Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-018-1100-0

ORIGINAL RESEARCH

Self-organized design of virtual reality simulator for identification and optimization of healthcare software components

Amit Kumar Srivastava[1] · **Shishir Kumar**[1] · **Masoumeh Zareapoor**[2]

Received: 15 March 2018 / Accepted: 27 September 2018

Abstract
As in the current trend, the virtual reality-based application is very popular in the medical healthcare system to generate a realistic virtual 3-D simulation environment that users can interact with specialized devices. The increasing demand for advancement in the requirement of the virtual environment of healthcare policies as well as the systems needs changes in the simulation environment. In reference to such requirement, the software industry needs improvement in the development process which reduces the effect of software cost, complexity and resource planning. From last few years, optimization in development of simulation environment's cost-benefit aspects is also the challenging area. In view of such issues, there are several guided (supervised) and unguided (unsupervised) algorithms are using evolutionary approaches and nature-inspired self-organized swarm intelligence approaches have been developed by the researchers for virtual real-time healthcare search based software system. However, there is still a gap in the development of such a simulation environment for identification of the software component. This paper proposes a self-organizing component identification technique using medoid based ant colony clustering algorithms. The proposed algorithm has been compared to classical centroid-based clustering (K-means,CRUD, and FCA) and evolutionary approach(genetic algorithm) on the case study of the virtual real-time healthcare system. For the accuracy and precision of the approach two already studied cases have also been used.

Keywords Software logical component · Clustering · Component-based software development · Design pattern

1 Introduction

Day to day changes in healthcare policies is having a measurable impact on the education and learning of healthcare professionals. According to Kaufmann (2001). In the last five decade the major change in medicine and nowadays it is recognized that double modification in a decade with update medical procedures. If we consider only the half-life of medical information which is to less where the 30 and 40 years average physician practices age and the nurse service respectively. Within this circumstance, the challenging task is to continue the education of healthcare professionals. Definitely demands of the changes are arises in every procedure of teaching process, performance evaluation strategies etc. All these re-structuring can be fulfilled easily by the virtual reality (VR) that is like a computer can generate a three-dimensional (3D) graphical environment from categorical information data. However, VR helps in several domains (like healthcare, Education and Military Services etc.) but it demands day to day updation. These 3D simulators also need updating because always significant changes are going on. This is a challenging task for the software industry to emerge all these changes in ethical and cost benefited aspects of IT applications of healthcare. Always needs changes in the repository of a logical component of the 3D simulator VR system.

The VR system consists of two major components i.e. Hardware and Software. In this paper, we are focused on the software component. The application of virtual reality software system has been suggested by the researchers in various dimension like the visual aid system for reducing

✉ Amit Kumar Srivastava
amitsri1983@gmail.com

Shishir Kumar
dr.shishir@yahoo.com

Masoumeh Zareapoor
mzarea222@gmail.com

[1] Jaypee University of Engineering and Technology, Guna, India

[2] Shanghai Jiao Tong University, Shanghai, China

Published online: 03 November 2018

the risk of neck injuries for computer users (Liu et al. 2018), agriculture product price prediction model (Yu et al. 2018) etc. From off the self, the identification of the required software component is an NP-complete problem (Cai et al. 2011). As describe by the Birkmeier and Overhage (2009) three categories of components are Business-oriented components (are concerned with business services and cognition of business process), Architectural or logical component (focused on logical characteristics that provide the software system needed to meet the business requirement) and third one Technical components (focus on technical requirement developed by use case for software services implementation and deployment aspects). The Architectural or logical component can be an attribute of objects that define the software system role function. After the prototype design, it will describe the target system.

During the development life cycle of VR system software architecture, a software designer is responsible to decompose a system using relevant software components. However, because of lack of perfection in IT regulation and experience about protocols of services it is an extremely difficult and error-prone task to identify and decompose logical component without any supporting tool (Birkmeier and Overhage 2009). In respect of overcome this difficulty, several methods have been suggested like defect density approach (Davis et al. 2015; Verma and Kumar 2014; Mandhan et al. 2015; Verma and Kumar 2017a, b, 2016), intelligent computational technique (Kumar et al. 2018), emotion extraction (Jain et al. 2017), taxonomic position of software vulnerability (Srivastava and Kumar 2018a) and to identify logical components (Shahmohammadi et al. 2010; Hasheminejad and Jalili 2013).

By search-based software engineering the identification of component maps as an optimization problem that can be handled by the meta-heuristic. According to the Harman et al. (2012b) the SBSE can map the various optimization problem of software engineering as the logical design of software program architecture that thinks by software architect as follows:

- How many minimum number of test classes are required to test the full functionality of the program?
- What is the optimum way to design the logical view which gives high cohesion and less coupling?
- How to plan the software requirement for balance the software cost with full customer satisfaction?
- What is the optimum resource to develop the software and how to schedule it?
- What is the best sequence of re-factoring steps to apply in system?

According to Harman and Jones (2001), the objective of search based software engineering starts with two important objectives are as (1) the problem formulation (2) search function (fitness function). On the basis of these objectives search based software engineer develop the optimization algorithm which gives the optimum result. In search optimization problems, where we do not have any guiding information about the selection of next state value then we choose the random search approach for optimum results. Often he fails to find global optimum results. For such situation, we take some guiding information in form of fitness function which provides the heuristic information about the search space.

Genetic Algorithm comes in the role for global search. It is also known as the variant of stochastic beam search. In the initial iteration, the Genetic algorithm starts with a set of N ($N = 1, 2, 3, 4, \ldots$) randomly generated states, called the population. Each individual chromosomes is represented as a string of 0s and 1s and sometimes also in floating point values. Each state is evaluated by the fitness function or evaluation function. The first iteration started with the randomly selected chromosomes. In some specific cases, the population may also be prepared by the selected individual of domain-specific information of the problem. Detail discussion about the evolutionary computation (Genetic Algorithm) for identification of logical component is discussed in related work in the next section of this paper.

In proposed work, the one important focus area is to identify the logical components for virtual real-time software system design. Because The Logical or architectural component can be an attribute of objects that define the virtual environment role function. After the prototype design, it will describe the target simulator system (Future target simulator for health care etc.). So it should reconfigure the logical design of the prototype system to balance the software complexity and cost etc. For assurance of quality and reliability of the target system we propose the bio-inspired swarm intelligence techniques for computation of software cohesion parameters of software design principles. For integration of optimization techniques, we used the emerging field search-based software engineering. As noticed in literature it seems to be a search based software engineering (SBSE) is an emerging field for the distributed system or virtual reality-based system development (Harman et al. 2012a). That is the objective of this paper which is given in section Problem Formulation. This paper also concentrates on dynamic reconfiguration of a software architecture using reuse software component using ant colony based clustering techniques.

The use case diagram and prototype class diagram is the prerequisite input of the proposed approach. In object-oriented modeling, the use case diagram will helpful to express the software requirement and it is a key point to prepare software design architecture (Shahmohammadi et al. 2010). The class diagram represents the static structure of the use

case diagram of any system. The class diagram explains the attributes and various operations and also the relationship in between the classes.

The remaining section of this paper is arranged as follows Sect. 2 presents problem formulation. Section 3 presents goal formulation. Section 4 describes the related work on the virtual reality scope in various field. Section 5 theoretical analysis of object-oriented software design measurements. Section 6: explain the medoid based logical component identification and optimization ant colony algorithm. Section 7: describe the measuring problem-solving performance of proposed algorithm. Section 8: perform the result evaluation and finally Sect. 9: presents the conclusions and future scope of work.

2 Problem formulation

A logical component is a group of classes that provide a common solution to a virtual system for health care problem. In Rational Unified Process (Kruchten 2004) several problem model is defined for the representation of requirement of a common software problem. According to the recommendation of RUP methodology, the class diagram modeling is suitable to define the static structure design of problem requirement which is common standard to communicate the end user requirement to other team members. The class diagram model consists the attributes, operation, and the relationship between the classes which defined what the system does and when the operation has been performed and which one from outside of the system that interacts with the system respectively. The analysis class diagram consists of the description of the all three classes: boundary (interface), control an entity class. The current trend to a design of class diagram is very simple by various tools like IBM rational rose, Visual paradigm 12.2 etc. There are various issues which need extra attention from a software developer. The main important issues are as follows:

- How to maintain the property of principles of software design i.e. maximum cohesiveness and minimum coupling.
- Estimation and evaluation of parameter of principles of software design is an NP-complete problem.
- The composition of the classes maintains the coupling between the object of the classes.
- Design of the reuse component in the composition of classes.
- Optimization in software cost according to the customer satisfaction.

All these issues and many more like them are a challenging task for a software developer. Out of these in this paper we are focusing on these issues as follows:

- To calculate the cohesion and coupling value of a prototype structure of class diagram and recommended the best configuration of class diagram structure with maximum cohesion and minimum coupling matrix.
- To identify the cluster of logical components (similar type of subsystem) on the basis of a design pattern.
- Systems have a dynamic re-configurable capability to rearrange the logical view of the software architecture.

The pictorial representation of the problem is as shown in Fig. 1.

3 Goal formulation

An end-user describes the own requirement in form of a story to a software developer. At the very beginning of the software development lifecycle, the software developer prepares a class diagram according to an end user requirement. The perfection in the design of the class diagram depends on the experience of developers. In this paper for a perfection of design with an optimum value of cohesion and coupling, we propose an approach on medoid based clustering technique which gives the best possible configuration of the component structure of the class diagram. With a prototype of a class diagram, identifies the subsystem of class hierarchies and prepare a cluster that completes the goal of the second problem. In the third step, on the basis of cluster and subsystem to reconfigure the logical view of the software architecture. In during the preparation of a logical view of a structure of the system have the self-organized capability to update the component structure if any optimum possible partitioning threshold achieved. The goal formulation is as given in Fig. 2.

4 Related work

Application of virtual reality (VR) demands have increased in the last few years. This growth of a VR system is just because of subsequent changes in every domain either it is education or medical or defense. This expected demand will continue in future with the innovative development in various domain like computer vision and image processing, control, education, healthcare (stress management etc.) (Bouchard et al. 2010), military (Baumann 1993; Lele 2013) etc. According to Oluleke Bamodu et al. the VR system consists of the two major components: The hardware and Software. The hardware component will be subcategorized into

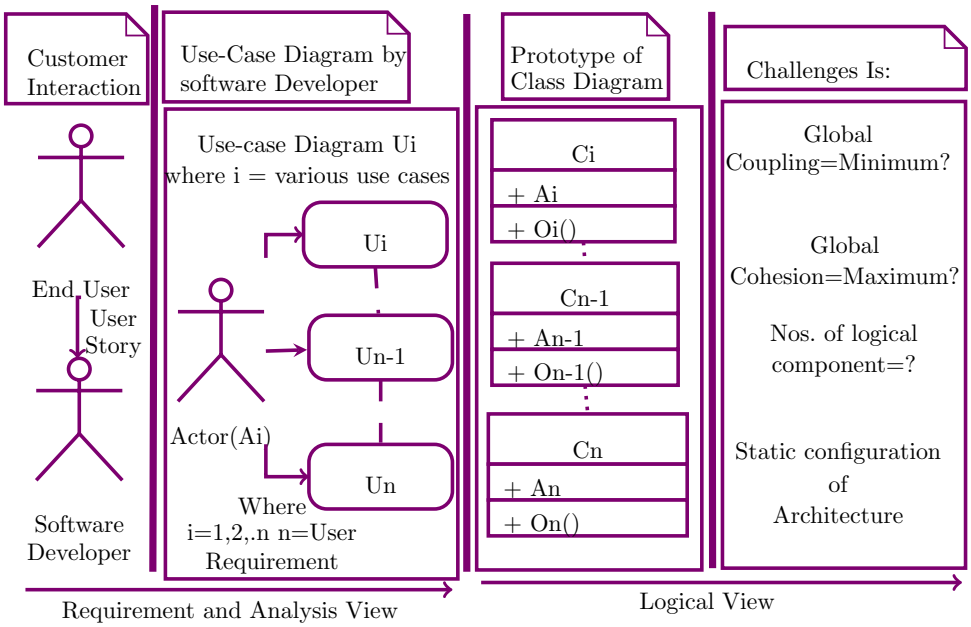

Fig. 1 Problem formulation

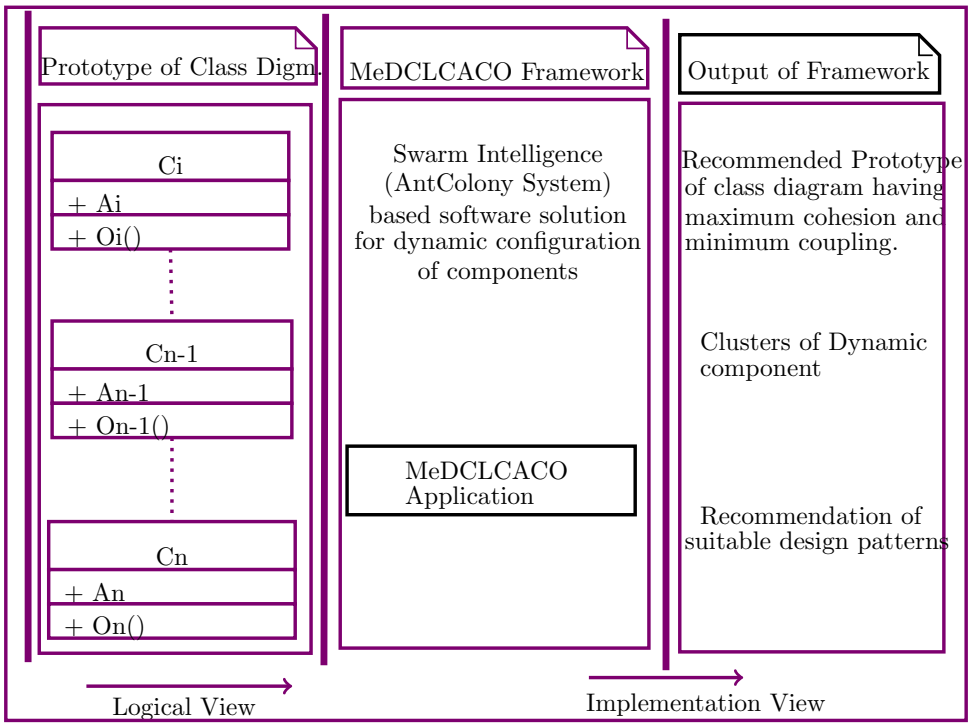

Fig. 2 Goal formulation

the two component i.e. VR engine and I/O devices, while the software component was divided into application software and database (Bamodu and Ye 2013). By the application of virtual maintenance, the industry can focus on cost-benefit aspects in terms of development and training of 3D VR systems. Kouroush et al. perform the review study on the benefit of cost aspects of VR systems (Jenab et al. 2016). In this paper, we have focused on the software component of the VR system. Virtual reality system software comprises of a various software component of features and systematically maintain the virtual environment and the database storage. The identification of such software component of VR tools and software is a challenging task. It is mapped with the new domain of software engineering i.e. search based software engineering (Harman and Clark 2004).

This paper presents an approach which is focused on identification of software component of VR system. The proposed approach is also applicable for any type of application software related to health care or education or stock trading. In the proposed approach as per the demand of data science and optimization in system software development, we used medoid based clustering technique using bio-inspired swarm intelligence. The bio-inspired swarm intelligence rank based ant colony optimization has been used for to group the component configuration.

The ant colony optimization (ACO) (Dorigo and Socha 2006) algorithms are the family of algorithms in which the artificial agents collaborate with each other which leads them to form an intelligent system. It was proposed by Italian scholar Marco Dorigo during his research study for solving the well-known traveling salesman problem. The characteristics and the behavior portrayed by artificial agents is the one that has been observed in nature, such as stimulating behavior (Jenab et al. 2016) within the ant colonies, beehives, flocks of birds and the different group of animals and insects. The kind of stigmergy is found in ant colonies where they use a chemical substance called trail pheromone which is a specific type of pheromone that some ants use for marking paths on the ground, for example, paths from food sources to the nest. The trail pheromone is perceived by neighbor ants and they tend to follow the path where the intensity of the pheromone is more. The ants while making a tour chooses a path which has the high concentration of pheromone trail, after a while, all ants start following the same path and hence converge to shorter tour.

Clustering techniques are the group of data into the cluster according to the medoid of a dataset which is the application of data mining and machine learning. The cluster will prepare on the basis of cost function either maximize or minimize the cost function value allocated the data instance accordingly. There are various application areas that have dealt with clustering algorithm like image processing, computer vision, deep learning, healthcare system etc.In literature there are several bio-inspired techniques which deal on evolutionary algorithms (Menendez et al. 2014). classical clustering techniques for identification of logical system software components of applications domains (Shahmohammadi et al. 2010; Srivastava and Kumar 2018b; Lee et al. 2001; Kim and Chang 2004).These approaches prepare a cluster on the basis of several features like actors and analysis classes using the fitness function factor of software design(high cohesion and low coupling). Hasheminejad and Jalili (2013) suggested an evolutionary computation approach for identification of logical components. SCI-GA has a number of characteristics like do not require a number of components in advance etc. It suggested the component diagram of the software system which is almost similar to the expert's component diagram.

Software logical component identification techniques are also dealt with classical clustering techniques like K-means, Hierarchical, Graph-Based method, and Fuzzy C-Means obtained the good result in form of architecture component design. All these clustering approaches have common weaknesses like manual adjustment of threshold on the basis of an external decision, pre-information for a number of components in advance and not efficient for multiple local minima or maxima landscape due to simple greedy and heuristic approach.

5 Theoretical analysis of VR healthcare system software design measurements

The identification of the logical component at requirement gathering and analysis phase of the software development life cycle is a challenging task. By various unified modeling diagram (use case model, class diagram and collaboration diagram etc.) the software developer can explain the requirement of end-user to the software design team. During this stage, the total reliability or quality depends on the experience of technical skills. In the proposed approach for the validity and reliability or cohesiveness of the design the proposed framework gives the recommendation of the cohesive and less coupled design. For the framework, the use case model is used as input prerequisite for the identification of logical components.

On the basis of sensitivity analysis of Shahmohammadi et al. (2010) use case features i.e. Actor and Entity class has the highest impact on the cohesiveness of the design. On the basis of the use case feature, the metrics of fitness concept (Harman and Clark 2004) is used for the mapping of the problem into the optimization problem. For the framework, a sample metrics is prepared on the basis of two use case parameter (Actor and Entity Class). For the theoretical analysis of fitness function, a sample use case model with the corresponding logical component diagram is assumed. The assumed use case model and logical component diagram are given in Fig. 1a, b. As shown in the figure in this use case model four use cases, four actors, and four entity classes, hence Sample proximity matrix M_{SPM} having 4 rows and 8 column binary matrix.

$$\mathrm{M}_{\mathrm{spm}} = \begin{array}{c} UseCase \\ UC1 \\ UC2 \\ UC3 \\ UC4 \end{array} \begin{array}{c} \begin{array}{cccccccc} A_1 & A_2 & A_3 & A_4 & EC_1 & EC_2 & EC_3 & EC_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

In M_{spm} matrix the Actor $\in A_1, A_2, A_3, A_4$, Entity classes are set of EC_1, EC_2, EC_3, EC_4 and use cases are represented by the set of UC_1, UC_2, UC_3, UC_4.

For improvement in the development process of the virtual reality of health care system software design have the techniques to measure the design concepts: software cohesion and coupling. Coupling defines the interdependence between the two classes and cohesion defines the consistency between the design classes of software design.

According to Chidamber and Kemerer 1994 suggested that two ontological definitions of software coupling and cohesion defined in Sects. 5.1 and 5.2.

5.1 Software cohesion

Cohesion according to the Bunge (1977, 1979) ontology the similarity of the two set of objects is to be the intersection of both sets of objects. This general principle defines the similarity between the two sets. The extension of Bunge's similarity methods to be the intersection of the sets of instance variables that are used by the methods. The instance variable is not the properties of the methods but it is consistent with the notion that methods of an object are intimately connected to its instance variables. The degree of similarity of methods represents the cohesiveness between the object class. Cohesion is shown by the listing 1.

$$LCOM = \begin{cases} \sum_{\forall UC_i \in comp_i} \sum_{\forall UC_j \in comp_j} \left(D(UC_i, UC_j) - S(UC_i, UC_j) \right) & if\ D > S \\ 0 & Else \end{cases} \tag{1}$$

Listing 1 The simple less cohesion class (Java source code)

```
//Less cohesion of class i.e. A.Java
public class A
{
int count=0;
public void display(int a)
{
System.out.println(a);
}

public void Show (count)
{
count+=count;
}
}
```

For the quality assurance, the measurement of software design is required. Clarke et al. (2003) have also suggested the metrics based approach for the mapping of such type of requirement problem to the search based software engineering. Various empirical metrics for object-oriented design had suggested by the researcher for measurement of software cohesion as given in Table 1.

In the above-mentioned Table 2 the four types of cohesion metrics designed for the cohesion computation of the analysis class diagram. In rationally unified process (Kruchten 2004) the end user requirements were expressed to the developer team by the use case model. This process is known as requirement analysis workflow.After that design team is responsible for to produce the high cohesive and less coupling design. The total reliability and quality of the design depend on the experience of the development team. In the proposed framework the dynamic computation of software cohesion is based On the Chidamber and Kemerer (1994) framework. The proposed measure of cohesion for use case model as follows:

Assume a component $Comp_1$ having set of use case UC_1, $UC_2 ... UC_i$, $i = 1, 2, 3, ..., n$ | n ∈ Use Cases defined by the developer to perform the end user requirement with binary features of entities then

where, D is dissimilarity of the pairs of use-cases have not similar features and S represents a similarity between the pairs of use cases have similar features.

For calculation of dissimilarity (D) of use case pair square Euclidean Equation (2) and for similarity (S) of use case pair Pearson and Hearon equation (3) is used.

$$D_{SquareEucleidian}(UC_i, UC_j) = \sqrt{1 - Sim(UC_i, UC_j)} \tag{2}$$

$$sim_{Pearson\&Heron\text{-}II}(UC_i, UC_j) = \cos\left(\frac{\pi \sqrt{bc}}{\sqrt{ad} + \sqrt{bc}} \right) \tag{3}$$

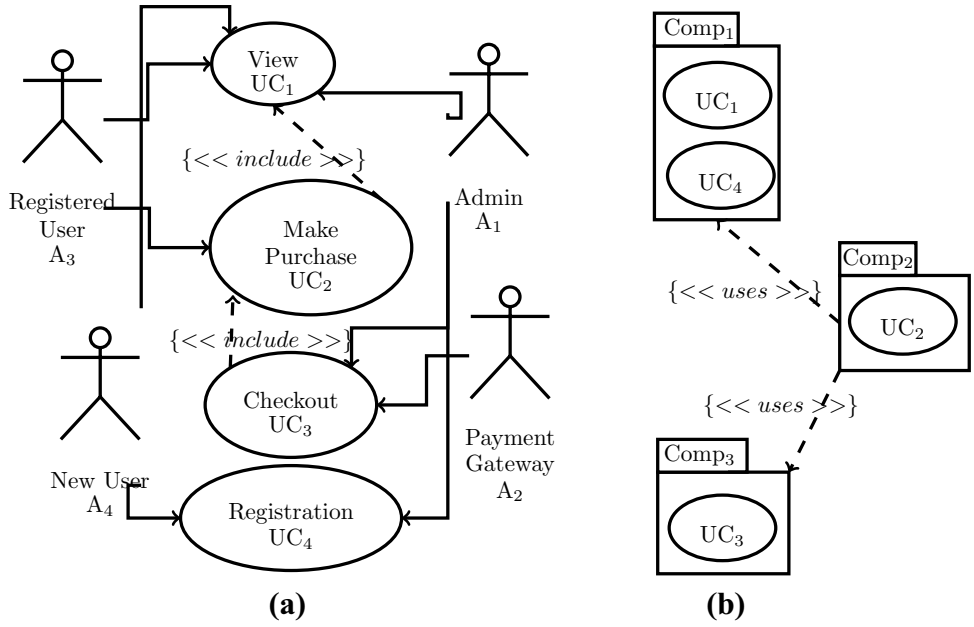

Fig. 3 **a** Sample health care insurance use case model. **b** The logical component of (**a**)

Table 1 Type of cohesion metric for object oriented design

Sr. No.	Design measurement	Metrics	Description	Source
1	Cohesion	Lack of cohesion	It measures the dissimilarity of methods in a class on the basis of variable instance or variables of a method (Aggarwal et al. 2006)	Chidamber and Kemerer (1994)
		Tight class cohesion	The percentage of pairs of public methods of the class which use the common attribute (Aggarwal et al. 2006)	Briand et al. (1998)
		Loose class cohesion	The percentage of pair of public methods of the class which are directly or indirectly connected (Aggarwal et al. 2006)	Briand et al. (1998)
		Information based cohesion	The number of invocations of other methods of the same class, weighted by the number of parameters of the invoked method (Aggarwal et al. 2006)	Lee (1995)

Table 2 The similarity matrix of the use case matrix M_{spm}

	UC_1	UC_2	UC_3	UC_4
UC_1	1	0.41	− 0.99	1
UC_2	0.41	1	− 0.06	
UC_3	− 0.99	− 0.06	1	
UC_4				1

where, a = number of variables present in both the object S_A & S_B value having 1.

b = number of variables present in object (S_A) and absent from object (S_B).

c = number of variables present in object (S_B) and absent from object (S_A)

d = number of variables absent from both the objects.

On the basis of Eq. (3) and Table 3 shows the similarity coefficient between the pair of use cases of matrix M_{spm}.

Similarly, we can calculate all value of the similarity matrix of the use case matrix M_{spm}

Hence the component cohesion is

$$CC_{cmp} = 1 - LCOM \tag{4}$$

Hence the software cohesion is:

$$SoftwareCohesion = \sum_{comp=1}^{N} \frac{(CC_{cmp})}{N} \tag{5}$$

where N represents the total number of component.

5.2 Software coupling

Coupling was defined in ontological terms: two objects are coupled if an only if one object depended on the other object, shown in Listing 1.

Listing 2 The Coupling in between two class (Java source code)

```
//Fist class of coupled relation i.e. A.Java
public class A
{
display(int a)
{
System.out.println(a);
}
}
// Second class of coupled relation i.e. B.Java
public class B
{
A a;
a= new A();
public void Show(){
a.display(10);
}
}
```

According to listing-2, the method of one class object using the methods or instance of another class object make the coupling.

In component-based software system design, the coupling describes the relationship or communications between the components by using the interface services. The coupling metric is describe by the various studies (Chidamber and Kemerer 1994; Briand et al. 1998, 1999). These studies focused on various quality features of components such as reusability and maintainability etc. However, a popular class coupling metric CBO (Coupling between objects) (Briand et al. 1998) is widely adopted by the software designer. CBO class metric frequently used to estimate the quality parameters such as fault-proneness,reusability, and maintainability of classes (Chidamber and Kemerer 1994; Briand et al. 1998, 1999). It also used by various software tools as OOMeter (Alghamdi et al. 2005) etc. JianFeng Cui and Chae 2010 suggested a component coupling rate (CCR). It describes a ratio of the coupled component to the total number of components within the system. According to Cui and Chae (2010), the two components are coupled if there is a uses relationship between them. So:

$$CCR(cmp_c) = \frac{|\ CP(cmp)\ |}{|\ S\ | - 1}, \quad 0 \le CCR(cmp) \le 1 \tag{6}$$

where $\mathrm{CCR}(\mathrm{cmp}_c)$ is the component coupling rate of cmp_c,$\mathrm{CP}(\mathrm{cmp}_c)$ represents the set of component coupled with cmp_c and S is the total number of component that compose the software system. Cui and Chae (2010) consider the uses relationship between the component but in this proposed work we are considering following relationships: *<< usage >><< realization >> << objectflow >><< include >><< interface >><< extend >> << generalization >><< abstraction >><< association >> and << aggregation >>*.

In *<< association >>* stereotypes we consider all category of association relationship like communication path,and directed association etc.For example in Fig. 3 the two component are coupled if there are a relationship in the the use case of components. The CCR value for Comp_1, Comp_2 and Comp_3 are 0, 1 and 1 respectively where $|\ CP(Comp_1)\ | = 0$,$|\ CP(Comp_1)\ | = 1$ and $|\ CP(Comp_1)\ | = 1$.

AlSharif et al. (2004) suggested total software coupling approach Eq. 7 which is also used by SCI-GA approach(Hasheminejad and Jalili 2013) to software coupling in boundary [0,1].

$$SoftwareCoupling = \sqrt{\frac{\sum_{c=1}^{n}(CCR(cmp_c))^2}{No.ofUsecases}} \tag{7}$$

They have used Euclidean norm distance since it gives the comparatively better result to average norm distance.

For the identified component of Fig. 3 the software coupling is equal to 1.

6 Medoid based logical component identification using nature bio-inspired optimization algorithm

One of the widely used nature bio-inspired swarm intelligence techniques is the Ant colony optimization(ACO) algorithm ((Dorigo and Gambardella 1997). It is based on the foraging behavior of ants that collaboratively can identify the smallest path from the nest to a food source. Here ACO has used for the identification of best optimum possible software component. In the proposed work the grouping of the use cases is done by the medoid based clustering techniques. This clustering technique performed by the ACO algorithm.

Table 3 The Online Broker System (OBS) use cases, entity classes and actors

Use case	Use case name	Entity class	Entity class name	Actor	Actor name
UC_1	User registration	ac1	Account	a1	Investor
UC_2	Login	ac2	Client	a2	Administrator
UC_3	Logout	ac3	Portfolio	a3	Exchange service
UC_4	Change password	ac4	Balance	a4	Analyst
UC_5	View account	ac5	Order		
UC_6	Change account info	ac6	Stock		
UC_7	Increase the account balance				
UC_8	Create account				
UC_9	Terminate account				
UC_{10}	View portfolio				
UC_{11}	Set performances and alerts				
UC_{12}	Retrieve stock info				
UC_{13}	Search stock				
UC_{14}	Update portfolio				
UC_{15}	Report				
UC_{16}	Report the highest price				
UC_{17}	Report the stocks to buy and sell				
UC_{18}	Report daily stock market				
UC_{19}	Get short term stock Prediction				
UC_{20}	Get long term stock Prediction				
UC_{21}	System recovery				
UC_{22}	Client reactivation				
UC_{23}	Manage client				
UC_{24}	Deactivate client				
UC_{25}	Send E-mail				
UC_{26}	Place order				
UC_{27}	View statistics				
UC_{28}	Compute benefit				
UC_{29}	Buy stock				
UC_{30}	Sell stock				

ACO algorithm has been used because it performs the global search in the solution space. There is minimal chance to stuck in the local minima and has the probability to give the better solution.

The proposed framework consists of the two type of algorithm. The first algorithm performs the grouping of classes on the basis of distance measure which is known as a cluster of classes of a logical component. and the second algorithm, recommends the best possible combination of the component according to the fitness function for a software architecture. The fitness function consists the software principles of design parameter i.e.software cohesion and coupling.

6.1 Clustering of classes

In recent trends, the clustering techniques are a popular approach for grouping of the similar type of data on the basis of distance measures. In literature, various clustering techniques were used for grouping of the task. The clusters have been identified on the basis of medoid. In the proposed algorithm, initially, the number of the cluster is not defined. It has been calculated on the basis of best-suited fitness function value by the randomly selected group of medoid. Initially proposed algorithm has worked on random medoid value by the random selection method and iteratively it will be optimized into the best possible optimum number of clusters.

6.1.1 Initialization of characteristics of ant for clustering of components

Each ant (A) has the following characteristics :

- A search space of component instance (C_i) where C_i is : $C_{instance} \epsilon\, C_1, C_2, C_3, \ldots C_n$
- A empty Medoid list (M_l),which is initially empty. $M_l = \{\phi\}$

Table 4 The Restaurant Automation System (RAS) use cases, entity classes and actors

Use case	Use case name	Entity class	Entity class name	Actor	Actor name
UC_1	Login	ac_1	Employee	a_1	System timer
UC_2	Logout	ac_2	Order	a_2	Chef
UC_3	Enter hour (time sheet)	ac_3	Table	a_3	Host
UC_4	Add employee	ac_4	Item	a_4	Waiter
UC_5	Modify employee	ac_5	Bill	a_5	Administrator/manager
UC_6	Remove employee	ac_6	Customer	a_6	Customer
UC_7	Pay employee	ac_7	Customer survey		
UC_8	View employee info	ac_8	Timer		
UC_9	Assign tables	ac_9	Food queue		
UC_{10}	Update floor plan	ac_{10}	Customer queue		
UC_{11}	Update item menu				
UC_{12}	Schedule special				
UC_{13}	View statistics				
UC_{14}	View customer survey				
UC_{15}	Rate employee				
UC_{16}	View menu				
UC_{17}	Create order				
UC_{18}	Update order				
UC_{19}	Pay check				
UC_{20}	Enter survey				
UC_{21}	Call waiter				
UC_{22}	Add customer to food queue				
UC_{23}	Remove customer from food queue				
UC_{24}	Seat customer				
UC_{25}	Add customer to queue				
UC_{26}	Remove customer from queue				
UC_{27}	View table chart				
UC_{28}	Show customer food queue				
UC_{29}	select order				
UC_{30}	Update order status				
UC_{31}	Update table status				
UC_{32}	Serve food				

- A tabular storage t_s for visited instance which will store the fitness function value and combination of the component of the cluster. $t_s \epsilon \forall_{medoid}$ [$FF_{value} \cup$ set of component]

6.1.2 Transition rule for composition of components

According to Dorigo et al. an ants have two category of search transition strategies: exploitation and exploration. In our proposed approach the transition rule for composition of components for the medoid assignment is given in Eq. 8. The medoid assignment $M_{ie\{1,2,\ldots n|n>totalnumberofcomponent\}}$ as per the pseudo random proportional rule (Dorigo and Gambardella 1997):

$$M_i = \begin{cases} argmax_{ue\{M_k(j)\}}\{\tau[j,u].\eta[j,u]^\beta\} & if\ q \le q_0 \\ p_k(j,u) & otherwise \end{cases} \quad (8)$$

where M_k(i) is the set of medoid that remain to be visited by ant k positioned on subset of medoid $M_j \subset M_i$, τ is the pheromone value between the edge of medoid j to component instance u. η is the inverse of the distance between j to n and p_k(i,u) is:

$$P_k(i,u) = \begin{cases} \frac{[\tau(i,u)].[\eta(i,u)]^\beta}{\sum_{v\epsilon M_k(j)}[\tau(i,v)].[\eta(i,v)]^\beta} & if\ v\epsilon M_k(j) \\ 0 & otherwise \end{cases} \quad (9)$$

where $P_k(i,u)$ is the probability of component instance i and u belongs to the set of component those having the possibility to be medoid.In iteration first the medoid selection is random assignment according to the Eq. 8 and 9 but in next iteration the assignment is on the basis of meta-heuristic fitness function Eq. (9):

$$Meta_{FF} = SoftwareCohesion - SoftwareCoupling \quad (10)$$

this assignment will be performed until and unless the total number of medoid is not greater than a total number of components. If in any case there are possible set medoid gives the optimum value of meta-heuristic fitness function the next selection of assignment of medoid is stopped and remaining component instance treated as an instance of any medoid. In the next subsection, the proposed algorithm for the composition of a component in logical view is given.

6.1.3 The proposed algorithm for composition of component

The steps of proposed algorithm is given in below Algorithm 1 listing:

7 Measuring problem-solving performance of proposed algorithm

In view of the measurement of a problem -solving performance of a proposed algorithm, each ant having selected medoids which take evaluation step approx $O(KN)$, where K is the number of ants and N is the number of component instance. In the next forward step, the algorithm assigns the component instance to closest medoid which required $O(KNM)$ steps, where M is the number of medoids. In the same steps the involvement of calculation of similarity index of each data instance to assigned medoids, which takes $O(KN)$. After assignment of a component instance to closest medoid, the evaluation of meta-heuristic fitness function value will take in generalize way

Algorithm 1 Proposed Algorithm

procedure COMPCOMPOSITION
  Initialization of pheromone matrix: $\tau_0 \leftarrow$ component cohesion (CC_{comp}).
  Initialization of each ant k: set the chosen medoids $M_j \leftarrow \phi$ and the list of available component instance $\text{Comp}_i \leftarrow \phi$.
For each ant check:
  if All component *($Comp_i == N$)* or all medoid ($\text{M}_l == k$) have been chosen **then**
  goto *step 5.*
  select next component instance i.
  choose a search approach.
  if i is selected as a medoid **then** add it to M_l & remove i to not visited list $notVisited_l$;
  Assign each component instance except medoid instance to its closest medoid on the basis of objective function and update the each representative ant of medoid:
  $O^k \leftarrow \sum_{i=1}^{N} argmin_i^{M_l} d(M_l^k, Comp_i)$.where O^k represents the objective function of ant k , M_l^k is the medoid instance of ant k and Comp_i represents i instance of component list.
For each ant check:
  if All component instance assigned to anyone of medoid **then** Calculate
    compCohesion();
    compCoupling();
    compComplexity();
Loop:For all possible combination of medoid and component instance
Calculate the meta-heuristic fitness function value.
  if ($metaH_{FF_i} \geq metaH_{FF_{i-1}}$) **then** Update $t_s(FF, Composition) \leftarrow$ ($metaH_{FF_i}$, Composition of component).
    $t_s \leftarrow t_{s-1}$
Update the pheromone trails.Only the best ranked ant add pheromone
  $\tau_{t+1} \leftarrow (1-\rho)\tau_t(M_l^k) + \sum_{h=1}^{K_r} \Delta\tau_t(M_l^{K_r}, u)^h$,$\Delta\tau_t(M_l^{K_r}, u)^h \leftarrow \frac{1}{J^h}$,Where ρ is the pheromone trails evaporation rate,$(0 < \rho < 1)$,t is the iteration counter , K is the number of ranked ants and J^h is the quality of the solution created by ant h.
Termination Condition:
  if (number of iteration > maximum number of iterations)$\cup$(t_s has not been changed atleast in last multiple iteration) **then return** t_s(FF,Composition of component)
    goto step 3.

After the successful completion of the algorithm, the best configuration of a logical view of system architecture has recommended to software architect. The tabular storage t_s data structure consist the best so far solution of metaheuristic fitness function value ($metaheuristic_{FF}$) and composition of the component for a logical view of software system architecture.

$O(KNM) + O(KNM)$ as $\simeq O(KNM)$. So simplified complexity for calculation of $metaheurisitic_{FF}$ is $O(KNM)$. Finally the best ranking of ant solution, which takes $O(KlogK)$ and the pheromone update by B_r ant (B_r is best rank ant) and visit all component instance have been required $O(B_rN)$. Since the whole process completed at least $T(assume)$ iteration then total computational complexity is:

Table 5 Comparison of results for identified no. of components and quality metric on Mojo criteria (Tzerpos and Holt 1999)

Case study	Method	No. of components	No. of different use-cases compared with experts	Quality metric
OBS	COMO (CRUD based) (Lee et al. 1999)	6	12 from 30 use cases	60
	Shahmohammadi et al. (clustering-based) (Shahmohammadi et al. 2010)	6	5 from 30 use cases	83
	Cai et al. (FCA based) (Cai et al. 2011)	5	4 from 30 use cases	87
	SCI-GA (evolutionary) (Hasheminejad and Jalili 2013)	6	1 from 30 use cases	97
	Expert	6	0	100
	Proposed approach	6	0	100
RAS	COMO (CRUD based) (Lee et al. 1999)	7	17 from 32 use cases	47
	Shahmohammadi et al. (clustering-based) (Shahmohammadi et al. 2010)	7	10 from 32 use cases	69
	Cai et al. (FCA based) (Cai et al. 2011)	7	7 from 32 use case	78
	SCI-GA (evolutionary) (Hasheminejad and Jalili 2013)	8	6 from 32 use cases	81
	Expert	7	0	100
	Proposed approach	7	2 from 32 use cases	94
HIS	COMO (CRUD based) (Lee et al. 1999)	97	66 from 110 use cases	40
	Shahmohammadi et al. (clustering-based) (Shahmohammadi et al. 2010)	97	48 from 110 use cases	57
	Cai et al. (FCA based) (Cai et al. 2011)	97	35 from 110 use case	71
	SCI-GA (evolutionary) (Hasheminejad and Jalili 2013)	100	21 from 110 use cases	81
	Expert	97	0	100
	Proposed approach	99	2 from 110 use cases	98

Table 6 The fitness function value of dynamic design along with software cohesion and software coupling

Case study	Method	Software cohesion	Software coupling	Fitness function
OBS	COMO (CRUD based) (Lee et al. 1999)	0.825	0.171	0.654
	Shahmohammadi et al. (clustering-based) (Shahmohammadi et al. 2010)	0.927	0.163	0.764
	Cai et al. (FCA Based) (Cai et al. 2011)	0.919	0.183	0.736
	SCI-GA (evolutionary) (Hasheminejad and Jalili 2013)	0.969	0.159	0.810
	Expert	0.966	0.159	0.807
	Proposed approach	0.984	0.159	0.825
RAS	COMO (CRUD based) (Lee et al. 1999)	0.846	0.169	0.677
	Shahmohammadi et al. (clustering-based) (Shahmohammadi et al. 2010)	0.859	0.154	0.705
	Cai et al. (FCA based) (Cai et al. 2011)	0.864	0.156	0.708
	SCI-GA (evolutionary) (Hasheminejad and Jalili 2013)	0.907	0.138	0.769
	Expert	0.886	0.144	0.742
	Proposed approach	0.947	0.138	0.809
HIS	COMO (CRUD based) (Lee et al. 1999)	0.646	0.158	0.488
	Shahmohammadi et al. (clustering-based) (Shahmohammadi et al. 2010)	0.669	0.154	0.515
	Cai et al. (FCA based) (Cai et al. 2011)	0.754	0.156	0.598
	SCI-GA (evolutionary) (Hasheminejad and Jalili 2013)	0.987	0.138	0.849
	Expert	0.986	0.142	0.844
	Proposed approach	0.997	0.098	0.899

$O(TKN) + O(TKNM) + O(TKM) + O(TKlogK) + O(TB_rkN)$ as $\approx O(TKNM) \geq O(TKN) \geq O(TKM) \geq O(TB_rkN) \geq O(TKLogK)$ the simplified complexity is $O(TKNM)$.

8 Result evaluation

The evaluation of the proposed approach has been described on the virtual environment of health care and other IT applications. Initially, to measure the performance of the proposed model we used two well known real-world cases i.e. Online Broker System (OBS), Restaurant Automation System (RAS). Both of the case studies also used by the Hasheminejad and Jalili (2013). After that we have identified the software component of the virtual reality health insurance system.

Online broker system (OBS) recommended the stock price on the basis of historical data. This system has the features to predict and suggest for short term and long term investors to make decisions while investing in the stock market. The OBS case study comprises 30 Use cases, 4 actors, 22 analysis classes and 6 entity classes which is the example of a virtual reality system for stock prediction. The already extracted use case information by Hasheminejad and Jalili (2013) have been given in Table 3: use cases, actors and entity classes. Restaurant Automation System (RAS) comprises the 32 Use case, 6 actors, 25 analysis classes and 10 entity classes. This is the example of a virtual reality system for the automaton of restaurant system. Hasheminejad and Jalili 2013 extracted information has given in Table 4. According to the Hasheminejad et al. observation for OBS and RAS cases, 3 expert software professional developers with the average of 5 years of working experience are considered as experts, and one of them with more experience than others combines solutions of all developers to reach one solution. Definitely, the aim of all software professionals is to maintain the property of software design (high software cohesion and low software coupling and complexity).

For comparison, we have used Hasheminejad et al. experimental results of various classical clustering techniques and suggested technique SCI-GA (Hasheminejad and Jalili 2013). In experimental result of Hasheminejad et al. the CRUD-Based (Lee et al. 1999; Lee 1995), Clustering-based (Shahmohammadi et al. 2010), and FCA Based (Cai et al. 2011) methods are considered. We compared the proposed approach with the evolutionary computation approach suggested by Hasheminejad and Jalili (2013): SCI-GA. The final comparison between the proposed and existing novel approaches result is given in Tables 5 and 6.

For OBS case, the identified components by experts, SCI-GA framework (Hasheminejad and Jalili 2013) and proposed approach MeDCLCACO comparatively almost similar shown in Table 5. In comparison with SCI-GA framework, the proposed approach makes a better cohesive medoid cluster of similar component configuration which is similar to an expert component diagram. However, in SCI-GA have one different use case configuration in component identified by experts and the proposed approach. Although the only software cohesion value i.e. 0.984 varies in the proposed approach with the values identified by SCI-GA for components (i.e., 0.969).The other both parameter are approximately similar value.

For RAS case, in Table 5, the components identified by experts, the components identified by SCI-GA (Hasheminejad and Jalili 2013) and proposed approach shows that SCI-GA automatically obtains eight components which are overestimated comparison to expert components configuration but by the proposed approach we have obtained seven components similar to experts configuration. However, SCI-GA divides the experts Administrator component into two cohesive components but not by the proposed approach. Indeed, dividing the complex Administrator component into two simpler components in the SCI-GA results leads to improve the values of Software Cohesion, Software Coupling, and Software Complexity metrics, i.e., 0.907, 0.138 and 0.0153, respectively, in contrast to the ones identified by experts for the components (i.e. 0.886, 0.144 and 0.0159, respectively).

9 Conclusions

By the additional advantage of meta-heuristic in search-based software engineering, the proposed approach recommend the best possible configuration of software design specification of system architecture by an identified logical component on the basis of medoid based ant colony optimization techniques. In this paper, we have presented a novel approach to identify logical components based on the meta-heuristic and data mining techniques.

In literature,novel evolutionary approach identified the logical component but the proposed approach recommends slightly more accurate with high software cohesion and low coupling.In result, it improves with measuring problem-solving performance in simplified complexity is $O(TKNM)$. In this paper, we have also focused on design configuration according to the design pattern which results in the improved software cohesion value.

The proposed approach has assumed the use-cases as the edge of the connected graph of entire components of a system so that each use case is encoded as a connected source of Ant. Initially in the first iteration randomly selected use cases are considered as medoid of other use cases.

The efficiency of the proposed approach was evaluated by using three virtual reality system OBS, RAS and Healthcare Insurance case studies. Therefore the evaluation results have demonstrated that it outperforms in comparison to other

methods such as FCABased, Clustering-Based methods and evolutionary computation method. Additionally, it has a feature to automatically identify the best optimal configuration number of logical components for all three case studies (see Table 5), as opposed to the other methods and recommends the self-organized design of virtual reality (VR) system.

References

Aggarwal KK, Singh Y, Kaur A, Malhotra R (2006) Empirical study of object-oriented metrics. J Object Technol 5(8):149–173

AlSharif Mohsen, Bond Walter P, Al-Otaiby Turky (2004) Assessing the complexity of software architecture. In: Proceedings of the 42nd annual Southeast regional conference. ACM, pp 98–103

Bamodu Oluleke, Ye Xu Ming (2013) Virtual reality and virtual reality system components. In: Advanced materials research, vol 765. Trans Tech Publ, pp 1169–1172

Baumann J (1993) Military applications of virtual reality. Retrieved from Hiti. Washington. Edu on April 20:2016

Birkmeier Dominik, Overhage Sven (2009) On component identification approaches–classification, state of the art, and comparison. In International symposium on component-based software engineering, pp 1–18. Springer

Bouchard S, Baus O, Bernier F, McCreary DR (2010) Selection of key stressors to develop virtual environments for practicing stress management skills with military personnel prior to deployment. Cyberpsychol Behav Soc Netw 13(1):83–94

Briand LC, Daly JW, Wüst JK (1999) A unified framework for coupling measurement in object-oriented systems. IEEE Trans Softw Eng 1:91–121

Briand LC, Daly JW, Wüst. J (1998) A unified framework for cohesion measurement in object-oriented systems. Empir Softw Eng 3(1):65–117

Bunge M (1977) Treatise on basic philosophy: ontology I: the furniture of the world, vol 3. Springer Science & Business Media, Berlin

Bunge M (1979) Treatise on basic philosophy, ontology ii: a world of systems, vol 4

Cai Z, Yang X, Wang X, Kavs AJ (2011) Afuzzy formal concept analysis based approach for business component identification. J Zhejiang Univ Sci C 12(9):707

Chidamber SR, Kemerer CF (1994) A metrics suite for object oriented design. IEEE Trans Softw Eng 20(6):476–493

Clarke J, Dolado JJ, Harman M, Hierons R, Jones B, Lumkin M, Mitchell B, Mancoridis S, Rees K, Roper M (2003) Reformulating software engineering as a search problem. IEE Proc Softw 150(3):161–175

Cui JF, Chae HS (2010) Component identification and evaluation for legacy systems–an empirical study-. IEICE Trans Inf Syst 93(12):3306–3320

Davis S, Srivastava AK, Kumar S (2015) Automation in cloud computing using constraint based agent. In: 2015 international conference on computing, communication & automation (ICCCA), pp 649–654. IEEE

Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evolut Comput 1(1):53–66

Dorigo M, Socha K (2006) An introduction to ant colony optimization

Harman M, Jones BF (2001) Search-based software engineering. Inf Softw Technol 43(14):833–839

Harman M, Mansouri SA, Zhang Y (2012a) Search-based software engineering: trends, techniques and applications. ACM Comput Surv (CSUR) 45(1):11

Harman M, Clark J (2004) Metrics are fitness functions too. In: null. IEEE, pp 58–69

Harman M, McMinn P, De Souza JT, Yoo S(2012b) Search based software engineering: techniques, taxonomy, tutorial. In: Empirical software engineering and verification. Springer, pp 1–59

Hasheminejad SMH, Jalili S (2013) Sci-ga: software component identification using genetic algorithm. J Object Technol 12(2):3–1

Jain VK, Kumar S, Fernandes SL (2017) Extraction of emotions from multilingual text using intelligent text processing and computational linguistics. J Comput Sci 21:316–326

Jarallah S, Alghamdi, Raimi A Rufai, Khan Sohel M (2005) Oometer: a software quality assurance tool. In: null, pp 190–191. IEEE

Jenab K, Moslehpour S, Khoury S (2016) Virtual maintenance, reality, and systems: a review. Int J Electr Comput Eng (IJECE) 6(6):2698–2707

Kaufmann CR (2001) Computers in surgical education and the operating room. Annales chirurgiae et gynaecologiae 90:141–146

Kim Soo Dong, Chang Soo Ho (2004) A systematic method to identify software components. In: Software engineering conference, 2004. 11th Asia-Pacific. IEEE, pp 538–545

Kruchten P (2004) The rational unified process: an introduction. Addison-Wesley Professional, Boston

Kumar S, Mahanti P, Wang S-J (2018) Intelligent computational techniques. J Comput Sci 25:201–203

Lee JK, Seung SJ, Kim SD, Hyun W, Han DH (2001) Component identification method with coupling and cohesion. In: Apsec. IEEE, pp 79

Lee SD, Yang YJ, Cho FS, Kim SD, Rhew SY (1999) Como: a uml-based component development methodology. In: Software engineering conference, 1999 (APSEC'99). Proceedings. Sixth Asia Pacific. IEEE, pp 54–61

Lee Y (1995) Measuring the coupling and cohesion of an object-oriented program based on information flow. In: Proc. Int'l Conf. Software quality, 1995

Lele A (2013) Virtual reality and its military utility. J Ambient Intell Humaniz Comput 4(1):17–26

Liu YP, Chen HC, Hung TY, Yu CY (2018) Development and assessment of a visual-aid system for reducing the risk of neck injuries for computer users. J Ambient Intell Humaniz Comput 1–9

Mandhan N, Verma DK, Kumar S (2015) Analysis of approach for predicting software defect density using static metrics. In: 2015 international conference on computing, communication & automation (ICCCA). IEEE, pp 880–886

Menendez HD, Barrero DF, Camacho D (2014) A genetic graph-based approach for partitional clustering. Int J Neural Syst 24(03):1430008

Shahmohammadi G, Jalili S, Hasheminejad SMH (2010) Identification of system software components using clustering approach. J Object Technol 9(6):77–98

Srivastava AK, Kumar S (2018a) An effective computational technique for taxonomic position of security vulnerability in software development. J Comput Sci 25:388–396

Srivastava AK, Kumar S (2018b) Dynamic reconfiguration of robot software component in real time distributed system using clustering techniques. Procedia Comput Sci 125:754–761

Tzerpos Vassilios, Holt Richard C (1999) Mojo: a distance metric for software clusterings. In: Sixth working conference on reverse engineering, 1999. Proceedings. IEEE, pp 187–193

Verma D, Kumar S (2014) An improved approach for reduction of defect density using optimal module sizes. Adv Softw Eng 2014:4

Verma DK, Kumar S (2016) Exponential relationship based approach for predictions of defect density using optimal module sizes. Proc Natl Acad Sci India Sect A Phys Sci 86(2):201–208

Verma D, Kumar S (2017a) Prediction of defect density for open source software using repository metrics. J Web Eng 16(3–4): 293–310

Verma Dinesh, Kumar Shishir (2017b) Empirical validation of defect density prediction using static code metrics. In: Advances in information sciences and service sciences

Yu Yanfeng, Zhou Haibo, Fu Jiangfan (2018) Research on agricultural product price forecasting model based on improved BP neural network. J Ambient Intell Humaniz Comput, pp 1–6