Requirements Prioritization and using Iteration Model for Successful Implementation of Requirements

Muhammad Yaseen[1], Noraini Ibrahim[2], Aida Mustapha[3]
Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia Parit Raya, 86400 Batu Pahat, Johor, Malaysia

***Abstract*—Requirements prioritization is ranking of software requirements in particular order. Prioritize requirements are easy to manage and implement while un-prioritized requirements are costly and consume much time as total estimation time of project can exceed. Because all requirements are depended on each other so total estimation time exceed when requirements wait for pre-requisite requirements. Priority of requirement also increases when other requirements wait for it but assigning low priority to needed requirements will delaying the project. Iteration model is software engineering (SE) process model in which all requirements are not developed at one time but are developed in phases. Only sufficient information or sub-requirements of particular user requirement (UR) can be needed for other user requirements (URs) so by implementing only the sufficient requirements in first phase will reduce waiting time. Hence total estimation time of the project will also reduce. In this research work, iteration model approach is used during prioritization to reduce total estimation time of project and to assure timely delivery of project. From the results it is concluded that not all sub-requirements of particular UR get same priority, but there are only few requirements that are important and should be given more priority.**

***Keywords*—Requirements prioritization; iteration model; user requirements; spanning trees; directed acyclic graph**

I. Introduction

Software requirements gathering and management is not an easy task and needs systematic approaches [14][21]. Requirement prioritization (RP) is an important activity during requirement management and is defined as is giving order or importance to requirements. RP helps in better management of requirements and make it easy for developers to rank requirements to assure timely delivery of software [1]. RP is not an easy task, many authors have worked on prioritization and suggested several techniques. There are four types of requirements that needs to be prioritize. The goal of every type of requirement is different. Business requirements (BRs) deals with benefits and cost issues of requirements. User requirements (URs) are requirements that come from users either in the form of features or modules. Functional requirements (FRs) are core requirements of system. FRs are the base of URs. FRs are requirements that system must do and must consist of while non-functional (NFRs) are supportive requirements that helps in better implementation of FRs. Techniques like 'Cost value ranking', 'Attribute goal oriented', 'Value oriented' are suggested for prioritizing BRs [2][3]. Some of the techniques like 'AHP', 'Binary tree', 'value based', 'genetic algorithm', are suitable for prioritizing URs and FRs [4][5][6] and techniques like 'QFD', 'Contextual preference based technique' are suggested for NFRs [7][8]. The big challenge for current prioritization techniques is scalability i.e. inability to handle large set of requirements [9]. The current techniques are not suitable for prioritizing FRs from developer's perspective i.e. based on internal structure of requirements.

FRs prioritization from developer's perspective is very necessary for easy management and timely availability of Pre-requisite requirements. In parallel software development, as all User requirements (URs) are related to each other, so one requirement become dependent on others and prioritization process become necessary.

A. *Iteration Model*

The basic idea behind this model is to develop a system through repeated cycles (iteration or phases) and in smaller portions at a time. Through this model, full software is not developed on one time, but only skeleton of whole software is developed and then subsequently requirements are implemented [10][11]. The first step is analysis phase during which all requirements are analysed and examined that which requirements to be implemented first and which should not. The second phase is the design phase in which proper design is made.

After the design, requirements are implemented and at the tested. After integration and deployment, requirements are analyzed for second iteration and then same process repeat itself. The detail of iteration model process is shown in Fig. 1.

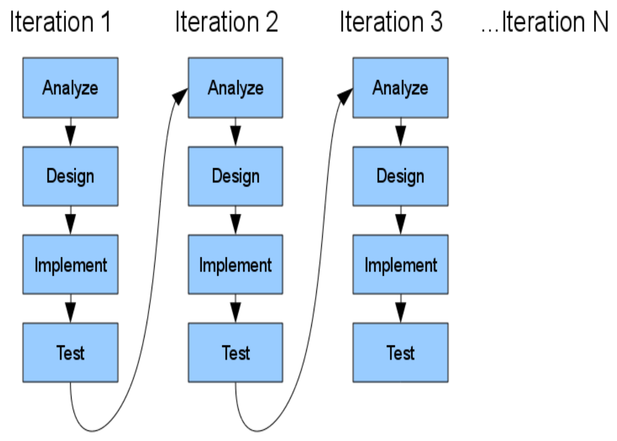

Fig. 1. Iteration Model Process.

II. Background Study

The Analytical Hierarchy Process (AHP) is the most famous, most used and simplest technique for Requirement Prioritization (RP). AHP-based prioritization is performed pairwise by comparing each and every requirement against each other. For *n* requirements, then *n* (*n*-1)/2 comparisons will be needed. AHP completes prioritization for each and every new requirement. For example, if the number of requirements are ten, then AHP will perform forty-five times comparisons of the requirements. If the requirements increase in size, so does the processing time. If the requirements size is in thousand, there will be 1000*(1000-1)/2 = 499,500 comparisons, which is both very time consuming and difficult to execute. Because the technique is time consuming, it is not scalable for big requirements due to the pairwise comparisons for every requirement [12]. In [13], the proposed framework arranges requirements on the basis of benefits and cost that represent requirement dependencies. The work highlighted six ways of dealing with dependencies. First is, cost and benefit value for requirements should be fixed value. Secondly, all requirements should be grouped independently to overcome the complexity issues during calculations. Third, benefit should be measured in relative terms such as dollars and time in hours. Fourth is performing the pairwise comparisons and finally fifth is the use of discrete values instead of continuous values like 1, 2, and 3. “Cumulative voting” or “100 dollars method” is a technique where stakeholders receive 100 dollars or points and they have to allocate dollars or points on all possible requirements just like voting mechanism. The requirement with high polls receive high priority [15].

Group decisions on prioritizing requirements are helpful. After getting remarks from stakeholders, group of people will analyze the requirements. At the end, on the basis of group decision, all the requirements can be prioritized accordingly [16].

Author presents algorithm of binary tree concept for requirement prioritization. Requirements are first arrange and then form a binary tree for that. Using sorting mechanism we can easily prioritization either in ascending or descending order all the requirements. Using this technique as compared to AHP is although difficult in use but very helpful because of the small number of comparisons as compare to AHP. This means for projects having many requirements, we can apply this technique having less amount of comparisons [4].

Value oriented technique focuses on the core value of the business to rank which requirements are more important from the other based on business values. Business stakeholders use simple scale of measuring the values of certain business requirements but they need a framework that can decide exactly which requirement is more important than other. In [16], the business values represent major requirements like security, customer satisfaction, speed, service, and integrity. The requirements are arranged from R_1 to R_n into a matrix of business value vs. score. The matrix will produce a total score for each requirement, which at then sorted as the final list of prioritized requirements [17].

III. Design of Research Methodology

The detail of research design and methodology is given in Fig. 2. The purpose of this design is to follow step by step instruction of prioritization and iteration model. Step by step process is explained as;

A. Requirements Collection

Gathering software requirements is the core task for any software construction [18]. Requirements can be collected by applying elicitation techniques. The collected requirements need proper management i.e. categorizing requirements and make relationship between different URs. Proper management of requirements will help in prioritizing requirements i.e. which requirements should be implemented first and which should not.

B. Graph-based Approach

Graph based approach is adopted for representing URs. Through directed acyclic graph (DAG) requirements are related to each other as shown in Fig. 3. Graphs are useful for representing and relating requirements [5]. In many studies, DAG are used by authors for relating different objects and entities [19]. From DAG one can easily identify which requirements are necessary for which other requirements.

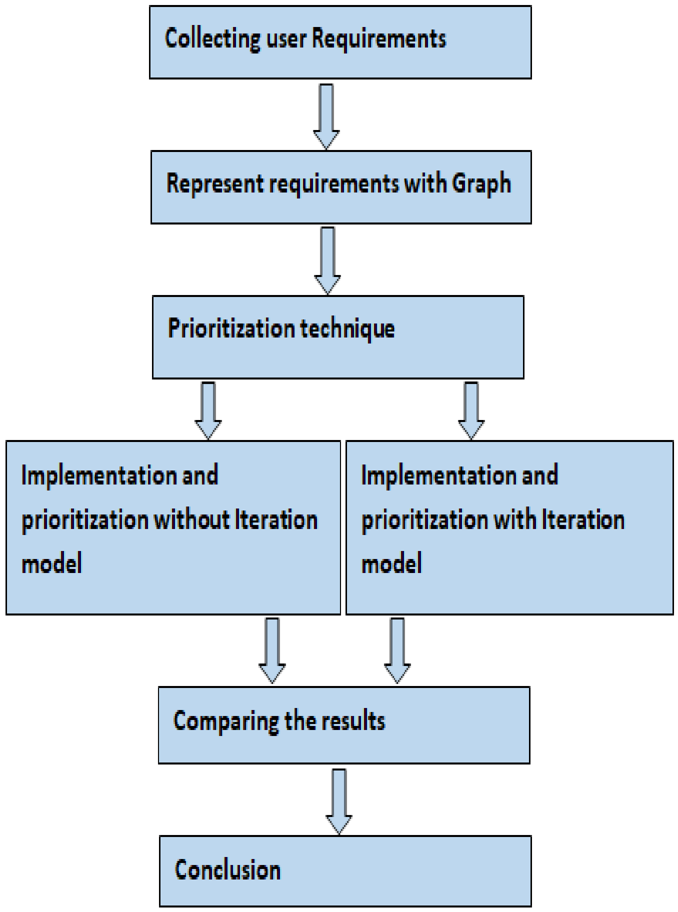
Collecting user Requirements
Represent requirements with Graph
Prioritization technique
Implementation and prioritization without Iteration model
Implementation and prioritization with Iteration model
Comparing the results
Conclusion

Fig. 2. Step by Step Research Design Process.

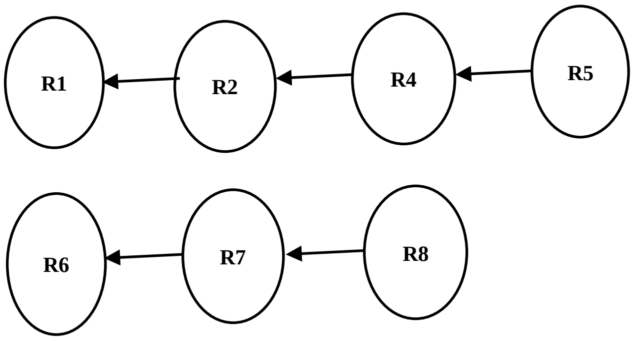

Fig. 3. Assigning Priority to Requirements in Graph.

In above graph, R1 is requirement that is needed for R2 and R3. While R4 need R2 for its implementation. This relation shows that for implementation of R2 and R3, R1 implementation and completion is must.

C. Requirement Prioritization

"Requirement which is pre-requisite for the completion of other requirement is assigned more priority". E.g. in Fig. 3.

R5 priority will be higher than R4 while R4 will get high priority than R2

1) Spanning tree concept: *Spanning trees* are special sub graphs of a graph that have several important properties. First, if T is a spanning tree of graph G, then T must span G, meaning T must contain every vertex in G. Second, T must be a sub graph of G. In other words, every edge that is in T must also appear in G. Third, if every edge in T also exists in G, then G is identical to T.

Priority of requirement can be found through spanning tree inside graph. Spanning tree inside graph will show a complete track for particular requirement through which it is needed to set of all other requirements.

Spanning trees can be formed either as a result of depth first searching (DFS) or breadth first searching (BFS). Record of any visiting node or requirement will be kept on stack. Using DFS, start traversing full leaves of particular branch. When dead point reaches, requirements of that branch will be pop out one by one until it reaches to start point of that branch. Similar process will be repeated for next branch. Dead point is that where requirements are no more required further for any requirement.

Find all possible trees from graph. Starting point will be the requirement which is required for other requirements such that the pre requisite requirements will come to the top as parent and all requirements for which pre requisite requirements are needed will look like a child's and sub child's. E.g. In this directed graph of Fig. 4, all possible spanning trees are;

Tree 1 will start from R2 and ends with R1 as R1 is not required for any other requirement.

Tree 2 will start from R4, passes R3 and ends with R1. Similarly will happen with R5.

Tree 3 will start from R6, now it has three paths, either to go R3 (using DFS or BFS) and then R1, either to go R7 or either R8.

In Fig. 5, priority of R6 will be greater than R3, R7 and R8. Priority of R8 will be greater than R9 and similarly R9 priority will be greater than R10, R11 and R12. Priority of R11 will be greater than R13 and R14. In between R10 and R11, priority of R10 will be slightly higher than R11 because it is needed for R11.

2) Assign numerical values to prioritized requirements: Ranking is technique used to rank requirements either in ascending or descending order of implementation [6]. Numerical values show the order in which requirements should be implemented. These values are not fixed, which means any value can be assigned in certain range. E.g. if 6 is considered maximum value which is highest priority value than R6 will be assigned value of 6. The value of R8 shall be less than 6 e.g. we can assign 5 to R8. Similarly R5 can be either assigned with value of 6 or 3 as this chain have three requirements. As R3 is common in both chains, we can either assign it value 5 or 2. Value 2 will be assigned in case when R5 is assigned 3. Either we assign 2 or 5, we can't implement requirement before its pre-requisites. The purpose of ranking is assigning implementation priority such that pre-requisite requirements will get more priority as compare to other requirements for which it is needed. This method is simple and appropriate in case where priority is given on the basis of its implementation from developer's perspective. Similarly all those requirements that have same implementation priority can be arranged in same group for simplification [20].

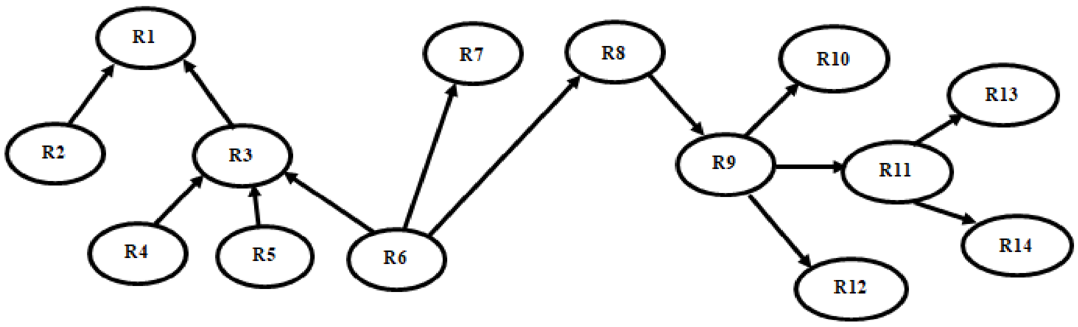

Fig. 4. Directed Graph Connecting Different Requirements.

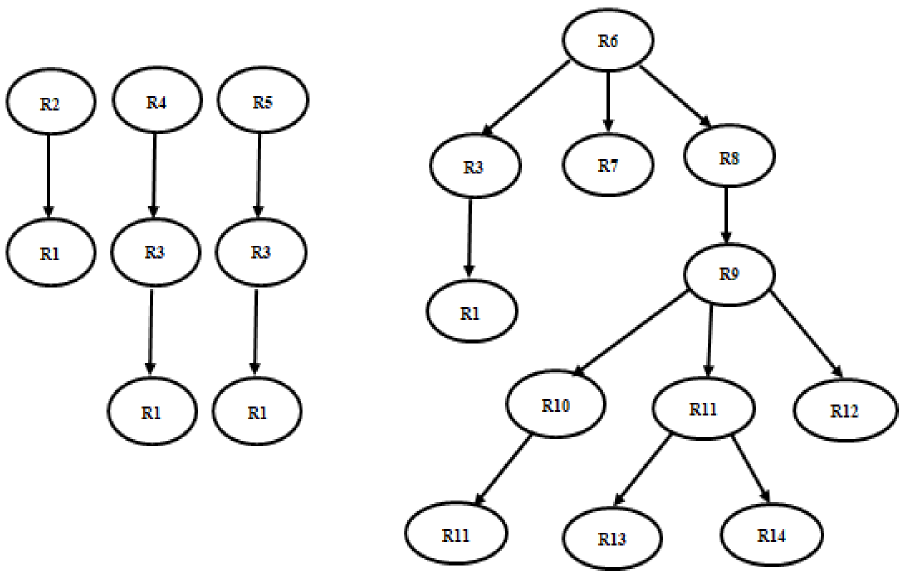

Fig. 5. Spanning Trees from Graph of Fig. 4.

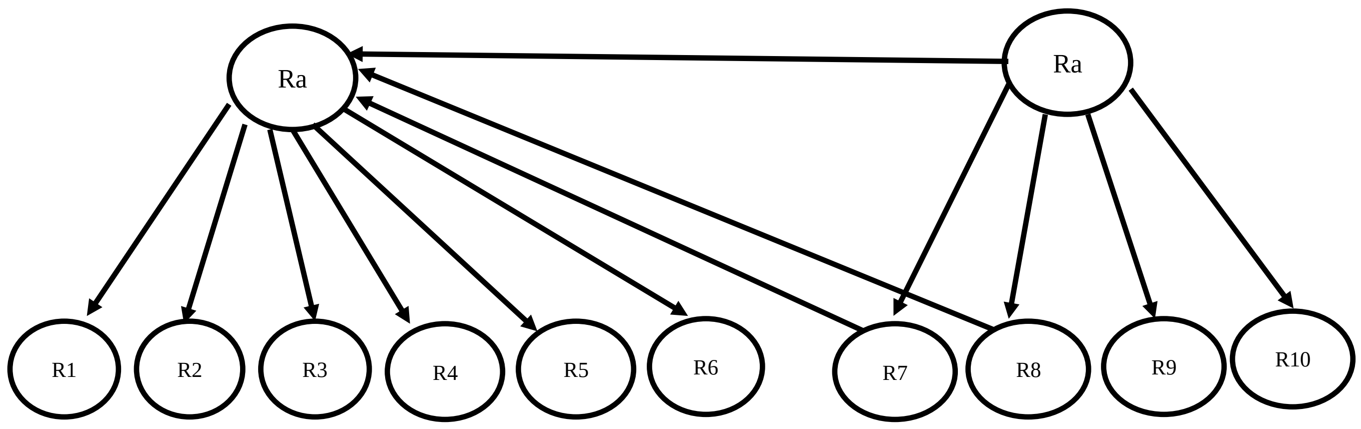

Fig. 6. Dependency of Sub-Requirements of Two user Requirements.

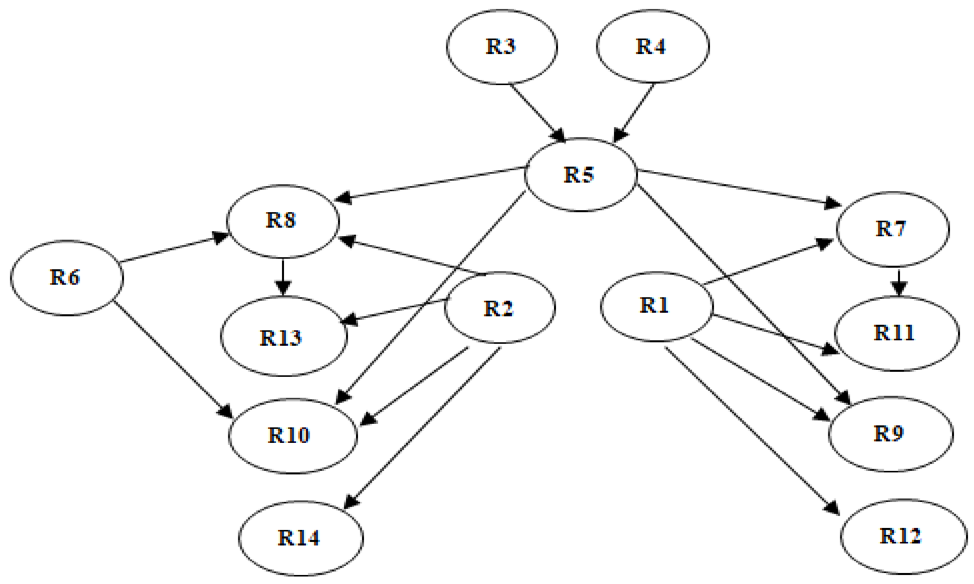

Fig. 7. Graphical Representation of Requirements for Mobile Shop.

3) Priority on the basis of importance of requirement: Although two requirements can have same implementation or chain priority such as R5 and R6 but for analyst the importance of one requirement can be greater than other. E.g. analyst can give more importance to R6 as it is required for too many other requirements or can give importance to R5 as this chain have lesser requirements and which can be deployed in time to user or available for other UR. If user or developer need a particular requirement earlier than priority should be assigned to that particular requirement. We can use any of the existing technique from literature while giving score to requirements on the basis of its importance. But at the end this requirement should be implemented in order of its implementation priority as discussed in section 3.2.

IV. Using Iteration Model

As stated iteration model is SE process model in which all the features or FRs of particular URs are not developed at one time but are implemented in different phases. Some important FRs can be implemented earlier and some can be implemented latterly in next phases. This model is applicable in that case where either all features are not required, or budget is too high that's why clients demand only for important features only.

During implementation, one requirement wait for other requirement and this waiting can delay the project so it will be better to implement the necessary features or FRs that are required for other requirements. Developer will not implement all requirements completely but will implement only necessary requirements.

During this phase when a team member finishes necessary FRs and other members start developing their requirements, the first team member can then implement the FRs in next phase. The detail of the iteration process is explained as.

In Fig. 6, the two URs, Ra and Rb are related to each other such that Rb is required for Ra. From Fig. 6 we can see that not all FRs of Rb are required for Ra but there are some requirements such as R7 and R8 that are required for Ra. Similarly not all but some requirements of Ra will be required for other requirements.

For example, let us suppose, average time of completion of the four FRs of Rb is 40 hours. Suppose average time consume by each requirement of Rb is 10 hours then Ra will wait for 40 hours to Rb. If Rb is implemented and delivers with R7 and R8 only, then waiting time of Ra will reduce to 20 hours and will be implemented in less time.

V. Experiment and Results

In order to validate the significance of iteration model during requirements implementation, experiment was conducted on requirements of mobile phones inventory management system. The presented technique were applied on requirements collected from mobile sales shop and represented with directed graph as shown in Fig. 7. Twenty seven URs were collected from mobile shop using background study and interview as elicitation technique.

A. Implementation Priority

Priority of requirements can be calculated from its position in spanning tree as discussed in Section 3.2. Requirements of particular trees are given below in decreasing order of priority.

1. R4>R5>R7>R11
2. R4>R5>R9
3. R4>R5>R10
4. R4>R5>R8>R13
5. R3>R5>R7>R11
6. R3>R5>R9
7. R3>R5>R10
8. R3>R5>R8>R13
9. R1>R7>R11
10. R1>R11
11. R1>R9
12. R1>R12
13. R2>R8>R13
14. R2>R13
15. R2>R10
16. R2>R14
17. R6>R8>R13
18. R6>R10

TABLE I. Requirements Detail of Mobile Shop

Functional Requirement	**Notation**	**Required for**	**Chain priority**	**Efforts required**	**Assign Team member**
Supplier	R1	R7,R9,R11,R12	4	10 hrs.	A
Customer	R2	R8,R10,R13,R14	4	10 hrs.	A
Product category	R3	R5	4	10 hrs.	A
Company	R4	R5	4	10 hrs.	A
Product	R5	R7,R8,R9,R10	3	10 hrs.	A
Sale man	R6	R8,R10	4	10 hrs.	A
Purchase	R7	R11	2	30 hrs.	B
Sale	R8	R13	2	30 hrs.	B
Purchase return	R9		2	30 hrs.	B
Sale return	R10		2	30 hrs.	B
Supplier debit	R11		1	20 hrs.	C
Supplier payment	R12		3	20 hrs.	C
Customer debit	R13		1	20 hrs.	C
Customer payment	R14		3	20 hrs.	C
Expenses	R15		4		

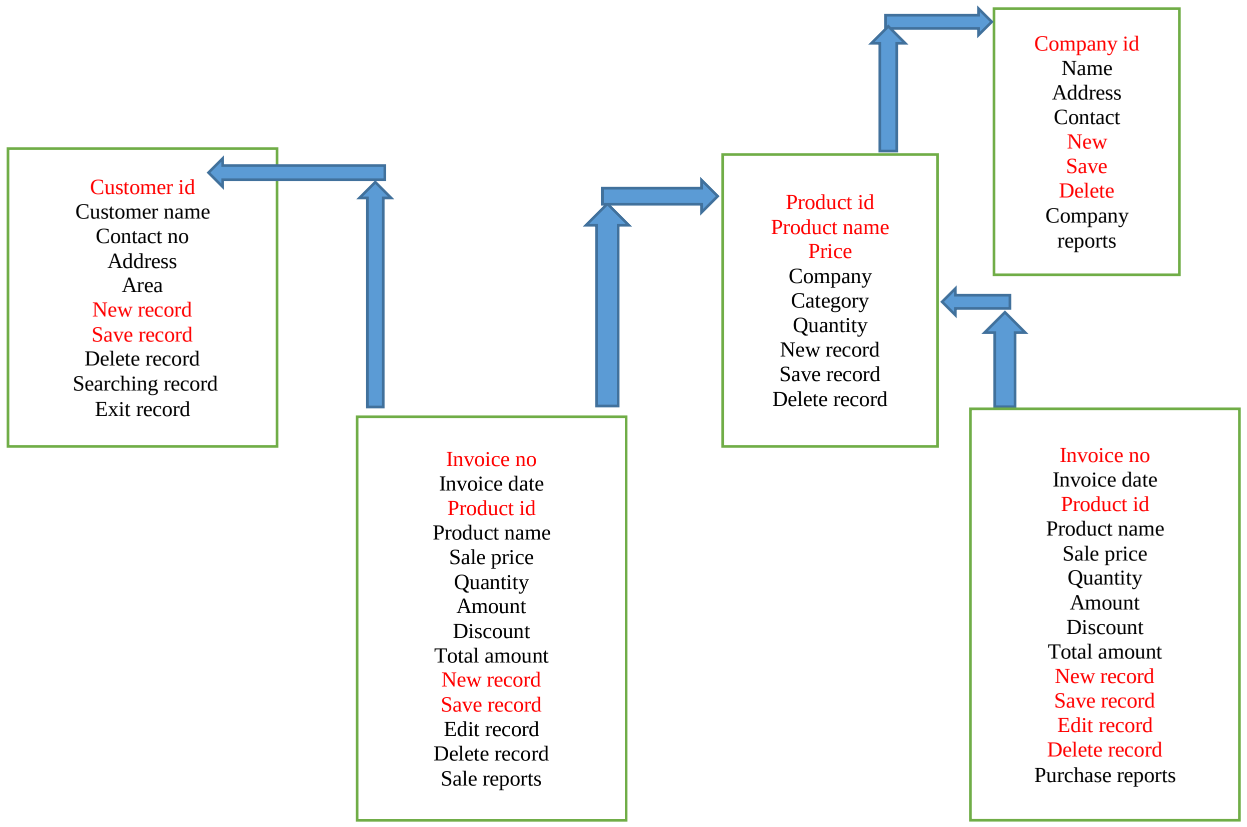

Fig. 8. Dependency of Requirements on Each Other.

From order of requirements as given in Section 5.1, implementation priority or chain priority can be assigned to requirements. Table 1 shows chain priorities of requirements. Suppose we distribute the requirements into three team's members i.e. A, B, C as shown in Table 1. Column 'efforts required' of Table 1 shows the approximated efforts in time hours required to complete requirement. These efforts/hours' time are calculated through time estimation (use case) model. Different authors in their studies have used use case estimation technique.

B. *Requirements Implementation without Iteration Model*

R7, R8, R9, and R10 of B need requirements of A. Similarly requirements of C also need requirements of A and B. Time estimation requirements are given as:

Estimation of A=10+10+10+10+10+10= 60 hrs.
Estimation of B= [60] + 30+30+30+30= 180 hrs.
Estimation of C= [60] + [60] + 20+20+20+20= 200 hrs.

Requirements of B actually take 120 hours but due to its dependency on A, delay of 60 hours occur. Similarly waiting time of C is 60 hours. Total estimation time will be equal to maximum time taken among A, B and C which is 200 hours.

C. *Requirements Implementation with Iteration Model*

Fig. 8 shows URs from Table 1. From Fig. 8, we can see that not all but few FRs are needed for the implementation URs.

In Fig. 8, red requirements are those FRs that are required for other UR which means for implementation of particular UR, red colour requirements should be implemented first. If we implement only red colour FRs instead of whole URs then this pre-requisite UR will be available in less time to other URs.

After implementing only necessary or required FRs, the average estimation time for the URs of A will be 5 hours instead of 10 hrs. Similarly estimation time for the URs of B will be 15 hours. Thus total estimation times of A, B, C will now.

A= 5+5+5+5+5+5=30
B= [30] +15+15+15+15=120
C= [30] + [30] +10+10+10+10=100

From above time estimations, we can see that URs of A are available to B and C on time and similarly URs of B are also available to C on time.

When B start developing R7, then in parallel A can implement the remaining FRs for all URs in its second iteration. But requirements R7, and R8 will not be completely implemented as they are required for C but will follow iteration model and will implement only necessary FRs similar to A. Similarly when C starts implementing requirements, during this B can implement the remaining FRs for all URs in second iteration. This parallel development in iteration or phases will reduce the project delay. Thus after comparing both results, we can conclude that giving importance or more priority to necessary FRs reduced delay and assured delivery of project in less time.

VI. Conclusion

RP play vital role in managing requirements especially when requirements are large in size. Requirements of one module or UR are either dependent or required for the requirements of other r. This dependency cause delay when requirements wait for other requirements and some requirements can wait for too long which can delay the whole project. If we adopt iteration model concept during implementation of requirements, some of the necessary features of requirements can be developed in less estimated time. In this research work, author says that there are few needed requirements that are necessary for other requirements, so instead of implementing all requirements it is better to implement only the necessary requirements of particular user requirement. The proposed idea applied on requirements for mobile shop. The results of with iteration and without iteration are compared. The decrease in total estimation time shows the advantages of using iteration model concept during RP and implementation.

Acknowledgment

This work is supported in partial by the Ministry of Education Malaysia under the Fundamental Research Grant Scheme (FRGS) Vot 1610.

References

[1] M. A. Awais, ‘Requirements Prioritization : Challenges and Techniques for Quality Software Development’, vol. 5, no. 2, pp. 14–21, 2016.

[2] 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.

[3] M. A. A. Elsood and H. A. Hefny, ‘A Goal-Based Technique for Requirements Prioritization’, 2014.

[4] 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.

[5] 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.

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

[7] C. E. Otero, E. Dell, A. Qureshi, and L. D. Otero, ‘A Quality-Based Requirement Prioritization Framework Using Binary Inputs’, pp. 0–5, 2010.

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

[9] P. Achimugu, A. Selamat, R. Ibrahim, and M. Naz, ‘A systematic literature review of software requirements prioritization research’, *Inf. Softw. Technol.*, vol. 56, no. 6, pp. 568–585, 2014.

[10] R. assignment/12. pd. Martin, ‘Iterative and incremental development (iid)’, *C++ Rep.*, vol. 11, no. 2, pp. 26–29, 1999.

[11] M. Alfonso and A. Botia, ‘An iterative and agile process model for teaching software engineering’, *18th Conf. Softw. Eng. Educ. Train.*, pp. 9–16, 2005.

[12] R. Prioritization and U. Hierarchical, ‘Requirements Prioritization Using Hierarchical Dependencies’, pp. 459–464, 2018.

[13] M. Daneva and A. Herrmann, ‘Requirements Prioritization Based on Benefit and Cost Prediction : A Method Classification Framework’, pp. 240–247, 2008.

[14] M. Yaseen, S. Baseer, and S. Sherin, ‘Critical challenges for requirement implementation in context of global software development: A systematic literature review’, ICOSST 2015 - 2015 Int. Conf. Open Source Syst. Technol. Proc., vol. 9, no. 6, pp. 120–125, 2016.

[15] R. M. Liaqat, ‘A Majority Voting Goal Based Technique for Requirement Prioritization’.

[16] A. Felfernig and G. Ninaus, ‘Group Recommendation Algorithms for Requirements Prioritization’, pp. 59–62, 2012.

[17] A. S. Danesh, ‘Requirements prioritization in on-line banking systems : using value-oriented framework’, pp. 158–161, 2009.

[18] 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.

[19] L. Arge and N. Zeh, ‘I / O-Efficient Strong Connectivity and Depth-First Search for Directed Planar Graphs’, 2003.

[20] P. Voola and V. B. A, ‘Study of aggregation algorithms for aggregating imprecise software requirements ’ priorities’, *Eur. J. Oper. Res.*, vol. 259, no. 3, pp. 1191–1199, 2017.

[21] M. Yaseen, S. Ali, Abdulah and N. Ullah. , ‘An Improved Framework for Requirement Implementation in the context of Global Software Development: A Systematic Literature Review Protocol, International Journal of Database Theory and Application, Vol.9, No.6 (2016), pp.161-170.