Prioritization of Software Functional Requirements from Developers Perspective

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

***Abstract*—Prioritizing software requirements is important and difficult task during requirements management phase of requirements engineering. To ensure timely delivery of project, software developers have to prioritize functional requirements. The importance of prioritization increases when size of requirements is big. Software for large enterprises like the Enterprise Resource Planning (ERP) systems are more likely to be developed by a team of software developers where large size requirements are distributed in parallel team members. However, requirements are dependent on each other, therefore development of pre-requisite requirements must be carefully timed and should be implemented first. Therefore, assigning importance and priority to some requirements over others is necessary so that requirements can be available on time to developers. This paper proposes a prioritization approach for functional requirements on the basis of their importance during implementation. The design of research method consists of Analytical Hierarchical Process (AHP) technique based on spanning trees. Through spanning trees, dependent requirements were linked in hierarchical structure and then AHP were applied. As a result of prioritization, requirements were distributed in such a way that dependency among requirements of developers were kept minimum as much as possible so that waiting time of requirements for their pre-requisite were reduced. With reduced effect of dependency in requirements of parallel developers, timely delivery of software projects can be assured.**

***Keywords*—Requirements prioritization; Functional Requirements (FRs); directed graph; spanning tree (ST); Analytical Hierarchical Process (AHP)**

I. Introduction

Requirements Engineering (RE) is a systematic way of collecting software requirements [1][2][3]. There are different types of software requirements [4][5][6]; Business Requirements (BRs) that deal with benefits of implementing requirements, Process Requirements (PRs) that deal with time and cost issues during development, Functional Requirements (FRs) that deal with the actual functionalities of the software, and finally Non-Functional Requirements (NFRs) that deal with requirements such as usability, security, and performance. The collected FRs need proper management in determining issues such as which requirements should be given higher priority, which team member will implement a particular requirement, when the requirements is expected to be delivered, how will the requirements be integrated and other concerns related to requirements management [7][8].

Requirements Prioritization (RP) is a task in RE that focuses on giving priority or ordering a group of requirements [9][10]. Techniques such as cost-value ranking, attribute goal-oriented, and value-oriented approaches work well for BRs in combination with high level FRs [11][12]. FRs are prioritized either from client's perspective or developer's perspective [13][14]. FRs from client's perspective are normally high level requirements that are also known as user requirements (URs). Techniques like the Analytical Hierarchical Process (AHP), binary trees, Genetic Algorithm (GA) are more suitable to prioritize FRs from user perspective [15][16][17]. Meanwhile, techniques like Quality Function Deployment (QFD) and contextual preference-based technique are suggested for prioritizing NFRs [18][19]. Although most of the techniques like AHP work well for small size requirements, they are not scalable and suitable to apply on large requirements. While machine learning techniques and intelligent based techniques such as Artificial Neural Networks (ANN) and SNIPR are suitable for prioritizing large-sized FRs, but they are not suitable techniques to prioritize FRs from developer's perspective where requirements are distributed in parallel development team [20][21][22].

As FRs are not isolated but inter-related so prioritization of FRs is necessary especially when parallel team members are assigned to implement the entire requirements. Giving importance and priority to some requirements over the others is necessary so that pre-requisite requirements can be available on time for other requirements. According to [23], successful projects are not only those that meet all their FRs and NFRs but timely delivery of these requirements is also necessary. Most of big size software's fail to deliver in time, thus proper management and prioritization of FRs from developer's perspective is necessary for successful implementation and delivery of any software project [24].

Although the current prioritization techniques are able to prioritize FRs from user perspective effectively in selecting particular modules or requirements, the same techniques are not either capable or applied to prioritize FRs from developer's perspective when it involves the internal structure and dependency of one requirement on others. Another problem is that most techniques are suitable for prioritizing small-sized requirements but not scalable for large set of requirements. Therefore, a new prioritization is needed for focusing on prioritizing FRs from developer perspective

within the setting of large size requirements especially in parallel developing projects.

Technique like AHP can be applied with pre-defined prioritization rules to FRs but it is not scalable for big-sized requirements. However, we can use technique such as AHP that pairwise compare requirements to prioritize requirements from developer's perspective.

To address this gap, this research work proposes a new approach to prioritize FRs using AHP but based on spanning trees, called the SAHP. The proposed prioritization approach will then be evaluated on FRs of ODOO ERP as case study. Finally, this paper will also investigate the scalability of SAHP in ERP systems by comparing time complexity of the SAHP with existing AHP. The remaining of this paper proceeds as follows. Section 2 presents preliminary studies related to AHP. Section 3 presents the proposed AHP based on Spanning Trees called the SAHP. Section 4 reports evaluation of prioritization experiments using requirements of ERP system. Section 5 presents efficient distribution of requirements in parallel team members and finally Section 6 concludes with some indication for future work.

II. Background Study

Analytical Hierarchical Process (AHP) is an organized decision-making method that is intended to compute complex multi-criteria decision problems. AHP is technique that is also applied efficiently in many other fields such as biology and social sciences for prioritization. In fact, AHP is the utmost frequently discussed prioritization technique within decision making in requirements engineering. AHP is led by comparing all possible pairs of hierarchically categorized entities such as requirements as well as stakeholders for obtaining comparative priorities for all objects [15].

Research in [25] revealed that AHP is capable of improving total time of calculations for pairwise comparisons of the requirements by using eigenvalues and matrix evaluation. The research also proposed Consistency Index (CI) to remove errors like inconsistency. Basically the requirements are arranged in groups called bins in the form of hierarchy. This form of prioritization although be helpful in those cases where requirements are not too much and we need to prioritize with the help of AHP. Number of comparisons will be less as compared to traditional AHP but still it fails large set of requirements.

According to [26], although we assign priorities to FRs, we can also assign priorities on the basis of PRs. The work discussed prioritization of PRs by considering both local priority and perspective priority and proposed the Correlation-Based Priority Assessment (CBPA) that prioritizes requirements from different stakeholder perspectives while to highlight the key issues among them. Two types of requirements were considered (1) from business point of view and (2) from management point of view. Increased profit, lead in competition, reduced cost of development, reduced time to development are business-oriented process requirements while maintaining a project within budget, on schedule, high customer satisfaction, increase productivity are management-oriented process requirements that are considered and prioritized in the research work by author. The relationship between different requirements, its prioritization and impact are discussed in the paper in the form of matrix. Apart from PRs, prioritization of requirements from multiple stakeholder's point of view is also discussed. High priority requirement needs more attention and leads to project success [26].

Apart from fully AHP-based solutions to prioritization of requirements, intelligent-based solution has also been proposed for prioritization of requirements collected from stakeholders by applying machine learning techniques to first group similar requirements, and then apply Artificial Neural Networks (ANN) for further prioritization. Finally, AHP was applied at the end for final comparisons. In first step, before clustering, stakeholders are requested to prepare requirements, then on the basis of profiles of stakeholders and through expert opinions using ANN, requirements can be prioritized [22].

Along with stakeholder preferences, it is also necessary to have prioritization which can handle dependencies in between requirements from user perspective. DRANK is an automated algorithm was presented to perform comparisons based on the importance of dependent requirements and compared the results with AHP and other techniques. Experiments proved that this technique is more efficient and scalable for large size URs [27].

Though many authors have used AHP and tried to reduce number of comparisons from different perspectives, AHP are still unable to cater prioritization of FRs during an active implementation software life cycle. Existing AHP implementation needs user input for pairwise comparison of requirements, while we need this process to be automatic i.e. to take input from its internal structure rather than user. The purpose of this study is to reduce this research gap to prioritize FRs from developer's perspective.

III. Proposed AHP based on Spanning Tree (SAHP)

This section proposes spanning trees based approach to represent FRs and then prioritized with AHP. Spanning tree represents hierarchal order and dependencies of all inter-related requirements. From spanning tree, one can easily pairwise compare requirements with AHP. FRs collected from any sources using appropriate elicitation technique and must be specified in the form of Software Requirement Specification (SRS). In this research, the FRs are represented as alphabets R1, R2, …, Rn and are enclosed in circles as nodes.

A. Spanning Trees

In graph theory, a spanning tree is a subset of graph. A graph G = (V; E) consists of finite set of vertices V and finite set of edges E. Edge is something that connects two vertices. Graphs are useful for the representation of any kind of data in particular sequence [28][29]. This research uses directed acyclic graphs (DAG) rather than cyclic graphs. Requirements are represented as vertices and 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, Fig. 1 shows the graphical representation of requirements through DAG. Cycles in requirements are not possible because if one requirement is needed for the implementation of other requirement than opposite is not possible e.g. if R1 is required for R2 and R2 is required for R3 than it is not possible that R3 will be required for R2 and R1. Graph based approach is also used in one of our previous research study to related FRs [30].

Spanning trees are special 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 [31]. Spanning trees can be formed simply either by performing breadth-first search (BFS) or depth-first search (DFS) or it can be formed directly from adjacency matrix. Because spanning trees use graph-based search algorithms that are only dependent on the number of vertices in the graph, the algorithms are considerably fast [32][33]. The general properties of spanning trees are as follows.

The resulting spanning trees from graph of Fig. 1 are shown in Fig. 2. From a spanning tree, one can easily see the need of particular requirement in relation to other requirements.

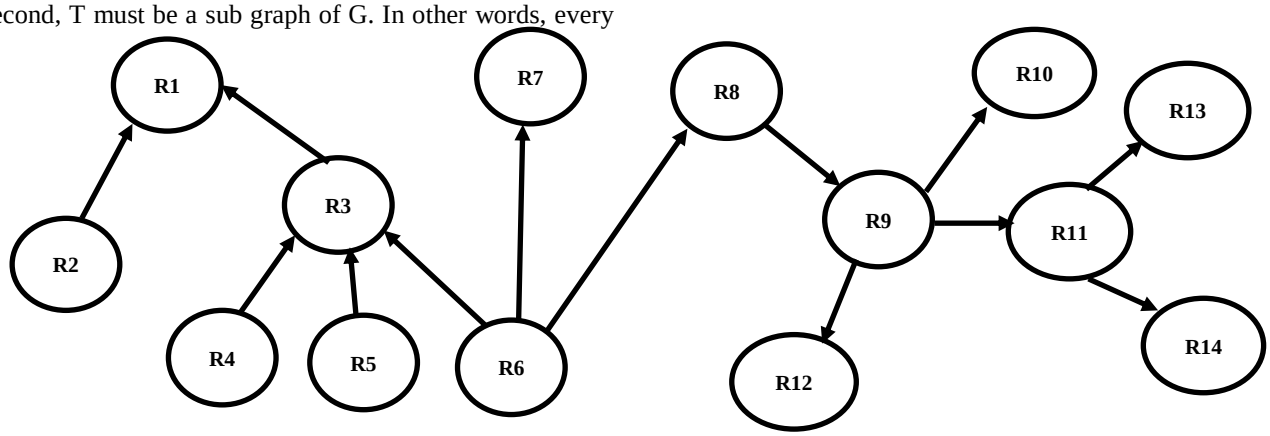

Fig. 1. Graph Connecting Requirements for Making Spanning Tree from Graphs.

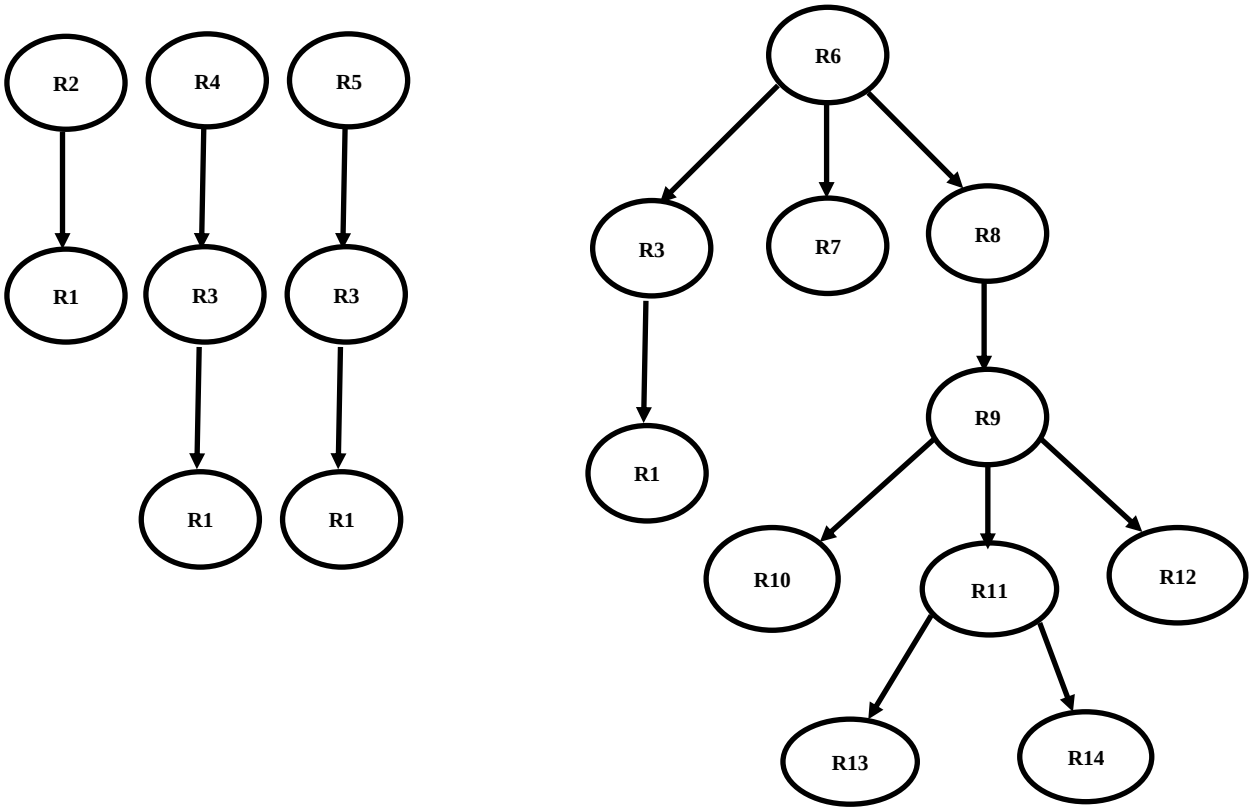

Fig. 2. Tree 1, Tree 2, Tree 3, Tree 4, Respectively.

B. Analytical Hierarchical Process (AHP)

Spanning trees will show the relationship of requirement with other requirements. As shown in Fig. 2, a finite number of spanning trees will be produced from directed graph. Next, AHP will be applied to individual trees or combination of many trees that have common requirements. The main idea is that while applying AHP to spanning tree, only depended requirements will be compared, hence resulting in optimal prioritization in a reduced time. For example, consider the spanning tree with starting node R6 shown in Fig. 2, R6 will be compared with R3, R1 and R7 as it is required for all these requirements. However, R7 will be not compared with R1 or R3 as there is no direct relation with these requirements. In this case, when R6 is compared with R3 or any other requirement, then there is no need to compare between R3 with R6. Requirements that are not depended can be considered as equal during comparison and assigned with value 1. This means with the help of spanning tree, the number of comparisons can be greatly reduced. AHP can be applied to either every spanning tree individually or combination of two or more trees if they have some requirements in common. We have five spanning trees as given in Fig. 2. AHP will be applied to first four spanning trees combined as they are related by some common requirements. First, apply AHP to Tree 5 starting with root R8 and then apply AHP to combined four trees. Table I shows requirements of Tree 5 for comparison and calculation.

From Table I, we can see that we can put value either 1 or greater than 1 while comparing any two requirements. We can only put 1 or greater value where 1 represents equal priority requirements and value greater than 1 represents those requirements that have not equal priorities. For instance, we can use values such as 2, 3, 4, …, n for requirements that are not equal in priorities. If we increase the value, the difference in both requirements will be increased. The value 2 is taken for requirement that is needed for other requirement. For instance, if R1 is required for R2 and R2 is required for R3, then we will put 2 for R1 against R2 and will put 4 for R1 against R3. The value 1 is taken for requirements that have either equal priority or not related and 2 against those requirements that need this particular requirement as well and value ½ for the reverse case. In this case, as R8 is required for R9, therefore the value is 2 against R9 for R8. Priority value for each requirement against other requirements is shown in Table I e.g. priority of R9 against R8 is 0.5 which means priority of R8 is double as compare to R9. For independent requirements like R10 and R12, we put value 1 because these requirements have no relation.

Next, the task is to calculate normalized values for each requirement by dividing the values of each column value in Table I by column sum. Column sum for each column is shown in Table II. For example, the value 1 in the first row and the first column will be divided by 2.5, which comes to 0.4. Consequently, normalized values for each requirement are shown in Table II. The column sum2 represents the averaging over normalized values for each row. The same process is then repeated for the combined four trees together and the values obtained in shown in Table III.

TABLE I. PAIRWISE COMPARISON FOR TREE 5

	R8	R9	R10	R11	R12	R13	R14
R8	1.000	2.000	4.000	4.000	4.000	8.000	8.000
R9	0.500	1.000	2.000	2.000	2.000	4.000	4.000
R10	0.250	0.500	1.000	1.000	1.000	1.000	1.000
R11	0.250	0.500	1.000	1.000	1.000	2.000	2.000
R12	0.250	0.500	1.000	1.000	1.000	1.000	1.000
R13	0.125	0.250	1.000	0.500	1.000	1.000	1.000
R14	0.125	0.250	1.000	0.500	1.000	1.000	1.000
Sum	2.500	5.000	11.000	10.000	11.000	18.000	18.000

TABLE II. NORMALIZATION AND AVERAGING AND FOR TREE 5

	R8	R9	R10	R11	R12	R13	R14	Sum2/ priority	Out of 1 (x = sum/7)	Z= (x/2)
R8	0.400	0.400	0.360	0.400	0.360	0.440	0.440	2.800	0.400	0.200
R9	0.200	0.200	0.180	0.200	0.180	0.220	0.220	1.400	0.200	0.100
R10	0.100	0.100	0.090	0.100	0.090	0.055	0.055	0.600	0.090	0.045
R11	0.100	0.100	0.090	0.100	0.090	0.110	0.110	0.700	0.100	0.050
R12	0.100	0.100	0.090	0.100	0.090	0.055	0.055	0.600	0.090	0.045
R13	0.050	0.050	0.090	0.050	0.090	0.055	0.055	0.440	0.060	0.030
R14	0.050	0.050	0.090	0.050	0.090	0.055	0.055	0.440	0.060	0.030

TABLE III. CALCULATING PRIORITIES OF TREE 1 TO TREE 4 (COMBINED)

	R1	R2	R3	R4	R5	R6	R7	Sum2/ priority	Out of 1 (y = sum/7)	Z = (y/2)
R1	0.055	0.076	0.050	0.040	0.040	0.043	0.125	0.430	0.060	0.030
R2	0.110	0.153	0.105	0.170	0.170	0.173	0.125	1.000	0.140	0.070
R3	0.110	0.153	0.105	0.086	0.086	0.086	0.125	0.751	0.110	0.055
R4	0.220	0.153	0.210	0.170	0.170	0.173	0.125	1.221	0.200	0.100
R5	0.220	0.153	0.210	0.170	0.170	0.173	0.125	1.221	0.200	0.100
R6	0.220	0.153	0.210	0.170	0.170	0.173	0.250	1.346	0.200	0.100
R7	0.055	0.153	0.105	0.170	0.170	0.086	0.125	0.864	0.123	0.062

The column sum2 also shows the priority value of every requirement of the spanning tree, in particular, or combination of spanning trees. The sum of these sum2 values will equal to number of requirements i.e. 7. These values can be divided on number of requirements to find priority of requirements out of 1. For considering whole set of requirements i.e. In 14 requirements, priority value will be divided on 2 (2 is sum value for all requirements priorities). Column value z for Table II and Table III shows priority out of 14 requirements. Priority out of 14 is calculated. Similarly, for calculating priority of requirement in 100, value 100 is multiplied.

C. Time Complexity of SAHP

Time complexity of AHP depends on total number of pairwise comparisons. With spanning tree, total number of comparisons are reduced because of limited number of relations. Either we consider combination of all spanning trees in one table or individual trees, the number of comparisons of dependent or related requirements will be always same (from adjacency matrix one can see how much relations exists). The number of comparisons in all cases will depend on how much relations of requirements in graph exist. In this example, as only 20 relations are possible, the total number of comparisons will equal to only 20. Therefore, in this way, number of necessary comparisons are reduced from n*(n-1) / 2, which was from 91 to only 20 in this example. This reduction in value shows the advantage of using spanning trees for related depended requirements only. Overall values and calculations during comparing requirements can be reduced by considering individual trees for prioritization as explained.

For given requirements set, maximum relations that can exists are equal to ((n-1) + (n-2) + (n-3) + . . . + (n-n)), where n are total number of requirements. This is possible when all requirements are connected point to point in chain like structure such that one requirement is dependent on other requirement. The value of n will be decremented and will be added until it reaches to 0. In such case, total number of comparisons will become n*(n-1) / 2 which is equal to number of comparisons of AHP. The minimum number of relations will be 0 in any requirements set. In such case, priority of all requirements will be consider to be equal i.e. 1. Fig. 3 shows number of comparisons of two techniques i.e. AHP without spanning trees by considering all requirements and AHP with spanning trees. Let's take 10 requirements. Minimum possible relations are 0 while maximum relations can be 45. Any number of relations can be possible between 0 and 45. The orange linear line of Fig. 3 shows that number of comparisons in this proposed approach is directly proportional to number of relations. It is equal to 45 i.e. case of AHP where maximum relations exist. In small set of requirements where requirements are few in amount, this is possible that maximum relations exist (number of relations reaches number of requirements) such that each requirement is point to point connected with other requirement but we rarely can see such number of relations in large set of requirements like ERP.

From this discussion, it can be concluded that by comparing only the depended requirements through spanning tree, the number of comparisons and calculations can be greatly reduced. Therefore, although total comparisons of dependent requirements are same in all cases, but as the entire project, the number of comparisons and calculated normalized values are not same due to independent requirements.

D. Requirements Priority

Priority is assigned to requirements on the basis of its position in spanning tree i.e. how much they are needed and dependent on other requirements. Requirements need can either increase breadth-wise or depth-wise. In either case, priority can increase but priority values in both cases can be different. Similarly, priority of requirement can decrease when requirements are dependent and wait for other requirements.

AHP can be applied for calculating priority of requirement on the basis of how much they are depended or required for other requirements. AHP is simple and accurate prioritizing technique that can find priority of requirements by comparing pairwise all requirements together. If requirement let say R1 is required for R2 and R2 is required for R3, then priority of R1 can be taken as double of R2 or it can be said that priority of R1 is two times as compared to R2 while R1 priority is 4 times as compare to R3. The following scenarios show different cases of requirements behavior as they change when applied with AHP.

Scenario 1: In this scenario priority of requirement is determined when its need for other requirements increases breadthwise. Breadthwise contain all requirements on same level with same priority. Two cases can be considered here, one with seven requirements and other with five requirements and calculate priorities.

Case 1: In this case, R1 is required for six other requirements with all requirements on same level with same priority as shown in Fig. 4.

Through AHP, we have calculated priority of R1 by comparing all seven requirements together which is equal to 1.75. R1 is considered to be double in priority as compare to individual requirements during pairwise comparison. The priority of all other requirements is shown in Table IV. Table IV summarizes priority values for all requirements.

Case 2: In this case, R1 is required for four other requirements with all requirements on same level or priority in Fig. 4. Now priority of R1 is reduced to 1.32 as shown in Table IV.

Scenario 2: In this scenario, requirements size increases depth wise. In case 01 of scenario 1, R1 is required for six other requirements in depth wise structure such that one requirement is depended on other requirement as shown in Fig. 5. In case 02, number of requirements that need R1 are reduced from six to four. Priority of R1 in first case comes out 3.5 while in second case it is 2.21. The priority of R1 in 2nd case of scenario 2 (required for four requirements) is still greater than case 01 of scenario 1 (required for six requirements). This shows priority increases with greater ratio depth wise and this has advantage because in scenario 1, R1 is available to all requirements after implementation but in scenario 2, it is not available to all requirements e.g. R7 in scenario 2 can't be implemented when R6 is not available but in scenario 1, all requirements are dependent on R1.

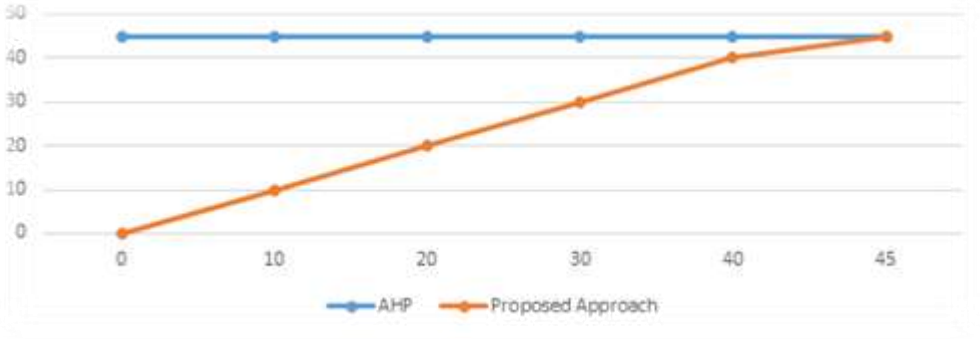

Fig. 3. Comparison of AHP and Proposed Approach.

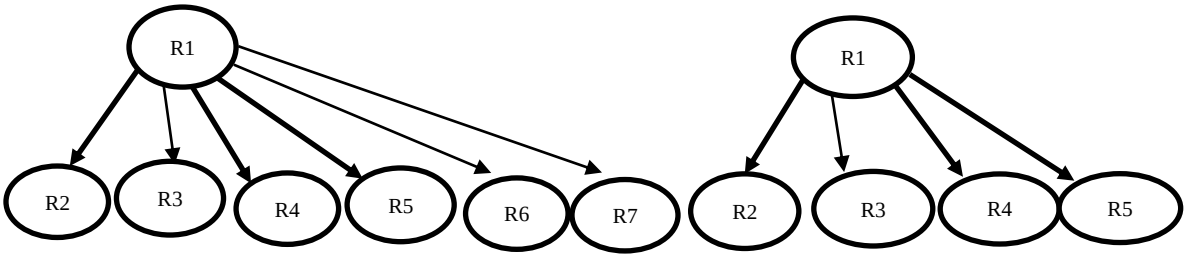

Fig. 4. Breadth-Wise Increase of Requirements.

TABLE IV. COMPARISON OF PRIORITY OF REQUIREMENTS AS RESULT OF AHP

Requirements	Scenario 1		Scenario 2		Scenario 3
	Case 1	Case 2	Case 1	Case 2	
R1	1.750	1.32	3.500	2.210	1.0
R2	0.875	0.68	1.750	1.245	0.5
R3	0.875	0.68	0.875	0.758	0.5
R4	0.875	0.68	0.437	0.515	0.5
R5	0.875	0.08	0.210	0.400	0.5
R6	0.875	x	0.105	0.900	2.0
R7	0.875	x	0.500	0.900	2.0

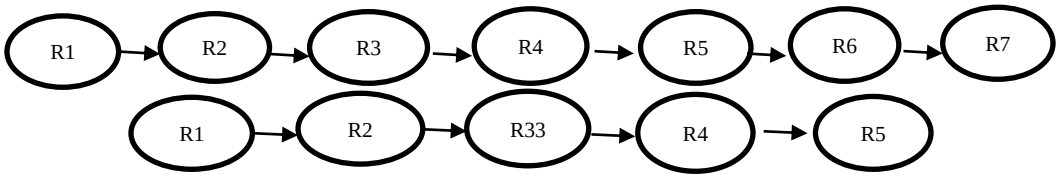

Fig. 5. Depth-Wise Increase of Requirements.

Scenario 3: Priority of requirement decreases when its dependency on other requirements increase. The reason is that during comparison against other requirements, sum of values are reciprocal of 1. Fig. 6 shows the priority of R1 against R6 and R7 will be equal to ½. The sum of reciprocal values will reduce the priority of requirement. Priority of R1 is now 1, which is minimum as compared to all cases. Priority of other requirements are shown in Table IV.

From values given in above Table IV, it can be concluded that requirement priority is associated with its increasing size but the ratio in which it increases depth wise is greater than breadth wise and it should be increase with high ratio in depth wise as compare to breadthwise because in breadthwise, the pre-requisite requirement is available for all requirements and the delay is not too much as compare to the case of depth wise where pre-requisite requirement is not available for all requirements and by delaying this requirement can delay the implementation of its requirements more in case of parallel developing project.

Similarly, if number of pre-requisite requirements and number of requirements for which particular requirement is needed are equal then priority of requirement will be equal. For example, in Fig. 7, the number of backward and forward requirements for R1 are equal, in all cases priority of requirement will be equal. With AHP, we have calculated priority of R1 that is 0.84 for all cases of Fig. 7.

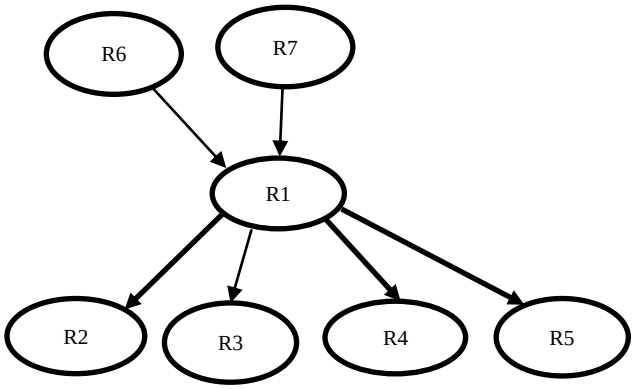

Fig. 6. Number of Pre-Requisite Connected with Requirement.

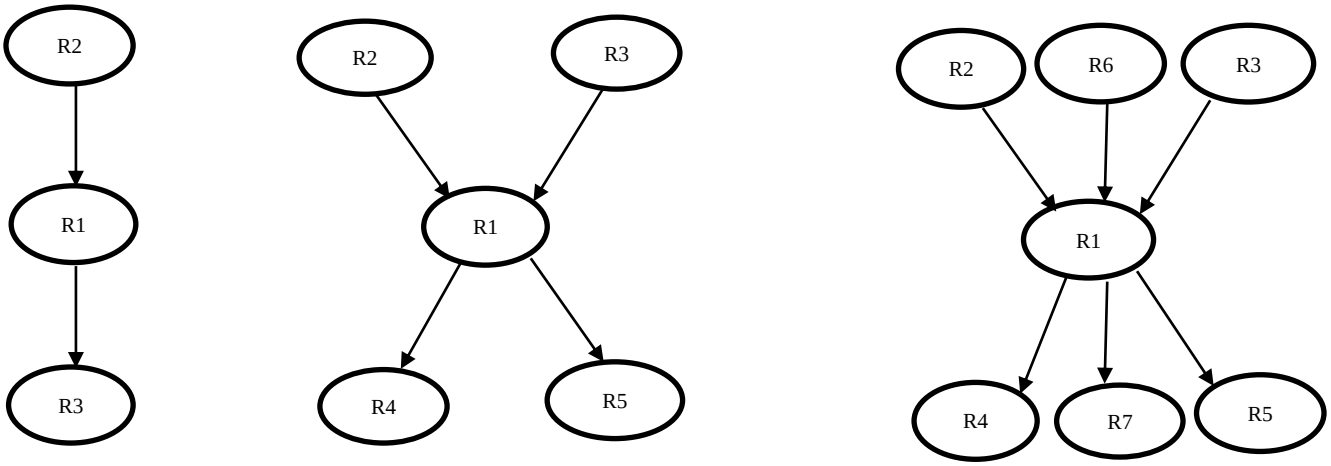

Fig. 7. Distribution of Requirements with Same Ratio.

IV. Validation of SAHP on ODOO ERP

SAHP was evaluated on requirements of On Demand Open Object (ODOO). ODOO is open source ERP software system that is used by millions of users for managing hundreds of possible enterprises and their resources. In many of research studies, authors used different modules of ODOO ERP [34][35][36]. In ERP system, all modules are integrated which shows that all the requirements should be inter-related. Modules of ERP are highest level URs that are further comprised of low level FRs. With spanning tree, we can relate FRs that can belong to any module. Module is just high level abstraction to which requirements of same nature belong e.g. customer and supplier creation are FRs that belong to HR module while customer sale and supplier sale are FRs that belong to sale management module. With spanning tree, we can relate these FRs that belong to different modules. Thus spanning tree does not show abstraction or high level representation of requirements because it relates only different requirements that belong to particular module. Selection of particular modules have impact on priority of their FRs. This means variations in selecting different modules by users have impact on FRs structure. The suggested prioritization approach will be applied on the FRs of ODOO to prioritize them. The modules of ERP consists of 96 FRs for this study as shown in Table V.

TABLE V. REQUIREMENTS OF ODOO ERP FOR HR MODULE

Notation	Requirement	Module No.	Required For	Tree	Notation	Requirement	Module No.	Required For	Tree
R1	employee creation	1	R81, R25, R23, R67, R2, R4, R10, R11, R12, R17, R18, R20, R21, R22, R7, R9, R8	T1	R69	sale return view	3		T10
R2	public information's of employee	1		T1	R42	purchase	4	R51, R59	T4, T5, T6
R3	employee personal info	1			R59	purchase view	4		T4, T5, T6
R4	contact info	1		T4	R60	purchase return	4	R68	T4
R5	job position	1		T2, T3	R68	purchase return view	4		T4
R6	department	1	R5, R61, R67	T2, T3	R34	product	5	R42, R60, R66, R35, R70, R71, R90,	
R7	job information's	1			R66	stock ledgers	5		T4
R8	manager	1	R5, R24, R67		R70	product transfer in	5		T4
R9	coach	1			R71	product transfer out	5		T4
R10	contract information's	1		T1	R56	company	5		
R11	contract reference information's	1		T1	R90	manufacturing orders	5		T4
R12	salary generation	1	R21	T1, T18	R24	project management	6	R26, R27, R28, R29	T3
R22	hr expenses	1	R23	T1	R25	add team members	6		T1
R23	hr expenses detail	1		T1	R26	extra information's	6		T3
R33	customer detail	1	R73, R55, R36, R35, R61, R64, R39	T10	R27	project stages	6		T3
R37	sales persons	1	R58, R63,R35		R28	view current task	6		T3
R41	supplier detail	1	R44, R65, R72, R42, R52, R60		R29	create a task	6	R31	T3
R43	sales man	1	R42,R44	T5	R30	extra information's	6		
R57	region	1	R58	R8	R31	tasks stages	6		T3
R58	area	1	R35	R7, R8	R93	directories for documents	7		T11
R80	job position in recruitment	1		T4	R94	documents history	7	R96	T15
R81	job	1		T1, T2	R95	documents attachments	7	R96	T14
R82	appraisal form	1			R91	fleet management	8	R92	T11
R83	create a job position	1			R92	vehicle repairing	8		
R84	recruitment form	1			R13	salary rules	9		T18

R85	job selection process	1			R14	salary structure	9	R12	T16
R86	link tracker	1			R15	salary categories	9	R12	T17
R87	mass mailing	1			R16	registers	9	R12, R13	T18
R88	contact	1			R21	hr payroll process	9		T1, T16, T17, T18
R89	business pipeline	1			R17	apply for leave	10	R19, R20	T1
R38	customer receipts	2		T10	R18	allocation request	10		T1
R39	customer payment	2	R55, R38	T10	R19	leave approval	10		T1
R40	supplier receipts	2		T12	R20	leave summary	10		T1
R44	supplier refund	2		T5, T6	R46	bank statement	11	R47	T9
R45	supplier payment	2	R40	T12	R47	bank detail	11	R49, R50, R53	T9
R52	supplier payment	2			R48	cash registers	11		
R53	journals accounts	2	R54	T9	R49	put money in	11		T9
R54	chart of accounts	2		T9, T10	R50	put money out	11		T9
R55	analytic accounts	2	R54	T10	R51	profit and lost	11		T4, T5, T6
R63	salesman ledgers	2		T7	R75	compose message	12		
R64	customer ledgers	2		T10	R76	message inbox	12	R79	T13
R65	supplier ledgers	2		T6	R77	message draft	12		
R67	hr expense management	2		T1, T2, T3	R78	sent messages	12		
R74	balance sheet	2			R79	message searching	12		T13
R32	customer invoice	3	R36		R72	order to suppliers	13		T6
R35	sale	3	R61, R62, R32		R73	order from customer			T10
R36	customer refund	3		T10	R96	documents attachment			T14, T15
R61	sale return	3	R69	T10					
R62	sale view	3							

A. Results and Discussion

Results of prioritization of ODOO ERP requirements after applying suggested framework using AHP and spanning tree combination have been calculated. Requirements are prioritized by applying the same criteria discussed.

1) Spanning trees: As the result, 8 spanning trees are constructed (T1, T2 up to T18) while 19 requirements are independent requirements which are neither required nor dependent on other requirements. The root and the detail requirements are given in Table VI. Spanning trees are categorized into different groups which are made on the basis of common requirements in different spanning trees. For example, in T1 and T2, the common requirement is R67. Similarly, R21 is common in T1, T16, T17 and T18. Six groups (A, B, C, D, E, and F) of different trees are made which are shown in Table VI.

2) Applying AHP to spanning trees: The column "priority" as shown in Table VII shows priority of requirements as a result of applying AHP on spanning tree. Priority of requirements in spanning trees are calculated. We have calculated priority of these requirements out of 100 as shown in Table VII.

TABLE VI. COMBINING REQUIREMENTS OF SPANNING TREES

Group	Tree	Root	Requirements	Efforts (Hours)
A	T1 T2 T3 T16 T17 T18	R1 R6 R8 R14 R15 R16	R81, R23, R25, R2, R4, R10, R11, R12, R17, R18, R19, R20, R22, R21, R67 R5, R67, R81, R5, R67, R24, R26, R27, R28, R29, R31 R21 R21 R12, R13, R21	720
B	T9 T8 T7 T10 T4 T5 T6	R46 R57 R37 R33 R34 R43 R41	R47, R49, R50, R53, R54 R58 R58, R63, R35, R61, R62, R32, R36, R69 R73, R55, R54, R35, R61, R62, R32, R36, R69, R64, R38, R39, R42, R51, R59, R60, R66, R68, R70, R71, R80, R90, R35, R61, R62, R32, R36, R69 R42, R51, R59, R44 R42, R51, R59, R44, R52, R60, R68	1230
C	T11	R92	R93	50
D	T12	R45	R40	50
E	T13	R76	R79	50
F	T14 T15	R95 R94	R96 R96	80
Individual requirements				470
Total efforts in man hours				2650

TABLE VII. REQUIREMENTS PRIORITY OF ODOO

Notation	Combined Priority (Out of 100)	Separate Priority (Out of 100)	Notation	Combined Priority (Out of 100)	Separate Priority (Out of 100)
R1	1.66	2.22	R62	0.72	0.84
R2	0.96	0.96	R69	0.62	0.79
R3	1.03	1.03	R42	0.9	0.91
R4	0.96	0.96	R59	0.77	0.75
R5	0.98	0.82	R60	0.9	0.90
R6	1.08	1.20	R68	0.81	0.80
R7	1.03	1.03	R34	2.37	2.90
R8	1.65	2.68	R66	0.9	0.90
R9	1.03	1.03	R70	**0.93**	0.94
R10	0.96	0.96	R71	**0.9**	0.90
R11	0.96	0.96	R56	1.03	1.03
R12	0.97	0.88	R90	0.9	0.90
R22	0.96	0.96	R24	1.16	1.34
R33	2.72	3.10	R25	0.96	0.96
R37	2.57	2.70	R26	0.92	0.73
R41	1.47	2.056	R27	0.92	0.73
R43	1.20	1.414	R28	0.92	0.73
R57	1.031	1.045	R29	0.96	0.78
R58	0.78	0.79	R30	1.03	1.03
R80	0.86	0.79	R31	0.92	0.61
R81	0.98	0.92	R93	0.72	0.72
R82	1.03	1.03	R94	1.23	1.23
R83	1.03	1.03	R95	1.23	1.23

R84	1.03	1.03	R91	1.03	1.03
R85	1.03	1.03	R92	1.37	1.37
R86	1.03	1.03	R13	0.97	0.90
R87	1.03	1.03	R14	1.01	1.07
R88	1.03	1.03	R15	1.01	1.07
R89	1.03	1.03	R16	1.1	1.44
R38	0.87	0.81	R21	0.90	0.88
R39	1.10	1.12	R17	1.02	1.06
R40	0.72	1.03	R18	0.96	0.96
R44	0.87	0.85	R19	0.95	0.88
R45	1.37	1.37	R20	0.96	0.88
R52	0.90	0.90	R46	1.6	2.63
R53	1.08	0.79	R47	1.15	1.17
R54	0.70	0.41	R48	1.03	1.03
R55	0.90	0.81	R49	0.89	0.60
R63	0.84	0.85	R50	0.89	0.60
R64	0.93	0.87	R51	0.77	0.75
R65	0.91	0.90	R75	1.03	1.03
R67	0.92	0.83	R76	1.37	1.37
R74	1.03	1.03	R77	1.03	1.03
R32	0.77	0.88	R78	1.03	1.03
R35	1.34	1.47	R79	0.72	0.72
R36	0.62	0.79	R72	0.91	0.90
R61	0.77	0.64	R73	0.93	0.88
R23	0.96	0.96	R96	0.618	0.62

B. Time Estimation

Time estimation is time taken by particular requirement to complete its implementation. Every requirement consume certain amount of efforts on the basis of which time can be calculated. Many models are suggested by authors for calculating efforts and time estimation of requirements and projects. We applied USE CASE point (UCP) estimation technique which was simple in use and more appropriate for our requirements. The UCP estimation method was presented initially in 1993 by Karner estimates efforts in person-hours based on use cases that primarily specify FRs of a system [11][12]. Use cases are assumed to be developed from scratch, be sufficiently detailed and typically have less than 10-12 transactions. The method has previous been used in numerous industrial software development projects. There have been promising outcomes and the method was highly accurate than expert estimates in industrial trials.

UCP defines the functional scope of the system to be developed. Attributes of a use case model may therefore serve as measures of the size and complexity of the functionality of a system. After following all steps of USE case estimation technique, effort in hours for each requirement is calculated. After approximation, we have divided requirements into three categories as follows.

- First category contains requirements that take approximately 20 hours to complete its implementation. This is time just needed to implement requirement with functionalities. This time contain unit and integration testing time.
- Second category contain requirements that contain requirements that take approximately 30 hours to complete its implementation.
- Third category contain requirements contain requirements that take approximately 60 hours to complete its implementation.

Completion time of particular module will be sum of time taken by all its requirements. This time reduces when the project is to be developed by parallel team members. But total actual time can exceed calculated time in parallel development projects because requirements are interrelated to each other's and waiting time for particular requirements can cause delay in projects. The purpose of prioritization is to minimize the delay or waiting time.

V. Distribution of Requirements in Parallel Developers

From results of prioritization we can conclude that not only priority value and order of requirements is necessary for reducing delays and assuring timely delivery of project but

distribution of requirements in team members is also necessary. Total delivery time of project is equal to maximum time taken by any team member to implement all requirements. Distribution of requirements as shown in Table VIII are not uniform e.g. actual time estimation of requirements of A = 410 hours, B = 610 hours, C = 880 hours and D = 670 hours. Total delivery time of the project can exceed from 880 hours due to waiting time which is the maximum time of team member C but total time can't be less than 880 hours. This is because C is given those requirements which take more time in hours. Efficient distribution will be in that case where everyone is given requirements with same efforts. The generalized formula we can make for equal distribution is as follows.

$$\text{Total efforts (for any team member)} = \text{Total efforts (man hours)} / 4.$$

Where total efforts (man hours) = Total efforts (for all requirements starts from R1 to Rn).

From this formula, we will get average time for every team member which becomes 660 hours. If every team member gets no more than 660 hours than in ideal case total estimation time of delivery of project can be 660 hours which is reduced from 880 hours. This means further adjustment will be needed to reduce time estimation more and for this purpose some requirements of C can be assigned to A.

Along with equal distribution of requirements, we should reduce dependency among requirements of different team members as much as possible. Requirements of A that are required for C can be adjusted and can be assigned to C. Similarly, some requirements of C can be adjusted and implemented by A. From the spanning tree, one can easily identify which requirements are dependent on each other, so dependent requirements can be assigned to same team members. In ideal case, distribution of requirements will be uniform and dependency between different team member requirements will be zero.

The best way to distribute requirements is thus assigning requirements of whole spanning tree to same team member. Requirements of spanning trees should be adjusted in such a way that every team member get requirements with equal weight of man hours. Team members can either implement big spanning tree requirements or requirements of many small spanning trees. If some trees requirements are distributed in more than one member than requirements should be prioritized in order to reduce the waiting time.

A. Combining and Splitting the Spanning Trees

If two or more than two trees have some common requirements than we can combine two trees and consider as one group. The reason is that common requirements are depended on requirements of more than one trees requirements and hence this dependency can increase waiting time and cause delays in parallel developing projects. Splitting process is taken when tree size is either big or difficult to assign all its requirements to single developer or sometimes small size trees are split to assure equal distribution of requirements. Table VIII shows how different trees are combined. Six groups were made as result of combining trees with common requirements. Total efforts in man hours for each group are also shown below. It is better to split tree at edge where two trees are combined for assigning requirements to different developers. For example, T9 and T10 are combined with R54, so the tree can be break here.

TABLE VIII. COMBINING AND SPLITTING OF REQUIREMENTS OF SPANNING TREES

No of team members	Efforts per team (hours) with equal distribution	Splitting of trees	Combining trees	Time estimation for implementing requirements	Time completion with prioritization
01	2650	NIL	All trees are considered	2650 hours	2650 hours
02	1325	NIL	Developer 1: [B + C + D] Developer 2: [A + E + F + 470 individuals]	1230 hours	1230 hours
03	880	Breaking of Group B: 1230 = 350 + 880	Developer 1: [880 of B] Developer 2: [350 of B + C + D + E + F + 300 individuals] Developer 3: [A+ 170 individuals]	880 hours	880 hours
04	660	Breaking of Group B: 1230 = 640 + 590 Breaking of Group A: 720 = 660 + 60	Developer 1: [640 of B + 20 individuals] Developer 2:[590 of B + C + 20 individuals] Developer 3:[660 of A] Developer 4: [60 of A + D + E + F + 430 individuals]	670 hours	670 hours

Common requirements can be assigned to any tree. Common requirements normally get low priority as they are dependent on other requirements. Similarly, T4 and T6 are combined with R42. In this case, we can break here (edge of R42) in order to equally distribute requirements. Common requirements can be adjusted with any tree requirements. But for equal distribution in terms of time efforts, especially in case where a single tree is quite large and needs to split, then we will split it. In such case, try to split an edge and assign those requirements that have significantly high priority difference from their parent requirements. It is better to split at edge where there exists quite big difference in priorities between two requirements. E.g. if tree T4 is to be split, there can many options, either to split edge at R60, R66, R35, R70, R71 or R90.

The difference in priorities between R34 and R35 is less as compare to other requirements because R35 is high priority requirement, so splitting at R35 can increase waiting time if R35 is assigned to different team member. Splitting at edge of low priority requirement and assigning it to other team member will decrease the effect of dependency and waiting time. For maintaining balance and equal distribution, more than one trees can be split e.g. T3 can be split along with T4 but at point where there exists quite difference in priority. Thus from values of SAHP, distributed priority can be determined requirements can be easily assigned to team members such that effect of dependency in requirements become low as much as possible.

B. Distribution of Requirements

Requirements will be distributed in such a way that there does not exist either relation between requirements of different team members or if relation exist, then requirements should be prioritized so that waiting time can be reduced and timely implementation of requirements can be assured. Few cases are considered for distribution of requirements as shown below.

1) Distribution of requirements in 2 team members: In distributing requirements based on efforts in man hours per team member, the value will be equal to 1325 hours i.e. half of total 2650 hours. There is no need to split any tree or group of trees because different groups can be managed to produce total efforts of 1325 hours. We can assign requirements of groups B, C, D to one developer for implementation and groups A, E, F along with 470 individual requirements to second developer. In this way two different developers will get independent requirements with no relationship between any two requirements.

2) Distribution of requirements on 3 team members: In this case efforts per team member will be equal to 880 hours. While distributing requirements on three developers, it is must to split large tree or group of trees to assure equal distribution of requirements on developers. Requirements with total efforts of 350 hours were separated from group B. The separated requirements from any tree of group based on values of SAHP. Group B requirements after splitting will remain with efforts of 880 hours. In this way two sub groups are made. Sub-group with 350 hours can be adjusted with groups C, D, E, F and 300 hours of individual requirements to comprise total of 880 hours. Similarly, requirements of group A can be implemented along with remaining individual requirements i.e. 170 hours. In such way equal distribution of requirements can be assured. After distributing requirements, it is necessary to prioritize it to reduce waiting time and delays in project.

3) Distribution of requirements in 4 team members: To assure equal distribution of requirements, every team member will get requirements of 660 hours. For equal distribution, we can split group B into two subgroups with 640 and 590 hours. Similarly group A can be split into two subgroups with 660 and 60 hours. Splitting Group A were necessary as requirements of A were exceeded from 660 hours. Efficient distribution and prioritization of requirements reduces the effect of dependency between requirements and waiting time in parallel developing projects which results in timely delivery of projects. Separated requirements are shown in Table IX.

TABLE IX. SEPARATED REQUIREMENTS OF SPANNING TREES

Number of Developers	A	B	C	D	E	F
2	nil	nil	nil	nil	nil	nil
3	nil	(R66, R70, R71, R80, R90, R42, R43, R44, R38, R52, R72) OR (R41, R44, R65, R72, R52, R42, R51, R59, R52, R60, R68)	nil	nil	nil	nil
4	R4, R10, R11	R66, R70, R71, R80, R90, R42, R43, R44, R38, R52, R72, R41, R64, R68, R65, R60	nil	nil	nil	nil

VI. Conclusion

This paper proposed an approach for prioritizing FRs using AHP based on spanning trees. The proposed approach of SAHP has been presented in detail with evaluation on ODOO ERP system. The proposed framework is capable of prioritizing large-sized FRs while in active development cycle. As FRs are inter-related, so prioritization will help in easy arrangement of requirements. Similarly, apart from its implementation priority, in which the requirement is pre-requisite for other requirements, if we compare two requirements that are totally independent of each other, then deciding about which requirement is more important is a very important task. Importance of requirement was measured from how much it can reduce delay or waiting time.

Prioritizing and implementing important requirements decrease not only total estimation time but also decrease non-critical delay. Although non-critical delay does not increase estimation time of the project, it affects the waiting time of requirements. Another big problem that needs to be solved is that how much the proposed technique is scalable of handling and prioritizing large requirements size. Prioritizing large size requirements on the basis of its importance was solved using AHP and spanning tree in combination. AHP is used because it can solve dependency issues of requirements as it statistically compares pairwise for each and every requirement against other requirements. Requirements were represented with directed graph and spanning tree. From spanning tree, it became easy to decide about not only which requirement was necessary for other requirement but it became easy to compare all neighbor requirements that belong to same tree. AHP was applied to each tree separately and only depended requirements were scored value greater than 1. Priority of all other requirements during comparison were considered equal.

The results were obtained and were evaluated on parallel developing requirements of ODOO ERP. From different cases for prioritized and un-prioritized requirements, we showed that the proposed framework not only deal with big size requirements but reduce all possible delays in projects. We have shown that how spanning tree can help in equal and efficient distribution of requirements in parallel developing team members so that the effect of dependency and waiting time of requirements can be reduced. In future, we aim to do more industrial based experiments in order to validate framework on big projects and get feedbacks from industry.

Acknowledgment

This paper is supported by Research Fund E15501, Research Management Centre, Universiti Tun Hussein Onn Malaysia.

References

[1] M. Yaseen, S. Baseer, and S. Sherin, ‘Critical Challenges for Requirement Implementation in Context of Global Software Development : A Systematic Literature Review’, pp. 120–125, 2015.

[2] M. Yaseen, M. Bacha, and Z. Ali, ‘REVIEW PAPER COORDINATION AND COLLABORATION PRACTICES IN GLOBAL By’, vol. 14, no. 2, 2020.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] M. Yaseen, A. Mustapha, A. U. Rahman, S. Khan, and W. Kamal, ‘Importance of Requirements Prioritization in Parallel Developing Software Projects’, vol. 9, no. 2, pp. 171–179, 2020.

[14] 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.

[15] 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.

[16] 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.

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

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

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

[20] N. Setiani and T. Dirgahayu, ‘Clustering Technique for Information Requirement Prioritization in Specific CMSs’, 2016.

[21] A. Perini, A. Susi, and P. Avesani, ‘A Machine Learning Approach to Software Requirements Prioritization’, vol. 39, no. 4, pp. 445–461, 2013.

[22] 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.

[23] H. Taherdoost and A. Keshavarzsaleh, ‘A Theoretical Review on IT Project Success / Failure Factors and Evaluating the Associated Risks’, 4th Int. Conf. Telecommun. Informatics, Sliema, Malta, no. August, pp. 80–88, 2015.

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

[25] M. A. Iqbal, A. M. Zaidi, and S. Murtaza, ‘A new requirement prioritization model for market driven products using analytical

hierarchical process', DSDE 2010 - Int. Conf. Data Storage Data Eng., pp. 142–149, 2010.

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

[27] 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.

[28] 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.

[29] S. Ma, J. Li, C. Hu, X. Lin, and J. Huai, 'Big graph search : challenges and techniques', 2015.

[30] M. Yaseen, A. Mustapha, S. Qureshi, A. Khan, and A. U. Rahman, 'A Graph Based Approach to Prioritization of Software Functional Requirements', vol. 9, no. 3, pp. 64–73, 2020.

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

[32] M. Usman, D. Sakethi, R. Yuniarti, and A. Cucus, 'The Hybrid of Depth First Search Technique and Kruskal ' s Algorithm for Solving The Multiperiod Degree Constrained Minimum Spanning Tree', no. Icidm, pp. 0–3, 2015.

[33] S. Dhingra, 'Finding Strongly Connected Components in a Social Network Graph', vol. 136, no. 7, pp. 1–5, 2016.

[34] E. Reitsma, P. Hilletofth, and U. Mukhtar, 'Implementation of enterprise resource planning using Odoo module sales and CRM . Case study : PT Ecosains Hayati Implementation of enterprise resource planning using Odoo module sales and CRM . Case study : PT Ecosains Hayati', 2017.

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

[36] 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.