International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *05 July 2023* ***Revised:*** *25 August 2023* ***Accepted:*** *15 September 2023*

Risk Prioritization using A FUZZY BASED Approach in Software Development Design Phase

Tabrez Khan[1]*, Mohd. Faisal
[1,2] Department of Computer Application, Integral University, Lucknow, India
[1]tabrez.khan.mail@gmail.com

Abstract The success of a software project's objective is directly proportional to the degree to which it satisfies all of the stakeholders' concerns regarding the project's requirements, including the budget, schedule, and overall performance. Risks can occur throughout the software development lifecycle (SDLC) phases and affect every phase. The design phase of the SDLC yields an overview of the software and can be defined as the software's blueprint. Different types of software have their own unique design phases and have different types of risks. With the high number of interacting components, complex systems have a greater propensity to be more volatile, which increases the risk. It is necessary to prioritize the risks in order of their severity levels. The issue at hand is the lack of effective methods to prioritize and mitigate the risk. Recent studies have suggested several methods for prioritizing risks, but it is clear that few of these have been implemented. These methods are overly complicated, time-consuming, prone to inconsistency, and challenging to put into practice. This paper proposes a novel Fuzzy-based approach to risk prioritization in the software design phase using MATLAB software. Fuzzy-based models have been shown to be more accurate than other techniques when using standard datasets to prioritize risks. Fuzzy-based methods that have been proposed take into account the characteristics of risks by modelling those characteristics as fuzzy variables.

Keywords Design phase, Fuzzy-Based, Inference System, MATLAB, Risk Prioritization, SDLC

I. INTRODUCTION

The software development team has a stake in the software's development, and each of them shares a common set of issues about the different risks. Prioritizing risks is the most important aspect of risk management throughout the design phase [1] [2] . After completing the process of effectively collecting risks, the next obstacle to overcome is prioritizing the risks in severity order. The objective of risk prioritization is, among other things, to investigate the most significant risks that can affect software development. It is necessary to prioritize the attributes that are related to the risks to account for the fact that there is a continual change in the risks that occur during the development of software. The software development team may resolve disagreements, prepare for staged delivery, and take appropriate trade-off decisions with the help of risk prioritization. They can choose to balance their approach by dropping or deferring low-priority risks in favor of later releasing and implementing higher-priority ones. After the risk has been successfully identified, the next step is prioritization, which involves ranking the severity of the risk according the expectations and demands of the software development team. The information has been used in a variety of different ways to rank the risks according to their relative significance. The implementation of high-priority risks comes first in the risk prioritization process, followed by low-priority risks [3]. It is tricky for a software team to prioritize risks, and it is even more difficult to achieve consensus among numerous stakeholders with varying expectations. The value of a risk to software is the first consideration when stakeholders prioritize its importance. When determining the order of risks, a software development team considers several different factors. The challenges come in determining what factors are used as a base and what the most suitable methods for risk prioritization are. Techniques for risk prioritizing, in general, examine the risk in light of both its costs and its potential benefits. There have been many different risk prioritization approaches proposed, but none have proven effective. Quantitative methods take a rational approach to collecting data and determining priority. Some approaches prioritize informal grouping and generalization. In this study, Fig. 1 shows the five different phases of the software risk prioritization fuzzy-based approach [4].

The following is a summary of this document: Section 2 presents the literature review and the key contributions of our research work. Section 3 focuses on the different risk prioritization techniques in the design phase. Section 4 discusses the background of Fuzzy Inference Systems (FIS). Section 5 explains risk prioritization using fuzzy inference systems, risk prioritization attributes, and a fuzzy-based risk prioritization algorithm. Section 6 presents the experimental analysis with a CMS software test case. Section 7 presents a comparison of risk prioritization techniques. The Conclusion and Future work are in Section 8.

II. LITERATURE REVIEW

It has been observed that risk prioritization is an extremely important aspect of business software development. However, there hasn't been much advancement, either conceptually or practically, in the procedures for risk prioritization. The fact that there are numerous risks associated with both large-scale and small-scale software is one issue [5].

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *05 July 2023* ***Revised:*** *25 August 2023* ***Accepted:*** *15 September 2023*

Therefore, it is necessary to determine or pick the risk that is mitigated in different software versions. In other words, there is a requirement to prioritize the risks, which means that the risks that have been gathered need to be arranged in some kind of order according to their level of importance. Risk prioritization decisions are important since diverse elements often conflict with each other. It is challenging to figure out which risks are most important. The aforementioned discussion makes it clear that the majority of the methods and strategies suggested in the recent risk prioritization methods are not widely used. Most of the methods are overly convoluted, tedious, incompatible, and challenging to put into practice. In this research, a novel method for early risk prioritization is provided to determine the best or most appropriate tools and techniques and show their shortcomings as well as their asset qualities for risk prioritizing methods in software development, based on the severity levels and various features of the risks. A comparative analysis of different software-related prioritizing methods and techniques with some commonly identified features is presented.

The key findings of the current investigation are summarized in Table 1, which may be found below. This table contains the titles of various research publications, together with the primary results of the authors and the limitations of the methodologies that they used.

Figure. 1 Proposed Methodology for Risk Prioritization

Table1 Summary of Major Findings and Contributing Factors from Previous Studies

Reference	Title	Factors	Finding	Limitations
[6]	Risk-based test case prioritization using a fuzzy expert system	System requirements, updating status, intricacy, safety	Enhancing risk-based test case prioritizing that uses a fuzzy expert system for risk prioritization	There have only four risk factors considered
[7]	A systematic literature review on requirement prioritization techniques and their empirical evaluation	Precision, scaling, effectiveness, enchantment, time taken, tolerance for faults, reliableness, and complexities	Empirical analysis of structured Requirements Prioritization (RP) techniques	There are fewer comparison to evaluate and contrast different RP approaches in real-world
[8]	Application of a fuzzy-logic based model for risk assessment in additive manufacturing R&D projects	Quality, schedule, and budget	Fuzzy inference is used in model based on fuzzy logic to determine the level of risk	The scope of model is restricted to the contextualization of the technological- temporal due to the changing of technology
[9]	Efficient fine Tuned Trapezoidal Fuzzy-Based Model for Failure Mode Effect Analysis risk Prioritization	Risk uncertainty, dependency, and budget	Failure mode effects analysis (FMEA) is a tool that has been used in the process of risk assessment to analyze failure modes	In the realm of limited software projects, the model exhibits impressive results
[10]	A risk evaluation method to prioritize failure modes based on failure data and a combination of	Severity, occurrence, and detection	The grey theory fuzzy set has been used for risk assessment	For analytical data risk assessment, the model needed more factors and a visual

	fuzzy sets theory and grey theory			interface
[11]	Comparison of Requirement Prioritization Techniques to Find Best Prioritization Technique	Ratio Scale	Analysis of several approaches for prioritization of requirements	Required additional factors to do the analysis of many approaches for risk prioritization
[12]	Development of Rule-Based Software Risk Assessment and Management Method with Fuzzy Inference System	Manpower, schedule, and budget	Risk analysis and management using fuzzy Inference system	There is a requirement for a survey on software risks and linguistic norms to obtain their input and evaluations
[13]	Cyber Security Risk Assessment using Multi Fuzzy Inference System	Risk probability vulnerabilities, impacts	Risk analysis performed using a fuzzy multi inference system	The proposed model is exclusively applied only to concerns regarding cyber threats
[14]	Assessment and Comparison of Fuzzy Based Test Suite Prioritization Methods for GUI Based Software	Coverage criteria based on events, interactions between events, and parameter value	Analysis and evaluation of fuzzy based test suite prioritization methods for GUI software	The proposal will only work for graphical user interface (GUI)
Proposed Method	Risk Prioritization using A Fuzzy Based Approach in Software Development Design Phase	Requirements Documents, Improper Architectural Design, ProgrammingLanguage, Physical Model Activity, Specifying Design Activity, Documenting Design Activity [15]	Risk Prioritization using Fuzzy Inference System (FIS)	More attributes and risk factors will be added to our methodology to improve risk prioritization, and other models will be compared.

Fuzzy inference system model are used in scientific and engineering areas due to their intuitive and heuristic nature and because they are more flexible than probability models. And its ability to handle ambiguity and linguistic variables is one reason why it has been put into practice. Fuzzy inference systems, which can be easily combined with other fuzzy systems [16] [17].

Key Contribution:-

This paper presents a risk prioritization methodology for software design with the following salient key contributions:

1. Several different types of risks and their risk factors can arise during the software design phase and have a significant impact on the software's schedule and budget. These risks are prioritized based on their severity level [18].
2. A fuzzy-based risk prioritization model has been proposed for the software design phase.
3. The fuzzy-based model has been used for numerous factors in risk prioritization.
4. The fuzzy-based model, which is often used to prioritize software risk, has been analyzed and compared with other existing risk prioritization models in this study.

III. DIFFERENT TECHNIQUES FOR PRIORITIZING RISKS

The number of risks in software is increasing due to the demands of different types of software. And there is a need for a method to prioritize these risks. Generally, all risks cannot be covered due to a lack of available time and resources. Researchers therefore strongly demand effective and trustworthy methods for prioritizing risks. Despite the fact that many different risk prioritization techniques have been proposed, there is no evidence to show that any of these techniques can help in prioritizing risks during the software design process. The software design process constantly changes during software development. So it is very important to prioritize the risk. In the following sections, we'll discuss some of the more well-known methods for risk prioritization[19].

A. Analytical Hierarchy Process

The Analytic Hierarchy Process (AHP) is a statistical method for overcoming decision-making issues when there are a large number of factors to take into consideration [20][21]. Risk prioritization is employed to prioritize the potential risk according to factors like probability and severity. In this method, the risks are compared to one another in pairs to determine at what point one risk becomes more important than the other. The procedure makes n comparisons based on n x (n-1)/2, where ‘n’ represents the total number of potential risks.

B. Hierarchy AHP

This approach prioritizes the risk similarly to AHP, with fewer comparisons. It establishes a relationship between the risks using the hierarchical structure, which reduces the amount of redundancy and the number of comparisons. AHP is less sensitive to judgment errors. It has the potential to reduce the number of unnecessary comparisons; moreover, there is a compromise in that it also increases the ability to identify conflicting conclusions [22].

C. Minimal Spanning Tree (MST)

The AHP method introduces redundancy, which indicates

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *05 July 2023* ***Revised:*** *25 August 2023* ***Accepted:*** *15 September 2023*

inconsistent judgments because it uses pair-wise comparisons. If decision-makers were totally consistent, this would not be the case. This redundancy is eliminated by using a minimally spanning tree [23]. Since all redundant elements have been eliminated, the decision maker in this approach constructs a minimal spanning tree in a directed graph, which requires just n-1 comparisons, requires the least employment, and is very fast.

D. Bubble Sort

Sorting risks by their priorities is one of the easiest and most fundamental strategies for sorting elements in risk prioritization. It is also one of the most straightforward methods. This method is similar to AHP in that it uses the same number of pair-wise comparisons and ranks risks on an ordinal scale based on their priority [24].

E. B-Tree Prioritize

This risk-prioritizing method incorporates risk dynamics. The risk continues to arise and never ends. The risk never stops and never goes away. It makes use of an algorithm based on a binary tree that provides a function for mapping risks in accordance with their respective priority values against a collection of all risks that have already been prioritized. It has the capability to run in real-time, which means that the prioritization process can begin even if not all of the risks have been resolved. Late-arriving risks might be proposed or eliminated later.

F. Priority Groups

The significance of one set of risks in software development may vary significantly from that of another. It reduces the required effort by grouping the risks according to approximate prioritization. The risk inside these sets is then rated using an appropriate method for ranking the risk once the sets have been ranked internally. Similar to the binary search tree, there is the same amount of comparison.

IV. FUZZY INFERENCE SYSTEMS (FIS)

The process of problem-solving might involve ambiguity, the granularity of knowledge, and fundamental forces; fuzzy logic can manage all of these aspects [25]. Its representation of a complicated system is purely empirical and relies more on knowledge and experience than on a technical understanding of the problem at hand [26]. A FIS is often known as a fuzzy inference system [27]. The ideas behind FIS are quite simple to understand. FIS employs an attempt at deriving answers from a knowledge base [28]. FIS has an input stage, a processing stage, and an output stage. The input stage is responsible for mapping the inputs, which may include subscription functions, a time limit for the appropriate collection of truth values, an execution time, and other such things. During the processing stage, the necessary rules are called, and results are generated for each rule individually. The outcomes of all the rules created are then combined. The final stage is called the output stage, and it is responsible for converting the combined result back into a particular set of outcome values. The following are the five steps of the inference system, which are shown in Fig. 2.

A. Fuzzifying the Inputs

Employing membership functions, fuzziness can be introduced into the inputs by identifying the level to which they belong in each of the relevant fuzzy sets. This procedure is known as "Fuzzing Inputs" [29]. It is possible to determine the extent to which each stated rule has complied with every aspect of the antecedents after fuzzifying the inputs.

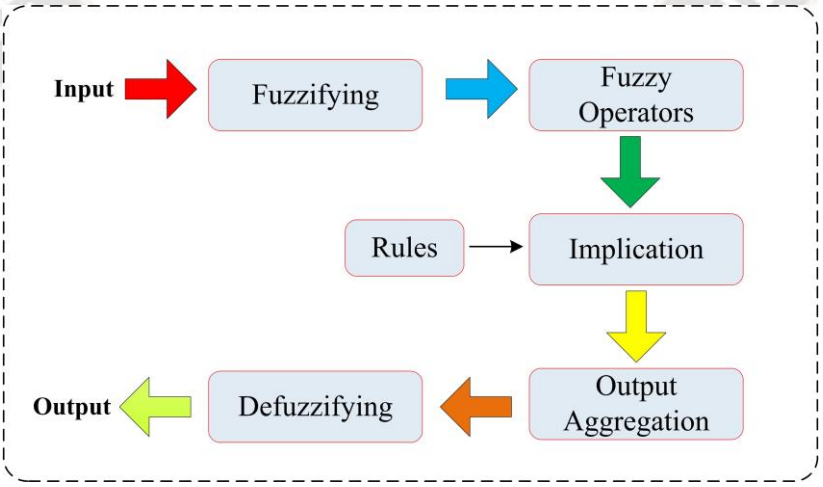

Figure 2 Fuzzy Inference System

B. Fuzzy Operators

The antecedents of a particular rule consist of more than one component, and then fuzzy operators have been used in order to generate a single value that accurately depicts the outcome of the component that constitutes the antecedent of that rule.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *05 July 2023* ***Revised:*** *25 August 2023* ***Accepted:*** *15 September 2023*

C. *Implication*

The output fuzzy set is subsequently modified by the implication function so that it corresponds to the antecedent's level.

D. *Output Aggregation*

Each rule in the Fuzzy Inference System (FIS) has been tested before a judgment can be made, and the findings from all of the rules must be pooled. A process aggregation combines multiple fuzzy sets, each representing the result of a set of rules, into a single fuzzy set.

E. *Defuzzifying*

A single value is generated from the aggregate fuzzy sets that are fed into the defuzzification procedure.

V. RISK PRIORITIZATION USING FUZZY INFERENCE SYSTEM (FIS)

Three linguistic variables are used as input in the proposed model for risk prioritization during the design process. The output stage is comprised of a single linguistic variable that is denoted with the abbreviation RP_l for priority rating. Through the application of the proper membership functions, the fuzzy set maps the input and output variables. For each linguistic phrase, the expert decides what form the membership function should take. The specification of FIS includes 27 rules. These rules are combined in order to produce the outcome. The priority rating RP_l is the result of the fuzzy inference system. When the RP_l value is higher, there is a greater potential for serious consequences. Within the initial phase of the design phase, estimates of high-priority risks are carried out.

A. *Attributes for Risks Prioritization*

When compared to the effects of the other factors on the software in which the risk is high, the fuzzy-based method of early risk prioritization has been offered as a solution [30]. The influence of an identified factor on risk prioritization is significant. Within the framework of the fuzzy-based approach, linguistic data are fed into the fuzzy-based system as its sources of information. Linguistic inputs make up the three factors that were chosen for prioritizing. There are three components that support the software. The risk can be prioritized in light of these considerations. After a careful review of many aspects based on their role, project support, and impact, several factors are defined following a critical review of the components that have been found. Table 2 displays the impact of various factors.

Table2 Prioritization risks factors

Attributes	Description	Priority Levels
Detectability	The levels where the risk is detectable	Low Level Medium Level High Level
Dependableness	Identified the levels to which different risks are dependent on one another	Low Level Medium Level High Level
Mitigation	Determine the risk mitigation level	Low Level Medium Level High Level

B. *Illustration of the Algorithm*

Step 1: Fuzzification of the input variables and FIS variable definition using the membership functioneditor. A well-defined boundary is considered a fuzzy set. A curve that is referred to as a membership function defines the mapping that takes place from every point in the input area to a membership value that falls between 0 and 1. Constructing, modifying, and observing FIS can all be accomplished with the help of the five graphical user interface (GUI) [14] tools that are included in the fuzzy logic toolbox, as follows:

i) Editor for the FIS
ii) Editor for the Membership Function
iii) Editor for Rules and a Viewer for Rules
iv) Surface View

Step 2: Prioritizing risks via rule sets in a Fuzzy Inference System.

The membership function editor is used to define the fuzzification and FIS variables that control the input variables. Input and output variables for the FIS are shown in Fig. 3.

FIS Detectability, Dependableness, and Mitigation of Input Variables are shown in Fig. 4, 5, and 6, and the variable RP_l from the FIS output is shown in Fig. 7.

Algorithm 1:

Algorithm: Fuzzy Based Risk Prioritization

```
Risk_prioritization ()
 {
   Input R={ r1,r2,r3…..rn}          // Risks in each module
   M= ={ M1,M2,M3…..Mn}              // Modules denoted by M
   m1= ={ r1,r2,r3…..rn}             // The module is associated with a set of risks
   m2= ={ r1,r2,r3…..rn}
   mn= ={ r1,r2,r3…..rn}
```

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 05 July 2023 **Revised:** 25 August 2023 **Accepted:** 15 September 2023*

```
    Fuzzy_risk_ prio()
    {
1   Fuzzification of the input variables and FIS
    variable definition using the membership
    function editor
2   Prioritizing risks via rule sets in a Fuzzy
    inference system.
3   Analyzing Rules
4   The aggregation of the rules' output.
5   Defuzzification of output
    }
}
```

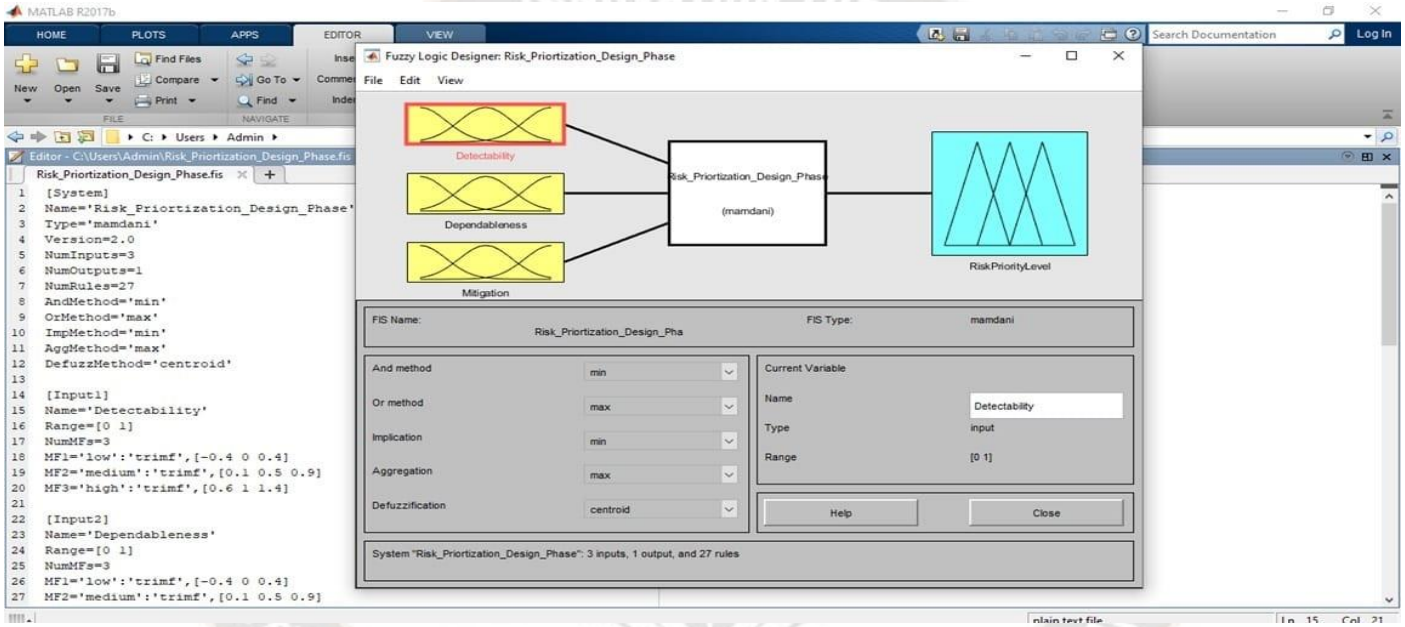

Figure 3 FIS Editor for Risk Prioritization

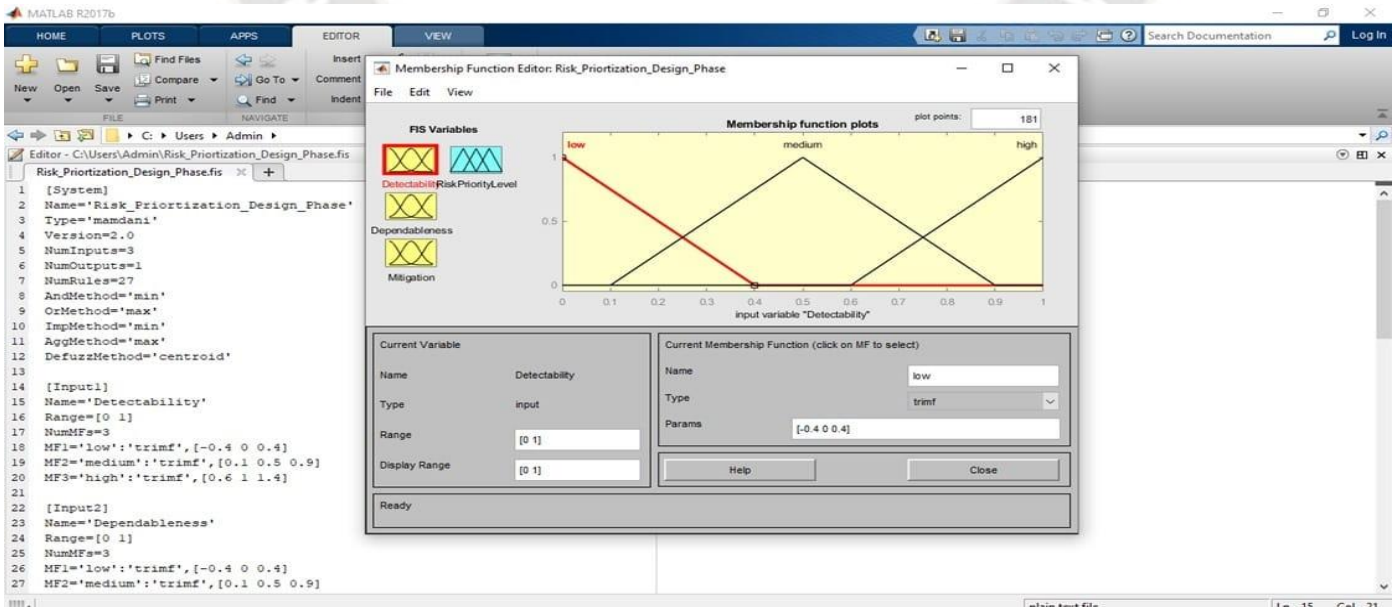

Figure 4 FIS Detectability of Input Variables

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *05 July 2023* ***Revised:*** *25 August 2023* ***Accepted:*** *15 September 2023*

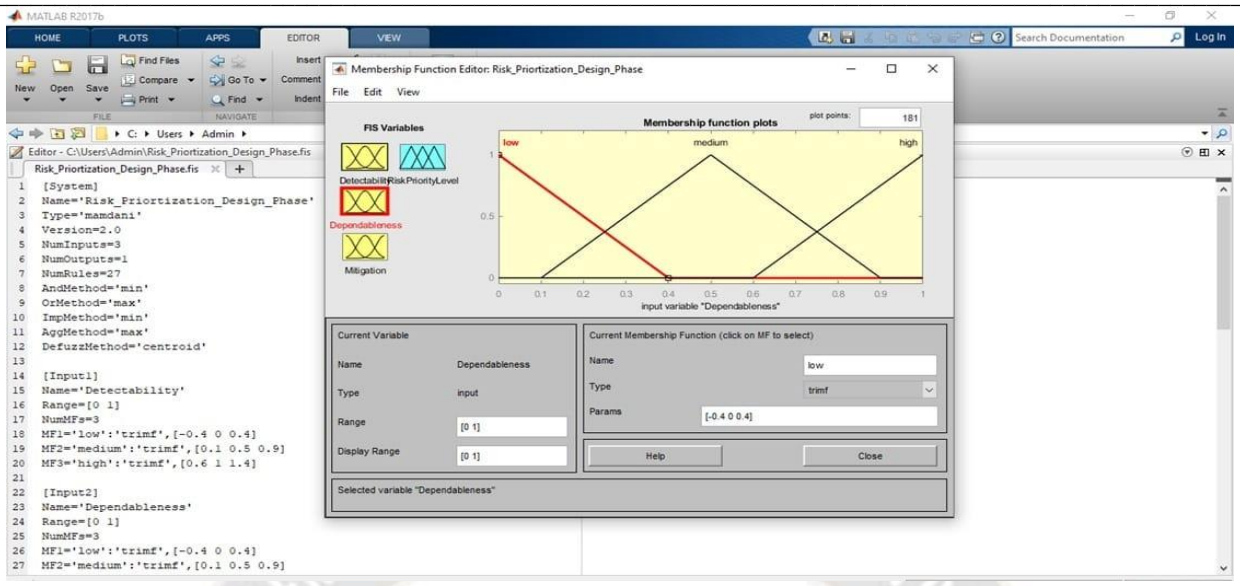

Figure 5 FIS Dependableness of Input Variables

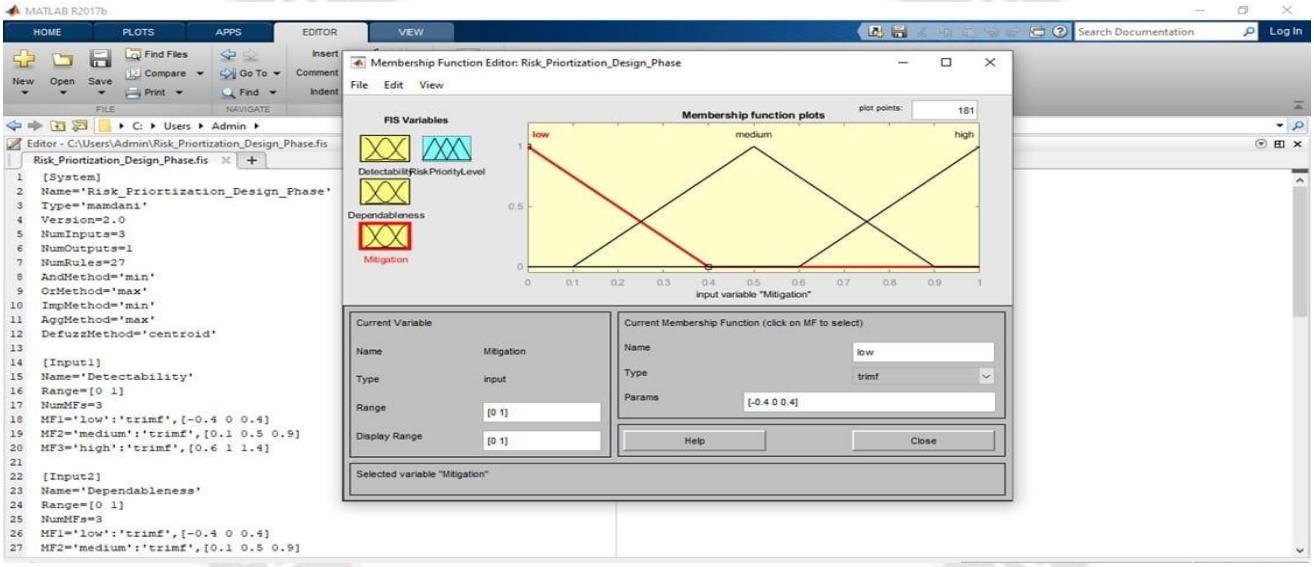

Figure 6 FIS Mitigation of Input Variables

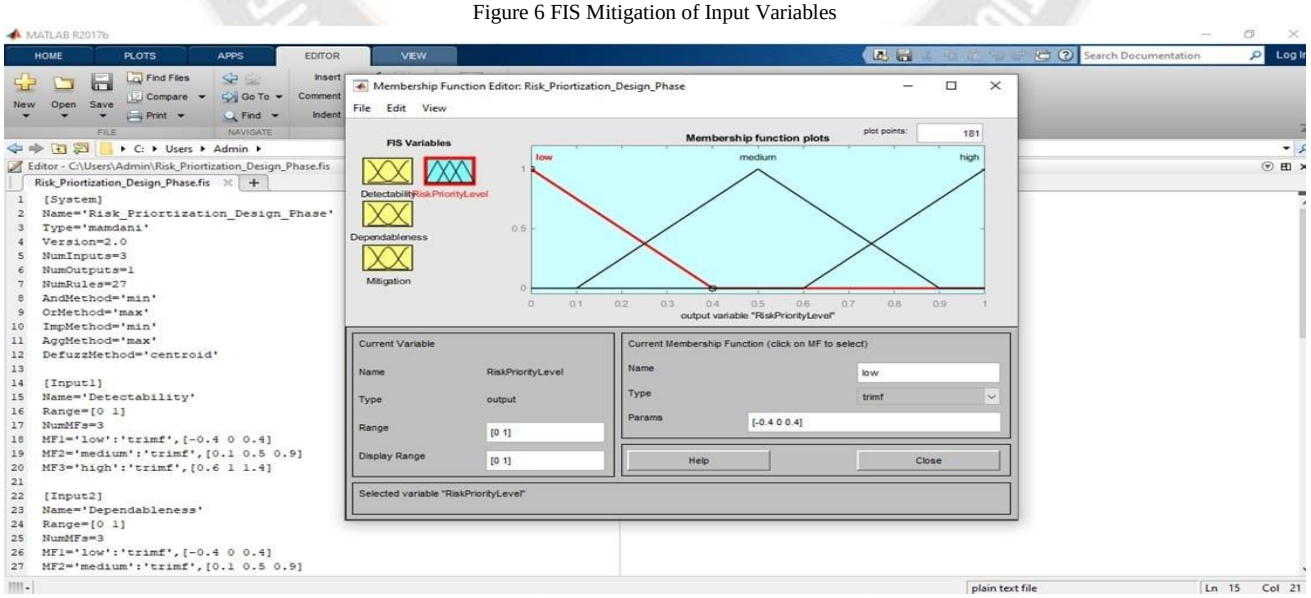

Figure 7 Variable RP_l from the FIS output

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *05 July 2023* ***Revised:*** *25 August 2023* ***Accepted:*** *15 September 2023*

Step 3: Analyzing rules the fuzzy deduction framework's rules for using Rule Editor for risk prioritization are shown in Table 3.

Table 3 Fuzzy Rules

Detectability	Dependableness	Mitigation	Risk Priority Level
Low	Low	Low	Low
Low	Low	Medium	Low
Low	Low	High	Low
Low	Medium	Low	Low
Low	Medium	Medium	Medium
Low	Medium	High	Medium
Low	High	Low	Medium
Low	High	Medium	Medium
Low	High	High	Medium
Medium	Low	Low	High
Medium	Low	Medium	Low
Medium	Low	High	Medium
Medium	Medium	Low	Medium
Medium	Medium	Medium	Medium
Medium	Medium	High	Medium
Medium	High	Low	Medium
Medium	High	Medium	Medium
Medium	High	High	High
High	Low	Low	Medium
High	Low	Medium	Medium
High	Low	High	Medium
High	Medium	Low	Medium
High	Medium	Medium	Medium
High	Medium	High	High
High	High	Low	Medium
High	High	Medium	High
High	High	High	High

Step 4: Risk-prioritizing fuzzy deductive editors' rules are shown in Fig. 8.

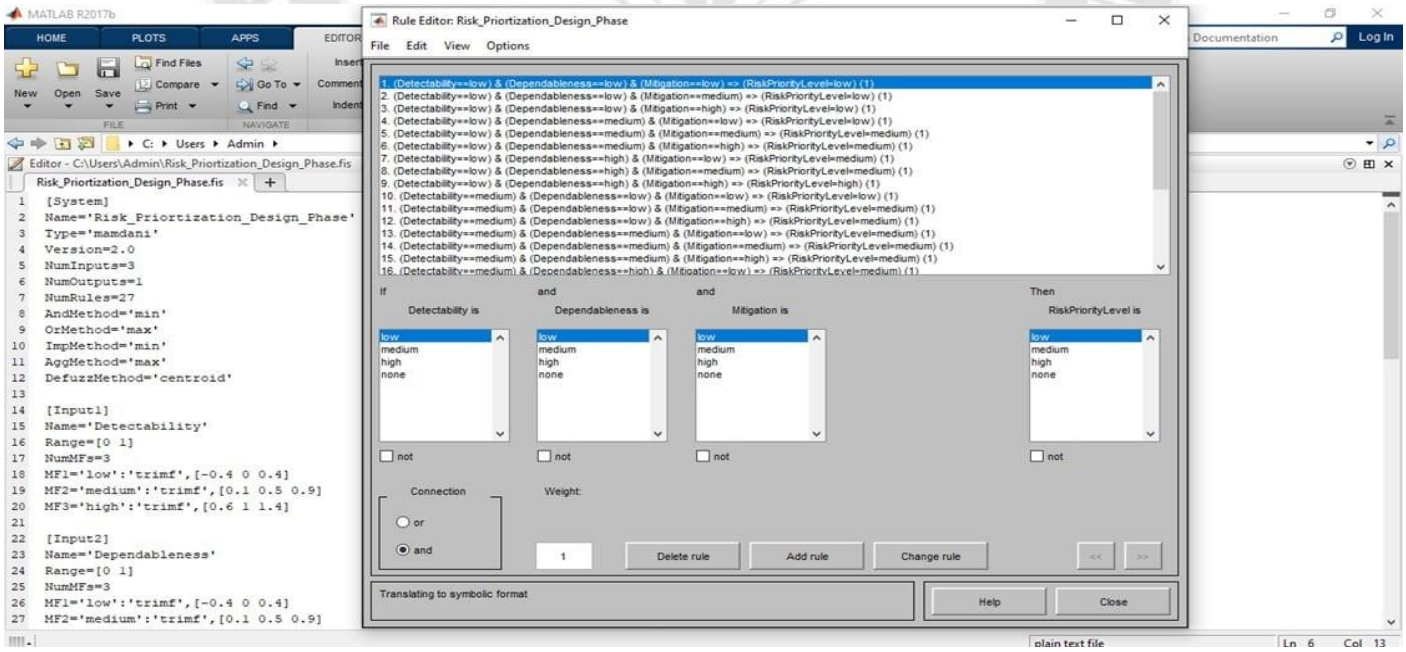

Fig. 8. Risk-prioritizing fuzzy deductive editors' rules

Step 5: Fig. 9 shows the analyzing rules of the aggregation output and the defuzzification of the resulting value.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *05 July 2023* ***Revised:*** *25 August 2023* ***Accepted:*** *15 September 2023*

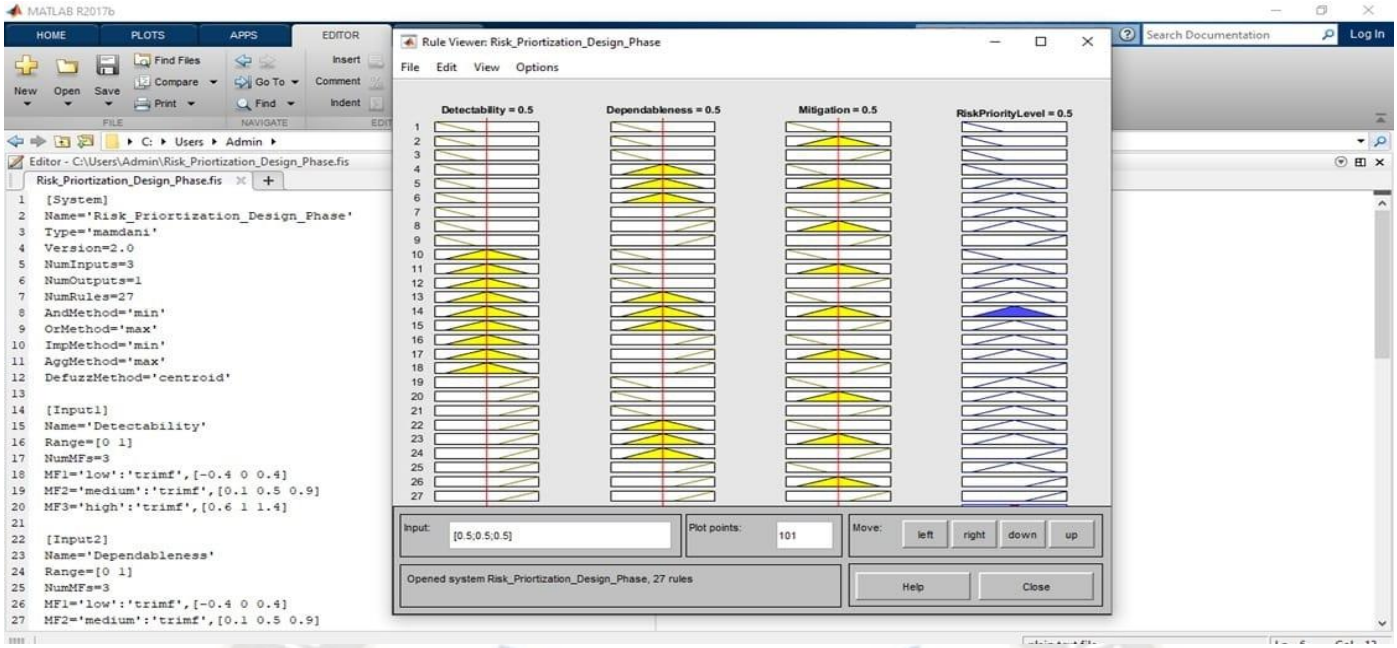

Figure 9 Aggregation of the Rule Output Based on the Rule's Evaluation

Step 6: Output defuzzification Fig. 10 shows the surface view of the risk prioritization level.

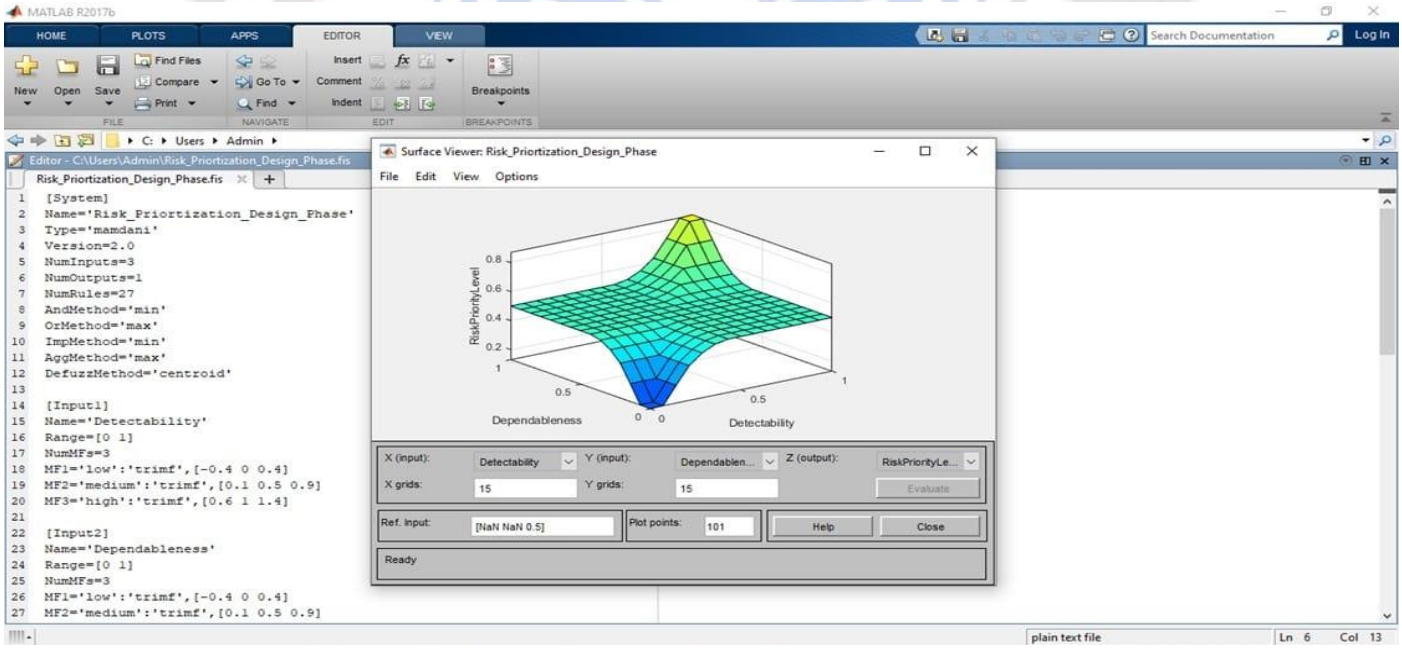

Figure 10 Surface view of risk prioritization Level

VI. EXPERIMENTAL ANALYSIS

A vital phase in the software development process is the prioritization of risk [31]. In this section, a risk prioritization model is developed with the use of a fuzzy logic-based algorithm in MATLAB with the assistance of the fuzzy logic toolbox. The idea of prioritization of these changing risks will undoubtedly assist the software development team in the providing of a more effective method for managing these risks. This is due to the fact that risk continues to change throughout the entirety of the SDLC [32].

The fuzzy-based approach to risk prioritizing has been proposed in earlier sections. The algorithm for the fuzzy risk prioritizing method is presented, and the outcomes are shown in Fig. 10. This subsection provides a case study of implementing the proposed fuzzy-based technique for risk prioritizing. The risk priority is determined using the suggested fuzzy-based inference system method shown in Figs. 3 to 10. To conduct an experimental analysis on a fuzzy-based strategy for risk prioritization, the CMS software has

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue:
Article Received: *05 July 2023* ***Revised:*** *25 August 2023* ***Accepted:*** *15 September 2023*

been chosen to serve as a case study. During the design phase, the risks associated with the CMS software were gathered and given priority using a fuzzy-based methodology. A few risks related to the CMS software are listed in Table 4. These risks are derived from the framework that has been proposed for the design phase. Now, the next obstacle to overcome is establishing a priority order for the CMS software's risks. At an early stage in the software design process, the accumulated risks are prioritized.

Table 4 CMS Risk at Design Phase[15]

Risk Factors	Risk_ID	Risk Description
Requirements Document	R_1	Requirement document is not clear for developers
	R_2	Ambiguity of requirements
	R_3	Lack of contribution of planning in the maintenance and changes of requirements
Improper Architectural Design	R_4	Improper Architectural design method choice
	R_5	Poor clustering of Architecture design affects the performance negatively
	R_6	Misalignment of components to decompose the software system into its major components
Programming Language	R_7	Improper choice of the PL
	R_8	Lack of knowledge of the PL and interface design
	R_9	Introduction of new technology about domain interface
Physical Model Activity	R_{10}	Large-sized components
	R_{11}	A lack of reusable components as expected
	R_{12}	Wrong methods prolong the duration of the decomposition into the main components of the system
Verifying Design Activity	R_{13}	Many feasible solutions are available and choosing the wrong solution.
	R_{14}	Lack of a reusable verifying design
	R_{15}	Delay in verifying design activity for important maintenance and decisions
Specifying Design Activity	R_{16}	A large amount of tramp data
	R_{17}	Omitting data processing functions
	R_{18}	Insufficient functions to ensure security, integrity, and availability of the database
Documenting Design Activity	R_{19}	Incomplete design document
	R_{20}	The ambiguity of the design document
	R_{21}	Changes in the design document

Table 5 Risk Prioritization Attributes, Risk Priority Rating and Risks Priority Level

Risk_id	Risk Prioritization Attributes	Values	Risk Priority Rating	Risk Priority Level
R_1	Detectability	5.9	6.2	Medium
	Dependableness	4.9		
	Mitigation	5.9		
R_2	Detectability	5.4	9	High
	Dependableness	8.6		
	Mitigation	8.8		
R_3	Detectability	8	7	High
	Dependableness	5		
	Mitigation	7		
R_4	Detectability	3	2	Low
	Dependableness	3.2		
	Mitigation	3		
R_5	Detectability	5	5.6	Medium
	Dependableness	7.5		
	Mitigation	6.7		
R_6	Detectability	8.5	9.2	High
	Dependableness	8		
	Mitigation	8.2		
R_7	Detectability	3.9	5.8	Medium
	Dependableness	3.6		
	Mitigation	4		
R_8	Detectability	2	3	Low
	Dependableness	5.6		
	Mitigation	3.5		
R_9	Detectability	2.5	6	Medium
	Dependableness	3.5		
	Mitigation	3.6		

R_{10}	Detectability	2.6	4	Medium
	Dependableness	2		
	Mitigation	7.9		
R_{11}	Detectability	4.6	6.7	Medium
	Dependableness	8.9		
	Mitigation	7.8		
R_{12}	Detectability	8.7	7.9	High
	Dependableness	4.7		
	Mitigation	8.1		
R_{13}	Detectability	2.3	3.3	Low
	Dependableness	1		
	Mitigation	5.6		
R_{14}	Detectability	3.5	3.3	Low
	Dependableness	1		
	Mitigation	5.9		
R_{15}	Detectability	6.7	4.4	Medium
	Dependableness	5.2		
	Mitigation	2		
R_{16}	Detectability	9	9	High
	Dependableness	8		
	Mitigation	7		
R_{17}	Detectability	6.1	6.1	Medium
	Dependableness	5.4		
	Mitigation	7.1		
R_{18}	Detectability	3	9	High
	Dependableness	9		
	Mitigation	9		
R_{19}	Detectability	4.5	6.9	Medium
	Dependableness	7.9		
	Mitigation	7.6		
R_{20}	Detectability	6	4	Medium
	Dependableness	5		
	Mitigation	2		
R_{21}	Detectability	7.7	8.9	High
	Dependableness	8.6		
	Mitigation	9.2		

Table 6 Risk Priority of CMS Software

Risk_id	Risk Priority Level
R2, R3, R6, R12, R16, R18, R21	High
R1, R5, R7, R9, R10, R11, R15, R17, R19, R20	Medium
R4, R8, R13, R14	Low

FIS has 27 rules defined, and the RP_I variable is used to defuzzify the output. The results of the analysis are displayed in Tables 4 and 5. Fuzzy logic systems have been used to derive extremely high RP_I values for risks R_2, R_3, R_6, R_{12}, R_{16}, R_{18}, and R_{21} as seen in Table 6. The initial step of deployment is mitigating these high-priority CMS risks. The risk assessment method uses fuzzy logic. When it comes to prioritizing risks, prioritization is an effective strategy, especially in large-scale software where the risk set might be quite extensive.

VII. COMPARISON OF RISK PRIORITIZATION TECHNIQUES

It can be seen from the findings that are presented in Table 2 that AHP [33] [34] is capable of providing the most dependable outcome out of the six approaches; the largest number of decisions and more time are required to complete them. The MST requires fewer decisions and less time. The output it produces is trustworthy, and it has the least amount of fault-tolerance. The process of prioritization can be completed with less work and in a shorter amount of time with this method; however, there is a greater possibility that valuable project resources and time will be misdirected as a result of the method's less trustworthy findings. The bubble sorting method is the simplest to implement, and it has the potential to produce results that are relatively reliable and have a relatively strong fault tolerance, but it also requires the most decisions. The middle ground is where you'll find hierarchical AHP and

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue:
Article Received: *05 July 2023* ***Revised:*** *25 August 2023* ***Accepted:*** *15 September 2023*

binary search trees. Compared to AHP and bubble sort, they yield less trustworthy outcomes, but they require fewer decisions and can be executed more quickly. It is clear that none of these six approaches to setting priorities are without flaws, but some are better than others. The concept behind a B-tree is that each of its internal nodes has the potential to hold a different number of child nodes. B-tree complexity is calculated as O (t*longt n), while AHP complexity is estimated as O (n2). It is seen from the comparison that the B-tree requires substantially fewer comparisons than the AHP does. When there are a high number of risks that need to be prioritized but only a limited number of comparisons are necessary, this is a crucial factor. Among these six prioritization techniques, it can be observed that none of them is flawless, especially for large-scale projects. Prioritizing the risk is the biggest issue in large-scale projects since the risk sets are so enormous. The reasoning above leads to the conclusion that, especially in large-scale projects, early risk prioritization requires an efficient method through which analysts may quickly prioritize the risk during the implementation phase. The suggested fuzzy-based method for risk prioritization uses fewer comparisons than existing methods, permits grouping of prioritized risks, and is trustworthy, simple to use, affordable, quick to complete, fault-tolerant, and suitable for large-scale software. Compared to other methodologies, the outcomes from a fuzzy-based approach are always apparent, and they help analysts prioritize requirements while making decisions. Table 7 shows the comparisons of risk prioritization techniques.

Table 7 Comparisons of Risk Prioritization

Technique's	Comparisons	Fuzziness	Complexity
Analytic Hierarchy Process	N*(n-1)/2	Not Supported	$O(n^2)$
Fuzzy AHP	<n*(n-1)/2	Supported	$O(n^2)$
Spanning Tree Matrix	n-1	Not Supported	O (log n)
Bubble Sort	n*(n-1)/2	Not Supported	O (n^2)
B-Tree	N*(n-1)/2	Not Supported	O (t* logt n)
Priority Group	n-1	Not Supported	O (t * long n)

VIII. CONCLUSION AND FUTURE WORK

It's quite challenging to predict and prioritize the risks associated with the software. Some risks that are not predicted and prioritized in the SDLC phases are exacerbated in the next phases of the SDLC. Risk prioritizing is a crucial step and a time-consuming process in the software development design process, and it improves the software development visibility. Improper or inadequate prioritization may lead to software schedule delays, budget overruns, poor quality of software, rework, and even software failure or abandonment. So it's important for the development team to prioritize risks accurately so they can mitigate and make accurate estimations of the risks. Therefore, the conclusion is that no single model or method is perfect for prioritizing the risk in the software development design phase because every model has different software design phases due to the differences in their requirements, features, and working procedures. This work attempts to prioritize the various risks inherent in the software design process by employing a fuzzy-based approach to prioritize the potential risks based on severity levels. With the help of the proposed fuzzy-based method, FIS is implemented in MATLAB for efficient risk prioritization for large-scale and small software projects according to the specifications of stakeholders. Different risk prioritization methods have been compared using different factors in all the stated methods. The results generated by fuzzy-based systems are accurate, reliable, and appropriate for application in a wide range of endeavors, and comparative analysis will be useful in assisting and providing basic ideas to the software development team in selecting the right risk-prioritizing techniques based on their needs in the software design phase. Table 6 shows that AHP is capable of providing the most reliable outcome out of the six distinct approaches evaluated according to numerous different factors.

The authors will extend this analysis by adding more prioritization methods in order to examine the efficacy of various approaches to risk prioritization and how a variety of features can increase the accuracy of risk prioritization. In the future, researchers may come up with a framework for predicting and prioritizing the most significant risks and their factors according to their severity level in the software design phase so that risk-free software can be developed.

Acknowledgement

This work is acknowledged under Integral University manuscript No. IU/R&D/2023-MCN0002117.

Reference:-

[1] T. Khan and M. Faisal, "The Essence of Risk Management in Software Development : A Comparative Study," 4th Springer Inetrnational Conf. Data, Eng. Appl., no. 1, pp. 1–12, 2022.

[2] B. H. Adbelrafe Elzamly, "Managing Software Project Risks Design Phase With Proposed Fuzzy Regression Analysis Techniques with Fuzzy Concepts.pdf," IRECOS, vol. Vol 8.N.11, 2013, [Online]. Available: https://www.researchgate.net/publication/260917920_Managing_Software_Project_Risks_Design_Phase_with_Proposed_Fuzzy_Regression_Analysis_Techniques_with_Fuzzy_Concepts

[3] S. N. Bhukya and S. Pabboju, "Software engineering: risk features in requirement engineering," Cluster Comput., vol. 22, pp. 14789–14801, 2019, doi: 10.1007/s10586-018-2417-3.

[4] S. A. Khan, W. Khan, and D. Pandey, "A Fuzzy Multi-Criteria Decision-Making Method for Managing Network Security Risk Perspective," no. October, pp. 115–140, 2020, doi: 10.4018/978-1-7998-2764-1.ch006.

[5] M. Pasha, G. Qaiser, and U. Pasha, "A critical analysis of software risk management techniques in large scale systems," IEEE Access, vol. 6, no. c, pp. 12412–12424, 2018, doi: 10.1109/ACCESS.2018.2805862.

[6] C. Hettiarachchi, H. Do, and B. Choi, "Risk-based test case prioritization using a fuzzy expert system," Inf. Softw. Technol., vol. 69, pp. 1–15, 2016, doi: 10.1016/j.infsof.2015.08.008.

[7] F. A. Bukhsh, Z. A. Bukhsh, and M. Daneva, "A systematic literature review on requirement prioritization techniques and their empirical evaluation," Comput. Stand. Interfaces, vol. 69, p. 103389, 2020, doi: 10.1016/j.csi.2019.103389.

[8] B. M. Moreno-Cabezali and J. M. Fernandez-Crehuet, "Application of a fuzzy-logic based model for risk assessment in additive manufacturing R&D projects," Comput. Ind. Eng., vol. 145, p. 106529, 2020, doi: 10.1016/j.cie.2020.106529.

[9] R. Subramanian, S. Taterh, D. Singh, and H. N. Lee, "Efficient Fine Tuned Trapezoidal Fuzzy-Based Model for Failure Mode Effect Analysis Risk Prioritization," IEEE Access, vol. 10, pp. 50037–50046, 2022, doi: 10.1109/ACCESS.2022.3172513.

[10] H. Wang, Y. M. Zhang, and Z. Yang, "A risk evaluation method to prioritize failure modes based on failure data and a combination of fuzzy sets theory and grey theory," Eng. Appl. Artif. Intell., vol. 82, no. 11, pp. 216–225, 2019, doi: 10.1016/j.engappai.2019.03.023.

[11] J. Ali Khan, I. Ur Rehman, Y. Hayat Khan, I. Javed Khan, and S. Rashid, "Comparison of Requirement Prioritization Techniques to Find Best Prioritization Technique," Int. J. Mod. Educ. Comput. Sci., vol. 7, no. 11, pp. 53–59, 2015, doi: 10.5815/ijmecs.2015.11.06.

[12] M. Batar, K. U. Birant, and A. H. Işik, "Development of Rule-Based Software Risk Assessment and Management Method with Fuzzy Inference System," Sci. Program., vol. 2021, 2021, doi: 10.1155/2021/5532197.

[13] H. Sallam, "Cyber Security Risk Assessment Using Multi Fuzzy Inference System," Int. J. Eng. Innov. Technol., vol. 4, no. 8, pp. 13–19, 2015, [Online]. Available: https://pdfs.semanticscholar.org/95cc/3661fd194263e27f0055f9cbcb0375f19192.pdf

[14] N. Chaudhary and O. P. Sangwan, "Assessment and Comparison of Fuzzy Based Test Suite Prioritization Method for GUI Based Software," Int. J. Adv. Comput. Sci. Appl., vol. 7, no. 1, pp. 222–225, 2016, doi: 10.14569/ijacsa.2016.070131.

[15] T. Khan and M. Faisal, "An efficient Bayesian network model (BNM) for software risk prediction in design phase development," Int. J. Inf. Technol., pp. 1–14, Apr. 2023, doi: 10.1007/S41870-023-01244-4/METRICS.

[16] T. Karimi, M. R. Fathi, and Y. Yahyazade, "Developing a Risk Management Model for Banking Software Development Projects Based on Fuzzy Inference System," J. Optim. Ind. Eng., vol. 13, no. 2, pp. 267–278, 2020, doi: 10.22094/JOIE.2020.1883892.1700.

[17] S. R. Chaudhari and M. E. Patil, "Comparative Analysis of Fuzzy Inference Systems for Air Conditioner," Int. J. Adv. Comput. Res., vol. 4, no. 4, pp. 922–927, 2014.

[18] S. Anas, S. Prakas, J. Yadav, and M. Waris, "Estimation of Software Security Risks through CVSS : A Design Phase Perspective Estimation of Software Security Risks through CVSS : A Design Phase Perspective," vol. 12, no. 4, pp. 894–901, 2021.

[19] A. Hudaib, R. Masadeh, M. H. Qasem, and A. Alzaqebah, "Requirements Prioritization Techniques Comparison," Mod. Appl. Sci., vol. 12, no. 2, p. 62, 2018, doi: 10.5539/mas.v12n2p62.

[20] A. A. Khan, M. Shameem, R. R. Kumar, S. Hussain, and X. Yan, "Fuzzy AHP based prioritization and taxonomy of software process improvement success factors in global software development," Appl. Soft Comput. J., vol. 83, p. 105648, 2019, doi: 10.1016/j.asoc.2019.105648.

[21] M. Shameem, A. A. Khan, M. Gulzarul Hasan, and M. A. Akbar, "Analytic hierarchy process based prioritisation and taxonomy of success factors for scaling agile methods in global software development," IET Softw., vol. 14, no. 4, pp. 389–401, 2020, doi: 10.1049/iet-sen.2019.0196.

[22] P. Chandani and C. Gupta, "Requirement Risk Prioritization Using Analytic Hierarchy Process: A Gateway to Identify Risky Requirements," 2018 11th Int. Conf. Contemp. Comput. IC3 2018, pp. 1–6, 2018, doi: 10.1109/IC3.2018.8530569.

[23] T. Hasuike and H. Katagiri, "Interactive decision making for uncertain minimum spanning tree problems with total importance based on a risk-management approach," Appl. Math. Model., vol. 37, no. 6, pp. 4548–4560, Mar. 2013, doi: 10.1016/J.APM.2012.09.051.

[24] I. Olaronke, I. Rhoda, and G. Ishaya, "An Appraisal of Software Requirement Prioritization Techniques," Asian J. Res. Comput. Sci., vol. 1, no. 1, pp. 1–16, 2018, doi: 10.9734/ajrcos/2018/v1i124717.

[25] D. Kalibatien and J. Miliauskait, "A Hybrid Systematic Review Approach on Complexity Issues in Data-Driven Fuzzy Inference Systems Development," Inform., vol. 32, no. 1, pp. 85–118, 2021, doi: 10.15388/21-INFOR444.

[26] N. Kaur, "A Fuzzy Logic Approach To Measure the Precise Testability Index of Software," International Journal of Engineering Science, 2011. https://www.researchgate.net/publication/50392266_A_FUZZY_LOGIC_APPROACH_TO_MEASURE_THE_PRECISE_TESTABILITY_INDEX_OF_SOFTWARE (accessed Jun. 22, 2023).

[27] P. Asadi, J. Rezaeian Zeidi, T. Mojibi, A. Yazdani-Chamzini, and J. Tamošaitienė, "Project risk evaluation by using a new fuzzy model based on Elena guideline," J. Civ. Eng. Manag., vol. 24, no. 4, pp. 284–300, 2018, doi:

10.3846/jcem.2018.3070.

[28] T. L. Jee, K. M. Tay, and C. P. Lim, "A New Two-Stage Fuzzy Inference System-Based Approach to Prioritize Failures in Failure Mode and Effect Analysis," IEEE Trans. Reliab., vol. 64, no. 3, pp. 869–877, 2015, doi: 10.1109/TR.2015.2420300.

[29] D. S. S. Rani and Others, "Neuro-Fuzzy based Software Risk Estimation Tool," Glob. J. Comput. Sci. Technol., vol. 13, no. 6, 2013.

[30] H. S. A. K. S. N. J. B. T. Gajanand Sharma Ashutosh Kumar, "Development of Fuzzy based intelligent System for Assessment of Risk Estimation in Software project for Hospitals Network," Eur. J. Mol. & Clin. Med., vol. 7, no. 4, pp. 1433–1442, 2020, [Online]. Available: https://ejmcm.com/article_1841.html

[31] M. M. De Carvalho and R. Rabechini Junior, "Impact of risk management on project performance: The importance of soft skills," Int. J. Prod. Res., vol. 53, no. 2, pp. 321–340, 2015, doi: 10.1080/00207543.2014.919423.

[32] A. Mathematics, "http://www.acadpubl.eu/hub/," vol. 119, no. 15, pp. 3005–3017, 2018.

[33] W. Min, Z. Jun, and Z. Wei, "The application of fuzzy comprehensive evaluation method in the software project risk assessment," ACM Int. Conf. Proceeding Ser., no. 388, pp. 76–79, 2017, doi: 10.1145/3034950.3035008.

[34] Ruby and Balkishan, "Role of Fuzzy Logic in Requirement Prioritization," Int. J. Innov. Res. Sci. Eng. Technol., vol. 4, no. 6, pp. 4290–4297, 2015, doi: 10.15680/IJIRSET.2015.0406099.