ISSN:2320-0790

COMPUSOFT
An International Journal of Advanced Computer Technology

# Design and Testing Analysis of Requirement Prioritizations Technique

**Dinesh Singh, Trilok Gaba**
M.D University, Rohtak

**Abstract:** With the growing need of software in our day to day life, the complexity of the software is increasing as well and also the number of requirements associated to the modern software projects. So, in order to overcome the increasing demands and the pressure on the software engineers and program managers to deliver the software to the customers on time and in given budget, there is a huge need to identify the most important requirements and establish their relative importance for implementation according to certain criteria. The existing techniques for requirement prioritization although provide consistent results but are difficult to use and implement. Whereas some existing techniques that are easy to apply lack structure to analyze the complex requirements. Moreover the available techniques lack user friendliness in the prioritization process. So in order to overcome these issues or problems, a hybrid approach of two available techniques was proposed in our earlier work. In this paper we analyzed the design of the proposed system and testing plan of the system. Use case diagram and control flow diagram are used to explain the structure of the approach.

## 1. INTRODUCTION

Requirements prioritization is an essential mechanism of agile software development approach which aims to maximize the value of the software delivered to the clients and accommodate the changing requirements. Agile software development is a group of software development methods that are based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change [6]. The Agile Manifesto reads as follows:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

The key points[] of the manifesto are discussed below:

- Individuals and interactions – in agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
- Working software – working software will be more useful and welcome than just presenting documents to clients in meetings.
- Customer collaboration – requirements cannot be fully collected at the beginning of the software development cycle, therefore continuous customer or stakeholder involvement is very important.
- Responding to change – agile development is focused on quick responses to change and continuous development.

According to Kent Beck, the Agile Manifesto is based on twelve principles:

- Customer satisfaction by rapid delivery of useful software
- Welcome changing requirements, even late in development
- Working software is delivered frequently (weeks rather than months)
- Working software is the principal measure of progress
- Sustainable development, able to maintain a constant pace

- Close, daily cooperation between business people and developers
- Face-to-face conversation is the best form of communication (co-location)
- Projects are built around motivated individuals, who should be trusted
- Continuous attention to technical excellence and good design
- Simplicity—the art of maximizing the amount of work not done—is essential
- Self-organizing teams
- Regular adaptation to changing circumstances

Our proposed technique [1] belongs to the software engineering domain. The purpose of our application was to effectively gather the software requirements from the customer and prioritize them. The requirements elicitation in the proposed technique was done in 2 ways i.e. simple text-based story-board form and graphical form where the user can exhibit the requirements by drawing use case diagrams on a customized graphical editor through user control toolbox. After implementing the requirement prioritization technique [1] we felt the need of analysis of design and generating the test case for testing the feasibility of the system. So, in this paper along with design and testing part, design constraints are also discussed.

### 1.1. Overview of the Implemented Technique

The implemented approach was a hybrid approach of two available techniques that works as follows [1]:

- Firstly, the requirements are gathered from the users associated with the software product in a two very effective and user friendly manner. The users can submit their requirements in simple text based story form or also can provide a text file (.txt file) that lists the requirements, and the second way is graphical form where the user can exhibit their requirements by drawing use case diagrams on a customized graphical editor through user control toolbox that is available to the customer.

- After the requirements are gathered from the users, the developer can select the valid set of requirements out of the user supplied requirements.

- Once the valid set if requirements are identified by the developer, he can perform the first level prioritization of the requirements by applying certain quality attributes and sub attributes on the valid requirements and calculate the desirability values associated with each requirement. The quality attributes [1] used are:

1. Type: This attributes describes the type of requirement and thus have 3 sub attributes i.e. Functional, Imposed, and Product.

2. Scope: This quality attribute deals with the impact of a particular requirement on the overall system. So, the requirements that affect more number of (or all)subsystems are determined to be of higher priority than requirements that affect minimal number of subsystems. Scope attribute is defined with the following sub attributes: Subsystem 1 (S1), Subsystem 2 (S2), Subsystem 2 (S3)… Subsystem n (Sn).

3. Customer Satisfaction: Customer satisfaction plays an important quality attribute of a system. The more the number of customers satisfied by a requirement, the greater is the desirability of the requirement. So, the sub attributes for this quality attributes are Customer 1 (C1), Customer 2 (C2), Customer 3 (C3)… Customer n (Cn).

4. Perceived Impact (PMF): This quality attribute is based on expert opinion. It considers all the leads which can be software, hardware, systems and asks them that if the particular requirement is perceived as a major functionality. Thus, the sub attributes of PMF are Lead 1(L1), Lead 2(L2), Lead 3(L3)….Lead n (Ln).

5. Application-Specific: Depending on the type application domain, the attributes that are important to a specific software application act as the sub attributes to this quality attribute. The sub attributes taken this research are: Usability (U),Performance (P), Safety (S), Security (S), Reliability, and Interoperability (I).

6. Penalties: Various types of penalties are associated with software requirements. This attributes analyses if a particular requirement has any penalty associated to it. These penalties are: Costly (C), Risky (R), and Complex (Cx).

One important point to note here is that for each of the requirement, at least one sub attribute of the applied quality attributes must be selected. The first level of prioritization results in a requirement priority sequence based on the quality attributes that are selected to be applied on the valid requirements as per the knowledge of the requirement analyst or the software developer. The selected attributes are ticked and they act as binary value input 1 and the nonselected ones are binary value input 0. Then the second level of prioritization takes place where the developer gathers the opinions of five distinct users about what should be the priority sequence as per their choices. As per the gathered user's priority sequence, the developer calculates the degree of disagreement for each user elicited sequence with respect to the requirements prioritization sequence obtained by the developer in the first level after applying quality attributes.

After the disagreement factor for each user is calculated, the developer selects that user elicited sequence as the final priority sequence whose disagreement value comes out to be minimum. One important point here is fixation of threshold value i.e. maximum value of the disagreement. If the disagreement value exceeds the threshold value i.e. 5 in our case for any of the five users, crossover and mutation operations are applied to the overall population which means all the five priority sequences till the disagreement value becomes less than threshold value. This makes the prioritization process more user friendly and is also easy to implement and moreover this process resolves the case of ties that was occurring in [2] due to insufficient knowledge about relationship between the requirements and thus eliminates the need of user intervention which in itself was a conflicting issue.

## 2. DETAIL DESIGN DESCRIPTION

This technique belongs to the software engineering domain. The purpose of our application is to effectively gather the software requirements from the customer and prioritize them. The requirements elicitation in the proposed technique can be done in 2 ways i.e. simple text-based story-board form and graphical form where the user can exhibit the requirements by drawing use case diagrams on a customized graphical editor through user control toolbox. After the proposed software requirements are gathered from the customer, they are evaluated on the basis of some quality attributes and sub-attributes in order to calculate the desirability values on the basis of which, they are prioritized. After prioritization of requirements is done at system level by the developer, we take the user's input in relation to their expected prioritization to make the prioritization process more users friendly and try to compare the prioritization of requirements from the developer and customer's perspective.

The overall objective of the application is to choose a requirement prioritization sequence that has the minimum disagreement value with respect to the system prioritization. So, both the system and user's perspective is taken into consideration while prioritizing the requirements for software that is to be delivered. We implement this application as console-based wherein we assume that the developer and customer are virtually the same entities. In other words, the customer furnishes requirements to the developer and the developer inputs them to this tool through text-based or graphical editor in order to extract the valid requirements, compute their desirability and minimum disagreement sequence is selected as best priority sequence.

To understand the requirement clearly we have developed a use case diagram and control flow diagram shown in fig 2.1 and fig 2.2 respectively.

- The use case diagram in fig.2.1 shows the interaction of the developer with the proposed tool. The developer acts as the actor who performs various functions shown in the ellipses.
- The diagram in fig.2.2 shows the flow of information over the tool and also depicts the databases associated with each step.

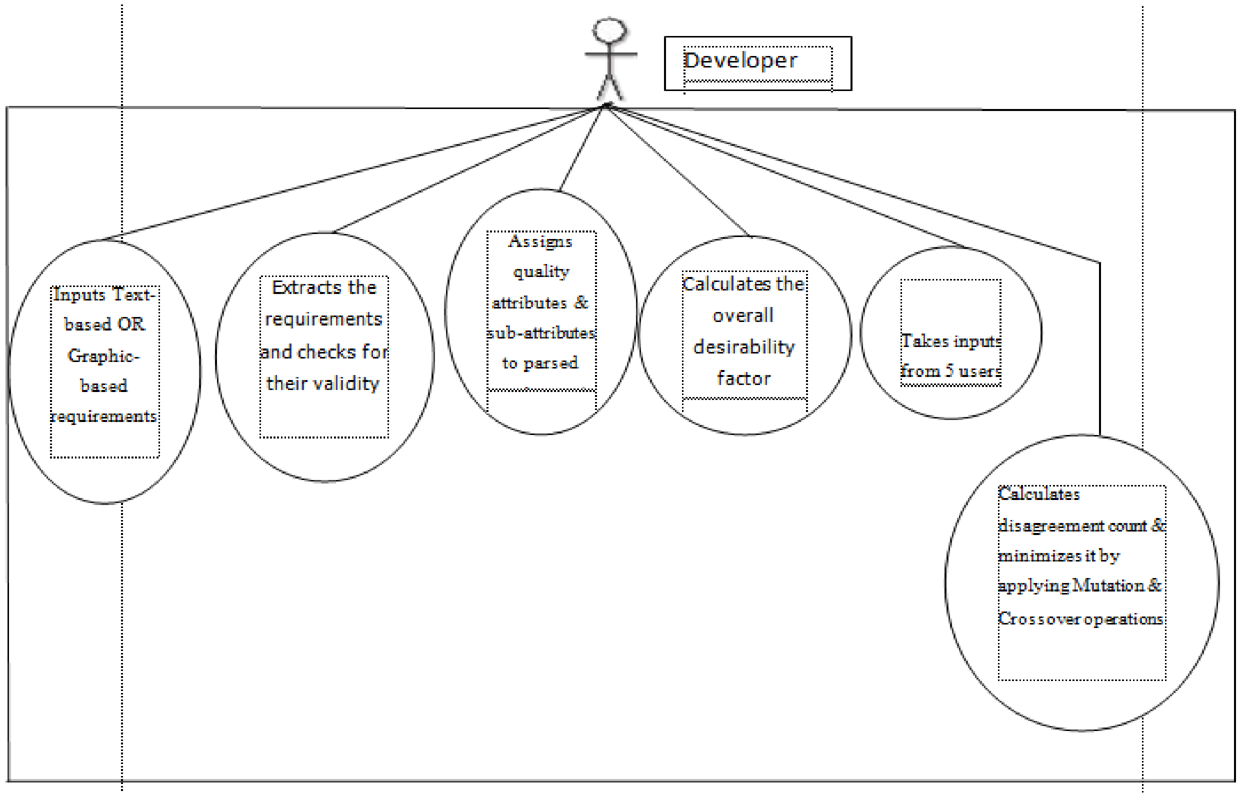
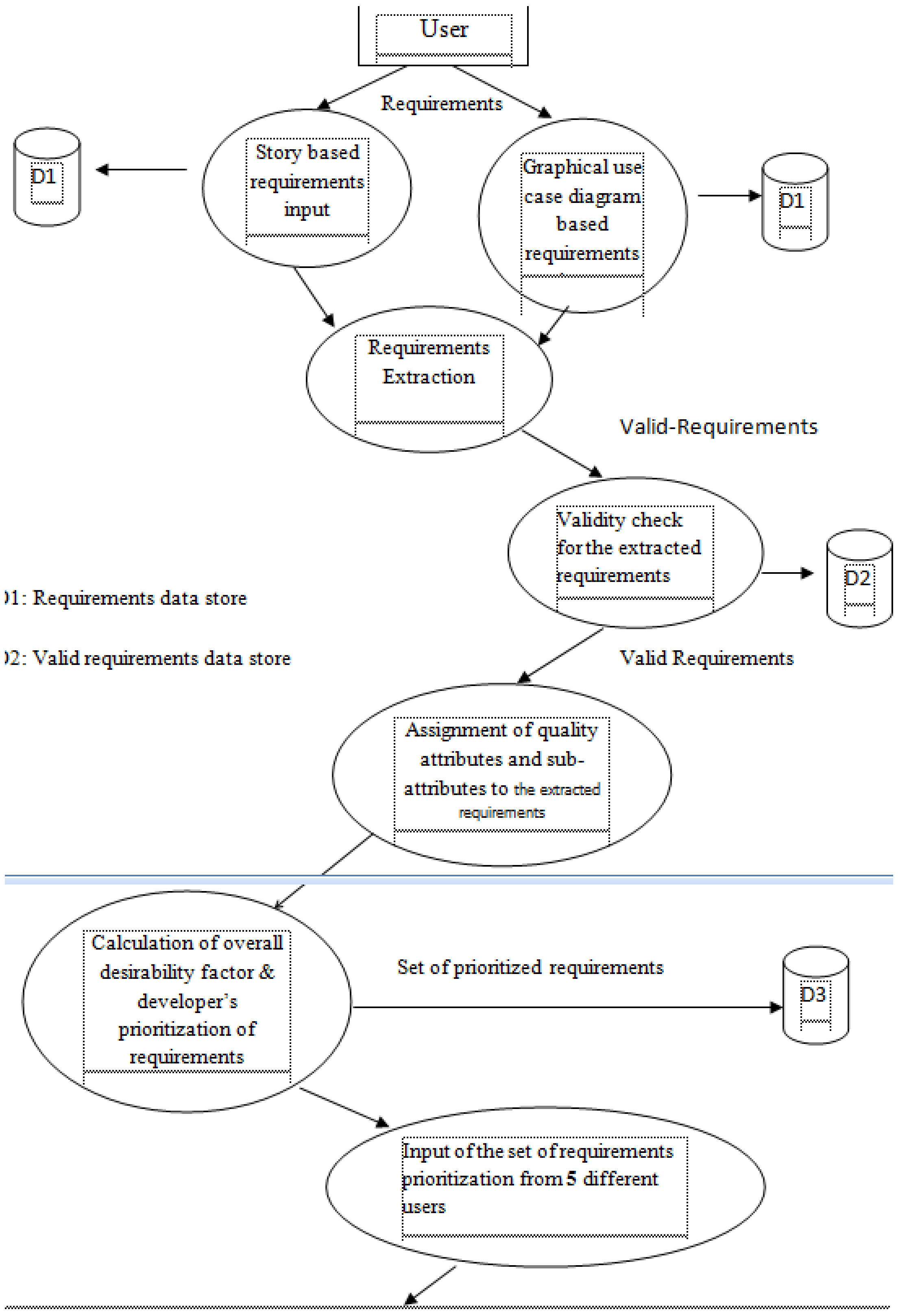

Developer
Inputs Text-based OR Graphic-based requirements
Extracts the requirements and checks for their validity
Assigns quality attributes & sub-attributes to parsed
Calculates the overall desirability factor
Takes inputs from 5 users
Calculates disagreement count & minimizes it by applying Mutation & Crossover operations

**Figure 1.1: Use case diagram**

There is an increasing need to develop a requirement prioritization technique that can be applicable to practical scenario. The technique should be easy to implement, provide consistent results. Moreover the existing techniques either have a very little or no role of customer in the requirement prioritization process. In order to overcome this problem, the proposed technique is easy to implement and rather than relying solely on the developer to obtain prioritized requirements, also take into account the user's perspective. The technique performs the prioritization at both the levels i.e. at developer level using quality attributes and at user level by taking the prioritization opinions of five distinct users.

## 2.1 Design Constraints

The design constraints in our project involve that the developer should be conveniently able to furnish the requirements both in the form of text-based input and graphical form. We have developed the text-based input in such a manner so that if the user wishes to type the requirements, he/she could do so. Otherwise, if the requirements are elaborated and are mentioned in some text file, the user may also input that text file containing the set of requirements. In the graphical form, we are giving convenience to the user in the form of user-defined controls that we have created in a customized fashion through DLL programming [4]. With the graphical editor, the user enjoys the privilege of creating customized use-case diagrams through drag 'n' drop and the controls can also be conveniently deleted through double-click action. Plus, after the requirements have been inputted or designed, their extraction has been done in such a manner so that the user can easily select the valid requirements among them and input them in the database. And after the parsing of requirements, we are providing an easy interface to the user so that he may assign the application of attributes and sub-attributes to the parsed requirements.


User
Requirements
Story based requirements input
Graphical use case diagram based requirements
D1
D1
Requirements Extraction
Valid-Requirements
Validity check for the extracted requirements
D2
)1: Requirements data store
)2: Valid requirements data store
Valid Requirements
Assignment of quality attributes and sub-attributes to the extracted requirements
Calculation of overall desirability factor & developer's prioritization of requirements
Set of prioritized requirements
D3
Input of the set of requirements prioritization from 5 different users

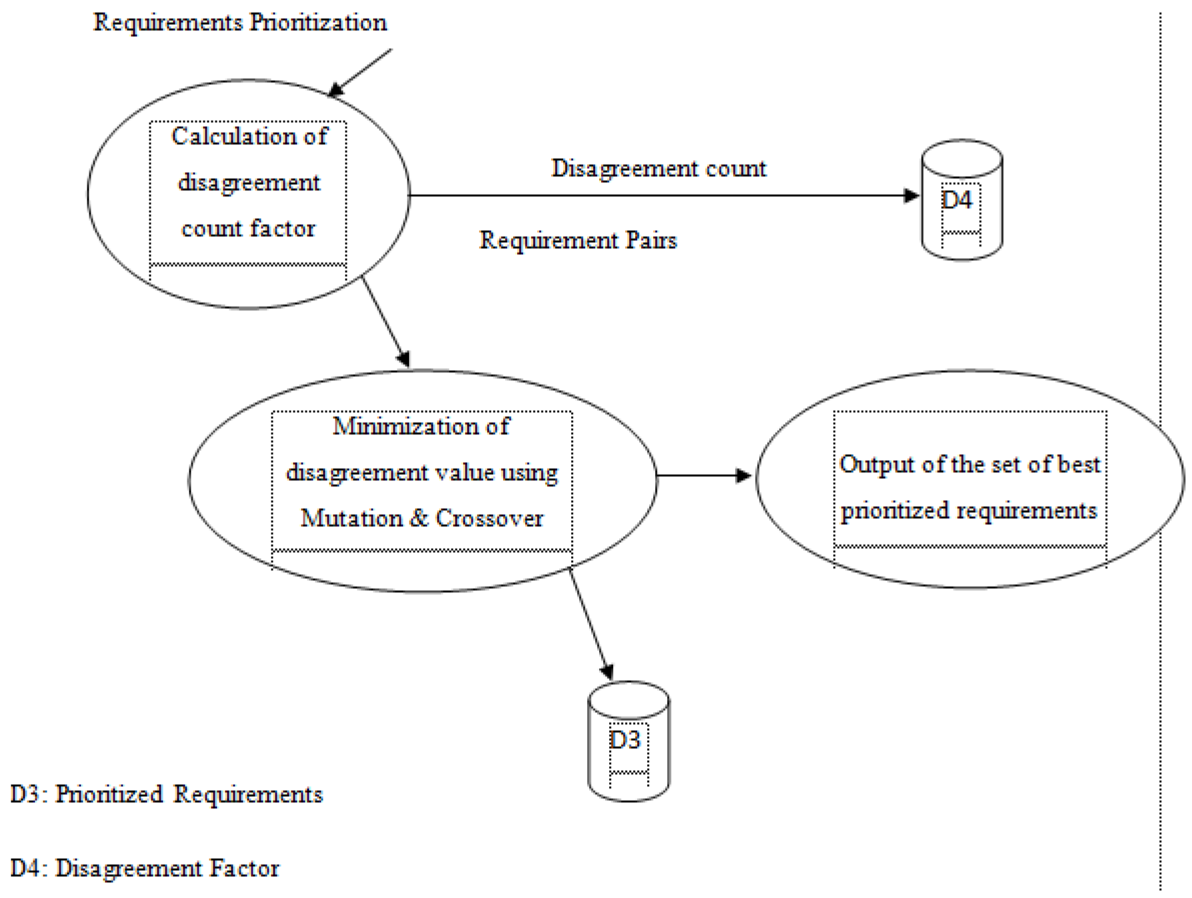

Requirements Prioritization
Calculation of disagreement count factor
Disagreement count
Requirement Pairs
D4
Minimization of disagreement value using Mutation & Crossover
Output of the set of best prioritized requirements
D3
D3: Prioritized Requirements
D4: Disagreement Factor

**Figure 2.2 Control Flow Diagram**

## 3. TESTING PLAN

In the testing of the technique, we created different test cases as below in order to perform module-wise testing. The test cases were formulated keeping the overall objectives of the applications into consideration. In other words, we tested a certain module to ensure that it should perform its own function in addition to some other related functionality with other modules, if necessary. We recorded our testing results by giving different inputs to the modules and observing the actual result as against the expected one. Wherever, the test result failed, we incorporated the essential modifications to correct it. The results are shown in Table 3.1 below:

**Table 3.1**

| S.NO | MODULE | INPUT | EXPECTED OUTPUT | ACTUAL OUTPUT | TEST RESULT |
|---|---|---|---|---|---|
| 1) | Project Title (Home Screen) | Blank | The user should be prompted through an error message | The error message is appearing | ok |
| 2) | Story based | Blank | The user should be prompted through an error | The error message is appearing | ok |

| | | | | | |
|---|---|---|---|---|---|
| | requirements input | | message | | |
| 3) | Story based requirements input | Proper requirements with full stops | The strings tokenization or parsing should happen at commas & full-stops | The requirements are parsed and extracted on full-stops with pairing | ok |
| 4) | Graphical based requirements input | Proper use-case diagrams with actors connected to their functions using arrows | The requirements should be extracted on the basis of distance between pixels of actors, arrows and oval symbols with respect to their adjoining diagrams | The requirements are parsed and extracted on the basis of distance between pixels of actors, arrows and oval symbols with respect to their adjoining diagrams | ok |
| 5) | Requirements Extraction & Validity | User's selection of valid requirements through a check box. | The checked requirements should be correctly stored in the database | The checked requirements are correctly getting stored in the database | ok |
| 6) | Application of quality attributes and sub-attributes to the extracted requirements | At least one attribute is not applied to set of extracted requirements | The user should be alerted and prompted about the same. | The alert message is generated in the form of a dialog box. | ok |
| 7) | Application of quality attributes and sub-attributes to the extracted requirements | At least one attribute is applied to set of extracted requirements | The overall desirability factor of the individual requirements should be successfully calculated | Overall desirability factor is successfully getting calculated | ok |
| 8) | Obtaining a set of 5 requirements prioritization from different users | The 5 requirements set are not correctly inputted in a proper, formatted manner | The user should be alerted about same | Prompt message in the form of dialog box is appearing | ok |
| 9) | Obtaining a set of 5 requirements prioritization from | 1 user inputs multiple sets of prioritized | An alert message should be generated | The alert message is getting generated that 1 user can | ok |

| | | | | | |
|---|---|---|---|---|---|
| | different users | requirements set | | furnish only 1 set of requirements prioritization | |
| 10) | Obtaining a set of 5 requirements prioritization from different users | If all the 5 requirements prioritization sets are inputted correctly in a formatted manner and one per user | Disagreement count factor should be correctly generated for each requirement set after its comparison with the main set of developer's perspective. | Disagreement count factor is correctly getting calculated and displayed in the form of a grid for each of the 5 inputted requirements prioritization sets | ok |
| 11) | Once the disagreement count factors are generated, they should be minimized using the Mutation & Crossover operations | After the counts are generated, they are correctly un-equalized and then applied the mutation and crossover functions for further minimization of the disagreement count | The best set of prioritized requirements set should be generated. | The best set is getting generated and outputted. | ok |

**3.1 Analysis**

The analysis of the system was done rigorously because this is such a phase where all the loopholes had to be discovered keeping company's objectives & challenges in mind. We performed the analysis in 02 parts i.e. Feasibility Analysis & Requirements Analysis.

- **Feasibility Analysis**

1) We studied the whole system & its objectives. Calculated the total time & resources incurred on every function being done manually.
2) Bifurcated the complete system into a list of functions & the users who operate on them.
3) Further subdivided all the functions into a list or source of requirements/inputs & clearly defined the output/expectation from each function.
4) The interaction, communication & dependency of all the functions between each other were carefully analysed in terms of sequence & information.
5) The source & flow of the information was determined & how would it be processed & used was considered.
6) Finally, we visualized the complete system with automated functions & compared the total time & resources being incurred to check the feasibility & see whether it is fulfilling all the necessary objectives.

- **Requirement Analysis**

1) This was a subset of feasibility analysis in which we defined a set of objectives for the complete system after thoroughly analyzing it.
2) All the objectives were further subdivided into a set of function(s).
3) The input(s) required by each function & the expected output(s)/behavior was/were clearly defined.
4) The source of information/input to every function was determined & its corresponding processing, usage & storage were also taken into account.
5) After this the interdependency & communication was finalized.

## Conclusion

The order in which requirements are implemented in a system affects the value of the software that is to be delivered. So, it is important to identify the important requirements and rank them as per their significance. Design of any of the application plays a important role in analyzing the system. In the paper design of the implemented technique is shown using the use case diagram and control diagram. Testing of the system to check the reliability of the technique is done by designing the different test cases and running those test cases on the system. .All the test case are documented and summarized using a table. In future, it may also include the principles of other software engineering techniques like the estimation of different software metrics like Effort, Time, People, and Cost etc apart from quality attributes.

## REFERENCES

[1] Dinesh Singh, Aman Jatain, "An Interactive Approach to Requirements Prioritization Using Quality Factors", International Journal in Foundations of Computer Science & Technology (IJFCST), Vol. 3, No.6, November 2013.

[2] Berntsson-Svensson R., Gorschek T., Regnell B., Torkar R., Shahrokni A., Feldt R.,"Quality Requirements in Industrial Practice – an extended interview study at eleven companies", IEEE Transactions on Software Engineering, vol. 38 no. 4, pp. 923-935, 2012.

[3] Ritu and Dr. Nasib Singh Gill,"A Comparison among Various Techniques to Prioritize the Requirements". IJCSMS International Journal of Computer Science & Management Studies, Vol. 12, Issue 03, Sept 2012 ISSN (Online): 2231 –5268.

[4] R. Wiegers, Software Requirements. Redmond: Microsoft Press, 1999.

[5] Mikko Vestola," A Comparison of Nine Basic Techniques for Requirements Prioritization", Helsinki University of Technology.

[6] Nancy R. Mead "Requirements Prioritization Case Study Using AHP" 2008 Carnegie Mellon University.

[7] Berntsson-Svensson R., Gorschek T., Regnell B., Torkar R., Shahrokni A., Feldt R.,"Quality Requirements in Industrial Practice – an extended interview study at eleven companies", IEEE Transactions on Software Engineering, vol. 38 no. 4, pp. 923-935, 2012.