

ANDRIAS

ISSN: 0721-6513

2015; Volume 40(3); Pp. 60-71

Received in 13 September 2015, Revised in 20 December 2015, Accepted in 24 January 2016

## **Identification of software systems components using a Self-Organizing Map competitive artificial neural network based on Cohesion and Coupling**

**Mohammad Ahmad Zadeh\***

**\*Corresponding Author**

Department of Computer, Bandar Abbas Branch, Islamic Azad University, Bandar Abbas, Iran

Email: Maz2140.Mail@Gmail.Com

**Gholam Reza Shahmohammadi**

Department of Information Technology, Olum Entezami Amin University, Tehran, Iran

Email: Shah\_Mohammadi@Yahoo.co.uk

**Mohammad Shayesteh**

Department of Computer, Bandar Abbas Branch, Islamic Azad University, Bandar Abbas, Iran

Email: Shayesteh.au@Gmail.Com

### **ABSTRACT**

One of the goals of software engineering is to provide a high quality system or a product. One of the methods to develop software systems is component base software development method, which software systems can be divided into parts called components using this method. Each part has a maximum internal cohesion and minimum communication with other components. This method can lead to save time, cost and reduce maintenance costs of the software system. In this study, we have proposed a method to identify the components based on SOM neural network approach and on the basis of coherence and Coupling measures, then, according to a new solution, components based on the use case are evaluated by applying on the three studies cases. The results show that, the use of the weighting method proposed and the SOM neural network approach will have good results to identify the components of the software system.

**Keywords: Software Engineering; Components; Neural Network; SOM; Clustering; Evaluation.**

### **1- INTRODUCTION**

The component base software engineering is one of the methods of software development, which are now widely used, it is a part of the software engineering, and defines, makes and implements the independent components for a software system using re-use based approach. A component may include a class, object, software services, functions, etc [1]. This leads to savings in time, cost, reliability and reduced costs related to software maintenance. Today, for companies having components that have features such as accuracy, efficiency, reusability and coherence and connectivity and appropriate complexity is an irrefutable fact. The software industry is developed so that minutes and seconds are very important in the process of software development. Having components that they can be trusted, and used them in several projects, will have a considerable impact on the speed of software development [2]. Meanwhile, today's software development teams use the exact timing for the development of their work. When you have a large software project, the appropriate components extraction becomes more difficult, hence there is need to an automated method to identify the components of our software systems. In this study, data from the Unified

Modeling Language Use Case diagrams that have been made in the design phase as appropriate is clustered using self-organized mapping neural network approach based on the criterion of cohesion and Coupling, and an appropriate method is presented to evaluate the components obtained by the proposed method and other methods.

## 2- SUGGESTED METHOD

In this section of the study, the proposed method is described, and the process of extracting data and identification of components are studied in detail, the stages of the implementation process of the proposed method, respectively includes preliminary data extraction, normalization of data, determining the number of clusters, SOM and assessment method based on the usecase.

### 2-1- THE INITIAL DATA EXTRACTION

In this study, data from the use case diagram has been extracted in the form of a matrix. The resulting matrix is largely similar to the adjacency matrix, the reason for the use of the matrix data is that clustering algorithms and its neural networks can be used easily as input data. [3] However, each row of the matrix represents a sample of data, and each column represents a property. Then, each sample is an  $n$ -dimensional vector. In the following, the matrix extracted with the relationship between them is discussed:

- Dependency relationship is shown as  $z$ .
- Inclusion relationship is shown as  $i$ .
- Extension relationship is shown as  $e$ .

Inclusion and extension relationships are very important, and considering the usecase, that have this type of communications, in a component can lead to reduced communication between the components, and increase the coherence or the interconnections between the components. The resulting matrix is a matrix  $n * n$ , symmetrical and its main diagonal is equal to zero. Preliminary data obtained from the use case diagrams of HACS which is expressed in Table (1) shows that this case study has 11 use case, which these use cases have dependence and extension relations, that in order will be shown as the signs of  $u$  and  $e$ . [4]

Table 1: Basic data about HACS studies

n	HACS Use Cases	1	2	3	4	5	6	7	8	9	10	11
1	Switch Light on/off	0	u	0	0	0	0	0	0	0	0	0
2	Set Intensity	u	0	0	0	0	0	0	0	0	0	0
3	On/Off	0	0	0	u	0	0	0	0	0	u	0
4	Set Temperature	0	0	u	0	0	0	0	0	0	0	0
5	Enabled/Disabled	0	0	0	0	0	u	0	0	0	u	0
6	Respond to Bugler Alert	0	0	0	0	u	0	0	0	0	0	0
7	Respond to Fire Alert	0	0	0	0	0	u	0	u	0	0	0
8	Send Alert user and police	0	0	0	0	u	u	0	0	0	0	0
9	Manage Remote Device	0	0	0	0	0	0	0	0	0	e	u
10	Manage Appliances	0	0	u	0	u	0	0	0	e	0	u
11	Access System	0	0	0	0	0	0	0	0	u	u	0

### 2-2- NORMALIZATION OF DATA

Normalization of Statistics has different meanings, that the simplest its application is data normalization or normalization of variables, and it consider data in the same domain when is not in a domain. In other words, it is possible that there are situations in data mining in which features in data have values in different in or range. These features or very large values may have a greater impact on the cost function compared to the values with smaller value. The problem will be removed with the attributes normalized so that their values are in the same domain. In this study, we want to

normal the data in the form of the row, because each row represents an instance of the data which will be provided for the clustering algorithm in the form of a row vector. To do this, we divide the total values of each row to the individual values of that row. In this study, to apply the data to the neural network, we normalize them using a universal approximate factor:

$$x_{nrmd} = 0.5 * \left( \frac{x - \bar{x}}{x_{max} - x_{min}} \right) + 0.5 \quad (1)$$

In this formula,  $x_{normal}$  is normalized value,  $x$  is the actual value of each parameter,  $x_{max}$  is the maximum value of the desired parameter, and  $x_{min}$  is the minimum value of the desired value [5]. We have used the following table to determine the desired values.

**Table 2: The table to determine the values of different types of use case diagram relationships**

The type of relation	Value	Symbol
Extension	6	E
Inclusion	8	I
Dependence	4	U
Association	1	I

Since that matrix elements obtained from the previous stage have abbreviation signs that were suggested by Table (2), at this stage, the amount of any symbol that represents the type of relationship is derived from Table (2) and is an alternative symbol [6, 14]. Then, we normalize on the basis of Table (1) the data in the matrix of primary data extraction step, which the results of normalization can be seen in Table (3). This table contains data normalized to the proposed approach which includes case study of HACS.

**Table 3: Data normalized of the HACS system**

n	1	2	3	4	5	6	7	8	9	10	11
1	0	1	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0.5	0	0	0	0	0	0.5	0
4	0	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0.5	0	0	0	0.5	0
6	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0	0	0.5	0	0.5	0	0	0
8	0	0	0	0	0.5	0.5	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0.6	0.4
10	0	0	0	0	0.28	0	0	0	0.42	0	0.28
11	0	0	0	0	0	0	0	0	0.5	0.5	0

### 2-3- DETERMINING THE INITIAL NUMBER OF CLUSTERS

In this study, an automated method must be proposed for determining the number of clusters to identify the components of the software system, hence, for this purpose, the following

$$k = \frac{n(\sum a_{ij}) + nN^3}{n \sum a_{ij}} \quad (2)$$

In the above equation,  $\sum a_{ij}$  is the sum of the values of the matrix,  $n$  is the non-zero data representing the number of connections in data matrix and  $N$  is the number of data matrix samples.

#### 2-4- SOM NEURAL NETWORK, CLUSTERING BASED ON COHERENCE AND COUPLING CRITERIA

In this study, we intend to identify components of a software system using self-organized map neural network technique. Since, our research can be done on a case study of HACK, and the case study is designed based on UML modeling language, we use the usecase diagram [7], because shows a overview of the system as comprehensive and without addressing the details. We use SOM neural network algorithms for clustering. This can consider input n-dimensional data as input vector to all neurons, and produce the final output [8]. To do this, we consider the number of clusters equal to the number of neurons in a neural network, and show them as c or w. Based on Figure (1) neurons in a single layer network compete with each other and a neuron will be selected as the winner neuron, which shows what input data is allocated to each cluster or component [8].

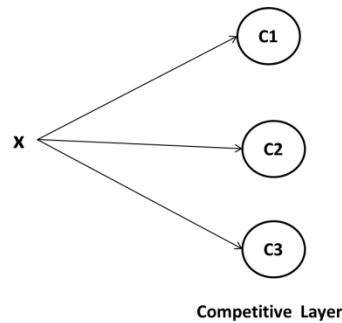


Figure 1: Competitive layer in a neural network

Each input such as  $x$  is an  $n$ -dimensional vector and is presented to the individual neurons, and the neurons that produces the greatest amount of output, is selected by the transfer function as the neuron that takes input data. [8] [7] it should be noted that our data are in the form of a symmetric sparse matrix. In the SOM Network, winner neuron admits the greatest impact, and its neighbors admit the less impact proportional to the distance from the winner neuron, and they are moved to the input data and the process is shown in Figure (2) [8].

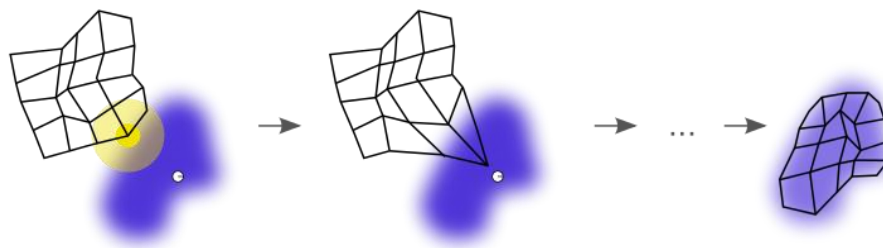


Figure 2: The process of changing the SOM neural network

In general, self-organizing map neural network operates in three main phases.

##### Competitive Phase

In this phase, what is the first phase of SOM neural network, the first input user case is applied in the form of an  $n$ -dimensional vector to all neurons? Neurons represent the center of the components. Each of them according to the FF criterion returns an amount based on internal cohesion and Coupling between components, according to the following formula. As long as data is not applied to a neuron, the FF benchmark to it is zero. When, on use case is applied to a neuron, based on the following criteria, the use case is allocated to a specified user:

- If the value of current  $FF_i$  is greater than the value of *pervious*  $FF_i$ , the UC User Case is allocated to the  $cmp_j$  component.
- If all values of current  $FF_i$  are equal to all the values of  $FF_i$ , the input User Case is allocated to the most privacy neuron.
- If all values of current  $FF_i$  are greater than the corresponding values of  $FF_i$ , the input component is allocated to a component randomly.

The relevant formulas for calculating the amount of FF criteria are expressed by the following equation:

$$\text{Component Cohesion} = \frac{\left( \sum_{i=1}^n \forall UC_i \in cmp_c, UC_j \in cmp_c, UC_{i,j} \neq 0 UC_{i,j} \right)}{n+1} \quad (3)$$

$$\text{Component Coupling} = \frac{\sum_{i=1}^n \forall UC_i \in cmp_c, UC_j \notin cmp_c, UC_{i,j} \neq 0 UC_{i,j}}{n} \quad (4)$$

$$FF = \text{Component Cohesion} - \text{Component Coupling} \quad (5)$$

In the above formula,  $UC_i$  is equal to the user case or the input vector,  $cmp_c$  is equal to the desired component and  $UC_{i,j}$  is element value of the user case i in the dimension j, and n is equal to the number of all use case assigned. In the above formula, the formula (3) is used for calculating the amount of components' cohesion, and formula (4) is used to calculate the amount of coupling between components, hence, the formula (5) is used for calculating the amount of FF. Thus, the process of assigning the use case is done to neurons. Finally, each neuron has a number of use case, which we call each of these categories as a component.

### Cooperative Phase

This phase is related to the cooperation between all neurons, and at this stage, each winner neuron must have a less impact on their neighboring and moves them to the desired data. [8] As a result, whatever the neurons of the neighbors, including the first, or second grade, or more neighbors, have more distance with the winner neuron, and are less irritating and moved towards relevant data. The following formula is used to determine the amount of cooperation between neurons and winner neuron: [8]

$$h_{ij}(x) = f(d_{ij}) \quad (6)$$

Where, i is index of the winner neuron, j the neuron of the desired neighboring,  $f(d_{ij})$  is a function to calculate the distance between winner neuron and neighbor's neuron [7]. The following two conditions are considered for function  $h_{ij}(x)$

$$\begin{cases} f(0) \geq f(d) & d \neq 0 \\ \lim_{d_{ij} \rightarrow \infty} h_{ij}(x) = 0 & d = 0 \end{cases} \quad (7)$$

The first condition: the distance of the neuron must be greater than zero, because if it is equal to zero, that means the distance of the winner neuron with itself. The second condition: when the neighbor is away, the value of this function is zero. It is possible that we consider immediate neighbors and consider the value of zero for the rest of the neighbors [8]. To calculate the value of function  $h_{ij}(x)$ , we use the Gaussian function.

$$h_{ij}(x) = \exp\left(-\frac{1}{2} * \frac{(d_{ij})^2}{2\sigma^2}\right) \quad (8)$$

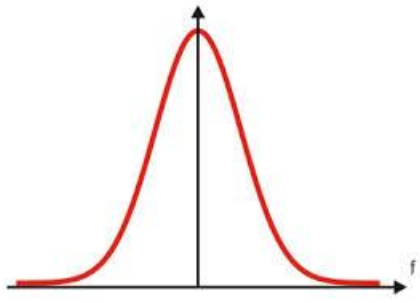


Figure 3: Gaussian function curve

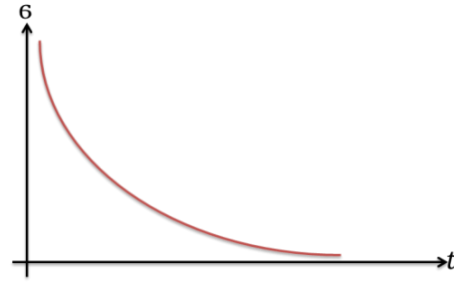


Figure 4: Reduced amount of Sigma curve

For calculating:

$$\sigma(t) = \sigma_0 * \exp\left(-\frac{t}{\tau}\right) \quad (9)$$

It is equal to 0.63. In the above formula,  $\tau$  is a time constant and decrease of  $\sigma$  reduces the width of the Gaussian curve above, and neurons act more local, ie any winner neuron has an impact on the less neighbors, but in that case, the value of  $\sigma$  is considered as a greater value, the winner neurons will act more for each input data and more neighbors will be affected. It is noteworthy that, when we want to optimize output, and gain a better result, we must consider two things in optimization. Firstly, the Exploration must be done in a larger environment and with more diverse responses, and second, we must have the right move to get or for Exploitation better responses. Based on these two principles, neither of the above two cases are not suitable. Our proposed solution is that  $\sigma$  or the standard deviation initially must be assumed as a larger number, and we must reduce its value over time. As a result, it makes the network converge quickly at first, and the convergence rate is reduced over time, and its accuracy increases. We want to gradually reduce the impacts on neighbors [8, 9].

### Adaptation Phase

The adaptation phase will be used to strengthen the connections between neurons, and a simple learning rule, called Kohonen is used:

$\Delta w$  is weight changes, ie what is the added to the weight.,  $\eta$  is an integer as a learning factor,  $Y$  equals the output (label),  $X$ , equal to the input. [8] Whatever the value of  $y$  is greater, the weight  $w$  will be more, whatever the input  $x$  is more, the weight  $w$  will be more, and these make the connection between two neurons be faster. With little change to adapt these calculations for no tag data, little change can be done [8].

$$w_j(t+1) = w_j(t) + \eta h_{ij}(x, t) * (x - w_j(t)) \quad (10)$$

$\eta$  is learning rate which we reduce it over time, and  $\tau$  is time constant that we propose this must be reduced over time:

$$\begin{cases} \eta = 0.1 \\ \tau = 1000 \end{cases} \quad (11)$$



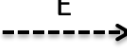
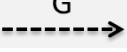

$\sigma_0$  amount is equal to the diameter of the neural network, which is dependent on the number of selected neurons, and  $\tau$  value is calculated based on the equation: [9]

$$\tau = \frac{1000}{\log \sigma_0} \quad (12)$$

### 3- EVALUATION OF THE COMPONENTS

It is possible that in the use case diagrams there is the relationship such as association, dependence, inclusion and extension between the two cases [10]. There are various relations between the various components in a user case diagram, including association, dependence, which will be described in detail. Since, in this study we try to obtain component diagram using the use case diagram, so evaluation components is not possible using current methods. Therefore, this method is generally based on the characteristics and methods of static and dynamic relationship between the components of a class. As a result, based on the types of relationships that exist between the use case of a obtained component, we try to assess the obtained components, and these relationships are presented in Table (4) with the recommended weight for them [10].

Table 4: Table of communication in the graph User

The type of relationship	UML Symbol	Abbreviation	The proposed weight	Quantity
Association		A	0.7	a
Inclusion		I	0.5	b
Extension		E	0.3	c
Globalization		G	0.9	d
Dependence		D	0.1	e

Based on the values and symbols in Table (4), the resulting components, which are equivalent to clustering results achieved by clustering algorithms, we propose the following calculations:

$$C_i = \frac{aA+bI+cE+dG+eD}{n} > 0 \quad (13)$$

$$Cohesion = \sum_{i=1}^c c_i \quad (14)$$

$$C_j = \frac{aA+bI+cE+dG+eD}{n} \geq 0 \quad (15)$$

$$Coupling = \sum_{k=1}^n c_j \quad (16)$$

$$M = Cohesion - Couplin \quad (17)$$

Where, n is the number of use case in the component, i and j are the clusters No and  $c_i$  is used to calculate the internal cohesion of the cluster based on the relationships between the use cases, and  $c_j$  is used to calculate the amount of communication between the cluster i with other clusters. As a result, whatever the amount of M relation is higher, the clustering is better.

### 4- EVALUATION OF THE PROPOSED METHOD

In this part of the study, we evaluate the proposed method. To do this, data is extracted from a case study based on the expressed method and it is normalized, and will be applied to the neural network and then evaluated.

#### 4-1- EVALUATION OF THE NUMBERS OF COMPONENTS

The number of components specified is used in the number of component identification methods, which with the expert opinion on the case studies of POS, QSB and HACS, are shown in Table (5).

**Table 5: Assessment criteria to determine the number of clusters**

Methods	QSB	HACS	POS
Expert opinion	4	3	6
Systematic method to identify the components of software systems	7	5	16
A method for Identification of system software components by clustering techniques	5	3	5
A method to identify software components by Genetic Algorithm	4	6	8
The proposed method for determining the number of components	4	4	6

By comparing the results in Table (5), we can see that the results of the proposed method are close to the results of expert opinion.

**Table 6: Calculating the amount of error of the methods for determining the number of clusters compared to the expert opinion**

Methods	HACS	Total errors	QSB	ES
Systematic method to identify the components of software systems	2	15	3	10
A method for Identification of system software components by clustering techniques	0	2	1	1
A method to identify software components by Genetic Algorithm	3	5	0	2
The proposed method for determining the number of components	1	1	0	0

According to Table (6), we can see that the total error obtained by the proposed method compared with the expert opinion is equal to 1, which has the best situation compared to other methods to identify.

#### 4-2- EVALUATION OF THE OBTAINED COMPONENTS

The average assessment of components to implement different algorithms for 5 times based on assessment of the component is shown in Table (7).

**Table 7: Assessment of the results to implement identifying the component methods for 5 times**

Case studies	SOM	SCI-GA	SH	Kim	The proposed method
HACS	0.1	-0.3	-4.8	-1.6	0.15
QSB	0.5	0.45	-6.3	-5.2	-0.5
POS	0.34	-4.5	-1.3	-7.4	-4.2
Average	0.31	-2.1	-4.13	-4.7	-1.51

According to the results in the table above, SOM network has the best average based on the proposed evaluation method, but since the network for data of our case studies is converged towards a cluster, and has led to create many privacy clusters [12], we have presented a proposed method by applying the new changes in the phases of implementation of this network, and clustered data based on it. The proposed approach will lead to better distribution of data in clusters, which it increases Coupling and reduces coherence compared to the base SOM method.

#### 4-3- EVALUATION OF THE PROPOSED METHOD

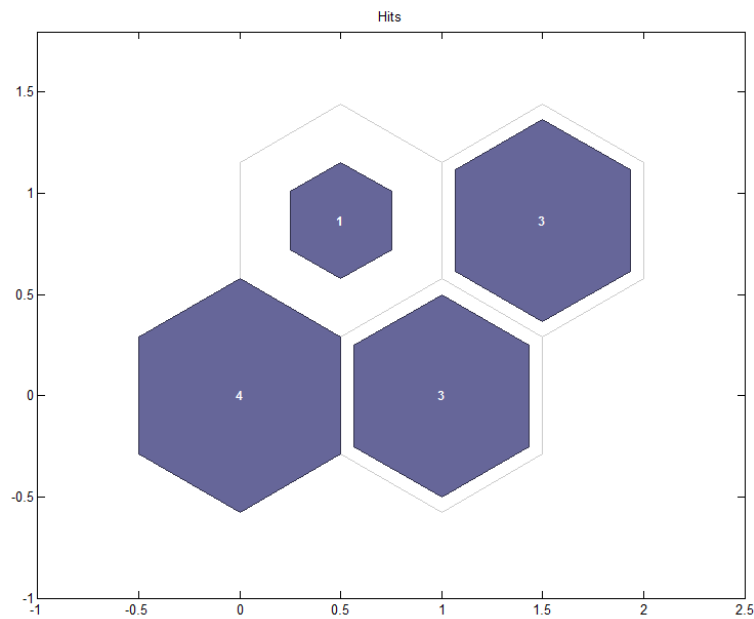
In this part of the study, we evaluate the proposed method. In recent studies, the criteria are suggested to compare the methods, including data mining, forecasting the appropriate number of components, coherence and coupling of components measures, unified Modeling Language diagrams such as use case diagram and class diagram [13]. In this paper, we present a method based SOM Neural network, which extracts the data in a simple process from use case diagram in the Unified Modeling Language, and identifies the software system components based on two criteria of coherence and Coupling of the components. The following table summarizes the features used in works done, and they are compared with the proposed method.



**Table 8: Final Table for component identification methods**

Methods	Use Case diagram	Class diagram	Coupling criterion	coherence criterion	No of Comp. achieved	Data Extraction
Systematic method to identify the components of software systems	-	√	-	-	Bad	Complex
A method for Identification of system software components by clustering	√	√	-	-	Good	Complex
A method to identify software components by Genetic Algorithm	√	-	√	√	It is not bad	Good
The proposed method for determining the number of components	√	-	√	√	Good	Simple

The results of the implementation of the proposed approach, and applying data of this case study on it show the good distribution of the data to the components, which the results will be examined.

**Figure 5: Data allocation to clusters**

The number of data assigned to each cluster is shown in the figure above, which, accordingly, the first cluster with one data has allocated the minimum numbers of data to itself, and the third cluster with 4 data has allocated the maximum numbers of data to itself. The above figure shows that data allocation to clusters is faced with better distribution, and the number of privacy clusters is reduced than other methods.

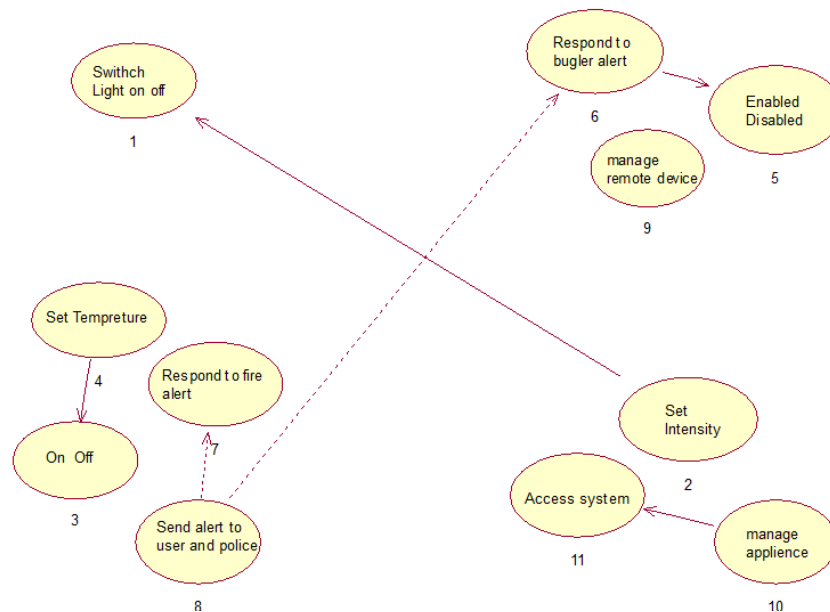


Figure 6: Results of clustering case study HACS for the proposed method

The results of distribution of use cases in components for the case study HACS data, show good distribution of the use cases compared to initial self-organized mapping neural network. Four components gained have only 2 of 6 possible connections. It should be noted that, all other connection are within components, so it increases the internal coherence of the components.

## 5- RESULTS

In this study, we have proposed a method to identify the components of the software system, in which the components are detected by an artificial neural network based on the criterion of cohesion and Coupling in the competitive phase. Determine the number of clusters was determined by a method, the amount determined by this method is very close to the expert suggested. The initial number of clusters was determined by a proposed method, which the amount determined by this method is very close to the value suggested by experts.

The proposed method has advantages, such as reduced time for data preparation, automatic determination of the components, the use of criteria of cohesion and Coupling for allocation of the use case to the desired component, relying on only one of the Unified Modeling Language diagrams called user case diagram, and not-using other graphs, and use case charts actors.

According to this study, we managed to identify the components of a software system using SOM artificial neural network approach. In fact, we have used a competitive neural network for clustering, and could prove that the use of neural network for data clustering and identification of the components of software systems, will lead to desired results.

## 6- THE WORKS DONE

Since, in this study, we use clustering techniques and adjacency matrix to identify the components of the software system. In this section, the methods are examined that create adjacency matrix using use case charts, cooperation diagram, and determine components of a software system by clustering techniques. A method for identification of software system components by clustering methods, was presented by Shah Mohammadi et al, It is a method for automatic determination of the software components, which performs the automatic identification using analysis phase of needs, and tries to

make maximum use of the features of the use case model, class diagrams and cooperative chart [1], SCI-GA method is presented by Hasheminejad et al which identifies component based on a genetic algorithm based on three criteria of Coupling, coherence and complexity. Systematic method is provided to identify the components by Kim et al., Which identifies components based on the characteristics of the class's charts [3].

## 7- RECOMMENDATIONS

According to research conducted on the component identification, and analysis of software and components obtained, and according to the method proposed in this study, it is suggested that, other criteria be used in addition to the criteria of coherence and Coupling such as complexity, reliability, reusability to identify the component by neural network in the phases of the neural network competition. Moreover, determination of the number of neurons in the network must be changed during the process of identifying the components in a dynamic manner. Since, software or diverse audience in the form of an actor may communicate with your software, it is better that the role of actors along with specific weights is taken into account in the process of data extraction.

## REFERENCES

- [1] Gholam Reza Shahmohammadi, Saeed Jalili and Seyed Mohammad Hosin Hasheminejad. "Identification of system Software Components using Clustering Approach." Journal of Object Technology, 2010.
- [2] Seyed Mohammad Hossein Hasheminejad, Saeed Jalili. "SCI-GA:Software Component Identification using Genetic Algorithm." Journal of Systems and Software, Volume 96, Pages 24–50, 2014.
- [3] Misook Choi , Eunsook Cho, "Component Identification Methods Applying Method Call Types between Classes." JOURNAL OF INFORMATION SCIENCE AND ENGINEERING, vol 22, 247-267 .2006.
- [4] S.M.H. Hasheminejad, S. Jalili, "CCIC: Clustering analysis classes to identify software components", Information and Software Technology, Vol 57, pp 329–351, 2015.
- [5] S. Kim, and S. Chang, "A Systematic Method to Identify Software Components", Proc of 11th Software Engineering Conf., pp. 538-545, 2004.
- [6] R.S. Pressman, "Software Engineering - A Practitioner's Approach", McGraw-Hill, 2001.
- [7] P. Kruchten," the Rational Unified Process: An Introduction", Third Edition, Addison-Wesley, 2003.
- [8] R. Xu and D. Wunsch, "Survey of Clustering Algorithms", IEEE Transactions on Neural Networks, Vol. 16, No. 3, pp. 645- 678, 2005.
- [9] 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.
- [10] B. Desgraupes, "Clustering Indices", Journal of University Paris Ouest,pp 14-28, 2013.
- [11] Kohonen T., Kaski S. and Lappalainen H., "Self-organized formation of various invariant feature filters in the adaptive-subspace SOM", Neural Computation 9, pp. 1321-1344. 1997.

- [12] Mukherjee A., "Self-organizing neural network for identification of natural modes", The Journal of Computing in Civil Engineering 11 (1), pp. 74-77. 1997.
- [13] Kohonen T., "Self-Organizing Maps, Springer series in Information Sciences", New York, Springer-Verlag, Vol. 30, pp. 501, 2001.
- [14] L. Bass, P. Clements, and R. Kazman, "Software Architecture in Practice", Second Edition, Addison-Wesley, 2003.
- [15] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, "Pattern Oriented Software Architecture: A system of Patters", Wiley Press, 1996