An Empirical Study on the Specification and Selection of Components Using Fuzzy Logic

Kendra Cooper[1], João W. Cangussu[1], Rong Lin[1,] Ganesan Sankaranarayanan[1]
Ragouramane Soundararadjane[1], and Eric Wong[1]

[1] The University of Texas at Dallas
Mail Station EC 31, 2601 N. Floyd Rd.
Richardson, Texas, USA
{kcooper, cangussu, rxl029000, sxg013900, rxs011610, ewong}@utdallas.edu

Abstract. The rigorous specification of components is necessary to support their selection, adaptation, and integration in component-based software engineering techniques. The specification needs to include the functional and non-functional attributes. The non-functional part of the specification is particularly challenging, as these attributes are often described subjectively, such as *Fast Performance* or *Low Memory*. Here, we propose the use of infinite value logic, fuzzy logic, to formally specify components. A significant advantage of fuzzy logic is that it supports linguistic variables, or hedges (e.g., terms such as slow, fast, very fast, etc.), which are convenient for describing non-functional attributes. In this paper, a new systematic approach for the specification of components using fuzzy logic is presented. First, an empirical study is conducted to gather data on five components that provide data compression capabilities; each uses a different algorithm (Arithmetic Encoding, Huffman, Wavelet, Fractal, and Burrows-Wheeler Transform). Data on the response time performance, memory use, compression ratio, and root mean square error are collected by executing the components on a collection of 75 images with different file formats and sizes. The data are fuzzified and represented as membership functions. The fuzzy component specifications are ranked using a set of test queries. Fuzzy multi-criteria decision making algorithms are going to be investigated for the selection of components in the next phase of the work.

1 Introduction

Component-based software engineering (CBSE) techniques are of keen interest to researchers and practitioners as they hold promise to support the timely, cost effective development of large-scale, complex systems. Such techniques are crucial to effectively meet the needs of rapidly changing business environments.

Effective CBSE techniques need to address a complex set of problems, as the issues are inter-related and span business, legal, and technical areas [1][2][3][4][5]. The pre-

diction of short and long term costs and benefits of using components needs to be supported, as project managers consider the cost of the components, quality of the vendor, lack of control over the support and evolution of the COTS components, licensing issues, and the associated risks. Technical issues include the specification, certification, selection, and composition of sets of components, and the impact on the concurrent, iterative modeling of various software engineering artifacts.

The term *component* has a wide variety of definitions in the literature [6]. Here, a software component is an independent, reusable blackbox that, ideally, has a comprehensive interface description including:

- A specification of the functional and non-functional capabilities of the component. The non-functional description includes quality of service attributes (response time performance, memory, etc.)

- A programmer interface description, which defines how to use the component (e.g., an API definition with method names, parameter lists, return types, etc.)

The specification of components has been considered using a variety of notations and approaches. Formal notations offer a means to specify components concisely and unambiguously. XML [7][8], fuzzy logic [8], first-order logic [9], the RESOLVE notation [10], as well as architectural description languages and coordination languages [11] have been proposed for formally representing component specifications. The well known Unified Modeling Language (UML), a semi-formal notation, has also been used to specify components [12]. Alternative approaches including interface definitions [13] and templates [14] have also been investigated, which are based on informal, English descriptions.

A key issue in the specification of components is the problem of how to represent and reason about the non-functional attributes, such as performance, adaptability, security, etc. These attributes are often described subjectively, using terms like "extremely fast," "very fast," "quite fast," and "moderately fast" rather than simply "fast" and "not fast." Fuzzy logic, developed as a solution to this kind of problem, provides an effective way to represent and reason with such vague and ambiguous terms. It provides a theoretical foundation for approximate reasoning using imprecise propositions, which is based on fuzzy set theory.

This paper presents an approach that utilizes fuzzy logic in the specification of software components. Fuzzy logic has been applied to a number of interesting problems in a wide variety of domains such as medicine, manufacturing, software engineering, etc. [15][16]. Currently, it has received very limited attention from the CBSE community. The approach described in Section 4 uses fuzzy logic to select components but does not present a well defined process to create fuzzy specifications for the non-functional attributes of the components [8].

The process to define the fuzzy specification of the components is accomplished in steps. The software components that provide specific functional capabilities are executed in order to collect data about their non-functional behavior. Data are collected for their response time performance, memory use, and quality attributes that are of particular interest for the component. For example, quality attributes for components that provide compression capabilities include the root mean square error (RMSE) and

compression ratios. The data are fuzzified (i.e., the data are represented as a collection of fuzzy membership functions) and the components are ranked using a set of test queries.

The approach is illustrated using a study involving a collection of components that provide data compression capabilities. Each component implements a different algorithm: Arithmetic Encoding, Huffman coding, Burrows-Wheeler Transform (BWT), Fractal Image Encoding, and Embedded zero-tree wavelet encoder. The image compression components are executed on a single platform with over 75 images (of various formats like jpg, pgm, raw, and bmp). Execution time (system time + user time) and memory usage for compression and decompression components, the compression ratio, and the RMSE are collected. Once gathered, the data are fuzzified (i.e., represented as fuzzy membership functions). Each fuzzy specification is composed of the requirements interface (functional and non-functional capabilities) and the programmer interface (e.g., API definition) represented as fuzzy membership functions. In addition, design and implementation constraints such as programming language, code form (e.g., source code, executable, byte code), as well as the execution environment (e.g., machine hardware = sun4u, OS version = 5.9, Processor type = sparc, Hardware = SUNW, Sun-Fire-280R) are captured in the specification. The fuzzy representations of the components are ranked using a set of test queries.

This paper is structured as follows. An overview of fuzzy logic is presented in Section 2 as background material. The approach to build and rank a collection of fuzzy components is presented in Section 3. The example study is illustrated in Section 4. Section 5 presents some related work while conclusions and future work are in Section 6.

2   Fuzzy Logic

When one admits that nothing is certain one must, I think also add that some things are more nearly certain than others.

- Bertrand Russell

Numerous approaches are available to represent and reason about uncertainty including probabilistic reasoning, neural network theory, and fuzzy logic [17]. Fuzzy logic was proposed by Zadeh in the 1960's [18] while working on the problem of natural language processing. Natural language, like many activities in the real world, is not easily translated into the absolute terms of 0 and 1 provided by classical logic. Fuzzy logic is well suited for representing and reasoning with vague and ambiguous conditions, where an exact true or false value cannot always be determined. Instead, degrees of truth and falsity are needed. Fuzzy logic offers infinite set of values [0.0..1.0].

Suppose, for example, you ask the following question to a group of people: ***Is the component very fast?*** The variety of responses reflects the individuals' experiences and knowledge. For example, the component vendor may reply 1 (true); an engineer in a high performance domain may reply 0.6 (somewhat true); an engineer in an informa-

tion system domain may reply 0.8 (quite true), etc. People can achieve this high level of abstraction when considering such questions; fuzzy logic has been compared to the human decision making process. The ultimate goal of fuzzy logic is to provide a theoretical foundation for approximate reasoning using imprecise propositions based on fuzzy set theory.

Fuzzy sets. A classical (crisp) set is defined as a collection of elements; each element either belongs or does not belong to a set. These classical sets can be described by enumerating the elements of the list, using conditions for membership (e.g., set A only contains elements less than 5), or by using a characteristic function to define the member elements, where 1 indicates membership and 0 indicates non-membership.

The definition of set membership is modified for fuzzy sets, in which a characteristic function gives a degree of membership for each element of a given set. Membership functions can be formed using piecewise lines (e.g., triangular or trapezoidal), Gaussian distribution, Bell, Sigmoid, quadratic and cubic polynomial curves, etc. In this paper, the generalized bell membership function is used, because it generated the least error in the fuzzification of the data collected for the components. The generalized bell function depends on three parameters a, b, and c as given by Equation (1). These parameters determine the shape of the curve. The parameter a determines the width of the curve, b is related to the slope at the point $c+a$, and c locates the center of the curve.

$$F(x; a, b, c) = \frac{1}{1 + \left| \dfrac{x - c}{a} \right|^{2b}} \tag{1}$$

Fuzzy Definitions. Basic definitions for fuzzy logic are presented below, followed by an example that clarifies the semantic distinction between fuzzy logic and probability theory.

Definition 1: Let X be some set of objects, with elements noted as x. Thus, X = {x}.

Definition 2: A fuzzy set A in X is characterized by a membership function mA(x) mA: x -> y, y ∈ R | 0.0 ≤ y ≤ 1.0   In other words, the membership function maps each element in X onto the real interval [0.0, 1.0]. As mA(x) approaches 1.0, the degree of membership of x in A increases.

Definition 3: Empty fuzzy set. A is EMPTY ⟺ ∀x mA(x) = 0.0.

Definition 4: Fuzzy set equality. A = B ⟺ ∀x mA(x) = mB(x)

Definition 5: Fuzzy subset. A is CONTAINED in B ⟺ ∀x mA(x) ≤ mB(x).

Definition 6: NOT operator (Complement of the fuzzy set). NOT (mA (x)) = 1 - mA(x).

Definition 7. OR operator (Union of fuzzy sets). C = A OR B, where: mC(x) = MAX(mA(x), mB(x)).

Definition 8: AND operator (Intersection of fuzzy sets). C = A AND B where: mC(x) = MIN(mA(x), mB(x)).

Definition 9: IMPLICATION operator. A $\rightarrow$ B = (NOT A) OR B = MAX{1-A, MIN{A,B}}

In a fuzzy system, the rules have the form: IF (x1 AND x2 AND ... AND *xn*) THEN *y*.

Fuzzy Logic vs. Probabilistic Theory. The last two fuzzy set operations, AND and OR, clearly illustrate the semantic differences from their counterparts in probabilistic theory. Suppose, for example, x = Fred, P is the fuzzy set of pleasant people, and A is the fuzzy set of athletic people. Let's use P(x) = 0.90 and A(x) = 0.90 to describe the probability that Fred is very pleasant and Fred is very athletic and mP(x) = 0.90 and mA(x) = 0.90 to describe the memberships functions that Fred is very pleasant and Fred is very athletic. The probabilistic calculation is P(x) * A(x) = 0.81, whereas the fuzzy result is MIN(mP(x), mA(x)) = 0.90. Therefore, the probabilistic calculation yields the statement:

If Fred is very pleasant and Fred is very athletic, then Fred is a quite pleasant, athletic person, using 0.81 to describe "quite".

The fuzzy calculation, however, yields:
If Fred is very pleasant and Fred is very athletic, then Fred is a very pleasant, athletic person, once again using 0.90 to describe "very".

As more factors are included into the equations (the fuzzy set of intelligent people, tall people, wealthy people, etc.), the result of a series of AND's approaches 0.0 in the probabilistic calculation, even if all factors are initially high. Our intuition for the problem better matches the fuzzy set calculation in which, for example, five factors of the value 0.90 ("very") ANDed together gives the answer 0.90 ("very"), not anything lower.

The probabilistic version of A OR B is (A+B - A*B), which approaches 1.0 as additional factors are considered. Again, the fuzzy set calculation uses the maximum of the membership values to limit the resulting membership degree.

It is important to note that the assignment of values to linguistic meanings (such as 0.90 to "very") and vice versa is subjective. Fuzzy systems do not claim to establish formal procedure for assignments at this level. What fuzzy logic does propose is to establish a formal method of operating on these values, once the primitives have been established.

3 Fuzzy Component Specification Process: A Case Study on Compression Components

A series of steps are defined to create a fuzzy specification that can be used to facilitate the analysis and ranking of components according to some specific criteria. Each step in our approach is described next along with a case study based on image compression algorithms.

Step 1 - **Component collection:** Before actually starting specifying components using fuzzy logic, a set of functionally equivalent components must be selected. Assume we are interested in a certain set of functionality $F=\{f_1, f_2, ..., f_n\}$. Then, a collection of components $C=\{c_1, c_2, ..., c_m\}$ that implements F is formed. In addition to F the components may present any other extra functionality.

The problem of compressing images is an interesting one. Some algorithms take less time but use more memory, while others behave the other way around. Quality issues, such as the compression ratio or RMSE, also have trade-offs in different algorithms. For example, an algorithm may be acceptable in terms of time and memory, but have a poor compression ratio. In some cases image restoration is compromised to achieve a good compression ratio. In short, Image Compression components have clear trade-offs, in terms of execution time, memory, compression ratio, and RMSE. Also since there are many well-established algorithms and their implementations are available over the Internet, image compression components have been chosen for this study and $F=\{\text{image compression}\}$.

A survey of image compression algorithms has been performed; the components have been chosen in such a way that there is a clear trade-off in terms of execution time, compression ratio, and the root mean square error (RMSE) between the original image and the uncompressed image. Available components have been selected, their non-functional features analyzed, and the results used to derive a fuzzy specification of these features. Five image compression algorithms have been chosen for this study: Arithmetic Encoding (AREC), Huffman coding (HUFF), Burrows-Wheeler Transform (BWT), Fractal Image Encoding (FRAC), and Embedded Zero-Tree Wavelet Encoder (WAVE). Therefore $C=\{\text{AREC,HUFF,BWT,FRAC,WAVE}\}$. Source code for these components is available over the Internet. As a first step, the credibility of the source code for both compression and un-compression 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. It should be clear that the availability of source code is not required for the fuzzy specification of the components.

Step 2 - **Input fuzzification:** Based on the functionality set F, the input attributes $A=\{a_1, a_2, ..., a_q\}$ affecting F are determined. Then, a body of distinct test inputs $T=\{t_1, t_2, ..., t_j\}$ is created to exercise all the components in C; data are collected. Neuro-adaptive learning techniques [16] are then used to create K_i membership functions for

each attribute a_i in A. The value of K_i depends on the characteristics of a_i and the desired granularity. For example, if a_i represents the size of the input, one could select $K_i =3$ for {small, medium, large}, $K_i =5$ for {small, medium, large, very large, extremely large}, or any other representative set of the linguistic characteristics of a_i.

The inputs affecting the functional and non-functional behavior of the distinct components are identified in this step. In the case of compression algorithms, two features are of interest. The first is the type of images a compression algorithm can handle (i.e., the file format). The second refers to how the size of an image affects the component's non-functional behavior. When searching for a compression component, one could think of the following query: "Search for a component that has low memory usage for large images". What is the size of a large image is the immediate question that comes from this query. As expected, the definition of a large (very small, small, medium, very large, etc.) image is not crisp but rather fuzzy, therefore justifying the fuzzy specification approach proposed here. This results in a single element set for $A=${image size}.

In order to fuzzify the input size, images of various sizes and types were downloaded from the Internet, where the size ranged from 11 KB to 4096 KB. Also, components depend on the image formats like raw or pgm. Due to the difficulty of finding images in raw or pgm formats, ReaConverter Pro v3.4 has been used to convert some images to the desired format. In effect, the study was conducted on 75 images of type jpg, raw, and pgm leading to $T=\{t_1,t_2,...,t_{74},t_{75}\}$. With these images three ($K=3$) membership functions have been generated (using MatLab Fuzzy Logic Toolbox) based on the size of the images. The choice of three membership functions (small, medium, and large) is arbitrary and done here to simplify the results of the case study. There is no restriction in selecting a different number of functions. Neuro-adaptive learning techniques [16] have been used to generate the size based membership function depicted in Figure 1. Also a generalized bell format has been selected for the membership functions due to its concise and powerful representation features. We are aware that a much larger set of images would be required to fully capture the fuzzy features of image size. However, the selected set appears to be large enough to demonstrate the applicability of the proposed approach.

Step 3 - **Non-functional attributes selection:** A set of non-functional attributes $NF=\{nf_1,nf_2,...,nf_p\}$ for the components is selected. This set must express important attributes related to the components. Also, some attributes in NF may not be applicable/identifiable to all the components

Based on the characteristics of compression algorithms, four attributes are analyzed here: $NF=${total execution time, compression ratio, maximum memory usage, 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.
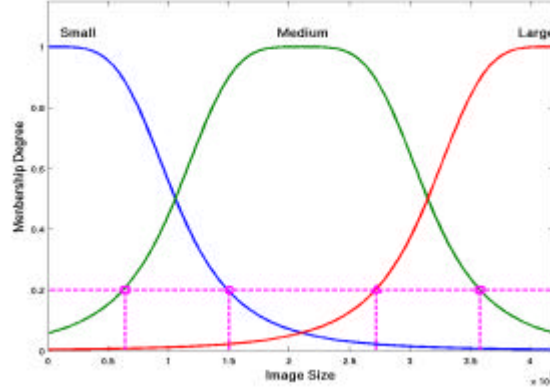
Figure 1: Fuzzification of input based on image size: three membership functions for small, medium, and large images.

Step 4 - **Component independent fuzzification of non-functional attributes:** In this step all the components in *C* are executed for all test inputs in *T*, and measurements of all attributes in *NF* are collected. As for the input fuzzification (Step 2), neuro-adaptive learning techniques are used to create K_i membership functions for each of the nf_i attributes in *NF*. Notice that the measurements for the *NF* attributes are collected for the set of components; the membership functions are created based on all combined results and not on individual components.

Once the features to be analyzed are selected in Step 3, the components are executed for all small images, and data for the four features are collected. The same is done for medium and large images. The selection of images (that is, the creation of the subsets *TS*) is done according to the membership functions in Figure 1. Notice that if any degree of membership is selected for the three functions, then the three sets of small, medium, and large images would be the same. Therefore, a cutoff for the membership degree has to be determined to decrease the overlap. In the case of Figure 1, a cutoff of 0.5 reduces the overlap to zero; due to the fuzzy characteristics of the problem, some overlap is desired. In general, the smaller the cutoff, the larger the overlap and consequently, the less distinct the results of the fuzzification. Experiments with distinct cutoffs have been conducted. We limit our description here to a specified cutoff of 0.2 for the degree of membership.

Using a 0.2 cutoff results in small images with size less than 1.5 MB, medium images with size ranging from 0.7 MB to 3.6 MB, and large images with size greater than 2.8 MB. Now, each component is executed for the three sets of images, and the collected data are used to generate membership functions for each set of images and features. Again, a generalized bell curve is used to create the membership functions. However, no learning technique needs to be applied in this case since the image sets already have the information we need. After the execution of a component for all small images, the average and standard deviation for a specific feature is computed and used to create the membership function. The same procedure is followed for medium and large size images. Figure 2 shows the results for the Wavelet component. As expected, the

execution time increases as the size of the images increases. In this case, we note a large overlap between the three curves. Compression ratio follows the same scheme as execution time (the larger the image, the better the compression ratio) but with more distinct membership functions, i.e., less overlap. As seen in Figure 2, the quality of the image measured by the RMSE is better for large images than for medium or small size images. Again, there is less overlap in this case than the membership functions for execution time. The memory usage for the Wavelet component presents very distinct results with almost no overlap between the curves.
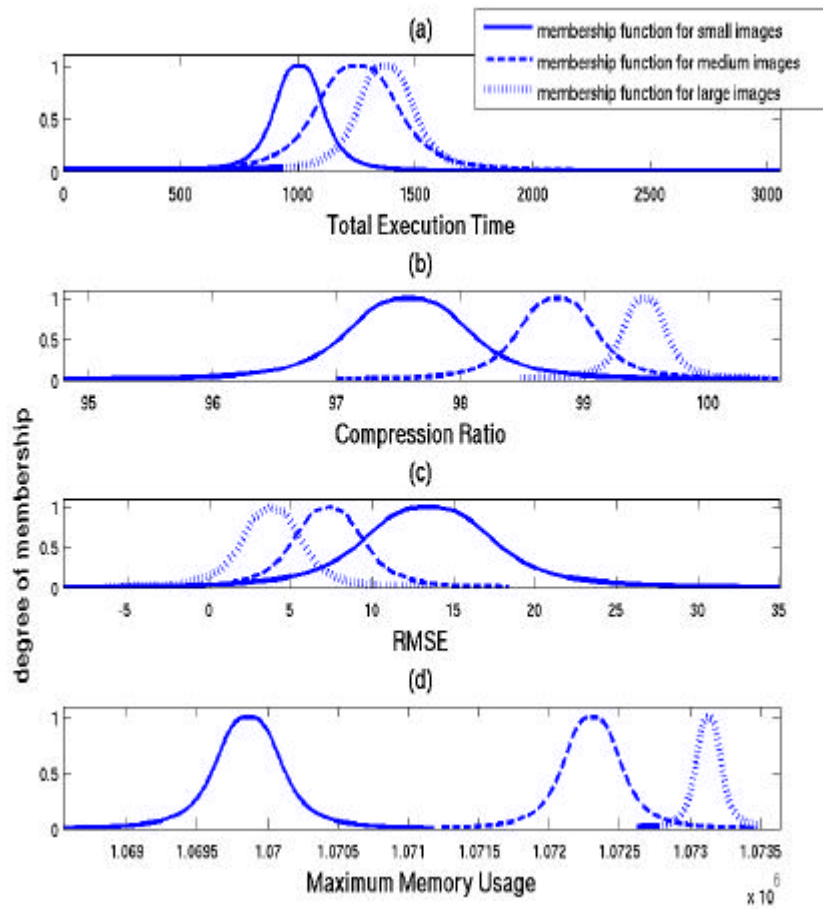

Figure 2: Results of the fuzzification of the Wavelet compression component: four selected non-functional attributes.

The results for the BWT component are presented in Figure 3. As can be observed there is no plot for the RMSE feature since BWT is a lossless approach. Also, the maximum memory usage is the same for all images and therefore constitutes a crisp value with no need for fuzzification. The results for total execution time and compres-

9

sion ratio are similar and show a better execution time (compression ratio) for small images with decreasing performance as the size of the images increase.

The results for the other three components (Huffman, Arithmetic Encoding, and Fractal) are quite similar and are not presented here.
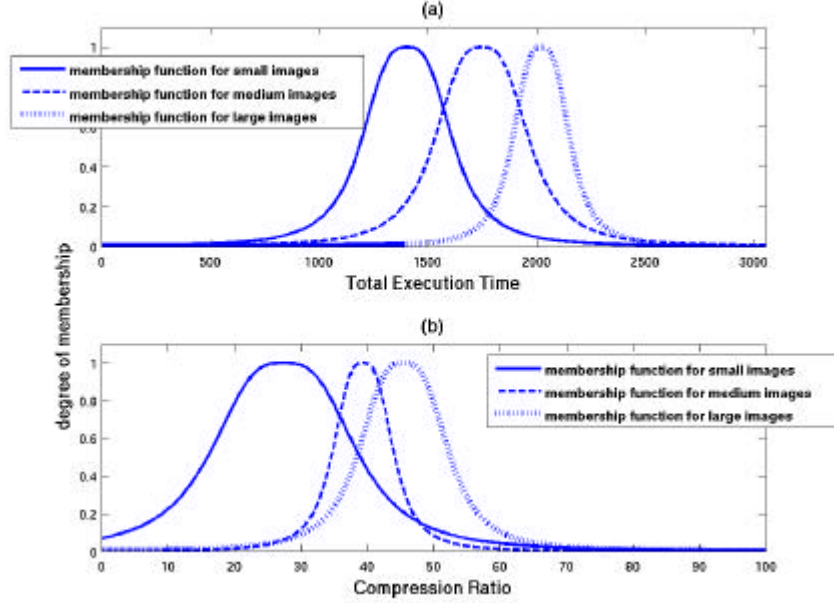

Figure 3: Results of the fuzzification of the BWT compression component: four selected non-functional attributes.

Step 5 - **Input dependent fuzzification of non-functional attributes:** The test inputs T are divided into K subsets $TS=\{Ts_1, Ts_2, ..., Ts_k\}$, where K is the number of membership functions used in the input fuzzification (Step 2). Overlap is expected among the subsets, and the amount of overlap can be determined by a cutoff value for the degree of membership. That is, an input t_i is added to a subset Ts_j only if the membership degree is greater than the cutoff. In general, the larger the value of the cutoff, the smaller the overlap. Each component c_l is executed for the inputs in the subset Ts_l. Measurements of all the non-functional attributes in NF are collected, and one membership function is created based on the values of the measurements. For example, mean and standard deviation values may be used to create generalized bell membership functions. The process is repeated for each subset TS_i in TS and for each component c_j in C.

Similar to image size, the non-functional attributes are also fuzzy by nature. That is, there is no crisp definition of what is a fast compression algorithm or an approach with moderate use of memory. Therefore, as it has been done for the input, neural-adaptive techniques are used to generate three membership functions for the four specified non-functional attributes, as can be seen in Figure 4.
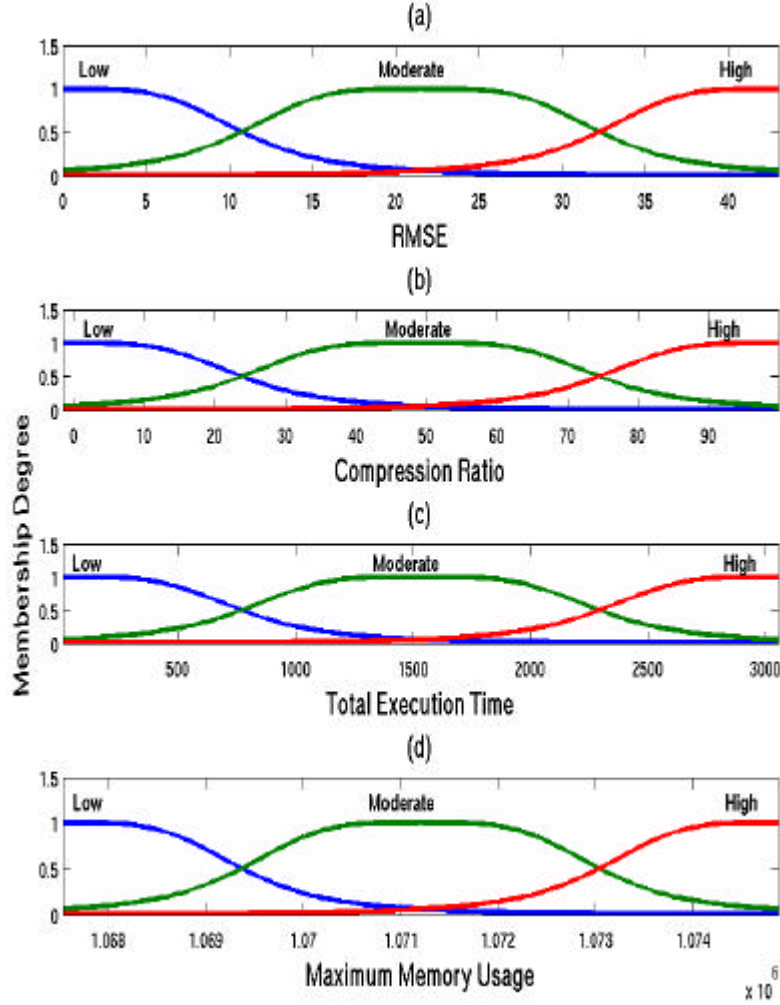
Figure 4: Component independent fuzzification of the four non-functional attributes selected.

Step 6 - **Components ranking:** Once the components are specified in fuzzy logic (Steps 1-5), we can search for components using linguistic rather than numeric variables/values. Any search with respect to the input attributes A and the non-functional attributes *NF* can be conducted. Since the focus of this paper is on the fuzzy specification, the ranking is restricted to individual non-functional attributes; multi-criteria techniques are the subject of future work. In order to rank the components, specified membership functions for a non-functional attribute are combined (using an OR fuzzy operator) and then merged (using an AND fuzzy operator) with the membership func-

tion associated with the specified input attribute. The resulting new membership functions (one for each component) represent the query results. Sorting the maximum or the mean value for each function produces the desired rank of components. The use of maximum or mean values may affect the ranking, but a discussion of which one is more appropriate is beyond the scope of this paper.

Up to this point we have fuzzy specifications for the input (Figure 1) and for each of the components and the specified non-functional attributes (Figure 2 and Figure 3), as well as a component independent specification for the same attributes (Figure 4). These specifications are now used to conduct queries and select/rank components according to some features. For example, consider a search is done according to the following criteria: a fast compression component. The size of the images is not specified in this case, and therefore the components should be considered for images of all sizes (small, medium, and large). A fuzzy OR operator is used to merge the three membership functions for Total Execution Time for each of the components. The results for the OR operator for the wavelet component are shown in Figure 5 (light dashed line - referred to hereafter as mf_1). A fast component is represented by the low membership function of Figure 4(c). The same curve is also plotted in Figure 5 (heavy dashed line - referred to hereafter as mf_2). The comparison of the wavelet results and the component independent specification is achieved by applying an AND fuzzy operator for mf_1 and mf_2 resulting in the new membership function represented by the solid line in Figure 5.
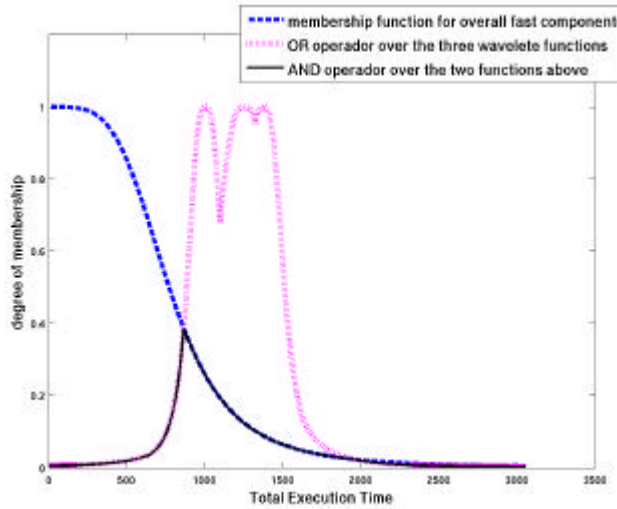

Figure 5: Membership function of the Wavelet component: searching for a fast component.

The same process is conducted for each of the components. The ranking of the components can be achieved by sorting the resulting membership functions by the largest value or by the mean value. The ranking results when searching for a fast component are shown in Table 1.

Table 1: Results of ranking the components according to two queries.

Query: fast component		
Rank	Largest Value	Mean Value
1	HUFF - 0.82	HUFF – 0.11
2	AREC – 0.56	AREC – 0.08
3	WAVE – 0.38	WAVE – 0.05
4	BWT – 0.21	BWT – 0.03
5	FRAC – 0.18	FRAC – 0.01

Query: high compression ratio for large images		
Rank	Largest Value	Mean Value
1	WAVE – 0.99	BWT – 0.026
2	BWT – 0.12	WAVE – 0.023
3	AREC – 0.05	AREC – 0.014
4	HUFF – 0.04	HUFF – 0.012

Now, consider we are searching for components with high compression ratio for large images. The process above is repeated, but there is no need to apply the OR operator to merge the membership functions for each of the components. In this case mf_1 is represented only by the Compression Ratio membership function for large images. The result for this new query is also shown in Table 1. Fractal is not displayed in the table due to lack of sufficient large images to collect meaningful results.

Additional test queries have been executed for memory usage and RMSE to rank the components. The results from these tests indicate the applicability of our approach to specify and rank components using fuzzy logic; they have not been included in the paper due to space constraints.

4 Related Work

A substantial body of work exists on the specification of components. Here, due to space constraints, we restrict the discussion of related material to the work done on IP-Based Design [18]. This technique is closer to our approach as it uses fuzzy logic in the selection of components.

Zhang, Benini, and Micheli [18] have identified the fuzzy characteristics of component selection and have proposed a technique based on IP (Intellectual Property) Design. Assuming an IP repository is available where specifications are represented using XML, the XML document is parsed into a Document Object Model tree (DOM tree). This tree is based on a grammar named Document Type Definition (DTD) and therefore represents the syntax structure of the specification. Further, semantic information is added to the tree.

Once the semantic trees for the components are available, the proximity of the trees to the query specification (also done in XML) is ordered using fuzzy logic. First, leaf nodes are scored using any specified fuzzy membership function. Then, the nodes are aggregated using an un-weighted or a weighted function. The second case is applicable when the user wants to emphasize some features of the desired components. As pointed out by the authors, the use of a specific fuzzy membership function does not need to be based on strong mathematical arguments, but mainly on the characteristics of the domain it represents. The ordered results from the query provide the user a

ranked list of components with a higher probability of fulfilling the design requirements.

An interesting approach is used for the fuzzy representation of non-numeric values. For example, to represent functionality as a fuzzy object, the number of occurrences and the probability of one occurrence of some keyword in the specification is used: $(1-p^n)$, where p is the probability and n is the number of occurrences.

The IP-Based approach and the approach proposed here are based on the same assumptions that component specification and behavior cannot be completely determined by a crisp function and fuzzy logic appears to properly address this problem. Both approaches have as the ultimate goal the selection of components that better match some specification. However, while the IP-Based approach relies on the existence of a XML specification, our approach includes steps to systematically select components, identify the inputs that affect the functional and non-functional behavior of the components, create input to exercise the components, collect and fuzzify data, and rank components [18].

5 Conclusions and Future Work

The rigorous specification of software components is necessary so that we can reason about them in useful ways, such as selecting individual components or collections of interacting components. Part of the complexity of this problem stems from the challenges in representing the uncertainty of the non-functional attributes in a meaningful way. Here, we present a systematic approach for the specification of components using fuzzy logic. A significant advantage of fuzzy logic is that it supports linguistic variables, or hedges (e.g., terms such as slow, fast, very fast, etc.), which are very convenient for describing non-functional attributes. Our process involves conducting an empirical study to gather data on five components that provide data compression capabilities; each uses a different algorithm (Arithmetic Encoding, Huffman, Wavelet, Fractal, and BWT). Data on non-functional attributes, response time performance, memory use, compression ratio, and root mean square error are collected by executing the components on a collection of images with different file formats and sizes. The data are fuzzified and represented as membership functions. The fuzzy component specifications are successfully ranked with respect to a set of test queries.

Limitations of the study include the fact that the components only provide one functional capability (data compression), and one execution environment has been used to collect data. In addition, the ability to select components based on multiple attributes is not addressed in this work.

There are a number of interesting directions for our future work. First, the problem of selecting components based on multiple attributes is going to be investigated using fuzzy multi-criteria decision making algorithms (FMCDM). In multi-criteria decision problems, relevant alternatives are evaluated according to a set of criteria, and the best alternative can be picked depending on the evaluation results. A study comparing a collection of FMCDM algorithms, including the Fuzzy Weighted Sum Method, Fuzzy

Weighted Product Method, and Fuzzy Analytic Hierarchy Process, is going to be conducted. Our proposed study will replicate the work presented by Triantaphyllou [20] and extend it with additional FMCDM algorithms. In addition, a study can be conducted to compare alternatives to the fuzzy logic evaluation of the components. This study would clarify the advantages and disadvantages of our approach. Second, the component specification can be refined with the addition of attributes, both functional and non-functional; the repository of components will be extended to include components with different functional capabilities. Component data will be collected on a more comprehensive collection of platforms (i.e., hardware/operating system); the behavior of the platforms can also be represented as fuzzy membership functions. Third, the process definition will also be extended to include entry and exit conditions, while refining the descriptions of the inputs, outputs, and purpose of each activity.

References

1. Crnkovic, I., Component-Based Software Engineering — New Challenges in Software Development: Journal of Computing & Information Technology; Sep2003, Vol. 11 Issue 3, p. 151-162.
2. Haddad, H.M. and Biberoglu, E., Component-based software engineering: issues and concerns, In Proceedings, International Conference on Software Engineering Research and Practice SERP'03, Las Vegas, USA, 2003, p. 391-397.
3. Heineman, G. T., Councill, W.T., *Component-Based Software Engineering – Putting the Pieces Together*, Addison Wesley, 2001.
4. Szyperski, C., *Component Software*, Addison-Wesley, 2 edition, 2002.
5. Voas, J., "The Challenges of Using COTS Software in Component-Based Development", *IEEE Computer*, June 1998, Vol. 31 Issue 6, pp. 44-45.
6. Carney, D. and Long, F. What Do You Mean by COTS? Finally a Useful Answer, *IEEE Software*, 17(2), 2000, 83-86.
7. Seacord, R., Mundie, D., & Boonsiri, S. "K-BACEE: Knowledge-Based Automated Component Ensemble Evaluation," in Proceedings, Workshop on Component-Based Software Engineering, Warsaw, Poland, 2001, p. 56-62.
8. Zhang, T., Benini, L., and de Micheli, G., "Component Selection and Matching for IP-Based Design", in Proceedings, Conference on Design, automation and test in Europe, Munich, Germany, 2001, p. 40 - 46.
9. Lau K. and Ornaghi M., "A Formal Approach to Software Component Specification", in Proceedings, Specification and Verification of Component-Based Systems Workshop, OOPSLA 2001, Tampa Bay, Florida, p. 88-96.
10. Addy, E. and Sitaraman, M., "Formal specification of COTS-based software: a case study", in Proceedings, *5th Symposium on Software Reusability*, 1999, pp. 83-91.
11. Poizat, P., Royer, J-C., and Salaün, G., "Formal Methods for Component Description, Coordination and Adaptation", in Proceedings, 1st International Workshop on Coordination and Adaptation Techniques for Software Entities, June 14, 2004, Oslo, Norway.

12. Cheesman, J. and Daniels, J., *UML Components A Simple Process for Specifying Component-Based Software*, Addison Wesley, 2000.
13. Han, J., "A Comprehensive interface definition framework for software components", in Proceedings, 1998 Asia-Pacific Software Engineering Conference, Taipei, Taiwan, pp. 110-117.
14. Kallio, P. and Niemela, E., "Documented Quality of COTS and OCM Components", in Proceedings, 4th ICSE Workshop on Component-Based Software Engineering, May 14-15, 2001, Toronto, Canada, available at http://www.sei.cmu.edu/pacc/CBSE4-Proceedings.html.
15. Carlsson, C. and Fuller, R., *Fuzzy Reasoning in Decision Making and Optimization*. Physica-Verlag, 2001.
16. Klir, G. and Yuan, B., *Fuzzy sets and fuzzy logic: theory and applications*, Prentice Hall 1995.
17. J. Halpern, *Reasoning about Uncertainty*, The MIT Press, 2003.
18. Zadeh, L.A., Fuzzy sets, in Information and Control, Vol. 8, 1965, pp. 338-353.
19. Jang, J.-S. R., ANFIS: Adaptive-Network-based Fuzzy Inference Systems, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 3, pp. 665-685, May 1993.
20. E. Triantaphyllou, http://www.directtextbook.com/title/prices/0792366077*Multi-Criteria Decision Making Methods: A comparative Study*, Kluwer Academic Publishers, 2000.