A Comparison of Nine Basic Techniques for Requirements Prioritization

Mikko Vestola
Helsinki University of Technology
Email: mikko.vestola@tkk.fi

***Abstract*—Requirements prioritization is an important activity in software development. Numerous different techniques to prioritize requirements exist. In this paper, we focus on the following research questions: 1) What empirically studied requirements prioritization techniques are presented in the literature? 2) How easy to use, accurate and scalable are these techniques? Total of nine basic requirements prioritization techniques were identified: 1) Numeral assignment technique, 2) Analytic hierarchy process (AHP), 3) Hierarchy AHP, 4) Minimal spanning tree, 5) Cumulative voting (CV), 6) Hierarchical cumulative voting (HCV), 7) Priority groups, 8) Binary priority list (BPL), 9) Bubble sort. The techniques are presented and analyzed based on the empirical studies from the literature. The results indicate that none of the techniques can be considered the best one. The best prioritization technique depends on the situation (e.g. number of requirements used or desired results scale). This study also reveals that there are several problems with the existing empirical studies, which make it hard to compare their results to each other. Therefore, this paper also presents recommended practices for future empirical studies about requirements prioritization techniques.**

***Index Terms*—Requirements, prioritization, techniques, methods, approaches, empirical, comparison**

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 [1]. 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.

This paper helps to decide the right technique for prioritizing requirements by answering the following research questions:

1) What empirically studied requirement prioritization techniques are presented in the literature?
2) Based on the empirical studies, how did these techniques perform, especially in terms of easy of use, accuracy and scalability?

Hopefully, this paper helps to choose the right prioritization technique and also helps to develop completely new prioritization methods using these prioritization techniques as basic building blocks. In addition, the results of this paper also guide which techniques need more empirical studies.

The rest of this paper is organized as follows. In Section II, we [1] introduce the classification of requirements prioritization approaches. The following three sections, Section III, Section IV and Section V, present different requirements prioritization techniques classified based on the results they provide (nominal, ordinal or ratio scale) and also analyze how these techniques are empirically studied. Finally, Section VI concludes the paper.

II. Requirements prioritization levels

According to [2] and [3], requirements prioritization approaches can be classified into four different abstraction levels, namely: prioritization activities, techniques, methods and processes. Approaches on higher levels utilize lower level activities and techniques.

Requirements prioritization *activities* are the lowest level approaches. This level refers to the underlying primitive activities which are used to rank requirements. One basic approach is to separately assign a rank to each requirement according to a specific criterion (e.g. importance for the customer). Another way is that requirements are ranked by assigning a preference value to pairs of requirements (pair-wise comparison). A third approach is to group each requirement to one specific class from a number of predefined classes of importance.

Prioritization *techniques* use the results from the lower level prioritization activities and possibly do some calculations to compute the requirements ordering. Three common scales to present the results are: nominal scale, ordinal scale and ratio scale. For example, the Analytic Hierarchy Process (AHP) is an example of a prioritization technique. It utilizes pair-wise comparison activity to calculate the priorities for requirements.

[1] Use of the plural pronoun is customary even in solely authored research papers and thus is also used in this paper.

The final result from AHP is a list of requirements based on a ratio scale. In other words, the priorities calculated with AHP can answer the question: ”How much important is this one requirement when compared to some other?” Another example of prioritization technique is the Binary priority list (BPL) approach. It also uses pair-wise comparison activity but does not require so many pair-wise comparisons than AHP and produces a ranked list of requirements based on an ordinal scale. Unlike AHP, the results of BPL can not answer the above mentioned question. In other words, the BPL technique can only tell that one requirement is more important than another but not to what extent. As we can see, these two different prioritization techniques both utilize pair-wise comparison but they use it in different ways and thus produce very different results.

On the next level of requirements prioritization approaches are the requirements prioritization *methods*. These methods are usually more sophisticated than techniques and are developed specifically for requirements prioritization. While a technique often focuses on prioritizing based on one aspect (e.g importance for customer), a method usually utilizes more variables (e.g. importance, cost, dependencies). For example, the Cost-Value approach is one example of prioritization methods. It presents requirements in a two dimensional cost-value diagram. Both cost and value for each requirement are computed with the AHP technique. Other examples of prioritization methods are Planning game, EVOLVE, Quantitative Win-Win and Wiegers' method.

On the highest abstraction level are the prioritization *processes*. This refers to the description of steps what needs to be done in an organization to prioritize requirements. For example, what are the roles of different stakeholders, in which order things needs to be done and how requirements prioritization suits to the organization's software process.

In the following sections, we focus on prioritization techniques. These are the basic building blocks for more sophisticated prioritization methods. The user should be noted that, for convenience, we might use the terms *technique* and *method* interchangeably later in this paper, although they precisely mean different. The techniques presented at the following sections are classified based on the results they give: nominal scale, ratio scale and ordinal scale. Total of nine techniques are included: Numeral assignment technique, Analytic hierarchy process (AHP), Hierarchy AHP, Minimal spanning tree, Cumulative voting (CV), Hierarchical cumulative voting (HCV), Priority groups, Binary priority list (BPL) and Bubble sort.

III. Nominal scale prioritization techniques

Nominal scale prioritization techniques produce lists of categories to which objects can be classified. In other words, requirements are categorized into groups based on their importance. All requirements in one priority group have equal priority. One can not tell if some requirement is more or less important than another within one priority group. Numeral assignment technique is the only technique included in this category. The MoSCoW technique is also included, however, it is basically a numeral assignment technique and thus is not included as a separate subsection.

A. Numeral assignment technique

The numeral assignment technique is said to be the most common technique for prioritization of requirements [4], [5]. It is a simple technique based on grouping requirements into different priority groups. The number of priority groups may vary, however, three groups is perhaps the most common division. Priority groups may be just priority numbers from 1 to 3 or they can be labeled, for example, as ”high”, ”medium”, and ”low”. The result of this technique is a collection of requirements classified into different priority groups. All requirements in one priority group have equal priority. The MoSCoW technique is an example of numeral assignment technique. It defines four priority groups: ”MUST have”, ”SHOULD have”, ”COULD have” and ”WON'T have”. Requirements are assigned to these groups based on the importance of having them implemented.

Even though the numeral assignment technique is considered to be the most common technique for prioritization of requirements, there is actually little empirical studies about the effectiveness of the technique. Three studies were identified which involved prioritizing requirements using numeral assignment technique.

The first one is a study by Karlsson where he compares the numeral assignment approach (using scale ranging from 1 to 5) to AHP [6]. The number of requirements to prioritize was 14 which all were real requirements from a real project. Interestingly, the study shows that numeral assignment technique is slower than AHP. On average, it took about twice as much time to perform the prioritization with numeral assignment technique than with AHP. Moreover, the numeral assignment technique was also seen to be less informative and inaccurate when compared to AHP.

The second empirical study about numeral assignment technique was performed by Danesh and Ahmad [7]. The settings of the study were basically identical to the Karlsson's study above. Danesh and Ahmad conducted two studies which both prioritized nine requirements. The results were also quite identical to Karlsson's: pair-wise comparison technique was seen more accurate and faster than the numeral assignment technique.

Both studies clearly indicate that the numeral assignment technique is not effective when the number of requirements is low (say 20 or less). There seems to be better techniques, like AHP, which provide more accurate results faster. To mix up, Hatton provides opposite results in her study [8] where practitioners prioritized total of 12 requirements with the following three prioritization methods: MoSCoW, AHP and Cumulative voting (CV). On the contrary to the previously mentioned studies, in Hutton's paper, the MoSCoW technique is considered the best method. It was the easiest and fastest one and provided the highest user confidence. However, Hatton points out in [9] that MoSCoW is probably best used in the early stages of projects when requirements are not specified in

a great degree of detail. Instead, in the later stages of projects, when stakeholders have greater understanding of the system developed, other methods such as CV and AHP might be better.

To sum up, it seems that the numeral assignment technique might be better when prioritizing medium or large number of requirements. However, more empirical studies should be performed to confirm this assumption.

IV. Ratio scale prioritization techniques

Ratio scale prioritization techniques produce ranked lists of requirements. These techniques can answer the question: "How much important is this one requirement when compared to another?". In other words, these techniques can provide the relative difference between requirements. The following techniques are included in this category: Analytic hierarchy process (AHP), Hierarchy AHP, Minimal spanning tree, Cumulative voting (CV) and Hierarchical cumulative voting (HCV).

A. Analytic hierarchy process (AHP)

Definitely the most widely studied requirements prioritization technique is the Analytic hierarchy process (AHP), which was developed by Saaty [10] and applied to software engineering field by Karlsson [6]. AHP is also commonly known as the pair-wise comparison technique [4], [6], however, it is a somewhat misleading term because pair-wise is nothing unique to AHP but also other techniques use it, such as Bubble sort and Binary priority list.

The basic idea of AHP is to calculate the priorities of requirements by comparing all unique pairs of requirements to estimate their relative importance. In other words, the person performing the comparison has to decide which requirement is more important, and to what extent using a scale 1-9.

	SR-1	SR-2	SR-3	SR-4	SR-5
SR-1	1	8	1/5	3	1
SR-2	1/8	1	1/5	1/7	1/7
SR-3	5	5	1	1	2
SR-4	1/3	7	1	1	1/2
SR-5	1	7	1/2	2	1

Fig. 1. An example matrix created with AHP. The participants using AHP need only to fill the upper highlighted half of the matrix. Each requirement is equal value when compared to itself (thus value 1 in the diagonal).

The steps required to prioritize with AHP are the following (see Figure 1 for more explanation) [6] :

1) Create an $n \times n$ matrix (n is the number of requirements) and insert the n requirements in the rows and columns of the matrix.
2) For each unique pair of requirements, for example, A and B, insert their relative intensity of importance in the position where the row of A meets the column B. At the same time, the reciprocal values are inserted to the transposed positions (e.g. if cell AB=4 then cell BA=1/4).
3) Finally, to get the relative priority of each requirement, calculate the eigenvalues of the resulting comparison matrix. The final result is therefore the relative priorities of the requirements (e.g. A=50%, B=12% and C=38%).

AHP requires total of $n \times (n-1)/2$ comparisons. Because pair-wise comparisons in AHP produce much redundancy, AHP also provides means to check the accuracy of the comparisons by calculating the consistency ratio.

Karlsson et al. [11] compares AHP to five other prioritization techniques: hierarchy AHP, minimal spanning tree, binary search, bubble sort and priority groups. Total of 13 requirements were prioritized with all these methods by three persons. Although AHP was the slowest approach when considering total time spent in prioritizing, Karlsson et al. find AHP to be the most promising approach, mainly because they find it trustworthy and fault tolerant. It also includes a consistency check and, as a ratio scale technique, it provides more informative results than any other tested method. Still, AHP is relatively easy to use (not so easy but not so hard either). Similar results are described in [6] and [7].

On the contrary to the previously mentioned studies, two studies describe entirely different results. Ahl [12] compares five prioritization methods: AHP, CV, binary search tree, Planning Game and Planning Game combined with AHP. The study used students as subjects to prioritize 13 requirements. The results of the study indicate that AHP is the worst candidate. It is difficult to handle, not scalable and slow. Similar results are provided by Hatton [8]. In Hatton's study, MoSCoW, AHP and CV were compared and AHP was the hardest to use, took the longest time to perform and contained the lowest user confidence. Two other studies confirm the problems with user confidence. Studies by Lehtola et al. [13] and Lena Karlsson et al. [14] found out that the practitioners felt a loss of control over the prioritization process when they used AHP. Lehtola et al. also pointed out that users found it difficult to estimate how much more valuable one requirement is than another and also that the practitioners seemed to mistrust the results they got with AHP.

To sum up, all studies admit that AHP is not scalable as such. Therefore, it is not really suitable for anything else than prioritizing small number of requirements (say, less than 20). However, the results from the empirical studies are quite conflicting that is AHP even reasonable approach to prioritize small number of requirements or do other techniques provide better results with less work.

B. Hierarchy AHP

The Analytic hierarchy process (AHP) described at the previous section does not really scale well because of the high number of required pair-wise comparisons ($n \times (n-1)/2$). The number of pair-wise comparisons grows exponentially: we need "only" 45 comparisons with 10 requirements but with five times more requirements the number of comparisons is as much as 1225. Clearly, AHP as such is not a reasonable

technique for prioritizing large or even medium number of requirements. Therefore, Karlsson et al. introduced the hierarchy AHP technique [11]. The technique uses AHP to prioritize requirements only at the same level of hierarchy. This can dramatically reduce the number of required comparisons since not all the requirements are compared pair-wise. However, the trade-off is that the ability to identify errors is also reduced because of the reduced number of redundant comparisons.

Unlike AHP, hierarchy AHP has not really been empirically researched. The only study is the one that presented this technique [11]. In that paper, 13 requirements were prioritized with the following techniques: hierarchy AHP, AHP, minimal spanning tree, binary search, bubble sort and priority groups. Hierarchy AHP was the second fastest technique. However, it was not considered so easy to use and, moreover, not as much reliable and fault tolerance as plain AHP. Therefore, using AHP for small number of requirements is preferred. Although the hierarchy AHP approach is definitely intended to be used with tens or hundreds of requirements, no empirical studies show how well hierarchy AHP actually performs when dealing with large number of requirements.

C. Minimal spanning tree

Another prioritization technique introduced by Karlsson et al. [11] is the minimal spanning tree technique. As described at section IV-A, AHP requires quite a lot of pair-wise comparisons and contains much redundancy. For example, if requirement A is more important than B and B is more important than C, comparing A and C is redundant because we already know that A is also more important than C. This redundancy helps to identify judgment errors but also creates scalability issues. This problem is what the minimal spanning tree technique tries to solve. The basic idea of minimal spanning tree method is that all the redundant comparisons from AHP (e.g. comparing A to C in the previous example) are not performed at all. This will dramatically reduce the number of comparisons to only $n - 1$ when compared to $n * (n - 1)/2$ required by AHP. The required comparisons can be constructed by creating a minimal spanning tree from the requirements. This reduced number of comparisons is still enough to calculate the relative intensity of importance between the requirements. However, the ability to identify inconsistent judgments disappears.

Like hierarchy AHP, also the minimal spanning tree technique has been empirically studied only in the paper introducing this technique [11]. In that study, the minimal spanning tree approach was the fastest one, however, subjective measures from practitioners indicate that the technique provided the least reliable results and also fault tolerance was poor. Based on this one study, it seems that the minimal spanning tree technique is not the best method for prioritizing small number of requirements. It is a fast method, however, if reliability and fault tolerance are more important than time consumption, better methods exist such as AHP. Because of better scalability, the minimal spanning tree technique might be better in prioritizing large number of requirements. On the other hand, no empirical studies support this assumption.

D. Cumulative voting (CV)

Cumulative voting (also know as the 100-point method or the Hundred-Dollar Test [4]) is a simple method for prioritizing software requirements. The basic idea is that the stakeholders participating at prioritization are given a number of imaginary units (100 dollars, 1000 points, etc.) which are distributed among the requirements to prioritize. The number of units assigned to a requirement represents its priority. The results are presented on a ratio scale which provides the information on how much one requirement is more/less important than another.

Ahl [12] compares CV to four other prioritization methods: binary search tree, AHP, Planning Game and Planning Game combined with AHP. When prioritizing total of 13 requirements, the test subjects thought that CV was an easy method to use and also one of the most accurate methods. CV was also one of the fastest methods, however, it was believed to be a bad candidate for handling large number of requirements.

In another study [8], 12 requirements were prioritized with the following techniques: CV, MoSCoW and AHP. The results from the study show that CV took longer time than MoSCoW but was faster than AHP. In overall, the study shows that CV is relatively easy to use and contains high user confidence. However, also in this study, the practitioners felt that CV probably does not scale well for large number of requirements. Similar results were obtained in another study by Hatton [9].

In a study conducted by Regnell et al. [15], the stakeholders prioritized 58 requirements with imaginary $100,000. The study raised some concerns about CV. Firstly, the stakeholders might lose overview as the number of requirements increases. Secondly, CV might be sensitive to so called ”shrewd tactics”, which means that stakeholders distribute their points based on how they think others will do it in order to get high priorities to their favorite requirements.

To sum up, CV seems to be an eligible technique to prioritize small and medium number of requirements. On the other hand, there are some concerns that CV does not scale well. Because of the lack of empirical studies, it is hard to say how well CV actually performs with many requirements when compared to other prioritization techniques.

E. Hierarchical cumulative voting (HCV)

Hierarchical cumulative voting (HCV) was first introduced by Berander and Jönsson [4]. HCV was developed to answer the scalability issues in Cumulative voting (CV). The idea of HCV is basically the same as behind CV. In other words, like with CV, the prioritization with HCV is conducted by distributing points to requirements. However, when using HCV, not all requirements are prioritized at the same time. Instead, requirements are classified to different levels of hierarchies and the prioritization is performed only within each hierarchy level. Figure 2 shows an example how to use HCV. Instead of prioritizing all five low-level requirements at the same time like in CV, requirements are divided into two prioritization blocks. Moreover, only requirements within a prioritization block are prioritized together.

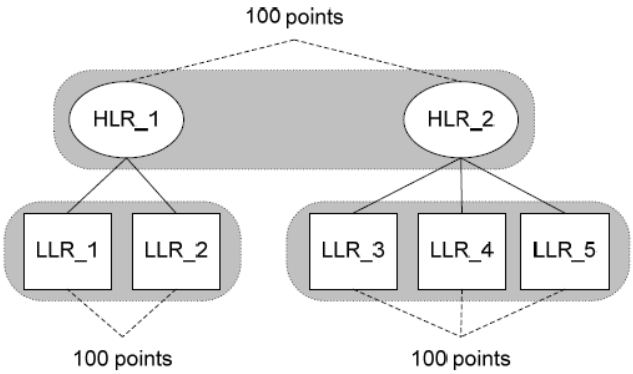

Fig. 2. An example how to use HCV. This example contains two levels of requirements: high-level requirements (HLR) and low-level requirements (LLR). Only requirements within a prioritization block (the grey areas) are prioritized together. [4]

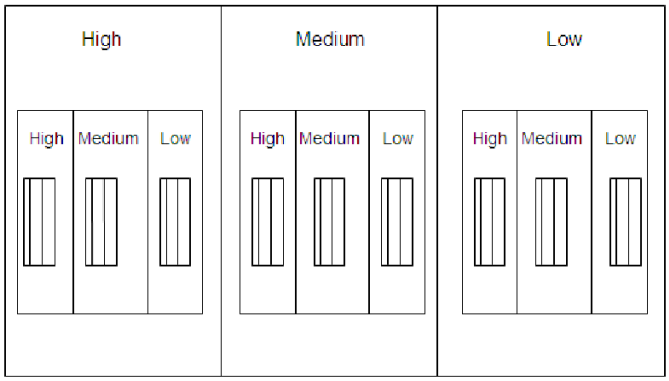

Fig. 3. Using priority groups. The original high, medium and low groups are further divided to subgroups [16].

Only one empirical study dealing with HCV was identified [5]. The study used master students to prioritize 27 requirements. In overall, HCV was seen to be reasonably easy to use and scalable. However, the main focus of the paper is on evaluating the effectiveness of HCV with and without a compensation factor. Therefore, the study does not really provide answers how well HCV performs when compared to other prioritization techniques, such as hierarchy AHP.

V. Ordinal scale prioritization techniques

Ordinal scale prioritization techniques produce ranked lists of requirements. Unlike ratio scale techniques, ordinal scale techniques can not answer the question ”How much important is this one requirement when compared to another?”. In other words, these techniques can only tell that one requirement is more important than another but not to what extent. The following techniques are included in this category: Priority groups, Binary priority list (BPL), Minimal spanning tree and Bubble sort.

A. Priority groups

The priority groups technique was first described by Karlsson et al. [11]. Despite the name of the technique, it does not actually produce groups of requirements as a final result. Instead, the outcome is a ranked list of requirements. The basic principle behind priority groups is the same as in numeral assignment technique: assign each requirement into one of the three groups: high, medium and low priority. However, while numeral assignment technique groups requirements into priority groups only once, priority groups technique does this repeatedly. Figure 3 shows the idea of priority groups.

The steps required to prioritize requirements using the priority groups approach are described below [11].

1) Gather all candidate requirements into one pile.
2) Put each requirement into one of the three groups: high, medium or low priority
3) In groups with more than one requirement, create three new subgroups (high, medium, low) and put the requirements within that group into these newly created subgroups.
4) Repeat step 3 recursively until there is only one requirement in each subgroup.
5) As presentation, just read the requirements from left to right.

The only empirical study about priority groups is the one conducted by Karlsson et al. [11] in which total of 13 requirements were prioritized with the following techniques: AHP, minimal spanning tree, binary search, bubble sort and priority groups. The study concludes that the priority groups technique is clearly the worst approach: it is quite slow to perform and hard to use. Moreover, the technique got clearly the lowest ranking when considering easy of use, reliability and fault tolerance. Based on this one study, the technique seems not to be suitable for prioritizing small number of requirements. Better methods exist, such as AHP or Bubble sort. However, one study is insufficient to draw any wide conclusions, especially how does the priority groups technique perform with large number of requirements.

B. Binary Priority List (BPL)

Binary Priority List (BPL) is a requirements prioritization technique described in [17]. It is basically the same as the Binary search tree (BST) approach which was introduced to the requirements prioritization area by Karlsson et al [11]. In this paper we use the term BPL instead of BST because the author of this paper believes that presenting the results in a horizontal binary priority list (see Figure 4) is easier to understand than using a vertical binary search tree (BST). For example, it is much easier to remember that high priority requirements go to top and low priority requirements go to bottom instead of right and left.

Applying the BPL technique to prioritize requirements consists of performing the following steps described in [17].

1) Collect all requirements to a single pile.

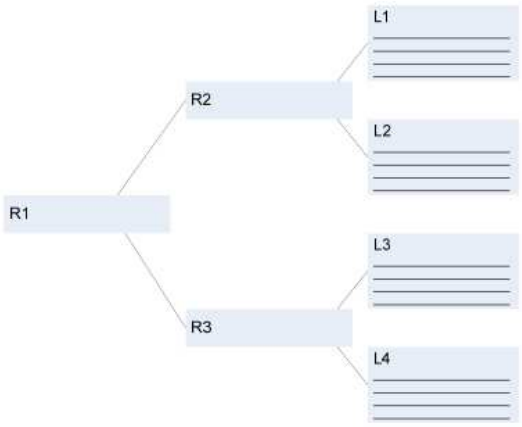

Fig. 4. Structure of Binary Priority List. For example, requirement R2 is more important than R1, R3, L3 and L4. [17]

2) Take any one of the requirements from the pile and put it as a root requirement.
3) Take another requirement and compare it to the root requirement.
4) If the requirement has a lower priority than the root requirement, compare it to the requirement below the root requirement. If the requirement has higher priority than the root requirement, compare it to the requirement above the root. This is continued until a place where there is no sub-requirement to compare with is encountered and the requirement can be finally placed to this position.
5) Steps 3 to 4 are repeated for all requirements.
6) At the end, traverse the list in top-down order to get the prioritized order of the requirements.

Total of three papers have empirically studied BPL. The first one is the often mentioned study by Karlsson et al. [11]. When comparing BPL, AHP, minimal spanning tree, bubble sort and priority groups, BPL scored relatively weak in terms of ease of use, time consumption, reliability and fault tolerance. After AHP, BPL was the second slowest method to prioritize with but did not provide as reliable and fault tolerance results as AHP.

Another already mentioned study [12], by Ahl, describes somewhat conflicting results. When comparing CV, BPL, AHP, Planning Game and Planning Game combined with AHP, Ahl comes to conclusion that BPL is the best out of the five techniques. It is one of the best methods to scale up, it is the easiest method to use and it gives the most accurate results. However, in the study, it was the second slowest method being only faster than AHP.

The latest empirical study about BPL is [17] in which BPL is compared to Wiegers' prioritization method based on the following three factors: time consumption, ease of use and subjective reliability of results. Two case studies were made where product managers prioritized 46 and 68 requirements with both methods. In both cases, BPL scored higher than Wiegers' method in all aspects of the comparison. BPL took only quarter to third of the time of Wiegers' method and was also easier to use.

Although the results from the three studies above are somewhat conflicting, BPL seems to be an eligible technique for prioritizing small and medium number of requirements. However, when dealing with a very large number of requirements, BPL might not be a feasible approach. For example, Bebensee expects that BPL is not practicable for numbers much more than 100 requirements [17].

C. Bubble sort

Bubble sort technique was first introduced by Karlsson et al. to the requirements prioritization area for ranking requirements [11]. Interestingly, the idea of bubble sort is closely related to AHP. Both AHP and bubble sort techniques utilize the pair-wise comparison activity. Moreover, both techniques require $n \times (n-1)/2$ pair-wise comparisons [11]. However, in bubble sort, the decision maker only has to determine which of the two requirements is more important, not to what extent like in AHP.

The steps required to prioritize requirements with bubble sort technique are the following [11]:

1) Outline the requirements in a vertical column.
2) Compare the top two requirements from the column with each other to determine which is the most important. If the lower requirement is more important than the top one, swap their positions.
3) Repeat this pairwise comparison and swapping for the second and third requirement, then third and fourth requirement and so on until the bottom of the column is reached.
4) If any of the requirements have been moved during steps 2 and 3, repeat the process for the whole column starting again from the top two requirements (step 2). Keep repeating this until no requirements are swapped during a complete pass through the column, which means that the requirements are now in priority order.

The result of the process is a ranked column of requirements where the most important requirement is at the top of the column and the least important one is at the bottom.

Bubble sort is yet another technique which is not empirically studied elsewhere than in the paper by Karlsson et al. [11]. In that study, bubble sort was actually one of the best methods when comparing time consumption and subjective measures. In terms of time consumption, bubble sort positioned at the middle being faster than AHP but slower than the minimal spanning tree technique. Bubble sort was the easiest method to use and provided both reliable and fault tolerant results.

For small number of requirements, bubble sort seems to be a viable alternative, however, Karlsson et al. point out that, like AHP, bubble sort technique might have difficulties to scale up. No empirical studies have tested bubble sort with medium or large number of requirements.

VI. Conclusions

In this paper, we introduced nine basic requirements prioritization techniques: Numeral assignment technique, Analytic hierarchy process (AHP), Hierarchy AHP, Minimal spanning tree, Cumulative voting (CV), Hierarchical cumulative voting (HCV), Priority groups, Binary priority list (BPL) and Bubble sort.

The summary of the techniques is presented in Table I. Based on the empirical evidence from the studies, the last column of the table describes the subjective opinion of how well the technique can prioritize different sizes of requirements sets. In other words, is the technique best suited for small (< 20), medium ($21 - 100$) or large (> 100) number of requirements.

TABLE I
Summary of presented requirements prioritization techniques.

Technique	Empirical studies	Result scale	Speed	Best suited for number of req.
Numeral assignment	[6], [8], [7]	Nominal	Average	Medium, Large
AHP	[6], [11], [13], [12], [8], [14], [7]	Ratio	Slow	Small
Hierarchy AHP	[11]	Ratio	Average	Medium, Large
Minimal spanning tree	[11]	Ratio	Fast	Medium, Large
CV	[15], [12], [8], [9]	Ratio	Fast	Small, Medium
HCV	[5]	Ratio	Fast	Large
Priority groups	[11]	Ordinal	Average	?
BPL	[11], [12], [17]	Ordinal	Average	Small, Medium
Bubble sort	[11]	Ordinal	Average	Small

Based on the empirical evidence, it is quite impossible to say, which of the methods is the best one. It really depends on the situation. For example, if you don't need to know the relative differences between requirements, using a ratio scale technique can be overkill and more simple methods, such as BPL might be enough. The most suitable method might also be a combination of techniques, such as first grouping requirements with numeral assignment technique and then ranking the most important ones with AHP.

Some of the studies provide conflicting results and some techniques are not studied empirically enough. The settings of the studies varied so much that it is hard to compare their results to each other. First of all, the studies used different sets of methods. There was only two studies ([11], [12]) which compared five or more techniques, others usually just two or three. Moreover, the variables used to compare the techniques were not always the same in all studies. For example, some of the studies did not measure scalability, others not accuracy.

Another problem is that almost all of the studies used a very small number of requirements (20 or less). None of the studies focused on large sets of requirements. Therefore, it is hard to say how the techniques perform in real-life projects which can have tens or even hundreds of requirements.

Further studies should focus on larger sets of requirements, say 50 or more. Furthermore, the studies should follow the research framework presented by Berander et al. [2]. The framework clearly describes the variables which should be used when comparing requirements prioritization techniques. By using this framework, it would be much easier to compare the results from the future studies.

It would be very interesting to see "the mother of all prioritization experiments" which would use the framework by Berander et al. to empirically study all the techniques presented in this paper. It would also be interesting to compare these basic single-criteria requirements prioritization techniques to more sophisticated prioritization methods (e.g. Wiegers' method or Cost-Value approach) to see how much more value do these multi-criteria methods actually provide.

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. Jönsson, "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] ——, "Choosing the right prioritisation method," in *Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on*, 26-28 2008, pp. 517 –526.

[10] T. Saaty, *The Analytic Hierarchy Process, Planning, Piority Setting, Resource Allocation*. New york: McGraw-Hill, 1980.

[11] J. Karlsson, C. Wohlin, and B. Regnell, "An evaluation of methods for prioritizing software requirements," *Information and Software Technology*, vol. 39, no. 14-15, pp. 939 – 947, 1998.

[12] V. Ahl, "An experimental comparison of five prioritization methods - investigating ease of use, accuracy and scalability," Master's thesis, Blekinge Institute of Technology, Ronneby, Sweden, 2005. [Online]. Available: http://www.bth.se/fou/cuppsats.nsf/all/86a759a57c335911c1257088005e42bc/$file/Master_thesis_Viggo_Ahl.pdf

[13] L. Lehtola and M. Kauppinen, "Empirical evaluation of two requirements prioritization methods in product development projects book series lecture notes in computer science." Springer Berlin / Heidelberg, 2004, pp. 161–170.

[14] L. Karlsson, T. Thelin, B. Regnell, P. Berander, and C. Wohlin, “Pair-wise comparisons versus planning game partitioning–experiments on requirements prioritisation techniques,” *Empirical Softw. Engg.*, vol. 12, no. 1, pp. 3–33, 2007.

[15] B. Regnell, M. Höst, J. N. och Dag, P. Beremark, and T. Hjelm, “An industrial case study on distributed prioritisation in market-driven requirements engineering for packaged software,” *Requirements Engineering*, vol. 6, no. 1, pp. 51–62, 2001.

[16] Q. Ma, “The effectiveness of requirements prioritization techniques for a medium to large number of requirements: a systematic literature review,” Master’s thesis, Auckland University of Technology, Auckland, New Zealand, 2009. [Online]. Available: http://hdl.handle.net/10292/833

[17] T. Bebensee, I. van de Weerd, and S. Brinkkemper, “Binary priority list for prioritizing software requirements,” 2010, unpublished. [Online]. Available: http://people.cs.uu.nl/sjaak/Papers/Software%20Product%20Management/ThomasBebensee/BPL_submitted.pdf