INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

PUGH Decision Trapezoidal Fuzzy and Gradient Reinforce Deep Learning for Large Scale Requirement Prioritization

Raghavendra Devadas[1,2]*, **Nagaraj G Cholli**[3]

1 Research Scholar, RV College of Engineering, Visveswaraya Technological University, Belgaum
2 Assistant Professor, Department of CSE, Presidency University, Bangalore
3 Associate Professor. Department of ISE, RV College of Engineering, Bangalore

OPEN ACCESS

Received: 21-09-2021
Accepted: 18-02-2022
Published: 25.03.2022

Citation: Devadas R, Cholli NG (2022) PUGH Decision Trapezoidal Fuzzy and Gradient Reinforce Deep Learning for Large Scale Requirement Prioritization. Indian Journal of Science and Technology 15(12): 542-553. https://doi.org/10.17485/IJST/v15i12.1757

* **Corresponding author**.

raghudevdas@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objective: To prioritize requirements for large scale software projects within time involving uncertainty in the opinions among different stakeholders. **Methods:** We propose Pugh Trapezoidal Fuzzy and Gradient Reinforce Learning (PTF-GRL) methods for large scale software requirement prioritization. A Pugh Decision-based Trapezoidal Fuzzy Requirement Selection model is designed, inputting the functional and non-functional requirements of the corresponding stakeholders. With the assistance of Trapezoidal Fuzzy Inference, the qualitative factors are mapped with the corresponding numeric factors, which increases the computational efficiency. **Findings:** Performance is analyzed based on four parameters: The first parameter is accuracy and our method showed improvement of 4%, 7% and 3% compared to JRD-SCRUM, IFS and SRP-Tackle respectively. The second parameter is prioritization time and found that our method had reduced time of 30%, 37% and 39% compared with existing methods. The third parameter is precision and it was found that our method improves precision by 6%, 10% and 5% compared with the other two methods. The final parameter we consider is the test suite execution and our method showed improvement of 12%, 19% and 5% compared with the existing two methods. Novelty/Applications: The originality of this work indicates the better performance along with the optimal test suite execution even considering the uncertainty factor in the proposed method compared with existing similar methods.

Keywords: Software Project; Pugh Decision Matrix; Trapezoidal Fuzzy Inference; Gradient Orientation; Reinforce Learning; Requirement Prioritization

1 Introduction

Agile software development (ASD) consists of incomplete requirements identification, vague requirements analysis, secondary requirement prioritization that has a minimum negative influence on the functionality of software systems and quality. A Joint Requirements Documents (JRD) and Systematic Customer Resolution Unraveling

Meeting (SCRUM) called, JRD-SCRUM was proposed in [1] that in turn incrementally identified, implemented, evolved, and managed system requirements via software development cycle.

An action research method was proposed with the objective of qualitatively validating the requirement prioritization based on data and user requirements acquired from a product-concentric Norway-based software company. With this, the individual responses for implementing JRD with SCRUM were proved to be highly accurate. However, during the requirement selection process, with the large amount of uncertainty involved between different stakeholders, a significant amount of precision is said to be compromised. To address this issue, in this work, a Pugh Decision-based Trapezoidal Fuzzy Requirement Selection model is proposed that in turn improves the precision rate by minimizing the uncertainty.

An Intuitionistic Fuzzy Approach (IFS) provided a scalable framework to ensure stakeholders' preferences in the form of priority values via multidimensional evaluation parameter was proposed in [2,3]. It also concentrated on the collaboration between stakeholders by taking into consideration both the intuitionistic and nonmembership parameter. Also, the dependency and developer's initial preference along with the technical constraints were considered.

With this reliability and robustness to errors were ensured. However, with expeditious changing large scale software projects with plentiful data including both functional and non-functional requirements the test suite execution was not ensured. Hence to address this issue, in this work a deep learning model called, Gradient Orientation-based Reinforce Requirement Prioritization is designed to flexibly procure more perceptions into individual test cases, therefore ensuring an optimal number of test suite execution.

The association of stakeholders is a key success factor for Software Engineering (SE). Numerous process mechanisms are within easy reach to represent Requirements Engineering (RE)activities. Key activities comprise requirements elicitation, prioritization, and negotiation. The requirement prioritization shows its significance in comparison to others. It also assists in determining the requirements to be incorporated into a project. In addition, prioritization strengthens the negotiation of requirements that focus on the resolution of disagreements by defining an agreement that meets all stakeholders.

A hybrid fuzzy analytical hierarchy process was designed in [4] via a multi-criteria decision-making approach for quantifying reusability in requirement prioritization. In the present-day situation, the concept of social networks has entirely metamorphosed into a new domain and hence has the potential to impact RE activities. In [5], an approach was proposed whereby the social network was exploited to carry out RE activities such as elicitation, prioritization, and negotiation. However, with a multi-stakeholder perspective, the prioritization process was not concerned with diversification. To address this issue, aspect-based requirement mining techniques were proposed in [6] taking into consideration the diversification aspect.

Requirements Specifications (RSs) are feasibly the most innermost artifacts to the requirement engineering procedures. This is due to the reason that an RS lays out the requisite features, potentialities, and aspects of a system-to-be. To facilitate understanding and links between stakeholders with distinct backgrounds and competencies, natural language is being used today.

A software effort estimation using Bayes with fuzzification of numbers was utilized in [7] based on the optimal control using Gene and Particle Swarm Optimization, therefore, contributing to the estimation accuracy. Yet another novel approach was proposed in [8] by utilizing Bayesian network classifiers to categorize the software defects. However, neither the existence nor a completely precise execution of such understandings is guaranteed. An automatic approach was designed in [9] using a machine learning-based approach, therefore reducing the time-consuming process. A case study for agile method adaptation exploring requirement prioritization for large scale software projects was investigated in [10]. A machine learning model was utilized in [11] to maintain the prediction accuracy of spatial information involving prioritization based on the significance of data. However, customization of stakeholders' specifications was said to be of great significance.

The contributions of the work include the following:

- The main contribution of the proposed Pugh Trapezoidal Fuzzy and Gradient Reinforce Learning (PTF-GRL) is used to handle uncertainty and test suite execution issue among different stakeholders during large scale software prioritization.
- To handle uncertainty issues among different stakeholders for large scale software projects employing Pugh Decision-based Trapezoidal Fuzzy Requirement Selection model in a precise and computationally efficient manner.
- To focus on issues concerning test suite execution among different stakeholders while involving large scale software using a deep learning model called, Gradient Orientation-based Reinforce Requirement Prioritization.
- Our experimental results have been computed on a software requirement dataset. By applying our PTF-GRL method on this dataset, we show that the method is deployable in large scale software project settings.

This study is organized as follows: Section 2 discusses about research gap and motivation; Section 3 describes the proposed Pugh Trapezoidal Fuzzy and Gradient Reinforce Learning (PTF-GRL) method with an elaborated discussion on how stakeholders and developers cooperate and provide initial requirements that are in turn employed in the algorithm to compute requirement

selection and final requirement prioritization. Section 4 introduces the experimental settings for implementing the PTF-GRL method followed by empirical validation and comparison of performance assessment of PTF-GRL with JRD-SCRUM and IFS. Finally, Section 5 presents the conclusion.

2 Research Gap and Motivation

The software configuration was developed to customize the software for dissimilar users. In [12], an integration of indicator-based evolution algorithm and the differential evolutionary algorithm was proposed to minimize the search space involved during prioritization. A systematic literature review of software requirements prioritization, its strengths, the limitation was investigated in depth in [13]. Yet another consistency prioritization model using fuzzy fault tree analysis was proposed in [14]. However, it was not found to be feasible enough for large scale agile development. To address this issue, a case study including different processes and purposes was designed in [15].

In recent years, software development organizations are modeling agile mechanisms in the global software development (GSD) environment with the purpose of meeting the requirements of the swiftly evolving and ceaselessly growing business framework. The specific objectives of the work proposed in [16] were to identify the crucial hurdles and design a prioritization-based taxonomy for modeling agile evolution in the GSD framework. An empirical study for agile requirements prioritization for large scale software projects was proposed in [17].

Arranging and identifying a set of releases with requirement prioritization denotes a demanding task owing to the reason that the requirements consist of their features, involving, technical priority, implementation cost, the significance of one or more stakeholders toinclude the requirement, and so on. In [18], Verbal Decision Analysis was employed for identifying reasonable solutions to resolve these types of issues. Yet another approach utilizing hybrid access was employed in [19] to prioritize both functional and non-functional requirements, therefore, reducing the time involved. Two factors, cost, and value were analyzed in [20] using a hybrid technique, therefore, reducing the time involved in the prioritization process.

A software project management framework was introduced in [21] for enhancing the success rate. But the time took was higher. A multiattribute framework was developed in [22] to verify and validate stakeholder objectives and constraints. The designed framework was enhancing the accuracy of the prioritization. However, it failed to handle the real-world project issue in engineering management. To solve the issue, the Test case prioritization (TCP) method was introduced in [23] to reduce the computational complexity and execution time. But the various latest cases were added to the test suite.

The novel nature-inspired optimization method called Bat algorithm was developed in [24] to enhance the software quality for reducing the time. However, it failed to handle the TCP issue.

Requirements Change Management in Agile (ARCM) was introduced in [25] for discovering the success factors. But the prioritization accuracy was not enhanced. The requirement prioritization method was introduced in [26] to focus on the scalability issues. Therefore, the prioritizationaccuracy was improved. The approach [27] designed focused on determining the importance of requirements with respect to different stakeholders.

New semiautomated scalable prioritisation technique called 'SRPTackle' was designed in [28]. SRPTackle provides a semiautomated process based on a combination of a constructed requirement priority value formulation function using a multi-criteria decision-making method, clustering algorithms (K-means and K-means++) and a binary search tree to reduce the need for expert involvement and increase efficiency. But requirement prioritization time was not minimized.

Fuzzy AHP methods were introduced in [29] for discovering the taxonomy of the SPI success factors. But the requirement prioritization accuracy was not considered.An analytic hierarchy process method was developed in [30] to recognize and prioritize the challenges. AHP approach was introduced in [31] to achieve the software organizations by using scale agile method in GSD. Fuzzy AHP based conceptual mapping was developed in [32] for handling the GSD to prioritize the empirically validated achievement factors. Interdependency among the large scale requirements and an approach to address volatility of requirements were in addressed in [33].

All distinct types of explorations illustrate that there is a requirement for methods that prioritize the requirements of all stakeholders by their viewpoint and control the semantic of large-scale software handling uncertainty and test suite execution with absoluteness and uniformity. Therefore, we utilized fuzzy and deep learning models to handle uncertainty and improve test suite execution among different stakeholders to reduce the human endeavor to control large scale software.

Motivated by the above research gaps, this research study has been performed with the objective to address the issue of uncertainty and test suite execution improvement management in the domain of large-scale software prioritization by selecting and prioritizing the key features and their categorical functional and non-functional requirements that could positively influence the software engineering activities.

3 Proposed Methodology

This section details the proposed Pugh Trapezoidal Fuzzy and Gradient Reinforce Learning (PTF-GRL) method. Qualitative software requirements obtained from the software requirement dataset are the essential inputs of the proposed method. The PTF-GRL method provides an interactive requirement selection and prioritization process capable of handling uncertainty and test case execution between different stakeholders during large scale software prioritization. Figure 1 shows the block diagram of the PTF-GRL method.

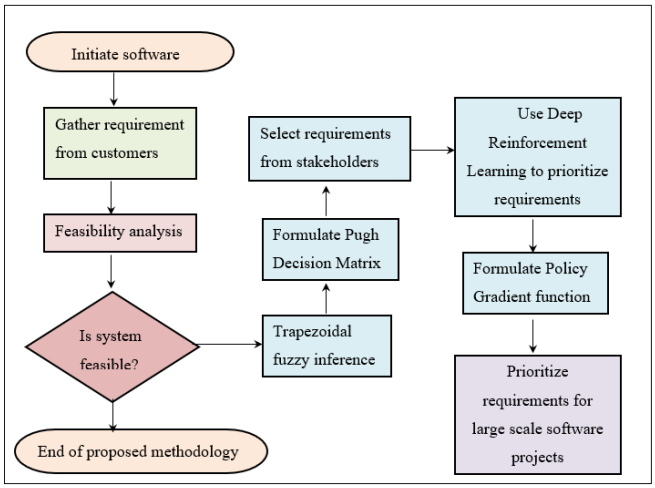

Fig 1. Block diagram of Pugh Trapezoidal Fuzzy and GradientReinforce Learning

As illustrated in Figure 1, the PTF-GRL method is split into two sections. First requirement selection is modeled by means of Pugh Decision-based Trapezoidal Fuzzy Requirement Selection model to handle uncertainty between different stakeholders. Followed bythe selected requirements, a requirement prioritization method is designed using Gradient Orientation-based Reinforce Requirement Prioritization model to address test suite execution. An elaborate description of the PTF-GRL method is provided in the following sub-sections.

3.1 Pugh Decision-based Trapezoidal Fuzzy Requirement Selection model

In the formulation of a large-scale software project, software developers must attempt to improve business value while maintaining a high degree of certainty that the product will be completed on time and within budget. Owing to this, restrictions management is frequently made mandatory to model the hard decision as to which stakeholders' opinions to consider and which opinion to ignore, therefore heavy concern towards the uncertainty aspect. In this work, a Pugh Decision-based Trapezoidal Fuzzy Requirement Selection model that deals with the uncertain situation between different stakeholders during large scale software prioritization is designed. The inferences of the Pugh Decision-based Trapezoidal Fuzzy Requirement Selection model dispense with the coarse-grained information and control the uncertainties in the decision-making issues on the basis of the discernments. Figure 2 shows the structure of Pugh Decision-based Trapezoidal Fuzzy Requirement Selection model.

As shown in Figure 2, let us consider a Trapezoidal Fuzzy Numbers (TFN) represented by a triad '$\left(T^{ls}, T^{ns}, T^{ms}\right)$' that denotes the membership function '$\mu T\,(R)$' of the software requirement for large scale software projects.

$$\mu_{\mathrm{T}}(\mathrm{R}) = \begin{cases} \dfrac{\mathrm{R} - \mathrm{T}^{\mathrm{l\ s}}}{\mathrm{T}^{\mathrm{ns}} - \mathrm{T}^{\mathrm{ls}}}, \mathrm{T}^{\mathrm{ls}} \leq \mathrm{R} \leq \mathrm{T}^{\mathrm{ns}} \\ \dfrac{\mathrm{T}^{\mathrm{ms}} - \mathrm{R}}{\mathrm{T}^{\mathrm{ms}} - \mathrm{T}^{\mathrm{ns}}}, \mathrm{T}^{\mathrm{ns}} \leq \mathrm{R} \leq \mathrm{T}^{\mathrm{ms}} \\ 0, \text{ Otherwise} \end{cases} \tag{1}$$

From equation (1), 'T^{ls}', 'T^{ns}' and 'T^{ms}' represent the least significance, nominal significance, and the most significant requirement values respectively. Let us further consider two Trapezoidal Fuzzy Numbers '$T_1 = \left(T_1^{ls}, T_1^{ns}, T_1^{ms}\right)$' and '$T_2 =$

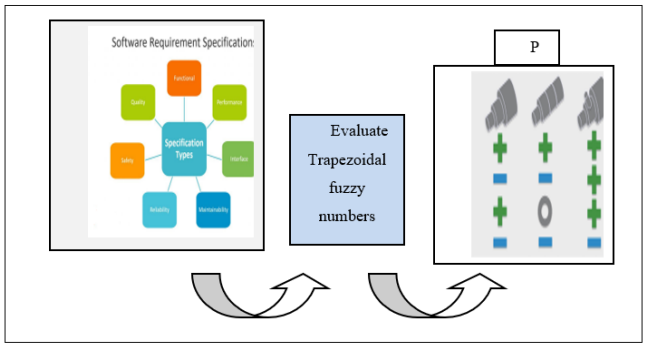

Fig 2. Structure of Pugh Decision-based Trapezoidal Fuzzy Requirement Selection

$\left(T_2^{ls},\ T_2^{ns},\ T_2^{ms}\right)$', then the distance between 'T_1' and 'T_2' is estimated as given below.

$$Dis\left(T_1, T_2\right) = \sqrt{\frac{1}{3}\left[\left(T_1^{ls} - T_2^{ls}\right)^2 + \left(T_1^{ns} - T_2^{ns}\right)^2 + \left(T_1^{ms} - T_2^{ms}\right)^2\right]} \tag{2}$$

The conventional deep dive [1] cannot handle stakeholders' uncertainty related to quantifying the priorities of different criteria. Owing to this reason, a fuzzy inference (FI) is integrated with the Pugh Decision Matrix model called, Fuzzy Inference-based Pugh Decision Matrix, which is used to select more accurate and adequate requirements using the minimum extent of possibility in uncertain problems. Trapezoidal Fuzzy Numbers are utilized in this model for selecting rankings of the benchmark and procuring priority weight of definite benchmark utilizing the Pugh Decision Matrix. As 'T_1' and 'T_2' are two Trapezoidal Fuzzy Numbers, then the extent of possibility is formulated as given below.

$$T_2 = \left(T_2^{ls},\ T_2^{ns},\ T_2^{ms}\right) \geq T_1 = \left(T_1^{ls},\ T_1^{ns},\ T_1^{ms}\right) \tag{3}$$

$$EoP\ \left(T_2 \geq T_1\right) = SUP\ \left(MIN(\mu T_1\left(R\right)), (\mu T_2\left(R\right))\right] \tag{4}$$

Owing to the reason that different types of software products include non-functional and functional requirements, then for each requirement extent values are analyzed by means of Pugh Decision Matrix as given below.

$$PDM\left(REVal\right) = \begin{bmatrix} SH1\left(SF^1_{(W_1)}\right) & SH2\left(SF^1_{(W_1)}\right) & SH3\left(SF^1_{(W_1)}\right) & \ldots & SHn\left(SF^1_{(W_1)}\right) \\ SH1\left(SF^2_{(W_2)}\right) & SH2\left(SF^2_{(W_2)}\right) & SH3\left(SF^2_{(W_2)}\right) & \ldots & SHn\left(SF^2_{(W_2)}\right) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ SH1\left(SF^n_{(W_n)}\right) & SH2\left(SF^n_{(W_n)}\right) & SH3\left(SF^n_{(W_n)}\right) & \ldots & SHn\left(SF^n_{(W_n)}\right) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ Agg\left(SH1\right] & Agg\left(SH2\right] & Agg\left(SH3\right] & \ldots & Agg\left(SHn\right] \end{bmatrix} \tag{5}$$

From equation (5), the holistic aggregates acquired from the Pugh Decision Matrices for each requirement extent values are utilized in formulating and checking the consistency between the pair-wise matrices and for de-fuzzifying the matrix to address uncertainty to a greater extent. The pseudo-code representation of Pugh Decision and Trapezoidal Fuzzy Requirement Selection is given in Algorithm 1.

As given in Algorithm 1, the objective is to select the relevant requirements among different stakeholders during large scale software prioritization with high precision in a computationally efficient manner. To attain this objective, the requirements of the customers are provided as input, first, using Pugh Decision and Trapezoidal Fuzzy Requirement Selection algorithm, with the aid of Trapezoidal Fuzzy Numbers the objects or the requirements are mapped between 0 and 1. Followed by which, the distance between the two TFN's requirements is obtained. Finally, the extent of possibility via Pugh Decision Matrix is estimated to obtain and select precise function, non-function requirements in a computationally efficient manner.

Input: Customers '$C = C_1, C_2, C_3, \ldots, C_m$', Requirements '$R = R_1, R_2, R_3, \ldots, R_n$'

Output: Precise and computationally efficient requirement selection

1: Begin 2: For each customer 'C' and requirements 'R' [including both functional 'FR' and non-functional requirements 'NFR'] 3: Extract the requirements 'R' from the customers 'C' 4: Obtain the rule using equation (1) 5: Estimate fuzzification inference via two Trapezoidal Fuzzy Numbers as in equation (2) for both functional 'FR' and non-functional requirements 'NFR' 6: Estimate the extent of possibility via Pugh Decision Matrix as in equations (3) and (4) for both 'FR' and 'NFR' //Perform de-fuzzification 7: Estimate requirement extent values via Pugh Decision Matrix as in equation (5) for both 'FR' and 'NFR' 8: Return relevant function requirements 'RFR' and non-functional requirements '$RNFR$' 9: End for 10: End

ALGORITHM 1. Pugh Decision and Trapezoidal Fuzzy Requirement Selection

3.2 Gradient Orientation-based Reinforce Requirement Prioritization model

The goal of requirement prioritization is to identify and prioritize the ordered sequence of test samples to increase the performance metric. With this purpose, in our work Gradient Orientation-based Reinforce Requirement Prioritization model is designed to disclose failure as early as possible and therefore contribute to test suite execution.

In the proposed model, let us consider a triad '$(TS, P(TS), f(S))$', where 'TS' represents the test samples, '$P(TS)$' is the permutation of test samples and '$f(S)$' is a function scaling '$P(TS)$' to rational numbers. Then, for the '$n-th$' factor, failing test samples are organized simultaneously resulting in a progression '$F_1, F_2, F_3, \ldots, F_n$' with '$F_i$' forming an organization of failing test samples in the '$i-th$' instance. Similarly, the same operation is performed for passing test cases, forming, '$P_1, P_2, P_3, \ldots, P_n$' with '$P_i$' forming an organization of passing test samples in the '$i-th$' instance. Every permutation of the input test suite as states is provided that the test suite 'T' with 'n' instances of test samples and the state space size is said to be '$n!$'. Figure 3 illustrates the distinct test suite permutations or the structure of permuted state space.

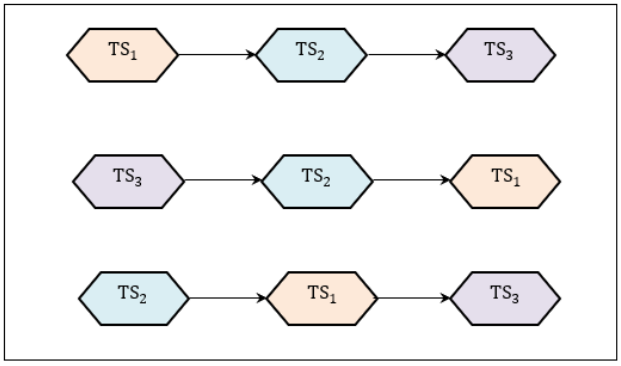

Fig 3. Test suite permutation

As specified in Figure 3, the states form the permutations of the input test suite, an action refers to the condition that it is the indicator for the Gradient Orientation-based Reinforce Requirement Prioritization model to progress from one permutation to another. In our work, the Gradient Orientation-based Reinforce Requirement Prioritization model is the state of input test suite switches between each other in an arbitrary pattern. The structure of arbitrary action learning is illustrated in Figure 4.

As the test samples are ranked on the basis of the accumulated decision cases as each test sample's individual reward and the test sample failure rewards are taken into consideration. In other words, failing test samples are more concentrated and rewarded than the passing test samples, therefore resulting in earlier execution. This is formulated as given below.

$$\mathrm{R}_{\mathrm{i}}^{\mathrm{TSF}}(\mathrm{TS}) = \begin{cases} 1 - \mathrm{TS.Dec_i}, & \text{if } \mathrm{TS} \in \mathrm{TS_i} \\ 0, & \text{Otherwise} \end{cases} \tag{6}$$

As given in equation (6), for each test samples 'TS' the reward function returns the test sample's decision as each test sample's individual reward. Then, for every test sample 'TS' in the set of failing test samples 'TSF', the weights of the failing test instances are increased for an intended proportion 'Pr^F', and accordingly, rewards are estimated. On the other hand, test samples belonging to passing test instances are increased to an intended proportion 'Pr^P'. Therefore, in our proposed work, we select '$Pr^F \geq Pr^P$' as higher proportions or weights have higher priority.

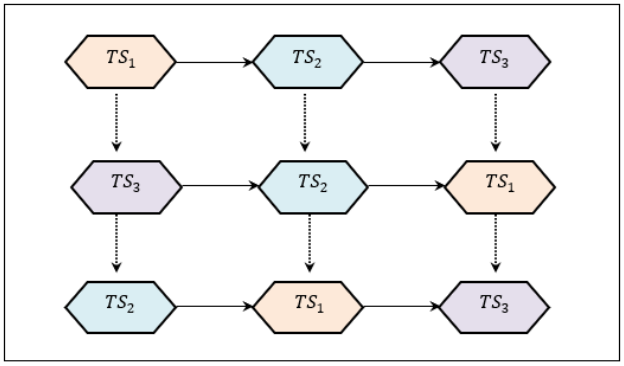

Fig 4. Structure of Arbitrary Action Learning

With this objective finally, a Gradient Orientation function is employed in our work to evaluate the policy 'Pol_θ' by means of a parameterized function estimator. The objective here remains in maximizing the objective function denoting the return represented by '$R(\alpha)$' (i.e., representing the sum of rewards) of the test sample orientations '$\alpha = (s_0, a_0, s_1, a_1, s_2, a_2, \ldots..,s_n a_n)$' selected by the policy 'Pol_θ'.

$$J(\theta) = \sum R(\alpha) = \sum_{i=1}^{n} r\ (s_i,\ a_i,\ s_{i+1}) \tag{7}$$

To maximize this objective function (i.e., considering test sample failure rewards than the pass test samples), the policy 'Pol_θ' only generates test sample orientations correlated with high returns '$R(\alpha)$' and circumventing those with low returns. The objective function employs the projection of the return overall probable test sample orientations. The likelihood that a test sample orientation is occasioned by the policy 'Pol_θ' is denoted as '$\rho_\theta(\alpha)$' mathematically formulated as given below.

$$\rho_\theta(\alpha) = \rho_\theta(s_0, a_0, s_1, a_1, s_2, a_2, \ldots..,s_n a_n) \tag{8}$$

$$= \rho_\theta(s_0) \prod_{i=1}^{n} Pol_\theta(s_i,\ a_i)\, Prob\ (s_{i+1} \mid s_i, a_i) \tag{9}$$

From equation (9), '$\rho_\theta(s_0)$' represents the initial probability starting in 's_0' and '$Prob\ (s_{i+1} \mid s_i,\ a_i)$' represents the transition probability. Finally, given a test sample, the proposed model evaluates the total weight. The larger the value, the higher the test sample, and the test instance is ranked in the resultant test suite, therefore contributing to robust and optimal requirement prioritization. The pseudo-code representation of Gradient Orientation-based Reinforce Requirement Prioritization is given below.

Input: Input: Customers '$C = C_1, C_2, C_3, \ldots., C_m$', Requirements '$R = R_1, R_2, R_3, \ldots., R_n$',

Output: Robust and optimal requirement prioritization

1: Initializerelevant function requirements 'RFR' and non-functional requirements '$RNFR$' 2: Initialize test samples 'TS', permutation of test samples '$P(TS)$' 3: Initialize failing test samples '$F_i = 0$' 4: Initialize passing test samples '$P_i = 0$' 5: Begin 6: Foreach customer 'C' and requirements 'R' [including both relevant function requirements 'RFR' and non-functional requirements '$RNFR$'] 7: Formulate the objective function as in equation (6) 8: Estimate return or sum of reward as in equation (7) 9: For each test sample orientation 10: Evaluate the gradient policy and rank resultant test suite as in equations (8) and (9) 11: Return prioritized results (s) 12: End for 13: End for 14: End

ALGORITHM 2. Gradient Orientation-based Reinforce Requirement Prioritization

As given in Algorithm 2, the objective is in prioritizing the software requirements involving large scale software projects. To attain this objective, a deep learning model with the Gradient Orientation function is designed that flexibly procure more perceptions into individual test cases or test samples, therefore resulting in an optimal number of test suite execution. Here, with the aid of Deep Reinforce learning failing test samples are highly concentrated than the passing test samples. This early execution is attained contributing to robust and optimal requirement prioritization.

4 Results and Discussion

In this section, the proposed Pugh Trapezoidal Fuzzy and Gradient Reinforce Learning (PTF-GRL) for large scale software requirement prioritization are illustrated employing the dataset https://www.kaggle.com/iamsouvik/software-requirements-dataset [3]. The performance is simulated in Python. An elaborate comparison is provided with the state-of-the-art methods, [Joint Requirements Documents (JRD) and Systematic Customer Resolution Unraveling Meeting (SCRUM) – JRD-SCRUM [1], Intuitionistic Fuzzy Approach (IFS) [2] and Semiautomated scalable prioritisation technique (SRPTackle) [28].

To ensure fair comparison, similar set of requirements are selected within the same environment with the aid of Intel(R) Core (TM) i3 CPU 2.13 GHz processor with 4 GB RAM for all three methods, PTF-GRL, JRD-SCRUM [1], and IFS [2], SRPTackle [28]. Simulation parameters employed for analyzing the results are requirement prioritization accuracy, requirement prioritization time, precision, and the number of test suite execution.

4.1 Performance analysis of requirement prioritization accuracy

Owing to the fact that the software quality is defined by the customer or stakeholder's satisfaction, with the purpose of improving the stakeholder's satisfaction under certain yardsticks like, time, resource, the software engineer requires to prioritize the requirements concerning large scale software projects. The benchmark for identifying the requirement prioritization remains in measuring the accuracy and this is mathematically expressed as given below.

$$RPA = \sum_{i=1}^{n} \frac{R_{AP}}{R_i} \tag{10}$$

From equation (10), the requirement prioritization accuracy 'RPA' is estimated on the basis of the requirements analyzed 'R_i' and the requirement prioritized in an accurate manner 'R_{AP}'. It is measured in terms of percentage (%). The measure of requirement prioritization accuracy has been tabulated in Table 1 . The measure of higher requirement prioritization accuracy indicates that the proposed PTF-GRL method is comparatively better than [1,2,28].

Table 1. Tabulation forrequirement prioritization accuracy using PTF-GRL, JRD-SCRUM [1], IFS [2] and SRPTackle [28].

Requirements	Requirement Prioritization Accuracy (%)			
	PTF-GRL	JRD-SCRUM	IFS	SRPTackle
60	96.66	93.37	91.66	94.66
120	95.89	91.29	89.99	93.39
180	95.29	90.59	89.39	89.39
240	95.04	90.39	88.29	92.89
300	94.19	90.04	88.04	93.19
360	94.04	91.19	87.39	92.09
420	95.29	91.89	89.29	93.39
480	95.59	92.39	90.19	93.59
540	96.04	91.19	89.09	94.29
600	94.19	90.59	88.04	92.09

Table 1 illustrates the figurative representation of requirement prioritization accuracy for three different methods, PTF-GRL, JRD-SCRUM [1], IFS [2] and SRPTackle [28] respectively. The Table 1shows the results for 600 distinct functional and non-functional requirements acquired at different time instances between different stakeholders. It can be noticed that increasing the number of requirements does not have any influence on the accuracy rate for all the three methods. However, with '60' requirements acquired from two different stakeholders involved in simulation and '58' requirements correctly prioritized using PTF-GRL, '56' prioritized correctly using JRD-SCRUM [1] and '55' prioritized correctly using IFS [2], the requirement prioritization accuracy was found to be '96.66', '93.33' and '91.66'. '94.66' respectively. Likewise, nine various performance results are observed. From these results, the requirement accuracy is said to be better using PTF-GRL when compared to [1], [2] and [28].

The reason behind the improvement in requirement prioritization accuracy is the application of the Pugh Decision and Trapezoidal Fuzzy Requirement Selection algorithm. By applying this algorithm, the lesser extent of possibility via Pugh Decision Matrix is employed for accurately selecting the requirements in a computationally efficient manner. In addition, the Pugh Decision Matrices are employed to carry out the consistency checking among pair-wise matrices. De-fuzzification is used

to reduce the uncertainty among stakeholders. This in turn improves the accuracy of requirement prioritization using PTF-GRL by 4% compared to [1], 7% compared to [2] and 3% compared to [28].

4.2 Performance analysis of requirement prioritization time

During the process of prioritization of requirements involving large scale software projects, time is the major criterion of concern. It is said that a great deal of time is consumed during the process of prioritizing requirements. Therefore, any requirement prioritization method necessitates time. This is mathematically expressed as given below.

$$RPT = \sum_{i=1}^{n} RP_i * Time\,(RP) \tag{11}$$

From equation (11), the requirement prioritization time 'RPT', is estimated on the basis of the requirements involved in large scale software projects 'RP_i' and the corresponding time consumption in requirement prioritization '$Time\,(RP)$'.It is measured in terms of milliseconds (ms). The measure of requirement prioritization time has been tabulated in Figure 5. The measure of minimum requirement prioritization time indicates that the proposed PTF-GRL method is more effective than [1], [2] and [28].

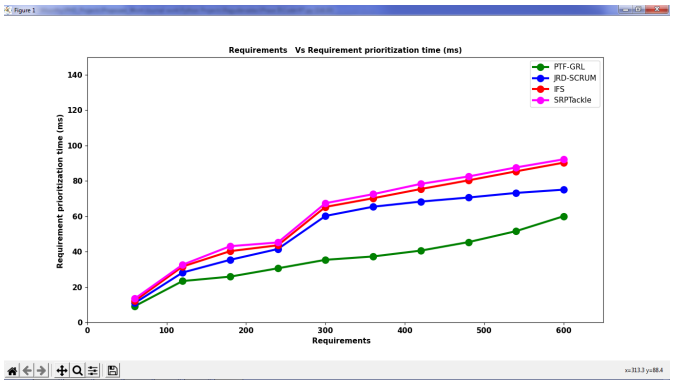

Fig 5. Requirement prioritization time

Figure 5 illustrates the figurative representation of requirement prioritization time using three different methods, PTF-GRL, JRD-SCRUM [1], IFS [2] and SRPTackle [28] respectively with respect to 600 different requirements involving both functional and non-functional types. A linear increase is found in the requirement prioritization time with the increase in the number of requirements provided as input during the simulation. This is because with the higher number of requirements to be prioritized, the larger number of requirements for different types of software products is formed and that results in a rise in the requirement prioritization time also. However, with simulations conducted for requirement prioritization time, the time consumed for a single requirement being '0.155ms' using PTF-GRL, '0.185ms' using $JRD - SCRUM$ [1], and '0.205ms' using IFS [2], and0.225ms using SRPTackle [28] the overall requirement prioritization time was observed to be '9.3 ms', ' 11.1 ms',12.3 ms' and '13.55' respectively. Similarly, the remaining nine runs are executed with different counts of requirements. The obtained performance of requirement prioritization time of the (PTF-GRL) method is compared to existing methods.

From the results, it is inferred that the requirement prioritization time is lesser using PTF-GRL than compared to [1,2,28]. The reason behind the improvement is due to the application of Trapezoidal Fuzzy Numbers applied to the Pugh Decision Matrix. The sufficient requirements are precisely selected by using Pugh Decision Matrix with minimum time even in case of uncertainty between stakeholders. With this, the requirement prioritization time using PTF-GRL is said to be reduced by 30% compared to [1], 37% compared to [2] and 40% compared to [28].

4.3 Performance analysis of precision

Precision is measured as the fraction of relevant requirements correctly and incorrectly prioritized among the number of requirements taken for experimental evaluation. The precision is mathematically estimated as given below,

$$P = \left[\frac{t_p}{t_p + f_p}\right] \tag{12}$$

From equation (12), the precision 'P' is measured on the basis of the true positive 't_p' (i.e., correctly prioritized) and false positive 'f_p' (i.e., incorrectly prioritized). The measure of precision has been tabulated in Table 2. The measure of better precision

indicates that the proposed PTF-GRL method is more efficient than [1,2,28].

Table 2. Tabulation for precision using PTF-GRL, JRD-SCRUM [1], IFS [2] and SRPTackle [28]

Require-ments	Precision (%)			
	PTF-GRL	$JRD-SCRUM$	IFS	SRPTackle
60	0.93	0.9	0.88	0.89
120	0.9	0.877	0.777	0.887
180	0.91	0.866	0.789	0.879
240	0.96	0.887	0.788	0.889
300	0.97	0.889	0.888	0.91
360	0.93	0.881	0.899	0.91
420	0.999	0.888	0.877	0.899
480	0.94	0.885	0.886	0.91
540	0.95	0.883	0.881	0.885
600	0.91	0.887	0.885	0.889

Table 2 shows the figurative portrayal of precision for three different methods, PTF-GRL, JRD-SCRUM [1], IFS [2] and SRPTackle [28] respectively. It is inferred that the precision rate is not found to be inversely or directly proportional to the requirement set as provided by two different stakeholders. This is due to the reason that the requirements utilized in our work consist of both functional and non-functional requirements and also both the negative and positive aspects involved in the non-functional requirements. Due to this inconsistency of precision between stakeholders for distinct requirements are said to persist. Nevertheless, simulation results show a true positive rate of 56, 54, and 53 and a similarly false positive rate of 4, 6, and 7 using three different methods respectively. For experimentation, the precision of PTF-GRL is '0.93 %', and the precision of JRD-SCRUM [1], IFS [2] and SRPTackle [28] being '0.9%', 0.88% and 0.89%.In the same way, a diversity of results is observed and the results are compared. The overall results show that the PTF-GRL improves the precision by 6% compared to [1] , 10% compared to [2] and 5% compared to [28] respectively .With these results, the incorrect requirement prioritization using PTF-GRL is found to be lesser than [1,2,28].

The reason behind improved precision value is due to utilization of Pugh Decision Matrices for each requirement extent value. Consistency checking between pair-wise matrices (i.e., between stakeholders) is also performed by de-fuzzifying the Pugh Decision Matrices for minimizing uncertainty involved between stakeholders. Only after this, the prioritization process for requirements is performed, the Gradient Orientation function, flexibly obtains more perceptions into individual test cases or test samples. With this function, according to distinct requirements of the stakeholders, correct prioritization is ensured using PTF-GRL.

4.4 Performance analysis of test suite execution

Test suite execution refers to the number of fault identification involved in prioritizing the requirements concerning large scale software projects. In other words, the minimum the test suite execution, the higher is the chance of obtaining the failure cases involved in prioritization and therefore improving the overall performance. A test suite refers to a collection of test samples (i.e., requirements) intended to test a set of behaviors (i.e., failing test instances) of software programs (i.e., large scale software projects). This is mathematically expressed as given.

$$\mathrm{TSE} = \sum_{i=1}^{n} \frac{\mathrm{Pr}^{F}}{\mathrm{R_i}} * 100 \tag{13}$$

From the equation (13), the test suite execution 'TSE' is measured based on the requirements that intends to be actual failing test instances 'R_i' and the probable failing test instances measured 'Pr^F'. It is measured in terms of percentage (%). The measure of test suite execution has been shown in Figure 6. Measure of higher test suite execution indicates that the proposed PTF-GRL method is more efficient than [1,2,28].

Finally, Figure 6isthe graphical plot of test suite execution against the requirements. As revealed in Figure 6, the test suite execution of four different methods PTF-GRL, JRD-SCRUM [1], IFS [2] and SRPTackle [28] are represented by three different colors namely blue, red, and green,pink color respectively. The requirements are taken in the horizontal direction whereas the performance of test suite execution is observed on the vertical axis. A test suite comprises a set of test cases that are grouped

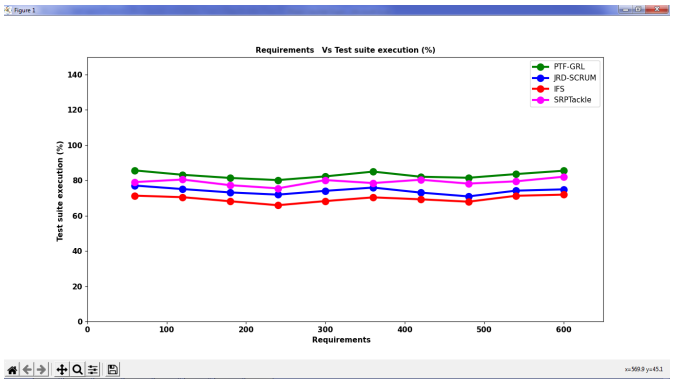

Fig 6. Graphical representation of test suite execution

for test execution purposes. In our work, the test suite consists of a set of requirements provided by different stakeholders for requirement prioritization. Also, the test plan assists in organizing test suite execution, and hence by including both the test samples and the test suite in our work, both the test suite execution and testing activities for requirement prioritization are said to be performed parallel. Therefore, increasing the number of requirements does not influence the test suite execution.

Moreover, by employing the Gradient Orientation function that flexibly procures more perceptions into individual test cases or test samples, large number of test suite execution are said to be ensured. Followed by, Deep Reinforce learning is used to focus the failing test samples than the passing test samples. Therefore, the test suite execution is obtained in an early manner. As a result, the test suite execution is improved by using PTF-GRL by 12% compared to [(1)], 19% compared to [(2)] and 5% compared to [(28)] respectively.

5 Conclusion

One of the most critical phases of requirement engineering is requirement prioritization owing to the fact that several executions have been employed for explicit requirement elicitation. However, none of the methods can be said to be efficient as each method possess their advantages and drawbacks. In this study, Pugh Trapezoidal Fuzzy and Gradient Reinforce Learning (PTF-GRL) method are proposed for large scale software requirement prioritization. The PTF-GRL method involved all the stakeholders and worked for any number of requirements, therefore contributing to both accuracy and time. Moreover, by utilizing Gradient Orientation-based Reinforce Requirement Prioritization more perceptions into individual test cases can be modeled therefore ensuring test suite execution. Experiments are performed to compare our proposed PTF-GRL method with the state-of-the-art methods. The results of this study verify the significance of our method to meet the emergent requirements in software requirement prioritization. Simulations results showed an improvement of 4%, 7% and 3% accuracy upon comparison with JRD-SCRUM, IFS and SRPTackle, therefore reducing the requirement prioritization time by 30% compared to JRD-SCRUM, 37% compared to IFS and 40% compared to SRPTackle. Moreover, the test suite execution was improved by 12% compared to JRD-SCRUM, 19% compared to IFS and 5% compared to SRPTackle respectively. The proposed work failed to reduce both the costs and duration of a project for addressing the high priority requirements.

References

1) Saeeda H, Dong J, Wang Y, Abid MA. A proposed framework for improved software requirements elicitation process in SCRUM: Implementation by a real-life Norway-based IT project. *Journal of Software: Evolution and Process*. 2020;32(7). Available from: https://dx.doi.org/10.1002/smr.2247.
2) Gupta A, Gupta C. A novel collaborative requirement prioritization approach to handle priority vagueness and inter-relationships. *Journal of King Saud University - Computer and Information Sciences*. 2019. Available from: https://dx.doi.org/10.1016/j.jksuci.2019.12.002.
3) Software Requirement Dataset . 2021. Available from: https://www.kaggle.com/iamsouvik/software-requirements-dataset.
4) Thapar SS, Sarangal H. Quantifying reusability of software components using hybrid fuzzy analytical hierarchy process (FAHP)-Metrics approach. *Applied Soft Computing*. 2020;88:105997–105997. Available from: https://dx.doi.org/10.1016/j.asoc.2019.105997.
5) Seyff N, Todoran I, Caluser K, Singer L, Glinz M. Using popular social network sites to support requirements elicitation, prioritization and negotiation. *Journal of Internet Services and Applications*. 2015;6(1):1–16. Available from: https://dx.doi.org/10.1186/s13174-015-0021-9.
6) Ali S, Hafeez Y, Asghar S, Nawaz A, Saeed S. Aspect-based requirements mining technique to improve prioritisation process: multi-stakeholder perspective. *IET Software*. 2020;14(5):482–492. Available from: https://dx.doi.org/10.1049/iet-sen.2019.0332.
7) Zare F, Khademizare H. Software effort estimation based on the optimal Bayesian belief network. *Applied Soft Computing*. 2016;49:968–980. doi:10.1016/j.asoc.2017.10.047.

8) Hernández-González J, Rodriguez D, Inza I, Harrison R, Lozano JA. Learning to classify software defects from crowds: A novel approach. *Applied Soft Computing*. 2018;62:579–591. Available from: https://dx.doi.org/10.1016/j.asoc.2017.10.047.
9) Abualhaija S, Arora C, Sabetzadeh M, Briand LC, Traynor M. Automated demarcation of requirements in textual specifications: a machine learning-based approach. *Empirical Software Engineering*. 2020;25(6):5454–5497. Available from: https://dx.doi.org/10.1007/s10664-020-09864-1.
10) Dingsøyr T, Moe NB, Fægri TE, Seim EA. Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. *Empirical Software Engineering*. 2018;23(1):490–520. Available from: https://dx.doi.org/10.1007/s10664-017-9524-2.
11) Shinkuma R, Nishio T. Data Assessment and Prioritization in Mobile Networks for Real-Time Prediction of Spatial Information with Machine Learning. *2019 IEEE First International Workshop on Network Meets Intelligent Computations (NMIC)*. 2019;92:1–19. doi:10.1016/j.asoc.2016.07.040.
12) Xue Y, Zhong J, Tan TH, Liu Y, Cai W, Chen M, et al. IBED: Combining IBEA and DE for optimal feature selection in software product line engineering. *Applied Soft Computing*. 2016;49:1215–1231. Available from: https://dx.doi.org/10.1016/j.asoc.2016.07.040.
13) Hujainah F, Bakar RBA, Abdulgabber MA, Zamli KZ. Software Requirements Prioritisation: A Systematic Literature Review on Significance, Stakeholders, Techniques and Challenges. *IEEE Access*. 2018;6:71497–71523. Available from: https://dx.doi.org/10.1109/access.2018.2881755.
14) Sahin B. Consistency control and expert consistency prioritization for FFTA by using extent analysis method of trapezoidal FAHP. *Applied Soft Computing*. 2017;56:46–54. doi:10.1007/s10664-016-9491-z.
15) Heikkilä VT, Paasivaara M, Lasssenius C, Damian D, Engblom C. Managing the requirements flow from strategy to release in large-scale agile development: a case study at Ericsson. *Empirical Software Engineering*. 2017;22(6):2892–2936. Available from: https://dx.doi.org/10.1007/s10664-016-9491-z.
16) Shameem M, Kumar RR, Nadeem M, Khan AA. Taxonomical classification of barriers for scaling agile methods in global software development environment using fuzzy analytic hierarchy process. *Applied Soft Computing*. 2020;90:106122–106122. Available from: https://dx.doi.org/10.1016/j.asoc.2020.106122.
17) Daneva M, van der Veen E, Amrit C, Ghaisas S, Sikkel K, Kumar R, et al. Agile requirements prioritization in large-scale outsourced system projects: An empirical study. *Journal of Systems and Software*. 2013;86(5):1333–1353. Available from: https://dx.doi.org/10.1016/j.jss.2012.12.046.
18) Barbosa PAM, Pinheiro PR, Silveira FRV, Filho MS. Selection and Prioritization of Software Requirements Applying Verbal Decision Analysis. *Complexity*. 2019;2019:1–20. Available from: https://dx.doi.org/10.1155/2019/2306213.
19) Dabbagh M, Lee SP. An Approach for Integrating the Prioritization of Functional and Nonfunctional Requirements. *The Scientific World Journal*. 2014;2014:1–13. Available from: https://dx.doi.org/10.1155/2014/737626.
20) Jamilah D, Michael I, Jasser MB, and. Software Requirements Prioritization Tool using a Hybrid Technique. *International Journal of Engineering and Advanced Technology*. 2019;9(1):1631–1635. Available from: https://dx.doi.org/10.35940/ijeat.a2634.109119.
21) Sarker KU. Explicit specification framework to manage software project effectively. *Indian Journal of Science and Technology*. 2020;13(36):3785–3800. doi:17485/IJST/v13i36.1244.
22) Tompkins M, Iammartino R, Fossaceca J. Multiattribute Framework for Requirements Elicitation in Phased Array Radar Systems. *IEEE Transactions on Engineering Management*. 2020;67(2):347–364. Available from: https://dx.doi.org/10.1109/tem.2018.2878688.
23) Zhou ZQ, Liu C, Chen TY, Tse TH, Susilo W. Beating Random Test Case Prioritization. *IEEE Transactions on Reliability*. 2021;70(2):654–675. Available from: https://dx.doi.org/10.1109/tr.2020.2979815.
24) Bajaj A, Sangwan OP. Test Case Prioritization Using Bat Algorithm. *Recent Advances in Computer Science and Communications*. 2021;14(2):593–598. Available from: https://dx.doi.org/10.2174/2213275912666190226154344.
25) Mughal S, Abbas A, Ahmad N, Khan SU. A Social Network Based Process to Minimize In-Group Biasedness During Requirement Engineering. *IEEE Access*. 2018;6:66870–66885. Available from: https://dx.doi.org/10.1109/access.2018.2879385.
26) Kamal T, Zhang Q, Akbar MA, Shafiq M, Gumaei A, Alsanad A. Identification and Prioritization of Agile Requirements Change Management Success Factors in the Domain of Global Software Development. *IEEE Access*. 2020;8:44714–44726. Available from: https://dx.doi.org/10.1109/access.2020.2976723.
27) AbdElazim K, Moawad R, Elfakharany E. A Framework for Requirements Prioritization Process in Agile Software Development. *Journal of Physics: Conference Series*. 2020;1454(1):012001–012001. Available from: https://dx.doi.org/10.1088/1742-6596/1454/1/012001.
28) Hujainah F, Bakar RBA, Nasser AB, Al-haimi B, Zamli KZ. SRPTackle: A semi-automated requirements prioritisation technique for scalable requirements of software system projects. *Information and Software Technology*. 2021;131:106501–106501. Available from: https://dx.doi.org/10.1016/j.infsof.2020.106501.
29) Khan AA, Shameem M, Kumar RR, Hussain S, Yan X. Fuzzy AHP based prioritization and taxonomy of software process improvement success factors in global software development. *Applied Soft Computing*. 2019;83:105648–105648. Available from: https://dx.doi.org/10.1016/j.asoc.2019.105648.
30) Shameem M, Kumar RR, Kumar C, Chandra B, Khan AA. Prioritizing challenges of agile process in distributed software development environment using analytic hierarchy process. *Journal of Software: Evolution and Process*. 2018;30(11):e1979–e1979. Available from: https://dx.doi.org/10.1002/smr.1979.
31) Shameem M, Khan AA, Hasan MG, Akbar MA. Analytic Hierarchy Process Based Prioritisation and Taxonomy of Success Factors for Scaling Agile Methods in Global Software Development. *IET Software*. 2020;14(4):389–401. Available from: https://dx.doi.org/10.1049/iet-sen.2019.0196.
32) Khan AA, Shameem M, Nadeem M, Akbar MA. Agile trends in Chinese global software development industry: Fuzzy AHP based conceptual mapping. *Applied Soft Computing*. 2021;102(3):107090–107090. Available from: https://dx.doi.org/10.1016/j.asoc.2021.107090.
33) Devadas R, Cholli N. Interdependency aware QUBIT and BROWNBOOST rank for large scale requirement prioritization. *International Journal of Computing and Digital Systems*. 2022;11:625–634. Available from: http://dx.doi.org/10.12785/ijcds/110150.