AN INTEGRATED APPROACH FOR SIMULTANEOUS PRIORITIZATION OF FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

MOHAMMAD DABBAGH

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2015

AN INTEGRATED APPROACH FOR SIMULTANEOUS PRIORITIZATION OF FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

MOHAMMAD DABBAGH

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2015

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: **MOHAMMAD DABBAGH** (I.C/Passport No:)

Registration/Matric No: **WHA110052**

Name of Degree: **DOCTOR OF PHILOSOPHY**

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

AN INTEGRATED APPROACH FOR SIMULTANEOUS PRIORITIZATION OF FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

Field of Study: **REQUIREMENTS PRIORITIZATION**

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

ABSTRACT

Requirements prioritization, as one of the important activities in the requirements engineering phase, is the process of giving precedence to one requirement over another to help accomplishing software projects on a predefined schedule. Over the recent past years, a substantial amount of research effort has been dedicated to proposing various approaches to perform requirements prioritization. Although these approaches have contributed a lot to the software development process, an in-depth study of current research work has found that they can only be applied with functional requirements or non-functional requirements separately though most have been adopted with respect to functional requirements. However, it is not an effective way to prioritize functional and non-functional requirements separately since both types of requirements are interrelated and have an influence on each other. Moreover, considering merely each type of requirements separately during the requirements prioritization process may lead to failure in the final software product, or at least, may result in a poor-quality system because both types of requirements have a serious impact on each other. To achieve a high-quality software system, both functional and non-functional requirements need to be taken into consideration together during the prioritization process. Hereupon, fulfilling this gap is considered as a major motivation toward conducting this research to provide software researchers and practitioners with an approach which is capable of integrating the process of prioritizing functional and non-functional requirements simultaneously.

In this research, an approach called Integrated Prioritization Approach (IPA), is proposed, which aims to simultaneously integrate the process of prioritizing functional and non-functional requirements. IPA allows practitioners to prioritize both functional and non-functional requirements simultaneously in an integrated manner by establishing their relationships, ultimately producing the prioritized lists of functional and non-

functional requirements separately. It utilizes triangular fuzzy number to prioritize non-functional requirements based on their importance degree for achieving functional requirements. Furthermore, IPA prioritizes functional requirements according to the importance degrees of non-functional requirements using weighted average decision matrix. Two successive controlled experiments were conducted to evaluate IPA against two state-of-the-art approaches. In the first experiment, IPA was compared with AHP-based approach, whereas in the second experiment, IPA was compared with HAM-based approach. In both experiments, evaluation was based on measuring three properties: actual time-consumption, accuracy of results, and ease of use.

Statistical analysis of the experimental results obtained from the two controlled experiments shows the better performance of IPA compared to both AHP-based approach and HAM-based approach, with respect to actual time-consumption, accuracy of results, and ease of use. IPA requires a time reduction of 43% and 20% compared to AHP-based approach and HAM-based approach respectively, to perform the prioritization task. Furthermore, IPA is validated to be easier to use and produces more accurate results compared to these two approaches at 95% confidence level (i.e. $p<0.05$). Results extracted from the conducted experiments provide practitioners with valuable information to choose the most appropriate approach for a given prioritization problem, and also could be used as a guideline by interested researchers for identifying trends before conducting a study in future.

ABSTRAK

Keperluan keutamaan, sebagai salah satu aktiviti penting dalam fasa kejuruteraan keperluan, adalah proses memberi keutamaan kepada satu keperluan ke atas yang lain untuk membantu menyelesaikan projek perisian mengikut jadual yang telah ditetapkan. Sejak beberapa tahun kebelakangan yang lalu, sejumlah besar usaha penyelidikan telah didedikasikan untuk mencadangkan pelbagai pendekatan untuk melaksanakan keperluan keutamaan. Walaupun pendekatan ini telah banyak menyumbang kepada proses pembangunan perisian, satu kajian yang mendalam dalam kerja penyelidikan semasa telah mendapati bahawa mereka hanya boleh digunakan dengan keperluan fungsian atau keperluan bukan fungsian secara berasingan sungguhpun kebanyakan telah diterima pakai berkenaan dengan keperluan fungsian. Walau bagaimanapun, ia bukanlah cara yang berkesan untuk mengutamakan keperluan fungsian dan bukan fungsian secara berasingan kerana kedua-dua jenis keperluan adalah saling berkaitan dan mempunyai pengaruh ke atas satu sama lain. Selain itu, mempertimbangkan hanya setiap jenis keperluan berasingan semasa proses keutamaan keperluan itu boleh membawa kepada kegagalan dalam produk perisian terakhir, atau sekurang-kurangnya, boleh menyebabkan sistem yang berkualiti rendah kerana kedua-dua jenis keperluan memberi kesan yang serius ke atas satu sama lain. Untuk mencapai sistem perisian yang berkualiti tinggi, kedua-dua keperluan fungsian dan bukan fungsian perlu diambil kira bersama-sama semasa proses keutamaan itu. Setelah itu, memenuhi jurang ini dianggap sebagai motivasi utama ke arah menjalankan kajian ini untuk menyediakan penyelidik dan pengamal perisian dengan pendekatan yang mampu menyepadukan proses keutamaan keperluan fungsian dan bukan fungsian pada masa yang sama.

Dalam kajian ini, pendekatan yang dikenali sebagai Pendekatan Keutamaan Bersepadu (IPA), adalah dicadangkan, yang bertujuan untuk mengintegrasikan proses keutamaan keperluan fungsian dan bukan fungsian pada masa yang sama. IPA

membolehkan pengamal untuk mengutamakan kedua-dua keperluan fungsi dan bukan fungsi dengan serentak secara bersepadu dengan mewujudkan hubungan mereka, akhirnya menghasilkan senarai keutamaan dalam keperluan fungsian dan bukan fungsian secara berasingan. Ia menggunakan nombor kabur segi tiga untuk mengutamakan keperluan bukan fungsi berdasarkan pada tahap kepentingan mereka untuk mencapai keperluan fungsian. Tambahan pula, IPA mengutamakan keperluan fungsian mengikut darjah kepentingan keperluan bukan fungsi dengan menggunakan wajaran matriks keputusan purata. Dua eksperimen terkawal telah dijalankan secara berturut-turut untuk menilai IPA terhadap dua pendekatan terkini. Dalam eksperimen pertama, IPA telah dibandingkan dengan pendekatan berasaskan-AHP, sedangkan dalam eksperimen kedua, IPA telah dibandingkan dengan pendekatan berasaskan HAM. Dalam kedua-dua eksperimen, penilaian adalah berdasarkan pada berukuran tiga sifat: masa penggunaan sebenar, ketepatan keputusan, dan kemudahan penggunaan.

Analisis statistik daripada keputusan eksperimen yang diperolehi daripada dua eksperimen terkawal tersebut menunjukkan prestasi yang lebih baik daripada IPA berbanding kedua-dua pendekatan berasaskan-AHP dan pendekatan berasaskan HAM, berkenaan dengan masa penggunaan sebenar, ketepatan keputusan, dan kemudahan penggunaan. IPA memerlukan pengurangan masa sebanyak 43% dan 20% berbanding dengan pendekatan berasaskan-AHP dan pendekatan berasaskan HAM masing-masing, untuk melaksanakan tugas keutamaan itu. Tambahan pula, IPA disahkan lebih mudah untuk digunakan dan menghasilkan keputusan yang lebih tepat berbanding dengan kedua-dua pendekatan pada 95% tahap keyakinan (iaitu $p < 0.05$). Keputusan yang terhasil daripada eksperimen terkawal itu dapat menyediakan kepada pengamal maklumat yang bernilai untuk memilih pendekatan yang paling sesuai untuk masalah keutamaan diberikan, dan juga boleh digunakan sebagai satu garis panduan oleh

penyelidik untuk mengenal pasti trend sebelum menjalankan kajian pada masa akan datang.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank God for given me the wisdom, knowledge and strength to complete this work successfully.

Secondly, I would like to express my profound gratitude to my supervisor, Professor Dr. Lee Sai Peck, whose constructive criticisms and invaluable contributions have enabled me to complete this work successfully. Thank you so much Prof. Lee for all your support right from the beginning of this thesis to the end and God bless you for all the patience and advice.

I owe a great debt of gratitude to my beloved parents, Masoud and Jaleh, for educating me to this level and all their support. Mum and Dad, I say God richly bless you for all that you have been doing for me. I also give special thanks to my sister, Nazanin, for her kind support and contribution to the success of my studies.

Mohammad, April 2015

TABLE OF CONTENTS

Abstract iii

Abstrak v

Acknowledgements viii

Table of Contents ix

List of Figures xiv

List of Tables xvii

List of Appendices xx

CHAPTER 1: INTRODUCTION 1

1.1 Introduction 1

1.2 Background 2

1.3 Motivation and Problem Statement 4

1.4 Research Scope 6

1.5 Research Objectives 7

1.6 Research Questions 8

1.7 Research Approach 9

1.8 Significance of the Research 11

1.9 Summary 11

CHAPTER 2: LITERATURE REVIEW 13

2.1 Basic Concepts in the Context of Requirements Prioritization 13

2.1.1 The Role of Requirements Prioritization in Requirements Engineering 13

2.1.2 Requirement 17

2.1.3 Requirements Prioritization 21

2.1.3.1 Aspects of Requirements Prioritization 24

2.1.3.2 Different Types of Product Developments in the Prioritization Process 28

2.2 Requirements Prioritization Approaches 31
2.2.1 Ratio Scale Approaches 33
2.2.2 Ordinal Scale Approaches 50
2.2.3 Nominal Scale Approaches 55
2.2.4 Interval Scale Approaches 59
2.2.5 Overview of Current Requirements Prioritization Approaches 60
2.3 Empirical Evaluations of Requirements Prioritization Approaches 63
2.3.1 The Evaluation of Minimal Spanning Tree, Bubble Sort, Binary Search Tree, Priority Groups, Hierarchy AHP, and AHP 63
2.3.2 The Evaluation of Hundred Dollar Method, Planning Game, AHP, Binary Search Tree, and PGcAHP 65
2.3.3 The Evaluation of Hundred Dollar Method, AHP, MoSCoW, and Simple Ranking 67
2.3.4 The Evaluation of AHP and Planning Game 68
2.3.5 The Evaluation of AHP and CBRank 71
2.3.6 Overview of Empirical Studies 73
2.4 Concluding Remarks 75
2.5 Summary 76
CHAPTER 3: RESEARCH METHODOLOGY 77
3.1 Preparation Phase 77
3.2 Approach Development and Validation Phase 79
3.2.1 Propose an Integrated Approach for Simultaneous Prioritization of Functional and Non-functional Requirements 80
3.2.2 Validate the Proposed Approach through a Case Study 81
3.2.3 Validate the Proposed Approach through Mathematical Theory 82
3.3 Evaluation Phase 82

3.3.1 Experiment Definition....83
3.3.2 Experiment Planning....84
3.3.3 Experiment Execution....84
3.3.4 Experiment Analysis....85
3.4 Summary....85
CHAPTER 4: THE PROPOSED APPROACH....87
4.1 Integrated Prioritization Approach (IPA)....87
4.1.1 Step 1: Identify stakeholders of software system....92
4.1.2 Step 2: Specify the weights of stakeholders using Analytic Hierarchy Process 92
4.1.3 Step 3: Identify functional and non-functional requirements....93
4.1.4 Step 4: Extract functional and non-functional requirements statements....94
4.1.5 Step 5: Construct the decision matrix....95
4.1.6 Step 6: Elicit the importance degree of each non-functional requirement with respect to each functional requirement to establish the relationship between functional and non-functional requirements....95
4.1.7 Step 7: Calculate NFRs ranking with respect to all FRs using triangular fuzzy number and alpha cut approach....97
4.1.8 Step 8: Compute FRs ranking using weighted average decision matrix and weights determined in Step 7....100
4.1.9 Step 9: Aggregate different prioritized lists of FRs and NFRs provided by various stakeholders to obtain final rankings of FRs and NFRs....101
4.2 Summary....103
CHAPTER 5: VALIDATION OF THE PROPOSED APPROACH....104
5.1 Validate the Proposed Approach through Case Study....104
5.2 Validate the Proposed Approach using Mathematical Theory....117
5.2.1 Mathematical Formulation of the Proposed Approach....117

5.2.2 Validation of the Proposed Approach using Graph Theory 120
5.3 Summary 124
CHAPTER 6: EVALUATION OF THE PROPOSED APPROACH THROUGH CONTROLLED EXPERIMENTS 125
6.1 Controlled Experiments 126
6.1.1 Experiment Definition 127
6.1.1.1 Definition of Experiment 1 127
6.1.1.2 Definition of Experiment 2 128
6.1.2 Experiment Planning 129
6.1.2.1 Context Selection 129
6.1.2.2 Hypothesis Formulation 130
6.1.2.3 Variables and Measures 131
6.1.2.4 Selection of subjects 142
6.1.2.5 Experiment design 143
6.1.2.6 Subjects 143
6.1.2.7 Objects 144
6.1.2.8 Instrumentation 145
6.1.2.9 Threats to validity 146
6.1.3 Experiment execution 150
6.1.4 Experiment results and analysis 151
6.1.4.1 Results of Experiment 1 152
6.1.4.2 Results of Experiment 2 157
6.1.5 Discussion 162
6.2 Summary 165
CHAPTER 7: CONCLUSION 167
7.1 Contributions 167

7.2 Implications 169

7.3 Achievement of Research Objectives 170

7.4 Limitation 173

7.5 Future work 173

REFRENCES 175

List of Publications and Presented Papers 185

Appendices 186

LIST OF FIGURES

Figure 1.1: The scope of the current thesis 7

Figure 1.2: Flow chart of research approach 10

Figure 2.1: The activities of requirements engineering process 15

Figure 2.2: An example of cost-value diagram 38

Figure 2.3: Value-oriented prioritization framework 40

Figure 2.4: EVOLVE approach to select requirements for each release 44

Figure 2.5: Steps of CBRank approach for prioritizing requirements 46

Figure 2.6: The general process of VIRP approach for requirements prioritization 47

Figure 3.1: Flow chart of the research methodology 78

Figure 3.2: A graphical overview of controlled experiment process 83

Figure 3.3: Overview of experiment planning 84

Figure 3.4: Process of experiment analysis 85

Figure 4.1: Flow charts of the proposed approach when applied A) in single decision-making problems; B) in group decision-making problems 90

Figure 4.2: A sketch of algorithm used for automating the IPA 91

Figure 4.3: Fuzzy priority vector, $\widetilde{F_x}$. 98

Figure 5.1: The NFRs' weights achieved by four stakeholders participated in the prioritization process 112

Figure 5.2: The final prioritized list of FRs ranked according to their relationships with NFRs 116

Figure 5.3: The final prioritized list of NFRs ranked according to their relationships with FRs 117

Figure 5.4: A sample of complete weighted graph for three functional requirements . 121

Figure 5.5: Flow chart of identifying the single path which indicates the final prioritized list 122

Figure 5.6: The completed weighted graph of non-functional requirements 123

Figure 5.7: The acyclic directed sub graph indicates the final prioritized list of non-functional requirements 123

Figure 5.8: The acyclic directed sub graph indicates the final prioritized list of functional requirements 124

Figure 6.1: The process of prioritizing functional and non-functional requirements using a) IPA; b) AHP-based approach; c) HAM-based approach 132

Figure 6.2: A picture of the visual user interface displaying the functional requirement, withdraw money, versus the non-functional requirement, availability, under analysis with TIPA 134

Figure 6.3: A picture of the visual user interface displaying two functional requirements, withdraw money, and check balance, under evaluation in CAHP 136

Figure 6.4: A picture of pairwise comparison of two non-functional requirements, Availability, and Security, using CHAM 139

Figure 6.5: A picture of the visual user interface displaying the functional requirement, withdraw money, versus the non-functional requirement, Availability, under analysis with CHAM 140

Figure 6.6: Boxplot of the actual time-consumption associated with the two evaluated prioritization approaches 152

Figure 6.7: Boxplot of the actual time-consumption associated with IPA and HAM-based approach 158

Figure A.1: Use case diagram of the TIPA software prototype 189

Figure A.2: User interface of the TIPA software prototype along with its Projects form 193

Figure A.3: A snapshot of Stakeholders windows form 194

Figure A.4: A graphical view of FRs windows form 195

Figure A.5: A snapshot of NFRs windows form 196

Figure A.6: A picture of pairwise comparisons of two stakeholders, User1 and User4, using TIPA 197

Figure A.7: Calculated weights of stakeholders using TIPA 198

Figure A.8: A picture of the visual user interface displaying the functional requirement, Transfer funds, versus the non-functional requirement, Security, under analysis with TIPA, for project, Banking system 199

Figure A.9: Prioritized list of functional and non-functional requirement calculated using TIPA 199

Figure A.10: A graphical view of the report that indicates the weights of stakeholders 200

Figure A.11: A visual interface of a report that shows the weights of non-functional requirements achieved by each stakeholder for project, Banking system 201

Figure A.12: A graphical view of a report that indicates the weights of functional requirements achieved by each stakeholder for project, Banking system 202

Figure A.13: A visual interface of the TIPA software prototype that shows the final prioritized lists of functional and non-functional requirements for a given project 203

Figure E.1: Post-test 1A of Experiment 1 for assessing ease of use of IPA 215

Figure E.2: Post-test 1A of Experiment 1 for assessing ease of use of AHP-based approach 215

Figure E.3: Post-test 2A of Experiment 1 for comparing ease of use of IPA and AHP-based approach 215

Figure E.4: Post-test 1B of Experiment 1 for assessing expected accuracy of IPA..... 216

Figure E.5: Post-test 1B of Experiment 1 for assessing expected accuracy of AHP-based approach 216

Figure E.6: Post-test 1A of Experiment 2 for assessing ease of use of IPA 216

Figure E.7: Post-test 1A of Experiment 2 for assessing ease of use of HAM-based approach 217

Figure E.8: Post-test 2A of Experiment 2 for comparing ease of use of IPA and HAM-based approach 217

Figure E.9: Post-test 1B of Experiment 2 for assessing expected accuracy of IPA..... 217

Figure E.10: Post-test 1B of Experiment 2 for assessing expected accuracy of HAM-based approach 217

LIST OF TABLES

Table 2.1: Overview of definitions of the term “Requirement”.....18

Table 2.2: Different categories of requirements 19

Table 2.3: Different definitions of functional requirements 19

Table 2.4: Different definitions of non-functional requirements.....20

Table 2.5: A comparison between bespoke and market-driven software development..29

Table 2.6: Classification of requirements prioritization approaches 32

Table 2.7: Overview of requirements prioritization approaches.....60

Table 2.8: Results of objective and subjective measurements.....65

Table 2.9: Initial results of controlled experiment.....66

Table 2.10: Results of times taken, median confidence, and median difficulty 68

Table 2.11: Properties of the prioritization approaches 68

Table 2.12: Results of the first controlled experiment for comparing AHP and Planning Game 70

Table 2.13: Results of evaluating tool-supported AHP and Planning Game.....71

Table 2.14: Results of the controlled experiment for comparing AHP and CBRank 72

Table 2.15: An overview of comparative evaluations of different prioritization approaches.....74

Table 4.1: Steps of IPA for integrating the prioritization of functional and non-functional requirements.....88

Table 4.2: Possible scales used for AHP’s pairwise comparison 93

Table 4.3: IPA nominal scale, IPA actual scale.....96

Table 4.4: Weighted average decision matrix for priority assessment of FRs.....100

Table 4.5: Weighted average matrix to aggregate different prioritized list of FRs 102

Table 4.6: Weighted average matrix to aggregate different prioritized list of NFRs ...102

Table 5.1: The functional requirements of the ATM, CDM, and CQM 107

Table 5.2: The non-functional requirements of the banking software system 107

Table 5.3: Filling up the four decision matrixes with nominal scale values 109

Table 5.4: Computation of NFRs' priority vectors with respect to all FRs 111

Table 5.5: Calculation of FRs' priority vectors with respect to NFRs 113

Table 5.6: Calculating the prioritized list of FRs 115

Table 5.7: Computing the prioritized list of NFRs 116

Table 6.1: Overview of the controlled experiments 127

Table 6.2: Scales used in AHP method 135

Table 6.3: Scales used in HAM method 138

Table 6.4: The paired comparison design used for controlled experiments 143

Table 6.5: Objects of the controlled experiments 145

Table 6.6: Average actual time-consumption for the prioritization task using IPA and AHP-based approach 152

Table 6.7: Normality test of data extracted from Experiment 1 using Shapiro-wilk test 153

Table 6.8: Results of ease of use collected from post-test 1A of Experiment 1 154

Table 6.9: Results of ease of use extracted from post-test 2A of Experiment 1 155

Table 6.10: Results of expected accuracy collected from post-test 1B of Experiment 1 155

Table 6.11: Results of perceived accuracy collected from post-test 2B of Experiment 1 156

Table 6.12: Analysis the effect of other variables on dependent variables of Experiment 1 157

Table 6.13: Average actual time-consumption for the prioritization task using IPA and HAM-based approach 158

Table 6.14: Normality test of data extracted from Experiment 2 using Shapiro-Wilk test 159

Table 6.15: Results of ease of use obtained from post-test 1A of Experiment 2 159

Table 6.16: Results of ease of use extracted from post-test 2A of Experiment 2 159

Table 6.17: Results of expected accuracy collected from post-test 1B of Experiment 2 160

Table 6.18: Results of perceived accuracy collected from post-test 2B of Experiment 2 161

Table 6.19: Analysis the effect of other variables on dependent variables of Experiment 2 161

Table 6.20: Summary of hypotheses testing of Experiment 1 for comparing IPA and AHP-based approach 162

Table 6.21: Overview of hypotheses testing of Experiment 2 for comparing IPA and HAM-based approach 164

Table B.1: Classification of empirical strategies 207

Table C.1: Results of time-consumption of IPA and AHP-based approaches collected from twenty subjects of Experiment 1 209

Table C.2: Results of ease of use of IPA and AHP-based approaches collected from twenty subjects of Experiment 1 using post-test 1A 210

Table C.3: Results of ease of use collected from post-test 2A for Experiment 1 210

Table C.4: Results of expected accuracy of IPA and AHP-based approaches collected from twenty subjects of Experiment 1 using post-test 1B 211

Table C.5: Results of perceived accuracy collected from post-test 2B for Experiment 1 211

Table D.1: Results of time-consumption of IPA and HAM-based approaches collected from twenty subjects of Experiment 2 212

Table D.2: Results of ease of use of IPA and HAM-based approaches collected from twenty subjects of Experiment 2 using post-test 1A 213

Table D.3: Results of ease of use collected from post-test 2A for Experiment 2 213

Table D.4: Results of expected accuracy of IPA and HAM-based approaches collected from twenty subjects of Experiment 2 using post-test 1B 214

Table D.5: Results of perceived accuracy collected from post-test 2B for Experiment 2 214

LIST OF APPENDICES

APPENDIX A: THE SOFTWARE PROTOTYPE 187

APPENDIX B: OVERVIEW OF EMPIRICAL STRATEGIES 204

APPENDIX C: RAW DATA OF CONTROLLED EXPERIMENT 1..... 209

APPENDIX D: RAW DATA OF CONTROLLED EXPERIMENT 2 212

APPENDIX E: POST QUESTIONNAIRES OF CONTROLLED EXPERIMENTS .. 215

CHAPTER 1: INTRODUCTION

1.1 Introduction

The ultimate goal of developing any high quality software system is to satisfy various stakeholders' needs and expectations (J. Karlsson & Ryan, 1997). Hence, managing the software requirements process plays a critical role towards the success of a software development project (Hofmann & Lehner, 2001). Requirements engineering, as a first step in the software development process, and its underlying activities can help practitioners to understand stakeholders' needs and develop high quality software in an economic manner.

However, in real-world software development, projects have to regularly deal with time and to meet market constraints, budget deadlines, restricted technology, and limited human power. Considering these constraints, projects often are not able to address all the requirements in one product release. When projects contain more requirements than can be addressed in one product release, requirement engineers have to make decisions on which requirements need to be considered first. In addition, during the initial stage of software development process, often it is not obvious which requirements strongly affect stakeholders' satisfaction among the candidate list of requirements.

Due to the constraints mentioned above, it could be a challenge for requirements engineers to decide which requirements lead to high stakeholders' satisfaction and need to be considered first. Therefore, to address this concern and in order to reduce the cost and duration of a software project as well, it is essential to address the high-priority requirements before considering the low-priority ones (Duan, Laurent, Cleland-Huang, & Kwiatkowski, 2009; Liu, Sun, Veera, Kyoya, & Noguchi, 2006). Requirements

prioritization can help to identify the most important requirements for a software system (Sommerville, 2007), and then proceed to develop the software according to these requirements. Hence, requirements prioritization has been recognized as one of the most important decision-making processes during the software development process (Achimugu, Selamat, Ibrahim, & Mahrin, 2014; Perini, Susi, & Avesani, 2013). Furthermore, the most frequently addressed topic in the requirements engineering domain is requirements prioritization (Babar, Ghazali, & Jawawi, 2014; Daneva, Damian, Marchetto, & Pastor, 2014).

This research is conducted to introduce a prioritization approach which addresses both functional and non-functional requirements. In other words, the new approach aims to facilitate software engineers with a way which can assist them to integrate the process of prioritizing functional and non-functional requirements simultaneously.

1.2 Background

As the complexity of software systems increases, practitioners are forced to make trade-offs between conflicting requirements in order to complete projects on predefined schedule. Priority assessment of requirements is one of the solutions which can be useful to assist practitioners to resolve trade-offs. Thus, requirements prioritization has become an increasingly important part of ensuring the success of a project, and therefore, various studies pointed out the importance of the problem of requirements prioritization in the software engineering domain. Requirements prioritization is defined as an activity during which the most important requirements for the system (or release) should be identified in order to maximize the stakeholders' satisfaction (Sommerville, 2007). To perform the prioritization process, stakeholders are requested to compare requirements using a scoring method with the aim of determining the importance value of each requirement. Prioritizing requirements prior to architecture design or

implementation phase provides many benefits throughout the software development process (Achimugu et al., 2014).

Even though requirements prioritization approaches have mostly been adopted with respect to functional requirements (Svensson et al., 2011; Thakurta, 2013), several studies have shown the significance of non-functional requirements in software projects (Joerg Doerr, Kerkow, Koenig, Olsson, & Suzuki, 2005; Glinz, 2007), and not correctly taking non-functional requirements into consideration is identified as one of the ten biggest risks in requirements engineering (Lawrence, Wiegers, & Ebert, 2001). Therefore, non-functional requirements need to be considered throughout the first phase of the software development process (i.e. requirements engineering phase). As such, it has been widely acknowledged that the achievement of non-functional requirements along with functional requirements is critical to the success of a software system (Berntsson Svensson, Olsson, & Regnell, 2013; Cysneiros & Sampaio do Prado Leite, 2004; Svensson et al., 2012). However, there are some differences between functional and non-functional requirements (Berander & Andrews, 2005; Lauesen, 2002):

- Functional requirements generally depend on particular functions whereas non-functional requirements generally have an effect on various functions (from a set of functions to the entire software system).
- Non-functional requirements are qualities that the functions or system should have, taking into the account that non-functional requirements are basically ineffective without functional requirements.
- From the implementation's point of view, functional requirements either operate or not, though non-functional requirements usually have a sliding value scale of good and bad.

- In many cases, non-functional requirements are usually in conflict with each other, hence trade-offs among this kind of requirements should be done.

The aforementioned differences of functional and non-functional requirements imply that non-functional requirements are more critical than functional requirements. Users of a system might be able to work with a function that is not capable of meeting their real expectations. However, failure of satisfying a non-functional requirement might make the whole system unworkable. In essence, a non-functional requirement presents a cross-cutting concern aspect that may have an effect on several functions or on the whole system. For instance, a non-functional requirement such as usability should be addressed through implementing some specific functions that enhance the capability of software system to be understood, learned, and used by its intended users. A non-functional requirement such as system's availability may affect the whole system. Considering the example of ATM software system, suppose that there is one functional requirement regarding *withdraw cash*. This functional requirement might be affected by several non-functional requirements such as *usability, performance* and *reliability*:

"*Usability*: Bank customer shall be able to *withdraw cash easily*."

"*Performance*: Bank customer shall be able to *withdraw cash* in *less than 3 seconds*."

"*Reliability* of *withdraw cash* must be high."

All the above explanations lead to the motivation to concentrate on both functional and non-functional requirements in this study and propose an approach which considers these two types of requirements (i.e. functional and non-functional) simultaneously during the prioritization process.

1.3 Motivation and Problem Statement

As discussed in the preceding sections, in almost every software project, budgetary deadlines and time-to-market constraints force practitioners to carefully prioritize

requirements and distinguish the high-priority requirements from the low-priority ones (Duan et al., 2009). So, requirement prioritization has been recognized as a critical but challenging activity in software product development (Svensson et al., 2012).

The requirements engineering community has classified the requirements of a software system into two main categories: functional requirements and non-functional requirements (Baskaran, 2014; Chung & do Prado Leite, 2009). The definition for a functional requirement specifies what the system should do; "A functional requirement specifies a function that a system or component must be able to perform." On the other hand, the definition for a non-functional requirement specifies how the system should behave; "A non-functional requirement is a statement of how a system must behave, it is a constraint upon the system behaviour." It has been widely acknowledged that a quality attribute such as reliability, modifiability, performance, or usability is a non-functional requirement of a software system (Capilla, Babar, & Pastor, 2012; Laplante, 2013).

Although, functional and non-functional requirements are very different, they have a serious impact on each other (Berander & Andrews, 2005). Considering merely functional requirements during the requirements prioritization process may lead to failure in the final product or at least may result in a poor-quality system. Hence, prioritizing these two types of requirements entirely together or separately might not be the best solution. For example, if there is one functional requirement about a specific function and one non-functional requirement regarding availability, it could be hard to prioritize between them since they are not at the same abstraction level. So, it is not an efficient way to prioritize both types of requirements together. In such cases, one may decide to prioritize them separately. Clearly, it is also not a good choice since both types of requirements have an impression on each other.

Over the recent past years, a substantial amount of research effort in the software engineering community has been dedicated into proposing various techniques, methods, and approaches to perform requirements prioritization. Although these approaches have contributed a lot to software development, an in-depth study of current research studies implies that:

1. The existing approaches can be used with both types of requirements separately.
2. Most of the current approaches have mostly been adopted with respect to functional requirements.
3. Considering both functional and non-functional requirements has received less attention in these works.

Hence, to support the advancement in this area and the maturity of requirements prioritization, a new approach needs to be developed with the purpose of addressing both functional and non-functional requirements during the prioritization stage. This research is primarily concerned with providing such an approach.

1.4 Research Scope

The box shaded in grey colour in Figure 1.1 shows the general focus of the current thesis which is actually the area of requirements prioritization. The figure also uses a hierarchy to demonstrate the role of requirements prioritization in the context of requirements engineering in particular and in the domain of software engineering in general. It should be noted that the presented hierarchy for showing the software development process is adapted by (Paetsch, Eberlein, & Maurer, 2003).

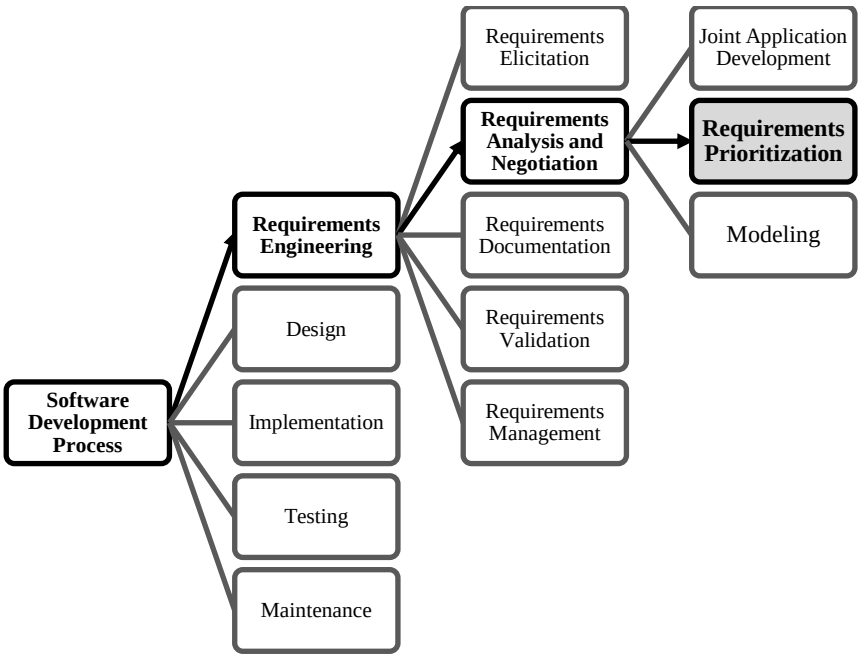


Figure 1.1: The scope of the current thesis

This research aims to introduce an approach by which practitioners are able to prioritize both functional and non-functional requirements simultaneously in an integrated manner. This approach investigates the dependencies between functional and non-functional requirements to perform the prioritization task. However, this study does not investigate requirements dependencies which might exist among individual functional requirements or non-functional requirements. In other words, this approach assumes that requirements (whether functional or non-functional) are independent. Nevertheless, in a case where some dependent functional or non-functional requirements need to be considered throughout the prioritization process, the remedy is to group those dependent requirements into a single requirement and then perform the prioritization task.

1.5 Research Objectives

The main purpose of this research is to provide the practitioners and researchers with an approach which is capable of integrating the prioritization of functional and non-

functional requirements simultaneously. To achieve the desired goal, the objectives of this research are determined as follows:

- **RO1**: To identify the current requirements prioritization approaches as well as several empirical evaluations of these approaches.
- **RO2**: To propose an approach by which integrating the process of prioritizing functional and non-functional requirements could be performed.
- **RO3**: To empirically evaluate the effectiveness of the proposed approach, through controlled experiments, in terms of time needed for performing the prioritization task, accuracy of the produced results and ease of use when compared to two similar state-of-the-art approaches.

1.6 Research Questions

The six research questions which form the basis of conducting this research are formulated as follows:

- **RQ1**: What are the current approaches used for requirements prioritization?
- **RQ2**: What are the descriptions and limitations of current requirements prioritization approaches?
- **RQ3**: What procedure should a software engineer follow to integrate the process of prioritizing functional and non-functional requirements?
- **RQ4**: How to perform the prioritization task within a reasonable amount of time?
- **RQ5**: How easily the prioritization task can be performed?
- **RQ6**: How accurately the prioritization task can produce the prioritized lists of functional requirements and non-functional requirements?

1.7 Research Approach

The main purpose of conducting this research is to provide researchers and/or practitioners with an approach which is capable of integrating the process of prioritizing functional and non-functional requirements simultaneously. To achieve the desired goal, three successive phases need to be accomplished throughout the research as illustrated in Figure 1.2. They are defined as follows: *preparation phase*, *approach development and validation phase*, and *evaluation phase*.

In the initial phase of conducting this research, called *preparation phase*, a detailed review of the literature is performed on the area of software requirements prioritization. This includes several studies that investigate different requirements prioritization approaches proposed in recent years to perform the requirements prioritization process. Besides this, a varied set of empirical evaluations of existing prioritization approaches, which were carried out by other researchers, are reviewed.

Three main activities which are conducted throughout the *approach development and validation phase* of this research aim to propose an approach for integrating the prioritization of functional and non-functional requirements, as well as validate the proposed approach through a case study, and through mathematical theory.

During the last phase of this research, i.e. *evaluation phase*, the effectiveness of the proposed approach is evaluated through conducting two controlled experiments. In both controlled experiments, the proposed approach is compared with two state-of-the-art alternatives in terms of actual time-consumption, accuracy of results, and ease of use.

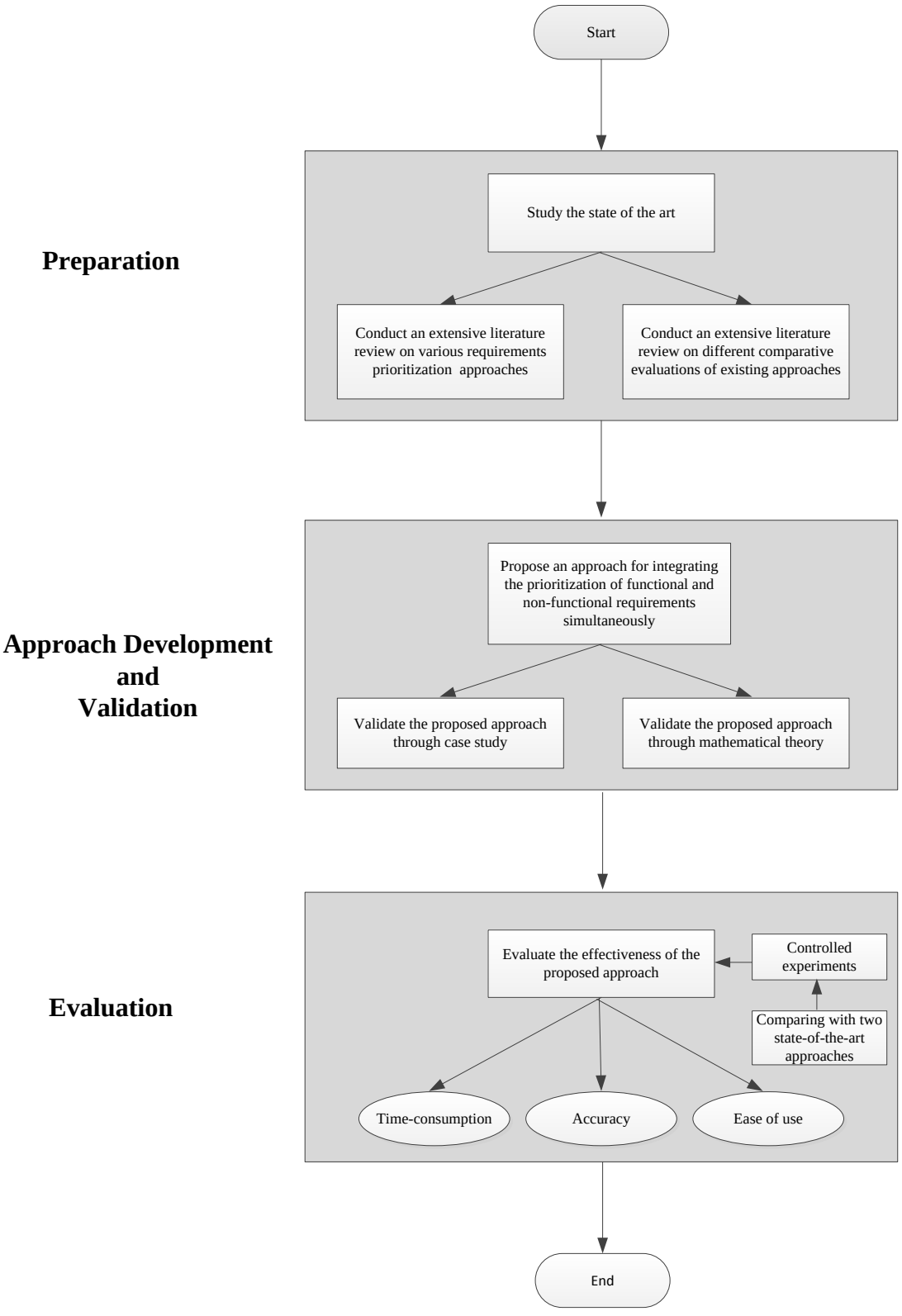


Figure 1.2: Flow chart of the research approach

1.8 Significance of the Research

The significance of this research is summarized as follows:

- To provide researchers with a body of knowledge of software requirements prioritization from two main perspectives: approaches as well as empirical studies.
- To provide practitioners with an approach that considers both functional and non-functional requirements simultaneously during the prioritization process.
- To assist researchers/practitioners in reducing the required time to perform the prioritization task.
- To help researchers/practitioners in improving the accuracy of the results provided by the proposed approach as well as enhancing the ease of use of the approach.

1.9 Summary

This chapter presented the primary elements of this thesis with the aim of providing insights into the research problem that this thesis addresses. The chapter also outlined the main objectives of this research along with the related research questions. Furthermore, it provided a brief research methodology that needs to be followed to achieve the desired objectives. It also presented the scope and significance of this research.

The remainder of this thesis is structured as follows. Chapter 2 of this thesis aims at presenting a thorough study on the current body of knowledge in the context of requirements prioritization. It introduces some basic concepts in the area of requirements prioritization, current requirements prioritization approaches, and empirical evaluations of existing prioritization approaches. Chapter 3 provides a

comprehensive description of the research methodology carried out in this research. Chapter 4 introduces the approach that is proposed in this research for integrating the process of prioritizing functional and non-functional requirements simultaneously. Chapter 5 describes the validation of the proposed approach through a case study as well as mathematical theory. Chapter 6 presents a thorough explanation of the two controlled experiments carried out in this research in order to evaluate the effectiveness of the proposed approach in an empirical manner. Finally, Chapter 7 describes the main conclusions of this research and outlines suggestions for future research.

CHAPTER 2: LITERATURE REVIEW

This chapter aims to present a thorough study on the current body of knowledge in the context of requirements prioritization. The basic concepts in the context of requirements prioritization are introduced first in order to express how this domain relates to the other activities and concepts in the area of software engineering, and also to demonstrate the definition and different aspects of requirements prioritization. Then, the current requirements prioritization approaches in the literature are explained and reviewed comprehensively to show how they prioritize requirements. This could be helpful to find out the research gap of this study. At the end of the chapter, some empirical evaluations of existing prioritization approaches, which were carried out by other researchers, are presented and reviewed to provide some guidelines on how to evaluate a given prioritization approach.

2.1 Basic Concepts in the Context of Requirements Prioritization

This section aims to provide fundamental concepts related to the domain of requirements prioritization. Initially, it presents the role of requirements prioritization in the area of requirements engineering. Afterwards, definitions for the terms *requirement*, *functional requirement*, and *non-functional requirement* are provided. Finally, the definition and aspects of requirements prioritization along with different types of product developments in the requirements prioritization process are discussed.

2.1.1 The Role of Requirements Prioritization in Requirements Engineering

A software development life cycle typically consists of five phases such as requirements engineering, designing, implementation, testing, and maintenance (Royce,

1970). Requirements engineering as the leading phase of the software development process is the phase in which requirements prioritization should be performed.

Many authors believe that the quality of a software product is mostly assessed by measuring its ability to satisfy different stakeholders' needs and expectations (Babar, Ramzan, & Ghayyur, 2011; J. Doerr, Hartkopf, Kerkow, Landmann, & Amthor, 2007; Sher, Jawawi, Mohamad, & Babar, 2014). One of the main challenges throughout the software development process is to find out the right needs of stakeholders and convert them into software requirements. Hence, managing the software requirements process in a solid, systematic and rigorous way could help practitioners to release high-quality products. Requirements engineering and its underlying activities play a critical role toward the success of a software project (Fellir, Nafil, & Touahni, 2014; Hofmann & Lehner, 2001).

Requirements engineering is not only engineering but actually it is a sort of multi-disciplinary strategy that covers some other disciplines and areas particularly the social and human sciences (Khan, 2006). In fact, it focuses on the social and cognitive sciences to be able to make use of its theoretical and practical background, knowledge, and methods for requirements elicitation, analysis, documentation, management and modelling. Many disciplines that have been applied in the context of requirements engineering are from sociology, anthropology, cognitive psychology, philosophy, human psychology and linguistics. These are generally well-organized and disciplined domains which have great contribution towards the improvements in the area of requirements engineering (Nuseibeh & Easterbrook, 2000). The terms *requirements* and *engineering* were initially joined together as a single phrase by Alford when he was developing the Software Requirements Engineering Method (SREM) (Alford, 1977). Requirements engineering was originally applied to information systems.

For a large number of software projects, it has been reported that the failure of software system is not due to the bugs found in the source code but rather because of the poorly-defined software requirements (Daneva et al., 2014). In other words, the failure of software project is mostly related to the problems within the requirements engineering phase. For example, in a research study conducted for a US Air Force project, it has been claimed that more than 40% of detected bugs were the results of errors in the requirements (Martens, 2011). The cost of discovering and fixing the requirements errors is a person-intensive and expensive job (Abirami, Shankari, Akshaya, & Sithika, 2015). Therefore, well-structured requirements engineering process can improve the success probability of any software system.

The requirements engineering process includes a structured set of activities which need to be done in order to discover, validate, and maintain a systems requirements document (Pressman, 2010). In (Gunda, 2008), the author introduced some general activities which form the requirements engineering process. As Figure 2.1 shows, these activities include requirements elicitation, requirements analysis and negotiation, requirements documentation, requirements validation, and requirements management. In the following paragraph, these activities are explained briefly.

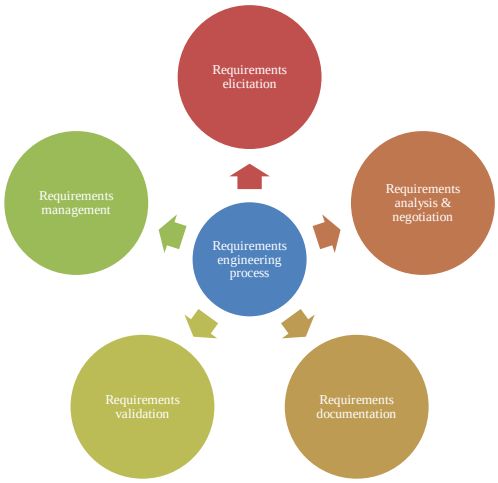


Figure 2.1: The activities of requirements engineering process

Requirements elicitation involves the identification of stakeholders' needs in order to transform them into software requirements (Teixeira, Saavedra, Ferreira, Simões, & Santos, 2014). Various techniques have been introduced for eliciting the software requirements (Rehman, Khan, & Riaz, 2013). Examples of these techniques include interview, use case/scenario, observation and social analysis, focus group, brainstorming, ethnography, and goal-based techniques (Anwar & Razali, 2014). During the requirements analysis and negotiation, each requirement is inspected to check its necessity, completeness, consistency, and feasibility (Pohl, 2010). The most common techniques used for requirements analysis are Joint Application Development (JAD) sessions, requirements prioritization, and requirements modelling. The goal of documenting the requirements is to provide a common basis for communication between stakeholders and developers. A well-defined requirements document should be unambiguous, complete, correct, understandable, consistent, concise, and feasible. Requirements validation is the process of checking the requirements to make sure that they are defined correctly as well as to ensure that they are the real needs of stakeholders. Requirements management involves with all activities concerned with change and version control, requirements traceability, and requirements status checking (Pohl, 2010).

However, there is not a unique requirements engineering process which is applicable for all organization (Aurum & Wohlin, 2003). In other words, different organizations or companies may apply a different requirements engineering process for developing their software products (e.g. requirements engineering process for web-based applications is not the same as requirements engineering process for safety critical systems or embedded systems) (Khan, 2006). The variety in the requirements engineering process is considered as an acceptable issue since there exists no ideal requirements engineering process. Requirements engineering may be impressed by many factors such as

application domain (Kotonya & Sommerville, 1998), organizational culture, disciplinary involvement, technical maturity, and market decisions as well as design decisions of the individuals (Gurp, Bosch, & Svahnberg, 2001). Furthermore, other factors like system acquisition, commercial, legal, and contractual issues can affect the requirements engineering process (Kotonya & Sommerville, 1998). The end result of requirements engineering process is the software requirements specification (SRS).

2.1.2 Requirement

In the context of requirements prioritization, the object to be prioritized by decision makers is a collection of requirements. This implies that the concept of requirement is fundamental in the area of requirements prioritization. Hence, before going through the concept of requirements prioritization, it would be useful to have an overview on the definitions provided in the literature for the term *requirement*. According to (Sommerville & Sawyer, 1997), the term *requirement* is referred to as "*What a system should do*". However, different authors and researchers offered various definitions from different points of view for the term *requirement*. These definitions are summarized and presented in Table 2.1.

Requirements can be classified into different categories. Table 2.2 illustrates some different categories of requirements. Based on the purpose of requirements prioritization, prioritization techniques or approaches can be applied on different classes of requirements.

Table 2.1: Overview of definitions of the term "*Requirement"*

Reference	Definition
(Abbott, 1986)	*"Any function, constraint, or other property that must be provided, met, or satisfied to fill the needs of the system's intended user(s)."*
(IEEE, 1990)	*"(1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2)."*
(A.M. Davis, 1993)	*"A user need or a necessary feature, function, or attribute of a system that can be sensed from a position external to that system."*
(S. IEEE, 1998)	*"A well-formed requirement is a statement of system functionality (a capability) that must be met or possessed by a system to satisfy a customer's need or to achieve a customer's objective, and that is qualified by measurable conditions and bounded by constraints."*
(Lethbridge & Laganiere, 2001)	*"A statement about the proposed system that all stakeholders agree must be made true in order for the user's problem to be adequately solved."*
(Sommerville, 2007)	*"The requirements for a system are the descriptions of the services provided by the system and its operational constraints. These requirements reflect the needs of customers for a system that helps solve some problem such as controlling a device, placing an order or finding information."*

Due to the fact that the quality of any software system mainly depends on fulfilling functional and non-functional requirements, these two types of requirements have received more attention in this thesis. Therefore, these two types of requirements (i.e. functional and non-functional requirements) are explained below in more detail. In addition, the differences between these two types of requirements are discussed.

A functional requirement describes a functional behaviour that a system or system component should be able to perform. In other words, functional requirements specify what the system should do; an activity that the system must do to provide its users with the required functionality. Functional requirements are also referred to as behavioural or operational requirements (Davis, 1993). Table 2.3 shows a review on different definitions of functional requirements that have been presented in the literature. It is

quite obvious that these definitions are in line with the definition that has been already presented in this thesis.

Table 2.2: Different categories of requirements (Aurum & Wohlin, 2005)

Requirements Classification
• *Functional requirements*---what the system should do • *Non-functional requirements*---constraints on the types of solutions that will meet functional requirements (e.g. performance, security, reliability) • *Goal level requirements*---related to business goal • *Domain level requirements*---related to problem area • *Product level requirements*---related to the product • *Design level requirements*---what to build • *Primary requirements*---elicited from stakeholders • *Derived requirements*---derived from primary requirements Some other classifications: • *Business requirements* versus *technical requirements* • *Product requirements* versus *process requirements*---i.e. business needs versus how people will interact with the system • *Role based requirements*, e.g. customer requirements, user requirements, system requirements, and security requirements.

Table 2.3: Different definitions of functional requirements

Reference	**Definition**
(IEEE, 1990)	*"A function that a system must be able to perform."*
(Robertson & Robertson, 2012)	*"What the product must do."*
(Sommerville, 2004)	*"What the system should do."*
(K. E. Wiegers, 2003)	*"A statement of a piece of required functionality or a behaviour that a system will exhibit under specific conditions."*
(Jacobson, Booch, & Rumbaugh, 1999)	*"A requirement that specifies an action that a system must be able to perform, without considering physical constraints; a requirement that specifies input/output behaviour of a system."*
(Anton, 1997)	*"Describe the behavioural aspects of a system."*

On the other hand, the other type of requirements, called non-functional requirements, express how good a software system must work. It has been widely acknowledged that a quality attribute such as reliability, modifiability, performance, or usability is a non-functional requirement of a software system (Capilla et al., 2012; Chung & do Prado Leite, 2009; Laplante, 2013). That is why non-functional requirements are sometime called quality attributes or quality requirements. Literature offers several definitions proposed by different scholars. These definitions are provided in Table 2.4. Referring to the definitions, it is clear that non-functional requirements are relating to properties or qualities that the software system must have when performing one or some functions

Table 2.4: Different definitions of non-functional requirements

Reference	**Definition**
(Anton, 1997)	*"Describe the non-behavioural aspects of a system, capturing the properties and constraints under which a system must operate."*
(A.M. Davis, 1993)	*"The required overall attributes of the system, including portability, reliability, eefficiency, human engineering, testability, understand- ability, and modifiability."*
(Robertson & Robertson, 2012)	*"A property, or quality, that the product must have, such as an appearance, or a speed or accuracy property."*
(K. E. Wiegers, 2003)	*"A description of a property or characteristic that a software system must exhibit or a constraint that it must respect, other than an observable system behaviour."*
(Jacobson et al., 1999)	*"A requirement that specifies system properties, such as environmental and implementation constraints, performance, platform dependencies, maintainability, extensibility, and reliability. A requirement that specifies physical constraints on a functional requirement."*
(Mylopoulos, Chung, & Nixon, 1992)	*"Global requirements on its development or operational cost, performance, reliability, maintainability, portability, and robustness."*

Although functional and non-functional requirements are two correlated concepts in software development process, there are also some differences between functional and non-functional requirements. These differences are listed below (Berander & Andrews, 2005; Lauesen, 2002).

- Functional requirements generally depend on particular functions whereas non-functional requirements generally have an effect on various functions (from a set of functions to the entire software system).
- Non-functional requirements are qualities that the functions or system should have, taking into the account that non-functional requirements are basically ineffective without functional requirements.
- From implementation point of view, functional requirements either operate or not, though non-functional requirements usually have a sliding value scale of good and bad.
- In many cases, non-functional requirements are usually in conflict with each other so that trade-offs among these kind of requirements should be done.

2.1.3 Requirements Prioritization

Requirements prioritization has not been only recognized as one of the major activities of requirements engineering process but also has been known as one of the most significant decision-making processes during the software development process. In software engineering community, projects are usually faced with time to market constraints, budget deadlines, restricted technology, and limited human power. Therefore, it would be challenging for requirements engineers to determine which requirements may provide a higher degree of satisfaction for stakeholders and should be addressed first. To tackle this concern, it is essential to address the high-priority requirements before considering the low-priority ones. Requirements prioritization facilitates requirements engineers with a solution to choose the most valuable requirements for a software system or a release planning.

By reviewing the literature, it has been discovered that different scholars proposed different definitions for the term *requirements prioritization*. According to (Sommerville, 2007), requirements prioritization has been defined as an activity in which the most important requirements can be identified. On the other hand, another definition presented by (Firesmith, 2004) implies that requirements prioritization is a process by which the implementation order of requirements can be determined. By analyzing the definitions proposed in the two studies, it can be found that the first definition is centralized on the importance of requirements to stakeholders. However, the second definition focuses on implementation order, relying on the point that in some situations there might be dependencies among requirements which make the implementation order of requirements different from the importance order of requirements to stakeholders. This thesis adopts Somerville's definition as the meaning of the requirements prioritization.

According to (Berander & Andrews, 2005), requirements prioritization provides many benefits and advantages including:

- It improves user involvement by engaging stakeholders in the process of identifying the most important requirements of a project.
- It facilitates stakeholders to assign resources according to the priorities of requirements.
- It allows stakeholders to determine the core requirements of the system.
- It helps in choosing an optimal ordered list of requirements which need to be implemented in consecutive releases.
- It provides support to make trade-off between the desired project scope and conflicting constraints such as resources, budget, schedule, time to market, and quality.

- It helps in counter-balancing the business profit of each requirement against its implementation cost.
- It aids in balancing implications of requirements on the software architecture and future evolution of the product and its related cost.
- It assists to choose only a subset of the requirements that meet the overall stakeholders' satisfaction.
- It helps in the estimation of predicted stakeholder's satisfaction.
- It assists to get a technical advantage and optimize market opportunity.
- It facilitates the development organization to minimize rework and schedule slippage and thereby improve stability.
- It helps to manage inconsistent requirements, concentrate on the negotiation process, and solve arguments among stakeholders.
- It allows determining the relative importance of each requirement to deliver the maximum importance at the minimum cost.

All the factors stated above represent the importance of requirements prioritization. In addition to the benefits mentioned above, some well-known authors in the domain of software engineering have argued regarding the significance of prioritizing requirements. For instance, Frederick P. Brooks mentioned (Brooks Jr, 1995), "*The hardest single part of building a software system is deciding precisely what to build…No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later*". Another famous author in the field of software engineering, Ed Yourdon, has recognized requirements prioritization as an extremely important issue (Yourdon, 1997). Sharif *et al*. believes that requirements prioritization is a key but often neglected issue in the requirements engineering research (Sharif, Zafar, & Zyad, 2014).

A number of researchers consider requirements prioritization as one of the most complicated activities throughout the requirements engineering process (Berander & Andrews, 2005). They claim that only a few number of software companies have utilized systematic, practical, and efficient methods for performing the prioritization task. At the same time, some authors believe that requirements prioritization has a medium level of complexity while some others consider it as an easy process during the software development process. Nonetheless, it has to be said that all of the researchers has identified requirements prioritization as a fundamental activity toward the success of any software project. Hence, in order to develop a cost-effective software system, it is crucial to prioritize requirements first and then proceed to develop the software according to those requirements.

2.1.3.1 Aspects of Requirements Prioritization

Variety of aspects can be taken into account when prioritizing requirements. Aspect can be defined as a property or attribute of requirements that can be used to prioritize requirements (Berander, 2004a). Some alternative terms have been used in the literature such as *factor* (Henry & Henry, 1993), *criteria* (Martinez, Pazos Arias, & Vilas, 2005), *element* (Egyed, 2003), and *parameter* (Tran & Sherif, 1995) to bring the same meaning for the term *aspect*. Some popular aspects that can be used to prioritize requirements include risk, time, cost, penalty, importance, dependency, volatility, abstraction level, strategic benefit, available resources, and market value. Prioritizing requirements based on only one aspect seems to be easier compared to the situations that multiple aspects need to be considered for prioritization (Khan, 2006). For example, if the goal is to prioritize requirements according to their importance to stakeholders, it is a straightforward task to decide which requirement is the most desirable. But, once taking other aspects into account such as cost, stakeholders need to change their minds to

prioritize requirements. This may result in changing the high-priority requirements into low-priority requirements if they are not cost-effective.

Authors of (Ruhe, Eberlein, & Pfahl, 2003) believe that, in many cases, aspects are dependent on each other, and might have interaction with each other (e.g. high quality might need high cost). Hence, modification in a single aspect might cause a modification in the other aspect. Due to the fact that every aspect could possibly have an impact on the success level of the ultimate software product, it is crucial to take several aspects into consideration to be able to enhance the success degree of the ultimate product. Although requirements can be prioritized based on multiple aspects, it is normally not useful and advisable to consider all the aspects (Berander & Andrews, 2005). The question on what aspects are essential to be considered for a prioritization problem is mainly dependent on the specific project.

The most significant aspects which might be suitable for any prioritization problem are explained below.

- *Importance*: Stakeholders can prioritize requirements according to their importance for the software system to find out which requirement is the most important one. Nevertheless, the term *importance* could be considered as a multi-dimensional principle (Ma, 2009), i.e. it can possess various interpretations from different points of view. For example, *importance* can be interpreted as urgency of implementation, importance of a requirement for a software system, or strategic importance of the product architecture. Therefore, prior to begin the prioritization process, it is necessary to clarify the meaning of *importance* to avoid misunderstanding among stakeholders.

- *Cost*: Cost is usually specified with the amount of money outlaid on implementing the requirements (Ma, 2009). Factors which may affect the cost include the ability to reuse existing code, the complexity of requirements, the extra resources required to implement the requirements, and the amount required for documentation and testing. Cost is specifically measured in terms of staff hours due to the fact that the major cost of developing any software system is associated with the number of spent hours.
- *Time*: Time can be defined as the period which is needed to successfully implement the requirements. In addition, time can be affected by other factors such as training time, level of parallelism in software development process, time required to develop infrastructure and support industry standards, and etc.
- *Risk*: Each software project encompasses a particular amount of risk. Risk can be prioritized to find which requirement contains the lowest risk. Risk management is generally rooted in project management where it has been utilized to deal with internal as well as external risks. Internal risks include market and technical risk whereas external risks involve suppliers and regulations risks. In particular, risk management could also be exploited when planning requirements into releases and products by determining risks that may lead to problems throughout the development process. Some instances of these risks are schedule risks, process risks, and performance risks. Risk degree of a software project can possibly be assessed by estimating the risk level of every single requirement .
- *Penalty*: Penalty is a major aspect which needs to be estimated in terms of how much requires to be spent if a requirement is not met (Ma, 2009). It is not correct to use penalty as the opposite meaning of importance. In some situations,

a requirement may have a low level of importance, but ignoring to meet that requirement might lead to a high penalty.

- *Volatility*: In some cases, volatility of requirements has been considered as a risk element and is also occasionally treated as portion of risk aspect (Berander & Andrews, 2005). Some authors believe that volatility needs to be evaluated independently and volatility of requirements must be taken into the consideration independently within the prioritization process (Lauesen, 2002). There are some different reasons that may result in requirement volatility. Examples of these reasons include user changes, legislative changes, business requirements changes, and market changes (Ruhe et al., 2003). Regardless of these reasons, volatile requirements could possibly influence the planning and stability of a software project and undoubtedly boost the costs. This is due to the reason that any changes during the development process will increase the cost of a software project.
- *Other aspects*: A collection of typical aspects which have been considered significant in the literature presented above. However, this collection is not a thorough list in which other aspects also can be included such as release theme, competence/resources, competitors, strategic benefit, and financial benefit. It has been recommended that stakeholders provide a comprehensive list of aspects before starting the process of decision making in a software company (Berander & Andrews, 2005). In addition, it is crucial for stakeholders to reach a consensus on the meaning of aspects and requirements as well because several studies have indicated that it is quite difficult to interpret the outcomes if there is no guideline regarding the meaning of aspects and requirements (Lehtola, Kauppinen, & Kujala, 2004).

- *Aspects combination*: In real cases, it is preferable to combine different aspects of a requirement before making decision on the final priority of that requirement. For instance, the Planning Game approach considers three aspects such as importance, cost, and risks of requirements to determine the final prioritized list of requirements (Beck, 2000). In cost-value approach (J. Karlsson & Ryan, 1997), requirements are prioritized based on importance to users and cost of implementation to select those requirements that possess the highest importance and lowest price. Moreover, some other studies declare that requirements should be prioritized based on importance and volatility aspects whereas others express that dependencies should be also considered (Carlshamre, Sandahl, Lindvall, Regnell, & Natt och Dag, 2001; Lauesen, 2002). The Wieger's approach (K. E. Wiegers, 2003) uses four aspects including benefit, penalty, cost, and risk to determine the relative value of each requirement. This approach allows different weights for different aspects in different situations in order to find out the most valuable requirement. Many other combinations of aspects can be considered when prioritizing requirements. Which aspects to use for combination is basically dependent on the specific situation, and it is really critical to determine the possible aspects that can be combined together efficiently to achieve the desired outcomes.

2.1.3.2 Different Types of Product Developments in the Prioritization Process

Requirements prioritization can be applied on two types of software development processes: bespoke software development and market-driven software development (Baskaran, 2014; Ma, 2009). For a bespoke project, only one or a few number of stakeholders may take part in the prioritization process, whereas in market-driven software development, every person in the entire world has the potential to be

considered as a stakeholder of project (Regnell & Brinkkemper, 2005). Literature distinguishes the differences between bespoke and market-driven software development which may inspire requirements prioritization (Carlshamre, 2001). These differences are given in Table 2.5.

Table 2.5: A comparison between bespoke and market-driven software development (Carlshamre, 2001)

Facet	**Bespoke development**	**Market-driven development**
Main stakeholder	Customer organization	Development organization
Users	Known or identifiable	Unknown, may not exist until product is on market
Distance to users	Usually small	Usually large
Requirements conception	Elicited, analysed, validated	Invented (by market pull or technology push)
Lifecycle	One release, then maintenance	Several releases as long as there is a market demand
Specific RE issues	Elicitation, modelling, validation, conflict resolution	Steady stream of requirements, prioritization, cost estimating, release planning
Primary goal	Compliance to specification	Time-to-market
Measures of success	Satisfaction, acceptance	Sales, market share

According to Table 2.5, there is a great difference between bespoke development and market-driven development which put them in two extremes. Therefore, different software projects should follow different ways to prioritize requirements. However, in majority of real situations, projects development place somewhere between these two extremes. Different kinds of markets involve different types of customers. Generally, three types of customer situations may exist while developing a software product: one customer, several known customers, and mass-market (Berander & Andrews, 2005).

In case of one customer development, the goal is to develop a software product for only one customer, and thereby requirements need to be prioritized based on the viewpoint of only one person. Most of the existing software processes are developed

based on one customer presuming that this customer is accessible at the time of developing the product. However, one major concern in one customer situation is that, in some cases, the customer who participates in the prioritization process is different from the one who really works with the system. For example, it could be conflicting in requirements priorities if a customer who prioritizes the requirements would be an employer but the person who really uses the system would be an employee. In such case, it would be advisable to engage the end user in the prioritization process, since not involving him/her in the prioritization process may lead to reduction of product's usability.

Regarding several known customers development, the prioritization process is more difficult because different customers who may take part in the prioritization process may have different preferences and viewpoints. In this situation, the major target is to come up with a prioritized list of requirements which is acceptable by each single customer. Engaging the viewpoints of all stakeholders is necessary towards the success of the software product.

In case of mass-market development, all customers are not accessible throughout the prioritization process (Kuusela & Savolainen, 2000). In this case, different sources can be used to extract necessary information for the prioritization process (Jobber & Ellis-Chadwick, 2012). These sources include marketing research (e.g. surveys, focus groups), competitor intelligence (e.g. information about competitors' strategies, benchmarking competitors' products), marketing intelligence (e.g. information from sales force, scientists) and internal records (e.g. shipments, sales records). To perform marketing research, it is essential to focus on an appropriate test sample which can accurately represent the actual characteristics of the whole market segment. For

instance, when developing for small companies, it is not useful to engage large companies in the surveys or focus groups.

2.2 Requirements Prioritization Approaches

In the preceding section, it has been shown that requirements prioritization has been recognized as one of the most important decision-making process in the software development life cycle. This is also confirmed by a great amount of requirements prioritization approaches that have been proposed in recent years. In this regard, this section aims to provide a thorough overview on the various prioritization approaches found in the literature. Before going through a detailed description of the current prioritization approaches, it should be highlighted that the term *approach* in this context refers to a general term for *technique* and *method* and they have been used to bring the same meaning.

Existing requirements prioritization approaches can be split into four different categories such as *ratio* scale approaches, *ordinal* scale approaches, *nominal* scale approaches, and *interval* scale approaches. This classification is derived from (Aasem, Ramzan, & Jaffar, 2010; Achimugu et al., 2014; L. Karlsson, Höst, & Regnell, 2006; Voola & Babu, 2012) and depending on the order relation, the approaches ultimately produced. Table 2.6 classifies the findings according to those scales.

Regarding *ratio* scale approaches, requirements are ranked according to their relative weights (Ma, 2009). The ratio scale is widely recognized to be the most desirable among all the other scales since it is capable of ordering, specifying relative ratios and distances among requirements. Examples of ratio scale approaches are AHP, hundred dollar method, minimal spanning tree, hierarchy AHP, cost-value approach, VOP, TOPSIS, case-based ranking, IGA, and EVOLVE. Using ratio scale prioritization

approaches provides the opportunity to accomplish all possible kinds of statistical calculations such as coefficient of variations, harmonic mean, geometric mean, and also exploit algorithms during the prioritization process (Aasem et al., 2010; Achimugu et al., 2014).

Table 2.6: Classification of requirements prioritization approaches

Requirements prioritization approaches			
Ratio scale	*Ordinal scale*	*Nominal scale*	*Interval scale*
AHP	Simple ranking	Numerical assignment	RUPA
Hundred dollar method	Bubble sort	Top-ten requirements	
Minimal spanning tree Hierarchy	Binary search tree	MoSCoW	
AHP	Planning Game	Requirements triage	
Cost-value approach	QFD		
VOP	CBPA		
Wieger's method			
IGA			
EVOLVE			
TOPSIS-based approach			
Case-based ranking			
Value-based intelligent requirements prioritization			
Fuzzy AHP			
Thakurta's approach			
Interrelationship-based approach			
HAM			

Ordinal scale approaches generate an ordered list of requirements (Ma, 2009). In other words, ordinal scale could be utilized to enhance the efficiency of nominal scale due to the reason that it provides knowledge regarding the ranking of requirements. Examples of these approaches include: simple ranking, bubble sort, binary search tree, B-tree, Planning Game, QFD, and correlation-based priority assessment framework. Using ordinal scale prioritization approaches provides the opportunity to compute the percentile and median of ordered requirements (Aasem et al., 2010).

For nominal scale approaches, requirements are allocated to distinct priority groups, taking into consideration that all requirements of each group have the same priority (Ma, 2009). These kinds of approaches are not able to show the importance degree of each requirement over the other requirement. Examples of these approaches are as

follows: numerical assignment, top-ten requirements, and MoSCoW. The nominal scale prioritization approaches allow computing chi square and mode of ordered requirements (Aasem et al., 2010).

Interval scale approaches include details regarding how big are the intervals between the ordered set of requirements in order to increase the calculation of the disparity that might exist between requirements (Achimugu et al., 2014). The interval scale keeps the order in the same way as ordinal scale does. RUPA is considered as an example of interval scale approach. The interval scale prioritization approaches allow calculating regression, correlation, standard deviation, mean, and analysis of variance (Aasem et al., 2010).

2.2.1 Ratio Scale Approaches

Analytical Hierarchy Process (AHP)

The Analytic Hierarchy Process (AHP), first proposed by Saaty (Saaty, 1980), has been recognized as a principled and organized multi-criteria decision-making method. AHP was initially applied to the domain of software engineering by Karlsson (J. Karlsson, 1996), and since then, it has been used to prioritize software requirements. The fundamental concept of AHP is to determine the priorities of all available requirements as well as their relative importance through comparing all possible pairs of requirements together. In practice, a decision maker who intends to use AHP should use a range of scales from 1 to 9 (where one indicates the same level of importance while nine expresses the most degree of importance) to decide which requirement is more important and to what extent.

To prioritize n requirements using AHP, the decision maker needs to perform $n \times (n-1)/2$ comparisons among requirements. So, in any prioritization case where

there would be a large number of requirements to prioritize, the number of comparisons would increase substantially. This issue has been considered as a major drawback associated with the AHP method. That is why some studies in the literature stated that AHP is not appropriate for prioritization of a large number of requirements (Lehtola & Kauppinen, 2004). In contrast, some researchers have attempted to figure out solutions to reduce the number of comparisons. As a result, some modifications of AHP method have been proposed with the aim of reducing the number of comparisons (Harker, 1987; Shen, Hoerl, & McConnell, 1992).

AHP has been recognized as the most cited prioritization technique among others (Achimugu et al., 2014). It is also capable of prioritizing requirements based on different aspects. In addition, it has been acknowledged that AHP is the most promising prioritization technique which produces trustworthy results (J. Karlsson, Wohlin, & Regnell, 1998). This is due to the fact that it is possible to calculate the consistency ratio when using AHP to improve the reliability of results. Further discussions regarding AHP is available in (J. Karlsson & Ryan, 1997; Saaty, 1980).

Hundred dollar method

Hundred dollar method is an uncomplicated prioritization method in which stakeholders receive 100 fictitious units and they are asked to allocate these units among candidate requirements (Berander & Andrews, 2005; Hatton, 2008). This method is also referred to as cumulative voting. The outcome of the prioritization process using hundred dollar method is shown in ratio scale. The fictitious units can be expressed in terms of different aspects such as cost of implementing requirements, time, importance, and so on. According to (Berander, 2004a), this method is considered complex in terms of sophistication, and fine in terms of granularity.

There would be some problems when a large of requirements need to be prioritized using this method (Berander & Andrews, 2005). For instance, if there are twenty candidate requirements, five units should be allocated to each requirement on the average. Regnell et al. encountered with this issue when they intended to prioritize 17 sets of requirements (Regnell, Höst, Och Dag, Beremark, & Hjelm, 2001). To solve the problem and have more flexibility in the prioritization process, they considered an imaginary amount of $100,000. So, this study indicated that in case of having a large number of requirements, the other amounts more than hundred units can be used to allow participants to have a straightforward prioritization. The other problem that might be happened when prioritizing many requirements using hundred dollar method is that an individual who works with the technique may make a mistake regarding the calculation of hundred points (Berander & Wohlin, 2004). This issue can be solved by facilitating the method's users with an automated tool which is capable of counting the number of remaining points (Berander & Andrews, 2005).

Using the Hundred dollar method, the prioritization process must be performed only one time on the same list of requirements since stakeholders may change their preferences in the second time especially when they do not find their desired requirements as high-priority requirements (Berander, 2004a). In such situation, the solution could be used to force stakeholders to allocate a limited amount of points for each requirement. However, this could be considered a risk because stakeholders are not able to prioritize requirements based on their actual preferences.

Minimal spanning tree

Minimal spanning tree is a prioritization approach which first proposed by (J. Karlsson et al., 1998). As explained before, AHP needs a great amount of pairwise comparisons to prioritize candidate requirements. So, AHP involves a large-scale

redundancy. Suppose that if requirement X is considered to be more important than requirement Y and requirement Y is also supposed to be more important than requirement Z, then it can be easily concluded that requirement X is more important than requirement Z. However, using AHP, the decision maker has to perform the pairwise comparison between requirement X and requirement Z. Although this redundancy assists to recognize the errors associated with stakeholder's judgments, it also causes scalability problems. Minimal spanning tree approach has been proposed with the aim of solving this problem.

The fundamental concept behind the Minimal spanning tree technique is that all the redundant pairwise comparisons of AHP method should not be performed (e.g. comparing requirement X to Z in the previously mentioned example) by relying on the assumption that the decision maker's judgement would be consistent. This would result in decreasing the number of pairwise comparisons from $n \times (n-1)/2$ needed by AHP to $(n-1)$ with Minimal spanning tree approach. The required number of comparisons can be represented by constructing a minimal spanning tree of requirements in a directed graph. This reduced number of comparisons between requirements is sufficient to compute the relative degree of importance between requirements. The Minimal spanning tree approach is considered a fast approach for prioritizing requirements due to the low number of comparisons needed by this approach. On the other hand, its capability to recognize inconsistent judgments is not high because redundancy has been eliminated.

Cost-value approach

The cost-value approach was proposed by (J. Karlsson & Ryan, 1997) to allow decision makers to give ranking to candidate requirements based on two aspects: the value of requirements to users and customers, and the implementation cost of

requirements for software developers. The evaluation of requirements based in value and cost is performed using AHP method. In fact, the outcome of this approach provides insights for software engineers to include or exclude some requirements in/from the first release of software product.

Using the cost-value approach involves five steps to prioritize software requirements. These steps are as follows:

1. Requirements engineers need to check the candidate requirements to make sure that they are defined completely and clearly.
2. Users and customers must utilize AHP method to estimate the relative value of candidate requirements.
3. Skilful software engineers should apply AHP method to determine the relative cost of implementing the candidate requirements.
4. A software engineer needs to apply AHP method to obtain the relative value as well as implementation cost of each candidate requirement, and then based on this information, draws a cost–value diagram. Example of this diagram is depicted in Figure 2.2.
5. The stakeholders exploit the cost-value diagram as a guideline for analysing requirements, and software managers use the information of the diagram to prioritize candidate requirements.

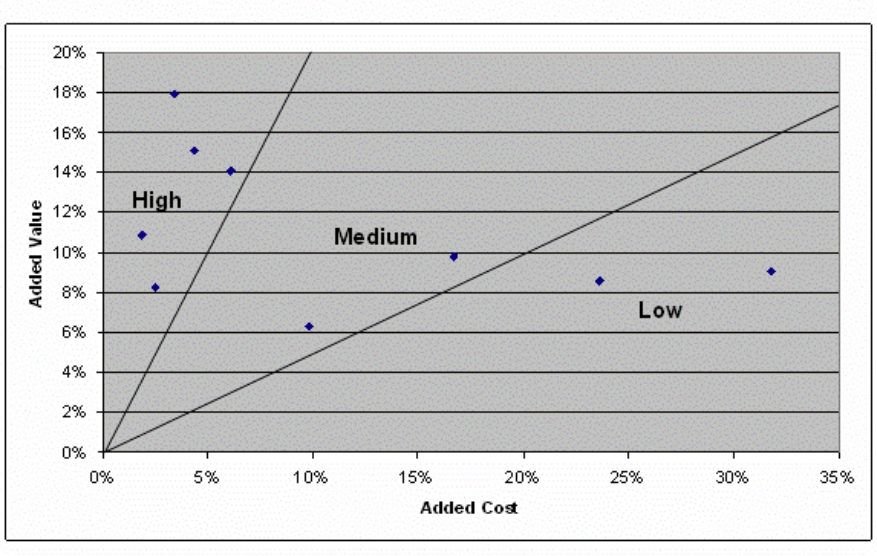


Figure 2.2: An example of cost-value diagram

Despite the fact that the cost-value approach can be considered as a robust and straightforward prioritization approach which provides valuable results to its users, there are still some drawbacks associated with this approach. In case of a large number of requirements, it would be a time-consuming approach which makes it tedious to use. In addition, the approach ignores the dependencies which may exist among requirements.

Hierarchy AHP

The Hierarchy AHP approach was introduced by (J. Karlsson et al., 1998) to overcome the scalability issues involved with the AHP method. Davis believes that in large-scale software projects, requirements are not organized in a flat structure (Alan M Davis, 1993). In other words, requirements tend to be placed in an organized hierarchy in which the more general requirements are put at the higher levels of the hierarchy while the more specific requirements are placed at the lower levels of the hierarchy. Hierarchy AHP approach utilizes the AHP method to prioritize requirements which are placed at the same level of hierarchy. This approach can improve the scalability of the AHP method by decreasing the number of comparisons needed for performing the

prioritization process. This is due to the reason that only requirements of the same level need to be compared together.

According to (J. Karlsson et al., 1998), using hierarchy AHP approach involves three stages explained below to prioritize requirements.

1. In *preparation* stage, all candidate requirements are organized in a hierarchy.
2. During *execution* stage, requirements that have been placed in each hierarchy are compared together using AHP scales (i.e. one to nine).
3. In *presentation* stage, AHP algorithm is applied at each hierarchy level to come up with a prioritized list of requirements for the related hierarchy level.

Value-Oriented Prioritization (VOP)

VOP provides a framework for requirements engineers to make decisions and prioritize requirements. VOP can be defined as a prioritization approach which prioritizes requirements based on their influence on particular business values that a company identifies (J. Azar, R. K. Smith, & D. Cordes, 2007). Figure 2.3 demonstrates the VOP framework.

VOP includes a two-step process including *establishing the framework* and *applying the framework* to prioritize requirements (J. Azar, R. Smith, & D. Cordes, 2007). The first step is called *establishing the framework* which involves with recognizing the core business values and then assigning weights to those business values. These weights are ranging from 1 (not important) to 10 (critical). Besides identifying and scoring the core business values, the framework also includes the identification and giving scores to the risk categories. The weight of risk is assigned in negative scale. VOP creates a decision

matrix using business core value as well as risk value. So, the outcome of the first step is a decision matrix. Then, in the second step (i.e. applying the framework), VOP calculates the final score associated with each candidate requirement. In practice, the final score of each requirement is calculated as the sum of its business values minus the sum of its risk values. The total core business value contribution is easily achieved as the product of each value's weight times the requirement's weight with respect to that core value. Besides, the sum of risk values of a requirement is the product of each risk's weight times the weight of that requirement with respect to that risk.

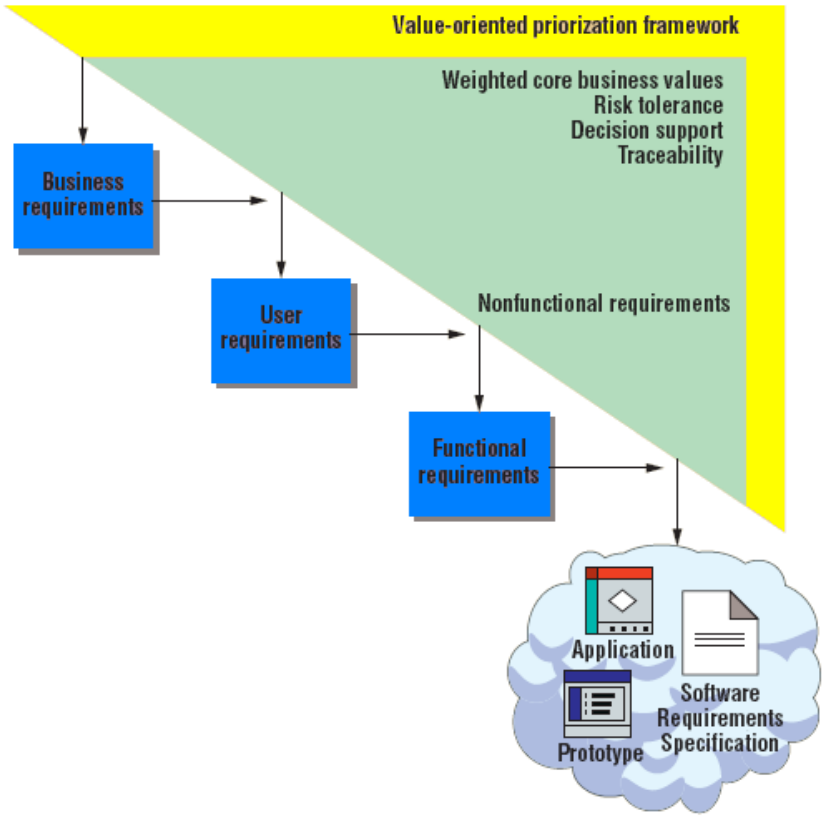


Figure 2.3: Value-oriented prioritization framework (Jim Azar et al., 2007)

Wieger's method

Wieger proposed a semi-quantitative analytical approach with the purpose of prioritizing requirements based on four aspects such as benefit, penalty, cost, and risk (K. E. Wiegers, 2003; K. Wiegers, 1999). However, it seems that this method is not

rigorous since it lacks of mathematical basis and it depends only on the estimation of stakeholders regarding the values of benefit, penalty, cost, and risk. Wieger's method is suitable for applying on small projects. Using this method, three main stakeholders need to participate in the prioritization process:

- *Project manager*, who manages the whole process.
- *Main customers*, who provides the values of benefit and penalty associated with each requirement.
- *Key developers*, who is in charge of providing the values of cost and risk associated with each requirement.

Wieger's method involves eight steps that need to be followed by a decision maker to prioritize requirements. These steps are as follows:

1. Make a list of requirements that need to be prioritized. These requirements should be at the same abstraction level.
2. Determine the benefit that every single requirement may provide to the customer. Benefit of each requirement is assessed by customers on a scale from 1 to 9 where one indicates the lowest benefit and nine represents the highest benefit.
3. Determine the penalty associated with each requirement if that requirement fails to fulfil. The penalty is also assessed by customers on a scale from 1 to 9.
4. Calculate the value of each requirement by adding the estimated benefit and penalty of that requirement.
5. Determine the implementation cost of each requirement by asking the developers to assign a value from 1 to 9.

6. Determine the risk level of each requirement by asking again the developers to assign a value from 1 to 9.
7. Calculate the final weight of each requirement using the following formula: $weight = value\%/(cost\% + risk\%)$.
8. Prioritize the candidate requirements in descending order using calculated weights.

Interactive genetic algorithm (IGA)

Tonella et al. proposed a search-based approach, called *Interactive Genetic Algorithm (IGA)*, for requirements prioritization (Tonella, Susi, & Palma, 2013). The IGA requirements prioritization approach is a kind of pairwise comparison method that gets the advantages of using a genetic algorithm to reduce the amount of pairwise comparisons that need to be elicited from the decision maker. It merely elicits those pairs of requirements which prevent the ambiguity of similarly ordered or even diversely ordered requirements. This can enhance the scalability of the approach when using for prioritizing a large number of requirements.

The disagreement among the ranking of requirements converted into a code form of an individual and the elicited pairs of requirements and their elementary constraints on the relative requirements ranking express the fitness function to be minimized (optimized). Both elicitation and optimization are performed simultaneously due to the reason that they have impact on each other. In other words, the most significant pairs of requirements that need to be elicited would be recognized once the genetic algorithm has started to optimize the ranking of requirements with regards to the existing constraints. From the other side, the new elicited pairs of requirements establish new constraints that need to be considered. Therefore, the fitness function exploited by IGA

is designed to work in an incremental way to satisfy the newly produced constraints. As a result, convergence of the genetic algorithm is not guaranteed and is dependent upon the balance of the fitness function over the time, during incremental knowledge utilization. The prioritization process brings into an end once disagreement becomes low and the specified time or elicitation budget finishes.

EVOLVE

Greer et al. proposed an evolutionary and iterative approach called EVOLVE that supports decision making in software release planning (Greer & Ruhe, 2004). This approach gets the advantages of integrating the genetic algorithms and iterative solution method. In practice, a genetic algorithm is used for all iterations to make optimal or closely optimal decisions (because genetic algorithms generally are not able to guarantee of making absolute optimal decisions) regarding the assignment of requirements for the next release. For all iterations, only requirements that satisfy constraints are considered. The main purpose of using the iterative solution method in EVOLVE is to provide the opportunities to apply any types of delayed modifications to requirements, changes to the prioritization of requirements achieved by various stakeholders, effort estimation and constraints which needed for all requirements, precedence and coupling constraints and any changes to the weights of stakeholders. The number of iterations is not fixed throughout the process. The EVOLVE process is illustrated in Figure 2.4.

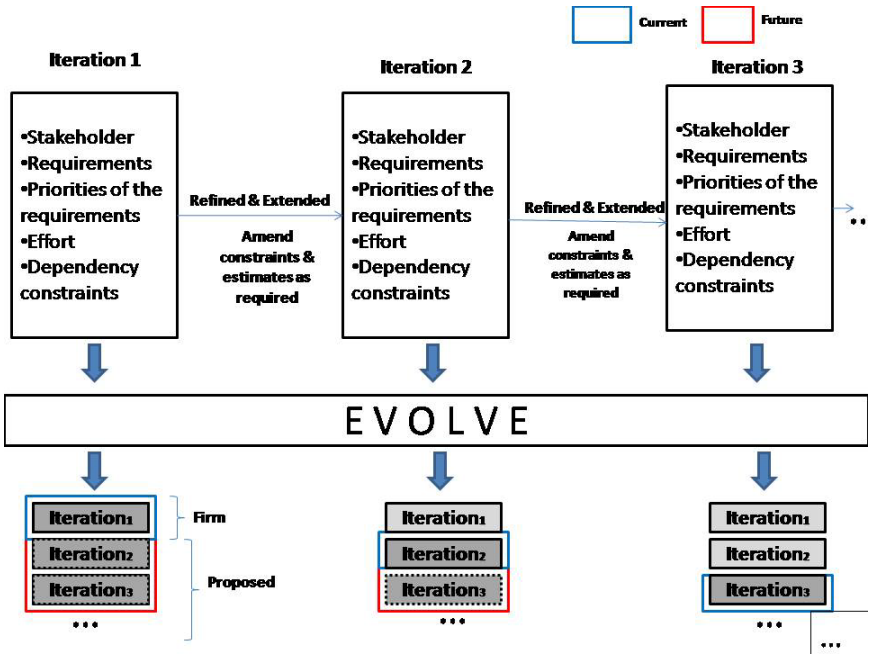


Figure 2.4: EVOLVE approach to select requirements for each release (Iqbal, Khan, & Khan, 2009)

Using EVOLVE approach offers many advantages such as:

- It utilizes genetic algorithms to perform an optimal prioritization process for several stakeholders who may have different viewpoints regarding the priorities of requirements.
- The approach considers various stakeholder opinions, risk constraints, effort constraints, and dependencies among requirements.
- It also provides uncertainty in estimating effort, letting stakeholders to select the confidence level when estimating effort.

TOPSIS-based approach

TOPSIS-based approach (Kukreja, 2013) is a two-step prioritization approach which has been proposed, on the basis of a decision-analysis framework called TOPSIS (Technique of Ordered Preference by Similarity to Ideal Solution) (Jahanshahloo, Lotfi, & Izadikhah, 2006), to prioritize system and software requirements. Initially, the process is started by decomposing the software system which needs to be developed

into some high-level features called Minimal Marketable Features (MMFs). Then, MMFs are furthered broken up into low-level requirements.

The MMFs are prioritized by business stakeholders with respect to business goals and they are assessed based on a scale from 1 to 9 where one indicates the lowest score and nine represents the highest score of MMF for satisfying a particular business goal. In addition, the requirements in the lowest level are also prioritized based on three aspects including relative penalty, business value, and ease of realization by cooperating technical and business stakeholders. The ease of realization is defined as the extent to which the requirement technologically, politically, socially, and economically is achievable.

Case-Based Ranking approach (CBRank)

The Case-Based Ranking approach (CBRank) is a prioritization approach which combines the pairwise comparison and machine learning techniques to calculate the final ordering of requirements (Avesani, Bazzanella, Perini, & Susi, 2005; Perini et al., 2013). The basic idea of integrating machine learning techniques with pairwise comparisons is to overcome the scalability issues concerned with pairwise comparisons. In other words, using machine learning techniques makes the approach applicable for a large number of requirements. Figure 2.5 depicts the main steps of CBRank approach for prioritizing software requirements.

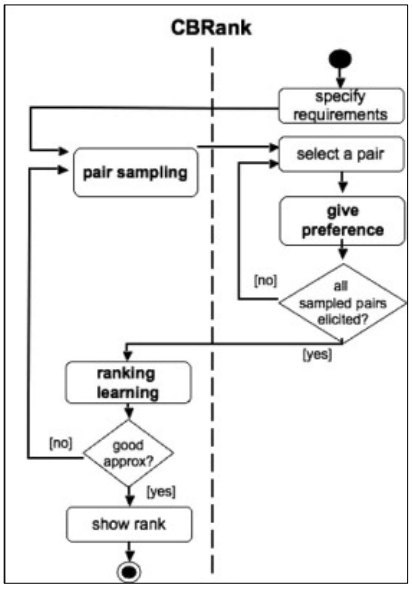


Figure 2.5: Steps of CBRank approach for prioritizing requirements (Perini, Susi, Ricca, & Bazzanella, 2007)

As can be seen in Figure 2.5, the input of the CBRank approach is a set of requirements that need to be prioritized whereas the output of the approach is an estimation of the final ordering. The pair sampling function is an automatic process in which a couple of requirements should be selected based on some pre-defined criteria. The decision maker must assess all pairs of requirements through an iterative process. The ranking learning function takes the preference values of the decision maker as an input, and produces an estimation of the final ordering of requirements as an output. The learning function works according to the boosting approach presented in (Avesani, Bazzanella, Perini, & Susi, 2004). The process is terminated once the ranking generated by the learning function can be recognized as a reasonable output. Otherwise, it should be considered as an input for further iteration.

Value-based Intelligent Requirements Prioritization (VIRP)

VIRP is essentially a three-level prioritization technique which exploits automated fuzzy logic-based system, stakeholders' preferences, and expert's opinions to iteratively prioritize requirements (Muhammad Ramzan, Jaffar, & Shahid, 2011). The iterative

process used in VIRP enhances the reliability of results produced by the prioritization approach. The main steps of the VIRP approach are presented in Figure 2.6.

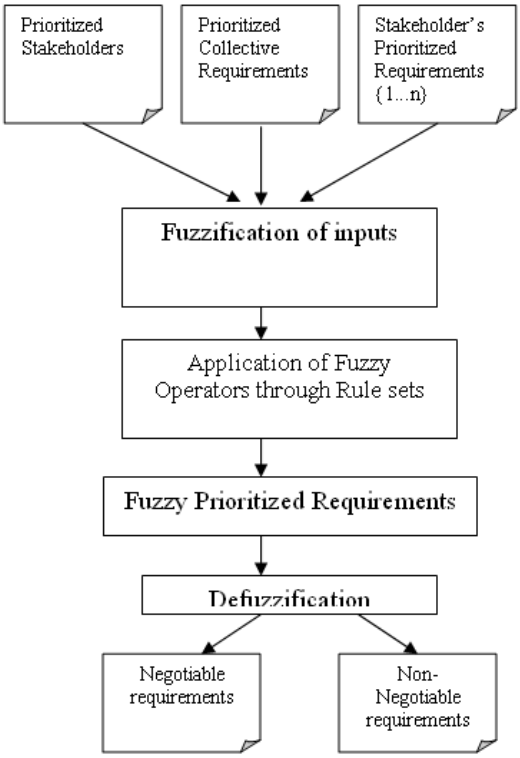


Figure 2.6: The general process of VIRP approach for requirements prioritization (M Ramzan, Jaffar, Iqbal, Anwar, & Shahid, 2009)

Fuzzy Analytical Hierarchy Process (FAHP)

Laarhoven and Pedrycz (Van Laarhoven & Pedrycz, 1983) proposed the Fuzzy Analytical Hierarchy Process in which AHP method and Fuzzy Theory were combined together in order to cope with the vagueness and indefiniteness associated with the opinions of decision makers. Zadeh (Zadeh, 1965) first proposed Fuzzy Theory, which is capable of getting indefinite opinions from stakeholders. After getting undetermined opinions as inputs, then fuzzy set theory is able to specify the extent to which these inputs belong to the appropriate fuzzy sets. The process should be continued by defuzzification process, which normally generates quantifiable weights in the form of numerical values. By integrating fuzzy set theory, AHP is able of managing the

fuzziness of the data involved in decision making in an efficient way. By proposing FAHP, the authors have shown that several concepts and things that exist in the real life may have fuzziness.

Thakurta's approach

Thakurta proposed a quantitative prioritization approach in which non-functional requirements have been focused (Thakurta, 2013). This approach aims at generating a prioritized set of non-functional requirements that need to be implemented for a software project by considering the opinions of stakeholders of both project and business organization. It also considers the objectives of both project and business organization. In addition, the approach is capable of providing quantitative information for determining the degree of the benefit that can be reached while making decision on accepting or ignoring a specific non-functional requirement to be considered in the final requirements of a particular project. The prioritization process additionally represents the importance degree of all business objectives as well as different non-functional requirements achieved by each stakeholder of business organization. The main difference of this approach from existing ones is that it considers the dependencies of non-functional requirements during the prioritization process. Nevertheless, the approach consists of six steps, as listed below, to prioritize non-functional requirements:

1. Identification of non-functional requirements
2. Construction of a project level scenario
3. Connecting scenario to business objectives
4. Calculation the weights of non-functional requirements of scenario
5. Applying a heuristic to ignore some non-functional requirements from a scenario

Interrelationship-based approach approach

Dabbagh and Lee proposed an AHP-based approach for prioritizing non-functional requirements (Dabbagh & Lee, 2013). In this approach, the interrelationships which may exist among non-functional requirements are considered during the prioritization process while non-functional requirements are prioritized based on their importance to the customers and users. In other words, the approach produces a consistent prioritized list of non-functional requirements in which there is no conflicting relationships among the output list. This approach consists of eight steps as described below:

1. Elicit a list of non-functional requirements that need to be prioritized.
2. Collect the preference value between each pair of non-functional requirements.
3. Apply AHP algorithm on the candidate list of non-functional requirements.
4. Insert the results of applying AHP into an array called initial set.
5. Select a non-functional requirement which poses the greatest value of the initial set as well as appending it into the result set.
6. Remove the selected non-functional requirement from the initial set.
7. Remove any non-functional requirement from the initial set which has negative impact on the high-priority non-functional requirement.
8. If the initial set is empty, the process is terminated and the result set is then used as guidance for other members of the project. Otherwise, further iteration would be needed. (go to step 5)

Hybrid Assessment Method (HAM)

HAM is a multiple criteria decision-making method in which the pairwise comparison decision matrix is integrated with the classical weighted average decision matrix to rank a collection of alternatives with respect to a set of criteria. HAM is a two-phase method which begins the prioritization task by eliciting the criteria and the alternatives (Ribeiro, Moreira, Van den Broek, & Pimentel, 2011). The second step performs trade-offs between criteria using pairwise comparisons. The third step is to calculate the criteria priority vector, normalize the respective weights and calculate the consistency ratio. These three steps belong to Phase 1 of the HAM and correspond to an automated determination of weights for the decision matrix. Phase 2 starts in step four, which elicits the contributions of each alternative with respect to each criterion, using a classical weighted average decision matrix. The fifth and final step is an aggregation process to determine the prioritization of alternatives (ranking) using a geometric aggregation operator. This step concludes the HAM's process by providing the ratings for each alternative.

2.2.2 Ordinal Scale Approaches

Simple ranking

Simple ranking method is a common prioritization method by which requirements are prioritized in the same way as the people rank objects in real life (Berander & Andrews, 2005). It means that for n requirements to be ranked, the most important requirement is assigned rank 1 while the least important one is assigned rank n (Hatton, 2008). Using this method, the priority of each requirement is unique, but it is not possible to have the relative importance of two requirements.

Simple ranking method has been considered as medium in terms of granularity and easy in terms of sophistication (Berander, 2004a). This method is suitable to be applied in the prioritization problems like in bespoke development where only one stakeholder involves in the prioritization process since it would be difficult to consider different viewpoints of distinct stakeholders (Khan, 2006). However, it is feasible to aggregate different views by calculating the mean priority of every single requirement which is not really advisable.

Bubble sort

Bubble sort has been basically identified as one the straightforward methods for sorting elements (Hopcroft, 1983) and it has been firstly applied in the area of software requirements prioritization in the study presented by (J. Karlsson et al., 1998). It is interesting to mention that there is a high degree of similarity between bubble sort and AHP method since the number of required pair-wise comparisons in both approaches is $n \times (n-1)/2$. The difference is that by using bubble sort, the decision maker does not need to specify the importance degree between each pair of requirements. Using bubble sort involves four steps as follows:

1. Put the requirements in a vertical column.
2. Compare two requirements from the top of column to find out which one is more important. If the higher requirement is less important than the lower one, change their positions in the column.
3. Continue this process until the bottom of the column is reached.
4. If the position of any requirement has been changed during steps 2 and 3, do the process again for the whole column starting from the top two requirements (step

2). Continue repeating the process until the positions of all requirements are fixed through the column.

The outcome of this process is a prioritized column of requirements where the most important requirement is placed at the top and the least important one is placed at the bottom of the column. The shortcoming of this method is that it is not suitable for prioritizing a large number of requirements.

Binary search tree

Binary tree is defined as a tree where each node contains at most two children and binary search tree is a particular kind of binary tree in which the nodes have labels. Binary search tree has been basically identified as a method for sorting the elements of a set (Hopcroft, 1983) and has been firstly introduced for prioritizing software requirements in (J. Karlsson et al., 1998).

To prioritize n requirements using binary search tree method, a binary search tree with n nodes should be constructed. To start the process, the first requirement needs to be placed in the top node of tree. Then, the next requirement should be compared to the top node. If it has a high priority than top node, it must be placed as the right children of the top node and if it has a low priority than top node, it should be put as the left children of the top node. This process should be repeated until all requirements are placed in the right positions of the binary search tree. Once the binary search tree is constructed completely, the prioritized list of requirements can be achieved by traversing the tree in *in-order*. The time complexity of prioritizing n requirements using binary search tree method is $O(n \log n)$ (Ma, 2009).

Planning Game

Planning Game (PG) has been applied in an Extreme Programming (XP) project with the purpose of making decisions on what to develop for a certain release. Using this technique, customers are first asked to elicit requirements. After eliciting requirements, the customers prioritize them according to three different groups: (1) the requirements that without considering them the software system is not able to work, (2) the requirements that are not considered as critical but produce essential business value, and (3) the requirements that would be better to be considered (Beck, 2000).

Simultaneously, the required time for implementing each requirement is estimated by developers. Then, requirements are prioritized into three different groups: (1) requirements that can be estimated accurately, (2) requirements that can be estimated moderately, and (3) requirements that cannot be estimated.

According to the requirements' importance, time estimation, and also determining the release date of system, the customers prioritize the requirements of each group and then make decision regarding the selection of which requirements for the next release (Newkirk & Martin, 2001). PG exploits a sorting algorithm such as numerical assignment (J. Karlsson, 1996) to prioritize the requirements into one of three groups. Then, the requirements of each group are compared together to produce an ordered list of requirements.

As explained before, the outcome of the PG technique is a prioritized list of requirements. This indicates that the requirements are listed as a ranking on an ordinal scale, without providing any information regarding the extent to which each requirement would be important than the other one.

Quality Function Deployment (QFD)

QFD is a significant method targeted at satisfying customer requirements and converting requirements into design objectives (Wang, Xie, & Goh, 1998). It has been identified as an effective management method in multi-criteria decision-making problems due to its clarity and simplicity. The basis of QFD is to create and deal with a management tool, called House Of Quality (HOQ) (Cohen & Cohen, 1995), which documents the translation of customer requirements into high-level technical specifications. In HOQ, customers are responsible for determining the relative importance of customer requirements. The way of determining the relative importance is not to perform pairwise comparison but to assign a number to each requirement which expresses the customer's viewpoints regarding that requirement. This helps to construct the prioritization matrix method which has been utilized in HOQ to provide the final weight of each requirement. Constructing the prioritization matrix method in HOQ involves some steps as follows:

1. List customer requirements (what).
2. List technical descriptors (how).
3. Develop a relationship matrix between what and how.
4. Develop an interrelationship matrix between how.
5. Develop prioritized customer requirements.
6. Develop prioritized technical descriptors.

Correlation-Based Priority Assessment (CBPA)

The CBPA method has been developed to perform the prioritization of software process requirements obtained from various groups of stakeholders by combining inter-

perspective relationships that exist among different requirements (Liu et al., 2006). Addressing the relationships among different requirements obtained from different stakeholders during the prioritization process could possibly help practitioners to achieve a general knowledge regarding the problem domain as well as the requirements of stakeholders for the software process.

In addition, the CBPA method was proposed to provide the opportunity for business organizations to prioritize the requirements gathered from various stakeholders with the aim of recognizing and concentrating on the most critical subjects elicited from different stakeholders. CBPA showed constant and changeless efficiency under circumstances where a wide range of requirements were gathered from various perspectives. By applying this method, those software process requirements that have more and solid effects on other requirements from various perspectives may be assigned as high-priority requirements. Satisfying these high-priority requirements can enhance the overall quality of the final product. The cooperative prioritization of process requirements prepares a noteworthy guideline to mitigate project risks that may have negative influence on the improvement of software process. It also assists to enhance the quality of the process, which provides many benefits for all associated stakeholders.

2.2.3 Nominal Scale Approaches

Numerical assignment

Numerical assignment has been recognized as a popular method for prioritizing requirements, as acknowledged by different studies such as (Berander & Andrews, 2005; Bradner, 1997; S. IEEE, 1998; L. Karlsson et al., 2006; Leffingwell & Widrig, 2000; Sommerville & Sawyer, 1997). Furthermore, it has been identified as a medium and very easy method in terms of granularity and sophistication, respectively (Berander,

2004a). The idea in numerical assignment method is to divide requirements into different priority groups. Although the number of groups may be different from case to case, choosing three groups is common in general. For instance, requirements can be categorized based on three groups such as *critical, standard*, and *optional*. Labelling groups with words such as high, medium, and low may confound stakeholders since different stakeholders could have their own interpretation regarding these terms. Hence, it is necessary to provide stakeholders with common definitions of each group before asking them to prioritize.

As mentioned in (Berander, 2004b), one of the shortcomings of this method is that most of the stakeholders have a tendency to label each requirement as a critical one. A solution to overcome this issue is to force stakeholders to choose only a limited number of requirements for each group. However, this may result in reducing the efficiency of the prioritization process. The outcome of numerical assignment method is represented using nominal scales so that the requirements of each group have the same priority.

Top-ten requirements

Using top-ten requirements approach for prioritizing requirements involves asking stakeholders to choose the most ten significant requirements among all candidate requirements from their points of view without specifying the relative importance between selected requirements (Berander & Andrews, 2005). This makes the approach particularly ideal for several stakeholders with the same level of importance (Lauesen, 2002). The intention of not prioritizing some more is that it could cause inessential conflicts when some stakeholders get support for their top priority and others only for their third priority. One might presume that conflicts could happen in any case if, for instance, one customer considers three top-ten requirements into the product whereas

another addresses six top-ten requirements into the product. Nevertheless, it is not just a good choice to get an average among all stakeholders because it could result in failing some stakeholders to get any of their top requirements (Lauesen, 2002). Alternatively, it is essential that a number of important requirements are fulfilled for every stakeholder. This might clearly lead to a situation that dissatisfies all customers rather than fulfilling a few customers entirely. The major challenge associated with this approach is to handle these issues.

MoSCoW

MoSCoW is based on numerical assignment and it has been suggested by (Hatton, 2007, 2008; Tudor & Walter, 2006). It is also integrated into the software development methodology, called Dynamic Systems Development Method (DSDM). The basic idea behind MoSCoW is to partition all candidate requirements into four priority groups including "MUST have", "SHOULD have", "COULD have", and "WON'T have" (Ma, 2009). MoSCow is an acronym stands for:

- "MUST have" represents that requirements of this group should be included in the project. Failure to satisfy these requirements equals to the failure for the whole project.
- "SHOULD have" indicates that the project would perform better if the requirements of this group are included.
- "COULD have" also represents that the project would perform better if the requirements of this group are included. But these requirements are less important than the requirements in the "SHOULD have" group.

- "WON'T have" is like a "wish list". It indicates that although the requirements of this group are good but they could not be considered in the current release. They might be included in the next release.

This method is not able to provide the relative importance of requirements of each group. In other words, all the requirements within each group have the same priority.

Requirements triage

A semi-automated prioritization approach has been proposed by (Duan et al., 2009) with the purpose of producing a prioritized set of requirements from a large list of unrefined stakeholders' demands using data mining and machine learning techniques. It has been also indicated that how the prioritized list of requirements might be utilized to notify and supply the triage process. The approach, which is referred to as Pirogov after one of the inventors of early triage practices, exploits clustering techniques to put requirements into several distinct classes that take the different and complicated roles played by separate requirements. For instance, one clustering technique classifies requirements according to feature sets; the other one captures and clusters non-functional requirements or early aspects; whereas others classify requirements based on user-defined aspects including high-level use cases, business goals, or available code samples. The final output result is that each requirement is placed into one or more feature sets, while a cross-cutting subset of requirements are located into supplementary classes. Using this approach, Stakeholders are in charge of specifying the relative weight of each cluster as well as determining the importance weight of each clustering method.

The approach provides some advantages. First, the required elicitation effort by stakeholders to perform the prioritization process is remarkably decreased. For example, to prioritize 1000 requirements using the binary search tree method, 10000 comparisons are needed while to prioritize the same number of requirements using the presented approach, only 500 comparisons should be done. The second advantage is that the approach allows stakeholders to make decisions at a higher level of abstraction, which are further automatically transformed down to the requirement level. The third advantage is that the approach enables using multiple criteria throughout the triage process. The last advantage of the approach is that decisions made at the clustering level can be used for different releases of the product, and can also be used to automatically filter next requirements and other stakeholder demands. On the other side, the restrictions of the approach are associated with the restrictions of underlying data mining and machine learning techniques which have been used in the approach.

2.2.4 Interval Scale Approaches

Requirements Uncertainty Prioritization Approach (RUPA)

The necessity for addressing uncertainty in requirements prioritization is introduced and established in a prioritization approach called Requirements Uncertainty Prioritization Approach (RUPA) (Voola & Babu, 2012). The idea of uncertainty and imprecision is described in this approach by introducing a probability distribution across individual scores and score intervals to determine the importance of requirements. This approach integrates extensive numerical assignment method (Voola & Babu, 2013) and interval evidential reasoning algorithms to prioritize requirements.

2.2.5 Overview of Current Requirements Prioritization Approaches

This section aims to provide a summary, as given in Table 2.7, on the existing requirements prioritization approaches that have been discussed thoroughly in the preceding sections. The explanation and limitation of each presented requirements prioritization approach are given in Table 2.7.

Table 2.7: Overview of requirements prioritization approaches

Source	Prioritization approach	Explanation	Disadvantage
(J. Karlsson, 1996)	Analytical hierarchy process	This approach compares all possible pairs of requirements to calculate the relative weight of each requirement	Not scalable when used for prioritizing a large number of requirements
(Berander & Andrews, 2005)	Hundred dollar method	Stakeholders receive 100 fictitious units and they are asked to allocate these units among candidate requirements	Does not work well for a large number of requirements
(J. Karlsson et al., 1998)	Minimal spanning tree	This approach uses a weighting score to order requirements in a directed graph	Not suitable to recognize inconsistent judgments
(J. Karlsson & Ryan, 1997)	Cost-value approach	Requirements are prioritized based on value to customers and implementation cost	It is time-consuming when applied for a large number of requirements
(Jim Azar et al., 2007)	Value-oriented prioritization	Requirements are prioritized according to their impact on business values	Not address requirements dependencies in the prioritization process
(J. Karlsson et al., 1998)	Hierarchy AHP	Requirements are placed in a hierarchy and then AHP is applied on each hierarchy	Not able to identify inconsistent judgments
(K. E. Wiegers, 2003)	Wieger's method	Requirements are prioritized based on benefit, penalty, cost, and risk	It lacks of mathematical basis and not suitable for large projects

Table 2.7, continued

Source	Prioritization approach	Explanation	Disadvantage
(Tonella et al., 2013)	Interactive genetic algorithm	It uses a genetic algorithm to reduce the amount of pairwise comparisons that need to be elicited from the decision maker	Did not apply on different case studies to show its effectiveness
(Greer & Ruhe, 2004)	EVOLVE	It integrates the genetic algorithms and iterative solution method to make optimal solutions	It suffers from time complexity
(Kukreja, 2013)	TOPSIS-based approach	Requirements are prioritized by stakeholders according to business goals	Did not address requirements dependencies
(Perini et al., 2013)	Case-based ranking	It exploits machine learning algorithms to reduce the number of pairwise comparison	It sacrifices the accuracy of results for decreasing the time
(Muhammad Ramzan et al., 2011)	Value-based intelligent requirements prioritization	It uses automated fuzzy logic-based system, stakeholders' preferences, and expert's opinions to iteratively prioritize requirements	Not scalable to be applied in large projects
(Van Laarhoven & Pedrycz, 1983)	Fuzzy AHP	It combines triangular fuzzy numbers and AHP algorithm to prioritize requirements	Not suitable for prioritizing a large number of requirements
(Thakurta, 2013)	Thakurta's approach	Non-functional requirements are prioritized based on business objective and stakeholders' ratings	Decisions might be non-optimal due to the subjectivity of inputs
(Dabbagh & Lee, 2013)	Interrelationship-based approach	Non-functional requirements are prioritized based on their interrelationships and AHP algorithm	It lacks of empirical evaluation
(Berander & Andrews, 2005)	Simple ranking	Requirements are prioritized from 1 to n	It lacks of providing relative importance between requirements and it is not recommended for a large number of requirements
(J. Karlsson et al., 1998)	Binary search tree	This method prioritizes requirements by traversing a binary tree of requirements in in-order	It is not able to show the relative weights of requirements

Table 2.7, continued

Source	Prioritization approach	Explanation	Disadvantage
(J. Karlsson et al., 1998)	Bubble sort	Requirements are prioritized according to the following steps: (1) Inserting requirements in a vertical vector (2) Performing the comparisons of requirements (3) Ordering requirements from bottom-up	It is not suitable for prioritizing a large number of requirements
(Beck, 2000)	Planning Game	Requirements are first categorized into three groups and then prioritized by relevant stakeholders	It lacks of providing information regarding the extent to which each requirement would be important than the other
(Wang et al., 1998)	Quality function deployment	It uses the prioritization matrix method to produce the final rank of each requirement	Not scalable for prioritizing a large number of requirements and not able to identify inconsistent judgments
(Liu et al., 2006)	Correlation-based priority assessment	Requirements are prioritized by synthesizing requirements correlations using relationship matrix	It lacks of addressing negative correlations between requirements during the prioritization process
(L. Karlsson et al., 2006)	Numerical assignment	Using this method, requirements are divided into different priority groups	Its shortcoming is that stakeholders tend to put requirements in the most important group
(Berander & Andrews, 2005)	Top-ten requirements	Stakeholders select their top-ten requirements from all candidate requirements	It is not effective when many stakeholders with conflicting viewpoints involve in the prioritization process
(Ma, 2009)	MoScoW	All candidate requirements are partitioned into four priority groups including "MUST have", "SHOULD have", "COULD have", and "WON'T have"	It is not able to provide the relative importance of requirements within each group
(Duan et al., 2009)	Requirements triage	Requirements are prioritized from a large list of unrefined stakeholders' demands using data mining and machine learning techniques	It is capable of making errors during the prioritization process

Table 2.7, continued

Source	Prioritization approach	Explanation	Disadvantage
(Voola & Babu, 2012)	Requirements uncertainty prioritization approach	It uses an extensive numerical assignment method and interval evidential reasoning algorithms to prioritize requirements	It is not scalable for prioritizing a large number of requirements

2.3 Empirical Evaluations of Requirements Prioritization Approaches

A number of empirical studies have been conducted in the literature to perform the comparative and experimental evaluations of the state-of-the-art prioritization techniques, methods and approaches. To compare the prioritization approaches, some properties of the techniques, methods and approaches, such as time-consumption, scalability, ease of use, and accuracy of results, have been measured throughout the empirical studies. This section aims to provide a review on the most important comparative evaluations of existing prioritization approaches which inspired this work.

2.3.1 The Evaluation of Minimal Spanning Tree, Bubble Sort, Binary Search Tree, Priority Groups, Hierarchy AHP, and AHP

An empirical study was conducted by (J. Karlsson et al., 1998) to compare six prioritization approaches including minimal spanning tree, bubble sort, binary search tree, priority groups, Hierarchy AHP, and AHP. The six approaches have been applied on thirteen requirements of telephony system. These requirements have been prioritized by three authors of the study. The prioritization was done based on the importance of requirements for customers. Each author applied the approaches randomly without any pre-defined execution orders.

During the study, the authors have used two types of measurements for comparative evaluations of the six prioritization approaches. These measurements included both

subjective and objective measurements. A range of ordinal numbers from one to six have been used for the measurements so that one represented the best choice whereas six represented the worst option. The results of performing the experimental study are provided in Table 2.8.

The authors considered *time consumption per decision*, *total time consumption*, and *required number of decisions* as objective measurements.

- *Time consumption per decision* represented the average time needed for making one decision using a given prioritization approach.
- *Total time consumption* indicated the total time needed for performing the prioritization process using a particular approach.
- *Required number of decisions* determined the total number of decisions needed to be made using each prioritization approach.

In addition, three properties such as *fault tolerance*, *reliability of results*, and *ease of use* have been selected as subjective measurements.

- *Fault tolerance* indicated the capability of recognizing inconsistent judgements.
- *Reliability of results* showed how much the result of a specific prioritization approach could be reliable.
- *Ease of use* described how much a prioritization approach would be easy for performing the prioritization process.

Table 2.8: Results of objective and subjective measurements (J. Karlsson et al., 1998)

Approach / Criteria	AHP	Hierarchy AHP	Spanning tree	Bubble sort	Binary search	Priority groups
Time consumption per decision	2	4	5	1	6	3
Total time consumption	6	2	1	3	5	4
Required number of decisions	78	26	12	78	293338	343536
Fault tolerance	1	3	6	2	4	5
Reliability of results	1	3	6	2	4	5
Ease of use	3	4	2	1	5	6

According to the results presented in Table 2.8, it can be concluded that AHP is able to produce trustworthy results compared to the other six approaches. However, it takes a long time to perform the whole prioritization process. Minimal spanning tree method seems to be the fastest method among others for requirements prioritization and also requires a minimum number of decisions to be made. Nevertheless, it lacks of providing reliable results and being fault tolerant. Bubble sort has been identified as an easiest method and can produce almost reliable results though it requires a large number of decisions. Binary search tree and hierarchy AHP can be considered as moderate prioritization approaches since they generate less reliable results in comparison with AHP and bubble sort. On the other side, they need less time than AHP and bubble sort to perform the prioritization process.

2.3.2 The Evaluation of Hundred Dollar Method, Planning Game, AHP, Binary Search Tree, and PGcAHP

Ahl (Ahl, 2005) conducted a controlled experiment to perform the comparative evaluation of five prioritization approaches including the hundred dollar method, Planning Game, AHP, binary search tree, and a combination of planning game and AHP. The five prioritization approaches have been applied on thirteen requirements of a

course registration system. Fourteen subjects including twelve bachelor and master students and two professionals participated in the controlled experiment.

In order to perform the comparative evaluation of the five prioritization approaches, the author measured four properties such as *scalability*, *accuracy of results*, *ease of use*, and *time consumption*, through the controlled experiment. A range of ordinal numbers from one to five have been used for measuring *scalability*, *accuracy of results*, and *ease of use* while *time consumption* has been assessed in terms of real time. Table 2.9 provides the average results obtained from executing the controlled experiment.

- *Scalability* has been measured by asking subjects regarding the scalability of a given prioritization approach when applied on a large number of requirements.
- *Accuracy of results* has been measured by asking participants to express his/her opinions about the accuracy of results produced by a particular prioritization approach.
- *Ease of use* has been measured by asking subjects to determine how easy a given prioritization approach could be used for performing the prioritization task.
- *Time consumption* has been measured by asking participants to keep record of how long it took them to apply a particular prioritization approach.

Table 2.9: Initial results of controlled experiment (Ahl, 2005)

Approach / **Criteria**	Hundred dollar	Planning Game	AHP	Binary search tree	PGcAHP
scalability	3	4	2	3	3
Accuracy of results	3	3	2	3	2
Ease of use	3	3	2	3	3
Time consumption	3.5	2.7	11	9.15	5.28

The initial results obtained from executing the controlled experiment have been analysed statistically by the author to provide more rigorous findings. According to the

final results of the study, the binary search tree method has been identified as the most suitable approach for prioritizing requirements. This could be due to the reasons that it could provide the most accurate results among other approaches, i.e. it took reasonable time for performing the prioritization task, it was an easy method to use, and it could also be easy to apply even for a large number of requirements. Even though the results have shown that the binary search tree method would be the best prioritization approach, some variables such as previous experiences of subjects on the prioritization approaches could have some effect on the final result. Another interesting point reported from this study was that the test subjects selected the combination of Planning Game and AHP, i.e. PGcAHP, as one of the best overall methods for requirements prioritization even though they did not give a high rank to PGcAHP when measuring the other properties. A reason for this could be that there were few requirements or few subjects to get a clear answer.

2.3.3 The Evaluation of Hundred Dollar Method, AHP, MoSCoW, and Simple Ranking

Another empirical study conducted by Hatton to evaluate the effectiveness of four prioritization approaches including the hundred dollar method, AHP, MoSCoW, and simple ranking method (Hatton, 2007). In practice, the effectiveness of these approaches has been evaluated by measuring three properties such as user confidence, the required time to complete the prioritization process, and ease of use. The four mentioned approaches have been applied on twelve requirements of a mobile phone software system. In addition, thirty one persons from a different range of careers, educational levels, genders, and ages have been considered as participants of the empirical study.

Each subject of the study was asked to apply each prioritization approach on the requirements of the mobile phone software system. Then, they were also asked to determine his/her confidence rate regarding each approach, record the time needed by each approach to perform the whole prioritization task, and specify the difficulty level of each approach.

The time needed by each prioritization approach was measured by asking the participants to record the time he/she commenced and terminated the prioritization process while working with each approach. To measure the difficulty (ease of use) of each approach, an ordinal scale ranging from one to ten was used, where one represented the lowest level and ten expressed the highest level of difficulty. The user-confidence level was also rated on an ordinal scale from one to ten, where one represented "not confident" and ten expressed "very confident". The results of the empirical study are presented in Table 2.10 and Table 2.11.

Table 2.10: Results of times taken, median confidence, and median difficulty (Hatton, 2007)

	Minimum time	Maximum time	Mean time	Standard deviation	Median confidence	Median difficulty
MoSCoW	1	5	1.78	1.083	8	2
Simple Ranking	1	4	1.5	0.73	8	3
Hundred dollar	1	8	3.6	2.42	7	4
AHP	7	22	14.03	4.4	2	9

Table 2.11: Properties of the prioritization approaches (Hatton, 2007)

	Simple ranking	MoSCoW	Hundred dollar	AHP
Ratio scale information			√	√
High confidence from user	√	√	√	
Consistent	√	√	√	√
Low difficulty	√	√	√	
Low effort	√	√	√	
Able to handle large numbers of alternatives		√		

According to the results presented in Table 2.10, it is clear that AHP requires the longest time to complete the prioritization task among the other four approaches. Regarding the mean time values, AHP takes much longer time to perform the prioritization process. MoSCoW and simple ranking have achieved the highest rate of confidence as well as the lowest degree of difficulty. AHP shows the highest rate of difficulty and the lowest confidence rate.

The results presented in Table 2.11 imply that the outcomes of all four approaches are consistent. This indicates that the outcomes of all four approaches reflect the actual viewpoints of participants regarding the priorities of requirements. MoSCoW has been identified as the easiest approach to use among the four other approaches. It gives high user confidence and it requires less time to complete. Simple ranking has been also recognized as an easy approach to use. Similar to MoSCoW, it gives high user confidence and it requires less time to complete. Even though the hundred dollar method needs longer time to complete and provides less user confidence than simple ranking and MoSCoW, it is almost easy to use, provides high user confidence, and requires less time to complete. AHP has been identified as the most difficult approach to use.

2.3.4 The Evaluation of AHP and Planning Game

In the study conducted by (L. Karlsson, Thelin, Regnell, Berander, & Wohlin, 2007), two controlled experiments have been carried out with the aim of evaluating three prioritization approaches. In the first controlled experiment, two prioritization approaches, AHP and Planning Game, have been compared to find out which one is more effective. Sixteen subjects including fifteen students and one professor

participated in the experiment. They have been asked to apply AHP and Planning Game on sixteen requirements of mobile phone.

In order to perform the comparative evaluation of the AHP and Planning Game, the authors measured one objective property including *time*, as well as two subjective properties, *accuracy of results*, and *ease of use*, through the controlled experiment. *Time* was measured by asking subjects to determine the start and end time of performing the prioritization task while using each approach. *Accuracy of results* was measured by a post-questionnaire which was submitted to the subjects a few weeks after the experiment. *Ease of use* was measured by asking the subjects immediately after the experiment "Which approach did you find easiest to use?" The results of the first controlled experiment are given in Table 2.12.

Table 2.12: Results of the first controlled experiment for comparing AHP and Planning Game (L. Karlsson et al., 2007)

	AHP	Planning game	p-value	direction
Time	26.7min	12.0min	0.0209	Planning Game
Ease of use	6%	75%	0.0023	Planning Game
Accuracy of results	19%	56%	0.2200	---------

According to the results provided in Table 2.12, it is clear that Planning Game shows a better performance compared to AHP in terms of ease of use and time-consumption. Regarding the accuracy of results produced by the two approaches, there is no evidence of superiority of one approach over another one. This indicates that the prioritized lists of requirements generated by the two approaches are closely similar.

In the second controlled experiment of this study, the tool-supported version of AHP has been compared to Planning Game. The motivation behind conducting the second controlled experiment was that even though the first experiment showed the superiority

of Planning Game over AHP, the authors thought that the tool-supported version of AHP could cover the defects of its manual version. The selected requirements of the second experiment were exactly the same as the first experiment while the subjects were different to prevent biasing the final results. Thirty subjects including 25 male and 5 female master students participated in the second experiment. Similar to the first experiment, three properties such as *time*, *ease of use*, and *accuracy of results* have been measured within the second experiment. Table 2.13 shows the results of the second experiment.

Table 2.13: Results of evaluating tool-supported AHP and Planning Game (L. Karlsson et al., 2007)

	Tool-supported AHP	Planning game	p-value	direction
Time	9.4min	11.3min	0.0400	Tool-supported AHP
Ease of use	53%	33%	0.2393	---------
Accuracy of results	37%	50%	0.3270	---------

By analysing the results presented in Table 2.13, it can be found that the tool-supported AHP is much faster than Planning Game in performing the prioritization task. However, the statistical tests of ease of use and accuracy of results indicate that there is not much difference between the two approaches with regards to these two properties.

2.3.5 The Evaluation of AHP and CBRank

Perini et al. conducted a controlled experiment to evaluate two tool-supported requirements prioritization approaches, AHP and CBRank (Perini, Ricca, & Susi, 2009). The experiment was carried out by participating twenty three well-experienced subjects including eight researchers and fifteen Ph.D. students. The subjects of the study have been asked to apply tool-supported versions of AHP and CBRank on a set of twenty requirements of the Compilation Compiler Advisor (CoCoA) project. To compare the

two approaches, the authors focused on measuring three properties such as *time-consumption*, *ease of use*, and *accuracy*.

Time-consumption of each requirements prioritization approach has been measured automatically by the software tool through recording the start and end time of working with a given prioritization approach. *Ease of use* was measured by a post-questionnaire through asking the subjects to identify which approach was easiest to use. *Accuracy* was measured in two ways including *expected* accuracy and *perceived* accuracy. *Expected* accuracy was measured immediately after the experiment by a post-questionnaire through which the subjects were asked to determine which approach produced the most accurate results. *Perceived* accuracy was measured after a long break by another post-questionnaire through which the subjects were presented with two prioritized list of twenty requirements provided during the working sessions with the two tool-supported versions of AHP and CBRank. The subjects were not aware of which approach produced a prioritized list. Then, each subject was asked to select a list which better showed his/her priority orders. This list identified the approach which produced most accurate results.

Once the experiment has been executed, the data was collected and then statistical tests were carried out on the obtained results to make rigorous conclusions. Table 2.14 presents the results of the controlled experiment.

Table 2.14: Results of the controlled experiment for comparing AHP and CBRank (Perini et al., 2009)

		Tool-supported AHP	Tool-supported CBRank	p-value	direction
Time-consumption		38.65min	10.78min	< 0.01	Tool-supported CBRank
Ease of use		5%	95%	< 0.01	Tool-supported CBRank
Accuracy	Expected	70%	30%	0.09	---------
	Perceived	100%	0%	< 0.01	Tool-supported AHP

Based on the results provided in Table 2.14, CBRank shows a better performance compared to AHP in terms of time-consumption and ease of use. Regarding the accuracy of results, AHP outperforms CBRank.

2.3.6 Overview of Empirical Studies

A summary on the most important comparative evaluations of different requirements prioritization approaches which have been explained above is illustrated in Table 2.15. They are appeared and sorted based on the year of publication.

Table 2.15: An overview of comparative evaluations of different prioritization approaches

Source	Requirements prioritization approaches under evaluation	Measured properties of the prioritization approaches	Number of requirements	Number of subjects participated in the study	Outcome
(J. Karlsson et al., 1998)	spanning tree matrix, bubble sort, binary search tree, priority groups, Hierarchy AHP, and AHP	reliability of results, required completion time, and ease of use	13 non-functional requirements of a small telephony system	3 scholars	AHP and bubble sort showed a better performance with respect to reliability of results, and ease of use. Spanning tree matrix required less time to perform the prioritization task.
(Ahl, 2005)	Hundred dollar method, Planning game, AHP, binary search tree, and PGcAHP	scalability, accuracy, ease of use, and time	13 requirements of course registration system	14 students and professionals	Planning game was the easiest method to apply. Hundred dollar and binary search were the most accurate methods. Binary search and planning game were the most scalable prioritization methods. The binary search tree was identified as the most effective method for requirements prioritizing.
(Hatton, 2007)	Hundred dollar method, AHP, MoSCoW, and simple ranking	User confidence, the time to complete the prioritizing process, and ease of use	12 mobile phone requirements	31 subjects	MoSCoW was identified as the most easiest to use among other methods. AHP was the slowest one among others. Simple ranking and MoSCoW indicated the best performance for user confidence.
(L. Karlsson et al., 2007)	AHP and Planning game in the first experiment, tool-supported AHP and Planning game in the second experiment	accuracy, ease of use, and time-consumption	16 mobile phone requirements	16 students and scholars in the first experiment, 30 master students in the second experiment	The automated version of pair-wise comparisons indicated a better performance in terms of time-consumption as well as ease of use whereas it could not declare obviously which one was more accurate.
(Perini et al., 2009)	Tool-supported versions of AHP and CBRank	accuracy of results, time-consumption, and ease of use	20 requirements of the Compilation Compiler Advisor (CoCoA) project	23 Ph.D. students and scholars	For time-consumption as well as ease of use, CBRank has overcome AHP, while for the accuracy AHP performed better.

2.4 Concluding Remarks

An in-depth studying and analysis of the current requirements prioritization approaches, which was discussed in Section 2.2, indicated that each of the existing approaches has its own benefits and disadvantages. However, one of the important gaps that has been neglected through all these approaches is that none of them has investigated and addressed the prioritization of both functional and non-functional requirements simultaneously during the prioritization process. In other words, it has been found that the current requirements prioritization approaches can only be applied with functional requirements or non-functional requirements separately. As discussed earlier, considering both functional and non-functional requirements is crucial since these two types of requirements are interrelated and have a serious impact on each other, and not properly addressing them together in the prioritization process could ultimately result in delivering low-quality software products. Hence, fulfilling this gap has been considered as a major motivation toward conducting this research to provide researchers and practitioners with an approach which is capable of integrating the prioritization of functional and non-functional requirements simultaneously. Therefore, this research aims to introduce an approach which allows practitioners to prioritize both functional and non-functional requirements simultaneously in an integrated manner by establishing their relationships, ultimately producing the prioritized lists of functional and non-functional requirements separately.

A thorough reviewing of the most important empirical evaluations of the existing requirements prioritization approaches, which was presented in Section 2.3, could provide guidelines on how to evaluate and compare a requirements prioritization approach which specifically could be beneficial for evaluating a new prioritization approach. In fact, by analyzing the empirical evaluations of different requirements

prioritization approaches, it has been found that three properties of requirements prioritization approaches have been measured during all of the empirical studies. Hence, this could be helpful to choose three properties, such as time-consumption, accuracy of results, and ease of use, as the most essential properties that need to be measured for evaluating a new prioritization approach. Furthermore, it has been discovered that AHP has been widely applied as a reference approach in empirical evaluations of different prioritization approaches. In addition, analyzing the comparative evaluations of different requirements prioritization approaches could contribute this research towards choosing the number of requirements and participants for evaluating a new prioritization approach.

2.5 Summary

This chapter provided an extensive literature review which has been conducted in this research on the area of requirements prioritization. In practice, the chapter started with discussing about the basic concepts such as requirements, requirements engineering and requirements prioritization. This could possibly assist researchers to get fundamental and general ideas regarding the current body of knowledge in the context of requirements prioritization. The chapter continued with presenting a literature review on the existing requirements prioritization approaches. This could bring some insights for conducting this research. First, it highlighted the strengths and weaknesses of the existing requirements prioritization approaches. Second, it was beneficial to find out the research gap of this study. Ultimately, a literature review on the most important empirical evaluations of the existing requirements prioritization approaches has been presented in this chapter. This was helpful to provide guidelines on how to evaluate a requirements prioritization approach.

CHAPTER 3: RESEARCH METHODOLOGY

The main purpose of conducting this research is to provide researchers and/or practitioners with an approach which is capable of integrating the process of prioritizing functional and non-functional requirements simultaneously. To achieve the desired goal, three successive phases required to be accomplished throughout the research as illustrated in Figure 3.1. They are defined as follows: preparation phase, approach development and validation phase, and evaluation phase. Each phase consists of some activities which have been performed to form the whole body of this research. Moreover, conducting each phase of the research assisted the researcher to address and achieve one research objective defined in Chapter 1. Figure 3.1 also sketches the relationship between the three phases of the research and the three research objectives, where the preparation phase has addressed the research objective 1, the approach development & validation phase has tackled the research objective 2, and the evaluation phase has tackled the research objective 3. The following sections explain the three mentioned phases in more detail.

3.1 Preparation Phase

In the initial phase of conducting this research, called preparation phase, a detailed review of the literature has been performed on the area of software requirements prioritization. This resulted in collecting:

- Several studies that investigate different requirements prioritization techniques, methods, and approaches proposed in recent years to perform the requirements prioritization process.

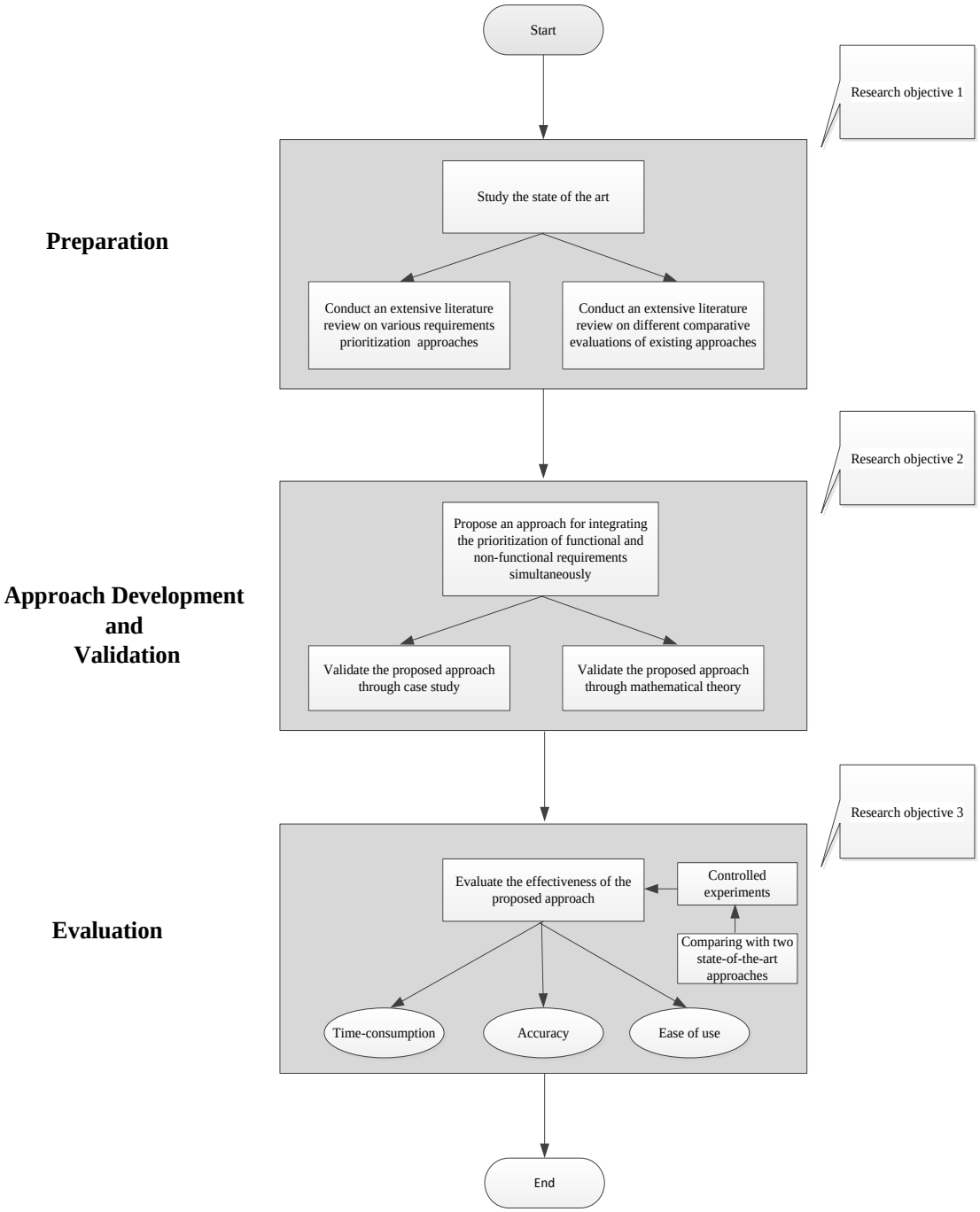


Figure 3.1: Flow chart of the research methodology

- A number of empirical studies that had been carried out and reported in order to perform the comparative evaluations of the existing techniques, methods, and approaches.

Conducting an extensive literature review, on the various requirements prioritization approaches, was useful to find out the research gap of this study. It should be pointed out that the detailed explanations of the current approaches are provided in Chapter 2. In fact, after the final review of the different requirements prioritization approaches, it has been found that addressing both functional and non-functional requirements within a single prioritization approach has received less attention in current prioritization approaches. This could be considered as a motivation to propose an approach in which prioritization of functional and non-functional requirements could be integrated simultaneously.

On the other hand, a review on the most important empirical evaluations of existing prioritization approaches, that inspired our work, has been conducted throughout the preparation phase. Generally, this review targeted toward determining the guidelines on how to evaluate and compare the requirements prioritization approaches. In particular, it contributed to this research by identifying the significant properties that need to be measured for evaluating a new requirements prioritization approach. Further discussions regarding the most important comparative evaluations of existing prioritization approaches are given in Chapter 2.

3.2 Approach Development and Validation Phase

Three main activities which have been conducted throughout the approach development and validation phase are proposing an integrated approach for simultaneous prioritization of functional and non-functional requirements, validating the proposed approach through a case study, and validating the proposed approach through mathematical theory. In the following, these activities are explained briefly.

3.2.1 Propose an Integrated Approach for Simultaneous Prioritization of Functional and Non-functional Requirements

As the beginning part of the approach development and validation phase, an approach has been proposed throughout this research with the aim of integrating the process of prioritizing functional and non-functional requirements simultaneously. This approach is called Integrated Prioritization Approach (IPA). IPA can be defined as an approach which prioritizes both functional and non-functional requirements simultaneously, producing two prioritized lists of functional requirements and non-functional requirements separately. In other words, by applying IPA on a candidate list of functional requirements as well as a candidate list of non-functional requirements, researchers/practitioners are able to obtain a prioritized list of non-functional requirements along with a prioritized list of functional requirements.

The initial motivation behind proposing such an approach is particularly based on the idea that non-functional requirements may affect several functional requirements (Berander & Andrews, 2005). Inspired by this point, the intention is to find out the extent in which each non-functional requirement may affect a given functional requirement. By determining such a value (i.e. the importance degree of a non-functional requirement for a given functional requirement), IPA aims to prioritize functional and non-functional requirements simultaneously according to the relationship which may exist among functional and non-functional requirements. It should be noted that IPA uses only one decision matrix to perform the prioritization task.

By establishing the relationship between functional and non-functional requirements, IPA produces a prioritized list of non-functional requirements by calculating the total importance degree of each non-functional requirement with respect to all related functional requirements. In other words, it means that a non-functional requirement which achieves the greatest total importance degree among all functional requirements

may be assigned as a high-priority non-functional requirement with respect to all functional requirements. Moreover, it provides a prioritized list of functional requirements with respect to all candidate non-functional requirements.

IPA is a nine-step approach which can be used in both single and group decision making problems. The nine steps of IPA are listed below. The whole description of each step is provided in Chapter 4.

1. Identify stakeholders of software system
2. Specify the weights of stakeholders using Analytic Hierarchy Process
3. Identify functional and non-functional requirements
4. Extract functional and non-functional requirements statements
5. Construct the decision matrix
6. Elicit the importance degree of each non-functional requirement (NFR) with respect to each functional requirement (FR) to establish the relationship between functional and non-functional requirements
7. Calculate NFRs ranking with respect to all FRs using triangular fuzzy number and alpha cut approach
8. Compute FRs ranking using weighted average decision matrix and weights determined in Step 7
9. Aggregate different prioritized lists of FRs and NFRs provided by various stakeholders to obtain final rankings of FRs and NFRs

3.2.2 Validate the Proposed Approach through a Case Study

In order to validate the proposed approach (i.e. IPA), it has been applied on a case study to demonstrate how the nine steps of the proposed approach could be utilized in practical solutions. Indeed, in this part of research, IPA has been applied on a collection of 15 functional requirements and 5 non-functional requirements of Automated Teller Machine (ATM), Cash Deposit Machine (CDM), and Check Deposit Machine (CQM).

Four users of ATM, CDM, and CQM, who played the role of stakeholders, participated in the case study. The detailed explanation of validating IPA through case study is given in Chapter 5.

3.2.3 Validate the Proposed Approach through Mathematical Theory

In this part of research, mathematical theory is used to show that the outcome of the proposed approach is proven to be valid. To perform such an activity, the proposed approach is expressed using mathematical formulation. Then, graph theory is utilized to prove that the final prioritized list of functional and non-functional requirements produced by the proposed approach are prioritized in a correct order. The validation of the proposed approach through mathematical theory is described in more detail in Chapter 5.

3.3 Evaluation Phase

Within the third and last phase of this research, i.e. evaluation phase, empirical studies have been carried out to evaluate the effectiveness of the proposed approach, IPA. The effectiveness of IPA has been assessed through performing two controlled experiments to compare it with the most familiar and relevant state-of-the-art alternatives, AHP-based approach and HAM-based approach. In both controlled experiments, evaluation was based on measuring three properties: actual time-consumption, accuracy of results, and ease of use. Figure 3.2 shows the process of conducting the controlled experiments, which highlights the basic activities to perform the controlled experiments.

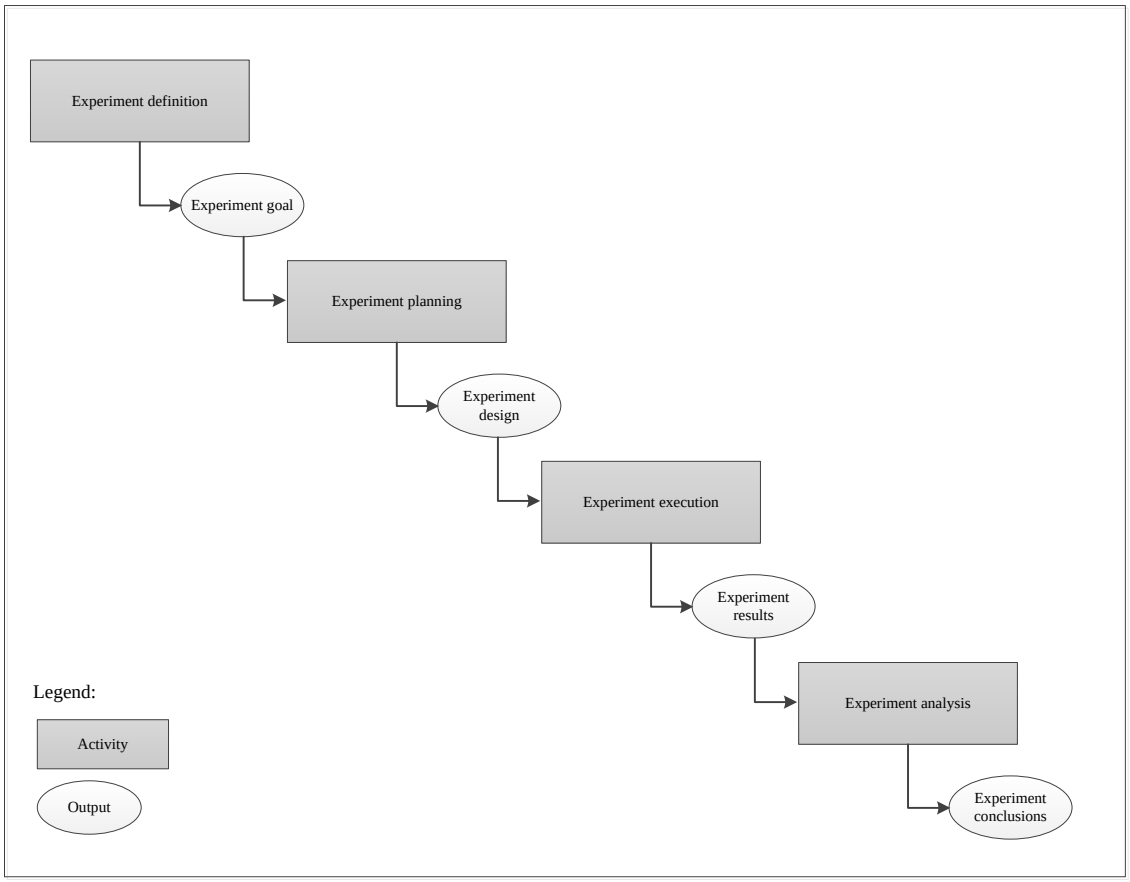


Figure 3.2: A graphical overview of controlled experiment process

According to Figure 3.2, conducting the controlled experiments within the evaluation phase of this research involved four main activities. The different activities are:

- Experiment definition
- Experiment planning
- Experiment execution
- Experiment analysis

3.3.1 Experiment Definition

This activity is related to define the controlled experiments. It specified the target and foundation of each controlled experiment. The goal of each controlled experiment was defined clearly.

3.3.2 Experiment Planning

The input for experiment planning is the experiment definition. In fact, after the experimental goal was determined, each controlled experiment needed to be planned in order to determine the design of the controlled experiment. The planning phase of each controlled experiment composed of seven steps, as depicted in Figure 3.3. These steps are further elaborated in Chapter 6.

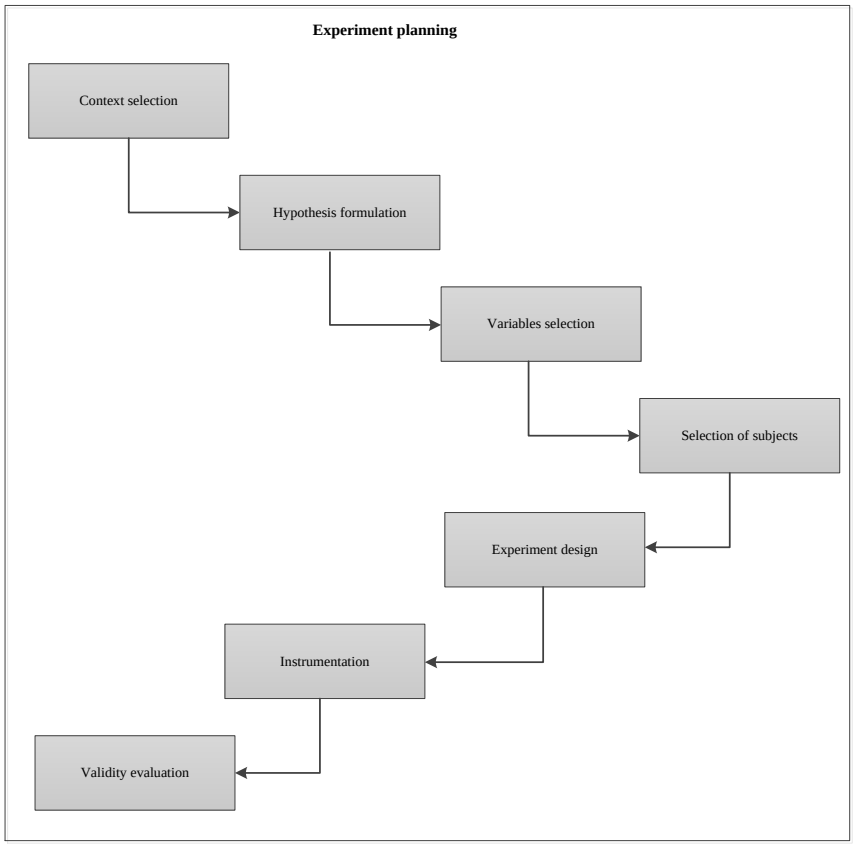


Figure 3.3: Overview of experiment planning

3.3.3 Experiment Execution

Once the controlled experiment has been planned and designed, it should be executed in order to collect the required data needed to be analysed. This is the step where the subjects of the study have to work with the proposed approach.

3.3.4 Experiment Analysis

The experimental data collected within the execution step provided the input to this activity. Consequently, as can be observed in Figure 3.4, the collected data has been interpreted and analysed in two ways: descriptive analysis and statistical analysis using hypothesis testing. The goal of performing such analysis was to make rigorous conclusions regarding the findings of the controlled experiments.

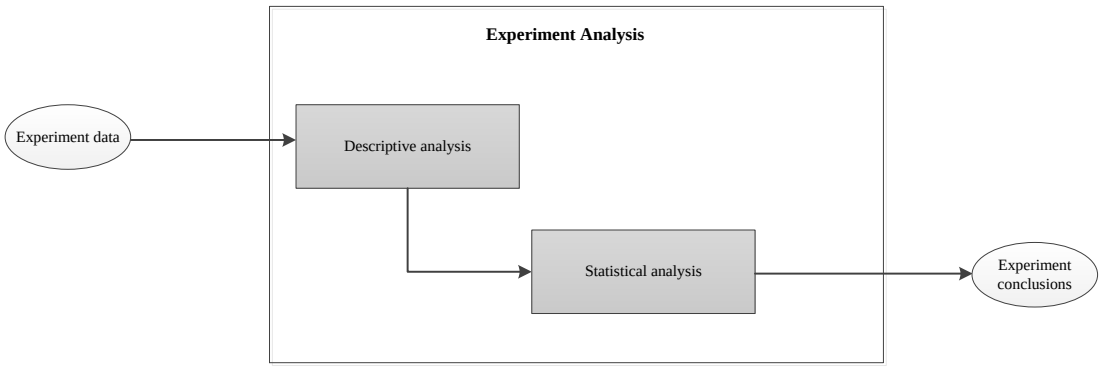


Figure 3.4: Process of experiment analysis

It should be highlighted that the detailed description of the controlled experiments which have been carried out during this phase is presented in Chapter 6.

3.4 Summary

This chapter provided a comprehensive overview of the research methodology which has been organized in the form of three phases such as preparation phase, approach development and validation phase, and evaluation phase. During the preparation phase, an extensive literature review has been performed on different requirements prioritization approaches proposed in recent years as well as various empirical studies dedicated to evaluate the existing approaches. Within the approach development and validation phase, an approach for integrating the prioritization of functional and non-functional requirements has been proposed. Moreover, in this phase, the proposed

approach has been validated in two ways including case study and mathematical theory. Finally, in the evaluation phase, the effectiveness of the proposed approach has been evaluated through conducting two controlled experiments. In both controlled experiments, the proposed approach has been compared with the other state-of-the-art alternatives in terms of actual time-consumption, accuracy of results, and ease of use.

CHAPTER 4: THE PROPOSED APPROACH

As explained in Chapter 2, over the recent past years, a substantial amount of research studies in the software engineering community have been dedicated into proposing numerous techniques, methods and approaches to perform the prioritization of software requirements (Perini et al., 2013; Thakurta, 2013; Tonella et al., 2013). A comprehensive and thorough study of current research studies insinuates that integrating the prioritization of functional and non-functional requirements has been neglected through all existing prioritization approaches proposed recently. Accordingly, to support the progression in this area and the maturity of requirements prioritization, a new approach needs to be proposed with the purpose of addressing both functional requirements along with non-functional requirements during the prioritization stage. Considering both functional and non-functional requirements is crucial during the prioritization process since these two types of requirements have great impression on each other, and not properly addressing them together in the prioritization process could ultimately result in releasing low-quality software products. In response to fulfil this demand, a new approach has been proposed throughout this research in order to integrate the process of prioritizing functional and non-functional requirements simultaneously. This approach is referred to as Integrated Prioritization Approach (IPA). This chapter basically describes the detailed description of IPA.

4.1 Integrated Prioritization Approach (IPA)

In order to integrate the prioritization of functional and non-functional requirements simultaneously, an approach has been proposed, namely, Integrated Prioritization Approach (IPA), consisting of nine steps, as shown in Table 4.1. The initial step is to identify system stakeholders who get involved in the prioritization process. The second step specifies the weight of each stakeholder using Analytic Hierarchy Process (AHP)

method. The third step is to identify functional requirements as well as non-functional requirements. The forth step is to extract functional and non-functional requirements statements. The fifth step is to set up the n functional requirements and the m non-functional requirements in the rows and columns of an $n \times m$ decision matrix. The sixth step performs the elicitation of the importance degree of each non-functional requirement with respect to each functional requirement with the aim of creating linkage between functional and non-functional requirements. The seventh step is an aggregation procedure to specify the prioritization of non-functional requirements ranking with respect to all functional requirements using triangular fuzzy number and alpha cut approach. This step provides a decision maker with a prioritized list of non-functional requirements with respect to all functional requirements. The eighth step produces a prioritized list of functional requirements by calculating the functional requirements priority vector and the respective normalized weights. The ninth and final step concludes the process by aggregating different prioritized lists of functional and non-functional requirements provided by various stakeholders to obtain final rankings of functional and non-functional requirements.

Table 4.1: Steps of IPA for integrating the prioritization of functional and non-functional requirements

Step #	**Description**
1	Identify stakeholders of software system
2	Specify the weights of stakeholders using Analytic Hierarchy Process
3	Identify functional and non-functional requirements
4	Extract functional and non-functional requirements statements
5	Construct the decision matrix
6	Elicit the importance degree of each non-functional requirement with respect to each functional requirement to establish the relationship between functional and non-functional requirements
7	Calculate NFRs ranking with respect to all FRs using triangular fuzzy number and alpha cut approach
8	Compute FRs ranking using weighted average decision matrix and weights determined in Step 7
9	Aggregate different prioritized lists of FRs and NFRs provided by various stakeholders to obtain final rankings of FRs and NFRs

Before going through the details of each step, it should be highlighted that IPA supports both single and group decision-making. In other words, the proposed approach, IPA, has the flexibility to apply whether in single or group decision-making problems. In single decision-making problems, only one stakeholder (i.e. decision maker) is involved in the prioritization process, whereas within group decision-making problems, several stakeholders may take part in the prioritization process.

If IPA is applied in any single decision-making problems, Step 1, Step 2, and Step 9 of the proposed approach should be skipped. Hence, in those situations, as illustrated in Figure 4.1A, the process would be started by Step 3, following by Step 4, Step 5, Step 6, Step 7 and finally will be terminated by Step 8. On the other hand, in group decision-making problems, as can be seen in Figure 4.1B, all of the steps (i.e. Step 1 to Step 9) must be accomplished consecutively.

The algorithm, given as pseudo-code in Figure 4.2, has been used as a basis for developing a software prototype to support the automation of the proposed approach. More information regarding the software prototype is provided in Appendix A.

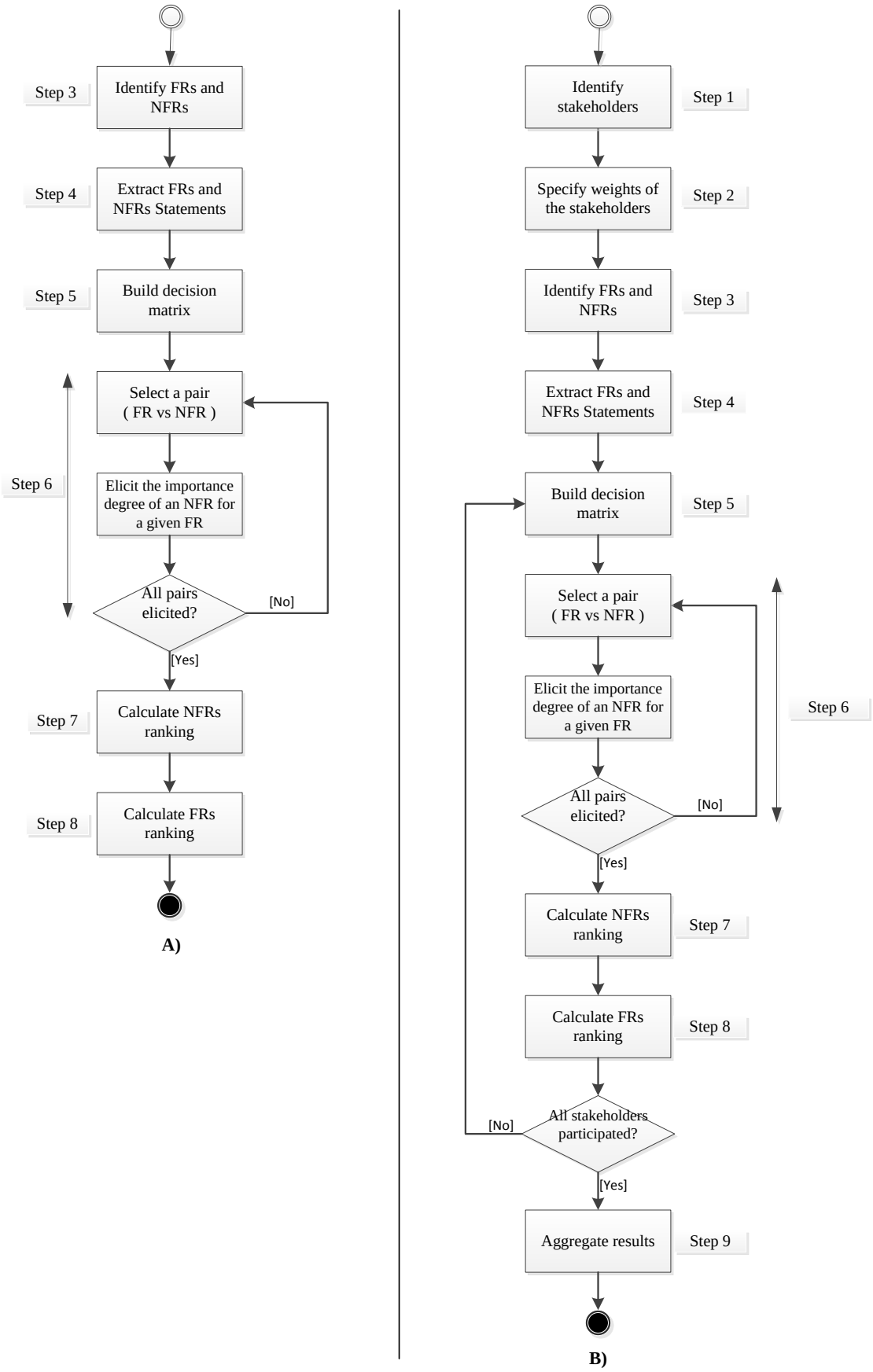


Figure 4.1: Flow charts of the proposed approach when applied A) in single decision-making problems; B) in group decision-making problems

Input
$\boldsymbol{FRs} = \{\boldsymbol{FR1_1}, \boldsymbol{FR_2}, \boldsymbol{FR_3}, \dots, \boldsymbol{FR_n}\}$
//a set of functional requirements
$\boldsymbol{NFRs} = \{\boldsymbol{NFR_1}, \boldsymbol{NFR_2}, \boldsymbol{NFR_3}, \dots, \boldsymbol{NFR_m}\}$
//a set of non-functional requirements
$\boldsymbol{S} = \{\boldsymbol{S_1}, \boldsymbol{S_2}, \boldsymbol{S_3}, \dots, \boldsymbol{S_k}\}$
//a set of stakeholders
Output
$\boldsymbol{P(FRs)}$
//prioritized list of FRs
$\boldsymbol{P(NFRs)}$
//prioritized list of NFRs
Begin

```
1:    N = NumberOfFunctionalRequirements
      //definition of parameter for the number of functional requirements
2:    M = NumberOfNonFunctionalRequirements
      //definition of parameter for the number of non-functional requirements
3:    K = NumberOfStakeholders
      //definition of parameter for the number of stakeholders
4:    If K > 1
5:        W(S) = AHP(S)
          //specifying the weights of stakeholders using AHP method
6:    Endif
7:    For k = 1 To K
          M = GenerateMatrix(N, M)
          //constructing the decision matrix with N rows and M columns
8:        For n = 1 To N
9:            For m = 1 To M
10:               M[n, m] = ImportanceDegree(NFR(m), FR(n))
                  //elicitation of the importance degree of NFR for with respect to each FR
11:           Endfor
12:       Endfor
13:       P(NFRs) = ComputRanking1(NFRs)
14:       //computation of NFRs ranking using triangular fuzzy number and alpha cut approach
15:       P(FRs) = ComputRanking2(FRs)
16:       // computation of FRs ranking using weighted average decision matrix
17:   Endfor
18:   If K > 1
19:       P(NFRs) = AggregateResultOfNFRs
20:       //aggregating different NFRs rankings provided by different stakeholders
21:       P(FRs) = AggregateResultOfFRs
22:       // aggregating different FRs rankings provided by different stakeholders
23:   Endif
24:   return P(FRs) and P(NFRs)
End
```

Figure 4.2: A sketch of algorithm used for automating the IPA

As can be observed in Table 4.1, using IPA to prioritize functional and non-functional requirements involves nine steps. The detailed explanation of these steps is illustrated in the following:

4.1.1 Step 1: Identify stakeholders of software system

Identification of stakeholders is a crucial activity associated with requirements elicitation process (Pacheco & Garcia, 2009). Hence, the initial step of the proposed approach is primarily concerned with identification of system stakeholders. During this step, the main stakeholders who may possibly take part in the prioritization process are recognized. To illustrate the proposed approach, the identified stakeholders are represented as S_1, S_2, S_3, …, S_k.

4.1.2 Step 2: Specify the weights of stakeholders using Analytic Hierarchy Process

As mentioned in the previous step, different stakeholders might participate in the prioritization process. These stakeholders could have different weights according to their significance on the final prioritized lists of functional and non-functional requirements. In the proposed approach, in order to perform the weighting process, Analytic Hierarchy Process (AHP) (J. Karlsson & Ryan, 1997) has been utilised to get the relative importance between different stakeholders (these stakeholders are identified in Step 1). AHP has been selected as a method for weighting the stakeholders due to the fact that the outcome that would be generated by this method is accurate, reliable, and trustworthy (Achimugu et al., 2014).

AHP as the most widely known MCDM (Multi Criteria Decision Making) method, exploits pairwise comparison strategy, allowing requirements engineers to compare all the available pairs of stakeholders together to figure out the weight of one stakeholder over another stakeholder. Having a collection of n stakeholders identified in Step 1, the first step in AHP is to construct an $n \times n$ matrix in which rows and columns indicate the available stakeholders. Then, the requirements engineer determines his/her judgment for each pair of stakeholders by identifying a preference value which is between one to

nine, where one expresses that the two stakeholders are equally important while nine represents the highest value of one stakeholder when compared to the other stakeholder. In fact, the requirements engineer needs to perform $n * (n-1)/2$ pairwise comparisons in total. The underlying values used for this purpose are presented in Table 4.2, which indicates a measure of specifying the requirements engineer's preference value for a given pair of stakeholders. Once all the possible pairs of stakeholders have been assessed, the final weight of each stakeholder is calculated throughout the calculation of the principal eigenvector of the matrix (i.e., the eigenvector with the greatest normalized eigenvalue). Each element of the principal eigenvector signifies the weight of the related stakeholder. The calculated weights of stakeholders using AHP are represented as $W_{s_1}, W_{s_2}, W_{s_3}, \ldots, W_{s_k}$.

Table 4.2: Possible scales used for AHP's pairwise comparison (J. Karlsson & Ryan, 1997)

Relative Intensity	**Definition**	**Explanation**
1	Of equal value	Two stakeholders are of equal value
3	Slightly more value	Experience slightly favors one stakeholder over another
5	Essential or strong value	Experience strongly favors one stakeholder over another
7	Very strong value	A stakeholder is strongly favored and its dominance is demonstrated in practice
9	Extreme value	The evidence favoring one over another is of the highest possible order of affirmation
2,4,6,8	Intermediate values between two adjacent judgments	When compromise is needed
Reciprocals	if stakeholder *i* has one of the above numbers assigned to it when compared with stakeholder *j*, then *j* has the reciprocal value when compared with *i.*	

4.1.3 Step 3: Identify functional and non-functional requirements

The third step of IPA is to identify the functional and non-functional requirements which are required to be prioritized for inclusion in a software system. Let n be the number of candidate functional requirements and m the number of candidate non-functional requirements. For demonstration of the process, we suppose that we have n candidate functional requirements: FR_1, FR_2, FR_3, …, FR_n, and m non-functional requirements: $NFR_1, NFR_2, NFR_3, \ldots, NFR_m$, which need to be ranked using IPA.

4.1.4 Step 4: Extract functional and non-functional requirements statements

In this step, in order to reach agreement among different stakeholders as well as avoid misunderstanding between them, both functional and non-functional requirements which were identified in Step 3, must be expressed in a structured and consistent way. Therefore, to define each functional requirement statement, a canonical form is provided and structured as follows:

<S> shall be able to <Functionality>.
must be able to

Where <S> is the entity to which this requirement applies such as stakeholder or system; <Functionality> is the main function of a functional requirement which was defined during the previous step and it can be represented by one of the following structures:

1. [V]
2. [V]+[O]
3. [V]+[C]
4. [V]+[A]
5. [V]+[O]+[A], where V is verb; O is object; C is Complement; A is adverb.

On the other hand, to extract non-functional requirements statements, the proposed approach offers three canonical forms as follows:

1. <non-functional attribute>: <S> shall be able to <Functionality> <constraint>.
must be able to

Where <non-functional attribute> includes, but not limited to Performance, Security, Availability, or Reliability; <constraint> is the condition of satisfying the requirement.

The other two ways for acquiring non-functional requirements statements are to integrate non-functional attributes (quality attributes) with <Functionality> as follows (Sadana & Liu, 2007):

2. <non-functional attribute> of <Functionality> shall be <constraint>.
must be

3. <Functionality> shall have <constraint><non-functional attribute>.
must have

4.1.5 Step 5: Construct the decision matrix

The fifth step of the proposed approach, IPA, is to generate an $n \times m$ decision matrix, namely, D, and insert the n functional requirements along with m non-functional requirements in the rows and columns of the decision matrix, respectively. Therefore, in this step, an $n \times m$ matrix is constructed, as shown in matrix D (4.1). The instructions on how to fill up the elements of the matrix D will be described in Step 6.

$$D = \begin{array}{c|cccccccc} & NFR_1 & NFR_2 & NFR_3 & NFR_4 & . & . & . & NFR_m \\ FR_1 & D_{11} & D_{12} & D_{13} & D_{14} & - & - & - & D_{1m} \\ FR_2 & D_{21} & D_{22} & D_{23} & D_{24} & - & - & - & D_{2m} \\ FR_3 & D_{31} & D_{32} & D_{33} & D_{34} & - & - & - & D_{3m} \\ FR_4 & D_{41} & D_{42} & D_{43} & D_{44} & - & - & - & D_{4m} \\ . & - & - & - & - & - & - & - & - \\ . & - & - & - & - & - & - & - & - \\ . & - & - & - & - & - & - & - & - \\ FR_n & D_{n1} & D_{n2} & D_{n3} & D_{n4} & - & - & - & D_{nm} \end{array} \quad (4.1)$$

4.1.6 Step 6: Elicit the importance degree of each non-functional requirement with respect to each functional requirement to establish the relationship between functional and non-functional requirements

As mentioned before, non-functional requirements have an impact on functional requirements. Accordingly, the main purpose of performing this step is to elicit this impact value. So, the sixth step of the proposed approach involves eliciting the decision maker's opinions for determining the importance degree of each non-functional requirement for a given functional requirement. To elicit such an extent, IPA uses two scales: nominal scale, and actual scale.

Nominal scale is an interface scale which is utilized in order to enhance the user-friendliness of IPA for interacting with decision makers so that the decision maker would not be aware of details regarding the actual scale. On the other side, the actual scale is a numerical scale which is used for internal calculations within IPA. In fact, IPA exploits a five-point scale as actual scale. Table 4.3 demonstrates these scales.

Table 4.3: IPA nominal scale, IPA actual scale

IPA nominal scale	IPA actual scale
Very high importance (VHI)	1
High importance (HI)	0.75
Low importance (LI)	0.5
Very low importance (VLI)	0.25
Negligible (NI)	0.001

For each pair of functional and non-functional requirements (selected from rows and columns of matrix D, respectively), the decision maker fulfils a number of activities to determine the importance degree of each non-functional requirement for achieving each associated functional requirement (look at activities "select a pair" and "elicit the importance degree of an non-functional requirement for a given functional requirement" in Figure 4.1). Each pair (i.e., functional requirement and non-functional requirement) is assigned a value belonging to the IPA nominal scale (see left column of Table 4.3) which represents a qualitative measure of importance relation between the corresponding functional and non-functional requirements. So, for all i and j with $1 \leq i \leq n$ and $1 \leq j \leq m$, the importance degree of the jth non-functional requirement for achieving the ith functional requirement would be assessed by a decision maker, leading to the value D_{ij} using Table 4.3 (e.g. if the value of D_{23}= "very high importance", it means that the decision maker believed that the non-functional requirement NFR3 has "very high importance" impact for achieving functional requirement FR2).

By accomplishing this step, the relationships between all pairs of functional and non-functional requirements are extracted. In addition, all elements of decision matrix, D, are filled.

4.1.7 Step 7: Calculate NFRs ranking with respect to all FRs using triangular fuzzy number and alpha cut approach

When all pairs of functional and non-functional requirements have been evaluated, IPA carries out the priority assessment of non-functional requirements with respect to all functional requirements, to obtain the weights for Step 8, using triangular fuzzy number (TFN) and alpha cut approach. In fact, during this step, IPA creates a prioritized list of non-functional requirements through calculating the total importance degree of each non-functional requirement with respect to all associated functional requirements. The rational behind this idea is that a non-functional requirement which achieves the highest total importance degree among all associated functional requirements could be assigned as a high-priority non-functional requirement. The following sub-steps illustrate a stepwise process of computing the priority vector of non-functional requirements:

Sub-step 1: Convert the elements of matrix D into numerical values

First, IPA converts all values of the decision matrix D, which were specified according to the nominal scale, into the corresponding actual scales, resulting in the matrix Ď.

Sub-step 2: Set up triangular fuzzy numbers (TFNs)

Until now, various types of fuzzy numbers have been presented by researchers which could be useful throughout the decision-making process. They are including Trapezoidal, Triangular, Sigmoid, and Gaussian (Mahmood, Ahmadi, Verma, Srividya,

& Kumar, 2013). In this study, triangular fuzzy number is adopted since it is the most popular fuzzy number among the various shapes of fuzzy numbers. It has been widely used in practical solutions, and it is also simple in terms of computation and concept.

Here, in order to aggregate the different importance degrees of each non-functional requirement for different functional requirements, the triangular fuzzy number (TFN) is calculated. TFN has an ability to aggregate the different opinions of a decision maker by means of fuzzy set theory. The triangular fuzzy number T_{x_i} is represented using the following equations ((4.2) and (4.3)):

$$T_{x_i} = (L_{x_i}, M_{x_i}, H_{x_i}),\ i = 1..m, \text{ and } L_{x_i}, M_{x_i}, H_{x_i} \in [0.001,1] \quad (4.2)$$

$$M_{x_i} = \sqrt[n]{D_{x_i}a.\, D_{x_i}b.\, D_{x_i}c \ldots D_{x_i}n} \quad (4.3)$$

where T_{x_i} indicates the triangular fuzzy number of non-functional requirement "x_i" ; L_{x_i} and H_{x_i} represent the lowest and highest values of non-functional requirement "x_i" respectively; M_{x_i} is generated by calculating the geometric mean of all values belonging to the non-functional requirement "x_i" (see Equation (4.3)); m is the total number of non-functional requirements; n is the total number of functional requirements; and $D_{x_i}a$ specifies an opinion of a decision maker toward the importance degree of the non-functional requirement "x_i" for achieving the functional requirement "a".

Sub-step 3: Constructing the fuzzy priority vector

After calculating the TFN value for each non-functional requirement, the fuzzy priority vector, namely, $\widetilde{F_x}$ is generated, as illustrated in Figure 4.3. Notice, the values of $\widetilde{F_x}$ are derived from Equation (4.2).

x_1	x_2	....	x_m
T_{x_1}	T_{x_2}	....	T_{x_m}

Figure 4.3: Fuzzy priority vector, $\widetilde{F_x}$.

Sub-step 4: Defuzzification process

IPA exploits the alpha cut approach proposed by (Liou & Wang, 1992), as shown in the Equation (4.4), to perform the defuzzification process. The defuzzification is accomplished in order to convert the calculated TFN values into quantifiable values, leading to the priority vector W.

$$\mathrm{W}\left(\tilde{F}_{x_i}\right) = [\beta \times f_{\alpha}\left(L_{x_i}\right) + (1-\beta) \times f_{\alpha}\left(H_{x_i}\right)], 0 \leq \alpha, \beta \leq 1 \qquad (4.4)$$

where $\mathrm{f}_{\alpha}\left(\mathrm{L}_{\mathrm{x_i}}\right) = \left(\mathrm{M}_{\mathrm{x_i}} - \mathrm{L}_{\mathrm{x_i}}\right) \times \alpha + \mathrm{L}_{\mathrm{x_i}}$, which indicates the left-end border value of alpha cut for $\tilde{\mathrm{F}}_{\mathrm{x_i}}$; and $\mathrm{f}_{\alpha}\left(\mathrm{H}_{\mathrm{x_i}}\right) = \mathrm{H}_{\mathrm{x_i}} - \left(\mathrm{H}_{\mathrm{x_i}} - \mathrm{M}_{\mathrm{x_i}}\right) \times \alpha$, which shows the right-end border value of alpha cut for $\tilde{\mathrm{F}}_{\mathrm{x_i}}$.

In this case, α and β correspond to preferences and risk tolerance of decision maker, respectively. Both of these values vary in range 0 and 1, in such a way that a lower value implies higher uncertainty in decision making. Due to the fact that preferences as well as risk tolerance are not the main issues of this study, a value of 0.5 is used for both α and β to indicate a balanced setting. This represents that the decision maker is neither very optimistic nor pessimistic of his/her viewpoints.

Finally, by normalizing the calculated priority vector, W, the vector NW of normalized weights is obtained using the following equation (4.5):

$$NW_j = \frac{w_j}{\sum_{j=1}^{m} w_j} \qquad (4.5)$$

By applying the steps stated above, a decision maker is provided with a prioritized list of non-functional requirements along with their corresponding importance values with respect to all existing functional requirements.

4.1.8 Step 8: Compute FRs ranking using weighted average decision matrix and weights determined in Step 7

During the previous steps, the priority value of each non-functional requirement with respect to all functional requirements was obtained (i.e. NW in Step 7). Furthermore, the importance degree of each non-functional requirement with regards to every individual functional requirement was elicited (i.e. elements of the matrix Ď). By gathering such data, the weighted average decision matrix, as indicated in Table 4.4, is generated in order to assist the process of calculating the priority vector of functional requirements (according to their relations with non-functional requirements).

Table 4.4: Weighted average decision matrix for priority assessment of FRs

$NFRs' Weights$	NW_1	NW_2	NW_3	NW_m
	NFR_1	NFR_2	NFR_3	NFR_m
FR_1	$\acute{D_{11}}$	$\acute{D_{12}}$	$\acute{D_{13}}$	$\acute{D_{1m}}$
FR_2	$\acute{D_{21}}$	$\acute{D_{22}}$	$\acute{D_{23}}$	$\acute{D_{2m}}$
....	....	....	....	....
FR_n	$\acute{D_{n1}}$	$\acute{D_{n2}}$	$\acute{D_{n3}}$	$\acute{D_{nm}}$

The geometric means need to be computed to perform the aggregation process which has been used to determine the final ranking of functional requirements in this step of IPA. For computing the geometric means, the researcher has used the calculated normalized priority vector of non-functional requirements (see *Sub-step 4*) for its weights, leading to the priority vector R, as represented using the following equation (4.6):

$$R_i = \prod_{j=1}^{m} \acute{D}_{ij}^{\,NW_j}, i = 1..n \quad (4.6)$$

Then the obtained vector R is normalized, giving the normalized priority vector of functional requirements, NR, to ensure that the final ranking values will be between 0 and 1:

$$NR_i = \frac{r_i}{\sum r_i} \quad (4.7)$$

The decreasing ordered functional requirements indicate the final ranking, where the most important functional requirement is the one with the highest NR value.

4.1.9 Step 9: Aggregate different prioritized lists of FRs and NFRs provided by various stakeholders to obtain final rankings of FRs and NFRs

As discussed before, different stakeholders may possibly participate in the prioritization process. Each of these stakeholders might have his/her individual opinion regarding the final prioritized lists of functional and non-functional requirements. On the other hand, each of these stakeholders could have different weights based on how important they have been recognized for a specific prioritization problem. Hence, the question on how to aggregate and combine different results provided by different stakeholders regarding the final prioritized lists of functional and non-functional requirements could be a challenge.

In the proposed approach, Weighted Average (WA) method (McZara, Sarkani, Holzer, & Eveleigh, 2014) has been applied in order to aggregate different prioritized lists of functional and non-functional requirements which were provided by different stakeholders.

Therefore, to obtain the single prioritized list of functional requirements from different prioritized lists of functional requirements provided by different stakeholders, the weighted average matrix needs to be generated as indicated in Table 4.5.

Table 4.5: Weighted average matrix to aggregate different prioritized list of FRs

$\boldsymbol{Stakeholders' Weights}$	$\boldsymbol{W_{s_1}}$	$\boldsymbol{W_{s_2}}$	$\boldsymbol{W_{s_3}}$	$\boldsymbol{W_{s_k}}$
FRs	$\boldsymbol{S_1}$	$\boldsymbol{S_2}$	$\boldsymbol{S_3}$	$\boldsymbol{S_k}$
$\boldsymbol{FR_1}$	NR_{11}	NR_{12}	NR_{13}	NR_{1k}
$\boldsymbol{FR_2}$	NR_{21}	NR_{22}	NR_{23}	NR_{2k}
….	….	….	….	….
$\boldsymbol{FR_n}$	NR_{n1}	NR_{n2}	NR_{n3}	NR_{nk}

Then, the following equation (4.8) is used to arrive at a final weight for each functional requirement associated with the prioritization problem:

$$UR_i = \sum_{j=1}^{k} NR_{ij} * W_{s_j} \, , i = 1..n \tag{4.8}$$

where UR_i indicates the final weight of functional requirement "R_i" ; n is the total number of functional requirements; k is the total number of stakeholders; NR_{ij} specifies the normalized weight of functional requirement "R_i" obtained by stakeholder j; and W_{s_j} is the weight of stakeholder j. It should be noted that the values of NR_{ij} and W_{s_j} have already computed through Step 8 and Step 2, respectively.

The same process needs to be done in order to produce the single prioritized list of non-functional requirements. So, the weighted average matrix is generated again, as shown in Table 4.6, but here for non-functional requirements.

Table 4.6: Weighted average matrix to aggregate different prioritized list of NFRs

$\boldsymbol{Stakeholders' Weights}$	$\boldsymbol{W_{s_1}}$	$\boldsymbol{W_{s_2}}$	$\boldsymbol{W_{s_3}}$	$\boldsymbol{W_{s_k}}$
NFRs	$\boldsymbol{S_1}$	$\boldsymbol{S_2}$	$\boldsymbol{S_3}$	$\boldsymbol{S_k}$
$\boldsymbol{NFR_1}$	NW_{11}	NW_{12}	NW_{13}	NW_{1k}
$\boldsymbol{NFR_2}$	NW_{21}	NW_{22}	NW_{23}	NW_{2k}
….	….	….	….	….
$\boldsymbol{NFR_m}$	NW_{m1}	NW_{m2}	NW_{m3}	NW_{mk}

Then, the following equation (4.9) can be used to obtain a final weight for each non-functional requirement associated with the prioritization problem:

$$UW_i = \sum_{j=1}^{k} NW_{ij} * W_{s_j} \, , i = 1..m \tag{4.9}$$

where UW_i indicates the final weight of non-functional requirement "W_i"; m is the total number of non-functional requirements; k is the total number of stakeholders; NW_{ij} specifies the normalized weight of non-functional requirement "W_i" obtained by stakeholder j; and W_{s_j} is the weight of stakeholder j. It should be noted that the values of NW_{ij} and W_{s_j} have already calculated through Step 7 and Step 2, respectively.

4.2 Summary

This chapter introduced the approach which has been proposed in this research for integrating the prioritization of functional and non-functional requirements. The proposed approach, called IPA, can be applied on a prioritization problem where one or several stakeholders may take part in the prioritization process. Therefore, it has the flexibility to be used in both single and group decision-making process.

CHAPTER 5: VALIDATION OF THE PROPOSED APPROACH

This chapter presents the detailed descriptions regarding the validation of the proposed approach (i.e. IPA) through case study (Section 5.1) and mathematical theory (Section 5.2).

5.1 Validate the Proposed Approach through Case Study

An intuitive comprehension of the proposed approach can be achieved by applying the IPA to a case study, step by step, to demonstrate how the nine steps of the IPA could be utilized for a practical prioritization problem. Therefore, IPA has been applied on a collection of 15 functional requirements and 5 non-functional requirements of Automated Teller Machine (ATM), Cash Deposit Machine (CDM), and Check Deposit Machine (CQM). Four users of ATM, CDM, and CQM, who played the role of stakeholders, participated in the case study. The detailed explanation of applying IPA on the requirements of ATM, CDM, and CQM is illustrated in the following:

Step 1: Identify stakeholders of software system

To initiate the process, four stakeholders participated in the prioritization process. All these stakeholders play the role of users of the banking software system. These stakeholders are represented as S_1, S_2, S_3, and S_4.

Step 2: Specify the weights of stakeholders using Analytic Hierarchy Process

The four identified stakeholders could have different weights according to their significance on the final prioritized lists of functional and non-functional requirements. In this step, AHP method is applied on the identified stakeholders in order to obtain their weights. Using AHP for performing the weighting process involves four steps.

1. *Insert the four stakeholders in the rows and columns of a* 4×4 *matrix*. The four stakeholders are inserted into the rows and columns of a matrix of order 4. So, in this case, a 4×4 matrix is generated.

2. *Exploit pairwise comparisons of all the stakeholders*. The basic scales used in AHP for pairwise comparisons are demonstrated in Table 4.2. Given a pair of stakeholders (e.g. S_1 and S_2), put the relative importance value of S_1 and S_2 in the position (S_1,S_2) where the row of S_1 joins the column of S_2. In position (S_2,S_1) put the reciprocal value, and in all positions in the main diagonal insert a "1". As mentioned before, for a matrix of order n, $n * (n-1)/2$ pairwise comparisons are needed in total. So, in this case, six pairwise comparisons are needed. In this example, each of the four stakeholders is assigned a different importance value. So, all values of the matrix are selected based on the scales presented in Table 4.2. As a result, a 4×4 pairwise matrix is formed like this:

$$\begin{array}{c|cccc} & \boldsymbol{S_1} & \boldsymbol{S_2} & \boldsymbol{S_3} & \boldsymbol{S_4} \\ \boldsymbol{S_1} & 1 & 1/2 & 2 & 3 \\ \boldsymbol{S_2} & 2 & 1 & 1/2 & 1/2 \\ \boldsymbol{S_3} & 1/2 & 2 & 1 & 1/3 \\ \boldsymbol{S_4} & 1/3 & 2 & 3 & 1 \end{array} \quad (5.1)$$

3. *Use averaging over normalized columns to estimate the eigenvalues of the matrix.* The method used in this step is called averaging over normalized columns, originally proposed by Thomas Saaty (Saaty, 1980). First, compute the summation of the n columns in the pairwise comparison matrix. Then, divide each element of the pairwise comparison matrix by the summation of the column that element belongs to, and finally compute the summation of each row:

	S_1	S_2	S_3	S_4	SUM
S_1	0.26	0.09	0.31	0.62	1.28
S_2	0.52	0.18	0.08	0.10	0.88
S_3	0.13	0.36	0.15	0.07	0.72
S_4	0.09	0.36	0.46	0.21	1.12

(5.2)

Then, in order to normalize the summation of each row, the summation of each row is divided by the number of stakeholders. The outcome of this computation is an estimation of the eigenvalues of the pairwise comparison matrix.

$$\frac{1}{4} \cdot \begin{bmatrix} 1.28 \\ 0.88 \\ 0.72 \\ 1.12 \end{bmatrix} = \begin{bmatrix} 0.32 \\ 0.22 \\ 0.18 \\ 0.28 \end{bmatrix} \quad (5.3)$$

4. *Assign each stakeholder to its relative weight according to the estimated eigenvalues.* By computing the estimated eigenvalues of the pairwise comparison matrix, the relative weights of four stakeholders are achieved as follows:
    - W_{s_1} has the weight of 0.32.
    - W_{s_2} has the weight of 0.22.
    - W_{s_3} has the weight of 0.18.
    - W_{s_4} has the weight of 0.28.

Step 3: Identify functional and non-functional requirements

The third step of IPA is to identify the functional and non-functional requirements of ATM, CDM, and CQM. Accordingly, 15 functional requirements and 5 non-functional requirements are identified to be prioritized. The identified functional and non-functional requirements are listed in Table 5.1 and Table 5.2, respectively.

Table 5.1: The functional requirements of the ATM, CDM, and CQM

Functional requirement ID	Functional requirement description
$\mathbf{FR_1}$	Withdraw cash
$\mathbf{FR_2}$	Check balance
$\mathbf{FR_3}$	Deposit cash
$\mathbf{FR_4}$	Transfer funds
$\mathbf{FR_5}$	Change PIN number
$\mathbf{FR_6}$	View transactions history
$\mathbf{FR_7}$	Bill payment
$\mathbf{FR_8}$	Print transaction receipt
$\mathbf{FR_9}$	Deposit cheque
$\mathbf{FR_{10}}$	Top up your mobile phone
$\mathbf{FR_{11}}$	Loan payment
$\mathbf{FR_{12}}$	Print transaction history
$\mathbf{FR_{13}}$	Change withdraw limit
$\mathbf{FR_{14}}$	Activate overseas services
$\mathbf{FR_{15}}$	Credit card payment

Table 5.2: The non-functional requirements of the banking software system

NFR ID	Name	Description
$\mathbf{NFR_1}$	Availability	The percentage of time that the software system is in operation to provide its intended function.
$\mathbf{NFR_2}$	Security	The extent to which access to the desired function by unauthorized persons can be controlled while still providing its function to users.
$\mathbf{NFR_3}$	Usability	The extent to which a user is able to understand, learn, use and being attracted to a function.
$\mathbf{NFR_4}$	Performance	The extent to which how fast the system can interact with the user to perform the desired function.
$\mathbf{NFR_5}$	Reliability	The extent to which the system can be expected to perform its intended function with required precision.

Step 4: Extract functional and non-functional statements

During this step, first, each functional requirement is redefined according to the canonical form introduced in the proposed approach. The total number of functional requirements was thus 15. Examples of such functional requirements statements are:

"FRS1: Bank customer shall be able to withdraw cash."

"FRS2: Bank customer shall be able to check balance."

"FRS3: Bank customer shall be able to deposit cash."

In addition, examples of extracted non-functional requirements statements could be as below.

"Usability: Bank customer shall be able to deposit cash easily."

"Performance: Bank customer shall be able to withdraw cash in less than 3 seconds."

"Reliability of check balance must be high."

Step 5: Construct the decision matrix

Here, within this step, four decision matrixes of 15×5 are generated. Then, 15 identified functional and 5 identified non-functional requirements are inserted into the rows and columns of the four separate decision matrixes, respectively. Four decision matrixes are constructed because four stakeholders participated in the prioritization process.

Step 6: Elicit the importance degree of each non-functional requirement with respect to each functional requirement to establish the relationship between functional and non-functional requirements

To fill up the elements of the decision matrixes constructed in Step 5, the judgments of four stakeholders are elicited and inserted into the matrixes, D_1, D_2, D_3, and D_4. These values are given in Table 5.3. It should be noted that numbers within parenthesis represent the actual scale values.

Table 5.3: Filling up the four decision matrixes with nominal scale values

Decision matrix	Functional requirements	Non-functional requirements				
		NFR_1	NFR_2	NFR_3	NFR_4	NFR_5
D_1	FR_1	VHI (1)	VHI (1)	VHI (1)	VHI (1)	VHI (1)
	FR_2	HI (0.75)	VHI (1)	HI (0.75)	HI (0.75)	VHI (1)
	FR_3	HI (0.75)	VHI (1)	VHI (1)	VHI (1)	VHI (1)
	FR_4	HI (0.75)	VHI (1)	VHI (1)	VHI (1)	VHI (1)
	FR_5	LI (0.5)	VHI (1)	HI (0.75)	HI (0.75)	VHI (1)
	FR_6	HI (0.75)	VHI (1)	HI (0.75)	HI (0.75)	VHI (1)
	FR_7	HI (0.75)	VHI (1)	HI (0.75)	HI (0.75)	VHI (1)
	FR_8	HI (0.75)	HI (0.75)	HI (0.75)	HI (0.75)	VHI (1)
	FR_9	LI (0.5)	VHI (1)	HI (0.75)	HI (0.75)	VHI (1)
	FR_{10}	HI (0.75)	VHI (1)	HI (0.75)	HI (0.75)	VHI (1)
	FR_{11}	LI (0.5)	VHI (1)	HI (0.75)	HI (0.75)	VHI (1)
	FR_{12}	LI (0.5)	VHI (1)	HI (0.75)	HI (0.75)	VHI (1)
	FR_{13}	LI (0.5)	VHI (1)	HI (0.75)	HI (0.75)	VHI (1)
	FR_{14}	VLI (0.25)	VHI (1)	HI (0.75)	HI (0.75)	VHI (1)
	FR_{15}	LI (0.5)	VHI (1)	HI (0.75)	HI (0.75)	VHI (1)
D_2	FR_1	HI (0.75)	LI (0.5)	LI (0.5)	HI (0.75)	VHI (1)
	FR_2	LI (0.5)	LI (0.5)	HI (0.75)	LI (0.5)	HI (0.75)
	FR_3	HI (0.75)	HI (0.75)	HI (0.75)	VHI (1)	VHI (1)
	FR_4	LI (0.5)	VHI (1)	HI (0.75)	HI (0.75)	VHI (1)
	FR_5	VLI (0.25)	LI (0.5)	VHI (1)	LI (0.5)	LI (0.5)
	FR_6	LI (0.5)	VLI (0.25)	LI (0.5)	HI (0.75)	HI (0.75)
	FR_7	HI (0.75)	HI (0.75)	HI (0.75)	HI (0.75)	VHI (1)
	FR_8	LI (0.5)	VLI (0.25)	VLI (0.25)	LI (0.5)	LI (0.5)
	FR_9	HI (0.75)	HI (0.75)	HI (0.75)	HI (0.75)	VHI (1)
	FR_{10}	VHI (1)	LI (0.5)	HI (0.75)	HI (0.75)	VHI (1)
	FR_{11}	VLI (0.25)	LI (0.5)	LI (0.5)	LI (0.5)	HI (0.75)
	FR_{12}	VLI (0.25)	LI (0.5)	LI (0.5)	HI (0.75)	HI (0.75)
	FR_{13}	HI (0.75)	HI (0.75)	HI (0.75)	HI (0.75)	VHI (1)
	FR_{14}	VLI (0.25)	VLI (0.25)	LI (0.5)	VLI (0.25)	HI (0.75)
	FR_{15}	HI (0.75)	HI (0.75)	LI (0.5)	LI (0.5)	VHI (1)
D_3	FR_1	HI (0.75)	VHI (1)	HI (0.75)	VHI (1)	VHI (1)
	FR_2	HI (0.75)	VHI (1)	HI (0.75)	HI (0.75)	HI (0.75)
	FR_3	VHI (1)	VHI (1)	HI (0.75)	VHI (1)	HI (0.75)
	FR_4	VHI (1)	VHI (1)	HI (0.75)	VHI (1)	VHI (1)
	FR_5	HI (0.75)	VHI (1)	VHI (1)	LI (0.5)	HI (0.75)
	FR_6	LI (0.5)	VHI (1)	HI (0.75)	LI (0.5)	LI (0.5)
	FR_7	HI (0.75)	HI (0.75)	LI (0.5)	HI (0.75)	HI (0.75)
	FR_8	HI (0.75)	HI (0.75)	VLI (0.25)	LI (0.5)	HI (0.75)
	FR_9	HI (0.75)	HI (0.75)	VLI (0.25)	HI (0.75)	HI (0.75)
	FR_{10}	HI (0.75)	LI (0.5)	NI (0.001)	HI (0.75)	VLI (0.25)
	FR_{11}	HI (0.75)	HI (0.75)	NI (0.001)	HI (0.75)	HI (0.75)

Table 5.3, continued

Decision matrix	Functional requirements	Non-functional requirements				
		NFR_1	NFR_2	NFR_3	NFR_4	NFR_5
	FR_{12}	LI (0.5)	HI (0.75)	NI (0.001)	LI (0.5)	HI (0.75)
	FR_{13}	LI (0.5)	HI (0.75)	NI (0.001)	LI (0.5)	HI (0.75)
	FR_{14}	LI (0.5)	HI (0.75)	VLI (0.25)	LI (0.5)	HI (0.75)
	FR_{15}	HI (0.75)	HI (0.75)	VLI (0.25)	HI (0.75)	HI (0.75)
D_4	FR_1	VHI (1)	VHI (1)	LI (0.5)	HI (0.75)	VHI (1)
	FR_2	HI (0.75)	HI (0.75)	LI (0.5)	LI (0.5)	HI (0.75)
	FR_3	VHI (1)	VHI (1)	HI (0.75)	HI (0.75)	VHI (1)
	FR_4	VHI (1)	VHI (1)	LI (0.5)	HI (0.75)	VHI (1)
	FR_5	HI (0.75)	VHI (1)	HI (0.75)	LI (0.5)	HI (0.75)
	FR_6	LI (0.5)	HI (0.75)	LI (0.5)	LI (0.5)	HI (0.75)
	FR_7	VHI (1)	VHI (1)	HI (0.75)	LI (0.5)	VHI (1)
	FR_8	HI (0.75)	LI (0.5)	LI (0.5)	LI (0.5)	LI (0.5)
	FR_9	VHI (1)	VHI (1)	HI (0.75)	LI (0.5)	VHI (1)
	FR_{10}	HI (0.75)	HI (0.75)	HI (0.75)	LI (0.5)	HI (0.75)
	FR_{11}	HI (0.75)	VHI (1)	HI (0.75)	LI (0.5)	VHI (1)
	FR_{12}	HI (0.75)	LI (0.5)	VLI (0.25)	LI (0.5)	LI (0.5)
	FR_{13}	HI (0.75)	VHI (1)	HI (0.75)	LI (0.5)	HI (0.75)
	FR_{14}	HI (0.75)	HI (0.75)	HI (0.75)	LI (0.5)	HI (0.75)
	FR_{15}	HI (0.75)	VHI (1)	HI (0.75)	HI (0.75)	VHI (1)

Step 7: Calculate NFRs ranking with respect to all FRs using triangular fuzzy number and alpha cut approach

To calculate the priority vector of non-functional requirements with respect to all functional requirements, the elements of matrix D are converted to actual scales (Sub-step 1), the TFN is calculated for each non-functional requirement (Sub-step 2), the fuzzy priority vector is constructed (Sub-step 3), and the defuzzification is done in order to achieve the priority vector W and NW (Sub-step 4). Table 5.3 shows Sub-step 1 where the elements of matrixes D_1, D_2, D_3, and D_4 with actual scales are indicated within parenthesis, while the values of $\widetilde{F_x}$, W (Equation (4.4)), and NW (Equation (4.5)) related to each non-functional requirement are represented in Table 5.4. By calculating the priority vector NW, we are provided with a prioritized list of non-functional

requirements for each stakeholder (numbers within parenthesis represent the priority of each non-functional requirement for considering during the development process). Thus, totally, four prioritized lists of non-functional requirements are achieved.

Table 5.4: Computation of NFRs' priority vectors with respect to all FRs

Decision matrix	Non-functional requirements	$\widetilde{\mathbf{F}}_{\mathbf{x}}$	$\boldsymbol{W}$	$\boldsymbol{NW}$
D_1	$\boldsymbol{NFR_1}$	(0.25,0.60415,1.0)	0.61	0.145 (5)
	$\boldsymbol{NFR_2}$	(0.75,0.981004,1.0)	0.93	0.221 (2)
	$\boldsymbol{NFR_3}$	(0.75,0.794417,1.0)	0.83	0.198 (3)
	$\boldsymbol{NFR_4}$	(0.75,0.794417,1.0)	0.83	0.198 (4)
	$\boldsymbol{NFR_5}$	(1.0,1.0,1.0)	1.0	0.238 (1)
D_2	$\boldsymbol{NFR_1}$	(0.25,0.511916,1.0)	0.568	0.178 (5)
	$\boldsymbol{NFR_2}$	(0.25,0.521829,1.0)	0.573	0.180 (4)
	$\boldsymbol{NFR_3}$	(0.25,0.60415,1.0)	0.615	0.193 (3)
	$\boldsymbol{NFR_4}$	(0.25,0.650059,1.0)	0.64	0.201 (2)
	$\boldsymbol{NFR_5}$	(0.5,0.828354,1.0)	0.79	0.248 (1)
D_3	$\boldsymbol{NFR_1}$	(0.5,0.699457,1.0)	0.72	0.227 (2)
	$\boldsymbol{NFR_2}$	(0.5,0.819025,1.0)	0.78	0.246 (1)
	$\boldsymbol{NFR_3}$	(0.001,0.095002,1.0)	0.3	0.095 (5)
	$\boldsymbol{NFR_4}$	(0.5,0.675479,1.0)	0.71	0.224 (3)
	$\boldsymbol{NFR_5}$	(0.25,0.704971,1.0)	0.66	0.208 (4)
D_4	$\boldsymbol{NFR_1}$	(0.5,0.803466,1.0)	0.777	0.222 (3)
	$\boldsymbol{NFR_2}$	(0.5,0.844394,1.0)	0.80	0.229 (1)
	$\boldsymbol{NFR_3}$	(0.25,0.608913,0.75)	0.55	0.157 (5)
	$\boldsymbol{NFR_4}$	(0.5,0.557091,0.75)	0.59	0.169 (4)
	$\boldsymbol{NFR_5}$	(0.5,0.812619,1.0)	0.781	0.223 (2)

Figure 5.1 summarizes the weights of non-functional requirements obtained by four stakeholders who participated in the prioritization process.

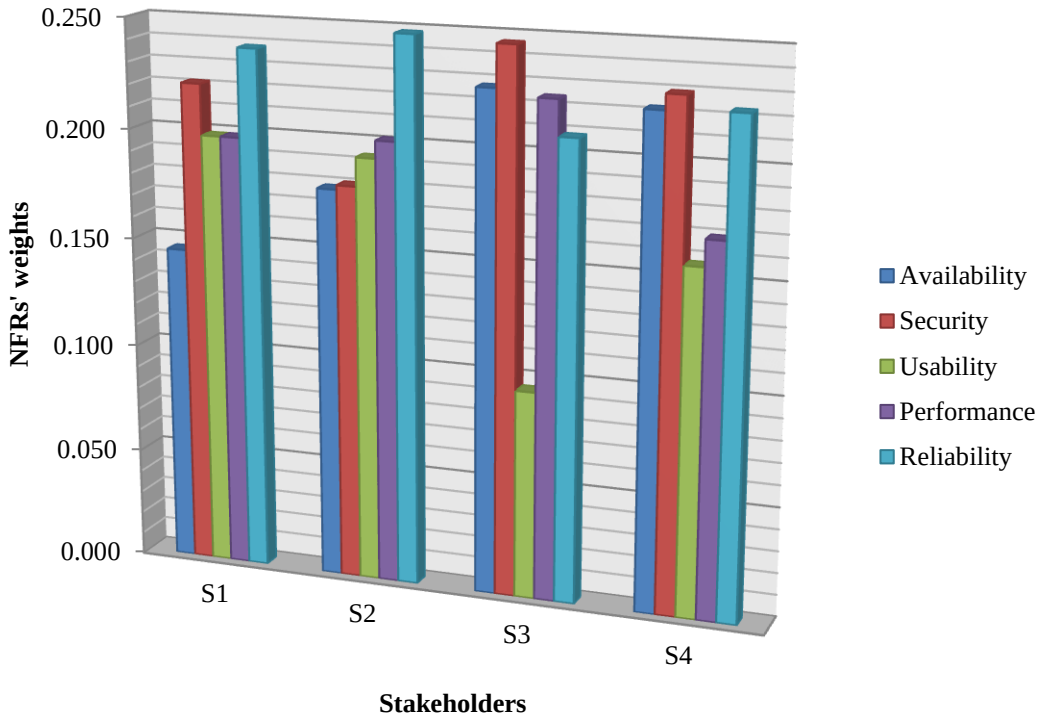


Figure 5.1: The NFRs' weights achieved by four stakeholders participated in the prioritization process

Step 8: Compute FRs ranking using weighted average decision matrix and weights determined in Step 7

In this step, the ranking (Equation (4.6)) along with the normalized ranking (Equation (4.7)) of each functional requirement for each stakeholder are calculated (see Table 5.3), using the values of Ď as well as NW which was achieved in Step 7. This step applies a classical weighted average matrix, where rows depict functional requirements while columns depict non-functional requirements. By performing this step, the prioritized list of functional requirements is also achieved for each stakeholder. The most right column of Table 5.5 (i.e. NR) indicates the weight of functional requirements obtained by each stakeholder.

Table 5.5: Calculation of FRs' priority vectors with respect to NFRs

Stakeholder	Functional requirements	Non-functional requirements						
		NFR_1	NFR_2	NFR_3	NFR_4	NFR_5	R	NR
	NFRs' weights	*0.145*	*0.221*	*0.198*	*0.198*	*0.238*		
S_1	FR_1	1	1	1	1	1	1.000	0.079
	FR_2	0.75	1	0.75	0.75	1	0.856	0.067
	FR_3	0.75	1	1	1	1	0.959	0.075
	FR_4	0.75	1	1	1	1	0.959	0.075
	FR_5	0.5	1	0.75	0.75	1	0.807	0.063
	FR_6	0.75	1	0.75	0.75	1	0.856	0.067303
	FR_7	0.75	1	0.75	0.75	1	0.856	0.067303
	FR_8	0.75	0.75	0.75	0.75	1	0.803	0.063157
	FR_9	0.5	1	0.75	0.75	1	0.807	0.06346
	FR_{10}	0.75	1	0.75	0.75	1	0.856	0.067303
	FR_{11}	0.5	1	0.75	0.75	1	0.807	0.06346
	FR_{12}	0.5	1	0.75	0.75	1	0.807	0.06346
	FR_{13}	0.5	1	0.75	0.75	1	0.807	0.06346
	FR_{14}	0.25	1	0.75	0.75	1	0.730	0.057392
	FR_{15}	0.5	1	0.75	0.75	1	0.807	0.06346
	NFRs' weights	*0.178*	*0.180*	*0.193*	*0.201*	*0.248*		
S_2	FR_1	0.75	0.5	0.5	0.75	1	0.692	0.072
	FR_2	0.5	0.5	0.75	0.5	0.75	0.598	0.062
	FR_3	0.75	0.75	0.75	1	1	0.853	0.089
	FR_4	0.5	1	0.75	0.75	1	0.789	0.082
	FR_5	0.25	0.5	1	0.5	0.5	0.505	0.052
	FR_6	0.5	0.25	0.5	0.75	0.75	0.529	0.054931
	FR_7	0.75	0.75	0.75	0.75	1	0.805	0.083563
	FR_8	0.5	0.25	0.25	0.5	0.5	0.386	0.040055
	FR_9	0.75	0.75	0.75	0.75	1	0.805	0.083563
	FR_{10}	1	0.5	0.75	0.75	1	0.788	0.081763
	FR_{11}	0.25	0.5	0.5	0.5	0.75	0.489	0.050702
	FR_{12}	0.25	0.5	0.5	0.75	0.75	0.530	0.055007
	FR_{13}	0.75	0.75	0.75	0.75	1	0.805	0.083563
	FR_{14}	0.25	0.25	0.5	0.25	0.75	0.375	0.038934
	FR_{15}	0.75	0.75	0.5	0.5	1	0.687	0.071225

Table 5.5, continued

Stakeholder	Functional requirements	Non-functional requirements						
		NFR_1	NFR_2	NFR_3	NFR_4	NFR_5	R	NR
	NFRs' weights	*0.227*	*0.246*	*0.095*	*0.224*	*0.208*		
S_3	FR_1	0.75	1	0.75	1	1	0.912	0.095
	FR_2	0.75	1	0.75	0.75	0.75	0.805	0.084
	FR_3	1	1	0.75	1	0.75	0.917	0.096
	FR_4	1	1	0.75	1	1	0.973	0.102
	FR_5	0.75	1	1	0.5	0.75	0.755	0.079
	FR_6	0.5	1	0.75	0.5	0.5	0.616	0.064296
	FR_7	0.75	0.75	0.5	0.75	0.75	0.722	0.075294
	FR_8	0.75	0.75	0.25	0.5	0.75	0.617	0.064375
	FR_9	0.75	0.75	0.25	0.75	0.75	0.676	0.070496
	FR_{10}	0.75	0.5	0.001	0.75	0.25	0.288	0.030047
	FR_{11}	0.75	0.75	0.001	0.75	0.75	0.400	0.041721
	FR_{12}	0.5	0.75	0.001	0.5	0.75	0.333	0.034749
	FR_{13}	0.5	0.75	0.001	0.5	0.75	0.333	0.034749
	FR_{14}	0.5	0.75	0.25	0.5	0.75	0.563	0.058715
	FR_{15}	0.75	0.75	0.25	0.75	0.75	0.676	0.070496
	NFRs' weights	*0.222*	*0.229*	*0.157*	*0.169*	*0.223*		
S_4	FR_1	1	1	0.5	0.75	1	0.854	0.077
	FR_2	0.75	0.75	0.5	0.5	0.75	0.657	0.059
	FR_3	1	1	0.75	0.75	1	0.910	0.082
	FR_4	1	1	0.5	0.75	1	0.854	0.077
	FR_5	0.75	1	0.75	0.5	0.75	0.748	0.067
	FR_6	0.5	0.75	0.5	0.5	0.75	0.601	0.053798
	FR_7	1	1	0.75	0.5	1	0.850	0.076158
	FR_8	0.75	0.5	0.5	0.5	0.5	0.547	0.049008
	FR_9	1	1	0.75	0.5	1	0.850	0.076158
	FR_{10}	0.75	0.75	0.75	0.5	0.75	0.700	0.062734
	FR_{11}	0.75	1	0.75	0.5	1	0.798	0.071446
	FR_{12}	0.75	0.5	0.25	0.5	0.5	0.491	0.043955
	FR_{13}	0.75	1	0.75	0.5	0.75	0.748	0.067006
	FR_{14}	0.75	0.75	0.75	0.5	0.75	0.700	0.062734
	FR_{15}	0.75	1	0.75	0.75	1	0.854	0.076513

Step 9: Aggregate different prioritized lists of FRs and NFRs provided by various stakeholders to obtain final rankings of FRs and NFRs

In the preceding steps, we have observed how to produce the prioritized list of functional requirements as well as non-functional requirements for each stakeholder.

Thus, in this case, four prioritized lists of functional requirements and four prioritized lists of non-functional requirements have been achieved due to the reason that four stakeholders participated in the prioritization process. But, there is still a need to aggregate these different prioritized lists of functional requirements as well non-functional requirements.

In response to aggregate four prioritized lists of functional requirements and produce a single prioritized list of functional requirements in which the opinions of all stakeholders have been considered, weighted average matrix is constructed as shown in Table 5.6. The most right column of Table 5.6 (i.e. UR) shows the final weights of functional requirements along with their priorities (numbers within parenthesis).

Table 5.6: Calculating the prioritized list of FRs

Stakeholders' weights	**0.32**	**0.22**	**0.18**	**0.28**	
Functional requirements	S_1	S_2	S_3	S_4	UR
FR_1	0.079	0.072	0.095	0.077	0.080 (3)
FR_2	0.067	0.062	0.084	0.059	0.067 (7)
FR_3	0.075	0.089	0.096	0.082	0.084 (1)
FR_4	0.075	0.082	0.102	0.077	0.082 (2)
FR_5	0.063	0.052	0.079	0.067	0.065 (8)
FR_6	0.067303	0.054931	0.064296	0.053798	0.060 (11)
FR_7	0.067303	0.083563	0.075294	0.076158	0.075 (4)
FR_8	0.063157	0.040055	0.064375	0.049008	0.054 (14)
FR_9	0.06346	0.083563	0.070496	0.076158	0.073 (5)
FR_{10}	0.067303	0.081763	0.030047	0.062734	0.062 (10)
FR_{11}	0.06346	0.050702	0.041721	0.071446	0.059 (12)
FR_{12}	0.06346	0.055007	0.034749	0.043955	0.051 (15)
FR_{13}	0.06346	0.083563	0.034749	0.067006	0.064 (9)
FR_{14}	0.057392	0.038934	0.058715	0.062734	0.055 (13)
FR_{15}	0.06346	0.071225	0.070496	0.076513	0.070 (6)

Similarly, in response to aggregate four prioritized lists of non-functional requirements and produce a single prioritized list of non-functional requirements in which the opinions of all stakeholders have been considered, weighted average matrix is constructed as shown in Table 5.7. The most right column of Table 5.7 (i.e. UW) shows

the final weights of non-functional requirements along with their priorities (numbers within parenthesis).

Table 5.7: Computing the prioritized list of NFRs

NFRs	Stakeholders' weights	**0.32**	**0.22**	**0.18**	**0.28**	
		S_1	S_2	S_3	S_4	UW
NFR_1		0.145	0.178	0.227	0.222	0.188 (4)
NFR_2		0.221	0.180	0.246	0.229	0.218 (2)
NFR_3		0.198	0.193	0.095	0.157	0.167 (5)
NFR_4		0.198	0.201	0.224	0.169	0.196 (3)
NFR_5		0.238	0.248	0.208	0.223	0.231 (1)

Therefore, the outcomes of applying IPA on the functional and non-functional requirements of the banking software system, while participating four stakeholders in the prioritization process are given in Figure 5.2 and Figure 5.3, where Figure 5.2 depicts the final prioritized list of functional requirements whereas Figure 5.3 demonstrates the final prioritized list of non-functional requirements.

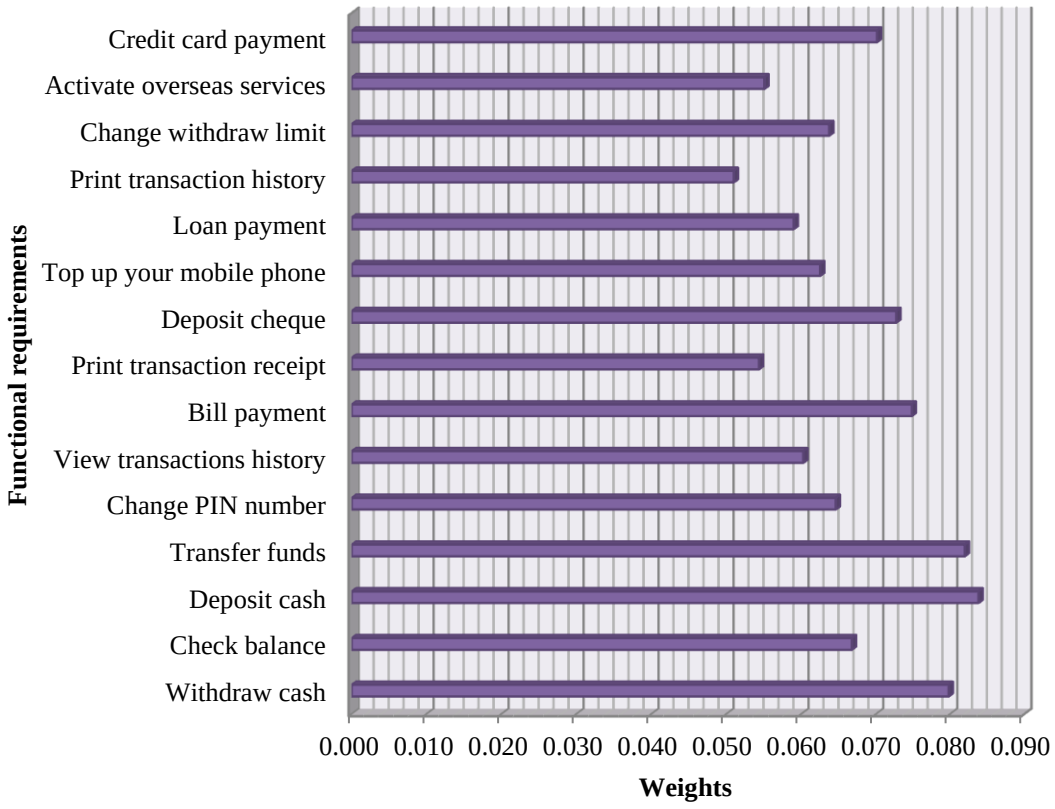


Figure 5.2: The final prioritized list of FRs ranked according to their relationships with NFRs

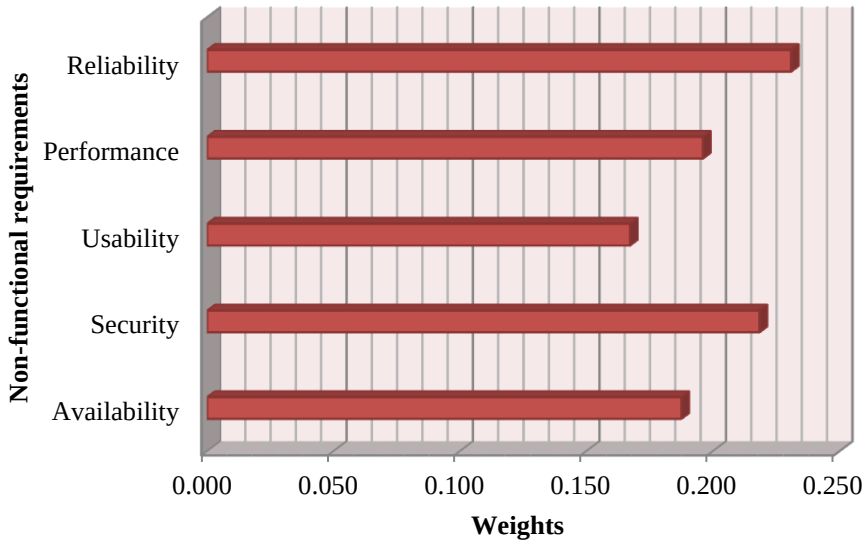


Figure 5.3: The final prioritized list of NFRs ranked according to their relationships with FRs

5.2 Validate the Proposed Approach using Mathematical Theory

The goal of this section is to provide mathematical evidence that the outcome of the proposed approach (i.e. IPA) is proven to be valid using graph theory. To achieve the desired goal, first, there is a need to express the proposed approach using mathematical formulation. Indeed, the mathematical formulation of the proposed approach could simplify the process of validating the proposed approach using graph theory. Then, graph theory is used in order to validate the outcome of the proposed approach.

5.2.1 Mathematical Formulation of the Proposed Approach

Let S be the set of stakeholders who participate in the prioritization process. It is denoted as:

$$S = \{S_1, S_2, S_3, \dots, S_k\} \tag{5.4}$$

where k is the total number of stakeholders.

These stakeholders might have different weights according to their significance for the prioritization problem. AHP has been applied in our approach in order to perform the process of weighting stakeholders. Thus, a function should be defined as follows:

$$F_{AHP}(s_i) = \{w_i \in [0,1] \; \forall s_i \in S, 1 \leq i \leq k\} \quad (5.5)$$

where $F_{AHP}(s_i)$ applies AHP for calculating the relative weight of the stakeholder s_i; w_i is a real number such that $0 \leq w_i \leq 1$; and k is the number of stakeholders.

Suppose that FR is the set of functional requirements which are required to be prioritized. It can be expressed as follows:

$$FR = \{FR_1, FR_2, FR_3, \dots, FR_n\} \quad (5.6)$$

where n is the total number of functional requirements.

Similarly, let NFR be the set of non-functional requirements which are required to be prioritized. It is denoted as:

$$NFR = \{NFR_1, NFR_2, NFR_3, \dots, NFR_m\} \quad (5.7)$$

where m is the total number of non-functional requirements.

However, there is a relationship between functional and non-functional requirements which should be established in terms of eliciting the importance degree of each non-functional requirement for a given functional requirement. This relationship can be expressed as follows:

$$F_M(FR_i, NFR_j) = \{m_{ij} \in (0.001,0.25,0.5,0.75,1) \; \forall FR_i \in FR, 1 \leq i \leq n, \forall NFR_j \in NFR, 1 \leq j \leq m\} \quad (5.8)$$

where $F_M(FR_i, NFR_j)$ indicates the importance degree of non-functional NFR_j for achieving functional requirement FR_i; m_{ij} is a real number belongs to IPA actual scale (see Table 4.3); finally n and m represent the number of functional and non-functional requirements, respectively.

Moreover, there is a need for defining functions aimed at calculating the weights of functional requirements as well non-functional requirements. Thus, initially, a function is defined in the following for computing the weights of non-functional requirements using triangular fuzzy number and alpha cut approach:

$$F_{TA}(NFR_j) = \{NW_j \in [0,1] \; \forall \; NFR_j \in NFR, \; 1 \leq j \leq m\} \tag{5.9}$$

where $F_{TA}(NFR_j)$ applies fuzzy triangular number and alpha cut approach to compute the weight of NFR_j; NW_j is a real number such that $0 \leq NW_j \leq 1$; and m is the number of non-functional requirements.

In a similar way, another function is needed to calculate the weights of functional requirements. This is formulated as follows:

$$F_{WA}(FR_i) = \{NR_i \in [0,1] \; \forall \; FR_i \in FR, \; 1 \leq i \leq n\} \tag{5.10}$$

where $F_{WA}(FR_i)$ uses weighted average decision matrix to calculate the weight of FR_i; NR_i is a real number such that $0 \leq NR_i \leq 1$; and n is the number of functional requirements.

Nevertheless, at the end of the process of prioritizing non-functional requirements, there is a need to aggregate different prioritized lists of non-functional requirements produced by different stakeholders. This is formulated as follows:

$$UW_i = \sum_{j=1}^{k} F_{TA}(NFR_{ij}) * F_{AHP}(s_j) \tag{5.11}$$

where UW_i is a real number such that $0 \leq UW_i \leq 1$ which represents the final weight of non-functional requirement W_i; and k is the number of stakeholders.

The same process needs to be done in order to produce the single prioritized list of functional requirements. It is expressed as follows:

$$UR_i = \sum_{j=1}^{k} F_{WA}(FR_{ij}) * F_{AHP}(s_j) \quad (5.12)$$

where UR_i is a real number such that $0 \leq UR_i \leq 1$ which indicates the final weight of functional requirement R_i; and k is the number of stakeholders.

5.2.2 Validation of the Proposed Approach using Graph Theory

In the following, graph theory is used to indicate that the outcome of the proposed approach is valid.

A graph $G = (V, E)$ is represented by two main sets, called vertices $V(G)$ and edges $E(G)$. In other words, vertices are the nodes of graph which connect together using edges.

In the context of this research, the set of requirements (functional or non-functional) represents the vertices of the graph where each pair of requirements (functional or non-functional) is connected together using an edge. Each edge of the graph has a weight which indicates the weights summation of two connected requirements. So, a complete weighted graph is formed. For a graph that contains n nodes, there are $n!$ possible paths, which may indicate the final prioritized list. For example, Figure 5.4 represents a complete weighted graph which is formed based on three functional requirements. So, there are 6 possible paths which might represent the final prioritized list. They are:

- FR1→ FR2→ FR3
- FR1→ FR3→ FR2

- FR2→ FR1→ FR3
- FR2→ FR3→ FR1
- FR3→ FR2→ FR1
- FR3→ FR1→ FR2

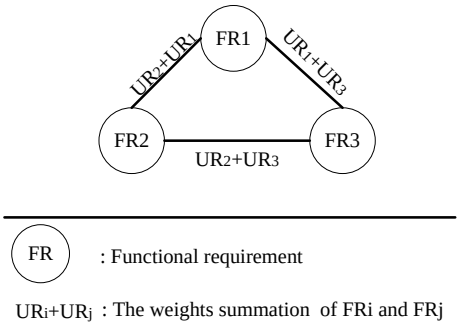


Figure 5.4: A sample of complete weighted graph for three functional requirements

Then, the intention is to find out the single path which indicates the prioritized list of functional requirements, produced by the proposed approach, among all possible paths. The procedure which needs to be followed to discover the single path is represented as a flowchart in Figure 5.5. Finally, the outcome would be represented in the form of an acyclic directed sub graph.

As mentioned before, the main goal of using graph theory is to prove that the final prioritized list of functional and non-functional requirements produced by the proposed approach have been sorted in a correct order. By applying the graph theory on the example presented in Section 5.1, two graphs need to be formed; one for non-functional requirements and the other one for functional requirements.

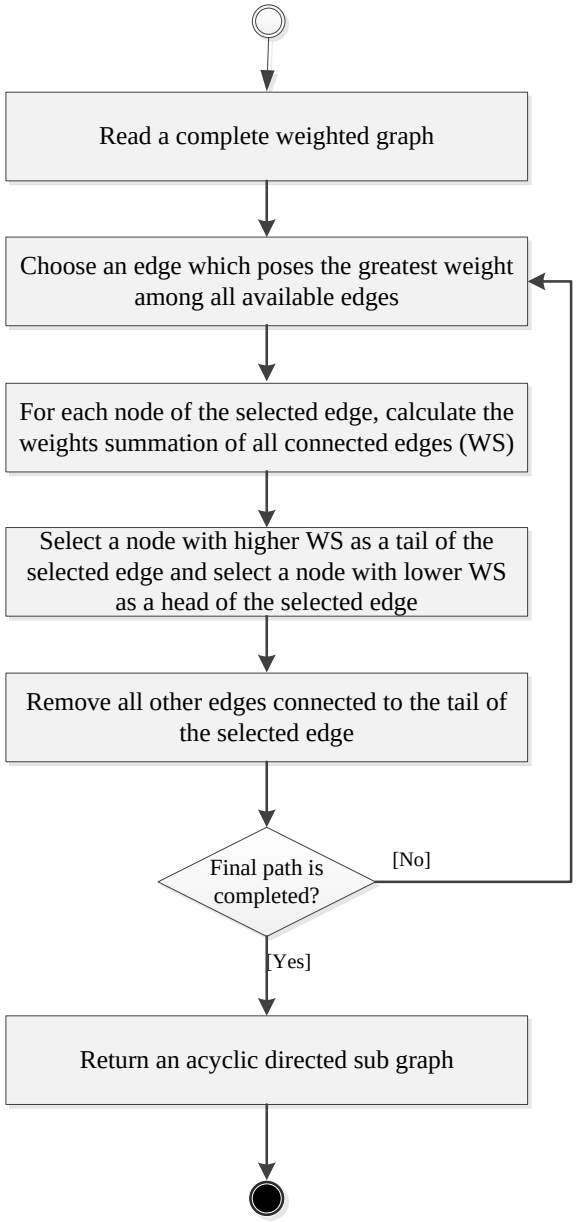


Figure 5.5: Flow chart of identifying the single path which indicates the final prioritized list

The complete weighted graph of non-functional requirements is represented in Figure 5.6. As can be seen in Figure 5.6, there are 120 possible paths which may indicate the final prioritized list of non-functional requirements. By following the procedure represented in Figure 5.5, the final prioritized list of non-functional requirements is achieved and represented in Figure 5.7. As can be observed in Figure 4.9, the prioritized list of non-functional requirements (NFR5→NFR2→NFR4→NFR1→NFR3) is the same as the prioritized list of non-functional produced by the proposed approach (see Table 5.7).

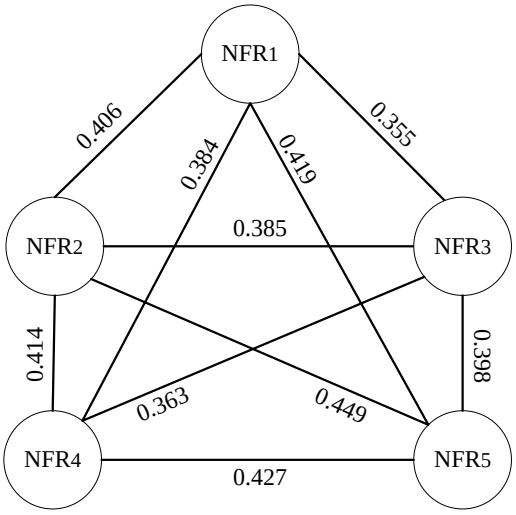


Figure 5.6: The completed weighted graph of non-functional requirements

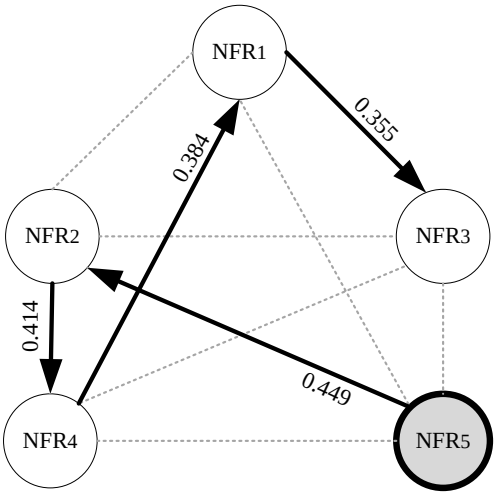


Figure 5.7: The acyclic directed sub graph indicates the final prioritized list of non-functional requirements

Similarly, in order to discover the final prioritized list of functional requirements of the example presented in Section 5.1 using graph theory, a complete weighted graph is formed which contains 15 nodes and 15! possible paths. By applying the procedure presented in Figure 5.5, the single path which indicates the final prioritized list of functional requirements is obtained and shown (the bold directed paths started by FR3) in Figure 5.8. It is quite obvious that this path (FR3→FR4→ FR1→FR7→

FR9→FR15→ FR2→FR5→ FR13→FR10→ FR6→FR11→ FR14→FR8→ FR12) is the same as the list produced and represented by the proposed approach in Table 5.6.

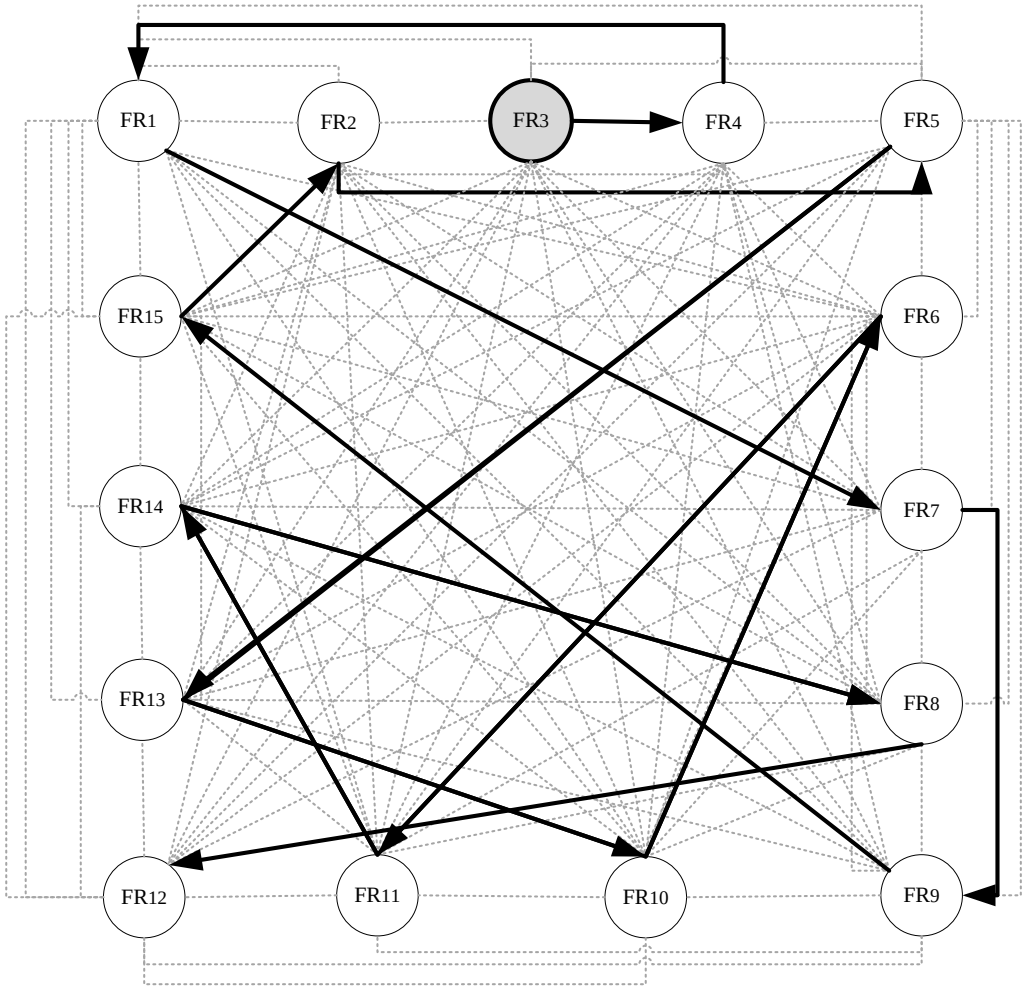


Figure 5.8: The acyclic directed sub graph indicates the final prioritized list of functional requirements

5.3 Summary

This chapter explained the application of IPA on the requirements of ATM, CDM, and CQM by participating four stakeholders to demonstrate how the nine steps of IPA could be applied in real cases. In addition, the chapter described the application of graph theory to provide mathematical evidence that the outcome of the proposed approach (i.e. IPA) is proven to be valid.

CHAPTER 6: EVALUATION OF THE PROPOSED APPROACH THROUGH CONTROLLED EXPERIMENTS

This research aims to contribute to software engineering community by proposing an approach which enables researchers and practitioners to integrate the prioritization of functional and non-functional requirements simultaneously. The detailed description of this approach has been provided in Chapter 4. However, improving the maturity of the whole body of exploration in this field of research will never be achieved unless the proposed approach is completely analyzed, evaluated, and compared and the demands for enhancement are recognized. In this respect, an in-depth evaluation of the proposed approach in an empirical manner would be needed.

The objective of this chapter is to present the evaluation the effectiveness of the proposed approach, i.e. IPA, in terms of time needed for performing the prioritization task, accuracy of the results produced by the approach and ease of use when compared to the other state-of-the-art approaches.

To achieve the desired goal, two successive controlled experiments have been performed during this research. Controlled experiment has been recognized as one of the most common empirical strategies for evaluating a new approach in the software engineering domain (Wohlin et al., 2012). Appendix B provides a general overview on the most common empirical strategies which can be used to investigate the evaluation of a new software engineering approach.

In the first controlled experiment, IPA has been compared with the state-of-the-art approach, called AHP-based approach. To compare these approaches, the focus was mainly on measuring three relevant properties such as the actual time-consumption, the accuracy of results, and ease of use. AHP-based approach has been exploited for comparative evaluation of IPA since AHP has been widely applied as a reference

method in empirical evaluations of different prioritization methods (J. Karlsson et al., 1998; L. Karlsson et al., 2007; Perini et al., 2009; Perini et al., 2007; Perini et al., 2013; Ribeiro et al., 2011; Tonella et al., 2013). Moreover, it is also found to be the most well-known and robust method in several domains (Barney, Petersen, Svahnberg, Aurum, & Barney, 2012).

As the results of the first controlled experiment indicated the superiority of IPA over AHP-based approach, the researcher conducted the second experiment with the aim of comparing IPA with the other state-of-the-art alternative, named HAM-based approach to find out whether IPA outperforms HAM-based approach with respect to the actual time-consumption, the accuracy of results, and ease of use. The results of the second experiment indicated statistically that IPA shows a better performance than HAM-based approach.

6.1 Controlled Experiments

This section illustrates in detail the two controlled experiments, which have been carried out to assess the actual time consumption, accuracy of results, and ease of use of the IPA. Table 6.1 summarizes the key components of the two controlled experiments. To conduct both experiments, the researcher has followed the guidelines proposed by (Wohlin et al., 2012), on how to define, plan, run, and analyze the results of an experiment in the software engineering domain. As can be observed in Table 6.1, the main difference between Experiment 1 and Experiment 2 is that we replaced AHP-based approach with HAM-based approach. Note that Experiment 2 was conducted almost six month after Experiment 1. The reason of not conducting only one experiment, which includes the three prioritization approaches, i.e. IPA, AHP-based approach, and HAM-based approach, is that on the time of conducting the first experiment, HAM-based approach had not been proposed. Moreover, conducting a

controlled experiment, which deals with three approaches, might make participants exhausted, and thereby, may bias the final results.

Table 6.1: Overview of the controlled experiments

	Experiment 1	**Experiment 2**
Goal	Analyze two tool-supported approaches for software requirements prioritization: IPA and AHP-based approach, with the goal of measuring the actual time-consumption, accuracy of results, and ease of use	Analyze two tool-supported approaches for software requirements prioritization: IPA and HAM-based approach, with the purpose of measuring the actual time-consumption, accuracy of results, and ease of use
Independent variables	IPA and AHP-based approach	IPA and HAM-based approach
Perspective	From the point of view of the decision maker and researcher (for both experiments)	
Context	Both experiments were executed using 20 real subjects prioritizing a collection of 20 real requirements including 15 functional requirements and 5 non-functional requirements of ATM, CDM, and CQM.	
Dependent variables	Actual time-consumption; accuracy of results and ease of use (for both experiments)	
Other variables	Execution order, experience in requirements prioritization (for both experiments)	

6.1.1 Experiment Definition

6.1.1.1 Definition of Experiment 1

The main goal of the first experiment is to analyze two approaches for software requirements prioritization, IPA and AHP-based approach. To achieve the desired goal, three properties are evaluated during the experiment: actual time-consumption, accuracy of results, and ease of use. The perspective is from the decision maker's and researcher's point of view, so that the decision maker and researcher would like to investigate the difference between IPA and AHP-based approach in terms of the three properties. Moreover, this investigation is useful for decision makers and researchers who may want to select an appropriate prioritization approach for a given requirements prioritization problem or to initiate a new prioritization approach in the future. The experiment has been carried out in the context of 20 Ph.D. students and research

scholars prioritizing a collection of 20 real requirements including 15 functional requirements and 5 non-functional requirements of ATM (Automated Teller Machine), CDM (Cash Deposit Machine), and CQM (Check Deposit Machine). Thus, this experiment is classified as blocked subject-object study.

As a summary, the goal of the first experiment is defined as:

Analyze the IPA and AHP-based approach for the purpose of evaluation with respect to actual time-consumption, accuracy of results, and ease of use from the point of view of the decision makers and researchers in the context of Ph.D. students and research scholars prioritizing requirements.

To achieve the defined goal of this experiment, the researcher needed to investigate for finding answers to the following research questions:

- **RQ1-1**: How fast are IPA and AHP-based approach when applied to perform the prioritization process?
- **RQ1-2**: Which approach, between IPA and AHP-based approach, is easier to use?
- **RQ1-3**: Which approach, between IPA and AHP-based approach, produces results that are more accurate?

6.1.1.2 Definition of Experiment 2

The principal purpose of the second experiment is to analyze the IPA and HAM-based approach for the purpose of evaluation with respect to actual time-consumption, accuracy of results, and ease of use from the point of view of the decision makers and researchers. The experiment has been done in the context of Ph.D. students prioritizing a collection of 20 real requirements including 15 functional requirements and 5 non-

functional requirements of ATM, CDM, and CQM. This experiment is classified as blocked subject-object study.

To achieve the desired goal, the researcher required to verify the following research questions:

- **RQ2-1**: How fast are IPA and HAM-based approach when applied to perform the prioritization process?
- **RQ2-2**: Which approach, between IPA and HAM-based approach, is easier to use?
- **RQ2-3**: Which approach, between IPA and HAM-based approach, produces results that are more accurate?

6.1.2 Experiment Planning

6.1.2.1 Context Selection

Both experiments have been done in a research laboratory environment, and hence the experiments were executed off-line (i.e. the experiments have not been done in an industrial setting). The experiments have been conducted with participating Ph.D. students and research scholars as subjects, and the experiments are classified as specific since they have focused on the requirements of ATM (Automated Teller Machine), CDM (Cash Deposit Machine), and CQM (Check Deposit Machine). The experiments have addressed a real problem since they have investigated the differences between two requirements prioritization approaches when applied on the same set of requirements.

6.1.2.2 Hypothesis Formulation

Depending on previously mentioned research questions, the following null and alternative hypotheses have been formulated for the controlled experiments:

Hypotheses of Experiment 1:

- *Null hypothesis* (H_{0time}): There is no significant difference between IPA and AHP-based approach with regards to the average actual time-consumption to conclude the prioritization task.

  Alternative hypothesis (H_{1time}): There is a significant difference between IPA and AHP-based approach with regards to the average actual time-consumption to conclude a prioritization task.

- *Null hypothesis* ($H_{0easeofuse}$): There is no significant difference between IPA and AHP-based approach in terms of ease of use.

  Alternative hypothesis ($H_{1easeofuse}$): There is a significant difference between IPA and AHP-based approach in terms of ease of use.

- *Null hypothesis* ($H_{0accuracy}$): The accuracy is equal for IPA and AHP-based approach.

  Alternative hypothesis ($H_{1accuracy}$): The accuracy is not equal for IPA and AHP-based approach.

Hypotheses of Experiment 2:

- *Null hypothesis* (H_{0time}): There is no significant difference between IPA and HAM-based approach with regards to the average actual time-consumption to conclude the prioritization task.

Alternative hypothesis (H_{1time}): There is a significant difference between IPA and HAM-based approach with regards to the average actual time-consumption to conclude a prioritization task.

- *Null hypothesis* ($H_{0easeofuse}$): There is no significant difference between IPA and HAM-based approach in terms of ease of use.

  Alternative hypothesis ($H_{1easeofuse}$): There is a significant difference between IPA and HAM-based approach in terms of ease of use.

- *Null hypothesis* ($H_{0accuracy}$): The accuracy is equal for IPA and HAM-based approach.

  Alternative hypothesis ($H_{1accuracy}$): The accuracy is not equal for IPA and HAM-based approach.

6.1.2.3 Variables and Measures

Similar to almost any kind of controlled experiments in software engineering domain, independent variables as well as dependent variables of the controlled experiments needed to be identified.

Independent variables:

The independent variables of the first experiment were *IPA* and *AHP-based approach* while the independent variables of the second experiment were *IPA* and *HAM-based approach*. Figure 6.1 sketches a graphical overview of these approaches. It should be highlighted that, in the following description of the three approaches, the term *non-functional requirement* refers to *system quality attributes* such as reliability, security, and etc. A brief description on the three approaches, IPA, AHP-based approach, and HAM-based approach, along with their implemented software prototypes are provided in the following.

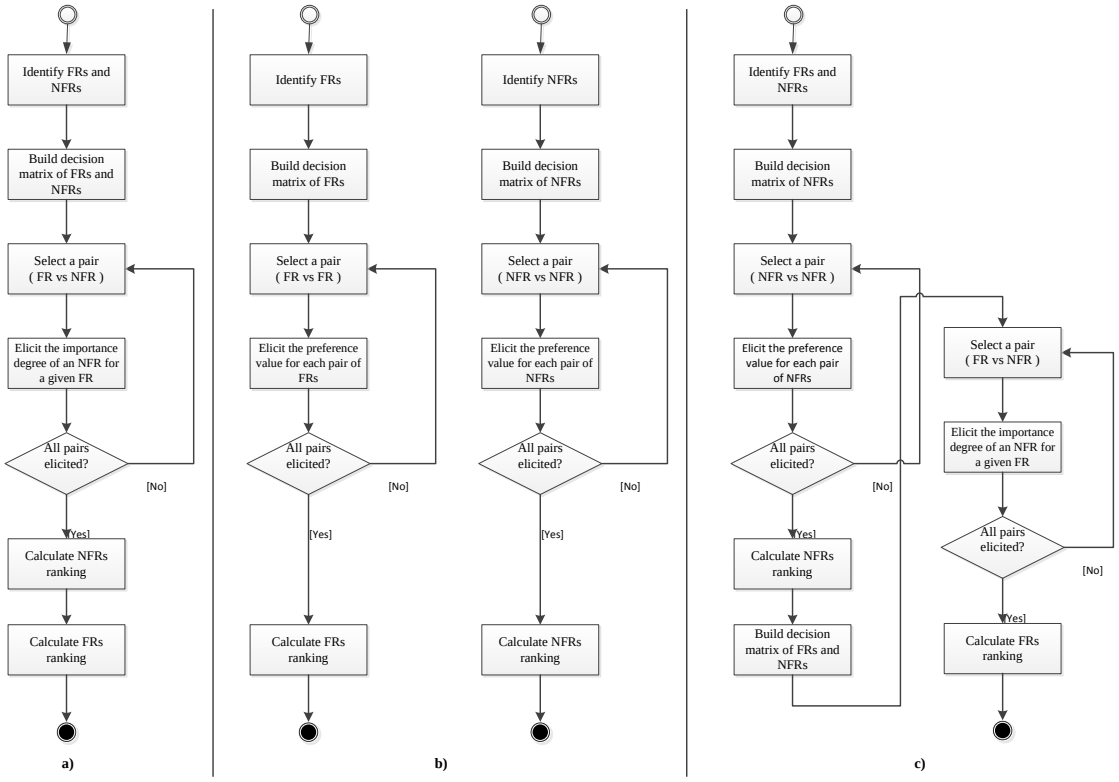


Figure 6.1: The process of prioritizing functional and non-functional requirements using a) IPA; b) AHP-based approach; c) HAM-based approach

- **IPA and its supporting tool**

In this research, an approach has been proposed in order to integrate the process of prioritizing functional and non-functional requirements simultaneously. This approach is called IPA. The detailed description of IPA has been provided in Chapter 4.

Figure 6.1a demonstrates the steps, which should be done by a decision maker to prioritize functional and non-functional requirements using IPA. As can be seen in Figure 6.1a, the process is started with identifying functional and non-functional requirements by a decision maker. Then, IPA puts the functional and non-functional requirements into rows and columns of the decision matrix, respectively. Afterward, the decision maker (the person who works with IPA) is requested to give his/her opinion regarding the importance degree of each non-functional requirement with respect to each functional requirement. In reality, the decision maker is asked to express his/her opinion according to five scales, called IPA nominal scale (see Table 4.3). Once the process of eliciting the importance degree of each non-functional requirement for all

functional requirements is completed, IPA begins to calculate the weight of each non-functional requirement, using triangular fuzzy number and alpha-cut approach, with the aim of producing the prioritized list of non-functional requirements. Lastly, the IPA's process is concluded with computing the weight of each functional requirement using weighted average decision matrix as well as the calculated weights of non-functional requirements, producing the prioritized list of functional requirements.

The IPA supporting prototype, which was utilized in this study, is TIPA (Tool-supported Integrated Prioritization Approach) that is a C#-based implementation of IPA algorithm. In fact, TIPA provides the opportunity for decision makers to computerize the particular steps of the IPA process shown in Figure 6.1a. The software prototype guides the user to express his/her judgments between all possible pairs of functional and non-functional requirements in a similar way as the IPA approach does. Figure 6.2 shows a picture of the TIPA visual user interface. A thorough explanation of TIPA is given in Appendix A.

The software prototype facilitates the decision maker in the whole prioritization process. Particularly, once the decision maker is authenticated by the system, he/she is able to define the candidate functional and non-functional requirements that need to be prioritized. Afterward, TIPA presents him/her an agenda of specifying the preference values. In this situation, the decision maker can view the explanation of functional and non-functional requirements for every possible pair (i.e. FR and NFR). So, the decision maker can specify her/his preference value by identifying the importance degree of every non-functional requirement for achieving each functional requirement according to IPA nominal scale (see Table 4.3), through choosing one of the radio buttons indicated in Figure 6.2. When the decision maker presses 'Submit', the subsequent pair of requirements (i.e. FR and NFR) is viewable. After finishing the evaluations of all requirements, the system is capable of calculating and showing the prioritized list of

functional requirements, non-functional requirements along with their corresponding weights.

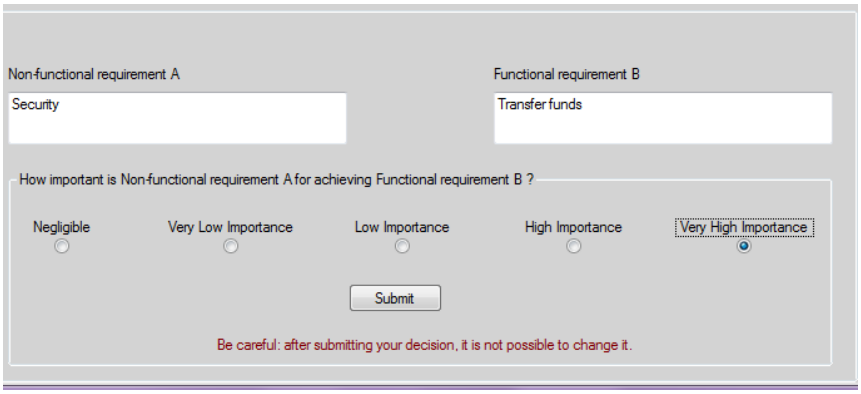


Figure 6.2: A picture of the visual user interface displaying the functional requirement, *Transfer funds*, versus the non-functional requirement, *Security*, under analysis with TIPA

- **AHP-based approach and its implementation**

In this section, a description of the AHP-based approach, which has been exploited in the first experiment, is given. This approach targeted at prioritizing functional and non-functional requirements separately. The AHP-based approach is basically proposed by the means of Analytic Hierarchy Process (AHP) method. The AHP method (Saaty, 1980) has been recognized as the most widely known MCDM (Multi Criteria Decision Making) method. This method exploits pairwise comparison strategy in such a way that the decision maker is required to compare all the available pairs of requirements together to figure out the relative weight of one requirement over another requirement.

Figure 5.1b sketches the steps of the AHP-based approach for prioritizing functional and non-functional requirements. In fact, to prioritize both functional and non-functional requirements using AHP-based approach, a decision maker needs to apply AHP method twice, first to apply AHP method on the identified functional requirements (see the left side of Figure 6.1b), and second to apply it on the candidate set of non-functional requirements (see the right side of Figure 6.1b). For example (see the left side of Figure 6.1b), having a collection of n functional requirements determined by a decision maker, the first step in AHP is to construct an $n \times n$ matrix which rows and

columns indicate the candidate functional requirements. Then, the decision maker determines his/her judgment for each pair of functional requirements by identifying a preference value which is between one to nine, where one expresses that the two functional requirements are equally important while nine represents the highest value of one functional requirement when compared to the other functional requirement. In fact, the decision maker has to perform $n \times (n-1)/2$ pairwise comparisons in total. The underlying values used for this purpose are presented in Table 6.2, which indicates a measure of specifying the decision maker's preference value for a given pair of requirements. Once all the possible pairs of functional requirements are assessed, the final priority order of functional requirements is calculated throughout the calculation of the principal eigenvector of the matrix (i.e., the eigenvector with the greatest normalized eigenvalue). Each element of the principal eigenvector signifies the priority value of the related functional requirement. The same procedure needs to be applied in order to prioritize non-functional requirements.

Table 6.2: Scales used in AHP method (J. Karlsson & Ryan, 1997)

Relative Intensity	**Definition**	**Explanation**
1	Of equal value	Two requirements are of equal value
3	Slightly more value	Experience slightly favors one requirement over another
5	Essential or strong value	Experience strongly favors one requirement over another
7	Very strong value	A requirement is strongly favored and its dominance is demonstrated in practice
9	Extreme value	The evidence favoring one over another is of the highest possible order of affirmation
2,4,6,8	Intermediate values between two adjacent judgments	When compromise is needed
Reciprocals	if requirement i has one of the above numbers assigned to it when compared with requirement j, then j has the reciprocal value when compared with i.	

In the following, the software prototype, which has been used as an implementation of AHP-based approach within this study, is described. In practice, CAHP (Csharp Analytic Hierarchy Process) has been exploited. It is a C#-based implementation of

AHP algorithm that enables the decision maker to computerize the particular steps of the AHP method. The software prototype directs the decision maker to specify pairwise comparisons between all possible pairs of requirements. Figure 6.3 indicates an image of the CAHP visual user interface where the decision maker is able to perform the pairwise comparison between each pair of requirements.

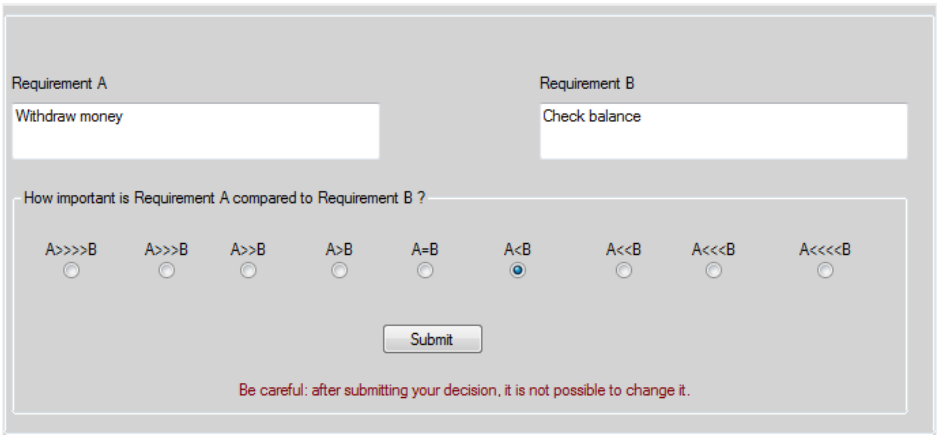


Figure 6.3: A picture of the visual user interface displaying two functional requirements, withdraw money, and check balance, under evaluation in CAHP

The CAHP software prototype supports the whole evaluation process. Note that the following process is applicable for either functional or non-functional requirements. Particularly, after the decision maker is authenticated by the system, he/she is able to define the candidate requirements (functional or non-functional) that need to be prioritized. Then, the software prototype displays the decision maker an agenda of $n \times (n-1)/2$ pairwise comparisons. The decision maker can view the explanation for each pair of requirements, determine her/his preference value by identifying the relative importance of one requirement over the other one according to AHP scales (see Table 6.2), through choosing one of the radio buttons shown in Figure 6.3. When the decision maker presses 'Submit', the subsequent pair of requirements is represented. When the evaluations of all requirements have been done completely, the system computes the final ranking of the requirements along with their priority values using AHP algorithm.

- **HAM-based approach and its supporting tool**

This section briefly explains the HAM-based approach, proposed by (Dabbagh, Lee, & Parizi, 2014) which has been used and analyzed in the second controlled experiment of this research. HAM-based approach is inspired from the Hybrid Assessment Method (HAM), first introduced in (Ribeiro et al., 2011).

In (Dabbagh et al., 2014), the authors proposed an approach by which indicated how HAM could be applied in the context of prioritizing functional and non-functional requirements, i.e., non-functional requirements are mapped to the criteria of HAM's process while functional requirements play the role of HAM's alternatives. This approach is called HAM-based approach.

The process of HAM-based approach for prioritizing functional and non-functional requirements is displayed in Figure 6.1c. The process is initialized with identifying functional and non-functional requirements that need to be prioritized by a decision maker. Then, HAM-based approach generates a pairwise comparison decision matrix, which rows and columns indicate the candidate non-functional requirements. Afterward, the decision maker is requested to determine his/her judgment for each pair of non-functional requirements by identifying a preference value according to HAM's scale (see Table 6.3). Once all the possible pairs of non-functional requirements are assessed, the process of prioritizing non-functional requirements can be done. After that, functional and non-functional requirements are inserted into rows and columns of the decision matrix, respectively. Then, the decision maker is asked to determine his/her judgment regarding the importance degree of each non-functional requirement with respect to each functional requirement based on the scales defined in Table 6.3. Once the process of eliciting the importance degree of each non-functional requirement for all

functional requirements is completed, the process is concluded by performing the prioritization of functional requirements.

Table 6.3: Scales used in HAM method (adapted form (Ribeiro et al., 2011))

HAM scale	**Interpretation**
9/1	Extremely high importance
9/3	Very high importance
9/5	High importance
9/7	Medium high importance
9/9	Equal importance
7/9	Medium low importance
5/9	Low importance
3/9	Very low importance
1/9	Extremely low importance

The software prototype, which has been used in this research, as an implementation of HAM-based approach, is CHAM (Csharp Hybrid Assessment Method). CHAM is implemented using C# programming language to assist decision makers by automating the particular steps of the HAM-based approach displayed in Figure 6.1c. The software prototype directs the decision maker to specify pairwise comparisons between all possible pairs of non-functional requirements as well as express his/her opinions between all possible pairs of functional and non-functional requirements in a similar way as the HAM-based approach does. Figure 6.4 and Figure 6.5 show two main pictures of the CHAM visual user interface.

The software prototype facilitates the decision maker in the whole prioritization process. Particularly, once the decision maker is authenticated by the system, he/she is able to define the candidate functional and non-functional requirements that need to be prioritized. Then, the software prototype shows the decision maker an agenda of $n \times (n-1)/2$ pairwise comparisons of non-functional requirements (see Figure 6.4). So, the decision maker can view the explanation for each pair of non-functional requirements, determine her/his preference value by identifying the relative importance

of one non-functional requirement over the other one through choosing one of the radio buttons shown in Figure 6.4. When the decision maker presses ‘Submit’, the subsequent pair of non-functional requirements is represented. When the evaluations of all non-requirements requirements have been done completely, the system computes the final ranking of the non-functional requirements along with their priority values. Afterward, CHAM provides the decision maker with the explanation of functional and non-functional requirements for every possible pair (i.e. FR and NFR). So, the decision maker can specify her/his preference value by identifying the importance degree of every non-functional requirement for achieving each functional requirement, through choosing one of the radio buttons indicated in Figure 6.5. When the decision maker presses ‘Submit’, the subsequent pair of requirements (i.e. FR and NFR) is viewable. After finishing the evaluations of all requirements, the system is capable of calculating and showing the prioritized list of functional requirements along with their corresponding weights.

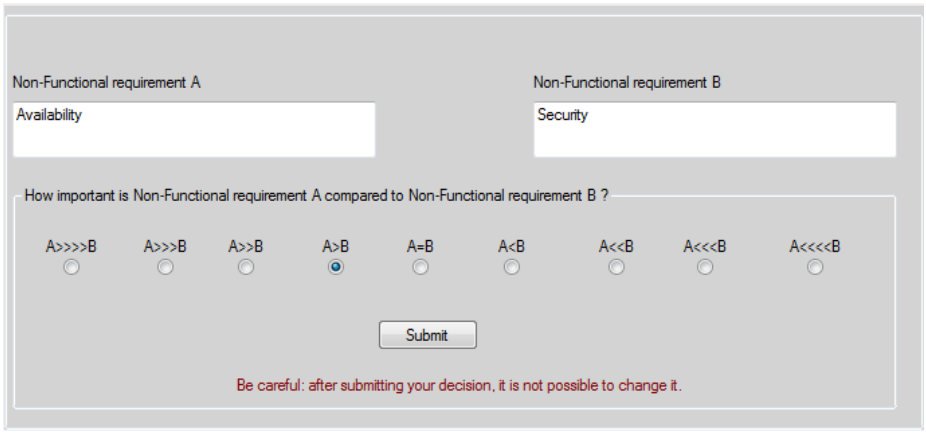


Figure 6.4: A picture of pairwise comparison of two non-functional requirements, *Availability*, and *Security*, using CHAM

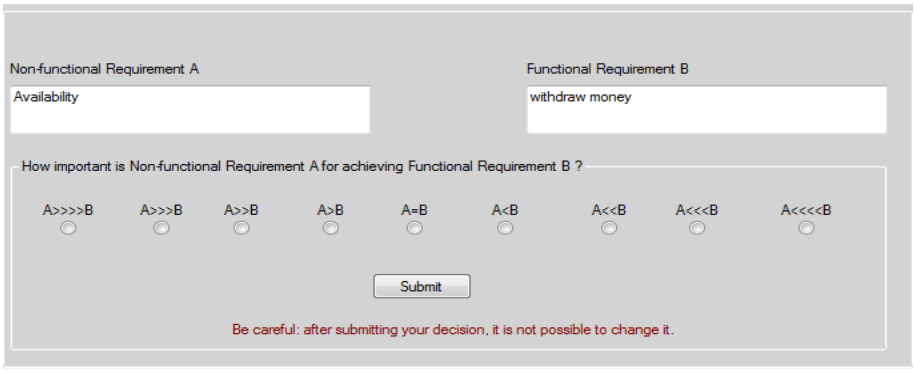


Figure 6.5: A picture of the visual user interface displaying the functional requirement, *withdraw money*, versus the non-functional requirement, *Availability*, under analysis with CHAM

Dependent variables and measures:

Based on the hypotheses formulated in Section 6.1.2.2, three dependent variables have been considered and measured in both controlled experiments: *actual time-consumption*, *ease of use*, and *accuracy of results*. By conducting a systematic mapping study on the various empirical studies within the requirements prioritization area, it was found that the most frequently measured dependent variables in these studies are the accuracy of results, the time needed to perform the prioritization task, and the ease of use (Pergher & Rossi, 2013). This is also in line with the analysis of comparative evaluations of different prioritization methods, which has been presented in Table 2.15, where the three mentioned variables have been measured in all reviewed studies. The reason for choosing these properties has been advocated by the fact that to make a prioritization process applicable in commercial software development, it should be fast and simple while providing accurate results (J. Karlsson & Ryan, 1997).

To measure the *actual time-consumption*, i.e. the first dependent variable, start time and end time of the prioritization task for each prioritization approach have been monitored automatically using the prioritization tools, and then their difference computed.

Ease of use represents how easy a decision maker is able to perform the prioritization process using a given prioritization approach. In both experiments, the second dependent variable, i.e. *ease of use*, was measured in two ways by the means of the two post-questionnaires: post-test 1A and post-test 2A (post-questionnaires are presented in Appendix E). Immediately after working with each tool-supported prioritization approach, the test subjects carried out the first post-test, i.e. post-test 1A, by answering the following question*: How easy was to perform the actual prioritization using the approach?* (In the experiments, the term *approach* is replaced with IPA or AHP-based approach or HAM-based approach) (The test subject was asked to choose an integer value ranging from one to five according to Likert scale (Likert, 1932) where one indicated very low and five represents very high). In the post-test 2A which has been done after working with both tool-supported prioritization approaches, the test subjects were asked to answer the following question: *Which approach did you find easier to use?* The test subject was asked to select one option:

- For Experiment 1: IPA, AHP-based approach, they are equal.
- For Experiment 2: IPA, HAM-based approach, they are equal.

In this context, a more accurate prioritization approach is the one that produces the ranking results (i.e. prioritized lists of functional and non-functional requirements) which better reflects the participants' opinions. Suppose that prioritization approach X is considered to produce more accurate results than prioritization approach Y. This brings the meaning that the prioritized list of functional requirements as well as the prioritized list of non-functional requirements which have been produced by using approach X are closer to the perception of the participants who have used both approaches X and Y. In both experiments of this study, the *accuracy of results*, as the third dependent variable, was measured in terms of *expected* accuracy and *perceived*

accuracy using two post-questionnaires: post-test 1B and post-test 2B. The *expected* accuracy was measured through the post-test 1B where each test subject was asked to answer the following question immediately after working with each tool-supported prioritization approach once he/she was provided with prioritized lists of functional and non-functional requirements produced by the given tool-supported approach based on his/her judgements: *How accurate did you find the results produced by the approach?* (In the experiment, the term *approach* is replaced with IPA or AHP-based approach or HAM-based approach) (The test subject was asked to choose an integer value ranging from one (very low) to five (very high) according to Likert scale (Likert, 1932)). In addition, the *perceived* accuracy was measured by the means of the second post-questionnaire, i.e. post-test 2B. The post-test 2B was done one week after each subject session. Therefore, each test subject was given two sheets of prioritized lists. Each sheet included the prioritized list of 15 functional requirements as well as the prioritized list of 5 non-functional requirements that had been generated by a given tool-supported approach based on the judgements of the subject who received the sheets. The critical point here is that the subjects were not aware of which tool produced the lists. They were only requested to label a sheet that included the lists (i.e. prioritized list of 15 functional requirements and prioritized list of 5 non-functional requirements) which better suited their views.

6.1.2.4 Selection of subjects

The sampling technique, which has been exploited in both experiments to select the test subjects, was convenience sampling. Convenience sampling is a type of non-probability sampling technique by which the most convenient and available persons are chosen as test subjects (Wohlin et al., 2012).

6.1.2.5 Experiment design

The design, which has been adopted in both experiments, is the paired comparison design (see Table 6.4), which is a particular type of one factor with two treatments (Wohlin et al., 2012). In this design, each subject applied two prioritization approaches (i.e. treatments) on the same set of requirements (i.e. objects). The order of executions was given at random to each subject in order to minimize the effect of the execution order on the final results.

Table 6.4: The paired comparison design used for controlled experiments

	Experiment 1		**Experiment 2**	
Group	Prioritization task 1	Prioritization task 2	Prioritization task 1	Prioritization task 2
1	IPA	AHP-based	IPA	HAM-based
2	AHP-based	IPA	HAM-based	IPA

6.1.2.6 Subjects

Both experiments have been performed with 20 real subjects. Participants of the first experiment included 16 Ph.D. candidates of Computer Science who have served as research assistants at the University of Malaya and 4 Ph.D holders in Software Engineering who are currently working as post-doc at University of Malaya, Kuala Lumpur, Malaysia. All the subjects of the second experiment were Ph.D. students of Computer Science at University of Malaya, Kuala Lumpur, Malaysia. It should be highlighted that the subjects of the second experiment were different from the subjects of the first experiment. This was done to prevent biasing the results that may happen due to the previous working experience of the subjects with approaches. All the subjects who participated in the experiments have had a good knowledge about different types of software requirements, requirements prioritization approaches and software engineering

domain in general. Therefore, we believe that the selected test subjects for the experiments could be considered close to professionals.

6.1.2.7 Objects

Same objects have been used in both experiments. In practice, 15 functional requirements and 5 non-functional requirements (totally twenty requirements) of ATM (Automated Teller Machine), CDM (Cash Deposit Machine), and CQM (Check Deposit Machine), were selected as the objects of the controlled experiments. ATM is a banking subsystem that allows bank customers to access basic bank services from remote locations. CDM and CQM are self-service banking subsystems that allow bank customers to make deposits and payment transactions using cash and check, respectively. Clearly, ATM, CDM, and CQM must provide some services (i.e. functional requirements) to their users. At the same time, these functional requirements should satisfy certain quality attributes (i.e. non-functional requirements) which could have great impression on the overall users' satisfaction.

Functional requirements of ATM, CDM, and CQM, which were used in the experiments, are given in Table 6.5 (see left column of Table 6.5). Non-functional requirements of ATM, CDM, and CQM express how well the functional requirements should be performed. For example, "*bank customer should be able to withdraw cash in less than 3 seconds*". This statement indicates that to perform functional requirement, *withdraw cash*, non-functional requirement, *performance*, needs to be considered as an important quality attribute. The non-functional requirements which have been chosen for this study were system quality attributes such as availability, security, usability, performance, and reliability. The definitions of non-functional requirements are provided in Table 6.5 (see the right column of Table 6.5).

These requirements were quite independent and high level, which have been chosen taking into consideration that they need to possibly be clear enough even for novice users. All requirements were represented as simple textual descriptions. The prioritization was performed without taking into account dependencies among requirements.

Table 6.5: Objects of the controlled experiments

Functional requirement	**Non-functional requirement**
Withdraw cash Check balance Deposit cash	Availability: the percentage of time that the system is in operation to provide its intended function
Transfer funds Change PIN number View transactions history	Security: the extent to which access to desired function by unauthorized persons can be controlled while still providing its function to users
Bill payment Print transaction receipt Deposit check	Usability: the extent to which a user is able to understand, learn, use and being attracted to a function
Top up mobile phone Loan payment Print transaction history	Performance: the extent to which how fast the system can interact with user to perform the desired function
Change withdraw limit Activate overseas services Credit card payment	Reliability: the extent to which the system can be expected to perform its intended function with required precision

6.1.2.8 Instrumentation

To perform each controlled experiment and monitor it as well, a software tool has been developed. The software tool not only included the implementations of the prioritization approaches used in the experiment, but also provided the participant with a sufficient amount of information regarding the definition of requirements prioritization, definitions of functional and non-functional requirements in general as well as explaining the differences between functional and non-functional requirements. In particular, the software tool included the definitions of 15 functional requirements

and 5 non-functional requirements, which were used as objects in each experiment. In addition, it provided the subject with a general introduction about the prioritization approaches of a given experiment as well as the guidelines on how they might work. Furthermore, to collect and measure the required data, i.e. the dependent variables which were explained in Section 6.1.2.3, some standard techniques and questionnaires were embedded in the designed tool.

6.1.2.9 Threats to validity

This section discusses the main threats to validity, which could bias the results of both controlled experiments: internal, external, construct and conclusion validity threats (Wohlin et al., 2012).

Internal validity concerns the relationship between the treatments and the outcome of the experiment. This type of validity threats, which could possibly bias the outcome of the experiments, is the fatigue effect. The subjects may become exhausted during each experiment, which might have impression on their concentration. In other words, the test subjects who performed the prioritization tasks with 20 requirements (15 FRs and 5 NFRs) could get tired and bored. To minimize the effect of this threat, the number of requirements was kept low with the purpose of conducting experiment in less than two hours to prevent the subjects form feeling fatigued. In addition, an obligatory break was considered between the two tasks to mitigate this threat. Moreover, the time of each subject session was arranged according to his/her preference so that he/she could be fresh at that time.

External validity focuses on theory about the relationship between the treatment and the outcome. Can the experiment be generalized outside the scope of the experiment? Threats to external validity may restrict the generalizability of the experiment to the

industrial setting. External validity threats must be taken into consideration when an experiment is needed to be conducted with participating students and researchers. Since the test subjects of the experiments were sampled from software engineering Ph.D. students and researchers, this type of threat needs to be addressed. Several studies have discussed the similarities and differences of using students or professionals in the software engineering experimentations. Some of them have shown that there is no significant difference between students and professionals (Svahnberg, Aurum, & Wohlin, 2008) while some others argued that the results of using students and professionals are not the same (Berander, 2004b). Another author (Tichy, 2000) argued that if the results of an experiment, which uses students as subjects, indicate that one approach overcomes the other one in terms of a given property, it is highly probable that practitioners would get the same conclusion. However, it is challenging to deduce that the results of conducted experiments within this study could be generalized to industrial environment. However, to minimize this threat, research students were selected as test subjects of the experiments taking into consideration that the results of using research students might be more reliable compared to classroom students (Danesh & Ahmad, 2009). In addition, the test subjects who participated in the experiments represented a population with sufficient education about requirements, requirements prioritization approaches used in the experiments, and have had industrial working experience. Therefore, selected test subjects of the experiments could be considered close to professionals. The other external threat is concerned with the small number of requirements used in the experiments. Despite rather small number of requirements (20) was used in the experiments to conquer the fatigue effect and as a result of that, amend the internal threat, it limits the chance of generalizing the results to cases where a larger couple of requirements need to be prioritized. In many practical situations, the total number of requirements is actually larger, and accordingly, the outcomes obtained in

this particular research could possibly be credible when the prioritization is conducted on a subset of the requirements of a large-scale system like the situation that only the requirements for a specific subsystem are needed to be prioritized. It is difficult to conclude that increasing the number of requirements would certainly result in exactly the identical results. Hence, future duplications and experimental studies need to be conducted to analyze the findings in some situations where more requirements would be prioritized. Furthermore, threats to external validity are also associated with the functional and non-functional requirements used as experimental objects. All of the subjects were familiar with the experimental objects (i.e. functional and non-functional requirements of ATM, CDM, and CQM). Though this makes the situation quite realistic, further investigations with various kinds of objects as well as subjects are needed to confirm or contradict the outcomes found in the experiments. Lastly, the time complexity of the algorithms as well as GUIs utilization are actually insignificant with regards to the time needed by a decision maker to perform the prioritization process. Hence, the calculated actual time-consumption pertains basically to the time of individuals' decision-making process. Therefore, this could not be considered as a threat to validity.

Construct validity threats concern the relationship between theory and observation. The objective dependent variable time was measured by the means of the prioritization tools automatically, as done also in (Perini et al., 2009). Dependent variables such as ease of use and accuracy are subjective variables, i.e. their measurement relies on how they are perceived by the subjects and could possibly influenced by the subjects' past expertise and information about a specific issue (Perini et al., 2009). Also, the ideal target ranking is not identified in advance, in general. Thus, these factors make it difficult to measure the ranking accuracy. In this research, an accurate prioritization approach is one that generates a priority order which better reflects the decision maker's

viewpoint, as in (Perini et al., 2009). Furthermore, following (Perini et al., 2009), accuracy has been measured in two ways. To collect the viewpoints of the test subjects regarding the ease of use and accuracy of prioritization approaches, standard questionnaires were designed. In a situation like the performed experiments where the test subjects are aware of measuring the time-consumption of performing the prioritization process, it is possible that the time-consumption could be affected. However, the test subjects of experiments were not aware of measuring the other two variables (i.e. ease of use and accuracy) when performing each experiment. Therefore, only the actual time-consumption may have been influenced.

The threats to conclusion validity are primarily related to the statistical analyses underlying the conclusion, measures, implementation, and unexpected interruptions during experiments execution. In the following, the researcher explains that none of these threats could affect the results. Robust and appropriate statistical tests were carried out to investigate the null hypotheses. In some situations, non-parametric tests were utilized in preference to parametric tests since the requirements to apply parametric tests were failed to meet. Furthermore, measures and implementation were considered to be reliable. Both objective and subjective measures were used. The researcher measured the objective dependent variable of the experiments, i.e. actual time-consumption, in an automatic way to make the outcomes more reliable. Moreover, to measure the subjective dependent variables of the experiments, which are ease of use and accuracy of results, the researcher designed some standard questionnaires and test (e.g. blind test used for measuring perceived accuracy). All of the participants used the same implementation of prioritization approaches as well. Each test subject was isolated in the research laboratory to make sure that nobody or nothing may disturb him/her and thereby influence the results. Mobile and any other smart devices were switched off. However, one threat, which may influence statistical power, is caused by the limited

number of test subjects who participated in the experiment, since only 20 subjects took part in the experiment. Therefore, further experiments would be carried out with participating more test subjects to have more rigorous statistical analysis.

6.1.3 Experiment execution

Before running the experiment, the researcher provided a brief presentation to each subject who participated in the experiment with the purpose of giving an introduction on the requirements prioritization definition and on available prioritization approaches, IPA, AHP-based approach (for only the participants of Experiment 1), or HAM-based approach (for only the participants Experiment 2). All selected functional and non-functional requirements of ATM, CDM, and CQM were explained to them. Moreover, the researcher provided a short instruction on how to work with the tools, TIPA, CAHP (for only the participants of Experiment 1), or CHAM (for only the participants of Experiment 2). These tools were also tested on a small number of requirements before applying them to the actual experiment. After that, each test subject was asked to fill up two pre-questionnaires: pre-questionnaire 1 and pre-questionnaire 2.

Pre-questionnaire 1 was designed in order to get some general information from each participant of the experiment. This information includes educational level, position, industrial experience, etc. The second pre-questionnaire, i.e. pre-questionnaire 2, targeted at obtaining information with respect to the familiarity of each subject regarding the requirements of ATM, CDM, and CQM. It also captured knowledge of each test subject about two prioritization approaches used in a given experiment.

The experiment occurred inside a research laboratory room provided with a computer. A computer with access to TIPA, CAHP (for only participants of Experiment 1), or CHAM (for only participants of Experiment 2) has been provided to every single

subject. The twenty requirements, including 15 functional requirements and 5 non-functional requirements of ATM, CDM, and CQM, were inserted in advance to the software tools. Given the same set of functional and non-functional requirements, the subject executed the two prioritization tasks sequentially. The order of executions was assigned randomly in order to minimize the effect of the order. Each subject's session took more than an hour, including preliminary presentation and providing instruction on the tools, along with the limited break of 5 minutes between the two prioritization tasks. An individual session was considered for each participant of the experiment.

After working with each tool-supported prioritization approach, each subject filled up the post-test 1 in order to measure the ease of use and expected accuracy of prioritization approaches. Furthermore, after working with both prioritization approaches, the first part of post-test 2, i.e. post-test 2A was given to each test subject to capture the ease of use again. Lastly, one week after the experiment, the second part of post-test 2, i.e. post-test 2B, was provided to each subject in order to capture the perceived accuracy.

6.1.4 Experiment results and analysis

This section describes the notable results achieved from the first experiment as well as the second experiment. For both experiments, the researcher initially performed descriptive analysis using Microsoft Excel. In addition, statistical analysis has been carried out using IBM SPSS Statistics version 21 to reject or accept the null hypotheses, which were formulated in Section 6.1.2.2. It should be noted that a 5% significance level was used for hypothesis testing. Raw data of Experiment 1 and Experiment 2 are given in Appendix C and Appendix D, respectively.

6.1.4.1 Results of Experiment 1

- **RQ1-1:** How fast are IPA and AHP-based approach when applied to perform the prioritization process?

Some intuition can be achieved by taking a look at the results indicated in Table 6.6 and the boxplot sketched in Figure 6.6 that compares the actual time-consumption to perform the prioritization task using IPA and AHP-based approach. These values were calculated automatically by the prioritization tools through recording the start time and end time of performing each prioritization task. It is quite obvious that the time required to perform the prioritization task is smaller with IPA than with AHP-based approach. As Table 6.6 shows, the difference in average actual time-consumption between the two approaches collected from 20 subjects of the first experiment is 390 seconds, which corresponds to a reduction of 43%. This is also shown in Figure 6.6 where the median value is higher for AHP-based approach than for IPA.

Table 6.6: Average actual time-consumption for the prioritization task using IPA and AHP-based approach

	IPA	**AHP-based approach**	**Difference (AHP-based,IPA)**
Actual time-consumption	509 sec	899 sec	390
%	-	-	43%

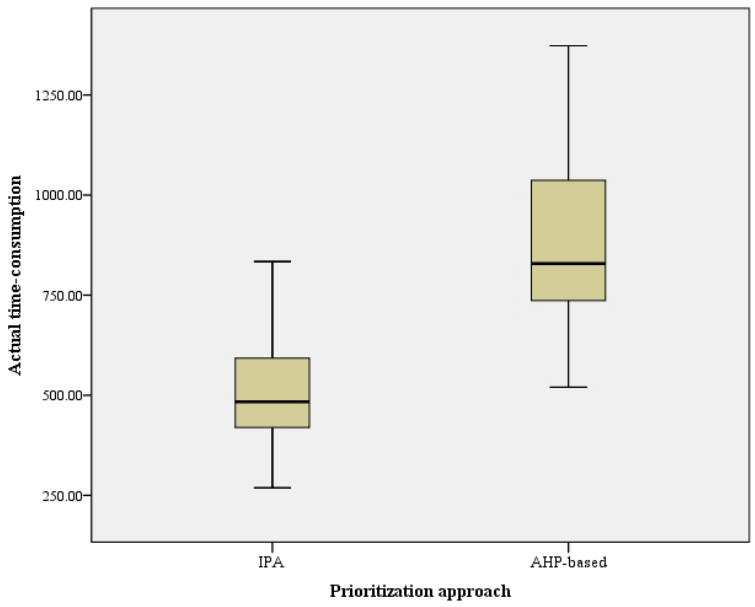


Figure 6.6: Boxplot of the actual time-consumption associated with the two evaluated prioritization approaches

Before starting to test the null hypothesis (H_{0time}) (see Section 6.1.2.2), the researcher checked the distribution of data to determine whether it is normal or not. To perform the normality test, the Shapiro-Wilk test was used. According to information gained from the Shapiro-Wilk test (see Table 6.7), it has been found that the distribution of data is normal with respect to actual time-consumption, as the p-value was greater than 0.05 for both IPA and AHP-based approach.

Table 6.7: Normality test of data extracted from Experiment 1 using Shapiro-wilk test

	Shapiro-Wilk														
	Time-consumption			Ease of use						Accuracy					
				post-test 1A			post-test 2A			expected			perceived		
	statics	df	Sig.	statics	df	Sig.	statics	df	Sig.	statics	df	Sig.	statics	df	Sig.
IPA	0.952	20	0.400	0.785	20	0.001	0.695	20	0.000	0.857	20	0.007	0.544	20	0.000
AHP-based	0.969	20	0.729	0.598	20	0.000				0.812	20	0.001			

According to Table 6.7, the data were normally distributed. On top of that, the design type that was used for this experiment was one factor with two treatments (see Section 6.1.2.5). One of the analysis methods, which appears to be suitable for such situation, is a parametric test called t-test. Therefore, to test the null hypothesis (H_{0time}), t-test was applied. The test's result shows that the difference between the average actual time-consumption of two approaches, IPA and AHP-based approach is significant, as the p-value turned out to be 0.034, which is less than a 5% significance level. Thus, the first null hypothesis was rejected and it could be easily concluded that the IPA approach is a faster approach than the AHP-based approach for performing the prioritization task.

- **RQ1-2:** Which approach, between IPA and AHP-based approach, is easier to use?

The ease of use was measured through two post questionnaires (see Section 6.1.2.3). In the first post questionnaire, post-test 1A, which was done immediately after working with each tool-supported approach, each participant was asked to identify his/her judgment regarding the approach's ease of use for performing the actual prioritization

using Likert scale. The results of this post questionnaire are provided in Table 6.8 where for IPA, majority of subjects believed that it has a high degree of ease of use for performing the prioritization, whereas for AHP-based approach, majority of participants found it a low-level of ease of use.

Table 6.8: Results of ease of use collected from post-test 1A of Experiment 1

Likert scale / Prioritization approach	1	2	3	4	5
IPA	-	5	1	11	3
AHP-based	-	14	1	5	-

To investigate the second null hypothesis ($H_{0easeofuse}$) (see Section 6.1.2.2), a non-parametric test, called Mann-Whitney test, was applied with considering the fact that the data were not distributed normally (see Table 6.7). As a result, the second null hypothesis was rejected since the significance value (p-value) calculated using Mann-Whitney test is 0.002, which is less than 0.05. Thus, it could be concluded that the IPA approach is easier to use than AHP-based approach.

To confirm the results regarding the ease of use, the researcher performed the post-test 2A after the subjects worked with both prioritization approaches. As illustrated in Section 6.1.2.3, they were requested to answer the following question: Which approach did you find easier to use (select one option: IPA, AHP-based approach, They are equal)? Among the 20 subjects, 14 found IPA easier to use than AHP-based approach. Only 5 subjects stated that AHP-based approach was easier than IPA and 1 subject found them equally easy. As can be observed in Table 6.9, this corresponds to that 70% of the participants found IPA easier, while 25% found AHP-based approach easier and 5% found the two approaches equally easy to use.

Table 6.9: Results of ease of use extracted from post-test 2A of Experiment 1

Ease of use	IPA	AHP-based	Equally easy
	14	5	1
%	70%	25%	5%

Due to the nature of variables as well as the fact that the data were not normally distributed (see Table 6.7), the researcher decided to investigate the second null hypothesis using Chi-Square test by comparing the number of responses in favour of IPA to the total number of responses. It became apparent that there is a statistically significance difference, as p-value=0.001 (<0.05). Hence, the second null hypothesis was rejected again and it could be concluded that the IPA approach is easier to use than AHP-based approach.

- **RQ1-3:** Which approach, between IPA and AHP-based approach, produces results that are more accurate?

As discussed in Section 6.1.2.3, accuracy of results produced by two approaches was measured in two ways, including expected accuracy and perceived accuracy. To measure the expected accuracy, the participants filled up the post-test 1B immediately after working with each tool-supported prioritization approach and seeing the results produced by a given approach. Table 6.10 summarizes the results collected from the post-test 1B which indicates the opinions of test subjects with respect to the expected accuracy of IPA and AHP-based approach. As can be seen in Table 6.10, the opinions of test subjects seem to be different for the two approaches. Therefore, in order to get understanding of which approach produces more accurate results, the third null hypothesis was tested statistically.

Table 6.10: Results of expected accuracy collected from post-test 1B of Experiment 1

Likert scale / Prioritization approach	1	2	3	4	5
IPA	-	4	3	12	1
AHP-based	-	8	8	2	2

To test the third null hypothesis ($H_{0accuracy}$) (see Section 6.1.2.2), the researcher decided to apply a non-parametric test, called Mann-Whitney test, taking into the account that the data were not distributed normally (see Table 6.7). Hence, it was observed that the difference between two approaches with respect to the expected accuracy is statically significant since the p-value turned out to be 0.033 (<0.05). Therefore, the third null hypothesis was rejected and it could be drawn the conclusion that IPA produces more accurate results than AHP-based approach.

Furthermore, the perceived accuracy was measured by the means of post-test 2B. Therefore, each test subject was given two pairs of prioritized lists, including the 15 prioritized functional requirements as well as the 5 prioritized non-functional requirements ordered based on the priority order that had been calculated by himself/herself during the subject session working with the two tools. The subjects were not aware of which tool produced the lists. They were only requested to label the list comprising the priority order which better suited their views. As Table 6.11 illustrates, 5 test subjects (25%) found the AHP-based approach lists are more accurate whereas 15 subjects (75%) declared that IPA was more accurate. Here also the difference is statistically significant, as the p-value turned out to be 0.025 (<0.05) in a Chi-Square test. Chi-Square test was used since the data were not normally distributed with respect to perceived accuracy (see Table 6.7). Therefore, the third null hypothesis was rejected and thus it could be concluded that IPA produces more accurate results than AHP-based approach.

Table 6.11: Results of perceived accuracy collected from post-test 2B of Experiment 1

Perceived accuracy	**IPA**	**AHP-based**	**Similar**
	15	5	0
%	75%	25%	0%

- Effects of other variables:

The researcher analyzed the effects of other variables, including execution order and experience in requirements prioritization, on dependent variables, i.e. actual time-consumption, ease of use, and accuracy of results, to investigate whether these variables affected the results of the experiment. Execution order indicates the order of executing the prioritization approaches by a subject. As mentioned before, half of the subjects started the prioritization task by executing IPA followed by AHP-based approach while the other subjects began with AHP-based approach followed by IPA. Moreover, the subjects who participated in the experiment could be classified into five categories (based on Likert scale (Likert, 1932)), according to their familiarity with requirements prioritization.

The results of analysis are presented in Table 6.12, where it can be concluded that neither execution order nor experience in requirements prioritization did not have any significant effect on time, ease of use, and accuracy of results, since the p-values are greater than 0.05. It should be noted that the analysis was performed using Mann-Whitney test.

Table 6.12: Analysis the effect of other variables on dependent variables of Experiment 1

Factor	Execution order (IPA first vs. AHP-based first)	Experience in requirements prioritization (very low, low, medium, high, very high)
Dependent variable	p-value	p-value
Time	0.45	0.528
Ease of use	0.423	0.792
Accuracy	1.0	0.584

6.1.4.2 Results of Experiment 2

- **RQ2-1:** How fast are IPA and HAM-based approach when applied to perform the prioritization process?

The average actual time-consumption of the two tool-supported prioritization approaches, i.e. IPA and HAM-based approach, was calculated and provided in Table 6.13 where the average actual time-consumption was 509 seconds and 635 seconds for IPA and HAM-based approach, respectively. In other words, IPA needed 20% less time than HAM-based approach to perform the prioritization task. This difference is also indicated in boxplots of Figure 6.7 where the median value is higher for HAM-based approach than IPA.

Table 6.13: Average actual time-consumption for the prioritization task using IPA and HAM-based approach

	IPA	**HAM-based approach**	**Difference(HAM-based,IPA)**
Actual time-consumption	509 sec	635 sec	126
%	-	-	20%

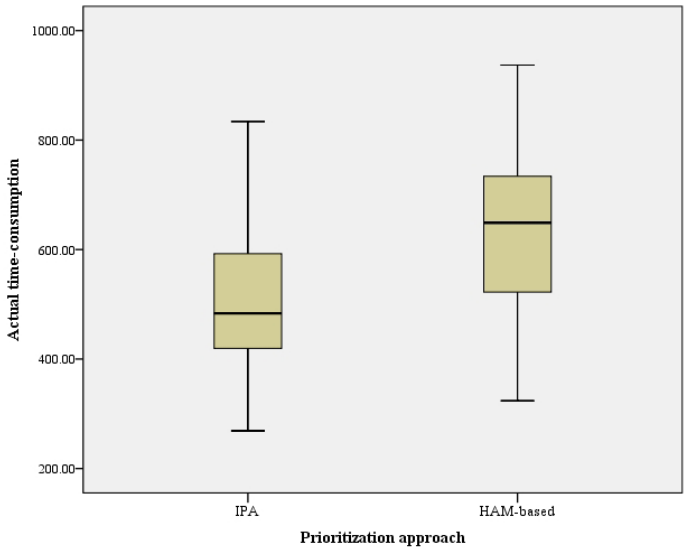


Figure 6.7: Boxplot of the actual time-consumption associated with IPA and HAM-based approach

Due to normally distribution of the data (see the most left columns of Table 6.14), the researcher decided to apply the parametric t-test to test the null hypothesis (H_{0time}) (see Section 6.1.2.2). The t-test's outcome indicated that the difference between the average actual time-consumption of IPA and HAM-based approach is significant on the 5% level, as the p-value turned out to be 0.016. Thus, the first null hypothesis was rejected and it could be simply drawn the conclusion that the IPA approach is faster than the HAM-based approach.

Table 6.14: Normality test of data extracted from Experiment 2 using Shapiro-Wilk test

	Shapiro-Wilk														
	Time-consumption			Ease of use						Accuracy					
				post-test 1A			post-test 2A			expected			perceived		
	statics	df	Sig.	statics	df	Sig.	statics	df	Sig.	statics	df	Sig.	statics	df	Sig.
IPA	0.952	20	0.400	0.785	20	0.001	0.771	20	0.000	0.785	20	0.001	0.495	20	0.000
HAM-based	0.984	20	0.971	0.800	20	0.001				0.816	20	0.002			

- **RQ2-2:** Which approach, between IPA and HAM-based approach, is easier to use?

After working with each tool-supported approach, participants filled up the first post questionnaire to answer the question "*How easy was to perform the actual prioritization using the approach?*". The results of this questionnaire are given in Table 6.15.

Table 6.15: Results of ease of use obtained from post-test 1A of Experiment 2

Likert scale Prioritization approach	1	2	3	4	5
IPA	-	5	1	11	3
HAM-based	-	8	8	4	-

Due to the non-normally distribution of the data (see the middle columns of Table 6.14), the non-parametric Mann-Whitney test was applied to investigate the second null hypothesis ($H_{0easeofuse}$) (see Section 6.1.2.2). The test's result showed that the difference between ease of use of IPA and HAM-based approach is significant due to the reason that the p-value calculated using Mann-Whitney test is 0.01, which is less than 5%. Thus, second null hypothesis was rejected and it could be concluded that the IPA approach is easier to use than HAM-based approach.

Moreover, the results extracted from the post-test 2A confirmed the results (see Table 6.16) where 60% of the test subjects found IPA easier, while 30% found HAM-based approach easier and 10% found the two approaches equally easy to use.

Table 6.16: Results of ease of use extracted from post-test 2A of Experiment 2

Ease of use	**IPA**	**HAM-based**	**Equally easy**
	12	6	2
%	60%	30%	10%

Here, the second null hypothesis was investigated again in a Chi-Square test because the data were not normally distributed (see the middle columns of Table 6.14) by comparing the number of responses in favour of IPA to the total number of responses. The test's result indicated that there is a statistically significance difference, as p-value is 0.022 (<0.05). Hence, the second null hypothesis was rejected and it could be concluded that the IPA approach is easier to use than HAM-based approach.

- **RQ2-3:** Which approach, between IPA and HAM-based approach, produces results that are more accurate?

The results collected from the post-test 1B, which illustrate the viewpoints of test subjects with respect to the expected accuracy of IPA and HAM-based approach are provided in Table 6.17. As can be seen in Table 6.17, it seems that there is a difference between these approaches in terms of expected accuracy. To investigate the difference between these approaches in a statistical manner, the non-parametric Mann-Whitney test, was applied since the distribution of the data was not normal (see the most right columns of Table 6.14). The test's result showed that the difference between two approaches with respect to the expected accuracy is statically significant since the p-value turned out be 0.048 (<0.05). Therefore, the third null hypothesis was rejected and it could be concluded that IPA produces more accurate results than HAM-based approach.

Table 6.17: Results of expected accuracy collected from post-test 1B of Experiment 2

Likert scale / Prioritization approach	1	2	3	4	5
IPA	-	5	1	11	3
HAM-based	-	5	9	6	-

However, the researcher also measured the perceived accuracy of IPA and HAM-based approach through a blind test by the means of the post-test 2B. The results of the

post-test 2B are given in Table 6.18 where it can be observed that 16 test subjects (80%) believed that the prioritized lists of functional and non-functional requirements produced by IPA were more close to their ideal rankings whereas only 4 subjects (20%) declared that results of HAM-based were more accurate. Here also the difference is statistically significant, as the p-value turned out to be 0.007 (<0.05) in a Chi-Square test. Chi-Square test was used because the data were not normally distributed with respect to perceived accuracy (see the most right columns of Table 6.14). Therefore, the third null hypothesis was rejected, and thus, it could be drawn the conclusion that IPA produces more accurate results than HAM-based approach.

Table 6.18: Results of perceived accuracy collected from post-test 2B of Experiment 2

Perceived accuracy	**IPA**	**HAM-based**	**Similar**
	16	4	0
%	80%	20%	0%

- Effects of other variables:

Median test was used to analyze the effects of execution order and experience of subjects in requirements prioritization, on actual time consumption, ease of use, and accuracy of results. As can be observed in Table 6.19, these variables did not significantly affect the dependent variables of the second experiment, as the p-values are greater than 0.05.

Table 6.19: Analysis the effect of other variables on dependent variables of Experiment 2

Factor	Execution order (IPA first vs. HAM-based first)	Experience in requirements prioritization (very low, low, medium, high, very high)
Dependent variable	p-value	p-value
Time	0.226	0.528
Ease of use	0.165	0.813
Accuracy	1.0	0.570

6.1.5 Discussion

Regarding Experiment 1, the main results extracted from this experiment are summarized in Table 6.20 where they are characterized in terms of hypothesis, dependent variable, statistical test, p-value, result and direction. Based on the results obtained from this experiment, IPA outperforms AHP-based approach with respect to the actual time-consumption taken for performing the prioritization task, ease of use, and accuracy of results perceived by test subjects. It should be highlighted that these results have been achieved in a situation where both IPA and AHP-based approach have been applied by the same set of test subjects on the same set of functional and non-functional requirements.

Table 6.20: Summary of hypotheses testing of Experiment 1 for comparing IPA and AHP-based approach

Hypothesis	Dependent variable	Statistical test	p-value	Result	Direction
$\mathbf{H_{0}}_{\text{time}}$	Actual time-consumption	T-test	0.034	rejected	IPA
$\boldsymbol{H_{0}}_{easeofuse}$	Ease of use	Mann-whitney Chi-square	0.002 0.001	rejected	IPA
$\mathbf{H_{0}}_{\text{accuracy}}$	Expected accuracy Perceived accuracy	Mann-whitney Chi-square	0.033 0.025	rejected	IPA

The observations concerning the actual time-consumption of the two prioritization approaches, IPA and AHP-based approach, indicate that, IPA performs better than the other approach, AHP-based approach as the first null hypothesis was rejected on a 0.034 significance value. The difference between the actual time-consumption of the two approaches is mostly based on the required number of decision-makings needed to be made by each test subject using these approaches. In this experiment, in order to prioritize 15 functional requirements as well as 5 non-functional requirements, each test subject needed to make 115 decisions using AHP-based approach whereas 75 decisions using IPA. The reason for the difference between IPA and AHP-based approach with respect to the required number of decision-makings is that IPA only needs one decision matrix in which both functional and non-functional requirements are included while

using AHP-based approach, two decision matrixes need to be constructed independently: one for functional requirements and the other one for non-functional requirements. Thus, this factor contributed the test subjects to reach the prioritized lists of functional and non-functional requirements produced by IPA faster than the prioritized lists produced by AHP-based approach. Furthermore, the different range of the values used for indicating a preference applied by the two approaches might have led to this variation in actual time-consumption since IPA uses a five-point scale (see Table 4.3) whereas AHP-based approach exploits a nine-point scale (see Table 6.2). It is obvious that selecting between five options requires less time than choosing among nine options.

A better performance of IPA with respect to AHP-based approach is also discovered when analyzing the results obtained from the dependent variable, ease of use. Initially, by measuring the ease of use using the post-test 1A, it can be concluded that the second null hypothesis was rejected with a p-value of 0.002. To confirm the findings regarding the ease of use of the IPA and AHP-based approach, the researcher also conducted the post-test 2A for measuring the ease of use in another way. It is worth pointing out that the second null hypothesis was rejected again and thus it can be drawn the conclusion that the IPA approach is easier to use than AHP-based approach. Two factors may bias the subjects' judgments on the ease of use of the two approaches: first, the number of decision-makings, and second, specifying the preference value in a short range. Both these factors could possibly put the IPA's ease of use in a higher level than AHP-based approach.

The third null hypothesis which refers to the statement that the accuracy is equal for IPA and AHP-based approach was also rejected. As can be observed in Table 6.20, regarding both the expected and perceived accuracy, the differences are statistically

significant since the p-values are 0.033 and 0.025, respectively. This indicates that the major number of the subjects selected IPA (see Table 6.10 and Table 6.11) as the most accurate approach compared to the AHP-based approach. This may be due to the reason that the triangular fuzzy numbers, which have been used in the underlying of IPA, could increase the accuracy of the results produced by this approach since the other approach, i.e. AHP-based approach, has not used triangular fuzzy numbers.

Regarding controlled Experiment 2, Table 6.21 summarizes the results achieved by the means of conducting the second experiment. In general, these results demonstrate the superiority of IPA over HAM-based approach when those approaches have been applied by the same set of subjects on the same set of functional and non-functional requirements.

Table 6.21: Overview of hypotheses testing of Experiment 2 for comparing IPA and HAM-based approach

Hypothesis	Dependent variable	Statistical test	p-value	Result	Direction
$\mathbf{H_{0}}_{\text{time}}$	Actual time-consumption	T-test	0.016	rejected	IPA
$\boldsymbol{H_{0}}_{easeofuse}$	Ease of use	Mann-whitney Chi-square	0.010 0.022	rejected	IPA
$\mathbf{H_{0}}_{\text{accuracy}}$	Expected accuracy Perceived accuracy	Mann-whitney Chi-square	0.048 0.007	rejected	IPA

In particular, the investigation of the first null hypothesis indicated that there is a significant difference between IPA and HAM-based approach in terms of actual time-consumption since the p-value is 0.016, and thereby IPA is the faster approach. The difference between the actual time-consumption of the two approaches is mostly based on the number of decision makings requested to the subject by these approaches, where it is 85 for HAM-based approach, and 75 for IPA. Furthermore, the different range of the values used for indicating a preference applied by the two approaches might have led to this variation in actual time-consumption.

Furthermore, the superiority of IPA over HAM-based approach can be observed with respect to the other two dependent variables, i.e. ease of use and accuracy of results since the second and third null hypotheses were rejected (see Table 6.21). For the first experiment, the main factors which might lead IPA to show a better performance comparing to AHP-based approach in terms of ease of use and accuracy of results, are discussed earlier. Those justifications are also valid for discussing the differences between IPA and HAM-based approach in terms of ease of use and accuracy of results.

6.2 Summary

This chapter provided a detailed description of the two controlled experiments that have been carried out in order to evaluate and compare the proposed prioritization approach, i.e. IPA, with the most familiar alternatives, called AHP-based approach, and HAM-based approach.

In the first experiment, IPA and AHP-based approach were analyzed and compared together in order to find out which one is an appropriate approach in terms of the actual time-consumption, the ease of use of an approach perceived by the subjects, and the accuracy of each approach's results. As the statistical analysis of the results obtained from the first experiment indicated the superiority of IPA over AHP-based approach in terms of actual time-consumption, ease of use, and accuracy of results, the researcher conducted the second experiment with the aim of comparing the IPA with the other comparable approach called HAM-based approach. To compare the IPA with HAM-based approach, once again the researcher concentrated on measuring the three properties, actual time-consumption, ease of use, and accuracy of results. The statistical analysis of the second experiment's results showed the better performance of IPA compared to HAM-based approach. Both of the experiments have been carried out with

20 experienced subjects on a set of 20 real requirements consisting of 15 functional requirements and 5 non-functional requirements.

CHAPTER 7: CONCLUSION

This chapter describes the main conclusions of this research by (i) presenting the key contributions of the research; (ii) explaining the practical and theoretical implications of this study; (iii) describing the achievements of the research objectives; (iv) discussing the limitation of the study; and (v) outlining some suggestions for future research.

7.1 Contributions

The software engineering community has been criticizing for lacking an approach which enables practitioners to integrate the prioritization of functional and non-functional requirements (Berander & Andrews, 2005; Pergher & Rossi, 2013; Pitangueira, Maciel, de Oliveira Barros, & Andrade, 2013; Thakurta, 2013). The approach introduced in this research, i.e. IPA, is a useful first step towards filling this gap. The IPA allows the practitioners to prioritize both functional and non-functional requirements simultaneously in an integrated manner by establishing their relationships, ultimately producing the prioritized lists of functional and non-functional requirements separately. The key contributions of the IPA over existing approaches are:

- IPA provides a requirements prioritization approach which considers both functional requirements and non-functional requirements simultaneously during the prioritization stage, using only one decision matrix.
- IPA establishes the relationship between functional and non-functional requirements to perform the prioritization task. This means that by using IPA, non-functional requirements are prioritized based on their importance degree for achieving functional requirements. Furthermore, functional requirements are prioritized in relation to non-functional requirements.

Moreover, the effectiveness of the proposed approach was empirically evaluated through conducting two controlled experiments aimed at comparing the IPA with the other similar state-of-the-art approaches, called AHP-based approach, and HAM-based approach. Three main properties were measured during the experiments: the actual time-consumption, the ease of use of an approach perceived by the subjects, and the accuracy of each approach's results. The experiments were conducted with 20 experienced subjects on a set of 20 real requirements of ATM, CDM, and CQM which consist of 15 functional requirements and 5 non-functional requirements. The main conclusion that can be drawn from the results of the experiments is that the IPA is superior to the AHP-based approach as well as HAM-based approach with respect to the actual time-consumption, ease of use, accuracy of results. In particular, the better performance of IPA compared to AHP-based approach and HAM-based approach with respect to the actual time-consumption is predominantly dependent on the required number of decision makings needed to be made by a decision maker. In fact, by using only one decision matrix, IPA produces a prioritized list of functional requirements as well as an ordered list of non-functional requirements, while AHP-based approach and HAM-based approach need two decision matrixes to perform the prioritization tasks. Another interesting point that indicates the superiority of IPA over the other two approaches is the accuracy of results produced by IPA. In other words, IPA does not sacrifice the accuracy of results towards decreasing the required time for performing the prioritization tasks. This implies that IPA is able to perform the prioritization of functional and non-functional requirements in a faster way compared to AHP-based approach and HAM-based approach, while at the same time, it is capable of generating reliable results. The application of triangular fuzzy number in the underlying steps of IPA plays a critical role for producing reliable results (prioritized lists of functional and non-functional requirements). The superiority of IPA over AHP-based approach and

HAM-based approach was also distinguished with respect to ease of use. Two aspects that may cause IPA to be identified as an easier approach to use are the number of decision makings and specifying the preference value in a short range. Both these factors could possibly put the IPA's ease of use in a higher level than the other two approaches.

7.2 Implications

This research offers some significant implications to practitioners and researchers. To requirements engineers, this study provides empirical evidence that adopting a prioritization approach which considers the mutual impact of functional and non-functional requirements during the prioritization process, as IPA does, would lead to beneficial results. The results of applying such an approach for a given software project, which are a prioritized list of functional requirements as well as a prioritized list of non-functional requirements, could assist software developers to concentrate on the most important functional requirements as the key component of the implementation phase, early in the life cycle rather than later when modifications are often difficult and impractical to accomplish. It would also help software architects to consider the most significant non-functional requirements as the main driver to design the system's software architecture and also simplify the selection of suitable guidelines for achieving the desired non-functional requirements. Requirements engineers might consider IPA as an appropriate prioritization approach in projects in which (i) the prioritization of both functional and non-functional requirements is required; (ii) the number of requirements is limited; (iii) the prioritization process needs to be fast and simple while preserving accuracy of results. The analysis of the specific results extracted from the conducted experiments would facilitate decision makers with valuable descriptive and statistical

information which could be useful to assist them to choose the most suitable prioritization approach for a given prioritization problem in an organization.

The findings of this study could be also used as a guideline by interested researchers for identifying trends before initiating a new prioritization approach in the future or evaluating existing ones. In addition, this study provides researchers with few lines for future research. As discussed earlier, it is challenging to claim that the results of this empirical study could be generalized to industrial settings due to the usage of students and researchers as test subjects. To have rigorous evidence of confirming or contradicting this claim, it would be of interest for researchers to investigate the replication of the study in industrial environments by participating professionals as subjects. It would be also worthwhile for researchers to conduct further empirical studies on a larger number of requirements to figure out how similar would be the results with the findings of this study.

7.3 Achievement of Research Objectives

As discussed in Chapter 1 (Section 1.5), three research objectives have been defined throughout this research. This section aims to provide the evidence that all defined research objectives have been tackled and achieved in this study.

To achieve the research objective 1, that is *to identify the current requirements prioritization approaches as well as several empirical evaluations of these approaches*, two research questions were formulated, i.e. RQ1 and RQ2 (see Section 1.6).

- **RQ1**: What are the current approaches used for requirements prioritization?

To investigate the above research question, an extensive literature review has been conducted on the various existing requirements prioritization approaches. Accordingly, the most important requirements prioritization approaches were identified and explained thoroughly in Chapter 2 (see Section 2.2). In fact, conducting an extensive literature review, on the various requirements prioritization approaches, contributed to this research towards finding out the research gap of this study. In particular, by comprehensively reviewing of the current requirements prioritization approaches, it was observed that addressing both functional and non-functional requirements within a single prioritization approach has received less attention in existing prioritization approaches.

- **RQ2**: What are the descriptions and limitations of current requirements prioritization approaches?

In order to find out the answer of this research question, the identified requirements prioritization approaches were analyzed and consequently their descriptions and limitations were captured and reported in Chapter 2 (see Section 2.2.5). This was useful towards discovering the strength and weakness of each prioritization approach.

Moreover, a literature review was conducted on the most important empirical evaluations of the existing requirements prioritization approaches (see Section 2.3). This contributed to this research by providing some insights. Initially, it highlighted the strengths and weaknesses of the existing requirements prioritization approaches when applied in real cases. In addition, it provided guidelines on how to evaluate and compare a requirements prioritization approach which specifically could be beneficial for evaluating the proposed approach of this research.

To attain the research objective 2, that is *to propose an approach by which integrating the process of prioritizing functional and non-functional requirements could be performed*, a research question was formulated, i.e. RQ3 (see Section 1.6).

- **RQ3**: What procedure does a software engineer should follow to integrate the process of prioritizing functional and non-functional requirements?

To investigate the above research question, a nine-step approach, called IPA, was proposed in this research with the aim of integrating the prioritization of functional and non-functional requirements. The detailed description of IPA was provided in Chapter 4.

To attain the research objective 3, that is *to empirically evaluate the effectiveness of the proposed approach, through controlled experiments, in terms of time needed for performing the prioritization task, accuracy of the produced results and ease of use when compared to two similar state-of-the-art approaches*, three research questions including RQ4, RQ5, and RQ6 were formulated (see Section 1.6).

- **RQ4**: How to perform the prioritization task within a reasonable amount of time?
- **RQ5**: How easily the prioritization task can be performed?
- **RQ6**: How accurately the prioritization task can produce the prioritized lists of functional requirements and non-functional requirements?

In order to investigate the above research questions and find out the required results, two controlled experiments were conducted in this research (see Section 6.1) with the aim of evaluating the IPA. Consequently, the results were collected, analyzed and thereby reported in Section 6.1.4.

7.4 Limitation

In the evaluation phase of this research, a rather small number of requirements, i.e. 20 requirements, were used as objects of the controlled experiments. One of the important issues needs to be taken into consideration when conducting experiments is to carry out experiments within a reasonable amount of time (L. Karlsson et al., 2007). Considering this factor could prevent choosing a larger number of requirements for the experiments of this study. However, some industrial projects would deal with the larger number of requirements during the prioritization process. Hence, the findings of the experiments could not be generalized to all industrial environments. The results of this study might be valid when a subset of functional and non-functional requirements of a large-scale system would be prioritized. Therefore, evaluation of the IPA in situations where a larger number of requirements would be prioritized could be considered as a potential suggestion for further empirical investigation of this approach.

7.5 Future work

As a future line of this research, (i) it would be worthwhile to investigate the replication of the controlled experiments on a larger number of requirements with participating professionals to get a sounder basis for our findings.

(ii) The AHP-based approach which was exploited in the first experiment relies on the idea that there is no relationship between functional and non-functional requirements. In other words, AHP-based approach prioritizes functional and non-functional requirements separately. As a future work, it would be interesting to modify this approach in a way that it gets advantage of using hierarchy for establishing the relationships. For example, in the case of having m non-functional requirements and n functional requirements, the first step could be to construct a m×m pairwise comparison

matrix and ask the decision maker to identify his/her judgments for each pair of non-functional requirements. Then, for each non-functional requirement, a n×n pairwise comparison matrix should be constructed for which each pair of functional requirements needs to be assessed with respect to a particular non-functional requirement. In this case, the total number of pairwise comparisons matrixes is m+1. Providing such improvement may result in enhancing the accuracy of results, but at the same time, it could lack in scalability issue. However, one solution to overcome the scalability issue is to exploit Incomplete AHP (IAHP) (Harker, 1987) with the aim of minimizing the pairwise comparisons.

(iii) The prioritization approach, which has been proposed and evaluated in this research, does not focus on the requirements dependencies. Hence, it would be of interest to propose a modified version of this approach in a way that dependencies among requirements would make certain to receive more attention.

REFRENCES

Aasem, Muhammad, Ramzan, Muhammad, & Jaffar, Arfan. (2010). Analysis and optimization of software requirements prioritization techniques. *2010 IEEE International Conference on Information and Emerging Technologies (ICIET)*, 1-6. doi: 10.1109/ICIET.2010.5625687.

Abbott, R. J. (1986). An Integrated Approach to Software Development. New York, USA: John Wiley.

Abirami, S, Shankari, G, Akshaya, S, & Sithika, M. (2015). Conceptual Modeling of Non-Functional Requirements from Natural Language Text *Computational Intelligence in Data Mining-Volume 3* (pp. 1-11): Springer.

Achimugu, Philip, Selamat, Ali, Ibrahim, Roliana, & Mahrin, Mohd Naz'ri. (2014). A systematic literature review of software requirements prioritization research. *Information and Software Technology, 56*(6), 568-585.

Ahl, Viggo. (2005). An experimental comparison of five prioritization methods. *Master's Thesis, School of Engineering, Blekinge Institute of Technology, Ronneby, Sweden*.

Alford, M. W. (1977). A Requirements Engineering Methodology for Real-Time Process Requirements. *IEEE Transactions on Software Engineering*, *3*(1), 60-69.

Anton, A. (1997). *Goal identication and renement in the specication of information systems.* (Ph.D. thesis), Georgia Institute of Technology.

Anwar, Fares, & Razali, Rozilawati. (2014). Requirements Elicitation Techniques Selection Survey. *New Trends in Software Methodologies, Tools and Techniques: Proceedings of the Thirteenth SoMeT_14*, 280-294. doi: 10.3233/978-1-61499-434-3-280.

Aurum, Aybüke, & Wohlin, Claes. (2003). The fundamental nature of requirements engineering activities as a decision-making process. *Information and Software Technology, 45*(14), 945-954.

Aurum, Aybüke, & Wohlin, Claes. (2005). Requirements engineering: Setting the context (pp. 1-15): Springer Berlin Heidelberg.

Avesani, Paolo, Bazzanella, Cinzia, Perini, Anna, & Susi, Angelo. (2004). Supporting the Requirements Prioritization Process. A Machine Learning approach. *SEKE*, 306-311.

Avesani, Paolo, Bazzanella, Cinzia, Perini, Anna, & Susi, Angelo. (2005). Facing scalability issues in requirements prioritization with machine learning techniques. *13th IEEE International Conference on Requirements Engineering*, 297-305.

Azar, J, Smith, RK, & Cordes, D. (2007). Value-Oriented Prioritization: A Framework for Providing Visibility and Decision Support in the Requirements Engineering

Process. *Department of Computer Science, University of Alabama, Technical report*.

Azar, Jim, Smith, Randy K, & Cordes, David. (2007). Value-oriented requirements prioritization in a small development organization. *IEEE Software, 24*(1), 32-37.

Babar, Muhammad Imran, Ghazali, Masitah, & Jawawi, Dayang NA. (2014). Systematic reviews in requirements engineering: A systematic review. *8th IEEE Malaysian Software Engineering Conference (MySEC)*, 43-48.

Babar, Muhammad Imran, Ramzan, Muhammad, & Ghayyur, Shabaz AK. (2011). Challenges and future trends in software requirements prioritization. *2011 IEEE International Conference on Computer Networks and Information Technology (ICCNIT)*, 319-324.

Babbie, Earl. (2013). *The basics of social research* (2 ed.). Belmont, CA:Wadsworth: Cengage Learning.

Barney, Sebastian, Petersen, Kai, Svahnberg, Mikael, Aurum, Aybüke, & Barney, Hamish. (2012). Software quality trade-offs: A systematic map. *Information and Software Technology, 54*(7), 651-662.

Baskaran, Saranya. (2014). A Survey on Prioritization Methodologies to Prioritize Non Functional Requirements. *International Journal of Computer Science and Business Informatics, 12*(1), 32-44.

Beck, Kent. (2000). *Extreme programming explained: embrace change*: Addison-Wesley Professional; US ed edition.

Berander, Patrik. (2004a). Prioritization of Stakeholder Needs in Software Engineering. *Understanding and Evaluation. Licenciate Thesis, Blekinge Institute of Technology, Sweden, Licentiate Series*(2004), 12.

Berander, Patrik. (2004b). Using students as subjects in requirements prioritization. *IEEE International Symposium on Empirical Software Engineering (ISESE'04)*, 167-176. doi: 10.1109/ISESE.2004.1334904.

Berander, Patrik, & Andrews, Anneliese. (2005). Requirements prioritization *Engineering and managing software requirements* (pp. 69-94): Springer.

Berander, Patrik, & Wohlin, Claes. (2004). *Differences in views between development roles in software process improvement-a quantitative comparison.* Paper presented at the Proceedings of the 8th International Conference on Empirical Assessment in Software Engineering (EASE 2004).

Berntsson Svensson, Richard, Olsson, Thomas, & Regnell, Björn. (2013). An investigation of how quality requirements are specified in industrial practice. *Information and Software Technology, 55*(7), 1224-1236.

Bradner, Scott. (1997). Key words for use in RFCs to Indicate Requirement Levels.

Brooks Jr, Frederick P. (1995). *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition, 2/E*: Pearson Education India.

Capilla, Rafael, Babar, Muhammad Ali, & Pastor, Oscar. (2012). Quality requirements engineering for systems and software architecting: methods, approaches, and tools. *Requirements Engineering, 17*(4), 255-258.

Carlshamre, Pär. (2001). A usability perspective on requirements engineering: from methodology to product development. *Ph D Dissertation, Department of Computer and Information Science, Linkoping University, Sweden*.

Carlshamre, Pär, Sandahl, Kristian, Lindvall, Mikael, Regnell, Björn, & Natt och Dag, Johan. (2001). *An industrial survey of requirements interdependencies in software product release planning.* Paper presented at the Fifth IEEE International Symposium on Requirements Engineering, 2001.

Chung, Lawrence, & do Prado Leite, Julio Cesar Sampaio. (2009). On non-functional requirements in software engineering *Conceptual modeling: Foundations and applications* (pp. 363-379): Springer.

Cohen, Lou, & Cohen, Lou. (1995). *Quality function deployment: how to make QFD work for you*: Addison-Wesley Reading, MA.

Cysneiros, Luiz Marcio, & Sampaio do Prado Leite, Julio Cesar. (2004). Nonfunctional requirements: From elicitation to conceptual models. *IEEE Transactions on Software Engineering, 30*(5), 328-350.

Dabbagh, Mohammad, & Lee, Sai Peck. (2013). *A consistent approach for prioritizing system quality attributes.* Paper presented at the 2013 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD).

Dabbagh, Mohammad, Lee, Sai Peck, & Parizi, Reza Meimandi. (2014). *Application of Hybrid Assessment Method for Priority Assessment of Functional and Non-Functional Requirements.* Paper presented at the 2014 International Conference on Information Science and Applications (ICISA).

Danesh, Amir Seyed, & Ahmad, Rodina. (2009). *Study of prioritization techniques using students as subjects.* Paper presented at the International Conference on Information Management and Engineering, 2009. ICIME'09. .

Daneva, Maya, Damian, Daniela, Marchetto, Alessandro, & Pastor, Oscar. (2014). Empirical research methodologies and studies in Requirements Engineering: How far did we come? *Journal of Systems and Software, 95*, 1-9.

Davis, A.M. (1993). *Software Requirements: Objects, Functions, and States*. Upper Saddle River, NJ: Prentice Hall.

Davis, Alan M. (1993). *Software requirements: objects, functions, and states*: PTR Prentice Hall.

Doerr, J., Hartkopf, S., Kerkow, D., Landmann, D., & Amthor, P. (2007). Built-in user satisfaction - Feature appraisal and prioritization with AMUSE. *15th Ieee International Requirements Engineering Conference, Proceedings*, 101-110. doi: Doi 10.1109/Re.2007.62

Doerr, Joerg, Kerkow, Daniel, Koenig, Tom, Olsson, Thomas, & Suzuki, Takeshi. (2005). *Non-functional requirements in industry-three case studies adopting an experience-based NFR method.* Paper presented at the 13th IEEE International Conference on Requirements Engineering. .

Duan, Chuan, Laurent, Paula, Cleland-Huang, Jane, & Kwiatkowski, Charles. (2009). Towards automated requirements prioritization and triage. *Requirements engineering, 14*(2), 73-89.

Egyed, Alexander. (2003). A scenario-driven approach to trace dependency analysis. *IEEE Transactions on Software Engineering, 29*(2), 116-132.

Fellir, Fadoua, Nafil, Khalid, & Touahni, Rajaa. (2014). *System requirements prioritization based on AHP*. Paper presented at the 2014 Third IEEE International Colloquium in Information Science and Technology (CIST).

Firesmith, Donald. (2004). Prioritizing Requirements. *Journal of Object Technology, 3*(8), 35-48.

Glinz, Martin. (2007). *On non-functional requirements.* Paper presented at the 15th IEEE International Requirements Engineering Conference, 2007. RE'07. .

Greer, Des, & Ruhe, Günther. (2004). Software release planning: an evolutionary and iterative approach. *Information and Software Technology, 46*(4), 243-253.

Gunda, Sai Ganesh. (2008). Requirements engineering: elicitation techniques. *Trollhattan: University West, Department of Technology, Mathematics and Computer Science*.

Gurp, J Van, Bosch, J, & Svahnberg, M. (2001). On the notion of variability in software product lines. *Proceedings of Working IEEE/IFIP Conference on Software Architecture*, 45-54.

Harker, Patrick T. (1987). Incomplete pairwise comparisons in the analytic hierarchy process. *Mathematical Modelling, 9*(11), 837-848.

Hatton, Sarah. (2007). Early prioritisation of goals *Advances in conceptual modeling–Foundations and applications* (pp. 235-244): Springer.

Hatton, Sarah. (2008). *Choosing the right prioritisation method.* Paper presented at the 19th Australian Conference on Software Engineering, 2008. ASWEC 2008. .

Henry, Joel, & Henry, Sallie. (1993). *Quantitative assessment of the software maintenance process and requirements volatility.* Paper presented at the Proceedings of the 1993 ACM conference on Computer science.

Hofmann, H. F., & Lehner, F. (2001). Requirements engineering as a success factor in software projects. *IEEE Software, 18*(4), 58-66. doi: Doi 10.1109/Ms.2001.936219

Hopcroft, John E. (1983). *Data structures and algorithms*. Boston, MA, USA: Addison-Wesley.

IEEE, 830 Std. (1998). IEEE Recommended Practice for Software Requirements Specifications.

IEEE. (1990). Standard Glossary of Software Engineering Terminology: IEEE Std.

Iqbal, A, Khan, FM, & Khan, SA. (2009). A critical analysis of techniques for requirement prioritization and open research issues. *International Journal of Reviews in Computing, 1*, 1-11.

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The Unied Software Development Process*: Addison-Wesley Longman Publishing Co., Inc.,.

Jahanshahloo, Gholam Reza, Lotfi, F Hosseinzadeh, & Izadikhah, Mohammad. (2006). Extension of the TOPSIS method for decision-making problems with fuzzy data. *Applied Mathematics and Computation, 181*(2), 1544-1551.

Jobber, David, & Ellis-Chadwick, Fiona. (2012). *Principles and practice of marketing*. Boca Raton, FL: McGraw-Hill Higher Education.

Karlsson, Joachim. (1996). *Software requirements prioritizing.* Paper presented at the Proceedings of the Second International Conference on Requirements Engineering, 1996.

Karlsson, Joachim, & Ryan, Kevin. (1997). A cost-value approach for prioritizing requirements. *IEEE Software, 14*(5), 67-74.

Karlsson, Joachim, Wohlin, Claes, & Regnell, Björn. (1998). An evaluation of methods for prioritizing software requirements. *Information and Software Technology, 39*(14), 939-947.

Karlsson, Lena, Höst, Martin, & Regnell, Björn. (2006). *Evaluating the practical use of different measurement scales in requirements prioritisation.* Paper presented at the Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering.

Karlsson, Lena, Thelin, Thomas, Regnell, Björn, Berander, Patrik, & Wohlin, Claes. (2007). Pair-wise comparisons versus planning game partitioning—experiments on requirements prioritisation techniques. *Empirical Software Engineering, 12*(1), 3-33.

Khan, Kashif Ahmed. (2006). *A Systematic Review of Software Requirements Prioritization.* (Master Thesis), Blekinge Institute of Technology, Sweden.

Kotonya, G, & Sommerville, I. (1998). *Requirements Engineering - Processes and Techniques*: John Wiley & Sons, New York, NY. USA.

Kukreja, Nupul. (2013). *Decision theoretic requirements prioritization A two-step approach for sliding towards value realization.* Paper presented at the 2013 35th International Conference on Software Engineering (ICSE).

Kuusela, Juha, & Savolainen, Juha. (2000). *Requirements engineering for product families.* Paper presented at the Proceedings of the 22nd international conference on Software engineering.

Laplante, Phillip A. (2013). *Requirements engineering for software and systems* (1 ed.). Redmond, WA: CRC Press.

Lauesen, Soren. (2002). *Software requirements: styles and techniques*. Essex Pearson Education.

Lawrence, Brian, Wiegers, Karl, & Ebert, Christof. (2001). The top risk of requirements engineering. *IEEE Software, 18*(6), 62-63.

Leffingwell, Dean, & Widrig, Don. (2000). *Managing software requirements: a unified approach*: Addison-Wesley Professional.

Lehtola, Laura, & Kauppinen, Marjo. (2004). Empirical evaluation of two requirements prioritization methods in product development projects *Software Process Improvement* (pp. 161-170): Springer.

Lehtola, Laura, Kauppinen, Marjo, & Kujala, Sari. (2004). Requirements prioritization challenges in practice *Product focused software process improvement* (pp. 497-508): Springer.

Lethbridge, T., & Laganiere, R. (2001). *Object oriented software engineering: practical software development using UML and Java*. England: Mc-Graw Hill Education.

Likert, Rensis. (1932). A technique for the measurement of attitudes. *Archives of psychology (140)*.

Liou, Tian-Shy, & Wang, Mao-Jiun J. (1992). Ranking fuzzy numbers with integral value. *Fuzzy sets and systems, 50*(3), 247-255.

Liu, Xiaoqing Frank, Sun, Yan, Veera, Chandra Sekhar, Kyoya, Yuji, & Noguchi, Kunio. (2006). Priority assessment of software process requirements from multiple perspectives. *Journal of Systems and Software, 79*(11), 1649-1660.

Ma, Qiao. (2009). *The effectiveness of requirements prioritization techniques for a medium to large number of requirements: a systematic literature review.* Auckland University of Technology.

Mahmood, YA, Ahmadi, Alireza, Verma, Ajit Kumar, Srividya, Ajit, & Kumar, Uday. (2013). Fuzzy fault tree analysis: a review of concept and application. *International Journal of System Assurance Engineering and Management, 4*(1), 19-32.

Marascuilo, Leonard A, & Serlin, Ronald C. (1988). *Statistical methods for the social and behavioral sciences*: WH Freeman/Times Books/Henry Holt & Co.

Martens, Nick. (2011). *The impact of non-functional requirements on project success.* Citeseer, Utrecht University, Netherland.

Martinez, Ana Belen Barragans, Pazos Arias, JJ, & Vilas, Ana Fernandez. (2005). *Merging requirements views with incompleteness and inconsistency.* Paper presented at the 2005 Australian Software Engineering Conference. .

McZara, Jason, Sarkani, Shahryar, Holzer, Thomas, & Eveleigh, Timothy. (2014). Software requirements prioritization and selection using linguistic tools and

constraint solvers—a controlled experiment. *Empirical Software Engineering*, 1-41.

Montgomery, Douglas C. (2008). *Design and analysis of experiments*: John Wiley & Sons.

Mylopoulos, John, Chung, Lawrence, & Nixon, Brian. (1992). Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Transactions on Software Engineering, 18*(6), 483-497.

Newkirk, James, & Martin, Robert C. (2001). Extreme programming in practice. *Object-oriented programming, systems, languages, and applications (Addendum)*, 25-26.

Nuseibeh, B, & Easterbrook, S. (2000). Requirements engineering: A roadmap. *Future of Software Engineering*, 35 – 46.

Pacheco, Carla, & Garcia, Ivan. (2009). *Effectiveness of stakeholder identification methods in requirements elicitation: experimental results derived from a methodical review.* Paper presented at the Eighth IEEE/ACIS International Conference on Computer and Information Science, 2009. .

Paetsch, Frauke, Eberlein, Armin, & Maurer, Frank. (2003). *Requirements engineering and agile software development.* Paper presented at the International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises.

Pergher, Massimiliano, & Rossi, Bruno. (2013). *Requirements prioritization in software engineering: A systematic mapping study.* Paper presented at the 2013 IEEE Third International Workshop on Empirical Requirements Engineering (EmpiRE).

Perini, Anna, Ricca, Filippo, & Susi, Angelo. (2009). Tool-supported requirements prioritization: Comparing the AHP and CBRank methods. *Information and Software Technology, 51*(6), 1021-1032.

Perini, Anna, Susi, A, Ricca, Filippo, & Bazzanella, C. (2007). *An empirical study to compare the accuracy of AHP and CBRanking techniques for requirements prioritization.* Paper presented at the Fifth International Workshop on Comparative Evaluation in Requirements Engineering, 2007. CERE'07. .

Perini, Anna, Susi, Angelo, & Avesani, Paolo. (2013). A machine learning approach to software requirements prioritization. *Software Engineering, IEEE Transactions on, 39*(4), 445-461.

Pfleeger, Shari Lawrence. (1995). Experimental design and analysis in software engineering. *Annals of Software Engineering, 1*(1), 219-253.

Pohl, Klaus. (2010). *Requirements engineering: fundamentals, principles, and techniques*: Springer Publishing Company, Incorporated.

Pressman, R. (2010). *Software Engineering: A Practitioner s Approach* (7th edition ed.): McGraw-Hill.

Ramzan, M, Jaffar, M Arfan, Iqbal, M Amjad, Anwar, Sajid, & Shahid, Arshad A. (2009). *Value based fuzzy requirement prioritization and its evaluation framework.* Paper presented at the Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on.

Ramzan, Muhammad, Jaffar, M Arfan, & Shahid, Arshad Ali. (2011). Value based intelligent requirement prioritization (VIRP): expert driven fuzzy logic based prioritization technique. *International Journal Of Innovative Computing, Information And Control, 7*(3).

Regnell, Björn, & Brinkkemper, Sjaak. (2005). Market-driven requirements engineering for software products *Engineering and managing software requirements* (pp. 287-308): Springer.

Regnell, Björn, Höst, Martin, Och Dag, Johan Natt, Beremark, Per, & Hjelm, Thomas. (2001). An industrial case study on distributed prioritisation in market-driven requirements engineering for packaged software. *Requirements Engineering, 6*(1), 51-62.

Rehman, Tousif, Khan, Muhammad Naeem Ahmed, & Riaz, Naveed. (2013). Analysis of Requirement Engineering Processes, Tools/Techniques and Methodologies. *International Journal of Information Technology and Computer Science (IJITCS), 5*(3), 40.

Ribeiro, Rita A, Moreira, Ana M, Van den Broek, Pim, & Pimentel, Afonso. (2011). Hybrid assessment method for software engineering decisions. *Decision Support Systems, 51*(1), 208-219.

Robertson, S., & Robertson, J. (2012). *Mastering the requirements process: Getting requirements right*: Addison-wesley.

Robson, C. (2002). *Real World Research: A Resource for Social Scientists and Practitioner-Researchers*: Wiley.

Royce. (1970). *Managing the Development of Large Software Systems: Concepts and Techniques.* Paper presented at the WESTCON, Los Angeles, California.

Ruhe, Günther, Eberlein, Armin, & Pfahl, Dietmar. (2003). Trade-off analysis for requirements selection. *International Journal of Software Engineering and Knowledge Engineering, 13*(04), 345-366.

Saaty, Thomas L. (1980). *The Analytic Hierarchy Process*. New York: McGraw-Hill.

Sadana, Vishal, & Liu, Xiaoqing Frank. (2007). *Analysis of conflicts among non-functional requirements using integrated analysis of functional and non-functional requirements.* Paper presented at the 31st Annual International Computer Software and Applications Conference, 2007.

Sharif, Naila, Zafar, Kashif, & Zyad, Waqas. (2014). *Optimization of requirement prioritization using Computational Intelligence technique.* Paper presented at the 2014 International Conference on Robotics and Emerging Allied Technologies in Engineering (iCREATE).

Shen, Yujin, Hoerl, AE, & McConnell, Wes. (1992). An incomplete design in the analytic hierarchy process. *Mathematical and computer modelling, 16*(5), 121-129.

Sher, Falak, Jawawi, Dayang NA, Mohamad, Radziah, & Babar, Muhammad Imran. (2014). *Requirements prioritization techniques and different aspects for prioritization a systematic literature review protocol.* Paper presented at the 2014 8th Malaysian Software Engineering Conference (MySEC).

Siegel, S., & Castellan, N.J. (1988). *Nonparametric Statistics for the Behavioral Sciences*: McGraw-Hill.

Sommerville, I. (2004). *Software Engineering*. USA: Pearson Addison Wesley.

Sommerville, I. (2007). *Software Engineering* (7th Edition ed.). USA: Pearson Addison Wesley.

Sommerville, I., & Sawyer, P. (1997). *Requirements Engineering: Good Practice Guide.*: Chichester, England: John Wiley & Sons Ltd.

Stake, Robert E. (1995). *The Art of Case Study Research*. Washington DC: SAGE Publications.

Svahnberg, Mikael, Aurum, Aybüke, & Wohlin, Claes. (2008). *Using students as subjects-an empirical evaluation.* Paper presented at the Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement.

Svensson, Richard Berntsson, Gorschek, Tony, Regnell, Björn, Torkar, Richard, Shahrokni, Ali, & Feldt, Robert. (2012). Quality Requirements in Industrial Practice-An Extended Interview Study at Eleven Companies. *IEEE Transactions on Software Engineering, 38*(4), 923-935.

Svensson, Richard Berntsson, Gorschek, Tony, Regnell, Björn, Torkar, Richard, Shahrokni, Ali, Feldt, Robert, & Aurum, Aybüke. (2011). *Prioritization of quality requirements: State of practice in eleven companies.* Paper presented at the 2011 19th IEEE International Conference on Requirements Engineering

Teixeira, Leonor, Saavedra, Vasco, Ferreira, Carlos, Simões, João, & Santos, Beatriz Sousa. (2014). Requirements Engineering Using Mockups and Prototyping Tools: Developing a Healthcare Web-Application *Human Interface and the Management of Information. Information and Knowledge Design and Evaluation* (pp. 652-663): Springer.

Thakurta, Rahul. (2013). A framework for prioritization of quality requirements for inclusion in a software project. *Software Quality Journal, 21*(4), 573-597.

Tichy, Walter F. (2000). Hints for reviewing empirical work in software engineering. *Empirical Software Engineering, 5*(4), 309-312.

Tonella, Paolo, Susi, Angelo, & Palma, Francis. (2013). Interactive requirements prioritization using a genetic algorithm. *Information and Software Technology, 55*(1), 173-187.

Tran, Tuyet-Lan, & Sherif, Joseph S. (1995). *Quality Function Deployment (QFD): an effective technique for requirements acquisition and reuse.* Paper presented at the Second IEEE International Software Engineering Standards Symposium, 1995.(ISESS'95)'Experience and Practice'.

Tudor, Dot, & Walter, George A. (2006). *Using an agile approach in a large, traditional organization.* Paper presented at the Agile Conference, 2006.

Van Laarhoven, PJM, & Pedrycz, Witold. (1983). A fuzzy extension of Saaty's priority theory. *Fuzzy sets and Systems, 11*(1), 199-227.

Voola, Persis, & Babu, A Vinaya. (2012). Requirements uncertainty prioritization approach: a novel approach for requirements prioritization. *Software Engineering: An International Journal (SEIJ) Vol, 2*, 37-49.

Voola, Persis, & Babu, A Vinaya. (2013). Comparison of requirements prioritization techniques employing different scales of measurement. *ACM SIGSOFT Software Engineering Notes, 38*(4), 1-10.

Wang, H, Xie, M, & Goh, TN. (1998). A comparative study of the prioritization matrix method and the analytic hierarchy process technique in quality function deployment. *Total Quality Management, 9*(6), 421-430.

Wiegers, K. E. (2003). *Software Requirements*. USA: Microsoft Press.

Wiegers, Karl. (1999). First things first: prioritizing requirements. *Software Development, 7*(9), 48-53.

Wohlin, Claes, Runeson, Per, Höst, Martin, Ohlsson, Magnus C, Regnell, Björn, & Wesslén, Anders. (2012). *Experimentation in software engineering*: Springer.

Yin, Robert K. (2014). *Case study research: Design and methods*. Washington DC: Sage publications.

Yourdon, E. (1997). Death March Projects *Keynote Address, ICSE, 97*.

Zadeh, Lotfi A. (1965). Fuzzy sets. *Information and control, 8*(3), 338-353.

Zelkowitz, Marvin V, & Wallace, Dolores R. (1998). Experimental models for validating technology. *Computer, 31*(5), 23-31.

LIST OF PUBLICATIONS AND PRESENTED PAPERS

Mohammad Dabbagh, Sai Peck Lee, Reza Meimandi Parizi. 2015. "*Functional and Non-Functional Requirements Prioritization: empirical evaluation of IPA, AHP-based, and HAM-based approaches*". Soft Computing, Springer, DOI: 10.1007/s00500-015-1760-z.

Mohammad Dabbagh and Sai Peck Lee. 2015. "*An Approach for Prioritizing NFRs According to Their Relationship with FRs*". Lecture Notes on Software Engineering, Vol. 3, No. 1, February 2015. pp. 1-5. DOI: 10.7763/LNSE.2015.V3.154.

Mohamad Dabbagh and Sai Peck Lee. 2014. "*An Approach for Integrating the Prioritization of Functional and Non- functional Requirements*". The Scientific World Journal. Volume 2014 (2014), Article ID 737626, 13 pages. http://dx.doi.org/10.1155/2014/737626.

Reza Meimandi Parizi, Sai Peck Lee, and Mohammad Dabbagh. 2014. "*Achievements and Challenges in State-of-the-Art Software Traceability Between Test and Code Artifacts*". IEEE Transactions on Reliability. Vol. 63, No. 4, Dec 2014. pp. 913-926.

Tahriri, Farzad, Mohammad Dabbagh, and Nader Ale Ebrahim. "*Supplier assessment and selection using fuzzy analytic hierarchy process in a steel manufacturing company*". Journal of Scientific Research and Reports 3.10 (2014): 1319-1338.

Mohammad Dabbagh, Sai Peck Lee, Reza Meimandi Parizi. 2014. "*Application of Hybrid Assessment Method for Priority Assessment of Functional and Non-Functional Requirements*". 2014 International Conference on Information Science and Applications (ICISA). 6-9 May 2014. Seoul, South Korea. pp. 1-4. IEEE Computer Society. ISBN 978-1- 4799- 4443-9. DOI 10.1109/ICISA.2014.6847365.

Mohammad Dabbagh and Sai Peck Lee. 2013. "*A Consistent Approach for Prioritizing System Quality Attributes*". 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). Honolulu, HI, USA, 1-3 July 2013. pp. 317 - 322.

Mohammad Dabbagh and Ali Mahjur. 2011. "*Filter Scheduling in the Flexible Stream Programming Language*, Vol. 6, Issue. 3, May 2011. pp. 306-313.

Mohamad Dabbagh. "*Introducing a new language for stream applications*". International Journal on Computer Science and Engineering 3.5 (2011): 2119-2126.

Appendices

APPENDIX A: THE SOFTWARE PROTOTYPE

Chapter 4 introduced and described the underlying steps of IPA. However, performing the underlying steps of IPA manually during the software development process is not a straightforward job and might result in achieving incorrect and unreliable prioritized lists of functional and non-functional requirements. Moreover, the large amount of mathematical calculations such as calculating triangular fuzzy number, applying alpha cut approach and so on, that need to be done while using the IPA might be considered as a person-power intensive, time-consuming and error-prone task. In this respect, a software prototype which is capable of automating the underlying steps of IPA would be needed.

The main focus of this section is to present the detailed development of the software prototype, called TIPA (Tool-supported Integrated Periodization Approach), which aims to automate the prioritization of functional and non-functional requirements using IPA. The software prototype, i.e. TIPA, can aggressively tackle the mentioned difficulties of manually using IPA by reducing the effort and time needed to perform the prioritization task. In addition, TIPA is able to improve the reliability of the results produced by the proposed approach.

TIPA needs to be developed to assist researchers and practitioners to computerize the approach that has been proposed in this research, to integrate the prioritization of functional and non-functional requirements simultaneously. In other words, for a given software project, TIPA is able to produce a prioritized list of functional requirements as well as a prioritized list of non-functional requirements automatically in the same process as the IPA does. As a first step to develop the software prototype, the main requirements of TIPA have been elicited and listed below.

- **R1:** The TIPA shall be able to allow requirements engineers to insert/delete/update/view software projects which their functional and non-functional requirements need to be prioritized.
- **R2:** The TIPA shall be able to allow requirements engineers to insert/delete/update/view potential stakeholders associated with a defined software project which their functional and non-functional requirements need to be prioritized.
- **R3:** The TIPA shall be able to allow requirements engineers to insert/delete/update/view functional requirements that need to be prioritized for a defined software project.
- **R4:** The TIPA shall be able to allow requirements engineers to insert/delete/update/view non-functional requirements that need to be prioritized for a defined software project.
- **R5:** The TIPA shall be able to allow requirements engineers to apply pairwise comparisons between all possible pairs of stakeholders using AHP scales.
- **R6:** The TIPA shall be able to calculate and show the final weight of each stakeholder associated with a defined software project using AHP algorithm.
- **R7:** The TIPA shall be able to allow each stakeholder to specify his/her preference value in terms of identifying the importance degree of each non-functional requirement for achieving each functional requirement using IPA nominal scales.
- **R8:** The TIPA shall be able to calculate and show the prioritized list of functional requirements, non-functional requirements, and their respective values related to each stakeholder.

- **R9:** The TIPA shall be able to allow requirements engineers to aggregate different prioritized lists of functional and non-functional requirements calculated by various stakeholders.
- **R10:** The TIPA shall be able to provide requirements engineers with a search engine equipped with various types of reports regarding software projects, stakeholders and their relative weights, functional and non-functional requirements of each project, prioritized lists of functional and non-functional requirements, and etc.

After eliciting the main requirements of TIPA, these requirements can be represented using use case diagram. Figure A.1 illustrates the use case diagram of TIPA.

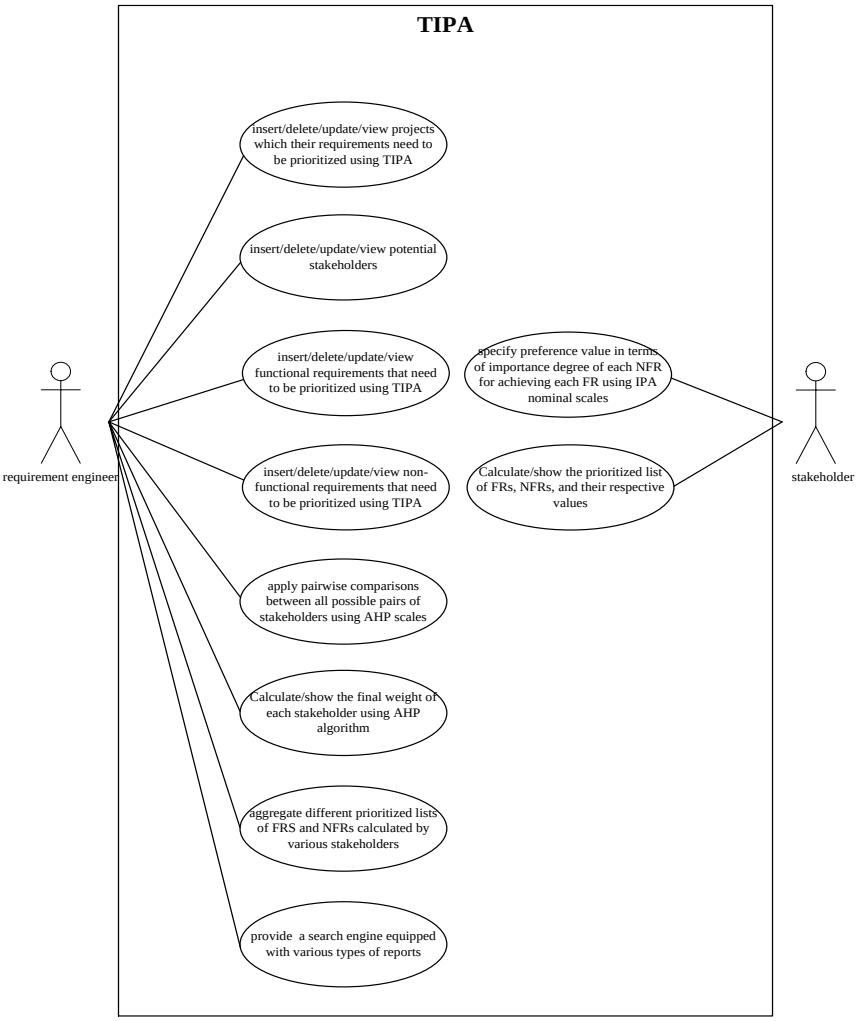


Figure A.1: Use case diagram of the TIPA software prototype

In the implementation phase of developing TIPA, the real software prototype has been implemented to satisfy the specified requirements. The first step of the implementation phase was to select the programming language that would be suitable for implementing the TIPA software prototype. Moreover, the data generated by the software prototype should have stored in a database. So, the second step was to select and design a database to store the TIPA's data. Finally, the third step was to code the required functions to implement the TIPA software prototype and design the user interface. A detailed explanation of these steps is provided below.

1) Selection of Programming Language

An object-oriented programming (OOP) language was selected to implement the TIPA software prototype. In fact, Visual C# programming language (version 5.0 released in August 2012), which is included in Microsoft Visual Studio 2012 and .Net framework 4.0, has been used to implement the TIPA software prototype. The reasons of choosing Visual C# as a programming language are listed below.

- The C# language is a multi-paradigm, modern, general-purpose, high-level, and object-oriented programming language for building applications using Visual Studio and .Net framework.
- The C# language provides many features for software engineering principles such as strong type checking, array bounds checking, detection of attempts to use uninitialized variables, and automatic garbage collection.
- The C# language is suitable for writing applications for both hosted, distributed and embedded systems, ranging from the very large that use sophisticated operating systems, down to the very small having dedicated functions.

- The C# language is suitable for developing software applications that need to be economical with respect to memory and processing power requirements.

2) Database Design

Microsoft SQL Server 2010 was utilized to store and retrieve the data generated by the TIPA. Indeed, Microsoft SQL Server is a relational database management system developed by Microsoft. Microsoft SQL Server offers many features such as:

- The SQL Server allows storing data in a collection of tables with typed columns supporting different data types including *Integer*, *Float*, *Decimal*, *Char*, *Varchar*, *Binary*, and *Text*.
- The SQL Server provides buffer management by which pages are buffered in memory to minimize disk input/output and thereby improves performance.
- The SQL Server allows multiple clients to access the same database concurrently while preserving the data integrity.
- The SQL Server offers querying using stored procedures so that it is capable of executing queries in the server side and not in the client side in order to decrease network traffic and improve performance.

In this regard, a database called TIPA_DBMS, was created using SQL Server 2010 with the aim of storing and retrieving data generated by the TIPA software prototype.

3) Coding and User Interface Design

To start coding the TIPA, three class libraries have been built. The first and lowest level class library, called *dataAccess*, included some classes to perform the transactions on the TIPA_DBMS. These transactions included inserting, updating, deleting, and retrieving data from/to the database. The second and middle level class library, named

business, was built to implement and execute the required operations and internal calculations of TIPA. Lastly, the third and highest level class library, called *presentation*, included some visual user interfaces to interact with end user and provide the tool's functionality to the person who works with the TIPA. In the following, the user interface design is explained.

The user interface of TIPA has a main visual menu consisting of four menu items such as *Basic information*, *AHP pairwise comparison*, *IPA decisions*, and *Search*. The main menu is viewable in Figure A.2. The TIPA's user can insert/update/delete/view the information of projects, stakeholders, functional requirements and non-functional requirements by the means of *Basic information* menu item. By using *AHP pairwise comparison* menu item, user of TIPA is able to perform pairwise comparisons between all possible pairs of stakeholders using AHP scales. Then, TIPA calculates and shows the weights of stakeholders for a given project using AHP algorithm. *IPA decisions* enables user to specify his/her judgments regarding the importance degree of each non-functional requirements for achieving each functional requirement. Afterward, TIPA computes and represents the prioritized list of functional requirements and prioritized list of non-functional requirements based on the internal calculation of IPA. Finally, the TIPA software prototype provides the user with a different range of reports through *Search* menu item.

The *Basic information* menu item comprises four sub items including *Projects*, *Stakeholders*, *FRs*, and *NFRs*. In practice, by clicking *Projects* sub item, a visual windows form is represented, as displayed in the middle of Figure A.2, in which the information of software projects is accessible. In fact, by clicking "Add a new project", the TIPA's user can define a new project in which functional and non-functional

requirements need to be prioritized using IPA. In addition, the TIPA's user may update the information of existing projects or delete current projects by pressing "Change existing project", and "Delete a project", respectively. In the bottom part of the *Projects* windows form, the whole information of the projects is provided in a data grid.

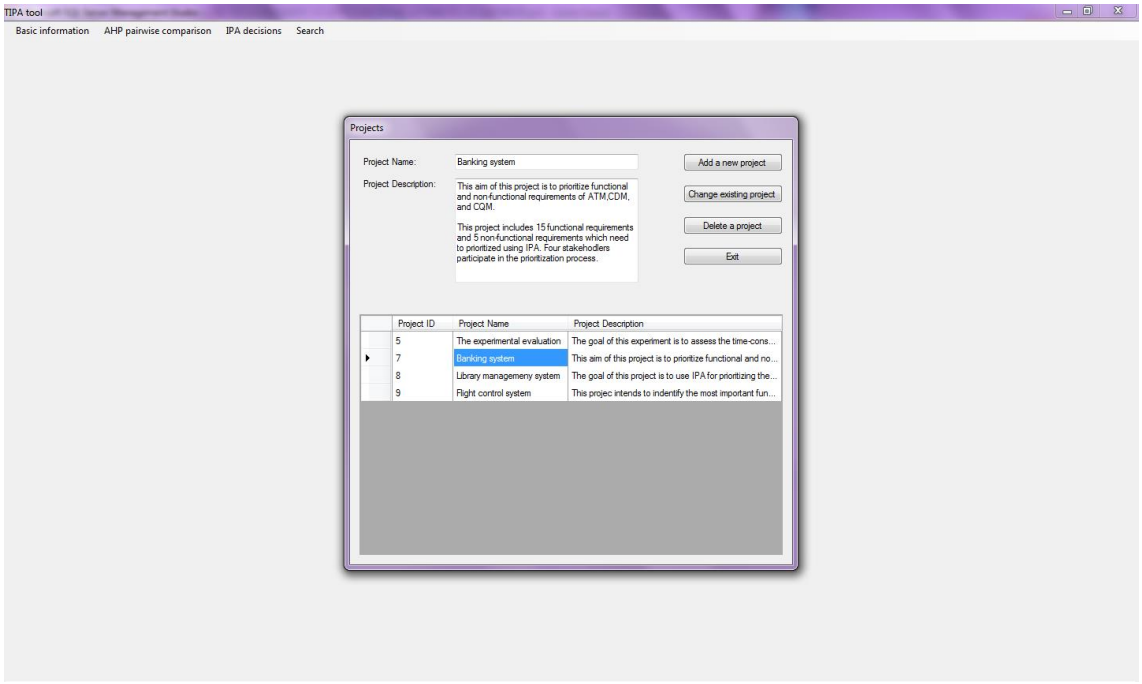


Figure A.2: User interface of the TIPA software prototype along with its *Projects* form

Each defined project may have some associated stakeholders. To define the stakeholders of a given project using TIPA, users need to press *Stakeholders* sub item (under *Basic information* menu item) to view the related visual windows form. This form is indicated in Figure A.3 where the defined projects are retrieved from the TIPA_DBMS and listed automatically in the left upper side of the windows form. Then the TIPA's user is able to define one or more stakeholders associated with the selected project by clicking "Add a new stakeholder" button. Once the user adds a new stakeholder, the information of the new stakeholder is shown in a data grid in the bottom of the *Stakeholders* windows form. Moreover, TIPA facilitates the users with other options such as updating and/or deleting the information of existing stakeholders by providing "Change a stakeholder", and "Delete a stakeholder" buttons.

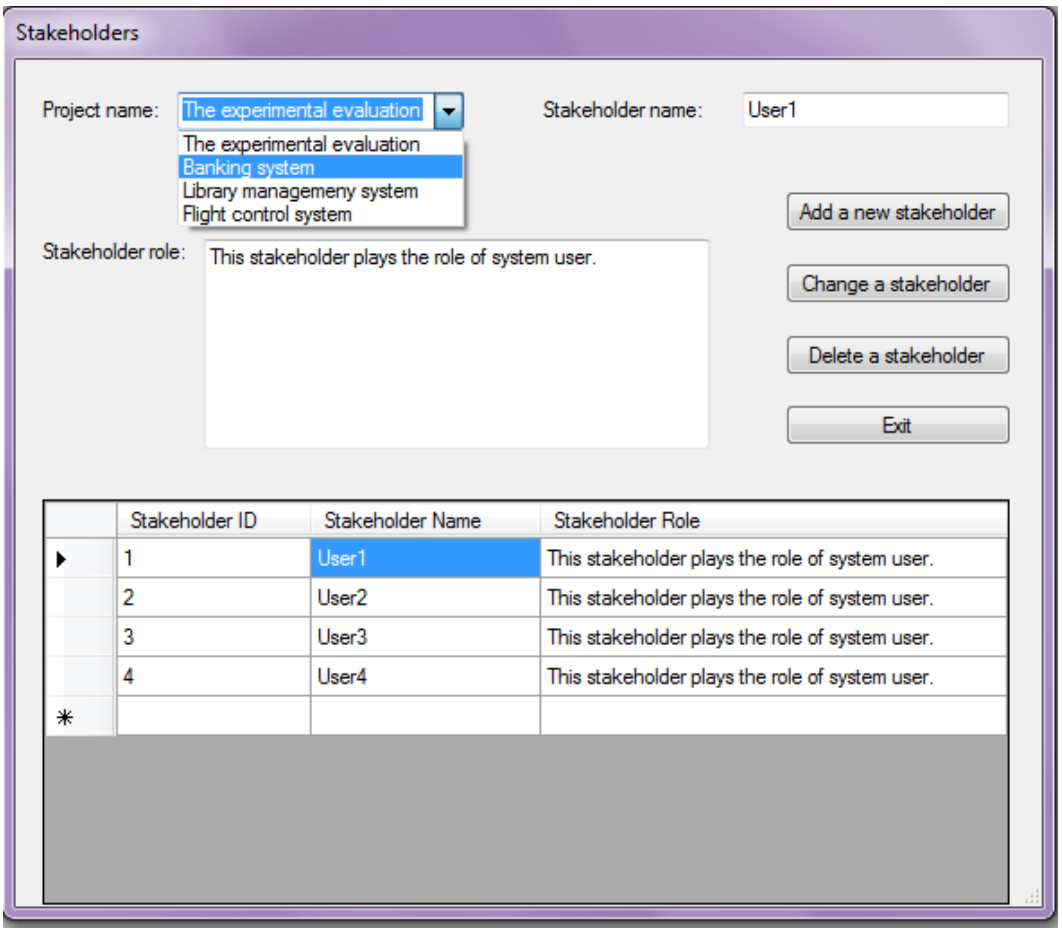


Figure A.3: A snapshot of *Stakeholders* windows form

The TIPA software prototype is also equipped with a windows form for inserting/updating/deleting/viewing functional requirements associated with each defined project. In practice, by clicking *FRs* sub item (under *Basic information* menu item), a windows form, as indicated in Figure A.4, is displayed by which users are able to perform required operations concerning with functional requirements of each selected project.

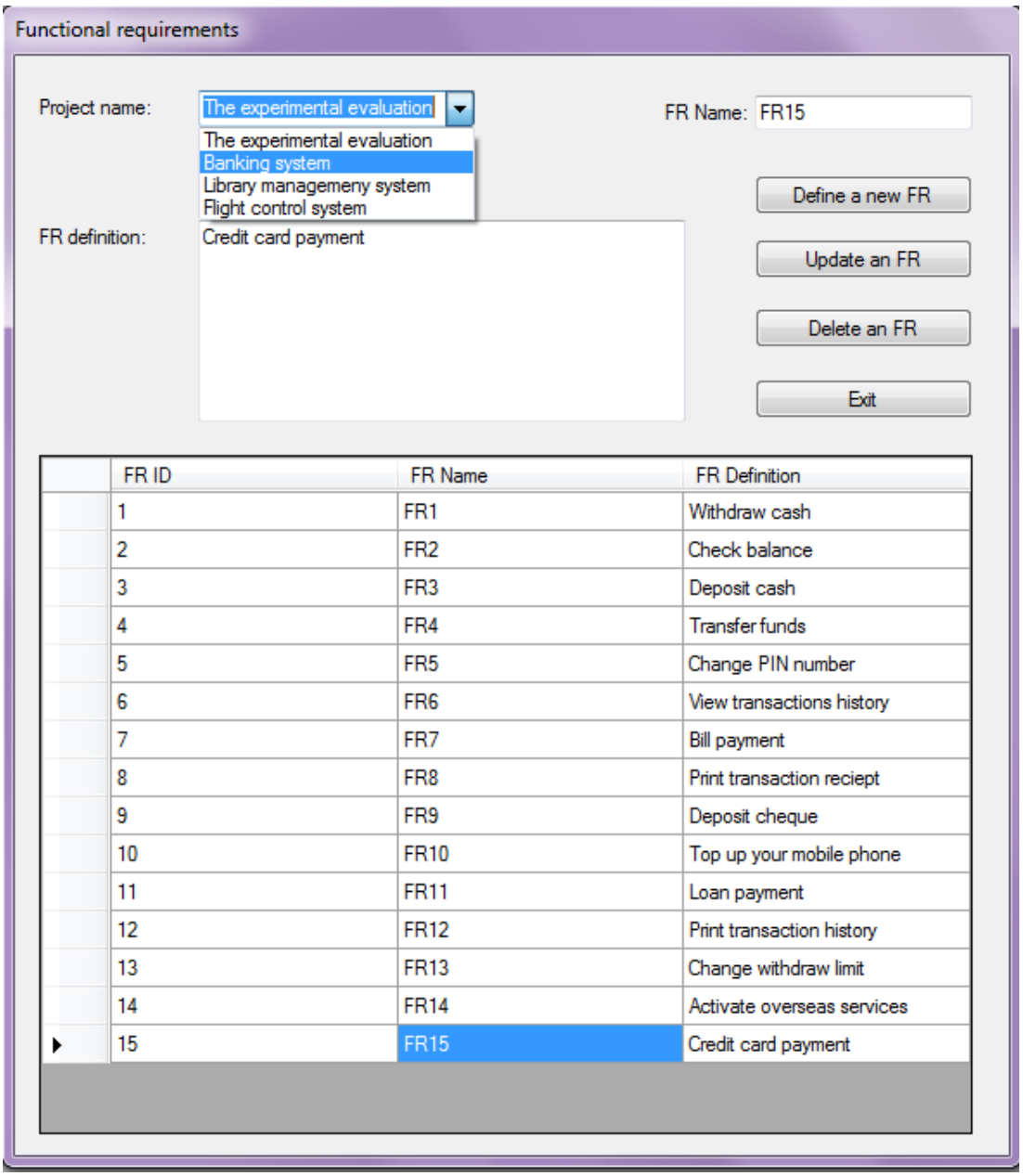


Figure A.4: A graphical view of *FRs* windows form

Similarly, a windows form is designed to define, modify, delete, and view the information of non-functional requirements associated with each project. This form is represented in Figure A.5. Using TIPA, it can be accessible through clicking *NFRs* sub item under *Basic information* menu item.

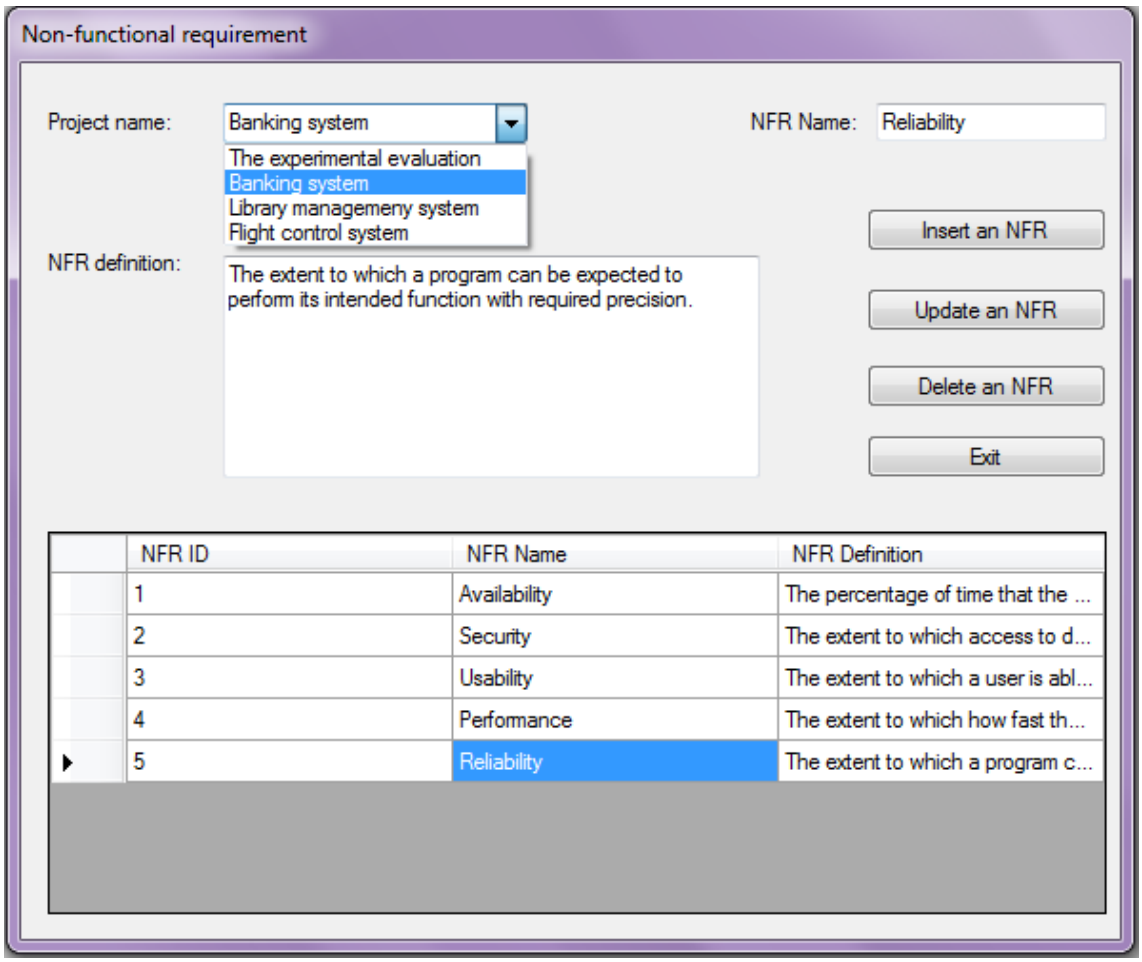


Figure A.5: A snapshot of *NFRs* windows form

The TIPA software prototype provides the ability to perform the pairwise comparisons between all possible pairs of stakeholders to specify the weights of stakeholders for a given project. Indeed, by clicking on the *AHP pairwise comparison* menu item, a visual windows form is represented, as indicated in Figure A.6, where the TIPA's user is able to perform pairwise comparisons between each pair of stakeholders for a selected project. In other words, after selecting the project (see the left upper side of Figure A.6), the TIPA software prototype displays the user an agenda of $n * (n - 1)/2$ pairwise comparisons. The TIPA's user can view the name of each pair of stakeholders, determine her/his preference value by identifying the relative importance of one stakeholder over the other one according to AHP scales, through choosing one of the radio buttons shown in Figure A.6. When the user presses "Submit", the subsequent pair of stakeholders is represented.

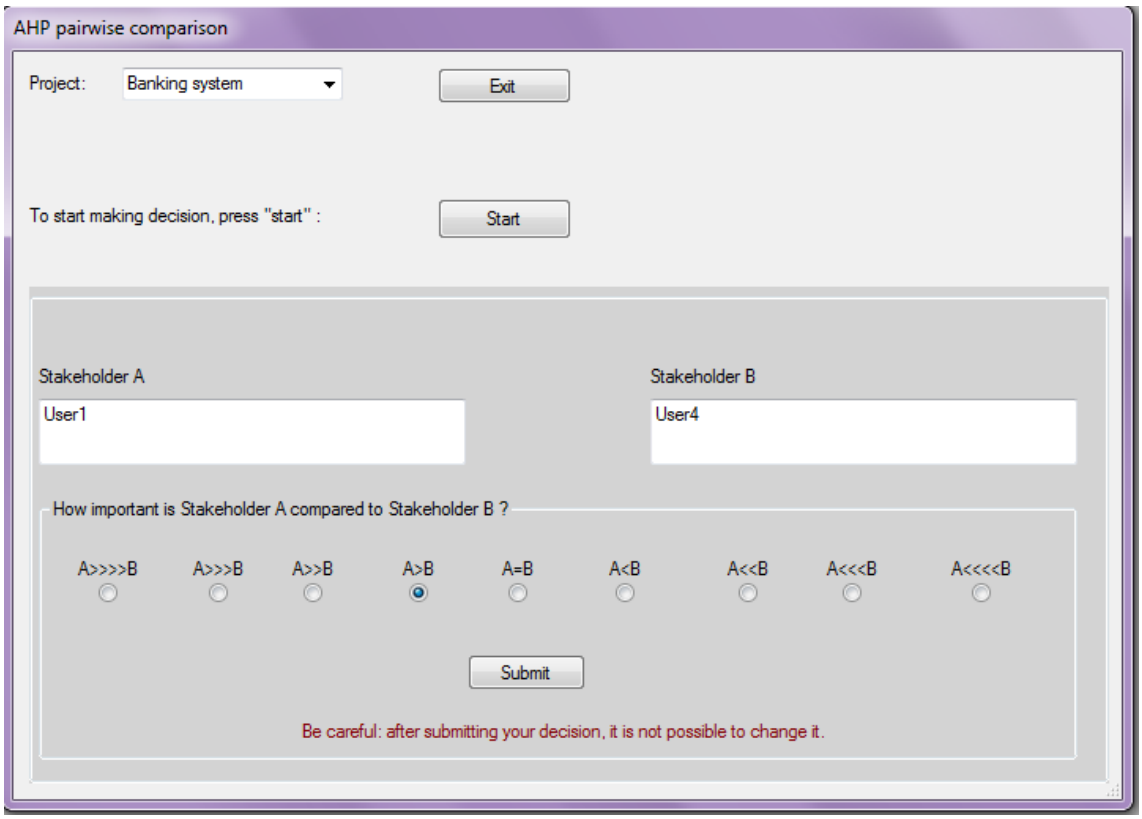


Figure A.6: A picture of pairwise comparisons of two stakeholders, *User1* and *User4*, using *TIPA*

When the pairwise comparisons of all stakeholders have been done completely, TIPA computes and shows automatically the weights of the stakeholders along with their ranking, as represented in Figure A.7, using AHP algorithm.

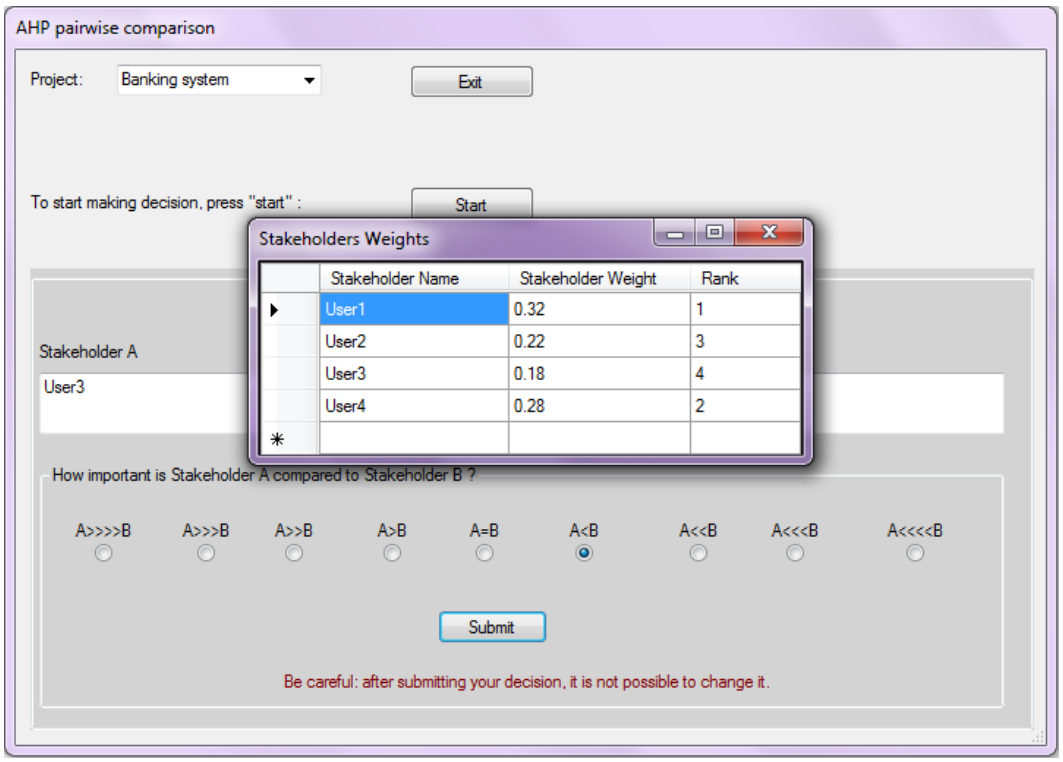


Figure A.7: Calculated weights of stakeholders using *TIPA*

By clicking on the *IPA decisions*, TIPA guides each stakeholder to express his/her judgments between all possible pairs of functional and non-functional requirements in a similar way as the IPA does. Figure A.8 shows a picture of the *IPA decisions* visual user interface. In practice, the stakeholder can view the explanation of functional and non-functional requirements for every possible pair (i.e. FR and NFR) for a selected project. So, the stakeholder can specify her/his preference value by identifying the importance degree of every non-functional requirement for achieving each functional requirement according to IPA nominal scale, through choosing one of the radio buttons indicated in Figure A.8. When the stakeholder presses 'Submit', the subsequent pair of requirements (i.e. FR and NFR) is viewable. After finishing the evaluations of all functional and non-functional requirements, TIPA is capable of calculating and showing the prioritized list of functional requirements, prioritized list of non-functional requirements along with their corresponding weights (see Figure A.9).

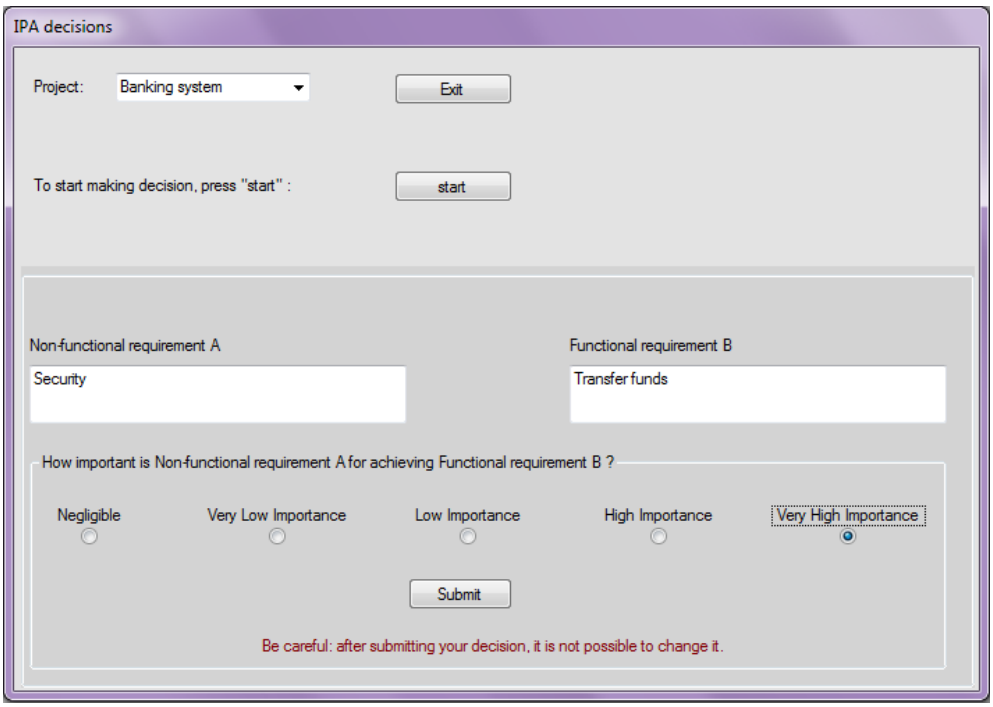
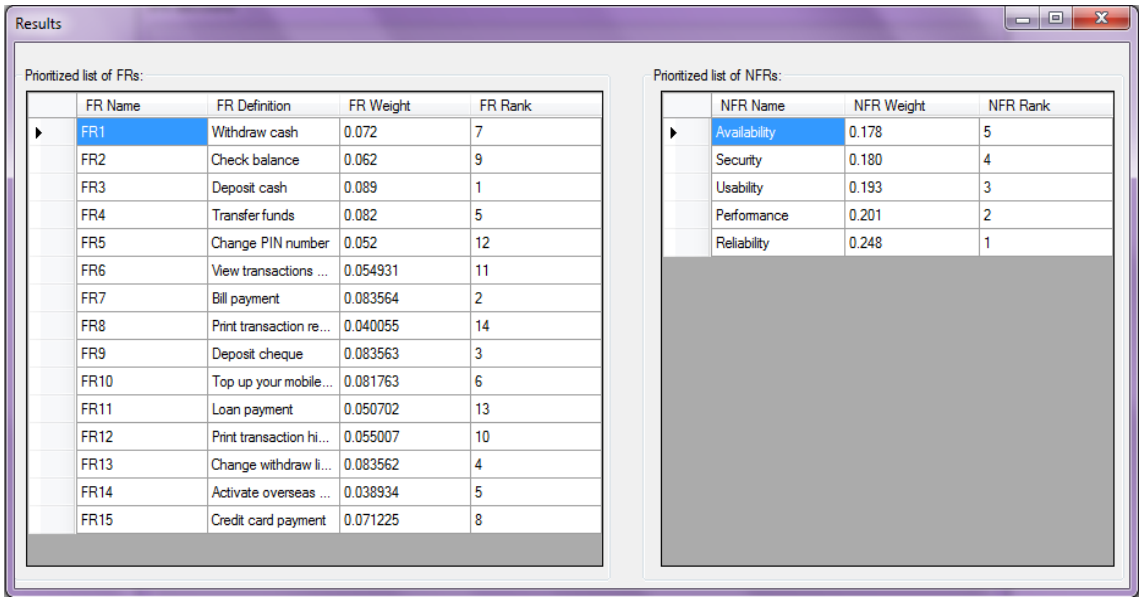


Figure A.8: A picture of the visual user interface displaying the functional requirement, *Transfer funds*, versus the non-functional requirement, *Security*, under analysis with *TIPA*, for project, *Banking system*



FR Name	FR Definition	FR Weight	FR Rank
FR1	Withdraw cash	0.072	7
FR2	Check balance	0.062	9
FR3	Deposit cash	0.089	1
FR4	Transfer funds	0.082	5
FR5	Change PIN number	0.052	12
FR6	View transactions ...	0.054931	11
FR7	Bill payment	0.083564	2
FR8	Print transaction re...	0.040055	14
FR9	Deposit cheque	0.083563	3
FR10	Top up your mobile...	0.081763	6
FR11	Loan payment	0.050702	13
FR12	Print transaction hi...	0.055007	10
FR13	Change withdraw li...	0.083562	4
FR14	Activate overseas ...	0.038934	5
FR15	Credit card payment	0.071225	8

NFR Name	NFR Weight	NFR Rank
Availability	0.178	5
Security	0.180	4
Usability	0.193	3
Performance	0.201	2
Reliability	0.248	1

Figure A.9: Prioritized list of functional and non-functional requirement calculated using *TIPA*

The TIPA software prototype is equipped with a *Search* menu item which provides some tabular and schematic reports automatically. These reports are including:

- Stakeholders weights for a given project

- Non-functional requirements weights achieved by each stakeholder for each project
- Functional requirements weights achieved by each stakeholder for a given project
- Aggregated weights of functional and non-functional requirements of each project

Figure A.10 shows a snapshot of the visual interface of TIPA which indicates the weight of each stakeholder in both tabular and graphical form for a selected project.

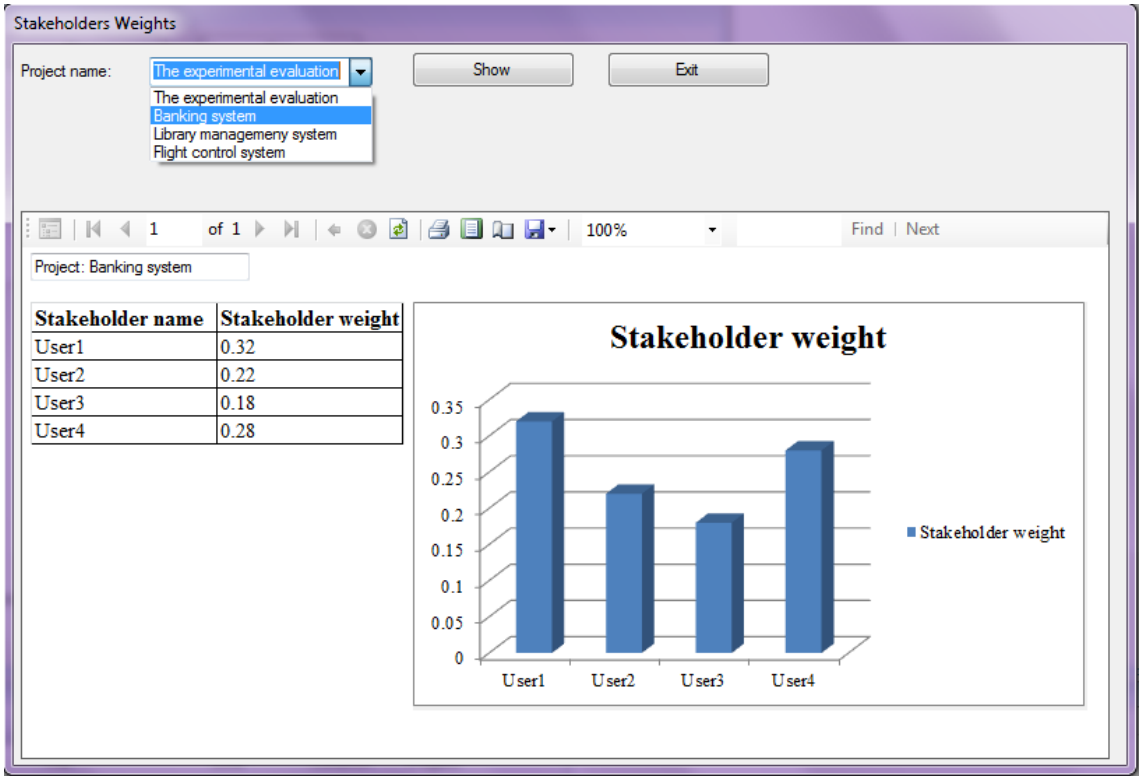


Figure A.10: A graphical view of the report that indicates the weights of stakeholders

Figure A.11 indicates the visual interface of a report which contains the weights of non-functional requirements achieved by each stakeholder for a selected project.

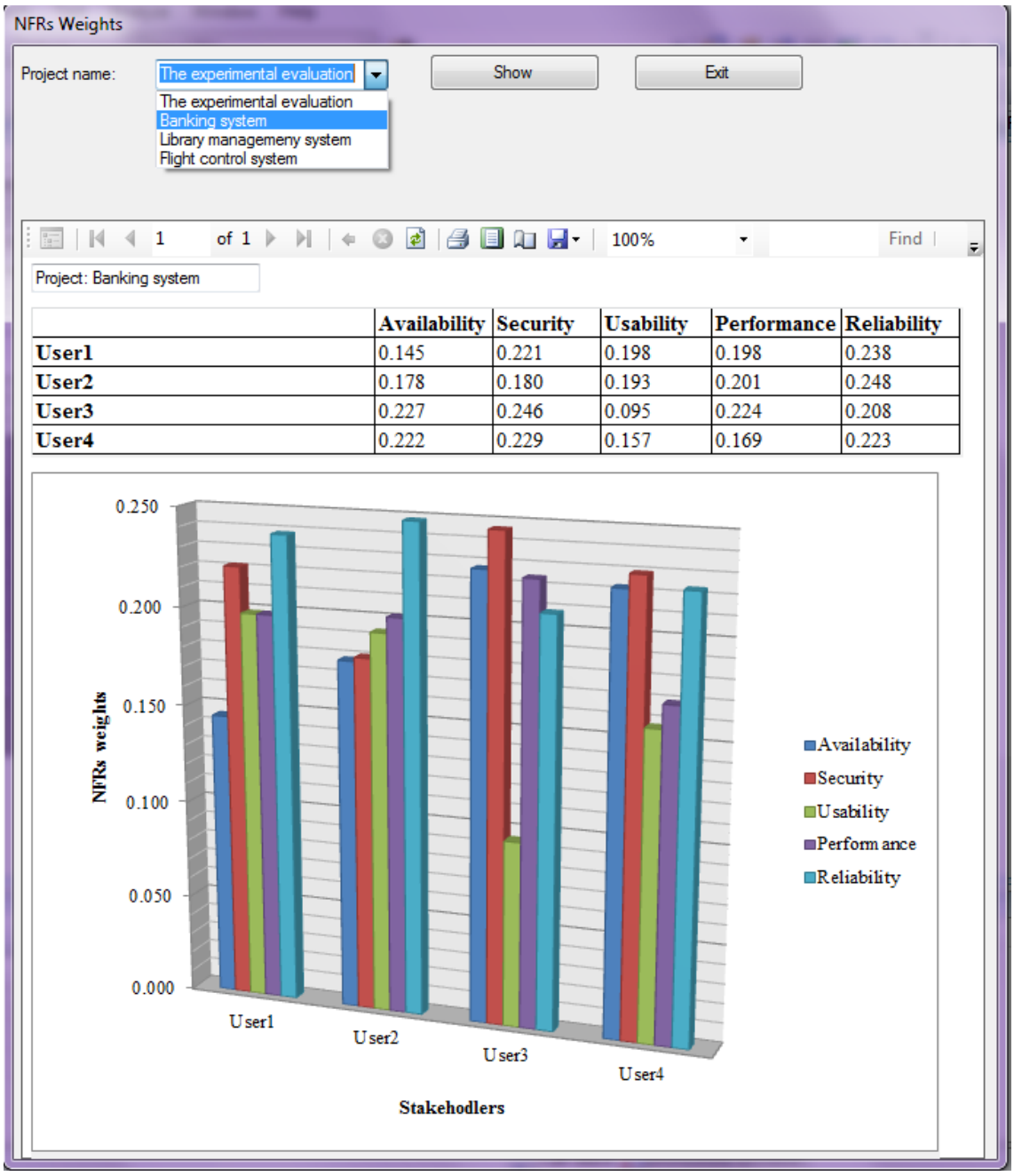
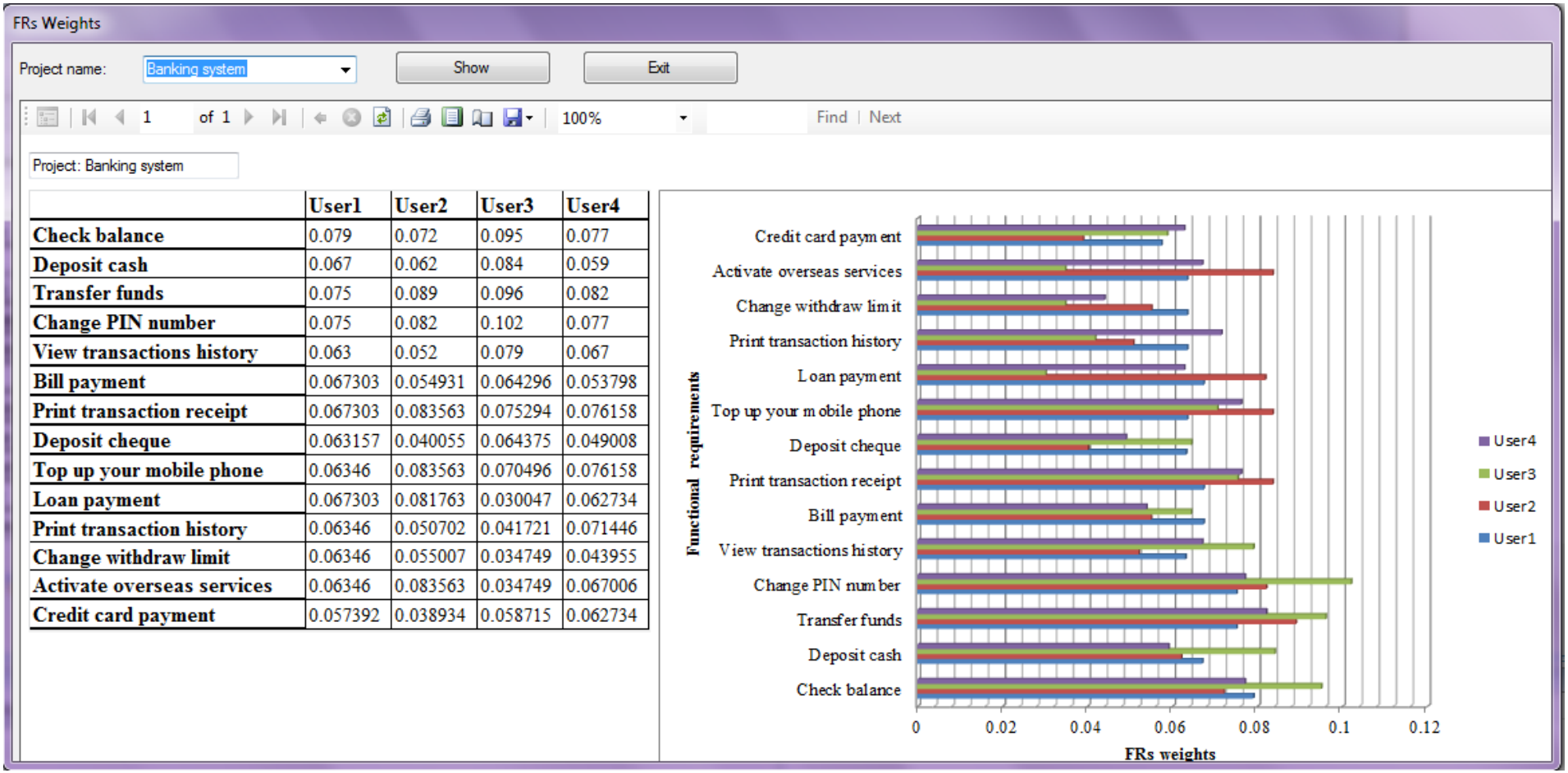


Figure A.11: A visual interface of a report that shows the weights of non-functional requirements achieved by each stakeholder for project, *Banking system*

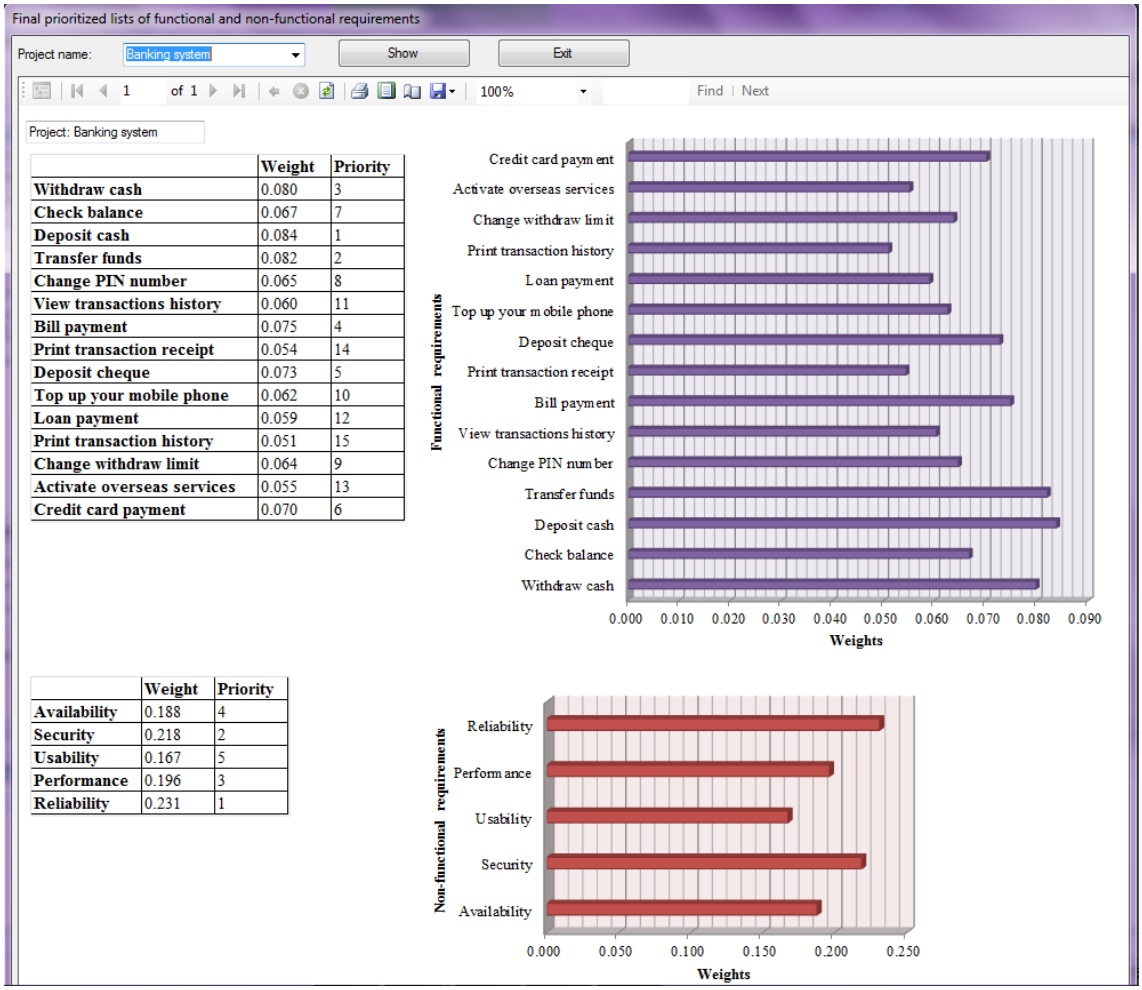
Similarly, Figure A.12 represents the graphical view of a report that shows the weights of functional requirements achieved by each stakeholder for a selected project.



	User1	User2	User3	User4
Check balance	0.079	0.072	0.095	0.077
Deposit cash	0.067	0.062	0.084	0.059
Transfer funds	0.075	0.089	0.096	0.082
Change PIN number	0.075	0.082	0.102	0.077
View transactions history	0.063	0.052	0.079	0.067
Bill payment	0.067303	0.054931	0.064296	0.053798
Print transaction receipt	0.067303	0.083563	0.075294	0.076158
Deposit cheque	0.063157	0.040055	0.064375	0.049008
Top up your mobile phone	0.06346	0.083563	0.070496	0.076158
Loan payment	0.067303	0.081763	0.030047	0.062734
Print transaction history	0.06346	0.050702	0.041721	0.071446
Change withdraw limit	0.06346	0.055007	0.034749	0.043955
Activate overseas services	0.06346	0.083563	0.034749	0.067006
Credit card payment	0.057392	0.038934	0.058715	0.062734

Figure A.12: view of a report that indicates the weights of functional requirements achieved by each stakeholder for project, *Banking system*

Lastly, as can be seen in Figure A.13, TIPA provides a report which contains the final and aggregated prioritized lists of functional and non-functional requirements for a selected project.



	Weight	Priority
Withdraw cash	0.080	3
Check balance	0.067	7
Deposit cash	0.084	1
Transfer funds	0.082	2
Change PIN number	0.065	8
View transactions history	0.060	11
Bill payment	0.075	4
Print transaction receipt	0.054	14
Deposit cheque	0.073	5
Top up your mobile phone	0.062	10
Loan payment	0.059	12
Print transaction history	0.051	15
Change withdraw limit	0.064	9
Activate overseas services	0.055	13
Credit card payment	0.070	6

	Weight	Priority
Availability	0.188	4
Security	0.218	2
Usability	0.167	5
Performance	0.196	3
Reliability	0.231	1

Figure A.13: A visual interface of the TIPA software prototype that shows the final prioritized lists of functional and non-functional requirements for a given project

APPENDIX B: OVERVIEW OF EMPIRICAL STRATEGIES

This section aims to provide a general overview on the most common empirical strategies which can be used for evaluating a new approach in the software engineering domain. It also presents the most commonly used statistical tests which can be exploited for analysing the results of a controlled experiment.

There are three different strategies that can be investigated to evaluate techniques, methods, and approaches (Robson, 2002; Wohlin et al., 2012). These strategies include survey, case study, and experiment. In this section, these strategies are explained briefly. Then they are compared according to different criteria.

- **Survey**. Surveys can be conducted when a technique or method has been already proposed and applied. So, the main goal of investigating a technique or method through surveys is to capture its current status. It is very popular to use surveys for market research and opinion polls. By using surveys, a researcher is not able to control over the execution or measurement. So, it is impossible to manipulate and change variables as in the other strategies (Wohlin et al., 2012). The two most common ways for collecting required data for the research through surveys are interviews and questionnaires. The main difference between interviews and questionnaires is that interviews are online whereas questionnaires are offline. In other words, by using interviews, a face-to-face or voice-to-voice live communication needs to be established in order to capture the required data, while in questionnaires, the data can be collected through some paper forms or electronic forms such as emails or web pages. The benefits of using interviews include achieving higher response rates and decreasing the number of "no answer". It is also possible for the interviewer to observe and ask questions. However, the drawback is time and cost of conducting interviews which restrict its flexibility. The results

obtained from the survey can be analysed to discover descriptive and explanatory conclusions. A comprehensive explanation of surveys are provided in (Babbie, 2013; Robson, 2002).

- **Case study**. Case study research can be conducted to keep track of activities, projects or assignments. Data is captured for a certain objective throughout the case study. After collecting necessary data through case study, statistical analyses such as principal component analysis and linear regression can be carried out to make rigorous conclusions. Indeed, case study research is a strategy by which main variables that may influence the outcome of research are captured and then the activity is documented (Stake, 1995; Yin, 2014). A case study research is an observational study, i.e. it can be conducted by observation and monitoring of an on-going project or activity. In general, case study research is a typical strategy that could be useful for empirical studies in several sciences like medicine, psychology, as well as sociology. In particular, within software engineering domain, case studies have become ideal for industrial evaluation of software engineering techniques, methods, and tools since they can easily prevent scale-up issues. The difference between case study and experiment is that a case study is an observational study while the experiment is a controlled study (Zelkowitz & Wallace, 1998). In addition, the level of control in case studies is lower than in controlled experiments. One benefit associated with case studies is that they are simple to plan. Nevertheless, the drawbacks are that the results are hard to generalize and more difficult to interpret, i.e. it is possible to discover the impacts in a particular circumstance, but it is not possible to generalize the results to general circumstances (Yin, 2014). A thorough description of case study research is given in (Robson, 2002; Yin, 2014).
- **Experiment**. An experiment is a formal, precise, and controlled investigation in which the significant variables are determined and manipulated (Wohlin et al., 2012).

Experiments are typically conducted in a laboratory environment where a researcher intends to have a high degree of control over the situation under investigation in a systematic manner. Once experiment's subjects are assigned to distinct treatments randomly, the main goal would be to manipulate and change one or more variables and monitor all other variables at fixed levels. The effect of the manipulation is measured, and based on the measurement; statistical analyses can be carried out to make a strong conclusion. A good example of an experiment within software engineering is to compare two approaches for requirements prioritization. For this kind of studies, some statistical tests can be carried out with the purpose of showing evidences that an approach is statistically better than the other with a given significance value. The difference between experiments and case studies is mainly based on the notation of a state variable (Pfleeger, 1995). In an experiment, the state variable may receive distinct values, whereas in a case study, the state variable can assume only one value, which is influenced by the actual project under investigation. Conducting a well-designed experiment is not a straightforward task and involves performing different steps systematically.

The presented strategies are compared and classified according to four criteria, which are derived from (Wohlin et al., 2012). These criteria are briefly explained below.

- **Execution control**. This factor indicates the degree of control which the researcher may have over the study under investigation. For instance, in a case study, the data is collected while the project is in execution mode. Therefore, if project is stopped due to any reason, the researcher is not able to continue his/her investigation. In contrast to case studies, the researcher has a high level of control in experiments.
- **Measurement control**. This criterion represents the degree of control which the researcher may have on decision of which measures to be selected, and to include or exclude those measures during execution of the study.

- **Investigation cost**. This factor expresses the cost associated with execution of investigation and it mainly depends on the size of investigation as well as required resources to carry out the investigation. For example, in a survey, the investigation cost is low, since it does not need any large amount of resources for carrying out the survey.
- **Ease of replication**. This aspect indicates the degree of possibility to replicate the investigation. The key goal of replication is to prove that the results of the original experiment are the same as the results collected for the new experiment with the same design, but with different and larger population.

Table B.1 classifies and compares the presented empirical strategies according to the mentioned criteria. This table can possibly be used as a guideline to assist researchers for selecting a suitable strategy for a given investigation.

Table B.1: Classification of empirical strategies

Criteria / Empirical strategies	Execution control	Measurement control	Investigation cost	Ease of replication
Survey	No	No	Low	High
Case study	No	Yes	Medium	Low
Experiment	Yes	Yes	High	High

As mentioned earlier, one of the advantages of carrying out a controlled experiment is that a researcher would be able to analyze the collected data statistically, in order to make a rigorous conclusion. To perform such an activity, there are some statistical tests which can be used to analyze the collected data. In the following, some of the most commonly used statistical tests are presented.

- **T-test**. This test has been considered as one of the most frequently applied parametric tests. The test can be applied to compare two sample means. This test is suitable in the case of one factor with two treatments (Montgomery, 2008).

- **Mann-Whitney**. This test is non-parametric alternative to the t-test. The Mann-Whitney test is further discussed in (Siegel & Castellan, 1988).
- **F-test**. This is a parametric test which could be applied to compare two sample distributions (Montgomery, 2008).
- **Paired t-test**. A t-test for a paired comparison design (Marascuilo & Serlin, 1988).
- **Wilcoxon**. This test is non-parametric alternative to the paired t-test (Siegel & Castellan, 1988).
- **Sign test**. This test is non-parametric alternative to the paired t-test. The sign test is a simpler alternative to the Wilcoxon test (Robson, 2002).
- **ANOVA** (Analysis Of VAriance). This is a parametric test which can be used for different design types including nested design, factorial design, one factor and blocking variable, and one factor with more than two treatments (Montgomery, 2008).
- **Kruskal-Wallis**. This is a non-parametric alternative to ANOVA in the case of one factor with more than two treatments (Siegel & Castellan, 1988).
- **Chi-Square**. This is a non-parametric test which can be applied in those situations where data are formed in terms of frequencies (Siegel & Castellan, 1988).

APPENDIX C: RAW DATA OF CONTROLLED EXPERIMENT 1

This section provides the raw data achieved from executing the first controlled experiment of this study which aimed at comparing IPA and AHP-based approaches in terms of time-consumption, ease of use, and accuracy of results.

C.1) TIME-CONSUMPTION

Table C.1: Results of time-consumption of IPA and AHP-based approaches collected from twenty subjects of Experiment 1

Subject	IPA (Sec)	AHP-based (Sec)
S1	632	947
S2	718	679
S3	269	520
S4	272	609
S5	408	988
S6	834	1137
S7	505	1255
S8	523	759
S9	543	659
S10	735	1373
S11	415	808
S12	379	1219
S13	684	850
S14	521	1054
S15	445	1002
S16	553	1020
S17	462	714
S18	428	797
S19	431	801
S20	424	791

C.2) EASE OF USE

Table C.2: Results of ease of use of IPA and AHP-based approaches collected from twenty subjects of Experiment 1 using post-test 1A

Subject	IPA	AHP-based
S1	4.00	2.00
S2	4.00	2.00
S3	4.00	2.00
S4	4.00	2.00
S5	5.00	2.00
S6	5.00	2.00
S7	5.00	2.00
S8	4.00	2.00
S9	4.00	2.00
S10	4.00	2.00
S11	4.00	2.00
S12	4.00	2.00
S13	4.00	2.00
S14	4.00	2.00
S15	2.00	4.00
S16	2.00	4.00
S17	2.00	4.00
S18	2.00	4.00
S19	3.00	3.00
S20	2.00	4.00

Table C.3: Results of ease of use collected from post-test 2A for Experiment 1

Subject	Ease of use
S1	IPA
S2	IPA
S3	IPA
S4	IPA
S5	IPA
S6	IPA
S7	IPA
S8	IPA
S9	IPA
S10	IPA
S11	IPA
S12	IPA
S13	IPA
S14	IPA
S15	AHP-based
S16	AHP-based
S17	AHP-based
S18	AHP-based
S19	Equal
S20	AHP-based

C.3) ACCURACY

Table C.4: Results of expected accuracy of IPA and AHP-based approaches collected from twenty subjects of Experiment 1 using post-test 1B

Subject	IPA	AHP-based
S1	4.00	2.00
S2	4.00	2.00
S3	4.00	2.00
S4	4.00	2.00
S5	4.00	2.00
S6	5.00	2.00
S7	3.00	2.00
S8	4.00	2.00
S9	4.00	3.00
S10	4.00	3.00
S11	4.00	3.00
S12	4.00	3.00
S13	4.00	3.00
S14	4.00	3.00
S15	3.00	3.00
S16	3.00	3.00
S17	2.00	4.00
S18	2.00	4.00
S19	2.00	5.00
S20	2.00	5.00

Table C.5: Results of perceived accuracy collected from post-test 2B for Experiment 1

Subject	Perceived accuracy
S1	IPA
S2	IPA
S3	IPA
S4	IPA
S5	IPA
S6	IPA
S7	IPA
S8	IPA
S9	IPA
S10	IPA
S11	IPA
S12	IPA
S13	IPA
S14	IPA
S15	AHP-based
S16	IPA
S17	AHP-based
S18	AHP-based
S19	AHP-based
S20	AHP-based

APPENDIX D: RAW DATA OF CONTROLLED EXPERIMENT 2

This section presents the raw data extracted from executing the second controlled experiment of this research which targeted at comparing IPA and HAM-based approaches in terms of time-consumption ease of use, and accuracy of results.

D.1) TIME-CONSUMPTION

Table D.1: Results of time-consumption of IPA and HAM-based approaches collected from twenty subjects of Experiment 2

Subject	IPA (Sec)	HAM-based (Sec)
S1	632	713
S2	718	818
S3	269	324
S4	272	345
S5	408	468
S6	834	937
S7	505	645
S8	523	599
S9	543	653
S10	735	802
S11	415	481
S12	379	523
S13	684	751
S14	521	885
S15	445	596
S16	553	659
S17	462	553
S18	428	522
S19	431	717
S20	424	708

D.2) EASE OF USE

Table D.2: Results of ease of use of IPA and HAM-based approaches collected from twenty subjects of Experiment 2 using post-test 1A

Subject	IPA	HAM-based
S1	4.00	2.00
S2	4.00	2.00
S3	4.00	2.00
S4	4.00	2.00
S5	5.00	2.00
S6	5.00	2.00
S7	5.00	2.00
S8	4.00	2.00
S9	4.00	3.00
S10	4.00	3.00
S11	4.00	3.00
S12	4.00	3.00
S13	4.00	3.00
S14	4.00	3.00
S15	3.00	3.00
S16	2.00	3.00
S17	2.00	4.00
S18	2.00	4.00
S19	2.00	4.00
S20	2.00	4.00

Table D.3: Results of ease of use collected from post-test 2A for Experiment 2

Subject	Ease of use
S1	IPA
S2	IPA
S3	IPA
S4	IPA
S5	IPA
S6	IPA
S7	IPA
S8	IPA
S9	IPA
S10	IPA
S11	IPA
S12	IPA
S13	Equal
S14	HAM-based
S15	HAM-based
S16	HAM-based
S17	HAM-based
S18	HAM-based
S19	Equal
S20	HAM-based

D.3) ACCURACY

Table D.4: Results of expected accuracy of IPA and HAM-based approaches collected from twenty subjects of Experiment 2 using post-test 1B

Subject	IPA	HAM-based
S1	4.00	3.00
S2	4.00	3.00
S3	4.00	3.00
S4	4.00	3.00
S5	5.00	3.00
S6	5.00	2.00
S7	5.00	2.00
S8	4.00	2.00
S9	4.00	2.00
S10	4.00	2.00
S11	4.00	3.00
S12	4.00	3.00
S13	4.00	3.00
S14	4.00	3.00
S15	3.00	4.00
S16	2.00	4.00
S17	2.00	4.00
S18	2.00	4.00
S19	2.00	4.00
S20	2.00	4.00

Table D.5: Results of perceived accuracy collected from post-test 2B for Experiment 2

Subject	Perceived accuracy
S1	IPA
S2	IPA
S3	IPA
S4	IPA
S5	IPA
S6	IPA
S7	IPA
S8	IPA
S9	IPA
S10	IPA
S11	IPA
S12	IPA
S13	IPA
S14	IPA
S15	IPA
S16	IPA
S17	HAM-based
S18	HAM-based
S19	HAM-based
S20	HAM-based

APPENDIX E: POST QUESTIONNAIRES OF CONTROLLED EXPERIMENTS

E.1) CONTROLLED EXPERIMENT 1

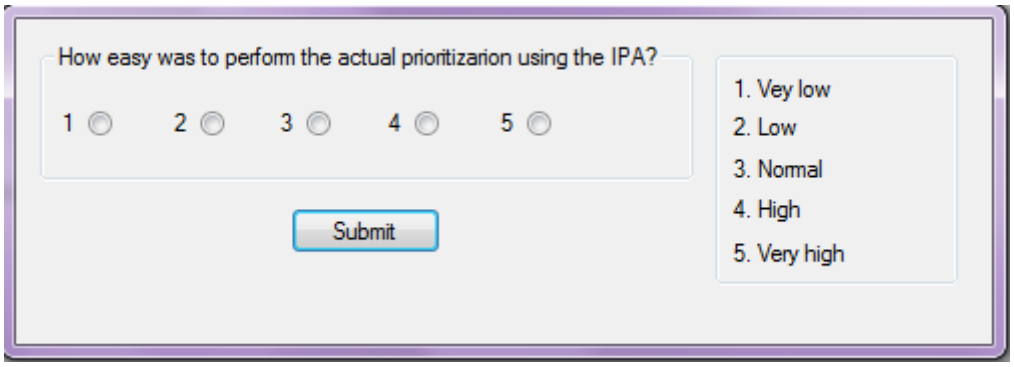


Figure E.1: Post-test 1A of Experiment 1 for assessing ease of use of IPA

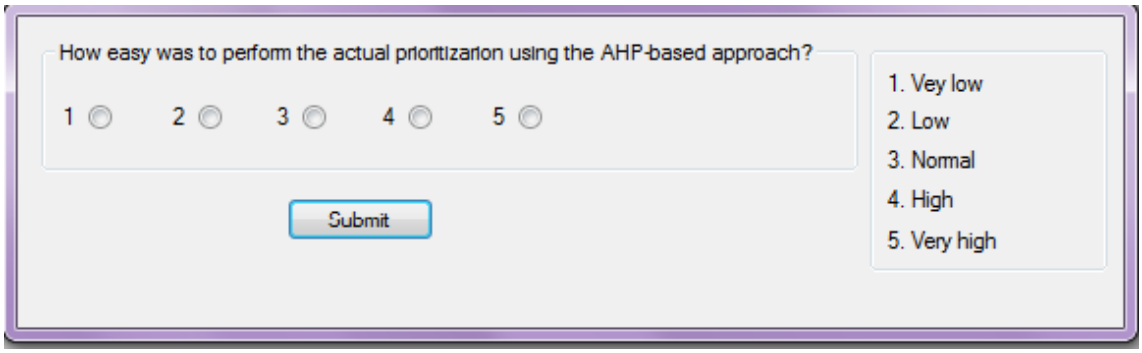


Figure E.2: Post-test 1A of Experiment 1 for assessing ease of use of AHP-based approach

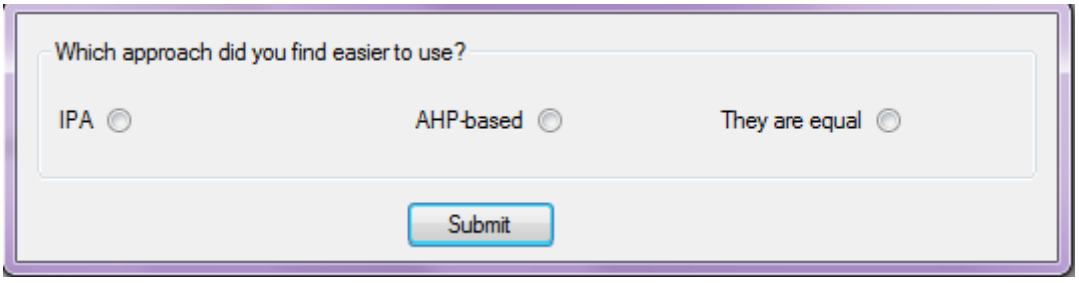


Figure E.3: Post-test 2A of Experiment 1 for comparing ease of use of IPA and AHP-based approach

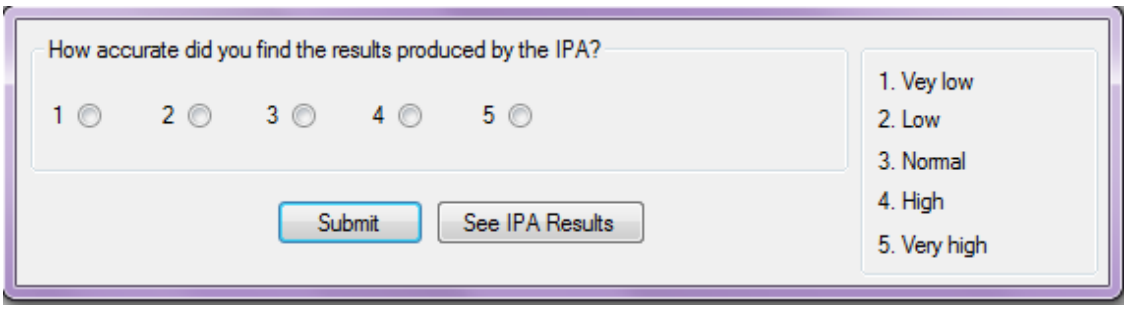


Figure E.4: Post-test 1B of Experiment 1 for assessing expected accuracy of IPA

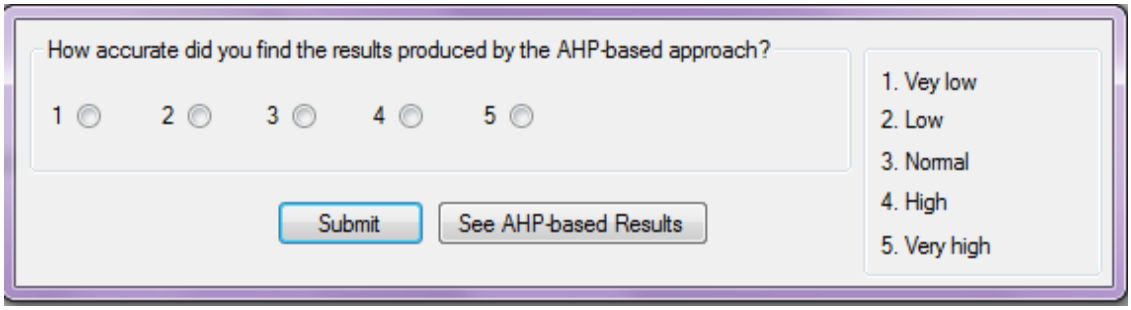


Figure E.5: Post-test 1B of Experiment 1 for assessing expected accuracy of AHP-based approach

D.2) CONTROLLED EXPERIMENT 2

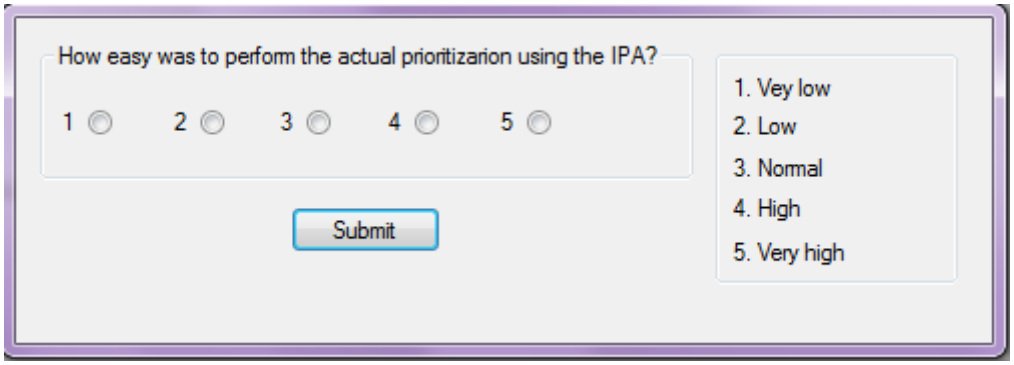


Figure E.6: Post-test 1A of Experiment 2 for assessing ease of use of IPA

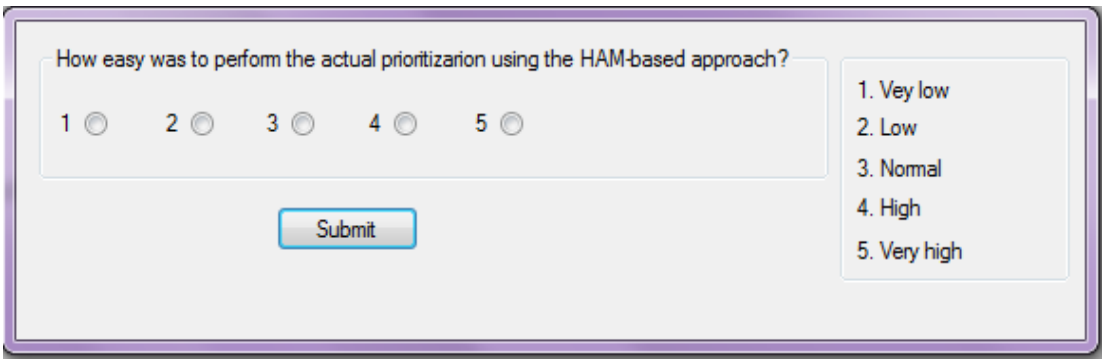


Figure E.7: Post-test 1A of Experiment 2 for assessing ease of use of HAM-based approach

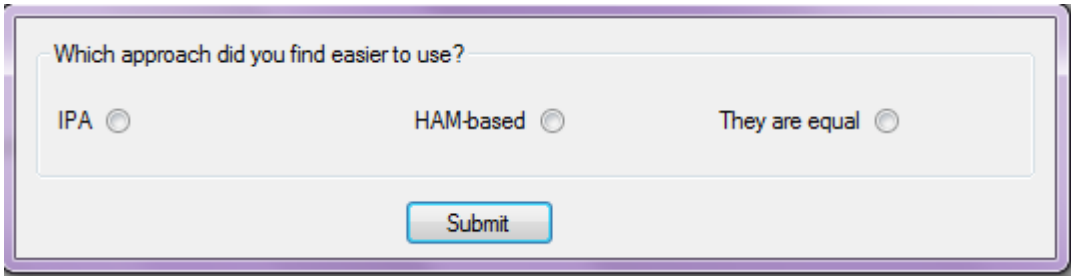


Figure E.8: Post-test 2A of Experiment 2 for comparing ease of use of IPA and HAM-based approach

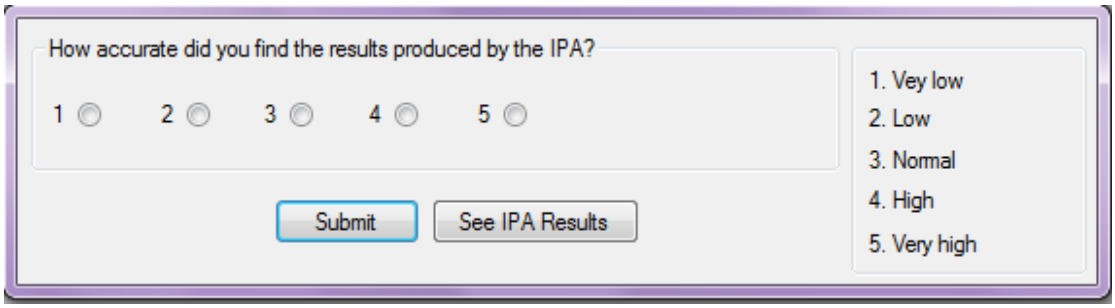


Figure E.9: Post-test 1B of Experiment 2 for assessing expected accuracy of IPA

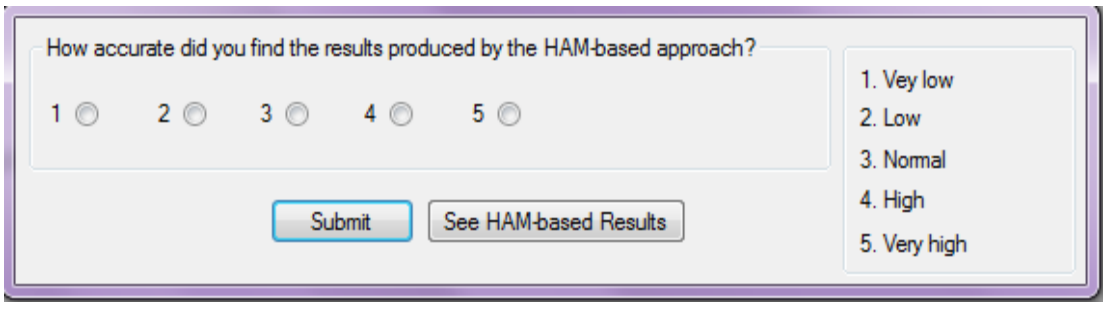


Figure E.10: Post-test 1B of Experiment 2 for assessing expected accuracy of HAM-based approach