A Monte Carlo investigation of one approach to prioritization for a large number of software requirements

A Yu Timofeeva¹, M Sh Murtazina¹

¹Novosibirsk State Technical University, 20, Karla Marksa ave., Novosibirsk, 630073, Russia

E-mail: a.timofeeva@corp.nstu.ru

Abstract. The paper presents an approach to prioritization for a large number of software requirements. It selects certain requirements as benchmarks. The selection is based on the analysis of a pairwise comparison matrix. Several selection rules are proposed. A Monte Carlo simulation was conducted to study the effects of the benchmark selection rules on rank correlation between true and estimated priorities. The results of the simulation suggest that the best way to select one benchmark is the rule based on the most consistent judgements independent of experimental conditions. Several benchmarks must be chosen in another way. This problem will be the subject of future research.

1. Introduction

Requirement prioritization is a very important part of a requirements engineering process. To succeed in developing a software product an implementation of the most significant requirements must be ensured first. The prioritization of requirements becomes an extremely difficult task in the context of the implementation of complex software products. There are a large number of requirements for such products. In this case, the set of requirements is constantly growing. Stakeholders need to compare the requirements and determine the importance of some requirements relative to others in order to prioritize the requirements.

Currently there are a number of approaches to requirement prioritization. The most well-known methods of requirement prioritization are the Analytic Hierarchy Process (AHP), the Quality Function Deployment (QFD), the planning game, the Binary Search Tree (BST), the hundred dollar method [1-2]. At the same time, the analytic hierarchy process is the most widely studied technique for requirement prioritization. This method provides a reliable result due to the possibility of computing the consistency of judgements across all pairwise comparisons. However, this method becomes extremely labor-intensive for prioritizing a large number of requirements.

This paper presents an approach to prioritization for a large number of software requirements. This approach is based on the selection of benchmarks from a set of requirements based on optimal criteria. In Section 2, the possibilities and disadvantages for the AHP are considered. In Section 3, we define the problem of incomplete pairwise comparison matrices. In Section 4, we propose the benchmark selection rules. In section 5, we describe the simulation study assumptions and design. In Section 6, we analyse the performance of proposed benchmark selection rules. This will be followed by drawing conclusions and proposing directions for further research.
2. Analytic Hierarchy Process

The Analytic Hierarchy Process is a multi-criteria decision making method developed by Saaty [3]. It uses the pairwise comparison matrix \( A = [a_{ij}]_{1 \leq i, j \leq n} \) for computing the priority (weight) vector \( w \) for \( n \) requirements. An element \( a_{ij} \) is a number associated with expert judgment about importance of \( i \)-th requirement compared to \( j \)-th one. The Fundamental Scale [3] converts verbally judgments into integers from one to nine. The experts assign only \( n(n-1) \) comparisons, diagonal elements of the matrix \( A \) are equal to one, reciprocals of all assigned comparisons are entered in the transpose positions:

\[
a_{ij} = 1/a_{ji}.
\]  

(1)

When the judgements do not contradict each other, \( a_{ij}a_{jk} = a_{ik} \ \forall i, j, k \) holds and a normalized matrix \[ N = [n_{ij}]_{1 \leq i, j \leq n} , \text{ where } n_{ij} = a_{ij} \left( \sum_{k=1}^{n} a_{kj} \right)^{-1} \], has identical columns. The priority vector can be calculated from any column of the normalized matrix. In this case:

\[
a_{i} = w_i / w_j.
\]  

(2)

Let us suppose that in the last column of consistent pairwise comparison matrix only the value \( a_{in} = w_i / w_n \) is known. Then other pairwise comparisons in the \( n \)-th column can be expressed by:

\[
a_{in} = \frac{w_i}{w_n}; \quad \frac{w_i}{w_j} = \frac{a_{in}}{a_{ij}}, \quad l = 2, \ldots, n-1.
\]  

(3)

There can be a certain level of inconsistency in the pairwise comparison matrix. Then the Eigenvector Method can be used to determine the approximation \( \hat{w} \) of \( w \):

\[
A\hat{w} = \lambda_{\text{max}}\hat{w}, \quad \lambda_{\text{max}} \geq n,
\]

where \( \lambda_{\text{max}} \) denotes the largest eigenvalue of \( A \) and \( \hat{w} \) denotes the eigenvector corresponding to \( \lambda_{\text{max}} \) normalized so that its components sum to unity. The AHP calculates the Consistency Ratio to test whether level of inconsistency is acceptable. It defined as:

\[
CR = \frac{(\lambda_{\text{max}} - n)/(n-1)}{RI},
\]  

(4)

where \( RI \) is the Random Consistency Index calculated based on a large number of randomly generated \( n \times n \) pairwise comparison matrices. For \( n \geq 5 \) allowable \( CR \) should be not more than about 0.10.

A major drawback of the AHP is that a large number of pairwise comparisons is needed to approximate the priority vector. In a study [4], a Monte Carlo simulation was performed using the Incomplete Pairwise Comparisons algorithm [5], to investigate the effect of reduced sets of pairwise comparisons in the AHP. The best results with the least number of pairwise comparisons can be obtained if one starts eliminating or not asking those comparisons that have the smallest \( a_{ij} \geq 1 \).

However, this process is not realizable in practice due to the fact that a full matrix of pairwise comparisons is unknown. On the contrary, the researcher starts with an identity matrix and adds new comparisons. Therefore, we further investigate an approach based on comparison with benchmarks, which makes it possible to significantly reduce the number of judgments required from an expert.

3. The problem of incomplete pairwise comparison matrices

Let a pairwise comparison matrix \( A^m \) be given for initial set of \( m \) requirements. When adding a new \((m+1)\)-th requirement, an expert needs to make \( m \) comparisons of this requirement with the available ones. In order to reduce a number of judgments, it is proposed to compare a new requirement not with all \( m \) requirements, but also with \( k \) benchmarks. It will be useful to consider other pairwise comparisons as missing values. The task is to fill in the pairwise comparison matrix.
Let the first $k$ requirements be chosen as benchmarks. Then a vector of the missing judgments $\mathbf{x}^{(m+1)} = \{x_{i(m+1)}, \ldots, x_{m-k(m+1)}\}$ contains $(m-k)$ elements. The incomplete pairwise comparison matrix can be written as:

$$
\mathbf{A}_k^{(m+1)} = \begin{pmatrix}
1 & \cdots & a_{i,k} & a_{i(k+1)} & \cdots & a_{i,m} & a_{i(m+1)} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1/a_{i,k} & \cdots & 1 & a_{k(k+1)} & \cdots & a_{k,m} & a_{k(m+1)} \\
1/a_{i(k+1)} & \cdots & 1/a_{(k+1)(k+1)} & 1 & \cdots & a_{(k+1)m} & x_{i(m+1)} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1/a_{i(m+1)} & \cdots & 1/a_{(m+1)(m+1)} & 1/a_{(m+1)(k+1)} & \cdots & 1 & x_{i(m-k)(m+1)} \\
1 & \cdots & 1 & 1 & \cdots & 1 & 1
\end{pmatrix}.
$$

The eigenvalue optimization problem is solved [6] to find unknown pairwise comparisons:

$$
\min_{\mathbf{x}^{(m+1)}} \lambda_{\text{max}}(\mathbf{A}_k^{(m+1)}(\mathbf{x}^{(m+1)}))
$$

$$
\frac{1}{9} \leq x_{i(m+1)} \leq 9, \quad i = 1, \ldots, m-k,
$$

where $\lambda_{\text{max}}(\mathbf{A}_k^{(m+1)}(\mathbf{x}^{(m+1)}))$ is the largest eigenvalue of the matrix $\mathbf{A}_k^{(m+1)}(\mathbf{x}^{(m+1)})$.

It is proposed to start from initial approximations (3) for the missing pairwise comparisons. Since there may be a certain level of inconsistency in the original matrix $\mathbf{A}^m$ of judgments, then the expression (3) will not give the same results for various benchmarks. It is reasonable to average all results based on the geometric mean:

$$
x_{i(m+1)} = \sqrt[m-k]{a_{i1(m+1)} \cdots a_{i(k+1)(m+1)}}, \quad i = 1, \ldots, m-k.
$$

After estimating the values of the missing entries by solving the problem (5)-(6), we obtain a complete pairwise comparison matrix $\hat{\mathbf{A}}_k^{(m+1)}$. In the next step, when adding a new $(m+2)$-th requirement, $\hat{\mathbf{A}}_k^{(m+1)}$ is used for calculations and the missing judgments are estimated only for the $(m+2)$-th column and row.

4. **Benchmark selection rules**

In order to test the hypothesis that the choice of the benchmarks significantly affects the results of the AHP, let us formulate some rules for selecting benchmarks in practice. The pairwise comparisons of benchmarks are only available. Therefore, the choice of the benchmarks should be based on the analysis of a pairwise comparison matrix $\mathbf{A}^m$.

**Rule #1.** Random selection. Any of candidate benchmarks can be selected with equal probability. When selecting $k$ benchmarks, a sample is drawn from the elements of $1, \ldots, m$ without replacement.

**Rule #2.** The most consistent judgments. This rule favors the requirements with the lowest level of inconsistency of judgments for which relative weights are the closest to the priority vector. The dissimilarity can be measured as the Manhattan distance $\Delta_j = \sum_m a_{ij} | \hat{w}_j - \hat{w}_i |$. Distance values $\Delta_1, \ldots, \Delta_m$ are sorted in ascending order. The first $k$ requirements in the ordered set are selected as benchmarks.

**Rule #3.** $k$-medoids based on the pairwise comparison matrix. In order to identify the benchmarks it is proposed to apply clustering algorithm. The pairwise comparison matrix is used for partition candidate benchmarks into $k$ groups. Since the benchmarks must belong to an original set of requirements, we use $k$-medoids clustering. Each row of the pairwise comparison matrix describes a requirement with a set of
weights. On the basis of pairwise (Euclidean) distances between rows, we obtain a distance matrix between requirements, which is the input for the clustering algorithm. The PAM algorithm [7] is used for partitioning by medoids. The quality of clustering is defined as the sum of within-cluster distances.

Rule #4. k-medoids based on the normalized pairwise comparison matrix. The same procedure as one for rule #3 except that the distance matrix calculates based on the normalized pairwise comparison matrix.

There have already been attempts to divide objects into clusters for reducing a number of judgements. However, the clusters and pivots method [8] requires that an expert must evaluate which alternatives are close enough, and the clusters can be constructed in a heuristic way only. In contrast, our approach is easier to implement in practice, and the benchmarks are selected in an optimal way.

5. Simulation study
To study the influence of the benchmark selection rules on the results of the AHP, a Monte Carlo simulation was performed. Elements of the true priority vector were generated as independent random variables drawn according to the beta distribution. Their values are normalized so that they sum is equal to one. The beta distribution parameters were chosen to control the asymmetry of weight distribution. The uniform distribution \((\alpha = \beta = 1)\) corresponds to the case when all alternatives are close enough to each other. Such a case was analyzed in [4]. Other simulations \((\alpha = 1, \beta = 3; \alpha = 1, \beta = 6)\) with right-tailed skew assume that there are clearly high priority requirements.

Based on the priority vector \(w^{true}\), the matrix \(A^{true}\) is computed by (2). It is perfectly consistent, but the pairwise comparisons do not correspond to the Fundamental Scale. Therefore, we applied the following procedure. First, elements of the matrix \(A^{true}\) that are only greater than one are converted:

\[
\alpha_{ij}^{true} = \begin{cases} 
\text{round}(\alpha_{ij}^{true}), & 1 < \alpha_{ij}^{true} < 9.5, \\
9, & \alpha_{ij}^{true} \geq 9.5, \\
\alpha_{ij}^{true}, & \alpha_{ij}^{true} \leq 1,
\end{cases}
\]

where \(\text{round}()\) is rounding to the nearest integer. Second, for satisfying property (1), all elements \(\alpha_{ij}^{true} \leq 1\) are replaced by \(1/\alpha_{ij}^{true}\). Thereafter, the matrix \(\tilde{A}^{true}\) will have a low inconsistency, and the priority vector calculated by this matrix will give the same order of the requirements as \(w^{true}\).

Suppose that an ideal expert can assign the pairwise comparisons so that they do not contradict each other, and the result is a matrix \(\tilde{A}^{true}\). In real life applications of the AHP, an expert gives some inconsistent judgments. Let’s generate them as random. First, elements of the pairwise comparison matrix that are only greater than or equal to one are replaced:

\[
\tilde{d}_{ij}^{error} = \begin{cases} 
\tilde{d}_{ij}^{true}, & \tilde{d}_{ij}^{true} \geq 1, \\
\tilde{d}_{ij}^{true} (1 - \tilde{l}_{ij}^{error}), & \tilde{d}_{ij}^{true} < 1,
\end{cases}
\]

where \(\tilde{\delta}_{ij}\) is a random number from one to nine generated from the uniform distribution, \(\tilde{l}_{ij}^{error}\) is a random variable distributed according to the Bernoulli distribution with success probability \(\gamma\). Three different values of \(\gamma\) (0.01, 0.05, and 0.07) were used. Second, for satisfying property (1), all elements \(\tilde{d}_{ij}^{error} : \tilde{d}_{ij}^{error} \geq 1\) are replaced by \(1/\tilde{d}_{ij}^{error}\).

However, such a distortion of the original matrix can lead to a high level of inconsistency. Therefore, the consistency of \(\tilde{A}^{error}\) was tested by (4). The initial set included \(m=6\) requirements. Three different matrix sizes \(n (10, 15, \) and \(20)\) were used. The true priority vector had the maximum number of elements 20. Therefore, the matrix \(\tilde{A}^{error}\) was 20×20, its submatrices of dimensions 6×6, 10×10, 15×15 were extracted. The consistency of these matrix and submatrices were tested. If at least in one case the inconsistency level was unacceptable, random errors were generated again. A new matrix \(\tilde{A}^{error}\) was
computed, and the consistency was tested again. And so on until the condition \( CR \leq 0.1 \) was met for all submatrices and the matrix.

For the submatrix of \( \tilde{A}^{\text{err}} \) of dimension \( 6 \times 6 \), \( k \) benchmarks were chosen according to the rules described above. The number of the benchmarks varied from 1 to 3. It was assumed that the pairwise comparisons with the selected benchmarks were known, they were taken from the full matrix. The other pairwise comparisons were estimated by solving the problem (5)-(6). Based on the obtained matrix, the priority vector was estimated. Finally, 500 replications of the data were generated according to the various levels of simulation design factor: \( n \), the beta distribution parameters, \( \gamma \), \( k \).

To evaluate the accuracy of estimated weights, we used the correlation performance measure. Since it is important that the requirements are ordered in the correct way, the Spearman’s rank correlation coefficient is calculated as the Pearson correlation coefficient between \( \text{rank}(\hat{w}^{\text{true}}) \) and \( \text{rank}(\hat{w}) \), where \( \text{rank}(\cdot) \) returns the ranks of the vector elements.

As a result of simulations, 162,000 values of the Spearman’s correlation coefficients were obtained for all combinations of factor levels. ANOVA model was estimated to study the influence of factors on the accuracy of approximating priority vector. The response was the Spearman’s correlation coefficient between the true priority vectors and calculated ones. The model included the main effects and the interaction effects. Identification of ANOVA model was performed using the least squares method. The simulation study and the analysis of its results were carried out using free software for statistical analysis R.

6. Results

Based on the results of ANOVA model identification we calculated the effects of the benchmark selection rules under various experimental conditions. At the base line (\( k = 1, n = 10 \), and \( \alpha = \beta = 1 \)), all the proposed methods have positive effects compared to a random choice (see Table 1). The most significant improvement is provided by the rule \#2 which takes into account consistency of judgements.

When the experimental conditions change, the effects of the benchmark selection rules will be different. Only an error fraction \( \gamma \) does not affect; the other conditions are important. In order to understand what advantages the benchmark selection rules have in comparison with a random choice under various experimental conditions, it is necessary to sum up the interaction effects and the main effects. The results are presented in Tables 1-2.

| Table 1. Estimated effects of the benchmark selection rules compared to a random choice, \( k = 1 \). |
|--------------------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
| Rule    | \( n = 10 \) | \( n = 15 \) | \( n = 20 \) | \( n = 10 \) | \( n = 15 \) | \( n = 20 \) |
| #2     | 0.0107*** | 0.0115*** | 0.0130*** | 0.0124*** | 0.0132*** | 0.0147*** |
| #3     | 0.0038**  | 0.0070*** | 0.0074*** | 0.0074*** | 0.0107*** | 0.0110*** |
| #4     | 0.0056*** | 0.0097*** | 0.0106*** | 0.0079*** | 0.0120*** | 0.0129*** |

*** Significant at 0.1% level, ** Significant at 1% level.

Table 1 shows that rule #2 of selecting the most consistent judgements is still the best independent of the pairwise comparison matrix size and the distribution of weights. For a highly skewed distribution of weights and large dimensions of the pairwise comparison matrix, rule #4 based on \( k \)-medoids becomes comparable to selection rule #2 (see Table 1).

It is interestingly that benchmark selection rules #2 and #3 will give worse results than a random selection when selecting two or three benchmarks for small pairwise comparison matrices. Table 2 shows that the effects are not as significant as with one benchmark. This means that several benchmarks must be chosen in another way. This applies especially to rule #2: several benchmarks with the lowest degree of judgment inconsistency give a negative or null effect compared to a random benchmark under
all experimental conditions (see Table 2).

| Rule | $n = 10$ | $n = 15$ | $n = 20$ | $n = 10$ | $n = 15$ | $n = 20$ |
|------|---------|---------|---------|---------|---------|---------|
| #2   | 0.0037  | 0.0029  | 0.0014  | 0.0020  | 0.0012  | 0.0003  |
| #3   | 0.0058  | 0.0026  | 0.0023  | 0.0022  | 0.0010  | 0.0013  |
| #4   | -0.0019 | 0.0021  | 0.0030  | 0.0004  | 0.0044  | 0.0054  |

Significant at 0.1% level, * Significant at 1% level, ** Significant at 5% level.

The selection rule based on $k$-medoids has a positive effect when the pairwise comparison matrix size is greater than ten. In addition, using the normalized pairwise comparison matrix as the input of the PAM algorithm provides some advantages.

7. Conclusion
The proposed rules for the benchmark selecting should be used in the cases where a number of requirements is very large, and it is required to significantly reduce a number of pairwise comparisons. If there are no clearly high priority requirements (the uniform or slightly skewed distribution of weights), then it is recommended to choose one benchmark with the lowest degree of judgment inconsistency. If the distribution of weights is highly skewed, then one benchmark can be chosen on the basis of $k$-medoids. The choice of several benchmarks should be based on other rules. Their development will be the direction of future research.

8. Acknowledgments
The reported study was funded by the Russian Ministry of Education and Science according to research project No. 2.2327.2017/4.6.

References
[1] Achimugu P, Selamat A, Ibrahim R, and Mahrin M N 2014 A systematic literature review of software requirements prioritization research *Information and Software Technology* **56**(6) 568-585
[2] Qaddoura R, Abu-Srhan A, Qasem M H, and Hudaib A 2017 Requirements Prioritization Techniques Review and Analysis *Proc. Int. Conf. on New Trends in Computing Sciences* (Amman: Jordan) pp 258-263
[3] Saaty T L, Sodenkamp M 2008 Making decisions in hierarchic and network systems *Int. J. of Applied Decision Sciences* **1**(1) 24-79
[4] Carmone Jr F J, Kara A, and Zanakis S H A 1997 A Monte Carlo investigation of incomplete pairwise comparison matrices in AHP *European Journal of Operational Research* **102**(3) 538-553
[5] Harker P T 1987 Incomplete pairwise comparisons in the analytic hierarchy process *Mathematical Modelling* **9**(11) 837-848
[6] Bozóki S, Fülöp J N, and RóNyai L 2010. On optimal completion of incomplete pairwise comparison matrices *Mathematical and Computer Modelling* **52**(1-2) 318-333
[7] Flach P 2012 *Machine learning: the art and science of algorithms that make sense of data* (Cambridge University Press)
[8] Ishizaka A 2012 Clusters and pivots for evaluating a large number of alternatives in AHP *Pesquisa Operacional* **32**(1) 87-102