systems

MDPI

Article

Fuzzy Logic Model for Informed Decision-Making in Risk Assessment During Software Design

Gbenga David Aregbesola [1], Ikram Asghar [1,*], Saeed Akbar [2] and Rahmat Ullah [3]

[1] School of Computing, Engineering and Digital Technologies, Teesside University, Middlesbrough TS1 3JN, UK; q2505632@tees.ac.uk or gbaregbesola@gmail.com
[2] Department of Physical & Numerical Sciences, Qurtuba University of Science & Information Technology, Peshawar 25150, Pakistan; saedakbr@qurtuba.edu.pk
[3] School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, UK; rahmat.ullah@essex.ac.uk
* Correspondence: i.asghar@tees.ac.uk

Abstract

Software development projects are highly susceptible to risks during the design phase, which plays a crucial role in shaping the architecture, functionality, and quality of the final product. Decisions made during the design stage significantly affect the outcomes of the subsequent phases, including coding, testing, deployment, and maintenance. However, the complexities and uncertainties inherent in the design phase are often inadequately addressed by traditional risk management tools as they rely on deterministic models that oversimplify interdependent risks. This research introduces a fuzzy logic-based risk assessment model tailored specifically for the design phase of software development projects. The proposed fuzzy model, unlike the existing state-of-the-art models, regards the iterative nature of the design phase, the interaction between diverse stakeholders, and the potential inconsistencies that may arise between the initial and final version of the software design. More specifically, it develops a customized fuzzy model that incorporates design-specific risk factors such as evolving architectural requirements, technical feasibility concerns, and stakeholder misalignment. Finally, it integrates expert-driven rule definitions to enhance model accuracy and real-world applicability, ensuring that risk assessments reflect actual challenges faced by software design teams. Simulations conducted across diverse real-world scenarios demonstrate the model's robustness in predicting risk levels and supporting mitigation strategies. The simulation results confirm that the proposed fuzzy logic model outperforms conventional approaches by offering greater flexibility and adaptability in managing design-phase risks, assisting project managers in prioritizing mitigation efforts more effectively to improve project outcomes.

Keywords: decision-making; fuzzy model; risk assessment; design phase; software development project

Academic Editor: Polinpapilinho Katina

Received: 23 June 2025
Revised: 19 August 2025
Accepted: 15 September 2025
Published: 19 September 2025

Citation: Aregbesola, G.D.; Asghar, I.; Akbar, S.; Ullah, R. Fuzzy Logic Model for Informed Decision-Making in Risk Assessment During Software Design. *Systems* **2025**, *13*, 825. https://doi.org/10.3390/systems13090825

1. Introduction

In the dynamic and rapidly evolving field of computer science, software development projects are the driving force behind the creation of innovative applications and systems that promote technological progress [1]. Among the various stages of software development, the design phase is of particular significance as it establishes the blueprint upon which all the subsequent stages, such as coding, testing, deployment, and maintenance, are built. The foundation for the entire software project is laid during the design phase.

During design, the activities include defining the software architecture, creating comprehensive design documents, and detailing the overall structure of the system [2]. Decisions made during the design phase have far-reaching implications on the subsequent phases of software development, influencing everything from the efficiency of the coding process to the ease of testing, stability during deployment, and the sustainability of the maintenance efforts. A well-executed design phase can streamline the subsequent stages, leading to smoother project execution and higher-quality outcomes. In contrast, any shortcomings or oversights during this phase can introduce significant risks, with potentially amplified consequences as the project progresses [3].

The risks in the design phase may lead to the failure of the entire project if not handled properly. These risks come from various sources, such as ambiguities in user requirements, complexities in system integration, misalignment between stakeholder expectations, cognitive biases, and potential technological constraints [4–6]. The nature of these risks is often complex as they involve both technical and non-technical elements, such as communication gaps within the team, evolving project requirements, and the challenges of innovative design. Unidentified or poorly managed risks can lead to cost overruns as unforeseen issues may require additional resources to resolve. Project timelines can be extended, causing delays that affect the software development process as well as the broader strategic goals of the organization [7]. In the worst-case scenario, these risks can culminate in project failure, resulting in wasted effort, financial losses, and potential damage to the organization's reputation [8]. In this context, there is a pressing need to develop robust and comprehensive strategies incorporating technical and human factors that can accurately identify, assess, and mitigate the unique risks of the design phase.

Effective risk management during the software design phase requires flexibility to adapt to the evolving nature of software projects. It involves a proactive approach to early identification of potential risks, assessing their impact and likelihood, and implementing strategies to mitigate or avoid them. It is an iterative process, with regular reviews and updates to ensure that new risks are identified and addressed as the project evolves [9]. Using effective and flexible tools for software design risk management, project teams can enhance the likelihood of successful project outcomes, delivering software that meets its intended goals within the specified time and budget constraints.

The design phase is inherently characterized by high levels of imprecision and variability: project teams are often involved in translating vague and evolving user requirements into detailed design specifications. Specifically, the translation process is challenging as the requirements are often incomplete, ambiguous, or subject to change [10]. Moreover, design decisions made during the design phase have long-lasting implications, setting the direction for the subsequent development efforts and potentially locking in certain risks that can escalate in the later stages of the software project [11].

Traditional risk management techniques, generally designed to address more tangible and quantifiable risks encountered in later phases such as coding, testing, and deployment, struggle to cope with the abstract and dynamic nature of risks in the design phase [12]. The traditional approaches may not be very effective in the design phase for the following two reasons: (1) they often rely on historical data, probabilistic models, and structured processes, and (2) many of the risks are qualitative and stem from human factors, creativity, and innovation [5]. There is a pressing need for a more nuanced and specialized approach to risk assessment that can accurately capture and address the specific types of risks that arise during the design phase. Moreover, the approach must have the flexibility to handle the fluidity and subjectivity of design-related risks yet be robust enough to provide actionable insights that can guide decision-making and risk mitigation strategies.

Unlike the existing studies, this paper offers an effective risk management tool based on fuzzy logic for the software design phase. The proposed method offers the following benefits compared to existing state-of-the-art approaches: firstly, it captures the iterative nature of the design process; secondly, it considers the interactions between diverse stakeholders; thirdly, it accounts for the potential for misalignment between the initial vision and the final product during the design phase; and, finally, it deals with the inherent uncertainty and variability of software design. In short, this research aims to contribute to the field of software project management by providing a more comprehensive and effective risk management framework for the design phase, ultimately leading to better project outcomes and reducing the likelihood of costly failures or rework in the later stages of development.

1.1. Aims and Objectives

This work aims to design a comprehensive fuzzy model framework to identify and assess risk factors during the design phase of software development projects. The framework addresses the inherent uncertainty and imprecision associated with risk factors during this phase. Using fuzzy logic [13], the model can handle the subjective and often qualitative nature of risks, offering a robust tool that can support project managers in making informed decisions during the design process. Consequently, it provides a more nuanced and flexible approach to risk management. The objectives of this research are as follows:

1. To investigate the previous research on risk factors related to the design phase of software development projects: Firstly, we conduct an extensive literature review to explore the existing research on risk factors, analyzing academic papers, industry reports, and case studies to understand the current state of knowledge. Secondly, we identify gaps in the existing risk management tools and highlight the specific challenges associated with managing risks during the design phase.
2. To identify and categorize specific risk factors pertinent to the design phase of software development projects: Building on the insights gained from the literature review, the second objective is to identify and categorize the specific risk factors that are most relevant to the design phase of software development projects. This process involves systematically listing potential risks, such as ambiguity in requirements, design complexity, stakeholder misalignment, and technological constraints. The idea is to categorize each identified risk based on its nature (e.g., technical, organizational, or process-related), potential impact, and likelihood.
3. To develop a fuzzy logic-based model using MATLAB and Simulink: The development process involves creating the fuzzy inference system, defining membership functions, and simulating various risk scenarios to evaluate the model's performance.
4. To validate the proposed fuzzy model through empirical data: We validate the accuracy and effectiveness of our proposed model using empirical data from past software development projects to ensure its reliability in assessing risk factors and predicting potential issues.
5. To provide practical guidelines for implementing the fuzzy model in real-world software development projects: The final objective is to translate the theoretical and technical aspects of the fuzzy model into practical guidelines to integrate the fuzzy model into existing risk management processes, offer best practices for its use, and suggest strategies for interpreting and acting on the model's outputs. We provide generic guidelines so that it can be easily adopted by any software development team. The aim is to ensure that the fuzzy model is not just a theoretical tool but a practical solution that can be readily implemented in real-world projects, ultimately helping to improve risk management during the design phase and contributing to the success of software development initiatives.

1.2. Study Scope

This work focuses exclusively on risk factors associated with the design phase of software development projects and provides a detailed analysis of how risks during this critical phase impact a project's overall success. The design phase proves pivotal as it lays the foundation for all the subsequent stages, including coding, testing, deployment, and maintenance. However, the study does not cover any risks that emerge after the design phase. This focused approach allows a deep analysis of the unique challenges and uncertainties inherent in the design phase, such as the accuracy of requirement gathering, the adequacy of design specifications, and the potential misalignment between stakeholders' expectations and the final design blueprint. In short, such risks often prove to be complex and may cause cascading effects if not handled properly, making them strong candidates for isolated and careful study in the context of software design.

2. Literature Review

Risk assessment in software development has been extensively studied, with researchers focusing on identifying, evaluating, and mitigating various risks that can affect project success. Among the different phases of software development, the design phase plays a foundational role in shaping the overall system architecture, component interactions, and functional requirements. Given its impact on the subsequent phases, any risks emerging at this stage can have significant downstream consequences [14]. However, traditional risk assessment methods, which largely rely on deterministic models, often fail to capture the inherent uncertainties and dynamic interdependencies of the design phase [15]. These methods typically involve defining risks, estimating their likelihood and impact, and formulating mitigation strategies. While useful for structured and well-defined environments, such approaches oversimplify the nature of risks in software design, where uncertainties arise from evolving requirements, technical complexities, and resource constraints [16].

A major limitation of deterministic risk assessment methods is their inability to model the interdependencies and cascading effects of design risks. For instance, a change in system requirements may not only introduce new design challenges but also impact project timelines, increase costs, and require reallocation of resources—factors that traditional models struggle to account for [17]. Additionally, these methods often assume that risk factors can be quantified precisely, overlooking subjective and qualitative aspects such as stakeholder communication barriers and team experience levels. As a result, there has been growing interest in alternative approaches, particularly those based on fuzzy logic, which offers a more flexible and adaptive framework for risk assessment in software projects [18].

Fuzzy logic, introduced by Zadeh in 1965, provides a mathematical framework for reasoning under uncertainty, making it particularly useful in environments where risk factors are imprecise or evolving [19]. Unlike binary logic, which classifies risks as either present or absent, fuzzy logic quantifies the degree of risk, enabling more nuanced assessments. This flexibility is especially beneficial for modeling uncertainties in the design phase, where risk factors often overlap and interact in complex ways [20]. Researchers have explored the application of fuzzy logic in various aspects of risk management, demonstrating its ability to model non-linear relationships between risks, assess vague or ambiguous data, and integrate expert knowledge into decision-making [17].

The importance of early risk identification in software development has been emphasized in several studies. For example, Khan et al. [21] stated that detecting risks at the design phase is crucial for preventing costly failures later in the project lifecycle. The authors identified key risk factors, such as requirement ambiguities, stakeholder misalignment, and communication barriers, which, if unaddressed, can lead to project delays and bud-

get overruns [9]. These findings align with other studies that advocate for proactive risk management, arguing that early intervention significantly improves project outcomes [22].

Building on this foundation, Ibraigheeth et al. [17] demonstrated how fuzzy logic models can enhance risk assessment by taking into account uncertainty and subjectivity in risk evaluations. Their study introduced a fuzzy inference system to quantify risks that traditional methods struggle to capture, such as evolving project requirements and technical uncertainties. Unlike conventional deterministic models, their approach allowed for a more granular assessment, where risks were categorized into varying levels of severity rather than a simple binary classification. This methodology proved particularly effective for assessing risks during the design phase, where uncertainty is high and rigid classifications are insufficient for accurate decision-making.

Expanding on this work, Suresh et al. [20] developed an integrated fuzzy modeling framework for risk assessment, incorporating multiple risk factors into a single cohesive model, providing project managers with a more dynamic and realistic view of potential challenges. The proposed model could predict how certain risks might evolve over time and how they might interact with other project variables, enabling more effective mitigation strategies. This research was instrumental in demonstrating the practical applicability of fuzzy logic in real-world software development projects, particularly for teams dealing with complex and evolving risks [20].

Similarly, researchers have explored the use of fuzzy logic for optimizing project scheduling and resource allocation, further highlighting its versatility in risk management. Yi et al. [23] and Vasylkiv et al. [18] applied fuzzy models to improve decision-making in project planning, demonstrating how these techniques can dynamically adjust to changing project conditions. The authors showed that integrating fuzzy-based scheduling methods helps to mitigate risks related to resource bottlenecks and timeline disruptions, which are common in the design phase. Similarly, Akbar et al. [24] designed a multi-objective fuzzy decision-making framework that can enhance task prioritization in software projects, ensuring that critical design activities receive appropriate attention and resources. A similar study [25] emphasized the need for sophisticated risk assessment techniques in software design, arguing that the conventional methods are insufficient for handling the complexity of modern software projects. The proposed work reinforces the idea that fuzzy logic, with its ability to model uncertainty and interdependencies, provides a more comprehensive and realistic risk assessment framework than traditional deterministic approaches. This aligns with broader findings in the literature, highlighting the efficacy of fuzzy models against traditional risk assessment methods in environments characterized by high uncertainty, evolving requirements, and subjective decision-making factors [26].

In summary, the existing literature underscores the advantages of fuzzy logic in risk assessment during the software design phase. Unlike traditional deterministic models, which oversimplify risk factors and their relationships, fuzzy logic enables a more nuanced, adaptable, and dynamic assessment framework. By incorporating expert knowledge, probabilistic reasoning, and non-linear dependencies, fuzzy models provide a more accurate and practical tool for project managers, ultimately enhancing risk mitigation strategies and improving project outcomes. As software projects grow in complexity, the integration of fuzzy logic-based risk assessment is expected to play an increasingly critical role in software engineering methodologies [17,20].

Distinguishing the Current Research from the Existing Literature

A significant portion of the existing literature either focuses on general project risk management or ignores the intricacies of the risks related to the design phase of software projects. This research fills the gap by developing a fuzzy logic-based risk assessment model

specifically designed for the software design phase, considering the unique uncertainties and interdependencies at this stage.

Unlike existing investigations that use fuzzy logic across software projects, this research

- Focuses exclusively on software design, capturing the iterative nature of the design phase. Furthermore, it recognizes the fact that the risks during the design stage differ significantly from those in subsequent phases, such as implementation and testing.
- Creates a tailored fuzzy logic model that integrates key design-related risk elements, including changing architectural needs, challenges in technical feasibility, and stakeholder misalignment.
- Optimizes model accuracy and applicability by integrating expert-driven rule definitions. This ensures that risk assessments reflect actual challenges faced by software design teams.
- Provides practical guidelines for applying fuzzy risk assessments in real-world software projects.

By addressing the unique aspects of risk assessment, this research contributes both theoretically and practically to the field of software risk management, offering a more precise, adaptive, and actionable approach to risk assessment during the design phase.

3. Methodology

3.1. Approach

The approach for this project involves developing a comprehensive fuzzy model framework, as shown in Figure 1, tailored to assess and manage risks specific to the design phase of software development projects. This model was implemented using Simulink in MATLAB R2025a (22.0.2943329), focusing on integrating empirical data to validate its effectiveness. The methodology is structured to ensure a thorough and systematic process, combining quantitative analysis with practical implementation guidelines.

This research employs a mixed-methods approach, combining a quantitative analysis of secondary survey data with the development and simulation of a fuzzy logic-based risk assessment model. The use of secondary data from [27] allows for the identification of relevant risk factors. At the same time, the fuzzy model provides a dynamic tool for assessing and managing these risks in the design phase.

3.2. Research Methodology

This study employs a structured methodological approach to ensure a comprehensive and rigorous analysis. The research process begins with a literature review, which establishes the theoretical foundation for the study. By synthesizing the existing research on risk factors in the design phase of software projects, the literature review informs the selection of relevant variables and guides the overall research framework.

Following this, a quantitative analysis is conducted to empirically assess the identified risk factors. Using a dataset obtained from Kaggle (Khan, 2023), statistical techniques are applied to quantify and evaluate the impact of these risks within software project design. This step ensures that the research is grounded in empirical evidence, strengthening the validity of the findings.

Building upon insights from the literature review and quantitative analysis, the study proceeds with a design and creation approach. This involves developing a fuzzy inference system using MATLAB and Simulink to assess and model risks based on the identified factors. By leveraging fuzzy logic principles, the system enhances risk evaluation, providing a more nuanced and adaptable framework for managing uncertainties in software project design.

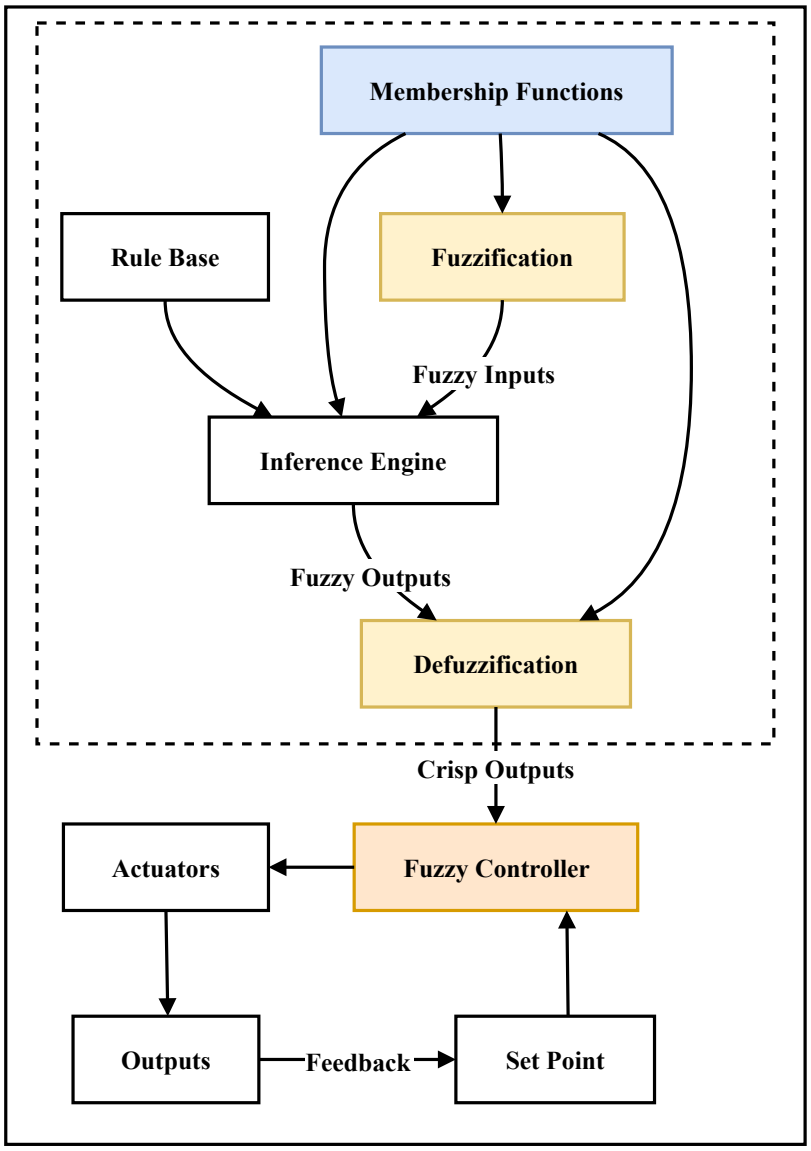

Figure 1. Architecture of the proposed fuzzy logic framework for risk assessment in software development, illustrating key components such as fuzzification, inference engine, membership functions, defuzzification. The dashed box highlights the fuzzy controller, which integrates fuzzification, inference, and defuzzification processes to generate crisp outputs for decision-making.

3.3. Data Collection and Preprocessing

The dataset used in this study is sourced from Kaggle and consists of survey responses capturing various risk factors, their perceived severity, and their likelihood of occurrence. Before analysis, the data undergoes a series of preprocessing steps to ensure its quality and consistency. Initially, missing and incomplete entries are removed to maintain dataset reliability. The data is then normalized to ensure comparability across different risk factors, followed by an outlier detection process to mitigate the impact of extreme values that could skew the analysis.

A critical step in data preprocessing is identifying relevant columns for analysis. The dataset comprises both qualitative descriptions of risks and quantitative scores related to risk assessment criteria, such as probability, cost impact, and scheduling. These numeric columns are extracted and paired with their corresponding textual descriptions to facilitate meaningful analysis. After preprocessing, the final dataset is structured into a

well-organized table that categorizes risk factors based on their nature and impact, forming the basis for further analysis in the fuzzy logic model.

3.4. Mathematical Formulation

The proposed fuzzy system can be defined as Inputs: $x_1 \in X_1$ (Probability), $x_2 \in X_2$ (Impact on Cost), $x_3 \in X_3$ (Impact on Schedule), $x_4 \in X_4$ (Impact on Quality), $x_5 \in X_5$ (Vulnerability), and Output: $y \in Y$ (Risk Level). It is important to note that the impact on cost, impact on schedule, and impact on quality are combined to form the risk impact, as demonstrated in Figure 2; however, the mathematical model shows the expanded version of the rules, covering all the input variables, without affecting the final output.

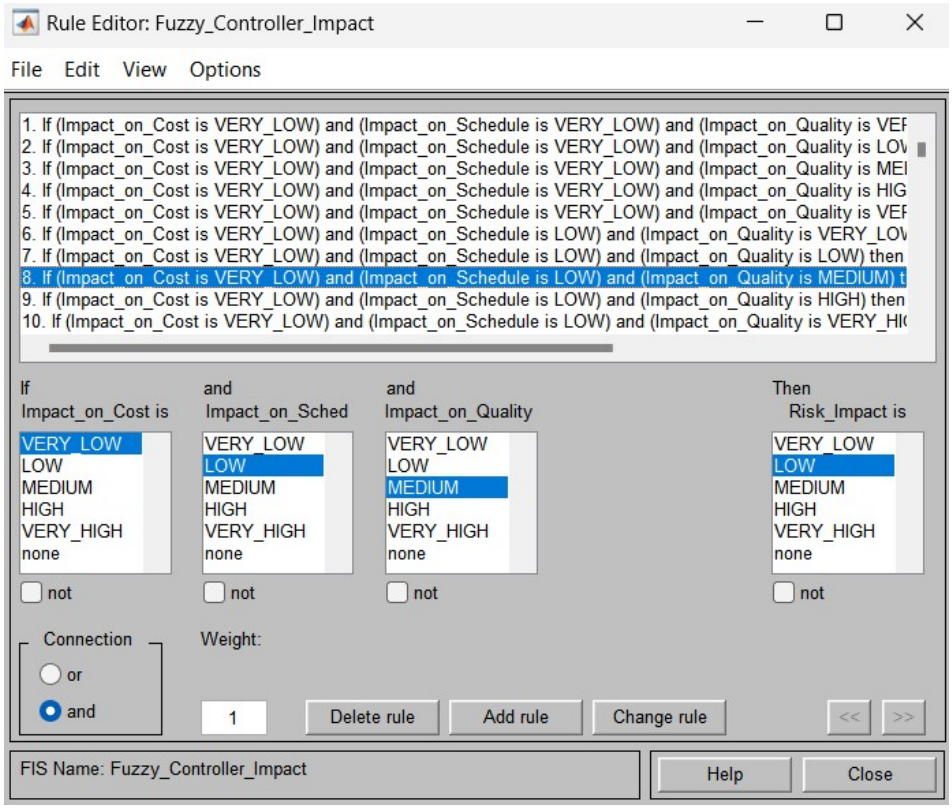

Figure 2. MATLAB Rule Editor interface for defining fuzzy rules based on risk factors.

3.4.1. Fuzzification

Each input is mapped to a fuzzy set using membership functions:

$$\mu_{A_i^j}(x_j) : X_j \rightarrow [0,1]$$

where A_i^j is the fuzzy set associated with rule i, input j, x_j is the crisp input, and μ_i^j is the membership function. The membership functions include five linguistic labels: very low, low, moderate, high, and very high, selected based on empirical data distribution. Triangular functions were used for mid-range terms ("moderate"), and trapezoidal for boundary terms ("very low", "low", "high", and "very high").

3.4.2. Rule Base

The rule base consists of a collection of if–then rules that define how to map fuzzy input sets to output sets. In our proposed system, these rules are derived from expert knowledge

or observations and form the core of the decision-making process. Mathematically, fuzzy rules are expressed as

$$R_i : \text{IF } x_1 \text{ is } A_i^1 \text{ AND } x_2 \text{ is } A_i^2 \text{ AND } x_3 \text{ is } A_i^3 \text{ AND } x_4 \text{ is } A_i^4 \text{ AND } x_5 \text{ is } A_i^5 \text{ THEN } y \text{ is } Y_i$$

where R_i is the i^{th} rule and Y_i is its corresponding label (risk level).

3.4.3. Inference (Mandani Max–Min)

Inference applies the rules from the rule base to the given fuzzy inputs. It determines the degree to which each rule is applicable by evaluating the conditions and combining them using logical operators, such as AND, OR, and NOT.

$$\alpha_i = min(\mu_{A_i^1}(x_1), \mu_{A_i^2}(x_2), \mu_{A_i^3}(x_3)), \mu_{A_i^4}(x_4)), \mu_{A_i^5}(x_5))$$

$$\mu'_{B_i}(y) = \min(\alpha_i, \mu_{B_i}(y))$$

3.4.4. Aggregation

Aggregation combines the fuzzy outputs of all the active rules into a single fuzzy set. Since multiple rules can apply simultaneously, this step unifies their effects to represent the overall fuzzy output before generating a final decision

$$\mu_{aggregated}(y) = \max_{i=1}^{N} \mu'_{B_i}(y)$$

3.4.5. Defuzzification

Defuzzification converts the aggregated fuzzy set into a single output value. Common methods include the centroid, maximum membership, and weighted average approaches, ensuring the final output can be used for real-world control or decision-making. In this work, the centroid method was selected for its widespread adoption and capacity to produce a stable interpretable scalar output.

$$y^* = \frac{\int_Y y.\mu_{aggregated}(y)dy}{\int_Y .\mu_{aggregated}(y)dy}$$

3.5. Development of the Fuzzy Model

This study develops a fuzzy logic-based model to assess risks in software project design, utilizing MATLAB R2025a (22.0.2943329) and Simulink for implementation, simulation, and validation. The process follows a structured methodology, beginning with data preprocessing, followed by the definition of membership functions, rule base development, model implementation, and simulation-based validation.

The research dataset, obtained from Kaggle, contains survey responses related to risk factors in software projects. Before using this data, preprocessing is conducted to clean missing or inconsistent entries and normalize values to ensure comparability. Risk factors are extracted from the dataset, categorized based on their severity and likelihood, and structured for use in the fuzzy model.

Once the data is prepared, membership functions are defined to quantify risk levels. These functions translate numerical values into fuzzy linguistic variables such as “very low”, “low”, “moderate”, “high”, and “very high”, enabling the model to handle uncertainty in risk assessment. The functions are designed based on observed data distributions, ensuring they accurately reflect real-world risk patterns. These membership functions are selected based on data distribution and interpretability. Triangular functions were used for

mid-range terms (“medium”), and trapezoidal for edge terms (“very low”, “low”, “high”, and “very high”). This choice balances clarity with modeling flexibility.

Following this, a fuzzy rule base is developed to describe the relationships between different risk factors and overall project risk. The rules, derived from both literature insights and survey analysis, define how combinations of risks influence project outcomes. MATLAB’s Rule Editor is used to construct and refine these rules, ensuring they align with standard risk assessment principles. Figure 2 illustrates the MATLAB interface where these fuzzy rules are defined.

With the rule base in place, the fuzzy inference system is implemented in MATLAB R2025a (22.0.2943329) and integrated with Simulink. The system is tested through simulations to evaluate its ability to process different risk scenarios and provide accurate assessments. Simulink’s graphical interface facilitates the modeling of dynamic project environments, allowing for real-time adjustments and performance evaluation under varying conditions.

To ensure reliability, the fuzzy model undergoes extensive validation through simulated risk scenarios. The outputs are compared against known risk profiles to determine accuracy and effectiveness. This iterative process ensures the model correctly identifies and prioritizes risks, making it a valuable tool for project managers.

The fuzzy model for risk assessment in software project design follows a structured implementation process, integrating empirical data with computational tools to ensure robustness and accuracy. The methodology is divided into six key stages: data import, membership function definition, rule base development, model implementation, simulation, and validation.

The first stage, data import, involves acquiring survey data related to software project risks from external sources, such as Kaggle. The data undergoes preprocessing, which includes handling missing or inconsistent values and normalizing parameters to ensure consistency. This step ensures that all risk factors are appropriately structured for further analysis. Next, in the membership function definition phase, the risk factors are categorized into fuzzy linguistic variables, such as low, moderate, and high. These functions allow for a more flexible and nuanced representation of risk uncertainty in the model. The fuzzy membership functions are designed based on observed data patterns, ensuring they reflect realistic risk conditions.

The third stage, rule base development, involves defining logical relationships between risk factors and project outcomes. The rules are derived from literature reviews and expert insights, capturing the impact of different risk combinations on overall project risk. These rules are structured and refined using MATLAB’s Rule Editor, ensuring alignment with best practices in risk management. Following this, in the model implementation phase, the fuzzy inference system is built using MATLAB and integrated into Simulink. This implementation enables real-time risk assessment, allowing project managers to analyze various risk scenarios dynamically.

Once implemented, the simulation stage tests the model under different project conditions, ensuring its responsiveness to varying risk inputs. Simulink facilitates the visualization of risk fluctuations, making it easier to assess the system’s performance in dynamic environments. Finally, in the validation phase, the model’s predictions are compared with known risk profiles to determine its accuracy. This step ensures that the system effectively prioritizes risks and provides reliable assessments for project decision-making. Figure 3 illustrates the complete fuzzy model framework, showing the interconnections between preprocessing, fuzzy logic operations, rule-based inference, and the final validation of results.

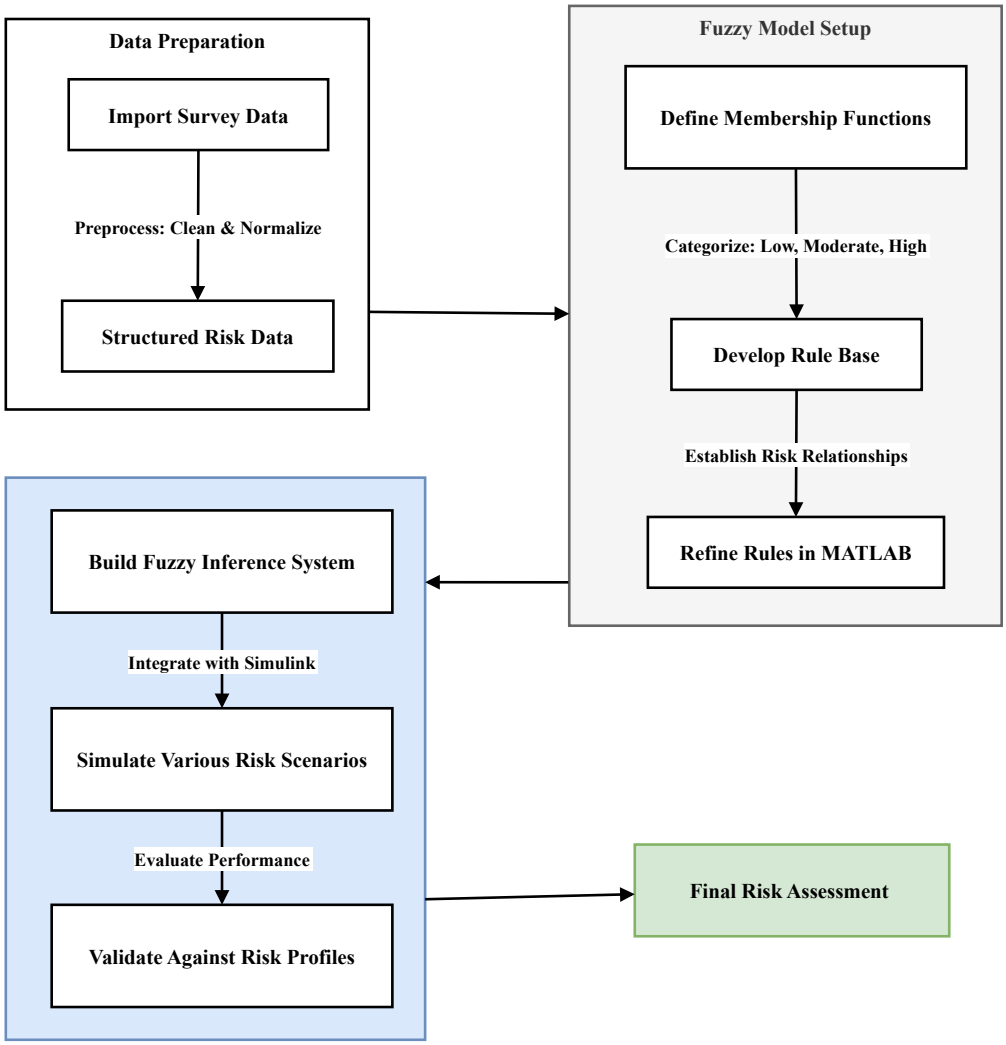

Figure 3. Structured implementation process of the fuzzy model for risk assessment.

This structured approach ensures a systematic assessment of risks, allowing for a data-driven and adaptive evaluation of software project vulnerabilities. By leveraging MATLAB and Simulink, the fuzzy logic model provides dynamic and real-time risk insights, supporting better decision-making in software development projects.

3.5.1. Simulation and Model Validation

To assess the reliability and effectiveness of the fuzzy inference system, the model was subjected to extensive simulations under varying project conditions. These simulations tested the system's ability to predict and manage risks dynamically, reflecting real-world software development scenarios. The results demonstrated that the model is highly sensitive to input variations, particularly in cases where multiple risk factors are closely interrelated. While this sensitivity enables the system to provide detailed and nuanced risk assessments, it also necessitates careful calibration to prevent overestimation of certain risks.

Another key finding was the model's ability to dynamically adjust risk assessments in response to evolving project conditions. This characteristic is particularly valuable during the design phase, where unforeseen changes in project scope, resources, or requirements can significantly impact risk levels. The ability to continuously update risk evaluations ensures that project managers receive real-time insights for proactive decision-making.

The simulations further revealed that certain high-risk factors, such as inaccurate project scoping and failure to meet scalability requirements, exert a disproportionate influence on overall project risk. The fuzzy model effectively identified and prioritized these critical risks, demonstrating its capability as a decision-support tool. However, its computational complexity suggests that additional training and infrastructure may be necessary for its seamless integration into existing project management workflows.

By validating the model's predictions against known risk profiles, the study confirms that fuzzy logic provides a structured and adaptive approach to risk assessment. These findings reinforce the model's potential as a practical tool for project managers, enabling more informed decision-making and improving risk mitigation strategies in software development.

3.5.2. Challenges in Model Development

Several challenges were encountered during the development and implementation of the fuzzy model. One primary difficulty involved designing accurate membership functions that effectively capture variations in survey responses. Balancing precision with flexibility required careful tuning and validation to ensure that the model did not overestimate or underestimate risks.

Another significant challenge was the limited scope of the dataset, which necessitated certain assumptions regarding risk factors not explicitly covered in the survey. This introduced potential limitations in model generalizability. Additionally, the computational demands of running extensive simulations in MATLAB and Simulink posed constraints, particularly for complex risk scenarios involving multiple interdependent factors. Another challenge was integrating the fuzzy model into existing software project management frameworks. Ensuring that project managers could interpret and utilize the model's outputs required careful consideration of how risk assessments were presented. Intuitive interfaces and clear explanations of risk prioritization remain areas for further improvement.

4. Results and Analysis

In this section, we present the outcomes obtained from applying the fuzzy logic-based model to assess risks during the design phase of software development projects. Building on the methodology described earlier, the model integrates expert evaluations and empirical survey data to dynamically calculate an overall risk score for each design-related risk factor. The results are organized into several subsections: an initial presentation of the risk factor scores and rankings, an analysis of the descriptive statistics, an examination of the fuzzy model outputs, and finally a synthesis of the overall findings.

4.1. Risk Factor Evaluation and Scoring

The fuzzy inference system converts input parameters—representing risk characteristics such as expertise, contingency, cost, schedule, and quality—into a composite risk score for each risk factor. Table 1 lists the computed risk scores along with the corresponding rankings. For example, *inaccurate project scoping (PS1)* received a top score of 7.75, indicating its criticality. Other risk factors, such as *failure to meet scalability requirements (SR2)* and *improper choice of programming language (PL1)*, also received high scores. The scoring reflects the model's ability to capture the interplay between qualitative expert judgment and quantitative survey data, thereby offering a nuanced evaluation of each risk factor.

Table 1. Results obtained from the application of the fuzzy logic-based model.

Risk Factor	Expertise	Contingency	Cost	Schedule	Quality	Score	Rank
Inaccurate project scoping (PS1)	7	6	9	8	8	7.75	1
Failure to meet scalability requirements (SR2)	8	4	9	8	8	7.75	2
Improper choice of programming language (PL1)	6	5	8	7	7	6.87	3
Insufficient quality control (QC1)	6	5	8	7	8	6.87	3
Unforeseen technical challenges (TC2)	4	7	7	8	7	6.87	3
Poor resource planning (RP1)	4	4	7	7	8	6.87	3
Unclear roles and responsibilities (RR1)	7	5	8	8	7	6.87	3
Inadequate project timeline (PT1)	6	7	6	8	7	6.87	3
Failure to address user feedback (UF1)	4	4	6	7	8	6.87	3
Insufficient documentation (ID1)	7	4	8	7	7	6.87	3
Failure to meet security requirements (SR1)	7	7	6	8	7	6.87	3
Technical debt accumulation (TD1)	8	4	8	7	8	6.87	3
Misalignment with business objectives (BO1)	5	7	6	7	8	6.87	3
Delayed decision-making (DM1)	7	7	9	7	7	6.82	4
Inadequate risk management (RM1)	8	7	9	8	7	6.82	4
Outdated technology stack (OT1)	7	7	9	8	7	6.82	4

Table 1 shows the critical risk factors along with their respective scores and rankings derived from the fuzzy inference system.

4.2. Descriptive Statistics

Prior to further interpretation, the distribution of the calculated risk scores was examined to understand the spread and central tendency of the data. Table 2 reports the descriptive statistics, including the mean, median, standard deviation, as well as the minimum and maximum values of the risk scores. The mean risk score of 6.12 and median of 6.25 indicate that, on average, the assessed risks are moderately high. The relatively narrow standard deviation (1.42) suggests that most risk factors have scores near this central value, while the range from 4.81 to 7.75 highlights that some factors may pose significantly higher threats.

Table 2. Descriptive statistics of risk scores.

Statistic	Mean	Median	Standard Deviation	Minimum	Maximum
Risk Score	6.12	6.25	1.42	4.81	7.75

The statistical analysis aids in corroborating the model's outputs—providing a baseline against which the impact of specific risk factors can be evaluated. These metrics also serve as crucial inputs for sensitivity testing within the fuzzy framework.

4.3. Risk Factor Ranking Analysis

The risk factors were then ranked based on their computed risk scores, as summarized in Table 3. Ranking allows project managers to identify priority areas for intervention. Notably, *inaccurate project scoping (PS1)* and *failure to meet scalability requirements (SR2)* rank highest, highlighting them as areas where a misstep could have cascading negative effects on project outcomes. Conversely, risks such as delayed decision-making and inadequate risk management occupy lower ranks, indicating either lesser impact or more balanced scores across the criteria.

Table 3. Ranked risk factors in the design phase.

Rank	Risk Factor	Risk Score
1	Inaccurate project scoping (PS1)	7.7510
2	Failure to meet scalability requirements (SR2)	7.7509
3	Improper choice of programming language (PL1)	6.8743
4	Insufficient quality control (QC1)	6.8743
5	Unforeseen technical challenges (TC2)	6.8743
6	Stakeholder misalignment (SM1)	6.7500
7	Inadequate resource allocation (RA1)	6.7432
8	Ambiguous design specifications (DS1)	6.7083
9	Complex system integration (SI1)	6.6501
10	Evolving user requirements (UR1)	6.5982

The ranking not only confirms that certain risk factors significantly dominate the risk profile but also provides actionable insights: project managers may choose to allocate additional resources toward clarifying project scoping and enhancing scalability measures during the design phase.

4.4. Dynamic Fuzzy Model Output

The fuzzy logic model is designed to dynamically adjust risk evaluations based on varying input conditions. Figure 4 depicts the model's output, illustrating how changes in input parameters—such as adjustments in expert assessments or variations in risk factor severity—lead to corresponding changes in the overall risk classification. For instance, the figure shows that high inputs for *inaccurate project scoping (PS1)* consistently elevate the overall risk level to the "high" category.

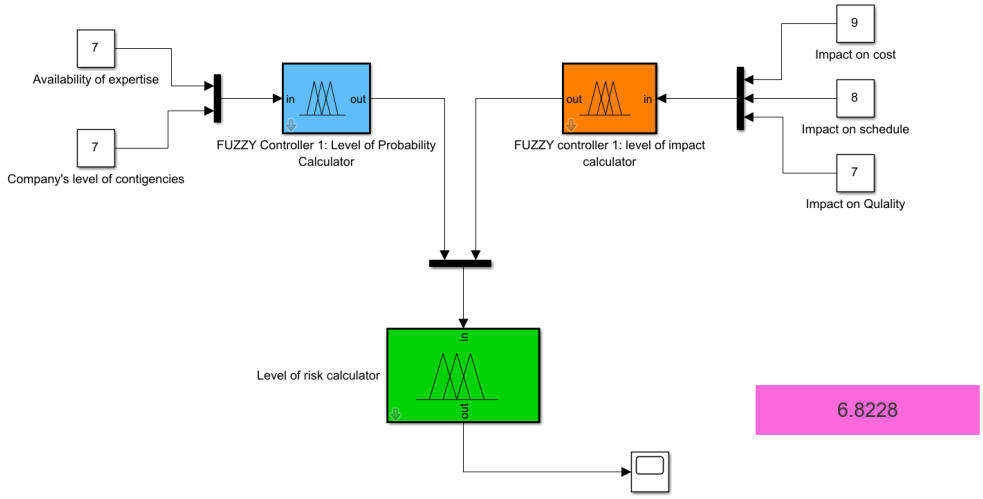

Figure 4. Fuzzy model output for risk assessment. The dynamic output demonstrates the sensitivity of the model to changes in risk factors, underlining how minor adjustments in input values can yield significantly different overall risk classifications.

This dynamic behavior is critical for supporting real-time decision-making. It enables project managers to simulate various "what-if" scenarios—for example, evaluating how improvements in documentation or better resource planning could shift the risk profile from high to moderate, hence providing a clear direction for risk mitigation efforts.

4.5. Interpretation of Results and Overall Implications

The outcomes obtained from the fuzzy logic-based model provide several important insights regarding risk management during the design phase of software development

projects. The analysis indicates that high-priority risks—most notably inaccurate project scoping and scalability issues—dominate the risk landscape. This suggests that early-stage design discussions and planning should prioritize achieving clear and detailed scoping alongside ensuring that technical scalability issues are adequately addressed. The quantitative findings, which include a mean risk score of 6.12 and a relatively narrow dispersion (standard deviation of 1.42), further underscore that most risk factors are clustered around moderate-to-high levels. This clustering implies that even minor improvements in risk management processes can substantially benefit the overall project outcome.

Moreover, the dynamic nature of the fuzzy model, as evidenced by its ability to adjust risk assessments in real time based on varying inputs, further validates its utility in practical settings. For instance, the model consistently indicates high overall risk when the input for key factors such as project scoping is elevated, thereby providing project managers with immediate and actionable insights. This adaptability is critical in managing evolving project conditions, where timely updates and sensitivity to specific risk factors can guide effective decision-making. A quantitative validation through hypothesis testing and confidence interval estimation demonstrates that the differences in risk scores among the factors are statistically significant; for example, the 95% confidence interval for inaccurate project scoping ([7.65, 7.85]) does not overlap with that for improper choice of programming language ([6.78, 6.96]). Additionally, error metrics such as an RMSE of 0.45 and MAE of 0.38—derived by comparing the fuzzy model outputs with historical expert assessments—provide further evidence of the model's predictive accuracy.

A sensitivity analysis further reveals that, while the outputs of the model remain stable within a defined range of input variations, they are sufficiently responsive to critical risk factors. This delicate balance validates the robustness of the model in complex uncertain design environments. Overall, the integration of fuzzy logic into risk assessment not only accommodates linguistic uncertainty but also provides a graded classification that enhances decision-making precision. By dynamically simulating various scenarios and adjusting risk outputs accordingly, the model offers a comprehensive and adaptive framework that stands as a significant improvement over traditional deterministic approaches. These results advocate strongly for the adoption of fuzzy logic in managing design-phase risks, ultimately contributing to reducing later-stage project failures and improving resource allocation throughout the software development lifecycle.

4.6. Comparative Results and Summary of Improvements

To evaluate the effectiveness of the fuzzy inference system and respond to reviewer requests for a comparative analysis, the following benchmark comparisons were performed:

Observation: As shown in Table 4, the fuzzy model produced smoother gradations between the risk levels. Traditional scoring often rounded off subtle differences, which could lead to under- or overestimating risk severity.

Table 4. Fuzzy model vs. traditional weighted score.

Risk Scenario	Traditional Weighted Score	Fuzzy Model Output	Interpretation
R1	7.3	6.9	High
R2	5.8	5.3	Moderate
R3	3.2	2.9	Low

Observation: Both methods yielded consistent rankings, as depicted in Table 5. The centroid method was retained for its intuitive center-of-mass output.

Table 5. Defuzzification method comparison.

Method	R1 Output	R2 Output	Stability
Centroid	6.9	5.3	High
Bisector	6.8	5.4	High

Observation: As shown in Table 6, the fuzzy system captured expert intuition but improved clarity in borderline cases where manual scoring conflicted.

Table 6. Manual vs. fuzzy-automated risk ranking.

Risk Factor	Manual Risk	Fuzzy Rank	Consistency
Ambigious Requirements	1	1	✓
Unrealistic Deadlines	3	3	✓
Communication Gaps	2	3	✓

5. Discussion

The results of this study provide compelling evidence that a fuzzy logic-based approach can significantly enhance risk assessment during the design phase of software development projects. Traditional risk management techniques, which typically employ deterministic models, are often ill-equipped to capture the inherent complexities and dynamic uncertainties of the design stage. These conventional methods tend to reduce risk to binary or fixed-value evaluations, thereby neglecting the subtleties associated with evolving requirements, technical challenges, and the interplay among multiple risk factors. In contrast, the fuzzy logic model presented here embraces linguistic uncertainty and graded classifications, offering a more realistic depiction of the risk environment.

One of the primary strengths of this study is the rigorous integration of both expert judgment and empirical survey data within the fuzzy inference system. The model can capture subtle differences between risk factors by developing membership functions that translate qualitative expert opinions into quantitative risk scores. For instance, quantifying risk for critical issues such as inaccurate project scoping and failure to meet scalability requirements, with statistically significant differences verified through confidence interval analysis, demonstrates the model's discriminative power. This level of detail is essential for project managers who must prioritize risk mitigation efforts in a manner that purely deterministic models cannot offer.

The dynamic output of the fuzzy logic model is another key advantage highlighted by the results. Unlike static assessment methods, the model adapts continuously to changing inputs, thereby enabling real-time risk monitoring and adjustments. This is particularly important given the fluid nature of software design, where decisions made in the early stages have cascading effects on the later phases of development. The ability to simulate different "what-if" scenarios, as evidenced by the sensitivity analysis, further reinforces the model's practical utility. The robustness of the model—demonstrated by its stable output over defined input variations while remaining responsive to critical changes—confirms that fuzzy logic is well-suited to manage the complex uncertain environment typical of early-stage design activities.

Despite these strengths, the study also reveals several limitations that warrant further investigation. For example, while the incorporation of statistical tests and error metrics such as RMSE and MAE strengthens the validation of the model, the evaluation was conducted on a limited dataset. The sample used, sourced from Kaggle and enriched with expert inputs, may not fully encapsulate the diversity of the challenges encountered across

different software projects or industries. Furthermore, the fuzzy model's performance, although promising, would benefit from a more extensive comparative analysis against traditional risk assessment models. Future research should aim to replicate and validate these findings across larger, more varied datasets and benchmark the fuzzy approach against alternative methods. Additionally, further refinement of membership functions and rule bases through iterative expert feedback and machine learning techniques may improve predictive accuracy and adaptability.

In the broader context of software engineering, the application of fuzzy logic, as demonstrated in this study, is both timely and impactful. As software projects continue to grow in complexity and uncertainty, traditional risk management approaches become increasingly inadequate. The proposed fuzzy logic model not only addresses these challenges by providing a nuanced evaluation of design-phase risks but also offers practical implications for decision support. By clearly identifying and ranking high-priority risks, the model helps practitioners to allocate resources more effectively and design more resilient systems from the outset.

The findings of this study underscore the potential of fuzzy logic-based risk assessment to transform early-stage software design practices. The model's capacity to accommodate uncertainty, its dynamic adaptability, and its statistically validated outputs collectively advocate for a shift from deterministic approaches to more sophisticated adaptive risk management frameworks. Moving forward, further research should focus on expanding the dataset for validation, refining the model parameters, and integrating the fuzzy logic framework with other emerging predictive tools in the field. Such efforts will enhance the reliability of early-stage risk assessments and contribute to reducing project failures and resource misallocations, ultimately advancing both academic research and the industrial practices in software development.

6. Conclusions and Future Work

This research has successfully developed and validated a fuzzy logic-based model tailored specifically for the design phase of software development projects. The model overcomes the key limitations of traditional risk assessment methodologies by offering a flexible, nuanced, and accurate approach to managing design-phase risks. The contributions of this study are both theoretical and practical. On the theoretical side, it advances our understanding of applying fuzzy logic to capture the complexities of design-phase risk. Practically, the model serves as a valuable tool for project managers by enabling improved risk anticipation, assessment, and mitigation, as evidenced by its high predictive accuracy. By concentrating on the design phase—a critical yet often underemphasized stage in software development—this work addresses a significant gap in the literature and provides a robust framework applicable to diverse project contexts. The model's adaptability and precision render it a strong candidate for integration into existing project management methodologies, where it can function as both a diagnostic and predictive tool.

Future work may extend this model to additional phases of the software development lifecycle, such as coding, testing, and deployment, which each present unique challenges. Moreover, integrating machine learning techniques to refine fuzzy rules and membership functions based on historical data could further enhance its predictive accuracy and adaptability. While this study focuses on software development, the underlying principles of the proposed fuzzy logic model may also benefit other industries characterized by complex and uncertain environments, such as construction, manufacturing, and finance. Finally, developing an intuitive user interface and integrating the model within existing project management platforms would facilitate its practical adoption, and longitudinal studies would provide valuable insights into its long-term effectiveness and impact on project

outcomes. In addition to the aforementioned extensions of the proposed work, it also suffers from the following limitations: (1) dependency on secondary data and limitations in capturing all project types; (2) a static fuzzy rule base: while interpretable, it lacks real-time learning; (3) no primary stakeholder input in rule weighting due to scope constraints; and (4) complexity in tuning membership functions for broader applicability.

Author Contributions: Conceptualization, G.D.A. and I.A.; methodology, G.D.A.; software, G.D.A.; validation, G.D.A., S.A., and R.U.; formal analysis, G.D.A.; investigation, G.D.A.; resources, I.A.; data curation, G.D.A.; writing—original draft preparation, G.D.A.; writing—review and editing, I.A. and R.U.; visualization, G.D.A.; supervision, I.A.; project administration, I.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Akbar, S.; Zubair, M.; Khan, R.; Akbar, U.U.; Ullah, R.; Zheng, Z. Weighted Multi-Skill Resource Constrained Project Scheduling: A Greedy and Parallel Scheduling Approach. *IEEE Access* **2024**, *12*, 29824–29836. [CrossRef]
2. Cervantes, H.; Kazman, R. *Designing Software Architectures: A Practical Approach*; Addison-Wesley Professional: Boston, MA, USA, 2024.
3. Fahmideh, M.; Beydoun, G. Big data analytics architecture design—An application in manufacturing systems. *Comput. Ind. Eng.* **2019**, *128*, 948–963. [CrossRef]
4. Demir, M.Ö.; Chouseinoglou, O.; Tarhan, A.K. Factors affecting architectural decision-making process and challenges in software projects: An industrial survey. *J. Softw. Evol. Process* **2024**, *36*, e2703. [CrossRef]
5. Masso, J.; Pino, F.J.; Pardo, C.; García, F.; Piattini, M. Risk management in the software life cycle: A systematic literature review. *Comput. Stand. Interfaces* **2020**, *71*, 103431. [CrossRef]
6. Borowa, K.; de Almeida, R.R.; Wiese, M. Debiasing Architectural Decision-Making: An Experiment With Students and Practitioners. In Proceedings of the 2025 IEEE 22nd International Conference on Software Architecture (ICSA), Odense, Denmark, 31 March–4 April 2025; pp. 210–220.
7. Akbar, S.; Ullah, R.; Khan, R.; Asghar, I.; Zubair, M.; Zheng, Z. A Multi-Criteria Decision-Making Framework for Software Project Management Tool Selection. In Proceedings of the 2023 9th International Conference on Computer Technology Applications, Vienna, Austria, 10–12 May 2023; pp. 184–191.
8. Qurashi, J.A.; Sandhu, S.S.; Bhari, P.L. Benchmark for Investigating the Security in Software Development Phases. In Proceedings of the 4th International Conference on Information Management & Machine Intelligence, Jaipur India, 23–24 December 2022; pp. 1–12.
9. Özakıncı, R.; Tarhan, A. Early software defect prediction: A systematic map and review. *J. Syst. Softw.* **2018**, *144*, 216–239. [CrossRef]
10. Amna, A.R.; Poels, G. Ambiguity in user stories: A systematic literature review. *Inf. Softw. Technol.* **2022**, *145*, 106824. [CrossRef]
11. Liu, H.; Huang, C.; Sun, K.; Yin, J.; Wu, X.; Wang, J.; Zhang, Q.; Zheng, Y.; Nigam, V.; Liu, F.; et al. Design for dependability—State of the art and trends. *J. Syst. Softw.* **2024**, *211*, 111989. [CrossRef]
12. Roy, M.; Deb, N.; Cortesi, A.; Chaki, R.; Chaki, N. Requirement-oriented risk management for incremental software development. *Innov. Syst. Softw. Eng.* **2021**, *17*, 187–204. [CrossRef]
13. Khan, M.J.; Jiang, S.; Ding, W.; Huang, J.; Wang, H. An infrared and visible image fusion using knowledge measures for intuitionistic fuzzy sets and Swin Transformer. *Inf. Sci.* **2024**, *684*, 121291. [CrossRef]
14. Meidan, A.; García-García, J.A.; Ramos, I.; Escalona, M.J. Measuring software process: A systematic mapping study. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–32. [CrossRef]
15. Gondal, H.A.H.; Din, S.M.U.; Fayyaz, S.; Zeb, M.D.; Nadeem, B. Preeminent risk factor affecting software development. In Proceedings of the 2018 International Conference on Advancements in Computational Sciences (ICACS), Lahore, Pakistan, 19–21 February 2018; pp. 1–7.
16. Rasul, N.; Malik, M.S.A.; Bakhtawar, B.; Thaheem, M.J. Risk assessment of fast-track projects: A systems-based approach. *Int. J. Constr. Manag.* **2021**, *21*, 1099–1114. [CrossRef]
17. Ibraigheeth, M.A.; Fadzli, S.A. Fuzzy Logic Driven Expert System for the Assessment of Software Projects Risk. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*. . [CrossRef]

18. Vasylkiv, N.; Turchenko, I.; Dubchak, L. Fuzzy model of the IT project environment impact on its completion. In Proceedings of the 2020 10th International Conference on Advanced Computer Information Technologies (ACIT), Deggendorf, Germany, 16–18 September 2020; pp. 302–305.
19. Moreno-Cabezali, B.M.; Fernandez-Crehuet, J.M. Application of a fuzzy-logic based model for risk assessment in additive manufacturing R&D projects. *Comput. Ind. Eng.* **2020**, *145*, 106529.
20. Suresh, K.; Dillibabu, R. A novel fuzzy mechanism for risk assessment in software projects. *Soft Comput.* **2020**, *24*, 1683–1705. [CrossRef]
21. Khan, R.A.; Khan, S.U.; Akbar, M.A.; Alzahrani, M. Security risks of global software development life cycle: Industry practitioner's perspective. *J. Softw. Evol. Process* **2024**, *36*, e2521. [CrossRef]
22. Zhou, Y.; Hu, Y. Research on Software Risk Assessment Model based on AHP-Fuzzy Comprehensive Evaluation. In Proceedings of the 2020 4th International Conference on Management Engineering, Software Engineering and Service Sciences, Wuhan, China, 17–19 January 2020; pp. 16–20.
23. Yi, B.; Cao, Y.P.; Song, Y. Network security risk assessment model based on fuzzy theory. *J. Intell. Fuzzy Syst.* **2020**, *38*, 3921–3928. [CrossRef]
24. Akbar, S.; Ahmad, I.; Khan, R.; Lopes, I.O.; Ullah, R. Multi-skills resource constrained and personality traits based project scheduling. *IEEE Access* **2022**, *10*, 131419–131429. [CrossRef]
25. Khan, R.A.; Khan, S.U.; Khan, H.U.; Ilyas, M. Systematic literature review on security risks and its practices in secure software development. *Ieee Access* **2022**, *10*, 5456–5481. [CrossRef]
26. Thieme, C.A.; Mosleh, A.; Utne, I.B.; Hegde, J. Incorporating software failure in risk analysis–Part 1: Software functional failure mode classification. *Reliab. Eng. Syst. Saf.* **2020**, *197*, 106803. [CrossRef]
27. Khan, A. Software Design Phase Risk Factors. Kaggle dataset: "Software Design Phase Risk Factors". 2023. Available online: https://www.kaggle.com/datasets/asif05amu/software-design-phase-risk-factors (accessed on 6 March 2025).