International Journal of Research and Development in Applied Science and Engineering (IJRDASE)
ISSN: 2454-6844

Parameter Optimization of Software Requirement by Using Fuzzy Algebra

Anika Bisht, Madan Kushwaha
Computer Science and Engineering
Bansal institute of engineering and technology, Lucknow, UP
anika.bisht95@gmail.com

Abstract: In this paper, we have presented a novel multi-level value based intelligent requirement prioritization technique using fuzzy logic and as a facilitating process, we have redefined the "value" of software to better meet its objectives. We have introduced and applied the concept of requirement value to prioritize requirements. We have performed extensive experimentation using our proposed technique along with existing techniques. The experiments have also shown that proposed technique is capable of delivering impressive prioritization under varying and often conflicting circumstances.

Keywords: Fuzzy Logic, Requirement Prioritization, Requirements engineering, RCF

1. Introduction:

Requirements engineering is a critical stage in the development of software because at this stage the purpose, functionality, and boundaries of the software are supposed to be fully identified, analyzed and defined. It has also been identified that most of the software projects fails to meet the real need are related to requirements engineering areas like capturing, analyzing, specifying, and managing requirements. In some life cycle models [11], feasibility study is the initial activity in the requirement engineering process that results in a feasibility report. If the development of the product is recommended by feasibility report, then requirement analysis can begin. In case of requirement analysis preceding feasibility studies we can expect an outside the box thinking. However, in such a scenario, feasibility should be determined before requirements are finalized.

Requirement engineering can be viewed as process of effectively finding and specifying objectives and purposed of the proposed solution. Zave [4] has defined RE in the following words:

"Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families."

Software Test & Evaluation Panel (STEP) defines requirement engineering [9] as:

"The disciplined application of scientific principles and techniques for developing, communicating, and managing requirements"

Loucopoulos and Champion [13] define requirements engineering as:

"The systematic process of developing requirements through an iterative process of analyzing a problem, documenting the resulting observations, and checking the accuracy of the understanding gained"

Requirement engineering according to Laplante [14] is "A subdiscipline of systems engineering and software engineering that is concerned with determining the goals, functions, and constraints of hardware and software systems"

In their work, Elizabeth Hull et al [10] define requirement engineering as

"A subset of system engineering concerned with discovering, development, tracing, analyzing, qualifying, communicating and managing requirements that define the system at successive levels of abstraction"

All these definitions mentioned above state the position of RE as a solid element in the software engineering elements that has a major contribution in achieving the real-world goals. Moreover, these refer RE a precise specification that establishes proper framework for requirement analysis, definition, validation and verification. The definitions, particularly one, given by Elizabeth Hull [10] also ensures that certain real life facts such as the always evolving nature of requirements and the need to reuse partial specification, as engineers often do in other branches of engineering. It is actually the same unique characteristic of requirements mentioned by Somerville in his work [9] where he states that

"The RE process varies immensely depending on the type of application being developed, the size and culture of the companies involved, and the software acquisition processes used"

2. Related Work:

AHP is a relative evaluation based measurable system to organize prerequisites for programming items. In

the event that we have n number of necessities, AHP makes n x (n-1)/2 correlations at every pecking order level. All things considered, we are normally working with prerequisites which have numerous goals. AHP fills in as an effective method in these sorts of circumstances by making pair savvy correlation to ascertain relative esteem and cost of every necessity against the other one. This altogether expansive number of examinations makes the strategy less compelling as increment in number of correlations dependably happens at the rate of O(n2). AHP is viewed as a five stage strategy.

1. Establish completeness of requirements.
2. Apply the pair-wise comparison method to assess the relative value.
3. Apply the pair-wise comparison method to assess the relative cost
4. Calculate each candidate requirement's relative value and implementations cost, and plot each on a cost-value diagram.
5. Use the cost-value diagram as a map for analyzing the candidate requirement

Countless have been made in later past to decide the adequacy of AHP for prerequisites prioritization.

Karlsson [1] has made various studies which have demonstrated the viability of this method in modern settings. In the meantime, some different studies [3] have discussed AHP as being troublesome, less effective and tedious. AHP can be considered as a profoundly refined and complex procedure which can build up prioritization at the level of individual necessities. Endeavors have been made to diminish the quantity of examinations. In any case, this has constantly improved the room for give and take. As we would like to think, this tradeoff is essential since a few correlations might very be required.

This method is more suitable in nature where a solitary partner is included. On the off chance that there are n number of prerequisites, these necessities are positioned from 1(most huge) to n (slightest noteworthy). This positioning is select in its temperament on the grounds that prerequisites are not positioned with respect to different necessities similar to the instance of AHP or total voting. Different methods like air pocket sort, brisk sort or twofold inquiry procedures can be utilized to accomplish this positioning. There are two noteworthy downsides connected with this procedure. To start with significant issue is that it can bring about a greater number of contentions than assentions when connected in a domain of numerous partners. The second downside is that necessities are seen and positioned in segregation. The effect of one necessity over the other doesn't assume any part in general prioritization. Since necessities can have different measurements to them so scientists have conceived an instrument of joining these measurements and ascertaining a mean need for every prerequisite [2]. This adjustment has its own particular confinements and additionally has been appeared in [5].

3. Methodology:

Programming Engineering (SE) goes for making programming items or their relics in a manner that these meet the prerequisites postured by partners while satisfying quality requirements forced on them. Keeping in mind the end goal to meet both these destinations, any product advancement infers its motivation and significance from the necessities postured by different partners. Prerequisite Engineering is a set up area of information inside of programming designing which sets up practices and standards for powerful necessity elicitation, displaying, detail, documentation and so forth. One imperative however regularly ignored routine of programming necessity building is prerequisite prioritization. A few necessity prioritization procedures have been introduced by creators. These systems are both quantitative and subjective in their tendency. Some understood prerequisite prioritization procedures incorporate Analytical Hierarchy Process (AHP), Cumulative Voting, Numerical Assignment, Ranking, Theory W, Requirement Triage, Wieger's Method and so forth. What's more, there are a few different methods which we should examine in this paper. Necessity prioritization empowers us to comprehend the importance of prerequisites opposite the framework to be created and among prerequisites also. With necessity prioritization, we can distinguish the center zones which require the vast majority of our consideration so as to build up an item which ideally meets the prerequisites of the partners. In the vast majority of the circumstances, because of spending plan and time limitations, it gets to be difficult to actualize every one of the prerequisites postured by partners. Likewise the way of numerous undertakings is such that necessities are actualized in an organized situation. In both of these situations, we require prerequisite prioritization [1]. We can organize prerequisite to acknowledge which necessities can be deferred or changed so that other earnest necessities can be actualized and to what degree. We can likewise utilize necessity prioritization to figure out which prerequisites to be actualized in before stages or later stages. We have been working with a few supported undertakings amid our examination. These ventures are confronted with both of the aforementioned circumstances. We have discovered it critical to organize necessities in their actual sense

keeping in mind the end goal to build up a significant and fruitful item.

Requirement prioritization was another practice in our particular advancement environment. In this way, the nature of our work obliged us to concentrate further into different prerequisites prioritization strategies with the goal that we can choose one which can best suit our unconventional improvement environment. Our finding was that there is serious lack of any trial results to figure out which strategy to lean toward. Hence, amid this time of innovative work, we concentrated on different prerequisite prioritization strategies and attempted to execute them on test level at different ventures. We soon understood that these procedures functioned admirably inside of specific circumstances however had some inalienable issues joined with them which made it difficult to actualize any of these over the association for every single diverse sort of tasks. The fundamental deterrents confronted by us while executing these methods were identified with expense, time and treatment of developing and inching prerequisites. Keeping in mind the end goal to beat these issues, one arrangement before us was to add to a falsely wise master driven necessity prioritization procedure. In one of our past works, we had exhibited the introductory portrayal of a "quality based prerequisites prioritization" system [23]. This procedure was all that much like Theory W. In this method the end clients and specialists were requested that organize their necessities based upon the worth that achievement of this prerequisite may have for the framework. The notable component of this procedure was an amalgamation of end clients and specialists during the time spent necessity prioritization. In any case, while actualizing this strategy, we experienced two noteworthy issues.

- The procedure created a considerable measure of contentions toward the end of prerequisite prioritization process. Struggle determination was a long and tedious procedure which should have been be taken after toward the end of each prioritization session.
- The strategy was totally manual. The prioritization was done through human try and component of human inclination was discernible.

While applying the strategy proposed in [23], we understood the requirement for a mechanized prerequisite prioritization method so as to overcome both the aforementioned constraints. In the ensuing productions, we chipped away at refining and upgrading this wise necessity prioritization approach [26].

3.1 Fuzzy Logic based Requirement Prioritization:

Following steps are executed in this third and final level of prioritization:

In the first and second level of prioritization, we achieve prioritization from the perspective of stakeholders and experts. However, both these steps involve extensive human input which can make the results more error prone. In order to further strengthen our prioritization results and reduce the manual nature of results, we make use of fuzzy logic for third level prioritization. In this approach requirement prioritization is modeled in the form of fuzzy rules. Based on Mamdani method, the approach is described using the following algorithm:

Start
Define Fuzzy Variables
Determine fuzzy variables, Requirement value, Stakeholder Priority and Requirement Priority
Establish fuzzy sets for these variables
Fuzzify each value in fuzzy sets using membership function
Generate knowledge Base using fuzzy rules
Build the system
Execute the system
Give input variable values
Get rule strength
Combine rule strength with output membership function
Find consequence of rules
Generate output
Combine consequences of variant rules
Generate output distribution by conflict resolution process
Defuzzify
Finish

In the first step, different variables (both input and consequent) for the system are
defined. For our problem we have two input variables namely, requirement value and stakeholder priority and one consequent namely requirement priority. Variable sets for all of these inputs and consequents are defined. . The values in these sets are fuzzified using appropriate membership function [16].

There are several fuzzy membership functions which are used in various problem environments. These include fuzzy centroid function, trapezoidal function, triangular function, bell shaped function etc. We have selected trapezoidal fuzzy membership function for our technique. Major reason to use this particular function is that in our particular problem situation, our maximum or minimum point is not just one value. Instead, several values can be at maximum position. Such a problem is best handled by trapezoidal function. Other functions such as bell shaped functions

have very little accommodation for maximum value or centroid has only one maximum value.
For example:

Rule1: if Requirement Value is very low and Stakeholder Priority is medium then Requirement Priority is low

Rule2: if Requirement Value is medium and Stakeholder Priority is low then Requirement Priority is medium

Rule3: if Requirement Value is high and Stakeholder Priority is medium then Requirement Priority is high

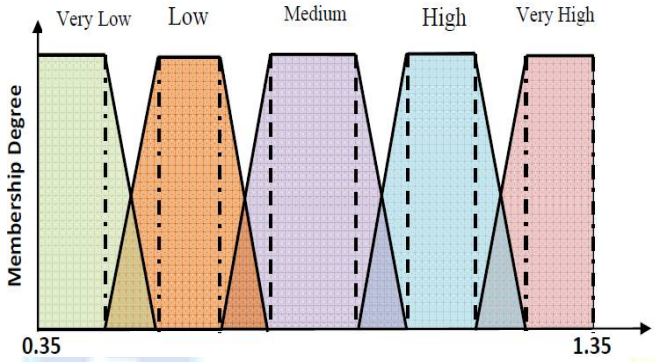

Fig 1. Fuzzy function distribution for VIRP

A complete set of such rules is generated which works as knowledge base for our system. To generate a prioritized list of requirements, each requirement is presented to the system which takes its requirement value and stakeholder priority as input and determines which rules to be fired. For each rule [5], the fuzzified inputs are combined to get the rule strength. These rule strengths are combined with the output membership function to find consequent fuzzified value for that rule using Max-Min method.
Once all fuzzified values for all fired rules have been determined, we get the defuzzified value of the consequent using the following equation:

$$y_i = \frac{\sum_{j=1}^{p} y_j \mu_A(y_j)}{\sum_{j=1}^{p} \mu_A(y_j)} \quad (1)$$

The purpose of defuzzification is to achieve a single crisp value from the fuzzified operations which involve several equations.

4. Result and Discussion:

We have given a brief overview of existing requirement prioritization techniques. We have described in detail the implementation of intelligent requirements prioritization technique. Several of these techniques are being applied in software industry for a while now. Some research studies and surveys to determine the usability of some of these techniques have also been conducted (as mentioned in previous section). So a significant mass of literature on these requirement prioritization techniques exists.

Table 1: Project Specific Requirement Specification Factors (pRCF)

Name	Description
Feasibility	The requirement is capable of being implemented within the constraints and resources
Modifiability	Requirement can undergo change to optimize the system without affecting the system adversely
Urgency	Degree of necessity of the requirement for system to be considered successful
Traceability	Requirement is such that subsequent function of the system can be traced to it. Requirements are less compound
Testability	Requirement can be tested and validated during testing phase. Independent test cases for the requirement can be generated.

However, no significant comparative study has so far appeared where all or most of the above mentioned techniques might have been applied to the same set of projects. This can be a very valuable study as it can determine that in what kind of development environment which specific requirement prioritization technique can yield best results. As we have already mentioned in the introduction section, we faced a severe problem of selecting suitable requirement prioritization technique for our projects. This search ultimately culminated in proposing and implementing our own intelligent requirement prioritization technique. We feel that it is need of the hour to catalogue the pros and cons of all existing as well as proposed technique at one place so that it becomes easier for software engineering community to evaluate and select one technique which better meets its needs.

Table 2: Requirement Specific Requirement Specification Factors (rRCF)

Name	Description
Completeness	The requirement statement has

	enough information to proceed to the nest development phase
Consistency	Requirement specifications use standard terminology and there are minimum conflicts due to statement and specifications
Understandability	Requirements are easy to describe and review. Requirements are grammatically correct with single and clear meaning
Within Scope	Requirement does not envisage something which is not described in original statement of scope
Non-Redundant	Requirement is not duplicated in complete or partially.

This section is dedicated to theoretical evaluation of requirement prioritization techniques. It is further subdivided into three subsections. In the first section, we have presented a theoretical evaluation of existing requirement prioritization techniques. In the second part, we have given a brief introduction of our intelligent requirement prioritization technique. In the last subsection, we have presented experimental results for our analysis of this technique with all existing ones.

5. Conclusion:

Requirement prioritization is one important activity of requirement engineering phase in software development. There are various requirement prioritization techniques in literature and practice. However, no significant comparative evaluation of these techniques has been made so far. In this work a new intelligent requirement prioritization technique has been proposed and described. This new technique for requirement prioritization is based on fuzzy logic and Bayesian network is a multilevel approach. In this technique, stakeholders, experts and fuzzy logic based system perform separate prioritizations. A comparative analysis based on experimental results conducted on several projects has also been presented. This analysis shows that in almost all different environments, intelligent requirement prioritization is able to exhibit better and impressive results. To extend this work towards classification of the prioritized requirements so that it can automatically classify requirements as critical, essential, peripheral etc. Work can also be done in such a way that prioritized requirements can be classified as non-negotiable and negotiable requirements. We are also developing Bayesian networks to perform classification. Using Bayesian networks can make this system highly evolvable and self-optimizing. In this work we propose a hybrid approach to optimize the requirement elicitation process. Work can also be done in such a way that prioritized requirements can be classified as non-negotiable and negotiable requirements. This new technique for requirement prioritization and classification is based on fuzzy logic. By this approach prioritized requirements can be classified as non-negotiable and negotiable requirements.

References:

[1] F. Brooks, No silver bullet: Essence and accidents of software engineering, IEEE Computer, vol.20, no.4, pp.10-19, 1987.
[2] J. Karlsson and K. Ryan, Supporting the selection of software requirements, Proc. of the 8th International Workshop on Software Speci_cation and Design, 1996.
[3] J. Karlsson and K. Ryan, A cost-value approach for prioritizing requirements, IEEE Software, vol.14, no.5, pp.67-75, 1997.
[4] X. Liu, C. C. Veera, Y. Sun, K. Noguchi and Y. Kyoya, Priority assessment of software requirements from multiple perspectives, Computer Software and Applications Conference, vol.1, pp.410-415, 2004.
[5] L. Fellows and I. Hooks, A case for priority classifying requirements, The 3rd International Conference on Requirements Engineering, pp.62-65, 1998.
[6] E. Yourdon, Death March Projects, Prentice Hall, 1997.
[7] M. Lubars, C. Potts and C. Richter, A review of the state of the practice in requirements modeling, Proc. of the IEEE Intemational Symposium of Requirements Engineering, pp.2-14, 1993.
[8] V. Ahl, An Experimental Comparison of Five Prioritization Methods, Master Thesis, School of Engineering, Blekinge Institute of Technology, Ronneby, Sweden, 2005.
[9] T. Saaty, The Analytic Hierarchy Process: Planning, Priority Setting, Resource, Allocation McGraw-Hill, New York, 1980.
[10] T. L. Saaty and G. Hu, Ranking by eigenvector versus other methods in the analytic hierarchy process, Applied Mathematical Letter, vol.11, no.4, pp.121-125, 1998.
[11] F. Hartwich, Weighting of agricultural research results: Strength and limitations of the analytic hierarchy process, Research in Development Economic and Policy, Discussion Paper, Grauer Verlag, Stuttgart, no.9, 1999.
[12] F. Hivert, J. Novelli and J. Thibon, The algebra of binary search tree, Theoretical Computer Science, vol.339, no.1, pp.3-10, 2005.
[13] L. Xiang, K. Ushijiam, T. Zhao, T. Zhang and C. Tang, O(1) time algorithm on BSR for constructing a random binary search tree, Proc. of the 4th International Conference on Parallel and Distributed Computing, Applications and Technologies, pp.599-602, 2003.
[14] I. Al-furaih, S. Aluru, S. Goil and S. Ranka, Parallel construction of multidimension binary search tree, IEEE

Trans. on Parallel and Distributed Systems, vol.11, no.2, pp.136-148, 2000.
[15] C. Lee, L. Hung, M. Chang, C. Shen and C. Tang, An improved algorithm for the maximum agreement subtree problem, Information Processing Letters, vol.94, no.5, pp.211-216, 2005.