International Journal of Computer Science and Software Engineering (IJCSSE), Volume 9, Issue 4, April 2020
ISSN (Online): 2409-4285 www.IJCSSE.org Page: 20-29

A Graph Based Approach to Prioritization of Software Functional Requirements

Muhammad Yaseen[1], Aida Mustapha[2], Sidra Qureshi[3], Abdullah Khan[4] and Atta Ur Rahman[5]

[1, 2] Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia.
[3] SubHealth (Pvt.) Ltd, KPK, Peshawar, Paksitan
[4] Department of Computer Science, Univeristy of Swat, Pakistan
[5] Dapartment of Software Engineering, Comsats University Islamabad, Pakistan

yaseen_cse11@yahoo.com

ABSTRACT

Requirements prioritization plays important role for a successful requirements implementation. Functional requirements, in specific, represents the specification of behavior between the inputs and outputs. They are prioritized based on the high-level requirements of the system and subsystems functionalities, as well as the type of software, expected users and the type of system where the software is used. Nonetheless, prioritization of functional requirements is very challenging considering a project where the requirements are huge. In practice, prioritization of functional requirements highly depends on stakeholders' preference for giving priorities in features selection instead of based on its internal structure and characteristics. This is because the key information in functional requirements concern on business processes, security, performance, data migration and conversion. This paper proposes a graph-based approach for prioritizing functional requirements using directed acyclic graphs for relating requirements with one another on the basis of its importance to overall project and how much it is required for other requirements.

The proposed approach is then evaluated in terms of total time estimation to project completion. The experimental results showed that the graph-based approach is able to effectively prioritize functional requirements with lower estimated project completion time as compared to non-prioritized requirements. The approach will help software vendors to deliver projects well within the total project duration.

Keywords: *Requirement Prioritization, Functional Requirements, Directed Graph.*

1. INTRODUCTION

Requirement prioritization (RP) is an important activity during requirements implementation phase in requirement engineering
[1][2]. RP is about giving priority to requirements for better time planning during implementation [3][4]. Without requirements prioritization, development phase could be costly as it takes more efforts to complete all the requirements within the specified timeframe [5][6]. Existing techniques from the literature have shown that they are not scalable for large set of requirements especially in dealing with dependency issues between the functional requirements. When one requirement is dependent on other requirements, prioritization on the basis of internal structure of the implementation becomes necessary [7][8]. There are three types of requirements; (1) business requirements which deals with benefits and cost issues of requirements along with time constraint, (2) functional requirements which are necessary for software system to develop, and (3) non-functional requirements that are not directly demanded but are necessary for ensuring quality product such as security and performance issues.

Along this line, research on requirements prioritization techniques depends on the type of requirement under study whether at business level [9][10][11], non-functional level [12][13][14] or at functional level [15][16][14]. Meanwhile, some of the techniques are able to cater for all types of requirements [17][18]. In addition to the types of requirements, requirements prioritization techniques also depends on the size of the requirements data. For example, AHP and cumulative voting are more suitable for a small set of functional requirements but often fail on large requirements due to high time consumption. From the implementation point of view, requirements prioritization has to consider how much a particular requirement deserves to be assigned with high priority considering the internal linking structure of the requirements especially when the size of requirements is huge [19]. For instance, certain requirements may have higher priority from the user's side, but gets assigned a lower priority from the developer's side. Therefore, there is need for a new requirements prioritization approach that considers both factors but is able to achieve high accuracy but at a lower time computational efforts.

This paper is set to propose a graph-based approach for prioritizing functional requirements by relating requirements with one another on the basis of its importance to the overall project and how much it is required by other requirements. The remainder of this

paper proceeds as follows. Section II presents the related work on requirements prioritization approach in requirements engineering. Section III presents the proposed graph-based functional requirements prioritization algorithm. Section IV presents the evaluation of the proposed approach, Section V discusses the results, and finally Section VI concludes with some indication for future work.

2. RELATED WORK

In requirements prioritization, cumulative voting is a method where stakeholders are given 100 points and they have to distribute these points on all requirements, similar to a voting mechanism. The requirement with high votes will be given more priority. However, the problem arise when the requirements are too large or the number of stakeholders are more than one. Another problem is biasness, whereby a stakeholder may vote a requirement that he or she likes, regardless of the importance of the requirements. Another challenge with the voting approach that the stakeholders may assign a zero to a particular requirement when they do not consider the requirement [20]. To address this issue, an improved statistical model called CODA was presented to solve the interdependencies arise among the requirements. In [21], requirements prioritization is proposed to be developed on the basis of benefits and cost of the requirement to the customers. In other words, the requirements are prioritized on the basis of how much efforts are needed and how much benefits they provide. In dealing with the dependencies between the requirements in terms of cost and its respective value, the prioritization algorithm used the concept of binary tree, whereby the requirements are first arranged in a sequence and then a binary tree is formed based on the sequence. This technique was compared to AHP but was found more difficult in use considering the structure of the binary tree. Nonetheless, this technique is very helpful due to the smaller number of comparisons needed as compared to AHP even though with increased number of requirements

In binary tree, prioritization was carried out based on sorting mechanism to prioritize the requirements whether in ascending or descending order [22]. The method is although consuming less time as compare to AHP but it is difficult to use. Also, the method is not automatic but user have to give input and arrange requirements in binary tree.

In determining the priority of requirements at the business level, the goals and criteria for the project and customer should be identified as earlier as possible. Some goals are fixed throughout the project while some goals can vary as time progresses due to external environment effects such as laws, stakeholders, diversity of customers, requirements and business constraints or new market needs. Due to hierarchical structure of an organization, different people can have different opinions and suggestions about the requirements. Along with stakeholder preferences it is also necessary to have prioritization on the basis of dependencies in requirements. DRANK an automated algorithm is presented to do comparisons on the importance of dependent requirements and compared the results with AHP and other techniques. Experiments proved that this technique is more efficient and takes less time [23].

In [24], the research proposes two matrices; for stakeholder goals' weight and criteria weight. This work is similar to AHP in terms of using weights for goals, the number of comparisons is a lot smaller as compared to technique by AHP. To avoid the issue of biasness when gathering priorities from individual stakeholders, group decisions have also found helpful. After getting remarks from individual stakeholder, a group of people will analyze the requirements. At the end, on the basis of group decision, all the requirements are prioritized accordingly [25].

Intelligent-based solution was proposed by [26] for prioritization of collected requirements from stakeholders through the use of machine learning technique. First, stakeholders were requested to prepare requirements. Next, clustering was applied to group similar requirements, then the Artificial Neural Network (ANN) was applied for second layer of before AHP was applied at the end for final comparisons. In another paper, clustering technique using the K-means algorithm for prioritization was proposed. Through clustering techniques, similar objects or requirements can be combined together by assigning them to whether low or high priority groups.

The case study were conducted on websites data of seven airports where priority were assigned on the basis of services in terms of passengers in airport. There were four clusters representing the level of importance i.e. very important, important, less important and not necessary [8].

[27] Proposed the use of Case-based Ranking (CBR), which combines stakeholders' preferences with requirement ordering approximations computed through machine learning approaches. The two goals were minimizing efforts during elicitation and to make partial orders in attributes. The findings reported that by using machine learning approach for requirements during elicitation was able to reduce the efforts during prioritization.

The literature also showed prioritization techniques focusing on the importance of non-functional or quality requirements prioritization. Assessing the importance of quality requirements is normally carried out based on quality survey. Standard quality requirements gets low priority but if attention is made to this side software can be made more successful. Quality criteria highly depends on the customer opinions of a particular quality attribute. Although many techniques like clustering are known to

solve problem with large requirements, they do not solve problems related to internal structure of functional requirements.

3. GRAPH BASED FUNCTIONAL REQUIREMENTS PRIORITIZATION

This paper proposes a graph-based approach to prioritization of functional requirements. The inputs are the functional requirements collected from any sources using appropriate elicitation technique and must be specified in the form of Software Requirement Specification (SRS). In this research, the functional requirements are represented as alphabets R1, R2, . . . , Rn enclosed in circles as nodes. Figure 1 shows notations used for representing the requirements in a graph form.

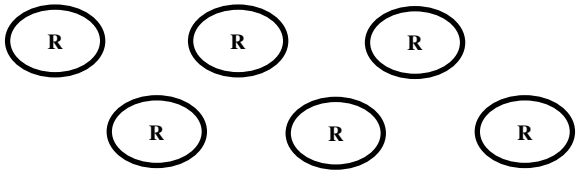

Fig. 1. Notations for representing requirements

3.1 Using DAG For Relating Requirements

A graph G = (V;E) consists of a finite set of vertices V and a finite set of edges E. Graphs are useful for the representation of any kind of data in particular sequence.
This research uses directed acyclic graphs (DAG) rather than cyclic graphs. A directed graph E is a set of ordered pairs of vertices (u; v). The arrows in the graph indicates the dependency of a requirement on another requirement. The requirement generates arrow and points to another requirement indicating that it is necessary or required for another requirement.
For example, R1 ← R2 indicates that R1 is depended on R2 or R2 is required for the completion of R1. Given the requirements collected as shown in Figure 1, Figure 2 shows the graph representation of requirements through directed acyclic graph.

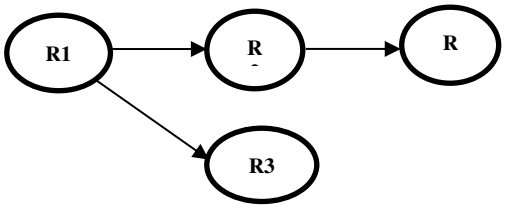

Fig. 2. Representing requirements through directed acyclic graph

3.2 Assigning Priority to a Specific Requirement

In order to assign priority value to requirements certain rules are defined. Following hypothesis are finalized for defining the rules of prioritization.
Hypothesis 1 (H1): Requirement generates higher number of arrows to other requirements is assigned with high priority i.e. prioritizing key and important requirements on the basis of how much it is required for other requirements as a whole.
Note that highlighting the requirements that are more important than others are extremely important because in a large software development projects, the development of requirements is running in parallel. This means specification of many requirements are needed and necessary for other requirements of the same or different modules, hence delaying implementation if the relationships between the requirements are not identified and prioritized. Figure 3 shows the relationships among the requirements.

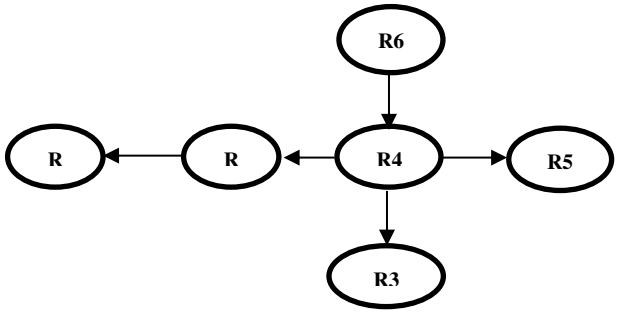

Fig. 3. Relationships among requirements

As shown in Figure 3, R4 will get higher priority score because three requirements are dependent on R4. This also signifies the importance of R4, whereby it is high because more requirements are depending on it. However, R4 depends on R6 for its completion, therefore R6 is assigned with higher priority than R4. Note that although there is only one requirement i.e. R4 is required for R6, R6 still gains higher priority. This proves that for assigning priority solely based on the number of dependencies is insufficient. This leads to the second hypothesis as follows.
Hypothesis 2 (H2): Requirement which is prerequisite for the completion of other requirement is assigned more priority. To demonstrate the hypothesis, Figure 4 shows two independent chain of requirements. R5 is assigned higher priority than R4 and R4 is assigned higher priority than R2, which in turns is assigned higher priority than R1. Similarly, R8 is assigned more priority than R7 and R6. However, when the development process is about to commence, which requirement R5 and R8 should be assigned with higher priority? Figure 4 proves that if modules or chains of requirements have same importance

than the requirements on the two chains will all receive the same priority although both the chains have different number of requirements. Similarly, R4 and R7 will gain the same priority while R6 and R2 will gain the same priority. Finally, the priority of both R6 and R2 will be greater than R1 because R2 is needed for R1.

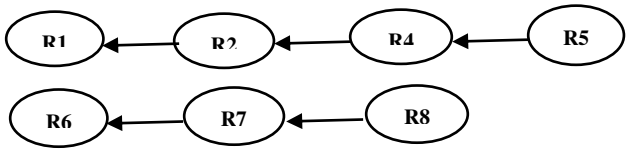

Fig. 4. Requirements needed for other requirements

Figure 5 shows the new representation of Figure 4 such as R7 is depended on R4 as well as R8. Now R5 and R8 will receive the same priority but the priority of R4 and R7 will be not equal. R4 will be assigned higher priority than R7. In this example, the requirement chain is not independent anymore but are depending on each other.

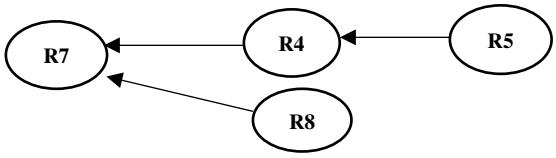

Fig. 5. Requirements with additional prioritization information

Consider that a new requirement R9 is now added to the existing chain of requirements as shown in Figure 6. According to H2, priority of R8 and R5 should be equal. However, between the two cases, R8 has two dependent requirements, therefore should be assigned with additional priority than R5 due to its importance.

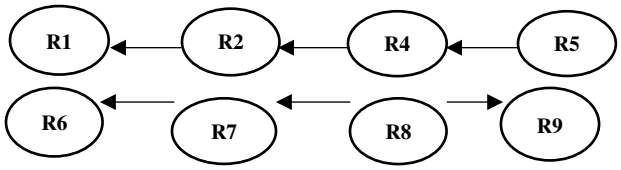

Fig. 6. A new requirement added in

Given all the additional information on priority scores or values among the requirements, Figure 7 shows the final graph structure to this example. Comparing the priority of R5 with R8, R5 now has more importance as compared to R8 as it is needed for two requirements while R8 is only required for R7. Therefore, it should be assigned higher priority but R8 is required for R7 and R7 receives more importance as compared to R1, R2, and R4 because it is required for three requirements and R6 also receives more importance as compare to these requirements. On this chain, total arrows or dependent requirements are five while in upper chain total arrows of importance are four. This indicates that R8 is more important hence should have higher priority as compared to R5. R8 will be assigned to group having higher priority than R5.

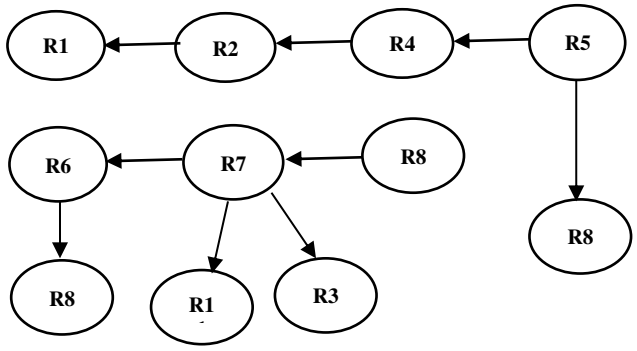

Fig. 7. Final requirements structure

In summary, a priority should be given not only on the basis of the number of chains in a set of requirements but also from the perspective of the importance in overall chain.

3.3 Adjacency Matrix

In this work, the functional requirements are represented using the adjacency matrix representation, |V| * |V| matrix A where, Aij=1 if arrow points from "i" towards "j" and it is zero otherwise.

Following [15] and [16], Table 1 shows the adjacency matrix for ten requirements for Figure 7 where both rows and columns represent every requirement against all other requirements. Requirements on right side of the column will points to other requirements on the above row for which it is required. The value is 1 against that requirement for which it is required while 0 for rest of requirements.

This matrix shows requirements needed for other requirements.

Table 1: Adjacency matrix

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R11
R1	-	0	0	0	0	0	0	0	0	0
R2	1	-	0	0	0	0	0	0	0	0
R3	0	0	-	0	0	0	0	0	0	0
R4	0	1	0	-	0	0	0	0	0	0
R5	0	0	0	1	-	0	0	0	1	0
R6	1	0	0	0	0	-	0	0	0	0
R7	0	0	1	0	0	1	-	0	0	1
R8	0	0	0	0	0	0	1	-	0	0
R9	0	0	0	0	0	0	0	0	-	0
R11	0	0	0	0	0	0	0	0	0	-

3.4 Priority Value Table

Given the adjacency matrix presented in Table 1, a prioritized adjacency matrix is produced as shown in Table 2. Based on hypothesis, we will put 1 not only for those requirements that are directly needed for other requirements but will put 1 against all other requirements which directly or indirectly need it and where the implementation of other requirements against this particular requirement become impossible. Thus, Following H1, this matrix maps the priority of requirements calculated not only on the basis of importance of one particular requirement but rather aggregated from all requirements within the same chain of requirement.

Table 2: Prioritized adjacency matrix

	R 1	R 2	R 3	R 4	R 5	R 6	R 7	R 8	R 9	R11	sum
R1	-	-	-	-	-	-	-	-	-	-	0
R2	1	-	-	-	-	-	-	-	-	-	1
R3	-	-	-	-	-	-	-	-	-	-	0
R4	1	1	-	-	-	-	-	-	-	-	2
R5	-	1	-	1	1	-	-	-	-	-	3
R6	1	-	-	-	-	-	-	-	-	-	1
R7	1	-	1	-	-	1	-	-	-	1	4
R8	1	-	1	-	-	1	1	-	-	1	5
R9	-	-	-	-	-	-	-	-	-	-	0
R11	-	-	-	-	-	-	-	-	-	-	0

Next, a priority value table will produced as shown in Table 3 which aggregates the priority values from the entire chain of requirements based on H2.

Table 3: Priority value table

R1	R2	R3	R4	R5	R6	R7	R8	R9	R11
1	2	2	3	4	2	3	4	3	0

3.5 Requirement Prioritization Algorithm

The algorithm for graph-based functional requirements prioritization proposed in this work has the following steps:

Step 1: Count number of requirements of each and every module (chain of requirements). Say max is number of maximum requirements of any chain i.e. if we have two modules, one contain 5 requirements and other contain 8 requirements than max will be assigned with value of 8. This value will be assigned to high priority requirement of every chain.

Step 2: Start prioritizing requirements of the first module (chain of requirements).

Step 3: Assign the high priority requirement in module with value of max.

Similarly the second most priority requirement in this chain will be assign with value of one less i.e. max-1. Continue the process by decrementing max value till the first or less priority requirement is assigned with lowest possible value.

Step 4: Continue the same process for 2nd module and all other modules, by assigning max value to the high priority requirement inside chain. Continue the same process for all requirements.

Step 5: Make groups of similar value requirements from all modules and keep in descending order of priority.

Step 6: Inside each group, further prioritize requirements in descending order on the basis of how much they are important. I.e. how much they are required for other requirements.

In summary, G1, R8 will get higher priority as compared to R5 because the equivalent importance of R8 is 5 while that of R5 is 3. Likewise, the priority of R4 will be higher than R9 in G2. Similarly, in G3, the priority of R3 will be lower than R2 and R6 but priority of both R2 and R6 will be the same as shown in Table 3.

4. EVALUATION

In order to evaluate the performance of the proposed graph-based algorithm for the functional requirements prioritization, the algorithm is applied on twenty seven functional requirements collected from mobile sales shop. The requirements were elicitated using the interview elicitation technique. The collected user requirements are shown in Table 4. The column “Required for" represents requirements which are required or pre-requisite for a specific requirement in the “Functional Requirement" column.

Table 4: Functional requirements for mobile shop

Functional Requirement	Notation	Required for
Sale main	R1	R3, R25
Customer	R2	R1, R12 , 24,25
Sales detail	R3	
Product	R4	R3, R7, R11, R13
Category	R5	R4
Company	R6	R4
Purchase detail	R7	
Purchase main	R8	R7, R26
Supplier	R9	R8, R10, R26, R27
Purchase return main	R10	R11, R26
Purchase return detail	R11	
Sale return main	R12	R13, R25
Sale return detail	R13	
Bank information	R14	R15, R16
Bank debit	R15	
Bank credit	R16	
Employee	R17	R21, R22, R23, R18
Sale man	R18	R1, R12,
Area	R19	R1, R12,

Expenses module	R20	
Employee Leave request	R21	
Employee salary structure	R22	
Employee salary generation	R23	
Customer payment	R24	
Customer debit	R25	
Supplier debit	R26	
Supplier payment	R27	

Table 5: Requirements priority table

Functional Requirement	Chain priority	Equivalent importance
Sale main	2	2
Customer	3	8
Sales detail	1	0
Product	2	4
Category	3	5
Company	3	5
Purchase detail	1	0
Purchase main	2	2
Supplier	3	8
Purchase return main	2	2
Purchase return detail	1	0
Sale return main	2	2
Sale return detail	1	0
Bank information	2	2
Bank debit	3	0
Bank credit	3	0
Employee	3	10
Sale man	3	7
Area	3	7
Expenses module	3	0
Employee Leave request	2	0
Employee salary structure	2	0
Employee salary generation	2	0
Customer payment	2	0
Customer debit	1	0
Supplier debit	1	0
Supplier payment	2	0

Figure 8 shows the graphical representation of requirements. Next, the chain priority and equivalent importance are calculated for giving priority to each requirement.

Table 5 shows the chain priority value and equivalent importance value. In table 6, requirements are shown in descending order of priority with G1 assigns the highest priority and G3 the lowest. In G1, R17 is assigned the highest priority while R16 is assigned the lowest priority value. Similarly in G2, R4 gets high priority while R27 gets the lowest. In G3, the highest priority is for R3 and the lowest is for R26. Overall the highest priority requirement is R17 and lowest is R26.

Implementing requirements that have been prioritized is crucial in order to manage requirements especially in cases where requirements are implemented in parallel by different teams, and each requirement is waiting for the pre-requisite requirements with higher or equivalent priority importance.

5. RESULTS AND DISCUSSION

Let the requirements to be developed by the four team members; A, B, C and D working in parallel. Given the information on effort estimation produced by the graph-based approach, Table 6 shows the how the functional requirements are grouped and prioritized so the requirements can be distributed among parallel teams and integrated once they are completed.

Table 6: Priority groups

Functional Requirements	Group	Functional Requirements	Group
R17	G1	R10	G2
R2	G1	R12	G2
R9	G1	R21	G2
R5	G1	R22	G2
R6	G1	R23	G2
R18	G1	R24	G2
R19	G1	R27	G2
R1	G1	R3	G3
R15	G1	R7	G3
R16	G1	R11	G3
R4	G2	R13	G3
R8	G2	R25	G3
R14	G2	R26	G3

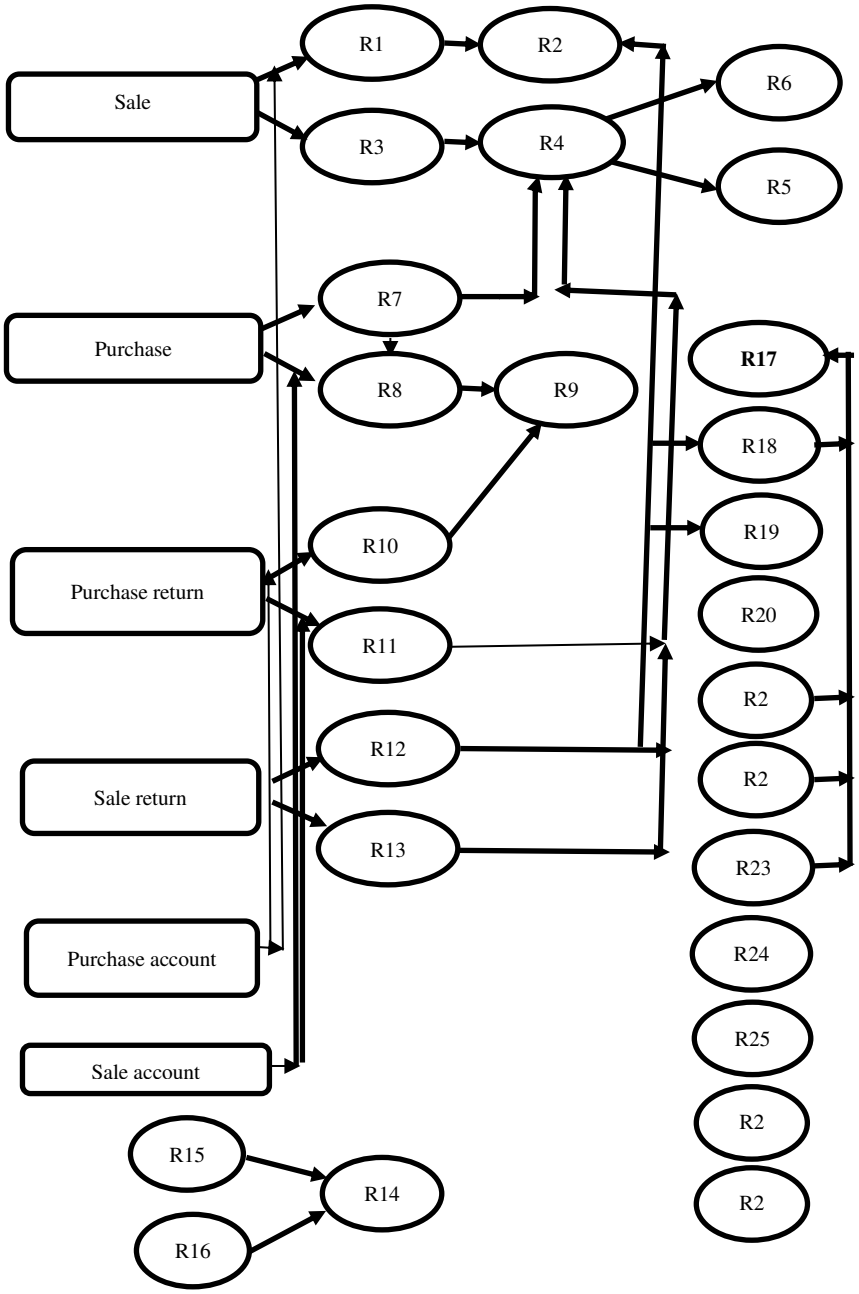

Fig. 8. Graphical representation of requirements for mobile shop

The effort estimation values as shown in Table 7 gives an approximation of how much time a particular requirement will take during an implementation. Suppose all the functional requirements are classified in three categories in accordance with efforts i.e. High, Average, and Low. The categories resulted from the graph-based requirement prioritization algorithm are made after calculation on the basis of functional point analysis and use cases. Low effort requirements takes 8 hours on average as calculated. Average effort requirements take 16 hours while and high effort requirements take 30 hours. Table 7 shows all the requirements with how much efforts required to implement them. The three different cases when distributing requirements are described as follows.

Table 7: Efforts calculated for requirements

Functional Requirements	Efforts	Functional Requirements	Efforts
R17	Low	R10	Low
R2	Low	R12	High
R9	Low	R21	Low
R5	Low	R22	Low
R6	Low	R23	Average
R18	Low	R24	Average

R19	Low	R27	Average
R1	Low	R3	High
R15	Average	R7	High
R16	Average	R11	High
R4	Low	R13	High
R8	Low	R25	Average
R14	Low	R26	Average

CASE 1 (WORST CASE): In this case, requirements are arranged in a way as shown in Table 8. The requirements in parallel are depended on each other e.g. R1, R3, R24, R25, R12, R13 are depending on R2, R4 which are implementing by C. In this case, R2 and R4 are assigned low priority. It can see that R2 and R4 also depended on R5 and R6, which means R5 and R6 both are both required for the completion of R2 and R4. Nonetheless, both requirements are assigned with low priority despite its high importance. The worst case is considered as worst case as the required requirements of C and D are assigned low priority.

Table 8: Distribution of requirements in Case 1

A	B	C	D
R18	R8	R17	R14
R1	R7	R20	R15
R3	R27	R21	R16
R24	R26	R22	R9
R25	R10	R23	R19
R12	R11	R4	R5
R13		R2	R6

CASE2: (AVERAGE CASE): In this case, requirements are arranged in a way as shown in Table 9. R5 and R6 of D are assigned with top priority but R2 and R4 are still assigned with low priority.

Table 9: Distribution of requirements in Case 2

A	B	C	D
R18	R8	R17	R5
R1	R7	R20	R6
R3	R27	R21	R14
R24	R26	R22	R15
R25	R10	R23	R16
R12	R11	R4	R9
R13		R2	R19

CASE3: (BEST CASE): In this case, requirements are arranged in a way as shown in Table 10. R2, R4, R5 and R6 all are assigned with high priority values and bring to the top in the table.

Table 10: Distribution of requirements in Case 3

A	B	C	D
R18	R8	R4	R5
R1	R7	R2	R6
R3	R27	R17	R14
R24	R26	R20	R15
R25	R10	R21	R16
R12	R11	R22	R9
R13		R23	R19

5.1 Time Efforts for All Requirements

In calculating the time effort of all the requirements, the total estimated time for requirements of A, B, C and D are calculated without considering any delay.
Total time estimation for A = 8 + 8 + 30 + 16 + 16 + 8 + 30 = 108 hours.
Total time estimation for B = 8 + 30 + 16 + 16 + 8 + 30 = 100 hours. Total time estimation for C = 8 + 16 + 8 + 8 + 16 + 8 + 8 = 72 hours. Total time estimation for D = 8 + 16 + 16 + 8 + 8 + 8 + 8 = 72 hours.

5.2 Total time estimation of project completion in case 01

As requirements of A are dependent on R2 and R4, they are then dependent on R5 and R6. This indicates that R5 and R6 should be implemented first. In Case 1, the priority of these requirements are very low with the total development time of D to completion is 72 hours. This means R2 and R4 have to wait for 72 hours in waiting for C to be implemented. R17, R20, R21, R22 and R23 will be implemented during this duration and will take a total of 56 hours (8 + 16 + 8 + 8 + 16 = 56). Now instead of 72, R4 and R2 will now wait for 16 hours (72 - 56 = 16). Total time completion of C will become 88 hours (72 + 16 = 88). Now after 80 hours, A can start implementing requirements.
As total time estimation in parallel project development will be equal to maximum, the entire team has to prioritize the cases so it will be near to total. For example, the total time of A is 108 hours but due to delay and finishing time of C, the total estimation time will now become 196 hours (108 + 88 = 196). This is similar to the case of B but total overall delay will be 80 hours.

5.3 Total time estimation of project completion in case 02

In this case, R5 and R6 which are pre-requisites for R2 and R4 will be assigned to D and will be given higher priority as shown in the Table 9. Now, the waiting time will be reduced to 16 hours, which is equivalent to the total time of implementation of R5 and R6. However, this time will not cause any delay for any requirement for C because implementation of requirements for C will take 56 hours. Nonetheless, the total time is equal to 72 hours. A will wait for 72 hours as all requirements are depending on R2 and R4, so total estimation time of the delivery will be equal to 180 hours (108 + 72 = 180). There is a difference of 16 hours in Case 1 and Case 2 showing that Case 2 is better than Case 1.

5.4 Total time estimation of project completion in case 03

In this case, R2, R4, R5 and R6 all are assigned high priority and brought to the top as shown in Table 10. R2 and R4 will wait for only 16 hours to D. Similarly, A will wait for 16 hours to C and the total waiting time will reach 32 hours. Total time estimation will be now equal to 140 hours (108+32 = 140). Now by comparing the best and worst case, there is a

difference of 56 working hours. Since in day there are 8 working hours, the difference found is nearly equal to 7 days. The results of all the cases proved that the proposed graph-based prioritization of functional requirements is able to reduce the total time estimation for any software project.

Figure 9 shows that out of these three cases, case 03 is considered to be best because in case 03, the overall estimation time of project is reduced. This shows the importance of prioritized requirements in parallel development.

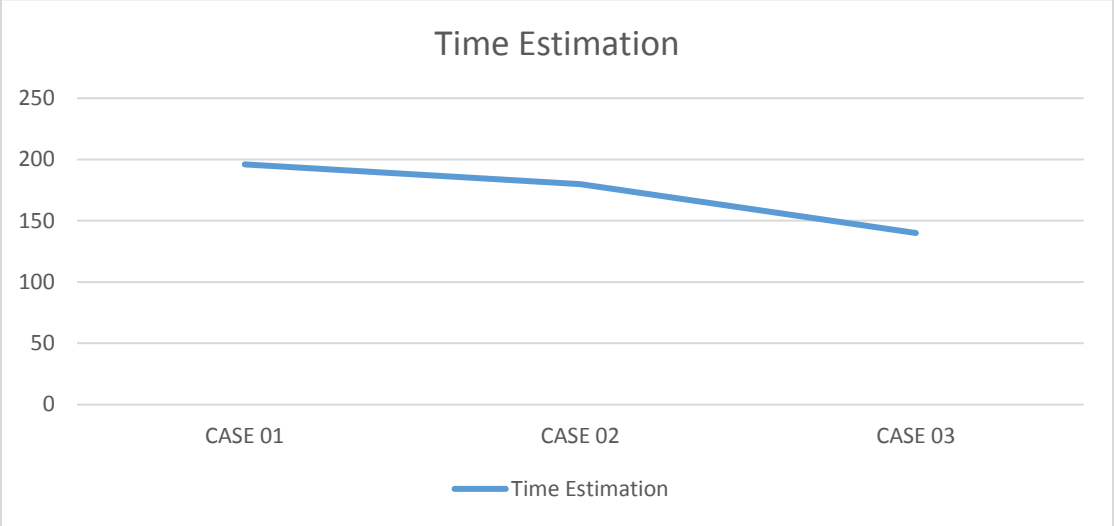

Fig. 9. Time estimation for all three cases

6. CONCLUSION

A new graph-based approach is presented for prioritizing functional requirements in requirement engineering on the basis of how much they are required for other requirements by marking them as key requirements. The proposed graph-based approach has solved dependency issues of one requirement with other, which are not addressed by other techniques. The proposed technique will not only prioritize requirements that are required for other requirements but will also prioritize them on the basis of how much important these requirements are in terms of how much they are needed for other requirements. The proposed approach was evaluated through a case study. The results concluded that requirements that are prioritized take lesser time to complete the project as compared to those requirements that are not prioritized. The graph-based approach will help software vendors in better implementation of requirements that will help them to deliver projects within the time frame. In future, the approach will be tested on big industrial projects like the Enterprise Resource Planning (ERP) systems.

REFERENCES

[1] M. Yaseen, A. Mustapha, and N. Ibrahim, 'Prioritization of Software Functional Requirements : Spanning Tree based Approach', vol. 10, no. 7, pp. 489–497, 2019.

[2] M. Yaseen, A. Mustapha, and N. Ibrahim, 'An Approach for Managing Large-Sized Software Requirements During Prioritization', 2018 IEEE Conf. Open Syst., pp. 98–103, 2019.

[3] N. Misaghian and H. Motameni, 'An approach for requirements prioritization based on tensor decomposition', Requir. Eng., 2016.

[4] 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.

[5] M. Yaseen, A. Mustapha, N. Ibrahim, and U. Farooq, 'International Journal of Advanced Trends in Computer Science and Engineering Effective Requirement Elicitation Process using Developed Open Source Software Systems', vol. 9, no. 1, 2020.

[6] M. Ramzan and M. A. Jaffar, 'Value Based Fuzzy Requirement Prioritization and its Evaluation Framework', pp. 1464–1468, 2009.

[7] M. Yaseen, I. Journal, M. Yaseen, A. Mustapha, M. A. Salamat, and N. Ibrahim, 'International Journal of Advanced Trends in Computer Science and Engineering Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse09912020.pdf Prioritization of Software Functional

Requirements : A Novel Approach using AHP and Spanning Tree', vol. 9, no. 1, 2020.
[8] N. Setiani and T. Dirgahayu, 'Clustering Technique for Information Requirement Prioritization in Specific CMSs', 2016.
[9] Z. Ali and M. Yaseen, 'Critical Challenges for Requirement Implementation in Global Software Development : A Systematic Literature Review Protocol with Preliminary Results', vol. 182, no. 48, pp. 17–23, 2019.
[10] M. Yaseen, Z. Ali, and M. Humayoun, 'Requirements Management Model (RMM): A Proposed Model for Successful Delivery of Software Projects', Int. J. Comput. Appl., vol. 178, no. 17, pp. 32–36, 2019.
[11] A. U. Rahman, M. Yaseen, and Z. Ali, 'Identification of Practices for Proper Implementation of Requirements in Global Software Development : A Systematic Literature Review Protocol', vol. 177, no. 13, pp. 53–58, 2019.
[12] M. Yaseen and M. A. Awan, 'Practices for Effective Software Project Management in Global Software Development : A Systematic Literature Review', vol. 177, no. 36, pp. 1–6, 2020.
[13] 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.
[14] 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.
[15] M. Yaseen and Z. Ali, 'Practices for Effective Communication during Requirements Elicitation in Global Software Development', vol. 8, no. 06, pp. 240–245, 2019.
[16] M. Yaseen, 'Effective Negotiations Practices in Global Software Development : A Systematic Literature Review', vol. 9, no. 1, pp. 87–91, 2020.
[17] M. Yaseen, S. Ali, . A., and N. Ullah, 'An Improved Framework for Requirement Implementation in the context of Global Software Development: A Systematic Literature Review Protocol', Int. J. Database Theory Appl., vol. 9, no. 6, pp. 161–170, 2016.
[18] M. Yaseen and U. Farooq, 'Requirement Elicitation Model (REM) in the Context of Global Software Development', Glob. J. Comput. Sci. Technol., vol. 1, no. 2, pp. 1–6, 2018.
[19] 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.
[20] P. Chatzipetrou, L. Angelis, P. Roveg??rd, and C. Wohlin, 'Prioritization of issues and requirements by cumulative voting: A compositional data analysis framework', Proc. - 36th EUROMICRO Conf. Softw. Eng. Adv. Appl. SEAA 2010, pp. 361–370, 2010.
[21] M. Daneva and A. Herrmann, 'Requirements Prioritization Based on Benefit and Cost Prediction : A Method Classification Framework', pp. 240–247, 2008.
[22] R. Beg, Q. Abbas, and R. P. Verma, 'An Approach for Requirement Prioritization using B-Tree', pp. 1216–1221, 2008.
[23] F. Shao, R. Peng, H. Lai, and B. Wang, 'The Journal of Systems and Software DRank : A semi-automated requirements prioritization method based on preferences and dependencies', vol. 126, pp. 141–156, 2017.
[24] M. A. A. Elsood and H. A. Hefny, 'A Goal-Based Technique for Requirements Prioritization', 2014.
[25] A. Felfernig and G. Ninaus, 'Group Recommendation Algorithms for Requirements Prioritization', pp. 59–62, 2012.
[26] 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.
[27] A. Perini, A. Susi, and P. Avesani, 'A Machine Learning Approach to Software Requirements Prioritization', vol. 39, no. 4, pp. 445–461, 2013.