Multi Criteria Selection of Components Using the Analytic Hierarchy Process

João W. Cangussu, Kendra C. Cooper, and Eric W. Wong

University of Texas at Dallas,
Department of Computer Science
Richardson TX 75083, USA,
{cangussu,kcooper,ewong}@utdallas.edu

Abstract. The Analytic Hierarchy Process (AHP) has been successfully used in the past for the selection of components, as presented in case studies in the literature. In this paper, an empirical study using AHP to rank components is presented. The components used in the study are for data compression; each implements one of the Arithmetic Encoding (AREC), Huffman coding (HUFF), Burrows-Wheeler Transform (BWT), Fractal Image Encoding (FRAC), and Embedded Zero-Tree Wavelet Encoder (EZW) algorithms. The ranking is a semi-automated approach that is based on using rigorously collected data for the components' behavior; selection criteria include maximum memory usage, total response time, and security properties (e.g., data integrity). The results provide a clear indication that AHP is appropriate for the task of selecting components when several criteria must be considered. Though the study is limited to select components based on multiple non-functional criteria, the approach can be expanded to include multiple functional criteria.

1 Introduction

The selection of components is recognized as a challenging problem in component based software engineering [1–3], as there are complex technical, legal, and business considerations that need to be simultaneously and iteratively addressed as development proceeds. When selecting a component, the number of criteria can be large. Established metaheuristic search techniques [4] from the artificial intelligence community have been proposed to search for software components including genetic algorithms [5, 6] and evolutionary algorithms [7]. Alternative approaches using multi-criteria decision making (MCDM) techniques have also been employed as a solution to this problem [2, 8–12]. One well known MCDM technique is the Analytic Hierarchy Process (AHP).

Case studies are available in the literature that report the successful use of the AHP approach [9–11]. The data supporting the description of the components in these studies appears to be obtained from vendor specifications, etc. Here, we rigorously collect data about the components as the foundation for their selection. We note that other sources [1, 13] have provided arguments against the use of the AHP approach for the selection of components; this is discussed in Section 4.

In this paper we present an empirical study for the ranking, using AHP, of components based on non-functional criteria. To the best of our knowledge, no such work is currently available. A relatively small number of empirical studies in component based software engineering have become available over the years, which include the representation and selection of UNIX tool components [14], support for regression testing of component based software [15], design of a knowledge base used for business components [16], variations in COTS-based software development processes used in industry [17], and the use of fuzzy logic to specify and select components based on a single selection criterion [18]. Due to the very limited number of empirical studies available in the area of component based software engineering, this study makes a substantial contribution to the literature.

The results of this study provide a clear indication of the suitability of AHP for this task. In addition, the approach can be extended to incorporate functional requirements and the possible integration of components when no suitable alternative implements all required functionality.

The remainder of this paper is organized as follows. A general description of AHP is presented in Section 2. An empirical study for the selection of components for non-functional requirements using AHP is the subject of Section 3. Section 4 presents relevant related work. Conclusions and extensions of the work described in this paper are addressed in Section 5.

2 Analytic Hierarchy Process (AHP)

The problem of selecting the best alternative from a set of options that are characterized by criteria that may be qualitative, quantified with different units of measure, and conflict with each other has been under investigation for centuries [19]. The decision making approaches proposed to address this problem are called multi-criteria decision making (MCDM) methods. There are numerous MCDM methods available including the Weighted Sum Method, Weighted Product Method and Analytic Hierarchy Process (AHP) [20, 21].

The AHP method has three main steps [22] (refer to Figure 1). The first step is to structure the decision making problem as a hierarchical decomposition, in which the objective or goal is at the top level, criteria used in the evaluation are in the middle levels, and the alternatives are at the lowest level. The simplest form used to structure a decision problem consists of three levels: the goal at the top level, criteria used for evaluation at the second level, and the alternatives at the third level. We use this form to present the second and third steps in the AHP.

The second step is to create decision tables at each level of the hierarchical decomposition. The matrices capture a series of pairwise comparisons (PC matrices) using relative data. The comparison can be made using a nine point scale or real data if available. The nine point scale includes: $[9, 8, 7, \ldots, 1/7, 1/8, 1/9]$,

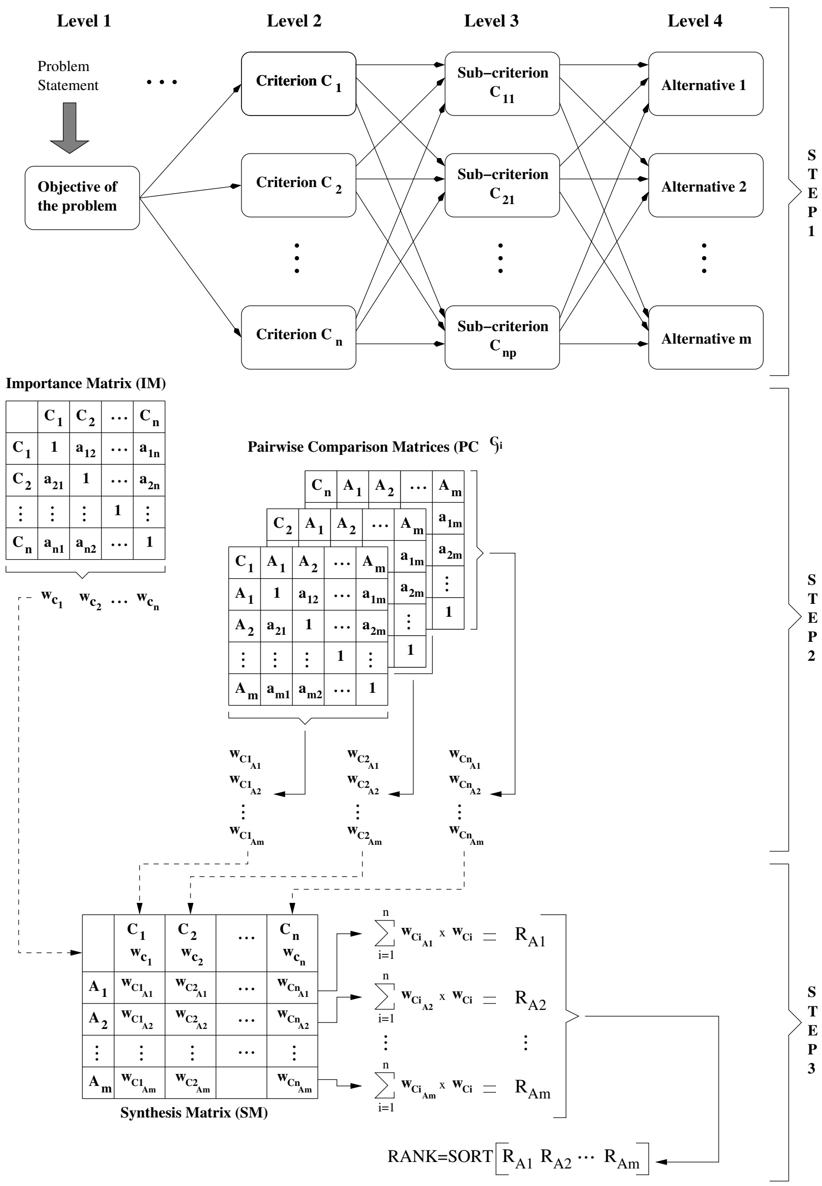

Fig. 1. General structure of the AHP approach.

where 9 means extreme preference, 7 means very strong preference, 5 means strong preference, and continues down to 1, which means no preference. The reciprocals of the above levels are also available. For example, if a comparison between “a” and “b” is evaluated as 7, then the comparison between “b” and “a” is 1/7.

For the top level of the decomposition, a single importance matrix (IM) is defined, which captures the relative importance of each criterion; these are considered in the second level. For example, if the evaluation criteria are response time (RTP), memory usage (MU), data security (DS), and data integrity (DI), then the decision maker compares their importance, two at a time in a four by four matrix. For example, a decision maker may determine that the RTP is strongly preferred to DI. In this case the pairwise comparison of RTP/DI has the value 5; DI/RTP has the inverse value 1/5. In AHP, the values captured in the IM reflect the relative importance of criteria, i.e., the requirements, from the decision maker’s perspective. The IM does not capture additional relationships among the criteria, such as synergistic or conflicting relationships. The criteria are assumed to be independent. This is likely to be the case in many decision making situations, in which the conflicting or synergistic relationships among the requirements are not well understood.

The IM for the top level provides a means to prioritize the requirements for the decision making process. Using the comparison data in the matrix a priority vector can be calculated, for example, with an eigenvector formulation. This priority vector is used in the third step, when the priorities are aggregated.

The decision tables for the middle level capture a pairwise comparison (PC) matrix for the alternative solutions for each evaluation criterion. Consequently, each evaluation criterion has a PC^{C_i}. For example, if there are four evaluation criteria and five alternatives to choose from, then this results in four PC that are five by five. The priority vector for each of these decision tables is calculated. Here again, an eigenvector calculation can be used. The priority vector can be represented in normalized form, such that the sum of the elements is equal to one. This vector is used in the third step, when the priorities are aggregated. In AHP, the values captured in a PC matrix do not explicitly capture other relationships such as synergistic or conflicting relationships that are present in the alternative solutions.

The third step aggregates, or synthesizes, the priorities so that the “best” alternative can be chosen. Options for the aggregation include a distributive mode and an ideal mode. As these two options have been shown to give the same result 92% of the time [23], we present the distributive mode here and refer the reader to [23] for a discussion of the ideal mode. The distributive mode calculation uses the priority vector calculated in step one as a weight vector, W, and the normalized priority vectors calculated in step two to define a new matrix A. Each element A_{ij} represents the normalized priority calculated in step two for alternative i and evaluation criteria j. To calculate an alternative’s overall priority, each row in the matrix is multiplied by the weight vector and the multiplied elements are added together:

$$Alternative\ priority\ i = \sum_{j=1}^{4} (w_j * A_{ij}) \qquad (1)$$

The alternative with the largest value represents the best alternative.

3 Empirical Study

An empirical study of how to select components based on non-functional criteria is addressed in this Section. First, a set of available components is described in Section 3.1. This is followed by Section 3.2, which presents all the steps to rank these components according to two distinct scenarios.

3.1 Compression Components

Though there exist a large number of compression techniques, the subset used in this section has been reduced to five compression components: Arithmetic Encoding (AREC), Huffman coding (HUFF), Burrows-Wheeler Transform (BWT), Fractal Image Encoding (FRAC), and Embedded Zero-Tree Wavelet Encoder (EZW). It is our understanding that they provide a broad spectrum in terms of the features of interest (e.g., response time, compression ratio, quality, etc). Source code for these components is available over the Internet. As a first step, the credibility of the source code for both compression and decompression has been assured based on available reviews and references given by various web sites, a thorough code walk through, and testing the components. In some cases the source code has been modified so that it could compile in the lab environment using the g++ compiler (with default settings) on Sun Solaris. A brief description of the components/techniques used in the case study follows.

- Huffman Coding: this is a lossless approach based on statistical features of the file to be compressed. The more frequently the occurrence of a symbol, the smaller the bit code used to represent it.
- Arithmetic Coding: differently from Huffman Coding, this approach assigns a floating point output number to a sequence of input symbols. The number of bits in the output number is directly proportional to the size and complexity of the sequence to be compressed. It is also a lossless method.
- Burrows-Wheeler Transform: this method is based on sorting the input sequence in a certain way to improve the efficiency of compression algorithms. As the previous two methods, it is also a lossless approach.
- Embedded Zero-Tree Wavelet Encoder: this approach is based on the use of wavelet transforms to compress images (2D-signals). Extensions to other signal dimensions is also possible. Images are compressed using a progressive encoding with increasing accuracy. The better the accuracy, the better the quality of the image. Therefore, this is not a lossless approach.

– Fractal Image Encoding: this method is based on the representation of an image as an iterated function system (IFS). Blocks of the image that are similar are then equally represented. This is also a loss data approach.

Based on the characteristics of compression algorithms, four attributes are analyzed here: total execution time (TET), compression ratio (CR), maximum memory usage (MU), and root mean-square error (RMSE), where total execution time is the combination of compression plus decompression time and RMSE is a quality measure. Other features can be easily included according to users' needs. A set of 271 images were collect to evaluate the components. To capture the performance for small, medium, and large images, their size were uniformly distributed in the range from 10KB to 10MB. Also, different types of images were used (e.g., raw, pgm, jpg, and png). To compute the averages and standard deviations in Table 1, each component was executed 10 times for all the images.

	TET (in s)		MU (in KB)		CR		RMSE	
	avg	std	avg	std	avg	std	avg	std
HUFF	111.3	132.4	2135270	21.63	1.167	0.379	0	0
AREC	180.3	237.3	2135103	3.37	1.248	0.555	0	0
BWT	473.3	371.1	2137885	603.5	3.334	13.277	0	0
FRAC	89.5	73.3	2138986	2155.99	58.427	104.68	50.55	32.43
EZW	380.4	485.4	2142957	4732	4.226	0.837	39.35	27.91

Table 1. Average and standard deviation results for the five compression components and each feature of interest.

3.2 AHP Steps

The three steps defined in Section 2 and seen in Figure 1 are presented next in the context of the selection of components for image compression.

AHP Step 1 At this point the structure of the problem has already been defined as a three level hierarchical decomposition. The problem is the selection of image compression components; the four criteria of interest have been defined. The data for each of the available alternatives have also been collected, providing 2710 data points for each alternative.

AHP Step 2 The first step after defining the features to be considered in the computation of the rank of the five components is to decide their relative importance. Using Saaty's scale [22], the user defines the relative importance of each feature compare to each other. Since four features are under consideration here, a 4×4 matrix, as seen in Table 2 results. For example, if the user defines that TET is two times more important than CR then $IM_{1,2} = 2$; consequently,

	TET	CR	RMSE	MU
TET	1	2	1/3	5
CR	1/2	1	4	3
RMSE	3	1/4	1	2
MU	1/5	1/3	1/2	1

Table 2. Importance Matrix (IM) for the four features of interest for scenario 1.

$IM_{1,2} = 1/2$ in Table 2. If quality (represented by RMSE) is three times more important than MU then $IM_{3,4} = 3$ and $IM_{4,3} = 1/3$, as seen in Table 2.

The goal of creating the IM is to facilitate the computation of the weights of each feature. It is, in general, difficult for the user to directly compute the weights. Using the AHP approach, the user decides the relative importance of just two features, which greatly eases the task. The weights can now be computed by finding the eigenvector associated with the largest eigenvalue of IM. In this case, the eigenvector is

$$x^{IM} = [\ 0.5373\ 0.6616\ 0.5035\ 0.1417\ ]^T \tag{2}$$

In order to avoid scale issues, the values in x^{IM} are normalized to 1 using to Eq. 3. The measurements under consideration here are all in a ratio scale which is closed for arithmetic operations. Therefore, the normalized results obtained from Eq. 3 maintain the same scale properties as the original data.

$$x_i^{IM^N} = \frac{x_i^{IM}}{\sum_{j=1}^{4} x_j^{IM}} \tag{3}$$

The new normalized weights x^{IM^N} for the features are now:

$$\begin{aligned} x^{IM^N} &= [\ w_{TET}\ w_{CR}\ w_{RMSE}\ w_{MU}\ ]^T \\ &= [\ 0.2914\ 0.3588\ 0.2730\ 0.0768\ ]^T \end{aligned} \tag{4}$$

As can be verified from Eq. 4, the sum of the weights is 1. Also, it is clear that compression ratio (the weight of CR is represented by the second value in the vector) plays a more important role in the users selection while memory usage (the weight of MU is represented by the fourth value in the vector) has almost no influence. The other two features have similar importance.

The next step on the AHP approach is to compute pair-wise comparison matrices (PC matrices) for each feature. Using the average values of TET from Table 1 we can compute the entries for PC^{TET}. Each entry is computed using Eq. 5 below.

$$PC_{i,j}^{TET} = \frac{TET_j}{TET_i} \tag{5}$$

where TET_k, for $k = \{1(AREC), 2(HUFF), 3(BWT), 4(EZW), 5(FRAC)\}$ is the average total execution time (extracted from Table 1) for each of the alternatives. For example, $TET_{AREC} = 180$ and $TET_{HUFF} = 111$ resulting in

TET	AREC	HUFF	BWT	EZW	FRAC
AREC	1.0000	0.6172	2.6252	2.1101	0.4945
HUFF	1.6203	1.0000	4.2536	3.4189	0.8012
BWT	0.3809	0.2351	1.0000	0.8038	0.1883
EZW	0.4739	0.2925	1.2441	1.0000	0.2343
FRAC	2.0224	1.2482	5.3093	4.2675	1.0000

Table 3. Pair-wise comparison (PC^{TET}) matrix for total execution time (TET) for the five available components.

$PC^{TET}_{2,1} = 1.62$. It should be noticed that execution time has an inverse effect, that is, the lower the execution time the better. Therefore, the numerator and denominator in Eq. 5 are inverted. The same is true for RMSE and MU. Table 3 presents the results of the pair-wise comparison for PC^{TET}. As before, to find the rank/weight for execution time for each alternative, we compute the normalized (to avoid scale issues when comparing, for example, execution time with RMSE) eigenvector associated with the largest eigenvalue for PC^{TET}. This results in the values in Eq. 6

$$x^{TET^N} = [\ 0.1819\ 0.2947\ 0.0693\ 0.0862\ 0.3679\ ]^T \qquad (6)$$

Based solely on execution time and Eq. 6, the rank for the components would be: 1st-FRAC, 2nd-HUFF, 3rd-AREC, 4th-EZW, and 5th-BWT. Sorting the execution time column of Table 1 leads to the exactly same order which is a good indication of the accuracy of the approach. Therefore, one could argue against the computation of eigenvalues and eigenvectors when it is much easier to simply sort the average execution time. This could be true when only one criterion is being considered, but does not apply for multi-criteria problems as the one described here. Therefore, we need to consider the computation of weights for the remaining features.

MU	AREC	HUFF	BWT	EZW	FRAC
AREC	1.0000	1.0001	1.0013	1.0037	1.0018
HUFF	0.9999	1.0000	1.0012	1.0036	1.0017
BWT	0.9987	0.9988	1.0000	1.0024	1.0005
EZW	0.9963	0.9964	0.9976	1.0000	0.9981
FRAC	0.9982	0.9983	0.9995	1.0019	1.0000

Table 4. Pair-wise comparison (PC^{MU}) matrix for memory usage (MU) for the five available components.

The pair-wise comparison matrices PC^{MU} for memory usage and PC^{RMSE} for root mean square error are computed similarly to what has been done to PC^{TET}. They are shown, respectively, in Tables 4 and 5. To avoid division

RMSE	AREC	HUFF	BWT	EZW	FRAC
AREC	1.0000	1.0000	1.0000	40.3588	51.5568
HUFF	1.0000	1.0000	1.0000	40.3588	51.5568
BWT	1.0000	1.0000	1.0000	40.3588	51.5568
EZW	0.0248	0.0248	0.0248	1.0000	1.2775
FRAC	0.0194	0.0194	0.0194	0.7828	1.0000

Table 5. Pair-wise comparison (PC^{RMSE}) matrix for root mean square error (RMSE) for the five available components.

by zero problems, a value 1 has been added to each entry, related to RMSE, in Table 1. Now, following exactly the same steps used in the computation of x^{TET^N}, the eigenvectors x^{MU^N} and x^{RMSE^N} are computed and the results are shown in Eqs. 7 and 8.

$$x^{MU^N} = [\ 0.2003\ 0.2003\ 0.2000\ 0.1995\ 0.1999\ ]^T \tag{7}$$

$$x^{RMSE^N} = [\ 0.3285\ 0.3285\ 0.3285\ 0.0081\ 0.0064\ ]^T \tag{8}$$

As expected, due to only small variations on the amount of memory used by each approach, there is little difference in the weights in x^{MU^N}. Also, as can be seen in Eq. 8, the weights for AREC, HUFF, and BWT are the same and are much larger than the weights for EZW and FRAC. This behavior is expected since the first three are lossless approaches.

The computation of the weights for Compression Ratio (CR) presents a small distinction when compare to the previous features. The computation of the weights for TET, MU, and RMSE are based on an inverted gain, i.e., the smaller the value, the better. This is captured in Eq. 5 by switching the numerator and denominator. In the case of CR (the larger the ratio the better), such inversion is not necessary which leads to the use Eq. 9.

$$PC^{CR}_{i,j} = \frac{CR_i}{CR_j} \tag{9}$$

CR	AREC	HUFF	BWT	EZW	FRAC
AREC	1.0000	1.0691	0.3743	0.2953	0.0214
HUFF	0.9354	1.0000	0.3501	0.2762	0.0200
BWT	2.6717	2.8563	1.0000	0.7890	0.0571
EZW	3.3861	3.6201	1.2674	1.0000	0.0723
FRAC	46.8079	50.0414	17.5198	13.8234	1.0000

Table 6. Pair-wise comparison (PC^{CR}) matrix for compression ratio (CR) for the five available components.

Using the values from Table 1 and Eq. 9 leads to matrix PC^{CR} presented in Table 6. The computation of the normalized eigenvector x^{CR^N} is done as before resulting in the values presented in Eq. 10

$$x^{CR^N} = [\ 0.0182\ 0.0171\ 0.0488\ 0.0618\ 0.8541\ ]^T \tag{10}$$

	TET $w_{TET} = 0.2914$	CR $w_{CR} = 0.3588$	RMSE $w_{RMSE} = 0.2730$	MU $w_{MU} = 0.0768$
AREC	0.1819	0.0182	0.3285	0.2003
HUFF	0.2947	0.0171	0.3285	0.2003
BWT	0.0693	0.0488	0.3285	0.2000
EZW	0.0862	0.0618	0.0081	0.1995
FRAC	0.3679	0.8541	0.0064	0.1999

Table 7. Synthesis Matrix (SM) for the ranking of the five compression components.

AHP Step 3 The final step in the computation of the rank for the five components is to combine the weights for each feature [w_{TET} w_{CR} w_{RMSE} w_{MU}] (line 1 in Table 7) with the weights computed for all the pair-wise comparison matrices. Each column in Table 7 correspond respectively to x^{TET^N}, x^{CR^N}, x^{RMSE^N}, and x^{MU^N}. Now, using Eq. 1 from Section 2 leads us to the results in Table 8. All the values used up to this point are referred to as Scenario 1 in Table 8.

Alternative	Scenario 1 - Rank		Scenario 2 - Rank	
Components	Value	Position	Value	Position
AREC	0.164627	(3)	0.236828	(2)
HUFF	0.197080	(2)	0.263210	(1)
BWT	0.142737	(4)	0.214710	(4)
EZW	0.064837	(5)	0.052667	(5)
FRAC	0.430719	(1)	0.232586	(3)

Table 8.

FRAC has the first position in the rank, as seen in Table 8. This is clearly the best choice as FRAC has the best response time (TET) and by far the best compression ratio (CR). These two features have the two highest weights, as seen in Eq. 4. The difference of these two features for FRAC is so large that it compensates the poor quality of the compressed images (due to a high RMSE). The second and third positions in the rank, respectively HUFF and AREC, present average execution time and reasonably compression ratio when compare to the two approaches in the bottom of the rank. In addition, they are lossless approaches justifying their rank placement. EZW is clearly the last place in the rank; it has poor response time, the compression ratio is not outstanding, and RMSE is high. We can conclude that, for this scenario, the rank computed using AHP has been able to capture the user's preference in an adequate manner.

Now let us assume a different scenario, refer hereafter as Scenario 2. To compute the rank of the components for a new scenario, only the importance matrix needs to be changed. That is, what is changing are the user's preferences (captured now by a new IM in Table 9) and not the comparative behavior of the components (still captured by PC^{TET}, PC^{CR}, PC^{RMSE}, and PC^{MU}). As can be seen in Table 9, quality (RMSE) has now a much higher importance than in the previous scenario.

	TET	CR	RMSE	MU
TET	1	2	1/5	5
CR	1/2	1	1/4	3
RMSE	5	4	1	2
MU	1/5	1/3	1/2	1

Table 9. Importance Matrix for the four features of interest for scenario 2.

The new normalized weights x^{IM^N} for scenario 2 are presented in Eq. 11.

$$x^{IM^N} = [\ 0.2353\ 0.1445\ 0.5240\ 0.0961\ ]^T \tag{11}$$

Replacing the values from Eq. 11 in Table 7 leads to the results of Scenario 2 in Table 8. As can be seen, the increase of the importance for quality (RMSE) results in two lossless components (HUFF and AREC) overtaking FRAC in the rank. Again, AHP has been able to properly select the best alternatives under a given scenario. Additional scenarios were also used in this study. In all cases, the AHP approach has been able to rank the components and identify the best alternative.

As we can see from the two scenarios described above, changes in IM lead to changes in the selection of components. Therefore it is important to know how sensitivity is IM to these changes. Initial experiments indicate that the sensitivity depends not only on the values of IM but also on the values for SM. That is, if the values in one column of SM are close, a small change in IM may change the rank of the components. However, if the values are far apart, changes in IM need to be more significant in order to affect the rank.

4 Related Work

Diverse approaches to component selection have been proposed such as processes that use MCDM [2, 8–12], keyword matching combined with knowledge based approaches [24, 25], analogy based approaches (including case based reasoning) [26–28], and fuzzy logic [18, 29].

The AHP, an established MCDM approach, has been adopted in component selection approaches that have reported successful case studies [9–11]. OTSO, for example, is one of the earliest component selection approaches that uses AHP [9]. The OTSO process is composed of subprocesses to search, screen, and evaluate component alternatives; there is an additional subprocess to rigorously define the evaluation criteria. The evaluation definition subprocess refines the requirements for the components into a hierarchical decomposition. The evaluation criteria include functional, quality (non-functional), business concerns, and relevant software architecture. The AHP was selected in OTSO for the component evaluation because it provided a systematic, validated MCDM approach. The OTSO approach has been applied in two case studies with Hughes corporation in a program that develops systems to integrate and make available Earth environment data from satellites [9]. One study has involved the selection of a hypertext browser tool. Initially, the search resulted in over 48 tools; the screening process reduced this to four tools which were evaluated with respect to evaluation criteria refined from the initial requirements. The evaluation required 322 pairwise comparisons by the evaluators. This was deemed feasible because AHP tool support was used. The results of the case study indicated that the AHP approach produced relevant information for component selection and the information was perceived as more reliable by decision makers. ArchDesigner [30, 31] is another example of a component selection approach based on AHP. Their results are also a indication of AHP's applicability for the component selection problem.

The strengths and limitations of using AHP have been discussed within the context of component selection [1, 13]. The strengths presented include that it only requires a pairwise comparisons of alternatives and a pairwise weighting of selection criteria, which reduces the burden on experts and it enables consistency analysis of the comparisons and weights, which allows the quality of the information on the criteria and alternatives to be assessed. In addition, the AHP allows the use of a relative ratio scale [1..9] or real data for the comparisons [23] and has been successfully used in component selection approaches, as reported in case studies.

The limitations presented include the assumption of independence among the evaluation criteria, the difficulty in scaling the approach to problems with a large number of comparisons, which would burden the experts in a manual approach, and determining if the AHP is the best approach among the many MCDM alternatives available. Stating these limitations, the approach has been questioned by some for its usefulness in component based software engineering.

Given the arguments that support and question the use of AHP in component selection, we believe a more extensive empirical study investigating the effectiveness of using the AHP approach for component selection in a semi-automated approach is needed.

5 Conclusions and Future Work

An empirical study is presented in this work to investigate using the AHP approach to select components using non-functional criteria. The application of AHP has been constrained to non-functional attributes of functionally equivalent components. The approach is semi-automated, to provide scalability. The importance matrix, which reflects the required behavior of the component, is created manually. Data about the non-functional behavior of a set of compression components are collected including memory usage, response time, root mean square error, and compression ratio; the data are used to automatically create the pairwise comparison decision tables for the criteria. The data collected about the behavior of the components (e.g., criteria such as memory and response time) reflect the intrinsic synergistic and conflicting relationships among the criteria in the components. Once the importance matrix and the pairwise comparison tables are available, the data are synthesized and a ranking is automatically calculated. The results of the study indicate the AHP is an effective ranking approach. This approach can be applied to the selection of components using other non-functional criteria, however, designing the data collection for the components' behavior may be non-trivial.

The limitations of the study include that a comparison on the use of AHP and alternative MCDM techniques is not considered. This study needs to be conducted in the future work. In addition, the limitation of using the AHP with respect to the assumption of independence is not addressed in this study. In the future, the composition of multiple components that will integrate together and provide a set of functional capabilities with minimal overlap will be investigated. The extension can be considered as follows. First, non-functional requirements can be used to rank the components as done in this paper, let us refer to this rank as R_{nf}. Now, let us assume that a set of K functionalities is desired $F = \{f_1, f_2, \ldots, f_K\}$. An IM matrix comparing the relative importance of each functionality f_i can be used to define their weights. PC matrices can then be constructed to verify which alternative implements each of the functionalities. The availability of these matrices allows for the computation of a new rank R_f accounting for the functional requirements. Ranks R_{nf} and R_f can now be combined to compute the final rank of the available components. Let us assume a component C_j is the first choice in that rank.

It is likely that the selected component C_j does not implement all the desired functions. In this case, the process above can be repeated for the remaining components but now focusing only on the functionality that is not implemented by C_j. This process can be repeated until all functional requirements have been satisfied. Clearly, there are compositional issues that still need to be addressed. However, the goal of this section is to present a potential expansion of the use of AHP and not to validate it.

References

1. C. Alves and A. Finkelstein, "Challenges in cots decision-making: a goal-driven requirements engineering perspective," in *Proceedings of the 14th international conference on Software engineering and knowledge Engineering*, (Ischia, Italy), pp. 789–794, 2002.
2. "Commercial-off-the-shelf (cots) evaluation, selection, and qualification process." Systems Engineering Process Office - Space and Naval Warfare Systems Center, October 2002.
3. F. Navarrete, P. Botella, and X. Franch, "How agile cots selection methods are (and can be)?," in *31st EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 160–167, 30 Aug.-3 Sept. 2005.
4. C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.
5. S. Wadekar and S. Gokhale, "Exploring cost and reliability tradeoffs in architectural alternatives using a genetic algorithm," in *Proceeding of 10th International Symposium on Software Reliability Engineering*, pp. 104–113, November 1999.
6. T. Tseng, W. Liang, C. Huang, and T. Chian, "Applying genetic algorithm for the development of the components-based embedded system," *Computer Standards & Interfaces*, vol. 27, no. 6, pp. 621–635, 2005.
7. W. Chang, C. Wu, and C. Chang, "Optimizing dynamic web service component composition by using evolutionary algorithms," in *Proceeding of 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 708–711, September 2005.
8. B. C. Phillips and S. M.Polen, "Add decision analysis to your cots selection process," *CrossTalk the journal of defense software engineering*, April 2002. available at: http://www.stsc.hill.af.mil/crosstalk/2002/04/index.html.
9. J. Kontio, "A cots selection method and experiences of its use," in *Proceedings of the 20th Annual Software Engineering Workshop*, (Maryland), November 1995.
10. C. Ncube and N. Maiden, "Guiding parallel requirements acquisition and cots software selection," in *Proceedings of the IEEE International Symposium on Requirements Engineering*, pp. 133–140, 1999.
11. A. Lozano-Tello and A. Gomez-Perez, "Baremo: how to choose the appropriate software component using the analytic hierarchy process," in *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, (Ischia, Italy), pp. 781 – 788, 2002.
12. C. Ncube and N. Maiden, "Selecting cots anti-virus software for an international bank: Some lessons learned," in *Proceedings 1st MPEC Workshop*, 2004.
13. L. C. Briand, "Cots evaluation and selection," in *Proceedings of the International Conference on Software Maintenance*, pp. 222–223, 1998.
14. W. Frakes and T. Pole, "An empirical study of representation methods for reusable software components," *IEEE Transactions on Software Engineering*, vol. 20, pp. 617–630, August 1994.
15. A. Orso, M. J. Harrold, D. Rosenblum, G. Rothermel, M. L. Soffa, and H. Do, "Using component metacontents to support the regression testing of component-based software," in *Proceedings of the IEEE International Conference on Software Maintenance (ICSM)*, November 2001.
16. P. Vitharana, F. M. Zahedi, and H. Jain, "Knowledge-based repository scheme for storing and retrieving business components: A theoretical design and an empirical

analysis," *IEEE Transactions on Software Engineering*, vol. 29, pp. 649–664, July 2003.

17. J. Li, F. O. Bjoernson, R. Conradi, and V. B. Kampenes, "An empirical study of variations in cots-based software development processes in norwegian it industry," in *Proceedings. 10th International Symposium on Software Metrics*, pp. 72 – 83, 2004.
18. K. Cooper, J. W. Cangussu, R. Lin, G. Sankaranarayanan, R. Soundararadjane, and E. Wong, "An empirical study on the specification and selection of components using fuzzy logic," in *Lecture Notes in Computer Science*, vol. 3489, pp. 155–170, Springer-Verlag, April 2005. Eighth International SIGSOFT Symposium on Component-based Software Engineering (CBSE 2005), Co-Located with ICSE-2005, St. Louis, Missouri, May 15-21, 2005.
19. K. R. Hammond, J.S. and H. Raiffa, *Smart Choices A Practical Guide to Making Better Decisions.* Harvard Business School Press, 1999.
20. M. Mollaghasemi and J. Pet-Edwards, *Making Multiple-Objective Decisions.* IEEE Computer Society Press, 1997.
21. E. Triantaphyllou, *Multi-criteria decision making methods: a comparative study.* Kluwer Academic Publishers, 2000.
22. T. Saaty and L. Vargas, *Methods, Concepts & Applications of the Analytic Hierarchy Process.* Kluwer Academic Publishers, 2001.
23. T. Saaty, *Fundamentals of Decision Making and Priority Theory.* RWS Publications, 1994.
24. R. Seacord, D. Mundie, and S. Boonsiri, "K-bacee: Knowledge-based automated component ensemble evaluation," in *Proceedings of the 2001 Workshop on Component-Based Software Engineering*, 2001.
25. L. Chung and K. Cooper, "A cots-aware requirements engineering process: a goal- and agent oriented approach," in *Proceedings of the International Council on Systems Engineering Symposium*, (Las Vegas, Nevada), 2002.
26. B. H. C. Cheng and J.-J. Jeng, "Reusing analogous components," *IEEE Transactions on Knowledge and Data Engineering*, vol. 9, pp. 341–349, March-April 1997.
27. P. Gonzalex, "Applying knowledge modelling and case-based reasoning to software reuse," *IEE Proceedings Software*, vol. 147, pp. 169–177, October 2000.
28. M. Gu, A. Aamodt, and X. Tong, "Component retrieval using conversational case-based reasoning," in *Proceedings of the ICIIP, International Conference on Intelligent Information Systems*, (Beijing, China), October 21-23 2004.
29. T. Zhang, L. Benini, and G. D. Micheli, "Component selection and matching for ip-based design," in *Proc. of Conference and Exhibition on Design, Automation and Test in Europe.*
30. A.Liu and I. Gorton, "Accelerating cots middleware technology acquisition: the i-mate process," *IEEE Software*, vol. 20, pp. 72–79, March/April 2003.
31. T.Al-Naeem, I.Gorton, M. A. Babar, F.Rahbi, and B. Boualem, "A quality-driven systematic approach for architecting distributed software applications," in *International Conference on Software Engineering (ICSE).*