International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 5 Issue 9|| September 2016 || PP. 38-43

Different Approaches of Software Requirement Prioritization

Mulugu.Narendhar [1], Dr.K.Anuradha[2]
[1]*Associate Professor & HOD, Scient Institute of Technology, CSE Dept, Hyderabad, TS, India,*
[2]*Professor & HOD, Griet, CSE Dept, Hyderabad, TS, India.*

Abstract: *In software system development, it can be a challenge for people to select the 'right' requirement among several or many options if it is not obvious which requirement is desirable. Requirements prioritization helps people to discover the most desirable requirements. It seems that most requirements prioritization techniques work well on a small number of requirements, but many of them have constraints on medium to large numbers of requirements. Requirement prioritization process is used to determine which candidate requirement of a software project should be included in a certain release, for this purpose different techniques are used. These techniques use different approaches and consider different factors for prioritization e.g. cost, value, risk, benefit etc.*
Keywords*: scalability, requirements prioritization, MoScow, AHP, Minimal spanning tree.*

I. INTRODUCTION

Requirements prioritization is an important activity in software development. Usually, the number of requirements from the customers exceeds the number of features that can be implemented within the given time and available resources. For that reason, some of the requested features will not be completed or they are moved to later releases. Therefore, the customer and the development teams must decide what is the most essential functionality which should be implemented as early as possible. In other words, the stakeholders should prioritize the requirements. There are numerous different techniques presented in the literature how to prioritize requirements. It might be difficult to pick the most suitable method because of the large number of them. Some methods are more time consuming than others but provide more accurate results. Some methods scale well to be used with larger number of requirements but provide very coarse results. In other words, none of the techniques can really be considered the best one but a practitioner must pick a technique that is the most suitable for his situation, for example, in terms of scalability, accuracy and time consumption.

Generally, not all the requirements contain equal user satisfaction. It is often not obvious which requirement contains high user satisfaction among hundreds or thousands of requirements. When only one stakeholder is involved in the project, it is relatively easy to make decisions since only one stakeholder's opinion needs to be considered. When more than one stakeholder is involved in the project, decisions can be harder to make, since different stakeholders have different perspectives. For example, project developers look for the requirements which can be implemented fast, financial managers look for the requirements with low cost, market managers look for the requirements with high market value, and end users look for the requirements which are easy use. One requirement may be of low cost, with short implementation time, but also 12 have low market value and be hard to use. Conversely, another requirement may have a high cost, but short time to be implemented, high market value and be easy to use. It can be a challenge for stakeholders to decide which requirements need to be implemented first. Requirements prioritization is a technique that can uncover the most important requirements to maximize the stakeholders' satisfaction.

II. ASPECTS OF REQUIREMENTS PRIORITIZATION

Berander and Andrews (2005) define an aspect as a property or attribute that can be used to prioritize requirements. Synonymous words used by other authors are "factor" (Henry & Henry, 1993) and "criteria" (Hatton, 2007). Requirements can be prioritized based on different aspects such as importance, time, cost, penalty, and risk. Some aspects are introduced below

I**mportance**

Stakeholders can prioritize requirements to find out which requirement is most important to them. However, the word "importance" can be a multifaceted concept which may have different meanings to different people. For example, importance could mean high market value, high quality of the product, or urgency of implementation among other things.. It is essential to specify the meaning of "importance" first to reduce the possibility of confusion when letting the stakeholders prioritize the requirements.

Time
Time can be the time spent on successfully implementing the candidate requirement. Wiegers (1999) notes time can also be influenced by other factors such as degree of parallelism in development, or staff training time.

Cost
Cost can be the money spent on successfully implementing the candidate requirement. Cost can be directly influenced by staff hours. Cost can also be influenced by other factors such as the extra resources needed in order to implement the requirements.

Penalty
Penalty is how much needs to be paid if a requirement is not fulfilled. Penalty is an important aspect that needs to be evaluated. Sometimes requirements may have low values, but failing to fulfill these requirements may cause a high penalty.

Risk
Mustafa and Al-Bahar (1991) define risk as the degree of likelihood that a project will fail to achieve its time, cost or quality goals. Each project contains a certain amount of risk. Boehm (1991) notes the risk contained in project development could be things such as personnel shortfalls, unrealistic schedules and budgets, developing the wrong functions and properties, continuing stream of requirements changes, or gold plating (adding more functionality or features than is necessary). Aurum and Wohlin (2003) say that the risk of each requirement of a project can be estimated in order to get an estimation of the risk level of the project. Risk can be prioritized to find which requirement contains the lowest risk. 14 Other aspects which can be considered include volatility, strategic benefit, market value, and available resources. Stakeholders can prioritize requirements based on a single or multiple aspects. Generally, prioritizing requirements based on a single aspect is easier than prioritizing requirements on multiple aspects. For example, when only considering a single aspect such as market value, a requirement that contains a high market value will have a high priority. But when considering more aspects such as cost, a high market value requirement may involve high cost. In this case, stakeholders may change their minds, and the high priority requirement may become a low priority requirement. Ruhe, Eberlein and Pfahl (2003) note that aspects are often not independent of each other, and may interact with each other (for example, high quality may require high cost). Change in one aspect may result in a change in another aspect. Since each aspect may have an influence on the degree of successful of the final product, it is essential to consider multiple aspects in order to increase the degree of success of the final product. Several aspects can be considered when prioritizing requirements, but generally it is not practical to consider all the aspects. Which aspects should be considered depends on the real situation.

III. TECHNIQUES OF REQUIREMENTS PRIORITIZATION

Several basic techniques on how to prioritize requirements are introduced. The prioritization techniques can be categorized as nominal scale, ordinal scale, and ratio scale.

Nominal Scale
For nominal scale methods, requirements are assigned to different priority groups, with all requirements in one priority group being of equal priority.

Numerical assignment
It is a simple requirements prioritization technique based on grouping requirements into different priority groups. The number of priority groups can vary, but three is common. For example, requirements can be grouped as "critical", "standard", and "optional". The results of numerical assignment are on a nominal scale. All requirements contained in one priority group represent equal priority. No further information shows that one requirement is of higher or lower priority than another requirement within one priority group.

MoScoW
MoScoW is a kind of numerical assignment and it is mentioned by DSDM Consortium (2009), Hatton (2007, 2008) and Tudor and Walter (2006). MoScoW currently 17 incorporates into the software development methodology DSDM (Dynamic Systems Development Method). The idea of MoScoW is that it groups all requirements into four priority groups "MUST have", "SHOULD have", "COULD have", and "WON'T have".
• "MUST have" means that requirements in this group must be contained in the project. Failure to deliver these requirements means the entire project would be a failure.
• "SHOULD have" means that the project would be nice if it contains the requirements in this group.

• "COULD have" also means that the project would be nice if it contains these requirements. But these requirements are less important than the requirements in the "SHOULD have" group.
• "WON'T have" is like a "wish list". It means that the requirements in this group are good requirements but they will not be implemented in the current stage. They may be implemented in the next release.
The results of MoScoW are on a nominal scale. All requirements contained in one priority group represent equal priority. No further information shows one requirement is of higher or lower priority than another requirement within one priority group.

Ordinal Scale
Ordinal scale methods result in an ordered list of requirements.

Simple ranking
Ranking elements is quite intuitive for most people as it can happen in people's lives. Berander and Andrews (2005) and Hatton (2008) mention simple ranking is that n requirements are simply ranked from 1…n, with the most important requirement ranked 1 and the least important requirement ranked n. This is a common requirements prioritization technique based on an ordinal scale.

Bubble sort
Bubble sort is mentioned by Aho, Hopcroft and Ullman (1983). It is a method for sorting elements. Karlsson et al. (1998) introduce this technique to the requirements prioritization area for ranking requirements. The idea of the bubble sort method for sorting requirements is that the users compare two requirements at a time and swap 18 them if the two requirements are in the wrong order. The comparisons continue until no more swaps are needed. The result of bubble sort is a list of ranked requirements. The average and worst case complexity for bubble sort is O (n^2).

Binary search tree
Another method for sorting elements that is mentioned by Aho et al. (1983) is binary search tree. A binary search tree is a tree in which each node contains at most two children. Karlsson et al. (1998) introduce this technique to the requirements prioritization area for ranking requirements. The idea of the binary search tree method for ranking requirements is that each node represents a requirement, all requirements placed in the left subtree of a node are of lower priority than the node priority, and all requirements placed in the right subtree of a node are of higher priority than that node priority. When performing the binary search tree method, first choose one requirement to be the top node. Then, select one unsorted requirement to compare with the top node. If that requirement is of lower priority than the top node, it searches the left subtree, but if that requirement is of higher priority than the top node, it searches the right subtree. The process is repeated until no further node needs to be compared and at that time the requirement can be inserted into the right position. The average complexity for binary search tree is O (n log n). Simple ranking, bubble sort and binary search tree methods are all used for ranking requirements. Simple ranking method is quite intuitive for people, bubble sort and binary search tree methods seem harder for people to use to rank requirements. One question which may come out is that if people can do simple ranking easily, why are bubble sort and binary search tree methods needed? The answer is that when there are a fairly small number of requirements needing to be prioritized, simple ranking seems easy for people to perform. But as the number of requirements increases, people may have difficulty remembering all the requirements. Psychology research (reviewed by Miller (1956)) shows that people have difficulty remembering more than seven (plus or minus two) elements. Hatton (2007) said that it would be difficult and probably incorrect for people to use the simple ranking method to rank 15 or more elements. If a large number of requirements need to be ranked, in order to get a high degree of accuracy, bubble sort and binary search tree seem more suitable than simple ranking.

Ratio Scale
The results of ratio scale methods can provide the relative difference between requirements.

Hundred Dollar Method
Hundred dollar method (also called cumulative voting) which is mentioned by Berander and Andrews (2005) and Hatton (2008) is a simple method for prioritizing requirements. The idea of the hundred dollar method is that each stakeholder is asked to assume he/she has $100 to distribute to the requirements. The result is presented on a ratio scale. The ratio scale result can provide the information on how much one requirement is more/less important than another one.

AHP

Another well-known prioritization technique based on ratio scale results is called Analytic Hierarchy Process (AHP). AHP is developed by Saaty and it is designed for complex decision making. The idea of AHP is that it compares all possible pairs of hierarchical requirements to determine the priority. When using AHP, the user first identifies the attributes and alternatives for each requirement and uses them to build a hierarchy. Then the user specifies his/her preference to each pair of the attributes by assigning a preference scale which is generally 1 to 9, where 1 indicates equal value and 9 indicates extreme value. The scale is shown in Table 3. After that AHP converts the user's evaluations to numerical values and a numerical priority is derived for each element of the hierarchy. Note that a redundancy might exist when using the AHP method to prioritize requirements, therefore a consistency ratio should be calculated after using the AHP method to judge if the prioritization is valid. If n requirements need to be prioritized, n*(n-1)/2 pair-wise comparisons are required when using the AHP method. Therefore the complexity of AHP is O (n^2).

Empirical studies performed by Karlsson and Ryan (1997) and Karlsson et al. (1998) show that AHP is time consuming. Some techniques try to reduce the number of comparisons in order to reduce the time consumed. Hierarchy AHP and minimal spanning tree have been developed for that purpose.

Hierarchy AHP

Davis notes that in large projects, requirements are often structured in a hierarchy, with the generalized requirements placed at the top of the hierarchy and the more specific requirements placed at the lower levels of the hierarchy. Hierarchy AHP, which is introduced by Karlsson et al. uses the AHP method to prioritize requirements only at the same level of hierarchy. This method can reduce the number of decisions compared with the AHP method, since not all the requirements are compared pair-wise. This can reduce the number of redundant comparisons, but the trade-off is that the ability to identify inconsistent judgments is also reduced.

Minimal Spanning Tree

Minimal spanning tree is another prioritization method which is introduced by Karlsson et al. The idea of minimal spanning tree method is that if the decisions are made perfectly consistent, the redundancy will not exist, and in this case the number of comparisons will reduce to only n-1 comparisons (n is the number of requirements). A minimal spanning tree constructs unique pairs of requirements. It is a directed graph which is minimally connected. Minimal spanning tree can reduce the number of pair wise comparisons dramatically compared with AHP. However, the ability to identify inconsistent judgments is low.

Cost-Value Approach

Karlsson and Ryan provide a method which is called the Cost-Value approach for prioritizing requirements. The basic idea of the Cost-Value approach is that each individual requirement is determined on two aspects: the value to the users and the cost of implementing the requirement. It uses the AHP technique to compare requirements pair-wise according to the relative values and costs. Empirical studies performed by Karlsson and Ryan show that the Cost-Value approach is time consuming.

IV. TECHNIQUES EVALUATION

The following section presents the findings from previous authors' evaluations of some prioritization techniques.

The Evaluation of AHP, Hierarchy AHP, Spanning Tree, Bubble Sort and Binary Search Tree Methods Karlsson et al. (1998) perform an experimental study to evaluate six prioritization methods: analytic hierarchy process (AHP), hierarchy AHP, spanning tree matrix, bubble sort, binary search tree, and priority groups. The priority groups method was not introduced in section 2.2 because it is different from general grouping requirements. For more details on priority groups method and why it was not introduced, please refer to Appendix A. Karlsson et al. use 13 quality requirements to evaluate the six prioritization methods. These requirements are prioritized by three authors independently. Each prioritization method is assigned randomly to each author. Each author only studies one method per week in order to minimize the risk of remembering the priorities of the requirements. Only importance to customers is considered as an aspect when prioritizing requirements. Two kinds of measurements are evaluated: objective measures and subjective measures. An ordinal scale from 1 to 6 is used for the measurement, where 1 indicates the best and 6 indicates the worst. The evaluation results are shown in Table 1 and Table 2. The objective measures are: required number of decisions, total time consumption, and time consumption per decision.

Required number of decisions measures the total number of decisions which need to be made for each prioritization method. The numbers of decisions for the first four methods are pre-defined. The numbers of decisions for the last two methods are counted by each author (therefore three numbers are presented in the result shown in Table 1.

• Total time consumption compares the time spent to complete the prioritization process among the six methods.

• Time consumption per decision compares the average time spent on each decision for each prioritization method among the six methods.

The subjective measures are: ease of use, reliability of results, and fault tolerance.
• Ease of use measures how easy a particular prioritization method is to use.

Table1-1: Objective measures (Karlsson et al.,)

Evaluation Criteria	AHP	Hierarchy AHP	Spanning Tree	BubbleSort	BinarySearch
Required number of decisions	78	26	12	78	29,33,38
Total time consumption(ordinal scale 1-6)	6	2	1	3	5
•Time consumption per decision(ordinal scale 1-6)	2	4	5	1	6

Table -2: Subjective measures (Karlsson et al., 1)

Evaluation Criteria	AHP	Hierarchy AHP	Spanning Tree	BubbleSort	BinarySearch
Ease of use	3	4	2	1	5
Reliability	1	3	6	2	4
Fault tolerance	1	3	6	2	4

From the results provided in Table 1 and Table 2, it is seen that AHP can provide the most reliable result of the six methods, but it requires the largest number of decisions and the longest time consumption. Minimum spanning tree involves the smallest number of decisions and the shortest amount of time consumption, but it provides the least reliable result and the lowest fault tolerance. Bubble sort is the easiest method to use and it can provide relatively reliable results and relatively good fault tolerance, but it involves the largest number of decisions (same as AHP). Hierarchy AHP and binary search tree reside in the middle. They produce less reliable results than AHP and bubble sort, but also take fewer decisions and less time to perform than AHP and bubble sort. It is seen that no one prioritization method is perfect among these six methods. Minimum spanning tree requires less effort and time to perform the prioritization process, but it contains a high risk of misdirecting project resources and time since it provides low reliable results. AHP and bubble sort methods can provide reliable results, but they need large amounts of effort and time to perform. When dealing with a small number of requirements, the amount of effort and time spent can be relatively small. But since the complexity of the AHP and bubble sort methods is high (both are $O(n^2)$), when dealing with large numbers of requirements the amount of effort and time spent may become unmanageable. Karlsson et al. also admit that AHP and bubble sort both contain a scale up problem. Among the six prioritization methods, it seems that no method is perfect for large numbers of requirements (AHP and bubble sort contain a scale up problem, and other methods contain a certain degree of accuracy problems).The Evaluation of Simple Ranking, MoSCoW, AHP and Hundred Dollar Methods.

Hatton (2007) conducts a case study to examine four prioritization methods: simple ranking, MoSCoW, AHP and Hundred dollar method. Each method is examined by three criteria: ease of use, the time to complete the prioritizing process and the user's confidence (user's confidence here means how deeply the user believes the prioritization result actually reflects his/her real priority). Twelve requirements related to mobile phone features are provided to the users. Each user is asked to use each method to prioritize requirements. Each user is also asked to record the time taken to use each method to perform the prioritization, the difficulty of each prioritization method, and the user's confidence rate for each method. A wide range of people from different ages, genders, levels of education and occupations are selected to be participants. Thirty one studies are completed and the results are used for data analysis. The time taken for each method is recorded by letting the user record the time he/she started and finished each prioritization process. An ordinal scale from 1 to 10 is used for the measurement of difficulty of the method, where 1 indicates "very easy" and 10 indicates "very difficult". The research findings are shown in Table 3

Table 3: Times taken (minutes), median confidence and median difficulty (1-10 Scale)

	Minimum time	Maximum Time	Mean Time	Standard Deviation	Median Confidence	Median Difficulty
MoScow	1	5	1.78	1.083	8	2
Simple Ranking	1	4	1.5	0.73	8	3
$100	1	8	3.6	2.42	7	4
AHP	7	22	14.03	4.4	2	9

From the results in Table 3, it is seen that AHP contains the longest completion time range of the four methods. For the mean time values, AHP takes much longer time to complete than any of the other methods.

V. CONCLUSION

It is almost impossible to implement all the requirements in one release. That is why some sort of prioritization process is needed to implement the most important requirements in the first release and leave the less important ones for the future releases. Requirements prioritization helps requirement engineers in this complex and crucial decision making situation. In this paper, we introduced eight basic requirements prioritization techniques: Simple ranking, Bubble sort, Binary search tree, Hundred Dollar Method, AHP, Hierarchy AHP, Cost-Value Approach, Minimal spanning tree .

VI. FUTURE WORK

During requirement prioritization process lots of factors has to be considered. E.g. cost, benefit, risk, effort, etc. there are lots of methods for prioritization but there isn't a single method that addresses all the factors necessary for prioritization process. Furthermore the prioritization process should not only considers the crucial factors but also the process must be easy to learn, easy to use, invisible to stakeholders to gain their interest and confidence, understandable to non-experts, reliable, efficient, scalable and flexible. Lots of factors can be added to this list. And most of the methods and techniques that are available now-a days are for small development organization. If the scale increases, these techniques get very difficult to implement and are less accurate. Hence lot of work has to be done in this field. Such techniques can be developed that uses all major factors listed above for prioritization, regardless of organization size, i.e. supports all size of development organizations.

REFERENCES

[1]. L. Lehtola, M. Kauppinen, and S. Kujala, "Requirements prioritization challenges in practice," in Proc. of 5th Intl Conf. On Product Focused Software Process Improvement (PROFES), 2004, pp. 497–508.

[2]. P. Berander, K. Khan, and L. Lehtola, "Towards a research framework on requirements prioritization," in SERPS06: Sixth Conference on Software Engineering Research and Practice in Sweden, 2006, pp. 39–48.

[3]. A. Perini, F. Ricca, A. Susi, and C. Bazzanella, "An empirical study to compare the accuracy of ahp and cbranking techniques for requirements prioritization," in CERE '07: Proceedings of the 2007 Fifth International Workshop on Comparative Evaluation in Requirements Engineering. Washington, DC, USA: IEEE Computer Society, 2007, pp. 23–35.

[4]. P. Berander and P. J¨onsson, "Hierarchical cumulative voting (hcv) prioritization of requirements in hierarchies," International Journal of Software Engineering & Knowledge Engineering, vol. 16, pp. 819–849, 2006.

[5]. P. Berander and M. Svahnberg, "Evaluating two ways of calculating priorities in requirements hierarchies - an experiment on hierarchical cumulative voting," J. Syst. Softw., vol. 82, no. 5, pp. 836–850, 2009.

[6]. J. Karlsson, "Software requirements prioritizing," in Requirements Engineering, 1996., Proceedings of the Second International Conference on, 15-18 1996, pp. 110 –116.

[7]. A. S. Danesh and R. Ahmad, "Study of prioritization techniques using students as subjects," in ICIME '09: Proceedings of the 2009 International Conference on Information Management and Engineering. Washington, DC, USA: IEEE Computer Society, 2009, pp. 390–394.

[8]. S. Hatton, "Early prioritisation of goals book series," in Advances in Conceptual Modeling Foundations and Applications, vol. 4802, 235- 244 2007, pp. 517 –526.

[9]. Leoucopoulos P. and Karakostas V. "System Requirements Engineering", Mc Graw-Hill, 1995

[10]. Martín A., Martínez C., Martínez Carod N., Aranda G., and Cechich A. "Classifying Groupware Tools to Improve Communication in Geographically Distributed Elicitation". IX Congreso Argentino en Ciencias de la Computación, CACIC 2003, La Plata, 6-10 Octubre 2003, (942-953).

[11]. Hui B., Lisakos S., and Mylopoulos J.. "Requirements Analysis for Customizable Software: A Goals-Skills-Preferences Framework". In Proceedings of the 11th IEEE International Requirements Engineering Conference, pages 117–126, 2003

[12]. Young R.. "Recommended Requirements Gathering Practices" . CroossTalk The Journal of Defense Software Engineering. April 2002. pag 9 -12 [13] Ruhe G., Ebertein A., Pfahl D. "Quantitative WinWin – A New Method for Decision Support in Requirements Negotiation". SEKE'02, Italy, July 2002. ACM.

[13]. S.E. Keller, L.G. Kahn, R.B. Panara, Specifying software qualityrequirements with metrics, in: R.H. Thayer and M. Dorfman (Eds.),

[14]. System and Software Requirements Engineering, 1990, pp. 145–163.

[15]. D.A. Garvin, Building a learning organization, Harvard Business

[16]. Review, July–August, 1993, pp. 78–91.

[17]. J. Karlsson, S. Olsson, K. Ryan, Improved practical support for largescale

[18]. requirements prioritizing, Requirements Eng. J. 2 (1) (1997)51–60.

[19]. P. Voola, A. Babu, Requirements uncertainty prioritization approach: a novel

[20]. approach for requirements prioritization, Softw. Eng.: Int. J. (SEIJ) 2 (2) (2012)37–49.

[21]. R. Thakurta, A framework for prioritization of quality requirements for inclusion in a software project, Softw. Qual. J. 21 (2012) 573–597.

[22]. M. Ramzan, A. Jaffar, A. Shahid, Value based intelligent requirement prioritization (VIRP): expert driven fuzzy logic based prioritization technique'', Int. J. Innovat. Comput. 7 (3) (2011) 1017–1038.

[23]. A. Perini, F. Ricca, A. Susi, Tool-supported requirements prioritization.

[24]. Comparing the AHP and CBRank method, Inf. Softw. Technol. 51 (2009) 1021–1032.

[25]. X. Liu, Y. Sun, C. Veera, Y. Kyoya, K. Noguchi, Priority assessment of software

[26]. process requirements from multiple perspectives, J. Syst. Softw. 79 (11)

[27]. (2006) 1649–1660.