I.J. Modern Education and Computer Science, **2018, 8, 33-40**
Published Online August 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2018.08.04

Clustering based Architecture for Software Component Selection

Jagdeep Kaur
Department of Computer Science and Engineering & Information Technology,
The NorthCap University Gurugram, Haryana
Email: jagdeep_kaur82@rediffmail.com

Pradeep Tomar
Department of Computer Science and Engineering, School of ICT Gautam Buddha University
Greater Noida, Uttar Pardesh
Email: parry.tomar@gmail.com

Received: 24 November 2017; Accepted: 10 July 2018; Published: 08 August 2018

Abstract—The component-based software engineering (CBSE) consists of component selection, qualification, adaptation, assembly and updating of components according to the requirements. The focus of this paper is software component selection only. Now-a-days many selection processes, techniques and algorithms are proposed for this task. This paper presents generalized software component selection architecture using clustering. The architecture is divided into four tiers namely Component Requirements and Component Selection Tier, Query and Decision Tier, Application logic tier with Clustering and Component Cluster Tier. The architecture offers manifold advantages like i) presenting a generalized architecture where the existing techniques can be applied, reducing the search space for the component selection. ii) It also illustrates the usage of clustering in the software component selection without the need for pre-specification of number of clusters and considering more than two features while clustering. iii)The cluster validation is performed to check the correctness of the clusters. This complete selection process is validated on a representative instance of set of components.

Index Terms—Component-based Software engineering, Component Selection Process, Clustering based Architecture for selection.

I. Introduction

Now-a-days software reuse has become a buzzword. The reusability is achieved in the form of source code, software architecture or software components. The component based software engineering is a methodology used for software development that advocates the use of software reuse. The main properties of the components are that they are independent, implementation is not visible, communication among the components take place through the interfaces. However, there are some challenges like certification of components, emergent property prediction and trade-off analysis of various component features for selection. The trade-off analysis or the software component selection is an active area of research among the researchers. The software selection process consists of finding the component that provides the desired functionality, from the finite set of components. Further, the selection process consists of selecting the most suitable component from the candidate components based on given requirement or constraints.

According to [1], CBSE emphasizes the "the 'buy, don't build' philosophy". CBSE approach is used to reuse the already developed and testable software codes to develop economical software, which can be developed within shortest 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 CBSD, development of software by writing code is replaced by selecting, assembling and integrating software components [2]. 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 [3]. The use of the 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 feel 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 CBSD and the success of the final system depends on the component selection process. The components are selected mainly based on its functionality from the repository, but the non- functional

properties or the information provided by the developers about the components also plays a crucial role in component selection. As the number of available components in the repository grows, the selection of a set of components based on a set of functional requirements and on the other hand, minimizing or maximizing other objectives like price, number of components etc. has become extremely difficult.

This paper presents a software component selection technique using clustering with a four tier architecture. The remainder of the paper is structured as follows. Section II presents the related work. Section III describes the four-tier architecture for component selection using clustering. Section IV discusses a representative instance and section V follows the conclusion.

II. Related Work

In the component based development, component selection paradigm plays a vital role. It emphasizes on composing the application from the already existing one. The major activities of CBD are component qualification, adaptation and composition which enhance the reusability. The component qualification deals with ensuring that the component performs its desired functionality and is according to desired quality characteristics like reliability, performance and usability. The component adaptation deals with adapting the components with the architectural design rules, and finally component composition deals with putting together the qualified, adapted and engineered components in the architecture developed for the application. Initially, the selection technique based on functional requirements was proposed. This was followed by numerous techniques covering various issues like conflicting goals, reducing the gap between stated requirements and actual requirements as in CARE [4]. In case of PECA [5], it helps the decision maker by establishing suitable criteria. The CSSP (COTS Software Selection Process) helps in evaluating as well as ranking the software components with risk management. In case of CSCC (Combined Selection of COTS Component) the interoperability among the in-house developed components and the 3rd party components are checked in order to avoid schedule delays [6]. In order to select whether the in house developed component is better or the market available component, a fuzzy based approach is used as given in [5]. Some selection techniques incorporate testing with selection like in IDM [7] and PRISM [8]. Another technique laid emphasis on the non-functional requirement usability [9]. Here, an index is proposed to rank the components based on its value. In the work done by Jadhav and Sonar [10] it takes into consideration the previous similar components searched using rule based and case based reasoning. In [11], the multi-criteria based decision is taken using cross referencing. The computational intelligent techniques also played a major role and selection techniques using them are proposed. As in case [12] entropy based fuzzy k modes algorithm is used to divide the components into clusters and then isolate the cluster nearest to user's choice. Few years back, fuzzy clustering based approach was developed. It uses metrics based on clustering analysis [13]. Recently, a case survey was presented [14] for making the choice among the different component sourcing options like in-house components, COTS, Open Source Software or outsourcing the components. Their results showed that the most of the solutions are deterministic and based on optimization. A very few non-deterministic solutions were also found. Another study [15] analyzes different COTS selection techniques and presents automatic component selection techniques. Apart from this researchers [16] have highlighted the trade-offs between different factors for selecting the in-house components, COTS, Open source Components or outsourcing them. The detailed analysis of research work carried out in last twenty eight years is presented by the researchers in [17]. The authors in [18] proposed a dynamic reconfiguration of robot software service by reuse software service component using clustering techniques. A multi objective optimization based algorithm is recently proposed to select the software products [19][20] . The authors [21] have demonstrated the Buy vs Build strategy for selecting components for a modular system using fuzzy bi-criteria optimization model. The researchers [22] have done an assessment of reusability of web service components that can help in component selection too.

After studying these techniques, following points are noted:

- The software component search space is quite large and it is a challenging task to narrow down the search space.
- There is a lack of generalized architecture for software component selection.
- The role of non-functional requirements in final decision making is very less.
- The subjective judgments can be replaced by incorporating fuzzy logic.

The proposed architecture is similar to the existing techniques in the following ways:

- Reduction in the search space.
- Consideration of extra-functional properties.

However, the proposed architecture differs in the following aspects with the existing techniques:

- Multi-attributes of varied types can be handled.
- The final component retrieved is having maximum inter-cluster distance and minimum intra-cluster distance.

III. Four-Tier Architecture for Software Component Selection

A component is selected based on the function it provides in Component Based Software Engineering (CBSE). Many techniques have been proposed for

component selection under varied situations to select the optimal component from component set of same functionality. Here, the selection process is viewed as architecture. It is divided into four tiers namely Component Requirements and Component Selection Tier, Query and Decision Tier, Application logic tier with Clustering and 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 prior declaration of clusters will be eliminated and the cluster validation is performed to check the correctness of the clusters. This architecture is validated on a case study of set of sorting and searching components.For a component selection problem the system designer considers several candidate components for which the data available includes estimates of the cost of acquisition, customer desirability, development time and expected revenue. The designer may also have information about the dependencies between components and may wish to include other factors in the decision making process, such as the priority given to each of the customers. From the set of all components, the designer must search for a subset that balances these competing concerns in the best way possible. 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 technique is comparable to the stock selection process for investment. This paper proposes a new architecture for the entire component selection which is similar to stock selection for investments, as the objective of stock selection 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. The architecture is motivated by the one proposed by the work for stock selection of investment [23].The architecture of component selection process has the following tiers: Component Requirements and Component Selection Tier, Query and Decision Tier, Application logic tier with Clustering and Component Cluster Tier as shown in Fig. 1. This paper proposes four-tier architecture, which helps the developers to make decision while selecting the components. The proposed 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.

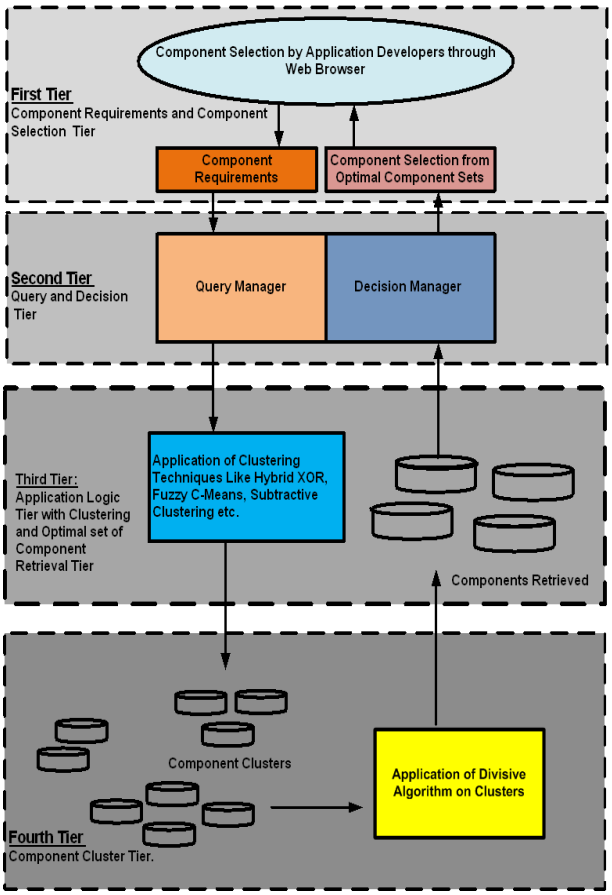

Fig.1. Four -Tier Architecture for Component selection using Clustering

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

A. First Tier: Component Requirements and Component Selection Tier

The first tier is known as Component requirements and Component Selection Tier as shown in Fig.2 to give the component requirements through the web browser using a natural language. The request for components is stated in a form of a query and as an outcome number of components will be returned from the optimal component set. This retrieval process can be further enhanced by using iteration for generating more alternative components for the clients and system analyst until a particular level of confidence is reached according to the original stated query.

This first tier will provide interface for client requirement and system analyst. It also presents the analysis of selected optimal set of component according to component requirement. In return, this interface provides possible number of solutions according to the problem that are feasible for construction of an optimal selection of components. The requirements can be stated in terms of features of the components described in the repository. For example, for the proposed case study the

features of the sorting and searching components can be Halstead program volume, time complexity, cyclomatic complexity and size of input list. In, return the optimal component sets are returned, as solution, to system analyst.

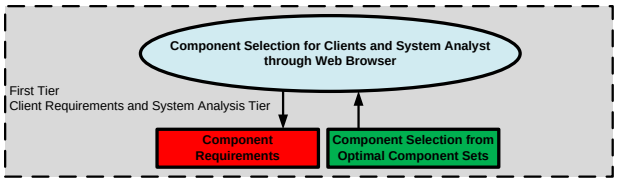

Fig.2. First Tier: Component Requirements and Component Selection Tier

B. Second Tier: Query And Decision Tier

The Second tier as shown in Fig. 3 is known as query and decision tier. It shows how the query manager and decision manager interacts which help in generating query and solution.

1) Query Manager

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.

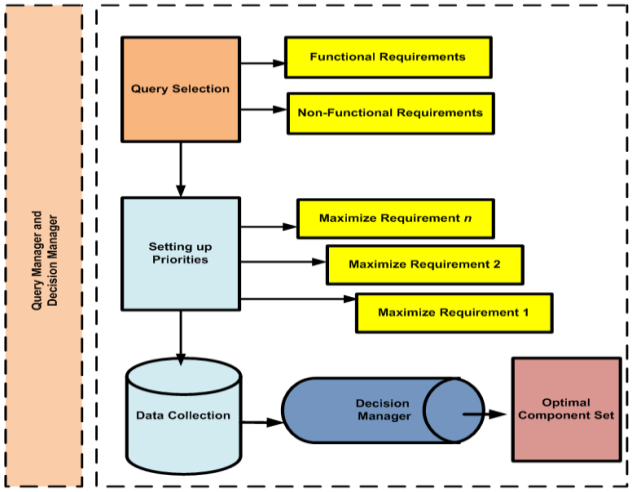

Fig.3. Queries and Decisions Formulation

The Fig. 3 shows the working of query manager and decision manager. The query manager performs query selection where the queries are formulated based on functional and non-functional requirements. Afterwards, the priorities are set for the requirements, identifying the most important ones from the least important. The requirements can be stated in terms of objectives need to be maximized or minimized. On the basis of priorities set by the users, data is collected with the help of application logic from the database tier and then decision manager will generate the solution.

The requirements can be stated in the following ways:

i) Functional Requirements: The user can state the requirements in terms of functionality required.

ii) 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 Halstead program volume, time complexity, cyclomatic complexity, input size etc.

2) Decision Manager

It 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.

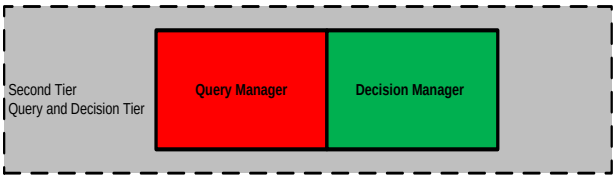

Fig.4. Second Tier: Query and Decision Tier

C. Third Tier: Application Logic Tier with Clustering using Hybrid XOR Similarity Function

The third tier as shown in Fig. 4 consists of *application of hybrid XOR similarity function* and *Components catalogue*. The clustering used in this architecture is based on the hybrid XOR similarity function. The catalogue is formed on the basis of ratio of inter component similarity to the intra component similarity after application of Divisive algorithm in the fourth tier. The application logic can be modified according to the given component set. Here in this tier a hybrid XOR similarity function based clustering technique is used. It forms the clusters of similar components from where the user can select the desired components. Some non-functional factors like coupling, cohesion, reliability, fault tolerance, component size, time, cost, compatibility, value of intra-modular coupling density etc. can be used.

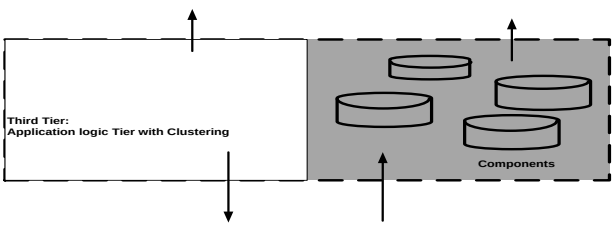

Fig.5. Third Tier: Application Logic Tier with clustering

The clusters thus formed will be having high cohesion and low coupling. Using the algorithm in [18] the process of clustering is applied on a case study. Another application logic can be application of other fuzzy clustering techniques like fuzzy c-means, subtractive clustering, as shown in Fig. 5.

D. Fourth Tier: Component Cluster Tier

It deals with the application logic to select the database from the different data sources. The third tier will generate three clusters and on these clusters Divisive Algorithm is applied, which is a type of hierarchical clustering. In this tier, firstly the cluster set that needs to be split further is chosen. For simplicity purpose the authors opt the biggest cluster. For n number of components, the number of possible bipartitions are 2^{n-1} - 1. For large number of components this will become computationally expensive, so to reduce the complications, the cluster is divided based on a specific feature in which the user is more interested. According to proposed methodology, it keep on dividing the cluster further until it reach a point when no more division is possible or it reaches a condition where the clusters have become singletons.

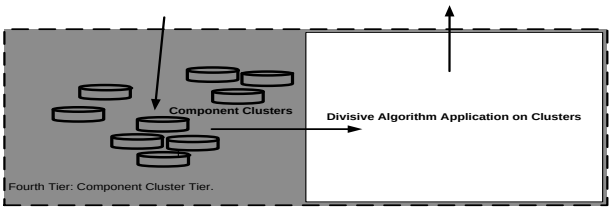

Fig.6. Fourth Tier: Component Cluster Tier

IV. Representative Instance

This complete architecture for software component selection can be validated using a set of software components. The java programs for sorting *Bubble sort, Merge Sort, Insertion sort, Quick Sort, Selection Sort, Heap sort, Radix Sort, Bucket Sort and Shell sort* are considered.

Table 1. Sorting Components and its features

Software Component	Halstead Metric	Cyclomatic complexity	Time Complexity	Size of Input
Bubble sort(SO1)	172	4	$O(n^2)$	SMALL
Merge Sort(SO2)	633	7	O(n log(n))	LARGE
Insertion sort(SO3)	164	3	$O(n^2)$	SMALL
Quick Sort(SO4)	367	7	$O(n^2)$	LARGE
Selection Sort(SO5)	175	4	$O(n^2)$	SMALL
Heap sort(SO6)	511	7	O(n log(n))	LARGE
Radix Sort(SO7)	642	10	$O(n^k)$	SMALL
Bucket Sort(SO8)	332	8	$O(n^k)$	LARGE
Shell sort(SO9)	299	5	O(n log(n))	LARGE

The main features of these components considered for selection purpose are Halstead program volume, cyclomatic complexity, time complexity and type of input list, as shown in Table 1 and Table 2. The clustering approach is the one taken from [18] in where a XOR based similarity function is used for finding extent of likeness between component cluster. A similarity matrix of the order n-1 by n is constructed for given n components. The input in the third tier is similarity matrix based on the given components and resultant is the set of clusters formed in the fourth tier.

A Application of Xor Clustering for Cluster Computation

The new clusters would be formed with the help of XOR similarity. Some pre-processing is required for the features of the components before applying the similarity matrix. The sorting components can be represented using the possible set of values for different features as follows:

- Halstead Metric: {1-199:A, 200-400:B, 401-650:C, 651 and above : D}
- Cyclomatic Complexity: {2-4: A, 5-7: B, 8-11: C, 50 and above : D
- Time Complexity: {$O(n^2)$: A, O(n log(n)): B, $O(n^K)$: C, O(n): D, O(log n): E
- Size of input list: {Small : A, Large: B }

Table 2. Updated Table for sorted components

Software Component	Halstead Metric	Cyclomatic complexity	Time Complexity	Size of Input
Bubble Sort(SO1)	A	A	A	A
Merge Sort(SO2)	C	B	B	B
Insertion Sort(SO3)	A	A	A	A
Quick Sort(SO4)	B	B	B	B
Selection Sort(SO5)	A	A	A	A
Heap Sort(SO6)	C	B	B	B
Radix Sort(SO7)	C	C	C	A
Bucket Sort(SO8)	B	C	C	B
Shell Sort(SO9)	B	B	B	B

Now, this information can be represented in the form of a Boolean matrix with rows indicating each component and column corresponding to each unique attribute of the software components. A matrix of D[9,9] for 9 components is formed and only the upper triangular region is considered. The cells are filled according to the similarity function S for which each component pairs form the input. The feature vector representation of the component set is shown and the feature vector of each component is replaced by count of number of zeros in the tri state feature vector. This is shown in the Table 4.

Table 3. XOR Similarity Measure

A	B	S(A,B)
0	0	Z
0	1	1
1	0	1
1	1	0

The feature vector representation of the component set is shown in the Table 3 and then the feature vector of each component is replaced by count of number of zeros in the tri state feature vector. The similarity between the components is computed using the hybrid XOR similarity function for example for SO1 and SO2 gets (1,Z,1,z,,1,1,z,Z,1,1,Z,Z,z,1,1)=* that is no similarity and for SO1 and SO3 after applying similarity function the result is (0,Z,Z,z,0,Z,Z,Z,0,Z,Z,Z,Z,0,z)=4.

Table 4. The similarity matrix with the feature vector replaced by count of 0s

	SO1	SO2	SO3	SO4	SO5	SO6	SO7	SO8	SO9
SO1	*	*	4	1	4	*	1	*	*
SO2	*	*	*	2	*	4	1	1	3
SO3	*	*	*	1	4	*	1	*	*
SO4	*	*	*	*	*	2	*	2	3
SO5	*	*	*	*	*	*	1	*	*
SO6	*	*	*	*	*	*	1	1	3
SO7	*	*	*	*	*	*	*	2	*
SO8	*	*	*	*	*	*	*	*	2
SO9	*	*	*	*	*	*	*	*	*

The entries of the similarity matrix as shown in Table 4 are made according to following:

Find the highest value from the Table 4, which is 4 here and target those cells as they form the best candidate solutions and replace those cells by *. The component pairs (SO1,SO3), (SO3,SO5), (SO1,SO5) and (SO2,SO6) are having the value 4. So, (SO1,SO3,SO5) is formed as the first set. Find the next highest value from the Table 4 which is 3 and target those cells as they form the best candidate solutions and replace those cells by *. Now consider only un-clustered components sets (SO2, SO4, SO6, SO7, SO8, SO9) and look for value 3 in corresponding cells. Now,(SO2,SO9), (SO4,SO9) (SO6,SO9) (SO8) have value 3, So cluster1 is formed as (SO1,SO2,SO3,SO4,SO5,SO6,SO9)

Find the next highest value from the Table which is 2 and target those cells as they form the best candidate solutions and replace those cells by *. Consider only un-clustered components sets (SO7,SO8), (SO7,SO8) (SO8,SO9). Now, no more components are left, the procedure stops now.

The clusters finally formed are:

1. R1 is (SO1, SO2, SO3, SO4, SO5, SO6, SO9)
2. R2 is (SO7, SO8)

Now, these clusters are taken from the component repository as explained in the fourth tier.

In the fourth tier, the first cluster set is taken for Divisive Algorithm application as it is the biggest among the others. There would have been 63 pairs of possible partitions (2^6 -1), the number of calculations would be very large. So, to ease the computations the cluster is divided according to a user specified criteria of time complexity. Referring to Table 4 the bipartition formed is (R_1^1, R_1^2) which is represented as R_1^1 is (SO1, SO3, SO4, SO5) and R_1^2, is (SO2, SO6,SO9).

Now, R_1^1 can be further divided based on another feature that is type of input list. It generates R_2^1 and R_2^2 which is represented as (SO1, SO3, SO5) and (SO4) respectively. Finally the new clusters generated are {(SO1, SO3, SO5), (SO4) and (SO2, SO6, SO9) }

The clustering will stop here as no progress can be made further.

Table 5. The final Candidate Component Clusters

Cluster No.	Clusters	Components Description
1	(SO1,SO3,SO5)	Bubble sort, Insertion sort, Selection Sort
2	(SO4)	Quick sort
3	(SO2,SO6,SO9)	Merge Sort, Heap Sort and Shell sort
4	(SO7, SO8)	Radix Sort, Bucket Sort

The final set of component clusters are shown in Table 5.

V. Conclusions

This paper proposed four-tier architecture to develop the component selection system which is capable of providing effective optimal set of components for component selection problem and CBSD. Here, the user specifies the requirements and as an output it gets the number of component clusters grouped with a greater degree of similarity with peer member of the cluster. The components are clustered together using hybrid XOR similarity function used for document clustering. The clusters so formed are further split using divisive algorithm based on a pre-specified criterion. This architecture fulfills the advantages as proposed in the abstract of the paper:

1. In the proposed architecture the existing XOR based similarity can be applied.
2. There is no need to specify the the number of clusters as in the traditional fuzzy c-means and subtractive clustering techniques.
3. The validation of the clusters is performed using Divisive Algorithm which is in This study can be further validated by using a large component set. The future work will consider the application of fuzzy clustering technique to deal with the cases when the same component tends to lie in two different clusters.

REFERENCES

[1] P. C Clements, “From Subroutines to Subsystems: Component-Based Software Development”, *American Programmer*, vol. 8, pp. 31-33, 1996.

[2] N. S. Gill, “Reusability Issues in Component-Based Development”, *ACM SIGSOFT Software Engineering Notes*, vol. 28, no. 6, pp. 30,2003.

[3] Crnkovic “Component-Based Software Engineering-New Challenges in Software Development”, *Software Focus*, vol. 2, no. 4, pp. 127-133, 2002.

[4] L. Chung and K. Cooper, “Defining Goals in A COTS-Aware Requirements Engineering Approach”, *Systems Engineering*, vol. 7,no.1, Wiley, pp. 61-83, 2004.

[5] Comella-Dorda S., Dean J., Morris E., Oberndorf P.(2002): ‘A Process for COTS Software Product Evaluation’, in proceedings of *International Conference on COTS-Based Software Systems*, Lecture Notes in Computer Science, vol. 2255, pp. 86-96.

[6] X. Burgués, C. Estay, X Franch, J. A. Pastor, C. Quer, “Combined Selection of COTS Components”, in proceedings of *International Conference on COTS-Based Software Systems*, Lecture Notes in Computer Science vol. 2255, pp. 54-64,2002.

[7] W. Zhiqiao, C.K. Kwong C. K., J. Tang and J. W. K Chan “Integrated Model for Software Component Selection with Simultaneous Consideration of Implementation and Verification”, *Computers and Operation Research*, vol. 39, no. pp. 3376-3393, 2012.

[8] R. W. Lichota, R. L Vesprini, and Swanson B. “ PRISM: Product Examination Process for Component Based Development ”, in proceedings of Symposium on Assessment of Software Tools and Technologies , pp. 61-69, 1997

[9] N. Upadhyay, B. M. Deshpande and V. P Agrawal, “Concurrent Usability Evaluation and Design of Software Component: a Digraph and Matrix Approach”, *IET Software*, vol. 5, no. 2, pp.188-200, 2011.

[10] S. Jadhav and R. M Sonar “Framework for Evaluation and Selection of the Software Packages: A Hybrid Knowledge Based System Approach”, *Journal of Systems and Software*, vol. 84, no.8, pp.1394-1407,2011.

[11] Becker,M. Kraxner, M. Plangg and A. Rauber, ”Improving Decision Support for Software Component Selection through Systematic Cross-Referencing and Analysis of Multiple Decision Criteria”, in *Proceedings of 46th Hawaii International Conference on System Sciences*, pp. 1193-1202,2013.

[12] Stylianou and A. S. Andreou, “A Hybrid software Component clustering and retrieval Scheme Using an Entropy-based Fuzzy k-modes Algorithm”, in *Proceedings of 19th IEEE International Conference on Tools with Artificial Intelligence*, 2007, pp. 202-209.

[13] C Serban, A Vescan and H. F. Pop,” A New Component Selection Algorithm Based on Metrics and Fuzzy Clustering”, *Creative Mathematics and Informatics*, vol. 1, no. 3,pp.505-510,2009.

[14] Petersen, Kai, Deepika Badampudi, Syed Shah, Krzysztof Wnuk, Tony Gorschek, Efi Papatheocharous, Jakob Axelsson, Severine Sentilles, Ivica Crnkovic, and Antonio Cicchetti. "Choosing Component Origins for Software Intensive Systems: In-house, COTS, OSS or Outsourcing?--A Case Survey." IEEE Transactions on Software Engineering (2017).

[15] C.Srinivas, V. Radhakrishna and C. V. Rao, “Clustering Software Components for Program Restructuring and Component Reuse Using Hybrid XOR Similarity Function”, in *Proceedings of AASRI Conference on Intelligent Systems and Control*, Vancouver, Canada , pp. 319-328, 2013.

[16] Badampudi, D., Wohlin, C. and Petersen, K., Software component decision-making: In-house, OSS, COTS or outsourcing-A systematic literature review. *Journal of Systems and Software*, 121, pp.105-124, 2016.

[17] Vale, T., Crnkovic, I., De Almeida, E.S., Neto, P.A.D.M.S., Cavalcanti, Y.C. and de Lemos Meira, S.R., Twenty-eight years of component-based software engineering. Journal of Systems and Software, 111, pp.128-148, 2016.

[18] Srivastava, A.K. and Kumar, S., Dynamic Reconfiguration of robot software component in real time distributed system using clustering techniques. Procedia Computer Science, 125, pp.754-761, 2018.

[19] Lian, X., Zhang, L., Jiang, J. and Goss, W., An approach for optimized feature selection in large-scale software product lines. Journal of System s and Software., pp. 636-651, 2017

[20] J. Kaur and P. Tomar,"Multi Objective Optimization Model using Preemptive Goal Programming for Software Component Selection", International Journal of Information Technology and Computer Science(IJITCS), vol.7, no.9, pp.31-37, 2015

[21] Jha, P.C., Bali, V., Narula, S. and Kalra, M., Optimal component selection based on cohesion & coupling for component based software system under build-or-buy scheme. Journal of Computational Science, 5(2), pp.233-242, 2014

[22] P Singh, P Tomar,"Web Service Component Reusability Evaluation: A Fuzzy Multi-Criteria Approach", International Journal of Information Technology and Computer Science(IJITCS), Vol.8, No.1, pp.40-47, 2016

[23] P. Tomar, D. K. Sharma and H.Sharma, “A Web Based Four-Tier Architecture Design for Stock Selection Decision Support System for Investments”, *Review of Business and Technology Research*, vol.5, no.1, pp. 116-121, 2011.

Authors’ Profiles

Jagdeep Kaur working as Assistant Professor in Computer Science and Engineering Department, The NorthCap University, Gurgram, Haryana, India; and is Ph.D. in Computer Science Engineering from School of Information and Communication Technology, Gautam Budhha University,Greater Noida,UP, India. She holds a Bachelor of Technology (B.Tech.) degree in Computer Science and Engineering from BCET, Gurdaspur, Punjab, India (2003). She obtained her Master of Technology degree in Computer Science from Department of Computer Science and Engineering, Punjabi University, Patiala India (2005). Her research interests include Component Based Software Engineering, Software Reuse, Software Testing and Software Process Metrics.

Dr. Pradeep Tomar is working as Faculty Member in the School of Information and Communication Technology, Gautam Buddha University, Greater Noida, INDIA since 2009. Dr. Tomar earned Ph.D. from Department of Computer Science &

Applications, M. D. University, Rohtak, Haryana, INDIA. He is also a member of IEEE, IEEE Computer Society, Computer Society of India (CSI), Indian Society for Technical Education (ISTE), Indian Science Congress Association (ISCA), International Association of Computer Science and Information Technology (IACSIT) and International Association of Engineering (IAENG). He served as a reviewer of journals and conferences and worked as advisory board members in national and international conferences. Two books "Teaching of Mathematics" and "Communication and Information Technology" at national levels have been authored by Dr. Tomar. Dr. Tomar has been awarded with Bharat Jyoti Award by India International Friendship Society in the field of Technology in 2012. He has been awarded for the Best Computer Faculty award by Govt. of Pondicherry and ASDF society. His biography is published in Who's Who Reference Asia, Volume II. Several technical sessions in national and international conferences had been chaired by Dr. Tomar and he delivered expert talks in FDP, workshops, national and international conferences. Three conferences have been organized by Dr. Tomar.

How to cite this paper: Jagdeep Kaur, Pradeep Tomar, " Clustering based Architecture for Software Component Selection ", International Journal of Modern Education and Computer Science(IJMECS), Vol.10, No.8, pp. 33-40, 2018.DOI: 10.5815/ijmecs.2018.08.04

