A Fuzzy-Based Approach for the Multilevel Component Selection Problem

Andreea Vescan(✉) and Camelia Şerban

Computer Science Department, Babes-Bolyai University,
1, M. Kogalniceanu, Cluj-Napoca, Romania
{avescan,camelia}@cs.ubbcluj.ro

Abstract. Component-based Software Engineering uses components to construct systems, being a means to increase productivity by promoting software reuse. This work deals with the Component Selection Problem in a multilevel system structure. A fuzzy-based approach is used to construct the required system starting from the set of requirements, using both functional and non-functional requirements. For each selection step, the fuzzy clustering approach groups similar components in order to select the best candidate component that provide the needed required interfaces. To evaluate our approach, we discuss a case study for building a Reservation System. We compare the fuzzy-based approach with an evolutionary-based approach using a metric that assess the overall architecture of the obtained systems, from the coupling and cohesion perspective.

Keywords: Component selection · Metrics · Fuzzy analysis · Architecture assessment

1 Introduction

Component-Based Software Engineering (CBSE) is concerned with designing, selecting and composing components [5]. As the popularity of this approach and hence number of commercially available software components grows, selecting a set of components to satisfy a set of requirements while minimizing a set of various objectives (as cost, number of used components) is becoming more difficult. The problems of identification and selection the right software components out of a range of choices to satisfy a set of requirements have received considerable attention in the field of CBSE during the last two decades [2,7].

In this paper, we address the Component Selection Problem (CSP) [14]. Informally, our problem is to select a set of components from a components repository which can satisfy a given set of requirements while minimizing/maximizing various criteria. Components are themselves compositions of components. This give rise to the idea of composition levels, where a component on level i may be decomposed (using more components) at level $i+1$ or compositions at level $i+1$ serves as component at level i. The multilevel approach was proposed in [13].

F. Martínez-Álvarez et al. (Eds.): HAIS 2016, LNAI 9648, pp. 463–474, 2016.
DOI: 10.1007/978-3-319-32034-2_39

The motivation to propose this approach is two fold. Firstly, this paper presents a systematic approach to describe component-based systems having a *multilevel structure*. This offers a means to abstract details not needed in a certain level. Secondly, the proposed approach takes also into consideration *non-functional requirements*: functionality metric and cost of a component, i.e. cost of the solution. The final obtained system is assessed by measuring coupling and cohesion that are internal attributes influencing reusability and maintenability external quality attributes.

The paper is organized as follows: Sect. 2 presents a short introduction on components and metrics, i.e. the used component model and the metrics used for selection. Section 3 presents the proposed approach, formulating the problem, presenting the metrics used for selection criteria and the fuzzy-based algorithm to construct the final system. Section 4 presents a case study. We have compared the obtained solution by our algorithm with the solution obtained by an evolutionary algorithm. Finally, Sect. 5 summarizes the contributions of this work and outlines directions for further research.

2 Background: Component Model and Metrics

To provide a discussion context for configuration we first describe the assumptions about components and their compositions. The metrics used in our fuzzy-based selection algorithm are also described in this section.

2.1 Component Model

A component is an independent software package that provides functionality via well defined interfaces. Assembling components [5] is called composition. Composition involves putting components together and connecting provided functionality to required functionality.

A graphical representation of our view of components is given in Fig. 1. See details about component specification elements in [13].

There are two type of components: *simple component* (specified by inports, outports and a function) and *compound component* - is a group of connected components in which the output of a component is used as input by another component from the assembly. In Fig. 1 we have designed the compound component by fill in the box. See [4,13] for more details.

2.2 Metrics for Selection Criteria and Architecture Evaluation

Selecting the best component alternative among components that sometimes provide similar functionalities needs to appeal to knowledge on quality attributes such as, for example, maintainability, reusability and extensibility. Therefore evaluation of these components is of utmost importance and software metrics are a means to quantify those attributes considered important for the system that will be built.

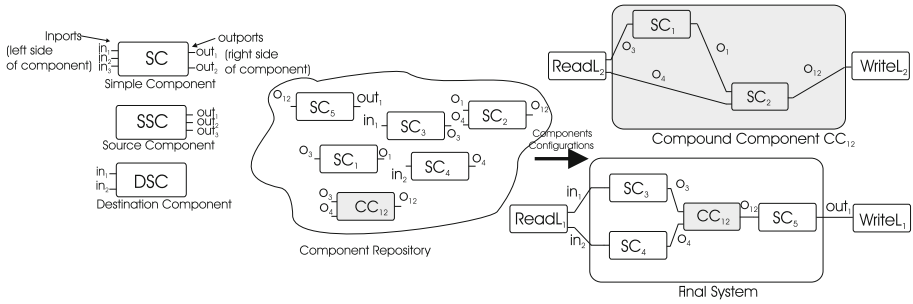

Fig. 1. Components graphical representation and components assembly construction reasoning.

Component Related Metrics. As it has already been mentioned above, we need to evaluate the available set of components in order select the best candidate. This evaluation is based on some criteria (quality attributes) that are important for the final system. These attributes are quantified using the metrics stated in the following:

- The *cost (C)* of a component metric is defined as the overall cost of acquisition and adaptation of that component.
- The *Functionality* metric (F) is defined as the ratio between the number of required services of the system that are provided by the component and the number of required services of the system.

The influence of these metrics values over the quality attributes previously mentioned has been discussed in [12].

Architecture Related Metrics. In our research, the system is built using a multilevel approach (see paper [13] for details). In such an hierarchical system, every compound component is decomposed into a subsystem that will be part of the next higher level. The first level solution defines the modules of the system. In this approach, a module can be viewed as a set of one or more components whose composition forms a subsystem of the final system. Each module has to provide a set of functionalities which define the module's interface [1].

Taking into account the interactions between components, the proposed decomposition approach brings us additional information that can be used to assess coupling and cohesion. It is well known that these internal attributes have a great impact [6] on external quality attributes like reusability, understandability and modularity which are of great importance for a component-based system [1,9,10]. A quantitative approach to measure and assess the solutions of software modularity based on the criteria of minimal coupling and maximal cohesion was proposed in [1]. Coupling measures the interactions among software modules while cohesion measured interactions among the software components which are within a software module. A good software system should possess software modules with high cohesion and low coupling. A highly cohesive module exhibits high

reusability and loosely coupled systems enable easy maintenance of a software system [11].

Therefore, the first level of decomposition solution minimizes coupling, as a direct consequence of minimizing the number of components for solution, and the decomposition of each composed component at the next levels maximizes the modules cohesion, identifying the interactions between components of the same module. To define coupling and cohesion metrics for software modules, we take into account the interactions between components. Therefore, coupling for a software module refers to the external interactions of components regarding with that module, whereas cohesion refers to internal interactions of components for that module. In this research, we use the Intramodular Coupling Density metric (ICD) proposed by Abreu and Goulao [1]. They proposed a quantitative approach for measuring coupling and cohesion at the module level. The ICD metric was also used by [8], adapted for component-based system approach.

The definition of **ICD metric** [8], used in our approach is given as follows:

$$ICD_j = \frac{(CI_{IN})_j + 1}{(CI_{IN})_j + (CI_{OUT})_j} \tag{1}$$

ICD_j is the intra-modular coupling density for the jth module; $(CI_{IN})_j$ is the number of component interactions within the jth module; and $(CI_{OUT})_j$ is the number of component interactions between the jth module and other modules.

As a remark, in order to satisfy the principle of "maximize cohesion within modules while minimizing the coupling between them", the value of ICD metric has to be as near as possible to the value 1.

3 Proposed Algorithm Description

3.1 Component Selection Problem Formulation

An informal specification of our aim is described next. It is needed to construct a final system specified by input data (that is given) and output data (what is required to compute). We can see the final system as a compound component and thus the input data becomes the required interfaces of the component and the output data becomes the provided interfaces and in this context we have the required interfaces as provided data and we need to configure the internal structure of the final compound component by offering the provided interfaces. In Fig. 2 all the above discussion is graphically represented.

A formal definition of the problem (seen as a compound component) [14] is as follows. Consider SR the set of final system requirements (the provided functionalities of the final compound component) as $SR = \{r_1, r_2, ..., r_n\}$ and SC the set of components (repository) available for selection as $SC = \{c_1, c_2, ..., c_m\}$. Each component c_i can satisfy a subset of the requirements from SR (the provided functionalities) denoted $SP_{c_i} = \{p_{i_1}, p_{i_2}, ..., p_{i_k}\}$ and has a set of requirements denoted $SR_{c_i} = \{r_{i_1}, r_{i_2}, ..., r_{i_h}\}$. The goal is to find a set of components Sol in such a way that every requirement r_j $(j = \overline{1,n})$ from the set SR can be

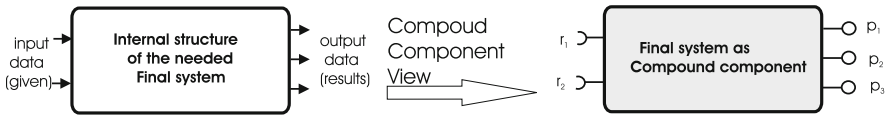

Fig. 2. Graphically representation of the problem formulation.

assigned a component c_i from Sol where r_j is in SP_{c_i} $(i = \overline{1,m})$, while minimizing/maximizing various criteria.

3.2 Metrics for Component Classification

In the following, each component c_i from SC is described by a 2-dimensional vector, $c_i = (F, C)$. Next, our goal is to group similar components regarding the defined attributes. To obtain this, we use a clustering approach [17]. The objects to be clustered are components from our repository and the characteristics of these objects are the corresponding metrics values. The next problem is the selection of one component out of a set of possibilities. After obtaining the final system we use ICD metric to measure the "degree" of reusability and maintenance of it.

3.3 Algorithm Description

In this section we propose a new algorithm for the Component Selection Problem defined in Sect. 3.1. Our algorithm is based on selected metrics and fuzzy clustering approach described before. The objects to be clustered are components, each component being identified by a vector of metrics values. Our focus is to group similar objects in order to select the best candidate. The fuzzy clustering algorithm (Fuzzy n-means algorithm) used to determine the fuzzy partition of the set of components is described in [3]. Taking these into account, we next give the proposed algorithm: SR-set of requirements and SC-set of components and their metrics values represent input, and Sol- obtained solution represents output.

```
program MF-CS (SR, SC, Sol)
 begin
     while (SR is not empty)
       CC := SelectCandidateComponents(SR,SC);
       ComputeMetrics(CC, SR);
       FuzzyPartitionDet(CC, A, B);
       c := SelectBestComponent(CC, A, B);
       AddToSol(Sol,c);
       UpdateSR(SR, c, CC);
end.
```

The *SelectCandidateComponents* subalgorithm selects the components that satisfy at least one of the requirements from the SR, i.e. provides the components (the CC set) that can offer functionalities from SR.

The *ComputeMetrics* subalgorithms computes the metrics values for the Functionality and Cost quality criteria.

The $FuzzyPartitionDet(CC, A, B)$ subalgorithm computes a fuzzy partition of the set CC. The fuzzy sets obtained after the first splitting are A, B, i.e. the candidate components set is partitioned in two sets, A, i.e. B considering Functionality and Cost metrics values.

The *SelectBestComponent* subalgorithm is described next. From the set of possible candidates we have to select one of them. There are two different cases that may appear: all these components belong to the same cluster, or some of them are in the first cluster and the others in the second one. For the first case, we select the component with the maximum membership degree for that class. Regarding the second case we proceed in the following way: the best candidate from each cluster is identified, and then some criteria (first Functionality, then Cost) are considered to choose one of them. For this reason, the component set is split in two clusters. In future research we will apply a divisive hierarchical algorithm and the initial partition will be further split.

The *AddToSol* algorithm adds the selected component to the final solution.

The *UpdateSR* algorithm updates the set of SR by the dependencies appeared by selecting component c and removes the requirements that were provided by selecting component c.

4 Case Study

In order to validate our approach we have used the following case study for building a *Reservation System* is developed. The system allows booking several types of items (hotel, car, ... etc.), by different types of customers, having thus different types of offers. Four modules that define the business logic of this system are identified: *Offer Module* (provides transactions on making a reservation and getting a notification), *LoyaltyPrg Module* (responsible for the loyalty program for old clients), *ReservationType Module* (managing different types of booking offers) and *Customer Module* (provides information about customers). Two of the four modules mentioned above, *LoyaltyPrg Module* and *Offer Module*, are described at level 1 as compound components which are further decomposed at next levels whereas, the modules *ReservationType* and *Customer Module* are simple components and remain unchanged over modules decomposition. The components and the structure of (one) solution may be found at [16].

4.1 Component Selection Problem Formulation

Having specified two input data (customerData, calendarData) and two output data (doneReservation, requestConfirmation) needed to be computed, and having a set of 126 available components, the goal is to find a subset of the given

components such that all the requirements are satisfied considering the functionality and cost criteria previously specified. The set of requirements $SR = \{r_3, r_4\}$ (view as provided interfaces $\{p_3, p_4\}$, see the discussion in Sect. 3.1) and the set of components $SC = \{c_0, c_1, c_2, c_3, c_4, c_5, c_6, ..., c_{126}\}$ are given. The final system has as input data (transformed in required interfaces) the set $\{r_1, r_2\}$.

Remark. The component repository may be found at [16]. There are many components that may provide the same functionality with different requirements interfaces. The components from the repository system have been numbered for better management and utilization of the algorithm.

4.2 Solution Obtained by the Proposed Algorithm

In the first step of the algorithm the set of requirements needed to be satisfied is: $\{r_3, r_4\}$. Considering the remark on Sect. 3.1, these requirements are "transformed" in provided interfaces, thus we "search" components that contain these provided interfaces.

Remark. The criteria that help us decide which of the two components should be chosen to be part of the final solution are based on two metrics values that quantify: functionality and cost. When the functionality criterion has the same value for both components, the cost criterion is considered. If all the criteria are equal one of the components is randomly selected.

Construction of Level 1. Applying the [3] algorithm we obtained the results in Table 1. The components that offer these requirements are split in the following clusters (representative components for a cluster are written in bold): A contains $\{c_{12},\ c_{16},\ c_3\}$ with the most representative being c_{12} and the cluster B contains $\{c_{25},\ c_{18},\ c_4,\ c_{24},\ c_{10},\ c_{13}\}$ with the most representative being c_{25}. To decide which of c_{12} and c_{25} components should be chosen, we consider the metrics-based criteria selection mention above. In this case, the c_{12} component is chosen due to the functionality criterion.

Table 1. The final partition (representative components for a cluster are written in bold) for the set of 9 components (level 1)

Class	c_{12}	c_{16}	c_3	c_4	c_{25}	c_{10}	c_{13}	c_{18}	c_{24}
A	**0.99**	**0.99**	**0.99**	0.09	0.02	0.03	0.16	0.23	0.50
B	0.01	0.01	0.01	**0.91**	**0.98**	**0.97**	**0.84**	**0.77**	**0.50**

The new set of remaining requirements is $\{r_5,\ r_7,\ r_8,\ r_{11}\}$, i.e. the provided interfaces $\{p_5,\ p_7,\ p_8,\ p_{11}\}$. The components that offer these requirements are again grouped in two clusters. Based on the same rules, between the c_6 and c_2 components, the component c_2 (that provides $\{p_7,\ p_8\}$ was chosen. A new partition is created for the remained set of provided interfaces, and between c_6 and

c_{21} components, the first one is selected (that provides $\{p_5\}$). From components $\{c_{21},\ c_{15},\ c_{22}\}$ that have the same Functionality metric value and the same Cost, one is chosen randomly, i.e. c_{21}.

The obtained solution (constructing first level) contains the components: $\{cc_{12}, cc_2, c_6, c_{21}\}$ and has the cost 153. Components c_2 and c_{12} are compound components.

Construction of Level 2. Until now we have constructed the first level of the system. The solution from the first level contains two compound components, thus we need to apply the same algorithm for each compound component.

The requirements needed to be satisfy for component C_2 are: $\{r_7, r_8\}$. The components that offer these requirements are split in the following clusters: A contains $\{c_{27},\ c_{54},\ c_{35},\ c_{53},\ c_{29},\ c_{49},\ c_{38},\ c_{44},\ c_{55}\}$ with the most representative being c_{27} and the cluster B contains $\{c_{43},\ c_{31},\ c_{37}\}$ with the most representative being c_{43}. In Table 2 the degrees are presented (representative components for a cluster are written in bold). To decide which of c_{27} and c_{43} components should be chosen, we consider the metrics-based criteria selection mention above. In this case, the c_{27} component is chosen due to the cost criterion.

Table 2. The final partition (representative components for a cluster are written in bold) for the set of 12 components (level 2)

Class	c_{38}	c_{29}	c_{49}	c_{37}	c_{43}	c_{44}	c_{27}	c_{54}	c_{35}	c_{31}	c_{53}	c_{55}
A	**0.98**	**0.98**	**0.98**	0.10	0.01	**0.90**	**0.99**	**0.99**	**0.99**	0.03	**0.99**	**0.79**
B	0.02	0.02	0.02	**0.90**	**0.99**	0.10	0.01	0.01	0.01	**0.97**	0.01	0.21

After choosing the c_{27} component, the only requirement needs to be satisfied is r_7. All the components that offer this functionality are again grouped in one cluster, thus the chosen component is c_{38}.

The new set of remaining requirements is $\{r_{37},\ r_{34}\}$, i.e. the provided interfaces $\{p_{37},\ p_{34}\}$. The components that offer these requirements are again grouped in two clusters. Based on the same rules, between the c_{33} and c_{26} components, the component c_{26} was chosen. After choosing the c_{26} component, the only requirement needs to be satisfied is r_{37}. All the components that offer this functionality are grouped in the same cluster, thus the chosen component is c_{33}.

The new set of remaining requirements is $\{r_6,\ r_{35}\}$, i.e. the provided interfaces $\{p_6,\ p_{35}\}$. The provided interface p_6 is provided by the final system, i.e. is satisfied. For the p_{35} the components have the same Functionality and Cost, thus one is chosen randomly, i.e. c_{50}.

The new set of remaining requirements is $\{r_{10},\ r_{36}\}$, i.e. the provided interfaces $\{p_{10},\ p_{36}\}$. The provided interface p_{10} is provided by the final system, i.e. is satisfied. For the p_{36} the components have the same Functionality and Cost, thus one is chosen randomly, i.e. c_{50} that also was previously selected.

The obtained solution (constructing component c_2) contains the components: $\{c_{27}, c_{38}, c_{26}, c_{33},\ c_{50}\}$ and has the cost 52.

The requirements needed to be satisfy for component C_{12} are: $\{r_3, r_4\}$. The components that offer these requirements are split in the following clusters: A contains $\{c_{59},\ c_{70},\ c_{74}\}$ with the most representative being c_{59} and the cluster B contains $\{c_{57},\ c_{60},\ c_{69},\ c_{75}\}$ with the most representative being c_{57}. To decide which of c_{59} and c_{57} components should be chosen, we consider the metrics-based criteria selection mention above. In this case, the c_{57} component is chosen due to the functionality criterion.

The new set of remaining requirements is $\{r_5,\ r_{54}\}$, i.e. the provided interfaces $\{p_5,\ p_{54}\}$. The provided interface p_5 is provided by the final system, i.e. is satisfied. For the p_{54} the components have the same Functionality and Cost, thus one is chosen randomly, i.e. c_{68} that also was previously selected.

The new set of remaining requirements is $\{r_{11},\ r_{55}\}$, i.e. the provided interfaces $\{p_{11},\ p_{55}\}$. The provided interface p_{11} is provided by the final system, i.e. is satisfied. For the p_{55} the components have the same Functionality and Cost, thus one is chosen randomly, i.e. c_{63} that also was previously selected.

The obtained solution (constructing component c_{12}) contains the components: $\{c_{57}, c_{68}, c_{63}\}$ and has the cost 56. Components c_{26}, c_{33} and c_{57} are compound components.

Construction of Level 3. Until now we have constructed the second level of the system. The solution from the second level contains two compound components, thus we need to apply the same algorithm for each compound component.

The requirement needed to be satisfy for component C_{26} is: $\{r_{34}\}$. The components that offer this requirement are split in the following clusters: A contains $\{c_{81}\}$ with the most representative being c_{81} and the cluster B contains $\{c_{77},\ c_{80},\ c_{84},\ c_{78}\}$ with the most representative being c_{77}. To decide which of c_{81} and c_{77} components should be chosen, we consider the metrics-based criteria selection mention above. In this case, the c_{77} component is chosen due to the cost criterion.

The new set of requirements is $\{r_{65}\}$, i.e. the provided interfaces $\{p_{65}\}$. The components have the same Functionality and Cost, thus one is chosen randomly, i.e. c_{83}.

The obtained solution (constructing component c_{26}) contains the components: $\{c_{77}, c_{83}\}$ and has the cost 15. There no other compound components.

The requirement needed to be satisfy for component C_{33} is: $\{r_{37}\}$. The components that offer this requirement are split in the following clusters: A contains $\{c_{88},\ c_{97},\ c_{90}\}$ with the most representative being c_{88} and the cluster B contains $\{c_{89},\ c_{95}\}$ with the most representative being c_{89}. To decide which of c_{88} and c_{89} components should be chosen, we consider the metrics-based criteria selection mention above. In this case, the c_{89} component is chosen due to the cost criterion.

The new set of requirements is $\{r_{72},\ r_{73},\ r_{74}\}$, i.e. the provided interfaces $\{p_{72},\ p_{73},\ p_{74}\}$. The components that offer these requirements are split in the following clusters: A contains $\{c_{101}\}$ and the cluster B contains $\{c_{94}\}$. To decide which of c_{101} and c_{94} components should be chosen, we consider the metrics-based criteria selection mention above. In this case, the c_{101} component is chosen

due to the functionality criterion. The remained set of requirements are provided by the component c_{94}.

The obtained solution (constructing component c_{33}) contains the components: $\{c_{89}, c_{101}, c_{94}\}$ and has the cost 26. There no other compound components.

The requirements needed to be satisfy for component C_{57} are: $\{r_3,\ r_4\}$. The components that offer this requirement are split in the following clusters: A contains $\{c_{104},\ c_{115},\ c_{107},\ c_{109}\}$ with the most representative being c_{104} and the cluster B contains $\{c_{123},\ c_{108},\ c_{113},\ c_{126}\}$ with the most representative being c_{123}. To decide which of c_{104} and c_{123} components should be chosen, we consider the metrics-based criteria selection mention above. In this case, the c_{104} component is chosen due to the functionality criterion.

The new set of requirements is $\{r_{78},\ r_{79}\}$, i.e. the provided interfaces $\{p_{78},\ p_{79}\}$. The components that offer these requirements are split in the following clusters: A contains $\{c_{114},\ c_{118},\ c_{111}\}$ and the cluster B contains $\{c_{110},\ c_{105},\ c_{116}\}$. To decide which of c_{114} and c_{110} components should be chosen, we consider the metrics-based criteria selection mention above. In this case, the c_{110} component is chosen due to the cost criterion.

The new set of requirements is $\{r_{79}\}$, i.e. the provided interfaces $\{p_{79}\}$. The components have the same Functionality and Cost, thus one is chosen randomly, i.e. c_{114}.

The obtained solution (constructing component c_{57}) contains the components: $\{c_{104}, c_{110}, c_{114}\}$ and has the cost 33. There no other compound components.

4.3 Comparison: Evolutionary Algorithm Vs. Fuzzy-Based Selection

The problem of selecting components from a set of available components was also discussed in several papers using various approaches. We compare our approach with an evolutionary-based approach.

An approach for the same selection problem that uses principles of evolutionary computation and multiobjective optimization was proposed in [15]. The problem is formulated as a multiple objective optimization problem having two objectives: the total cost of the components used and the number of components used. Both objectives are to be minimized.

In order to compare our approach with those mention before, we describe in Table 3 the obtained solution for the evolutionary algorithm and in Table 4 the solution obtained for the fuzzy-based approach.

Result Analysis: The solutions obtained with the evolutionary algorithm (EA) approach and with the fuzzy-based approach are compared from the cost and ICD (internal metric value that influence the external quality attributes reusability and maintenability) perspectives: EA approach obtained the solution with cost 153 and ICD=0.5313 and the Fuzzy-based approach the solution with cost 159 and ICD=0.5376.

Thus, we select a solution based on one criteria (that is important to us, either [reusability&maintainability] or cost) on the expense of the other criteria. There is a trade-off between cost and [reusability&maintainability].

Table 3. ICD metric values for best solution with cost = 153 using evolutionary algorithm.

CI_{IN}	m_1	m_2	m_3	m_4
L_1	0	0	0	0
L_2	0	4	0	2
L_3	0	7	0	7
CI_{OUT}	2	5	3	4
ICD_m	0.5000	0.6667	0.3333	0.6250
ICD	0.5313			

Table 4. ICD metric values for the obtained solution with cost = 159 using the fuzzy-based selection algorithm.

CI_{IN}	m_1	m_2	m_3	m_4
L_1	0	0	0	0
L_2	0	4	0	2
L_3	0	8	0	7
CI_{OUT}	2	5	3	4
ICD_m	0.5000	0.6923	0.3333	0.6250
ICD	0.5376			

5 Conclusions and Future Work

A new algorithm based on metrics and fuzzy clustering analysis that addresses the problem of component selection in the multilevel system structure was proposed in this paper. We use both functional and non-functional criteria when selecting the best candidate component for each selection step.

We evaluate our approach using a case study. We have compared our fuzzy-based approach with an evolutionary based approach: we have obtained better solutions from different perspective: considering either cost or reusability and maintainability quality attributes.

We will focus our future work on three main fronts: to apply this approach for more case studies; to apply other fuzzy clustering algorithms in order to obtain the needed classification, and to select more relevant metrics.

References

1. e Abreu, F.B., Goulao, M.: Coupling and cohesion as modularization drivers: are we being over-persuaded? In: Conference on Software Maintenance and Reengineering, pp. 47–57 (2001)
2. Becker, C., Rauber, A.: Improving component selection and monitoring with controlled experimentation and automated measurements. Inf. Softw. Technol. **52**(6), 641–655 (2010)
3. Bezdek, J.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York (1981)
4. Cox, P., Song, B.: A formal model for component-based software. In: Symposium on Visual/Multimedia Approaches to Programming and Software Engineering, pp. 304–311 (2001)
5. Crnkovic, I.: Building Reliable Component-Based Software Systems. Artech House Inc., Norwood (2002)
6. Fenton, N.E.: Software Metrics: A Rigorous Approach. Chapman and Hall, London (1991)
7. Iribarne, L., Troya, J., Vallecillo, A.: Selecting software components with multiple interfaces. In: The EUROMICRO Conference Component-Based Software Engineering, pp. 26–32 (2002)

8. Kwong, C., Mu, L., Tang, J., Luo, X.: Optimization of software components selection for component-based software system development. Comput. Ind. Eng. **58**(1), 618–624 (2010)
9. Bertrand, M.: Software Engineering. Addison-Wesley, Longman Publishing Co., Inc., Boston (2001)
10. Parsa, S., Bushehrian, O.: A framework to investigate and evaluate genetic clustering algorithms for automatic modularization of software systems. In: International Conference on Computational Science, pp. 699–702 (2004)
11. Seker, R., van der Merwe, A.J., Kotze, P., Tanik, M.M., Paul, R.: Assessment of coupling and cohesion for component based software by using Shannon languages. J. Integr. Des. Process Sci. **8**, 33–43 (2004)
12. Vescan, A.: A metrics-based evolutionary approach for the component selection problem. In: The International Conference on Computer Modelling and Simulation, pp. 83–88 (2009)
13. Vescan, A., Grosan, C.: Evolutionary multiobjective approach for multilevel component composition. Studia Univ. Babes-Bolyai, Informatica **LV**(4), 18–32 (2010)
14. Vescan, A., Grosan, C., Yang, S.: A hybrid evolutionary multiobjective approach for the dynamic component selection problem. In: The International Conference on Hybrid Intelligent Systems, pp. 714–721 (2011)
15. Vescan, A., Serban, C.: Multilevel component selection optimisation and metrics-based architecture evaluation. Soft Comput. J. 0, 1–1 (2015, submitted)
16. Vescan, A., Serban, C.: Details on case study for the multilevel componentselection optimisation approach (2016). http://www.cs.ubbcluj.ro/~avescan/?q=node/95
17. Zadeh, L.: Fuzzy sets. Inf. Control **8**, 338–353 (1965)