An Interactive Approach to Requirements Prioritization Using Quality Factors

Dinesh Singh¹ and Aman Jatain²

¹Department of Computer Science, M.D.University, Rohtak
²Department of Computer Science and Information Technology, ITM University, Gurgaon

ABSTRACT

As the prevalence of software increases, so does the complexity and the number of requirements associated to the software project. This presents a dilemma for the developers to clearly identify and prioritize the most important requirements in order to deliver the project in given amount of resources and time. A number of prioritization methods have been proposed which provide consistent results, but they are very difficult and complex to implement in practical scenarios as well as lack proper structure to analyze the requirements properly. In this study, the users can provide their requirements in two forms: text based story form and use case form. Moreover, the existing prioritization techniques have a very little or no interaction with the users. So, in this paper an attempt has been made to make the prioritization process user interactive by adding a second level of prioritization where after the developer has properly analyzed and ranked the requirements on the basis of quality attributes in the first level, takes the opinion of distinct user’s about the requirements priority sequence. The developer then calculates the disagreement value associated with each user sequence in order to find out the final priority sequence.

KEYWORDS

Requirement, Prioritization, Requirements Engineering, Quality attributes, Desirability, Disagreement, AHP, Genetic Algorithm.

1. INTRODUCTION

Software engineering decision support plays an important role in developing valuable software as it is important to make right decisions for the development of right product. Requirement engineering is one of the most important activity carried out in the requirement engineering stage that supports this decision making process [1]. A requirement can be defined as a feature that the system must have or a constraint that it must satisfy for it to be accepted by the client or customer. Requirement Prioritization facilitates the requirement engineering process that helps in making crucial decisions about requirements associated with the development process of a software system [2]. Requirement Prioritization can be defined as the process of selecting the most important requirements that are crucial for customer satisfaction in a software application as the order in which requirements are implemented in a system affects the value of the software delivered [15]. 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 [10]. So, in order to overcome the increasing demands and the pressure on the

DOI:10.5121/ijfcst.2013.3603
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 [15]. The existing techniques
for requirement prioritization although provide consistent results but are difficult to use and
implement [8]. 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 is proposed. The proposed technique provides two mechanisms in which the
gathered requirements can be fed into the prioritization tool i.e. 1. Text Based Story Form where
the gathered requirements can be inputted by simply typing them in the text box provided or by
simply browse the text file containing the user’s requirements and, 2. Use Case Form where a
graphics editor is provided containing user-defined graphical use-case controls which follows
simple drag and drop mechanism. Once the gathered requirements are fed into the tool, the two
levels of prioritization takes place.

2. RELATED WORK

While delivering a market-driven software product development, there is a requirement overload
situation as there are a large number of requirements, even more than what can be dealt with.
Even in bespoke software development, the numbers of requirements are usually more than that
of the available resources to fulfill them. So, there are always questions like what requirements to
choose and prioritize as well as what features should be included in the product. This scenario is
when requirements prioritization comes in. There are a lot of techniques which have been
researched upon in order to prioritize requirements. Berander and Aurum [ ] have listed various
techniques like numerical assignment where the prioritization is based on grouping the
requirements into different categories, where three groups used in [R. Wiegers, Software
Requirements. Redmond: Microsoft Press, 1999] are high, medium and low.; ranking which
means the most important requirements are ranked first, and the less important requirements are
ranked last, on an ordinal scale; top-ten requirements; Analytical Hierarchy Process technique in
which the pairs of requirements are compared according to their importance and these
comparisons provide an understanding of each requirements share of the total value[2], and
Cumulative Voting or the 100$-method [8] in which the stakeholders participating in the
prioritization are given a number of imaginary units like 100 dollars, 1000 points, etc. which are
distributed among the requirements that are to be prioritized. A number of units that are assigned
to a requirement represent its priority. The results of the prioritization are represented on a ratio
scale that gives the information of how much a requirement is more/less important than other.
Various other researchers have worked in the field of requirements prioritization.

Carlos, Erica et al [1] proposed a Quality-Based Requirement Prioritization approach for
evaluating the quality of requirements in software projects based on multiple quality evaluation
criteria. Specifically, it presents a methodology that uses the concept of “Desirability Functions”
that acts as a unified measurement to represent how well requirements meet the quality attributes
and how important the quality attributes are for the project. These quality attributes are applied to
individual requirements based on the knowledge of the requirements analysts. The selected
quality attributes for the requirements form a metric for evaluating the importance of
requirements and based on the values calculated by the formula proposed in this paper,
requirements are ranked accordingly with relative importance for a particular software project. By
modifying the parameters of the desirability functions, quality and priority of requirements can be evaluated. Overall, the approach presented proved to be a feasible technique for efficiently evaluating the quality and priority of requirements in software projects and also can be extended to include additional quality attributes.

Paolo, Angelo[2] have proposed an interactive genetic algorithm that exploits the user knowledge to prioritize requirements. They performed a pair wise comparison on the information useful to prioritize the requirements for a software system. They have applied the proposed algorithm to a real case study, consisting of a non-trivial number of requirements, whereas AHP is hardly applicable due to a system with large number of requirements. This approach scaled to the size of the considered case study and produced a result that outperforms GA (i.e., a genetic algorithm which optimizes satisfaction of the initial constraints, without gathering further constraints from the user). Specifically, by eliciting between 50 and 100 pair wise comparisons from the user it was possible to obtain a substantially better ordering of the prioritized requirements.

Azeem Ahmad, Aamir Shahzad et al [3] considered various attributes that can affect requirements prioritization while dealing with geographically distributed Stakeholders. They analyzed that people related to a software product make their contribution to SDLC (Software Development Life cycle) either by staying at a particular place or by being geographically distributed at different locations. The term Globally Distributed Stakeholders refers to those stakeholders, who are related to a project but are distributed in different locations all over the world. These distributed locations can be within same area or in a continent. These distributions can also be located all over the world where each stakeholder can have his/her own perception about the software requirements as well as requirements expectations for a software project. This paper provides a framework that can be used to identify those requirements that can play an essential role in a product success during distributed development.

Joseph and Charlotte[] discussed various techniques for requirements elicitation. Requirements elicitation is an essential mechanism that tells the developer what customers really need from the product that is to be delivered to them. Once the requirements are gathered, the developers identify the properties that must be included in the software in order to meet the customer requirements. This paper discussed various techniques of gathering the knowledge from the user’s requirements along with their advantages and limitations. The techniques discussed in this paper include introspection that is the most common source of information; but experiences indicate that it can be very misleading. Interviews and questionnaires are other types of techniques that are widely used to gather user requirement data, and sometimes protocol analysis is also used. All the techniques are good to use but this paper suggests that techniques like interaction, conversation and discourse analyses are detailed and more accurate to use. Ravi Prakash Verma, Alok Joshi et al [] proposed an algorithm using B-Tree for reduction in number of comparisons required for requirement prioritization. Large software projects have large number of requirements and moreover these requirements keep on changing. So, there is immense pressure on the developers to deliver the software on time as well as manage the changing requirements. In order to resolve these issues, requirements are prioritized relative to each other. Most of the requirement prioritization approaches use pair wise comparison to achieve relative prioritization among the set of requirements but its disadvantage is that it consumes a lot of resources because a large number of comparisons are if there are changes in requirements. This paper proposed an algorithm using B-tree which effectively reduced the
number of comparisons required to prioritize the requirements. This method prioritized n requirements in just \( \log_2 t \times \log t(n) \) comparisons where prioritize the requirement, which when compared with the complexity of AHP comparisons comes out to be less.

Muhammad Atif Iqbal, Athar Mohsin Zaidi and Dr. Saeed Murtaza presented a new model based on AHP to prioritize requirements in a market driven software development where a large number of requirements coming from different users, customers and competitors imposes a burden on the requirement engineering process. So it is necessary for the product management to analyze and select those requirements that are aligned with the overall business goals and dispose of the other unimportant ones as soon as possible. This paper presents a MDRPM (Market Driven Requirement Prioritization Model) based on Analytic Hierarchy Process in a sequential stepwise manner which effectively produced the required results and reduced the number of comparisons required to prioritize the requirements as compared to traditional AHP.

3 PROPOSED TECHNIQUE

The proposed technique can be considered as a hybrid approach to requirements prioritization where the first level of prioritization takes place at developer level and the second level of prioritization incorporates the role of user in order to add interactivity to the prioritization process. Before the prioritization process begins, the developer needs to gather requirements from the users. The proposed technique provides two mechanisms in which the gathered requirements can be fed into the prioritization tool i.e. 1. Text Based Story Form where the gathered requirements can be inputted by simply typing them in the text box provided or by simply browse the text file containing the user’s requirements and, 2. Use Case Form where a graphics editor is provided containing user-defined graphical use-case controls which follows simple drag and drop mechanism. Once the gathered requirements are fed into the tool, the two levels of prioritization takes place.

This work has been developed using C# (C-sharp.NET) technology with SQL Server 2008 as the back-end database. The application executes on Visual Studio 2010 IDE. This application has been developed on the principles of object-oriented techniques wherein we define our own classes for performing any specific function, import some existing, pre-defined, inbuilt classes for including their properties in the project. We have defined the architecture in such a manner so that every class performs functions which are cleanly separated from the functions of the other class. We have used the database interface layer provided by the .NET technology in the form of LINQ2SQL. LINQ to SQL is a component of .NET Framework version 3.5 which provides a runtime infrastructure to manage relational data as objects. LINQ to SQL maps the data model of a relational database to an object model expressed in the programming language of the developer using .NET technology. When the application runs, LINQ to SQL translates into SQL the language-integrated queries in the object model and sends them to the database for execution. When the database returns the results, LINQ to SQL translates them back to objects that you can work with in your own programming language. Developers using Visual Studio typically use the Object Relational Designer, which provides a user interface for implementing many of the features of LINQ to SQL. Moreover, a graphical editor has been built up that enables the user to furnish the requirements in the form of use-cases. The graphical editor contains user-defined controls which we have coded and compiled in separate C# files and then integrated or mapped them within our project through addition of their references. The resources
used within the application have been kept in a separate folder in order to include them from their designated paths. In this way, the applications consist of user-defined classes which inherit the inbuilt classes and use their objects to call functions for performing specific operations. The extraction of requirements from the use case uses the custom logic 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 this pixel distance between various symbols.

### 3.1 Requirement Gathering and Selection of Quality Attributes

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
   a. Requirements that affect minimal number of subsystems.
   b. Scope attribute is defined with the following sub attributes: Subsystem 1 (S1), Subsystem 2 (S2), and 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. Figure 3.1 shows the interaction of the developer with the proposed tool. The developer acts as the actor who performs various functions shown in the ellipses.

![Figure 3.1: Developer Interaction with Proposed Tool](image)

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 non-selected 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.

5. RESULTS

The entire approach for the requirements prioritization can be presented in a series of sequential steps that are mentioned below:
STEP 1

First step is requirement elicitation phase. In this phase, the developer interacts with the users to whom the software product is to be delivered in order to gather the requirements from different users.

STEP 2

Once the developer has obtained the user’s requirements, he needs to input them to the requirements prioritization tool. This process is automated here. The developer has two ways to feed in the gathered requirements:

a) Text Based Story Form
b) Use Case Form

The developer enters the home screen of the prioritization tool and enters the project name. The developer can then select the mode in which the requirements are to be fed into the project i.e. Text based story form or Use case form.

If the developer selects the Story form, he/she will be leaded to the next page where one of the two options can be followed to input the requirements in text form:

1. Either the requirements can be typed in the text box provided or,
2. Developer can click on the browse text file button and can select any text file which could have been provided by the users to whom the project is to be delivered. This text file can contain the user requirements in text form separated by full stops (.)

Fig.4.1.Home Screen
Fig. 4.2. Developer types the requirements in textbox

Fig. 4.2 shows the case when the developer types the requirements into the text box. In case when the developer has a text file containing the requirements as shown in Fig. 4.3, he/she can simply click on the browse text file button.

Fig. 4.3. Text File containing user requirements

Once the developer clicks the browse text file button, a dialog box open up where the user can browse through the system and select the requirement document text file from the appropriate location.

Fig. 4.4. Browsing the user requirements text file

After the text file is selected, and open button is clicked in the dialog box, the contents of the text file is copied into the text box provided on the requirement elicitation page.
STEP 3

Once the gathered requirements are presented to the tool, the tool parses the text and extracts the requirements by splitting and tokenization of text wherever a full-stop (.) occurs in the text-based sentence. As the developer clicks on the get requirements button the requirements present in the text box splits wherever the full stop (.) occurs in the text and are listed automatically on the same page in the space provided below the text box.

Once the requirements are extracted, the developer or the requirement analyst possesses the relevant knowledge to judge the valid set of requirements from the requirements that are obtained from different users. Based on his/her knowledge the requirement analysts for the project can select the valid requirements among all the extracted requirements by simply checking the check box against each valid requirement. The selected valid requirements are then saved by clicking on the save requirements button. When the save requirements in clicked, the finalized requirements i.e. the set of developer selected valid requirements are inputted to the first level of the requirements prioritization. For ex: in the Fig.4.6 below the developer selects REQ 1, REQ 3, REQ 4, REQ 5, REQ 6, REQ 7, REQ 8 as the valid set of requirements leaving REQ 2. So, all the selected requirements i.e. except REQ 2 will go through the two prioritization levels.

But if the developer selects the Use Case form as in Fig.4.7:
The developer will see a graphical editor where he can simply drag and drop the functionality provided in the toolbox on right hand side and show the interactions or relationship between various users and their requirements in graphical form.

After graphically drawing the relationship between the user and their requirements, developer simply clicks the get requirement button and the requirement description along with the concerned user name is displayed in the space provided below on the same page. There can be multiple actors with the same name in order to incorporate the large no. of requirements of a single user. The requirements are extracted on the basis of distance between pixels of actors, arrows and oval symbols with respect to their adjoining diagrams. Once the developer gets the extracted requirements from the use case diagram, he can simply select the valid set of requirements among them using his own knowledge in a similar way as he did in case of Story form elicitation. The developer after ticking the checkboxes in front of the valid requirements clicks on the Save Requirements Button in order to process and prioritize the selected requirements.
STEP 4: After the valid requirements are chosen by the developer, the first level of requirement prioritisation takes place. The valid requirements are fed to the quality attributes [2] as shown in fig.4.10. The developer then selects the quality attributes that must be applied to a particular requirement based on the relevant knowledge. Sub attributes of each attributes are listed; the developer can simply select the appropriate ones as per each requirement based on his knowledge.

One important point to be noted here is that for each quality attribute for a particular requirement, at least one sub attribute must be selected. Otherwise, an error message appears indicating which quality attribute for which particular requirement is missing. This is shown in fig.4.10 above. If the quality attributes for each requirement is furnished properly without missing any attribute, the developer clicks on the Save Requirement attributes button and the disagreement value for each requirement is calculated first for each attribute and then the overall desirability value. The formula [1] used to calculate the desirability value for first five quality attributes is:

\[
d_{y_{ij}} = \begin{cases} 
0 & y_{ij} \leq L \\
\left(\frac{y_{ij} - L}{T - L}\right)^{r_{j}} & L \leq y_{ij} \leq T \\
1 & y_{ij} > T 
\end{cases}
\]

And for the last quality attribute (penalty), formula [1] used is:

\[
d_{y_{ij}} = \begin{cases} 
1 & y_{ij} < T \\
\left(\frac{U - y_{ij}}{U - T}\right)^{r_{j}} & T \leq y_{ij} \leq U \\
0 & y_{ij} > U 
\end{cases}
\]
CALCULATION OF DESIRABILITY VALUE:

The value that the developer selects acts as binary input 1. The calculation desirability value for REQ 1 is done in the following manner. For REQ1:

For QA1 as per values in Table.3.2 [1];

\[ Y_{ij} = \frac{1}{3} = 0.667 \] (the no. of sub attributes selected/ total no. of sub attributes)

After calculating this value, the formula [1] mentioned above is applied where:

\[ L=0, T=100, U=100 \] and \[ r=1. \]

This gives the desirability value of REQ 1 for QA1.Similarly the desirability value for each quality attribute is calculated depending upon the sub attributes ticked by the developer and the values in Table.3.2.After desirability values for each attribute is calculated, the overall desirability is calculated as the geometric mean of desirability values of all the quality attributes for a particular requirement. Once the desirability value for each requirement is calculated, they are arranged in descending order to get the requirements priority sequence which is displayed in the message box that prompts when save requirement

If the minimum disagreement value is greater than the threshold value, then crossover and mutation operation of the genetic algorithm is applied on the entire population i.e. the entire 6 user’s priority sequence in order to decrease the disagreement values.
MUTATION OPERATION: The mutation process [2] uses requirement-pair-swap operator, which selects two requirements and swap their position in the mutated individual. The selection of two requirements to be swapped is done randomly.

CROSSOVER OPERATION: The crossover Process [2] uses the cut-head/fill-in-tail and the cut tail/fill-in-head operators, which selects a cut point randomly in the chromosome /sequence of the first individual, keep either the head or the tail, and fill-in the tail (head) with the missing requirements, ordered according to the order found in the second individual/sequence to be crossed over. These mutation and crossover operations are applied iteratively to the entire population i.e. 6 users priority sequence till the minimum disagreement value becomes less than the threshold value or the maximum limit till these operations can be applied is reached(whichever comes first). This case is shown in fig.4.14 below, where the min value of disagreement is 7.

![Fig.4.14.Disagreement value greater than threshold value](image)

So, as the user clicks the get final prioritization button, he is prompted with the message box saying “Mutation and Crossover will be applied.”

![Fig.4.15.Mutation and Crossover message box appears](image)
After the mutation and crossover operations are applied, the revised disagreement values are reflected in Fig. 4.16. But the minimum value now comes out to be 11 which is still greater than the threshold value. So, when the get final Prioritization Sequence button is clicked same message is prompted: “Mutation and Crossover will be applied.”

So, mutation and crossover is applied again on the entire 6 sequences and the result is shown in Fig. 4.18 where the minimum disagreement value now comes out to be 4 which is less than threshold.
Now when Get final Prioritization Sequence button is clicked, the final sequence is displayed i.e. the one with minimum disagreement value: R7, R5, R4, R1, R2, R6, R3.

After the mutation and crossover are applied, again the disagreement values are calculated and the final priority sequence is displayed i.e. the one having minimum disagreement value. Using this technique does not require any human input whenever the ties occur as in [2] because the ties were occurring in 2 due to incomplete relationship between the requirements, whereas in the proposed approach the input to the second level of prioritization is a complete sequence in itself; so whenever the pairs are formed there is a relationship between each requirement.

6. CONCLUSION AND FUTURE SCOPE

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. The proposed approach performs a two level requirement prioritization where the first level is performed by the developer after gathering the requirements from different users. The second level prioritization involves the opinion of the customers or the users about the requirements associated with the particular product. This is an effective and easy to implement technique that considers various quality attributes to prioritize requirements by measuring the quality of each requirement and also is user friendly as their opinion is given importance while deciding the final prioritization sequence. The approach is simple to use and implement and can be extended to include additional attributes as well as the number of user’s whose opinions are taken. Moreover, it can be further enhanced to work on private cloud environment where several users are furnishing their requirements from several different locations. In future, it may also include the principles of other software engineering techniques like the estimation of different software metrics like Effort, Time, People, Cost etc apart from quality attributes.

REFERENCES


