ISSN 2278-3091
Volume 9, No.1, January – February 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse09912020.pdf
https://doi.org/10.30534/ijatcse/2020/09912020

# Prioritization of Software Functional Requirements: A Novel Approach using AHP and Spanning Tree

**Muhammad Yaseen, Aida Mustapha, Mohamad Aizi Salamat, Noraini Ibrahim**
Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Batu Pahat, Johor, Malaysia.
yaseen_cse11@yahoo.com, aidam@uthm.edu.my, aizi@uthm.edu.my, noraini@uthm.edu.my

## ABSTRACT

Requirements prioritization is an important activity conducted during requirements management phase of requirement engineering. Prioritization of requirements is necessary to include certain features in software while exclude unnecessary requirements. Similarly for developers of software project, prioritization of requirements is also important. Prioritization of small size requirements is easy but difficult process when deal with large size requirements due to time complexity of prioritization technique. AHP (Analytic hierarchy process) is an efficient technique which has been used from several years by many authors. AHP is simple in use and yield accurate results but one of difficulty with AHP is too much comparisons that make is difficult to use for large size requirements. In this research study, an improved AHP technique is designed to diminish number of comparisons and time complexity of AHP with spanning tree based approach. Reducing total comparisons with AHP will make it more suitable to prioritize large size software requirements.

**Key words:** Requirements engineering, requirements prioritization, functional requirements, AHP, MST.

## 1. INTRODUCTION

Requirement Engineering (RE) is important phase in software engineering where requirements are collected from clients in more systematic way [1] [2]. Requirement prioritization (RP) is an important activity during RE where requirements are assigned net importance of priority for implementation [2]. To implement quality software project in limited budget and time, giving importance and priority to requirements become necessary [3][4]. Importance of prioritization increase more and more when size of requirements become larger [5][6]. Techniques such as cost-value ranking, aattribute goal-oriented, value-oriented are designed to prioritize business requirements [7][8]. Prioritization techniques such as binary tree, value-based, and genetic algorithm designed to prioritize user requirements during elicitation [9][10][11]. Finally, prioritization techniques such as QFD and ccontextual preference based technique are specifically designed for non-functional requirements [12][13].

The Analytical Hierarchy Process (AHP) is prioritization technique suitable to prioritize any type of software requirements. It is an organized decision-making method that is intended to compute complex multi-criteria decision problems. The method was formerly suggested by Thomas Saaty [14]. Even though the results for AHP is very accurate, it can only cater for small-sized requirements. AHP pairwise compare all requirements and calculate net importance of requirements with respect to other requirements e.g. if total number of requirements are m than total pairwise comparisons will reach to m*(m-1)/2. Due to too much comparisons, AHP is not suitable for prioritization of large size requirements. AHP is considered to be suitable and scalable for prioritizing small size requirements [13].

The purpose of this research is to enhance AHP so that it can be applied for large-sized FRs. Although AHP is not suitable technique to be applied efficiently for large size functional requirements, but when number of comparisons of requirements is reduced before applying AHP, then the technique can be efficiently applied to prioritize large size requirements with reduce time complexity. When comparing large size requirements, not all requirements are related to each other, so we can separate different requirements with spanning tree and then AHP can be applied efficiently [15]. The remaining of this paper will proceed as follows. Section 2 presents the work related to AHP and requirements prioritization. Section 3 presents the materials and methods. Section 4 presents the time complexity and validation results and finally Section 5 concludes with some direction for future works.

## 2. BACKGROUND STUDY

AHP is applied efficiently in many studies e.g. in one of their studies, author used AHP to prioritize process requirements. Both local and perspective priority was considered to prioritize software process requirements using AHP. For this purpose, CBPA framework was designed which capture process requirements from various perspective from stakeholders and then prioritize it. Process requirements are prioritized from perspective of both business and management such as increase profit, lead in completion, reduction in development cost and time to deliver software are taken as

business oriented process requirements while schedule, satisfaction of customers and increase of productivity and benefits from software's are taken as business oriented process requirements. Matrix is drawn to compare all requirements against each other's. Similarly another matrix is drawn to include multiple stakeholders and their point of view and then all requirements are pairwise compared and net priority of all requirements are calculated[16].

In another study, AHP is used in combination with Artificial Neural Network (ANN) to reduce efforts during requirements elicitation from clients [17]. For this purpose, author applied ANN to group requirements of same nature or category and then AHP is applied to group of requirements rather than individual requirements. In first phase of prioritization, clients are requested for the preparation of all possible requirements to be included in software. Along with this, client's preferences from different perspective are collected. Purpose of ANN is to remove all conflicts from requirements and make arrange requirements in groups and then AHP were efficiently applied. Although technique is suitable for large size requirements but too much complex in use. Similarly if size of requirements in same group increase, time complexity of AHP increase. For particular scenarios this technique is preferable but not better to prioritize all types of requirements from different perspectives.

Not always software requirements are implemented in single version. Due to too much requirements and limited budget and time restrictions or customers need of some features on urgent basis, requirements need to be prioritized based on goals of the stakeholders. Certain goals are stable during the whole project whereas several goals changes with passage of time because of the affection from environment such as laws, stakeholders, variety of clients, requirements and business restrictions, market necessities etc and thus prioritization become necessary. This shows goals of stakeholders varies. In his study, author applied AHP to prioritize requirements by considering various goals [7].

In another paper author have prioritized functional requirements as relation and based on nonfunctional requirements using AHP method of pairwise comparisons [18]. Every functional requirement will be check against its associated nonfunctional requirement and if the priority of nonfunctional requirement is found greater than functional requirement. This shows that paying attention to nonfunctional requirements can change priority for functional requirements.

## 3. MATERIALS AND METHODS

Figure 1 shows step by step process for the current research work. The detail of framework is discussed as follows.

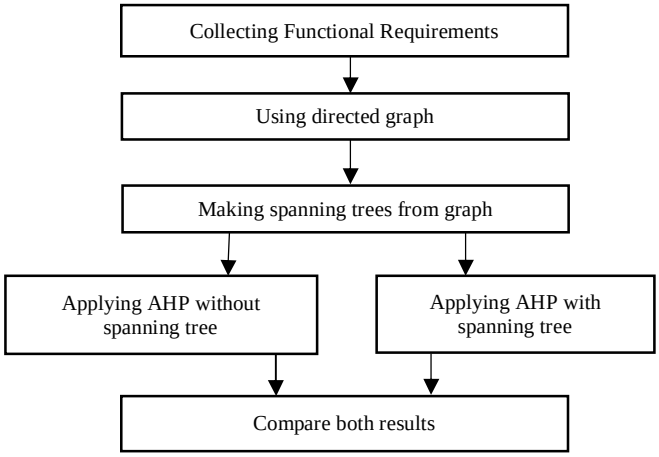


**Figure 1:** Step-by-step research design process

### 3.1 Functional Requirements (FRs) Collection

Functional requirements (FRs) are low level requirements that belong to different high level user requirements [19]. FRs are core requirements of any software system. Developers of software project deals with FRs [20]. For this research study, we have considered FRs from development perspective to validate our proposed technique of AHP using spanning trees.

### 3.2 Representation of FRs

Circular rounded shape notations are used for representing FRs for this study and alphabets $R_1$, $R_2$, …, $R_n$ are used for denoting requirements as shown in figure 2. Many other studies also used similar notations for the representation of FRs [10].

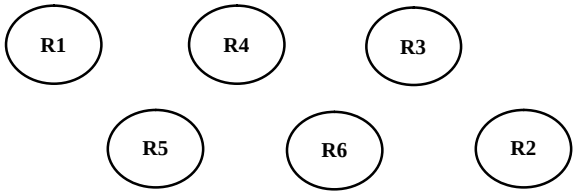


**Figure 2:** Requirements Prioritization

FRs are connected and inter-related with directed acyclic graph (DAG) as shown in figure 3. E.g. in in Figure 2, R5 is required for R3 and R6 or we can say that both R3 and R6 need R5 for implementation. Through DAG, we can easily relate requirements with one another.

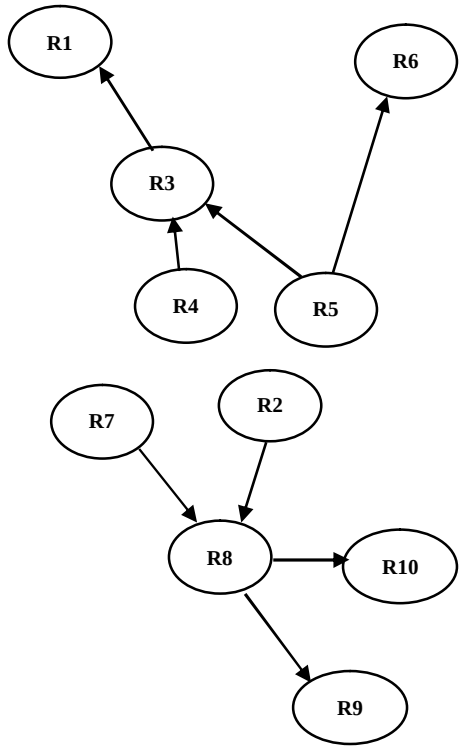


**Figure 3:** Graphical representation of requirements

### 3.2 Adjacency Matrix

In graph theory of computer science, an adjacency matrix is a square matrix used for representing a finite graph. This matrix shows which element of the graph is directed towards other elements of the graph. Graph consist of vertices and edges. Vertices in this study shows FRs while edge shows relation of one requirement with other requirement. In Table 1, the value 1 represents the relationship of requirement with another requirement while 0 shows no relation. For example, R4 is required for R3, so we placed 1 in column R3 and row R4.

### 3.3 Spanning Tree

Spanning trees represents special sub-graphs that possess numerous significant characteristics. For example, if T is a spanning tree belongs to graph G, then T must span G, which shows that T must cover all vertices inside G. Second, T must be a sub graph of G. Third, if every edge in T also occurs inside G, this shows G is similar to T. In many studies, authors used spanning tree algorithms for directed graph [21][22]. With the help of adjacency matrix, we can show all those requirements through spanning tree for which a particular requirement is needed. E.g. from table 1 we can see that R2 is needed for R8 which is then needed for R9 and R10. Figure 4 shows possible spanning trees resulted from figure 3. Spanning trees formation from directed graph either follow depth first search (DFS) or breadth first search (BFS) algorithm [23].

**Table 1:** Adjacency matrix for requirements of Figure 3

| | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 |
|---|---|---|---|---|---|---|---|---|---|---|
| R1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R5 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| R6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 3.4 Apply AHP to Spanning Tree

Through the process of AHP, each and every requirement is pairwise compared. In this case when AHP is applied to requirements of spanning trees only dependent requirements are compared. For the rest of requirements that are not dependent on each other will be consider as equal by default. E.g. in Tree 2, R1 and R6 will be consider of the equal priority although both belongs to same tree. As we have four possible spanning trees as shown in figure 4, we can apply AHP to individual trees or combination of trees that contain common requirements E.g. R3 is common is first two trees while R8 in next two trees. AHP is applied for all cases in this research work.

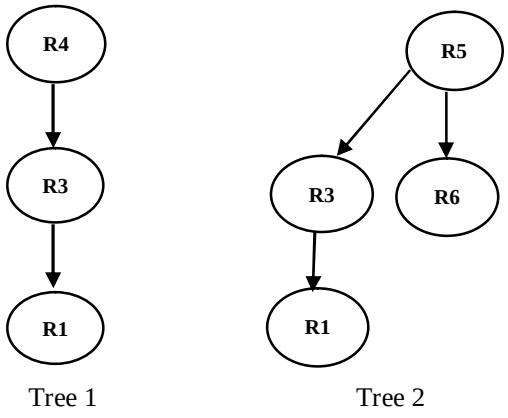


**Figure 4:** Spanning tree resulted from graph of Figure 3

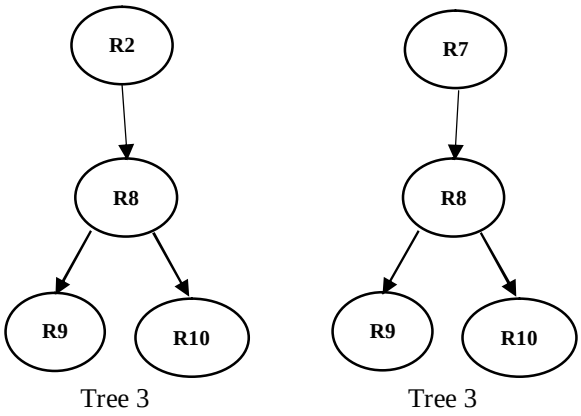


**Figure 4:** Spanning tree resulted from graph of Figure 3 (con't)

Now we are separately taking two different groups of requirements. Table 2 shows requirements of first two spanning trees while table 3 shows requirements of the next two spanning trees.

**Table 2:** Pairwise comparison of first two spanning trees

| | R1 | R3 | R4 | R5 | R6 |
|---|---|---|---|---|---|
| R1 | 1 | .5 | .25 | .25 | 1 |
| R3 | 2 | 1.0 | .50 | .50 | 1 |
| R4 | 4 | 2.0 | 1.00 | 1.00 | 1 |
| R5 | 4 | 2.0 | 1.00 | 1.00 | 2 |
| R6 | 1 | 1.0 | 1.00 | .50 | 1 |
| Sum | **12** | **6.5** | **3.75** | **3.25** | **6** |

**Table 3:** Averaging and normalization for Table 2 values

| | R1 | R3 | R4 | R5 | R6 | Sum |
|---|---|---|---|---|---|---|
| R1 | .083 | .076 | .066 | .076 | .166 | .467 |
| R3 | .166 | .153 | .133 | .153 | .166 | .771 |
| R4 | .333 | .307 | .266 | .307 | .166 | 1.380 |
| R5 | .333 | .307 | .266 | .307 | .333 | 1.550 |
| R6 | .083 | .153 | .266 | .153 | .166 | .581 |

In table 2, all requirements are pairwise compared. If requirement is needed for other requirement than its priority will be greater than it. In this study we have consider it double i.e. 2 but we can select any value from 2 to 9. E.g. R3 is required for R1, so R3 priority will be two times as compare to R1 while R4 priority will be 4 times than R1 as its priority is double as compare to R3. For those requirements that are independent i.e. not related either belong to same or different trees, we will put 1 against these requirements. When all values after comparing FRs are placed in table, then normalized values for each requirement is calculated by dividing priority value of each requirement against other requirement by rows sum for each column. E.g. for R1 against R1, its priority value i.e. 1 will be divided on 12. Normalized values for each requirement is calculated and given in table 3.

In next step, averaging or sum of normalized values are calculated by adding normalized values of requirements belong to all columns for each row. Column sum in table 3 shows averaging over normalized value. We can find priority out of 1 by dividing it on number of requirements which is 6. Now repeat the same process for Tree 3 and Tree 4 in combination and the results are given in table 4 and table 5 respectively.

**Table 4:** Pairwise comparison of 3rd and 4th spanning trees

| | R2 | R7 | R8 | R9 | R10 |
|---|---|---|---|---|---|
| R2 | 1.00 | 1.00 | 2.00 | 4 | 4 |
| R7 | 1.00 | 1.00 | 2.00 | 4 | 4 |
| R8 | .50 | .50 | 1.00 | 2 | 2 |
| R9 | .25 | .25 | .50 | 1 | 1 |
| R10 | .25 | .25 | .50 | 1 | 1 |
| Sum | **3.00** | **3.00** | **6.00** | **12** | **12** |

**Table 5:** Averaging and normalization for Table 4 values

| | R2 | R7 | R8 | R9 | R10 | Sum |
|---|---|---|---|---|---|---|
| R2 | .333 | .333 | .333 | .333 | .333 | 1.66 |
| R7 | .333 | .333 | .333 | .333 | .333 | 1.66 |
| R8 | .166 | .166 | .166 | .166 | .166 | .83 |
| R9 | .083 | .083 | .083 | .083 | .083 | .41 |
| R10 | .083 | .083 | .083 | .083 | .083 | .41 |

Table 6 summarize priority values for each requirement as calculated and shown in table 3 and table 5. R2 get high value while R9 and R10 get low values.

**Table 6:** Priority values assigned for two combined trees

| Requirement | Priority |
|---|---|
| R2 | 1.660 |
| R7 | 1.660 |
| R5 | 1.550 |
| R4 | 1.380 |
| R8 | 0.830 |
| R3 | 0.771 |
| R6 | 0.581 |
| R1 | 0.467 |
| R9 | 0.410 |
| R10 | 0.410 |

We can validate result from figure 4 easily. R2 and R7 got highest values in 3rd and 4th trees respectively while R8 got value less than R2 and R7, because both of them are required for R8. R2 priority is slightly higher than R5, although both are required for same number of requirements but chain structure is different. Similarly R5 have high priority than R4 because R5 is needed for more requirements. Similarly R6 priority will be lower than R3 because R3 is required for R1 while R6 is required for none of the requirement.

Now calculate priority values of requirements separately for each tree. Table 7 shows prioritize values obtained as a result of prioritization from AHP by repeating the same process for Tree 1, Tree 2, Tree 3 and Tree 4 individually.

**Table 7:** Priority values assigned for each individual trees

| Requirement | Priority |
|---|---|
| R2 | 2.00 |
| R7 | 2.00 |
| R5 | 1.80 |
| R4 | 1.70 |
| R8 | 1.00 |
| R3 | 0.90 and 0.85 |
| R6 | 0.77 |
| R1 | 0.56 and 0.42 |
| R9 | 0.50 |
| R10 | 0.50 |

Now consider whole set of requirements by considering all trees together, and compare each and every requirement against others. Table 8 shows summary of prioritization as a result of AHP by considering combination of four trees.

**Table 8:** Priority values by considering all requirements together

| Requirement | Priority |
|---|---|
| R2 | 1.334 |
| R7 | 1.334 |
| R5 | 1.244 |
| R4 | 1.153 |
| R8 | 1.000 |
| R3 | 0.838 |
| R6 | 0.837 |
| R1 | 0.563 |
| R9 | 0.600 |
| R10 | 0.600 |

## 4. TIME COMPLEXITY AND VALIDATION OF RESULTS

From table 6, table 7 and table 10, we can see that order of priority is same in all cases, either we prioritize requirements of combined trees or separately result is same in all cases. Some minor changes like R9 is given more priority as compare to R1 in second and third cases while in first case R1 is given slightly higher priority. But as whole results are same. Number of comparisons of depended requirements are although same in all cases but as whole number of calculated normalized values are different in all cases. E.g. when we compare all 10 requirements with each other's then total calculations will be equal to 10*10 which is 100. When we consider another case in which requirements of combined spanning trees are compared only i.e. 5 requirements in each tree as shown in table 2 and table 3 respectively. The total calculations reduce to 25. This is because in first case, each and every requirement is compared with all other requirements either belong to same tree or different tree but in second case requirements in two combined spanning trees are compared only against others which reduced the total comparisons and calculations.

In our study although priority values order is same in all cases but in real software's where software requirements size is large, there can be much difference in results, so which the results of which case is more accurate? If we consider whole set of requirements in one table than along with too much comparisons biasness can increase as we have to put 1 against all independent requirements and in big size requirements number of 1's of independent requirements will be very much so it can have negative impact on results.

Now what will happen if we consider individual trees separately instead of two combined spanning tree each? There are two disadvantages, first is that number of calculations can increases as we are repeating calculations after comparison of common requirements of more than one tree e.g. in Tree 1 and Tree 2, R3 and R1 are common, so in separate tables we are repeating calculations for them although comparisons are done once. Second disadvantage is that for common requirements like R3, we are calculating priorities separately e.g. one for Tree 1 and another for Tree 2 separately which can decrease or increase the overall priority of R3 as we are considering its priority multi times separately.

Now if we combine trees with common requirements like we combined first two trees, then overall impact of requirement will be not affected and biasness will be decrease. Also, in combined trees calculations will be not repeated although it will be lesser (half) as compare to combined case of all requirements.

## 5. CONCLUSION

So we can conclude from our study that AHP is easy and better technique to prioritize requirements but takes too much time due to too much comparisons and calculations but if we compare only dependent FRs, then we can reduce number of comparisons and calculations to minimum and the results will be more accurate. With the help of spanning tree, we have related dependent requirements. With help of spanning tree and AHP, we have efficiently prioritized FRs with accuracy and minimum time complexity. From the current work, we have concluded that comparing requirements of either individual tree or combination of trees with common requirements reduces time complexity and produce accurate results as compare to that case where we consider whole set of requirements for comparisons and prioritization.

In future we aim to apply the suggested framework to FRs of Enterprise Resource Planning (ERP) system. ERP system developers, who avoid to use AHP because of its time complexity and number of comparisons, will be able to use it easily.

## ACKNOWLEDGEMENT

This paper is supported by the Ministry of Education, Malaysia under the Fundamental Research Grant Scheme FRGS/1/2019/ICT01/UTHM/02/1

## REFERENCES

[1] M. . Yaseen, S. . Baseer, S. . Ali, S. U. . Khan, and Abdullahb, 'Requirement implementation model (RIM) in the context of global software development', *2015 Int. Conf. Inf. Commun. Technol. ICICT 2015*, 2015. https://doi.org/10.1109/ICICT.2015.7469573

[2] M. Yaseen, N. Ibrahim, and A. Mustapha, 'Requirements Prioritization and using Iteration Model for Successful Implementation of Requirements', *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 1, pp. 121–127, 2019.

[3] M. Yaseen, A. Mustapha, and N. Ibrahim, 'Prioritization of Software Functional Requirements : Spanning Tree based Approach', vol. 10, no. 7, pp. 489–497, 2019. https://doi.org/10.14569/IJACSA.2019.0100767

[4] M. Yaseen, A. Mustapha, and N. Ibrahim, 'MINIMIZING INTER-DEPENDENCY ISSUES OF REQUIREMENTS IN PARALLEL DEVELOPING SOFTWARE PROJECTS WITH AHP', vol. 8, no.

Viii, 2019.

[5] M. Yaseen, A. Mustapha, and N. Ibrahim, 'An Approach for Managing Large-Sized Software Requirements During Prioritization', no. 1, pp. 98–103, 2019.

[6] B. Manoj, K. V. K. Sasikanth, M. V. Subbarao, and V. Jyothi Prakash, 'Analysis of data science with the use of big data', *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 7, no. 6, pp. 87–90, 2018. https://doi.org/10.30534/ijatcse/2018/02762018

[7] N. Garg, M. Sadiq, and P. Agarwal, 'GOASREP : Goal Oriented Approach for Software Requirements Elicitation and Prioritization Using Analytic Hierarchy Process', pp. 281–287, 2017.

[8] M. A. A. Elsood and H. A. Hefny, 'A Goal-Based Technique for Requirements Prioritization', 2014. https://doi.org/10.1109/INFOS.2014.7036697

[9] R. Beg, R. P. Verma, and A. Joshi, 'Reduction in number of comparisons for requirement prioritization using B-Tree', no. March, pp. 6–7, 2009.

[10] P. Tonella, A. Susi, and F. Palma, 'Interactive requirements prioritization using a genetic algorithm', *Inf. Softw. Technol.*, vol. 55, no. 1, pp. 173–187, 2013.

[11] A. K. Massey, P. N. Otto, and A. I. Antón, 'Prioritizing Legal Requirements', vol. 1936, no. 111, 2010.

[12] C. E. Otero, E. Dell, A. Qureshi, and L. D. Otero, 'A Quality-Based Requirement Prioritization Framework Using Binary Inputs', pp. 0–5, 2010. https://doi.org/10.1109/AMS.2010.48

[13] F. Dalpiaz, 'Contextual Requirements Prioritization and Its Application to Smart Homes', vol. 1, pp. 94–109, 2017.

[14] 'AHP.pdf'. .

[15] A. Perini, F. Ricca, and A. Susi, 'Tool-supported requirements prioritization : Comparing the AHP and CBRank methods', *Inf. Softw. Technol.*, vol. 51, no. 6, pp. 1021–1032, 2009.

[16] X. Frank, Y. Sun, and C. Sekhar, 'Priority assessment of software process requirements from multiple perspectives', vol. 79, pp. 1649–1660, 2006.

[17] M. I. Babar, M. Ghazali, D. N. A. Jawawi, S. M. Shamsuddin, and N. Ibrahim, 'Knowledge-Based Systems PHandler : An expert system for a scalable software requirements prioritization process', *KNOWLEDGE-BASED Syst.*, 2015. https://doi.org/10.1016/j.knosys.2015.04.010

[18] F. Fillir, '6 \ VWHP UHTXLUHPHQWV SULRULWL ] DWLRQ EDVHG RQ $+ 3', pp. 163–167, 2014.

[19] M. Yaseen and Z. Ali, 'Success Factors during Requirements Implementation in Global Software Development : A Systematic Literature Review', vol. 8, no. 3, pp. 56–68, 2019.

[20] Z. Ali, M. Yaseen, and S. Ahmed, 'Effective communication as critical success factor during requirement elicitation in global software development', vol. 8, no. 03, pp. 108–115, 2019.

[21] S. Kapoor and H. Ramesh, 'Algorithmica An Algorithm for Enumerating All Spanning Trees of a Directed Graph 1', pp. 120–130, 2000.

[22] C. J. Quinn, N. Kiyavash, and T. P. Coleman, 'Efficient methods to compute optimal tree approximations of directed information graphs', *IEEE Trans. Signal Process.*, vol. 61, no. 12, pp. 3173–3182, 2013. https://doi.org/10.1109/TSP.2013.2259161

[23] C. Papamanthou, 'Depth First Search & Directed Acyclic Graphs', 2004.