# Prioritization of Software Functional Requirements: Spanning Tree based Approach

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

***Abstract*—Requirements prioritization shows significant role during effective implementation of requirements. Prioritization of requirements is not easy process particularly when requirements are large in size. The current methods of prioritization face limitations as the current prioritization techniques for functional requirements rely on the responses of stakeholders instead of prioritizing requirements on the basis of internal dependencies of one requirement on other requirements. Moreover, there is need to classify requirements on the basis of their importance i.e. how much they are needed for other requirements or dependent on other requirements. Requirements are first represented with spanning trees and then prioritized. Suggested spanning tree based approach is evaluated on requirements of ODOO ERP. Requirements are assigned to four developers. Time estimation with and without prioritization are calculated. The difference in time estimation with prioritization and without prioritization shows the significance of prioritization of functional requirements.**

***Keywords*—Requirements prioritization; functional requirements; spanning tree***

## I. Introduction

Requirement engineering is important and critical phase of software engineering which deals with how requirements should be collected from users in more discipline and systematic way [1][2][3]. The collected requirements should be properly managed before implementation and in this regard prioritization of requirements becomes more essential [4][5]. Requirements prioritization deals with assigning priority to requirements [6]. As software development becoming more complex in the recent years, prioritization have carried high significance in managing requirements successfully [7]. Elicitation and prioritization are two core activities of RE [8][9][10]. In a large software development projects such as an Enterprise Resource Planning (ERP), requirements are huge and prioritization process becomes much difficult [11]. When the stakeholder wishes to implement each and every requirement within a limited time and a limited budget, prioritization of the requirements become necessary [12][13][14]. As all types of requirements are inter-dependent with each other, there is critical need of collaboration among software developers and stakeholders during requirements prioritization especially when requirements [13]. Many techniques are suggested by authors to prioritize requirements, some techniques are suitable for prioritizing business requirements [15][16], some techniques are suitable for functional requirements and some techniques are suitable for nonfunctional requirements (NFRs) [17]. No such technique is either applied or suggested for functional requirements that can solve dependency issues of requirements in parallel developing large software systems for timely delivery. The objective of current research study is to propose an efficient approach of prioritizing software functional requirements from development perspective.

The remaining of this paper proceeds as follows. Section 2 presents background study conducted. Section 3 presents design of the research methodology. Section 4 discusses requirements prioritization. Section 5 presents case study conducted and finally Section 6 concludes with some indication for future work.

## II. Background Study

AHP is the most common and applied technique identified from literature. AHP is scalable for small size requirements and face time complexity problems when size of requirements is large. As AHP pairwise compare each requirement against all other requirements so time complexity increases with increase size of requirements. Total number of comparisons with AHP is equal to n * (n-1)/2 e.g. if requirements are 10, then total comparisons will be 45. If total requirements are 20, total comparisons will become 190 and thus number of comparisons increases with increase size of the requirements [18][19].

Cumulative voting (CV) or 100 dollar is a technique [20][21] in which 100 dollars or points are given to the stakeholders and they have to assign these dollars or points to specific requirement. Requirements that are assigned more dollars will acquire high priority while those requirements that are assigned with less dollars will acquire low priority. Even though this technique is very simple in use, but it works better for small size requirements where determining the priorities of requirements is not tough and when size of requirements is too large, it becomes difficult to prioritize with voting method. This technique is user based technique because it is subjective to the inputs of users. Another big issue that can arise with this method is that stakeholders may assign dollars to some requirements that are not so important and may ignore some high priority requirements. Stakeholder can assign zero to some requirements. When the number of stakeholders are more than one, then distributing dollars on requirements may cause conflicts.

Using numerical assignment (NA) technique, requirements are categorized into high, medium and low priority groups and numerically assigned requirements to these groups. Inside each group, all requirements are considered to be same in priority [22].

Group discussions and decision are also helpful to prioritize requirements. After getting remarks from stakeholders or experts, group of experts will analyze the requirements in which each group member will score for that. At the end on the basis of group decision and score, all the requirements will be prioritized accordingly [23].

In another research study, requirement ranking function with graph is applied using binary search algorithm for comparing the customer feedback and thus priority and with original order of requirements in priority list is calculated. The main goal is to reduce the difference between true and estimated value of priority [24].

Assigning priority to NFRs is that much important as much of assigning priority to functional requirements. Using method similar to AHP, author has defined three steps for assigning priority to NFRs. 1) Based on pairwise comparison, assign values to different NFRs. 2) Based on functional requirements, assign priority values to NFRs. 3) Calculate priority by matrix multiplication. Well-organized prioritization of NFRs is presented [25].

Machine learning approach is presented during requirements elicitation phase in order to reduce the efforts during prioritization. Case-Based Ranking (CBR) is discussed which combines stakeholder preferences with requirements ordering approximations computed through machine learning approaches [6].

Using fuzzy logic and decision tree, the idea and detail evaluation of framework is presented which can examine various prioritization techniques. It is an intelligent approach for prioritizing newly upcoming requirements by getting inputs as parameters. On the basis of different parameters under different scenarios, this technique will decide that which technique is best under specified conditions. The condition can be type or size of requirements [26].

Although a lot of work is done to prioritize different types of requirements but still no work is done to prioritize requirements from developers perspective especially in parallel development where multiple team members work in parallel and assigning low priority to important requirements can delay whole project.

## III. Design of Research Methodology

Fig. 1 shows the step by step approach of resign design.

Requirement elicitation process

Elicitation is the first phase for collecting user requirements for any software system. Various elicitation techniques such as background study, interview are applied to collect requirements from users. The quality of software product and its timely delivery depends on how well requirements are collected.

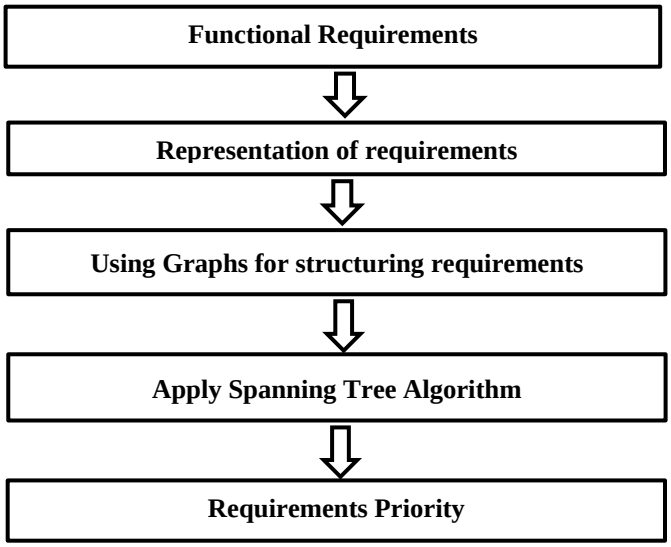


Fig. 1. Research Design Process.

### A. Representation of Requirements

The collected requirements from users are represented with symbols e.g. R1, R2, R3, Rn. with surrounded round shape as shown in Fig. 2.

### B. Using Directed Graph for Structuring Requirements

A graph is a pictorial diagram of a set of objects that are inter-related. The interrelated entities are characterized by points named as **vertices**, and the links that inter-relate the vertices are called **edges**.

- **Nodes** are typically represented by circles or ovals (though technically they can be any shape of your choosing). In this study requirements represents nodes i.e. R1, R2, R3 represent nodes of the graph.
- **Edges** are the connections or links between the nodes. An edge links two nodes. They are generally represented by lines, or lines with arrows.

Directed acyclic graph is a graph without having any cycles (a cycle is a complete circuit). When succeeding the graph from node to node, you will certainly not visit the same node twice. A directed acyclic graph is an acyclic graph that has a direction as well as an absence of cycles [27][28] [29]**.**

- **Vertices set** = {R1, R2, R3, R4, R5, R6, R7}.
- **Edge set** = {(R1, R2), (R1, R3), (R2, R4), (R2, R5), (R3, R6), (R4, R7), (R5, R7), (R6, R7)}.

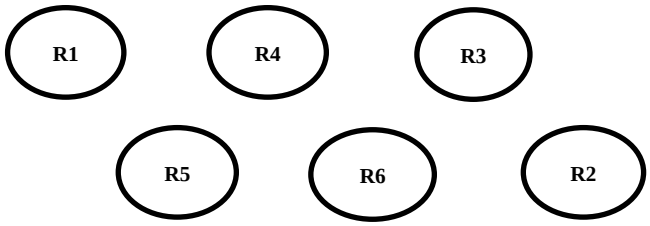


Fig. 2. Representation of Requirements using Specific Notations.

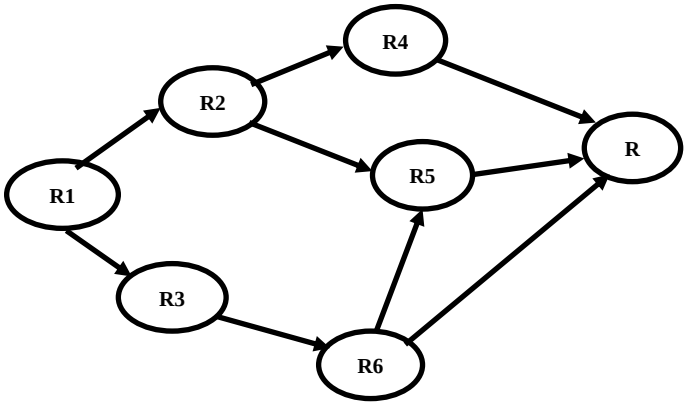


Fig. 3. Requirements Linked through Directed Acyclic Graphs.

Requirements inside directed graph can be either depended on other requirements or either needed for other requirements or can be both. Dependent requirements are those requirements that rely on other requirements for implementation and needed requirements are those on which other requirements are dependent. The requirement that points to some requirement is needed requirement while requirement on the arrow side is dependent. In Fig. 3, R1 is required for completion of R2 and R3, while R2 is required for the completion of R4, R5 and R3 is required for completion of R6. Similarly for the implementation of R5, R2 and R6 are needed. R4, R5 and R6 are needed for R7.

The reason for considering directed graphs instead of undirected is because undirected graphs points in both direction and it is not possible that a requirement is consider both depended as well as needed at the same time. During requirements implementation, cycles are not possible e.g. if there are three requirements such as R1, R2 and R3 as shown in Fig. 3. Consider if R2 is required for R1, R6 is required for R4 and R4 is again required for R2, R3 and R5 then cycle will create which means for R1 implementation, R2 should be implemented first but for R2, R4 should be implemented first.

### *C. Spanning Tree Formation*

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.

Spanning trees can be found in linear time by simply performing breadth-first search or depth-first search. These graph search algorithms are only dependent on the number of vertices in the graph, so they are quite fast.

There are a few general properties of spanning trees. Find all possible trees from graph. Starting point will be the requirement which is needed for other requirements such that the pre requisite requirements will come to the top (parent). The pre-requisite requirements will be the parent of all those requirements for which they are needed.

In below graph of Fig. 4, R2 is required for R1 but R1 is not required for other requirements so first tree will include only R1 and R2. Similarly R3 is also required for R1 and R4, R5 and R6 all are needed for R3, so from this point onwards three trees are possible. First will contain R4, R3, R1, second R5, R3, R1 and the third one with R6, R3, R1. As R6 is also needed for other requirements, so the child's of R6 will increase which will include R7 and R8. R8 is now needed for R9, so R9 will become child of R8 and further R9 is required for R10, R11 and R12 so all these will be the child's of R9. R11 is required for R13 and R14. R10 and R11 are child requirements of R9.

Thus by following either depth first searching (DFS) or breadth first searching (BFS) algorithm, the resulted spanning trees are shown in Fig. 5. With DFS, after the visit of R6, it can visit either of R3, R7 and R8, suppose it visit R3, and then it can't visit any of R7 and R8 before the child node of R3. After that it will visit R7, as it has no further child's, so it will go and visit R8 and then R9. Now using DFS, it can visit any of R10, R11 and R12. After visit of R10, it will visit R11 which is the child of R10. From R11, it will visit R13 and then R14. In last it will visit R12.

The same problem can be solved through BFS. Let's take example of tree 3. By applying BFS, it visits R6, then R3, R7 and R8 and then R1 and R9. After R9 visit, it visit R10, R12 and R11 and at the end it will visit R13 and R14.

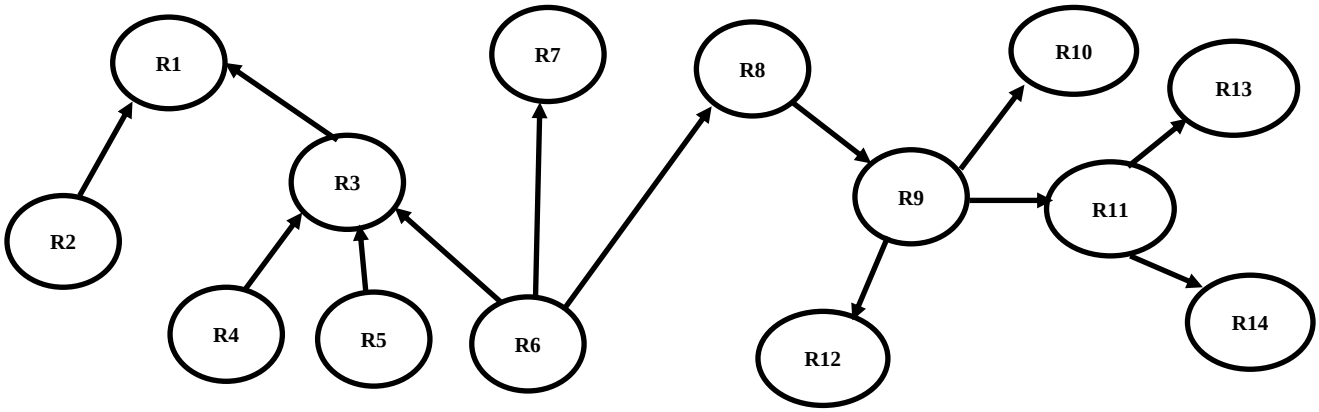


Fig. 4. Graph Connecting Requirements for Making Spanning Tree from Graphs.

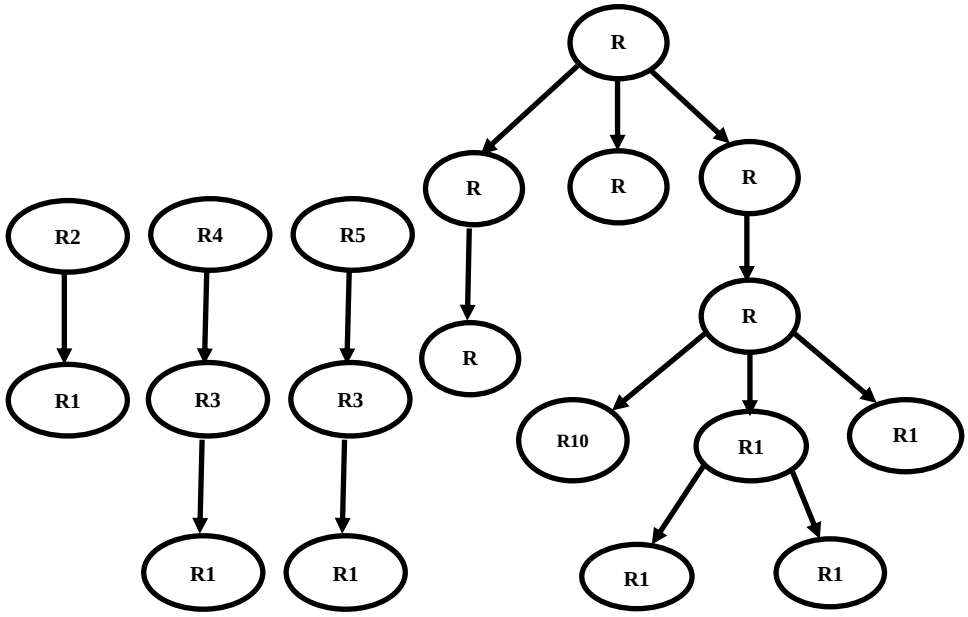


Fig. 5. Tree 1, Tree 2, Tree 3, Tree 4 Respectively.

## IV. Requirements Prioritization

In Fig. 5, priority of pre-requisite requirements will be greater than priority of requirements for which they are needed e.g. R4 priority will be greater than R3 while R3 priority will be greater than R1. In R4 and R6, priority of R6 should be greater because it is needed for greater number of requirements. In this case, R6 is although needed for three requirements R3, R7 and R8 but these requirements are further needed for other requirements. So requirement priority will be calculated from its overall need either directly or not. Similarly dependency of requirement in spanning tree will reduce its priority, e.g. priority of R13 and R14 will be lower than priority of R10 and R12 because R10 and R12 are dependent on three while R13 and R14 are dependent on four requirements.

Requirements prioritization has a significant role in successful implementation of software projects. Quality and success of any software projects is not only associated with how much software meets its functional requirements but it is also associated with timely delivery of software projects. Timely delivery of any project can be assured when time estimation of whole project is correct and along with it requirements waiting time for other requirements is minimum.

Requirements of parallel team developers can be inter-related and this can increase the waiting time of requirements if pre-requisite requirements are not available in time.

Requirements need for other requirements can easily be calculated by counting number of child nodes and similarly requirements dependency on other requirements can be calculated by counting number of parent nodes of requirement in tree. E.g. in tree 04 of Fig. 5, the parent nodes of R9 are two and child nodes are total five. In this case, although child nodes of R9 are three but R11 further has two more child nodes, so in this way total child become five for R9. In similar way, parent node of R9 is R8 and parent of R8 is R6, so in this way, total requirements on which R9 is dependent are two. Now question arise what will be the net priority of R9 in this case because only from child nodes, the priority of requirements can't be determined because if we ignore dependent requirements than results can be biased e.g. if there are two requirements and both are needed for same number of other requirements but dependency of these requirements on other requirements is different, then requirement that is less dependent on other requirements will get higher priority. From the difference of child nodes and parent nodes values, net priority of requirement can be calculated. The net priority of R9 will be equal to 3 in this way. Similarly priority of R3 will be equal to 2 because parent nodes of R3 are three and child nodes are one, the difference will equal to 2.

Now if child nodes of requirements such as R1 are zero and parent nodes are more than zero. In such case priority will be in negative. E.g. here the priority of R7 will be equal to -1. Priority of independent requirements will be 0 and thus requirements with negative priority will be given low priority as compare to requirements with 0 priority. Priority of R1 will equal to -5. For R1, parent R3 is repeated in three trees, but it will be counted as 1.

Negative or zero priority of requirements can be adjusted by adding positive same number with all requirements. E.g. To adjusted priority of R1 from -5 to 1, value 6 can be added to this. Thus value 6 will be added with all requirements in similar way. Net priority of R3 will be equal to 8. Requirements order with adjusted and without adjustment of priorities will be same.

## V. Case Study

The suggested approach is evaluated on requirements ODOO open source ERP system. It consist of ninety six (96) high level functional requirements. The prioritization algorithm is applied to these high level functional requirements only. The detail is given below in Table I.

TABLE. I. REQUIREMENTS DETAIL OF ODOO ERP

| Requirements | Required for | Requirements | Required for | Requirements | Required for |
|---|---|---|---|---|---|
| R1 (Employee) | R2,R4,R10, R11,R12, R17,R18, R20, R21, R22, R23, R25, R67, R81, | R33 (customer detail) | R24, R35, R36, R39,R55, R61, R64, R73, R90, R55 | R65 (supplier ledgers) | |
| R2 (Public information's of employee) | | R34 (products detail) | R35, R42, R60, R66,R70,R71, R91, R61 | R66 (stock ledgers) | |
| R3 (Employee personal info) | | R35 (sale) | R32, R51, R61, R62, | R67 (HR expense management) | |
| R4 (Contact info) | | R36 (customer refund) | | R68 (purchase return view) | |
| R5 (Job position) | | R37 (Sales persons) | R35, R36, R58, R63, | R69 (sale return view) | |
| R6 (Department) | R5, R81, R67 | R38 (customer receipts) | | R70 (Transfer In) | |
| R7 (Job information's) | | R39 (customer payment) | R38, R55 | R71 (Transfer out) | |
| R8 (Manager) | R5, R24, R67 | R40 (supplier receipts) | | R72 (order to suppliers) | |
| R9 (Coach) | | R41 (supplier detail) | R42, R44, R52, R60, R65, R72 | R73 (order from customer) | |
| R10 (Contract information's) | | R42 (purchase) | R51, R59 | R74 () | |
| R11 (Contract reference information's ) | | R43 (Sales man) | R42, R44 | R75 (Balance sheet) | |
| R12 (Salary generation) | R21, | R44 (supplier refund) | | R76 (compose message) | R79 |
| R13 (Salary rules) | | R45 (supplier payment) | R40, | R77 (message inbox) | R80 |
| R14 (Salary structure) | R12 | R46 (bank statement) | R47 | R78 (message Draft) | |
| R15 (Salary categories) | R12 | R47 (bank detail) | R49, R50, R53 | R79 (sent messages) | |
| R16 (Registers) | R12, R13, | R48 (cash registers) | | R80 (message Searching) | |
| R17 (Apply for leave) | R19,R20, | R49 (put money in) | | R81 (Job position in recruitment) | |
| R18 (Allocation request) | | R50 (put money out) | | R82 (Job) | |
| R19 (Approval) | | R51 (Profit and lost) | | R83 (appraisal form) | |
| R20 (Leave summary) | | R52 (supplier payment) | | R84 (create a job position) | |
| R21 (HR payroll) | | R53 (Journals accounts) | R54 | R85 (Recruitment form) | |
| R22 (HR Expenses) | | R54 (Chart of accounts) | | R86 (Job selection process) | |
| R23 (HR expenses) | | R55 (Analytic accounts) | R54 | R87 (Link tracker) | |
| R24 (Project management) | R26, R27, R28, R29 | R56 (company) | | R88 (Mass mailing) | |
| R25 (Add team members) | | R57 (region) | R58 | R89 (contacts) | |
| R26 (Extra information's) | | R58 (Area) | | R90 (business pipeline) | |
| R27 (Project stages) | | R59 (purchase view) | | R91 (manufacturing orders) | |
| R28 (View current task) | | R60 (purchase return) | R68, | R92 (fleet management) | R93, |
| R29 (create a task) | R31, | R61 (sale return) | R69 | R93 (Vehicle repairing) | |
| R30 (Extra information's) | | R62 (sale view) | | R94 (Directories for documents) | R96 |
| R31 (Tasks stages) | | R63 (salesman ledgers) | | R95 (Documents history) | R96 |
| R32 (customer invoice) | R36 | R64 (customer ledgers) | | R96 (Documents attachments) | |

From the information's of Table I, directed graph can be easily drawn and can identify all possible number of spanning trees. Table II shows the resulted 18 spanning trees from requirements of ODOO ERP. Requirements are further categorized into groups on the basis of common requirements.

By applying prioritization algorithm as explained above, requirements of ODOO ERP are prioritized accordingly as shown in Table III.

Calculated priorities of requirements as a result of apply prioritization technique using spanning tree are shown in Table III. Priorities are then adjusted such that minimum priority is 1. An experiment was conducted on parallel developing software requirements of ODOO using priority values from Table III. Requirements of software are distributed in four developers i.e. A, B, C and D as shown in Table V in such that there exists dependency between requirements of developers. Before applying prioritization algorithm, efforts in hours needed to implement all these individual requirements are calculated using USE CASE POINT estimation technique as shown in Table IV. Difference of total estimation time of these developers and overall project before and after prioritization will show significance of spanning tree based prioritization approach.

Requirements are distributed in such way that requirements of C and D are dependent on A while requirements of B are totally independent as shown in Table V.

**Case 1:** In this case, all requirements of A, B, C and D are arranged in ascending order of priorities i.e. requirements are not prioritized (except pre-requisite requirements that should be implemented first) as shown in Table V.

**Case 2:** In this case, all requirements of A, B, C and D of Table V are prioritized in descending order of priorities such that requirements of every team member are fully prioritized.

Time estimation based on sum of time estimation of all requirements for each developer in both cases is shown in Table VI.

Total estimation time of the project depends on the maximum time completion of any developers. From Fig. 6, for case 01, maximum time taken by developer D is 1750 hours and for case 02, maximum time taken by developer C is 940 hours. The delay or exceed in time estimation is case 01 is due to waiting time of requirements for their pre-requisites while after prioritization, delway is reduced due to reduction in waiting time for requirements.

TABLE. II. RESULTED SPANNING TREES

| Tree | Root | Requirements |
|---|---|---|
| T1 | R1 | R81, R23, R25, R2, R4, R10, R11, R12, R17, R18, R19, R20, R22, R21, R67 |
| T2 | R6 | R5, R67, R81, |
| T3 | R8 | R5, R67, R24, R26, R27, R28, R29, R31 |
| T4 | R14 | R21 |
| T5 | R15 | R21 |
| T6 | R16 | R12, R13, R21 |
| T7 | R46 | R47, R49, R50, R53, R54 |
| T8 | R57 | R58 |
| T9 | R37 | R58, R63, R35, R61, R62, R32, R36, R69 |
| T10 | R33 | R73, R55, R54, R35, R61, R62, R32, R36, R69, R64, R38, R39, |
| T11 | R34 | R42, R51, R59, R60, R66, R68, R70, R71, R80, R90, R35, R61, R62, R32, R36, R69 |
| T12 | R43 | R42, R51, R59, R44 |
| T13 | R41 | R42, R51, R59, R44, R52, R60, R68 |
| T14 | R92 | R93 |
| T15 | R45 | R40 |
| T16 | R76 | R79 |
| T17 | R95 | R96 |
| T18 | R94 | R96 |

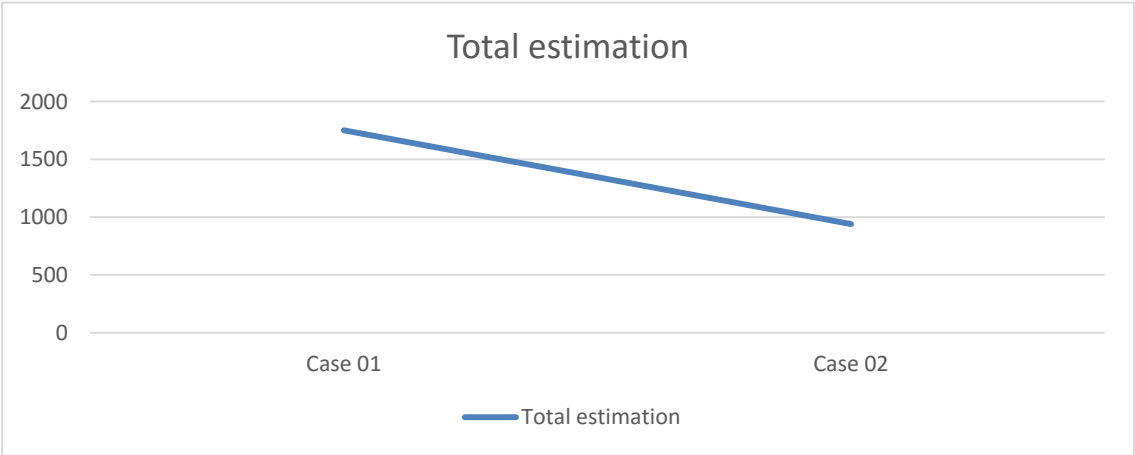


Fig. 6. Total Estimation Time of Requirements for Case 01 and Case 02.

TABLE. III. PRIORITY AND IMPORTANCE VALUES ASSIGNED ON THE BASIS OF CHILD NODES

| Requirement | Total child's/priority | Adjusted priority | Requirement | Total child's/priority | Adjusted priority |
|---|---|---|---|---|---|
| R1 | 24 | 30 | R49 | -2 | 4 |
| R2 | -1 | 5 | R50 | -2 | 4 |
| R3 | 0 | 6 | R51 | -2 | 4 |
| R4 | -1 | 5 | R52 | -1 | 5 |
| R5 | -1 | 5 | R53 | -1 | 5 |
| R6 | 3 | 9 | R54 | -3 | 3 |
| R7 | 0 | 6 | R55 | -1 | 5 |
| R8 | 8 | 14 | R56 | 0 | 6 |
| R9 | 0 | 6 | R57 | 1 | 7 |
| R10 | -1 | 5 | R58 | -1 | 5 |
| R11 | -1 | 5 | R59 | -5 | 1 |
| R12 | 0 | 6 | R60 | 0 | 6 |
| R13 | 0 | 6 | R61 | -3 | 3 |
| R14 | 0 | 6 | R62 | -4 | 2 |
| R15 | 0 | 6 | R63 | -1 | 5 |
| R16 | 0 | 6 | R64 | -1 | 5 |
| R17 | 1 | 7 | R65 | -1 | 5 |
| R18 | -1 | 5 | R66 | -1 | 5 |
| R19 | 0 | 6 | R67 | -1 | 5 |
| R20 | -2 | 4 | R68 | -2 | 4 |
| R21 | -3 | 3 | R69 | -5 | 1 |
| R22 | -1 | 5 | R70 | 0 | 6 |
| R23 | -1 | 5 | R71 | -1 | 5 |
| R24 | 4 | 10 | R72 | -1 | 5 |
| R25 | -1 | 5 | R73 | -1 | 5 |
| R26 | -2 | 4 | R74 | 0 | 6 |
| R27 | -2 | 4 | R75 | 0 | 6 |
| R28 | -2 | 4 | R76 | 1 | 7 |
| R29 | -1 | 5 | R77 | 0 | 6 |
| R30 | 0 | 6 | R78 | 0 | 6 |
| R31 | 0 | 6 | R79 | -1 | 5 |
| R32 | -3 | 3 | R80 | -2 | 4 |
| R33 | 13 | 19 | R81 | -1 | 5 |
| R34 | 16 | 22 | R82 | 0 | 6 |
| R35 | 3 | 9 | R83 | 0 | 6 |
| R36 | -5 | 1 | R84 | 0 | 6 |
| R37 | 8 | 14 | R85 | 0 | 6 |
| R38 | -2 | 4 | R86 | 0 | 6 |
| R39 | 2 | 8 | R87 | 0 | 6 |
| R40 | -1 | 5 | R88 | 0 | 6 |
| R41 | 9 | 15 | R89 | 0 | 6 |
| R42 | 1 | 7 | R90 | -1 | 5 |
| R43 | 4 | 10 | R91 | 0 | 6 |
| R44 | -1 | 5 | R92 | 1 | 7 |
| R45 | 1 | 7 | R93 | -1 | 5 |
| R46 | 5 | 11 | R94 | 1 | 7 |
| R47 | 4 | 10 | R95 | 0 | 6 |
| R48 | 0 | 6 | R96 | -1 | 5 |

TABLE. IV. TIME ESTIMATION FOR EACH REQUIREMENT

| Requirement | Efforts/hours | Requirement | Efforts/hours | Requirement | Efforts/hours | Requirement | Efforts/hours |
|---|---|---|---|---|---|---|---|
| R1 | 20 | R25 | 20 | R49 | 30 | R73 | 30 |
| R2 | 20 | R26 | 20 | R50 | 30 | R74 | 30 |
| R3 | 20 | R27 | 20 | R51 | 30 | R75 | 30 |
| R4 | 20 | R28 | 20 | R52 | 30 | R76 | 30 |
| R5 | 20 | R29 | 20 | R53 | 30 | R77 | 20 |
| R6 | 20 | R30 | 20 | R54 | 30 | R78 | 20 |
| R7 | 20 | R31 | 20 | R55 | 30 | R79 | 30 |
| R8 | 20 | R32 | 30 | R56 | 20 | R80 | 30 |
| R9 | 20 | R33 | 20 | R57 | 20 | R81 | 30 |
| R10 | 20 | R34 | 20 | R58 | 20 | R82 | 20 |
| R11 | 20 | R35 | 60 | R59 | 30 | R83 | 30 |
| R12 | 20 | R36 | 60 | R60 | 60 | R84 | 30 |
| R13 | 20 | R37 | 20 | R61 | 60 | R85 | 20 |
| R14 | 20 | R38 | 30 | R62 | 30 | R86 | 30 |
| R15 | 20 | R39 | 30 | R63 | 30 | R87 | 20 |
| R16 | 20 | R40 | 30 | R64 | 30 | R88 | 20 |
| R17 | 30 | R41 | 20 | R65 | 30 | R89 | 20 |
| R18 | 30 | R42 | 60 | R66 | 30 | R90 | 30 |
| R19 | 30 | R43 | 20 | R67 | 30 | R91 | 30 |
| R20 | 20 | R44 | 30 | R68 | 30 | R92 | 30 |
| R21 | 60 | R45 | 30 | R69 | 30 | R93 | 20 |
| R22 | 30 | R46 | 20 | R70 | 30 | R94 | 30 |
| R23 | 30 | R47 | 20 | R71 | 30 | R95 | 20 |
| R24 | 20 | R48 | 20 | R72 | 30 | R96 | 20 |

TABLE. V. REQUIREMENTS DISTRIBUTION IN FOUR DEVELOPERS (CASE 01)

| A | B | C | D |
|---|---|---|---|
| Requirement | Requirement | Requirement | Requirement |
| R79 | R1 | R63 | R51 |
| R31 | R21 | R64 | R71 |
| R27 | R19 | R73 | R90 |
| R26 | R25 | R66 | R70 |
| R28 | R11 | R45 | R80 |
| R58 | R10 | R40 | R55 |
| R29 | R4 | R75 | R74 |
| R5 | R2 | R39 | R91 |
| R67 | R23 | R38 | R48 |
| R81 | R22 | R43 | R82 |
| R30 | R20 | R44 | R83 |
| R77 | R18 | R41 | R88 |
| R78 | R13 | R65 | R89 |
| R56 | R12 | R72 | R86 |
| R6 | R17 | R60 | R87 |
| R24 | R15 | R68 | R84 |
| R57 | R14 | R52 | R85 |
| R76 | R3 | R42 | R95 |
| R37 | R7 | R59 | R94 |
| R8 | R9 | R35 | R96 |
| R33 | R16 | R61 | R92 |
| R34 | | R32 | R93 |
| | | R69 | R46 |
| | | R36 | R47 |
| | | R62 | R49 |
| | | | R50 |
| | | | R53 |
| | | | R54 |

TABLE. VI. TIME ESTIMATION OF EACH DEVELOPERS RESPECTIVELY

| **A** | | **B** | | **C** | | **D** | |
|---|---|---|---|---|---|---|---|
| Case 01 | Case 02 | Case 01 | Case 02 | Case 01 | Case 02 | Case 01 | Case 02 |
| 470 hours | 470 hours | 520 hours | 520 hours | 1320 hours | 940 hours | 1750 hours | 730 hours |

## VI. CONCLUSION

In this research work, functional requirements of software are prioritized from developer's perspective using spanning trees. Functional requirements are inter-related with directed graph and were converted to spanning trees. Based on prioritization technique using spanning trees, requirements of ODOO ERP are prioritized accordingly. Prioritized requirements reduce inter-dependency issues and delays and thus assure timely delivery of projects. An experiment was conducted with four developers and requirements were distributed such that there exist dependency in requirements of different developers. Time estimation of each requirement was calculated using use case point estimation technique. Total estimation time of each developer was calculated for both prioritized and un-prioritized requirements. There found a significant difference in total estimation time in both cases which shows the importance of prioritization and its effect on overall estimation time. In future work, spanning concept will be used to distribute functional requirements in more efficient way on parallel distributing team members.

## REFERENCES

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

[2] M. Yaseen, R. Naseem, Z. Ali, and G. Ullah, 'IDENTIFICATION OF CHALLENGES DURING REQUIREMENTS IMPLEMENTATION IN GLOBAL SOFTWARE DEVELOPMENT : A SYSTEMATIC', vol. 4, no. 1, pp. 23–40, 2019.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[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, 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.

[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] N. Garg, 'Recent Advancements in Requirement Elicitation and Prioritization Techniques', pp. 237–240, 2015.

[12] N. Setiani and T. Dirgahayu, 'Clustering Technique for Information Requirement Prioritization in Specific CMSs', 2016.

[13] J. Erazo, H. Arboleda, and F. J. Pino, 'Analysis of the Software Implementation Process for ERP Systems', vol. 1, pp. 297–312, 2017.

[14] S. Parthasarathy and M. Daneva, 'The Journal of Systems and Software An approach to estimation of degree of customization for ERP projects using prioritized requirements', vol. 117, pp. 471–487, 2016.

[15] 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.

[16] M. A. A. Elsood and H. A. Hefny, 'A Goal-Based Technique for Requirements Prioritization', 2014.

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

[18] R. Prioritization and U. Hierarchical, 'Requirements Prioritization Using Hierarchical Dependencies', pp. 459–464, 2018.

[19] 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.

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

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

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

[24] T. Bebensee, I. Van De Weerd, and S. Brinkkemper, 'Binary Priority List for Prioritizing Software Requirements', pp. 67–78, 2010.

[25] X. J. Frp, F. Edujdlqlqj, W. Uhtxluhphqwv, and I. R. U. Vrphrqh, ') XQFWLRQDO 5HTXLUHPHQWV', vol. 6, pp. 793–797, 2017.

[26] S. Dhingra and M. Madan, 'Selection of Prioritization Technique for Software Requirement using Fuzzy Logic and Decision Tree', 2016.

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

[28] M. Rainey, 'A Work-Efficient Algorithm for Parallel Unordered Depth-First Search', 2015.

[29] M. Weigel, 'Connectivity algorithm with depth first search ( DFS ) on simple graphs Connectivity algorithm with depth first search ( DFS ) on simple graphs', pp. 4–8, 2018.