WhaleRank: an optimisation based ranking approach for software requirements prioritisation

R. Vijay Anand* and M. Dinakaran

SITE, VIT University,
Vellore, Tamilnadu, India
Email: vijayanand.r@vit.ac.in
Email: dinakaran.m@vit.ac.in
*Corresponding author

Abstract: Requirement prioritisation is one of the major areas in the software product development process. Ranking methods employed for prioritising orders the requirements based on their importance. Ranking the requirements contributes in enhancing the quality of the product using additional features and to attain customer satisfaction. However, effectiveness is the growing concern in requirement prioritisation. This paper proposes a ranking method WhaleRank to rank the requirement and the effectiveness of ranking gets enhanced using the whale optimisation algorithm (WOA). The WhaleRank method uses four ranking functions based on dictionary words, similarity measure, perception of the manager, and the newly updated requirements that are combined to form a linear rank using the ranking constants. WOA determines the optimal weights of the ranking constants that promote to determine the optimal rank for the requirements. Experimentation with the methods like CBRank, average rank, WhaleRank, and GA provide a comparative performance analysis that proves that the proposed WhaleRank outperforms all methods in terms of accuracy and disagreement measure (NDA) and the values of accuracy and NDA is 83.33% and 16.24% respectively.

Keywords: WhaleRank; whale rank optimisation; WOA; similarity matrix; requirement updates; weighed ranking constants.


Biographical notes: R. Vijay Anand received his BTech in Information Technology from Madras University, Chennai, India in 2004, and MTech in Information Technology from VIT University, Vellore, India in 2006. He has been in the software industries (HCL, CISCO) field for more than eight years and currently working as an Assistant Professor (Senior) in VIT University. Currently, he is pursuing his PhD at VIT University, Vellore, India. He has published more than seven research papers in reputed international journals. His research interests include software testing, software engineering, data structures and software quality.

M. Dinakaran completed his BTech (IT), MTech (IT-Networking) in Vellore Institute of Technology and PhD in Anna University, Chennai, Tamil Nadu and India. He worked in TATA Consultancy Services as an Assistant System Engineer from September 2006 to July 2009. Currently, he is working as an Associate Professor and Head of the Department of IT, School of Information Technology, VIT University.
1 Introduction

Information technology plays a significant role in our day-to-day life and software engineering is one of the major branches. Software engineering is an interesting field, which aims at designing software to meet the consumer demands. There are a lot of challenges in the field of software engineering by considering the degree of the software products (Ejnioui et al., 2013). In most of the software products, one can find different categories of the stakeholders, who aim at receiving their requested software product that prioritises their requirements at the final delivery. Any of the following members like the users, customers, project managers, product managers, developers, and testers are the stakeholders. They hold positions in software engineering, and therefore, their requirements for the products are based on their immediate need or emergency (Ejnioui et al., 2013). However, the complexity of the software system gets increased, and therefore, the practitioners face extra pressure to develop software with the demanded requirement and in addition, to complete the task within the predefined time. This pressure makes the software requirement prioritisation, a prime importance and it is one of the best solutions to avoid the trade-offs between the conflicting requirements. In other words, one can say that the requirement prioritisation plays a vital role in the decision-making process corresponding to software development. The importance of the requirement prioritisation is highlighted in a number of research studies that pointed the problems of requirements prioritisation in the software engineering domain over the years (Dabbagh et al., 2016).

Particularly, requirements prioritisation plays a vital role in every aspect of our life mainly, in the project selection process, requirement finalisation, module development and design, software testing, implementation, and in turn, in the post-implementation processes (Aasem et al., 2010). Thus, it is very clear that the requirement prioritisation is the complex decision-making process (Perini et al., 2013). The objective of requirement prioritisation is that it identifies the most significant requirement of the software from the list and extracts the significant knowledge of the system to impact the effectiveness (Perini et al., 2009). The most important points to consider during the requirement prioritisation include: identifying the important requirement from the given activities, to determine the order of the requirements to implement during the software release, to achieve the settlement regarding the scope of the project with their existing limitations. The limitation list includes the schedule, budget, time, and quality. In addition, the objective aims at establishing equilibrium between the cost and benefit of a specific requirement and therefore, to accept this equilibrium balance as possible equilibrium to implement. Moreover, the objective includes the reduction of the impacts caused by any one of the requirements over the architecture of the software and the final deliveries mainly, on the cost of the product. Finally, to meet the optimum satisfaction of the customers and to determine the factors that meets the customer satisfaction (Forouzani et al., 2012).
WhaleRank: an optimisation based ranking approach for software

Since prioritising the requirements is the important concern in planning the release, there are various methods to undergo prioritising. Some of the methods include analytical hierarchy process (AHP), the weighted sum method and cost-value analysis. In situations when the requirements are not dependent on each other, scheduling the release planning is an easy task that duly depends on the priorities generated using the above techniques. These techniques establish the priority based on two criteria, namely available resources and the delivery date (Shao et al., 2017). Additionally, Case-Based Ranking (CBRank) provides the following solutions. It formally defines the prioritisation problem to solve, and then describes the machine learning method used. Then, it characterises the prioritisation process that gets supported using CBRank, important points of the method gets assessed through a comprehensive overview of the empirical measurements, and finally, CBRank gets positioned based on the state-of-the-art of the requirement prioritisation methods (Perini et al., 2013). However, CBRank faces a lot of problems with the scalability, and the accuracy reduces while ranking the requirements. When comparing AHP and the CBRank, CBRank is advantageous as it yields better accuracy and appears as a good method of handling the trade-off between the decision makers (Perini et al., 2009).

This paper concentrates on software requirement prioritisation that uses an efficient prioritising method named as WhaleRank. It uses four ranking functions based on the dictionary words, a similarity matrix, the perception of the manager, and the newly updated requirements. This novel ranking method WhaleRank overcomes the problems faced by the existing ranking methods like AHP (Perini et al., 2009) and CBRank (Perini et al., 2013). These ranking functions form four different ranks, which then combines to form a linear rank function. The linear rank function is the combination of the four rank functions that gets combined using the ranking constants. These ranking constants when summed equals to one. The ranking constants are optimally determined using the WOA. WOA determines the optimal position, and it is a population-based algorithm, and it overcomes the performance of PSO, ACO, and so on. Finally, the best rank order gets retrieved using the fitness function. This method of ranking based on WhaleRank achieves customer satisfaction by efficiently prioritising the requirements based on the customer review (Pimentel et al., 2017). It is clear that this method contributes to enhance the quality of the product.

The contribution of the paper is as follows:

1. **WhaleRank method**: the main contribution of the paper is the WhaleRank approach of ranking the requirements uses the WOA for effectively ranking the requirements.

2. **Ranking functions**: the major contribution is that this paper uses four ranking functions. Mainly, a new ranking function is introduced that define the newly updated requirements in the ranking process. The other ranking functions depend on the dictionary words, similarity matrix, and the perception of the manager.

The organisation of the paper is as follows: Section 1 introduces the background, motivation, and contributions of the work. Section 2 presents the review of the existing methods and the challenges faced by them. The proposed method is introduced in Section 3, and it depicts the importance of the proposed method, Section 4 provides the results and discussion of the proposed method with the comparative analysis, and Section 5 concludes the paper.
2 Motivation

2.1 Literature review

This section presents the existing methods involved in ranking the requirements. The prioritisation techniques include the machine learning techniques, tool-supported requirements prioritization methods, and so on.

Perini et al. (2013) proposed a method for a requirements prioritisation called CBRank that combined the project’s stakeholder’s preferences with requirements ordering approximations and was computed using the machine learning techniques. This method improved the accuracy, and it supported the adaptive elicitation process. However, effectiveness gets affected in the complex real settings.

Perini et al. (2009) presented an empirical study that aimed at evaluating the state-of-the-art tool-supported requirements prioritisation methods, AHP and CBRank. It focussed on three measures, namely ease of use, consumption time, and the accuracy. Comparative results proved that the ease of use and the consumption time for the CBRank was better than for AHP whereas the accuracy of the AHP was better than CBRank. However, the shortcoming was that it did not consider cost factor for prioritisation.

Shao et al. (2017) proposed a method named as the drank for prioritising requirements based on the dependencies. This method developed an attributes tree that made the prioritisation work easier. It used the RankBoost for the calculation of the prioritisation based on the stakeholder’s preferences, and finally, an algorithm based on the weighted PageRank was used that analysed the dependencies persisting between the requirements. However, DRank supports only the contribution dependencies and business dependencies between the requirements.

Misaghian and Motameni (2016) proposed a tensor decomposition method that considered the simultaneous effect that existed between the functional, non-functional requirements and stakeholders, which was found to have different preferences using a three-order tensor with a multi-way analysis. In order to evaluate the method, controlled experiment was carried out. The proposed method attained good quality compared with the state-of-the-art. But it required human interference in prioritising the preferences.

Dabbagh et al. (2016) conducted two experiments that aimed at evaluating the prioritising approaches. Comparison of the IPA approach with the HAM-based approach in terms of three parameters, namely time-consumption, accuracy, and ease of use proved that IPA outperformed other approaches, but there was no hierarchical relationship between the functional and the non-functional requirements.

Khan et al. (2016) proposed RePizer, a framework for prioritisation of software requirements. It assisted the engineer's inaccurate decision-making through the retrieval of the historical data. It provided an overview of the entire project. It improved the decision-making process, but it lacked an automatic system for selecting the requirements.

Babar et al. (2015) proposed an expert system named as the Priority Handler (PHandler) for requirement prioritisation. This method used a technique known as the value-based intelligent requirement prioritisation technique, neural network and analytical hierarchical process for improving the scalability of the method. However, it required domain knowledge that remained a disadvantage of this method.
Easmin et al. (2014) proposed a scheme for prioritising the requirements to address the feedback issue that impacted on true ranking. This approach used the stakeholder’s feedback using the binary search technique. The method arranged the requirements based on the customer requirement and decreased the stakeholder’s feedback through the reduction of the ranking difference. However, dynamism was the major shortcoming of this method.

2.2 Challenges

The main challenge is the domain knowledge for prioritising the requirements, which raises the need for the human interference in the decision-making process (Babar et al., 2015), this, in turn, poses the need for the automatic selection (Khan et al., 2016) of performing the prioritising technique.

Another problem associated with the prioritising technique is regarding the large set of requirements (Babar et al., 2011). The existing prioritising techniques solve only the small sets of the requirement under the name as toy applications, but the problem is with handling large sets of requirements (Ahuja et al., 2017).

The major problem faced is the prioritising time and ease of use. Human interference in the process of prioritising increases the time to prioritise and in addition increases the cost involved in prioritising (Inoki et al., 2014).

The most significant challenge faced using the CBRank method of prioritising is regarding the order accuracy, and it restricts the number of requirements. In the case of the CBRank, the number of requirements is limited to ten, which stands a major issue (Perini et al., 2009).

3 Proposed methodology: WhaleRank – a method for prioritising the requirements using the ranking functions

This section depicts the overview of the proposed WhaleRank method of prioritising the software requirements. Requirement prioritisation is the process carried out in the development of software projects in order to determine the importance of certain requirements at the stage of the release. The prioritisation method ensures customer satisfaction and the resources gets utilised in an efficient way. In other words, requirement prioritisation is the order in which the requirements get arranged based on their importance. This method of prioritisation orders the requirements upon the successive delivery of the products that improves the quality and adds value to the product. Prioritisation takes a long way that considers the expectations of the stakeholders, risk, cost, dependencies of the requirements among each other, and so on. The existing method for performing the prioritisation using CBRank (Perini et al., 2013) ranks the requirements of the stakeholder with good accuracy but the effectiveness of the method gets affected with the complex settings, and when new requirements get added to the list, this existing method fails to update the rank for prioritising. Moreover, the existing methods failed to address the non-monotonic conditions availing in the elicitation process and in addition, the ranking methods used are less effective. In order to overcome the abovementioned issues, this paper uses a new method named as the WhaleRank that improves the effectiveness of ranking through the usage of four ranking
functions. WhaleRank not only increases the effectiveness but also it ensures that the optimisation algorithm used here determines the optimum level rank to meet the consumer satisfaction and it enables an effective trade-off between the developer and the stakeholder.

**Figure 1** System overview – requirement prioritisation using WhaleRank-based ranking functions (see online version for colours)

Figure 1 shows the system overview of the proposed method that uses the WhaleRank method. The input to the WhaleRank is the set of documents that carries the requirements of the stakeholder, perception of the manager, and the number of updates performed. The input to the WhaleRank formulates the priority based on four ranking functions that extract the features based on the frequency of the words in the document, pair-to-pair similarity measure, $M_p$, and the number of updates. All these four rank functions provide four different ranking priorities, which gets merged to form a linear rank function using the ranking theory. The WOA determines the best-ranked priority using the iterative search mechanism. It includes a fitness function that gets computed using the mean square error of the original training set and the optimised requirement priority obtained using WOA.

### 3.1 Requirements

This section presents the requirements of the stakeholder. The input to the WhaleRank is the documents carrying the requirements, the perception of the manager, and a number of updates. The requirements get analysed for determining the important requirement, to consider and implement at the time of the release in order to ensure the customer satisfaction and the quality of the software. Documents hold the list of requirements to prioritise that are in the textual format. Let us assume there is $q$ number of requirements represented as:

$$ r = \{r_1, r_2, r_3, \ldots, r_q\}; r_j $$

(1)
where, \( r \) represents the set of all requirements, \( q \) is the total number of requirements and \( 1 \leq i \leq q \). It involves the set of all requirements presented in the text format in the documents. These documents possess the manager perception that holds the low, medium and higher perception. In addition, it holds the number of updates in the requirements, \( R_u \). These inputs with the requirement list, manager perception, and the number of requirement updates undergo ranking operations. The input gets ranked based on the importance of the requirement. The requirement with the highest priority indicates that the particular requirement enhances the quality of the product and hence, it gets included at the time of the release.

3.2 Requirement ranking functions

This section depicts the four ranking functions required to prioritise the requirements. The existing method used for prioritising the requirements of the stakeholder suffers from the issue of unable to update the priorities of the newly updated requirement and it lacked automatic ranking. To overcome the abovementioned issues of the existing methods, this paper employs four ranking functions. The ranking functions use the dictionary words, similarity measure, perception of the manager, and the updates to prioritise the requirements. The dictionary-based ranking determines the rank based on the similar words in the requirements. Likewise, the similarity measure depends on the pair-to-pair similarity obtained using the similarity matrix, manager perception depends on the manual perception of the manager based on the importance of the requirement, and the rank based on requirement updates uses the number of the updated requirements to rank the requirements. Hence, the four ranking functions generate four different rank orders for the requirements. These rank orders get merged to form the single linear rank function that is the sum of the product of the rank coefficients and the rank functions. This linear rank function, which is the combination of all the rank undergoes optimisation search using the WOA that determines the best rank for ranking the requirements.

3.2.1 Dictionary-based ranking function

This ranking method uses the dictionary for ranking the requirements. The requirement gets matched with the dictionary word, and this ranking function determines the most similar requirement in the requirement list while comparing all the documents indicating that the requirement that gets higher priority is more important requirement and it is essential to implement at the time of the release. The ranking function determines the rank based on the common requirements that exist in all the documents. The requirement that is present in most of all the documents indicates that the particular requirement ranks good priority.

\[
R_k (r_i) = \frac{D \cap r_i}{|D|} \tag{2}
\]

Where, \( q \) is the total number of requirements presented for prioritisation, \( D \) is the dictionary word used to determine the similar words, and \( R_k \) is the rank function that computes the priority based on the dictionary words. \( || \) indicates the approximate number of the total dictionary words used in defining the rank to the requirements.
3.2.2 Pair-to-pair similarity measure

The similarity measure gets computed based on the cosine similarity function. It ranks the requirement based on the similar measure through determining the frequency of the requirement. It compares the requirements and updates the value one in case of any similarity between both the requirements. If there is no similarity exists between the requirements, the similarity measure is zero. Ranking depends on the preference of the requirements. In other words, when the preference of any of the requirement is high, the value of similarity gets updated as one else −1. Upon no similarity, the similarity measure becomes zero. Finally, the similarity measure of a requirement gets updated by taking the average of the similarity measure of the requirement along the column. Figure 2 shows the similarity matrix.

**Figure 2** Similarity measure matrix (see online version for colours)

The requirement pair is, $\tau(r_j, r_k)$. The similarity matrix gets filled based on the following condition:

$$
\tau(r_j, r_k) = \begin{cases} 
-1, & \text{when } r_k \prec r_j \\
0, & \text{no order preference between } r_j \text{ and } r_k \\
1, & r_j \prec r_k
\end{cases}
$$

From the above description, $r_j \prec r_k$ denotes that the preference of the requirement $r_k$ is high than the requirement $r_j$. The requirement pair depends on the similarity matrix, and the ranking of the most similar pair depends on the following formula. It is the average of the similarity measure of the requirement column-wise.

$$R_z(r_q) = \frac{1}{q} \sum_{i=1}^{q} P_{i}^{p}$$

Where, $q$ is the number of requirements on the list and $P_{i}^{p}$ is the pair-to-pair similarity between the requirements in the similarity matrix of size ($q \times q$). The similarity matrix gets evaluated, and the rank of the individual requirement gets computed using the
average of the entire column in the similarity matrix. Ranking depends on this average value obtained column-wise. Greater the value of the similarity measure for the requirement, higher is the rank.

3.2.3 Ranking function based on manager perception

This ranking mainly depends on the manager perception. The perception is mainly defined in high, medium, and low. It specifies the requirement based on their importance. The manager orders the requirements based on the stakeholder or customer review. The requirement with the high-level perception gets ranked first, and that requirement gains good consideration for implementation at the time of release. It considers the manager perception, which means the priority given manually based on the importance and value of the requirement.

\[ R_3 (r_q) = \left( \frac{M_p}{\text{Max. fuzzy}} \right) \]  

Where, \( M_p \) is the manager perception, \( r_q \) is the \( q \)th requirement. Maximum fuzzy is nothing but the maximum value of the perception and it retrieves the value of the high-level perception.

3.2.4 Number of updates

In the existing methods like CBRank (Perini et al., 2009), the requirements get prioritised but failed to rank the newly updated requirement. To overcome this issue faced using the existing methods, the proposed WhaleRank plays a significant role in ranking the newly updated requirements. The newly updated requirement gets considered in this framework and gets ranked using the formula through dividing the new update to the maximum updates.

\[ R_4 (r_q) = \frac{|\text{updates}|}{\max_{i=1}^{q}(\text{updates})} \]  

Where, \( r_q \) is the \( q \)th requirement. Based on these ranking functions, the requirements get ranked. Upon each update of the requirement, the importance of the newly updated requirement gets retrieved, and that newly updated requirement gets considered for implementation. The main advantage of the newly updated requirement is that it helps to enhance the features of the product and mainly, contributes in implementing new features to attain good customer satisfaction. As a result of these ranking functions, four ranking gets extracted. These four rankings altogether form a linear ranking function.

3.3 Linear rank function

Linear rank function is the combination of the four rank functions combined using the weighed ranking constants. The ranking constants are \( \alpha, \beta, \gamma, \eta \). The sum of the weights of the ranking constants is one. The linear ranking function is:

\[ R = \alpha R_1 + \beta R_2 + \gamma R_3 + \eta R_4 \]
where $\alpha$, $\beta$, $\gamma$, and $\eta$ are the ranking constants. $R_1$, $R_2$, $R_3$, and $R_4$ are the ranking functions determined using the dictionary words, similarity matrix, perception of the manager, and the newly updated requirements. The function based on the newly updated requirements is the major contribution of the paper. Perception of the manager includes the preference of the customer and their idea regarding the importance of the requirement. The values of these constants get selected based on the optimisation procedure. The optimisation technique used to determine the best rank is the WOA that determines the value of the ranking constant $\alpha$, $\beta$, $\gamma$, $\eta$. The optimal weight of the ranking constants determines the best rank.

3.4 WhaleRank optimisation for prioritising the requirements

This section presents the WOA for prioritising the requirements. The main objective of the WOA is to compute the optimal weights of the ranking constants $\alpha$, $\beta$, $\gamma$, $\eta$ such that it improves the effectiveness of ranking the requirements. Effectiveness in ranking improves the quality of the product. WOA is the population-based meta-heuristic algorithm that depends on the behavior of the humpback whales. WOA overcomes the issues of the existing optimisation algorithms (Mirjalili and Lewis, 2016). The existing optimisation algorithms like PSO, ACO (Perini et al., 2013) provide the best solution only in their path, and some of the iterative solutions are the candidate solutions. In addition, once a new population gets determined, the information of the previous iteration was lost. The advantages of WOA include: it is capable of exploring the search space globally and explores the hot areas in the search space and in addition, it establishes a proper balance between the exploration process and the exploitation process.

3.4.1 Solution encoding

Requirement prioritisation plays a vital role in the prioritising the requirements of the stakeholder to implement them at the time of the release. For enhancing the prioritisation and quality of the product for achieving the customer satisfaction, a better ranking mechanism gets employed. The main aim of this section is to determine the optimal rank for the requirements. The solution encoding chooses the best values for the ranking constants for optimal rank. The solution encoding is as in Figure 3. The linear ranks obtained from the previous steps possess four ranking constants whose sum is one. The ranking constants get determined using the search mechanism and the value that defines the rank efficiently gets considered for ranking the requirements.

Figure 3 Solution encoding of the ranking coefficients (see online version for colours)

where $\alpha$, $\beta$, $\gamma$, and $\eta$ are the ranking constants, and the sum of the ranking constants equals to one.
3.4.2 Flowchart of WOA

This section presents the flowchart of the optimisation method that uses the WOA optimisation algorithm (Mirjalili and Lewis, 2016). WOA includes three phases like encircling, exploitation, and the exploration phases. The main objective of the WOA is about determining the weights of the ranking constants $\alpha$, $\beta$, $\gamma$, $\eta$ that prioritise the requirements to improve the quality of the product. This section discusses the steps of WOA.

1. **Population initialisation**: this is the first step. The whale population gets initialised. Let $W_n$ be the population of whales and the let $q$ be the number of whales in the search with $1 \leq n \leq q$. The whales are the agents that search for the prey based on the global search mechanism.

2. **Random selection of the current best solution**: the first step is the encircling phase. In the encircling phase, the localisation of the prey takes place. In other words, the humpback whales determine the position of the prey for encircling them. In this phase, the humpback whales randomly fix the position of the prey, which remains as the current best solution and the optimal position of the prey. Once the current best position gets assigned, the other search agents update their position with respect to the optimal best solution. The distance between the whale and the current best prey is:

$$
\overrightarrow{O_p} = \overrightarrow{A} \bullet \overrightarrow{P^*(T)} - \overrightarrow{P(T)}
$$

(8)

where $\overrightarrow{O_p}$ is the optimal position of the prey, which is the current best solution assigned. $\overrightarrow{A}$ is the search coefficient vector, $\overrightarrow{P^*(T)}$ is the best current position of the prey. $\overrightarrow{P(T)}$ is the position vector that defines the position of the whale, and the position update of the search agent on the optimal position of the prey is:

$$
\overrightarrow{P(T+1)} = \overrightarrow{P^*(T)} - \overrightarrow{G} \bullet \overrightarrow{A}
$$

(9)

where, $\overrightarrow{A}$ and $\overrightarrow{G}$ is the search coefficient vector, $\overrightarrow{P(T+1)}$ is the updated position of the search agent on the current best solution, and:

$$
\overrightarrow{G} = 2\overrightarrow{s} \bullet \overrightarrow{\bar{r}} - \overrightarrow{s}
$$

(10)

$$
\overrightarrow{A} = 2 \bullet \overrightarrow{\bar{r}}
$$

(11)

where, $\overrightarrow{s}$ is the coefficient vector whose value reduces linearly with increase in the number of iterations, and it takes the value two initially. $\overrightarrow{\bar{r}}$ is the random vector that varies in the interval $0 \leq \overrightarrow{\bar{r}} \leq 1$. Localisation of the search agent in the search space based on the best current position depends on the equation (9). The search for the prey gets extended in the search area using the random vector, $\overrightarrow{\bar{r}}$. The localisation of the various search agents that are present in the search space around the current best solution depends on the coefficient vectors $\overrightarrow{A}$ and $\overrightarrow{G}$ respectively. By varying the values of $\overrightarrow{A}$ and $\overrightarrow{G}$, the different search agents gets localised and the position of the search agents using equation (9). This process enhances the encircling process.
Position update based on the search area: the position update depends upon the global search mechanism. Initially, the location of the prey gets localised. When $p < 0.5$ and within the search space $\bar{G} < 1$, the whale is in the encircling phase, and the position of the search agents gets updated. It depends on the random search mechanism and uses the position update formula in the equation (8) of the encircling phase. When $p < 0.5$ the whale searches the prey globally with respect to the other whales randomly, this is the exploration phase. When $p < 0.5$ and $\bar{G} > 1$, the whale locates the prey within the search space and updates the position using the formula in the exploitation phase.

a Exploitation phase: in the exploitation phase, it describes the bubble-net behaviour of the humpback whales, which is the specific hunting mechanism of the humpback whales. This mechanism is characterised by the bubbles that spiral in the prey path for attacking the prey. In this phase, it discusses two steps, namely the encircling the prey using the bubbles and the spiral position update.

- Step 1: encircling using the bubbles: encirclement phase is attained through the linear decrease of the search coefficient vector $\bar{s}$ from two to zero with the increasing number of iterations. In this paper, $\bar{G}$ is the random coefficient that varies in the interval $[-1, 1]$. In other words, any search agent gets defined anywhere in this random interval. This step locates the prey in the search area.

- Step 2: spiral position update: the spiral position update occurs, once the prey gets encircled. The main purpose of this step is to determine the position of the prey in the search area. Based on this position, the whale develops bubble in the spiral shape that promotes the helical-shaped motion of the whale. The distance between the prey and the whale is:

$$P(T + 1) = O^*_p \cdot e^s \cdot \cos(2\pi l) + P^s(T)$$

$$O^*_p = |P^*(T) - P(T)|$$

where, $O^*_p$ is the current best solution that defines the distance between the whale and the prey, $s$ is a constant that defines the spiral shape of the motion of the whale, and $r$ is the random number, $\cos(2\pi l)$ in the equation denotes the spiral shape, in simple terms, it is the spiral length, $P(T + 1)$ is the current position of the prey, and $P^s(T)$ is the position of the whale, $\cdot$ denotes the element-wise multiplication. The humpback whale encircles the prey and undergoes the attack, and there is a 50% chance of catching the prey. In technical terms, the probability of shifting between the encircling phase and the spiral position update of the whale is 50% during optimisation. Thus, the condition for position update is:

$$\begin{align*}
\bar{P}(T + 1) &= \begin{cases} 
P^s(T) - G \cdot O^*_p & p < 0.5 \\
O^*_p \cdot e^s \cdot \cos(2\pi l) + P^s(T) & p > 0.5
\end{cases}
\end{align*}$$

where, $p$ is the probability value and the value of the probability is $0 \leq p \leq 1$. 

b  *Exploration phase:* in the exploration phase, the random search mechanism of the humpback whales depends on the variation of \( \overline{G} \). Random position depends on the position the humpback whales. In this phase, position update depends on the random search and not on the current best solution. In this exploration phase, the search coefficient vector \( \overline{G} \) uses the value greater than one. Hence, \( |\overline{G}| > 1 \) determines the global search for the prey. In the exploration phase, the position update gets performed based on the global search.

\[
O_p = \left[ A \cdot P - \overline{P} \right]
\]  

(15)

The equation (15) is the optimum position based on random search. \( \overline{P} \) is the random position of the random whale, and \( P \) is the position vector. The position of the prey during global search is:

\[
P(T+1) = \overline{P} - \overline{G} \cdot O_p
\]  

(16)

where, \( P(T + 1) \) is the position of the prey during the global search, \( O_p \) is the distance between the random whale and the random search agent during the global search.

4  *Fitness evaluation:* the fitness of the rank gets determined, and the best rank gets selected based on the fitness formula. For evaluating the fitness, the original training dataset is necessary. The fitness uses the original training dataset with the rank for the requirements, to determine the mean square error between the original training set and the optimised rank that holds the ordered requirement. The value of the fitness function varies between the values zero and one respectively.

\[
F = \text{MSE}(O, P^*(T + 1))
\]  

(17)

where, \( F \) is the fitness function, \( \text{MSE} \) is the mean square error, \( O \) is the original rank, and \( P^*(T + 1) \) is the determined rank using the optimisation algorithm. The fitness of the derived rank gets analysed, and the best rank gets retrieved. The rank obtained for the requirements possess efficient ranking. The requirement with the higher priority gets implemented at the time of the release to attain the customer satisfaction and to improve the quality of the product.

Figure 4 shows the flowchart of the WOA. The first step is the population initialisation, and there are \( q \) number of populations. The parameters like the search coefficient vector \( \overline{G} \), \( \overline{s} \), \( \overline{A} \), random number, \( r \), probability number, \( p \) gets updated for all search agents. Initially, check whether all the whales have the probability of reaching the prey. The parameter \( \overline{G} \) determines the presence of search agents in the search area and the value is limited in the range \(-1 \leq \overline{G} \leq 1\). Locate the prey and determine the position of the prey. This analysis possesses two phases like exploration phase and the exploitation phase. The fitness gets evaluated, and the best position gets updated. In this paper, the best position is the best rank required to prioritise the requirements for the efficient delivery of the product to the customer or the stakeholder.
Figure 4 Flowchart – steps involved in WOA

Figure 5 shows the pseudo code of the proposed method of ranking the software requirements. The main objective of the requirement prioritisation relies on the fact that it ensures the quality of the product and highlights the importance of the requirements such that it ensures the implementation of the important requirement at the time of release to meet the customer satisfaction. Initially, the parameters get initialised and the input, which is the combination of the requirements, the perception of the manager, and newly updated requirements, is provided to rank. There are four rank functions that rank the requirements based on the dictionary word, similarity measure, the perception of the manager, and the newly updated requirements. The main advantage of WhaleRank is that it considers and ranks the newly updated requirements. It does not require any human interference. The rank functions yield four rank that forms a linear rank function using the ranking constants $\alpha, \beta, \gamma, \eta$. The sum of these rank constants equals to one. They get
computed using WOA (Mirjalili and Lewis, 2016). The fitness function determines the best rank for the requirements. This method improves the efficiency, and it enhances the quality of the product.

Figure 5  Pseudocode for WhaleRank (see online version for colours)

<table>
<thead>
<tr>
<th>WhaleRANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  Input: $r_i$; $1 \leq i \leq q$, $M_x$, $R_i$</td>
</tr>
<tr>
<td>2  Output: Best solution $P(T+1)$</td>
</tr>
<tr>
<td>3  Parameters: $r$, $R$</td>
</tr>
<tr>
<td>4  Procedure</td>
</tr>
<tr>
<td>5  Start</td>
</tr>
<tr>
<td>6  Read the requirements, $r_i$.</td>
</tr>
<tr>
<td>8  1. Determine dictionary-based rank, $R_1(r_x)$</td>
</tr>
<tr>
<td>9  2. Compute the pair-to-pair similarity-based rank, $R_2(r_x)$</td>
</tr>
<tr>
<td>10 3. Compute the perception-based rank, $R_3(r_x)$</td>
</tr>
<tr>
<td>11 4. Compute the rank for newly updated requirements, $R_4(r_x)$</td>
</tr>
<tr>
<td>12 Determine the linear Rank $R$</td>
</tr>
<tr>
<td>13 Apply WOA to determine the values of $\alpha$, $\beta$, $\gamma$, $\eta$.</td>
</tr>
<tr>
<td>14 Evaluate the fitness, $F$; $0 \leq F \leq 1$.</td>
</tr>
<tr>
<td>15 Return the best position $P(T+1)$</td>
</tr>
<tr>
<td>16 End</td>
</tr>
</tbody>
</table>

4 Results and discussion

This section depicts the results and discussion of the proposed method with the comparative analysis of the existing methods. The comparison is in terms of accuracy and disagreement measure (NDA) (Perini et al., 2013).

4.1 Experimental setup

The ranking performance of the proposed WhaleRank gets evaluated using the experiment. Experimentation is developed in the personal computer with Intel Core i-3 processor 4GB RAM and Windows 8 operating system in the JAVA environment. The dataset used is the CM1 data (http://promise.site.uottawa.ca/SERepository/datasets-page.html).

4.2 Evaluation metrics

This section presents the metrics taken for analysis. The analysis metrics include the NDA and the accuracy.
4.2.1 NDA

It is the measure of the disagreement measure existing between the order relations of the same requirement sets. It compares the calculated rank with the original rank order. The formula is:

\[ NDA = \frac{1}{U_d} \times TDA(P, Q) \]  

(18)

where, \( NDA \) is the disagreement measure; \( 0 \leq NDA \leq 1 \), \( U_d \) is the original rankset, \( TDA \) is the total disagreement measure.

\[ TDA(P, Q) = \sum_{j=1}^{q} \sum_{k=1}^{q} D_{P,Q}(r_j, r_k) \]  

(19)

\[ D_{P,Q}(r_j, r_k) = \begin{cases} 1 & \text{if } P(r_j) < P(r_k) \text{ & } Q(r_j) > Q(r_k) \text{ or } P(r_j) > P(r_k) \text{ & } P(r_j) < P(r_k) \\ 0 & \text{otherwise} \end{cases} \]  

(20)

where \( P \) and \( Q \) are the order relations and \( D_{P,Q}(r_j, r_k) \) is the disagreement between the requirement pairs. \( j, k \) are the \( q \) number requirements in the input. The disagreement measure determines the accuracy of the computed rank when compared to the target rank.

4.2.2 Accuracy

It is the measure of the accuracy of the computed rank. If the order of the computed rank matches the original order, it returns the value 1, or it returns zero. Formula for accuracy is:

\[ \text{Accuracy} = \frac{1}{q} \sum_{i=1}^{q} \sigma_i \]  

(21)

where

\[ \sigma_i = \begin{cases} 1 & \text{if } R(P) = R(Q) \\ 0 & \text{otherwise} \end{cases} \]  

(22)

where \( q \) is the total number of requirements, \( R(P) \) and \( R(Q) \) are the rank of the requirement relations \( P \) and \( Q \) respectively.

4.3 Methods taken for comparison

The results of the proposed WhaleRank get compared with the existing methods like the CBRank, average rank, and the genetic algorithm (GA). The performance comparison is in terms of the accuracy and the NDA to prove the effectiveness of the proposed method.
4.4 Performance analysis

This section depicts the performance evaluation of the proposed WhaleRank in terms of the performance metrics, and the proposed method gets compared with the existing methods to evaluate the effectiveness of the proposed method.

4.4.1 Performance analysis based on NDA for varying population size

Figure 6 shows the comparative analysis of the ranking methods that depends on the varying size of the whale population. When the population size is four, the values of NDA attained using the methods like WOA, CBRank, GA, and average rank is 20.64%, 21.06%, 21.69%, and 22.61% respectively for 40 numbers of requirements. For 80 requirements, the values of NDA for the abovementioned methods are 18.06%, 21.06%, 21.69%, and 22.61% respectively. It makes clear that the value of NDA is the minimum for the proposed WOA when compared with the other existing methods.

![Comparative graph of NDA for varying population size](image)

When the population size is eight, the values of NDA attained by the methods WOA, CBRank, GA, and average rank with the 60 requirements is 19.11, 20.03, 20.06, and 22.06 respectively. Similarly, when there are 100 requirements, the values of NDA for the methods WOA, CBRank, GA, and average rank are 16.39%, 18.38%, 18.63%, and 19.36% respectively. When the population size gets increased further to 16, the values of
NDA attained using the methods WOA, CBRank, GA, and average rank with 60 requirements attained 19.02%, 20.01%, 19.96%, and 21.53% respectively. The performance comparison reveals that the proposed WOA attained a minimum NDA when compared with the other existing methods, which proved the effectiveness of the proposed method. In addition, it is very clear that with the increase in the population size the NDA value decreases. Hence, the proposed method is the best method compared with the other methods.

4.4.2 Performance analysis based on accuracy for varying population size

This section depicts the performance analysis of the proposed method with the other existing methods in terms of the accuracy. Figure 7 shows the comparative analysis of accuracy for varying population size. For analysis, the population size considered is 4, 8, 12, and 16 respectively. When the population size is 4, the accuracy attained for the methods, WOA, CBRank, GA, and average rank with the number of requirements 40 is 79.65%, 78.29%, 78.16%, and 77.34% respectively. For 100 requirements, the accuracy obtained using the abovementioned methods are 82.94%, 81.37%, 80.93%, and 80.07% respectively. This numerical interpretation makes it clear that the accuracy of the proposed method is high and with the higher number of requirements, the proposed method attains higher accuracy.

Figure 7 Comparative graph of accuracy for varying population size, (a) accuracy with whale population size = 4 (b) accuracy with whale population size = 8 (c) accuracy with whale population size = 12 (d) accuracy with whale population size=16 (see online version for colours)
When the population size is 16, the accuracy attained using the methods WOA, CBRank, GA, and average rank is 80.03%, 78.32%, 78.35%, and 77.63% respectively for 40 requirements. For 80 requirements, the accuracy achieved using the comparative method is 81.37%, 81.03%, 80.19%, and 80.02% respectively. It is clearly evident that the proposed method attained good accuracy compared with the other existing methods. Moreover, with the increasing number of requirements, the accuracy is found to increase. Hence, the proposed WOA possess good accuracy in ranking the requirements.

### 4.5 Discussion

This section discusses the overall comparative performance of the ranking methods WOA, CBRank, GA, and average rank. The parameters namely accuracy and NDA defines the effectiveness of the proposed method. The proposed method outperforms all other existing methods in terms of accuracy. Analysis of the ranking methods for varying population size proves that the proposed methods possess a higher accuracy and minimum NDA compared with the other existing methods. The higher accuracy of 83.33% is attained when the population size is 16 and with 100 requirements. It is evident that with for higher number of requirements, the proposed method achieved good accuracy. Moreover, the NDA of the proposed method is minimum, and the value of NDA is 16.24% when the population size is 16. This NDA value for the proposed method is low compared with the other existing methods.

<table>
<thead>
<tr>
<th>Methods</th>
<th>NDA %</th>
<th>Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n = 4</td>
<td>n = 8</td>
</tr>
<tr>
<td>WOA</td>
<td>16.39</td>
<td>16.39</td>
</tr>
<tr>
<td>CBRank</td>
<td>18.38</td>
<td>18.38</td>
</tr>
<tr>
<td>GA</td>
<td>18.63</td>
<td>18.63</td>
</tr>
<tr>
<td>Average rank</td>
<td>19.36</td>
<td>19.36</td>
</tr>
</tbody>
</table>

### 5 Conclusions

This paper uses a method named as WhaleRank to prioritise the requirements for enhancing the quality of the product at the time of the release. WhaleRank uses four ranking functions, namely the dictionary-based, similarity matrix-based, perception-based, and newly updated requirement-based ranking functions. These rank obtained using the four ranking functions is combined to form a linear rank using four ranking constants. For improving the effectiveness of ranking, this paper uses WOA optimisation algorithm to determine the weights of the ranking constants. The optimal weights of the rank provide the optimal solution for ranking the requirements. Experimental results of the proposed WhaleRank get compared with the existing methods like CBRank, GA, and average rank. The performance analysis in terms of accuracy and NDA prove the effectiveness of the proposed method. This method of ranking the
requirements is effective in improving the quality of the product and hence, brings customer satisfaction. The proposed method outperforms the existing methods, with a minimum NDA and greater accuracy and the value of accuracy and NDP is 83.33% and 16.24% respectively.

References


