Scientific Research Publishing

Journal of Software Engineering and Applications, 2021, 14, 267-276
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

How to Support Communication among Stakeholders during Software Requirements Prioritization

Philip Achimugu[1], Oluwatolani Achimugu[2], Mohammed Ahmed Taiye[1], Sseggujja Husssein[3], Grace Tam-Nurseman[4], Saheed Adekeye[4]

[1]Department of Computer Science, Air Force Institute of Technology Kaduna, Kaduna, Nigeria
[2]Department of Information and Communication Engineering, Air Force Institute of Technology Kaduna, Kaduna, Nigeria
[3]Department of Computer Science, Islamic University in Uganda, Kampala, Uganda
[4]Department of Computer Science, Lead City University, Ibadan, Nigeria
Email: p.achimugu@afit.edu.ng

How to cite this paper: Achimugu, P., Achimugu, O., Taiye, M.A., Husssein, S., Tam-Nurseman, G. and Adekeye, S. (2021) How to Support Communication among Stakeholders during Software Requirements Prioritization. *Journal of Software Engineering and Applications*, **14**, 267-276.
https://doi.org/10.4236/jsea.2021.147016

Received: March 8, 2021
Accepted: July 5, 2021
Published: July 8, 2021

Abstract

Existing prioritization techniques do not support communication among stakeholders and this makes it difficult for stakeholders to understand the meaning and essence of requirements before prioritization commences. When this happens, the ordered list of requirements can be misleading. The aim of this research is to develop a method capable of supporting and computing ranks of requirements based on the criteria defined for each requirement. The proposed method is developed based on fuzzy logic. Results show that ordered requirements reproduced ranks with strong correlations when compared to their linguistic values provided by the stakeholders. The contribution of this paper centers on an improved way of prioritizing requirements with understanding.

Keywords

Requirements, Prioritization, Software, Fuzzy Logic, Stakeholders

1. Introduction

During requirements elicitation, there are more prospective requirements specified for implementation by relevant stakeholders with limited time and resources. Therefore, an ordered list of requirements must be considered for implementation. This process is referred to as requirements prioritization. It is considered to be a complex multi-criteria decision making process [1].

There are so many advantages of prioritizing requirements before architecture

design or coding. Prioritization aids the implementation of a software system with preferential requirements of stakeholders [2] [3]. Also, the challenges associated with software development such as limited resources, inadequate budget, insufficiently skilled programmers among others make requirements prioritization really important [4]. It can help in planning software releases since not all the elicited requirements can be implemented in a single release due to some of these challenges [5] [6]. It also enhances budget control and scheduling [1]. Therefore, determining which, among a pool of requirements to be implemented first and the order of implementation is necessary to avoid breach of contract or agreement during software development. Furthermore, software products that are developed based on prioritized requirements can be expected to have a lower probability of being rejected. To prioritize requirements, stakeholders will have to compare them in order to determine their relative importance through a weight scale which is eventually used to compute the prioritized requirements [7]. These comparisons become complex with an increase in the number of requirements [8].

Software system's acceptability level is mostly determined by how well the developed system has met or satisfied user's requirements. Hence, eliciting and prioritizing the appropriate requirements and scheduling right releases with the correct functionalities are critical success factors for building acceptable systems. In other words, when vague requirements are implemented, the resulting system will fall short of user's expectations. Many software development projects have enormous prospective requirements that may be practically impossible to deliver within the expected time frame and budget [1] [9]. It, therefore, becomes highly necessary to source appropriate measures for planning and rating requirements in an efficient way.

2. Related Works

Many requirements prioritization techniques exist in the literature. All of these techniques utilize a ranking process to prioritize candidate requirements. The ranking process is usually executed by assigning relative weights across requirements based on pre-defined criteria. From the literature; analytic hierarchy process (AHP) is the most prominently used technique. However, this technique suffers bad scalability. This is due to the fact that, AHP executes ranking by considering the criteria that are defined through an assessment of the relative priorities between pairs of requirements. This becomes impracticable as the number of requirements increases. It also does not support requirements evolution or rank reversals but provide efficient or reliable results [10] [11]. Also, most techniques suffer from rank reversals. This term refers to the inability of a technique to update rank status of ordered requirements whenever a requirement is added or deleted from the list. Prominent techniques that suffer from this limitation are case base ranking [1]; interactive genetic algorithm prioritization technique [9]; Binary search tree [10]; cost value approach [6] and evolve [12]. Furthermore, existing techniques are prone to computational errors [13] probably due to lack

of robust algorithms. Karlsson *et al.* [10] conducted some researches where certain prioritization techniques were empirically evaluated. From their output, they reported that, most of the prioritization techniques apart from AHP and bubble sorts produce unreliable or misleading results while AHP and bubble sorts were also time consuming. The authors concluded that; techniques like hierarchy AHP, spanning tree, binary search tree, priority groups produce unreliable results and are difficult to implement. Babar *et al.* [11] were also of the opinion that, techniques like requirement triage, value intelligent prioritization and fuzzy logic based techniques are also error prone due to their reliance on experts and are time consuming too. Other requirements prioritization techniques such as planning game, wieger's method and requirements triage also possesses the ability to accurately prioritize requirements but cannot update ranks whenever requirements evolve [10] [11]. Most prioritization techniques do not support communication among stakeholders. One of the most recent works in requirements prioritization research reported communication among stakeholders as part of the limitations of their technique [1]. This can lead to generation of vague results. Communication has to do with the ability of all relevant stakeholders to fully understand the meaning and essence of each requirement before prioritization commences.

A study has been conducted to determine the requirements prioritization techniques used in software industry and identify aspects or evaluation criteria to choose the best technique according to the environment [14]. According to this study, techniques like cost value ranking, value oriented prioritization, cumulative voting, and MoSCoW were the most desirable and eminent techniques used to determine requirements ranks. The key aspect or evaluation criteria for the choice of these techniques were customer preference, business value, reliability of results, ease of use and time consumption rate, consistency, cost, benefit, penalty, technical risk and judgments on participants' experiences. Two systematic literature reviews revealed that, communication among stakeholders, scalability, complexity, uncertainty, time consumption, starvation and dependency issues among requirements, limited researches on the non-functional requirements, lower automation approach and conflict between stakeholders are critical problems associated with existing requirements prioritization techniques [15] [16].

3. Proposed Method

The value of requirements is computed based on their weight frequencies and a mean score is obtained to determine the final rank. The process flow of the proposed method is depicted in **Figure 1**.

To start the process, the following procedures are observed to elicit requirements used to construct pair-wise comparison for each stakeholder:

1) Generating requirements: The elicitor or architect articulates the description of the project's problem to the stakeholders both in written and verbal form. They now lead the stakeholders to express their thoughts in brief phrases or statements. Each person quietly documents requirements.

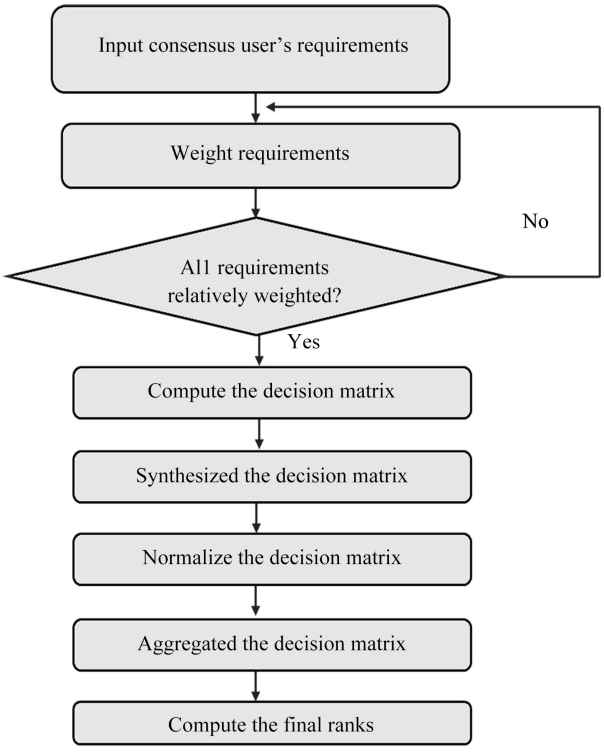

Figure 1. Process flow.

2) Recording requirements: Stakeholders engage in a round-robin feedback meeting to precisely elicit requirements (without deliberations at this point). The architect or elicitor then collates these requirements from all the stakeholders.

3) Discussing requirements: The documented requirements are then deliberated upon to determine clarity and relevance. For each requirement, the architect or developer asks for comments, questions or criticisms. This will allow stakeholders to express themselves in order to determine precise requirements.

4) Rating requirements: These requirements are parameterized as R_{ij}, where $i = 1, 2, 3, \cdots, n$ (n = total number of requirements) and rating confidence x_{ij} where $j = 1, 2, 3, \cdots, m$ (m = total number of stakeholders).

Based on these definitions and operations, the proposed steps of prioritizing software requirements are enumerated below:

Step 1: Requirements are weighted with Table 1 to determine their weights from the stakeholders and a decision matrix is constructed using Equation (3.1).

$$\tilde{A}^k = \left[\tilde{a}_{ij}\right]^k \tag{3.1}$$

$\tilde{A}^k$, represent the computed decision matrix of the form: $a_{ij} = La_{ij}, Ma_{ij}, Ua_{ij}$; k stands for the numbers of relevant stakeholders while $\tilde{a}_{ij}$ is the fuzzified local weights of all the criteria allotted by relevant stakeholders.

Step 2: It is important to note that, the weights allotted by the stakeholders are subjective and based on their knowledge or understanding of the requirements in the set. It is therefore necessary to employ the average score technique to synthesize the fuzzy performance values of k stakeholders using Equation (3.2).

Table 1. Weight scale.

Variables	Rank	TFNs	Crisp Value
VERY HIGH (VH)	5	(0.9, 1.0, 1.0)	[0.97]
HIGH (H)	4	(0.7, 0.9, 1.0)	[0.87]
MEDIUM (M)	3	(0.5, 0.7, 0.9)	[0.70]
LOW (L)	2	(0.3, 0.5, 0.7)	[0.50]
SUPER LOW (SL)	1	(0.1, 0.3, 0.5)	[0.30]

$$\omega_j = \frac{1}{k}\left[\sum_{i=1}^{k} a_{ij}\right],\ i = 1,\cdots,n;\ j = 1,\cdots,m \tag{3.2}$$

where $\omega_j = L\omega_j, M\omega_j, U\omega_j$ indicates the synthesized relative fuzzy weights of the jth requirement.

Step 3: The fuzzy numbers are defuzzified with Equation (3.3).

$$\Xi_{w_j} = \frac{L\omega_j M\omega_j U\omega_j}{\omega_j} \tag{3.3}$$

where, w_j is the synthesized weight of the jth, requirement.

Step 4: The defuzzification of the weights obtained from Step 3 are normalized using Equation (3.4).

$$\varpi_j = \frac{W_j}{\sum_{j=1}^{n} \varpi_j = 1},\ i = 1,\cdots,n;\ j = 1,\cdots,m \tag{3.4}$$

Step 5: After normalization, Equation (3.5) is used to determine the global ranks of requirements, which is the aggregation of weights across all the project stakeholders.

$$Gw_{ij} = \frac{\varpi_j\left(L\omega_j M\omega_j U\right)}{3} \tag{3.5}$$

Step 6: The final scores which determines the final ranks of requirements are computed using Equation (3.6). It gives rise to the ordered list of requirements.

$$Fw_{ij} = \sqrt{\frac{\sum a_{ij}}{k}} \tag{3.6}$$

where, a_{ij} is the sum of global weights while k stands for the number of stakeholders.

4. Empirical Evaluation

To illustrate the concepts of the proposed method, a fictional datasets consisting of 9 stakeholder's ratings across 10 requirements is used as shown in Table 2. The following explanations clarifies the computational processes involved in obtaining relative weights of requirements:

1) The stakeholders are asked to weight requirements using the linguistic values depicted in Table 2;

2) Thereafter, the linguistic values are converted to its corresponding triangu-

lar fuzzy numbers as shown in Table 3;

3) Since the relative weights of stakeholders are subjective, the triangular fuzzy weights are synthesized. This gives rise to the results displayed in Table 4;

4) The triangular fuzzy numbers are defuzzified into a crisp value for accurate calculation. The defuzzified weights are shown in Table 5;

5) The defuzzified weights are then normalized to ensure that, summation of the decision matrix is equal to 1 (Table 6);

6) The global weights are calculated as shown in Table 7 while the final scores are depicted in Table 8.

To reduce subjective biasness and deal with qualitative influential criteria for rating requirements in subjective environments, fuzzy sets theory and linguistic values quantified with triangular fuzzy numbers were used to determine the relative weights of requirements. These results will provide stakeholders and software developers with useful insight for decision making regarding which requirements are meant to be implemented in first, second, third or subsequent releases. The empirical results also help software developers in implementing or producing software with preferential requirements of stakeholders within time and budget.

Table 2. Linguistic variables for weights of requirements.

	S^1	S^2	S^3	S^4	S^5	S^6	S^7	S^8	S^9
R_1	VH	H	VH	F	F	H	F	L	H
R_2	H	F	F	H	VH	L	H	H	F
R_3	H	VH	H	VH	EH	F	H	H	F
R_4	F	H	VH	EH	H	H	F	L	F
R_5	VH	VH	EH	EH	VH	VH	H	F	F
R_6	VH	H	H	F	F	L	F	L	H
R_7	H	VH	VH	F	H	H	F	L	H
R_8	VH	H	H	VH	H	H	H	L	H
R_9	VH	VH	H	EH	VH	VH	H	H	VH
R_{10}	EH	EH	VH	H	VH	EH	F	VH	H

Table 3. Corresponding TFNs for relative weights of requirements.

	S^1	S^2	S^3	S^4	S^5	S^6	S^7	S^8	S^9
R_1	(0.7, 0.9, 1.0)	(0.5, 0.7, 0.9)	(0.7, 0.9, 1.0)	(0.3, 0.5, 0.7)	(0.3, 0.5, 0.7)	(0.5, 0.7, 0.9)	(0.3, 0.5, 0.7)	(0.1, 0.3, 0.5)	(0.5, 0.7, 0.9)
R_2	(0.5, 0.7, 0.9)	(0.3, 0.5, 0.7)	(0.3, 0.5, 0.7)	(0.5, 0.7, 0.9)	(0.7, 0.9, 1.0)	(0.1, 0.3, 0.5)	(0.5, 0.7, 0.9)	(0.5, 0.7, 0.9)	(0.3, 0.5, 0.7)
R_3	(0.5, 0.7, 0.9)	(0.7, 0.9, 1.0)	(0.5, 0.7, 0.9)	(0.7, 0.9, 1.0)	(0.9, 1.0, 1.0)	(0.3, 0.5, 0.7)	(0.5, 0.7, 0.9)	(0.5, 0.7, 0.9)	(0.3, 0.5, 0.7)
R_4	(0.3, 0.5, 0.7)	(0.5, 0.7, 0.9)	(0.7, 0.9, 1.0)	(0.9, 1.0, 1.0)	(0.5, 0.7, 0.9)	(0.5, 0.7, 0.9)	(0.3, 0.5, 0.7)	(0.1, 0.3, 0.5)	(0.3, 0.5, 0.7)
R_5	(0.7, 0.9, 1.0)	(0.7, 0.9, 1.0)	(0.9, 1.0, 1.0)	(0.9, 1.0, 1.0)	(0.7, 0.9, 1.0)	(0.7, 0.9, 1.0)	(0.5, 0.7, 0.9)	(0.3, 0.5, 0.7)	(0.3, 0.5, 0.7)
R_6	(0.7, 0.9, 1.0)	(0.5, 0.7, 0.9)	(0.5, 0.7, 0.9)	(0.3, 0.5, 0.7)	(0.3, 0.5, 0.7)	(0.1, 0.3, 0.5)	(0.3, 0.5, 0.7)	(0.1, 0.3, 0.5)	(0.5, 0.7, 0.9)
R_7	(0.5, 0.7, 0.9)	(0.7, 0.9, 1.0)	(0.7, 0.9, 1.0)	(0.3, 0.5, 0.7)	(0.5, 0.7, 0.9)	(0.5, 0.7, 0.9)	(0.3, 0.5, 0.7)	(0.1, 0.3, 0.5)	(0.5, 0.7, 0.9)
R_8	(0.7, 0.9, 1.0)	(0.5, 0.7, 0.9)	(0.5, 0.7, 0.9)	(0.7, 0.9, 1.0)	(0.5, 0.7, 0.9)	(0.5, 0.7, 0.9)	(0.5, 0.7, 0.9)	(0.1, 0.3, 0.5)	(0.5, 0.7, 0.9)
R_9	(0.7, 0.9, 1.0)	(0.7, 0.9, 1.0)	(0.5, 0.7, 0.9)	(0.9, 1.0, 1.0)	(0.7, 0.9, 1.0)	(0.7, 0.9, 1.0)	(0.5, 0.7, 0.9)	(0.5, 0.7, 0.9)	(0.7, 0.9, 1.0)
R_{10}	(0.9, 1.0, 1.0)	(0.9, 1.0, 1.0)	(0.7, 0.9, 1.0)	(0.5, 0.7, 0.9)	(0.7, 0.9, 1.0)	(0.9, 1.0, 1.0)	(0.3, 0.5, 0.7)	(0.7, 0.9, 1.0)	(0.5, 0.7, 0.9)

Table 4. Synthesized fuzzy weights.

	S^1	S^2	S^3	S^4	S^5	S^6	S^7	S^8	S^9
R_1	0.289	0.233	0.289	0.167	0.167	0.233	0.167	0.100	0.233
R_2	0.233	0.167	0.167	0.233	0.289	0.10	0.233	0.233	0.167
R_3	0.233	0.289	0.233	0.289	0.322	0.167	0.233	0.233	0.167
R_4	0.167	0.233	0.289	0.322	0.233	0.233	0.167	0.100	0.167
R_5	0.289	0.289	0.322	0.322	0.289	0.289	0.233	0.167	0.167
R_6	0.289	0.233	0.233	0.167	0.167	0.10	0.167	0.100	0.233
R_7	0.233	0.289	0.289	0.167	0.233	0.233	0.167	0.100	0.233
R_8	0.289	0.233	0.233	0.289	0.233	0.233	0.233	0.100	0.233
R_9	0.289	0.289	0.233	0.322	0.289	0.289	0.233	0.233	0.289
R_{10}	0.322	0.322	0.289	0.233	0.289	0.322	0.167	0.289	0.233

Table 5. Defuzzified aggregated triangular fuzzy numbers.

	S^1	S^2	S^3	S^4	S^5	S^6	S^7	S^8	S^9
R_1	(2.4, 3.1, 3.5)	(1.7, 2.4, 3.1)	(2.4, 3.1, 3.5)	(1.0, 1.7, 2.4)	(1.0, 1.7, 2.4)	(1.7, 2.4, 3.1)	(1.0, 1.7, 2.4)	(0.3, 1.0, 1.7)	(1.7, 2.4, 3.1)
R_2	(1.7, 2.4, 3.1)	(1.0, 1.7, 2.4)	(1.0, 1.7, 2.4)	(1.7, 2.4, 3.1)	(2.4, 3.1, 3.5)	(0.3, 1.0, 1.7)	(1.7, 2.4, 3.1)	(1.7, 2.4, 3.1)	(1.0, 1.7, 2.4)
R_3	(1.7, 2.4, 3.1)	(2.4, 3.1, 3.5)	(1.7, 2.4, 3.1)	(2.4, 3.1, 3.5)	(3.1, 3.5, 3.5)	(1.0, 1.7, 2.4)	(1.7, 2.4, 3.1)	(1.7, 2.4, 3.1)	(1.0, 1.7, 2.4)
R_4	(1.0, 1.7, 2.4)	(1.7, 2.4, 3.1)	(2.4, 3.1, 3.5)	(3.1, 3.5, 3.5)	(1.7, 2.4, 3.1)	(1.7, 2.4, 3.1)	(1.0, 1.7, 2.4)	(0.3, 1.0, 1.7)	(1.0, 1.7, 2.4)
R_5	(2.4, 3.1, 3.5)	(2.4, 3.1, 3.5)	(3.1, 3.5, 3.5)	(3.1, 3.5, 3.5)	(2.4, 3.1, 3.5)	(2.4, 3.1, 3.5)	(1.7, 2.4, 3.1)	(1.0, 1.7, 2.4)	(1.0, 1.7, 2.4)
R_6	(2.4, 3.1, 3.5)	(1.7, 2.4, 3.1)	(1.7, 2.4, 3.1)	(1.0, 1.7, 2.4)	(1.0, 1.7, 2.4)	(0.3, 1.0, 1.7)	(1.0, 1.7, 2.4)	(0.3, 1.0, 1.7)	(1.7, 2.4, 3.1)
R_7	(1.7, 2.4, 3.1)	(2.4, 3.1, 3.5)	(2.4, 3.1, 3.5)	(1.0, 1.7, 2.4)	(1.7, 2.4, 3.1)	(1.7, 2.4, 3.1)	(1.0, 1.7, 2.4)	(0.3, 1.0, 1.7)	(1.7, 2.4, 3.1)
R_8	(2.4, 3.1, 3.5)	(1.7, 2.4, 3.1)	(1.7, 2.4, 3.1)	(2.4, 3.1, 3.5)	(1.7, 2.4, 3.1)	(1.7, 2.4, 3.1)	(1.7, 2.4, 3.1)	(0.3, 1.0, 1.7)	(1.7, 2.4, 3.1)
R_9	(2.4, 3.1, 3.5)	(2.4, 3.1, 3.5)	(1.7, 2.4, 3.1)	(3.1, 3.5, 3.5)	(2.4, 3.1, 3.5)	(2.4, 3.1, 3.5)	(1.7, 2.4, 3.1)	(1.7, 2.4, 3.1)	(2.4, 3.1, 3.5)
R_{10}	(3.1, 3.5, 3.5)	(3.1, 3.5, 3.5)	(2.4, 3.1, 3.5)	(1.7, 2.4, 3.1)	(2.4, 3.1, 3.5)	(3.1, 3.5, 3.5)	(1.0, 1.7, 2.4)	(2.4, 3.1, 3.5)	(1.7, 2.4, 3.1)

Table 6. Normalized relative weights.

	S^1	S^2	S^3	S^4	S^5	S^6	S^7	S^8	S^9
R_1	(0.27, 0.34, 0.39)	(0.24, 0.33, 0.43)	(0.27, 0.34, 0.39)	(0.20, 0.33, 0.47)	(0.20, 0.33, 0.47)	(0.24, 0.33, 0.43)	(0.20, 0.33, 0.47)	(0.10, 0.33, 0.57)	(0.24, 0.33, 0.43)
R_2	(0.24, 0.33, 0.43)	(0.20, 0.33, 0.47)	(0.20, 0.33, 0.47)	(0.24, 0.33, 0.43)	(0.27, 0.34, 0.39)	(0.10, 0.33, 0.57)	(0.24, 0.33, 0.43)	(0.24, 0.33, 0.43)	(0.20, 0.33, 0.47)
R_3	(0.24, 0.33, 0.43)	(0.27, 0.34, 0.39)	(0.24, 0.33, 0.43)	(0.27, 0.34, 0.39)	(0.31, 0.35, 0.35)	(0.20, 0.33, 0.47)	(0.24, 0.33, 0.43)	(0.24, 0.33, 0.43)	(0.20, 0.33, 0.47)
R_4	(0.20, 0.33, 0.47)	(0.24, 0.33, 0.43)	(0.27, 0.34, 0.39)	(0.31, 0.35, 0.35)	(0.24, 0.33, 0.43)	(0.24, 0.33, 0.43)	(0.20, 0.33, 0.47)	(0.10, 0.33, 0.57)	(0.20, 0.33, 0.47)
R_5	(0.27, 0.34, 0.39)	(0.27, 0.34, 0.39)	(0.31, 0.35, 0.35)	(0.31, 0.35, 0.35)	(0.27, 0.34, 0.39)	(0.27, 0.34, 0.39)	(0.24, 0.33, 0.43)	(0.20, 0.33, 0.47)	(0.20, 0.33, 0.47)
R_6	(0.27, 0.34, 0.39)	(0.24, 0.33, 0.43)	(0.24, 0.33, 0.43)	(0.20, 0.33, 0.47)	(0.20, 0.33, 0.47)	(0.10, 0.33, 0.57)	(0.20, 0.33, 0.47)	(0.10, 0.33, 0.57)	(0.24, 0.33, 0.43)
R_7	(0.24, 0.33, 0.43)	(0.27, 0.34, 0.39)	(0.27, 0.34, 0.39)	(0.20, 0.33, 0.47)	(0.24, 0.33, 0.43)	(0.24, 0.33, 0.43)	(0.20, 0.33, 0.47)	(0.10, 0.33, 0.57)	(0.24, 0.33, 0.43)
R_8	(0.27, 0.34, 0.39)	(0.24, 0.33, 0.43)	(0.24, 0.33, 0.43)	(0.27, 0.34, 0.39)	(0.24, 0.33, 0.43)	(0.24, 0.33, 0.43)	(0.24, 0.33, 0.43)	(0.10, 0.33, 0.57)	(0.24, 0.33, 0.43)
R_9	(0.27, 0.34, 0.39)	(0.27, 0.34, 0.39)	(0.24, 0.33, 0.43)	(0.31, 0.35, 0.35)	(0.27, 0.34, 0.39)	(0.27, 0.34, 0.39)	(0.24, 0.33, 0.43)	(0.24, 0.33, 0.43)	(0.27, 0.34, 0.39)
R_{10}	(0.31, 0.35, 0.35)	(0.31, 0.35, 0.35)	(0.27, 0.34, 0.39)	(0.24, 0.33, 0.43)	(0.27, 0.34, 0.39)	(0.31, 0.35, 0.35)	(0.20, 0.33, 0.47)	(0.27, 0.34, 0.39)	(0.24, 0.33, 0.43)

Table 7. Global weights.

	S^1	S^2	S^3	S^4	S^5	S^6	S^7	S^8	S^9
R_1	0.74	0.71	0.74	0.69	0.69	0.71	0.69	0.62	0.71
R_2	0.71	0.69	0.69	0.71	0.74	0.62	0.71	0.71	0.69
R_3	0.71	0.74	0.71	0.74	0.78	0.69	0.71	0.71	0.69
R_4	0.69	0.71	0.74	0.78	0.71	0.71	0.69	0.62	0.69
R_5	0.74	0.74	0.78	0.78	0.74	0.74	0.71	0.69	0.69
R_6	0.74	0.71	0.71	0.69	0.69	0.62	0.69	0.62	0.71
R_7	0.71	0.74	0.74	0.69	0.71	0.71	0.69	0.62	0.71
R_8	0.74	0.71	0.71	0.74	0.71	0.71	0.71	0.62	0.71
R_9	0.74	0.74	0.71	0.78	0.74	0.74	0.71	0.71	0.74
R_{10}	0.78	0.78	0.74	0.71	0.74	0.78	0.69	0.74	0.71

Table 8. Final weights.

	Fw_{ij}	*Rank*
R_1	0.279	6
R_2	0.278	7
R_3	0.283	3
R_4	0.278	7
R_5	0.286	2
R_6	0.277	8
R_7	0.280	5
R_8	0.280	5
R_9	0.286	2
R_{10}	0.287	1

Therefore, the strengths of this method are reduced disagreements or discrepancies among ranked requirements and dealing with uncertainties associated with decision making processes. The proposed method is also reliable as seen by comparing the final output in Table 8 with the linguistic values in Table 2. There is a strong correlation between the prioritized requirements generated by the proposed method and the linguistic values provided by stakeholders.

5. Conclusion and Future Work

To avoid breach of agreement or contract in software development projects, stakeholders can converge to prioritize specified requirements. This is due to the fact that not all the specified requirements are always implementable in a single release. Software requirements prioritization is exercised by relative comparisons, where preference weights are assigned against requirements to reflect their values as perceived by stakeholders. It is therefore considered to be a multi-criteria decision making process. The rationale for undertaking this research arose from the fact that existing prioritization techniques are challenged by their inability of addressing communication among stakeholders during prioritization. In the

proposed method, relative weights of requirements are determined by a linguistic scale and prioritization takes place by analyzing the coefficient values of requirements. The outputs generated by the proposed method have an acceptable level of accuracy. Thus, prioritization helps stakeholders plan for software release phases with respect to available budget, time and skilled programmers. Validation of the proposed method was based on an illustrative example. Finally, the proposed method is able to classify ranked requirements with the computation of maximum, minimum, mean and aggregated scores to display prioritized requirements. Future work borders on implementation of a prototype tool and validation in a real environment.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Perini, A., Susi, A. and Avesani, P. (2013) A Machine Learning Approach to Software Requirements Prioritization. *IEEE Transactions on Software Engineering*, **39**, 445-461. https://doi.org/10.1109/TSE.2012.52

[2] Ahl, V. (2005) An Experimental Comparison of Five Prioritization Methods-Investigating Ease of Use, Accuracy and Scalability. Master's Thesis, School of Engineering, Blekinge Institute of Technology, Sweden.

[3] Thakurta, R. (2012) A Framework for Prioritization of Quality Requirements for Inclusion in a Software Project. *Software Quality Journal*, **21**, 573-597. https://doi.org/10.1007/s11219-012-9188-5

[4] Karlsson, L., Thelin, T., Regnell, B., Berander, P. and Wohlin, C. (2007) Pair-Wise Comparisons versus Planning Game Partitioning-Experiments on Requirements Prioritisation Techniques. *Empirical Software Engineering*, **12**, 3-33. https://doi.org/10.1007/s10664-006-7240-4

[5] Berander, P., Khan, K.A. and Lehtola, L. (2006) Towards a Research Framework on Requirements Prioritization. *Proceedings of Sixth Conference on Software Engineering Research and Practice in Sweden* (*SERPS 06*).

[6] Karlsson, J. and Ryan, K. (1997) A Cost-Value Approach for Prioritizing Requirements. *IEEE Software*, **14**, 67-74. https://doi.org/10.1109/52.605933

[7] Kobayashi, M. and Maekawa, M. (2001) Need-Based Requirements Change Management. *Proceedings ECBS: Eighth Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, 171-178.

[8] Kassel, N.W. and Malloy, B.A. (2003) An Approach to Automate Requirements Elicitation and Specification. *Proceedings of the 7th IASTED International Conference on Software Engineering and Applications*, Marina Del Rey, 3-5 November 2003.

[9] Tonella, P., Susi, A. and Palma, F. (2013) Interactive Requirements Prioritization Using a Genetic Algorithm, Information and Software Technology. *Information and Software Technology*, **55**, 173-187. https://doi.org/10.1016/j.infsof.2012.07.003

[10] Karlsson, J., Wohlin, C. and Regnell, B. (1998) An Evaluation of Methods for Prioritizing Software Requirements. *Information and Software Technology*, **39**, 939-947. https://doi.org/10.1016/S0950-5849(97)00053-0

[11] Babar, M., Ramzan, M. and Ghayyur, S. (2011) Challenges and Future Trends in Software Requirements Prioritization. *International Conference on Computer Networks and Information Technology*, **2011**, 319-324. https://doi.org/10.1109/ICCNIT.2011.6020888

[12] Greer, D. and Ruhe, G. (2004) Software Release Planning: An Evolutionary and Iterative Approach. *Information and Software Technology*, **46**, 243-253. https://doi.org/10.1016/j.infsof.2003.07.002

[13] Ramzan, M., Jaffar, A. and Shahid, A. (2011) Value Based Intelligent Requirement Prioritization (VIRP): Expert Driven Fuzzy Logic Based Prioritization Technique. *International Journal of Innovative Computing*, **7**, 1017-1038.

[14] Saher, N., Baharom, F. and Romli, R. (2020) Guideline for the Selection of Requirement Prioritization Techniques in Agile Software Development: An Empirical Research. *International Journal of Recent Technology and Engineering* (*IJRTE*), **8**, 3381-3388. https://doi.org/10.35940/ijrte.E6634.018520

[15] Borhan, N.H., Zulzalil, H. and Sa'adah Hassan, N.M.A. (2019) Requirements Prioritization Techniques Focusing on Agile Software Development: A Systematic Literature. *International Journal of Scientific and Technology Research*, **8**, 2118-2125.

[16] Achimugu, P., Selamat, A., Ibrahim, R. and Mahrin, M.N.R. (2014) A Systematic Literature Review of Software Requirements Prioritization Research. *Information and Software Technology*, **56**, 568-585. https://doi.org/10.1016/j.infsof.2014.02.001