SRPTackle: A semi-automated requirements prioritisation technique for scalable requirements of software system projects

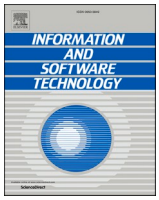
Citation for the original published paper (version of record):
Hujainah, F., Binti Abu Bakar, R., Nasser, A. et al (2021). SRPTackle: A semi-automated requirements prioritisation technique for scalable requirements of software system projects. Information and Software Technology, 131. http://dx.doi.org/10.1016/j.infsof.2020.106501

Information and Software Technology 131 (2021) 106501

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



SRPTackle: A semi-automated requirements prioritisation technique for scalable requirements of software system projects

Fadhl Hujainah [a,*], Rohani Binti Abu Bakar [b], Abdullah B. Nasser [b], Basheer Al-haimi [c], Kamal Z. Zamli [b]

[a] Computer Science and Engineering Department, Chalmers and University of Gothenburg, 41296 Gothenburg, Sweden
[b] Faculty of Computing, Universiti Malaysia Pahang, Kuantan 26300, Malaysia
[c] School of Management, Hebei University, Baoding, China

A B S T R A C T

Context: Requirement prioritisation (RP) is often used to select the most important system requirements as perceived by system stakeholders. RP plays a vital role in ensuring the development of a quality system with defined constraints. However, a closer look at existing RP techniques reveals that these techniques suffer from some key challenges, such as scalability, lack of quantification, insufficient prioritisation of participating stakeholders, overreliance on the participation of professional expertise, lack of automation and excessive time consumption. These key challenges serve as the motivation for the present research.
Objective: This study aims to propose a new semiautomated scalable prioritisation technique called 'SRPTackle' to address the key challenges.
Method: SRPTackle provides a semiautomated process based on a combination of a constructed requirement priority value formulation function using a multi-criteria decision-making method (i.e. weighted sum model), clustering algorithms (K-means and K-means++) and a binary search tree to minimise the need for expert involvement and increase efficiency. The effectiveness of SRPTackle is assessed by conducting seven experiments using a benchmark dataset from a large actual software project.
Results: Experiment results reveal that SRPTackle can obtain 93.0% and 94.65% as minimum and maximum accuracy percentages, respectively. These values are better than those of alternative techniques. The findings also demonstrate the capability of SRPTackle to prioritise large-scale requirements with reduced time consumption and its effectiveness in addressing the key challenges in comparison with other techniques.
Conclusion: With the time effectiveness, ability to scale well with numerous requirements, automation and clear implementation guidelines of SRPTackle, project managers can perform RP for large-scale requirements in a proper manner, without necessitating an extensive amount of effort (e.g. tedious manual processes, need for the involvement of experts and time workload).

1. Introduction

To ensure the fulfilment of the stakeholders' requirements, various decisions have to be made through software development [1,2]. Securing stakeholders' core requirements is a primary driver to achieve good quality in a system [3,4]. Many system projects have several requirements, and implementing all of them with limited resources (e.g. insufficient budget, time and technical staff) is difficult [5,6]. Thus, requirement prioritisation (RP) is often executed to assist requirement engineers in determining the order in which to implement requirements as perceived by the stakeholders of a system. In RP, the most important or highest risk requirement is selected to produce a quality system [5,7]. RP is a crucial process in software development for decision-making because information on priorities is critical for project managers to resolve conflicts, plan for staged deliveries and make necessary trade-offs [5,8]. Thus, the influence of RP cannot be overstated [5,9].

RP is a complex decision-making process [5,10]. To execute such process, various techniques, such as StakeRare [11] and Drank [12], have been proposed. Although useful, existing RP techniques suffer from key challenges, such as lack of scalability (i.e. ability to manage numerous requirements); lack of time efficiency, especially in prioritising a large set of requirements; lack of stakeholder quantification and prioritisation (SQP) processes for evaluating the effects of participating stakeholders in prioritising requirements; heavy reliance on the involvement of experts in conducting the prioritisation process; and lack of automation [3, 5, 13, 14].

The scalability issue has a critical effect on the prioritisation process in industrial projects because majority of current industrial projects have numerous requirements [5,13]. A scalable RP technique should be able to work with a large set of requirements when performing RP within a reasonable time whilst producing accurate results [3, 5, 13]. In RP, sets of requirements are categorised into three different categories, i.e. small

* Corresponding author.
E-mail address: fadelhogina@gmail.com (F. Hujainah).

https://doi.org/10.1016/j.infsof.2020.106501
Received 29 March 2020; Received in revised form 23 November 2020; Accepted 24 November 2020
Available online 29 November 2020

(number of requirements < 15), medium (15 <= number of requirements < 50) and large (number of requirements >= 50) sets, as defined in [15]. Despite the fact that software products have become more complex (e.g. containing a large set of requirements), most existing RP techniques can only work well with a few requirements; such techniques include the analytic hierarchy process (AHP) technique [10, 13]. Fig. 1 presents the percentage of the acceptable handling of the scalability issue within RP techniques, as highlighted in a study on existing 107 RP techniques in [5,13].

As shown in Fig. 1, the percentage of the poor handling of scalability issues on the basis of the 107 RP techniques is 93%, with only 7% successful ones. amongst the successful ones, existing techniques, such as PHandler [15] and StakeRare [11], still face issues regarding manual processing and heavy reliance on the participation of experts in the prioritisation process (i.e. in assigning the priority value for each requirement, classifying the requirements, and evaluating the influence of participating stakeholders in the prioritisation). The latter raises the issue of potential bias being introduced by experts because such bias can influence the accuracy of a technique [5,16]. Furthermore, for most current techniques, certain factors, such as the number of comparisons, time, lack of automation and heavy reliance on expert participation, play key roles with regard to the scalability problem [5, 13, 17]. In most cases, the stakeholders of the system must rate the importance degree and make comparisons between each requirement. This process becomes even more complex as the number of requirements increases [3, 5]. Additionally, a manual process that heavily relies heavily on experts to conduct the procedure and measurements and assess the relative priority value of each requirement can also be counterproductive and time consuming. Therefore, having to handle a large set of requirements makes these RP techniques unmanageable [13,17]. Excessive reliance on professional expertise is not preferred due to threats to the validity of the technique if an expertise shortage occurs [6, 15, 18]. In such conditions, interpreting and understanding the requirements for the technique's initiation and implementation can become difficult. Moreover, such approach increases the likelihood of implementing the technique in an improper and/or nonprofessional manner, thus directly affecting the quality of results [5, 6, 15]. The potential biases induced by experts arise in evaluating the respective impact degree of the participating stakeholders in RP and specifying the priority values of the requirements or classifying the requirements. These biased assessments can also influence the quality of the prioritised results with respect to specifying accurate requirement priority values (RPVs) and the quality of the most important requirements to be developed to secure a successful software system project [5, 6, 15].

In consideration of numerous requirements, a lack of automation during the RP process influences the efficiency of the technique because RP becomes complicated due to the effort required [5, 13, 17]. Manual RP affects time efficiency [5, 13, 19]. An increase in the number of requirements considerably affects time efficiency because of the complexity of implementing prioritisation. Such complexity is related to the high number of comparisons and the manual processes required to execute the computational calculations for specifying RPVs and producing a prioritised list of requirements [5, 12, 13, 15]. Furthermore, the selection of stakeholders who are involved in the prioritisation process of the requirements is crucial to securing accurate RP results [5, 6, 11, 15]. A stakeholder's influence on system requirements and development success varies from one stakeholder to another; moreover, the number of participating stakeholders of diverse types can be enormous, with each stakeholder interpreting their requirements differently [6, 11, 20]. Hence, an SQP process is conducted, with an aim to identify a stakeholder priority value (SPV) for each stakeholder and prioritise the stakeholders [6, 18, 20]. This process assists in identifying stakeholders who have a greater influence on project success, leading to the selection of the most essential requirements for important stakeholders and thus a successful system [5, 6, 15]. However, most of the existing RP techniques do not execute an SQP process. A few techniques perform the SQP process manually and rely heavily on substantial professional human intervention to specify the SPV of each participating stakeholder. This approach provides high abstract details without providing standard measurement criteria for quantifying and prioritising the stakeholders on the basis of the SQP attributes; however, it is not time efficient [5, 21, 22].

To cope with these key challenges, this study aims to propose a new semiautomated technique named 'SRPTackle'. SRPTackle presents a new process for prioritising requirements on the basis of a combination of the following: a constructed RPV formulation using a multi-criteria decision-making method (i.e. weighted sum model [WSM]), a classifying algorithm (i.e. K-means and K-means++) and a binary search tree (BST). The contributions of this work can be summarised as follows:

- A new RP technique called 'SRPTackle' is proposed. This technique presents low-level details for the automatic implementation of prioritisation for scalable requirements in software system projects, without requiring a considerable amount of effort (e.g. need for experts' participation and/or a tedious manual process in addition to potential human faults in the manual process and time workload). SRPTackle supports the RP process in system projects that contain various types of stakeholders with competing interests and limited resources (i.e. where each stakeholder defines their needs differently). SRPTackle also evaluates the influences of participating stakeholders by identifying their SPV values automatically with minimal expert involvement. The SPV values are considered in specifying the priority value of each requirement. To produce a prioritised list of requirements, SRPTackle employs a clustering algorithm (K-means and K-means++) and BST, along with a constructed RPV formulation function, on the basis of the WSM method.
- The development of the automation implementation tool (SRPTackle-Tool) along with presenting clear implementation

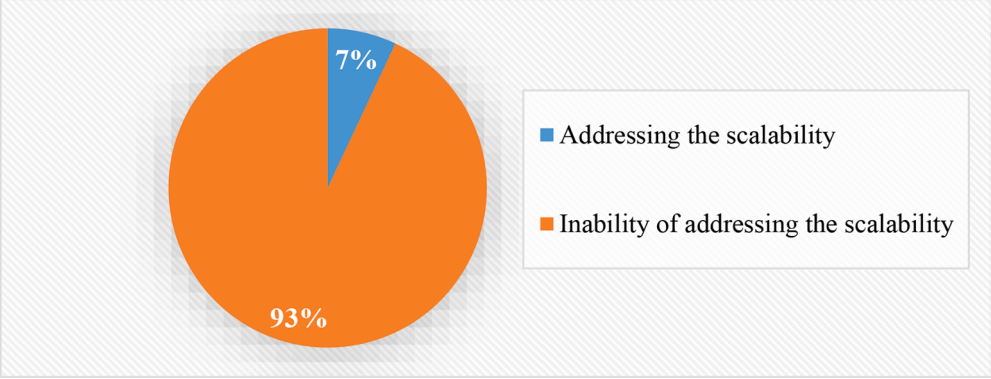


Fig. 1. Percentage of the existence of the scalability issue in RP techniques.

guidelines for automating the SRPTackle process and supporting straightforward implementation of the proposed technique in industrial and academic sectors.
- The empirical evaluation of SRPTackle is based on a large real software project to affirm its capability to address the key issues in existing RP techniques.

The remainder of this paper is structured as follows: Section 2 describes related work on RP. Section 3 presents a detailed description of SRPTackle, with respect to the proposed process. Section 4 illustrates an automation implementation tool developed for SRPTackle. Section 5 discusses an assessment of the proposed SRPTackle technique. Section 6 enumerates the experimental results. Section 7 presents a detailed discussion about the trends observed during the experimental analysis and comparison, along with essential clarifications associated to the base of the proposed SRPTackle technique. Section 8 elaborates on threats to validity. Section 9 explores the managerial contributions of SRPTackle. Lastly, Section 10 concludes the study and provides recommendations for future work.

2. Related work

The execution of RP in software development has led to an efficient negotiation of precise requirements [1,23]. These precise requirement negotiations assist software engineers in eliminating unnecessary or contradictory requirements [5,13]. Additionally, an RP process can assist in securing an effective implementation schedule, allowing project managers to modify project resources and delivery dates on the basis of environmental circumstances [5,15]. Project managers also improve stakeholders' satisfaction by conducting the RP process and increasing the likelihood that their preferred requirements are implemented. Hence, RP makes the rejection of projects after development less likely through the creation of clear and precise requirements [13].

To date, various techniques have been introduced to perform a prioritisation process for the requirements of a system. The most straightforward and common RP technique is perhaps the one that is based on numerical ranking [5, 13, 14]. This technique performs the prioritisation process manually by classifying the requirements into three priority groups: high, middle and low groups [5, 13, 15]. The classification is performed according to stakeholders' preferences and expert judgement [5,13]. Likewise, a priority group technique executes the RP process by initially categorising the requirements into three groups [5,13]. Unlike the numerical assignment technique, the priority group technique groups requirements into new groups repeatedly, until only one requirement remains in each categorised group [5,13]. The ranking technique is another RP technique that prioritises requirements in a manner similar to the numerical assignment technique. The difference is in using a linear approach, where a value of 1 will be assigned to the most important requirement, a value of 2 to the second most important requirement, and so on, until the least important requirement is assigned with the value of n, which indicates the complete number of requirements in the set [15]. Owing to the difficulty in aligning the views of several stakeholders, the ranking technique can only be convenient when performing the RP process with a single participating stakeholder [5, 13, 14]. In the top-ten technique, the prioritisation is performed by a number of various stakeholders, who are responsible for selecting their own top ten requirements [5,13]. These techniques (numerical assignment, top-ten and priority group and ranking) are suitable for a small dataset of requirements and have inabilities and/or disadvantages in dealing with large-scale requirements, specifying the relative priority value of each requirement and catering the SQP process to evaluate participating stakeholders [5,13].

As an improvement of the numerical ranking approach, several works in the literature adopt the AHP technique. Specifically, the AHP technique performs the prioritisation process by manually executing pairwise comparisons on the basis of experts' judgements, without considering the SQP process for participating stakeholders [5,13]. Although the AHP technique is considered one of the best techniques in terms of reliability, the AHP is not suitable for a large number of requirements, as of the number of pairwise comparisons will increase as the number of requirements increases [5,13]. Moreover, the AHP has issues on time consumption and complexity when used with a large scale of requirements [5,13]. As such, a hierarchy AHP (HAHP) technique has been presented, with the main purpose of addressing the scalability issues in AHP techniques [14, 15, 24]. HAHP succeeded in reducing the number of comparisons by only comparing requirements on the same hierarchy level [5,13]. However, the HAHP technique is less reliable and fault tolerant compared with AHP, and is more difficult to apply [5,13]. In a similar work, the ReDCCahp introduces a new method of dynamic consistency checking to eliminate the redundant AHP comparisons, ultimately minimising the number of pairwise comparisons [10]. Combining AHP with neural network, the PHandler approach automated the assignment of priority value for each requirement [8,15].

Apart from adopting neural networks, some works focus on machine learning approaches. In CBRanking, machine learning approach is applied to predict preference values of selected pairs and to generate approximate ranks for requirements [25]. In other related works, the interactive genetic algorithm (GA) technique optimises the list of prioritised requirements by minimising the number of comparisons [26]. Extending the existing work on StakeRare [11] and Saffron [27] (i.e. techniques based on the application of social networks and collaborative filtering), Lim et al. applied the GA to rank the impact of each participating stakeholder in assigning the priority value for each requirement [28]. Similarly, the WCW approach adopts a hybrid grey wolf and whale optimisation algorithm to rank requirements [29].

Based on the aforementioned works, a number of general observations can be deduced. Firstly, most, if not all, of the related techniques treat all participating stakeholders at par, and the same impact degree is assigned to all entities. Thus, their credibility falls short as different stakeholders may have different contribution and focus within the RP process [5]. Secondly, despite the usefulness of these techniques, one major challenge in these techniques is that they require deep expertise and knowledge to initiate, interpret and execute their prioritisation processes. This challenge incorporates human nature biases induced by experts, or threats related to the technique's validity in the absence of human experts [5,15]. Finally, the majority of these approaches lack an automated prioritisation process and are not cost-effective with respect to time utilisation [5].

Supporting the aforementioned observations, various review studies have been conducted to critically analyse the strengths and challenges of existing RP techniques. Some common and recent studies include those of Sufian et al. [14], Achimugu et al. [13], and Hujainah et al. [5], who presented critical and comprehensive analyses of 40, 49, and 108 RP techniques, respectively . The findings of these studies, along with those of other studies [9, 21, 30, 31], demonstrated that the existing techniques face major limitations as regards to scalability, cost effectiveness in terms of time consumption, heavy reliance on experts to initiate and execute the prioritisation process, the lack of an SQP process for evaluating the impacts of stakeholders on system requirements and a lack of automation. Therefore, to address these limitations, this paper proposes a new semi-automated technique for scalable requirements (SRPTackle), and constructs a new automation implementation tool (SRPTackle-Tool) for providing clear implementation guidelines to automate the SRPTackle process and support straightforward execution of the technique in industrial and academic sectors. The following section provides a detailed explanation of SRPTackle.

3. Proposed SRPTackle technique

Supporting fully automated prioritisation can mitigate repetitive manual operations but should not impede human experts' judgement and stakeholders' involvement during the prioritisation process.

However, in RP, stakeholders' participation and expert roles cannot be totally eliminated due to the participation of software product where human expertise is important, and the prioritisation process has to be conducted on the basis of stakeholder preferences [5, 12, 13]. Thus, and based on [5, 12, 13], recommendation and attempts on automating parts of the RP process and the essential elements of conducting RP reported in [5,32], the prioritisation process of the SRPTackle is constructed as a semi-automated process to address the RP key limitations.

The process of the SRPTackle technique is illustrated in Fig. 2. It consists of two main phases: pre- and post-prioritisation phases. The aim of conducting the process of the SRPTackle is related to establish the semi-automated process of SRPTackle in prioritising the requirements. The pre-prioritisation phase is constructed initially in the SRPTackle technique to collect stakeholders' preferences, in which participating stakeholders assign an initial weighting value to each requirement, which should be obtained to perform the prioritisation process. Thus, this phase is considered the basis for performing the full process of the pre-prioritisation phase in an automated manner. This step can assist to automate the process and minimise the need for expert participation in terms of assessing stakeholders' impacts and formulating the priority value of each requirement, and classifying and generating the prioritised list of requirements. The following subsections provide the detailed description of each phase of the SRPTackle technique.

3.1. Pre-prioritisation phase

The aim of this phase is to obtain initial weighting values for the requirements from stakeholders, which will be used as inputs in the post-prioritisation phase. In the SRPTackle technique, participating stakeholders assign an initial weighting value to each requirement on the basis of two prioritisation criteria: importance and cost. These defined criteria are the most significant prioritisation criteria in prioritising requirements [5,33].

The usage of these two criteria (importance and cost) can assist in guaranteeing the production of a balanced list of prioritised requirements based on the perspectives of all stakeholder types. In this example, the stakeholder types are categorised as functional beneficiaries, technical stakeholders and commercial stakeholders [5, 33, 34]. The importance criteria are used to prioritise requirements according to their importance to the needs of functional beneficiary stakeholders (users and customers) to estimate their expected satisfaction. In contrast, prioritising requirements on the basis of cost criteria is performed by technical (i.e. development teams) and commercial (i.e. business analysts and marketing managers) stakeholders to specify the priority order of the requirements according to the required cost for each requirement to be implemented [5,34]. Most current industrial companies aim to prioritise requirements on the basis of the importance criteria to obtain stakeholders' expectations and to use the cost criteria to prioritise requirements according to the required cost for implementing each requirement [5,34]. In SRPTackle, functional beneficiary stakeholders should assign the weight value for each requirement on the basis of the importance criteria. This weight value is denoted as the requirement importance weight value (RIWV) and refers to the importance of the requirement to the functional beneficiary stakeholders (e.g. whether it is to be implemented and delivered first). In contrast, technical and commercial stakeholders assign a weight value to each requirement according to the cost of implementation. This assigned weight value is denoted as the requirement cost weight value (RCWV). The weight values are employed on a scale of 1 (lowest weighting value) to 5 (highest weighting value). The RIWV and RCWV serve as inputs for conducting the, post-prioritisation phase.

3.2. Post-prioritisation phase

In this phase, the prioritisation process is executed, based on the initial weight values of the requirements. The full implementation of this phase is conducted in four steps: specifying the SPV for each stakeholder, formulating the RPV, classifying and generating the prioritised list of requirements by employing the K-means and K-means++ algorithms, and applying the BST algorithm. The following subsections

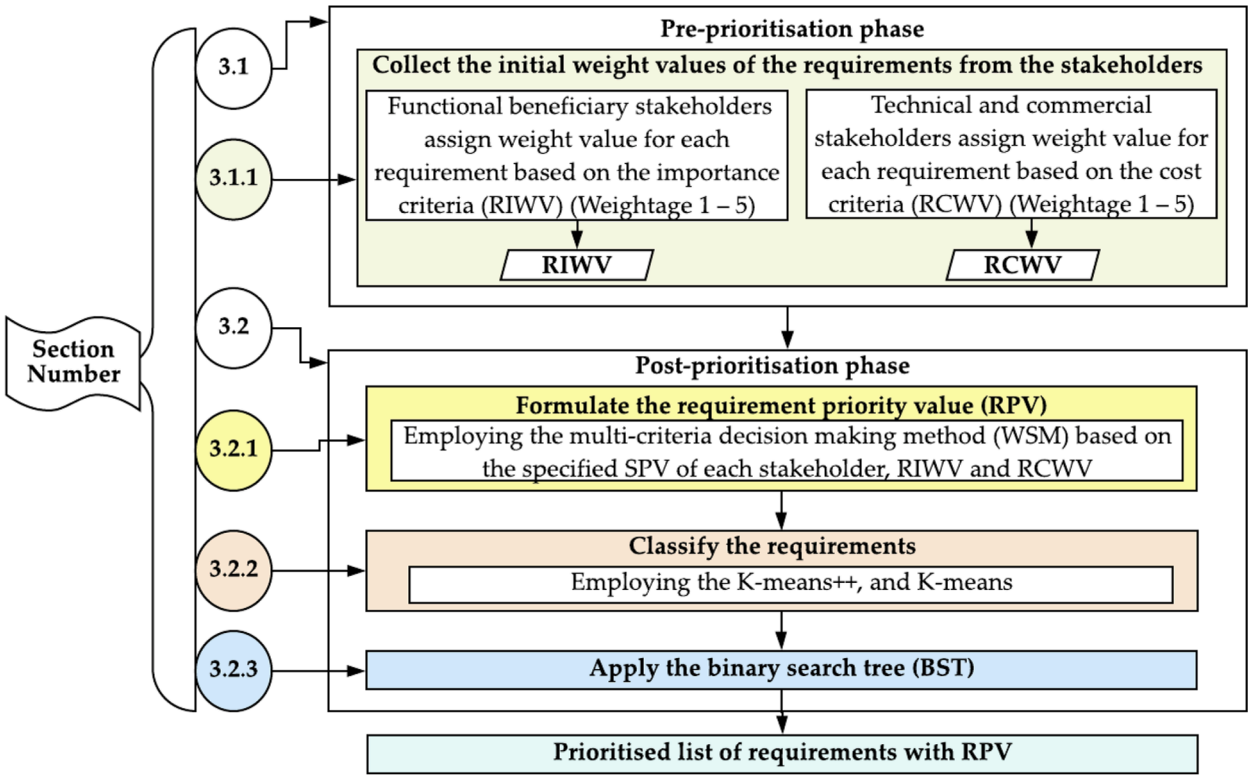


Fig. 2. Process of the proposed SRPTackle technique.

explain the details of each step.

3.2.1. Formulation of the requirement priority value

In this step, the RPV for each requirement is formulated by applying the WSM method, which is considered one of the most commonly applied and useful methods in performing the process of multi-criteria decision-making [35–38]. The WSM method has silent properties of simplicity and time efficiency, i.e. the priority of variants can be evaluated and revealed over listed criteria in a shorter time with an easily applicable computational process relative to other multi-criteria decision-making methods, such as AHP [35–38].

Additionally, unlike existing RP techniques, SRPTackle evaluates the influences of participating stakeholders in the RP process, which are quantified and prioritised by identifying the impact degree (SPV value) of each stakeholder in the RP. The SPV denotes the impact degree (priority) value of each participating stakeholder (SPV) in the RP process. The SPV values are considered in specifying the priority value of each requirement. SRPTackle uses StakeQP to conduct the SQP process to evaluate the influences of participating stakeholders by identifying the SPV of each participating stakeholder in the prioritisation of requirements. The StakeQP is a newer SQP technique with an automation implementation tool (StakeQP-AIT) that evaluates stakeholders' impacts on the basis of attributes (called StakeQP attributes): role influence, positional power, interest and experience with respect to the knowledge and education background attributes, as documented in [6]. StakeQP provides a semi-automated process for quantifying and prioritising stakeholders on the basis of the new attribute measurement criteria for each attribute (AMC). StakeQP also employs a multi-attribute decision-making method, namely, the 'technique of order preference similarity to the ideal solution' to specify the SPV of each stakeholder. The AMC refers to the measurement used to assess stakeholders' influence(s) on each attribute [6]. A higher SPV indicates a more significant stakeholder. The calculation of RPV is executed on the basis of the WSM method and considers the SPV of each participating stakeholder, as shown in Eq. (1), in which three inputs are used to identify the RPV: RIWV, RCWV and SPV.

$$RPV_i = \left(\sum RIWV_{i,s} * SPV_s\right) + \left(\sum RCWV_{i,s} * SPV_s\right) \quad (1)$$

Here,

RIWV refers to the weight value of the i^{th} requirement that is given from the s^{th} stakeholder, based on the importance criteria;

RCWV refers to the weight value of the i^{th} requirement that is given from the s^{th} stakeholder, based on the cost criteria; and

SPV refers to priority value associated with the s^{th} stakeholder.

The RIWV and RCWV are obtained from the pre-prioritisation phase (as described in Section 3.2). The SPV of each participating stakeholder is obtained by executing the StakeQP technique in [6]. The output of this step is the RPV for each requirement, which will be used as an input for the next step in classifying the requirements.

3.2.2. Classifying the requirements

Clustering refers to unsupervised learning classification, with the aim of classifying unlabelled data points into several classifications with respect to information found in the data that describes the points and their relations, such as a suitable similarity measure [39–42]. One commonly used clustering algorithm is K-means [39,43]. The K-means algorithm aims to cluster data points into a number of clusters, where each cluster has its own centroids, and in which each point is grouped to the cluster with the nearest centroid [40,44]. K-means has salient advantages in simplicity (simple and easy to implement for handling practical problems) [39, 45, 46], speed (computationally fast) [39, 45, 46], and efficiency in working with large datasets that contain numeric values [39, 43, 46], as compared to other clustering methods, such as K-medoids, partitioning around medoids, 'clustering large applications', and fuzzy clustering. Therefore, K-means is used in this study to cluster the requirements based on their obtained RPVs, in which the numeric values represent the priority value of each requirement.

The cluster centroids are considered as crucial elements in the K-means algorithm. The performance of the K-means algorithm with respect to the speed (time utilisation with number of iterations) and accuracy to find the optimal clustering can be affected by the random initialisation of the clusters' centroids. In that regard, improper centroid initialisation can lead to drawbacks in terms of slower convergence (requiring a high number of iterations to converge), empty clusters, and a higher probability of getting stuck in bad local minima, which can lead to a low-accuracy result in clustering the data [39,43]. Hence, the K-means algorithm is considered to be sensitive to the selection of initial centroids; however, this can be managed by adopting a proper initialisation method for initialising the cluster centroids instead of random selection [39,43]. The K-means++ algorithm is presented in [43] as an initialisation method for centroid selection in the K-means algorithm. The K-means++ algorithm adds value by improving the accuracy and the speed of the K-means algorithm, as proven in [43]. The initialisation method of K-means++ has been proven to address the sensitivity to initial centroid selection. In particular, the performance using the K-means++ initialisation method substantially outperforms K-means with random centroid initialisation in terms of producing more accurate clustering solutions and requiring less time consumption for conducting the clustering process, as fewer iterations are required to help achieve local search convergence [43]. Moreover, the process of K-means++ is fast and easy to implement in real practice [39,43]. As a result, K-means++ is selected in SRPTackle as the initialisation method for the centroid selection of the K-means algorithm.

In SRPTackle, the number of clusters (k) is specified to be three, based on the numerical assignment technique, which recommends to group the requirements into three levels (high, medium, and low) [5, 13]. However, in the proposed SRPTackle, the requirements are classified based on their obtained RPV values, rather than by asking a stakeholder to classify them as in the NA technique. With a large number of clusters (e.g. 100 clusters), the speed performance of the K-means++ can be reduced because of the high number of iterations required, which can lead to some time utilisation constraints in the application of massive data practices [39,43]. This may relate to the sequential nature of the K-means++ algorithm, i.e. making an iteration over the data to initiate each centroid. In this regard, 99 iterations are required to find the centroids for the 100th cluster, as the number of iterations is equal to (K-1) in K-means++ [39,43]. However, with the specified number of clusters in SRPTackle (three clusters), the K-means ++ algorithm can perform the initialisation for three cluster centroids without consuming excessive time, as the K-means++ algorithm will need only two iterations to initialise the three cluster centroids (the first centroid is assigned uniformly at random from the data points, and then two iterations to initialise the other two centroids(k-1)) [39,43]. The classification of each requirement to each group (cluster) is performed based on the process of K-means++ and K-means algorithms. Fig. 3 presents the pseudo code of the process for clustering the requirements into three clusters based on one element, i.e. the RPV value of each requirement.

The implementation steps of the clustering process start with the initialisation of centroids (c1, c2, c3) for the three clusters [43]. The initialisation mechanism of the K-means++ algorithm is presented in lines 1 to 5 of Fig. 3. The first centroid (c1) is initialised randomly to any requirement RPV value (y_r) from Y, as shown in line 1. The steps in lines 2 to 5 are recursively executed to initialise the next centroids, until all centroids of the defined clusters have been chosen. In line 3, the distance y_r to the nearest defined centroid ci is computed using the Euclidean distance as shown in Eq. (2). This is the most common definition, owing to its computational simplicity, i.e. its straightforward manner in computing the distance at each iteration [43,44]. The probability for each y_r to be the next is calculated based on Eq. (3), as in line 4. The y_r that has the highest probability (farthest distance) from the defined centroids is selected to be the next centroid. The core idea in K-means++

Clustering Pseudo Code Using K-means++ and K-means

Input: Number of cluster (K), set of Requirements' RPV (Y) = $\{y_1, .y_2, , y_r\}$
Output: Three clusters of requirements (low, medium, high)
Start
1 Choose the first cluster centroid (c_1) uniformly at random from Y
2 **Repeat**
3 For each y_r, compute the distance y to the nearest cluster centroid c_i using $E(y_r) = \sqrt{(c_i - y_r)^2}$
4 Choose the next cluster centroid c_i , selecting $c_i = y_r \in$ Y with probability $\frac{E(y_r)^2}{\sum_{y \in Y} E(y)^2}$
5 **Until** all the total of cluster centroids have been chosen
6 **Do**
7 For each y, compute the distance y_r to each defined cluster centroid c_i using $E_{ir} = \sqrt{(c_i - y_r)^2}$
8 Assign each y to the cluster with the closest centroid c_i.
9 Update each cluster centroids c_i by taking the average of the all assigned y_r in each cluster.
10 **While** (no longer changes in the cluster centroids).
11 **End**

Fig. 3. Clustering pseudo code using K-means++ and K-means.

is to select centroids one-by-one in a controlled fashion, where the set of initialised centroids bias the choice of the next centroid stochastically, as K-means++ tries to select from the far-away specified clusters' centroids.

$$E(y_r) = \sqrt{(c_i - y_r)^2} \quad (2)$$

In the above,
c_i is the centroid of the ith cluster;
y_r is the RPV of the rth requirement in the set Y; and
$E(y_r)$ is the distance y_r to the nearest cluster centroid c_i.

$$P(y_r) = \frac{E(y_r)^2}{\sum_{y \in Y} E(y)^2} \quad (3)$$

Here,
y_r is the RPV of the rth requirement in the set Y;
$E(y_r)$ is the distance from y_r to the nearest cluster centroid c_i; and
$P(y_r)$ is the probability of selecting y_r to be the next centroid.

After obtaining the three initial clusters' centroids, the K-means algorithm executes the process of clustering, as shown in lines 6–10 of Fig. 3. In line 7, the distance from y to each defined centroid c_i is computed using the Euclidean distance in Eq. (4). Based on the distance calculation, each y_r is grouped to the cluster with nearest centroid, as shown in line 8. In line 9, each defined cluster centroid c_i is recalculated by taking the average of all of its assigned y_r values using Eq. (5). The steps in lines 7 to 9 are repeatedly executed until there will no further changes in the cluster centroid assignments. Then, the final clustering is presented by clustering the requirements into three clusters with respect to their RP values. With the use of the K-means++ initialisation mechanism for the centroids, K-means is able to produce an accurate requirement clustering result with fewer iterations, as the K-means++ mechanism has been proven to enhance the performance of K-means in terms of accuracy and speed [43].

$$E_{ir} = \sqrt{(c_i - y_r)^2} \quad (4)$$

In the above,
c_i is the centroid of the ith cluster;
E_{ir} is the distance y_r to the ith cluster centroid;
y_r is the RPV of the rth requirement that belongs to the ith cluster centroid c_i; and
m_i is the number of all assigned y_r values in the ith cluster centroid c_i.

$$c_i = \frac{1}{m_i} \sum_{y_r \in c_i} y_r \quad (5)$$

Here,
c_i is the centroid of the ith cluster;
y_r is the RPV of the rth requirement that belongs to ith cluster centroid c_i; and
m_i is the number of all of the assigned y_r values in the ith cluster centroid c_i.

3.2.3. Applying the binary search tree

Several methods are used to sort the requirements, such as binary priority List, BST, bubble sort, spanning tree matrix and AHP. The BST has better effectiveness in dealing with a large number of data that need to be sorted in a ranked list with fewer comparisons. This advantage leads to less time consumption compared with alternative methods such as the bubble sort, binary priority list, spanning tree matrix and AHP, as proven in mathematical complexity analysis and empirical experimental comparison documented in [13, 17, 24, 47]. The complexity of the total number of BST comparisons is O(log n) in the best case, where the BST is balanced, and O(n Log n) in the unbalanced BST, where n is the number of requirements [13, 17, 24, 47]. Meanwhile, the total number of comparisons using AHP or bubble sort is equivalent to (n * (n-1)/2), which is technically difficult to implement and consumes considerable time [13, 17, 24, 47]. However, a drawback of using the BST in RP is a simplified ranking of requirements (without revealing to what extent each requirement is more essential than another), as no priority value is assigned to each requirement [5,13]. In SRPTackle, the priority value of each requirement 'RPV' is obtained from the previous steps of the technique. Thus, the BST is selected to rank the requirements in each cluster. Fig. 4 illustrates the pseudocode of the BST algorithm, which presents the application steps of the BST for producing a ranked list of requirements on the basis of their RPV.

The BST is executed first on the requirements in the high cluster category to produce a ranked list of requirements to be implemented in

BST Algorithm Pseudo Code

```
    Input: Un-order presentation of the requirements RPV(x)
    Output: Descending presentation of the requirements RPV
    Starts
1   Create the tree by selecting x randomly from the requirements' RPV list as root node value
2     Do
3     Take another x requirement's RPV
4       If x is less than the root node value
5       Compare x with left subtree node value.
6         If the subtree is empty
7         Create a leaf node with new requirement's RPV
8         Else
9         If requirement's RPV is less than left subtree node value
10            add the x to the left of the left subtree
11        Else
12            add the x to the right of the left subtree
13        End
14      Else if x is greater than or equals to the root node value
15      Compare x with right subtree value.
16        If the subtree is empty
17        Create a leaf node with new requirement's RPV
18        Else
19        If x is less than right subtree value
20            add the x to the left of the right subtree
21        Else
22            add the x to right of the right subtree.
23        End
24      End
25    While (requirements RPV list != null )
26    If Tree != empty
27      Recursively print right subtree
28      Visit node
29      Recursively print left subtree
30    End
31  End
```

Fig. 4. Binary search tree pseudo code.

the high cluster, as the priority of the requirements in the high cluster are greater than those in the other two clusters (middle and low clusters). This is followed by the middle cluster, which contains requirements with higher priorities than those in the low cluster. The low cluster is sorted last. The process starts with step in line 1 of Fig. 4, by randomly selecting one requirement's RPV as the root node. Then, the steps from lines 4 to 25 are executed recursively, to add each requirement's RPV. Each requirement's RPV is selected and compared to the root node. If the requirement's RPV is less important than the root node, it is compared to the left subtree node value. If the requirement's RPV is less than the root node value, it is compared to the right subtree value; otherwise, it is compared to the left subtree. If the node has no subtree, the requirement's RPV is inserted as the new leaf node. Otherwise, the requirement's RPV is added to the right or left of the subtree; if it is less, then it is added to the left, and if not, it is added to the right. This process is repeated until all requirements' RPVs have been compared and inserted in the BST. The steps in lines 26 to 30 show the process for sorting the requirements in a ranked list with a descending mode, where the sorting process is executed by traversing through the entire BST and recursively printing the right subtree, followed by visiting the root and then recursively printing the left subtree.

4. SRPTackle-tool

The SRPTackle-Tool automation implementation tool is developed to automate the prioritisation in the SRPTackle technique. The SRPTackle-Tool is constructed in the C# programming language with a .NET environment, HTML language for the graphical user interface (GUI), and a Microsoft SQL Server as the database engine. Fig. 5 shows the process of implementing the prioritisation in the SRPTackle technique using the SRPTackle-Tool. The SRPTackle-Tool is comprised of two main components: the GUI, and the automation engine. These two components contain two phases: an input phase, and a process and output phase. The details of these phases are presented as follows.

1 Phase 1, input phase: In this step, the files (in Excel format) of the list of the requirements, initial requirements' weighting values, and SPV of participating stakeholders are uploaded into the GUI of the SRPTackle-Tool, to be stored in the constructed database. The

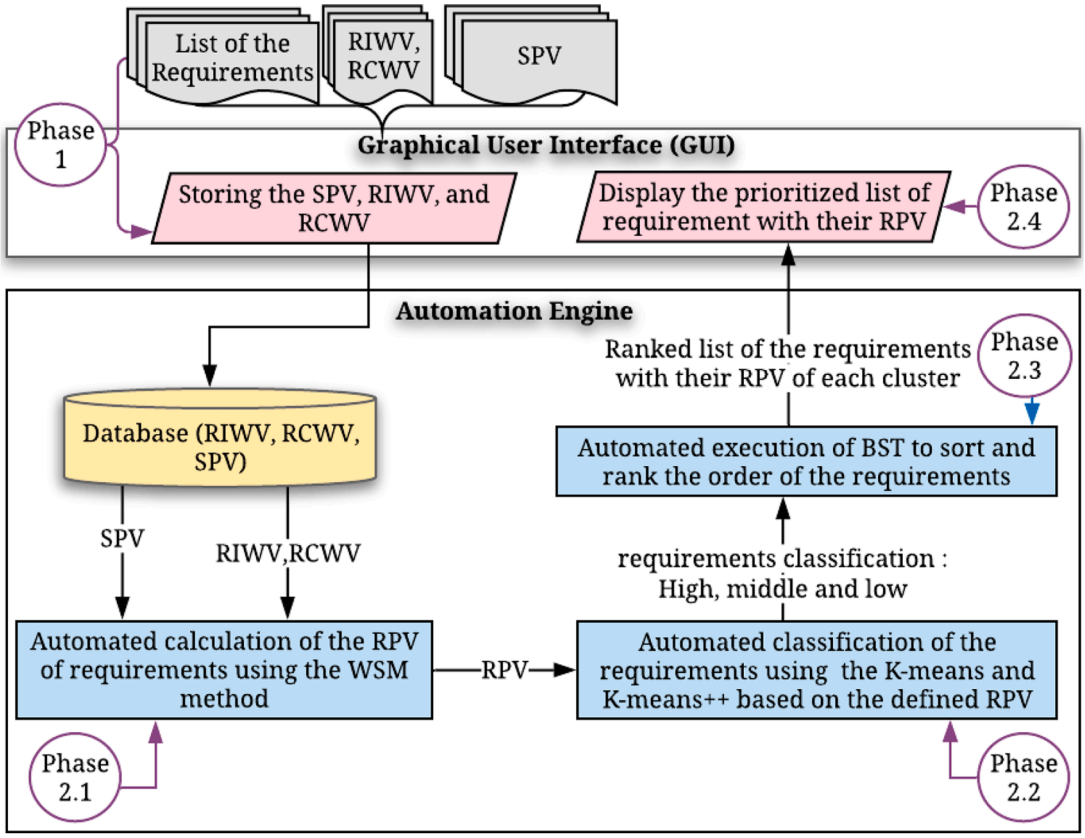


Fig. 5. Implementation structure of the SRPTackle-tool.

requirements list file contains the ID and title for all requirements that need to be prioritised. The file for the requirement weighting values contains the weighting values of each requirement (RIWV and RCWV), and should be provided by the stakeholders who are involved in prioritisation process of those requirements. The SPV file includes the SPV of participating stakeholders is calculated by executing the process of StakeQP-AIT as described in the StakeQP study [6].

2 Phase 2, process and output phase/Execution of the post-prioritisation phase of the SRPTackle technique: After uploading and storing the above-mentioned files, the SRPTackle-Tool directly implements the steps of the post-prioritisation phase, and displays the results via the following automated steps:

- 2.1 Automated calculation of the RPV for each requirement: The SPV of participating stakeholders in RP process, RIWV, and RCWV are extracted from the constructed database to automatically calculate the RPV of each requirement. The details of the RPV calculation were illustrated in Section 3.2.1).
- 2.2 Automated classification of the requirements: Based on the specified RPV, the requirements are automatically classified into three clusters (high, medium, low) by employing of the K-means++ and K-means algorithms, as described distinctly in Section 3.2.2.
- 2.3 Automated execution of BST: To sort and rank the order of the requirements in each cluster and produce the prioritised list of requirements, the BSTs are executed automatically. This step was described in Section 3.2.3.
- 2.4 Presentation of the result: The prioritised list of the requirements, along with their specified RPVs, are exposed on the GUI of the SRPTackle-Tool.

5. Experimental studies

In this section, we elaborate the precise definition and design of the experiments on the basis of the standard guidelines proposed by Wohlin et al. [48] on how to report and document experimentations in software engineering. Table 1 shows an overview of the design of the experiments, in which the key elements of the experiments are summarised. The following subsections will articulate in detail the reported key elements.

5.1. Experiment definition

The experiments were motivated by the goal of evaluating the effectiveness of SRPTackle compared with that of StakeRare [11], Lim et al. GA [28], Saffron [27] and OSA [49]. The effectiveness of a prioritisation technique represents its capability to produce rapid and accurate prioritisation results. Hence, the evaluation is conducted to measure the accuracy of results and time consumption. Correspondingly, the research questions of our experimentation are as follows:

Table 1
Overview of the design of the experimentation.

Goal	Analysing the techniques for software requirement prioritisation: SRPTackle, StakeRare [11], Lim et al. GA [28], Saffron [27] and optimal solution analysis (OSA) [49], with the goal of measuring the accuracy of results and time consumption.
Independent variables	SRPTackle and existing RP techniques used: StakeRare, Lim et al. GA, Saffron and OSA.
context	ralic benchmark dataset: 122 requirements of ralic industrial software project, including 49 general requirements and 73 specific requirements.
dependant variables	accuracy and time consumption

- Research question 1 (RQ1): Are the prioritisation results produced by SRPTackle more accurate (compared with the ground truth) than the prioritisation results produced by StakeRare, Lim et al. GA and Saffron techniques?
- Research question 2 (RQ2): Is SRPTackle less time-consuming than Lim et al. GA and OSA?

5.2. Hypothesis formulation

Corresponding to the formulated research questions, the following null hypotheses are proposed:

- $H1_{0Accuracy}$: The accuracy of the SRPTackle and a particular technique (StakeRare, Lim et al. GA and Saffron) are the same.
- $H2_{0Time}$: SRPTackle's time requirement to produce the prioritisation list is the same as that required by a particular technique (Lim et al. GA and OSA).

If the null hypothesis can be rejected with comparatively high conviction, then an alternative hypothesis can be formulated as follows:

- $H1_{1Accuracy}$: The accuracy of the SRPTackle and a particular technique (Lim et al. GA and OSA) are not the same.
- $H2_{1Time}$: SRPTackle's time requirement to produce the prioritisation list is not the same as that required by a particular technique (Lim et al. GA and OSA).

5.3. Variables and measures

The independent variables of our experiments are SRPTackle, whereas the alternative techniques include StakeRare [11], Lim et al. GA [28], Saffron [27] and OSA [49]. These specific techniques are considered relevant to SRPTackle, as they were validated in terms of suitability to the RALIC dataset, which will be used as a benchmark dataset in our experiments. These specific techniques were evaluated using medium- and large-size requirements as in the RALIC dataset and the same accuracy measurement method (to find the accuracy result) to assess the performance as the present study in the accuracy evaluation process. To the best of our knowledge, the specific techniques for comparison are the most common techniques that secured the best results published using the RALIC benchmark dataset.

In our experiment, two dependant variables were considered: time consumption and accuracy. These variables are commonly used in evaluating the effectiveness of the RP technique in terms of the capability to work with large-scale data by producing a fast and accurate prioritised list of requirements. The most frequently measured dependant variables in RP are accuracy of results and time needed to perform the prioritisation task. This can be related to the fact that a prioritisation process applicable in commercial software development should be fast and capable of providing accurate results [85].

Time consumption refers to the time consumed by the technique in prioritising the requirements to produce a ranked list of requirements. As can be observed from the constructed hypotheses and research questions, the time consumption performance of SRPTackle is compared with two alternative techniques (i.e. Lim et al.'s GA [28] and OSA [49]) that perform RP with less time consumption (e.g. AHP). Compared with StakeRare and Saffron (which were not included), Lim et al.'s GA and OSA have the same features as SRPTackle in terms of performing RP with a supported automation tool.

As for the accuracy of the dependant variable, SRPTackle was measured to verify its ability to produce an accurate prioritised list of requirements in relation to other selected techniques. The stakeholders' agreement (satisfaction or perception) method and comparison with actual results can be used to measure the technique's accuracy in the RP domain. The former is related to the assessment of accuracy performance by evaluating the produced result from the stakeholders' point of view, where the percentage of stakeholders' agreements (or disagreements) on the produced result are revealed. The latter is conducted by comparing the produced result of the technique with the actual result of the used projects' dataset. In this research, the second method was selected to assess the accuracy performance of SRPTackle. The used RALIC dataset has the actual result of the RALIC project; this result is known as the ground truth of the prioritised requirements and the stakeholders and is built on the basis of a rigorous and systematic process based on the stakeholders' satisfaction or agreements. In addition, the selected method facilitated a fair comparison with other existing techniques that used the RALIC dataset in evaluating their performance against the actual data (ground truth) of RALIC.

Prioritisation accuracy was measured by comparing the ranked prioritised list of requirements produced by a specific technique (SRPTackle, StakeRare, Lim et al.'s GA and Saffron) with the ground truth list of the requirements. The accuracy of requirement prioritisation is the degree of similarity between the prioritised list of requirements (ranked list of the general and specific requirements) produced by a specific technique and the prioritisation in the ground truth list. The ground truth list presents the actual prioritised list of the RALIC requirements (ranked list of the general and specific requirements), as derived from the RALIC project documentation in [50]. The statistical measure of the Pearson correlation coefficient was selected to determine the degree of similarity because it is considered the best method for measuring the association between two or more variables [51]. In addition, it has been used by existing RP techniques for the same purpose, i.e. finding the accuracy of their prioritised result [11,27]. The value of the correlation coefficient (p) ranged between +1 and −1, where +1 refers to a perfect positive correlation, −1 indicates a perfect negative correlation, and 0 refers to no correlation. As shown in Eq. (6), the Pearson correlation coefficient formula was used to calculate the correlation coefficient value (p).

$$P = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (6)$$

In the above equation,

P is the Pearson correlation coefficient value (accuracy performance);

n is the number of samples of requirements in each list (both lists consist of the same number of requirements);

x_i is the rank for requirement i in the list of SRPTackle; and

y_i is the rank for requirement i in the ground truth list.

5.4. Objects

The experiments were conducted using a benchmark dataset of the RALIC industrial software project. The RALIC benchmark dataset project is considered a large-scale software project [22,50]. This project was initiated and developed as a new access control system at University College London. The documentation reports of the RALIC project are provided in the RALIC dataset and in the thesis of Soo Lim, which can be accessed in [11, 22, 50]. To the best of our knowledge, the RALIC benchmark dataset is one of the first available and complete datasets in the RP field and contains a large set of requirements that need to be prioritised.

Compared with other datasets, the RALIC benchmark dataset contains detailed information on numerous requirements and stakeholders in the RALIC project. The ground truths that present the actual priority ranks of the RALIC requirements, can be used to evaluate performance accuracy. Hence, the RALIC dataset has been used to evaluate some existing RP techniques, as in [11, 27, 28]. On this basis, the RALIC dataset was used in this study to evaluate SRPTackle and achieve a fair performance comparison between SRPTackle and existing RP techniques. A list of 122 requirements is provided in the truth table of the RALIC requirements, and it presents the actual results of the

prioritisation of the requirements [11,50]. These requirements were categorised as general and specific requirements, where each general requirement may have its own specific requirements. A total of 49 general and 73 specific requirements was noted. In this research, the medium and large requirement sets of the RALIC dataset were used to evaluate the performance of SRPTackle.

5.5. Subjects

A total of 127 individuals were reported as stakeholders of the RALIC project [11]. However, as stated in the documentation reports of the RALIC benchmark dataset project, 87 stakeholders were identified to be involved in the prioritisation process of the RALIC requirements by providing initial rating value for system requirements [11, 22, 50]. The full details of the stakeholders' profiles and the requirements' ratings from each participating stakeholder are reported in the documentation reports of the RALIC project, which can be obtained in [11, 22, 50]. To conduct a systematic and fair performance evaluation and comparison amongst SRPTackle, StakeRare, Lim et al.'s GA, Saffron and OSA, the experiments were conducted with considering the participation of the same 87 stakeholders whose profiles and requirements' ratings were considered in the prioritization process of the conducted experiments in this study.

5.6. Experiment execution

As shown in Table 2, seven experiments were conducted using the RALIC requirements. The experiments aimed for a systematic and fair performance evaluation and comparison amongst SRPTackle, StakeRare, Lim et al.'s GA, Saffron and OSA. These experiments were executed separately, and each experiment implemented the proposed SRPTackle technique on the basis of the defined steps of SRPTackle-Tool, which was presented in Section 4. Each experiment was conducted within a specific objective and used a certain size of requirement set of the RALIC dataset.

Exp. 1 and Exp. 5 were conducted to compare the performance of SRPTackle with that of StakeRare, Lim et al.'s GA and OSA. The experimental procedure of this experiment follows that of [11, 28, 49] to achieve a fair comparison with the reported results. Exp. 1 was executed using a medium set of requirements (containing 49 general requirements), which is the same size of requirements used in assessing StakeRare, Lim et al.'s GA and OSA. By contrast, Exp. 5 was implemented with the same number of RALIC requirements (large set of requirements, including all of the specific requirements) used in evaluating the performance of StakeRare and Lim et al.'s GA. For Lim et al.'s GA, experiments were conducted with different initialisation values for the GA used in the prioritisation process. However, we selected the best-reported result of Lim et al.'s GA for comparison with the proposed SRPTackle.

Furthermore, Exp. 2, Exp. 3 and Exp. 4 were performed to evaluate SRPTackle and compare its accuracy results with the Saffron technique. These experiments were conducted on the basis of the experimental procedure in [27] to obtain a fair comparison with the results of the Saffron technique published therein. In this manner, Exp. 2, Exp. 3 and Exp. 4 were implemented with large sets of requirements, consisting of 50, 59 and 65 requirements, respectively. By contrast, the aim of constructing Exp. 6 is to assess SRPTackle's performance in comparison with that of Saffron and OSA. Hence, to ensure a fair comparison, Exp. 6 was implemented in accordance with the experimental procedure in [27, 49]. Correspondingly, Exp. 6 was implemented to prioritise 80 general requirements and their specific requirements. Eventually, in accordance with the experimental procedure in [11], Exp. 7 was executed to assess the performance of SRPTackle and compared it with that of StakeRare using a large set of requirements, including the complete list of 122 requirements (i.e. 49 general requirements and their 73 specific requirements).

Table 2
Experimental details.

Experiment No	Size of requirement set	Number of requirements	RALIC requirement types
Experiment 1 (Exp. 1)	Medium	49	General requirements
Experiment 2 (Exp. 2)	Large	50	Specific requirements
Experiment 3 (Exp. 3)	Large	65	Specific requirements
Experiment 4 (Exp. 4)	Large	70	Specific requirements
Experiment 5 (Exp. 5)	Large	73	Specific requirements
Experiment 6 (Exp. 6)	Large	80	General requirements and their specific requirements
Experiment 7 (Exp. 7)	Large	122	General requirements and their specific requirements

6. Experimental results

In this section, we analyse and summarise the results obtained from the conducted experiments to answer the research questions presented in Section 5.1. We performed statistical analysis using SPSS 22 [52] to test the stated hypotheses. A two-tailed one-sample *t*-test was used to test the defined hypotheses. Considered to be the most recommended significance level by researchers and scientists [12,53], 5% (0.05) statistical significance level (P) was set for hypothesis testing. The selection criteria of 5% (0.05) significance level are presented in Table 3.

6.1. RQ1: are the prioritisation results produced by SRPTackle more accurate (compared with the ground truth) than the prioritisation results produced by stakerare, Lim et al.'s GA and saffron?

Table 4 presents a comparison of the accuracy results from SRPTackle with those from three alternative techniques (StakeRare, Lim et al.'s GA and Saffron) for each conducted experiment; the Pearson correlation coefficient was used, as described in Section 5.3. In Exp.1 and Exp.5, the accuracy performance of SRPTackle is compared with that from StakeRare and Lim et al.'s GA. The accuracy performance of these two techniques was evaluated with the same number of RALIC requirements as stated in Section 5.5.

The accuracy results in Exp. 1 and Exp. 5 (as depicted in Table 4) reveal that the proposed SRPTackle has better accuracy performance than the other two techniques. For Exp. 1 and Exp. 5, the accuracy results of SRPTackle are 0.9465 and 0.9371, respectively; those for StakeRare are 0.50 and 0.71, respectively; those for Lim et al.'s GA are 0.9228 and 0.9135, respectively. The accuracy performance of SRPTackle was also compared with the Saffron technique, using the same number of requirements as in Exp. 2, 3, 4 and 6 for SRPTackle, as described in Section 5.5. The efficiency of SRPTackle is higher than that of the existing technique; the accuracy results for SRPTackle are 0.9399, 0.9381, 0.93 and 0.9392 for Exp. 2, Exp. 3, Exp. 4 and Exp. 6, respectively, whereas those for Saffron are 0.9231, 0.8156, 0.7669 and 0.7669, respectively.

Table 5 presents the statistical results of the *t*-test of the SRPTackle in comparison with each specific technique (StakeRare, Lim et al.'s GA,

Table 3
Selection criteria.

P-value Criteria	Result
$p < 0.05$	Reject null hypothesis
$p \geq 0.05$	Do not reject null hypothesis

Table 4
Accuracy results produced by SRPTackle, StakeRare Lim et al.'s GA and saffron.

Experiment	Technique	Accuracy Result
Exp. 1	StakeRare	0.50
	Lim et al.'s GA	0.9228
	SRPTackle	**0.9465**
Exp. 2	Saffron	0.923
	SRPTackle	**0.9399**
Exp. 3	Saffron	0.8156
	SRPTackle	**0.9381**
Exp. 4	Saffron	0.7669
	SRPTackle	**0.93**
Exp. 6	Saffron	0.6756
	SRPTackle	**0.9392**
Exp. 5	StakeRare	0.71
	Lim et al.'s GA	0.9135
	SRPTackle	**0.9371**
Exp. 7	StakeRare	0.6050
	SRPTackle	**0.9449**

Table 5
Results of t-tests for accuracy.

Technique	Mean	Std. Deviation	P-value
StakeRare	0.6050	0.1050	**0.010**
SRPTackle	**0.9428**	0.0050	**0.000**
Lim et al. GA	0.9185	0.0064	**0.003**
SRPTackle	**0.9418**	0.0066	**0.003**
Saffron	0.7953	0.1031	**0.001**
SRPTackle	**0.9364**	0.0043	**0.000**

Saffron) based on the accuracy results of conducted experiments in Table 4. As shown in Table 5, the obtained P values of SRPTackle and specific techniques are less than the 0.05 significance level. Thus, we can conclude that the first null hypothesis ($H1_{0Accuracy}$) is rejected at a significance level of 0.05, and the accuracy of SRPTackle is significantly higher than that of StakeRare, Lim et al.'s GA and Saffron.

Additionally, Fig. 6 depicts the average improvement percentages of SRPTackle relative to each of the selected techniques, with respect to the accuracy performance. In measuring the average accuracy improvement percentages, the accuracy improvement percentage is firstly measured on the basis of the obtained experimental accuracy results of each technique in each experiment, as presented in Table 4. Eq. (7), which is well-known for measuring the improvement percentage of performance testing for a technique [15, 54–56], was used to calculate the accuracy improvement percentages.

$$AIP_{i,j} = \frac{PT_j - ET_{i,j}}{ET_{i,j}} \times 100 \quad (7)$$

In the above equation,

$AIP_{i,j}$ is the accuracy percentage improvement of the SRPT technique against the i^{th} specific technique in the j^{th} experiment;

PT_j is the accuracy result of the proposed SRPTackle technique in the j^{th} experiment; and

$ET_{i,j}$ is the accuracy result of the i^{th} specific technique in the j^{th} experiment.

Subsequently, the average accuracy improvement of SRPTackle with respect to each of the specific technique on all experiments was calculated, as illustrated in Fig. 6. The accuracy efficiency of SRPTackle is 2.58%, 59.16% and 19.28% better than that of StakeRare, Lim et al.'s GA and Saffron, respectively. Additionally, the overall average performance of SRPTackle against all selected techniques demonstrates that the accuracy performance of SPRTackle is generally better than that of StakeRare, Lim et al.'s GA and Saffron in terms of accuracy at a percentage of 27.01%.

6.2. RQ2: is SRPTackle less time consuming than Lim et al.'s GA and OSA?

Fig. 7 presents a comparative time consumption performance of SRPTackle and two alternative techniques (Lim et al.'s GA and OSA) for producing the final prioritised list of requirements. In particular, the time consumption of SRPTackle is 5.02 s to prioritise 49 requirements of Exp. 1, whereas those for OSA and Lim et al.'s GA are 25.89 and 200 s, respectively. For Exp. 5, SRPTackle consumes 5.48 s, whereas Lim et al.'s GA consumes 200 s. For Exp. 6, the time consumption of SRPTackle is 5.52 s, whereas that for OSA is 40.64 s. As shown in Fig. 7, the performance of SRPTackle is more effective than that of the other two techniques, insofar as consuming less time for prioritising the medium and large sets of requirements.

We applied *t*-test to test the second null hypothesis ($H2_{0Time}$). Table 6 depicts the t–test results based on the time consumption results of the conducted experiments (Fig. 7). The t-test's results in Table 6 demonstrate that the hypothesis $H2_{0Time}$ should be rejected at a 0.05 significance level because the obtained P-values are less than a 0.05 significance level. The results also reveal that on the average, SRPTackle requires less time than Lim et al.'s GA and OSA. Hence, the statistical analyses evidently demonstrate that SRPTackle is a faster technique in performing prioritisation task than Lim et al.'s GA and OSA.

7. Discussion

Throughout this section, the potential phenomena that could articulate the trends achieved in the experiments' analysis and comparisons are distinctly discussed, and the base of the proposed SRPTackle technique is clarified.

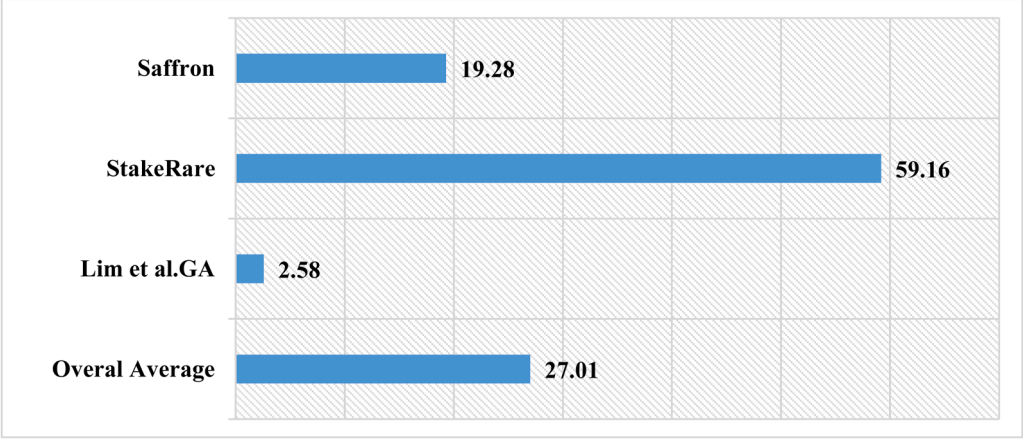


Fig. 6. Overall improvement percentages.

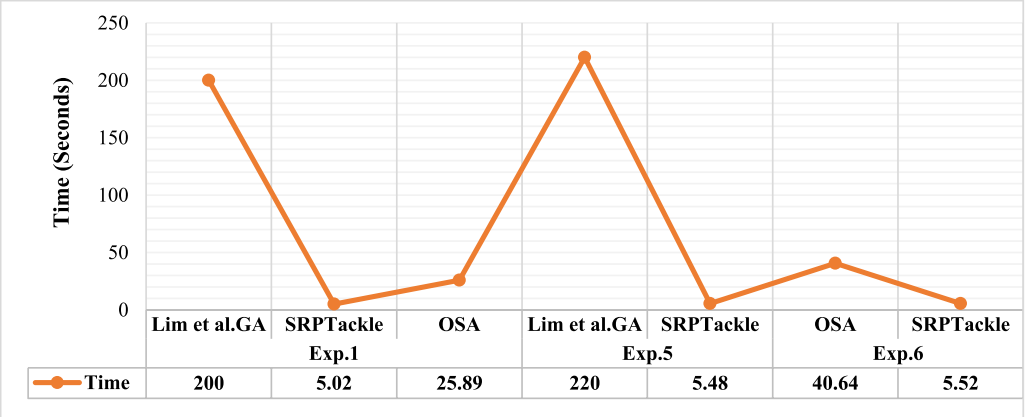


Fig. 7. Time Consumption performance of SRPTackle, Lim et al.'s GA and OSA for producing the final prioritised list of requirements.

Table 6
Results of t-tests for time consumption.

Technique	Mean	Std. Deviation	P-value
Lim et al.GA	210.00	14.1421	**0.030**
SRPTackle	**5.25**	0.3252	**0.028**
OSA	33.265	10.4298	**0.139**
SRPTackle	**5.27**	0.3535	**0.030**

In terms of accuracy performance, the statistical analyses of the experimental results demonstrate that the accuracy of SRPTacle is better than that of StakeRare, Lim et al.'s GA and Saffron. This result can be attributed to the fact that SRPTackle is comprehensive, and it minimises the risk caused by a lack of experience. In SRPTackle, the formulation of the RPV led to not being excessively dependant on experts for identifying the RPV values of the requirements. The RPV formulation is constructed by applying the WSM method on the basis of the input requirement weights given by the stakeholders and the specified SPV of stakeholders by StakeQP. In contrast to existing techniques, the SPV of each participating stakeholder in the RP process is considered and executed in SRPTackle via the new SQP technique. The quantification and prioritisation of the participating stakeholders using StakeQP facilitates the identification of an accurate a stakeholder's influence (SPV of each participating stakeholder) because StakeQP provides the ability to identify an accurate list of SPVs for the stakeholders whilst minimising the need for expert participation in measuring the SPV, as validated in [6]. Then, the identified SPVs of the stakeholders are used in specifying the RPV of each requirement. Thus, measuring the influence of each stakeholder amongst the participating stakeholders accurately contributes to generate the accurately prioritised list of requirements in comparison with StakeRare, Lim et al.'s GA and Saffron. Consequently, SRPTackle can conduct the RP process whilst evaluating the influence of the various types of involved stakeholders in RP, in which the stakeholders have different influences on the success of development projects and have limited and competing resources. In this regard, classifying the requirements and producing the prioritised list thereof on the basis of the identified RPVs via the clustering algorithm (K-means and K-means++), along with the BST, allows the SRPTackle technique to function without being heavily reliant on expert intervention in assigning the RPV for each requirement or in classifying, prioritising and generating the ranked list of requirements. By contrast, the RP processes of existing RP techniques depend too heavily on the involvement of professional expertise and require deep knowledge to initiate the process, thus negatively influencing the reliability of the techniques in producing accurate results due to the bias induced by the judgement of the expert in making various decisions and cases where expertise is deficient [5, 6, 15].

The experimental results show that the efficiency of SRPTackle is superior to Lim et al.'s GA and OSA in terms of the time consumption for producing the prioritised list of requirements. This observation is true even though some of these techniques are implemented with semi-automated execution types. This phenomenon could be attributed to the semiautomated process that was adopted herein to conduct the RP process with the use of the developed automation tool (SRPTackle-Tool). Executing the process of the SRPTackle technique with the automation SRPTackle-tool facilitates the automatic prioritisation, i.e. eliminating the manual process. Additionally, the ability of StakeQP to perform the SQP process for participating stakeholders with less time consumption reduces the time consumption for quantifying and prioritising the stakeholders in the prioritisation process of the SRPTackle. By contrast, Lim et al.'s GA and OSA heavily rely on professional expertise in conducting the SQP process by specifying the SPV values of stakeholders on the basis of the manual inputs of the experts. Moreover, the utilisation of the speed features of the clustering algorithms in categorising and prioritising the requirements minimises the time required to produce a ranked list of requirements in comparison with Lim et al.'s GA and OSA. These latter techniques perform pairwise comparisons and/or rely on experts to perform the process, thus consuming a considerable amount of time. Therefore, SRPTackle requires less time than either Lim et al.'s GA or OSA.

Furthermore, certain factors, such as the number of comparisons, time, lack of automation and overreliance on expert involvement, play key roles in the scalability issue for most existing techniques [5, 13, 17]. For instance, in various RP techniques, such as the bubble sort, AHP and pairwise comparison, the prioritisation process is conducted by evaluating the relative priorities between pairs of requirements [5, 13, 24]. The number of comparisons increases dramatically as the number of requirements increases, making the prioritisation process highly complex and tiring. Consequently, the scalability of the prioritisation process is affected [10, 13, 24]. Additionally, a manual process with heavy reliance on experts to perform the process and the calculations to identify the relative priority value of each requirement can be complex and time consuming [13, 17, 57]. Hence, having to manage hundreds of requirements makes these RP techniques unmanageable [5, 13, 17]. In addition, when conducting an RP process, a scalable RP technique should work with a large set of requirements without consuming considerable amount of time and producing accurate results [58]. In SRPTackle, the combination of the RPV formulation function, K-means, K-means++, BST and the developed automation tool addresses the lack of scalability to a good extent. In particular, the results illustrate the ability of this technique to handle a large set of requirements efficiently, i.e. with reduced time consumption and improved accuracy in comparison with other existing scalable techniques. The RPV formulation is used to calculate the priority value of each requirement using the WSM

method based on the input requirement weights given by the stakeholders, specified SPV of stakeholders, without performing any pairwise comparisons in formulating the RPV. This approach is beneficial because pairwise comparisons disrupt the effectiveness of dealing with a large set of requirements as the number of requirements increases and raises issues of complexity in terms of implementing prioritisation and time consumption. Moreover, using the BST and clustering algorithm (K-means++ and K-means) in SRPTackle not only reduces expert biases by minimising the experts' participation in generating the ranked list of requirements but also enables SRPTackle to work with a large set of requirements. The K-means algorithm provides the ability to cluster numerous requirements in the RALIC dataset into three specified clusters on the basis the specified RPV of each requirement; such ability is shown in the experimentation results with K-means++ initialisation.

Additionally, the BST is used in this research for sorting requirements. The effectiveness of the BST in managing numerous requirements to be sorted in a prioritised list, with fewer comparisons as compared to the AHP and bubble sort techniques, makes SRPTackle more effective in dealing with a large set of requirements. With the use of 122 RALIC dataset requirements in the experiment, the complexity of the total number of comparisons for BST is O(log n) in the best case, where the constructed BST is balanced, and O(n Log n) with an unbalanced BST, where n is the number of the requirements [17, 24, 59, 60]. The total number of comparisons using AHP or bubble sort is equal to (n * (n-1)/2), which is difficult to implement practically and consumes a considerable amount of time [15,24].

Insofar as StakeRare, OSA, Lim et al. GA, and Saffron, these techniques are found to be less are found to be less time consuming and more accurate in producing prioritisation results as compared to other techniques, such as AHP. However, the statistical results of the conducted experiments demonstrate the ability of SRPTackle to generate accurate results with less time consumption. Moreover, SRPTackle has salient properties in comparison with these alternative techniques in terms of being more effective in addressing the need for substantial professional participation in implementing the RP process; the alternative techniques are heavily dependant on good expertise to initiate and execute the prioritisation process of the technique.

8. Threats to validity

Experiment-based research is often subject to different types of validity threats (i.e. conclusion, internal, construct and external validity) [48]. In this research, we attempted to minimise and eliminate these threats as much as possible; however, a few of these threats are beyond our control.

External threats: Threats to external validity rise when experiments cannot be generalised to various forms of real-world problems. The threat here is that we cannot guarantee that the used benchmark constitutes all types of real-world applications in software development sectors. To repress this threat, we select a benchmark dataset of an actual large software project (RALIC), which is a well-known benchmark dataset in the RP domain. To the best of our knowledge, the benchmark dataset of RALIC is one of the realistic, available and complete datasets in RP and SQP domains; it includes detailed information of numerous requirements and stakeholders of the RALIC project. Thus, the RALIC benchmark dataset is commonly utilised for evaluations and selected from real developed software system projects that are initiated and developed as new access control system at University College London. However, SRPTackle should be tested in additional industrial projects that are related to the different types of software project practices to enhance external validity.

Internal threats: Threats to internal validity relates to factors which affect experiments without our awareness and/or which are beyond our control. One threat to internal validity comes from the measurement of time consumption for prioritising the requirements for each technique; such measurement is highly subjective to the running environment. Thus, the implementation of all techniques must be conducted in the same prioritisation environment. To minimise this threat, we compared the time consumption performance of SRPTackle with the tools of the three techniques evaluated with the RALIC dataset and with the same number of requirements as the present study. The comparison could present an indication for prioritisation time; however, another threat here is related to the implementation of language differences in the techniques' tools. In addition, although the performance of these techniques' tools can be influenced by the specifications of the machine (e.g. desktop or laptop) used to run the tools, the execution of the tools compared herein were conducted on a machine with specifications defined by the authors to be the minimum specifications required to execute the tools efficiently.

Construct validity: Construct validity threats are related to the application–theory relationship. One of the construct validity threats arises from automation process issues. The automation process, although timesaving and with minimal to no human intervention, may elicit an issue of producing an unpredictable processing error or a low-quality result in case the implementer has not followed the process implementation structure of SRPTackle. The reason is that the automated machine cannot execute a flexible variety of tasks because it is restricted to execute the task on the basis of what it has been programmed to do. Similarly, the automation process of SRPTackle is implemented in the full process of prioritising the requirements on the basis of the defined criteria that are used to obtain RIWV and RCWV, and the specified SPV of the participating stakeholder. Hence, the SRPTackle automation process has the capability of completing the process accordingly if the RIWV, RCWV, and the SPV have been upload on the basis of the stated process implementation structure of the SRPTackle. To reduce this threat, the full process of the SRPTackle is distinctly elaborated to assist the implementer in obtaining the SPV, RIWV and RCWV on the basis of the defined criteria. Thus, the implementer is recommended to read the given elucidation of the SRPTackle implementation process judiciously. Another threat here is related to the unforeseen costs that would be needed to keep the SRPTackle-tool's processes up to date because the costs disbursed in upgrading with a new protocol would entail high operating costs in relation to the research and development that needs to be conducted.

Conclusion validity: Conclusion threats involve the relationship between the treatment and the outcome. The threats here are related to the conducted comparison with existing RP techniques. Within RP, various techniques are compared. However, we were unable to compare our approach with all these techniques due to different reasons, such as the unavailability of the source code of these techniques for public use. To mitigate this threat, we compared the performance of the proposed SRPTackle technique with those techniques considered to be the most relevant to SRPTackle, as these selected techniques were evaluated using the RALIC dataset with the same size of requirements and the same accuracy measurement method as the present study. Moreover, these compared techniques execute their own SQP processes during the prioritisation of requirements similar to SRPTackle. To the best of our knowledge, the above-mentioned techniques and benchmarks for comparison are the best results published so far using the RALIC benchmark dataset.

9. SRPTackle Managerial contributions

Based on the performance evaluations, it is evident that SRPTackle can introduce a number of contributions to the managerial side in the development process of software system projects. SRPTackle is one of the first techniques to conduct the RP process without being exceedingly reliant on the participation of human expertise. By using the clear implementation details from the constructed RPV formulation function for specifying the RPV of each requirement, the classification algorithm using K-means and K-means++, and the BST for classifying and prioritising the requirements, a project manager can produce a prioritised list

of requirements with minimal expert participation. Thus, SRPTackle can enable the project manager to minimise cost expenses in project development that are typically required, e.g. contracting an expert to initiate and execute the RP process.

Additionally, as SRPTackle can generate more accurate prioritisation results than the alternative techniques insofar as producing a prioritised list of requirements with classification levels and RPV values for large-scale projects, the project manager can determine highly important requirements that need to be implemented early in the project development process. This will consequently assist the project manager in optimising the usage of limited resources effectively during the development process, and in constructing an effective plan for the financial implications, requirements, and staged deliveries. This allows for the expansion of projects and excellent outputs, increasing the likelihood of securing a successful system project.

Furthermore, with the SRPTackle features of time effectiveness, the ability to scale well with a large number of requirements, automation, and clear implementation guidelines, the project manager can perform the RP process for projects with large-scale requirements in a professional and proper manner, without necessitating an extensive amount of effort (e.g. time workloads, tiring tedious manual processes, the need for the involvement of professional expertise, computational complexity, and likelihood of human errors). Eventually, with the presence of the SRPTackle technique and its automation tools, the development process of a software system project will possess a lower likelihood of failure from shortcomings, such as a shortage of expertise, an omission of important requirements, biased results of the RP, and limited constraints (e.g. time and budget constraints).

10. Conclusion and future directions

In this research, a new semi-automated RP technique (SRPTackle) and automation implementation tool (SRPTackle-Tool) were proposed to address challenges in existing RP techniques, such as scalability, excessive reliance on expert intervention, time consumption, a lack of automation, and a lack of a SQP process for evaluating the stakeholder impact in prioritising the requirements. The proposed SRPTackle provides a semi-automated process for prioritising a large set of requirements without a manual process, while minimising the need for expert intervention in assigning the priority values to requirements, classifying the requirements, conducting the SQP process, and producing a ranked list of requirements. The proposed SRPTackle technique is based on a combination of the RPV formulation function using the WSM method, clustering algorithms (K-means and K-means++), and the BST. The WSM method utilised to specify the RPV value of each requirement is based on the defined SPV of each stakeholder and the assigned initial weight value of the requirement that is obtained from the participating stakeholders. To classify the requirements into three defined levels (high, medium, and low), the K-means and K-means++ algorithms were employed to classify the requirements based on their specified RPVs. Lastly, the BST algorithm was employed to sort the requirements, and produce the prioritised list of requirements. Seven experiments were conducted to assess the performance of SRPTackle with medium and large sets of requirements from the RALIC benchmark dataset. The findings demonstrate that SRPTackle can handle a large set of requirements and produce results that are more accurate in less time, and is more effective in addressing the defined RP limitations as compared to the other existing RP techniques.

Through the conducted experiments and comprehensive literature exploration in this research, several limitations have been revealed, leading to future trends that can be suggested to extend this research. A potential future trend for improving SRPTackle performance is catering to the independencies of the requirements. Handling requirement interdependencies is another important consideration in RP [5, 13, 23]. The proposed SRPTackle assumes that all the requirements are independent and places concerns regarding interdependencies as future work. Moreover, as revealed herein, most existing RP techniques fail to address requirement interdependencies. Thus, the SRPTackle technique is recommended to handle the dependencies amongst requirements automatically, especially with a large set of requirements.

Other future trends can focus on extending the implications of SRPTackle to different project datasets. In this research, SRPTackle was applied to the RALIC benchmark dataset, which is from a large actual software project. However, with limited resources and other constraints in accessing other benchmark datasets in the RP and SQP domains, we did not apply the proposed SRPTackle technique to other project datasets. Thus, we suggest expanding the implications of SRPTackle to different datasets of software projects. Additionally, the implications of the proposed technique in various global software project practices are desirable for improved applicability due to the encouraging evaluation results obtained herein.

CRediT authorship contribution statement

Fadhl Hujainah: Methodology, Investigation, Conceptualization, Writing - original draft, Software, Writing - review & editing, Validation, Formal analysis. **Rohani Binti Abu Bakar:** Conceptualization, Resources, Funding acquisition. **Abdullah B. Nasser:** Resources, Funding acquisition, Visualization. **Basheer Al-haimi:** Formal analysis, Visualization, Data curation. **Kamal Z. Zamli:** Validation, Writing - review & editing, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors appreciate the efforts of the Ministry of Education Malaysia, Universiti Malaysia Pahang (UMP) and Ministry of Higher Education Yemen for supporting this research. This work is supported in part by the Fundamental Research Grant from the Ministry of Education Malaysia under Grants: RDU1901209 FRGS/1/2019/ICT02/UMP/02/13, RDU190164, and the PGRS170393 Grant from UMP. The authors also give special thanks to Professor Soo Ling Lim for her kind support in providing the benchmark dataset for this work.

Reference

[1] F. Gomariz-Castillo, I. Garrigós, J.-A. Aguilar, J. Zubcoff, S. Casteleyn, J.-N. Mazón, Evaluating different i*-based approaches for selecting functional requirements while balancing and optimizing non-functional requirements: a controlled experiment, Inf. Softw. Technol. 106 (2019) 68–84, https://doi.org/10.1016/j.infsof.2018.09.004.

[2] J. Medeiros, A. Vasconcelos, C. Silva, M. Goulão, Requirements specification for developers in agile projects: evaluation by two industrial case studies, Inf. Softw. Technol. 117 (2020), https://doi.org/10.1016/j.infsof.2019.106194.

[3] F.A. Bukhsh, Z.A. Bukhsh, M. Daneva, A systematic literature review on requirement prioritization techniques and their empirical evaluation, Comput. Stand. Interfaces. (2020), 103389, https://doi.org/10.1016/j.csi.2019.103389.

[4] R.C. Motta, K.M. de Oliveira, G.H. Travassos, A conceptual perspective on interoperability in context-aware software systems, Inf. Softw. Technol. 114 (2019) 231–257, https://doi.org/10.1016/j.infsof.2019.07.001.

[5] F. Hujainah, R.B.A. Bakar, M.A. Abdulgabber, K.Z. Zamli, Software requirements prioritisation: a systematic literature review on significance, stakeholders, techniques and challenges, IEEE Access 6 (2018) 71497–71523, https://doi.org/10.1109/ACCESS.2018.2881755.

[6] F. Hujainah, R.B.A. Bakar, M.A. Abdulgabber, StakeQP: a semi-automated stakeholder quantification and prioritisation technique for requirement selection in software system projects, Decis. Support Syst. 121 (2019) 94–108, https://doi.org/10.1016/j.dss.2019.04.009.

[7] F. Sher, D.N.A. Jawawi, R. Mohammad, Requirements prioritization aspects quantification for value-based software developments, J. Theor. Appl. Inf. Technol. 97 (2019) 3969–3979.

[8] L. Alawneh, Requirements prioritization using hierarchical dependencies, in: S. Latifi (Ed.), Inf. Technol. - New Gener., Springer International Publishing, Cham, 2018, pp. 459–464.

[9] H. Tufail, I. Qasim, M.F. Masood, S. Tanvir, W.H. Butt, Towards the selection of optimum requirements prioritization technique: a comparative analysis, in: 5th Int. Conf. Inf. Manag. ICIM 2019, 2019, pp. 227–231, https://doi.org/10.1109/INFOMAN.2019.8714709.

[10] I. Ibriwesh, S.-.B. Ho, I. Chai, Overcoming scalability issues in analytic hierarchy process with ReDCCahp: an empirical investigation, Arab. J. Sci. Eng. (2018) 1–17, https://doi.org/10.1007/s13369-018-3283-2.

[11] S.L. Lim, A. Finkelstein, StakeRare : using social networks and collaborative filtering for large-scale requirements eliciation, IEEE Trans. Softw. Eng. 38 (2012) 707–735.

[12] F. Shao, R. Peng, H. Lai, B. Wang, DRank: a semi-automated requirements prioritization method based on preferences and dependencies, J. Syst. Softw. 126 (2017) 141–156, https://doi.org/10.1016/j.jss.2016.09.043.

[13] P. Achimugu, A. Selamat, R. Ibrahim, M.N. Mahrin, A systematic literature review of software requirements prioritization research, Inf. Softw. Technol. 56 (2014) 568–585, https://doi.org/10.1016/j.infsof.2014.02.001.

[14] M. Sufian, Z. Khan, S. Rehman, W. Haider Butt, A systematic literature review: software requirements prioritization techniques, in: Proc. - 2018 Int. Conf. Front. Inf. Technol. FIT 2018, IEEE, 2019, pp. 35–40, https://doi.org/10.1109/FIT.2018.00014.

[15] M.I. Babar, M. Ghazali, D.N. a. Jawawi, S.M. Shamsuddin, N. Ibrahim, PHandler: an expert system for a scalable software requirements prioritization process, Knowl.-Based Syst. 84 (2015) 179–202, https://doi.org/10.1016/j.knosys.2015.04.010.

[16] M. Aasem, M. Ramzan, A. Jaffar, Analysis and optimization of software requirements prioritization techniques, in: Int. Conf. Inf. Emerg. Technol., 2010, pp. 1–6, https://doi.org/10.1109/ICIET.2010.5625687.

[17] Q. Ma, The effectiveness of requirements prioritization techniques for a medium to large number of requirements : a systematic literature review, Master's Thesis, School of Computing and Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand, 2009.

[18] F. Hujainah, R.B. Abu Bakar, B. Al-haimi, M.A. Abdulgabber, Stakeholder quantification and prioritisation research: a systematic literature review, Inf. Softw. Technol. 102 (2018) 85–99, https://doi.org/10.1016/j.infsof.2018.05.008.

[19] S... Forouzani, R... Ahmad, S... Forouzani, N... Gazerani, Design of a teaching framework for software requirement prioritization, in: Proc. - 2012 8th Int. Conf. Comput. Technol. Inf. Manag. ICCM 2012, 2012, pp. 787–793. http://www.scopus.com/inward/record.url?eid=2-s2.0-84867025210&partnerID=40&md5=347940a3a8a79a5eb31fb427897ced76.

[20] M.I. Babar, M. Ghazali, D.N.A. Jawawi, K. Bin Zaheer, StakeMeter: value-based stakeholder identification and quantification framework for value-based software systems, PLoS ONE 10 (2015) 1–33, https://doi.org/10.1371/journal.pone.0121344.

[21] F. Hujainah, R.B.A. Bakar, B. Al-Haimi, M.A. Abdulgabber, Investigation of stakeholder analysis in requirement prioritization techniques, Adv. Sci. Lett. 24 (2018) 7227–7231.

[22] S.L. Lim, D. Quercia, A. Finkelstein, StakeNet: using social networks to analyse the stakeholders of large-scale software projects, in: Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng. - ICSE '10, 2010, pp. 295–304, https://doi.org/10.1145/1806799.1806844.

[23] F. Hujainah, R.B.A. Bakar, M.A. Abdulgabber, Investigation of requirements interdependencies in existing techniques of requirements prioritization, Teh. Vjesn. 26 (2019) 1186–1190, https://doi.org/10.17559/TV-20171129125407.

[24] J. Karlsson, C. Wohlin, B. Regnell, An evaluation of methods for prioritizing software requirements, Inf. Softw. Technol. 39 (1998) 939–947, https://doi.org/10.1016/S0950-5849(97)00053-0.

[25] A. Perini, A. Susi, P. Avesani, A machine learning approach to software requirements prioritization, IEEE Trans. Softw. Eng. 39 (2013) 445–461, https://doi.org/10.1109/TSE.2012.52.

[26] P. Tonella, A. Susi, F. Palma, Interactive requirements prioritization using a genetic algorithm, Inf. Softw. Technol. 55 (2013) 173–187, https://doi.org/10.1016/j.infsof.2012.07.003.

[27] S.A. Asif, Z. Masud, R. Easmin, A.U. Gias, SAFFRON : a semi-automated framework for software requirements prioritization, Int. J. Adv. Comput. Sci. Appl. 8 (2017) 491–499.

[28] S. Lim, M. Harman, A. Susi, Using genetic algorithms to search for key stakeholders in large-scale software projects, in: Aligning Enterp. Syst. Softw. Archit., 2012, pp. 118–134, https://doi.org/10.4018/978-1-4666-2199-2.ch007.

[29] A. Hudaib, R. Masadeh, A. Alzaqebah, WGW: a hybrid approach based on whale and grey wolf optimization algorithms for requirements prioritization, Adv. Syst. Sci. Appl. 18 (2018) 63–83, https://doi.org/10.25728/assa.2018.18.2.576.

[30] R. Qaddoura, A. Abu-Srhan, M.H. Qasem, A. Hudaib, Requirements prioritization techniques review and analysis, in: 2017 Int. Conf. New Trends Comput. Sci., 2017, pp. 258–263, https://doi.org/10.1109/ICTCS.2017.55.

[31] M. Yousuf, M.U. Bokhari, M. Zeyauddin, An analysis of software requirements prioritization techniques: a detailed survey, in: Int. Conf. Comput. Sustain. Glob. Dev., 2016, pp. 3966–3970.

[32] N.M. Carod, A. Cechich, A classification framework for software requirements prioritization approaches, Rev. Colomb. Comput. 10 (2009) 3283–3285, https://doi.org/10.1007/978-3-642-04425-0_55.

[33] F. Hujainah, R.B. Abu Bakar, B. Al-Haimi, A.B. Nasser, Analyzing requirement prioritization techniques based on the used aspects, Res. J. Appl. Sci. 11 (2016) 327–332, https://doi.org/10.3923/rjasci.2016.327.332.

[34] R.B. Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, R. Feldt, A. Aurum, Prioritization of quality requirements: state of practice in eleven companies, in: Proc. 2011 IEEE 19th Int. Requir. Eng. Conf. RE 2011, 2011, pp. 69–78, https://doi.org/10.1109/RE.2011.6051652.

[35] A.A. Zaidan, B.B. Zaidan, M. Hussain, A.M. Al-Haiqi, M.L. Mat Kiah, M. Abdulnabi, Multi-criteria analysis for OS-EMR software selection problem: a comparative study, Decis. Support Syst. 78 (2015) 15–27, https://doi.org/10.1016/j.dss.2015.07.002.

[36] Z. Chourabi, F. Khedher, A. Babay, M. Cheikhrouhou, Multi-criteria decision making in workforce choice using AHP, WSM and WPM, J. Text. Inst. 110 (2019) 1092–1101, https://doi.org/10.1080/00405000.2018.1541434.

[37] F. Tscheikner-Gratl, P. Egger, W. Rauch, M. Kleidorfer, Comparison of multi-criteria decision support methods for integrated rehabilitation prioritization, Water (Switzerland) 9 (2017), https://doi.org/10.3390/w9020068.

[38] A. Ishizaka, P. Nemery, Multi-Criteria Decision Analysis Methods, Wiley, 2013.

[39] M.E. Celebi, H.A. Kingravi, P.A. Vela, A comparative study of efficient initialization methods for the k-means clustering algorithm, Expert Syst. Appl. 40 (2013) 200–210, https://doi.org/10.1016/j.eswa.2012.07.021.

[40] L. Rokach, O. Maimon, Clustering methods, in: O. Maimon, L. Rokach (Eds.), Data Min. Knowl. Discov. Handb., Springer, Boston, MA, 2010, pp. 321–352, https://doi.org/10.1007/0-387-25465-X_15.

[41] E. Alhroob, M.F. Mohammed, C.P. Lim, H. Tao, A critical review on selected fuzzy min-max neural networks and their significance and challenges in pattern classification, IEEE Access 7 (2019) 56129–56146, https://doi.org/10.1109/access.2019.2911955.

[42] E. Alhroob, N.A. Ghani, Fuzzy min-max classifier based on new membership function for pattern classification: a conceptual solution, in: 2018 8th IEEE Int. Conf. Control Syst. Comput. Eng., 2018, pp. 131–135, https://doi.org/10.1109/ICCSCE.2018.8685029.

[43] D. Arthur, S. Vassilvitskii, K-Means++: the advantages of careful seeding, in: Proc. Eighteenth Annu. ACM-SIAM Symp. Discret. Algorithms, 2007, p. 1027, https://doi.org/10.1145/1283383.1283494. –1025.

[44] S.Z. Selim, M.A. Ismail, K-means-type algorithms: a generalized convergence theorem and characterization of local optimality, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-6 (1984) 81–87, https://doi.org/10.1109/TPAMI.1984.4767478.

[45] R. Xu, D. Wunsch, Survey of clustering algorithms, IEEE Trans. Neural Networks. 16 (2005) 645–678, https://doi.org/10.1109/TNN.2005.845141.

[46] D. Xu, Y. Tian, A comprehensive survey of clustering algorithms, Ann. Data Sci. 2 (2015) 165–193, https://doi.org/10.1007/s40745-015-0040-1.

[47] V. Ahl, An experimental comparison of five prioritization methods – investigating ease of use, accuracy and scalability, Master's Thesis, School of Engineering, Blekinge Institute of Technology, Sweden, 2005.

[48] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Springer-Verlag, Berlin Heidelberg, 2012.

[49] V. Veerappa, Clustering methods for requirements selection and optimisation, Ph. D. Thesis. School of Computer Science and Engineering, University College London, London WC1E 6BT, UK, 2012.

[50] S. Lim, Social networks and collaborative filtering for large-scale requirements elicitation, Ph.D. Thesis, School of Computer Science and Engineering, University of New South Wales, Sydney, Austrailia., 2010. http://discovery.ucl.ac.uk/1329883/.

[51] J.D. Evans, Straightforward Statistics For the Behavioral Sciences, Thomson Brooks/Cole Publishing Co, 1996.

[52] IBM Corp., IBM SPSS Statistics for Windows, Armonk, 22, NY IBM Corp., 2013, p. 0.

[53] J.F. Hair, C.M. Ringle, M. Sarstedt, PLS-SEM: indeed a silver bullet, J. Mark. Theory Pract. 19 (2011) 139–152, https://doi.org/10.2753/MTP1069-6679190202.

[54] D. De Angelis, Y. Grinstein, Relative performance evaluation in CEO compensation: evidence from the 2006 disclosure rules, Johnson Sch. Res. Pap. Ser. (2011), https://doi.org/10.2139/ssrn.1710386, 39-2010.

[55] F.M. Tice, Explicit relative performance evaluation and managerial decision-making : evidence from firm performance and investments, 2017. https://papers.ssrn.com/abstract=2645956.

[56] G. Gong, L.Y. Li, J.Y. Shin, Relative performance evaluation and related peer groups in executive compensation contracts, Account. Rev. 83 (2011) 1007–1043, https://doi.org/10.2308/accr.00000042.

[57] F. Hujainah, R.A. Bakar, E. Alhroob, B. Al-haimi, A.B. Nasser, Interrelated elements in formulating an efficient requirements prioritization technique: review, in: 2020 10th IEEE Int. Conf. Control Syst. Comput. Eng., 2020, pp. 97–101, https://doi.org/10.1109/ICCSCE50387.2020.9204955.

[58] N. Kukreja, B. Boehm, S.S. Payyavula, S. Padmanabhuni, Selecting an appropriate framework for value-based requirements prioritization, in: 2012 20th IEEE Int. Requir. Eng. Conf., 2012, pp. 303–308, https://doi.org/10.1109/RE.2012.6345819.

[59] R. Beg, Q. Abbas, R.P. Verma, An approach for requirement prioritization using B-tree, in: Proc. - 1st Int. Conf. Emerg. Trends Eng. Technol. ICETET 2008, 2008, pp. 1216–1221, https://doi.org/10.1109/ICETET.2008.158.

[60] G. Kaur, S. Bawa, A survey of requirement prioritization methods, Int. J. Eng. Res. Technol. 2 (2013) 958–962.