A HYBRID TECHNIQUE USING MINIMAL SPANNING TREE AND ANALYTIC HIERARCHICAL PROCESS TO IMPROVE FUNCTIONAL REQUIREMENTS PRIORITIZATION

MUHAMMAD YASEEN

UNIVERSITI TUN HUSSEIN ONN MALAYSIA

UNIVERSITI TUN HUSSEIN ONN MALAYSIA

STATUS CONFIRMATION FOR THESIS
DOCTOR OF PHILOSOPHY

A HYBRID TECHNIQUE USING MINIMAL SPANNING TREE AND ANALYTIC HIERARCHICAL PROCESS TO IMPROVE FUNCTIONAL REQUIREMENTS PRIORITIZATION

ACADEMIC SESSION: 2020/2021

I, MUHAMMAD YASEEN, agree to allow Thesis to be kept at the Library under the following terms:

1. This Thesis is the property of the Universiti Tun Hussein Onn Malaysia.
2. The library has the right to make copies for educational purposes only.
3. The library is allowed to make copies of this Thesis for educational exchange between higher educational institutions.
4. The library is allowed to make available full text access of the digital copy via the internet by Universiti Tun Hussein Onn Malaysia in downloadable format provided that the Thesis is not subject to an embargo. Should an embargo be in place, the digital copy will only be made available as set out above once the embargo has expired.
5. ** Please Mark (√)

☐ CONFIDENTIAL (Contains information of high security or of great importance to Malaysia as STIPULATED under the OFFICIAL SECRET ACT 1972) Title and Abstract only

☐ RESTRICTED (Contains restricted information as determined by the organization/institution where research was conducted)-Title, Abstract and Introduction only

☐ EMBARGO _____ (date) until _____ (date)

☑ FREE ACCESS

Approved by,

(WRITER'S SIGNATURE)
MUHAMMAD YASEEN

(SUPERVISOR'S SIGNATURE)
ASSOC. PROF. DR. AIDA MUSTAPHA

Permanent Address:

GULKADA No.1,
MINGORA SWAT,
PAKISTAN

Date : 16 / MAY / 2021

Date: 16 / MAY / 2021

NOTE: ** If this Thesis is classified as CONFIDENTIAL or RESTRICTED, please attach the letter from the relevant authority/organization stating reasons and duration for such classification

This thesis has been examined on 15 SEPTEMBER 2020 and is sufficient in fulfilling the scope and quality for the purpose of awarding the Degree of Doctor of Philosophy.

Chairperson:

ASSOC. PROF. TS. DR. HAIRULNIZAM BIN MAHDIN
Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia

Examiners:

PROF. DR. ABDUL AZIM ABD. GHANI
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia

PROF. DR. ROSZIATI IBRAHIM
Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia

A HYBRID TECHNIQUE USING MINIMAL SPANNING TREE AND ANALYTIC HIERARCHICAL PROCESS TO IMPROVE FUNCTIONAL REQUIREMENTS PRIORITIZATION

MUHAMMAD YASEEN

A thesis submitted in
fulfillment of the requirement for the award of the
Doctor of Philosophy in Information Technology

Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia

MAY 2021

I hereby declare that the work in this thesis is my own except for quotations and summaries which have been duly acknowledged

Student : ……

MUHAMMAD YASEEN

Date : 16 – MAY- 2021 ……

Supervisor : ……

ASSOC. PROF. DR. AIDA MUSTAPHA

Co Supervisor : ……

DR. NORAINI IBRAHIM

DEDICATION

To my mother and father.

ACKNOWLEDGEMENT

All praise and thanks are due to Almighty Allah. Firstly, I would like to express my sincere gratitude to my supervisor Assoc. Prof. Ts. Dr. Aida Mustapha for the continuous support of my Ph.D study and related research, for her patience, motivation, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis. Besides my supervisor, I would like to thank Dr. Noraini Ibrahim, for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

ABSTRACT

Software for large enterprises such as the Enterprise Resource Planning (ERP) is more likely to be developed by a team of software developers where the functional requirements (FRs) are distributed in parallel developers. Therefore, development of pre-requisite FRs must be carefully timed to see which requirement is to be implemented first by assigning priority to some FRs over others, so that FRs can be made available on time to parallel developers. Well-known prioritization technique such as the Analytic Hierarchical Process (AHP), although accurate, is not scalable for large set of FRs as in ERP due to high number of pairwise comparisons when the size of FRs is more than ten or twelve. To address this issue, this research proposes a hybrid prioritization technique of Minimal Spanning Trees (MST) and AHP called the Spanning Analytic Hierarchical Process (SAHP) for FRs prioritization by exploiting MST capability to prioritize large size software FRs with smaller pairwise comparisons but with more consistent results. Using Numerical Assignment (NA) technique, prioritized FRs from SAHP are assigned to priority groups such that top priority groups contain high priority FRs and low priority groups contain low priority FRs. Low priority group of FRs are dependent on high priority groups. As a result, within each priority group, inter-dependencies in FRs are reduced for parallel developers. Implementing high priority groups will reduce number of dependencies in FRs among the lower priority groups. The proposed technique is evaluated based on ERP case study and the results showed that SAHP reduces estimation time of parallel developers as compared to AHP and other techniques. This shows that SAHP is scalable to cater large number of pairwise comparisons for large systems like ERP.

ABSTRAK

Perisian bagi perusahaan besar seperti *Enterprise Resource Planning* (ERP) kebiasaannya dibangunkan oleh satu pasukan pembangun perisian di mana keperluan fungsian (FRs) diagihkan secara selari. Oleh yang demikian, pembangunan FR prasyarat mestilah diatur dengan cermat bagi menentukan keperluan yang perlu dibangunkan terlebih dahulu dengan cara memberi keutamaan kepada sesetengah FR berbanding yang lain dalam memastikan FR tersebut sedia digunakan oleh rakan pembangun sewaktu pembangunan selari. Teknik pengutamaan terkenal seperti *Analytic Hierarchical Process* (AHP), walaupun berkejituan tinggi, adalah tidak terskala untuk keperluan besar seperti ERP disebabkan perbandingan berpasang apabila saiz FR lebih daripada sepuluh atau dua belas. Bagi menangani isu ini, penyelidikan ini mencadangkan satu kaedah gabungan *Minimal Spanning Trees* (MST) dan AHP yang dipanggil called *Spanning Analytic Hierarchical Process* (SAHP) dengan mengeksploitasi keupayaan MST dalam pengutamaan FR bersaiz besar dengan kompleksiti masa yang minimal. Ketidakbolehpercayaan MST juga dibantu oleh pengutamaan AHP yang berkejituan tinggi. Dengan menggunakan teknik *Numerical Assignment* (NA), FR yang telah diberi keutamaan diagih kepada kumpulan berdasarkan keutamaan yang mana kumpulan berkeutamaan tinggi akan mendapat FR yang berkepentingan tinggi dan sebaliknya. Sebagai hasilnya, dalam setiap kumpulan, kadar kebergantungan keperluan fungsian dapat dikurangkan dalam pembangunan selari. Pembangunan kumpulan berkeutamaan tinggi juga mengurangkan kadar kebergantungan FR di kalangan kumpulan berkeutamaan rendah. Kaedah yang dicadangkan ini dinilai berdasarkan kajian kes ERP dan keputusan kajian menunjukkan kaedah SAHP berjaya mengurangkan anggaran masa dan kadar kelewatan dalam pembangunan selari berbanding dengan AHP dan teknik-teknik lain. Ini menunjukkan bahawa SAHP adalah terskala bagi menangani jumlah perbandingan berpasang yang banyak untuk sistem besar seperti ERP.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AHP	Analytic Hierarchical Process
BFS	Breadth-first Search
BPL	Binary Priority List
BRs	Business Requirements
BST	Binary Search Tree
CI	Consistency Index
CR	Consistency Ratio
CV	Cumulative voting
DAG	Directed Acyclic Graph
DFS	Depth-first Search
ERP	Enterprise Resource Planning
FRs	Functional Requirements
MST	Minimal Spanning Tree
NA	Numerical Assignment
NA	Numerical assignment
NFRs	Non-functional Requirements
PRs	Process Requirements
RCPSP	Resource Constraint Project Scheduling Problem
RI	Random Index
SAHP	Spanning tree based on AHP
TCF	Technical Complexity Factor
UAW	Unadjusted Actor Weight
UCP	Use Case Points
URs	User Requirements
UUCW	Unadjusted use case weight
V	Eigenvector

LIST OF APPENDICES

LIST OF PUBLICATIONS

1. **Muhammad Yaseen**, Aida Mustapha and Noraini Ibrahim. 2020. Prioritization of Software Functional Requirements from Developers Perspective. *International Journal of Advanced Computer Science and Applications (IJACSA)*, Vol. 11, No. 9, pp. 210-224.
2. **Muhammad Yaseen**, Aida Mustapha and Noraini Ibrahim. 2020. Prioritization of Software Functional Requirements: A Novel Approach using AHP and Spanning Tree. *International Journal of Advance Trends in Computer Science and Engineering*, Vol. 9, No. 1, pp. 51-56.
3. **Muhammad Yaseen**, Aida Mustapha, Noraini Ibrahim and Umar Farooq. 2020. Effective Requirement Elicitation Process using Developed Open Source Software Systems. *International Journal of Advance Trends in Computer Science and Engineering*, Vol. 9, No. 1.1, pp. 542-549.
4. **Muhammad Yaseen**, Aida Mustapha and Noraini Ibrahim. 2019. Minimizing Inter-Dependency Issues of Requirements in Parallel Developing Software Projects with AHP, *Compusoft*, Vol. 8, No. 7, pp. 3317-3323.
5. **Muhammad Yaseen**, Aida Mustapha and Noraini Ibrahim. 2019. Prioritization of Software Functional Requirements: Spanning Tree-based Approach. *International Journal of Advanced Computer Science and Applications (IJACSA)*, Vol. 10, No. 7, pp. 489-497.
6. **Muhammad Yaseen**, Aida Mustapha and Noraini Ibrahim. 2019. Requirements Prioritization and using Iteration Model for Successful Implementation of Requirements. *International Journal of Advanced Computer Science and Applications (IJACSA)*, Vol. 10, No. 1, pp. 121-127.
7. **Muhammad Yaseen**, Aida Mustapha and Noraini Ibrahim. 2018. An Approach for Managing Large-Sized Software Requirements During Prioritization, in *Proceedings of the 2018 IEEE Conference on Open* Systems (ICOS), pp. 98-103.

CHAPTER 1

INTRODUCTION

1.1 Overview

Requirements are the pre-requisites for any software system [1]. The quality of software system depends upon how well requirements are collected and implemented during the Requirement Engineering (RE) phase of a software development project [2][3]. Requirement elicitation in RE is considered to be the most critical and important step due to the reason that most of software project failures are traced back to improper requirements collection [4][5]. If the reasons of failure of software are traced back to improper requirements collection, then it becomes very costly to fix changes in requirements at a later stage [6][7].

Software requirements can be divided into different types i.e. Business Requirements (BRs), User Requirements (URs), Functional Requirements (FRs) and Non-Functional Requirements (NFRs). BRs deal with benefits and objectives of the software to the clients or business [8][9]. URs are high level requirements or features of the intended software [10]. FRs are the core requirements of software system [11][12]. Developers of the software system deal with FRs [13] while NFRs define quality attributes of the software such as usability and security [13][14].

When planning for a software development project involve large size requirements such as an Enterprise Resource Planning (ERP) system, URs are prioritized from the perspective of BRs based on cost, time or value of requirements for making release plan [8]. Release plan define which requirements to be included in particular release and which should not. As FRs belong to URs are inter-related,

identifying inter-dependencies and prioritization of FRs also become necessary from the perspective of BRs [15].

After elicitation and release plan development, FRs are scheduled for implementation. Accurate cost and time estimation of URs can only be assured if dependencies in FRs are considered [16]. The quality of software project depends on how well project time and cost are estimated and gets delivered within the estimated timeframe. As FRs are inter-related, thus it is necessary to consider dependencies in FRs when scheduling the resources such as developers and effort needed to implement all FRs [17].

1.2 Research Motivation

Cost and time estimation are considered to be success factor as well as reason of failure of big projects [18]. According to [19], one of the factors that the release planner must consider is the inter-dependencies in FRs. It is imperative to capture all dependencies and keep FRs in hierarchical order so that better release plan can be prepared. According to survey [20], 90% of big software projects exceed their deadlines either by ignoring dependencies during the beginning of the projects or FRs are not well prioritized during parallel development, thus increasing waiting time of requirements when other depended pre-requisite FRs are over estimated and delayed [21].

There could be several reasons in software project delays such as improper scheduling and improper effort estimation of FRs but one of the main reason of challenged or failed projects is that dependencies in FRs are ignored during scheduling [21]. When working in a parallel team, pre-requisite FRs should be given high priority. Due to its significance, prioritization of FRs become necessary. FRs inter-dependency is a crucial aspect during prioritization, since prioritization without managing these dependencies in FRs will affect the quality of the results of prioritization. For timely development of project, it is necessary to reduce delays while implementing FRs and that can be possible if pre-requisite FRs are available in time for parallel developers. Delay rate in projects increases when the number of FRs and inter-dependencies in FRs increases. So, prioritization is required in order to reduce the effect of dependencies as much low as possible. Giving priority to timely important FRs in big software projects with large number of FRs are essential and in this context no efficient work has been conducted [22]. According to [17], the probability of delays in

software projects increases when the number of dependencies among FRs increases, therefore considering constraints dependencies in FRs before scheduling is very important.

Literature survey in investigating inter-dependencies in FRs during prioritization reported only in three out of 65 studies that dealt with inter-dependencies in FRs from different perspectives [23]. These techniques include interactive requirements prioritization, value-based requirements prioritization and multiple aspects-based requirements prioritization. In interactive requirements prioritization, directed graph-based approach is used to inter-relate FRs and all FRs are assigned as high, medium or low priority by experts [15]. Although FRs are inter-related with this approach but it does not assign net priority value (importance) to FRs and depends on experts who assign priority to FRs which makes it not suitable for large size FRs.

In value-based requirements prioritization, two depended FRs will be closed in bracket with comma as separator e.g. (R2, R1) which shows R2 is needed for R1 [24]. This approach shows importance of pre-requisite FRs from perspective of BRs prioritization. it is used for FRs selection during making a release plan from BRs perspective. Finally, in multiple aspects-based requirements prioritization, URs are collected by considering dependencies in URs from business perspective such as sales, marketing, simplicity or innovativeness rather than FRs [25]. Interactive requirements prioritization and value-based requirements prioritization approaches are manual-based approaches, which does not assign net priority values to FRs.

Another problem with these techniques is scalability, whereby the techniques are not scalable for large-sized FRs due to the manual prioritization of the FRs. These techniques also do not provide any solution from parallel development perspective of FRs prioritization. According to Wiegers [14], new prioritization process is required for FRs which will help the project manager to resolve conflicts in FRs before making a release plan and scheduling. According to [26], 20% requirements are responsible for 80% dependencies and thus it is necessary to identify high priority FRs before implementation. Therefore, top priority FRs need to be captured and assigned into high priority groups. It will not only reduce number of dependencies for low priority group requirements but when pre-requisite FRs are delayed probability of delay rate will be not too much affected.

1.3 Problem Statement

Current work in FRs prioritization lack the abilities to solve dependency issues in parallel software development and to identify timely important FRs that may decrease waiting time of parallel developers for their pre-requisites. There is a need to prioritize FRs before scheduling and implementation. In value-based approach, inter-related FRs are only represented before implementation without assigning net importance. According to [23], interactive requirements prioritization using Directed Acyclic Graph (DAG) can provide such a good solution of handling inter-dependencies in FRs during prioritization as compared to other techniques. Participation of many professional analysts are required to conduct the process.

Prioritization technique such as the Analytic Hierarchical Process (AHP) calculate relative priority of one requirement with respect to other requirements in pairwise comparison in effort to reduce the effect of inter-dependencies in FRs for parallel developing software. Although AHP assure accuracy as compared to other techniques, it requires a high number of comparisons hence is not considered to be scalable for large-sized requirements [27][29]. High number of comparisons also affect the probability of inconsistency in judgments, along with the quality of prioritized FRs [30]. According to [31], excess of the pair-wise correlations permits a consistency pair-wise examinations in AHP deliver much more repetition. Since AHP uses a hierarchical structure, it is more suitable for problems that can be structured hierarchically, but clearly inadequate to deal large size FRs with a lot of inter-dependencies among FRs.

Minimal Spanning Tree (MST) prioritization technique is capable to prioritize large size software requirements by reducing pairwise comparisons [32]. This is achieved by presenting the FRs in the form of Directed Acyclic Graph (DAG), which can then easily be converted to possible number of spanning trees. However, MST is scalable for large size requirements by keeping requirements in order. It does not assign net importance or priority value to requirements with respect to all other requirements and thus not considered to be accurate prioritization technique [31][27].

To address these issues with AHP and MST, this research proposed a hybrid prioritization technique of MST and AHP for FRs prioritization. This is carried out by exploiting the capabilities of AHP and MST to prioritize large size software

requirements with small number of pairwise comparisons in FRs but with higher accuracy and consistency in results.

1.4 Research Objectives

The main aim of this thesis is to propose a hybrid approach of Minimum Spanning Tree (MST) and Analytic Hierarchical Process (AHP) specific for FRs prioritization. In order to achieve the aim, the following objectives are to be fulfilled:

1. To design a hybrid technique of MST and AHP called SAHP for FRs prioritization.
2. To assign prioritized FRs from SAHP to priority groups using Numerical Assignment (NA) for parallel software development.
3. To evaluate and compare the proposed FRs prioritization technique based on consistency index, scalability, accuracy, and time estimation and delay rate of parallel software development via the ODOO ERP as case study.

1.5 Scope and Limitations of Research

This research is scoped to FRs in large enterprise using the ODOO ERP as the case study. Although there be many factors that cause delays in software projects such as limited availability of software project resources, improper effort estimation of requirements and incompetency of software developers to implement their requirements in estimated time. This research work is also scoped to delays in software projects due to waiting time of FRs for their pre-requisite requirements in a parallel team development setup.

Proposed prioritization technique SAHP, AHP, MST, BPL and BST are applied on FRs of ODOO ERP. All steps of SAHP using ODOO ERP are applied manually accordingly and are given in section 4.3. Using RCPSP model for FRs scheduling, time estimation of parallel developers is calculated manually without experts' participation.

1.6 Organization of Thesis

This chapter gives the overview with motivation for functional requirements (FRs) prioritization during parallel software development project, problem statements and research objectives along with the scope and limitation of the research. The remaining chapters in this thesis are organized as follows:

- Chapter 2 discusses the background study of FRs conducted in detail.
- Chapter 3 presents the research methodology designed to achieve the thesis objectives.
- Chapter 4 presents design of SAHP and implementation of SAHP using ODOO ERP as case study.
- Chapter 5 presents detail discussion and analysis of results.
- Chapter 6 presents conclusion and contribution of current research work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter the background study of the different prioritization techniques is presented in detail. Some of the techniques that are suitable for one type of requirements are not applicable for other types. The purpose of this chapter is to overview and discuss the limitations of current prioritization techniques suitable for functional requirements (FRs) prioritization.

Overview of different types of requirements is given in Section 2.2. In Section 2.3, causes of delays in software projects due to FRs are discussed. FRs priority and its significance are given in Section 2.4. In Section 2.5, FRs prioritization approaches are presented. Prioritization techniques suitable for FRs are discussed in Section 2.6. AHP in combination with other techniques are discussed in Section 2.7. Comparative analysis of related work is presented in Section 2.8. In Section 2.9, mathematical laws and properties used in design of SAHP are discussed. Summary of the chapter is presented in Section 2.10.

2.2 Types of Requirements

There are four types of software requirements, which are Business Requirements (BRs), User Requirements (URs), Functional Requirements (FRs), and Non-Functional Requirements (NFRs) [9][10][14].

- User Requirements (URs): URs are the requirements that come from users for software system. URs are high level requirements or features of software [33][34]. Selection of URs for software system are based on business requirements (BRs) such as selection of URs can be based on cost of implementation of requirements or benefits of particular requirements to users or business [8].
- Functional Requirements (FRs): FRs are the core requirements of software system. FRs are the requirements which the system must do and must have in order to fulfill high level URs. Developers of the system deals with FRs [16][13]. Not all but some FRs are inter-related. There are few top priority FRs that are responsible for majority of dependencies for low priority FRs [19][26]. URs are high level requirements that come from clients while FRs are low level requirements that must be implemented for every user requirement. FRs of large size software's such as ODOO ERP are difficult to be developed by one developer and thus assigned to parallel development team [35].
- Non-functional Requirements (NFRs): NFRs are the requirements which may not be directly demanded by user or clients but they are necessary for system and helps to implement FRs successfully [36][37]. Quality of system depends on how well the system meets its NFRs. Examples are performance, usability, portability, inter-operability and security etc. FRs and NFRs together are also known as system requirements. So, for a system both FRs and NFRs are necessary [13].

2.3 Software Projects Failure due to Delay in FRs

Accurate cost and time estimation are considered to be success factor and reason of failure of big projects. Success of any software project is measured not only with how much software fulfill its requirements and quality. Also, successful project is the one

which is delivered in estimated time and which is cost effective [38][39]. The Standish Group study of 2009 reported that only 34% of software projects are successful, 44% are challenged and 22% are failed [4]. According to [40], the reasons of failure of software projects can be classified into three categories i.e. failure to meet the approved schedule, failure of achieving cost objectives and failure of defining clear project scope. The author [40] conducted survey on criteria of software projects success on projects managers and analysts. Although for analysts, the most critical success criteria were not meet user requirements but for project managers, time and cost were considered as most important factors of success for any software project.

According to [39], there are three factors that define the success of any software project i.e. time, cost and requirements criteria. Reasons that become the cause of projects delay should be evaluated at the beginning of the project. According to [41], effective management plays an important role in making plan for any project. This means requirements management is the key success factor and the way they are implemented define the quality of any system. According to [42], project success is probably the most commonly conversed topic in the field of project management. Countless measures have been presented to prompt the success of a project and most common of these success factors are meeting schedule, budget, and performance goals.

There can be many reasons of exceed in deadlines and cost such as improper effort estimation of FRs [43][44], developers experience and incompetency [45], technical complexities of projects not considered in start [46] and ignoring dependencies in FRs [17]. Ignoring dependencies in FRs during requirements collection can increase the estimated time and cost of software projects. Literature survey conducted by [20], shows that effort estimation and dependencies in FRs are two main causes of requirements that causes delay in software projects. According to [17], most of the big software projects exceed their deadlines because constraint dependencies in FRs are ignored during scheduling.

2.4 FRs Priority and its Significance in Parallel Development

Priority of FRs is giving importance to one requirement over another [47]. Not all but some FRs are inter-related such that implementation of one requirement need other FRs and thus, pre-requisite FRs should be given high priority[16]. In parallel

development projects where multiple teams or members work and implement requirements in parallel, FRs should be prioritized because two or more inter-related or depended FRs assigned to parallel developers can delay project. According to [35], it is the characteristic of large size software systems that it will be developed by parallel development teams. The structure of parallel development has profound effect on both the quality and timely delivery of a software product.

In parallel developing projects, many developers work in parallel to implement requirements. Requirement that is implemented by one developer can be required for other requirement of different team members. In this way, by delaying the needed requirement of other team member, requirements waiting time will increase, thus it can cause critical delay which at the end will exceed the project deadline. Understanding tasks and FRs dependencies in parallel development is must for developers in order to deliver software in estimated or less amount of time. For timely development of project, it is necessary to reduce delays while implementing FRs and that can be possible if pre-requisite FRs are available in time for the dependent requirements in parallel development. According to [14], the purpose of systematically dealing with requirements interdependencies is to improve decisions made during software development and to support early detection of potential problems due to requirements interdependencies. Managing requirements interdependencies is about identifying, storing, and maintaining information about how requirements relate to and affect each other. Delay rate in projects increase when number and dependencies in FRs increase and thus prioritization management of FRs is required in order to reduce the effect of dependencies as much low as possible.

According to [17], rate of late delivery of requirements and software projects are directly proportional to the total number of dependencies in requirements of large size requirements. When the number of dependencies in requirements increases, probability of timely delivery of projects increases. Thus, it is necessary to prioritize requirements, so that timely delivery of projects can be assured. According to [48], coordination between parallel developers is must while implementing software requirements as FRs of software are inter-related. More coordination is needed when this dependency increases, and less coordination is needed when effect of dependency is reduced. This indicates that for decreasing the effect of dependencies in parallel development, requirements of parallel development should be prioritized.

According to [49], identification and management of requirements dependencies fundamental challenge in software development organizations especially when large requirements are to be implemented by parallel development team. Reducing dependencies between two different developers is must. This dependency can either be between two requirements or either two team members can share same resource. In either case, timely development of project can be affected. According to [16], although management of FRs inter-dependencies are necessary but several issues should be addressed such as handling too much inter-dependencies in FRs, which are difficult and time consuming. So scalable approach should be required. According to [26] , there are few top priority FRs that are responsible for 75% inter-dependencies in low priority FRs. Thus, identification of these top priority FRs become necessary.

2.5 Prioritization Techniques Suggested for FRs

Some techniques are designed specifically for prioritizing particular type of requirements, while some techniques are designed for prioritizing all types of requirements. However, even there are a lot of techniques suggested to prioritize URs from different perspective of BRs and PRs such as Goal-based technique [50], Value-based technique [24][51], New Lanchester Theory [52], Machine learning based prioritization techniques such as Case-Based Ranking (CBR) [53], Clustering technique K-mean [54], DRANK [55] , Triage [56], Artificial Neural Network (ANN) [57] and Interactive Genetic Algorithms (IGA) [15].

According to survey results of more than fifty prioritization techniques [19], only three studies have considered dependencies in requirements during prioritization. Prioritization techniques that have considered dependencies in requirements during prioritization are Multi-Aspects based prioritization [25], Value-based Requirements Prioritization [24] and Interactive Requirements Prioritization [15].

2.5.1 Multiple Aspects-based Prioritization

In Multiple aspects-based Prioritization, dependencies in requirements are measured on the basis of business or features dependencies where the technique allow the

stakeholders to assign a value of the dependency to each requirement individually. For example, Dependencies in technical factors such as cost, time, associated risks, available resources, complex, and effort required. But, in this way, management of dependencies is weak, as the dependencies between the requirements have to be taken based on the dependency rate of the requirement on others instead of letting the stakeholders or experts directly give a value of the dependency to each requirement separately deprived of seeing the possible inter-dependencies of FRs together. This volatility of requirements selection and prioritization process is crucial and very difficult to handle. Important factors that manage the volatility include market trends, changing needs of the users.

2.5.2 Value-based Requirements Selection

Based on values of high-level user requirements, low level FRs are prioritized as these FRs are inter-related. Each dependent relation is represented by a pair of two requirements separated by comma.

This technique addresses the dependencies in FRs during requirements selection process. For small size requirements handling dependencies in Value-based Requirements Prioritization is not a problem but for large size requirements, it is difficult to fill dependencies due to too many pairs to check for dependencies. Dependencies are represented using comma (,) separated lists of requirements. E.g. If R1, R2 and R3 are FRs that are to be implemented such that R1 depends on R4, and R5, R2 depends on R6, R7, and R8. Similarly, R4 further depends on R9 and R10. Possible pair of dependencies will be,

Pair 1: (R1, R4).
Pair 2: (R1, R5).
Pair 3: (R2, R6).
Pair 4: (R2, R7).
Pair 5: (R2, R8).
Pair 6: (R4, R9).
Pair 7: (R4, R10).

2.5.3 Interactive Requirements Prioritization

In Interactive Requirements Prioritization, prioritization of FRs considered dependencies in FRs and for which graph-based approach is used for inter-relating and prioritizing FRs. In this technique, first of all requirements are represented with DAG and then from precedence order of implementation, requirements are numerically assigned into three priority groups by experts i.e. high, medium and low. Requirements with high priority should precede those with medium priority and medium should precede those with low priority. For example, in Figure 2.1, R1 is required for R4 and R5 while R4 and R5 are required for R2 and R3 respectively. Requirements are categorized into three groups i.e. High priority, medium priority and low priority. Requirements are assigned in such a way that R1 is assigned to high priority group, R4 and R5 are assigned to medium priority and R2 and R3 are assigned to low priority groups respectively. The requirements assigned to different priority groups are shown in Table 2.1.

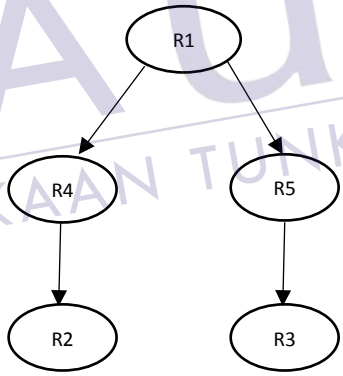

Figure 2.1: FRs related with directed graph

Table 2.1: Requirements assigned into Priority Groups from DAG

Requirement	Priority
R1	High
R4	Medium
R5	Medium
R2	Low
R3	Low

2.6 Prioritization Techniques Suitable for FRs Prioritization

In this section, prioritization techniques that will serve as the base techniques in this research are reviewed in detail, which are the Analytic Hierarchical Process (AHP), Numerical Assignments (NA), and Minimum Spanning Tree (MST). The presented techniques in this section are suitable to prioritize any type of requirements and thus these techniques are suitable for FRs prioritization.

2.6.1 Analytic Hierarchical Process (AHP)

AHP is an organized decision-making method that is intended to compute complex multi-criteria decision problems [11]. The method was formerly suggested by Thomas Saaty [58]. A judgment or comparison is numerical representation of relation among between two requirements. A matrix is drawn to compare any two requirements based on their importance. AHP consists of five phases as shown in Figure 2.2.

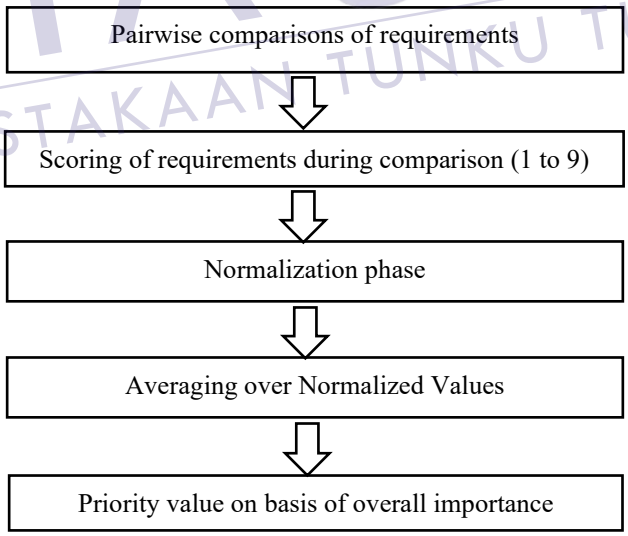

Figure 2.2: Phases of AHP [58][61]

(i) Pairwise comparison of requirements

Definition 1 [59]. *The pairwise comparison matrix for a decision maker with n requirements is an n * n matrix* $A = [a_{ij}]$ *is such that:*

$$\begin{aligned} a_{ij=} &> 0 && \text{For } i, j = 1, \ldots, n, \text{ and} \\ a_{ij=} &= 1/a_{ji} && \text{For } i, j = 1, \ldots, n \end{aligned} \tag{2.1}$$

If $a_{ij} > 0$, this shows matrix is a positive matrix with no negative element and zero number in comparison. In comparison matrix, $a_{ij=} = 1/a_{ji}$ means that for every comparison, there must be a reciprocal value.

(ii) Scoring of requirements during comparison (1 to 9)

When we compare two requirements pairwise, we put priority score value. Users (stakeholders) or experts put priority score between 1 and 9 to requirements on the basis of scale values as shown in Table 2. 2. Value 1 shows equal importance while 3 shows there exist slightly difference in importance while 9 shows there exist extreme variation in importance between any two requirements. Importance value can be any value between 1 and 9.

Table 2.2: Scale for Prioritizing Requirements

How Important	Definition
1	Equal Importance
2	Intermediate value between 1 and 3
3	moderate difference in importance
4	Intermediate value between 3 and 5
5	Essential difference in importance
6	Intermediate value between 5 and 7
7	Major difference in importance
8	Intermediate value between 7 and 9
9	Extreme difference in importance

During this phase, all requirements are pairwise compared together and priority score are assigned. We will put value 1 for those requirements that have equal importance and if priority of requirement is greater than other requirement, then we will put value greater than 1. Suppose we want to compare requirement R1 with R2 and R1 is of high importance as compared to R2 than any value between 2 and 9 is possible for R1 against R2. E.g. if priority of R1 against R2 is 2, this means R1 is double in priority as

compared to R2 and similarly R2 will be half in priority as compared to R1 thus we will put ½ for R2 against R1. E.g. Priority score assigned to R1, R2 and R3 are as shown in Table 2.3.

Table 2.3: Pairwise comparison

	R1	R2	R3
R1	1.00	2.0	4.0
R2	0.50	1.0	2.0
R3	0.25	0.5	1.0
Sum	1.75	3.5	7.0

Definition 2 [59]. *The weight vector* $\boldsymbol{w}$ *is of the form:*

$$w_i = \begin{bmatrix} w1 \\ w2 \\ wn \end{bmatrix}$$

where w1, w2 and w3 are total net weights of each requirement. On the basis of these weights, score is assigned to all requirements in comparison with other requirements and net priority is calculated. E.g. In Table 2.3, priority of R1 against R2 is 2, which means if weight score of R1 is 8, then weight of R2 must be 4.

(iii) Normalization phase

In this phase normalized values for each requirement against other requirement is calculated by dividing priority value of requirement with rows sum for particular column to which other requirement belongs. E.g. For requirements of Table 2.3, normalized values are shown in Table 2.4.

Table 2.4: Normalized values

	R1	R2	R3
R1	0.571	0.571	0.571
R2	0.285	0.285	0.285
R3	0.142	0.142	0.142

(iv) Averaging over Normalized Values

In this phase, normalized values for each requirement against all requirements of each column is added to give net priority in all requirements. The column sum shows priority values for each requirement. The sum of priority values of all requirements is equal to number of requirements. As number of requirements are 4, so sum will be equal to 4. R1 gets high priority while R3 gets lowest priority. Sum of normalized values for Table 2.4 is shown in Table 2.5.

Table 2.5: Sum of Normalized values

	R1	R2	R3	Sum
R1	0.571	0.571	0.571	1.713
R2	0.285	0.285	0.285	0.855
R3	0.142	0.142	0.142	0.427
Sum				4

Definition 3 [60]. *Eigenvector is a vector that contains all prioritized requirements such that:*

$$v = [\,R1, \quad R2, \quad Rn] \tag{2.2}$$

where $R_i > 0$ and $\sum_{i=1}^{n} R_i = 1$.

Eigenvector is a matrix that contains final prioritized requirements after comparisons. E.g. In Table 2.5, column sum represents eigenvector with all prioritized requirements. The above condition is true only when decision maker is consistent in comparisons. Instead of taking weight matrices, we can take comparison matrix and eigenvector.

Definition 4 [60]. *Eigenvalue* λ *is a scalar associated with eigenvector such that:*

$$\mathrm{Av} = \lambda \mathrm{v} \tag{2.3}$$

where λ is total number of requirements i.e. *n* but only for consistent matrix. A is comparison matrix and v show eigenvector.

Definition 5 [60]**.** *The principal eigenvalue denoted* λ_{max}*, of a comparison matrix A is the largest eigenvalue of that matrix.* λ_{max} *can be calculated as given below.*

$$\lambda_{max} = \frac{(\lambda 1 + \lambda 2 + \lambda 3)}{3} \tag{2.4}$$

where ƛ1, ƛ2, and ƛ1 are associated eigenvalues with each requirement of eigenvector.

Theorem 1. *For a reciprocal n by n matrix with all entries greater than zero, the principal eigenvalue,* λ_{max}*ill be greater than or equal to n. That is,* $n \leq \lambda_{max}$ [59].

Theorem 2. *A is consistent if and only if* $n = \lambda_{max}$ [58]

Based on Definition 6, Theorem 1 and Theorem 2, we can calculate λ_{max}. λ_{max} equal to *n* shows matrix is consistent while for inconsistent matrixes, its value is greater than *n*. Increase in value of λ_{max} shows more and more inconsistency in judgments.

Consistency Index (CI) and Consistency Ratio (CR) are calculated to measure the inconsistency in judgments using the following Equation 2.5 and Equation 2.6 [61].

$$\text{CI} = (\lambda_{max} - n) / n - 1 \tag{2.5}$$

$$CR = \frac{CI}{IR} \tag{2.6}$$

where IR is random index scale defined for a particular matrix size as shown in Table A8 of Appendix. If CR value become 0, this shows CI is zero, $n = \lambda_{max}$ and matrix is fully consistent with no judgmental errors. When value of CR is greater than 0, this shows matrix has inconsistencies in requirements priority but if value of CR is less than 0.1, then this inconsistency is acceptable but when CR > 01, this shows quite variation in judgments.

Table 2.5 shows that each pair of requirements only needs to be compared once with total of $n(n\text{-}1)/2$ comparisons, where *n* is the number of requirements. For example, if the number of requirements is ten then AHP will perform forty-five times comparisons of the requirements. If the requirements increase in size, so does the processing time. If the requirements are thousand, there will be 1000*(1000-1)/2 = 499,500 comparisons, which is both very time consuming and difficult to execute.

Since prioritizations are seldom consistent when making comparative judgments, AHP assumes inconsistency in judgments and allows for the calculation of a consistency ratio to re-evaluate inconsistent judgments. This consistency check is possible due to the redundancy of the pairwise comparisons. However, this redundancy also results in a dramatic increase in the number of comparisons as the number of requirements increases. Though, by reducing the number of redundant comparisons, the ability to classify inconsistent judgments can also be reduced accordingly. In many studies, authors applied AHP to prioritize small size requirements. Discussion in the light of literature review in AHP is given below.

[28] prioritized software system requirements i.e. FRs along with NFRs using AHP. NFRs are normally ignored when prioritizing FRs, but NFRs are as much necessary as its FRs. Every FRs have some NFRs associated. From this study, author concluded that although NFRs are ignored when comparing FRs, prioritizing FRs in the context of NFRs is as much necessary as FRs. In another research study, AHP is applied for prioritizing NFRs [62].

In [29], author efficiently applied AHP for prioritizing requirements during elicitation phase. In another paper, the author prioritized FRs in relation with NFRs using AHP method of pairwise comparisons [28]. Every functional requirement will be compared against its associated nonfunctional requirement and thus all FRs will be prioritized accordingly based on NFRs. This shows that paying attention to nonfunctional requirements can change priority for functional requirements. In another research study [63], improved approach of goal oriented for requirement prioritization during elicitation is discussed. Bagheri [64] used concepts of AHP in his study and identified best possible features for software product in product line. According to [30], accuracy and consistency of final prioritized requirements can be affected due to large number of comparisons.

AHP is thus applied on requirements but with size no more than few. For large size requirements software's, AHP is not applied due to scalability issues. Furthermore, AHP is not applied to prioritize FRs in parallel development.

2.6.2 Numerical Assignment (NA)

Using NA, requirements are categorized into different groups with high, medium, and low priority. Number of priority groups can vary. The priority of requirements inside each group is considered the same but priority of high priority group is greater than medium which is greater than low priority group. In this technique, number of groups can be increased but there are no criteria defined on the basis of which requirements can be assigned to group. But the technique helps in easy management of requirements i.e. which set of requirements to be implemented first and which should not.

This technique can be used in combination with other techniques in order to better prioritize requirements e.g. requirements at initial stage can be categorized into different priority groups and inside groups, therefore any of the available technique from literature can be applied to further prioritize requirements of each group. So in this way, within a group, the requirements can be ordered [65]. For example, given 100 requirements ([R1, R2, R3, …, R100]), these requirements are assigned into four priority groups by applying NA technique in such a way as shown as follows.

Group 1 = [R1, R2 … R25];

Group 2 = [R26, R27 … R50];

Group 3 = [R51, R52 … R75];

Group 4 = [R76, R77 … R100];

Priority order or value is assigned to each group e.g. Group 1 > Group 2 > Group 3 > Group 4. Inside each group, priority of all requirements will be considered as same. According to [31], sub-groups are also possible and this way of creating sub-groups is known as priority groups e.g. Group 1 = (Priority group 1 = [R1, R2 … R10], Priority group 2 = [R11, R12 …, R25]) such that Priority group 1 > Priority group 2.

Massey [66] suggested a NA technique for prioritizing legal requirements. This technique can be used by novel or current systems that must fulfill the laws or rules to rank the requirements concerning to those laws and regulations. The author in his study concentrated on EHR (electronic health records) systems, which must obey with HIPAA (health insurance portability accountability Act). Specifically, he applied technique on the 63 requirements defined for iTrust, an open source EHR system.

Outcome of research shows that the iTrust requirements had 17 requirements with no mapping to appropriate component of the legal text. 19 requirements demanded additional modification and 27 requirements are legally implementation-ready. In addition, this work proves that numeral assignment method of prioritizing requirements might be beneficial in the legitimate domain.

2.6.3 Minimal Spanning Tree (MST)

This technique was introduced by [32], to prioritize large size requirements. In MST, all requirements are arranged in MST in order such that priority of one requirement is higher than priority of other requirement. As compared to AHP, MST does not compare all requirements, but only necessary requirements are compared to make order of all requirements together. Redundancies in MST based approach can be easily reduced. Figure 2.3 shows steps of prioritization with MST.

Begin

Step 1: As preparation, outline *n*-1 unique pairs of requirements so that minimal spanning tree can be constructed.

Step 2: As execution, compare all outlined pairs of requirements using the scale from 1 to 9.

Step 3: As presentation, compute the priorities by taking the mean of the existing priorities of all possible ways in which they are connected.

End

Figure 2.3: Prioritization with MST [32]

In step 1, *n*-1 pairs of requirements will be outlined so that minimal spanning tree can be constructed. In step 2, *n*-1 pair of requirements will be compared using scale from 1 to 9 where 1 shows minimum importance while 9 shows extreme importance. E.g. if requirement A is of high priority as compared to B and B is of high priority as compared to C, then there is no need to compare requirement A with C because priority of A will already be higher than that of C. Total pairwise comparisons with MST depends on the number of edges in a tree. Total comparisons in MST is given in Equation 2.7 where *n* shows total requirements in MST.

$$\text{Total comparisons} = n - 1 \tag{2.7}$$

Thus, when the number of requirements in MST increases, total comparisons increases. Figure 2.4 shows R1, R2, R3 and R4 are arranged in order in MST such that priority of R1 is greater than R2 while R2 priority is greater than R3 and R4. This shows priority of R1 will be greater than R3 and R4. There is no need to compare R1 with R3 and R4. In step 3 of MST, we can compute missing priorities e.g. R1 with respect to R3 and R4.

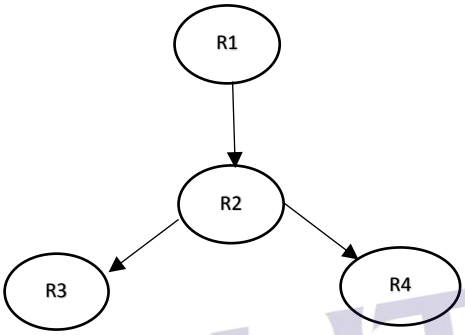

Figure 2.4: Requirements represented with MST

For example, if priority of R1 in comparison with R2 is 9 and priority of R2 in comparison with R3 is also 9 then geometric mean will also become 9. Thus, without comparing R1 and R3, priority of R1 in comparison with R3 will become 9. Next, if the developer has to decide between R3 and R4 the one with higher priority, than there are two ways. If R2 is more important than R3 as compared to R4, this means priority of R4 will be greater than R3 but if priority of R2 in comparison with R4 is greater than R3, then priority of R3 will be greater than R4. Priority of R3 and R4 will be equal if priority of R2 in comparison with both R3 and R4 are equal.

- **Directed Acyclic Graph (DAG)**

Graph based approaches can be used to represent and relates requirements with one another. Graph is comprised of vertices (V) and edges (E). Vertices shows the entity or node of a system while through edges different vertices are linked together. Based on the types of links and connectivity. There are two types of graphs; directed graph and undirected graph [67]. An undirected graph is the one in which a set of objects

(called vertices or nodes) that are connected together, where all the edges are bidirectional [68]. In directed graph [69][68], the edges that connect two vertices have a direction that points from one vertex to other vertex as shown in Figure 2.5.

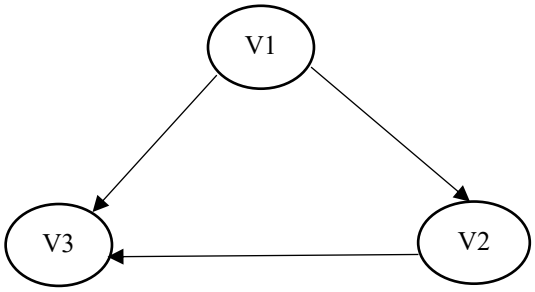

Figure 2.5: Directed Acyclic Graph

Directed graph can either be cyclic or acyclic. Directed acyclic graph (DAG) is one which doesn't possess any recurrent cycle. For FRs, we can use only DAG because repeated cycles in FRs inter-connectivity is not possible as requirement need can be only uni-directional.

In Graph Theory and Computer Science, an adjacency matrix is a square matrix used to represent a finite graph [70]. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph. For a simple graph with vertex set V, the adjacency matrix is a square $|V| \times |V|$ matrix A such that its element A_{ij} is one when there is an edge from vertex *i* to vertex *j*, and zero when there is no edge.

- **MST from Directed Acyclic Graph (DAG)**

MST are special sub-graphs 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* [67][69]. MST can be formed simply either by performing breadth-first search (BFS) or depth-first search (DFS). These graph search algorithms are only dependent on the number of vertices in the graph, so they are quite fast [71][72]. There are a few general properties of spanning trees as given as follows.

- A connected graph can have more than one spanning trees.

- All possible spanning trees for a graph *G* have the same number of edges and vertices.
- Spanning trees do not have any cycles.
- Spanning trees are all minimally connected. That is, if any one edge is removed, the spanning tree will no longer be connected.
- Adding any edge to the spanning tree will create a cycle. So, a spanning tree is maximally acyclic.

This study used the concept of DAG and MST to represent and interrelated software FRs. Through spanning trees, one can easily structure all the dependent requirements.

2.6.4 Cumulative Voting (CV)

Cumulative voting (CV) is a method where stakeholders are given 100 dollars and they have to distribute on all possible requirements just like voting mechanism [73][74]. This technique is also known as 100 dollars method. The requirement with high votes will be given more priority. Refer to Figure 2.3, R1 will get high number of votes because it is required for all other requirements. In this way, priority votes of R2 will be higher than R3 and R4. Votes of R3 and R4 can be equal in case of FRs as these requirements are not dependent and inter-related. E.g. R1 gets 40 votes, R2 gets 30 votes while R3 and R4 gets equal votes of 15 each. This number of votes can be any but R1 votes be higher than R2 and R2 votes should be higher than R3 and R4.

2.6.5 Binary Priority List (BPL)

Binary Priority List (BPL) is prioritization technique where Requirement with high priority is kept on top side of that requirement while requirement with low priority is kept bottom of the root requirement selected. According to author [75], if many requirements have similar priorities, then list of requirements can be considered either on left or left side. In this way, upcoming requirements are compared and kept on either down or top side. In this way, one can draw hierarchy of requirements in order of their priorities. Step by step approach of prioritization with BPL is given as follows.

REFERENCES

[1] Younas, M., Jawawi, D. N. A., Shah, M. A., Mustafa, A., Awais, M., Ishfaq, M. K., & Wakil, K., "Elicitation of Nonfunctional Requirements in Agile Development using Cloud Computing Environment," IEEE Access, 8, 209153-209162, 2020.

[2] H. S. Dar, "Reducing Ambiguity in Requirements Elicitation via Gamification," IEEE, 28 Requirement Engineering Conference, pp. 440–444, 2020.

[3] S. M. A. Jaya, "An integrated approach towards automated software requirements elicitation from unstructured documents," *J. Ambient Intell. Humaniz. Comput.*, no. 2015.

[4] A. Hussain and E. Mkpojiogu, "Requirements : Towards an understanding on why software projects fail," International Conference on Applied Science and Technology (ICAST'16), no. August, pp. 1–7, 2016..

[5] A. Poth and A. Riel, "Quality Requirements Elicitation by Ideation of Product Quality Risks with Design Thinking." *28th IEEE Requirement Engineering Conference*, 2020.

[6] M. Suhaib, "Investigation and Analysis of the Requirement Engineering in Software Development Process and its Systematic Requirements Elicitation Approach," I*nternational Journal of Scientific & Technology Research*, vol. 9, no. 04, pp. 2723–2726, 2020.

[7] A. M. R. M. Jadi, A. Abdulrahman, and S. N. Bhatti, "systematic approach in change management for elicitation and prioritization of requirements in agile," *Journal of Software Engineering & Intelligent Systems*, vol. 5, no. 1, pp. 18–24, 2020.

[8] J. Karlsson, "A Cost–Value Approach for Prioritizing Requirements," *IEEE Software*, vol.14, no.5, pp.67-74, 1997.

[9] X. Frank, Y. Sun, and C. Sekhar, "Priority assessment of software process requirements from multiple perspectives," *Journal of Systems and Software,*

vol. 79, pp. 1649–1660, 2006..

[10] S. C. Allala, J. P. Sotomayor, D. Santiago, T. M. King "Towards Transforming User Requirements to Test Cases Using MDE and NLP," *2019 IEEE 43rd Annu. Comput. Softw. Appl. Conf.*, vol. 2, pp. 350–355, 2019.

[11] S. Mahmudova and Z. Jabrailova, "Development of an algorithm using the AHP method for selecting software according to its functionality," *Soft Comput.*, vol. 24, no. 11, pp. 8495–8502, 2020.

[12] N. Rahimi, F. Eassa, and L. Elrefaei, "An ensemble machine learning technique for functional requirement classification," *Symmetry (Basel).*, vol. 12, no. 10, pp. 1–26, 2020.

[13] I. Reinhartz and B. Mark, "Extracting core requirements for software product lines," *Springer journal of Requirement Engineering*, pp. 47-65, 2019.

[14] K. Ayub, F. Azam, M. W. Anwar, A. Amjad, and M. S. Jahan, "A Novel Approach for Software Requirement Prioritization Based Upon Non Functional Requirements," *Proc. - 2019 7th Int. Conf. Softw. Eng. Res. Innov. CONISOFT 2019*, pp. 8–15, 2019.

[15] 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.

[16] Å. G. Dahlstedt and A. Persson, "5 Requirements Interdependencies : State of the Art and Future Challenges." S*pringer, Engineering and Managing Software Requirements,* pp 95-116, 2005.

[17] C. Li, M. van den Akker, S. Brinkkemper, and G. Diepen, "An integrated approach for requirement selection and scheduling in software release planning," *Springer journal of Requirement Engineering*, vol. 15, no. 4, pp. 375–396.

[18] 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.

[19] Pa¨r Carlshamre, "Release planning in market-driven software product development: Provoking an understanding," *Springer journal of Requirement Engineering*., vol. 6, pp. 139–151, 2002.

[20] W. Shah, "Time Estimation as Critical Factor of Software Failure and Success : A Systematic Literature Review Protocol with Preliminary Results," International Journal of Computer Science Engineering, vol. 8, no. 02, pp. 36–

40, 2019.

[21] P. Savolainen and J. J. Ahonen, “Software development project success and failure from the supplier ’ s perspective : A systematic literature review,” International Journal of Project Management, vol. 30, no. 4, pp. 458–469, 2012.

[22] M. Dabbagh and S. P. Lee, “A consistent approach for prioritizing system quality attributes,” SNPD 2013 - 14th ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput., pp. 317–322.

[23] F. Hujainah, R. Binti, A. B. U. Bakar, and M. A. Abdulgabber, “Investigation of Requirements Interdependencies in Existing Techniques of Requirements Prioritization,” Teh. Vjesn. - Tech. Gaz., vol. 26, no. 4, pp. 1186–1190, 2019.

[24] N. Kukreja, S. Swaroop, B. Boehm, and S. Padmanabhuni, “Value-Based Requirements Prioritization : Usage Experiences,” Procedia Comput. Sci., vol. 16, pp. 806–813, 2013.

[25] F. Sher, D. N. A. Jawawi, R. Mohamad, and M. I. Babar, “Multi-aspects based requirements priortization technique for value-based software developments,” Proc. - 2014 Int. Conf. Emerg. Technol. ICET 2014, pp. 1–6, 2014.

[26] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. Natt och Dag, “An industrial survey of requirements interdependencies in software product release planning,” Proc. IEEE Int. Conf. Requir. Eng., pp. 84–91, 2001.

[27] J. Ali Khan, I. Ur Rehman, Y. Hayat Khan, I. Javed Khan, and S. Rashid, “Comparison of Requirement Prioritization Techniques to Find Best Prioritization Technique,” Int. J. Mod. Educ. Comput. Sci., vol. 7, no. 11, pp. 53–59, 2015.

[28] F.Fellir, K.Nafil, R.Touahni,"Systems Requirements Prioritization based on AHP" 2014 Third IEEE International Colloquium in Information Science and Technology (CIST), pp. 163–167, 2015.

[29] M. Sadiq, S. Ghafir, and M. Shahid, “An approach for eliciting software requirements and its prioritization using analytic hierarchy process,” ARTCom 2009 - Int. Conf. Adv. Recent Technol. Commun. Comput., pp. 790–795, 2009.

[30] N. Jan, I. Inayat, and M. Abbas, “An Empirical Evaluation of Requirements Prioritization Techniques,” Mark. Brand. Res., vol. 7, pp. 11–23, 2020.

[31] S. Khan, S. Ahmad, J. Ahmad, and M. Haider, “A Critical Survey on Requirement Prioritization Techniques,” ACEIT Conference Proceeding, pp. 270–273, 2016.

[32] J. Karlsson, C. Wohlin, and B. Regnell, “An evaluation of methods for prioritizing software requirements,” Inf. Softw. Technol., vol. 39, no. 14–15, pp. 939–947, 1998.

[33] C. Series, “Structural Equation Modeling Technique for Social Application of Information Technology User Requirements’ Identification,” Journal of Physics and Conference Series, 2019.

[34] I. O. P. C. Series and M. Science, “User requirements elicitation in web-based Participatory Geographic Information System interface design User requirements elicitation in web-based Participatory Geographic Information System interface design,” 2020.

[35] A. Observational, C. Study, and M. Hill, “Parallel Changes in Large Scale Software Development :” ACM Transactions on Software Engineering and Methodology, vol. 10, no. 3, 2001.

[36] M. Binkhonain and L. Zhao, “A review of machine learning algorithms for identification and classification of non-functional requirements,” Expert Syst. with Appl. X, vol. 1, 2019.

[37] K. B. Ijaz, I. Inayat, and F. Allah Bukhsh, “Non-functional Requirements Prioritization: A Systematic Literature Review,” Proc. - 45th Euromicro Conf. Softw. Eng. Adv. Appl. SEAA 2019, pp. 379–386, 2019.

[38] N. Agarwal and U. Rathod, “Defining ‘success’ for software projects: An exploratory revelation,” Int. J. Proj. Manag., vol. 24, no. 4, pp. 358–370, 2006.

[39] K. de Bakker, A. Boonstra, and H. Wortmann, “Does risk management contribute to IT project success? A meta-analysis of empirical evidence,” Int. J. Proj. Manag., vol. 28, no. 5, pp. 493–503, 2010.

[40] J. Wateridge, “How can IS/IT projects be measured for success?,” Int. J. Proj. Manag., vol. 16, no. 1, pp. 59–63, 1998.

[41] L. Crawford, “Profiling the Competent Project Manager,” Proj. Manag. Res. Turn Millenium Proc. PMI Res. Conf., no. January 2009, pp. 3–15, 2000.

[42] V. Botes, “All the world’s a stage,” Accounting, Auditing and Accountability Journal, vol. 25, no. 7. p. 1236, 2012.

[43] R. Conradi, “Effort Estimation of Use Cases for Incremental Large-Scale Software Development,” 27th International Conference on Software Engineering, no. 7491, pp. 303–311, 2005.

[44] A. Javed, M. A. Ullah, and A.- Rehman, “Factors Affecting Software Cost

Estimation in Developing Countries," Int. J. Inf. Technol. Comput. Sci., vol. 5, no. 5, pp. 54–59, 2013.

[45] M. Choetkiertikul, H. K. Dam, T. Tran, and A. Ghose, "Predicting delays in software projects using networked classification," Proc. - 2015 30th IEEE/ACM Int. Conf. Autom. Softw. Eng. ASE 2015, pp. 353–364, 2016.

[46] A. Gumaei, B. Almaslukh, and N. Tagoug, "An Empirical Study of Software Cost Estimation in Saudi Arabia Software Industry," no. 6, pp. 44–48, 2015.

[47] D. Port, A. Olkov, and T. Menzies, "Using simulation to investigate requirements prioritization strategies," ASE 2008 - 23rd IEEE/ACM Int. Conf. Autom. Softw. Eng. Proc., pp. 268–277, 2008.

[48] K. Blincoe, G. Valetto, and D. Damian, "Do All Task Dependencies Require Coordination ? The Role of Task Properties in Identifying Critical Coordination Needs in Software Projects," pp. 213–223.

[49] M. Cataldo, J. D. Herbsleb, and K. M. Carley, "Socio-Technical Congruence : A Framework for Assessing the Impact of Technical and Work Dependencies on Software Development Productivity Socio-Technical Congruence : A Framework for Assessing the Impact of Technical and Work Dependencies on Software Deve," ESEM, DPP.2-11, 2008.

[50] M. A. A. Elsood and H. A. Hefny, "A Goal-Based Technique for Requirements Prioritization," 9th International Conference on Informatics and Systems, 2014.

[51] S. S. Payyavula and F. Si, "Application of Value Based Requirement Prioritization in a Banking Product implementation," Third International Conference on Services in Emerging Markets, pp. 157–161, 2012.

[52] T. M. Fehlmann, "New Lanchester Theory for Requirements Prioritization," 2nd International Workshop on Software Product Management, pp. 1–6, 2008.

[53] 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.

[54] N. Setiani and T. Dirgahayu, "Clustering Technique for Information Requirement Prioritization in Specific CMSs," International Conference o Data and Software Engineering, 2016.

[55] F. Shao, R. Peng, H. Lai, and B. Wang, "A semi-automated requirements prioritization method based on preferences and dependencies," The Journal of Systems and Software, vol. 126, pp. 141–156, 2017.

[56] C. Duan and Æ. P. Laurent, “Towards automated requirements prioritization and triage,” Requirements Engineering, pp. 73–89, 2009..

[57] 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.” Knowledge-Based Syst., 2015.

[58] T. L. Saaty, “How to make a decision : The Analytic Hierarchy Process,” European Journal of Operational Research, vol. 48, 1990.

[59] T. L. Saaty, “A scaling method for priorities in hierarchical structures,” J. Math. Psychol., vol. 15, no. 3, pp. 234–281, 1977.

[60] S. Philadelphia and W. Pennsylvania, “A Scaling Method for Priorities in Hierarchical Structures,” Journal of Mathematical Psychology, vol. 281, pp. 234–281, 1977.

[61] 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.

[62] T. Z. Win, R. Mohamed, and J. Sallim, “Requirement Prioritization Based on Non-Functional Requirement Classification Using Hierarchy AHP,” IOP Conf. Ser. Mater. Sci. Eng., vol. 769, no. 1, 2020.

[63] N. Garg, M. Sadiq, and P. Agarwal, “GOASREP : Goal Oriented Approach for Software Requirements Elicitation and Prioritization Using Analytic Hierarchy Process,” 5th International Conference on Frontiers in Intelligent Computing, pp. 281–287, 2017.

[64] E. Bagheri, M. Asadi, D. Gasevic, and S. Soltani, “Stratified Analytic Hierarchy Process : Prioritization and Selection of Software Features,” International Conference on Software Product Lines, pp. 300–315, 2010.

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

[66] A. K. Massey, P. N. Otto, and A. I. Antón, “Prioritizing Legal Requirements,” IEEE, 2009 Second International Workshop on Requirements Engineering and Law , vol. 1936, no. 111, 2010.

[67] E. W. Myers, “FINDING ALL SPANNING TREES OF,” SIAM Journal on Computing, vol. 7, no. 3, pp. 280–287, 1978.

[68] H. Zhang, S. Member, T. Feng, G. Yang, and S. Member, "Distributed Cooperative Optimal Control for Multiagent Systems on Directed Graphs : An Inverse Optimal Approach," IEEE Transactions of Cybernetics, vol. 45, no. 7, pp. 1315–1326, 2015.

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

[70] R. Ramanathan, "Network Signatures from Image Representation of Adjacency Matrices: Deep/Transfer Learning for Subgraph Classification," no. i, 2018.

[71] 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.

[72] S. Dhingra, "Finding Strongly Connected Components in a Social Network Graph," International Journal of Computer Applications, vol. 136, no. 7, pp. 1–5, 2016.

[73] P. Chatzipetrou, L. Angelis, P. Rovegrd, 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.

[74] M. S. Jahan, F. Azam, M. W. Anwar, A. Amjad, and K. Ayub, "A Novel Approach for Software Requirement Prioritization," Proc. - 2019 7th Int. Conf. Softw. Eng. Res. Innov. CONISOFT 2019, pp. 1–7, 2019.

[75] T. Bebensee, I. Van De Weerd, and S. Brinkkemper, "Binary Priority List for Prioritizing Software Requirements," International Working Conference on Requirements Engineering: Foundation for Software Quality pp. 67–78, 2010.

[76] H. Pakizehkar, M. M. Sadrabadi, R. Z. Mehrjardi, and A. E. Eshaghieh, "The Application of Integration of Kano's Model, AHP Technique and QFD Matrix in Prioritizing the Bank's Substructions," Procedia - Soc. Behav. Sci., vol. 230, no. May, pp. 159–166, 2016.

[77] L. Lehtola and M. Kauppinen, "Suitability of requirements prioritization methods for market-driven software product development," Softw. Process Improv. Pract., vol. 11, no. 1, pp. 7–19, 2006.

[78] F. A. Bukhsh, Z. A. Bukhsh, and M. Daneva, "A systematic literature review on requirement prioritization techniques and their empirical evaluation," Comput. Stand. Interfaces, vol. 69, p. 103389, 2020.

[79] A. Hudaib, R. Masadeh and M. H. Qasem, “Requirements Prioritization Techniques Comparison,” Mod. Appl. Sci., vol. 12, no. 2, p. 62, 2018.

[80] A. S. Danesh and R. Ahmad, “Study of prioritization techniques using students as subjects,” , Proc. - 2009 Int. Conf. Inf. Manag. Eng. ICIME 2009, vol. 1, pp. 390–394, 2009.

[81] R. M. Liaqat, “A Majority Voting Goal Based Technique for Requirement Prioritization.”, 22nd Int. Conference on Automation and Computing, 2016.

[82] Y. V. Singh, B. Kumar, S. Chand, and J. Kumar, “A Comparative Analysis and Proposing ‘ANN Fuzzy AHP Model’ for Requirements Prioritization,” Int. J. Inf. Technol. Comput. Sci., vol. 10, no. 4, pp. 55–65, 2018.

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

[84] F. A. Cowell, “On the Structure of Additive Inequality Measures,” The Review Economic Studies, vol. 47, no. 3, pp. 521–531, 2015.

[85] S. Chern, S. Fu, and D. Tang, “Some inequalities for k -colored partition functions,” Ramanujan J., no. December 2017, 2018.

[86] C. Bessenrodt and K. Ono, “Maximal Multiplicative Properties of Partitions,” Annals of Combinatorics , vol. 20, pp. 59–64, 2016.

[87] K. Rohni, G. Sinaga, W. Witjaksono, and F. M. Al-anshary, “Payroll Administration System Implementation using ODOO At Pt. Primarindo Asia Infrastructure, Tbk With Rapid Application,” 1978.

[88] J. S. Informasi, S. Tinggi, T. Terpadu, and N. Fikri, “Perbandingan Modul Payroll Open ERP (Odoo) dengan Modul Payroll Adempiere,” Jurnal Sains, Teknologi dan Industri, vol. 13, no. 2, pp. 136–145, 2016.

[89] Y. Kendengis and L. W. Santoso, “Integration Between ERP Software and Business Intelligence in Odoo ERP : Case Study A Distribution Company,” Advances in Natural and Applied Sciences, pp. 1–4, 2018.

[90] N. Kountouridou, P. Antoniou, and I. Stamelos, “A comprehensive approach for implementing an open source ERP in a Greek industry,” ACM Int. Conf. Proceeding Ser., 2016.

[91] M. Guerci and M. Pedrini, “The consensus between Italian HR and sustainability managers on HR management for sustainability: Driven change – towards a ‘strong’ HR management system,” The International Journal of

Human. vol.39, no.13, pp. 37–41, 2013.

[92] K. Alaqeel et al., “Configurations and implementation of payroll system using open source erp : a case study of Koperasi PT Sri Configurations and implementation of payroll system using open source erp : a case study of Koperasi PT Sri,” IOP Conf. Series: Materials Science and Engineering, 2017.

[93] E. D. Santos and S. R. B. Oliveira, “Gamification and evaluation the use of the function points analysis technique in software quality subjects: The experimental studies,” ACM Int. Conf. Proceeding Ser., pp. 354–362, 2018.

[94] H. R. Ismaeel and A. S. Jamil, “Software Engineering Cost Estimation Using COCOMO II Model Al-Nahrain University,” Al-Mansour Journal, no. April, vol. 3, no.10, pp.86-111, 2007.

[95] F. Wang, X. Yang, X. Zhu, and L. Chen, “Extended use case points method for software cost estimation,” Proc. - 2009 Int. Conf. Comput. Intell. Softw. Eng. CiSE 2009, pp. 0–4, 2009.

[96] A. B. Nassif, L. F. Capretz, D. Ho, and M. Azzeh, “A treeboost model for software effort estimation based on use case points,” Proc. - 2012 11th Int. Conf. Mach. Learn. Appl. ICMLA 2012, vol. 2, pp. 314–319, 2012.

[97] O. Kikelomo and J. Olalekan, “Application of Critical Path Method (CPM) and Project Evaluation Review Techniques (PERT) in Project Planning and Scheduling,” J. Math. Stat. Sci., vol. 6, no. 1, pp. 1–8, 2012.

[98] M. Simion, G. Vasile, C. Dinu, and R. Scutariu, “CPM and PERT techniques for small-scale R&D projects,” Int. Symp. "The Environ. Ind., no. SIMI 2019, pp. 166–174, 2019.

[99] S. Mohan, M. Gopalakrishnan, H. Balasubramanian, and A. Chandrashekar, “A lognormal approximation of activity duration in PERT using two time estimates,” J. Oper. Res. Soc., vol. 58, no. 6, pp. 827–831, 2007.

[100] S. Atin and R. Lubis, “Implementation of Critical Path Method in Project Planning and Scheduling,” IOP Conf. Ser. Mater. Sci. Eng., vol. 662, no. 2, 2019.

[101] P. L. Primandaria and Sholiq, “Effort Distribution to Estimate Cost in Small to Medium Software Development Project with Use Case Points,” Procedia Comput. Sci., vol. 72, pp. 78–85, 2015.

[102] R. K. Clemmons, “Project estimation with Use Case Points,” The journal of defence software engineering, vol. 19, no. 2, pp. 18–22, 2006.

APPENDIX

Table A1: Classification of use-cases on the basis of number of transactions

Use-case Complexity	Number of Transactions	Use-case Weight
Simple	<= 3	5
Average	4 to 7	10
Complex	>7	15

Table A2: Classification of requirements of ODOO based on number of transactions

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

Table A3: Classification of Actors for each use-cases

Actor Complexity	Example	Actor Weight
Simple	A System with defined API	1
Average	A System interacting through a Protocol	2
Complex	A User interacting through GUI	3

Table A4: Factors of technical complexity

Factor	Description	Weight	Rated Values (0 to 5) (RV)	Impact (I = W * RV)
T1	Distributed System	2.0	0.0	0.0
T2	Response time or throughput performance objectives	1.0	4.0	4.0
T3	End user efficiency	1.0	4.0	4.0
T4	Complex internal processing	1.0	0.0	0.0
T5	Code must be reusable	1.0	4.0	4.0
T6	Easy to install	0.5	2.0	1.0
T7	Easy to use	0.5	4.0	2.0
T8	Portable	2.0	1.0	2.0
T9	Easy to change	1.0	4.0	4.0
T10	concurrent	1.0	0.0	0.0
T11	Includes special security objectives	1.0	0.0	0.0
T12	Provides direct access for third parties	1.0	0.0	0.0
T13	Special user training facilities are required	1.0	0.0	0.0
	Total Technical Factor (TFactor)			21.0

Table A5: Factors for environmental complexity

Factor	Description	Weight	Rated values (0 to 5) (RV)	Impact (I = W * RV)
F1	Familiar with the project model that is used	1.5	5.0	7.5
F2	Application experience	0.5	5.0	2.5
F3	Object-oriented experience	1.0	5.0	5.0
F4	Lead analyst capability	0.5	5.0	2.5
F5	Stable requirements	2.0	5.0	10.0
F6	Part-time staff	-1.0	4.0	-4.0
F7	Difficult programming language	-1.0	0.0	0.0
Total Environment Factor (EFactor)				**23.5**

Table A6: Pairwise comparing FRs with MST, SAHP

Notation	Compared with	Notation	Compared with
R1		R51	R42
R2	R1	R52	R41
R3		R53	R54
R4	R1	R54	
R5	R6, R8	R55	R54
R6	R6	R56	
R7		R57	
R8		R58	R37, R57
R9		R59	R42
R10	R1	R60	R34, R41
R11	R1	R61	R35
R12	R1, R16	R62	R35
R13	R16	R63	R37
R14		R64	R33
R15		R65	R41
R16		R66	R34
R17	R1	R67	R1, R8
R18	R1	R68	R60
R19	R17	R69	R61
R20	R17, R1	R70	R34
R21	R12, R14, R15	R71	R34
R22	R1	R72	R41
R23	R1	R73	R33
R24	R8	R74	
R25	R1	R75	
R26	R24	R76	
R27	R24	R77	
R28	R24	R78	
R29	R24	R79	R76
R30		R80	R70
R31	R29	R81	R1, R6
R32	R35	R82	
R33		R83	
R34		R84	
R35	R34, R33, R37	R85	
R36	R32, R33	R86	
R37		R87	
R38	R39	R88	
R39	R33	R89	
R40	R45	R90	R34
R41		R91	
R42	R34, R43, R41	R92	R91
R43		R93	R92
R44	R43, R41	R94	R93
R45		R95	R94
R46		R96	R95
R47	R46	R97	
R48		R98	
R49	R47	R99	R100
R50	R47	R100	

Table A7: Pairwise comparing FRs with BPL, BST

Notation	Compared with	Notation	Compared with
R1		R51	R42, R34, R43, R41
R2	R1	R52	R41
R3		R53	R54
R4	R1	R54	
R5	R6, R8	R55	R54
R6	R6	R56	
R7		R57	
R8		R58	R37, R57
R9		R59	R42, R34, R43, R41
R10	R1	R60	R34, R41
R11	R1	R61	R35, R34, R37
R12	R1, R16	R62	R35, R34, R37
R13	R16	R63	R37
R14		R64	R33
R15		R65	R41
R16		R66	R34, R34
R17	R1	R67	R1, R8
R18	R1	R68	R60, R34, R41
R19	R17, R1	R69	R61, R35, R34, R37
R20	R17, R1,	R70	R34
R21	R12, R14, R15, R1, R16	R71	R34
R22	R1	R72	R41
R23	R1	R73	R33
R24	R8	R74	
R25	R1	R75	
R26	R24, R8	R76	
R27	R24, R8	R77	
R28	R24, R8	R78	
R29	R24, R8	R79	R76
R30		R80	R70, R34
R31	R29, R24, R29	R81	R1, R6
R32	R35, R34, R37	R82	
R33		R83	
R34		R84	
R35	R34, R33, R37	R85	
R36	R32, R33, R35, R34, R37	R86	
R37		R87	
R38	R39, R33	R88	
R39	R33	R89	
R40	R45	R90	R34
R41		R91	
R42	R34, R43, R41	R92	R91
R43		R93	R92
R44	R43, R41	R94	R93
R45		R95	R94, R93
R46		R96	R95, R94, R93
R47	R46	R97	
R48		R98	
R49	R47, R46	R99	R100
R50	R47, R46	R100	

Table A8: Scale for Random Index (RI)

Requirement (n)	RI
1	0
2	0
3	0.58
4	0.9
5	1.12
6	1.24
7	1.32
8	1.41
9	1.45
10	1.49
11	1.51
12	1.48
13	1.56
14	1.57
15	1.59
16	1.60
17	1.62
18	1.62
19	1.63
20	1.65

Table A9: Delay of 5 UCP in UN-prioritized FRs

Group A		Group B		Group C		Group D		Group E	
Completion time		Completion time		Completion time		Completion time		Completion time	
R1	9	R25	4+9	R23	6.5+9	R2	4+9	R18	6.5+9
R4	4	R10	4	R22	6.5	R11	4	R17	11.5
R67	6.5+31	R20	4+10	R16	9	R19	6.5+10	R21	9+36.5
R13	4	R6	9	R12	4	R14	9	R28	4
R15	9	R81	6.5	R8,	9	R5	4+1.5	R66	6.5
R24	9	R29	4+26	R26	4+28.5	R27	4+24.5	R90	6.5
R31	4	R34	9	R41	9	R42	14+44	R33	9
R35	14+22	R70	6.5	R37	9	R71	6.5	R61	9+14
R59	6.5	R68	6.5+11.5	R60	9	R62	6.5	R52	4
R36	9	R80	6.5	R51	6.5+31	R69	6.5	R63	6.5
R65	6.5	R43	9	R32	6.5	R72	6.5	R39	11.5
R64	6.5	R44	6.5	R58	4+6.5	R49	6.5	R53	6.5+4
R47	4	R46	4	R50	6.5	R38	6.5	R79	6.5+22
R57	9	R73	6.5	R76	11.5	R55	6.5	R93	9
R45	11.5	R54	11.5	R40	6.5	R92	6.5	R100	6.5+32
R96	4+41.5	R91	11.5	R94	11.5+9	R73	11.5	R30	4
R99	11.5	R95	4+37..5	R7	4	R75	6.5	R77	4
R48	4	R9	4	R56	4	R84	6.5	R85	4
R78	4	R74	6.5	R82	4	R98	6.5	R88	4
R86	6.5	R83	6.5	R87	4				
		R97	6.5	R89	4				
237		230.5		223		217.5		246	

Table A10: Delay of 5 UCP in top priority group with 5 FRs (SAHP)

Group A		Group B		Group C		Group D		Group E	
Completion time		Completion time		Completion time		Completion time		Completion time	
R1	4	R34	9	R41	9	R2	4+4	R33	9
R4	4	R25	4	R8,	9	R11	4	R18	6.5
R67	6.5+18	R10	4	R37	9	R19	6.5+10	R17	6.5
R13	4	R20	4+5	R23	6.5	R14	4	R21	9+26
R15	4	R6	4	R22	6.5	R5	4	R28	4
R24	4	R81	6.5	R16	4	R27	4+8	R66	6.5
R31	4	R29 R24	4+16	R12	4	R42	9+38	R90	6.5
R35	9	R70	6.5	R26	4+18.5	R71	6.5	R61	9
R59	6.5+38	R68	6.5+6.5	R60	9	R62	6.5	R52	4
R36	9	R80	6.5	R51	6.5+16	R69	6.5	R63	6.5
R65	6.5	R43	4	R32	6.5	R7	6.5	R39	6.5
R64	6.5	R44	6.5	R58	4+23.5	R49	6.5+4.5	R53	6.5+10 +32.5
R47	4	R46	4	R50	6.5	R38	6.5	R79	6.5
R57	4	R73	6.5	R76	6.5	R55	6.5	R93	4
R45	6.5	R54	6.5	R40	6.5	R92	6.5	R100	6.5+21
R96	4+31.5	R91	6.5	R94	6.5+4	R73	6.5	R30	4
R99	6.5	R95	4+49.5	R7	4	R75	6.5	R77	4
R48	4	R9	4	R56	4	R84	6.5	R85	4
R78	4	R74	6.5	R82	4	R98	6.5	R88	4
R86	6.5	R83	6.5	R87	4				
		R97	6.5	R89	4				
195		**193.5**		**186**		**178**		**203**	

Table A11: Delay of 5 UCP in top priority group with 15 FRs (SAHP)

Group A		Group B		Group C		Group D		Group E	
Completion time		Completion time		Completion time		Completion time		Completion time	
R1	9	R34	9	R8,	9	R14	9	R33	9
R15	9	R6	9	R37	9	R2	4	R93	9
R45	11.5	R43	9	R41	9	R11	4	R18	6.5
R4	4	R54	11.5	R16	9	R19	6.5+14	R17	6.5
R67	6.5	R91	11.5	R23	6.5	R5	4	R21	9+15.5
R13	4	R25	4	R12	4	R27	4+6.5	R28	4
R24	4	R10	4	R22	6.5	R42	9	R66	6.5
R31	4+24.5	R20	4	R26	4	R71	6.5	R90	6.5
R35	9	R81	6.5	R60	9	R62	6.5	R61	9
R59	6.5	R29	4	R51	6.5	R69	6.5	R52	4
R36	9	R70	6.5	R32	6.5	R72	6.5	R63	6.5
R65	6.5	R68	6.5	R58	4+43	R49	6.5+31	R39	6.5
R64	6.5	R80	6.5	R50	6.5	R38	6.5	R53	6.5
R47	4	R44	6.5	R76	6.5	R55	6.5	R79	6.5+34
R57	4	R46	4	R40	6.5	R92	6.5	R100	6.5+21
R96	4+34	R73	6.5	R94	6.5	R73	6.5	R30	4
R99	6.5	R95 R94	4+43	R7	4	R75	6.5	R77	4
R48	4	R9	4	R56	4	R84	6.5	R85	4
R78	4	R74	6.5	R82	4	R98	6.5	R88	4
R86	6.5	R83	6.5	R87	4				
		R97	6.5	R89	4				
181		179.5		172		170		189	

Table A12: Delay of 5 UCP in top priority group with 20 FRs (SAHP)

Group A		Group B		Group C		Group D		Group E	
Completion time		Completion time		Completion time		Completion time		Completion time	
R1	9	R34	9	R41	9	R14	9	R33	9
R15	9	R6	9	R8,	9	R2	4	R93	9
R45	11.5	R43	9	R16	9	R11	4	R39	11.5
R24	9	R54	11.5	R37	9	R19	6.5+25.5	R18	6.5
R99	11.5	R91	11.5	R76	11.5	R5	4	R17	6.5
R35	14	R25	4	R23	6.5	R27	4	R21	9+22
R4	4	R10	4	R22	6.5	R42	9	R28	4
R67	6.5	R20	4	R12	4	R71	6.5	R66	6.5
R13	4	R81	6.5	R26	4	R62	6.5	R90	6.5
R31	4	R29	4	R60	9	R69	6.5	R61	9
R59	6.5	R70	6.5	R51	6.5	R72	6.5	R52	4
R36	9	R68	6.5	R32	6.5	R49,	6.5+23	R63	6.5
R65	6.5	R80	6.5	R58	4+25	R38	6.5	R53	6.5
R64	6.5	R44	6.5	R50	6.5	R55,	6.5	R79	6.5
R47	4	R46	4	R40	6.5	R92	6.5	R100	6.5
R57	4	R73	6.5	R94	6.5	R73	6.5	R30	4
R96	4+24	R95	4+30	R7	4	R75	6.5	R77	4
R48	4	R9	4	R56	4	R84	6.5	R85	4
R78	4	R74	6.5	R82	4	R98	6.5	R88	4
R86	6.5	R83	6.5	R87	4				
		R97	6.5	R89	4				
161.5		166.5		159		167		145.5	

Table A13: Delay of 5 UCP in top priority group with 25 FRs (SAHP)

Group A		Group B		Group C		Group D		Group E	
Completion time		Completion time		Completion time		Completion time		Completion time	
R1	9	R34	9	R8	9	R14	9	R33	9
R4	4	R6	9	R37	9	R42	14	R93	9
R15	9	R43	9	R41	9	R2	4	R39	11.5
R45	11.5	R54	11.5	R16	9	R11	4	R17	11.5
R24	9	R91	11.5	R76	11.5	R19	6.5	R18	6.5
R99	11.5	R25	4	R94	11.5	R5	4	R21	9+22
R35	14	R10	4	R23	6.5	R27	4	R28	4
R57	9	R20	4	R12	4	R71	6.5	R66	6.5
R47	4	R81	6.5	R22	6.5	R62	6.5	R90	6.5
R67	6.5	R29	4	R26	4	R69	6.5+24	R61	9
R13	4	R70	6.5	R60	9	R72	6.5	R52	4
R31	4	R68	6.5	R51	6.5	R49	6.5	R63	6.5
R59	6.5	R80	6.5	R32	6.5	R38	6.5	R53	6.5
R36	9	R44	6.5	R58	4	R55	6.5	R79	6.5
R65	6.5	R46	4	R50	6.5	R92	6.5	R100	6.5
R64	6.5	R73	6.5	R40	6.5	R73	6.5	R30	4
R96	4	R95	4	R7	4	R75	6.5	R77	4
R48	4	R9	4	R56	4	R84	6.5	R85	4
R78	4	R74	6.5	R82	4	R98	6.5	R88	4
R86	6.5	R83	6.5	R87	4				
		R97	6.5	R89	4				
142.5		**136.5**		**139**		**147.5**		**150**	

Table A14: Delay of 5 UCP in top priority group with 25 FRs (AHP)

Group A		Group B		Group C		Group D		Group E	
Completion time		Completion time		Completion time		Completion time		Completion time	
R1	9	R34	9	R41	9	R42	9	R33	9
R35	14	R46	9	R8	9	R14	9	R93	9
R24	9	R43	9	R37	9	R2	4	R3	11.5
R57	9	R54	11.5	R16	9	R11	4	R18	6.5
R45	11.5	R6	9	R60	14	R19	6.5+16.5	R17	6.5
R15	9	R91	11.5	R94	6.5	R5	4	R21	9+19
R4	4	R97	11.5	R23	6.5	R27	4	R28	4
R67	6.5	R70	11.5	R12	4	R71	6.5	R66	6.5
R13	4	R25	4	R22	6.5	R62	6.5	R90	6.5
R31	4+28.5	R10	4	R26	4	R69	6.5+26.5	R61	9
R59	6.5	R20	4	R51	6.5	R72	6.5	R52	4
R36	9	R81	6.5	R32	6.5	R49	6.5+31.5	R63	6.5
R65	6.5	R29	4	R58	4	R38	6.5	R53	6.5
R64	6.5	R68	6.5	R50	6.5+46.5	R55	6.5	R79	6.5+40.5
R47	4	R80	6.5	R76	6.5	R92	6.5	R100	6.5
R96	4	R44	6.5	R40	6.5	R73	6.5	R30	4
R99	6.5	R73	6.5	R7	4	R75	6.5	R77	4
R48	4	R95	4	R56	4	R84	6.5	R85	4
R78	4	R9	4	R82	4	R98	6.5	R88	4
R86	6.5	R74	6.5	R87	4				
		R83	6.5	R89	4				
162		151.5		180.5		193		183	

Table A15: Delay of 5 UCP in top priority group with 25 FRs (MST)

Group A		Group B		Group C		Group D		Group E	
Completion time		Completion time		Completion time		Completion time		Completion time	
R1	9	R34	9	R8,	9	R71	11.5+9	R33	9
R24	9	R43	9	R37	9	R2	9	R93	9
R35	14	R46	9	R94	11.5	R11	4	R66	11.5
R78	4	R73	11.5	R41	9	R19	6.5+43.5	R90	11.5
R64	11.5	R25	9	R16	9	R14	4	R63	11.5
R4	4	R70	11.5	R23	11.5	R27	4	R39	11.5
R67	6.5+33	R10	4	R12	4	R42	9	R18	6.5
R13	4	R20	4+14	R22	6.5	R62	6.5	R17	6.5
R15	4	R6	4	R26	4	R69	6.5+14.5	R21	9+22.5
R31	4	R81	6.5	R60	9	R72	6.5	R28	4
R59	6.5	R29	4	R51	6.5+18	R49	6.5	R61	9
R36	9+4	R68	6.5	R32	6.5	R5	9	R52	4
R65	6.5	R80	6.5	R58	4+23.5	R38	6.5	R53	6.5
R47	4	R44	6.5	R50	6.5	R55	6.5	R79	6.5+22
R57	4	R54	6.5	R76	6.5	R92	6.5	R100,	6.5
R45	6.5	R91	6.5	R40	6.5	R73	6.5	R30	4
R96	4	R95	4	R7	4	R75	6.5	R77	4
R99	6.5	R9	4	R56	4	R84	6.5	R85	4
R48	4	R74	6.5	R82	4	R98	6.5	R88	4
R86	6.5	R83	6.5	R87	4				
		R97	6.5	R89	4				
164.5		162		180.5		195.5		183	