Software Requirements Prioritizing

Joachim Karlsson*

SoftLab ab
Datalinjen 1
S-583 30 Linköping, Sweden
Internet: joachim@softlab.se

Abstract

The importance of candidate software requirements can vary by orders of magnitude, yet most software providers do not have accurate and efficient means for selecting among them. This papers describes a case study at Ericsson Radio Systems AB of two techniques for software requirements prioritizing as a means for determining the importance of candidate requirements, a pair-wise comparison technique and a numeral assignment technique. The results from the case study indicate that the pair-wise comparison technique is an efficient, informative and accurate means for finding the candidate requirements importance. We therefore recommend the pair-wise comparison technique for software requirements prioritizing. At Ericsson we have extended its use in other development projects.

1. Introduction

In a majority of all software development projects there are more candidate requirements than time and resources allows for and which the customers are willing to pay for. A function can always be added, reliability can always improved and the user interface can always be enhanced. Of the candidate requirements the importance for customer satisfaction can vary by orders of magnitude [6]. Some of them may be critical for the success of the forthcoming software system, while others may merely be adornments. The set of candidate requirements most likely to achieve customer satisfaction must consequently be selected for implementation.

Most software providers neither aim to separate the important requirements from the less important, nor do they have accurate and efficient means for doing this. As a consequence, it can be problematic to achieve the best possible software system.

Previous affiliation: Department of Computer and Information Science, Linköping University, S-581 83 Linköping, Sweden.

The need for prioritizing the requirements according to their perceived importance is highly acknowledged in the literature [3, 5, 11]. The means available for performing requirements prioritizing are, however, not completely investigated. Still, an accurate and efficient technique for software requirements prioritizing would give a software provider advantages which include:

- A clear means for selecting an optimized set of software requirements for implementation.
- Support for software engineers to resolve trade-offs in subsequent development [2].
- An aid for project managers in estimating the expected customer satisfaction of the software system before it is released.

To gain a better understanding of how to efficiently and accurately prioritize software requirements, we performed a case study at Ericsson Radio Systems AB involving two techniques developed for the purpose. The first technique, a *pair-wise comparison technique*, is based on the analytic hierarchy process [9], and the second technique employs a *numeral assignment technique*. The pair-wise comparison technique requires a person to determine the *relative* importance of all pair of the candidate requirements, i.e., the person has to answer the question "which of the two requirements is more important and to what extent?" The numeral assignment technique requires a person to determine the *absolute* importance of the candidate requirements, i.e., the person has to answer the question "how important is this requirement?"

The case study was performed in the RAN-project at Ericsson Radio Systems AB. 14 candidate requirements were selected for the case study. Five participants performed the pair-wise comparison technique, and three of these performed the numeral assignment technique on a different occasion. The results from the case study indicate that the pair-wise comparison technique is faster and more accurate than the numeral assignment technique. Using the pair-wise comparison technique the system developers are forced to make trade-offs between requirements; all requirements cannot be equally important. Moreover, consensus about importance of candidate requirements is higher using the pair-wise comparison technique. Those using both techniques subjectively favor the pair-wise comparison technique, mainly due to the effectiveness of determining relative priorities rather than absolute ones.

Since the case study the pair-wise comparison technique has been applied to other projects and Ericsson Radio Systems AB is currently planning to extend its use in the requirements engineering process.

1.1 Outline of this paper

This paper continues with two sections that describe the pair-wise comparison technique and the numeral assignment technique, respectively. Section four describes the case study using the two techniques in the RAN-project. Section five describes the results from the case study, and finally the sixth and seventh sections complete the paper by describing conclusions from the case study and providing recommendations.

2. The pair-wise comparison technique

The first proposed technique for prioritizing software requirements, the *pair-wise comparison technique*, is a technique based on the analytic hierarchy process, AHP, [9].

In this technique the candidate requirements are compared pair-wise to estimate their *relative* importance. The person thus has to decide which requirement is more important, and to what extent. The scale used for the pair-wise comparisons is outlined in table 1. For instance, intensity of importance 5 means that one requirement is essentially or strongly more important than another.

Intensity of importance	Definition
1	Equal importance
3	Moderate importance of one over another
5	Essential or strong importance
7	Very strong importance
9	Extreme importance
2, 4, 6, 8	Intermediate values between the two adjacent judgements
Reciprocals	If requirement *i* has one of the above numbers assigned to it when compared with requirement *j*, then *j* has the reciprocal value when compared with *i*.

Table 1: The fundamental scale used for pair-wise comparisons in the AHP.

Table 1: The fundamental scale used for pair-wise comparisons in the AHP.

Since this technique prescribes pair-wise comparisons of all candidate requirements, the required number of comparisons grows polynomally. For a software system with n candidate requirements, $n \cdot (n-1)/2$ pair-wise comparisons are needed.

To illustrate the concept of pair-wise comparisons, assume there are three candidate requirements; *A*, *B* and *C*. The first step is to pair-wise compare the three requirements according to the scale in table 1. First *A* is compared to *B*, then *A* to *C*, and finally *B* to *C*. Assume the following relationships have been determined:

- *A* is essentially more important than *B* (intensity of importance 5).
- *C* is moderately more important than *A* (intensity of importance 1/3; notice the reciprocal value).
- *C* is very strongly more important than *B* (intensity of importance 1/7; notice the reciprocal value).

Intuitively, one can tell that *C* is more important than *A*, which is more important than *B*. It is, however, difficult to more precisely establish the difference in importance of the three candidate requirements.

2.1 Calculating the requirements priorities

When the candidate software requirements have been pair-wise compared, their relative priorities are to be calculated. This is done by inserting the n candidate requirements in the rows and columns of a matrix of order n, a comparison matrix. For each pair of requirements, e.g., *A* and *B*, their relative intensity of importance is inserted in the position where the row of *A* meets the column of *B*. In the transposed positions, the reciprocal values of the pair-wise comparisons are inserted. Since a requirement is equally important when compared to itself, a '1' is inserted in the main diagonal.

Inserting the relative importances as determined by the pair-wise comparisons in the previous example, the comparison matrix is as follows:

	A	*B*	*C*
A	1	5	1/3
B	1/5	1	1/7
C	3	7	1

To calculate the relative priority of each candidate requirement involves calculating the eigenvalues of the resulting comparison matrix. The eigenvalues of a comparison matrix represent the priorities of each of the candidate requirements [9]. Since calculating the eigenvalues may require significant computing power, they can be approximated using a technique known as averaging over normalized columns [9].

An important feature of the AHP is that the resulting requirements priorities are relative and based on a ratio scale. That is, the requirements' priorities always add up to 100%, and a requirement with a priority of 30% consequently represents 30% of the total importance. Moreover, a requirement of priority 30% is three times as important as a requirement with a priority of 10%. The pair-wise comparison technique consequently provides a good means for discussing each candidate requirement's ability to achieve customer satisfaction.

The eigenvalues of the comparison matrix and thus the resulting priorities of the three requirements in the example are:

- Requirement *A*. 28%
- Requirement *B*. 7%
- Requirement *C*. 65%

From the example, we determined requirement *C* as the most important, being about nine times as important as requirement *B* and about twice as important as requirement *A*. Moreover, requirement *A* is evaluated as being about four times as important as requirement *B*.

2.2 Estimating the accuracy of the comparisons

Another important feature of the AHP is the redundancy of the pair-wise comparisons which makes the AHP much less sensitive to judgement errors [8]. Assume, for instance, requirement *A* is determined as being moderately more important than *B*, and requirement B is determined as being moderately more important than *C*. If, then, requirement *C* is determined as being moderately more important than *A*, a judgement error has definitely occurred. Such judgement errors of the pair-wise comparisons can be measured and thus identified using the AHP. This is accomplished by calculating the *consistency index* of the comparison matrix and subsequently calculating the *consistency ratio* [9].

The consistency index, *CI*, of the comparison matrix is calculated as $CI = (\lambda_{max} - n)/(n-1)$, where λ_{max} denotes the maximum principal eigenvalue, and n denotes the number of candidate requirements. The closer λ_{max} is to n, the more accurate the resulting priorities. Then, the consistency ratio is calculated as the ratio of the consistency index to the average consistency index of a randomly generated reciprocal matrix of the same order. For a more thorough and illustrative description of how the consistency index and the consistency ratio are calculated, see [9].

Continuing the previous example, the consistency index is calculated at 0.03 and the consistency ratio at 0.06. A consistency ratio below 0.10 is considered acceptable. In industrial-strength applications of the AHP higher consistency ratios are, however, rather common.

3. The numeral assignment technique

The second proposed technique for prioritizing software requirements, the *numeral assignment technique*, is based on the principle that each requirement is assigned a symbol representing the requirement's perceived importance. This approach is common in Quality Function Deployment (QFD) where prioritizing of candidate requirements is required, see, e.g., [4, 10].

Several variants based on the numeral assignment technique exist. A straightforward approach to the technique is presented by Brackett [1], who suggests that requirements should be classified as *mandatory*, *desirable*, or *inessential*. An approach using finer granularity is to assign each requirement a number on a scale ranging from 1 to 5, where the numbers indicate [6]:

- *5*. Mandatory (the customer cannot do without it).
- *4*. Very important (the customer does not want to be without it).
- *3*. Rather important (the customer would appreciate it).
- *2*. Not important (the customer would accept its absence).
- *1*. Does not matter.

For the case-study of software requirements prioritizing we chose to apply the latter approach (the scale ranging from 1 to 5) in the RAN-project.

4. The case-study

Over the period ranging from August, 1994 to June, 1995 I participated in the RAN-project at Ericsson Radio Systems AB. On average six senior developers took active part in the project. One of many purposes of the project was to discover requirements from an operation and maintenance perspective on radio access networks. Another purpose was to infuse and evaluate new methods and techniques in the requirements engineering process.

For this case-study, 14 candidate requirements were selected as being relevant.

4.1 Objectives

The state-of-the-practice at Ericsson Radio Systems AB is to rank-order the candidate requirements on a scale ranging from 1 to 3, where 1 denotes highest priority. In most projects time and resources are available to implement only those candidate requirements that fall into the highest priority category. Thus the current prioritizing process is primarily a means for selecting and discarding candidate requirements for implementation. Unfortunately, little or no guarantee can be given that requirements that by right belong to the most important category do not fall into another category, or vice versa.

The objectives of this case study were to gain a better understanding of software requirements prioritizing, and to find an efficient and accurate means for prioritizing requirements. Two techniques for prioritizing requirements were selected for the case study, a pair-wise comparison technique and a numeral assignment technique.

Six broad categories of issues were of particular interest in the case study:

- Which prioritizing technique is faster?
- Which prioritizing technique yields the more informative results?
- Which prioritizing technique is more trustworthy?
- What are the participants' subjective responses to the two prioritizing techniques?
- Are there any other positive side effects when prioritizing candidate requirements?
- If any, which prioritizing technique do we recommend for future use, and why?

4.2 The pilot project

The RAN-project was selected as a pilot project for this study. The major contributing factors to this choice were the fact that I had participated in the project previously and was familiar with its progress; that the RAN-project is both important and typical for Ericsson Radio Systems AB; and finally the fact that the participants had a positive attitude to performing a case study of this kind.

When the requirements for the RAN-project were discovered and agreed upon by the participants, we chose 14 candidate requirements for the case study. These were of the type:

- "The system must provide information about which RAN-services are implemented."
- "The system must provide information about the extent to which each RAN-service is used."
- "The system must provide information about what coverage can be used."

In many software projects at Ericsson Radio Systems AB there are no given customers. Thus, the developers have to prioritize the candidate requirements they believe the customers will prefer. Obviously having prospective customers available is preferable, but in the given context using system developers is a viable alternative.

4.3 Applying the techniques

Five participants in the RAN-project performed the pair-wise comparison technique and three of them the numeral assignment technique. Two of the participants were not able to participate when we applied the numeral assignment technique.

First each of the candidate requirements was presented by the project manager who explained them and led discussions with the other participants. This was to clarify the requirements and reduce the risk of misinterpreting the requirements. The two prioritizing techniques were then individually applied to the project.

The pair-wise comparison technique

Before the pair-wise technique was applied, the five participants were given a short introduction on how to perform pair-wise comparisons. They were not informed about the analytic hierarchy process, nor all the purposes of performing the case-study. In total each of the participants had to perform $14 \cdot (14-1)/2 = 91$ pair-wise comparisons of the candidate requirements.

The participants then individually performed pair-wise comparisons of the 14 candidate requirements. These were outlined in a given order, but the participants were free to work with them in any order. Thus retraction was possible during their work. The session was not moderated and the participants worked at their own pace. Discussions were allowed, even though very few arose.

When the 14 requirements had been pair-wise compared, the importance distribution and the consistency ratio values were calculated using a simple C-program developed for the purpose.

The numeral assignment technique

Before the numeral assignment technique was applied, the five participants were given a short introduction on how to assign the symbols 1–5. The 14 candidate requirements were outlined in a given order on a single slide and all of them were visible simultaneously.

The participants then assigned each requirement a symbol corresponding to how they believed the customer would evaluate the requirements. The session was moderated and discussion formed, of course, a natural part of the process. When an issue was raised, the topic was discussed until consensus was reached. The participants were not restricted to assigning the requirements in a strict order, but were allowed to backtrack if they wanted to change priorities.

5. Results from the case study

The case study of software requirements prioritizing in the RAN-project involved five participants using the pair-wise comparison technique and three of these using the numeral assignment technique. Two of the participants were unable to participate when applying the numeral assignment technique. Therefore, only the results from those who participated in both applications are reported.

General information about the case study is outlined in table 2 (pair-wise comparison) and in table 3 (numeral assignment).

	Person A	Person B	Person C	*Average*	*Standard deviation*
Total time to perform the pair-wise comparisons (minutes)	35	45	35	**38**	6
Average time to perform one pair-wise comparison (seconds)	12	15	12	**13**	2
Consistency index	0.35	0.22	0.32	**0.30**	0.07
Consistency ratio	0.21	0.13	0.19	**0.18**	0.04

Table 2: Information about the pair-wise comparisons.

Total time to perform the numeral assignments	1 hour and 23 minutes
Average time to perform one numeral assignment	5 minutes and 56 seconds

Table 3: Information about the numeral assignments.

The results from the priority determination are numerically outlined in tables 4 and 5, and graphically illustrated in figures 1 and 2.

Requirements priority Requirements no:	Person A	Person B	Person C	*Average*	*Standard deviation*
1.	13%	13%	11%	**12%**	1%
2.	10%	8%	1%	**7%**	5%
3.	6%	5%	3%	**5%**	2%
4.	9%	10%	3%	**7%**	4%
5.	12%	10%	14%	**12%**	2%
6.	13%	18%	18%	**16%**	3%
7.	6%	1%	2%	**3%**	3%
8.	3%	2%	4%	**3%**	1%
9.	5%	2%	4%	**4%**	2%
10.	1%	5%	6%	**4%**	3%
11.	2%	1%	1%	**1%**	0%
12.	2%	1%	1%	**1%**	0%
13.	14%	22%	28%	**21%**	7%
14.	4%	2%	4%	**3%**	1%

Table 4: Requirements priority according to the pair-wise comparison technique.

Requirements priority Requirements no:	Assigned priority
1.	5
2.	3
3.	3
4.	4
5.	5
6.	5
7.	2
8.	2
9.	2
10.	2
11.	1
12.	2
13.	5
14.	2

Table 5: Requirements priority according to the numeral assignment technique.

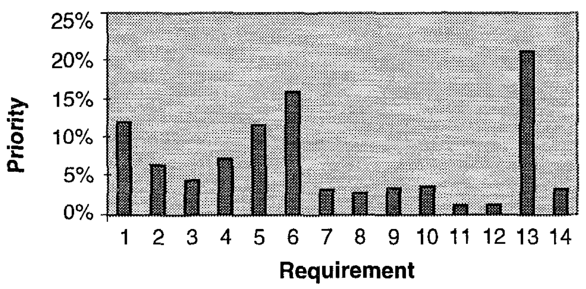

Figure 1. The candidate requirements average priority according to the pair-wise comparison technique.

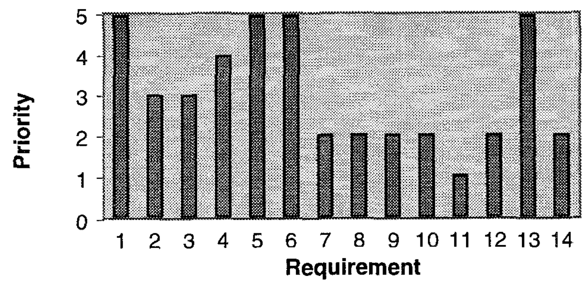

Figure 2. The candidate requirements priority according to the numeral assignment technique.

The results of the pair-wise comparison technique (as outlined in table 4) indicate that the importance of requirements can vary by orders of magnitude. The four most important requirements, numbers 1, 5, 6 and 13, add up to **61%** of the total importance. Moreover, the five least important requirements, numbers 7, 8, 11, 12 and 14, account for only **11%** of the total importance. Of the pair-wise comparisons, requirement 13 is determined about twenty times as important as requirements 11 and 12.

The results of the numeral assignment technique (as outlined in table 5) show that four requirements are considered mandatory, 1, 5, 6 and 13, while requirement 11 is hardly considered to have any importance at all. The other requirements are considered either very important, rather important or not important.

Figures 1 and 2 indicate that the results of the two techniques agree rather well. The pair-wise comparison technique yields, however, both additional and more accurate information. From table 4 we can, for instance, see that requirements 9 and 10 are (according to the judgements in the pair-wise comparisons) four times as important as requirement 12, even though all three were assigned priority 2 (in the numeral assignments). On the other hand, we see that requirement 1 is determined only about twice as important as requirement number 2 even though they are assigned priority 5 and 3, respectively. Thus, the difference in importance of requirements assigned the same priority (using the numeral assignment technique) can be larger than the difference in importance of requirements assigned different priorities. This observation indicates that the pair-wise comparison technique is more accurate and more informative than the numeral assignment technique.

6. Conclusions from the comparison

The results from this case study indicate that the pair-wise comparison technique is faster than the numeral assignment technique. Still, the number of required pair-wise comparisons is much higher than the required numeral assignments. One reason for this is that determining relative information is easier and faster to carry out than determining absolute information, as claimed by he participants in the case study. Another reason is that the participants' subjective opinions about the numbers in the numeral assignment techniques differ widely. Often we were caught up in abstract discussions about what a '4' and a '5' really stand for and how to accurately separate them.

We also found that the pair-wise comparison technique yields more informative results. This is mainly due to the fact that the candidate requirements priorities are based on a ratio scale and also that the priorities always add up to 100%. Also, the differences in priority among the candidate requirements seem to be much clearer using the pair-wise comparison technique. Using the results from the pair-wise comparison technique, we could discuss the requirements much more objectively than we could using the numeral assignment technique. For instance, we could discuss issues such as "if requirement A is so many times more important than requirement B, is it then cost-efficient to include requirement B at all?"

Using the numeral assignment technique, the priorities of the candidate requirements varied widely. Requirement 2, for instance, was first proposed as a '1' but later refined to a '3'. Requirement 11 was the first (and only) requirement assigned a '1', this was mainly due to the how the requirements was formulated. Requirement 8 was interesting: suggestions for assignment varied from a '1' to a '4', but finally settled at a 2. A reason for the many variations is that the requirements are written in English since it is the standard corporate language. Often the participants interpreted the candidate requirements slightly differently and thus assessed them differently.

The participants were doubtful about the trustworthiness of the numeral assignment technique. Moreover, the participants stated that a current risk is that all requirements are considered high priority, because if they were not important, the participants would have otherwise done a bad job developing the requirements.

The developers had similar comments about the pair-wise comparison technique:

- They found the notion of pair-wise comparisons was sound and appealing.
- They claimed that is was easier to use the pair-wise comparison technique than the numeral assignment technique.
- They argued it was tedious to carry out all the required pair-wise comparisons. They lost concentration a number of times during the process.
- They sometimes backtracked in order to check the consistency of their pair-wise comparisons.
- One participant believes he varied his evaluations using the pair-wise comparison technique too much, because of sheer tedium.

Prior to the case study all participants agreed upon, and understood, the candidate requirements. During the pair-wise comparisons, the participants were forced to contemplate the details of each requirement. In this process we found that the requirements were much more ambiguous than previously expected. Prior to the case study we were convinced we had interpreted the requirements in the same way. But when the pair-wise comparisons started, we were forced to face off our interpretations with those of the other participants. Then it became clear that almost each participant had his own interpretation of the candidate requirements. Thus the pair-wise comparison technique helped us identify requirements that were stated in too ambiguous a way and needed to be reformulated. In particular, one requirement needed to be completely rewritten.

In conclusion, the pair-wise comparison technique is favorable. It is easier to carry out and yields more informative results for the participants to discuss and consider. The pair-wise comparison technique also seems to be deemed more trustworthy than the numeral assignment technique and also appears to be less time-consuming.

7. Recommendations

Since we have found that the importance of candidate requirements can vary by orders of magnitude, we strongly recommend that these differences are quantified in the requirements engineering process. A viable strategy for this is to prioritize the requirements using the pair-wise comparison technique. Using this approach the differences in requirements importance become clearly quantified and software managers thus have an explicit means for establishing major requirements from others. A positive side effect of the prioritizing process is that the participants have to consider the candidate requirements from different points of view, and are thus able to detect deficiencies that were not previously detected. It is planned at Ericsson to extend the use of the pair-wise comparison technique, and a tool for evaluating requirements from various perspectives is being built [7].

As with most techniques, there is always room for improvement. One suggestion was that the ordering of the pair-wise comparisons should be random, as this would force the participants to think more thoroughly and also ease the tedium. It would, however, probably increase the required time. Another emerging idea was to discard some clearly unimportant requirements during the pair-wise comparison process thus reducing the time required.

8. Acknowledgments

This work has been supported by Ericsson Radio Systems AB and The Swedish National Board for Industrial and Technical Development, Dnr 93-3280.

9. References

[1] Brackett, J. W. (1990). Software requirements. Technical Report SEI-CM-19-1.2, Software Engineering Institute, Carnegie Mellon University, USA.

[2] Curtis, B., Krasner, H., and Iscoe, N. (1988). A field study of the software design process for large systems. *Communications of the ACM*, 31(11):1268–1287.

[3] Davis, A. M. (1993). *Software Requirements: Objects, Functions and States*. Prentice-Hall International, Inc., Englewood Cliffs, New Jersey.

[4] Hauser, J. R. and Clausing, D. (1988). The house of quality. *Harvard Business Review*, May-June issue, pp. 63–73.

[5] Holbrook III, H. (1990). A scenario-based methodology for conducting requirements elicitation. *ACM SIGSOFT*, 18(1):95–104.

[6] Karlsson, J. (1995). Towards a Strategy for Software Requirements Selection. Licentiate Thesis 513, Department of Computer and Information Science, Linköping University, Sweden.

[7] Karlsson, J. and Ryan, K. (1996). Supporting the Selection of Software Requirements. In *8th International Workshop on Software Specification and Design*. IEEE Computer Society Press.

[8] Millet, I. and Harker, P. (1990). Globally effective questioning in the analytic hierarchy process. *European Journal of Operational Research*, 48(1):88–97.

[9] Saaty, T. L. (1980). *The Analytic Hierarchy Process*. McGraw-Hill, Inc.

[10] Sullivan, L. (1986). Quality function deployment. *Quality Progress*, June issue, pp. 39–50.

[11] Zave, P. (1995). Classification of research efforts in requirements engineering. In *Second IEEE International Symposium on Requirements Engineering*, pages 214–216. IEEE Computer Society Press.