A Simulation-Based Fuzzy Multiplet Attribute Decision Making for Prioritizing Software Requirements

Abdel Ejnioui¹, Carlos E. Otero¹ and Luis D. Otero²
¹Information Technology, University of South Florida, Lakeland, Florida, USA
acjnioui@poly.usf.edu, cotero@poly.usf.edu
²Engineering Systems, Florida Institute of Technology, Melbourne, Florida, USA
lotero@fit.edu

Abstract—It is well known that most of the approaches proposed in recent research to prioritize software requirements have not been widely adopted. These approaches are too complex and time consuming, or inconsistent and difficult to implement. This paper proposes a new approach to prioritize requirements that is practical and easily implementable. Whereas most proposed approaches quantify requirements in precise and crisp parameters, this paper takes into consideration the imprecise nature of requirements by modeling their attributes as fuzzy variables. As such, these variables are integrated into a fuzzy multi-attribute decision making problem in which the requirements represented as attributes are ranked via the expected value operator of a fuzzy variable. The expected values of the attributes in the problem are computed by numerical simulation. This approach is easily extendable to include other attributes and can be easily customized as a decision making tool for software project managers.

Keywords: Requirements Engineering, Requirements Prioritization, Fuzzy Variables, Expected Value, Fuzzy Simulation, Software Quality

I. INTRODUCTION

As information technology penetrates every aspect of daily life, software engineering is becoming a vital discipline for creating software that meets customer expectations. This discipline faces numerous challenges considering the degree to which software products have evolved. In most software projects, there is a diversity of stakeholders who have vested interest in seeing their requirements included in the final deliverables of the product. These stakeholders can be users, customers, project managers, product managers, developers and testers. Because of their positions in the software engineering process, these stakeholders formulate requirements that address their immediate sphere of concern. As a result, these requirements can be numerous, diverse in nature, may consume various amounts of resources, bring different benefits and impose different costs [1, 2]. To make matters worse, many of these requirements impose conflicting demands on software projects. Considering these requirements with their budgets in resources and times, it is not unreasonable to expect that these requirements will have different priorities. In practice, stakeholders have to find approaches by which these requirements are prioritized. Only requirements with high priorities can be implemented in earlier releases while requirements with lower priorities can be left out to late releases [3].

For some time, many attempts have been made to find effective ways to prioritize software requirements [3-5]. Some methods proposed for requirement prioritization are qualitative in nature [6, 7]. These methods rely mostly on the direct qualitative assessment of stakeholders or indirect assessment from experts who collected data from stakeholders. Although these methods can prioritize requirements in efficient ways, they fail to reveal the differences in priorities between the requirements [3]. Others have proposed quantitative methods for prioritizing requirements that can be quite effective [5, 8-11]. While some of these quantitative methods can display a high degree of consistency, they tend to be complex and impractical [12]. On the other hand, there are several quantitative methods that are informal and easy to adopt, but lack structure and consistency [13-15].

Although quantitative methods attempt to overcome the subjective nature of qualitative methods, they nevertheless suffer from the need to use precisely measurable parameters in order to prioritize software requirements. In reality, requirements are rarely quantified in precise values. This is particularly true for quality requirements, which are often vague and intangible [16, 17]. In this context, it would be reasonable to approach the challenge of requirement prioritization using fuzzy quantities instead of using crisp values. To this end, the theory of fuzzy sets can provide a number of mathematical means by which requirements can be expressed in fuzzy ways and still allow the derivation of proven methods for prioritizing requirements.

This paper proposes a novel approach for prioritizing software requirements based on the formulation of a fuzzy multiple attribute decision-making problem. Rooted in credibility theory, this approach relies on the expected value method for computing the priorities of each requirement. The remainder of this paper is as follows. Section II provides a summary of previous approaches for prioritizing requirements. Section III provides several concepts in credibility theory. Section IV explains how the expected value operator can be used to rank fuzzy variables. Section V describes the formulation of the problem. Section VI provides the proposed solution to address this problem formulation. Section VII concludes
the paper by providing a conclusion and future directions of this work.

II. RELATED WORK

As software has become more complex, and project managers are forced to make concessions and trade-offs to complete projects on schedule, requirements prioritization has become an increasingly important part of ensuring the success of a project. There are many compelling arguments as to why requirements prioritization is necessary. One of the most compelling is made by Kriegers. He argues that limited resources inevitably mean that some requirements cannot be implemented, and that the decisions about which requirements are the most important are better made in early development stages rather than in "emergency mode" towards the end of a project [5].

Most requirements prioritization methods involve examining requirements through the framework of benefit and cost [2, 12, 13]. In other words, requirements are analyzed on the basis of how much benefit that fulfilling the requirement will provide to the customer, as well as any costs associated with its implementation. This information is then used in some manner to rank the requirements in terms of their importance.

There are a number of methods that currently exist for approaching requirements prioritization. Many of these methods are quantitative, and employ a very systematic approach to gathering data and assigning values to various factors associated with requirements in order to compute a priority [2]. Other methods rely on making somewhat informal generalizations and groupings before trying to assign priorities. This is typically done to reduce the amount of time necessary to compute priorities, but may sacrifice some consistency [18].

One of the most consistent methods that have been developed is the Analytic Hierarchy Process (AHP) [19-21]. All possible pairs of requirements are enumerated, and then the perceived importance of each requirement is ranked in relationship to its pair. The most important requirement from each pair is assigned a value, while the requirement of lesser importance is given the reciprocal of that value. The redundacy of AHP does produce consistent requirements; however it also makes the process impractical for all but small projects [3, 12].

Several other methods employ a variation of the pairwise comparisons performed for AHP. Hierarchy AHP is the most closely related; the process is nearly identical to AHP, except that requirements are first subjectively prioritized as low, medium, or high. Pair-wise comparisons are then performed on the requirements of each group [4]. Other algorithms, such as a binary search tree, and bubble-sort have also been used to compare requirements in pairs. With the exception of bubble sort, these methods require fewer comparisons than AHP. However they are still not feasible for larger projects, nor do they provide the same level of consistency [4, 9].

Total Quality Management (TQM) and Quality Function Deployment (QFD) are two other quantitative methods used to prioritize requirements. TQM ranks requirements against a set of criteria that have been deemed necessary for the success of a project [8]. A priority rank is then determined based on the weight of the success criteria and the requirement. QFD correlates the value a proposed product feature has to a customer with specific requirements in order to determine priority. These methods are regarded as robust, however it is well known that the time and commitment needed to execute them has prevented their wide-scale adoption by organizations [8].

Despite the myriad of methods that have been proposed, research suggests that none have gained universal acclaim, nor have they been widely adopted [3]. While some methods like AHP, QFD, and TQM seem to produce more consistent prioritizations [8, 12], they are also complex, time-consuming, and difficult to implement [8]. Other less formal methods may save time initially, but could cause problems in the later stages of a project if appropriate factors are not accounted for. Lehtola’s study on the practical challenges of requirement prioritization methods suggests that project managers do not have access to a method that is both simple and effective [22]. In order to increase the effectiveness of requirements prioritization, new methods need to be developed that save time, yet preserve the accuracy that more robust methods currently offer.

One of the latest works to propose a solution to the requirement prioritization problem is presented in [23]. In this work, the authors present a binary valued approach for prioritizing requirements that fuses benefits and costs of each requirement to provide a holistic approach to prioritization. Specifically, once requirements are elicited, a set of quality attributes are identified as evaluation criteria. These attributes are defined in terms of many different features, where each feature is determined to be present or not. Once all features are identified, each requirement is evaluated against each feature using a simple binary scale (i.e., 0 or 1). Requirements that satisfy the highest number of features would expose a higher level of quality (or priority) for that particular quality attribute. Once all requirements are evaluated and measurements computed for all features, the approach used desirability functions to fuse all measurements into one unified value that is representative of the overall quality of the requirement. This unified value is computed by using a set of desirability functions that take into consideration the priority of each quality attribute. Therefore, the resulting priority of each requirement is derived from decision-makers’ goals for a specific software project. While this approach is practical and quite easy to implement, it is too simplistic since it accounts only for the presence (value 1) or absence (value 0) of features in quality attributes. In reality, information about features related to quality attributes is imprecise and in most cases incomplete.

III. PRELIMINARIES

This section introduces basic concepts in credibility theory that are critical in developing the approach...
proposed in this paper. Most of these concepts have been published in [21-23].

A. Credibility Space

Let $\Theta$ be a nonempty set and $P(\Theta)$ its power set. Each element in $P(\Theta)$ is called an event. The credibility of an event $A$, denoted by $Cr\{A\}$, is a number that represents the credibility that $A$ will occur. To ensure that $Cr\{A\}$ has certain mathematical properties, the following axioms must be admitted:

**Axiom 1.** (Normality) $Cr\{\emptyset\} = 1$.

**Axiom 2.** (Monotonicity) $Cr\{A\} \leq Cr\{B\}$ whenever $A \subset B$.

**Axiom 3.** (Self-Duality) $Cr\{A\} + Cr\{A^c\} = 1$ for any event $A \in P(\Theta)$ where $A^c$ is the set complement of $A$.

**Axiom 4.** (Maximality) $Cr\{\bigcup A_i\} = \sup_i, Cr\{A_i\}$ for any events $\{A_i\}$ with $\sup_i, Cr\{A_i\} < 0.5$.

**Definition 1** The set function $Cr$ is called a credibility measure if it satisfies the normality, monotonicity, self-duality and maximality axioms.

**Definition 2** Let $\Theta$ be a nonempty set, $P(\Theta)$ the power set of $\Theta$, and $Cr$ a credibility measure. Then the triplet $(\Theta, P(\Theta), Cr)$ is called a credibility space.

B. Fuzzy Variables and Credibility Distributions

**Definition 3** A fuzzy variable is defined as a (measurable) function from a credibility space $(\Theta, P(\Theta), Cr)$ to the set of real numbers.

**Theorem 1** (Credibility Inversion) Let $\xi$ be a fuzzy variable with membership function $\mu$. Then for any set $B$ of real numbers, we have

$$Cr\{\xi \in B\} = \frac{1}{2} \left( \sup_{x \in B} \mu(x) + 1 - \sup_{x \in B^c} \mu(x) \right)$$

where $B^c$ is the complement set of $B$. The proof of this theorem can be found in [24, 25].

**Definition 4** The credibility distribution $\Phi: \mathbb{R} \rightarrow [0,1]$ of a fuzzy variable $\xi$ is defined by

$$\Phi(x) = Cr\{\theta \in \Theta \mid \xi(\theta) \leq x\} = \frac{1}{2} \left( \sup_{y \leq x} \mu(y) + 1 - \sup_{y > x} \mu(y) \right),$$

for all $x \in \mathbb{R}$.

C. Expected Value

In this section, we introduce the expected value operator of a fuzzy variable. Although there are many ways to define an expected value operator, we choose to focus on the most general definition of this operator [24-26]. This definition is applicable to both continuous and discrete fuzzy variables [24].

**Definition 5** Let $\xi$ be a fuzzy variable. Then the expected value of $\xi$ is defined as

$$E[\xi] = \int_{-\infty}^{+\infty} Cr\{\xi \geq r\}dr - \int_{-\infty}^{0} Cr\{\xi \leq r\}dr$$

provided that at least one of the two integrals is finite.

**Remark 1** Let $\xi$ and $\eta$ be independent fuzzy variables with finite expected values. Then for any numbers $a$ and $b$, we have

$$E[a\xi + b\eta] = aE[\xi] + bE[\eta]$$

This property is called the linearity of expected value operator of fuzzy variables.

D. Ranking of Fuzzy Variables

Contrary to the set of real numbers, fuzzy variables do not have a natural order in a fuzzy world. As such, several approaches were devised to rank fuzzy variables [27, 28]. One approach is based on the expected value operator of a fuzzy variable [26].

**Definition 6** (Expected Value Criterion) Let $\xi$ and $\eta$ be fuzzy variables with finite expected values. We say $\xi > \eta$ if and only if $E[\xi] > E[\eta]$ where $E$ is the expected value operator of a fuzzy variable.

IV. EXPECTED VALUE OF A FUZZY VECTOR

This section introduces fuzzy simulation as a possible technique to compute the expected value of a fuzzy vector based on its credibility.

**Definition 7** ($\alpha$-level set) Let $\xi$ be a fuzzy variable such that $\xi: \Theta \rightarrow [0,1]$, $\mu_\xi$ its membership function and $\Theta$ as defined above. An $\alpha$-level set of $\xi$ where $\alpha \in [0,1]$ is defined as

$$a\xi = \{ \theta \in \Theta \mid \mu_\xi(\theta) \geq \alpha \}.$$

**Definition 8** Let $f: \mathbb{R}^n \rightarrow \mathbb{R}$ be a function, and $\xi_1, \xi_2, ..., \xi_n$ fuzzy variables on the credibility space $(\Theta, P(\Theta), Cr)$. Then $\xi = f(\xi_1, \xi_2, ..., \xi_n)$ is a fuzzy variable defined as

$$\xi(\theta) = f(\xi_1(\theta_1), \xi_2(\theta_2), ..., \xi_n(\theta_n))$$

for any $\theta \in \Theta$.

**Definition 9** An $n$-dimensional fuzzy vector is defined as a function from a credibility space $(\Theta, P(\Theta), Cr)$ to the set of $n$-dimensional real vectors.

**Theorem 2** The vector $(\xi_1, \xi_2, ..., \xi_n)$ is a fuzzy vector if and only if $\xi_1, \xi_2, ..., \xi_n$ are fuzzy variables.

The proof of this theorem can be found in [24, 25]. Assume that $x$ is a decision vector, $\xi$ is a fuzzy vector and
\( f(x, \xi) \) is a return function. We are interested in computing the uncertain function \( U: x \rightarrow E[f(x, \xi)] \). As the expected value of a fuzzy vector, this function can be used to solve fuzzy multiple attribute decision problems using ranking criteria based on the expected values of fuzzy variables [25].

1) **Computing The Expected Value of a Fuzzy Vector**

Suppose \( \xi = (\xi_1, \xi_2, \ldots, \xi_n) \) is a fuzzy vector with membership function \( \mu \) while \( f \) is a function. We can generate a fuzzy simulation to compute the expected value \( E[f(\xi)] \). To do so, we randomly generate \( u_k \) from the \( \alpha \)-level sets of \( \xi \) for \( k = 1, 2, \ldots, N \) where \( \alpha \) is a small number while \( N \) is a large number. Then for any numbers \( r \geq 0 \), the credibility values \( \text{Cr}(f(\xi) \geq r) \) and \( \text{Cr}(f(\xi) \leq r) \) may be estimated by using the samples obtained from random generation of \( N \) values of \( r \). Then, we can use simulation to calculate the expected values as [25]:

\[
E[f(\xi)] = \int_0^{+\infty} \text{Cr}(f(\xi) \geq r)dr - \int_{-\infty}^0 \text{Cr}(f(\xi) \leq r)dr
\]

2) **Fuzzy Simulation of the Expected Value**

The estimation above can be used to design the simulation algorithm described below [29]. Let \( \xi = (\xi_1, \xi_2, \ldots, \xi_n) \) be a fuzzy vector and \( f: \mathbb{R}^n \rightarrow \mathbb{R} \) be a function. The expected value \( E[f(\xi)] = E[f(\xi_1, \xi_2, \ldots, \xi_n)] \) can be estimated using the following algorithm:

Set \( E = 0 \).

Generate randomly each \( u_i \) from the \( \alpha \)-level sets of \( \xi \) for \( i = 1, 2, \ldots, m \) and \( j = 1, 2, \ldots, n \). This generates the matrix \( U = [u_{ij}]_{m \times n} \). We denote by \( u_i = (u_{i1}, u_{i2}, \ldots, u_{in}) \) the \( i \)-th row vector in \( U \).

For each row in \( U \)

- compute \( f(u_i) \) for \( i = 1, 2, \ldots, m \).

EndFor

Set \( a = \min f(u_i) \) for \( i = 1, 2, \ldots, m \).

Set \( b = \max f(u_i) \) for \( i = 1, 2, \ldots, m \).

For 1 to \( N \)

- Generate randomly \( r \) from \([a, b]\).

  \begin{align*}
  \text{If } r &\geq 0 \\
  \text{Set } E &= E + \text{Cr}(f(\xi) \geq r).
  \end{align*}

  \begin{align*}
  \text{ElseIf } r &< 0 \text{ then} \\
  \text{Set } E &= E - \text{Cr}(f(\xi) \geq r).
  \end{align*}

EndFor

Set \( E[f(\xi_1, \xi_2, \ldots, \xi_n)] = \max (a, 0) + \min (b, 0) + E \cdot (b - a)/N \).

Note that \( f(u_i) \) can be computed using the membership functions of the fuzzy variables \( \xi_1, \xi_2, \ldots, \xi_n \) as follows:

\[
f(u_i) = \min_{1 \leq j \leq n} \left\{ \mu_j(u_{ij}) \mid u_i = (u_{i1}, u_{i2}, \ldots, u_{in}) \right\}
\]

V. **PROBLEM FORMULATION**

Let \( R = \{r_1, r_2, \ldots, r_m\} \) and \( Q = \{q_1, q_2, \ldots, q_n\} \) be the set of requirements and quality attributes respectively. Associated with each quality attribute \( q_j, j = 1, 2, \ldots, n \), a set of features \( F_j = \{f_{j1}, f_{j2}, \ldots, f_{jk_j}\} \) where each feature \( f_k \), \( k = 1, 2, \ldots, k_j \), is characterized by a fuzzy number \( \nu_k \).

Also associated with each attribute is a fuzzy weight \( w_j \) where the set \( W = \{w_1, w_2, \ldots, w_n\} \) is a set of weights. Our objective is to combine the fuzzy numbers of all features within an attribute resulting in a single fuzzy number. The results will be a \( m \times n \) matrix of fuzzy numbers.

A. **Combination of Feature Fuzzy Numbers**

The problem of combining a set of fuzzy numbers into a single one arises when expert opinions or imprecise estimates of a given quantity can be combined in either additive or nonadditive manner by means of fuzzy number representation [30]. While several techniques have been proposed to address this problem, we choose to use the simplest one called crisp weighting.

1) **Crisp Weighting**

This approach consists of assigning weights \( w_k \) to the individual fuzzy numbers. In this case, a combination rule can be applied on the fuzzy numbers of the feature set \( F_j \) as follows:

\[
C_j = \sum_{k=1}^{k_j} w_k \nu_k
\]

where \( \sum_{k=1}^{k_j} w_k = 1 \) and \( C_j \) is the combined fuzzy number.

Note that the weights used in this approach are distinct from the weights in \( W \) defined above. Although this is not the ideal approach, it does at least preserve agreement among the values in the combined fuzzy numbers [30].

2) **Problem Matrix**

After crisp weighting of the fuzzy numbers of all features in every attribute, we obtain \( m \times n \) matrix of fuzzy numbers \( A = [C_{ij}]_{m \times n} \) where the \( m \) rows and \( n \) columns represent requirements and attributes respectively. In addition to this matrix, we have the vector \( W = \{w_1, w_2, \ldots, w_n\} \) based on the set \( W \) defined above. This formulation is known as the fuzzy multiple attribute decision-making problem (FADM) [31].

VI. **PROPOSED SOLUTION**

This section describes the solution proposed to address the FADM problem.

A. **Matrix Normalization**

In most FADM problems expressed in matrix form, normalization is necessary in order to transform the matrix and weight vector numbers to comparable values. In our case, normalization is based on the expected value operator [29]. For each fuzzy number \( C_{ij} \) in \( A \), transform this number as follows:

\[
\eta_{ij} = \frac{C_{ij}}{\sqrt{\sum_{i=1}^{m} (E[C_{ij}]^2)}},
\]

for \( i = 1, 2, \ldots, n \) and \( j = 1, 2, \ldots, m \).

If \( C_{ij} \) is a cost:
\[ \eta_{ij} = \frac{p_j - C_{ij}}{\sqrt{\sum_{i=1}^{m} E[C_{ij}]}} \]  
for \( i = 1, 2, \ldots, n, \ j = 1, 2, \ldots, m \), and 
\[ p_j = \max_{1 \leq i \leq m} \sup \{x_{ij} \mid \mu_i(x_{ij}) > 0\} \]. Note that both expressions use in their denominators the expected values of the fuzzy variables of each column or attribute. Also, note that the \( \mu_i \) are the membership functions of the fuzzy variables representing the attributes. The obtained normalized matrix is \( B = [\eta_{ij}]_{m \times n} \).

**B. Weight Normalization**

Similar to the problem matrix, the weight vector must be normalized as follows:

\[ \omega_j = \frac{w_j}{\sum_{i=1}^{n} E[w_j]} \]  
for \( j = 1, 2, \ldots, n \). The final normalized weight vector is \( \omega = [\omega_1, \omega_2, \ldots, \omega_n] \).

**C. Expected Value Method**

Given a normalized matrix of fuzzy numbers and a normalized weight vector, a simple additive weighting approach can be used to compute the following \( m \) fuzzy variables as follows [29]:

\[ f_i = \sum_{j=1}^{n} \omega_j \eta_{ij} \]  
for \( i = 1, 2, \ldots, m \). Each fuzzy variable can be viewed as the real-value function associated with each requirement. A utility value function \( E[f_i], i = 1, 2, \ldots, m \), based on the expected value operator can be devised to rank the \( m \) fuzzy variables. This utility \( E[f_i] \) can be computed using the simulation algorithm presented in section IV.2 or computed directly if the real-value function \( f_i \) is an equipossibly, triangular or trapezoidal fuzzy variable. For instance, if the fuzzy variables of the attributes are triangular variables, the real-valued function \( f \) can be computed as follows if we assume \( \eta_{ij} = (a_{ij}, b_{ij}, c_{ij}) \) and \( \omega_j = (d_{j1}, e_{j1}, g_{j1}) \):

\[ f_i = \left( \sum_{j=1}^{n} a_{ij} d_{j1} + \sum_{j=1}^{n} b_{ij} e_{j1} + \sum_{j=1}^{n} c_{ij} g_{j1} \right) \]  
for \( i = 1, 2, \ldots, m \). In this case, the utility function of \( f_i \) can be computed as:

\[ E[f_i] = \left( \frac{1}{4} \sum_{j=1}^{n} a_{ij} d_{j1} + 2 \sum_{j=1}^{n} b_{ij} e_{j1} + \sum_{j=1}^{n} c_{ij} g_{j1} \right) \]  
for \( i = 1, 2, \ldots, m \).

**D. Summary of the Proposed Solution**

We assume that we are given a list of requirements, attributes and their features. The proposed solution to prioritize these requirements can be summarized as follows:

1. Get the fuzzy number of each feature.
2. For each attribute
   - Apply the crisp weighting technique to combine the fuzzy numbers of its features.
   - Derive the membership function from the combined fuzzy number.
3. Using the derived membership functions of the attribute, perform the fuzzy simulation algorithm.
4. Normalize the problem matrix by using the expected value of the fuzzy variables generated by the simulation algorithm.
5. Apply the expected value method on the normalized matrix.
6. Sort the utility functions in non-decreasing order.

The requirements that have the highest values from their corresponding utility functions have the highest priority.

**VII. CONCLUSION**

This paper presents a novel approach to prioritize requirements based on ranking fuzzy numbers expressing the importance of an attribute in a requirement using its expected value. This ranking requires the approximation of these expected values via simulation by randomly picking values in the credibility distributions of these numbers. This approach has the capacity of fusing multiple criteria and features to provide a holistic view of quality requirements. In addition, this approach can be easily extended to include other types of requirements not considered in this paper.

This approach is currently being implemented in a custom decision-making software tool that is flexible and easy to use. This tool will be used in a case study of a software project in order to study the quality of the results it produces when different scenarios of prioritization are considered.

**VIII. REFERENCES**


