Journal of Object Technology
Published by ETH Zurich, Chair of Software Engineering © Jot, 2010
Online athttp://www.jot.fm

Identification of System Software Components Using Clustering Approach

Gholam Reza Shahmohammadi[a] Saeed Jalili[a]
Seyed Mohammad Hossein Hasheminejad[a]

a. Tarbiat Modares University, Iran

Abstract
The selection of software architecture style is an important decision of design stage, and has a significant impact on various system quality attributes. To determine software architecture based on architectural style selection, the software functionalities have to be distributed among the components of software architecture. In this paper, a method based on the clustering of use cases is proposed to identify software components and their responsibilities. To select a proper clustering method, first the proposed method is performed on a number of software systems using different clustering methods, and the results are verified by expert opinion, and the best method is recommended. By sensitivity analysis, the effect of features on accuracy of clustering is evaluated. Finally, to determine the appropriate number of clusters (i.e. the number of software components), metrics of the interior cohesion of clusters and the coupling among them are used. Advantages of the proposed method include; 1) no need for weighting the features, 2) sensitivity analysis of the effect of features on clustering accuracy, and 3) presentation of a clear method to identify software components and their responsibilities.

Keywords. Automatic components identification, Clustering of use-cases, Software architecture.

Gholam Reza Shahmohammadi, Saeed Jalili, Seyed Mohammad Hossein Hasheminejad. Identification of System Software Components Using Clustering Approach. In Journal of Object Technology, vol. 9, no. 6, 2010, pages 77–98. Available at http://dx.doi.org/10.5381/jot.2010.9.6.a4

1 Introduction

Software architecture is a fundamental artifact in the software life cycle with an essential role in supporting quality attributes of the final software product. Making use of architecture styles is one of the ways to design software systems and guarantee the satisfaction of their quality attributes [1]. After architectural style selection, only type of software architecture organization is specified. Then software components and their responsibilities need to be identified. On the other hand, component-based development (CBD) is nowadays an effective solution for the subject of development and maintenance of information systems [2]. A component is a basic block that can be designed and if necessary be combined with other components [3]. Partitioning software system to components, while effective on later software development stages, has a central role in defining the system architecture. Component identification is one of the most difficult tasks in the software development process [4]. Indeed a few systematic components identification methods have been presented, and there are no automatic tools to help experts for identifying the components, and components identification is usually made based on expert experience.

In [4], a hierarchical clustering technique is proposed based on graph using edge, i.e. class relationships can be weighed according to their types. To identify business object component, the technique uses the concept of resemblance degree between business objects. The resemblance degree depends on the relationship among the objects either dynamics or statics. The static relationships between business objects can be classified as general, compound, aggregation and association, and have different weights. The technique is composed of two steps: 1) The object-activity relation matrix is used to calculate the dynamic resemblance degree, and 2) To identify business object components, the technique uses a hierarchical clustering technique based on business object relation graph according to the edge strength metric. The strength of edge considers both cohesion and coupling between business objects. However, the technique could be impractical to medium/big systems since it relies on the number of possible pairs of business objects.

In [5], software components have been determined using use case model, object model, and dynamic model (i.e. collaboration diagram). For clustering related functions, the functional dependency of use cases is calculated and related use cases are clustered.

In [6], the static and dynamic relationships between the different classes in the UML domain model are used for clustering related classes in components. Static relationship measures the relationship strength using different weights, and dynamic relationship measures the frequency of message exchange at runtime. To compute the overall strength of relationship between classes, the results of static and dynamic relationships between classes are combined.

In [7], a method to decompose requirements to components is presented. In this work hierarchical clustering algorithms are used for partitioning a system. Also they demonstrate different ways of defining and populating the input matrix to feed into the clustering technique.

In [8], for identifying business stable components, a novel framework is presented based on Formal Concept Analysis (FCA). In this work, FCA is used to develop a framework to identify components from use cases and class diagram.

In [9], a development process component is explained based on use cases and business type models that includes a method to identify business components. Inter-class relationship is used as the main factor for identifying components. The core class serves as the center of each clustering, and the responsibility derived from use cases is used to guide the process.

In [10,11,12] the Business Component Identification (BCI) method, as well as the further development to *BCI-3D*, are described by Albani et al.. BCI is the first and a crucial step in the business component modeling process. It starts by generating a set of essential elements from the business domain, rather on the basis of a high-level business domain model. Information about data objects, process steps and actors, plus their relationships is mapped onto vertices and edges of a graph. Weights are assigned to the edges depending on the relation type and the designers preferences. Using a heuristic from graph theory, the graph is partitioned and components are identified. The BCI-3D tool supports the designer in the identification process.

Disadvantages of most of these methods are: 1) Lack of validation of the results of the clustering application on some software systems 2) Lack of an approach for determining the number of system components; 3) No sensitivity analysis of the effect of features on accuracy of clustering; 4) High dependency of the method on expert opinion; 5) Need manual weighting of the features used in clustering and 6) No evaluation of the effect of using different clustering methods.

Since use cases indicate functionality of a system and a set of related use cases shows the coherent functionality of the system, allocating the related use cases to components is a suitable way to determine components of the system. Therefore, in this paper a method for automatic identification of system software components is proposed based on the use case model (in analysis phase). In this method, at first using the system analysis model including use case model, class diagram and collaboration diagram, some features are extracted. Then, using the proposed method and applying various clustering methods, use cases are clustered in several components. To evaluate the clustering methods, the components that result from clustering are compared to the ones specified by the experts and the method with most conformity with the expert opinion is selected. In most methods, the number of clusters (K) is an input parameter of clustering. But for partitioning a system to some components, the number of these components is not specified beforehand. Thus, in the proposed method, clustering is repeated for different values of K, and

then the most appropriate value of K (number of components) is chosen regarding high cohesion of components and low coupling among them. In order to increase clustering accuracy, the effect of features on accuracy of clustering is determined using sensitivity analysis on use case features. Finally, by choosing a proper feature selection method, minimum features achieving the required accuracy in clustering are selected.

Next, we present clustering in the second part. The proposed method for determining components and the evaluation model of the proposed method are presented in sections 3 and 4, respectively. In section 5, the conclusion is presented and the proposed method is compared with other methods.

2 Clustering

In order to understand new objects and phenomena, their features are described, and then compared to other known objects or phenomena, based on similarity or dissimilarity [13]. All of the clustering methods include three common key steps: 1) Determine the object features and data collection, 2) Compute the similarity coefficients of the data set, and 3) Execute the clustering method.

Each input data set consists of an object-attribute matrix in which objects are the entities grouped based on their similarities. Attributes are the properties of the objects. A similarity coefficient for a given pair of objects shows the degree of similarity or dissimilarity between these two objects, depending on the way the data are represented. The similarity coefficient could be qualitative or quantitative. A data object is described by a set of features represented as a vector. The features are quantitative or qualitative, continuous or binary, nominal or ordinal. A feature type determines the corresponding measure mechanism.

2.1 Similarity and Dissimilarity Measures

To join (separate) the most similar (dissimilar) objects of a data set X in some cluster, clustering algorithms apply a function that can make a quantitative measure among vectors. This quantitative measure is arranged in a matrix called proximity matrix. Two types of quantitative measures are Similarity Measures, and Dissimilarity Measures. In other words, for a data set with N input patterns, an $N \times N$ symmetric matrix called proximity matrix can be defined where (i, j)-th element represents the similarity or dissimilarity measure for the i-th and j-th patterns (i,j=1,...,N). So, the relationship between objects is represented in a proximity matrix, in which rows and columns correspond to objects. If the objects are considered as points in a d-dimensional space, each element of the proximity

matrix represents the distance between pairs of points [13].

Similarity Measures. The Similarity Measures are used to find similar pairs of objects in X. s_{ij} is called similarity coefficient. The higher the similarity between objects i and j, the higher the s_{ij} value. For all objects i and j, a similarity measure must satisfy the following conditions:

- $0 \leq s_{ij} \leq 1$
- $s_{ii} = 1$
- $s_{ij} = s_{ji}$

Dissimilarity Measures. Dissimilarity Measures are used to find dissimilar pairs of objects in X. The dissimilarity coefficient, d_{ij}, is small when objects i and j are alike, otherwise, d_{ij} becomes larger. A dissimilarity measure must satisfy the following conditions:

- $0 \leq d_{ij} \leq 1$
- $d_{ii} = 0$
- $d_{ij} = d_{ji}$

Typically, distance functions are used to measure continuous features, while similarity measures are more important for qualitative features [13]. Selection of different measures is problem dependent [13]. For binary features, the similarity measure is commonly used. Let us assume that a number of parameters with two binary indexes are used for counting features in two objects. For example, n_{00} and n_{11} denote the number of simultaneous absence and presence of features in two objects respectively, and n_{01} and n_{10} count the features presented only in one object. The equations (1) and (2) show two types of commonly used similarity measures for data points. w=1 for simple matching coefficient, w=2 for Rogers and Tanimoto measure and w=1/2 for Gower and Legendre measure are used in equation (1). These measures compute the match between two objects directly.

$$S_{ij} = \frac{n_{11} + n_{00}}{n_{11} + n_{00} + w(n_{10} + n_{01})} \tag{1}$$

Equation (2) focuses on the co-occurrence features while ignoring the effect of co-absence. w=1 for Jaccard coefficient, w=2 for Sokal and Sneath measure and w=1/2 for Gower and Legendre measure are used in equation (2).

$$S_{ij} = \frac{n_{11}}{n_{11} + w(n_{10} + n_{01})} \tag{2}$$

2.2 Clustering Methods

In this section, some of the main clustering methods are introduced.

A- Hierarchical Clustering (HC). In this method, hierarchical structure of data is

organized according to a proximity matrix. HC algorithms organize data into a hierarchical structure according to the proximity matrix. The results of HC are usually depicted by a binary tree or dendrogram. The root node of the dendrogram represents the whole data set and each leaf node is regarded as a data object. The intermediate nodes, thus, describe the extent that the objects are proximal to each other; and the height of the dendrogram usually expresses the distance between each pair of objects or clusters, or an object and a cluster. The ultimate clustering results can be obtained by cutting the dendrogram at different levels. HC algorithms are mainly classified as agglomerative methods and divisive methods [13]. Agglomerative clustering starts with N clusters and each of them includes exactly one object. A series of merge operations then follow that finally lead all objects to the same group. Based on the different definitions for distance between two clusters, there are many agglomerative clustering algorithms. Let C_i and C_j be two clusters, and let $|C_i|$ and $|C_j|$ denote the number of objects that each one have. Let $d(C_i,C_j)$ denote the dissimilarity measures between clusters C_i and C_j , and d(i, j) the dissimilarity measure between two objets i, and j where i is an object of C_i and j is an object of C_j. The simplest method is single linkage (SLINK) technique. In the SLINK method, the distance between two clusters is computed by the equation (3). The common problem of classical HC algorithms is lack of robustness and they are, hence, sensitive to noise and outliers. Once an object is assigned to a cluster, it will not be considered again, which means that HC algorithms are not capable of correcting possible previous misclassifications [13].

$$d(c_i,c_j) = \min_{i\in c_i,\, j\in c_j} d(i,j) \tag{3}$$

B- Squared Error—Based Clustering. Partitional clustering assigns a set of objects into clusters with no hierarchical structure. The optimal partition, based on some specific criterion, can be found by enumerating all possibilities. However, this method is impossible in practice, due to expensive computation. Thus, heuristic algorithms have been developed in order to seek approximate solutions. One of the important factors in partitional clustering is the criterion function**.** The sum of squared error functions is one of the most widely used criteria [13]. The main problem of partitional methods is uncertainty of the clustering solution to randomly selected cluster centers. The K-means algorithm belongs to this category. This method is very simple and can be easily implemented in solving many practical problems. But there is no efficient and universal method for identifying the initial partitions and the number of K clusters. The iteratively optimal procedure of K-means cannot guarantee convergence to a global optimum. K-means is sensitive to outliers and noise. Thus, many variants of K-means have appeared in order to overcome these obstacles. K-way clustering algorithms with the repeated bisection (*RB, RBR*) and direct clustering (*DIR*) are expansion of this method that are introduced briefly[14].

RB Clustering Method. In this method, the desired k-way clustering solution is computed by performing a sequence of $k - 1$ repeated bisections. In each step, the cluster is selected for further partitioning is the one whose bisection will optimize the value of the overall clustering criterion function. In this method, the criterion function is locally optimized within each bisection. This process continues until the desired number of clusters is found.

RBR Clustering Method. In this method, the desired k-way clustering solution is computed in a fashion similar to the repeated-bisecting method but at the end, the overall solution is globally optimized.

Direct Clustering Method. In this method, the desired k-way clustering solution is computed by simultaneously finding all k clusters. In general, computing a k-way clustering directly is slower than clustering via repeated bisections.

C- Graph-based Clustering Method. The clustering problems can be described by means of graphs. Nodes of a weighted graph correspond to data points in the pattern space, and edges reflect the proximities between each pair of data points. If the dissimilarity matrix is defined as a threshold value, the graph is simplified to an unweighted threshold graph. Graph theory is used for hierarchical and non-hierarchical clustering [14].

D- Fuzzy Clustering Method. In this method, the object can belong to all of the clusters with a certain degree of membership. This is mainly useful when the boundaries among the clusters are not well separated and ambiguous. Moreover, the memberships may help us discover more sophisticated relations between a given object and the disclosed clusters. FCM is one of the most popular fuzzy clustering algorithms [15]. FCM attempts to find a partition (i.e., c fuzzy clusters) for a set of data points $x_j \in R^d$, j=1,..., N while minimizing the cost function. FCM suffers from the presence of noise and outliers and the difficulty to identify the initial partitions.

E- Neural Networks-Based Clustering. In competitive neural networks, active neurons reinforce their neighborhood within certain regions while suppressing the activities of other neurons. A typical example is self-organizing feature map (SOFM)[13].

2.3 Methods to Determine the Number of Clusters

In most methods, the number of clusters (K) is the input parameter of clustering. But the quality of resulting clusters is largely dependent on the estimation of K. So

many attempts have been made to estimate the appropriate K. For the data points that can be effectively projected onto a two-dimensional Euclidean space, direct observations can provide good insight on the value of K but only to a small scope of applications.

Most proposed methods have presented formulas that emphasize on the compactness within the cluster and separation between clusters, and the comprehensive effect of several factors such as defined squares error, geometric or statistical feature of data and the number of patterns. Two of them are briefly introduced as follows:

A- CH Index[13]. This index is computed by equation (4),where N is the total number of patterns and $Tr(S_B)$ and $Tr(S_W)$ are the trace of the between and within class scatter matrix, respectively. The K that maximizes the value of CH(K) is selected as the optimal.

$$\mathrm{CH}(K) = \frac{\mathrm{Tr}(S_B)}{K-1} \Big/ \frac{\mathrm{Tr}(S_W)}{N-K} \tag{4}$$

B- Ray and Turi index[16]. In this index, the optimal K value is calculated by equation (5). In this equation, *Intra* is the average intra-cluster distance measure that we want to minimize and is computed by equation (6). N is the number of patterns, and z_i is the cluster centre of cluster C_i. *Inter* is distance between cluster centers calculated by equation (7). Meanwhile, we want to maximize the inter-cluster distance, i.e., the minimum distance between any two cluster centers. The K that minimizes the value of *validity* measure is selected as the optimal in k-means clustering.

$$Validity = \frac{Intra}{Inter} \tag{5}$$

$$Intra = \frac{1}{N}\sum_{i=1}^{k}\sum_{x \in C_i} \left\| x - z_i \right\|^2 \tag{6}$$

$$Inter = \min\left(\left\| z_i - z_j \right\|^2\right), i = 1,2,...,K-1 \quad and \quad j = i+1,...,K \tag{7}$$

3 Automatic Determination of System Software Components

In this section, the proposed method for clustering use cases, or in other words, automatic determination of system software components is presented. Software functions clustering is done using artifacts of requirements analysis phase, so all features of the use case model, class diagram and collaboration diagram (if any) are used in clustering.

Each use case indicates a section of system functionality. So, use cases are the main way to express system functionality. Each use case is composed of a number of scenarios in the system, producing a measurable value for a particular actor. A set of use cases describes the complete functionality of the system. Each actor is a coherent set of roles played by the users during interaction with use cases [17]. Each use case diagram shows interaction of the system with external entities and system functionality from user viewpoint. Considering the above statements, software components of the system are identified by relying on identification of coherent use cases of the system. Thus, use cases of the system are stimulators of the proposed method to identify software components of the system.

The stages of the proposed method are: 1) Extraction of use cases features, 2) Construction of proximity matrix of use cases and 3) Clustering system use cases, which are individually introduced.

3.1 Extraction of Use Cases Features

By evaluation of artifacts of requirements analysis phase including use case model, class diagram and collaboration diagram, the following features can be defined for use cases clustering. Features 1 to 4 are binary and other features are continuous.

1– Actors. Use cases initiated or called by the same actor are more related than other use cases because the actors usually play similar roles in the system. So, each actor is considered as a feature, taking a value 1 or 0 based on its presence or absence in the use case.

2 – Entity classes. Use cases working with the same data are more related than other use cases. So, each entity class is considered as a feature taking a value 1 or 0 based on its presence or absence in the use case.

3 – Control classes. In each use case, the class or classes are responsible for coordination of activities between interface classes and entity classes, known as control classes. Use cases controlled by the same control class are more related than other use cases. Each control class is considered as a feature taking a value 1 or 0 based on its presence or absence in the use case.

4 – Relationship between use cases. Based on relationship between use cases, the following features can be extracted:

- If several use cases U_i are related to U_j use case in an extend relationship, a new feature is added to existing use cases features and its value is 1 for U_j and related use cases, and 0 for other use cases.
- If several use cases are specialized from a generalized use case, a new feature is added to existing use cases features and its value is 1 for them and 0 for other use cases.
- If U_i and U_j use cases are related through include relationship, the relationship between U_j and use cases other than U_i should be investigated. U_j may also be

included by U_k (as shown in Figure 1). In this case, if U_i has a relatively strong relationship with U_k (at least 2 or more shared features), a new feature is added to the existing use cases features and its value is 1 for U_i and U_j and 0 for other use cases.

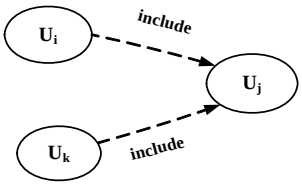

Figure1. Include relationship between use cases

5 – Weight of Control Class(WCC). Considering the number of entity classes and interface classes managed by each control class, a weight is assigned to each control class using equation (8), where Nec_i and Nic_i are, the number of entity and interface classes under control of control class i respectively; and m and l are total number of entity and interface classes of the system, respectively.

$$WCC_i = \frac{Nec_i + Nic_i}{\sum_{j=1}^{m} Nec_j + \sum_{j=1}^{l} Nic_j} \tag{8}$$

6 –Association Weight of Use Case(AWUC). This feature is calculated by equation (9), where Ncc_i is the number of control classes of each use case, $Naec_i$ *is* the number of relationships between entity classes of the use case and Nec_i is the number of entity classes of the use case (each control class has an association with entity classes of the use case). The variable u is the number of use cases of the system and the denominator of the fraction is total dependency of all use cases of the system.

$$AWUC_i = \frac{Ncc_i + Naec_i + Nec_i}{\sum_{j=1}^{u} (Ncc_i + Naec_i + Nec_i)} \tag{9}$$

7 –The similarity rate of each use case with other use cases. This feature is computed in terms of binary features (1 to 4 features) using equation (2) and coefficient of Jaccard. In this equation, n_{11} is the number of binary features with a value of 1 in both use cases, n_{01} is the number of binary features having a value of 0 for the first use case and 1 for the others; and the inverse relation exists for n_{10}. Since similarity of each use case with the other (N-1) use cases is calculated, (N-1) features are added to existing features.

3.2 Constructing Proximity Matrix of Use Cases

As mentioned in section 2, clustering is done based on either features matrix or proximity matrix (similarity/dissimilarity) of objects. As discussed in the previous step, some of the features are continuous and some are binary. In clustering objects with mixed features (both binary and continuous features), we can either map all these features into the interval (0, 1) and use distance measures, or transform them into binary features and use similarity functions. The problem of both methods is the information loss [13]. We can construct similarity matrix for binary features and dissimilarity (distance) matrix for continuous features, then convert dissimilarity matrix to similarity matrix, and use equation (10) to combine them in a single similarity matrix [18]. w_1 and w_2 values are positive weights determined concerning the importance of matrices. Also, s_1 and s_2 are binary and continuous similarity matrices, respectively.

$$s(i,j) = \frac{w_1 s_1(i,j) + w_2 s_2(i,j)}{w_1 + w_2} \tag{10}$$

Thus, proximity matrix is created as follows:

A – Constructing similarity matrix of binary features. The similarity matrix of binary features (features 1 to 4) is formed using equation (2) and coefficient of Jaccard.

B – Constructing distance matrix of continuous features. For continuous features (features 5 to 7), the cosine distance measurement is used in which for each X matrix with dimensions $m \times n$, the distance between every two feature vectors of x_r and x_s is calculated using equation (11).

$$d_{rs} = \left(1 - \frac{x_r . x_s'}{\sqrt{x_r' x_r}\sqrt{x_s' x_s}}\right) \tag{11}$$

C – Converting distance matrix of stage (B) to similarity matrix. Distance matrix of stage (B) is converted to similarity matrix by converting each distance element to similarity element using equation (12) in which d_{ij} is the distance between i and j use cases.

$$S_{ij} = 1\text{-}d_{ij} \tag{12}$$

D – Combining similarity matrices of stages (A) and (C). Using equation (10), the similarity matrices of stages (A) and (C) are combined.

3.3 Clustering System Use Cases

In Section 3-2, use cases similarity matrix was established. This matrix, which is created by the programs written in Matlab tool, is the main input of most clustering methods used in this study. For clustering use cases of the system, the

following clustering methods are used: (1) RBR, (2) RB, (3) Agglomerative (Agglo), (4) Direct, (5) Graph-based, (6) FCM and (7) Competitive Neural Network (CNN). The best clustering method is chosen based on the assessment performed in Section 4.

4 Evaluation of the Proposed Method

In the previous section, the method proposed to determine the system software components was described. Several clustering methods have been used in our proposed method. To select the best clustering method, first the results of partitioning the functionalities of several software systems to their components are compared to expert opinion using the introduced methods. Then, the method with most conformity with the expert opinion will be selected. In addition, using criteria based on high cluster cohesion and low cluster coupling, the suitable number of clusters is determined. In addition , the effect of each feature on accuracy of clustering is determined using sensitivity analysis. Finally, the set of features close to the optimum that contribute to clustering precision while being minimal is determined.

In methods (1) to (5), clustering is done based on the similarity matrix using CLUTO tool and various optimization functions [14, 19, 20]. CLUTO is a software package for clustering low and high dimensional datasets and for analyzing the characteristics of different clusters. In most clustering methods of CLUTO, the clustering problem is treated as an optimization process, which seeks to maximize or minimize a particular clustering criterion function defined either globally or locally over the entire clustering solution space. CLUTO provides seven different criterion functions (h2, h1, g'1, g1, e1, i2, i1) that can be used in both partitional and agglomerative clustering methods. In addition, CLUTO provides some of traditional local criteria such as SLINK that can be used in the agglomerative clustering. Also, CLUTO provides graph-partitioning-based clustering algorithms. In FCM and CNN methods, clustering is done based on features matrix of use cases using MATLAB software [21].

4.1 Evaluation Method

The steps of evaluation method are as follows:

A- **Determining the Best Clustering Method**. In this step, first, the results of applying use case clustering methods on some software systems are compared to the desired expert clustering, and then the method that shows the most conformity with the result of expert clustering will be selected as the best method.

The error of a clustering method is computed by equation (13), where CE_j and CT_j are the set of use cases of *j-th* component from expert and clustering method view, respectively.Δ symbol is the symmetric difference of two sets. The number of clusters(K) in each system is determined based on expert opinion. The computation of error of the clustering methods are done by programs written in Matlab tool.

$$Error = \frac{1}{2}\sum_{j=1}^{K}\left\|CE_j \Delta CT_j\right\| \tag{13}$$

The overall performance(i.e. the mean error) of a clustering method on some software systems is calculated in terms of error by equation (14). In this equation, $NCE_{i,j}$ is the error of clustering method i on *j-th* system, NUC_j is the number of use cases of *j-th* system and NS is the number of systems. Since in equation (14) we consider normalized errors so lower QCF_i shows the higher quality of i-th clustering method.

$$QCF_i = \frac{1}{NS}\sum_{j=1}^{NS}\frac{NCE_{i,j}}{NUC_j} \tag{14}$$

B- **Sensitivity Analysis**. In this stage, by eliminating each feature, its effect in clustering is examined and the features with negative effects or no effect in clustering are identified and removed.

C- **Determining Approximately Minimal Features Set**. To select a feature set that while being approximately minimal its accuracy is sufficient for clustering, the sequence backward selection (*SBS*) method [22] is used. In *SBS* greedy method, we begin with all features and repeatedly eliminate a feature, until the best performance of clustering is reached.

D- **Determining Suitable Number of Clusters.** In section 2-3, two methods where introduced to determine the number of clusters. In this stage, by applying these methods, the number of clusters for sample software systems is determined and the suitable method is selected.

4.2 Sample Software Systems

In this section, the proposed method is validated using four software systems of a software development company in Iran consisting of: (1) Education System(ES), (2) Online Stock Brokerage System(OSBS), (3) Chain-Store System(CSS), and (4) Home Appliance Control System(HACS). The number of different features of each system is shown in columns 4 to 11 of Table 1. The second column shows the number of use cases and third column shows the number of components of each system. Note that for each use case, there is one control class weight feature and one use case association weight feature.

Table 1. Characteristics of the sample software systems

Systems												
System	Number of		Number of			Number of different relationship among use cases			Number of		Similarity Rate of each Use case	Total number of Features
	Use cases	Components	Actors	Entity classes	Control classes	Extend	Specialization, Generalization	Include	Weight of control class	Association weight of use case		
ES	53	6	10	17	24	4	0	0	1	1	52	109
OSBS	23	4	3	6	10	2	0	7	1	1	22	52
CSS	21	4	5	18	11	0	0	0	1	1	20	56
HACS	11	3	4	6	7	0	0	0	1	1	10	29

4.3 Evaluation of Clustering Methods

We considered the systems introduced in Table 1 one by one. For each system, first we extracted its features and then produced its proximity matrix following the four steps introduced in section 3-2. Finally we applied all the methods introduced in section 2-3, except FCM and CNN methods, on the proximity matrix of the systems. Note that in equation (10), we considered w_1=w_2=0.5 and in FCM method we defuzzification process in order to assign each use case to the most related cluster. Table 3 shows the results of clustering methods

The values shown in the column 3 to 6 of Table 2 are clustering errors computed based on equation (13). Note that the number of components in each system is determined by experts.

The results of use cases clustering by RBR, RB, Direct and Graph-based methods reveal that in each of these methods, the average error per criterion functions (QCF) i1, i2, h1, and h2 is the same. Thus, only the results of h2 criterion function are presented in Table 2. Average error of RBR, RB and Direct methods for other criterion functions is higher than 0.141, so they were not shown in Table 2.

According to the results of Table 2, and based on equation (14), RBR and Direct methods with criterion functions i1, i2, h1, and h2 have the most conformity with experts opinion. Thus, these methods with desired criterion functions are recommended.

For more clarification, OSBS is concisely described. This system facilitates its users i.e., individual investors to trade the stocks online through the internet such as buy, sell, etc. With a good OSBS, any person can learn everything to become a smart investor, and buy and sell stocks online. One of the clear software component of this system is the reporting component, that is responsible of preparing the useful reports and it consists of several use cases such as: economic

statistic & trends, trading report, last day indices stock. We apply our proposed method on this system and the number of clusters is determined by expert. we obtain components similar to expert opinion. For example, the use cases of a component are the same as use cases of reporting component.

Table 2. Clustering Results of systems use cases with different clustering methods

Clustering Method	Criterion Function	Clustering Error				Average Error of Clustering Method QCF
		ES (53)	OSBS (23)	CSS (21)	HACS (11)	
RBR	h2	6	2	0	0	0.05
RB	h2	6	3	0	3	0.129
Direct	h2	6	2	0	2	0.095
Graph-based	h2	6	7	7	0	0.188
Agglo	i2	6	3	4	0	0.109
FCM	-	7	1	4	4	0.182
CNN	-	14	6	5	4	0.282

4.4 Determining the Appropriate Number of Clusters

As stated in section 2-3, the basis of determining the number of clusters is the intra-cluster compactness and inter-cluster coupling. To automatically determine the number of clusters, CH and Ray indices are used. Table 3 shows the number of components of the sample software systems based on expert opinion and these indices. According to Table 3, the results of CH index is far from the expert opinion so it is not a suitable method. The results of ray index are close to expert opinions, so we accept the results of this index.

Table 3. The number of components in sample software systems

System	Number of Use cases	Opinion expert	Number of Components			
			CH index		Ray index	
				Difference		Difference
ES	53	6	5	-1	5	-1
OSBS	23	4	10	+6	5	+1
CSS	21	4	2	-2	4	0
HACS	11	3	6	+3	3	0

4.5 Sensitivity Analysis

For sensitivity analysis, by eliminating each feature, its effect on accuracy of clustering is evaluated and features with negative effect or features without effect

upon clustering are identified and deleted. Table 4 shows features and their effects in clustering. Absence of feature is shown by "-"symbol.
Results of sensitivity analysis show that: (1) The effect of features presented in rows 1 to 3 and 7 of Table 4 in use case clustering is significant, and (2) The effect of features presented in rows 4 to 6 of Table 4 is negligible compared to other features.

Table 4. Features and their Effect in Clustering

row	System / Features		ES			OSBS			CSS			HACS		
			Feature Impact on Clustering			Feature Impact on Clustering			Feature Impact on Clustering			Feature Impact on Clustering		
			Positive	Negative	No effect	Positive	Negative	No effect	Positive	Negative	No effect	Positive	Negative	No effect
1	Actor		•			•			•			•		
2	Entity Classes		•			•					•			•
3	Control Classes		•			•			•					•
4	Different Relationship among Use cases	Extend			•			•			-			-
		Generalization Specialization			-			-			-			-
		Include			-		•				-			-
5	Weight of Control Class				•			•			•			•
6	Association weight of use case				•			•			•			•
7	Similarity Rate of each Use Case with other Use Cases				•	•			•					•

Table 5 shows quantitative results of sensitivity analysis in terms of number of errors resulting from inclusion or exclusion of features in clustering. Note that in Table 5 similarity rate of use cases with each other is computed based on binary features. Table 5 shows that Actors, Control classes and Entity classes are important feature.

Table 5. Quantitative results of features sensitivity analysis in terms of the number of errors in clustering

System	All Feature	Only Binary Feature	Only Continuous Features	All Features without			Binary Features without			Similarity Rate of each Use Case with other Use Cases without		
				Actor	Control	Entity	Actor	Control	Entity	Actor	Control	Entity
ES	0	0	1	1	5	5	1	3	7	1	3	5
OSBS	0	1	0	6	3	7	5	3	7	5	3	7
CSS	0	1	0	11	1	0	9	1	0	9	1	0
HACS	0	0	0	3	0	0	4	0	0	3	0	0

4.5.1 Sensitivity Analysis of Weight of Binary and Continuous Similarity Matrices

In equation (10) of step (B)-(C), in order to combine binary and continuous similarity matrices, weight of these matrices was considered equal. As a case we take into consideration OSBS to assess the effect of changes in matrices weights on accuracy of system functions clustering. Figure 3 depicts changes in clustering error where the weight of binary similarity matrix changes from 0.05 to 0.95. This figure shows that weight of binary similarity less than 0.6 for system 2 is desired.

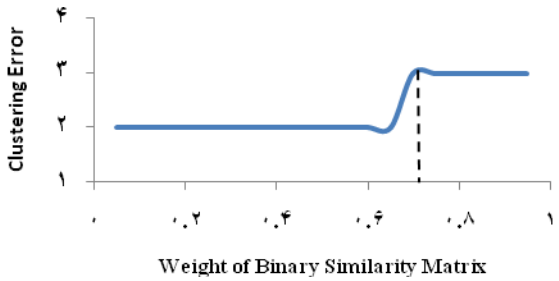

Figure 3. Sensitivity analysis diagram of weight impact of binary similarity matrix in OSBS

According to Sensitivity analysis results, allocation of weight 0.5 for binary and continuous similarity matrices is appropriate.

4.6 Approximately Minimal Feature Set

The set of features close to optimum for each system are determined and listed in Table 6 using the SBS method.

Table 6. Minimum Features Set for Functions Clustering

System	Actors		Entity classes		Control classes	
	Number	Minimum	Number	Minimum	Number	Minimum
ES	10	3	17	2	24	11
OSBS	3	2	6	4	10	1
CSS	5	4	18	1	11	0
HACS	4	1	6	1	7	1

4.7 Comparison of Results to Kim Method

We applied Kim et al. work [5] on four sample systems. We considered same weight for all features. The results are shown in Table 7. The proposed method

achieves better result than Kim method.

Table 7. Comparison of the proposed method and Kim method

Method	Systems Error			
	ES	OSBS	CSS	HACS
The Proposed Method	6	2	0	0
Kim Method	9	8	4	0

Advantages of the proposed method in comparison with the related works are as follows:

1-Presentation of a structured method to determine system software components.

2–Extraction of more features for clustering. The proposed method uses more features than other related works, and determines their effect in clustering through sensitivity analysis.

3- Using different clustering methods and choosing the best method in terms of the highest conformity to expert opinion.

4–Verifying the results of clustering methods with expert opinion and ensuring accuracy of the proposed method.

5–Using some sample software systems for validating the proposed method.

6-Sensitivity analysis by elimination of every feature and assessment of the effect of their elimination in increasing or decreasing the accuracy of clustering.

7-Elimination of weight assignment to features in clustering.

4.8 Extension

For further research, pre-conditions and post-conditions of each use case are also considered as a new feature. Use cases with similar pre-conditions/ post-conditions are more related than other use cases. Each pre-condition/ post-condition is considered a feature taking a value 1 or 0 based on its presence or absence in the use case. In sample software systems, only use cases of OSBS had pre-conditions/ post-conditions. So considering preconditions/ post-conditions of each use case, the clustering was repeated, the results show a decrease in clustering error. In the RBR and Direct clustering methods and RB method, the clustering errors became 0, 0, and 1 respectively. Thus, this feature can also be used in use case clustering.

4.9 Related Works

Evaluation of previous works [4-12] shows that: (1) clustering results have not been compared with expert opinion; (2) the presented methods have not been validated using a number of software systems; (3) various clustering methods have not been used; and (4) the effect of features on accuracy of clustering is not determined using sensitivity analysis, (5) there has been no guideline for determining the number of clusters, and (6) using less features, while these shortcomings have been addressed in this research. Related works were introduced in introduction section. The problems of these methods, in addition to the points mentioned, are as follows:

- The presented formula for calculating static and dynamic relationships in method [4] rigorously requires weighting relation types.
- Method [5] has not been validated by case study and it required weighting and did not give any guidelines in this regard.
- Method [6], 1) has not presented any guidelines to determine weight values (specially priority between types of relations between classes) and count the number of messages sent.
- In [7], the dependencies among requirements are manually identified. Also, coupling and dependency are treated the same way. But the assumed hypothesis is wrong.
- In method [8], the features used in identifying components and details of clustering method have not been presented.
- Method [9] provides high level guidelines, and it relies on domain experts in applying the guidelines.
- Nevertheless, the tool for implementing the BCI-3D method [12] cannot generate the information directly. Currently, all the information for identifying the business component is generated manually, due to the inability of communication between different platforms of the BCI-3D.

5 Conclusion

In this paper, a method was proposed to automatically determine system software components based on clustering of use cases features. First, the system use cases features were extracted and the components were determined based on the proposed method using different clustering methods. Then, the appropriate clustering method was selected by comparison of clustering methods results with expert opinion. To determine the appropriate number of clusters, metrics of the

interior cohesion of clusters and the coupling among them are used. By sensitivity analysis, the effect of each feature on accuracy of clustering was determined and finally the closest to optimum set of features providing the required accuracy in clustering were determined using the SBS method. The case studies conducted with four software systems, while validating the method, showed that *RBR* and Direct clustering methods that are extensions of K-means method have the most conformity with expert opinion. So, they were selected and recommended as the most appropriate methods. Innovation of this research is to propose a systematic method to determine system software components with specifications mentioned.

Acknowledgement

This work has been supported in-part by the Iranian Telecommunication Research Center (ITRC).

References

[1] M. Shaw، and D. Garlan. Software Architecture: Perspectives on an Emerging Discipline, Prentice Hall، 1996.

[2] L. Peng, Z. Tong, and Y. Zhang, "Design of Business Component Identification Method with Graph Segmentation", 3rd Int. Conf. on Intelligent System and Knowledge Engineering, pp. 296-301, 2008.

[3] R. Wu,"Componentization and Semantic Mediation", 33th Annual Conf. of the IEEE Industrial Electronics Society, Taiwan, pp. 111-116, 2007.

[4] M. Fan-Chao, Z. Den-Chen, and X. Xiao-Fei, "Business Component Identification of Enterprise Information System: A hierarchical clustering method", Proc. Of the 2005 IEEE Int. Conf. on e-Business Engineering, pp. 473-480, 2005.

[5] S. Kim, and S. Chang, "A Systematic Method to Identify Software Components", Proc of 11th Software Engineering Conf., pp. 538-545, 2004.

[6] H. Jain, and N. Chalimeda , "Business Component Identification – A Formal Approach", proc of the 5th IEEE Int. Conf. on Enterprise Distributed Object Computing, p.183, 2001.

[7] C. H. Lung, M. Zaman, and A. Nandi. Applications of Clustering Techniques to Software Partitioning, Recovery and Restructuring, J. Syst. Soft., 73(2):227-244, 2004.

[8] H. S. Hamza, "A Framework for Identifying Reusable Software Components Using Formal Concept Analysis", 6th International Conference on Information Technology: New Generations,pp. 813-818 ,2009.

[9] J. Cheesman, and J. Daniels, UML Components. A Simple Process for Specifying Component-Based Software, Addison-Wesley, Upper Saddle River, 2001.

[10] A. Albani, J.L. Dietz, and J. M. Zaha, "Identifying Business Components on the Basis of an Enterprise Ontology", Interoperability of enterprise software and applications, pp. 335-347. Springer, 2006.

[11] A. Albani, and J.L. Dietz, "The Benefit of Enterprise Ontology in Identifying", IFIP World Computing Conference, Santiago de Chile, Chile, 2006.

[12] A. Albani, S. Overhage, and D. Birkmeier, "Towards a Systematic Method for Identifying Business Components", Component-Based Software Engineering. LNCS 5282, pp. 262-277. Springer, Heidelberg, 2008.

[13] R. Xu and D. Wunsch, "Survey of Clustering Algorithms," IEEE Transactions on Neural Networks, Vol. 16, No. 3, MAY 2005, pp. 645- 678.

[14] G. Karypis, CLUTO: A Clustering Toolkit. Dept. of Computer Science, University of Minnesota, USA, 2002.

[15] F. Höppner, F. Klawonn, and R. Kruse, Fuzzy Cluster Analysis: Methods for Classification, Data Analysis, and Image Recognition, New York: Wiley, 1999.

[16] S. Ray, and R.H. Turi, "Determination of Number of Clusters in K-means Clustering and Application in Colour Image segmentation", Proc. of the 4th Int. Conf. on Advances in Pattern Recognition and Digital Techniques , Calcutta, India, pp. 137-143, 1999.

[17] OMG. OMG Unified Modeling Language Specification. March 2000.

[18] L. Kaufman, P. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, John, 2005.

[19] Y. Zhao, and G. Karypis, Criterion Functions for Document Clustering: Experiments and Analysis, http://citeseer.nj.nec.com/zhao02criterion.html, 2002.

[20] M. Steinbach, G.Karypis, and V. Kumar,"A Comparison of Document Clustering Techniques".KDDWorkshop on Text Mining,Vol. 34, pp. 35, 2000.

[21] H. Demuth, and M.Beale, "Neural Network Toolbox, For Use with MATLAB", Version 8, 2008.

[22] R. Caruana and D. Freitag, "Greedy Attribute Selection", Int. Conf. on Machine Learning, pp. 28-36, 1994.

About the authors

Saeed Jalili received the Ph.D. degree from Bradford University in 1991 and the M.Sc. degree in computer science from Sharif University of Technology in 1979. Since 1992, he has been assistant professor at the Tarbiat Modares University. His main research interests are software testing, software runtime verification and quantitative evaluation of software architecture. E-mail: Sjalili@modares.ac.ir

Gholam Reza Shahmohammadi received the Ph.D. degree from Tarbiat Modares University (TMU) in 2010 and the M.Sc. degree in oftware engineering from TMU in 2001, and the B.Sc. degree in Software engineering from Ferdowsi University of Mashhad in 1990. His main research interests are software engineering, quantitative evaluation of software Architecture, software metrics and software cost estimation. E-mail: Shahmohamadi@modares.ac.ir.

Seyed Mohammad Hossein Hasheminejad is a Ph.D. Candidate of computer engineering at Tarbiat Modares University (TMU). He received the M.Sc. degree in Software engineering from TMU in 2009, and the B.Sc. degree in Software engineering from Tarbiat Moalem University in 2007. His main research interests are Formal Methods for Software Engineering, Object-Oriented Analysis and Design, Search Based Software Engineering, and Self-Adaptive Systems. E-mail: SMH.Hasheminejad@Modares.ac.ir