Software Requirement Prioritization using Machine Learning

Deepali Singh
Department of Computer Engineering and Applications
GLA University
Mathura, India
deeplali.panwar@gla.ac.in

Ashish Sharma
Department of Computer Engineering and Applications
GLA University
Mathura, India
ashish.sharma@gla.ac.in

Abstract—Requirement engineering plays a very important role in software development life cycle (SDLC). Generally, software projects suffer with the problem of various types and categories of requirements and are also delimited by constraints like time and budget. To deal with this type of requirement complexity, project managers need to prioritize the requirements of the proposed software effectively. To decide about prioritization and consideration of a set of requirements is a strategic concern. This process is known as requirements prioritization. This paper proposes a new approach to requirements prioritization called Gradient Descent Ranking (GDRank), which combines project's stakeholders preferences with Functional and Non-Functional requirements, their ordering and approximations are estimated through machine learning techniques. For validation purpose the proposal is compared with various other prominent requirement prioritization methods.

Keywords—Quality Function Deployment (QFD), Requirements Prioritization, Machine Learning

I. INTRODUCTION AND RELATED WORK

Fred Brooks [1] once said, “The hardest single part of building a software system is deciding what to build…. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later”. Hence, in order to develop quality and cost-effective software, it is imperative to provide prioritization to customer’s requirements to select the best possible set of requirements from a set of all requirements [2]. The software product quality is often determined by the capability to satisfy the necessities of the customers and users [3, 4].

Fundamentally, Functional and Non-Functional requirements are diverse in nature. The functional requirements illustrate the activities of the system as it relates to the system's functionality. It describes what a software system be supposed to do [5]. Whereas Non-Functional requirements place restriction on how the system will do so. Non-Functional requirements are habitually known as quality attributes in the field of software architecture. Quality attributes and functionality is orthogonal [6]. In this paper we study how to achieve a requirement prioritization while trying to assure both quality attributes and functional requirements. The purpose of this paper is to present a novel idea for Requirement Prioritization using machine learning technique. We proposed a method called Gradient - Descent Ranking (GDRank). It acquires a priority elicitation process which is very flexible. Further, a comprehensive literature survey of leading papers related to Requirement Prioritization is presented in this section and is shown in Table I. All approaches of requirement prioritization discussed in [7, 11, 13, 14, 15, 16, 17, 18, 19, 20] works on assignment of rank to requirements set for a given candidate requirement set.

<table>
<thead>
<tr>
<th>Requirement Prioritization Methods</th>
<th>Criteria</th>
<th>Rank Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality Function Deployment [7]</td>
<td>Customer voice is translated into engineer voice</td>
<td>Numerical Assignment</td>
</tr>
<tr>
<td>Planning Game [18]</td>
<td>Focus on Customer Preferences and Development Time</td>
<td>Numerical Assignment and Basic Ranking</td>
</tr>
<tr>
<td>Triage [14]</td>
<td>No. of Criteria e.g. Business Target, Development Time</td>
<td>Numerical Assignment and Basic Ranking</td>
</tr>
<tr>
<td>Fairness Analysis [17]</td>
<td>Focus on Stakeholders Objectives and Goals</td>
<td>Pareto Optimal Search Based Software Engineering</td>
</tr>
<tr>
<td>Power [16]</td>
<td>Focus on Stakeholders Objectives and Target</td>
<td>Basic Ranking</td>
</tr>
<tr>
<td>Win-Win Approach [13]</td>
<td>No. of Criteria e.g. Business Value, Development Efforts</td>
<td>AHP Approach</td>
</tr>
<tr>
<td>Watershed Method [19]</td>
<td>Based on Implementation Cost and Risk, and Value for the Customer</td>
<td>Numerical Assignment and Method Rules</td>
</tr>
<tr>
<td>CRB Rank [20]</td>
<td>Based on Domain Adaptive</td>
<td>RankBoost Machine Learning Technique</td>
</tr>
<tr>
<td>GDRank</td>
<td>Based on Domain Adaptive</td>
<td>Gradient Machine Learning Technique</td>
</tr>
</tbody>
</table>

II. PROPOSED METHOD

The proposed GDRank method sets on an outline first commences in [12] that prop up in making decisions for ordering a set of items, like software requirements or product features. An iterative prioritization method is proposed by this
framework that can handle different ordering criteria as well as decisions by individual or group of decision makers (domain experts).

A. Concept

We take a finite collection of Functional Requirements \( \text{FReq} = \{ f_1^+, \ldots, f_n^+ \} \) and Non-Functional Requirements \( \text{NReq} = \{ f_1^-, \ldots, f_m^- \} \) that is to be ranked and for that we describe the Universe of a set of requirement pairs \( \mathcal{U} = \{ f_i^+, f_j^- \}; i < j \}. \) We entitle the organized relation among two requirements that can be obtained from stakeholder priority. It is then formally define in terms of the function \( \theta(f_i^+, f_j^-) \), where \( \theta: \mathcal{U} \rightarrow \{-1, 0, 1\} \), with the following denotation for its assessments: \( \theta(f_i^+, f_j^-) \) is equal to -1, if \( f_j^- < f_i^+ \); is equal to 1, if \( f_i^+ < f_j^- \) and is equal to 0, when there is no preference of order between \( f_i^+ \) and \( f_j^- \).

An Unordered Requirements couple is a couple of requirements \( (f_i^+, f_j^-) \) for which priority is not known yet as the domain expert has not assigned a priority yet.

B. The Process of Prioritization

The GDRank prioritization procedure infuses human behavior with machine calculation. The procedure is drafted in Fig. 4, where five footsteps are characterized as rectangles. The common artifacts in effort and production are: the set of Requirements (FReq and NReq), the set of Ranking Functions (F), the Priorities by experts (\( \xi_T \), encoding the requirement characteristics, and the Approximated Rank (\( \hat{H}_T \)) where \( T \) corresponds to the last process iteration or the Final Approximated Rank (H).

The process is based on five steps which are as follows:

1. Requirement Elicitation. We consider requirement Elicitation by Quality Function Deployment (QFD) approach. Translation of subjective quality criteria into quantifiable and measureable objectives that can be used in designing & manufacturing the product is the main Goal of QFD. In this “The voice of the customer translated into the voice of the engineer [7].” QFD procedure includes putting together a "House of Quality"[8] like one shown in Fig. 1, which is for the advancement of a climbing harness [9].

2. Balancing Functional and Non-Functional Requirement. Balancing of Functional and Non-Functional Requirements is done with the help of Pattern Driven Architectural Partitioning (PDAP) [10]. Generally, choices over architecture that satisfy a functional requirement may clash with a quality attribute, or reverse. Such sides effects ought to be take care by product architect and proper balancing decisions should be made. Architectural pattern helps us to avoid such problem to a certain level.

3. Pair Sampling. A set of Sampled Requirements Pairs is selected from the set of Requirements (Functional and Non-Functional) whose relative preference is unknown. Pair Sampling is done by the help of Analytical Hierarchy Process (AHP) [11].

Step1. Place n Functional Requirements in the rows and m Non-Functional Requirements in columns of an n x m matrix. We will assume four Functional Requirements: FReq1 to FReq4, and four Non-Functional Requirements: NReq1 to NReq4.

Step2. Carry out pair wise comparisons of all the Functional and Non-Functional Requirements. For this purpose the primary scale used is shown in Table II. Insert determined relative intensity value for each set of pair of requirements in the position (FReq1, NReq2) where the row of FReq1 meets the column of NReq2. Insert the reciprocal value in position (FReq2, NReq1) and insert “1” in all positions in the main diagonal. Thus, it look like as shown in Fig. 2:

Step3. To represent the criterion distribution, guess the eigenvalues of the matrix with the help of performing averaging over normalized columns as shown in Fig. 3:

![Figure 1. House of Quality defined by QFD [9].](image1)

![Figure 2. Requirement in Matrix form.](image2)

![Figure 3.](image3)
Afterward divide each row sum with the number of requirements to normalize the sum of the rows. This calculation results in estimation of Eigen values of matrix and is referred to as the priority matrix.

\[
\frac{1.05}{1.98} = 1.26, \quad \frac{0.34}{0.62} = 0.56, \quad \frac{0.09}{0.16} = 0.56
\]

Step 4. Based on the estimated Eigen values, each requirement gets its relative value.

4. Priority Elicitation. The input is taken by the Pair Sampling step as a collection of Sampled Requirements Pairs produced and produces output on the basis of the Priorities stated by a domain expert (or decision makers) as a set of Ordered Requirements Pairs.

5. Priority Learning. Given a set of Ranking Function and partial elicited stakeholder priority, an approximation rank of the known preferences is produced by the learning algorithm and then the Final corresponding Approximated Rank for the requirement.

The output gives an estimate of the exact ranking, known as Approximated Rank, and may turn out to be the input for a further iteration of the process. The iteration stops when the result of the learning step is regarded as accurate, and the output is considered as the Final Approximated Rank.

<table>
<thead>
<tr>
<th>TABLE II. SCALE FOR PAIRWISE COMPARISONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative intensity</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>2, 4, 6, 8</td>
</tr>
<tr>
<td>Reciprocals</td>
</tr>
</tbody>
</table>

C. The Priority Learning Technique

This step yields an approximation of a preference structure, adapting the Ranknet approach described in [12]. The Ranknet method is capable of producing more accurate rank forecasts. Next, we provide an instinctive description of the Ranknet approach by drawing the Ranknet algorithm, displayed as pseudo code in Algorithm 1.

Algorithm 1. A sketch of the Ranknet algorithm
Input:

\[
\text{Req}' = \{r_1, \ldots, r_n\}
\]

The set of elicited Requirements

\[
F = \{f_1, \ldots, f_k\}
\]

Partial orders defining constraints and priorities upon Req'

\[
\mathcal{E} = \{(r_i, r_j); i < j \mid \theta(f_i, f_j) \neq 0\}
\]

A subsample of obtained pair - wise preferences

Output:

\[
\text{H}(r) \quad (H: \text{Req}' \rightarrow \text{IR})
\]

A ranking function described upon Req'

Begin

1. \(X = \text{initialize} (\mathcal{E})\)

   Weighting of obtained pairs \(\mathcal{E}\)

2. \(T = \text{Maximum Number Of Cycles}\)

   Number of learning cycles

3. For \(i = 1\) to \(T\)

4. \(w_i = \text{learn} w\) that minimizes the square error

\[
E[w] \geq \frac{1}{2} \sum_{d \in D} (t - o)^2
\]

Where \(w\) is weight, \(E\) is error value, \(o\) is the output of the linear unit and \(t\) is the training example

5. Calculate the gradient of \(E\) with respect to the vector \(\{w_0, \ldots, w_n\}\)

\[
\nabla E[w] \approx \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots \frac{\partial E}{\partial w_n}
\]

6. \(w = w + \Delta w\)

Here \(\eta\) is the learning rate which is positive constant, which find out the step size in the gradient descent search.

7. End For

8. \(H(r) = \text{synthesis of ranking function}\)

\[
\sum_{k=0}^{K} w_k x_k
\]

9. return \(H(r)\)

End

The GDRank algorithm performs \(T\) cycles in which the input is the set of ranking function \(F\) and the output is Final Approximated Rank.

III. RESULTS

The Result is generated set of requirements, which are obtained on the basis of graph plotted between the no. of requirements and the accuracy in rank shown in Fig.5. In GDRank approach, exact requirements are elicited with the help of QFD approach. We consider the functional and Non - functional requirements of the obtained requirements and balance them. Therefore the ranks which are learned from this approach are more accurate as compared to the previous two approaches. Finally, in Fig. 6 the graph which is plotted between the no. of requirements and the error rate, describes the gap between the previous rank and the current rank.
IV. CONCLUSION AND FUTURE WORK

In this paper, we present a complement description of the proposed GDRank procedure for requirements prioritization. The GDRank is used to learn the ranks of the ordered sets of requirements which provide the optimal error rates. The GDRank procedure has been sited with respect to the other methods which are available for requirements prioritizations, with particular reference to AHP approach and CBRank approach. Some assumptions are made in this paper that the requirements are well elicited and no requirements are further going to be added. The potential advantage of GDRank method is the ranking with respect to Functional and Non-Functional Requirements. But on the other hand, as the set of requirements increases, then the efforts needed by the human evaluators when pair preferences are obtained grows rapidly. In future, the issues like requirement dependencies, renewing requirements rank when new requirements are included and when the view of the expert changes and Scalability must be further considered.

REFERENCES