Grey Wolf Algorithm for Requirements Prioritization

Raja Masadeh[1], Abdullah Alzaqebah[1] & Amjad Hudaib[2]

[1] Computer Science Department, the World Islamic Sciences and Education University, Amman, Jordan

[2] University of Jordan, Computer Science Department, Amman, Jordan

Correspondence: Abdullah Alzaqebah, 2the World Islamic Sciences and Education University, Computer Center Department, Amman, Jordan. E-mail: abdullah.zaqebah@wise.edu.jo

Abstract

Requirement prioritization is one of the most important approach in the process of requirement engineering due to use it in order to prioritize the execution sort of requirements with taking into account the viewpoints of stakeholders. Thus, in this study, grey wolf optimization (GWO) algorithm is applied in order to prioritize the requirements of a software project. GWO imitates the hunting behavior of grey wolves in nature. Which distinct from others that it has dominant leadership hierarchy which contains four main types; alpha, beta delta and omega wolves. In this paper, a proposed algorithm is presented to prioritize the requirements into ordered list. Furthermore, it is compared and evaluated with analytical hierarchy process (AHP) technique in terms of average running time and dataset size. The findings display that the RP-GWO performs better than AHP mechanism by approximately (30%).

Keywords: requirements analysis, requirement prioritization, prioritization techniques, optimization, meta-heuristic, grey wolf optimization

1. Introduction

Software Engineering (SE) includes one of the most significant fields that is Requirement Engineering (RE). In addition; Software Development Life Cycle (SDLC) considered RE as the most significant stage of the life cycle (Sommerville& Sawyer, 1997). RE consists of requirements identification, elicitation and validation of requirements and requirements documentations. The whole software projects, development process have many restrictions such as budget and time in order to acquire the market production; this leads to forward the software projects as consecutive versions. However, in the large software projects, many stakeholders participate in order to make decisions of what should be developed firstly which makes a difficulty in decision making. This difficulty motivates the software engineers to put the suitable order for the given requirements in an efficient way to make the right decision about the delivery and development of the particular project (Sommerville& Sawyer, 1997) (Svensson et al., 2011) (Greer & Bustard, 1997).

Thus, RP is the most significant part of RE which comes under the requirement analysis phase. Hence, the most significant part in requirement analysis is RE, because it includes the most significant part that is RP. In order to build a software project and submit a good system that meets the client needs. In case, the project has rigid execution plan, inadequate resources and customer's expectations in a high level, then the most significant characteristics should be published in early enough time. This reason leads us to comprehend the significance of priority ordering of the requirements (Sommerville& Sawyer, 1997).

The present of requirement prioritization process award the stakeholder's high chance to be involved in deciding which requirements a software should be included based on the importance and the impact of development process and this participation aids the stakeholders to transact with restrictions and limitations of project's resources. Moreover, negotiate the contentions between viewpoints which effect in the software development. Thus, these viewpoints come from different goals and roles of stakeholders. Priority in projects that have large number of requirements becomes a priority for the success or failure based on the project constraints and limitations (Svensson et al., 2011). The main goal of this participation is to order the requirements according to their importance and useful execution order (Sommerville& Sawyer, 1997) (Svensson et al., 2011) (Wiegers, 1999). Accordingly, clustering technique can be used in order to group the requirements depends on their importance. In other words, each cluster contains the similar data objects. In addition; the clusters are different

from each other's (Jain et al., 1999), (Emami&Derakhshan, 2015). For the time being, many mechanism are suggested by researchers such as meta-heuristic techniques and heuristics techniques in order to solve the problems that happen as a consequence of complicated datasets. However, meta-heuristic techniques are the most used in order to solve any problem of optimizations; which aims to define the optimal solutions for fulfilling data clusters and decreases the issues of local minima. One of the recently meta-heuristic algorithm is the grey wolf optimization (GWO) algorithm that proposed by (Mirjalili et al., 2014). Which simulate the hunting behavior of grey wolves in nature. Usually, these wolves live in pack within cluster size between 5 and 12. Moreover, these wolves split the dominant leadership hierarchy into four main kinds: alpha, beta, delta and omega. Where alpha denotes as the leader wolf and the decision maker. While beta wolf aids the higher level for making the decision. Delta wolves submit the information to the top levels. All other wolves in the pack join to omega.

The rest of this paper is organized as follows: section II includes the related work, while section III outlines the grey wolf optimization (GWO) Algorithm. The proposed algorithm "RP-GWO" algorithm is discussed in Section IV. Section V represents the discussion of results of the proposed algorithm. VI Finally, the last section draws the conclusion of this study.

2. Related Work

A lot of mechanisms are proposed and developed in order to prioritize the requirements for a specific software project. Some of these mechanisms are more appropriate for a software project that contains small number of requirements in turn others are more suitable for a software that has large number of requirements. Stakeholders play a significant role in order to make decisions to analyze the given requirement, then designate numbers that reflect their importance using an appropriate technique. The most common RP mechanisms are clustered into three classes (Vestola, 2010); nominal scale, ordinal scale and ratio scale. Nominal scale mechanisms make number of groups to which object can be grouped (Karlsson et al., 1998).

In other meaning, the project requirements can be classified into classes depends on their importance. This class contains only two techniques; MoSCoW and Numeral assignment techniques. While ordinal scale prioritization mechanisms create arranged lists of requirements. Moreover, this class can sort the requirements in terms of their importance but not to what amplitude. The most common techniques in ordinal scale class are bubble sort, minimal spanning tree and priority group mechanisms. Finally, ratio scale mechanism creates sorted lists similar to ordinal scale mechanisms, but this class can provide the relative diversity. The most popular techniques in ratio scale class are Analytic hierarchy process (AHP), Hierarchical cumulative voting (HCV) and Hierarchy AHP (Hudaib et al. 2018).

The most popular and traditional mechanism is numerical assignment mechanism. In this mechanism, all requirements that have the same priority associate the same group. However, does not inform which requirement is less or more priority than the others within the similar priority (Ma, 2009). This mechanism is mentioned by many authors such as Karlsson et al. (Karlsson et al., 2006) andSommerville and Sawyer (Sommerville& Sawyer, 1997). Moreover, the most mechanism that is similar to numerical assignment mechanism is the priority groups mechanism which assigns each requirement in one of the basic groups; high, medium and low groups. However, the difference between these techniques is that numerical assignment classes the requirements once; while the priority groups classify the given requirements repeatedly (Karlsson et al., 1998).

(Karlsson et al., 1998) presented the priority classifications in the empirical study that prioritize 13 requirements. This study concludes that priority groups' mechanism is the worst mechanism in terms of slow to execute and hard to use. Moreover, this technique is not appropriate to apply on a software project that has a small number of requirements. While (Karlsson et al., 1998) draws that AHP mechanism consumes time. Regarding to this study and its conclusion, many mechanisms are improved to solve this problem by minimizing the number of rapprochements such as minimal spanning tree and hierarchy AHP techniques. Another technique is compared in this study (Karlsson et al., 1998) which is bubble sort technique that is the softest to used and provided fault tolerance and reliable. However, this technique is faster than AHP from the time consuming side but slower than minimal spanning tree mechanism. This leads that bubble sort technique is appropriate for a software project that includes a small number of requirements. According to this study (Karlsson et al., 1998), the minimal spanning tree is suitable for a project that contains large number of requirements. In other words, this technique is not appropriate for small number of requirements. However, it is suitable for small number of requirements if the fault tolerance and reliability are more important than time consumption. Furthermore, binary search tree is compared and studied in this study (Karlsson et al., 1998) which is used to prioritize the list of requirements.

3. Grey Wolf Optimization

Grey Wolf Optimization (GWO) algorithm is one of the recently meta-heuristic mechanisms that is suggested by Mirjalili and Lewis in 2014 (Mirjalili et al., 2014). This technique mimics the foraging behavior of grey wolves in nature. The main goal of this algorithm is setting the optimal solution for a given issue by utilizing a population of search agents.

In nature, these wolves live in packs; each pack size is 5-12 in average. The main distinction between GWO and other meta-heuristic optimization algorithms is the social dominant hierarchy which its main purpose is to develop the candidate solution during each iteration. Thus, the GWO imitates the hunting behavior of grey wolves in terms of finding and attacking prey (Mirjalili et al., 2014) (Goldbogen et al., 2013). The social hierarchy of wolves and their responsible is illustrated in the following Fig.1.
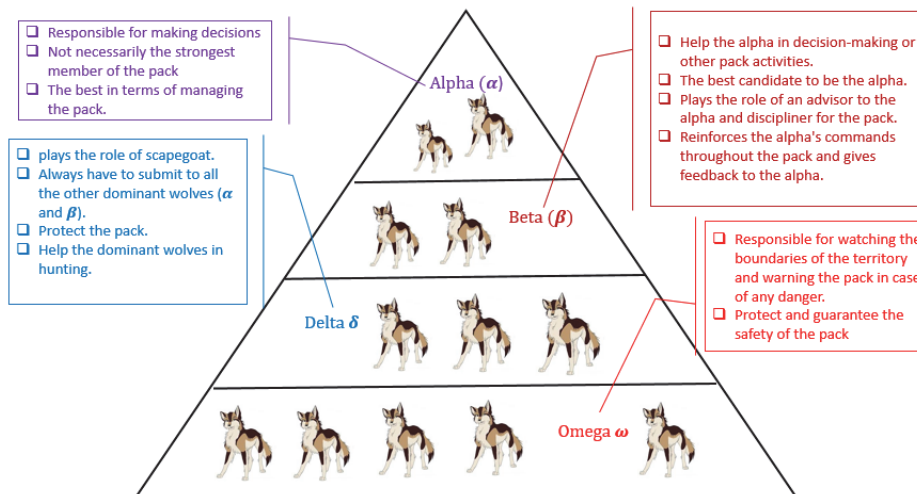

Figure 1. The social dominant hierarchy of grey wolves (Mirjalili et al., 2014)

Based on (Mirjalili et al., 2014), each pack includes the following:

• Alpha wolves are controlling wolves that are followed by the other wolves in the pack. Moreover, they responsible to make decisions. Thus, they display the best candidate solutions.

• Beta wolves assist the upper level to make decisions and act as a bridge between the higher level which contains alpha wolves and the lower level that includes the rest of wolves in the pack. Thus, they represent the second best candidate solutions.

• Delta wolves accountable to deliver information to the higher levels; alpha and beta. In addition to that, they display the third best candidate solutions.

• Omega wolves responsible to deliver information to the higher levels. Moreover, they represent the reminder of solutions.

In addition to the social dominant hierarchy of grey wolves, pack hunting is another pleasant attitude of grey wolves. According to (Muro et al., 2011) the major stages of grey wolves foraging behavior are as follow:

• Tracking, chasing, and approaching the prey.

• Following, encircling, and harassing the prey until it stops moving.

• Attack towards the prey.

Thus, GWO acts the foraging technique of the grey wolves mathematical which is used to solve complex optimization problem. The optimal solution is considered as prey for the given issue.

The three upper levels motions emulate the prey surrounding by grey wolves, which is the following mathematical equation is proposed in this regard (Mirjalili et al., 2014).

$$\vec{D} = \left| \vec{C}.\vec{X}_p(t) - \vec{X}(t) \right|, \; with \; \vec{C} = 2.\vec{r}_2 \tag{1}$$

Where t indicates the current iteration, Xp displays the prey position vector, X represents the grey wolf position,

C is a coefficient vector and r2 is denoted as a random vector in [0, 1]. Thus, the finding of vector D is employed in order to shift the specific element toward or far from the region that the best solution is placed that indicates the prey by applying the following formula (Mirjalili et al., 2014):

$$\vec{X}(t+1) = \left| \vec{X}_p(t) - \vec{A}\cdot\vec{D} \right| \text{ , with } \vec{A} = 2\vec{a}\cdot\vec{r}_1 - \vec{a} \tag{2}$$

Where r1 is chosen randomly in [0, 1] and *a* is decreased from 2 to 0 over the range of iterations. In case |A| < 1, this means the exploitation phase is matched and emulate the behavior of assaulting the prey. While, in case |A|> 1, this means that the exploration phase is matched and simulate the grey wolves spacing from the prey. The proposed values of *A* are in the range [-2, 2]. Thus, calculating the following mathematical equations in order to identify the top three levels α, β and δ (Mirjalili et al., 2014).

$$\vec{D}_\alpha = \left| \vec{C}_1\cdot\vec{X}_\alpha - \vec{X} \right| \text{With} \vec{X}_1 = \vec{X}_\alpha - \vec{A}_\alpha\cdot(\vec{D}_\alpha) \tag{3}$$

$$\vec{D}_\beta = \left| \vec{C}_2\cdot\vec{X}_\beta - \vec{X} \right| \text{With} \vec{X}_2 = \vec{X}_\beta - \vec{A}_\beta\cdot(\vec{D}_\beta) \tag{4}$$

$$\vec{D}_\delta = \left| \vec{C}_3\cdot\vec{X}_\delta - \vec{X} \right| \text{With} \vec{X}_3 = \vec{X}_\delta - \vec{A}_\delta\cdot(\vec{D}_\delta) \tag{5}$$

In order to mathematically simulation the foraging process of wolves, suppose that α, β and δ have adequate knowledge about the potential location of the prey. Furthermore, the first three top solutions that obtained are stored and oblige the other agents to update their positions accordingly. The following mathematical expression is suggested in this regard (Mirjalili et al., 2014):

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{6}$$

4. The Proposed Algorithm "RP-GWO"

As shown in Fig.2, this section presents a proposed algorithm which targets to prioritize the requirements which are given for a particular project. In this study, the suggested algorithm is analyzed in theory and tested on different datasets size. In order to obtain the optimal requirements prioritization, the grey wolf optimization algorithm is stratified. Figures (3 -6) illustrate the pseudo-code for "RP-GWO" algorithm.
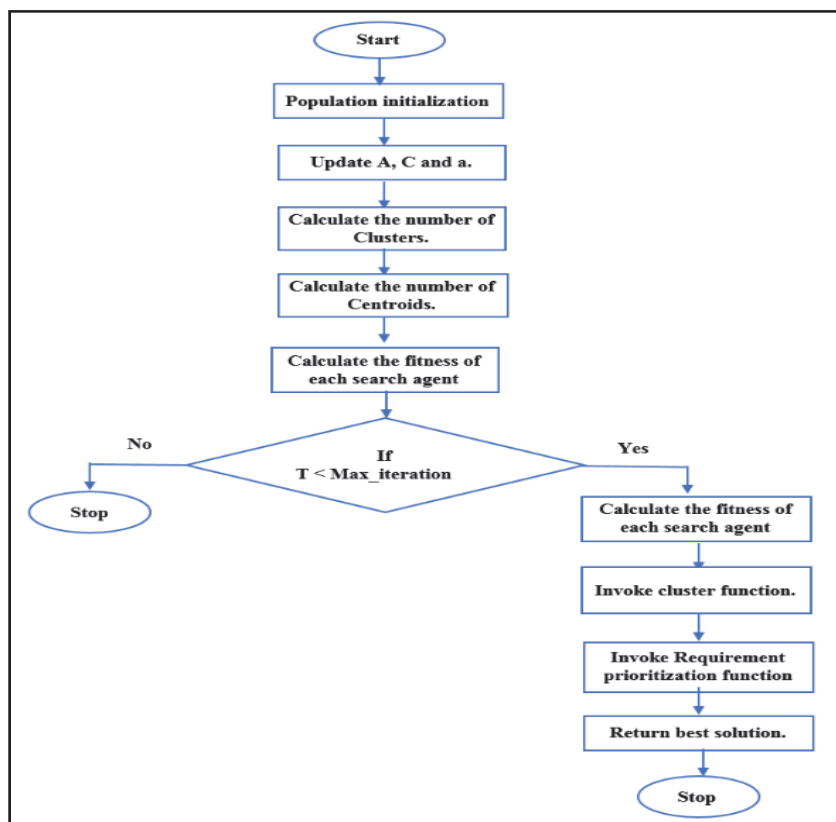

Figure 2. Flowchart of "RP-GWO" Algorithm

Initialize the grey wolves population X_i (i= 1, 2, 3… n)

Initialize A, C and a.

Compute the number of clusters by Eq. (7)

Compute the number of centroids by Eq. (8)

Compute the fitness function for each wolf X_i

Invoke cluster function

Invoke Requirements Prioritization function

Return the best solution

Figure 3. Pseudo-code for "RP-GWO" Algorithm

4.1 Initialization Stage

As shown in Fig.3, initialize the population of grey wolves that is selected randomly.

4.2 Clustering

The wolves in every pack are from 7 to 15 wolves in average according to (Mirjalili et al., 2014). Thus, K-means cluster is used in this study in order to group each cluster and to be has 5 to 12 nodes in average. Firstly, the number of clusters is computed using Eq. (7). Secondly, the number of centroid is calculated by using Eq. (8) which means it is equal the number of clusters. Therefore, centroids are selected randomly. Thus, each wolf that initialized associated to its nearest centroid by using Eq. (9) (Yassien et al., 2017) (Masadeh et al., 2017).

$$\text{Number of clusters} = \text{Ceilling}\left(\frac{\text{number of wolves}}{12}\right) \tag{7}$$

$$\text{Number of centroids} = \text{Number of clusters} \tag{8}$$

$$\text{Distance} = ||X_i - \text{Centroid}_j|| \tag{9}$$

Where the double notations thatappear in Eq. (9) indicates that it is a function of Euclidean distance, Xi represents wolf (i) and Centroid$_j$ indicates to a centroid in cluster j.

Each cluster M detects α, β and δ randomly that represent the three best solutions. During the iterations, the other search agents update their locations around the victim based on the location of α, β and δ using Eqs. (1-6). Thus, each cluster has three solutions α, β and δ which are locally optimal. Then, compare the three best solutions that represent the local optimal solutions with all other locally optimal solutions in other clusters. The minimum local optimal solution denotes as the first global optimum. While the second minimum local optimal indicates to the second global optimal and so on. Fig.4 shows the Pseudo-code of clustering function.

1. For each cluster M

2. Detect the first three best search agents α, β and δ.

3. While (iteration < = max_iteration)

4. Update the location of the current search agents by the Eqs. (4, 5 and 6).

6. Compute the fitness of all search agents.

7. Update a, A and C.

8. Update α, β, and δ.

9. Iteration= iteration +1.

10. End while.

11. Return α for each cluster M.

Figure 4. Clustering function

4.3 Fitness Function

As shown in Fig.5, the main goal of fitness function in this study is measure the distance between each wolf and each centroid in order to join the closest one and to be a member of this cluster.

1. For each Wolf (i)
2. Calculate the distance between each wolf (i) and all centroids by the Eq. (9)
3. If Wolf (i) is not assigned
4. Assign Wolf (i) to its closest centroid.
5. End For (if i= max number of wolves)

Figure 5. Fitness function

4.4 Requirements Prioritization Function

Each search agent denotes to a requirement from the both sides of specific factors and weight of importance. In other words, it is point to the significance of requirements in the development operation for specific project. In this study, the weight of the requirement indicates to the ratio of cost over value for a particular given requirement. Cost value based technique is used in this study similar to AHP technique. Moreover, in order to compare the performance between the suggested algorithm and AHP technique; which is the most popular technique that is based on cost value based mechanism. In addition, AHP mechanism based on pairwise comparison and the sorted set of given requirements ascending which depends on the calculated ratio by using Eq. (10).

$$\text{Ratio} = \frac{\text{Cost}}{\text{Value}} \tag{10}$$

As shown in Fig.6, during iterations, the requirement prioritization function works until all clusters going to be empty and each cluster has three best solutions α, β and δ; α indicates to the requirement that has the first local minimum ratio while β denotes as the second local minimum ratio and δ represents the third local minimum ratio. Then the three top solutions in each cluster will compare with other clusters top solutions. Whereas; the requirement which has a minimum ratio comparing with other ratios will be prioritized as the most significant and drop it from its cluster and the search space. The following mathematical formula is suggested in this regard.

$$\text{Global Min Ratio} = \text{MIN } \{\text{Local Min Ratio (1)} \dots \text{Local Min Ratio (M)}\} \tag{11}$$

Where *M* denotes as the number of clusters.

1.　While each cluster is nonempty (1,2,3,…N)

2.　　　For each cluster M

3.　　　　Arrange the requirements by the Eq. (10)

4.　　　　Select the requirement which has minimum ratio

5.　　　End For

6.　　Select Minimum of minimums ratio by the Eq.(11)

7.　Remove the selected one from search space

Figure 6. Requirements prioritization function

5. Results and Discussions

MATLAB program based as simulation program was ameliorated to evaluate the performance of "RP-GWO" by using various dataset sizes. Furthermore, the dataset size was between 100 and 1000 requirements. Each scenario was repeated 10 tests and raising the size of data set by 100, same as prior studies (Dorigo et al., 2008). These datasets are chosen randomly in terms of cost and value for each requirement. The simulation program that used in this work was implemented on portable computer with the following specifications; Intel (R) core (TM) i7-4510U CPU with 2.40 GHz, 16 GB RAM and Windows 8.1, 64-bit operating system.

In this study, AHP mechanism is selected in order to compare its performance with the suggested algorithm "RP-GWO" from side the time consumption because it is one of the most popular and classical mechanism ; thatused to prioritize the requirements that given for specific project (Perini et al., 2013). In addition, this mechanism is used because pairwise comparison method that estimates a cost or a value comparing one requirement with others requirements had been delicate by (Karlsson et al., 1998)(Perini et al., 2013)(Karlsson et al., 2006). Fig.7 Displays the average run time in seconds between RP-GWO and AHP mechanism for several datasets. It is clear from Table 1 and Fig.6 that the suggested algorithm is achieved better performance comparing to AHP mechanism, especially for huge size of dataset.

Table 1. Average run time in seconds for "RP-GWO" algorithm compared to AHP technique for various datasets

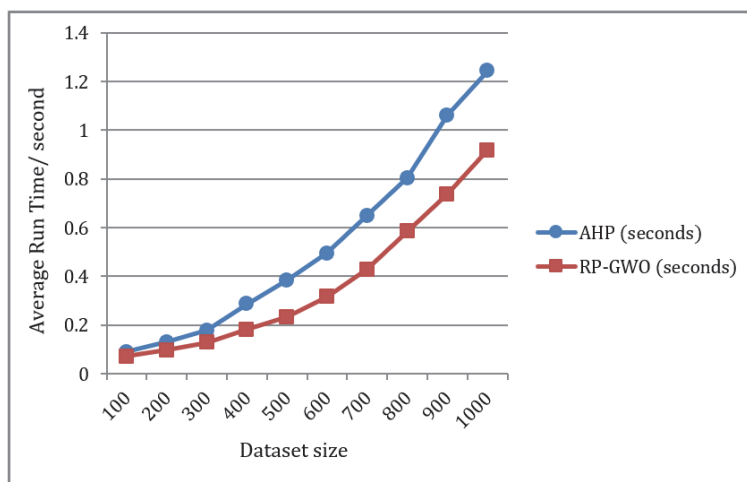| Dataset	AHP (seconds)	RP-GWO (seconds)
100	0.09064	0.0719
200	0.13126	0.09688
300	0.17814	0.12814
400	0.2875	0.18126
500	0.3844	0.2344
600	0.49376	0.31564
700	0.65002	0.42816
800	0.80314	0.58438
900	1.0625	0.73752
1000	1.24376	0.91876
Average:	0.532512	0.369704

Figure 7. Average running time for experimental results for "RP-GWO" compared with AHP technique in seconds

6. Conclusion

The basic phase in any software project is requirement engineering which is complicated phase in software developmental because it includes different activities and transacting with stakeholders. While, the requirement prioritization (RP) is the most important and base phase to submit a good system that accepts the client needs. Accordingly to that, GWO is applied and used as a meta-heuristic technique in order to prioritize the requirements. The results in this work represent that the RP-GWO performs better than the AHP technique in terms of time consuming to rank the requirements and assign number to each requirement which one is more or less priority than the others requirements.

References

Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., & Winfield, A. (Eds.). (2008). Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium,

September 22-24, 2008, Proceedings (Vol. 5217). Springer.

Emami, H., & Derakhshan, F. (2015). Integrating fuzzy K-means, particle swarm optimization, and imperialist competitive algorithm for data clustering. *Arabian Journal for Science and Engineering, 40*(12), 3545-3554.

Goldbogen, J. A., Friedlaender, A. S., Calambokidis, J., McKenna, M. F., Simon, M., &Nowacek, D. P. (2013). Integrative approaches to the study of baleen whale diving behavior, feeding performance, and foraging ecology. *BioScience, 63*(2), 90-100.

Greer, D., & Bustard, D. W. (1997). SERUM-Software engineering risk: Understanding and management. *The International Journal of Project & Business Risk, 1*(4), 373-388.

Hatton, S. (2007). Early prioritisation of goals. Advances in conceptual modeling–Foundations and applications, 235-244.

Hudaib, A., Masadeh, R., Haj Qasem, M., & Alzaqebah, A. (2018). Requirements Prioritization Techniques Comparison. *Modern Applied Science, 12*(2).

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR), 31*(3), 264-323.

Karlsson, J., Wohlin, C., & Regnell, B. (1998). An evaluation of methods for prioritizing software requirements. *Information and Software Technology, 39*(14-15), 939-947.

Karlsson, L., Höst, M., & Regnell, B. (2006, September). Evaluating the practical use of different measurement scales in requirements prioritisation. In Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering (pp. 326-335).ACM.

Ma, Q. (2009). The effectiveness of requirements prioritization techniques for a medium to large number of requirements: a systematic literature review (Doctoral dissertation, Auckland University of Technology).

Masadeh, R., Sharieh, A., & Sliet, A. (2017). Grey wolf optimization applied to the maximum flow problem. *International Journal of Advanced and Applied Sciences, 4*(7), 95-100.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46-61.

Muro, C., Escobedo, R., Spector, L., & Coppinger, R. P. (2011). Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations. *Behavioural processes, 88*(3), 192-197.

Perini, A., Susi, A., & Avesani, P. (2013). A machine learning approach to software requirements prioritization. *IEEE Transactions on Software Engineering, 39*(4), 445-461.

Sommerville, I., & Sawyer, P. (1997). Requirements engineering: a good practice guide. John Wiley & Sons, Inc.

Svensson, R. B., Gorschek, T., Regnell, B., Torkar, R., Shahrokni, A., Feldt, R., & Aurum, A. (2011, August). Prioritization of quality requirements: State of practice in eleven companies. In Requirements Engineering Conference (RE), 2011 19th IEEE International (pp. 69-78). IEEE.

Vestola, M. (2010). A comparison of nine basic techniques for requirements prioritization. Helsinki University of Technology.

Wiegers, K. (1999). First things first: prioritizing requirements. *Software Development, 7*(9), 48-53.

Yassien, E., Masadeh, R., Alzaqebah, A., & Shaheen, A. (2017). Grey Wolf Optimization Applied to the 0/1 Knapsack Problem. *International Journal of Computer Applications, 169*(5).

Copyrights