A Survey of Requirement Prioritization Methods

Gurkiran Kaur
Computer Science and Engineering Department
Thapar University, India

Seema Bawa
Computer Science and Engineering Department
Thapar University, India

Abstract

This paper presents result of a systematic review of literature of certain effective prioritization techniques. Seven prioritization strategies have been reported: Analytic Hierarchy Process, Value Oriented Prioritization, Cumulative Voting, Numerical Assignment Technique, Binary Search Tree, Planning Game and B tree prioritization. A comparative study of these methods is also presented.

Keywords*:* Requirements Prioritization, Requirement Analysis, Prioritization Techniques.

1. Introduction

When requirements are elicited, it often yields more requirements than can be implemented and deployed at once. The requirements need to be prioritized so that the most significant ones are implemented in the earliest product releases [1]. During a project, decision makers in software development need to make many different decisions regarding the release plan. Many issues such as available resources, milestones, conflicting stakeholder views, available market opportunity, risks, product strategies, and costs need to be taken into consideration when planning future releases. Unfortunately, there is a lack of simple and effective techniques for requirement's prioritization, which could be used for release planning [2].

2. Prioritization techniques

This section describes the prioritization techniques examined.

2.1 Numerical Assignment Technique

The numeral assignment technique is also called top ten prioritization technique and is based on the attitude that each requirement is assigned a figure representing the requirement's professed importance [3]. This approach is common in Quality Function Deployment where prioritizing of candidate requirements is required [14]. Several variants on the numeral assignment technique exist. A clear-cut approach to the method is presented by Brackett, who advises that requirements should be arranged as mandatory, desirable, or inessential [16]. The techniques allocate each requirement a number on a scale of 1 to 5, where the numbers indicate:

I. Does not matter.
II. Not important.
III. Rather important.
IV. Very important
V. Mandatory [10][18].

2.2 Analytical Hierarchical Process

The Analytic Hierarchy Process (AHP) was introduced by Saaty [4] in 1980. In AHP the candidate requirements are compared pair wise and to which degree one of the requirements are more important than the other requirement. Saaty [4] explains that intensity of importance should be according to the Table 1.Regnell et al [5] state that even though this is a good technique with many advantages like reliability it has major disadvantage of not being able to cater with environment having multiple stakeholders, hence it has to be modified in one way or another. The problem hasn't been solved yet[20].

Since this technique prescribes pair-wise comparisons of all candidate requirements, the number of comparisons grows up to polynomial. For a software system with *n* candidate requirements, *n. (n* - 1)/2 pair-wise comparisons are needed [10] [13]- [15] [19].

Iě

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.

2.3 Value Oriented Prioritization

Value Oriented Prioritization (VOP) uses a framework that gives requirement engineers a foundation for prioritizing and making decision about requirements. It provides visibility for all stakeholders during decision making, eliminating arguments over individual requirements. It emphasizes the core business values. The first step in setting up a value oriented prioritization process is to set up a structure for identifying the business's core values and the relative relationships among those values [14]. VOP uses the relationships that exist between core business values to assess and prioritize requirements and ensure their traceability. Company professional identify the core business values and use a simple ordinal scale to weight them according to their importance to the organization [10] [12]. The core business values as identified by Jim Azar are sales, customer satisfaction, marketing, strategic and integrity. VBSE applies to all phases of a software engineering activity, from assessing stakeholders' value propositions during requirements engineering through prioritizing those propositions during quality assurance. VOP connects the VBSE approach directly to requirements engineering [6].

2.4 Cumulative Voting

The Cumulative Voting(CV) is also called 100-Point Method or Hundred-Dollar test. It is presented by Leffingwell and Widrig, is a simple, basic and spontaneously attractive voting scheme where each stakeholder is given a constant amount i.e. 100 of imaginary units that he or she can use for voting in favour of the most important issues [7]. In this way, the amount of money assigned to an issue represents the respondent's relative preference (and therefore prioritization) in relation to the other issues. The points can be distributed in any way that the stakeholder desires. Each stakeholder is free to put the whole amount given to him or her on only one issue of dominating importance [14]. It is also possible for a stakeholder to distribute equally the amount to many of, or even to all of the issues.

The procedure may result to issues that are assigned zero units showing that the specific stakeholder considers these issues completely unimportant [15]. The zeros are generally a problem in this kind of data but the overall belief of CV is to allow stakeholders to spread freely their total amount without further restrictions[10][21][18].

2.5 Binary Search Tree

Binary Search Tree (BST) is an algorithmetic method with the purpose to store information, which then could be retrieved or searched after. The Binary search tree usually is either empty, or has one or two child nodes. The child nodes to the right have greater value than the root node, and the child nodes to the left have less value then the root node. Each child node is in itself a root node to its child node. If a node does not have any child nodes, it is called a leaf. This makes it possible to search in the BST recursively. The benefit for using BST, when prioritizing requirements, is that with n requirements, it takes only $n \log n$ [8] comparisons until all the requirements have been inserted in order. That makes BST a fast candidate, which could be good if there is a lot of requirement to prioritize among, i.e. BST could easily scale up to thousands of requirements, and still be a very fast candidate. There is one important thing to know about the BST algorithm, which is that a tree needs to be balanced to have the shortest insertion time [15].

A balanced BST is a BST where no leaf is more than a certain amount farther from the root than any other leaf. After a node has been inserted or deleted the tree might have to be rebalanced if but only if the BST would reach an unbalanced stated. The reason for this is that the insertion of a node should be most favourable, i.e. $\log n$ [18].

The problem with binary search tree is that it can be know which requirement is more favourable but the extent to which the requirement is important cannot be known and therefore the comparison is just ordinal[10][14].

2.6 Btree prioritization

Btree (BT) 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 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 [15].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 [18] . 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 [14].

2.7 Planning Game

In Planning Game (PG) requirements are written by the customer on a card. Then the customer divides the cards that are the requirements into three different piles. As Beck presented, the piles should have the names; "those without which the system will not function", "those that are less essential but provide significant business value" and "those that would be nice to have" [9]. The programmer is also involved who takes into account the time each requirement would take to implement and then he sorts the requirements into three different piles i.e. sort by risk, with the names; "those that can be estimated precisely", "those that can be estimated reasonably well" and "those that cannot be estimated at all" [14].

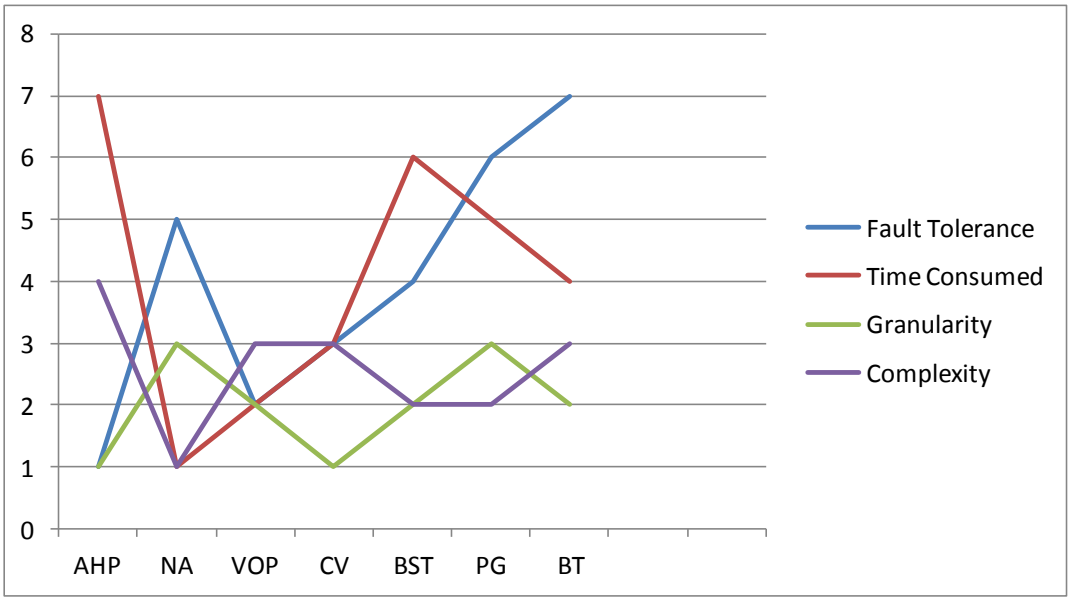
The customers then decide which requirement should be included in a particular release. Since planning game takes one requirement and then decides which pile the requirement belongs to and each requirement is not being compared to any other requirement, the time to prioritize *n* requirements is *n* comparisons. This means that PG is very flexible and can scale up to rather high numbers of requirements, without taking too long time to prioritize them all[10][15].

3. Comparison of requirement prioritization methods

Table 2 summarizes the seven prioritization techniques based on scale, fault tolerance(ordinal scale 1-7), granularity, complexity and time consumed (ordinal Scale 1-7).

ı ĕ

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



Iě ě

4. Conclusion:

In this paper, an overview of existing prioritization techniques is presented and then analyzed in the light of software requirement prioritization. These techniques can be categorized as Manual vs. algorithmic processes, easy to complex regarding complexity, Fine to Coarse in terms of Granularity, Ratio and ordinal in terms of scale. However, this area is still juvenile. More work need to be done in order to improve the current state of research and to improve the effectiveness of the prioritization strategies.

5. References

[1] J. Siddiqi and M.C. Shekaran, "Requirements engineering: the emerging wisdom," IEEE Software 1996.

[2] J. Karlsson and K. Ryan, "Prioritizing requirements using a cost-value approach," IEEE Software 1997.

[3] L. Sullivan, Quality function deployment: A system to assure that customer needs derive the product design and production process, Quality Progress.

[4] T.L. Saaty, The Analytic Hierarchy Process: Planning, Priority Setting, Resources, Allocation, McGraw-Hill, Inc. 1980.

[5] J. Karlsson, C. Wohlin and B. Regnell, "An evaluation of methods for prioritizing software requirements," Information and Software Technology 1998.

[6] J. Azar, R. K. Smith and D. Cordes, "Value Oriented Requirements Prioritization in a Small Development Organization," IEEE Software 2007.

[7] D. Leffingwell and D. Widrig, Managing Software Requirements: A Use Case Approach, 2nd ed., Addison-Wesley, Boston, USA, 2003.

[8] T. Standish, Data Structures in Java, Addison-Wesley, Boston, USA, 1997.

[9] K. Beck, Extreme programming: explained, 7th ed., Addison-Wesley, Boston, USA, 2001.

[10] Patrik Berander and Anneliese Andrews, "All about requirement prioritization," Springer.

[11] Md. Rizwan Beg 1, Qamar Abbas 2, Ravi Prakash Verma , "An Approach for Requirement Prioritization using B-Tree," IEEE Software 2008.

[12] Andrea Herrmann, Maya Daneva ," Requirements Prioritization Based on Benefit and Cost Prediction:An Agenda for Future Research"IEEE Software 2008.

[13] Mohd. Sadiq ,"An Approach for Eliciting Software Requirements and its Prioritization using Analytic Hierarchy Process" IEEE Sofware 2009.

[14] Muhammad Aasem, Muhammad Ramzan, Arfan Jaffar ,"Analysis and optimization of software requirements prioritization techniques" IEEE Software 2010.

[15] Azeem Ahmad, Aamir Shahzad, V. Kumar Padmanabhuni, Ali Mansoor, Sushma Joseph , Zaki Arshad," Requirements Prioritization with Respect to Geographically Distributed Stakeholders" IEEE software 2011.

[16] Richard Berntsson Svensson1, Tony Gorschek2, Björn Regnell1, Richard Torkar2, Ali Shahrokni3, Robert Feldt3, Aybuke Aurum4 "Prioritization of Quality Requirements: State of Practice in Eleven Companies"IEEE Software 2011.

[17] Nilofar Mulla "Comparison of various Elicitation Techniques and Requirement Prioritisation Techniques" IJERT May 2012 .

[18] Joachim Karlssona,b,*, Claes Wohlinb, Bjo¨rn Regnellc "An evaluation of methods for prioritizing software requirements" ELSEVIER 1998.

[19] "Analytical Hierarchy Process " Pearson education limited 2008.

[20] Azeem Ahmad, Magnus Goransson, Aamir Shahzad ," Limitations of the Analytic Hierarchy Process Technique with Respect to Geographically Distributed Stakeholders" World Academy of Science, Engineering and Technology 2010.

[21] Panagiota Chatzipetrou, Per Rovegard "Prioritization of Issues and Requirements by Cumulative Voting: A Compositional Data Analysis Framework" 2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications.