Journal of Artificial Intelligence and Computational Technology

Journal homepage: https://ojs.omgfzc.com/index.php/JAICT

Machine Learning Approach Integrated with MoSCoW Method for Parallel Requirements Prioritization

Kawthar Ishag Ali Fadlallah[1*], Moawia Elfaki Yahia Eldow[2], Anwer Mustafa Hilal[3], Khalid Mohammed Osman Saeed[4], Mohammed Mohammed Osman Mokhtar[5]

[1]Faulty of Computer Science &Information Technology, Department of Computer Science, Omdurman Islamic University, Omdurman, Sudan

[2]Faculty of Mathematical Science, University of Khartoum, Khartoum 11115, Sudan, University of North Texas, Denton, TX, USA

[3]Department of Information System, Omdurman Islamic University, Omdurman, Sudan

[4]Department of Information System, Omdurman Islamic University, Omdurman, Sudan

[5]Department of Information System, Omdurman Islamic University, Omdurman, Sudan

*Correspondence: E-mail: kawthar1140@gmail.com

Article Info

Article History:
Submitted/Received 11-Jan 2025
Revised in revised format 15 Feb2025
Accepted 02 Mar 2025
Available 02-Mar 2025
Publication Date 01 Apr 2025

Keyword:
Requirements Prioritization,
Evaluation,
Implementation time,
Accuracy

Cite this article: *Fadlallah, K. I. A., & Eldow, M. E. Y. (2025). Machine Learning Approach Integrated with MoSCoW Method for Parallel Requirements Prioritization. Journal of Artificial Intelligence and Computational Technology, 1(1).*

ABSTRACT

Requirements prioritization (RP) is one of the vital activities carried out through requirements engineering process. Requirements prioritization includes the selection of requirements that are reflected more important from elicited list of stakeholders' requirements. Making an incorrect selection will not only reduction the quality of the developed software but it will also earn extra cost for refinement processes in later stages. Thus, requirements prioritization would aid to determine the most appropriate requirements in different software product releases. Many research focusing on prioritizing the requirements using one or several criteria like time, dependency, and scalability. However, most of these studies address sequential prioritization only. To the best of our knowledge, no research has explored parallel ranking in prioritization, which allows for simultaneous requirements implementation, thereby reducing implementation time. Furthermore, as the volume of requirements grows, scalability becomes a critical issue. Manual prioritization is time-consuming and increases the likelihood of overlooking essential. Machine learning is increasingly popular for automating requirements prioritization. In this study we developed automated parallel requirements prioritization approach (APRP) for determine the requirements priority level in parallel format using Random Forest classifier based MoSCoW method (RF-MM). The proposed approach consists of two main modules, data elicitation and pre-processing module and prioritization module, which include established weight assignment, MoSCoW parallel prioritization, and classifier methods. Experiments on the industrial dataset (Testcase MIS system with priority) revealed that the total implementation time for sequential ranking was 76.0 seconds, whereas it was reduced to 33.0 seconds for parallel ranking. Thus, parallel ranking reduced implementation time by more than half. We achieved a maximum accuracy of 94.87%, precision of 92.31%, and recall of 92.31%.

1. Introduction

Requirement prioritization (RP) is one of the vital activities carried out through requirements engineering process. Requirement prioritization includes the selection of requirements that are reflected more important from a collected list of stakeholders' requirements. Making an incorrect selection will not only reduction the quality of the developed software but it will also earn extra cost for refinement processes in later stages. Thus, requirements prioritization would aid to determine the most appropriate requirements in different software product releases.

The concept RP appeared with the increased requests of complex software systems by stakeholders (Aurum, 2005), and due to huge numbers of software requirements. Hence, the issue of scalability in prioritizing the requirements is become essential. Scalability denotes the capability to handle a large number of requirements i.e. more than 100 (Lunarejo, 1021). Accordingly, because the requirements amount is huge and manual prioritization is considered time-consuming, and the possibility of missing necessary requirement is high. Therefore, computer-aided prioritizing requirements is required, as it can help the software engineer to proficiently detect the rank of each requirement (Achimugu et al., 2014; Shao et al., 2017). Machine learning is becoming more popular as it assists in providing an automatic solution in the area of requirements prioritization. The focus of this research is to provide efficient automated parallel requirements prioritization approach(APRP) with high performance. This work is an extension of our previous study (Fadlallah et al., 2024), and therefore extends this previously completed work by providing an experimental evaluation for priority determination and parallel ranking these requirements to provide more appropriate and usable systems through providing graphical user interface (GUI).

Several machine learning techniques for RP have been presented in the literature such as (Shao et al., 2017; Gupta & Gupta., 20١٨; Chua et al., 2022). Any technique uses one or many criteria in prioritizing requirements. However, all techniques have limitations, not only associated with scalability and requirements dependencies (Achimugu et al., 2014; Shao et al., 2017), but they consider mostly the sequential prioritization format. Also due to ignorance on the impact of the requirements prioritization on reducing software implementation time. According to (Fadlallah et al., 2024) there is no work focused on the impact on requirements prioritization on the software implementation time.

Our proposed approach reduces the software implementation time, due to the awareness of parallel ranking process in RP. It concerns the determining and categorizing the requirements priority levels into in parallel format which permit the batch implementation using Random Forest classifier based MoSCoW method. We apply our approach to the Testcase MIS system with priority dataset.

We develop a machine learning approach integrated with MoSCoW Method, that determine the requirements priority level in parallel format from an input Testcase MIS system with priority dataset after putting the values in an appropriate pattern, then removes null values and applies rules to determine the parallel priority level.

Our study aims to develop a machine learning approach relied on the software requirements document to determine the proper priority level by determining the MoSCoW categories (Must, Should, Could, and Wouldn't) as priority levels (High, Medium, and low) in parallel format. Parallel format, which reduce the implementation time through simultaneous requirements implementation. The total implementation time we achieved were equal to 33 seconds in parallel ranking formant; whereas The total implementation time were equal to 76.0 seconds when ranking in sequential format. We achieved a maximum measure of 94.87%, 92.31%, and 92.31% of average accuracy, precision and recall, respectively.

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

2. Related Works

Machine learning techniques have been deeply used to develop requirements prioritization approaches (Qayyum, & Qureshi, 2018; Jan et al., 2020; Bukhsh et al., 20٢٠). These approaches were believed very effective due to the automation and tools supporting their implementation. Machine learning and data mining algorithms has the capability to automation, resolve the conflicts between the stakeholders and developers (Hujainah et al., 2018; Sher et al., 20٢٠), etc., high level of handling the scalability problem (Hudaib et al., 2018; Hujainah et al., 2018; Sher et al., 20٢٠; Amelia& Mohamed 2022). We reviewed and documented the previous research in this field in (Fadlallah, & Eldow, 2024). Few existing semi-automated and full-automated approaches and models for requirements prioritization are being studied in this section.

(Shao et al., 2017) developed a semi-automatic requirements prioritization approach, called DRank. DRank takes the dependencies among requirements and the stakeholders' preferences into consideration. DRank employs RankBoost algorithm to produce requirement prioritization formula in subjective manner. DRank producing requirement dependency graphs (RDGs) based on the contribution dependencies and business dependencies among the requirements, next analyze the contribution order to compute the contribution of each requirement by adopting PageRank algorithm to finally assimilate the final requirements prioritization. A controlled experiment made to validate the DRank efficiency constructed on comparisons with Case Based Ranking, AHP, and EVOLVE. The results provided more effectiveness as compared with alternative approaches. However, this work seems to provide only the contribution and business dependencies. Further, the authors highlight that their approach is still motivated to the issue of subjectivity especially in the process of requirements evaluation.

(Gupta & Gupta., 20١٨) proposed a semi-automated dependency based collaborative requirement prioritization approach termed (CDBR), which uses an execute-before-after (EBA) connection among requirements, linguistic values, and a machine learning algorithm to reduce inconsistencies in views among stakeholder and developer and improve final priority estimation. The CDBR focused on three major problems which are usually overlooked in an existing research: dependencies between requirements, stakeholder and developer's collaboration and scalability. To obtain final agreeable implementation priorities, CDBR uses the Particle Swarm Optimization (PSO) algorithm to lessen disputes among stakeholders and developers' ranking. To evaluate the approach's performance tow scenario are chosen: in the first one, nine different requirement sets are applied to evaluate the suggested approach's performance in terms of managing scalability. stakeholders and developer's priority for all these requirements is determined randomly. The developer's priority is computed using a dependency matrix that is similarly created randomly while keeping the density of the matrix in mind. The higher the density, the further dependencies there are in the system, and hence the more complex it is. In the second scenario, validation on the case study of cargo booking management in a warehouse (CBMW) is chosen, the CDBR, interactive genetic algorithm (IGA), and Analytic hierarchy process (AHP) are used. The precision of the results is identified by comparing the CDBR against AHP and IGA priority lists using the Analysis of Variance (ANOVA) test method. The outcomes are accurate and equivalent in terms of scalability, accuracy, and stakeholder and developer variances levels. In terms of efficiency and processing time, CDBR beats AHP and IGA. Despite an improvements detected in prioritization results, and processing time together. However, it ignores the errors (false positive rate) implicated during the prioritization process and the absent of supporting tool is also seen.

(Hujainah et al., 2021) proposed a new semi-automated scalable prioritization model named, SRPTackle, and automation implementation tool (SRPTackle-Tool). SRPTackle consists of four steps:1) specification of stakeholder priority value (SPV) for each stakeholder. 2) preparation requirement priority value (RPV) for each requirement utilizing weighted sum model (WSM); 3) produce prioritised list of requirements using K-means and K-means++ algorithms. 4) implement the binary search tree (BST) algorithm SRPTackle developed to handle the main challenges of the prioritization process such as scalability, time consumption, restricted dependence on expert participation, and lack of automation.

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

The efficiency of SRPTackle is measured through established seven experiments using the RALIC benchmark dataset of a large actual software project. Experiment outcomes expose that SRPTackle able to get 93.0% and 94.65% as least and high accuracy percentages, respectively. The outcomes also highlight the ability of SRPTackle to prioritize large-scale requirements with minimum computation time, and its increase efficiency when compared with other techniques. However, the dependency between requirements is negated.

(Chua et al., 2022) introduced semi-automated framework for requirements prioritization named (SARiP), which aimed to automate the activities in software requirements prioritization (SRP) process. The proposed SARiP emphases on the parts related to prediction of requirements priority group and ranks requirements using classification tree and ranking algorithm. The SARiP framework initiate with data pre-processing and analysis of elicited data. Then a requirements list will be supplied. Consequently, the requirements list will be prioritized using the SARiP framework. The implementation of SARiP framework contains two prioritization phases manual and automatic. In the manual phase, the requirements prioritized manually by the project team and stakeholders The manual process using MoSCoW technique, numerical assignment technique and Kano model. The output of the manual phase represents an initial prioritization list, which used as an input for automatic prioritization. The automatic phase ranks the requirements using classification tree and ranking algorithm. Finally, the SARiP framework has been well evaluated in the government sector as case study. However, the authors state that the SARiP does not store the requirements prioritization results in the database. Additionally, the traceability to trace the requirements changes not considered. Further, it has limitations regarding the subjective use of ordinal scales and rankings, and it ignore the dependency among requirements.

(Devadas & Cholli., 2022) presented a novel method called the Interdependency-aware Qubit and BrownRoost Rank (IQ-BR) method to prioritize the large set of requirements and interdependency among them. IQ-BR begin with the selected functional and non-functional requirements that are required to be prioritized, using the Interdependencyaware Qubit requirement selection model. Each customer (or stakeholder) offers a score for each requirement and a rank is made established on the weighting of that customer. Interdependencies are acquired using the Eels function. Finally, the BrownBoost Rank prioritization learning model is then adopted to rank the selected requirements. Fairly good result had been achieved in precisely prioritizing requirements and reducing the noise in a large set requirement prioritization. Performance analysis comparing IQ-BR, Intuitionistic Fuzzy Approach (IFS) and Dependency Based Collaborative Requirement (CDBR) and establish that the result for IQ-BR outperform CDBR, and IFS in term of accuracy which was detected to be at between 93.15% and 95%, 90%, and 91.66% respectively. However, it fails to address uncertainty and test suite execution issue among different stakeholders for large scale software requirements prioritization and the absent of supporting tool is also seen.

We developed a classification approach concerning the parallel ranking in prioritization, which permit the simultaneous requirements implementation that reducing the implementation time. We aim to gain high rate of reducing the implementation time, high level of accuracy and other factors represented in determining priority level (high, Medium, and low), such as scalability, automation and ease of use.

3. Proposed approach

The proposed automated prioritization approach consists of two main modules, data elicitation and pre-processing module and prioritization module, which include established weight assignment, MoSCoW prioritization, MoSCoW parallel prioritization, classifier methods.
Our approach begins with first module, data elicitation and preprocessing, followed by prioritization module, which contains established priority assignment by the stakeholders and developer for each requirement. Followed by applying the MoSCoW method and MoSCoW parallel prioritization; Finally, we will apply a classification technique called Random Forest classifier based MoSCoW method.

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

Figure 1 shows a general diagram of the constructed approach. The work concerning each part is explained in detail in the following sections.

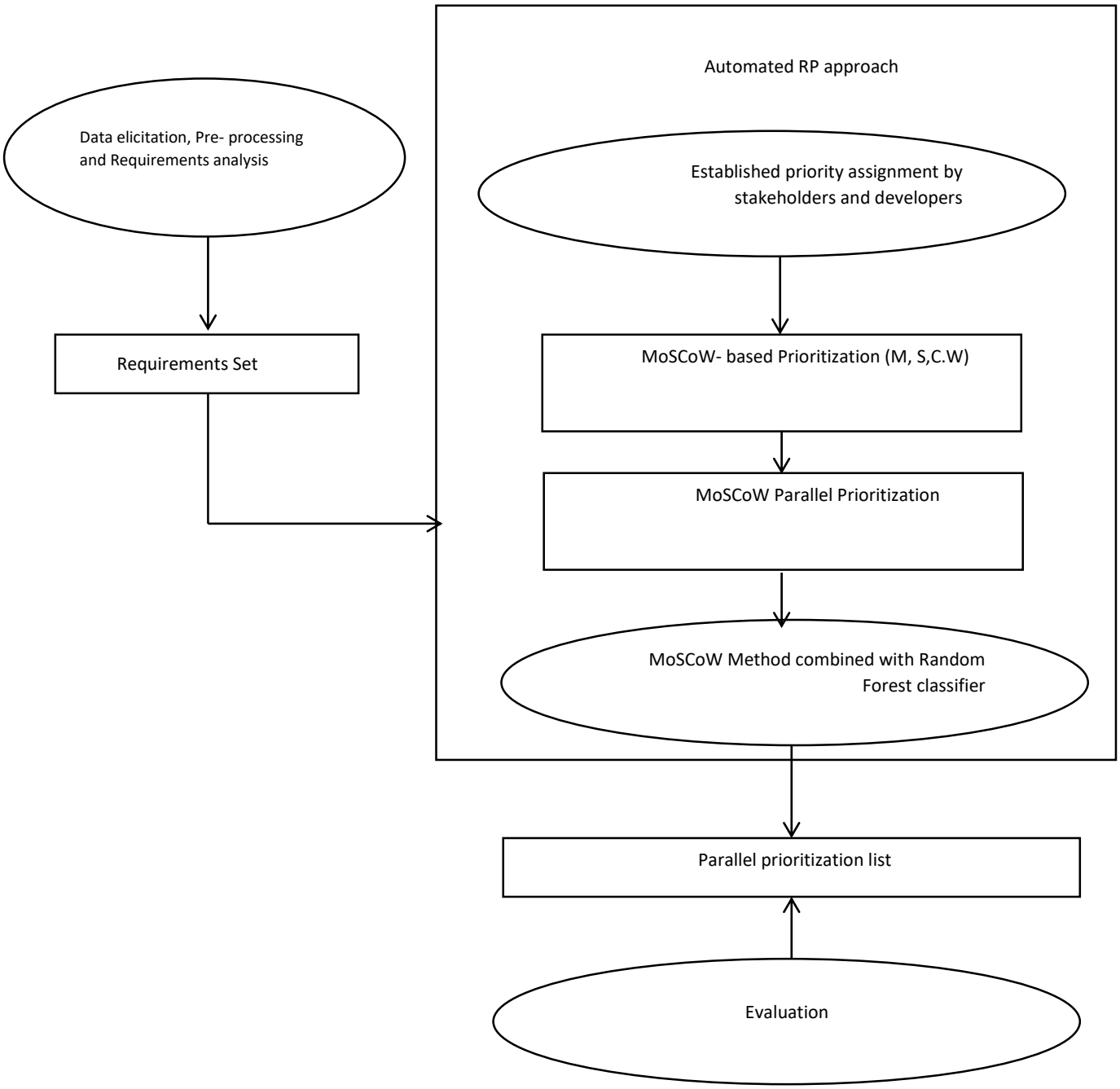

Figure 1. A general diagram of the prioritization approach.

3.1Elcitation and Pre-processing module

This module begins with data gaining and pre-processing, followed by requirements analysis. When both stages are completed, a requirements set will be issued. This stage involves removing the noisy

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

data of the requirements features, handling the missing values of the requirements features in order to attain optimal classification results. Requirements analysis is conducted to ensure that the produced requirements shall be atomic, uniquely identified, complete, consistent and unambiguous. Requirements analysis step is significant to guarantee all the requirements are clear, clean and ready for the prioritization process. This module also contains dependency analysis, and criteria assignment.

3.2 Prioritization module

In this research, we developed a machine learning approach based on an integration of the MoSCoW method and Random Forest classifier for automated parallel requirement prioritization(APRP). The proposed approach improves and enhances the existing approaches in order to overcome the limitations of existing approaches. The prioritization module comprises: Established priority assignment by the stakeholders and developer, MoSCoW prioritization, MoSCoW parallel prioritization .and classification technique. We applied our approach to (Testcase MIS system with priority) industrial dataset.

3.2.1 Established priority assignment by the stakeholders and developer

Stakeholders indicate ones who will describe the look of organization and will document the requirements (Gupta & Gupta., 20١٨). Stakeholders' emphasis customer satisfaction in term of urgency, needs, and business values (Keertipati et al., 2016; Gupta & Gupta., 20١٨; Gambo et al., 2021); Whereas developer denotes engineers who develop or maintain related systems (Ahmad et al., 2022). Developers are concerned with project features such as effort and cost (Keertipati et al., 2016; (Devadas & Cholli., 2022). Stakeholders and developer's collaboration has become an essential tool assists in bridging the gap of understanding and allocating a weight for each prioritization criteria. Therefore, stakeholders and developer's asked to pick a weight for various prioritization criteria such as MoSCoW, time, cost, complexity, and importance. Stakeholders and developer's collaboration in RP lead to success software system, since it minimizes the difference of view between stakeholder and developers for better estimation of priority acceptable to both. Hence, a refined priority weight for each requirement can be obtained. The proposed prioritization criteria and their description are represented in Table 1. The total weight will select by stakeholders for all requirements criteria will be 100 percent. Table 2 shows an example of criteria weight for the proposed requirements prioritization approach.

Table 1: Prioritization criteria and their description

Criteria	**Description**
MoSCoW	Measure the impact of requirements using MoSCoW Method
Cost	Measure the cost required in developing the application
Time	Measure the time required to develop application
Complexity	Measure the requirements complexity
Importance	Measure the importance of requirements.

Table 2: Example of criteria weight for requirements prioritization

Criteria	**Weight %**
MoSCoW	10%
Cost	15%
Time	20%
Complexity	25%
Importance	30%
Total	100%

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

Developers implement matrix based on dependence graph to represent dependency among requirements. The stakeholders and developer's initial priority are considered as ''perfect" preferences. These preferences will use as guide for estimation in MoSCoW ranking.

3.2.2 MoSCoW based Prioritization

Within the stakeholder and developer's initial ranking in mind, MoSCoW method will progress to identify and classify the requirements accordingly. Moscow characterises a guideline to confirm the basic importance of the requirements. The key stakeholders distribute the requirements into four categories (Must, Should, Could and Wouldn't) and three level (High, Medium, and Low) by applying MoSCoW method. MoSCOW categories represented in Table 3.

Table 3: Rating criteria for MoSCoW Method categories

MoSCoW category	**Meaning**
Must-Have (M)	Identify the requirements that are absolutely critical for the success of project.
Should-Have (S)	List the requirements that are important but not as critical as the Must-Have items.
Could-Have (C)	Include requirements that are desirable but not essential for the project's immediate success.
Won't-Have (W)	This category includes requirements that are explicitly excluded from the current project scope.

The rating method for MoSCoW method is depending on the criteria (e.g: 1 - the maximum important priority, 4 - the minimum important priority) indicated in Table 4.

Table 4: Rating criteria for MoSCoW Method

Criteria	**Rate**
Must	1
Should	2
Could	3
Won't	4

The stakeholders and developers will then use numerical assignment techniques to score the rating for further prioritization criteria as defined in (Chua et al., 2022), table 4 shows rating criteria. The rating method for these criteria is based on the criteria in table 5 with the scale of 1 to 10 (e.g: 1 - the fewer important priority, 10 - the greatly important priority). The rating method for importance criteria is based on the criteria (e.g:3 - the very important priority, 1 - the less important priority) (Chua et al., 2022), in Table 6.

Table 5.: Rating criteria for cost, time, and complexity

Criteria	Rating
Cost	1-10
Time	1-10
Complexity	1-10

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

Table 6: Rating criteria for importance

Criteria	Rating
High	3
Medium	2
Low	1

The prioritization part will begin with computing the rate based on the initial set of prioritized requirements by the stakeholders and developers. The weight criteria and rating method recognized previously will be used in computing the rate for requirements prioritization. The formula for computing each criteria of the requirement are defined as equations defined in (Chua et al., 2022) (1), (2), (3), (4), and (5).

$MoSCoW \rightarrow score/4 * moscow\_weight$ (1)

$Cost \rightarrow score\ /\ 10 * cost\_weight$ (2)

$Time \rightarrow score\ /\ 10 * time\_weight$ (3)

$Complexity \rightarrow score\ /\ 10 * complexity\_weight$ (4)

$Importance \rightarrow score\ /\ 3 * importance\_weight$ (5)

The total rates for requirements criteria will be computed using equation (6) when the criteria of each requirement are calculated.

$Total\ Score \rightarrow MoSCoW + Cost + Time + Complexity + Importance$ (6)

3.2.3 MoSCoW Parallel Prioritization

Parallel prioritization concepts implemented when the requirements in the same MoSCoW category and the same priority level with no dependencies. Therefore, it could be implemented simultaneously to reduce the implementation time, which may contribute in accelerating the other higher level in software development. Then calculate implementation time: To confirm the hypothesis of simultaneous implementation of the requirements in parallel format could reduce the implementation time, the approach must calculate implementation time using mathematical equations.

To calculate the implementation time, the time must be estimated for every requirement. The estimated time allocated for each requirement in each category (Must Have, Should Have, Could Have and Wouldn't), based on required resources like complexity and cost. Then the (star_ time and finish_ time) must be estimated for each MoSCoW category as following:

In Must category(M) the star_ time==0, finsh_time ==finsh_time_m

In Should category(S) the star_ time== finsh_time_m, finsh_time== finsh_time_s

In Could category(C) the star_ time== finsh_time_s, finsh_time== finsh_time_C.

The Wouldn't category could be dropped. Calculate the total time for each Category through making summation of the estimated times for all requirements in each category (Must Have, Should Have, Could Have). The requirements within the same MoSCoW category and with the same priority levels when no dependencies can be considered for simultaneous implementation to reduce the implementation time.

3.2.4 A classification technique using Random Forest-based MoSCoW Method (RF-MM)

Using Random Forest-based on MoSCoW Method and (RF-MM) to improve the performance of MoSCoW method in the RP level determination and classification. In addition, MoSCoW method allows flexibility by categorizing requirements into different priority levels, making it easier to manage scope and resources. Random Forest can capture complex relationships between requirements and

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

prioritize them based on historical patterns. The integrated method was initiated by using 8 features given as input to MoSCoW Method and the output weight of MoSCoW optimized by Random Forest classifier. The classification process utilizes machine learning techniques in generating the priority levels and final priority ranks of requirements which contains two parts, prioritization and prediction. Specifically, Random Forest classifier -based MoSCoW Method (RF-MM) is used. The purpose of this classification approach is to predict the priority levels. The second part of the proposed RF-MM approach automatic prioritization is prediction of requirements priority level. The final priority rank will be calculated as a result represented as output of Random Forest-based MoSCoW Method (RF-MM). It uses these MoSCoW ranking for estimate of its final output concluded series of iterations. Classification done using and 70% split (i.e. 70% training, 30% testing). The training data consists of 70% of the original data (current requirements) and the other 30% of the data (new requirements) will be used as testing data.

3.3 Implementation: (The proposed Automated Parallel Requirements Proposed Approach) (APRP)

Our proposed approach contains two techniques the first one concerned with dividing the requirements into four categories according to MoSCoW method namely Must have, should have, could have and Wouldn't have. The second techniques use the output of the first techniques to determine the final priority of the requirements in the document in parallel format.

Our proposed approach uses the output of the MoSCoW parallel prioritization as input to determine the final priority of the requirements in the document in parallel format. There are several tow steps Training, Testing and Prediction illustrated as following:

Training: Train a Random Forest classifier using the Testcase MIS system with priority dataset, where the input features are the attributes of requirements, and the target variable is the priority level (High, Medium, and Low) according to MoSCoW method category and the implementation time estimation.

We built these classifiers using the Scikit-Learn library, a well-established machine learning package for Python. Scikit-Learn offers numerous utility functions for data preprocessing, validation, and metric computations. The goal of our study is to achieve the best combination of the MoSCoW method and the Random Forest classifier as supervised learning techniques for prioritizing and classifying software requirements, with an added focus on estimating implementation time. The algorithm for our approach is shown in Figure 2.

Testing and Prediction: Once our approach is trained and evaluated, I can use it to predict the priority levels of new requirements based on their attributes.

Classification Approach Algorithm
Input: the data list gained from our MoSCoW parallel prioritization
Output: Pattern as values
Input prioritization criteria values (Cost, Time, Complexity and Importance).
Find pattern as values using Regular Expressions Array
If Importance is 1 Then
Req in Must MoSCoW_category == 'M'
if priority == 'High':
m_h.allocted(req) in array
elif priority == 'Medium':
m_m. allocted(req) in array
elif priority == 'Low':
m_l. allocted(req) in array
Else IF Importance is 2 Then

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

```
Req in Should MoSCoW_category == 'S'
if priority == 'High':
  s_h. allocted(req) in array
elif priority == 'Medium':
   s_m. allocted(req) in array
elif priority == 'Low':
s_l.allocted(req) in array
Else IF Importance is 3 Then
Req in Could MoSCoW_category == 'C'
if priority == 'High':
c_h. allocted(req) in array
elif priority == 'Medium':
c_m. allocted(req) in array
elif priority == 'Low':
c_l. allocted(req) in array
Else IF Importance is 4 Then
if priority == 'High':
w_h. allocted(req) in array
elif priority == 'Medium':
w_m. allocted(req) in array
elif priority == 'Low':
w_l. allocted(req) in array
calculate the total implementation time for requirements within each MoSCoW category considering
dependencies
Drop Wouldn't category
for each MoSCoW category  (M, S, C):
if No of Req in the same MoSCoW class with the same priority level & No dependency among them:
Rank them in parallel format.
END.
```

Figure 2. Classification approach algorithm

3.4 Approach Evaluation

To approve the hypothesis that ranking requirements in parallel format can reduce implementation time, the approach calculates this time using mathematical equations. We computed the total implementation time for the "Must," "Should," and "Could" categories according to the MoSCoW method and compared the results. Figures 3 and 4 illustrates these findings.

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

id	weights	moscow	priority	time	start_time	finsh_time	cost	complexity
48	1	M	High	5	0	5	105	3
54	1	M	High	4	0	4	140	5
60	1	M	High	5	0	5	175	5
80	1	M	High	2.5	0	2.5	17.5	1
104	1	M	High	5	0	5	175	5
122	1	M	High	1	0	1	7	1
140	1	M	High	4	0	4	84	3
9	1	M	Medium	4	0	4	84	3
17	1	M	Medium	4.5	0	4.5	31.5	1
21	1	M	Medium	3	0	3	105	5
22	1	M	Medium	3	0	3	105	5
96	2	S	Low	5	5	10	35	1
114	2	S	Low	4	5	9	28	1
2	3	C	High	4	10	14	84	3
30	3	C	High	2.5	10	12.5	17.5	1
42	3	C	High	2.5	10	12.5	87.5	5
108	3	C	High	1	10	11	35	5
144	3	C	High	1	10	11	35	5
1	3	C	Medium	8	10	18	168	3
5	3	C	Medium	4	10	14	140	5
8	3	C	Medium	2.5	10	12.5	52.5	3
12	3	C	Medium	4	10	14	84	3
13	3	C	Medium	3	10	13	105	5

total Moscow: 218.75
total cost: 2392.95
total time: 115.15
total complexity: 102.5
total importance: 291.66
Time: 33.0

Figure 3. parallel ranking implementation

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

id	weights	moscow	priority	time	start_time	finsh_time	cost	complexity
48	1	M	High	5	0	5	105	3
54	1	M	High	4	0	4	140	5
60	1	M	High	5	0	5	175	5
80	1	M	High	2.5	0	2.5	17.5	1
104	1	M	High	5	0	5	175	5
122	1	M	High	1	0	1	7	1
140	1	M	High	4	0	4	84	3
9	1	M	Medium	4	0	4	84	3
17	1	M	Medium	4.5	0	4.5	31.5	1
21	1	M	Medium	3	0	3	105	5
136	2	S	Medium	2.5	5	7.5	87.5	5
139	2	S	Medium	4	5	9	140	5
145	2	S	Medium	5	5	10	175	5
149	2	S	Medium	2	5	7	14	1
96	2	S	Low	5	5	10	35	1
114	2	S	Low	4	5	9	28	1
2	3	C	High	4	10	14	84	3
30	3	C	High	2.5	10	12.5	17.5	1
42	3	C	High	2.5	10	12.5	87.5	5
108	3	C	High	1	10	11	35	5
144	3	C	High	1	10	11	35	5
1	3	C	Medium	8	10	18	168	3
5	3	C	Medium	4	10	14	140	5

total Moscow: 218.75
total cost: 2392.95
total time: 115.15
total complexity: 102.5
total importance: 291.66
Time: 76.0

Figure 4. Sequential ranking implementation

To evaluate the performance of the classification and priority determination algorithm, you should know two essential terms in any measurements, Accuracy (Acc), and Precision (P). They have a distinct meaning, as we will discuss in the subsequence lines. For our algorithm, accuracy had been calculated in expressions of positives and negatives as follows:

Accuracy= (TP+TN) / (TP+TN+FP+FN)

Where, TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives table 7 signify the confusion matrix

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

Calculating accuracy for the classification algorithm that classified requirement priority as correctly predicted requirements priority (the positive class) or falsely predicted requirements priority (the negative class):

Table 7: Confusion matrix

True Positives(TP)	False Positives(FP)
Reality: correctly determines the requirement priority level ML algorithm predicted: right priority. Number of TP result: 36	Reality: correctly determines the requirement priority level ML model predicted: the wrong priority. Number of FP result: 3
False Negative (FN)	True Negative(TN)
Reality: imprecisely determine the requirement priority level ML algorithm predicted: incorrectly recognized wrong priority. Number of FN result: 3	Reality: imprecisely determine the requirement priority level ML model predicted: correctly recognized wrong priority. Number of TN result: 75

Accuracy= (TP+TN)/ (TP+TN+FP+FN) = (36+75)/ (36+75+3+3) =94.87%
Accuracy comes out to 0.9487, or 94.87%. That indicates our algorithm is doing a great job of determining the requirements priority level.

Precision(P): Is the ratio of the number of related instances retrieved to the total number of unrelated and related records retrieved (Sarhan et al., 2016). It seems to be the vital metric of our algorithm as it determines the requirements for specific release.

$$P = \frac{TP}{(TP + FP)} \times 100$$
$$P = \frac{36}{(36+3)} \times 10$$
=0.92307
= 92.31

Recall (R) (Sarhan et al., 2016) also known as Sensitivity: Is the ratio of the number of related instances retrieved to the total number of existing relevant instances, defined as:
$$P = \frac{TP}{(TP+FN)} \times 100$$

$$P = \frac{36}{(36+3)} \times 100$$

=0.92307
= 92.31
Accuracy (A): Is the fraction of true results against the total number of cases evaluated, defined as:
$$ACC = \frac{TP + TN}{(TP + TN + FP + FN)} \times 100$$

Where TP: the number of true positives, FN: the number of false negatives, TN: the number of true negatives, FP: the number of false positives.

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

$$ACC = \frac{36 + 75}{(36 + 75 + 3 + 3)} \times 100$$

= 94.87

Accuracy comes out to 94.87%. That means our algorithm is doing a great job of determining the requirements priority level from SRP document.
We achieved a maximum measure of 94.87%, 92.31%, and 92.31% of average accuracy, precision, and recall, respectively, as represented in figure 5.

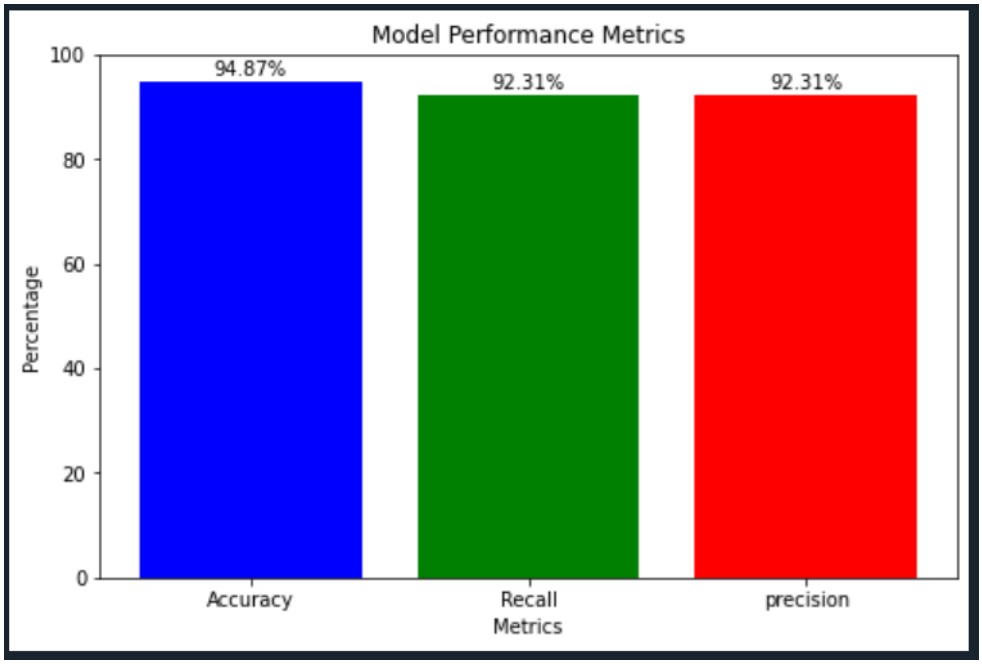

Figure 5. The performance metrics

The ease of use requires an interface for interaction between the approach and users, Thus, the Graphical User Interface (GUI) of the APRP approach main page is designed as denoted in Figure 6. APRP approach is able to handle different datasets as long as cleaning the dataset. An Example of requirements parallel prioritization result is presented in Figure 7.

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

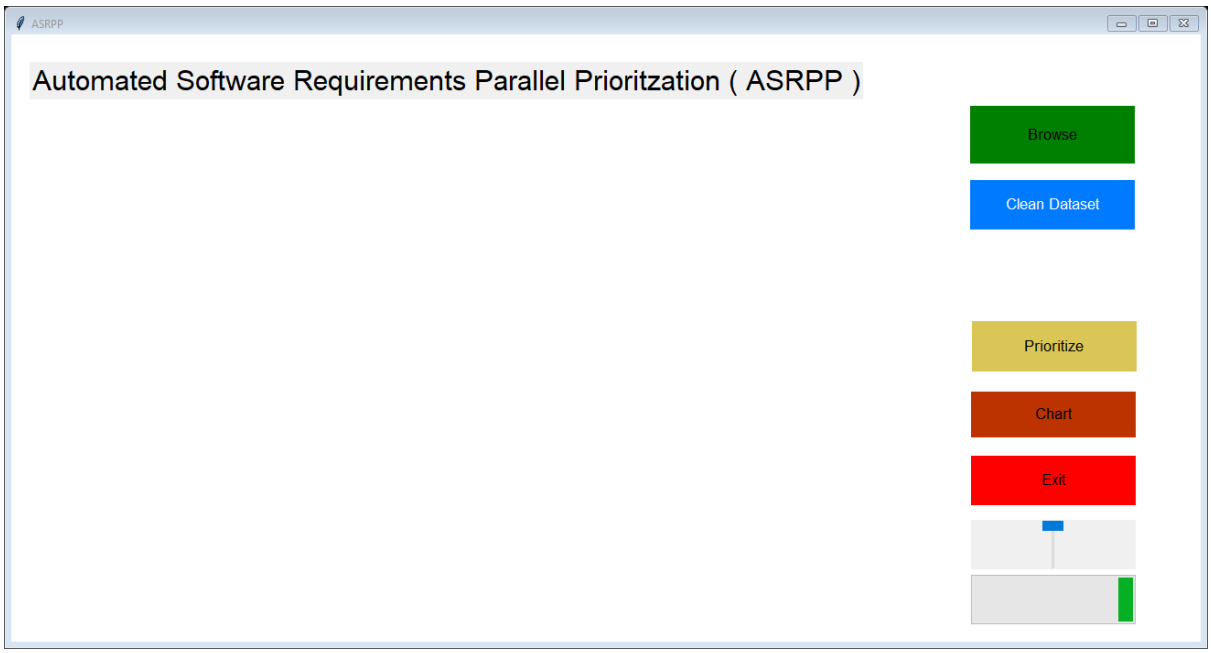

Figure 6. Graphical User Interface (GUI) of APRP approach

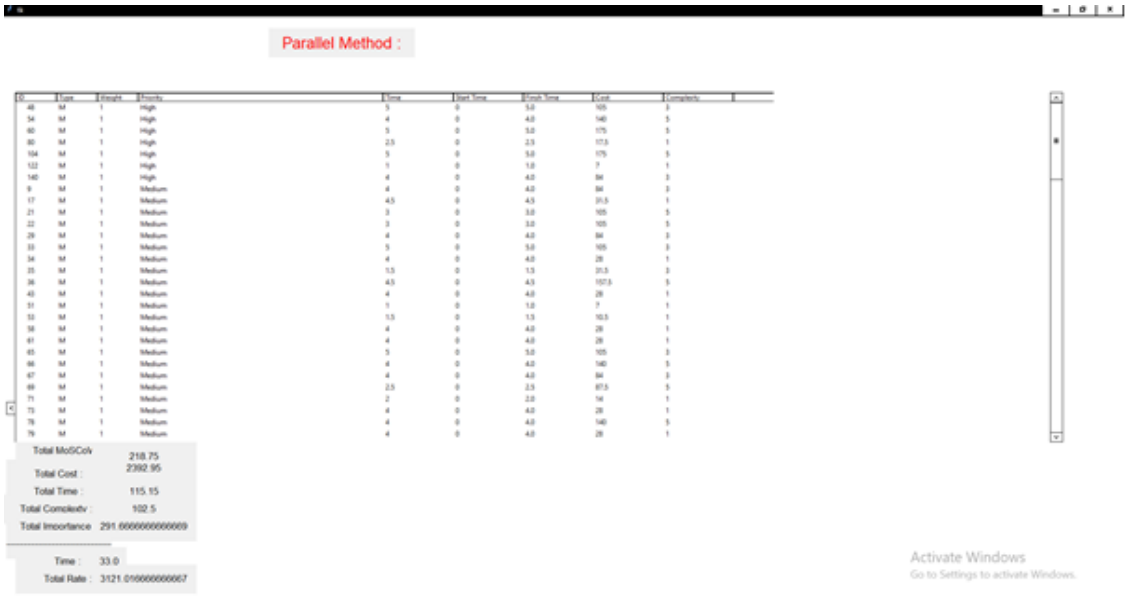

Figure 7. Example of requirements parallel prioritization result

4. Results Discussion

Experiments on automated requirements priority detection were conducted using the Testcase MIS system with a priority dataset. We trained the requirements prioritization approach on 1,314 requirements within this dataset and observed a promising reduction in implementation time. Specifically, parallel ranking reduced implementation time to 33 seconds, compared to 76 seconds in a

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

sequential setup. This demonstrates that parallel ranking can cut implementation time by more than half, proving its effectiveness in optimizing requirements prioritization. When we evaluate the performance in classification and priority level determination, it achieved a maximum measure of 94.87%, 92.31%, and 92.31% of average accuracy, precision and recall, respectively.

A strength of our approach is that unlike many other prioritization methods, as it beats CDBR, SARiP, and IQ-BR in term of concern the impact of priority ranking on software development cycles, accuracy, ease of use, as well as its ability of handling projects with massive number of requirements. Further, our approach introduces the parallel ranking concept in RP, which lead to reducing implementation time to more than have along with high level of accuracy.

Table 7: Study Comparisons

Factor	**CDBR(2018)**	**SARiP (2022)**	**IQ-BR(2022)**	**Proposed Approach APRP (2024)**
Automation	Semi-automated	Semi-automated	Full-automated	Full-automated
Prioritization criteria	Stakeholders and developer's Communication, Dependency. Scalability, and Importance	Stakeholders prioritization, Cost, Time, Complexity, Risk, and Importance	Customer prioritization, Dependency Accuracy, Prioritization time.	Stakeholders and developer's criteria, Dependency, Cost, Time, Complexity, Importance, and Scalability
No of input Requirements	100	Not estimated	60	>149
Scalability	Considered	Not Considered	Considered	Considered
Parallel RP	-	-	-	Considered
Awareness with the effect of RP on software developments cycle	-	-	-	Considered
Implementation time	-	-	-	Considered
Accuracy	90%	Not estimated	93.15	94.87

5.Conclusion

In this paper, we designed a machine learning approach integrated with MoSCoW method, capable of determining requirement priority levels in a parallel format, significantly reducing implementation time. When measuring this reduction, we found that sequential ranking required 76 seconds, while parallel ranking reduced this to 33 seconds. This demonstrates that parallel ranking can cut implementation time by more than half. The approach was applied to a Testcase MIS system with a priority dataset, providing valuable insights into the impact of priority ranking on software development cycles. By applying the classification, we achieved a maximum measure of 94.87%, 92.31%, and 92.31% of average accuracy, precision and recall, respectively.

Based on our outcomes, ML within RP promising to address present challenges and uncover new opportunities for improvement. Further, this research sets the stage for future enhancements in the field RP based computational intelligence. Future work will improve the approach performance by storing the requirements and prioritization results in a database in order to evaluate the approach effectiveness and refine it over time.

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

6. Acknowledgment

This work was supported by the Deanship of Scientific Research and the Deanship of Faulty of Computer Science &Information Technology at Omdurman Islamic University, Omdurman, Sudan.

Funding:

"This research received no external funding"

References

Aurum, A. (2005). Engineering and managing software requirements (Vol. 1). C. Wohlin (Ed.). Heidelberg: Springer.

Lunarejo, M. I. L. (2021, September). Requirements prioritization based on multiple criteria using Artificial Intelligence techniques. In 2021 IEEE 29th International Requirements Engineering Conference (RE) (pp. 480-485). IEEE.

Achimugu, P., Selamat, A., Ibrahim, R., & Mahrin, M. N. R. (2014). A systematic literature review of software requirements prioritization research. Information and software technology, 56(6), 568-585.IEEE.

Shao, F., Peng, R., Lai, H., & Wang, B. (2017). DRank: A semi-automated requirements prioritization method based on preferences and dependencies. Journal of Systems and Software, 126,141-156.

Fadlallah, KIA., Sharif, M. M., & Eldow, M. E. Y. (2024). Developing Parallel Requirements Prioritization Machine Learning Model Integrating with MoSCoW Method, Journal of Artificial Intelligence and Computational Technology, 1(1).

Fadlallah, K. I. A., & Eldow, M. E. Y. (2024). Machine learning: A survey of requirements prioritization: A review study. Journal of Artificial Intelligence and Computational Technology, 1(1).

Gupta, A., & Gupta, C. (2018). CDBR: A semi-automated collaborative execute-before-after dependency-based requirement prioritization approach. Journal of King Saud University-Computer and Information Sciences, 34(2), 421-432.

Chua, F. F., Lim, T. Y., Tajuddin, B., & Yanuarifiani, A. P. (2022). Incorporating semi-automated approach for effective software requirements prioritization: A framework design. Journal of Informatics anAd Web Engineering, 1(1), 1-15.

Qayyum, S., & Qureshi, A (2018, November). A survey on machine learning based requirement prioritization techniques. In Proceedings of the 2018 International Conference on Computational Intelligent Systems (pp. 51-55).

Jan, N., Inayat I., and Abbas, M. (2020). An Empirical Evaluation of Requirements Prioritization Techniques. Marketing and Branding Research,7 (1), 11.

Bukhsh, F. A., Bukhsh, Z. A., & Daneva, M. (2020). A systematic literature review on requirement prioritization techniques and their empirical evaluation. Computer Standards & Interfaces, 69, 103389.

Hujainah, F., Bakar, R.B.A., Al-haimi, B.& Abdulgabber,M.A.(2018). Stakeholder quantification and prioritisation research: A systematic literature review.Information and Software Technology ,102, 85–99, https://doi.org/10.1016/j.infsof.2018.05. 008.

Sher, F., Jawawi, D. N., Mohammad, R., Babar, M. I., Kazmi, R., & Shah, M. A. (2020). Multiaspects Intelligent Requirements Prioritization Technique for Value Based Software Systems. In Intelligent Technologies and Applications: Second International Conference, INTAP 2019.

Amelia, T., & Mohamed, R. (2022). A Review: Requirements Prioritization Criteria Within Collaboration Perspective. Journal homepage: http://www.pertanika.upm.edu.my/.

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243

Hujainah, F., Bakar, R. B. A., Nasser, A. B.Al-haimi, B., & Zamli, K. Z. (2021). SRPTackle: A semi-automated requirements prioritization technique for scalable requirements of software system projects. Information and Software Technology, 131, 106501.

Devadas, R., & Cholli, N. G. (2022). Interdependency Aware Qubit and Brownboost Rank Requirement Learning for Large Scale Software Requirement Prioritization. International Journal of Computing and Digital Systems, 11(1), 625-635.

Keertipati, S., Savarimuthu, B. T. R., &. Licorish. S. A. (2016, June). Approaches for prioritizing feature improvements extracted from app reviews. In Proceedings of the 20 th international conference on evaluation and assessment in software engineering (pp. 1-6).

Gambo, I. P., Ikono, R., Iroju, O. G., Omodunbi, T. O., & Zohoun, O. K. (2021). Hybridized ranking model for prioritizing functional software requirements: Case study approach. International Journal of Software Innovation (IJSI), 9(4), 19-49.

Ahmad, S., Rizawanti, R., Woodings, T., & Jalil, I. E. A. (2022). MCBRank Method to Improve Software Requirements Prioritization. International Journal of Advanced Computer Science and Applications, 13(7).

Sarhan ,I., El-Sonbaty ,Y., & Abou El-Nasr, M. (2016, November). Semi-Supervised Pattern-Based Algorithm for Arabic Relation Extraction. In 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI). (pp. 177-183). IEEE.

DOI: https://doi.org/10.70274/jaict.2025.2.1.
p- ISSN 34562-3478 e- ISSN 5443-1243