The Verification of Fuzzy Clustering Algorithm according to Requirements for Component Selection Process

Jagdeep Kaur
CSE & IT Dept., The NorthCap University, Gurgaon, Haryana, India.

Pradeep Tomar
Department of Computer Science & Engineering, School of ICT, Gautam Buddha University, Greater Noida, Uttar-Pradesh, India.

Abstract
The component-based software engineering (CBSE) consists of component selection, qualification, adaptation, assembly and updation of components according to requirements. The component selection is an important part of CBSE. The components are selected mainly based on its functionality from the repository but the non-functional properties and the information provided by the publishers about the components also play a crucial role in component selection. The focus of this paper is to verify the FREFCOSCO algorithm for software component selection process. The FREFCOSCO is Fuzzy relation-based fuzzy clustering of software components in which fuzzy clusters are formed based on fuzzy relations. This algorithm is verified with the four tier component selection process. Now-a-days many selection processes, techniques and algorithms are proposed for this task. It also highlights the pre-processing required for the features of the components. The main advantage of this combined approach is reduced computational time of selection and formation of smaller clusters. Hence the component selection process becomes easier. It is validated on a small case study taken from an online repository.

Keywords: Component-Based Software Engineering, Component Selection Algorithm, Component Selection Process, Fuzzy Clustering, COTS

INTRODUCTION
According to Clements [1], Component-Based Software Engineering (CBSE) emphasizes the “the buy, don’t build’ philosophy”. CBSE approach is used to reuse the already developed and testable software code to develop economical software, which can be developed within shorter time and reduce time-to-launch, to increase the quality of the Component-Based Software (CBS). So, Component Based Software Development (CBSD) is the foremost approach of CBSE which advocates acquisition, adaptation and integration of reusable and testable software components to swiftly develop and deploy complex CBS with least engineering techniques, efforts and cost. According to Gill [2] by using CBSD, development of software by writing code is replaced by selecting, assembling and integrating software components. As stated by Crnkovic [3] the development of CBS from reusable components requires development process models and methodologies not only in relation to the development/maintenance aspects, but also to the entire component and various aspects of CBS lifecycle. The use of superior development model and methodology reduce the time and cost by enhancing the productivity and quality of CBS. In CBSE with better development models and methodologies, researchers and practitioners feels to use a better selection process of components which can be selected from in-house developed component repository or could be purchased from Commercial-Off-The-Shelf (COTS) vendors. So, component selection is one of the most crucial steps in CBSE and success of the final system depends on the component selection process.

The software selection process consists of searching the components that provides desired functionality, from the finite set of component set. The selection process helps in selection of the optimal set of components from the third party repository, on the basis of given set of requirement or constraints. The components are selected mainly based on its functionality from the repository but the non-functional properties and the information provided by the publishers about the components also play a crucial role in component selection. As the number of available components in the repository grows, the selection of set of components offering required functionality and managing the conflicting criteria among the components has become extremely difficult. In order to select the optimal set of component having multiple attributes, some of the criteria may be of conflicting nature. An appropriate classification technique is required. Some observations are made from the existing software component selection processes based on clustering.

- The search space can be reduced by usage of clustering technique, which results in reducing the number of components for retrieval.
- The techniques using clustering need to specify prior the number of cluster centres. Even the clusters need to be validated.

The role of non-functional requirements while selection of components is very less and wherever it is considered the final decision is based on subjective judgements of the administrators which can be imprecise sometimes. The fuzzy logic can be incorporated in the selection process to eliminate the imprecision caused by human judgements. There are many component selection techniques and algorithm that exists and
that was the motivation for designing an architecture which can accommodate all the techniques based on clustering. There is a four tier architecture proposed by the authors earlier. The four tiers are: client requirements and system analysis tier, query and decision tier, application logic tier with clustering and component cluster tier[4]. The next section discusses about the clustering techniques in software components.

**SURVEY OF CLUSTERING IN COMPONENT SELECTION**

According to Nakkrasae S et al. [5] the components in software component repository are classified into similar component cluster group with the help of Fuzzy Subtractive Clustering algorithm. The classification index is proposed based on the center of each cluster and it is compared with the required software component. The component giving the closest match is further used for component selection process. Stylianou and Andreou[6], proposed a hybrid software component clustering and retrieval scheme using an entropy-based fuzzy k-modes algorithm. It employs an entropy-based fuzzy k-modes clustering Algorithm, which has two main purposes. The first task is to calculate the number of clusters from the given dataset. The second task is to partition the software components into clusters with degrees of membership and find the final cluster centers. The techniques stops when it finds the cluster centers closer to user’s search preferences.

According to Serban[7] component selection process is solved using metrics and some fuzzy clustering algorithm. Initially components are selected based on some quality attributes of the components. Some metrics are defined to quantify the features of components considered for selection. The metrics used are cost, provided service utilization, required services utilization and functionality. The components are considered as objects to be clustered and by using the fuzzy clustering algorithm appropriate clusters are formed.

According to the Seth et al. [8], a fuzzy rule based model can be used for estimating the efforts in selecting components for application development. The factors considered for selection includes reusability, portability, functionality, security and performance.

Serban et al. [9], presents a formal approach for component based assessment. The main components of this model are: the assessment domain, objectives, formal definitions of metrics and measurement result analysis method. Here system entities are identified, and then properties of these entities are identified using formal specifications. Lastly a problem is identified while interpreting the assessment results so fuzzy clustering analysis is proposed to place a component in more than one cluster and hence reducing the rigidity of threshold values of metrics. This model is scalable and general as other properties and interactions can be added easily. Here, in this section the application of computational intelligent technique in software component selection algorithms is presented. The main advantage of using such techniques is that the rigidity introduced due to use of some pre-defined metrics can be eliminated by using computational intelligent techniques. Moreover, the dependency on human judgment, which may vary, according to the expert’s experience, can also be eliminated.

**THE FOUR TIER ARCHITECTURE FOR COMPONENT SELECTION PROCESS**

A component is selected based on the functionality it provides in CBSE. Many techniques have been proposed for component selection under varied situations to select the optimal component from component set of same functionality. This chapter proposes architecture for component selection process. Here, architecture is divided into four tiers: first is client requirements and system analysis tier, second is query and decision tier, third is application logic tier with clustering and fourth is component cluster tier. The current selection techniques using clustering suffers from major demerits of specifying the number of clusters beforehand and the selection process depending on subjective judgement of application administrators. The proposed architecture will be better as the need for apriori declaration of clusters will be eliminated and the cluster validation is performed to check the correctness of the clusters. This architecture for software component selection is validated on a case study of set of sorting and searching components. As stated by Stylianou and Andreou[6], for a component selection problem the application developer selects the component from the available component set on the basis of different factors like the cost, functionality, development time, dependencies among the components and priorities associated with the components. From the set of all components, the application developer selects those that balance these conflicting attributes of the component. The developer may also want to rank (or prioritize) the components in some way based upon these trade-offs. For systems with more than a few simple components the search space is unmanageably large and complex, with the consequence that no designer can be expected to find optimal choices that balance the constraints without some form of automated support. Hence, informally the component selection problem is to select a set of components from available component repository which can satisfy the requirements while minimizing the sum of costs of selected components.

The component selection process is comparable to the stock selection process for investment. In stock selection the objective is to maximize the total return on investment and minimizing the risk while maintaining an appropriate degree of portfolio diversification. Similarly, the component selection process deals with minimizing the cost and selection is done on the basis of required functionalities and other associated non-functional requirement. The second tier strategic decisions including goals and priorities of the investor. The third tier deals with the application logic that deals with the query generated by the user for selection of appropriate database according to the user requirements. The application logic selects the database from the different data sources. This architecture will be used for developing a CBS to support client requirements for component selection. This architecture will facilitate desired client input and will suggest optimal choice of component for CBSD.
Finally, it helps in forwarding the result as solution to the requested web browser for analysis. The database tier will contain all the data from all third party organisation related to user requirements. The database tier uses data source from different web sites related to component repositories. Each tier is treated as a different subsystem. The different tiers of the architecture and how they work and interact with each other in order to extract the most optimal component are explained in the following sub-section.

COMPONENT SELECTION USING FREFCOSCO ALGORITHM
The main idea of FREFCOSCO (Fuzzy RElation based Fuzzy Clustering Of Software COmponents), is to make a fuzzy similarity relation matrix based on number of features. The main steps of the algorithm are:
Step 1. The input set of features are preprocessed for further processing.
Step 2. To calculate the similarity among the clusters use the fuzzy transitive closure Computation.
Step 3. To generate fuzzy partition set of the matrix.
Step 4. To generate the fuzzy clusters.
Step 5. Terminate the clustering process.
Step 6. Return the number of Clusters.
Step 7. Apply Silhouette Coefficient on the components.

The algorithm takes input in the form of matrix of components; the features can be mix of numerical as well as categorical types. The first step consists of normalizing the data, in terms of converting the categorical data to numerical one and assigning appropriate labels to the features. Further, the fuzzy relation matrix is formed by comparing the attributes of two components. The second step involves finding the fuzzy transitive closure so that it can be used as a similarity measure. The third step is to generate the partitions from the matrix obtained in step 2. The entries in the partition matrix can be obtained by using set similar to property. The fourth step, generates the final fuzzy clusters as the components are repeated in many clusters. The main contributions of this algorithm are as follows:
- The use of clustering has condensed the search space and multi-features can be processed instead of focusing on one or two features, as compared to built-in functions of Fuzzy c-means and subtractive clustering.
- In the existing algorithms the similarity matrices are difficult to be applied on categorical data. But using the proposed algorithm it can be achieved easily.
- There is no need for minimization or maximization of objective functions.
- There is a dependency of clustering results on the distance metric for similarity or dissimilarity. It is removed using the proposed algorithm.
- The simple fuzzy clustering technique requires the need of mentioning the number of cluster centres beforehand, in case of Fuzzy c-means and the radii of the cluster in case of subtractive clustering. The new algorithm eliminates this need.

VERIFICATION OF THE ALGORITHM FOR THE COMPONENT SELECTION PROCESS
The FREFCOSCO (Fuzzy relation-based fuzzy clustering of Software Components) Algorithm is applied using the four tier architecture. Here, in this section it is discussed how this interaction actually takes place.

First tier
The first tier is known as client requirements and the system analyst tier. The component requirements are stated in terms of functional and non-functional requirements. The application developer will state the requirements and the different desirable non-functional requirements of interest like download rating, review-based rating, bestseller rating. The application developer will state the component requirements in terms of features of the available components in the repository. This sub-part of the first tier, provides possible number of solutions according to the component requirements that are feasible for construction of an optimal selection of components. In, return the optimal component sets are given, as solution, to the application developer. The results are provided after the complete application of the algorithm and/or the validation index on the components. With the application of validation index it will help the system analyst in decision making to get the optimal component to be used in component-based system.

Second tier
Second tier is known as query and decision tier that deals with the query manager and decision manager which help in generating query and solution. The query manager will interpret queries from the first tier and extract its semantics. The query manager deals with strategic decisions including goals and priorities of the client. For example, the goals are related to price, relevance, download rating, best seller rating etc. The goals are converted into the form of queries by the query manager. The query may consist of required functionality, goals and constraints. The query manager performs query selection where the queries are formulated based on functional requirements of the components and non-functional requirements like cost, performance, portability, maintainability etc.
The requirements can be interpreted by the query manager in the following ways:

- Functional Requirements: The user can state the requirements in terms of functionality required.
- Non-Functional Requirements: The non-functional requirements for the black box components can be: best seller rating, download rating, review rating etc. whereas for the white box components, the non-functional requirements can be: program volume, time complexity, cyclomatic complexity, input size etc.

The decision manager helps in final decision making either by finding components that directly matches the formulated query or finding a near optimal component set by analyzing co-occurrence, correlation and hidden criteria across different components. The final optimized subset of component is presented to the clients and system analyst for their consideration. In the future, this optimized set of component can be compared with the degree of confidence stated by the clients and system analyst to discard the components that fall below a threshold level.

Third tier
The third tier mainly deals with applying an appropriate clustering method. Many clustering based similarity functions can be applied in this tier. Here, it consists of application of FREFCOSCO algorithm and retrieving the set of components. The Application Logic Tier can be modified according to the given component set. In the Retrieved Component, components are retrieved on the basis of the application of silhouette coefficient in the fourth tier.

Fourth tier
The fourth tier is divided into two parts; the first part is the component cluster tier which represents the cluster of components formed after the application logic. The second part represents the components after the cluster validation index application. Component Cluster Tier deals with the application logic to select the database from the different data sources. The resultant of the clustering algorithms is presented here. In the given case study, the clusters after the application of FREFCOSCO are presented. Here, other similarity factors like Sorensen dice, Jaccard measure, hybrid Sorensen dice etc. Any other fuzzy clustering technique, genetic algorithm or computation intelligence based technique viz. fuzzy logic related, neural networks, fuzzy sets, neuro-fuzzy systems can be applied. Further on these components, the validation indices are applied. The third tier will generate three clusters and on these clusters silhouette coefficient is applied. In this tier, firstly the cluster set that needs to be split further is chosen. These updated component clusters are supplied back to the third tier and finally the decision manager in the second tier will provide components to the application developer according to their choice.

To understand the complete flow of information through this architecture, the detailed flow in given in the Figure 1. Here, the boxes represent the different processes and the arrows represent the sequential order in which the processes will occur. The process starts with the application developer stating the requirements in the interface and end when he/she gets the optimal set of components as the final solution. After stating the requirements, the query manager interprets the query. The application logic is applied on the given components taken from the repository, to generate the clusters. Once the clusters are retrieved, with the help of some techniques validation of the clusters are done. The final retrieved components are analyzed by the decision manager and presented to the application developer.

According to the four tier architecture, these requirements are stated in the first tier. Keeping this functionality in mind, more than two hundred components are retrieved from the online repository. The query manager in the second tier will extract the non-functional requirements like price, bestseller rating, download rating, user rating, relevance, technology used, platform, publisher, compatibility, licensing, support, specifications, availability of release notes and support. The features considered for selection are: Price, distributed on, publisher, disk space and compatibility. This input data is shown in Table 1.

In the third tier, the FREFCOSCO algorithm is applied. The resulting fuzzy partitions are generated in the fourth tier. The larger partitions need to be broken into smaller parts and for that purpose a threshold of similarity is applied, considering a higher degree of similarity, the value of 0.75 is taken. After the application of the threshold the fuzzy clusters are generated and given to the third In the third tier, the FREFCOSCO algorithm is applied. The resulting fuzzy partitions are generated in the fourth tier. The larger partitions need to be broken into smaller parts and for that purpose a threshold of similarity is applied, considering a higher degree of similarity, the value of 0.75 is taken. After the application of the threshold the fuzzy clusters are generated and given to the third tier. The decision manger in the third tier will apply the silhouette coefficient in order to present the final optimized subset of components to the application developer in the first tier.

AN ILLUSTRATIVE CASE STUDY
Consider a case study where the application developer is looking for components for creating digital dashboards and data presentation applications.

Table 1: Component Set for Charting features

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Price (2008)</th>
<th>Distributed on</th>
<th>Publisher</th>
<th>Disk Space</th>
<th>Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proessential</td>
<td>221,000</td>
<td>Gigasoft</td>
<td>15</td>
<td>Windows</td>
<td></td>
</tr>
<tr>
<td>StudioFX premium subscription</td>
<td>89,900</td>
<td>2009</td>
<td>Software FX</td>
<td>100</td>
<td>Windows,iOS,MacOS</td>
</tr>
<tr>
<td>Actipro Micro charts silverlight</td>
<td>2,000</td>
<td>2007</td>
<td>Actipro</td>
<td>30</td>
<td>Windows</td>
</tr>
<tr>
<td>Nevron vision for share point</td>
<td>116,000</td>
<td>2008</td>
<td>Nevron</td>
<td>50</td>
<td>Windows</td>
</tr>
<tr>
<td>Nevron chart for .net enterprise</td>
<td>129,900</td>
<td>2008</td>
<td>Nevron</td>
<td>50</td>
<td>Windows</td>
</tr>
</tbody>
</table>
The non-functional requirements like price, bestseller rating, download rating, user rating, relevance, technology used, platform, publisher, compatibility, licensing, support, specifications, availability of release notes and support. The features considered for selection are: Price, distributed on, publisher, disk space and compatibility. This input data is shown in table 1.

These components are taken from an online repository. The initial match to the functionality has shown more than two hundred components, but for simplicity we have taken only ten components. However, this algorithm can be scaled to work on more than 200 components.

The component related data is processed. The price range is categorized as high (more than one lakh), medium (Fifty thousand to one lakh) and low (less than fifty thousand). The distribution year is categorized in category 1 (2002, 2004), category 2 (2005, 2007) and category 3 (2008, 2009). The disk space low (10 MB, 15 MB, 20 MB), medium (30MB, 32MB) and High (50MB and 100 MB). The compatibility is decided as one or zero depending upon the compatibility is with windows, linux or iOS. This dataset is represented in a preprocessed way as shown in table 2. This is further represented in a matrix form, for processing by the FREFCOSCO algorithm.

The algorithm is implemented in R2014b, the final clusters and components are shown in the form of a graph as in figure 1.

The complete input dataset can be represented in the form of a matrix as shown below.

The normalized matrix is formed on the basis of normalization procedure as explained follows. The input for it is the dataset of components along with their features. It is taken in the form of matrices of values. For instance, if four features are taken into consideration for set of ten components, then the input is a 10x 10 matrix. In order to apply normalization, the rule used is as follows:

If i and j sets are having n attributes and m is the number of common attributes, then \( \mu(i, j) \) can be defined as m/n. That means each entry of the normalized matrix is calculated by taking the ratio of common features to the total number of features taken into consideration. For example, for the above mentioned input, if these are two features common between component 2 and component 3, then the entry corresponding to \( \mu(2, 3) \) is calculated as 2/4 i.e. 0.5. Similarly, other values are calculated. Formally, this procedure can be summarized as:

Procedure: Normalize (\( \Delta \))

Input: A dataset \( \Delta \)

Output: A normalized matrix with I columns and J rows where each entry Aij is represented as If i and j sets are having

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Price</th>
<th>Distributed on</th>
<th>Disk Space (MB)</th>
<th>Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>High</td>
<td>3</td>
<td>Low</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>Medium</td>
<td>3</td>
<td>High</td>
<td>2</td>
</tr>
<tr>
<td>C3</td>
<td>Medium</td>
<td>2</td>
<td>Medium</td>
<td>1</td>
</tr>
<tr>
<td>C4</td>
<td>High</td>
<td>3</td>
<td>High</td>
<td>1</td>
</tr>
<tr>
<td>C5</td>
<td>High</td>
<td>3</td>
<td>High</td>
<td>1</td>
</tr>
<tr>
<td>C6</td>
<td>Low</td>
<td>1</td>
<td>Low</td>
<td>2</td>
</tr>
<tr>
<td>C7</td>
<td>Medium</td>
<td>1</td>
<td>Low</td>
<td>1</td>
</tr>
<tr>
<td>C8</td>
<td>Low</td>
<td>1</td>
<td>Medium</td>
<td>1</td>
</tr>
<tr>
<td>C9</td>
<td>Low</td>
<td>2</td>
<td>Low</td>
<td>2</td>
</tr>
<tr>
<td>C10</td>
<td>Medium</td>
<td>3</td>
<td>High</td>
<td>1</td>
</tr>
</tbody>
</table>
n attributes and m is the number of common attributes, then 
\( \mu(i, j) \) can be defined as \( m/n \). On this input matrix, further operations of the FREFCOSCO are applied and finally the fuzzy clusters as shown in figure 1, are generated. Further on the individual components of the clusters, the silhouette coefficient is applied. Now, the application developer can select the component which has maximum cohesion within the cluster and minimum coupling between the components of the clusters.

CONCLUSION
To enhance the software reusability component based software engineering (CBSE) is an important paradigm. To achieve it fully a proper architecture is required which can be used to select the components from the repository. The software component repositories also play a vital role in component selection. The more organized, filtering mechanisms and access means are provided in the repository the lesser the efforts and time is required to search for the component. This current paper shows the usage of the four-tier architecture for component selection in line with the FREFCOSCO algorithm. This paper highlights the use of preprocessing techniques for the algorithm. It also shows how the mix of numerical and categorical datatypes can be handled. This work can be extended by considering the situation when the values of all the features under consideration are not available.

REFERENCES