Gradus Vol. 12, No. 2 (2025)
ISSN 2064-8014

A Comprehensive Survey on Fuzzy Logic Applications in Software Requirements Engineering

Zsolt Csaba Johanyák[1,2,*] and **Zlatko Čović**[3]

[1] Department of Information Technologies, GAMF Faculty of Engineering and Computer Science, John von Neumann University, Hungary
[2] Institute of Mechatronics and Vehicle Engineering, Bánki Donát Faculty of Mechanical and Safety Engineering, Óbuda University, Hungary
[3] Subotica Tech - College of Applied Sciences, Serbia

https://doi.org/10.47833/2025.2.CSC.008

Keywords:
fuzzy logic,
requirements engineering,
software engineering,
uncertainty management

Article history:
Received: 13 October 2025
Revised: 15 November 2025
Accepted: 17 November 2025

Abstract: *The process of software development extends beyond defining system functions, as user requirements are often expressed in ordinary language that can be unclear or open to varying interpretations. The research reported in this paper examines how fuzzy logic enables the main Software Requirements Engineering processes to convert ambiguous and subjective inputs into quantifiable and structured data. This approach enables stakeholders to achieve clearer requirements by using fuzzy sets, linguistic terms, and inference rules to detect inconsistencies early and support decision-making. The results indicate that fuzzy techniques need domain experts to establish essential parameters, and they require additional computational resources to operate effectively as a promising solution for managing uncertainty in requirements engineering.*

1. Introduction

The first step in software development is to identify the need of the customer, i.e. determine exactly what the system must accomplish. This activity is called software requirements engineering (Requirement Engineering, RE) [1] and appears mainly in the early development phases. These phases include the identification, analysis, documentation, verification, and management of customer requirements. The main goal of RE is to ensure that all stakeholders-whether they are clients, project managers, designers, or developers-reach a common understanding of the software goals. Although this shared vision may seem straightforward, it remains difficult to achieve because of differences in communication styles and the natural complexity inherent to software projects.

One of the biggest challenges found in RE is the requirement ambiguity. When stakeholders express their expectations in an everyday language, misunderstandings can occur easily [2]. For example, expressions such as "the system must have high availability," "it must respond quickly," or "the interface must be user-friendly" are often used, but they lack measurable definitions.

For instance, the phrase "high availability" can be interpreted differently by different people, which could lead to misunderstandings, conflicting expectations, communication problems, and finally the produced software does not meet user needs.

Conventional methods for managing requirements depend on rigid binary logic, assuming that information must be entirely precise before advancing to subsequent development stages. This expectation becomes problematic when the available information is incomplete or subjective. Ideas are forced into true-or-false categories, which often leads to errors, missing features, and dissatisfied users. Since projects continuously become larger and more complex, these rigid methods are increasingly unable to handle the uncertainty and variety found in real situations [3].

Several researchers suggested that the concept of fuzzy logic (FL) could offer a proper tool to address this problem. Unlike traditional logic, which recognizes only absolute truth or falseness, fuzzy logic permits thinking in degrees. Thus, partial truths between 0 and 1 can be considered. For example,

* Corresponding author: johanyak.csaba@nje.hu

instead of simply restricting the evaluation of the speed of a website to "fast" or "slow," fuzzy logic allows the definition of intermediate categories (values) as well like slow, moderately slow, slightly slow, medium, slightly fast, moderately fast and in the case of each of them, a numeric value can be specified that indicates how sure the speaker is about associating the current speed with the given label. This approach reflects more closely the way people naturally describe and evaluate their experiences [4].

Several benefits arise when vague human expressions are transformed into machine-readable form through fuzzy logic:

- *Priority determination*: Requirements can be ranked based on values given by users [5].
- *Feasibility assessment*: The system determines whether desired functions can be performed without applying rigid binary judgments [6].
- *Conflict detection*: Contradictions between loosely defined requirements are identified early in the process [7].

Although fuzzy logic has advantages, widespread adoption has not been reported in industrial software development so far [8]. Thus, a significant gap remains between research theories and practice of daily development.

This paper addresses that gap by systematically reviewing the application of fuzzy logic across the five core stages of requirements engineering: elicitation, analysis, specification, validation, and management. Beyond presenting existing methods, we discuss their strengths, limitations, and integration with complementary techniques such as social network analysis, case-based reasoning, and goal-oriented modeling.

The rest of this paper is organized as follows. Section 2 covers background information related to fuzzy logic and RE. Section 3 presents the reported applications of FL in the different steps of RE based on the available literature. Section 4 synthesizes these findings, highlighting how fuzzy logic improves transparency, reduces ambiguity, and improves decision-making throughout the requirement lifecycle. Section 5 concludes with key challenges—such as reliance on expert knowledge and lack of standardized tools—and outlines future directions, including automated membership function generation, improved tool support, and broader industrial adoption. By bridging theoretical foundations with practical considerations, this survey aims to provide a comprehensive framework for leveraging fuzzy logic in human-centered, uncertainty-aware software development.

2. Background and Theoretical Foundations

2.1. Fuzzy Logic

Lotfi Zadeh proposed fuzzy logic in 1965 [9] as a mathematical approach to manage scenarios characterized by uncertainty, vagueness, or partial truth. While traditional Boolean logic classifies objects/values as either "true" (1) or "false" (0), i.e. a given value either belongs a to a set (category) or not, FL makes possible a more granulated expression of one's opinion by the help of membership values from the interval [0, 1]. This feature allows for a more accurate modeling of human thinking. The application of interval fuzzy modeling enables the representation of linear dynamic system uncertainties [10] as well.

For example, lets's assume a system that has to notify users of incoming messages or alerts within 5 seconds of receipt. The satisfaction of this requirement is measured with the help of a so-called linguistic variable named *Real-Time Notifications*. Five levels, i.e., linguistic terms, were defined for this task. They are *Instantaneous*, *Prompt*, *Adequate*, *Tolerable*, *Unacceptable*. Next, for each linguistic term a membership function is defined that expresses on a domain (also called universe of discourse) how much each delivery time value of that domain belongs to that category. The linguistic term and the underlying membership function together are called fuzzy set. The group of fuzzy sets defined on a universe of discourse is called a partition. A given value of the universe of discourse can belong to more than one fuzzy sets. In our case (see Fig. 1), if the delivery time is 6 sec it can be characterized as *Prompt* with a membership value of 0.8 (denoted as $\mu_{Prompt}(6) = 0.2$) and as *Adequate* with a membership value of 0.2 (denoted as $\mu_{Adequate}(6) = 0.2$). In this example, so called triangle type membership functions were used. However, one can choose from wide range of types when designing a fuzzy system. The most used types are: singleton, triangular, trapezoidal, gaussian, bell-shaped, sigmoidal, z-shaped, s-shaped, pi-shaped, polynomial, exponential. Membership functions typically are described by specifying their parameters. For example, the membership function of the fuzzy set with the linguistic term *Prompt* is $\mu_{Prompt}(x) = \{0, 5, 10\}$, because its parameters are the breakpoints of the curve describing it.

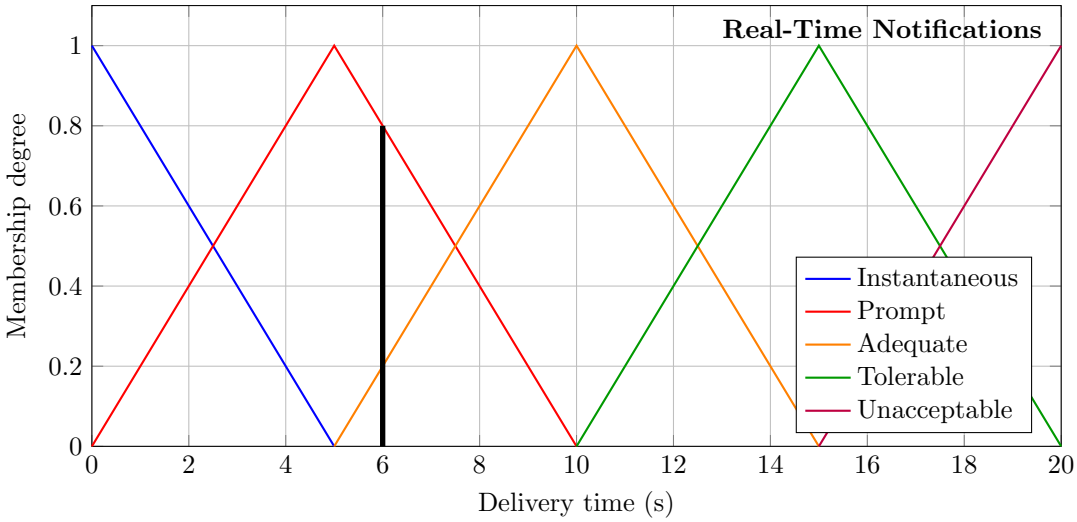

Figure 1: Fuzzy partition for the linguistic variable Real-Time Notifications.

The foundation of fuzzy logic systems rests upon reasoning that is based on "if-then" rules. This reasoning pattern mimics human thinking. Let's assume that the satisfaction level of the user described by the linguistic variable Requirement Satisfaction Level (linguistic terms: *Very Poor*, *Poor*, *Fair*, *Good*, *Excellent*) is determined by two factors that can be described by the linguistic variables Real-Time Notifications and UI Responsiveness (linguistic terms: *Instantaneous*, *Fast*, *Moderate*, *Slow*, *Very Slow*). An example rule describing the relation between these variables is

R : **IF** *Real-Time Notifications* **is** *Adequate* **AND** *UI Responsiveness* **is** *Moderate* **THEN** *Requirement Satisfaction Level* **is Fair**.

In the case of rules with two premises each having five linguistic values $5 * 5 = 25$ rules are necessary to fully describe the input-output relation. The collection of the available rules is called a rule base. Due to possible non-zero memberships in two fuzzy sets in case of a given input it is possible that at least two rules are used to determine the final outcome (Satisfaction level in our case). This process is called fuzzy inference. Although several inference methods are available Mamdani type inference [11]. Its key idea is that first a firing strength is determined for each rule.

$$\alpha_k = \min\left(\mu_{A_1^k}(x_{RTN}),\ \mu_{A_2^k}(x_{UI})\right), \quad \forall k \in \{1, \ldots, n_R\}, \tag{1}$$

where A_1^k and A_2^k are the first and second antecedent sets of the kth rule, x_{RTN} and x_{UI} are the current input values, and n_R denote the number of rules. In the equation, the abbreviated linguistic variable names are used to make it more compact and readable. Next, the aggregated output membership function is calculated as

$$\mu_{\mathrm{B}}(y) = \max_k \left(\min\left(\alpha_k,\ \mu_{B^k}(y)\right)\right), \quad \forall k \in \{1, \ldots, n_R\}, \tag{2}$$

where $\mu_{B^k}(y)$ is the consequent fuzzy set of the of the kth rule. Finally, the resulting membership function is defuzzified, i.e. a crisp output value is calculated from the previously aggregated membership function. Here, one can choose from several methods, like center of area, bisector of area, mean of maximum, smallest of maximum), largest of maximum, weighted average, height method, center of sums, center of largest area, first of maximum, etc. The equation of the center of area method [12] is

$$y^* = \frac{\int_Y y\,\mu_{\mathrm{B}}(y)\,\mathrm{d}y}{\int_Y \mu_{\mathrm{B}}(y)\,\mathrm{d}y}, \tag{3}$$

where y^* is the defuzzified crisp value.

2.2. Requirements Engineering Fundamentals

One of the most important and systematic processes in software development is requirements engineering (RE). The goal of this process is to discover, precisely document, and maintain software needs throughout the entire development process [13]. According to the INCOSE (International Council on Systems Engineering), requirements engineering serves as a cornerstone of systems engineering practice. It specifies what the system must be able to do and the constraints within which it must operate. The process consists of five main steps [14]:

- *Elicitation*: gathering user needs and expectations,
- *Analysis*: checking requirements for completeness, contradictions, and consistency,
- *Specification*: precisely and clearly documenting requirements,
- *Validation*: ensuring requirements match user expectations,
- *Management*: tracking changes and versions throughout the project.

The connection between these steps is presented in Fig. 2.

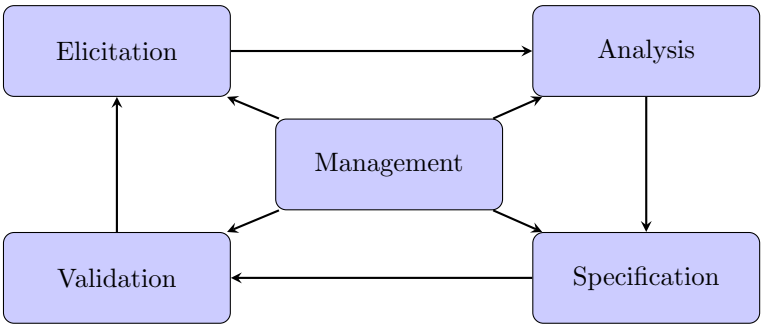

Figure 2: The Requirements Engineering process [15].

As modern software systems usually are complex, the development environment changes continuously. Organizations must work with many stakeholders, and these stakeholders often see priorities differently. Meanwhile, market, security, and usability demands must be met.

Classical RE approaches apply binary logic. Therefore, they are inadequate for processing requirements expressed in uncertain or naturally ambiguous language. Expressions like "user-friendly," "fast," or "secure" are necessary but difficult to measure. This situation causes serious problems during the acceptance (validation) process.

Systematic solutions to these problems are offered by fuzzy logic principles. They allow requirements to be evaluated not only on "meets/does not meet" grounds, but also according to satisfaction levels. A mathematical background for managing uncertainties, inaccuracies, and subjective factors is provided by fuzzy tools. In this way, the system can connect the logic of human language with technical specifications.

3. Fuzzy Logic Applications in Requirements Engineering

A fuzzy logic based requirements engineering support could be particularly useful because the theoretical foundations of FL enable the creation of models that account for human judgment errors and real uncertainties occurring during software development. Thus the development team can handle uncertainty at every stage, and uncertainty can be addressed from discovering needs to project management.

3.1. Elicitation

The elicitation stage of RE focuses on collecting user expectations about system behavior. This task is challenging because people often struggle to articulate their requirements clearly [7]. Fuzzy logic proves to be particularly valuable here. It can formally handle the vagueness and ambiguity inherent in natural language [7].

When describing their needs, users typically employ words that can easily be interpreted as linguistic values (e.g. "low," "medium," "high"). These terms are assigned membership functions. Due to this

assignment, rigid decisions between two extremes are not necessary. A fuzzy system based approach allows the developer team to evaluate the described needs following human thinking patterns [5].

When a user says, "The system should be fast enough, but not too complicated,", an analyst can assign fuzzy linguistic terms and membership functions to these characteristics. For example, $\mu_{fast} = \{2, 3, 4\}$ and $\mu_{complicated} = \{1, 2, 3\}$ might be assigned. These values then become usable in measurable form. Research has shown that fuzzy sets, when combined with social network theory [1], help identify the most important stakeholders in a project. When many actors are involved and differing opinions appear this approach is particularly helpful. Language uncertainty and vague expressions also can be addressed effectively. For example, a concrete value can be assigned by an analyst to a fuzzy phrase ("early in the month = around the 5th day"). The participants then rate, on a 0-to-10 scale, how acceptable this definition is to them. Finally, this rating is converted into a fuzzy membership value. If the rating indicates that the requirement cannot be reliably validated, it must be clarified or reformulated. Extended fuzzy sets, such as intuitive or neutrosophic fuzzy sets, are increasingly recommended by future research [7]. These sets can handle hesitation alongside uncertainty.

3.2. Analysis

The goal of the analysis step is to review the collected requirements for completeness, clarity, and consistency, as well as to prioritize requirements, and to identify dependencies. Because this process often contains complex and imprecise information, fuzzy logic is an ideal tool to manage it [7]. Often, requirement selection appears as a multi-criteria decision problem, where many factors must be considered simultaneously. The Fuzzy TOPSIS method [16] can be applied by analysts for this purpose. Thus, requirements can be ranked even when data is uncertain. For example, security needs can be ranked using this method. Stakeholders can use fuzzy decision-making to evaluate competing system requirements through analysis of both system objectives and limitations [17].

A major difficulty is encountered in analysis because requirements often have value dependencies. When one requirement is changed or deleted, others are affected. These connections are difficult to define precisely. The fuzzy-based requirement selection (FBRS) method [18] addresses this issue by using so called Fuzzy value graphs (FVG) and Integer Linear Programming (ILP). Value dependencies can be represented in a vague but manageable way through these graphs. In addition, Complex Fuzzy Logic (CFL) systems can generalize the fuzzy space to serve as a medium for clustering specifications-related information using a feature space based on linguistic variables [19]. For instance, a centroid update approach to K-means clustering was proposed to improve cluster center accuracy in dynamic datasets [20].

Non-functional requirements can be ranked by importance using fuzzy-soft set theory [8]. Performance, security, and usability are examples of such requirements.

3.3. Specification

Requirements become precise, documented descriptions during this phase. Fuzzy logic helps greatly in this phase. It is especially useful when many factors must be coordinated and multiple complex trade-offs that contain inherent uncertainty must be dealt with. Cost, quality, and time are examples of such factors. They are not always clearly measurable. Complex fuzzy logic (CFL) [19] can manage relationships simultaneously. For example, "good quality but high cost and yet difficult to implement" can be represented. Traditional, one-dimensional fuzzy logic is sometimes insufficient. Multiple properties can each have their own fuzzy value through CFL. For instance, "This feature is very useful, but implementation is extremely difficult" can be expressed in this way.

Security requirements can also be modeled using fuzzy logic by using the Security Requirement Models (SRMs) proposed by [21]. Some needs can be met partially or gradually. Furthermore the Goal-oriented requirements engineering (GORE) can also be extended using fuzzy tools [22]. Goals, such as cost, deadline, and quality, are covered by linguistic variables in this approach. The desired goal state, which is called fuzzy desired situation (FDS). The overall satisfaction measure increases as the similarity values between the required level of an attribute and the FDS increase.

Linguistic priorities ("weak," "normal," "strong") are determined by a fuzzy inference system (FIS) in the PAPS framework, for example. Then these values are converted into a required degree of satisfaction (RDS) value [21]. Partial fulfillment of some requirements is ensured to be possible if circumstances allow. When resources are limited, this method helps systematically choose which needs to implement.

3.4. Validation

Before development begins, documented requirements are checked to ensure they truly meet user needs and quality standards. This is the goal of the validation step [3]. Fuzzy logic provides an effective tool in this phase. It makes uncertain comparison data quantifiable. One known approach involves integrating fuzzy logic into case-based reasoning (CBR) systems. For example, the iSRS (Inspection of Software Requirements Specification) system was developed using this approach [3]. Fuzzy logic serves as a disambiguation tool for this system. When a new requirement is examined, it is compared to earlier cases that are stored in a database. The similarity is decided based on fuzzy value. Partial matches are also considered. The similarity measure might range from 30 to 90 percent. If complete agreement exists, the previous solution can be reused. If only partial agreement is found, a modified solution is created. The most similar cases form the basis of this modified solution. Accuracy is improved and ambiguous cases are clarified through fuzzy logic in this manner.

Quality control in early phases can also use fuzzy logic [1]. During development, for instance, error numbers or density (defect density) can be estimated. A testing approach called fuzzy testing [23] can be used to evaluate completeness. If the value is close to 1, the program is nearly complete and few changes are needed. For example, if the completeness value of a module is 0.88, roughly 12 percent redesign is needed.

3.5. Management

Throughout a project, an ongoing process called requirement management includes tracking changes, resources, and priorities [15]. The primary focus is on establishing systematic procedures for capturing changes, evaluating their impact on project scope and schedule, and making informed decisions about approval or rejection [24]. This management activity involves maintaining traceability links between requirements and system components to facilitate impact analysis. Requirements prioritization determines the implementation sequence based on multiple criteria such as business value, technical feasibility, resource availability, and stakeholder preferences, and this prioritization must be revisited when requirements change or new information emerges [25].

When requirements change during a project, fuzzy logic permits the systematic re-assessment of all requirement priorities without requiring crisp, definitive data, as fuzzy membership functions accommodate imprecise expert opinions and incomplete information [26]. The fuzzy analytical hierarchy process (FAHP) has proven particularly effective for this purpose, allowing practitioners to reduce uncertainty and vague opinions of requirements management experts when continuously adjusting requirement priorities in response to scope changes and emerging project constraints [26]. This approach supports iterative re-prioritization cycles characteristic of agile and evolutionary development methodologies, where requirements are evaluated multiple times throughout the project lifecycle to accommodate scope evolution and changing stakeholder needs while maintaining systematic, quantifiable decision-making even under conditions of incomplete knowledge [25].

4. Conclusion

This study offers a comprehensive overview of that application possibilities of fuzzy logic in software requirements engineering. Based on the results, uncertain and ambiguous information can be effectively handled by fuzzy logic at every stage of the process. Mathematical models can be built for unclear needs, and multi-factor decision-making is supported.

The reviewed research demonstrates the flexibility and effectiveness of fuzzy logic in requirements engineering. It offers a mathematical framework for managing uncertainties that arise in human communication and decision-making. The entire RE process—from elicitation and analysis to specification, validation, and management—shows improvement when fuzzy logic is applied. Vague, linguistically stated user expectations such as "fast," "stable," or "easy to use" can be transformed into measurable forms. This reduces contradictions, improves communication, and enhances requirement quality. Consequently, subjective properties like satisfaction, importance, or completeness become quantifiable. Fuzzy sets and linguistic variables assist in gathering uncertain opinions during elicitation, while methods such as Fuzzy TOPSIS, fuzzy graphs, and fuzzy decision models support analysis and prioritization. These techniques make decision-making more transparent by clearly showing influencing factors. FL also applies well to assessing software component completeness, evaluating security risks, and improving quality assurance.

Especially effective is the combination of fuzzy inference systems, fuzzy graphs, and fuzzy decision models. These tools provide strong support for ranking, specifying, and validating requirements. By

working with different truth levels, fuzzy logic surpasses binary (yes/no) approaches, producing more accurate and measurable requirements and improving user satisfaction. Quality assurance also benefits, as fuzzy-based validation and testing methods offer reliable numerical indicators for evaluating system completeness and reliability.

Despite these benefits, several limitations persist. Many fuzzy-based methods rely heavily on expert knowledge; membership functions and rules often depend on expert input, which can lead to inconsistency if experts disagree or lack experience. Computational demands also increase significantly with large datasets. As a result, industrial adoption of FL in field of RE is still limited. Standard guidelines are lacking, built-in software support is scarce, and automation requires further development.

Future research should focus on three key areas to advance fuzzy logic-based requirements engineering:

- Developing automatic membership function determination methods,
- Enhancing fuzzy tool functionality,
- Implementing fuzzy techniques in projects of various sizes and real industrial settings.

Fuzzy logic provides a solid, flexible framework for requirements engineering. With its help, analysts can respond effectively to uncertainty while maintaining accuracy. Ongoing research and practical applications are essential for successful industrial adoption. Although challenges remain, fuzzy logic presents a promising pathway toward more reliable and human-centered software development.

References

[1] T. Hassan, C. W. Mohammad, and M. Sadiq, "Using Social Network and Fuzzy Set Theory for Elicitation and Prioritization of Software Requirements," *International Journal of Computer Theory and Engineering*, vol. 14, no. 3, pp. 126–134, 2022, ISSN: 17938201. DOI: 10.7763/IJCTE.2022.V14.1319

[2] N. Iqbal and J. Sang, "Fuzzy Logic Testing Approach for Measuring Software Completeness," en, *Symmetry*, vol. 13, no. 4, p. 604, Apr. 2021, ISSN: 2073-8994. DOI: 10.3390/sym13040604

[3] D. E. Tamir, C. J. Mueller, and A. Kandel, "Complex Fuzzy Logic Reasoning-Based Methodologies for Quantitative Software Requirements Specifications," in *Computational Intelligence and Quantitative Software Engineering*, W. Pedrycz, G. Succi, and A. Sillitti, Eds., vol. 617, Series Title: Studies in Computational Intelligence, Cham: Springer International Publishing, 2016, pp. 153–172, ISBN: 978-3-319-25962-8 978-3-319-25964-2. DOI: 10.1007/978-3-319-25964-2_8

[4] G. D. Rottoli and C. Casanova, "Multi-criteria and Multi-expert Requirement Prioritization using Fuzzy Linguistic Labels," *ParadigmPlus*, vol. 3, no. 1, pp. 1–18, Feb. 2022, ISSN: 2711-4627. DOI: 10.55969/paradigmplus.v3n1a1

[5] D. C. Lima, F. Freitas, G. Campos, and J. Souza, "A Fuzzy Approach to Requirements Prioritization," in *Search Based Software Engineering*, M. B. Cohen and M. Ó Cinnéide, Eds., vol. 6956, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 64–69, ISBN: 978-3-642-23715-7 978-3-642-23716-4. DOI: 10.1007/978-3-642-23716-4_8

[6] D. Mougouei, D. M. Powers, and E. Mougouei, "A fuzzy framework for prioritization and partial selection of security requirements in software projects," en, *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 2, pp. 2671–2686, Sep. 2019, ISSN: 1064-1246, 1875-8967. DOI: 10.3233/JIFS-182907

[7] Y. Ahmad, W. Nasir, and S. Husain, "A Fuzzy base Approach to Reduce the Domain of Ambiguities in Software Requirement," in *2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, Yogyakarta, Indonesia: IEEE, Nov. 2018, pp. 674–678, ISBN: 978-1-5386-7422-2. DOI: 10.1109/ISRITI.2018.8864235

[8] M.-Y. Hsieh, Y.-C. Hsu, and C.-T. Lin, "Risk assessment in new software development projects at the front end: A fuzzy logic approach," en, *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, no. 2, pp. 295–305, Apr. 2018, ISSN: 1868-5137, 1868-5145. DOI: 10.1007/s12652-016-0372-5

[9] L. Zadeh, "Fuzzy sets," en, *Information and Control*, vol. 8, no. 3, pp. 338–353, Jun. 1965, ISSN: 00199958. DOI: 10.1016/S0019-9958(65)90241-X

[10] I. Skrjanc, S. Blazic, and O. Agamennoni, "Interval Fuzzy Modeling Applied to Wiener Models With Uncertainties," en, *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 5, pp. 1092–1095, Oct. 2005, Blazic, ISSN: 1083-4419. DOI: 10.1109/TSMCB.2005.850166

[11] E. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," en, *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, Jan. 1975, ISSN: 00207373. DOI: 10.1016/S0020-7373(75)80002-2

[12] A. Patel and B. Mohan, "Some numerical aspects of center of area defuzzification method," *Fuzzy Sets and Systems*, vol. 132, no. 3, pp. 401–409, 2002, ISSN: 0165-0114. DOI: https://doi.org/10.1016/S0165-0114(02)00107-0

[13] I. Udousoro, "Effective Requirement Engineering Process Model in Software Engineering," en, *Software Engineering*, vol. 8, no. 1, p. 1, 2020, ISSN: 2376-8029. DOI: 10.11648/j.se.20200801.11

[14] B. Nuseibeh and S. Easterbrook, "Requirements engineering: A roadmap," en, in *Proceedings of the Conference on The Future of Software Engineering*, Limerick Ireland: ACM, May 2000, pp. 35–46, ISBN: 978-1-58113-253-3. DOI: 10.1145/336512.336523

[15] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*, 2nd ed. Springer, 2025, ISBN: 978-3-662-69204-2.

[16] M. T. J. Ansari, F. A. Al-Zahrani, D. Pandey, and A. Agrawal, "A fuzzy TOPSIS based analysis toward selection of effective security requirements engineering approach for trustworthy healthcare software development," en, *BMC Medical Informatics and Decision Making*, vol. 20, no. 1, p. 236, Dec. 2020, ISSN: 1472-6947. DOI: 10.1186/s12911-020-01209-8

[17] V. R. Saxena, R. Yadav, A. Parveen, and M. Sadiq, "A Mathematical Model for the Selection of Software Requirements Elicitation Techniques," in *2024 14th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India: IEEE, Jan. 2024, pp. 130–135, ISBN: 979-8-3503-4483-7. DOI: 10.1109/Confluence60223.2024.10463258

[18] D. Mougouei, A. Ghose, H. Dam, M. Fahmideh, and D. Powers, "A Fuzzy-Based Requirement Selection Method for Considering Value Dependencies in Software Release Planning," in *2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Luxembourg, Luxembourg: IEEE, Jul. 2021, pp. 1–6, ISBN: 978-1-6654-4407-1. DOI: 10.1109/FUZZ45933.2021.9494422

[19] R. K. Yadav, Department of Computer Science, Mewar University, Gangrar, Chittorgarh – 312901, Rajasthan, India, S. Niranjan, and Department of Computer Science, Mewar University, Gangrar, Chittorgarh – 312901, Rajasthan, India, "Project Effort Estimation using COCOMO-2 Metrics with Fuzzy Logic," *Indian Journal of Science and Technology*, vol. 10, no. 29, pp. 1–8, Feb. 2017, ISSN: 0974-5645, 0974-6846. DOI: 10.17485/ijst/2017/v10i29/115882

[20] I.-D. Borlea, R.-E. Precup, F. Dragan, and A.-B. Borlea, "Centroid Update Approach to K-Means Clustering," en, *Advances in Electrical and Computer Engineering*, vol. 17, no. 4, pp. 3–10, 2017, Precup, ISSN: 1582-7445, 1844-7600. DOI: 10.4316/AECE.2017.04001

[21] M. Sadiq, V. Susheela Devi, J. Ahmad, and C. W. Mohammad, "Fuzzy logic driven security requirements engineering process," en, *Journal of Information and Optimization Sciences*, vol. 42, no. 7, pp. 1685–1707, Oct. 2021, ISSN: 0252-2667, 2169-0103. DOI: 10.1080/02522667.2021.1972618

[22] S. Dhingra, Savithri G, M. Madan, and Manjula R, "Selection of prioritization technique for software requirement using Fuzzy Logic and Decision Tree," in *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*, Coimbatore, India: IEEE, Nov. 2016, pp. 1–11, ISBN: 978-1-5090-4556-3. DOI: 10.1109/GET.2016.7916822

[23] M. Arif, C. W. Mohammad, and M. Sadiq, "Software requirements modeling from the selected set of requirements using fuzzy based approach," *ECTI Transactions on Computer and Information Technology (ECTI-CIT)*, vol. 16, no. 2, pp. 152–164, Jun. 2022, Section: Research Article. DOI: 10.37936/ecti-cit.2022162.247272

[24] S. Jayatilleke and R. Lai, "A systematic review of requirements change management," *Information and Software Technology*, vol. 93, pp. 163–185, 2018. DOI: 10.1016/j.infsof.2017.09.004

[25] A. Gupta and C. Gupta, "A novel collaborative requirement prioritization approach to handle priority vagueness and inter-relationships," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 5, pp. 2288–2297, 2022, ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2019.12.002

[26] M. A. Akbar, M. Shameem, A. A. Khan, M. Nadeem, A. Alsanad, and A. Gumaei, "A fuzzy analytical hierarchy process to prioritize the success factors of requirement change management in global software development," en, *J. Softw. (Malden)*, vol. 33, no. 2, Feb. 2021. DOI: https://doi.org/10.1002/smr.2292