Automation of Software Requirement Prioritization using B-Trees

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Gurkiran Kaur
(Roll No. 801132010)

Under the supervision of:
Dr.Seema Bawa
Professor
Dean of Student Affairs

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2013

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "*Automation of Software Requirement Prioritization using B-Trees*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Seema Bawa* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Gurkiran Kaur
12/July/2013
(Gurkiran Kaur)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Seema Bawa
12/July/2013
(Dr.Seema Bawa)
Professor,
CSE Department

Countersigned by

(Dr. Maninder Singh)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

It surely has been a fairly long journey from the inception till the completion of this master thesis. During all this time of ups and downs, I was encouraged and given a lot of support by a lot of people, which I would like to thank with all my heart and sincere feelings.

Firstly I would like to thank my Guide, Dr.Seema Bawa for giving me this opportunity and having faith in me. I am compelled to say that without her intelligent guidance, comments and insightful feedback, this thesis would not have been as good as it is now. With her help and guidance I began to explore systematic reviews and requirements prioritization theories, which later became the foundation for the master thesis. For this I am grateful, otherwise I may still have been wandering around without seeing the end of the path that lay in front of me. In addition to that, I would personally like to thank Ms.Shivani Goel, Ms.Inderveer Channa for providing me sound knowledge of the subject. In addition to that I am highly grateful to Dr. Ravikiran for teaching me the business and management. I would like to thank all the Thapar academic staff who gave me a sound understanding for all the knowledge that I gained while my stay at Thapar. My personal thanks to my class mates especially Haresh Bhai for all their support.

Finally, I would like to thank every member of my family especially my dad, my mother and my brother. I m obliged to my grandfather who I have just lost for guiding me and explaining things about life and world around us. For all his advice, care and help, I am eternally grateful. It's not complete unless I thank God for giving me life and making all this possible. Thank you all for everything!

Abstract

In software system development, it can be a challenge to select the 'right' requirement among several or many options if it is not obvious which requirement is desirable. Requirements prioritization helps us to discover the most desirable requirements. Software developers are seldom able to implement stakeholders' requirements fully when time and resources are limited. To solve the problem, requirement engineers together with the stakeholders must prioritize requirements. It seems that most requirements prioritization techniques work well on a small number of requirements, but many of them have constraints on medium to large numbers of requirements.

We have presented a tool based on the 100 dollar technique and Btree method to deal with prioritization of requirements. The 100 dollar approach consists of a method that gives priority values to the requirements according to the stakeholders, in order to aid system management and software quality. Every stakeholder has its own needs and therefore gives different importance value to different requirements. The 100 dollar method allows an input from all the stakeholders for the requirements. The priority values help the system analysts, developers and customers make decisions about the requirements. Based on the values given by the approach, requirements can be skipped or developed later.

The need to prioritize increases with the number of requirements so it is difficult to remember the priority values and to rank them according to the values. Knowing the rank of the requirements, the plan of the releases can be done. It is possible to plan by knowing which functions are critical and how they can be distributed. To find out the rank and the order in which these priorities can be implemented a method of Btree is used. Btree will help in ordering the requirements and also will allow an easy way to add, edit or delete the requirements.

Table of contents

Chapter 1 Introduction 1

1 Software Engineering 1

1.1 Software 3
1.1.1 Software Characteristics 3
1.2 Requirements Engineering 4
1.2.1. The Cost of Fixing Errors 5
1.3 Requirements Prioritization 7

1.4 The Usefulness of Requirement Prioritization 8

1.5 Aspect of Requirement Prioritization 10

1.6 Requirements Prioritization, Stakeholders and Roles 12

1.7 Organisation of Thesis 14

Chapter 2 Literature Review 16

2.1 Techniques of Requirements Prioritization 17
2.1.1 Numerical Assignment Method 17
2.1.1.1 Numerical Assignment Technique 17
2.1.1.2 MoScoW 17

2.1.2 Pairwise Comparison Technique 18
2.1.2.1 Simple Ranking 19
2.1.2.2 Bubble Sort 19
2.1.2.3 Analytic Hierarchy Process 20
2.1.2.4 Hierarchy *AHP* 20
2.1.2.5 Minimal Spanning Tree 22
2.1.2.6 Cost-Value Approach 22

2.1.3 Mixed Methods 23
2.1.3.1Binary search tree 23
2.1.3.2 BTree 23
2.1.3.3 Planning Game 24
2.1.3.4 Hundred Dollar Method 24

2.1.4 Comparison between the Techniques 24

26

Chapter 3 Purposed Requirement Prioritization Tool

3.1 Research Gaps 29

3.2 Problem Formulation 29

3.3 Objectives of Thesis 29

3.4 Methodology 30

Chapter 4 Design and Implementation 30

4.1 Features of the Lets Tree Prioritise Tool 31

4.2 UML Specification: Data Flow Diagram 31

4.3 Working of the Implemented Tool 31

4.3.1 User Authentication 33

4.3.2 New Project 33

4.3.3 Direct method 33

4.3.3.1 New Project Unit 34

4.3.3.2 Enter the number of requirements 34

4.3.3.3 Enter the Name & Priority Value of Requirement 35

4.3.3.4 Display Data 35

4.3.3.5 Edit, Delete or Add the Requirement 36

4.3.3.6 Prioritise the Requirements 36

4.3.4 100 Dollar Method 38

4.3.4.1 Selecting No Of Requirements & Number Of Stakeholders 38

4.3.4.2 Name of the Requirements 39

4.3.4.3 Stakeholders Form 39

4.3.4.4 The Display Form 40

4.3.4.5 Prioritised Data 40

4.3.5 Open Project 41

Chapter 5 Testing and Experimental Results 41

5.1 Testing 42

5.2 Test Cases 42

5.3 Type of testing done on the Proposed Tool 43

5.3.1 Unit Testing 53

5.3.2 Integration Testing 53

5.3.3 System Testing 54

5.3.4 Alpha Testing 55

55

Chapter 6 Conclusion and Future Scope 56

6.1 Conclusion 56

6.2 Future Scope 57

References 58

List of papers published 62

.

List of figure

1.1: Development cycle of the project 3

1.2: Bathtub Curve of Failure 4

1.3: Requirement Engineering 6

1.4: Cost of Fixing Errors 7

1.5 Requirement Prioritization 9

1.6 Usefulness of Requirement prioritization 12

1.7 Stakeholders 15

2.1 MoScoW Prioritization 18

2.2: Cumulative Voting 25

2.3 Comparison between the prioritization techniques based on certain parameters 27

2.4 Value Oriented Prioritization matrix 28

4.1: Data Flow Diagram of the Project 32

4.2: Snapshot of User Authentication 33

4.3: Snapshot of New Project Module 34

4.4: Snapshot of New Project 34

4.5:Snapshot of Number of Requirements 35

4.6: Snapshot of data entry of requirement and the priority value 35

4.7: Snapshot of Display data 36

4.8: Snapshot of Edit, add or delete data 37

4.9: Snapshot of Prioritised Requirements 38

4.10: Snapshot of selecting 100 dollar method 38

4.11: Snapshot of data entry 39

4.12: Snapshot of Requirements Form 39

4.13: Snapshot of Stakeholders form 40

4.14: Snapshot of Display data 40

4.15: Snapshot of Prioritised Data 41

4.16: Snapshot of Open Project Module. 41

5.1: Cost of testing at every lifecycle phase 42

5.2 Login Authentication 47

5.3 Login authentication 48

5.4 Main Form 48

5.5: Invalid Requirement 49

5.6: Priority Range 49

5.7: Incomplete Form 50

5.8: Selection Query 50

5.9: Deletion Query 51

5.10: Data entry check 51

5.11: Incomplete form 52

5.12: Total exceeding 100 check 52

5.13: Total less than 100 check 53

5.14 Unit testing 54

5.15 Integration Testing 54

5.16 System Testing 55

List of Table

2.1 Fundamental Scale used for AHP 22

2.2 Comparison between Different Techniques 26

5.1 Test Cases of the Tool 45

Chapter 1

Introduction

The purpose of this chapter is to introduce the Software and Requirements Engineering area to the reader. The chapter also presents requirements prioritization along with the different techniques for prioritizing different requirements under different situations. In addition to that, the chapter throws light on different roles of the stakeholders and takes into consideration different issues related to different types of stakeholders.

1. Software Engineering

"Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software that is, the application of engineering to software.

Engineering is the analysis, design, construction, verification, and management of technical entities. In spite of the entity that is to be engineered, the following questions must be asked and answered:

• What is the problem to be solved?

• What are the characteristics of the entity that is used to solve the problem?

• How will the entity be realized?

• How will the entity be constructed?

• What approach will be used to uncover errors that were made in the design and construction of the entity?

• How will the entity be supported over the long term, when corrections, adaptations, and

Enhancements are requested by users of the entity?

The phases of Software development process are:

The **definition phase** focuses on what. The software developer attempts to identify what information is to be processed what function and performance are desired, what system behaviour can be expected, what interfaces are to be established, what design constraints exist, and what validation criteria are required to define a successful system. The key requirements of the system and the software are identified. The methods applied during the definition phase will vary depending upon the software engineering hypothesis that is applied, three major tasks will occur in some form i.e. System or information engineering, software project planning and requirements analysis.

The **development phase** focuses on how i.e. during development a software engineer attempts to define how data are to be structured, how function is to be implemented as a software architecture, how procedural details are to be implemented, how interfaces are to be characterized, how the design will be translated into a programming language and how testing will be performed. The methods applied during the development phase will vary, but three specific technical tasks should always occur

- Software design
- Code generation
- Software testing.

The **maintenance phase** focuses on change that is associated with error correction, adaptations required as the software's environment evolves and changes due to enhancements brought above by changing customer requirements. The maintenance phase reapplies the steps of the definition and development phases, but does so for the existing software.

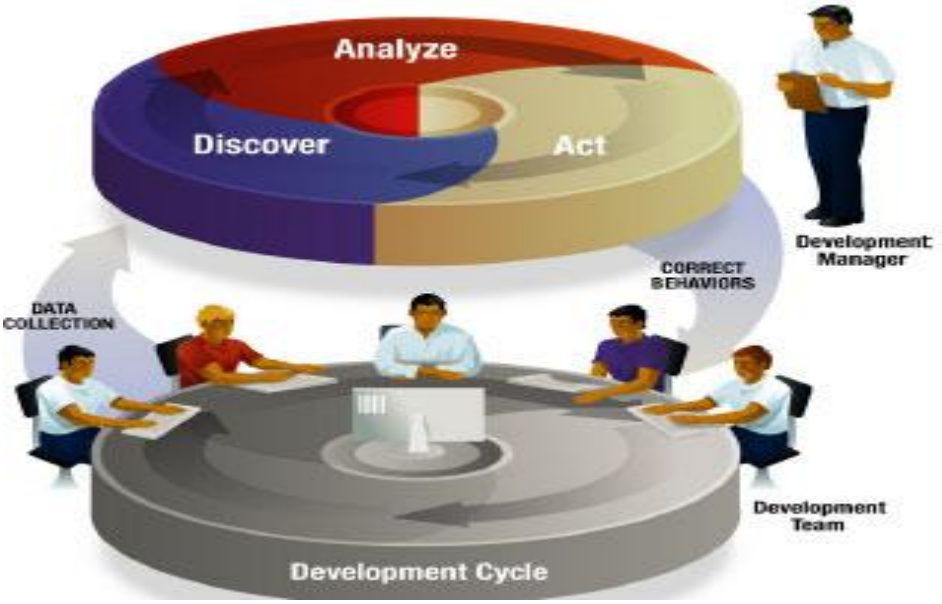


Fig 1.1: Development cycle of the project

1.1 Software

Software is:

i. Instructions that when executed provide needed function and performance

ii. Data structures that enable the programs to adequately manipulate information and

iii. Documents that describe the operation and use of the programs.

1.1.1 Software Characteristics

i. Software is developed or engineered; it is not manufactured in the classical sense. Although some similarities exist between software development and hardware manufacture the two activities are fundamentally different. In both activities high quality is achieved through good design but the manufacturing phase for hardware can introduce quality problems that are nonexistent (or easily corrected) for software. Both activities depend on people but the relationship between people applied and work accomplished is entirely different.

ii. Software doesn't wear out; Figure 1.1 depicts failure rate as a function of time for hardware. The relationship, often called the "bathtub curve", indicates that hardware exhibits relatively high failure rates early in its life (these failures are

often design or manufacturing defects) defects are corrected, and the failure rate drops to a steady state level (hopeful, quite low) for some period of time.

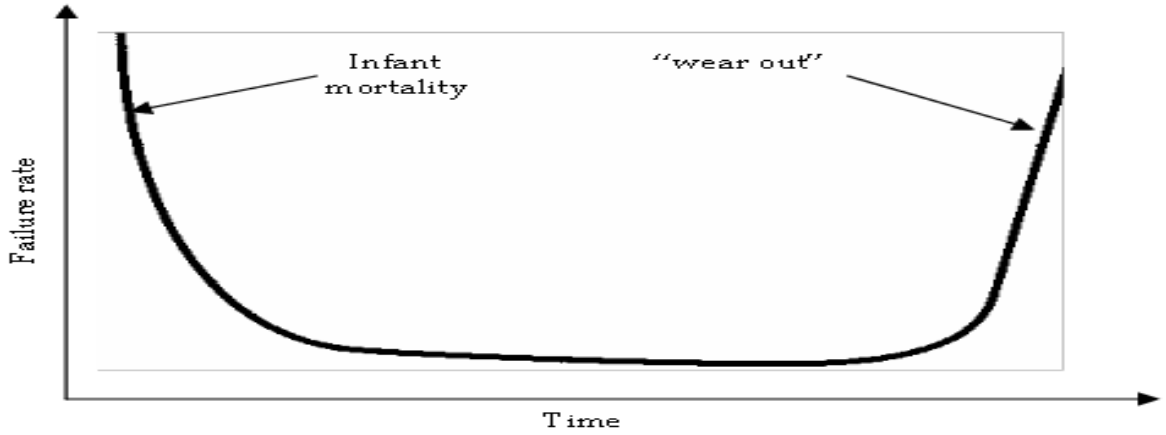


Fig 1.2: Bathtub Curve of Failure.

As time passes, however, the failure rate rises again as hardware components suffer from the cumulative effects of dust, vibration, abuse, temperature extremes, and many other environmental maladies. Stated simply, the hardware begins to wear out.

iii. Most software is custom–built rather than being assembled from existing components; sadly, software designers are not afforded the luxury described above. With few exceptions there are no catalogues of software components. It is possible to order off- the –shelf software, but only as a complete unit not as components that can be reassembled into new programs. Although much has been written about "software reusability we are only beginning to see successful implementations of the concept".

1.2. Requirements Engineering

The main objective during software development always has been to achieve the high quality in developed software and make more efficient the development and organizational processes. The software community is determined for the discovery of ideal suite of processes and methodologies to deliver high quality software with high reliability, efficiency, and within time and budget constraints. Failures are often caused by the ever-increasing complexity of the software and the solutions it

tends to provide [1]. Quality is highly subjective and is also related to software as well. Software unlike any other product is an artefact of a software vendor and is targeted for a single or a wide variety of audience and people. Some software developers still continue to believe that the software quality is something you begin to ponder and worry about after the product has been implemented and the code has been generated for the system [2].

The main challenge of the software engineering community is to satisfy the customer needs and possibly exceed his expectations [3] in an economic, rapid and profitable manner [4].The development of the rigorous and defined methodologies has the potential of fulfilling the customer needs and possibly exceeding them. Requirements engineering and its underlying activities can help organizations in realizing customer needs and developing quality software in an economic manner.

Requirements engineering is not just engineering but rather a multi-disciplinary approach that encompasses other fields and domains especially the human and social sciences. It draws on the social and cognitive sciences in order to endow withtheoretical ground and practical knowledge and techniques for requirements elicitation, analysis, documentation, management and modelling. Some of the fields used in the requirements engineering are cognitive psychology, sociology, human psychology and linguistics. These are well structured and disciplined areas that contribute a lot towards the advancements in requirements engineering [5]. Software Requirement is generally described as: what the system is required to do along with the environment it is made to operate in. Requirements provide the description of the system, its behaviour, application domain information, system constraints, specifications and attributes [6]. They also signify the quality that the product must exhibit [7] and the people who are affected by the usage of the system.

In enormous and majority of the cases reported, system failures are not subject to incompetent people (developers), organizations and the software engineering methodologies itself. In fact, the reality is that they are the result of the problems with requirements of the system [6]. Requirements engineering aim is to make system requirements clear and understandable so that they reflect the actual needs

of the stakeholder.

Effectiveness, flexibility and applicability of a system largely depend on the correct understanding of the needs of the systems stakeholders and it has been a long established fact [8].Requirements Engineering is regarded as the front end activity and is mostly the outcome of the system engineering process. The challenge faced by the software community is whether we have specified a system that is indeed the true reflection of the customer needs [2]. There is no fix answer to that but a good, sound and solid requirements engineering process has the potential of turning the customer voice into a realizable system.

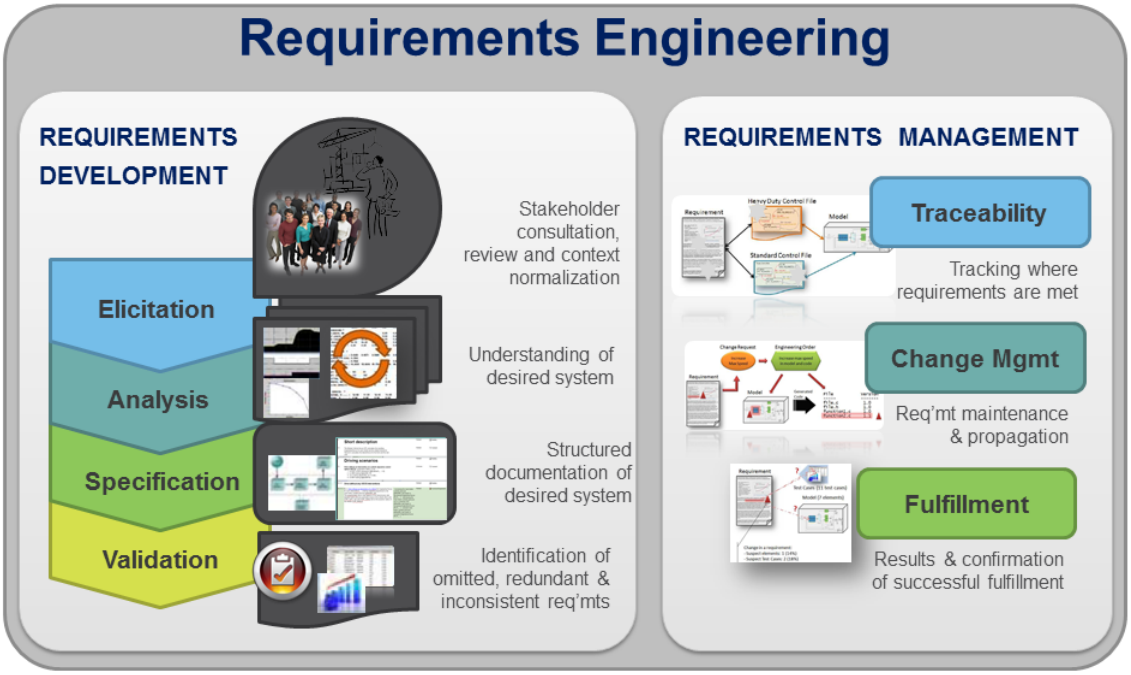


Fig 1.3: Requirement Engineering

Requirements engineering process is different for different products and organizations [9] e.g. requirements engineering for embedded systems or safety critical system is different from requirements engineering for web-based solutions. The variability in the requirements engineering process is often regarded as good as there exists no ideal requirements engineering process and is based on the technical maturity, disciplinary involvement, organizational culture, application domain [6], design decisions [10] of the individuals as well as the market situation.

In addition to that, requirements engineering is also heavily influenced by system acquisition and the commercial, legal and contractual issues [6].

In our normal day life, we are mostly faced with different situations when we have to make decisions between different choices e.g. if we buy a new car, some people go for speed, some go for luxury, others go for performance and yet others go for the cost. One of the major activities within the requirements engineering process is to use requirements prioritization that helps in specifying the right requirement.

1.2.1. The Cost of Fixing Errors

Most of the projects followed a fairly standard sequence of phases: requirements analysis, design, coding, development testing, acceptance testing and operation.

Errors made in a particular phase cost more to fix the longer they are left undetected, so that, for Example, a programming error that is not discovered until acceptance testing will cost ten times as much to fix than if it is found during programming. A requirements error not found until after delivery will cost a hundred times more to fix. The explanation is fairly simple, the longer a problem goes unnoticed, the more subsequent design decisions are based on it, and hence the more work it will take to unpick them.

Therefore the earlier an error is detected the better will be the payoff.

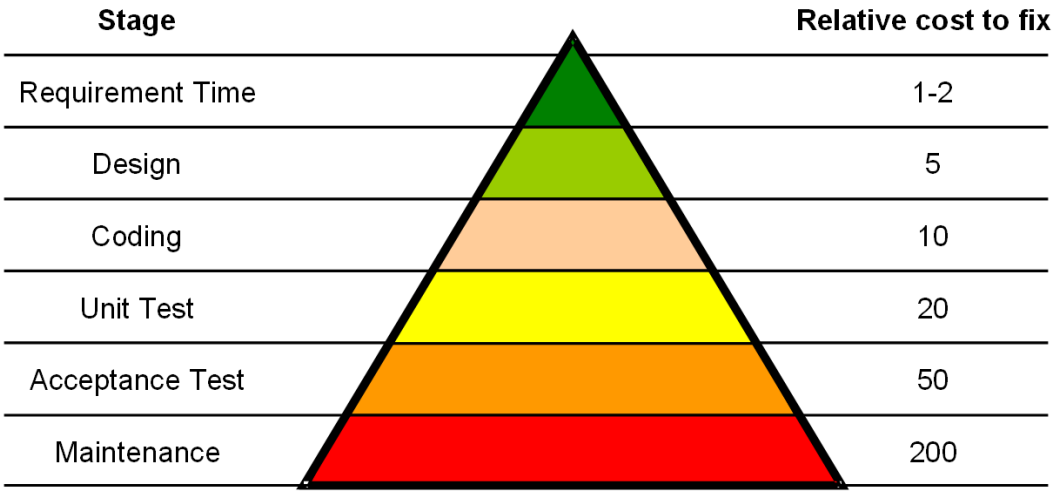


Fig1.4: Cost of Fixing Errors

1.3 Requirements Prioritization

According to Fred Brooks the hardest single part of building a software system is deciding what to build. No other part of the work so fails the resulting system if done wrong. No other part is harder to repair later [11]. Therefore it is really necessary to select the right requirements from a set of requirements in order to develop cost effective quality software. One way of accomplishing this is to prioritize the requirements to enable the selection of optimal set [12]. There is a wealth of literature that states the importance of prioritizing requirements. One of the most recognized authors in the field of software engineering, Ed Yourdon believes that prioritization of requirements is an extremely important issue [13,14] and that prioritization of requirements was a topic that came up repeatedly among the market-driven projects they reviewed. Siddiqi says prioritization as a key but neglected issue in requirements engineering research [15].

Requirements at the earlier stages are vague and evolve and become more specific and clear with the passage of time. There is no specific phase where requirements prioritization takes place; rather it is an iterative process which is performed throughout the product development life cycle [16] and between different versions of the same product as it is evolved [17].

Berander argues that requirements prioritization is not just asking the different stakeholders about their different priorities related to the system requirements. It is also a social process which requires considering other issue like organizational setup, organizational, market value, organizational issues, stakeholders personalities and agendas [12]. It also requires tradeoffs during conflicting situations between the different stakeholders to reach to a situation when the candidate requirements have been selected which will eventually form a system.

Berander provides a comprehensive list of advantages of requirements prioritization. Requirements prioritization helps in selecting the final candidate requirements from a set of many requirements, which can be realized within time and budget constraints they help stakeholders decide the final core requirements of the software. They also help in selecting optimal requirements sets, which be implemented in successive releases. Requirements prioritization helps in balancing the business benefits of each

requirement against the cost of implementing [12]. Requirements prioritization often involves negotiation process to handle contradictory requirements in order to resolve disagreement between stakeholders [6].With requirements prioritization, the developing organization can not only minimize rework and plan stability but it can also get a technical advantage and optimize the market opportunity [12]. All these factors are a key to success in the changing business trends and in keeping an organization ahead of other business competitors. It should be kept in mind that requirements must be gathered and discovered before the start of the project development because their discovery during development or after deployment posses an overhead for the organization implementing the system [18] and is inefficient. Of course, rectification is possible using change management, however this adds to the cost of development process [12]. Another way of handling such volatility of requirements is using incremental methodologies or agile methods like Extreme Programming [19], but these methodologies call for the presence of customer on site all the times that is not possible in many situations. Therefore requirements should be prioritized in order to find the final set of requirements and develop a system based on those requirements.

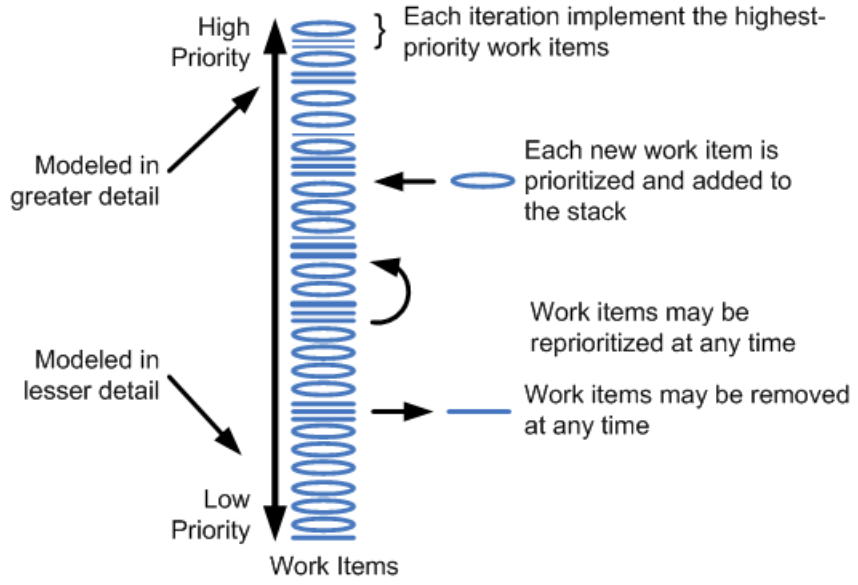


Fig 1.5: Requirement Prioritization

Requirements prioritization involves both quantitative techniques and qualitative negotiation method [20].The quantitative techniques also require negotiations between different stakeholders, and assign numeric number to requirements. Requirements prioritization also helps in detecting requirements ambiguities, unsuitability and abstraction [21].

Different people may see the meaning of requirements prioritization from different angles. It defines requirements prioritization as the activity during which the most important requirements can be revealed which defines requirements prioritization as the process to determine the implementation order of the requirements for implementing the system or the process to determine the order of importance of the requirements to the stakeholders. Compare the definitions provided by the two authors, [6] mainly focuses on the importance of the requirements to the users. But sometimes requirements contain dependencies. This may result in the implementation order of a system being different from the order of importance to the stakeholders. For example, in order to implement the high value requirement A , requirement B needs to be implemented first even though requirement B is of low value in itself. System developers can prioritize requirements to see which requirement can be implement first.

1.4 The Usefulness of Requirement Prioritization

Priority is the relative right of a requirement to the utilization of limited resources. Resources in this definition are taken to mean all types of resources, including time, money and human resources. In other words, if the resources are unlimited, we can have all, therefore there is no need to prioritize requirements. But generally, projects face limited resources such as short timelines, small budgets, restricted human power, and limited technology. As a result, projects often contain more candidate requirements than can be implemented in one product release time. Stakeholders need to decide which requirements should be implemented. Requirements prioritization helps the project developers to select the final candidate requirements within resource constraints. One project may contain hundreds or even thousands of requirements. Generally, not all the requirements contain equal user satisfaction. It is often not obvious which requirement contains high user satisfaction among hundreds or thousands of requirements. When only

one stakeholder is involved in the project, it is relatively easy to make decisions since only one stakeholder's opinion needs to be considered. When more than one stakeholder is involved in the project, decisions can be harder to make, since different stakeholders have different perspectives.

For example, project developers look for the requirements which can be implemented fast, financial managers look for the requirements with low cost, market managers look for the requirements with high market value, and end users look for the requirements which are easy use. One requirement may be of low cost, with short implementation time, but also have low market value and be hard to use. Conversely, another requirement may have a high cost, but short time to be implemented, high market value and be easy to use. It can be a challenge for stakeholders to decide which requirements need to be implemented first. Requirements prioritization is a technique that can uncover the most important requirements to maximize the stakeholders' satisfaction.

Nowadays, projects are still suffering low success rates. Chaos Report indicates only 32% of all projects are "successful" in the sense they are delivered on time, on budget, and with the required features and functions. 44% are described as "challenged" meaning they are delivered late, over budget, and/or with less than the required features and functions. The remaining 24% failures, being terminated before completion or delivered but never used. Ten main factors causing challenged or failed projects are unveiled. Four of them are lack of user involvement, lack of resources, unrealistic expectations, and changing requirements and specifications. Requirements prioritization increases user involvement by letting the stakeholders decide which requirements the project should contain. It helps stakeholders hold realistic expectations by letting the stakeholders understand the current constraints on resources and accepting the trade-off decisions on conflicting perspectives. Karlsson and Ryan (1997) think it helps stakeholders allocate resources based on the priorities of the requirements [38]. Karlsson, Wohlin and Regnell (1998) think it helps stakeholders detect requirements defects, such as misunderstanding or ambiguous requirements, to reduce the number of changes to requirements and specifications in the later stage of projects [34]. Hatton (2007) says requirements prioritization has become an essential step in the software development process in order to reduce software failure. The requirements prioritization has been recognised as one of

the most important decision making processes in the software development process. Fig 1.5 shows how important a requirement is and what the priority of its implementation is. It's based on:

- Business value
- Difficulty of implementation

  Easy to develop having high business values are to be implemented with the highest priority.

  Difficult to develop (Costly, Time Consuming, requiring huge effort) and having low business value are at lower priority.

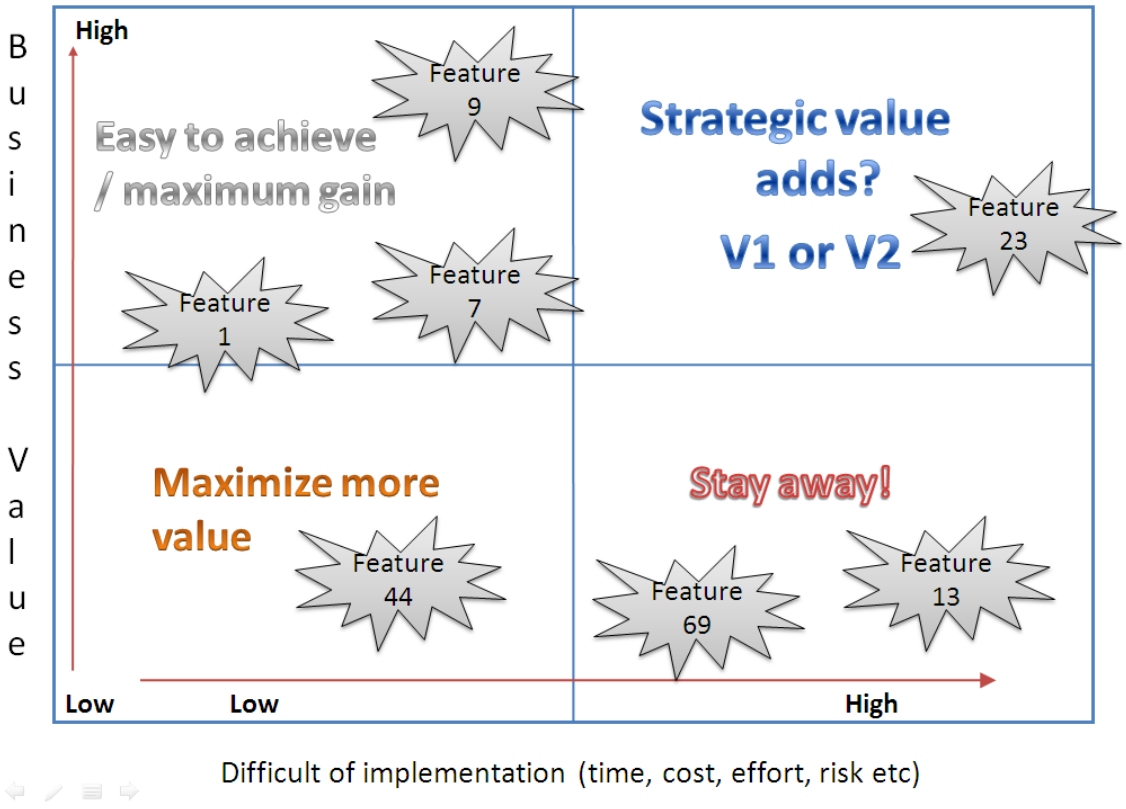


Fig 1.6 Usefulness of Requirement Prioritization

1.5 Aspect of Requirement Prioritization

Requirements can be prioritized based on the different aspects of the project. Other names used to describe "aspect" are 'element' [22], 'factor' [23], 'criteria' [24] and 'parameter' [25] etc. Aspect can be defined as a property or attribute of a requirement

[12].Examples of different requirements aspects include: time, cost, penalty, importance, volatility, risk, dependency, abstraction level etc. Prioritizing requirements on single aspect is easier since it is much easier to decide which is the most needed e.g. just considering importance in terms of implementation, it is easier to see which requirement is more important in comparison to others. However importance is also dependent on certain factors like requirements complexity, reusing existing code, testing amount, value it brings to the customer or organization etc [12]. Therefore one can easily see as the aspects become more, deciding and prioritization becomes tedious and cumbersome and involves much more insight and requires much more effort.

Although different aspects can be prioritized, however it is not always desirable and practical to consider all aspects. Rather it is dependent on the specific situation, product specification, stakeholders viewpoint [12] e.g. while considering importance aspect, stakeholders should prioritize which requirements are the most important and will become part of the system. However importance is highly subjective thing that varies considerably from one stakeholder to another [16]. Volatility is another crucial aspect that relates to the changing requirements [26] and depends on market trends, user changes, or getting deeper understanding of the requirements etc. Therefore, prioritization of requirements based just on importance and volatility becomes time consuming and challenging since each aspect is not mutual and also dependent on many other factors. In a real time situation, it is always necessary to combine different aspects of requirements together before deciding to implement it directly, later or simply not to include it in the system at all. There are many ways of combining different aspects using different prioritization techniques however, they are situation dependent, product dependent and stakeholder dependent [12]. Requirements prioritization should also consider business issues and implementation issues. Business issues might involve financial benefits for the developing organization, market trends and focus, competitors, regulations whereas implementation issues mostly involve implementation cost, cost if not implemented, available resources etc [16]. Another important aspect to be considered while prioritizing requirements is the customer perspective along with the perspectives of

developers and financial personals [27]. Customers provide vital information about the user/customer value; developers are better suited for the technical details whereas financial personals know about the cost, budgets and related risks [27]. In addition, all those perspectives can be involved and combined that adds value to the project and that have stake in the project or product [12].

Requirements prioritization is dependent on many situations, different stakeholders and their associated roles and is discussed in detail in the next section.

1.6 Requirements Prioritization, Stakeholders and Roles

Customer is the person or a group of people that are involved in a project. They are benefitted from the project in some way while developing software product, three customer situations exists [12]. These are:

- One Customer
- Several Known Customers
- Mass-Market Development

The situation of bespoke development and mass-market development are the two extremes. In majority of the situations using bespoke development, a system is developed for one customer (organization) which will be used by many people e.g. the software for the ATM machine will be developed for a particular bank but will be used by the clients of that bank. In mass-market development, the software is developed for potentially everyone in the whole world. But as mentioned earlier, these are the two extremes and there exits many situations in between the two, e.g. in the telecommunication and health sector, there could be many known customers which will use the software but they do not correspond to either bespoke or market driven development. These do not qualify for bespoke development since many operators or hospitals will be the customers and at the same time the developed software will not be developed for a mass market [12].

Different situations require different prioritization mechanisms either alone or in combination to be employed in order to come up with right requirements because of the nature of product and involvement of different stakeholders. In one customer situation, prioritization is easier as there are only one customer's priorities

that are to be considered. However, one of the main issues to be kept under consideration is that the customer and the end-users are not always the same [12].Thus their priorities are not same all the times. Therefore it is always advisable to engage end users in the prioritization process [28]. In case of several known customers, prioritization becomes difficult because of the different personalities as well as conflicting viewpoints, objectives and preferences of the individuals. The challenge here is to arrive at a win-win situation in which each stakeholder agrees to the requirements [29], [30] and it requires trade-offs based on different criteria [28].

Fig 1.7 shows the stakeholders that are associated in a project. The number of stakeholders varies from project to project.

Some Projects may have lesser no. of stakeholders associated and some projects may be more critical and hence may have multiple stakeholder.

The two categories of stakeholders are as follows:

Internal: The people or the organisation directly benefited from the project or the software.

External: the people or the organisation that are indirectly benefited from the software.

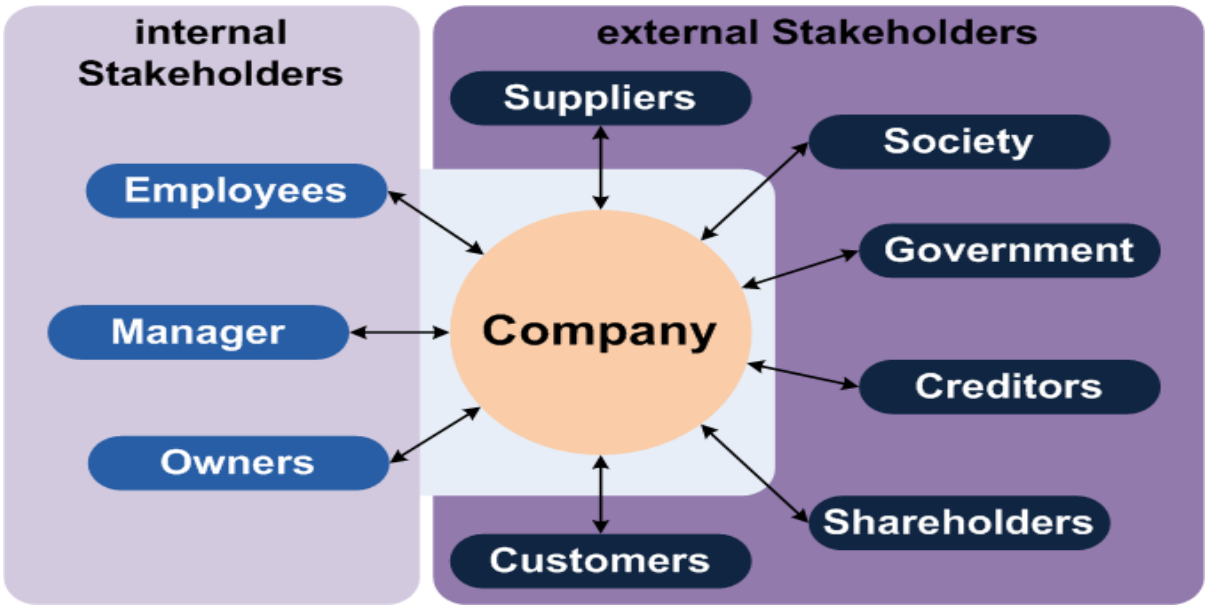


Fig 1.7 Stakeholders

1.8 Organisation of Thesis

The Thesis is organised in chapters. Every chapter is further divided in sections and subsections. Tables and Figures are provided wherever required and are properly referenced.

Chapter 1 is the introduction of the Thesis. In this chapter introduction of the area in which the Thesis is given is studied.

Chapter 2 is the literature survey of the Thesis. It describes various approaches and techniques for requirement prioritization. Two basic methods are given in the survey. These methods have further techniques and are given along.

Chapter 3 throws light on gaps in existing research and then briefly describes the problem statement. Objectives of the Thesis are mentioned. Then methodology used to fulfil those objectives is given. Every objective has a corresponding methodology .

Chapter4 demonstrates the implementation of the Lets_Tree_Prioritise tool in c#.net. Snapshots are given from the software to illustrate the functionality. Snapshots are properly indexed in list of Figures.

Chapter 5 shows the testing and the results obtained during implementation. Test case is the methodology used for the testing. Snapshots along with the test cases are drawn to show the behaviour of the system at that particular point.

Chapter 6 concludes the work done in the Thesis and also gives some future direction of the work.

Chapter 2

Literature Review

The purpose of this chapter is to review the existing techniques and methods of requirement prioritization.

2.1 Techniques of Requirements Prioritization

Several basic techniques on how to prioritize requirements are introduced in the following section.

2.1.1 Numerical Assignment Method

The Numerical assignment technique is based on the principle that each requirement is assigned a symbol representing the requirement's perceived importance.

2.1.1.1 Numerical Assignment Technique

Numerical assignment is one of the most common prioritization techniques, which also has been suggested in RFC 2119 [31].Numerical assignment is based on ordinal scale. In terms of granularity, this technique is medium and very easy in terms of sophistication [12].

In this technique, requirements are grouped together in different groups, which the stakeholders can relate to for trustworthy planning. Mostly three groups are used namely critical, standard and optional [12]. Use of relative terms like low, medium and high can create confusion for the stakeholders as they have different perceptions of such relative terms. Clear definition of each term and group can minimize the confusion [6].

A potential problem with the technique is with dividing the requirements into three groups (e.g. critical, standard, and optional). Since stakeholders tend to think everything is crucial, therefore they might put majority of the requirements in the critical group [32], which may lead to majority of requirements to be implemented. This will also eventually result in losing the cost benefits and real benefit of prioritization is lost. A way out of this possible problem is to put restriction on the

number of requirements in each group. But this forces the stakeholders not to prioritize according to their own priorities [33].Thus a substitution has to be made in such a situation.

All the requirements in each group are at the same level of priority and this might be another problem with this technique. However by using e.g. Ranking or more sophisticated techniques like AHP or 100 $ test requirements in a single group can be prioritized.

It is a simple requirements prioritization technique based on grouping requirements into different priority groups. The number of priority groups can vary, but three is common. For example, requirements can be grouped as "critical", "standard", and "optional". The results of numerical assignment are on a nominal scale. All requirements contained in one priority group represent equal priority. No further information shows that one requirement is of higher or lower priority than another requirement within one priority group.

2.1.1.1MoScoW

MoScoW is a kind of numerical assignment and it is mentioned by DSDM Consortium (2009). MoScoW currently incorporates into the software development methodology DSDM (Dynamic Systems Development Method). The idea of MoScoW is that it groups all requirements into four priority groups "MUST have", "SHOULD have", "COULD have", and "WON'T have".

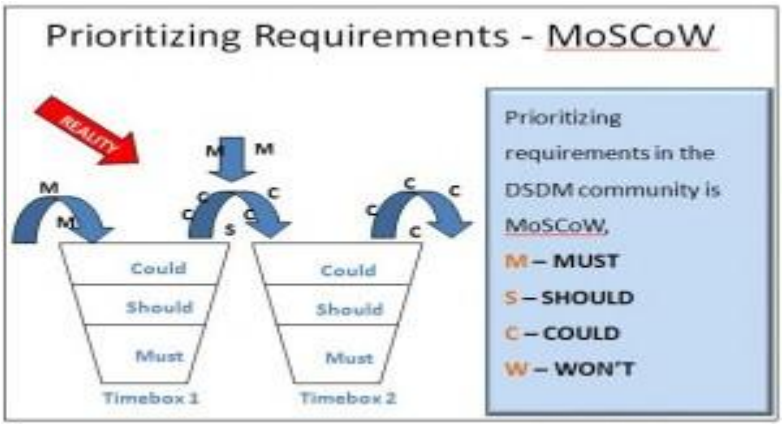


Fig 2.1 MoScoW Prioritization

• “MUST have” means that requirements in this group must be contained in the project. Failure to deliver these requirements means the entire project would be a failure.

• “SHOULD have” means that the project would be nice if it contains the requirements in this group.

• “COULD have” also means that the project would be nice if it contains these requirements. But these requirements are less important than the requirements in the “SHOULD have” group.

• “WON’T have” is like a “wish list”. It means that the requirements in this group are good requirements but they will not be implemented in the current stage. They may be implemented in the next release.

The results of MoScoW are on a nominal scale. All requirements contained in one priority group represent equal priority. No further information shows one requirement is of higher or lower priority than another requirement within one priority group.

2.1.2 Pairwise Comparison Technique

In this technique the candidate requirements are compared pair-wise to estimate their *relative* importance. The person thus has to decide which requirement is more important, and to what extent.

2.1.2.1 Simple Ranking

Based on medium in granularity and easy in terms of sophistication [12], requirements are prioritized without any ties in the rank. In this technique, the most important requirement is ranked 1 and the least important requirement is ranked as n (for n requirements). In contrast to Numerical Assignment each requirement has an exclusive rank; however relative ranking is not possible in this technique as in case of AHP and Cumulative Voting.

In order to obtain the ranks of the requirements, different algorithms like bubble sort, binary search tree [34], Quick sort and other sorting techniques [35] may be used.

This technique is a more suitable candidate when the priorities of a single stakeholder have to be prioritized, which is mostly the case in the bespoke development. However, in ranking, multiple aspects can be combined by taking the

mean priority of each requirement. This might result in ties in the requirements that this method wants to avoid [12]. This can be seen as one of the shortcomings of this technique.

Ranking elements is quite intuitive for most people as it can happen in people's lives. Berander and Andrews (2005) and Hatton (2008) mention simple ranking is that n requirements are simply ranked from 1…n, with the most important requirement ranked 1 and the least important requirement ranked n. This is a common requirements prioritization technique based on an ordinal scale.

2.1.2.2 Bubble Sort

Bubble sort is a method for sorting elements. Karlsson et al. (1998) introduce this technique to the requirements prioritization area for ranking requirements. The idea of the bubble sort method for sorting requirements is that the users compare two requirements at a time and swap them if the two requirements are in the wrong order. The comparisons continue until no more swaps are needed. The result of bubble sort is a list of ranked requirements. The average and worst case complexity for bubble sort is O (n^2) [34].

2.1.2.3 Analytic Hierarchy Process

Analytic Hierarchy Process is a systematic statistical technique based on relative assessment that has been used to prioritize software requirements in software community [28]. The AHP is a powerful and flexible decision making process to help people set priorities and make the best decision when both qualitative and quantitative aspects of a decision need to be considered [36]. By reducing complex decisions to a series of one-on-one comparisons, AHP helps decision makers arrive at the best decision. With AHP, one can synthesize the results, which provide a clear rationale for choosing the candidate requirements. It is very complex in terms of sophistication and fine in terms of granularity [12].

During the process, considering *n* requirements, $n \times (n - 1) / 2$ comparisons are to be made at each hierarchy level. This is often seen as a draw back in this process because with the increase in the number of requirements, the number of comparisons increases with magnitude of O (n^2) [35].

AHP can be used to prioritize requirements on the basis of different aspects and

there have been number of studies which have reported the use of it in the industrial setting and real projects [37], [38] as an efficient technique. In contrast to that, there are studies as well which have reported AHP as not being efficient and more difficult to use [20]. In another study by Karlsson, AHP was reported more time consuming and difficult to use in certain situations considering aspects of cost and value [21]. Therefore there is a need for more experimentation and industrial case studies to actually come to a final conclusion for its effectiveness under different situations.

There have been previous studies that have tried to minimize the comparisons [38], but by reducing the comparisons, there is a risk of inclusion of judgment errors. In Analytic Hierarchy Process, all comparisons should be made directly or indirectly, thus all requirements are considered. However, all requirements might not be compared with all other requirements in pair-wise comparisons. Therefore there is always a need of a trade-off between which comparisons to leave and which to perform. Requirements can also be accessed directly against particular criteria rather than relative to each other. Using such absolute measurement has a number of distinct advantages over the relative assessment. One – fewer assessments are needed i.e. one for each requirements against each criterion. Another – it avoids the need to compare unrelated requirements [39].

The idea of AHP is that it compares all possible pairs of hierarchical requirements to determine the priority. When using AHP, the user first identifies the attributes and alternatives for each requirement and uses them to build a hierarchy. Then the user specifies his/her preference to each pair of the attributes by assigning a preference scale which is generally 1 to 9, where 1 indicates equal value and 9 indicates extreme value. The scale is shown in Table 3. After that AHP converts the user's evaluations to numerical values and a numerical priority is derived for each element of the hierarchy. Note that a redundancy might exist when using the AHP method to prioritize requirements, therefore a consistency ratio should be calculated after using the AHP method to judge if the prioritization is valid. If n requirements need to be prioritized, n*(n-1)/2 pair-wise comparisons are required when using the AHP method. Therefore the complexity of AHP is O (n^2).

Table 2.1 Fundamental Scale used for AHP

Sr no	How important	Description
1	1	Equal Importance
2	3	Moderate difference in importance
3	5	Essential difference in importance
4	7	Major difference in importance
5	9	Extreme difference in importance
6	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.

Empirical studies performed by Karlsson and Ryan (1997) and Karlsson et al. (1998) show that AHP is time consuming. Some techniques try to reduce the number of comparisons in order to reduce the time consumed. Hierarchy AHP and minimal spanning tree have been developed for that purpose.

2.1.2.4 Hierarchy *AHP*

In large projects, requirements are often structured in a hierarchy, with the generalized requirements placed at the top of the hierarchy and the more specific requirements placed at the lower levels of the hierarchy. Hierarchy AHP, which is introduced by Karlsson et al. (1998), uses the AHP method to prioritize requirements only at the same level of hierarchy. This method can reduce the number of decisions compared with the AHP method, since not all the requirements are compared pair-wise. This can reduce the number of redundant comparisons, but the trade-off is that the ability to identify inconsistent judgments is also reduced [34].

2.1.2.5 Minimal Spanning Tree

Minimal spanning tree is another prioritization method which is introduced by Karlsson et al. (1998). The idea of minimal spanning tree method is that if the decisions are made perfectly consistent, the redundancy will not exist, and in this case the number of

comparisons will reduce to only n-1 comparisons (n is the number of requirements). A minimal spanning tree constructs unique pairs of requirements. It is a directed graph which is minimally connected. Minimal spanning tree can reduce the number of pair-wise comparisons dramatically compared with AHP. However, the ability to identify inconsistent judgments is low [34].

2.1.2.6 Cost-Value Approach

Karlsson and Ryan (1997) provide a method which is called the Cost-Value approach for prioritizing requirements. The basic idea of the Cost-Value approach is that each individual requirement is determined on two aspects: the value to the users and the cost of implementing the requirement. It uses the AHP technique to compare requirements pair-wise according to the relative values and costs. Empirical studies performed by Karlsson and Ryan (1997) show that the Cost-Value approach is time consuming.

2.1.3 Mixed Methods

Some other methods of requirement prioritization that do not fall in the category of numerical assignment or the pairwise comparison technique are as follows:

2.1.3.1Binary search tree

Another method for sorting elements is binary search tree. A binary search tree is a tree in which each node contains at most two children. Karlsson (1998) introduced this technique to the requirements prioritization area for ranking requirements. The idea of the binary search tree method for ranking requirements is that each node represents a requirement, all requirements placed in the left sub tree of a node are of lower priority than the node priority, and all requirements placed in the right sub tree of a node are of higher priority than that node priority. When performing the binary search tree method, first choose one requirement to be the top node. Then, select one unsorted requirement to compare with the top node. If that requirement is of lower priority than the top node, it searches the left sub tree, but if that requirement is of higher priority than the top node, it searches the right sub tree. The process is repeated until no further node needs to be compared and at that time the requirement can be inserted into the right position. The average complexity for binary search tree is O (n log n). Simple ranking, bubble sort and binary search tree methods are all used for ranking requirements [38, 34].

2.1.3.2 BTree

Btree approach is a systematic way in which the number of comparison required by can be kept low. The Btree prioritization technique proposed by Md.Rizwan Beg [40] incorporates certain features like dynamic incoming requirements i.e. the requirements that never solidify and it also handles if certain requirements are dropped during runtime. Btree Prioritization uses the similar concept as that of Binary tree but it is more balanced and provides certain other advantages over the Binary tree prioritization .It provides the run time capability i.e. if there are few hundreds of requirements waiting till they are finalized is not required and prioritization can be started with as many requirements available at the moment. It also gives the control to keep the number of comparison fixed for prioritizing a requirement if the total number of requirements is known in advance. Another major advantage of Btree Prioritization is that the result of this technique is more presentable then other techniques, even if they are not finalized yet.

2.1.3.3 Planning Game

Beck (1999) introduces a prioritization method, named Planning Game, which is based on a combination of prioritization techniques. Planning Game is mostly used in agile projects. The idea of Planning Game is that it combines the numerical assignment technique and ranking technique together to perform the requirements prioritization. Requirements are first prioritized into three groups:

(1) Those without which the system will not function,

(2) Those that are less essential but provide significant business value and

(3) Those that would be nice to have. After assigning the requirements into three groups, requirements are simply ranked in each group [19].

2.1.3.4 Hundred Dollar Method

Hundred dollar method (also called cumulative voting) is a simple method for prioritizing requirements. The idea of the hundred dollar method is that each stakeholder is asked to assume he/she has $100 to distribute to the requirements. The result is presented on a ratio scale. The ratio scale result can provide the information on how much one requirement is more/less important than another one. The cumulative voting or the 100-dollar test is a straightforward technique in which the stakeholders are provided 100

imaginary units. This technique is based on the ratio scale. This prioritization technique is complex in terms of sophistication and fine in terms of granularity [12]. These imaginary units could be different aspects (e.g. money cost of implementation, importance, penalty, hours etc).The imaginary units are then distributed among the requirements and the sum total at the end should be hundred [33]. However this also poses problems if the number of requirements is greater than hundred. Also, sometimes during the prioritization process, for large number of requirements, the person doing the prioritization can miscalculate, and the sum may come greater or less than hundred (Berander and Wohlin 2004). However use of automated tools, which can keep the count, can avoid this situation.

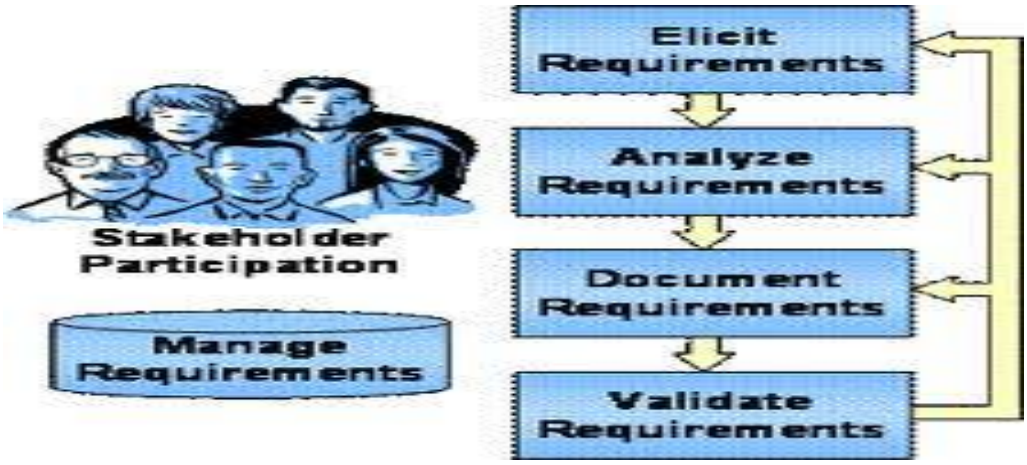


Fig 2.2: Cumulative Voting

Also during a study conducted by Regnell et al. they faced the problem of distributing hundred points between 17 set of requirements [28]. Since the number of requirements was large, so they used a fictitious amount of $ 100,000. Therefore this study showed that amount other than 100 can be used for larger number of requirements and participants of the study were also positive about the technique.

One shortcoming of this technique is that stakeholders might put all their units on their one or more favorite requirements which other stakeholders do not prioritize as highly, thus biasing the prioritization process. This can be avoided by limiting the

number of units to be put on a single requirement. However, this results in forcing the stakeholders not to prioritize according to their own priorities [33].

Also, prioritization should be done once on the same set of requirements because stakeholders might bias their evaluation during the second time if they do not get their favorite requirement during the first round [12].

2.1.4 Comparison between the Techniques

Table 2.1 illustrates the comparison between the techniques on the base of certain parameters that are scale

- Fault tolerance
- Granularity
- Complexity
- Time consumed

Table 2.2 Comparison between Different Techniques

Prioritization method	**Scale**	**Fault Tolerance Ordinal Scale (1-7)**	**Granularity**	**Complexity**	**Time Consumed Ordinal Scale (1-7)**
Analytical hierarchy process	Ratio	1	Fine	Very complex	7
Numerical assignment	Ratio	5	Coarse	Very easy	1
Value oriented prioritization	Ratio	2	Medium	Complex	2
Cumulative voting	Ordinal	3	Fine	Complex	3
Binary Search	Ordinal	4	Medium	Easy	6
Planning Game	Ordinal	6	Coarse	Easy	5
Btree prioritization	Ordinal	7	Medium	Complex	4

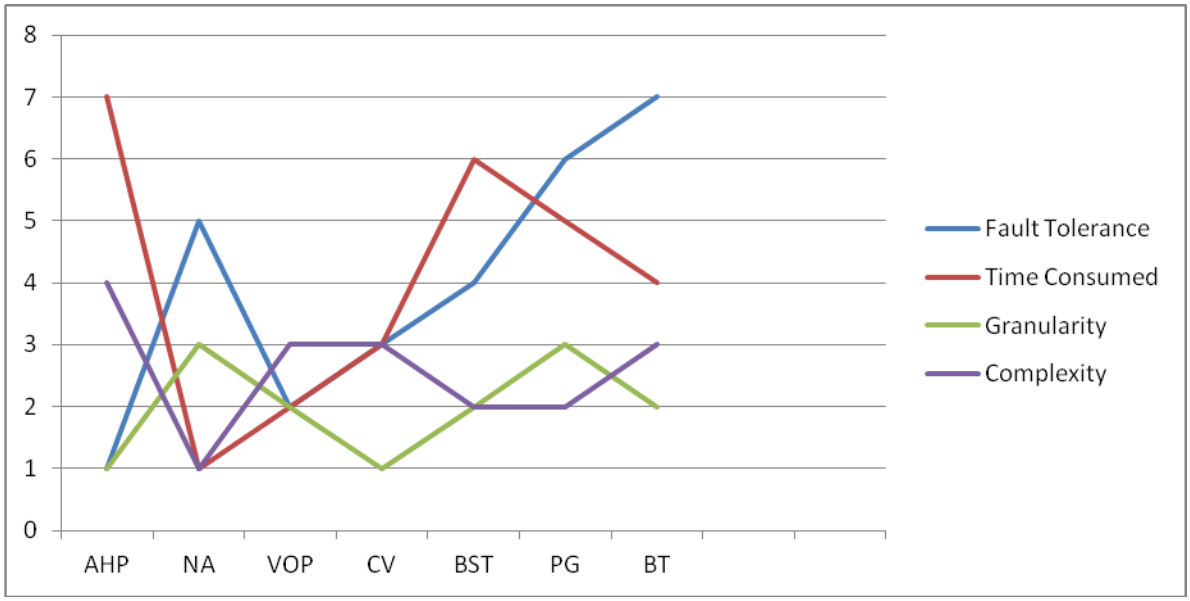


Fig 2.3 Comparison between the prioritization techniques based on certain parameters.

Jung won park in 1999 [41] proposed a tool DCPT i.e. Distributed collaboration and prioritization tool. This tool was developed to support the requirement prioritization when the stakeholders are at distant locations. This model and hence tool collaborates two techniques that are

- Bin Model
- Voting Model
- Cocomo II model

Jim Azar in 2007[42] gave a method called value oriented prioritization that was designed for small development organisations. Value oriented prioritization technique takes in consideration certain parameters. These are the values that an organisation holds its dearest. These parameters can be different for every organisation.

The risks are also calculated. Two major risks associated with the project are as follows:

- Business Risk: Risk associated with profit from that particular project. It is anticipated based on certain factors like competition, sales volume, climate and government factors.
- Technical Risk: Risk associated with understanding and employing the knowledge about the project by the developer.

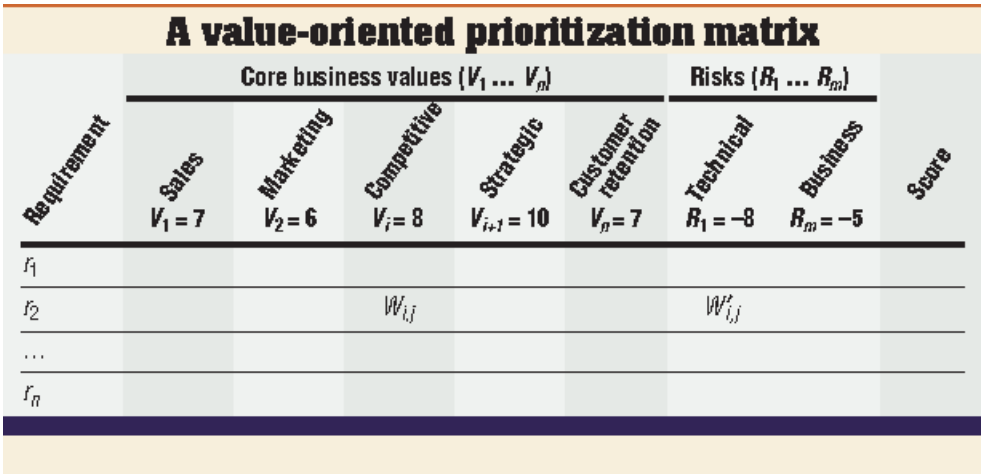


Fig 2.4 Value Oriented Prioritization matrix [42]

Certain organisations can be more inclined toward Customer Retention while others are towards business. For each requirement its corresponding value is calculated and the score is generated. The highest score requirement is implemented with highest priority.

The other factors are:

Sales, Marketing, Competitive, strategic, Technical, Business

Chapter 3

Proposed Requirement Prioritization Tool

The purpose of this chapter is to study the problem statement of the Thesis. The chapter also gives an insight of objectives and the methodology used**.** It will provide the insight of the research gaps that have been observed during literature review.

3.1 Research Gaps

During Literature review the following research gaps have been found:

I. Cumulative voting (100 Dollar method) has not been implemented and automated.
II. Btree Method has not been implemented and automated.
III. Most of the methods are made for stakeholders present at same geographical area. There is a need to develop methods for geographically distributed stakeholders.
IV. There is a need to improve effectiveness of cumulative voting.
V. No empirical studies have been done on these methods.
VI. Test of effectiveness (precision) of the methods of prioritization is not done.

3.2 Problem Formulation

There are various techniques for requirement prioritization based on parameters like size of project, number of stakeholders, Geographical distribution of customers etc. Every parameter call for a different requirement prioritization technique. But none of the techniques are automated and therefore certain discrepancies like wastage of time and paper occurs. Human errors are also encountered in manual systems. Moreover the arrangement of the prioritized requirements is difficult in manual systems. The addition and removal of requirement as per need is not possible in manual systems. To remove such discrepancies the prioritization methods can be automated so that more accurate results can be obtained. Moreover a GUI is a better way to present and implement.

3.3 Objectives of Thesis

I. To study the existing requirement prioritization techniques.

II. To propose a tool based on previous paper.

III. To implement the proposed tool.

IV. To test and validate the implemented tool.

3.4 Methodology

I. Existing requirement prioritization techniques have been studied in literature review. During literature review certain gaps were found and few of them are implemented by the tool.

II. A tool has been proposed to remove some research gaps. A tool is developed named Lets_Tree_Prioritise in c#.net and sql server.

III. Implementation process is as follows

   a) Collect the requirements from the stakeholders and assign their priority values according to the values given by the stakeholder.

   b) Prioritize the requirements and the priority values given by the stakeholders using Btree.

IV. Testing on the tool has been done. Test cases are performed the tool for various values and results have been observed.

Chapter 4

Design and Implementation Details

This chapter includes the implementation details for the requirement prioritization using Btree. C#.net and SQL server is used for implementation. It integrates computation, visualisation and programming in easy to use environment. As discussed in literature Review there are a lot of methods for requirement prioritisation having their own merits and demerits. In our implementation we have used Btree Prioritization for requirement prioritization.

4.1 Features of the Lets_Tree_Prioriritise Tool

The main features of the proposed tool are:

I. It contains a dedicated database for storage of all the previous entered software projects.

II. It contains an interface, which provides various techniques to enter the requirements.

III. The interface provides a Cumulative voting technique in which the user can enter the value of each requirement for multiple requirements and calculate the result for whole lot of requirements.

IV. The tool allows a direct method to enter the requirements with proper range of values.

V. The system implements a very effective sorting technique Btree to sort the requirements according to the priority of the requirement.

VI. The implementation also allows updation, addition and deletion of requirements and their respective values.

4.2 UML Specification: Data Flow Diagram

The Steps acquired in the data flow diagrams

I. Acquire Requirements: In this step the Requirements are collected and entered in the tool by the data administrator.

II. Choose a method: Appropriate method is choosed that can be Simple Ranking or Cumulative Voting (100 Dollar method).

III. Collecting and Measuring and requirements: The requirements are calculated as per the method choosed.

IV. Prioritizing requirements: Requirements are modified if required and prioritized thereafter.

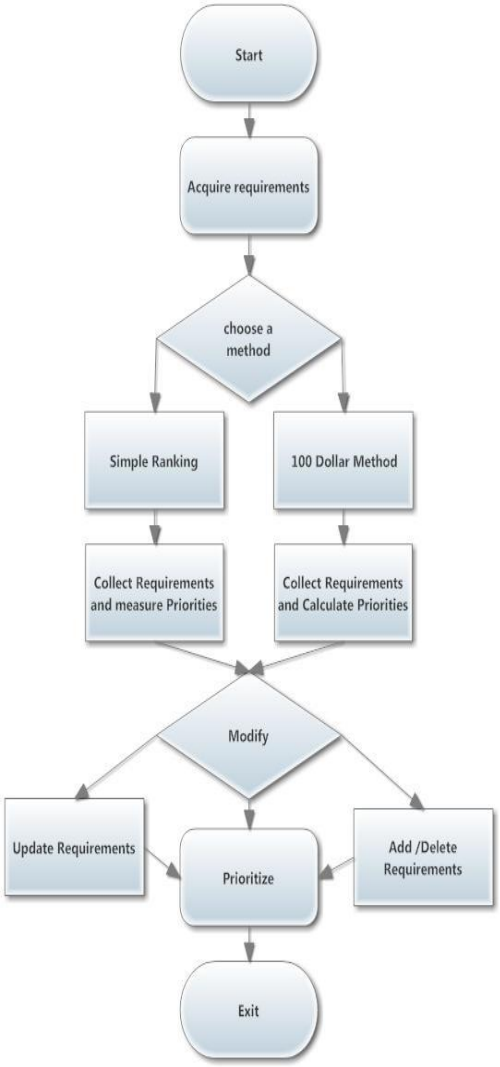


Fig 4.1: Data Flow Diagram of the Project

4.3 Working of the Implemented Tool

Lets_Tree_Prioritise is a software tool, the working and functionalities of its various units are described in this section.

4.3.1 User Authentication

The first step for any user is to authenticate herself so that he may be given the access rights to not only to view but also to enter and edit the requirement. User name and password is the minimum requirement to enter and use the tool. The authentication screen is given below that takes input the admin name and password and let the admin enter the tool.

Fig 4.2: Snapshot of User Authentication

4.3.2 New Project

The new project unit allows user to choose a method from two methods Direct or 100 Dollar Method to enter the name of the project and requirements with their respective values.

Fig 4.3: Snapshot of New Project Module

4.3.3 Direct method

Direct method allows the admin to enter the name of requirements and there priority value directly. Any number of requirements can be entered but the values can be between specified ranges.

4.3.3.1 New Project Unit

New Project unit will ask inquire the user the name of the new project that is to be created. It will check if the project already exists. If the same project exists it will notify the user and will ask for a new name for the project.

Fig 4.4: Snapshot of New Project

4.3.3.2 Enter the Number of Requirements

In this Text Box the number of requirements for a particular project is entered.

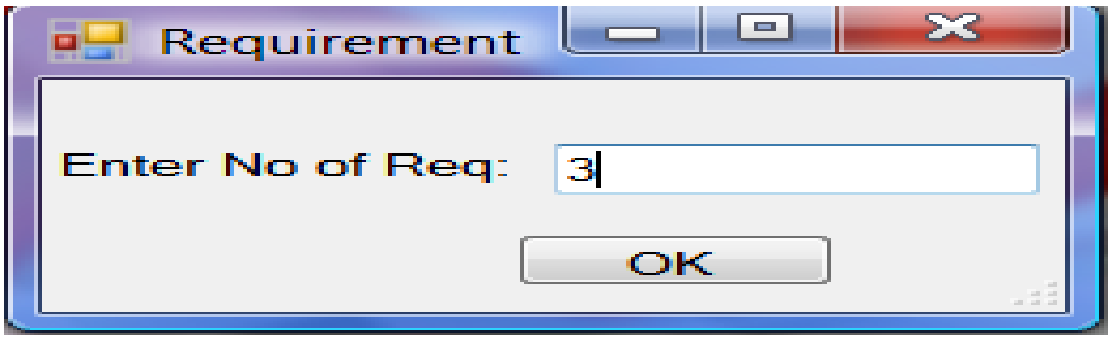


Fig 4.5: Snapshot of Number of Requirements

4.3.3.3 Enter the Name and Priority Value of the Requirement.

In this form the appropriate name and priority value is filled. This priority value can be based on any criteria inside an organisation where

i. Stakeholders are not the major issue.

ii. Some parameters are set for prioritizing requirements.

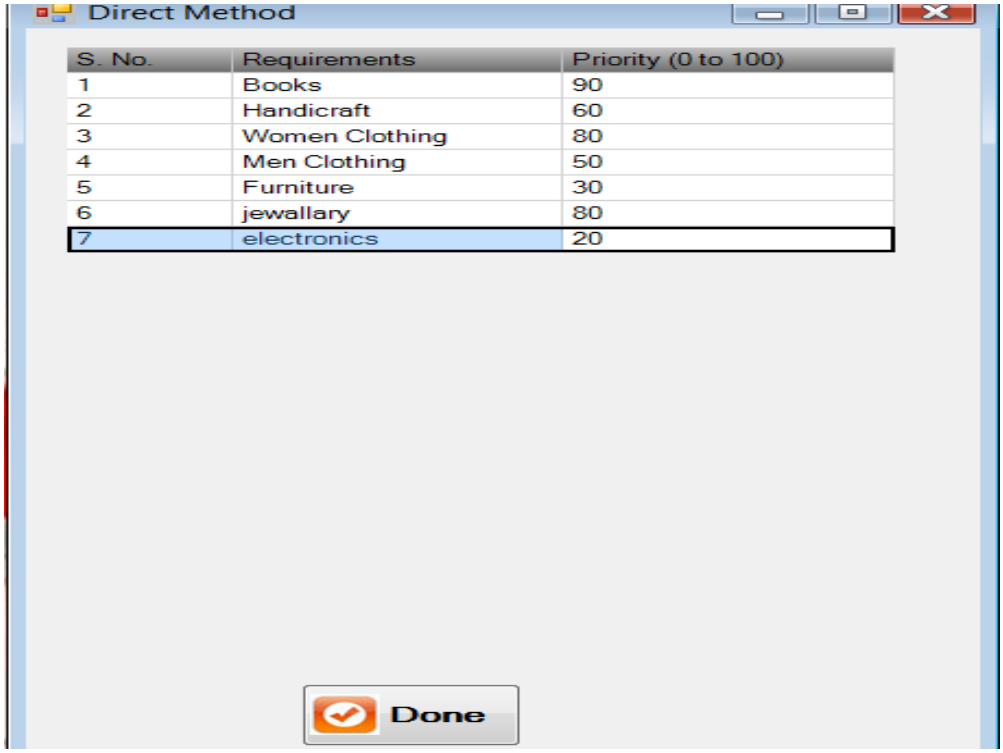


Fig 4.6: Snapshot of Data Entry of Requirement and the Priority Value

4.3.3.4 Display Data

Display data shows the name of the requirements along with the priority values. The comparison of the priority values of the component is also displayed to give a pictorial representation of the requirements. Requirements are on X axis of the comparison chart and the priorities are on the Y axis.

Display form also allows the following functions:

- Edit : edit will allow user to edit the name Of the Requirement or the Priority value of an already existing Requirement.
- Delete: Tool will ask user to select the Requirement and then will delete the component.
- Add : Add will prompt the user the name of the Requirement and the priority value and will add the same in the requirement database.

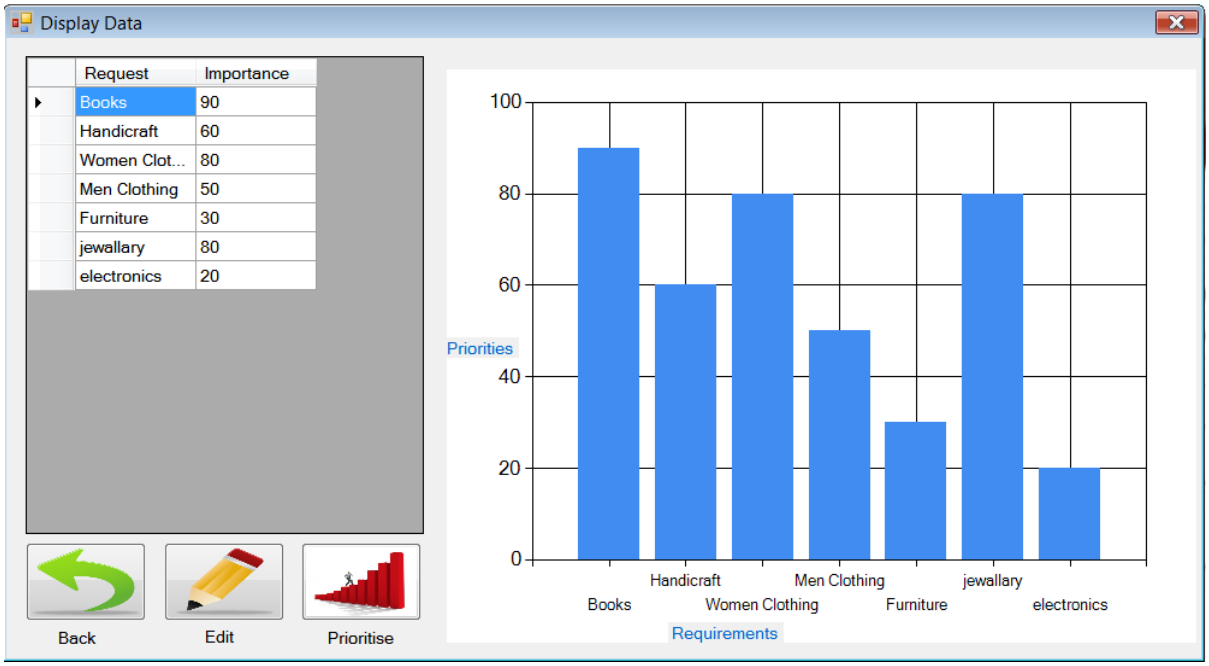


Fig 4.7: Snapshot of Display data

4.3.3.5 Edit, Delete or Add the Requirement

This form allows the user to edit, add or delete the requirement and priority.

Edit: to edit the name and the value of the requirement.

- Tool prompts to select a requirement.
- Selected requirement name and value can be changed.
- It will be updated.

Add: to add a Requirement and a priority value.

- Name and corresponding value of the requirement can be added.

Delete: to delete the requirement.

- Tool prompts to select a requirement.
- Selected requirement name and value can be deleted.
- It will be updated.

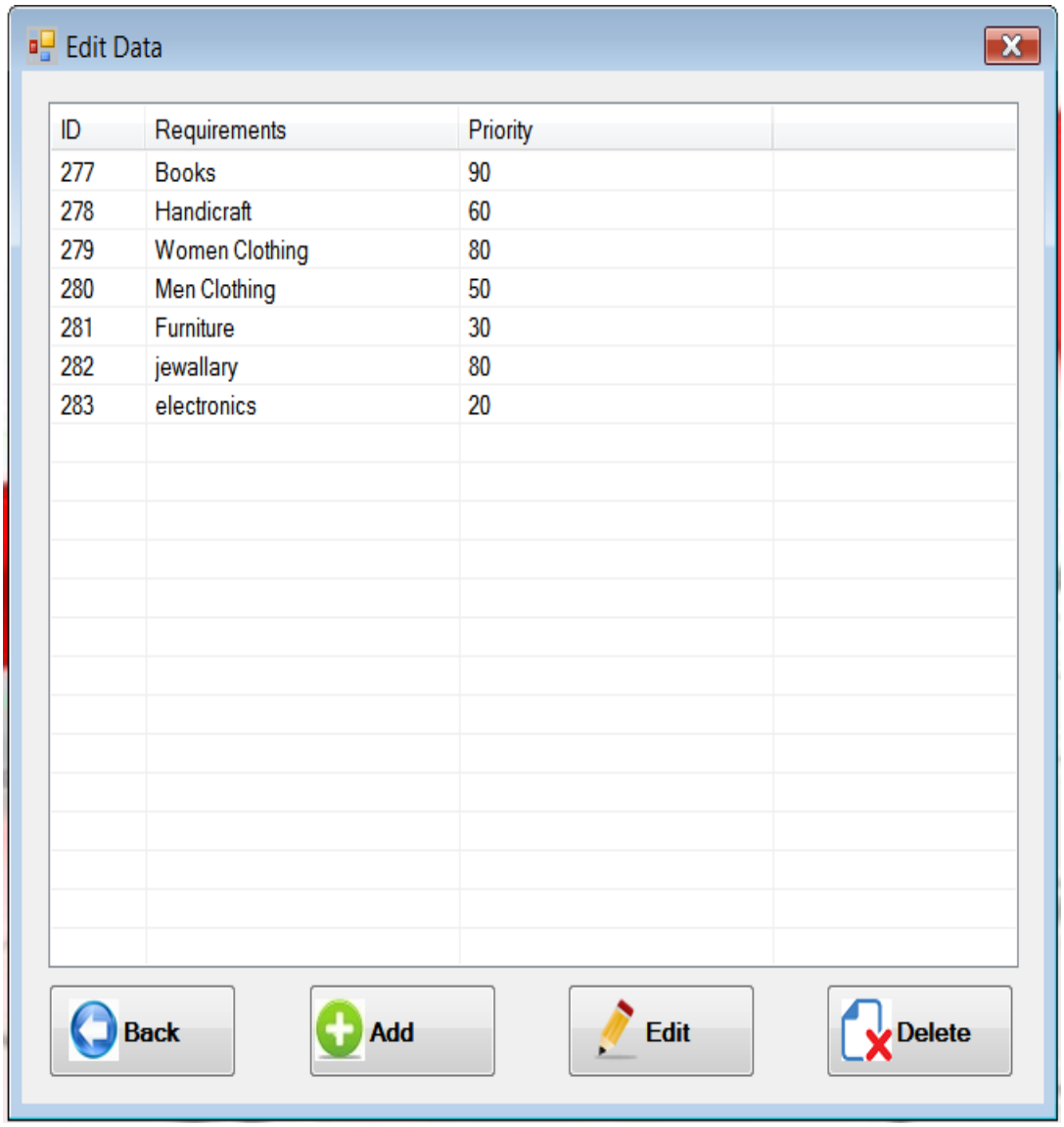


Fig 4.8: Snapshot of Edit, Add or Delete Data.

4.3.3.6 Prioritise the Requirements

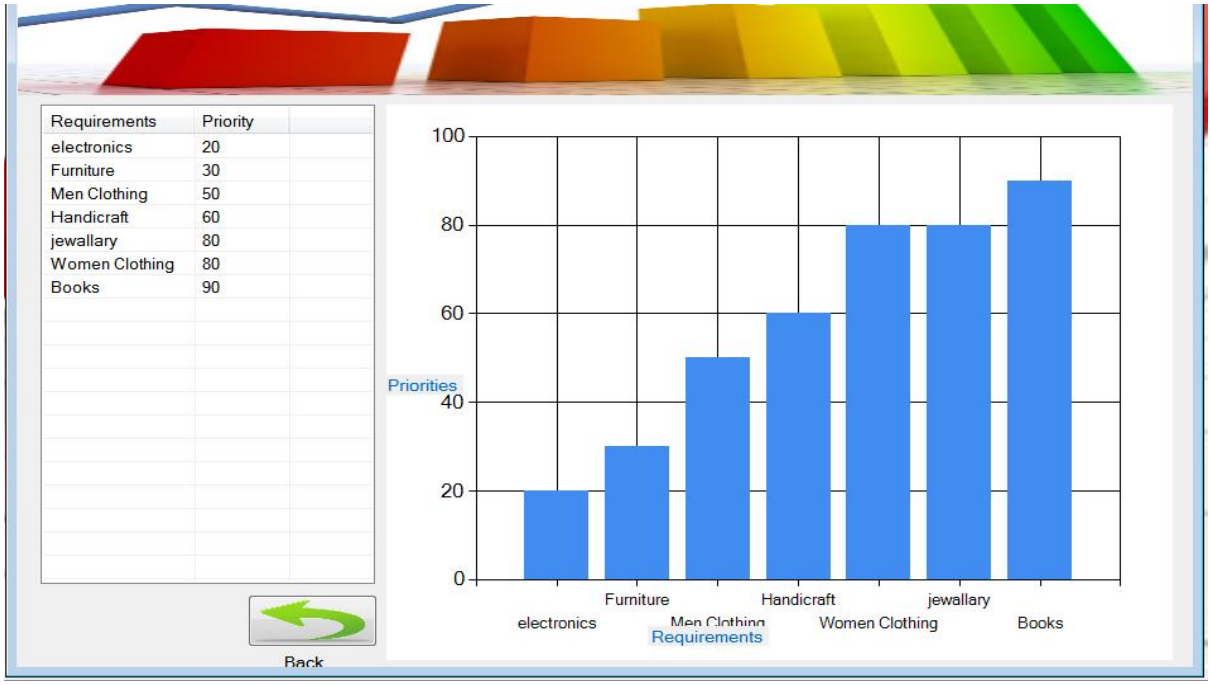


Fig 4.9: Snapshot of Prioritised Requirements

4.3.4 100 Dollar Method: The 100-dollar test is a technique where the stakeholders are given 100 imaginary units to distribute between the requirements.

Fig 4.10: Snapshot of Selecting 100 Dollar Method.

4.3.4.1 Selecting the Number of Requirements and Number of Stakeholders:

The Tool prompts the user to fill the number of requirements and number of stakeholders associated with the project.

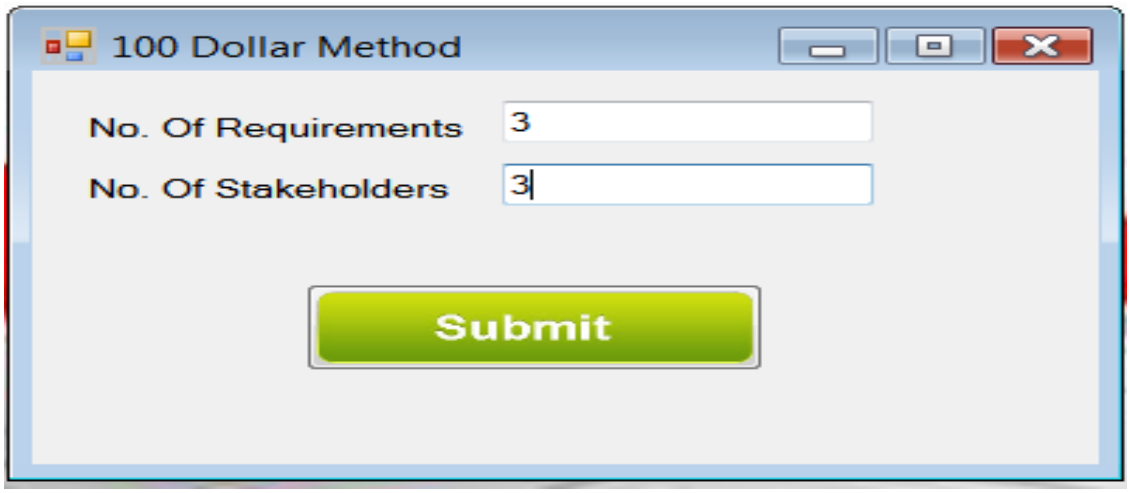


Fig 4.11: Snapshot of data entry

4.3.4.2 Name of the Requirements:

This form prompts the user to enter the details of the requirements.

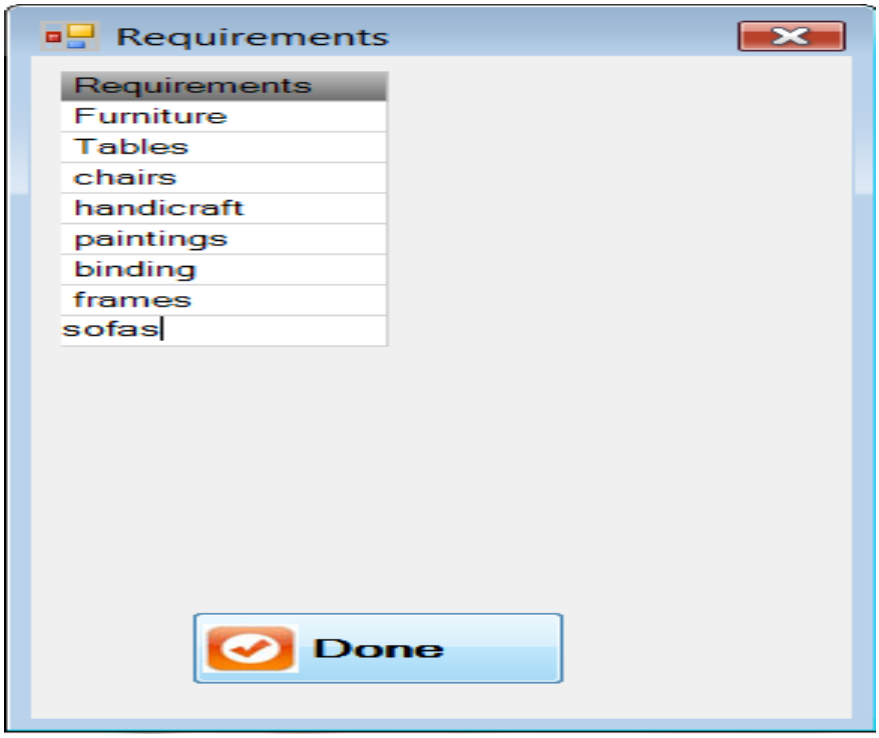


Fig 4.12: Snapshot of the Requirements Form

4.3.4.3 Stakeholders Form

This form lets the admin fill the values given by the stakeholders for the respective requirement.

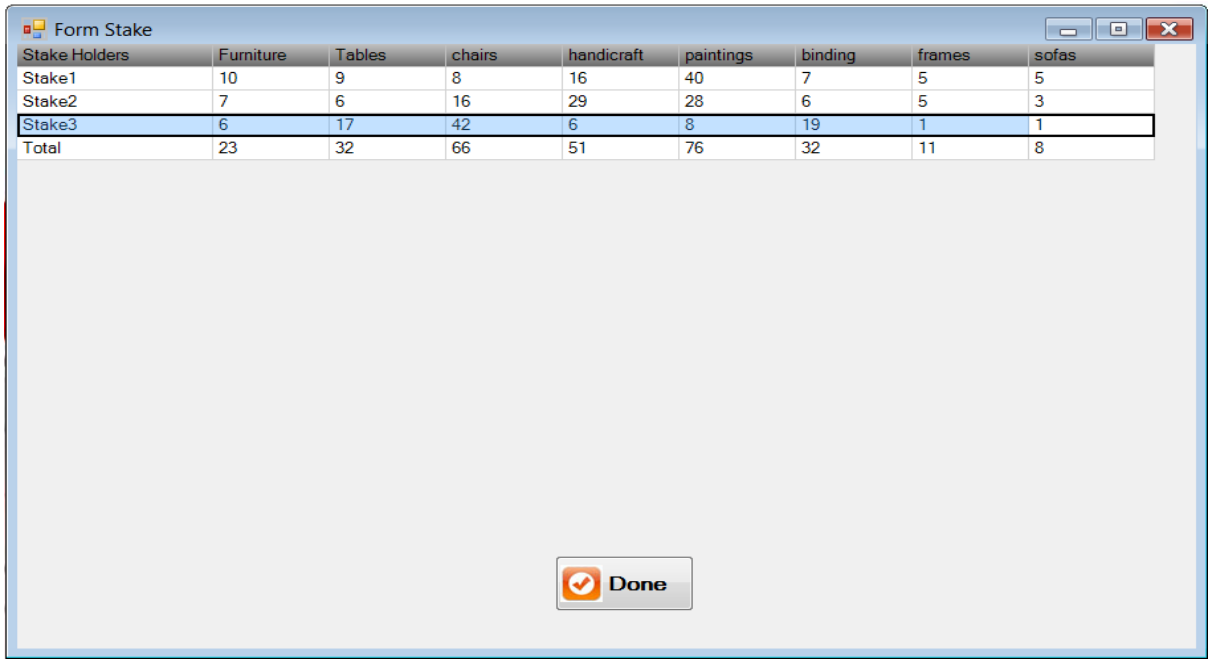


Fig 4.13: Snapshot of Stakeholders form

4.3.4.4 The Display Form:

It displays the requirement with the respective priority value.

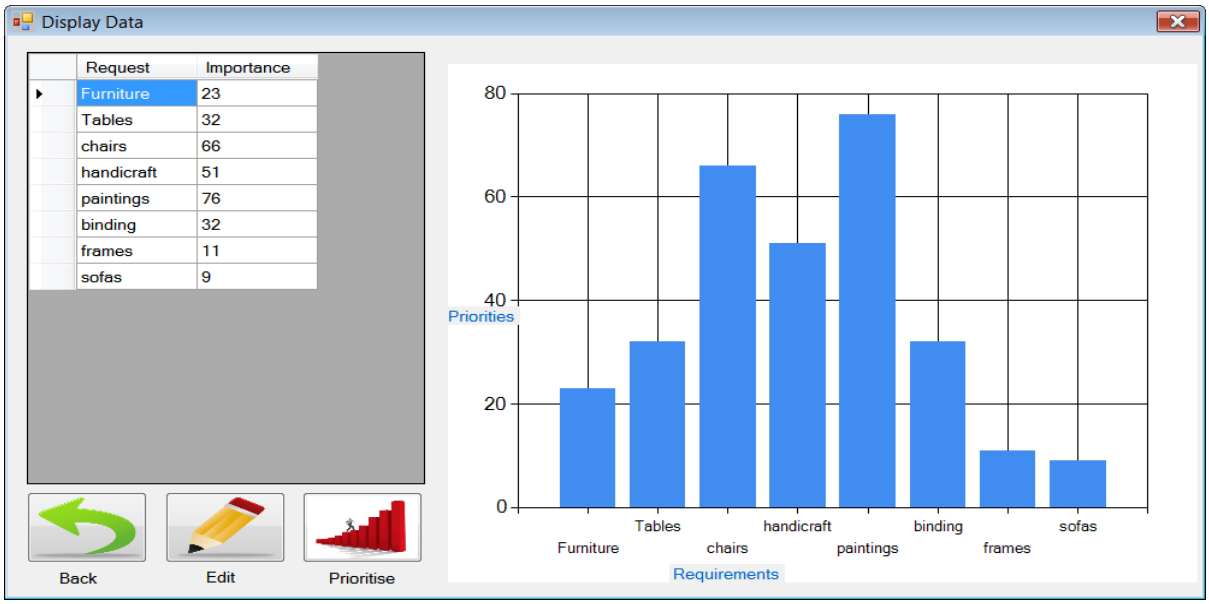


Fig 4.14: Snapshot of Display data

4.3.4.5 Prioritised Data

The Final prioritised values are displayed.

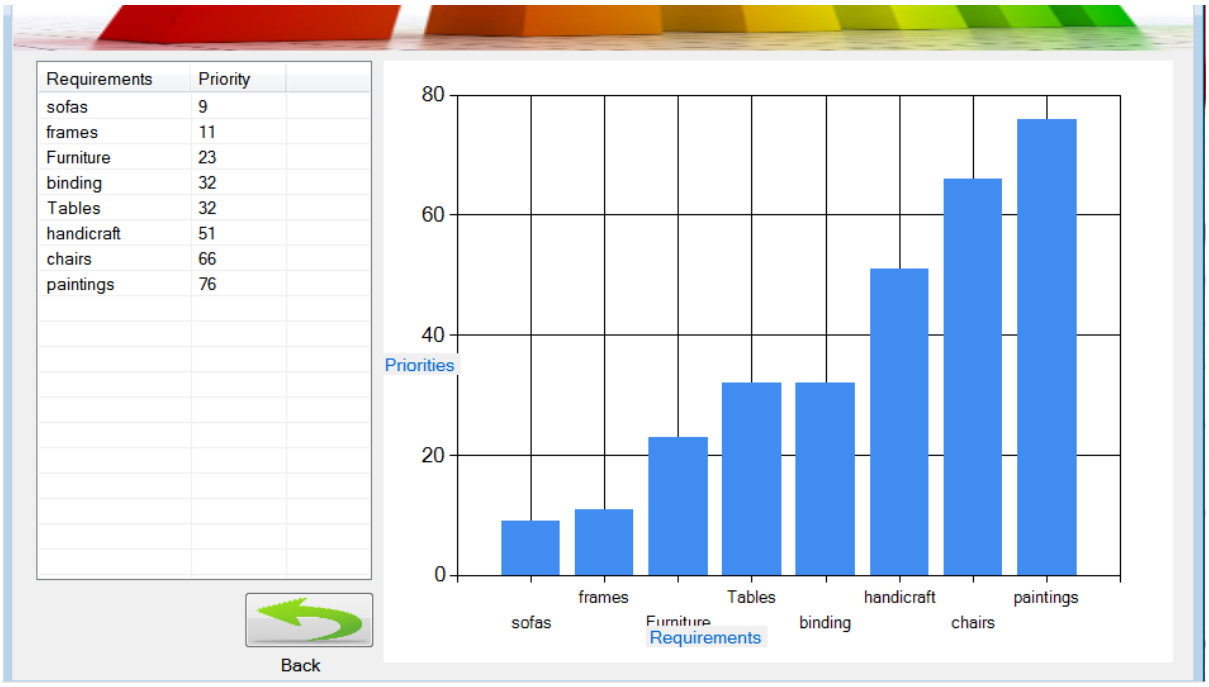


Fig 4.15: Snapshot of Prioritised Data

4.3.5 Open Project:

The Open Project screen displays the previously stored projects in the database.

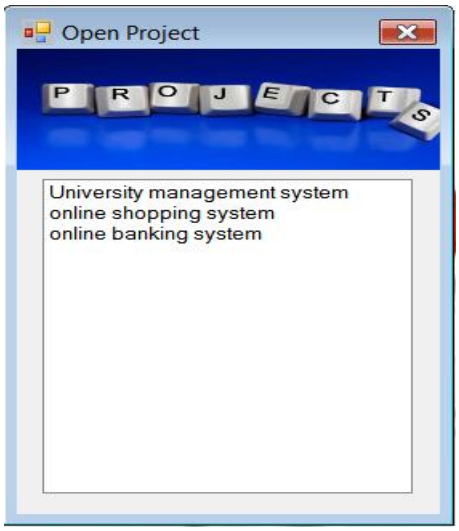


Fig 4.16: Snapshot of Open Project Module.

Chapter 5

Testing and Experimental Results

This chapter elaborates the types of testing done on the tool.

5.1 Testing

Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Although crucial to software quality and widely deployed by programmers and testers, software testing still remains an art, due to limited understanding of the principles of software. The difficulty in software testing stems from the complexity of software: we cannot completely test a program with moderate complexity. Testing is more than just debugging. The purpose of testing can be quality assurance, verification and validation, or reliability estimation. Testing can be used as a generic metric as well. Correctness testing and reliability testing are two major areas of testing. Software testing is a trade-off between budget, time and quality.

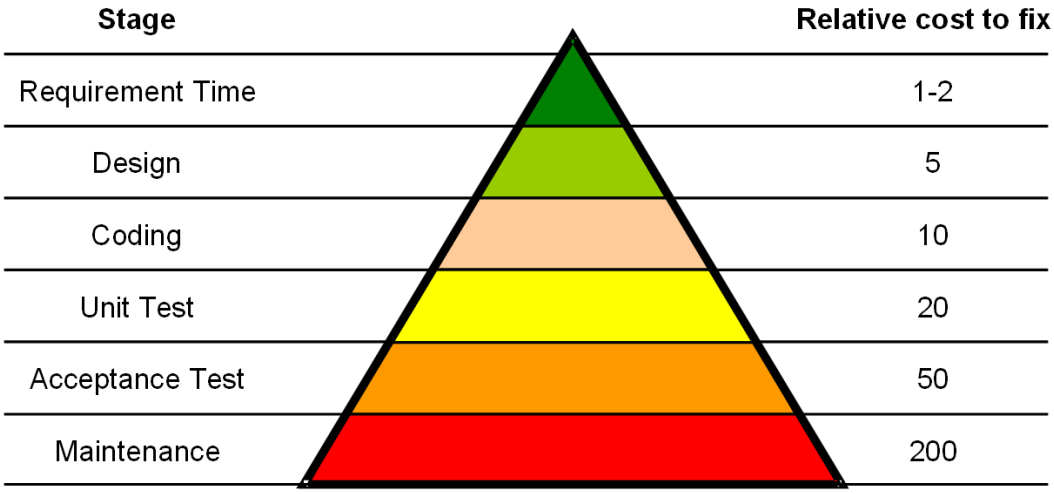


Fig 5.1: Cost of testing at every lifecycle phase

5.2 Test Cases

The test cases prepared and performed on the tool Lets_Tree_Prioritise are as follows:

Table 5.1 Test Cases of the Tool

Test ID	The Test Case	Input to be given	Expected output
1	Login Authentication	Username and Password	If Username and password are correct then Opens Btree Prioritization Page
2	Login Authentication	Username and Password	If username is correct and password is not correct then Display message “sorry !! You are not authenticated”[Fig5.2]
3	Login Authentication	Username and Password	If username is not correct and password is correct then Display message sorry !! You are not authenticated”
4	Login Authentication	Username and Password	If username is not filled and password is filled then Display message “please fill all the fields”[Fig 5.3]
5	Login Authentication	Username Password	If username is filled and password is not filled then Display message “please fill all the fields “and does not proceed.

Test ID	The Test Case	Input to be given	Expected output
6	Choose Direct method	Choosing direct method	New project dialog box opens[Fig 5.4]
7	Choose 100 Dollar method	Choose 100 dollar method	New project dialog box opens
8	Open Project	Choose open project	Open project form showing the Existing projects opens
9	Windows close all	Choose windows close all	Closes all the open forms
10	Windows arrange icons	Choose windows arrange icons	Allows to activate desired form from multiple forms
11	Exit	Choose Exit button	Closes the software
Test ID	**The Test Case**	**Input to be given**	**Expected output**
12	Entering the requirement and priorities in Direct method	Entering all the requirements in characters	Saves the requirement
13	Entering the requirement and priorities in Direct Method	Entering the requirements in character and integer	Display message "Invalid Requirement", deletes the numerical value and allows user to enter again.[Fig 5.5]
14	Entering the requirement and priorities in Direct Method	Entering the requirement in integers	Display message "Invalid Requirement"
15	Entering the requirement and priorities in Direct Method	Entering the priority value between 0-100	Saves the requirement and the corresponding priority value.

16	Entering the requirement and priorities in Direct Method	Entering the priority value more than 100	Display message "priority should be between 0 to 100"[Fig5.6]
17	Entering the requirement and priorities in Direct Method	Clicking Done without filling all the requirements	Display message " please fill all the Requirements" [Fig5.7]
18	Editing the requirement or Priority	Clicking the edit button	Display message "please select an item first"[Fig 5.8]
19	Editing the requirement or Priority	Clicking the delete button	Display message "please select an item first"
20	Editing the requirement or Priority	Choosing the value to delete and clicking delete	Display message " are you sure to delete Requirement having id=id of requirement"[Fig 5.9]
Test ID	**The Test Case**	**Input to be given**	**Expected output**
21	Entering the number of Requirement and the number of stakeholders.	Entering the Number of requirements in characters	Characters will not be entered.
22	Entering the number of Requirement and the number of stakeholders.	Entering the Number of Stakeholders in characters	Characters will not be entered
23	Entering the Requirement in 100 dollar method.	Entering characters and integers	Display message "Only alphabets are allowed.All didgits removed. You can

			make changes if you want to", deletes the numerical value and allows user to enter again [Fig 5.10]
24	Entering the Requirement in 100 dollar method.	Entering integers	Display message "Only alphabets are allowed.All didgits removed. You can make changes if you want to", deletes the numerical value and allows user to enter again. [Fig5.10]
25	Entering the Requirement in 100 dollar method.	Clicking enter without filling up all the requirements	Display message "please fill all the Requirements"and will not proceed further.[Fig 5.11]
26	Entering Priority value of the requirements in Stakeholders form	Entering characters in value of the requirement	Characters will be removed and will return 0 value
27	Entering Priority value of the requirements in Stakeholders form	Entering characters and integers in the value of the requirement	Characters will be deleted and only integer value will be left.
28	Entering Priority value of the requirements in Stakeholders form	Entering value exceeding a sum of 100	Display message "total exceeds 100" and calculates the remaining value of

			that requirement. [Fig 5.12]
29	Entering Priority value of the requirements in Stakeholders form	Clicking Done button without entering all the requirements	Display message "total of all the requirements should be 100"[Fig 5.13]
30	Editing the requirement or Priority	Clicking the edit button	Display message "please select an item first"
31	Editing the requirement or Priority	Clicking the delete button	Display message "please select an item first"
32	Editing the requirement or Priority	Choosing the value to delete and clicking delete	Display message " are you sure to delete Requirement having id=id of requirement"

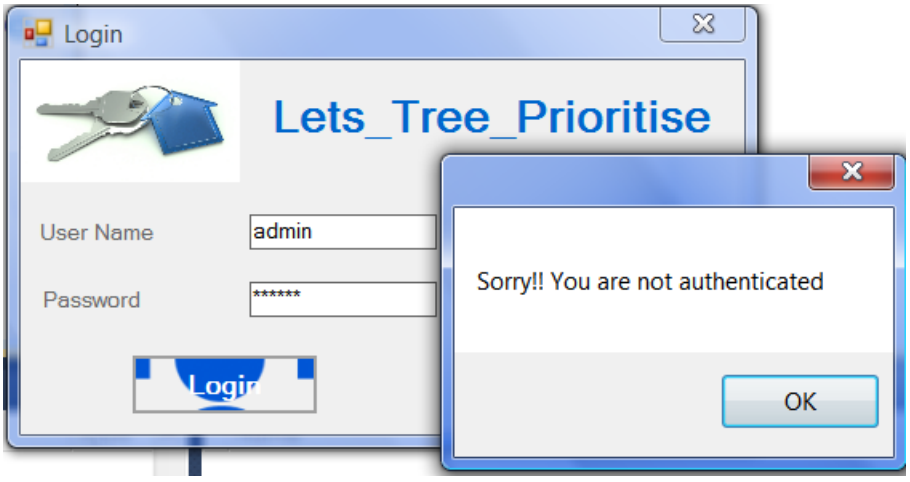


Fig 5.2 Login Authentication

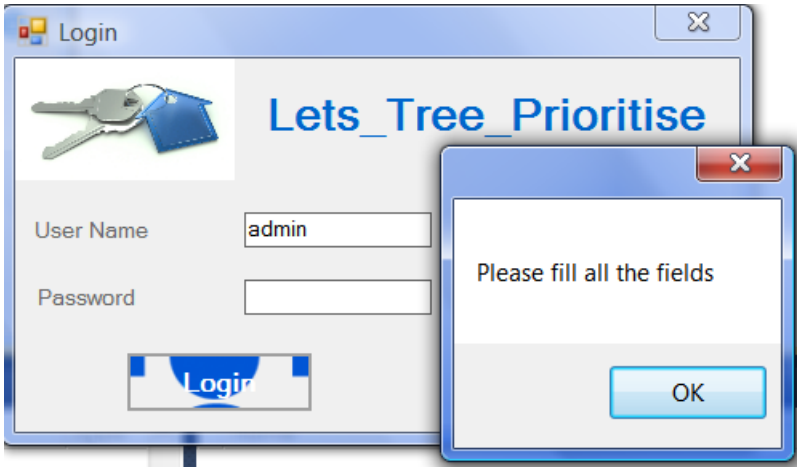


Fig 5.3 Login authentication

Figure 5.4 Main Form

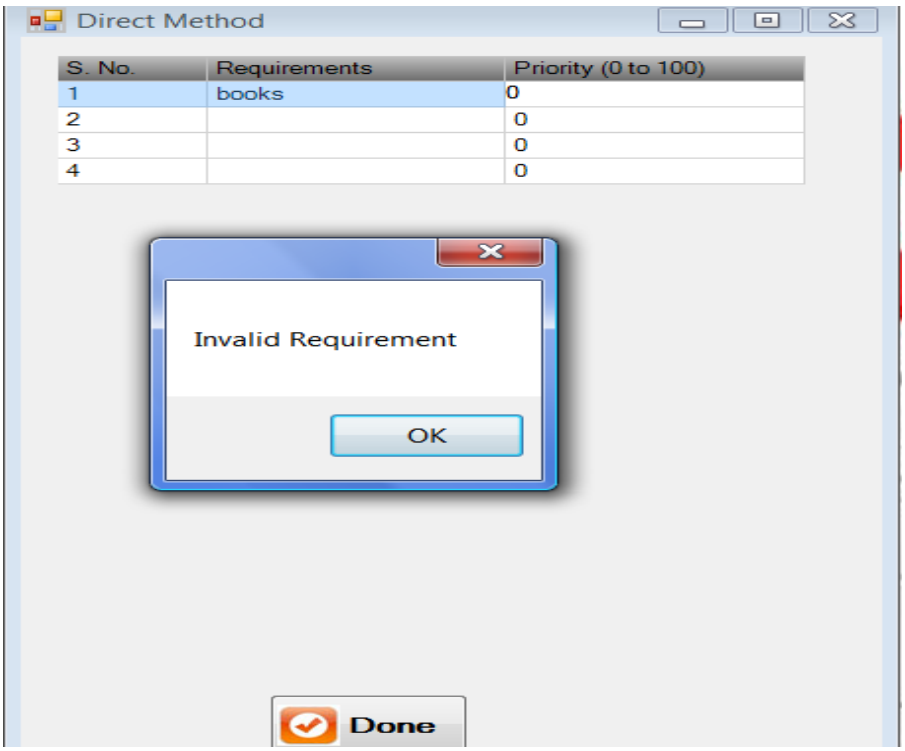


Fig 5.5: Invalid Requirement

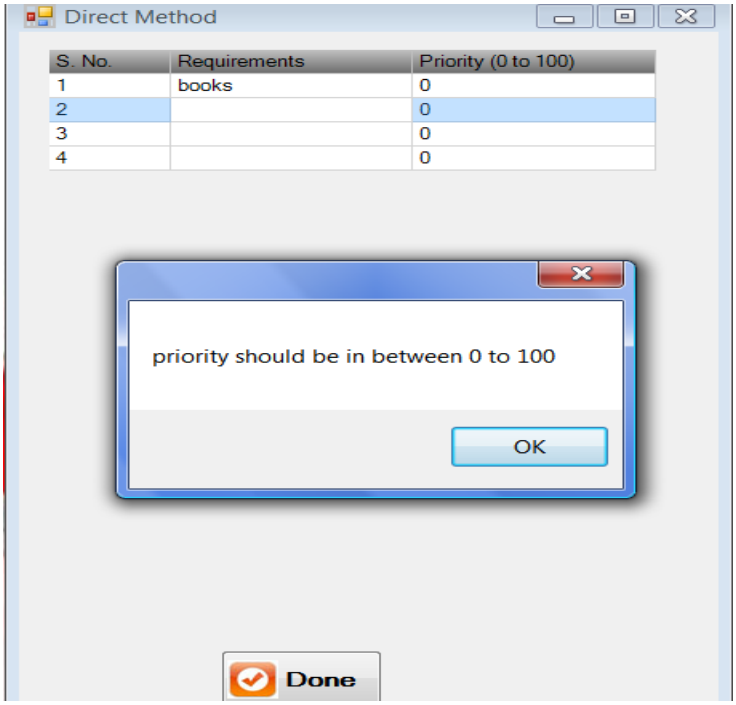


Fig 5.6: Priority Range

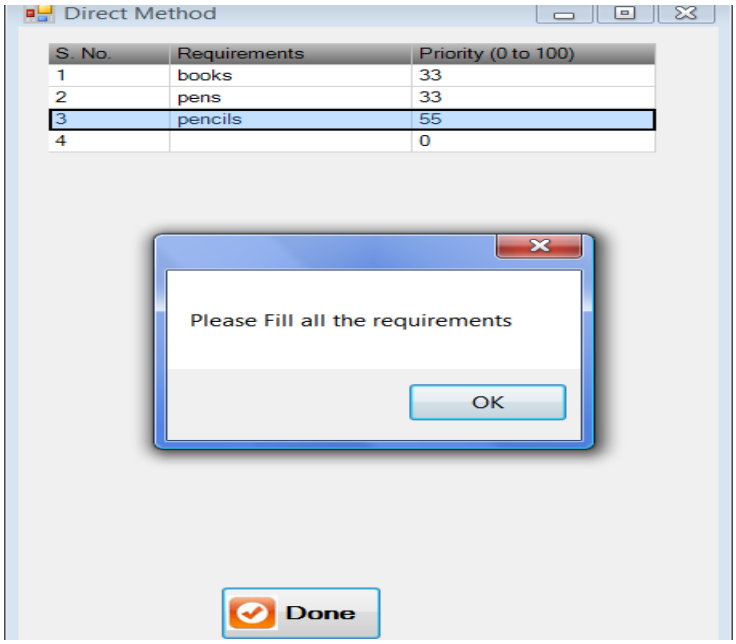


Fig 5.7: Incomplete Form

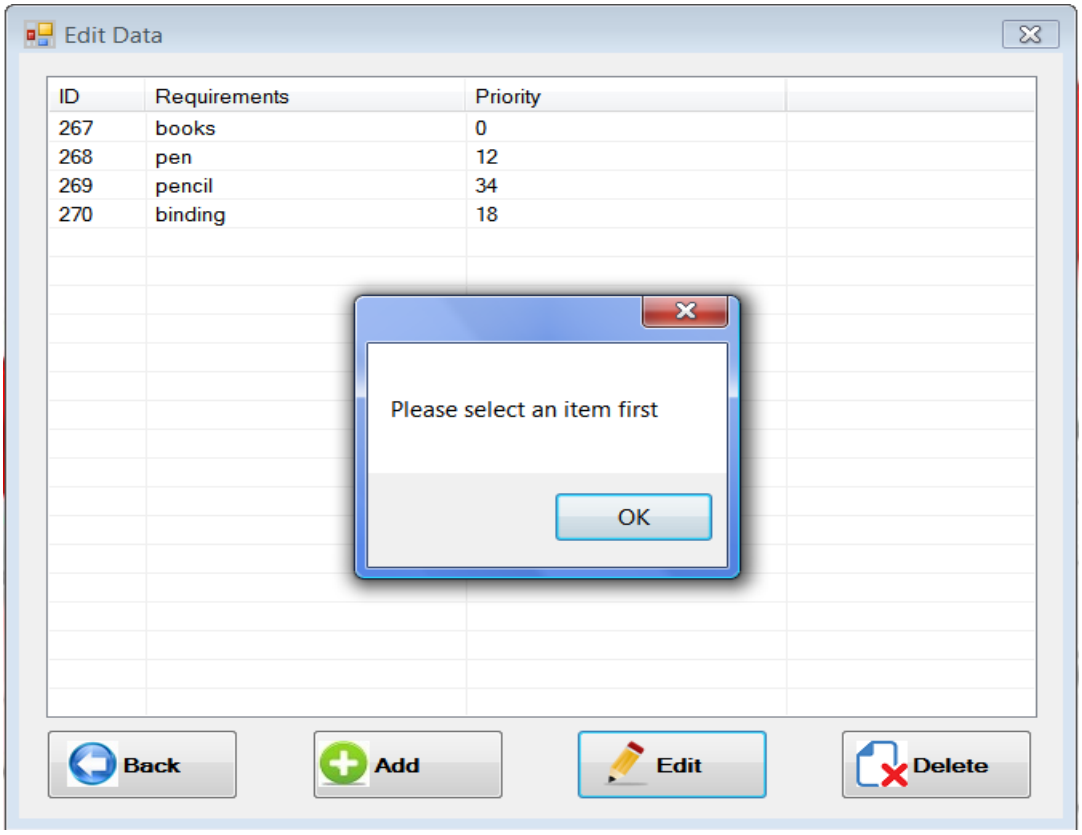


Fig 5.8: Selection Query

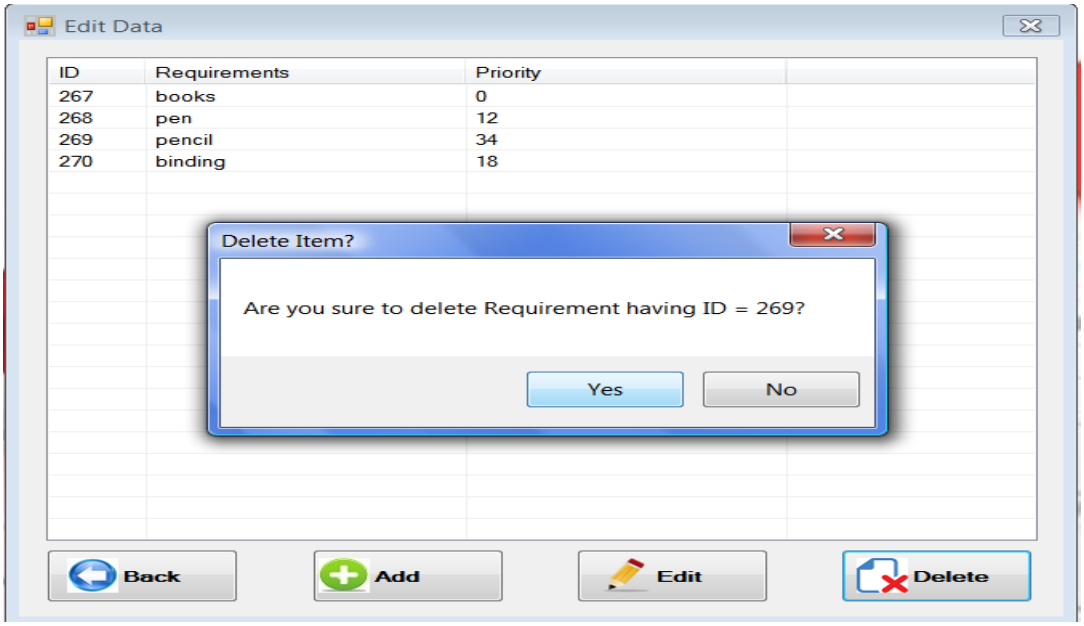


Fig 5.9: Deletion Query

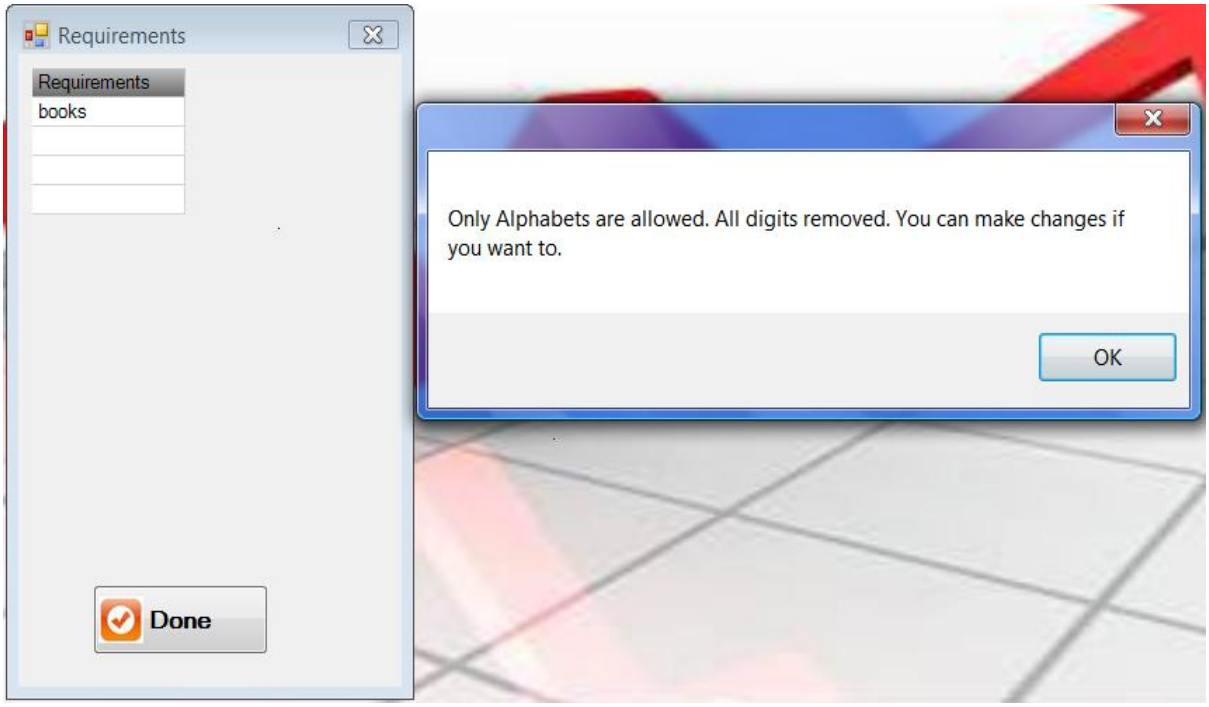
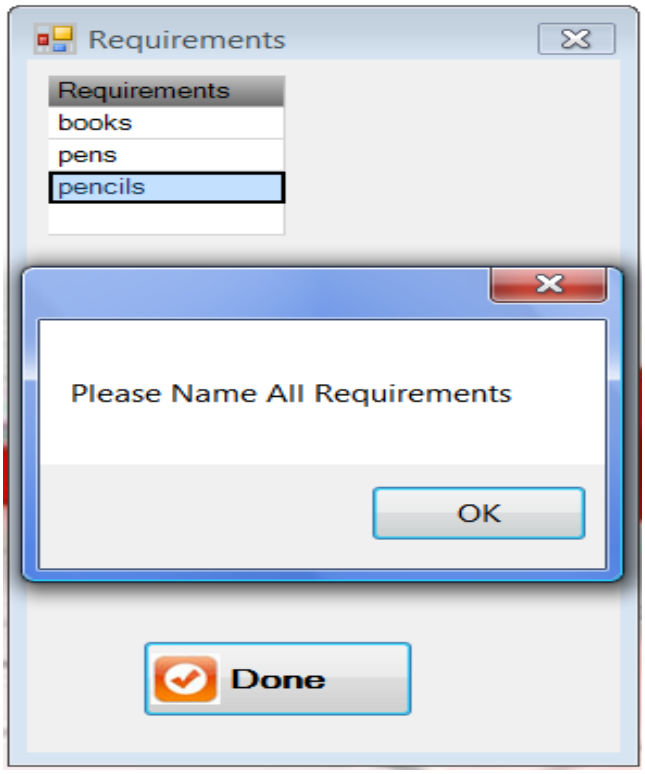
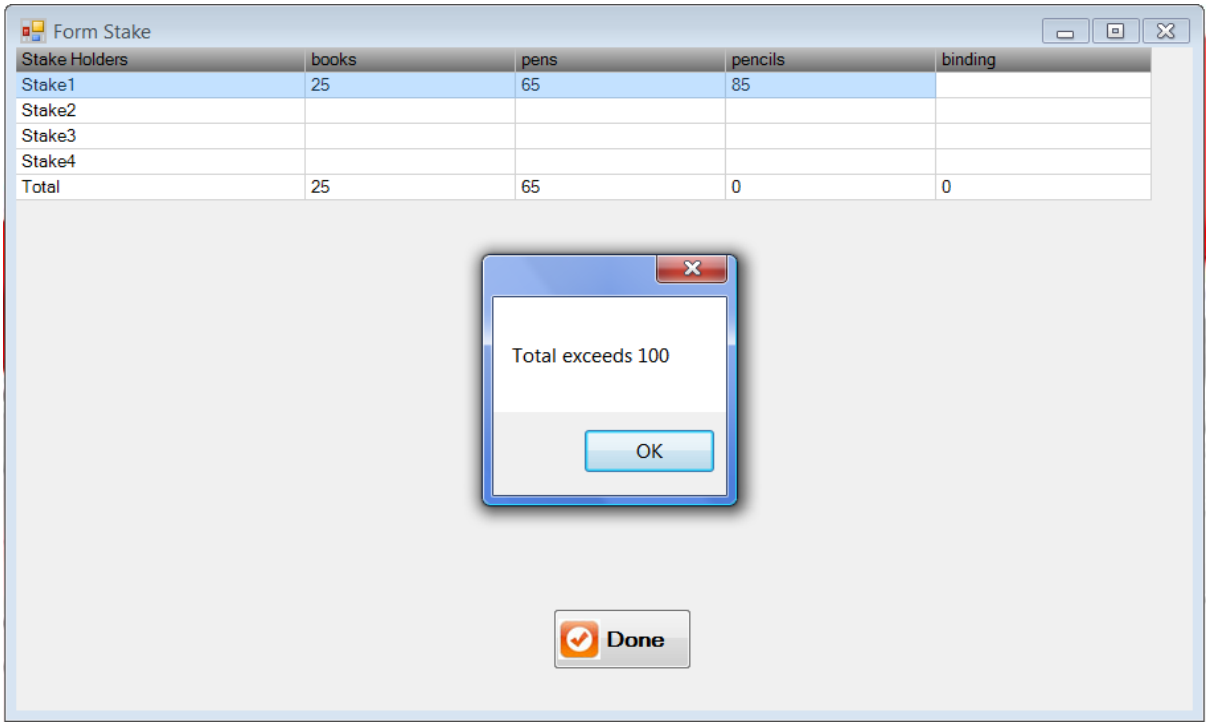


Fig 5.10: Data entry check



5.11: Incomplete form



5.12: Total exceeding 100 check

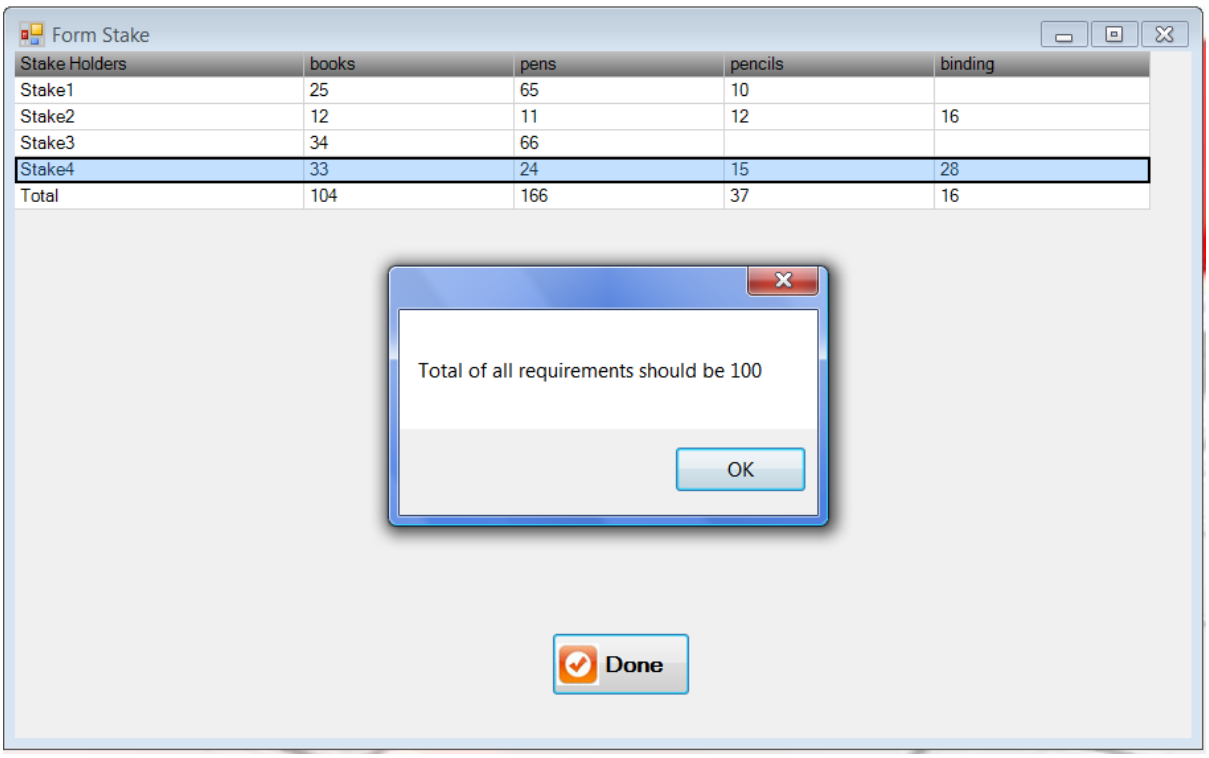


Fig 5.13: Total less than 100 check

5.3 Type of testing done on the Proposed Tool

The testing done on the tool Lets_Tree_Prioritise are as follows:

1. Unit testing
2. System Testing
3. Alpha testing
4. Integration testing

5.3.1 Unit Testing

The different units of the systems have been tested in this testing. The units are:

- Admin module
- Open project module
- Direct method
- 100 Dollar method
- Btree prioritization

The result of unit testing indicates that the individual units of the tool are working correctly as per requirement.

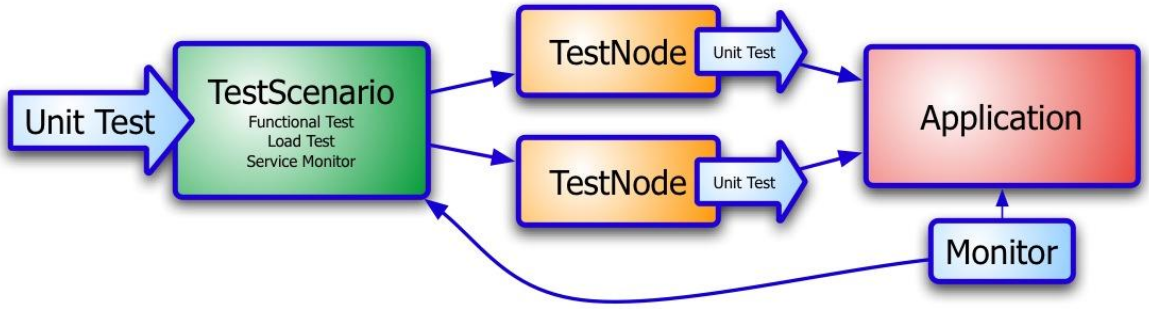


Fig 5.14 Unit testing

5.3.2 Integration Testing

Integration Testing is the phase in software testing in which individual software modules are combined and tested as a group.

It occurs after unit testing and before system testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing. Integration testing is a logical extension of unit testing. The entire system that is the entire tool after in integration is tested as an integrated unit. In the Fig below that colored blocks are the units and it they fit each other correctly and the project is giving the desired output then the integration testing is said to be correct.

Fig 5.15 Integration Testing

5.3.3 System Testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

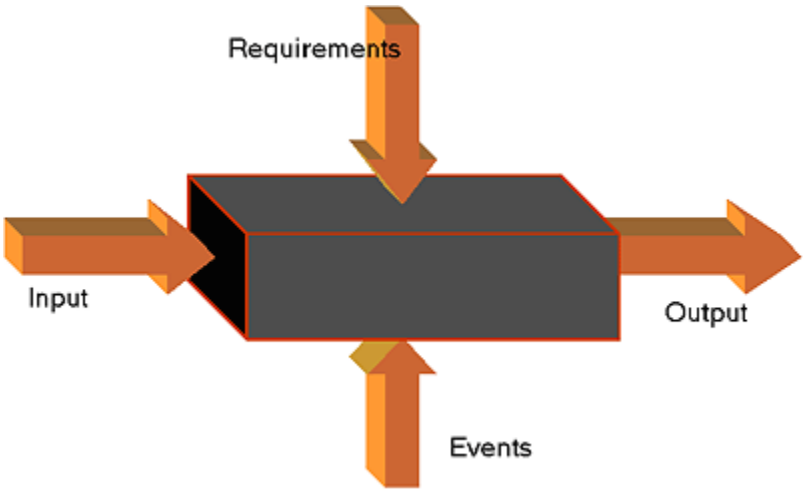


Fig 5.16 System Testing

5.3.4 Alpha Testing

Alpha testing is simulated or actual operational testing by potential users or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

Result of the test cases

The above described test cases were performed on the tool. The result depicts that the software is working correctly. The result is *pass* for every test case.

The unit test, integration test, system test and alpha test performed on the tool Lets_Tree_Prioritise results *pass.*

Chapter 6

Conclusion and Future Scope

This chapter is a conclusion of the work done and the future scope of the area specified.

6.1 Conclusion

In this thesis we presented a method based on the 100 dollar technique and Btree method to deal with prioritization of requirements. The 100 dollar approach consists of a method that gives priority values to the requirements, in order to aid system management and software quality. Every stakeholder has its own needs and therefore gives different importance value to different requirements. The 100 dollar method allows an input from all the stakeholders for the requirements. The priority values help the system analysts, developers and customers make decisions about the requirements. Based on the values given by the approach, requirements can be skipped or developed later. The 100 dollar method is the best if we have multiple requirements and many stakeholders. It's very difficult to understand the importance given by the stakeholder to a particular requirement. The stakeholders themselves cannot distribute the requirements correctly manually and eventually it becomes very difficult for the developer to notify. Therefore cumulative method is good to collect requirements.

The need to prioritize increases with the number of requirements so it is difficult to remember the priority values and to rank them according to the values. Knowing the rank of the requirements, the plan of the releases can be done. It is possible to plan by knowing which functions are critical and how they can be distributed. To find out the rank and the order in which these priorities can be implemented a method of Btree is used. Btree will help in ordering the requirements and also will allow an easy way to add, edit or delete the requirements. Btree Method is better than other sorting methods like bubble sort, quicksort, binary tree search as the complexity of Btree is o (log n).

Then, the schedule can be defined, adding the activities in the indicated order over successive releases. The priority information is very important when developing strategies and in consequence, works in benefit of the

- Software quality
- Fulfilling the stakeholder's expectations and constraints.

Prioritization of requirements requires some effort and work from the customer, but it brings benefits to the project and in consequence to the customer. The follow-on reduction in effort will be considerably more than the effort expended establishing priorities.

6.2 Future Scope

There are many methods to prioritize the requirements that are studied in the thesis but there is a lack of automated tools that can be used in collaborating environment. Therefore more work can be done in the field of automation. Other methods of prioritization like bubble sort, quick sort can be added in the tool mentioned in this Thesis. Empirical tests and verification can be done on the methods. Some methodology can be made to handle Stakeholders distributed geographically.

REFERENCES

[1] Peckham, J., and Lloyd, S. J. (2003) Practicing Software Engineering in the 21st Century. IRM Press. ISBN 1-931777-66-7.

[2] Pressman, R. S. (2001) "Software Engineering: A practitioners approach." 5th Edition. MacGraw-Hill. ISBN 0-07-118182-2.

[3] Bergman, B., and Klefsjö, B. (2003) "Quality - From customer needs to customer satisfaction" Student literature AB, Lund, Sweden.

[4] Valenti, S (2002). "Successful Software Engineering". IRM Press. ISBN 1-931777-33-0.

[5] Nuseibeh, B., and Easterbrook, S. (2000) "Requirements Engineering: A Roadmap", Proceedings of International Conference on Software Engineering (ICSE'00), ACM Press.

[6] Kotonya, G. and Summerville, I. (1998) "Requirements Engineering - Processes and Techniques". John Wiley & Sons, New York, NY. USA.

[7] Robertson, S., and Robertson, J. (1999) "Mastering the Requirements Process". ACM Press, London, UK.

[8] Loucopoulos, P. and Karakostas, V. (1995) "System Requirements Engineering" McGraw-Hill.

[9] Aurum, A. and Wohlin, C. (2003) "The fundamental nature of requirements engineering activities as a decision-making process", Information and Software Technology.

[10] Van Gurp, J.Bosch, J.; and Stahlberg, M. (2001) "On the notion of variability in software product lines". Proceedings of Working IEEE/IFIP Conference on Software Architecture.

[11] Brooks, F. P. (1995) “The Mythical Man-Month” Essays on Software Engineering, Addison-Wesley Longman, Boston, MA, USA.

[12] Berander, P. (2004a) “Prioritization of Stakeholder Needs in Software Engineering, Understanding and Evaluation”. Blekinge Institute of Technology. ISBN: 91-7295-052-8. Licentiate Series No 2004:12.

[13] Yourdon, E. (1997) “Death March Projects”, Prentice Hall.

[14] Lubars, M., Potts, C., and Richter, C. (1993) “A Review of the State of the Practice in Requirements Modelling”. Proceedings of the IEEE International Symposium of Requirements Engineering.

[15] Siddiqi, J. and Shekaran, M. (1996) “Requirements Engineering: The Emerging Wisdom”, IEEE Software.

[16] Lehtola L. and Kauppinen, M. (2004) “Empirical Evaluation of Two Requirements Prioritization Methods in Product Development Methods”. 11th European Conference on Software Process Improvement, (EuroSPI’04).

[17] Kuusela J. and Savolainen, J. (2000) “Requirements Engineering For Product Families” Proceedings - International Conference on Software Engineering.

[18] Boehm, B. W. (1981) “Software Engineering economics” Prentice Hall, Englewood Cliffs, NJ, USA.

[19] Beck, K. (1999) “Extreme Programming Explained” Addison-Wesley, Upper Saddle River, NJ, USA.

[20] Lehtola, L., and Kauppinen, M. (2004) “Empirical Evaluation of Two Requirements Prioritization Methods in Product Development Methods”. 11th European Conference on Software Process Improvement, (EuroSPI’04).

[21] Karlsson L., Berander P., Regnell B., and Wohlin C. (2004) “Requirements Prioritization: An Experiment on Exhaustive Pair-Wise Comparisons versus Planning

Game Partitioning". 8th International conference on Empirical Assessment in Software Engineering (EASE'04).

[22] Egyed, A. (2003) "A Scenario-Driven Approach to Traceability", IEEE Transactions on Software Engineering (TSE).

[23] Henry, J. and Henry, S. (1993) "Quantitative assessment of the software maintenance process and requirements volatility". In Proceedings of the ACM Conference on Computer Science.

[24] Barragans, Fernandez (2005) "Merging Requirements Views with Incompleteness and Inconsistency", Proceedings Software Engineering Conference.

[25] Tran, T.; and Sherif, J. S. (1995) "Quality Function Deployment (QFD): an effective technique for requirements acquisition and reuse", Proceedings of 2nd IEEE International Conference on 'Experience and Practice'.

[26] Christel, M. G. and Kang, K. C. (1992) "Issues in Requirements Elicitation", Software Engineering Institute, CMU/SEI-92-TR-12 0709.

[27] Davis, A. M. (2003) "The art of requirements triage", IEEE Computer, 36(3).

[28] Regnell B., Höst M., Nattoch Dag J., Beremark P., and Hjelm T. (2001) "An Industrial Case Study on Distributed Prioritisation in Market-Driven Requirements Engineering for Packaged Software", Requirements Engineering.

[29] Boehm, B. W., Grünbacher, P., and Briggs, B. (2001): "Developing Groupware for Requirements Negotiation: Lessons Learned", IEEE Software.

[30] Gruenbacher, P. (2000) "Collaborative Requirements Negotiation with Easy WinWin". Proceedings 11th International Workshop on Database and Expert Systems Applications.

[31] Bradner, S. RFC 2119, http://www.ietf.org/rfc/rfc2119.txt. Last Visited (September 25th, 2005).

[32] Berander, P. (2004b) “Using students as subjects in requirements prioritization” 2004 International Symposium on Empirical Software Engineering.

[33] Leffingwell, D., and Widrig, D. (2000) “Managing Software Requirements – A Unified Approach”.

[34] Karlsson, J., Wohlin, C., and Regnell, B. (1998) “An evaluation of Methods for Prioritizing Software Requirements”, Information and Software Technology.

[35] Hubbard, J. R. (2000) “Theory and Problems of Data Structures with C++”. McGraw-Hill, NY, USA

[36] Bagchi, P. and Rao, R. P. (1992) "Decision making in mergers: An application of the analytic hierarchy process”: Managerial and Decision Economics.

[37] Karlsson, J. (1996) “Software requirements prioritizing” Proceedings of the Second International Conference on Requirements Engineering.

[38] Karlsson, J.; Olsson, S., and Ryan, K. (1997) “Improved Practical Support for Large-Scale Requirements Prioritizing”, Requirement Engineering.

[39] Greer, D., Bustard, D. W. and Sunazuka, T. (1999) “Prioritisation of system changes using cost-benefit and risk assessments”. Proceedings IEEE International Symposium on Requirements Engineering.

[40] Md. Rizwan Beg (2008) “An Approach for Requirement Prioritization using B-Tree”. IEEE Software.

[41] Jung won Park(1999) “ Supporting distributed collaborative prioritization”IEEE Software.

[42]Jim Azar (2007) “Value oriented requirement prioritization in a small development organisation” IEEE Software.

List of Publications

1. Gurkiran Kaur, Seema Bawa," A Survey of Requirement Prioritization Methods", International Journal of Engineering Research and Technology ISSN: 2278- 0181, May 2013.(Published)

2. Gurkiran Kaur, Seema Bawa, "Automation of Software Requirement Prioritization using B-Trees" International Journal of Advanced Research in Computer Science and Software Engineering ISSN 2277 128X, July 2013. (Communicated)