*Computer Systems Science & Engineering*
DOI: 10.32604/csse.2023.025163
***Article***

Tech Science Press

# A Component Selection Framework of Cohesion and Coupling Metrics

**M. Iyyappan[1], Arvind Kumar[1], Sultan Ahmad[2,*], Sudan Jha[3], Bader Alouffi[4] and Abdullah Alharbi[5]**

[1]Department of Computer Science and Engineering, SRM University, Delhi-NCR, Sonepat, 131029, India
[2]Department of Computer Science, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Alkharj, 11942, Saudi Arabia
[3]School of Sciences, Christ (Deemed to be University), Delhi-NCR, Ghaziabad, 201003, India
[4]Department of Computer Science, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif, 21944, Saudi Arabia
[5]Department of Information Technology, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif, 21944, Saudi Arabia
*Corresponding Author: Sultan Ahmad. Email: s.alisher@psau.edu.sa
Received: 14 November 2021; Accepted: 05 January 2022

**Abstract:** Component-based software engineering is concerned with the development of software that can satisfy the customer prerequisites through reuse or independent development. Coupling and cohesion measurements are primarily used to analyse the better software design quality, increase the reliability and reduced system software complexity. The complexity measurement of cohesion and coupling component to analyze the relationship between the component module. In this paper, proposed the component selection framework of Hexa-oval optimization algorithm for selecting the suitable components from the repository. It measures the interface density modules of coupling and cohesion in a modular software system. This cohesion measurement has been taken into two parameters for analyzing the result of complexity, with the help of low cohesion and high cohesion. In coupling measures between the component of inside parameters and outside parameters. The final process of coupling and cohesion, the measured values were used for the average calculation of components parameter. This paper measures the complexity of direct and indirect interaction among the component as well as the proposed algorithm selecting the optimal component for the repository. The better result is observed for high cohesion and low coupling in component-based software engineering.

**Keywords:** Component-based software system; coupling metric; cohesion metric; complexity component; interface module density

## 1 Introduction

In component-based software system (CBSS) is mainly used for the development of major projects and less complexities of system [1]. The independent development of the traditional system used higher cost, poor quality, and risk as well as the failure of the system [2]. The CBSS provides better quality, lower cost for the development, increased reliability and reduced the complexity of the large-scale project [3].

The modularity approaches are used for the CBSS development where each module is more controllable and developed by integration components. A selection of suitable components with the help of measuring the coupling and cohesion density based on an optimization algorithm. Commercial off the shelf (COTS) component or in house components are available in the market it is used for the higher quality of software development also adopt for the latest technologies [4]. The C&C play a major role in determining the system quality in terms of reliability and availability [5]. Cohesion and coupling depends among all the components [6]. High cohesion is desirable as it accounts for improved strength and quality of the module, it is based on measurement of an object-oriented system [7]. A building block of the software component depends on the software object and modules to perform the specific functions [8]. Metrics play an important role in measurements take into the place of the controlling software engineering process [9]. The software quality metrics of packages are considering in to the development [10]. The system maintenance need to follow the metric parameters to gain the better modularity [11]. The selecting appropriate component is major challenge in software design and development [12]. The component composition metric(CCM) software development focus on the effort calculation of good and average category of CBSD system and component ratio metric(CRM) determine the mean of the component [13]. In the software component, the budget is main concern for reliable software. The complexity of two sets of metrics: component packing density (CPD) metric and component interaction density (CID) [14]. This metric analyse provide the larger support for the commercial software and delivery to customer need to follow certain protocols. Itinclude the characteristic of metric calculation among the software component, which provide a best commercial application for the business products.

This research article is organized as follows. In Section 2, reviewed the literature survey of existing research work. In Sections 3, proposed layout of component selection framework with the help of optimization technique. Section 4 discuss the proposed algorithm for Hexa-oval optimization selection algorithm which is applicable of the various parameter. Section 5, measure the interface density module of coupling and cohesion metric value. Section 6, proposed cohesion measurement for classes of direct and indirect interaction. Section 7, proposed coupling measurement for classes of direct and indirect interaction. Section 8, describes the complexity of coupling & cohesion measurement using the average component. In Section 9, discuss the conclusion and future work of the experimental study.

## 2 Literature Review

Gui et al. [15] in this paper discuss the evaluation of new coupling and cohesion metrics for a software component. The line of cohesion and tight class cohesion incorporate an indirect relationship between methods. Evaluation of software components between direct and indirect relationships is measured with the help of metrics. The weighted method per class is primarily concerned with the total number of methods and the method's priority. In object-oriented design, find the average calculation of metric. This methodology lacks implementation phases. Optimum selection of the component based on high reliability, lower cost, improved development phase, and deliver on time of the better quality of software and reliable system Chi et al. [16]. Analyzed the system response and cost for the development without any system failure. Umesh et al. [17] component selection based on lower complexity of overall software system can be reduced with the help of integration process, testing effort, and increased maintainability. Navneet et al. [18] represented component specification are system understandability and quality of application. Gill et al. [19] complexity level of the system increases based on the dependencies among the program module. Two categories: internal dependencies and external dependencies. In external process two metrics for black-box components: component dependency metric (CDM) and component interaction dependency metric (CIDM). Chillar et al. [20] in this paper metric shows the interaction with other components as well as quantify interface aspect of a component. Chidamber et al. [21] in this paper line of cohesion metric value calculate the number of pairs among methods in the class using without any

instance variable. Also discussed the depth of inheritance metrics used for measurement based on hierarchy. Hierarchy is related to the parent node as well as the child node. Sometimes this type of node process is complex at the level of measurement. Weyuker's concept of monotonicity failed to measure the inheritance process. Patel et al. [22] in this paper configuration of components are mainly used for the system maintenance as well as software installation. Software configuration will vary from system to system. So researcher cannot able to conclude the complexity of CBSE. This paper will provide fundamental information about their constituents and internal structure of methods or attributes. The sharing of components is used in component coupling. Chen et al. [23] discuss the complexity metric for software systems and propose a complexity-based cohesion measurement for software systems. However, the weyuker's methodology cannot be used in this paper.

## 3 Component Selection Framework

In this diagram of the proposed algorithm main concern is about the component selections framework which is 'α' denotes directly proportional to the Hexa-oval optimization algorithm. Which contain the various attribute like quality, reliability, and complexity of the system can be able to store in the warehouse of optimal components. This selection framework depends on the requirements which are collected from the user. The initial process contains the commercial off-the-shelf component which is used for the development CBSD system in Fig. 1. Also, it contains the client analysis and specification about that user analysis. The first process will get started. Based on the needs of the user, the software component repository is maintained consisting of in-house components and COTS components on the system repository.

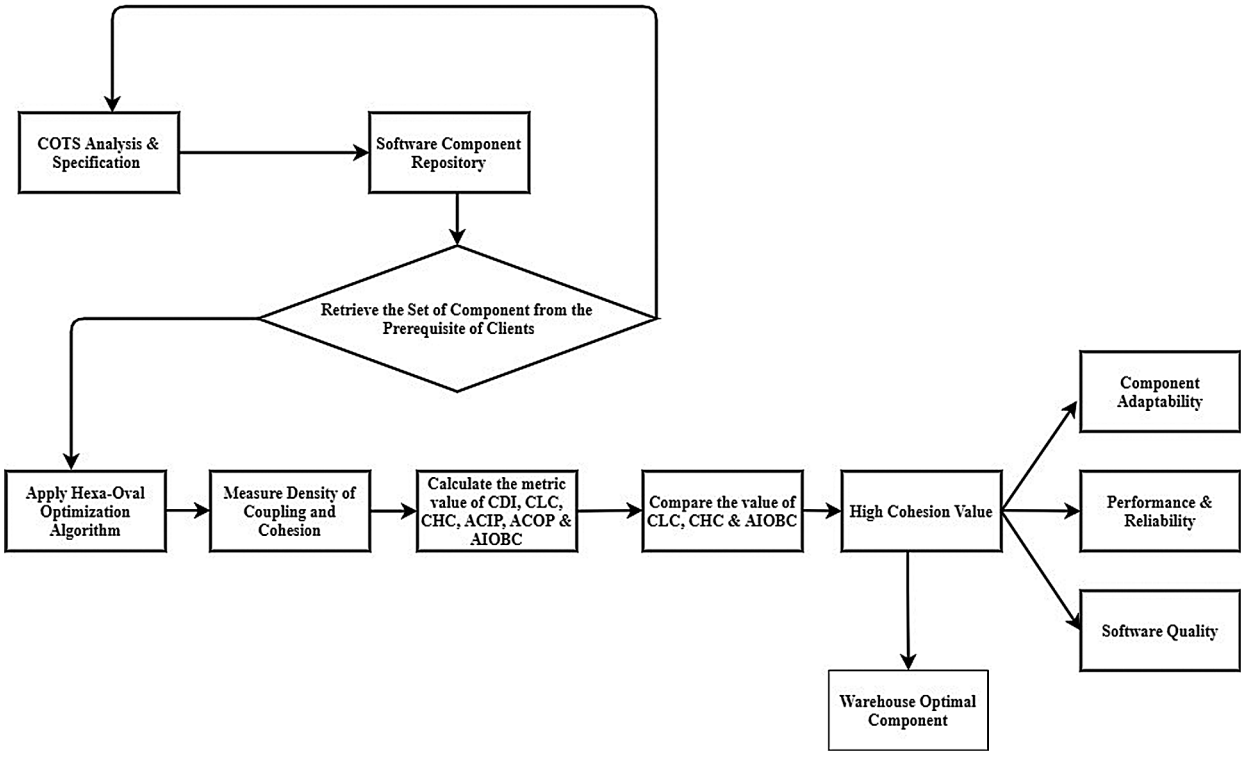



**Figure 1:** Proposed optimization of Hexa-oval algorithm model

The retrieve the set of the component from the software repository, according to the prerequisite of the client because it holds the various in-house component and COTS components. This retrieve set of processes able to measure the cyclomatic complexity as well as an adjacency matrix. The standard value of high complexity is 10. If the value comes nearby 10 then that kind of component can be considered to be somewhat complex. But if the calculated value comes out nearby 5 then its complexity value can also be calculated by using the proposed Hexa-oval optimization algorithm and the complexity of components for the development purpose selection can be analyzed. After applying it to the HOO algorithm, the various parameter related to the algorithm is proposed to calculate complexity level of the metric. This step of the HOO algorithm followed measuring the density of coupling and cohesion metric. After measuring density value moved to the classes of direct and indirect, low complexity and high complexity metric value measurement. Also, these metric values focused on the average calculation input parameter, output parameter, and both component parameters. The result of the calculated metric values then moved into the next of comparing the metric value of complexity of low component, complexity of high component and average of in & out both component parameter. The final comparison of the high cohesion value is better than the low cohesion value it is moved into the warehouse of optimal component selection framework. It is directly proportional to the component adaptability, performance of the system & reliability of the software, and improved software quality for the CBSD system.

## 4 Proposed Algorithm for Hexa-Oval Optimization Selection Algorithm

Step1:- Begin step refer the Fig. 2.

Step2:- Select [CPSC, CAS, CCR, RSCE]

CPSC - Client Prerequisite for Software Component

CAS – Component Analysis & Specification

CCR – COTS Component Repository

RSCE – Retrieve the Set of Component with Example

Step3:- CPSC = {1 to n}

// n is the number of the component in the repository

Step4:- Sol = null

// Solution is null at this time because no optimal component is selected according to Client requirements

Step5:- Retrieve the set of the component on the prerequisite of the client with general examples.

Step6:- Apply on Hexa-Oval Optimization Selection Algorithm based on step 2.

Step7:- Density measurement of components

$$MCCD = \frac{CCIin}{CCIout}$$

Step8:- Calculate the various parameter like classes of Direct or Indirect Interaction

$$CDI = \frac{R(R-1)}{2}$$

Step9:- Complexity of Low Cohesion measurement using Direct cohesion metric value

$$CLC = \frac{R(D)}{CDI}$$

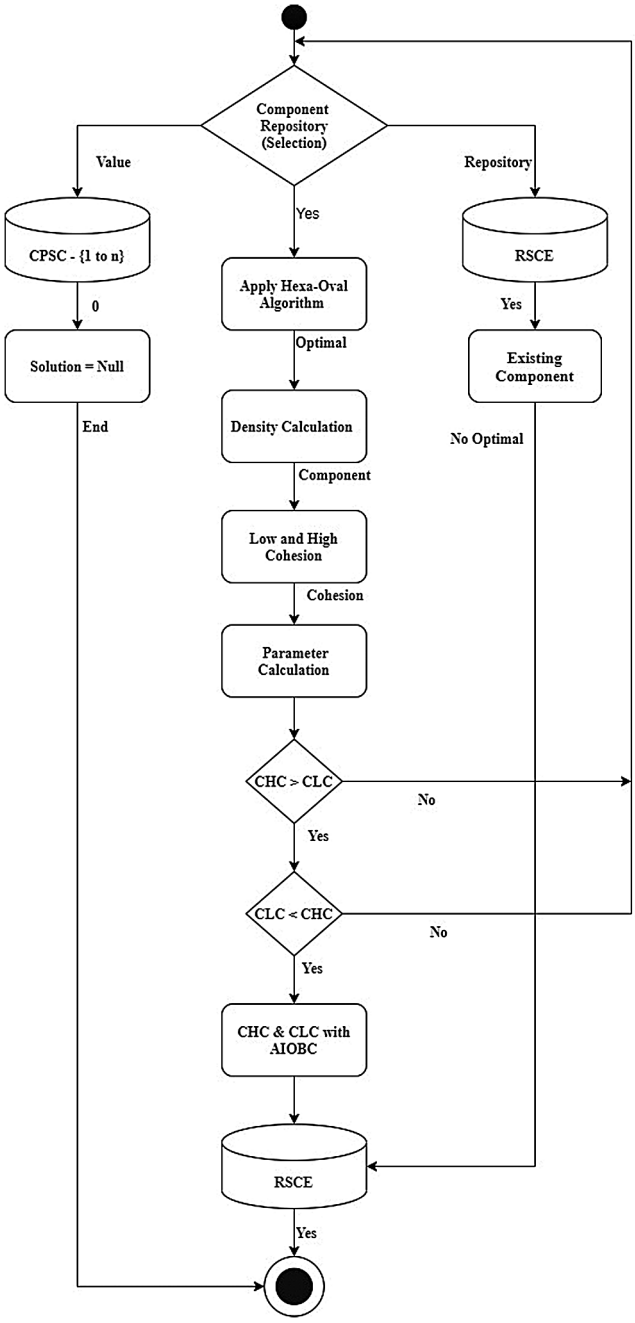



**Figure 2:** Flowchart of proposed Hexa-oval algorithm

Step10:- Complexity of High Cohesion using both Direct and Indirect cohesion metric value

$$CHC = \frac{R(DUI)}{CDI}$$

Step11:- Average measurement of Component In- Parameter for coupling & cohesion metric value

$$ACIP = \sum_{i=0}^{M} CIPi/m$$

Step12:- Average measurement of Component Out- Parameter for coupling & cohesion complexity metric value

$$ACOP = \sum_{i=0}^{M} COPi/m$$

Step13:- Average of In-Parameter and Out-Parameter of both component

$$AIOBC = ACIP + ACOP$$

Step14:- Compare the value of CLC and CHC with the support of **AIOBC**. If **CLC** must be less **CHC.** If **YES** Component will be selected. **NO,** then the component will not be selected.

Step15:- Solution = **RSCE** (Retrieve a set of Component engineering)

// Assign the YES value within a Component set to the solution

Step16:- If Solution meets the Client Prerequisite, go to step 18

Step17:- Repeat the Step 4 to 16.

Step18:- End

## 5 Measure the Interface Density module

The software architecture of the modular approach is developed using the component-based software system. The independently developed components are connected logically to another component module also a limited module for the software system. COTS or in-house components are supposed to any one of the modules for measuring the threshold value. Because this threshold value was used to measure the reliability of the system, reducing the cost and time to deliver the software. CBSS approach of software development using Inheritance concept of top-down strategy for analyzing the functional requirements. The components should be chosen in such a way that there is maximum interaction between components within software modules and minimal interaction between software modules in Fig. 3. This method focuses primarily on the metric measurement of interface density between each module.

IDM – Interface Density Module

S – Software Architecture

N – Each subsystem can have a highest of N elements.

***Density Measurement of Components***

In this paradigm, a quantitative measure is proposed. The following is the evaluate the interaction and adhesion is being used to examine the relationship between the Interface density modules (IDM1, IDM2, IDM3, IDM4, and IDM5) of a modular software system:

$$MCCD = \frac{CCIin}{CCIout} \tag{1}$$

Where CCIin is the number of coupling and cohesion interaction input within modules, and CCIout is the number of coupling and cohesion interaction output between the distinct modules.

$$MCCD = \frac{18}{18} = 1.$$

The above result is validated with the help of Inter-modular density and Intra-modular density between the cohesion and coupling measurement. This module measurement used the four different developed components of IDM1, IDM2, IDM3, IDM4, and IDM5 for the density analysis of the Inside module and the Distinct module of a software system.

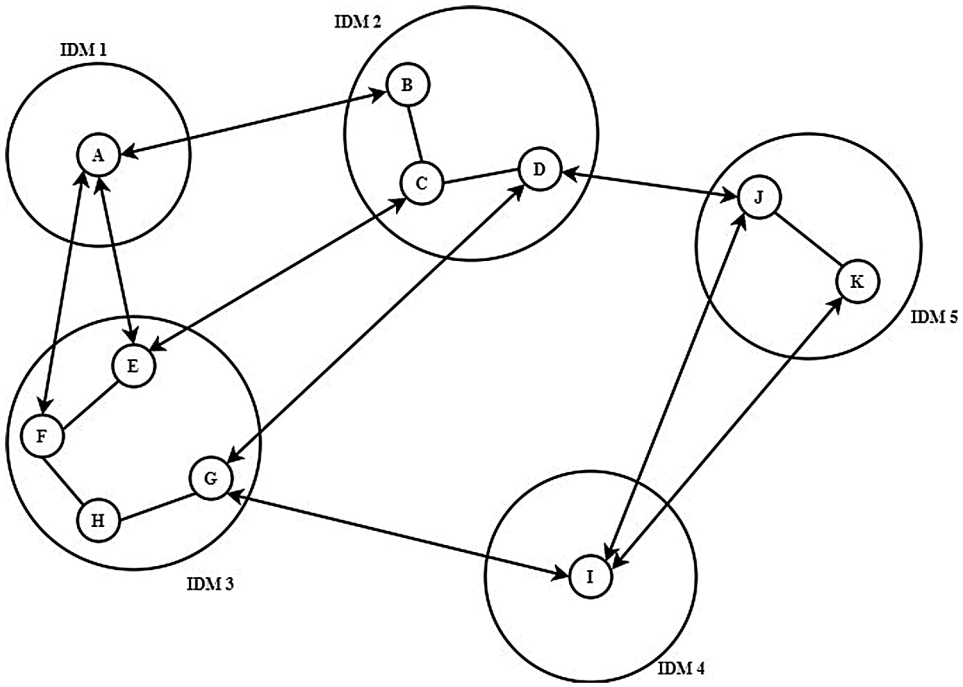



**Figure 3:** Coupling and cohesion density module

## 6 Cohesion Measurement for Direct and Indirect Interaction Classes

The package of direct interaction methods uses the class and indirect interaction uses the methods by importing the particular packages. CDI represents direct interaction or indirect interaction for the measurement of metric package level of cohesion.

$$CDI = \frac{R(R-1)}{2} \tag{2}$$

R = represents the client request to access either one interaction or both interaction for the complexity metrics. In the case study, classes of a total number of used direct or indirect connections and can be calculated as Pa (Ea+1, ra+1) / R(R-1) to analyse the elements, relation and threshold ranking of the software product. Then, in the case study, R represents a client request for a total number of packages. In the package measurement of hierarchical level represent the Ea+1 = 1 and ra+1 = 1, so this can be used to measure the client direct request of direct or indirect connections 2 / R(R-1) has been taken from the concept of TCB (Tough Category Bonding) and SCB (Soft Category Bonding) metric.

$$CDI = \frac{6(6-1)}{2} = 15$$

Here the average count of packages = 6. So the value as per the client request R = 6.

### 6.1 To Measure the Complexity of Cohesion

The higher the value of CLC and CHC wills, the lower the complexity of software because it adheres to terminology. The following are the definitions of the parameters used in the metrics calculation:

### 6.2 Complexity of Low Cohesion

In the term, CLC mentions complexity measurement with the help of low cohesion metric value. The CLC packages using the client request along with direct interaction to find the average using the classes of direct or indirect interactions.

$$CLC = \frac{R(D)}{CDI} \tag{3}$$

**R [D]:-** It is the client direct request for low cohesion measurement between the packages.

$$CLC = \frac{6(3)}{15} = 1.2$$

The value of the Client request = 6, Direct interaction value D = 3 and CDI = 15.

### 6.3 Complexity of High Cohesion

To check the complexity level of high cohesion packages using the client request along with direct and indirect interaction to find the average using the classes of direct or indirect interactions.

$$CHC = \frac{R(DUI)}{CDI} \tag{4}$$

**R [DUI]: -** It is the client's request of direct and indirect for high cohesion measurement based on complexity.

$$CHC = \frac{6(3+1)}{15} = 1.6$$

The value of the client request = 6, Direct interaction value = 3, Indirect interaction value = 1 and CDI = 15. Finally, there will be a comparison between the values of CLC and CHC. Value of CLC (1.2) ≤ Value of CHC (1.6). Therefore, from here, it can be concluded that the value of CLC comes out to be lower than the value of CHC and the high value of CHC will lead us to the low complexity of software. Similarly, these values for the rest of the five classes have been summarized in the Tab. 1 given below as:

**Table 1:** Comparison of cohesion metric value

| Classes | Methods(m) | CDI | R(D) | CLC | CHC | IS CLC<CHC |
|---|---|---|---|---|---|---|
| C1 | 6 | 15 | 3 | 1 | 1.6 | 1.6 YES |
| C2 | 5 | 10 | 2 | 1 | 1.5 | 1.5 YES |
| C3 | 6 | 15 | 3 | 1 | 1.6 | 1.6 YES |
| C4 | 5 | 10 | 2 | 1 | 1.5 | 1.5 YES |
| C5 | 5 | 10 | 2 | 1 | 1.5 | 1.5 YES |

## 7 Coupling Measurement Developed for Direct and Indirect Interaction Classes

Cohesion and coupling are primarily employed in the creation of component-based software systems. Two components are connected in this manner if and only if at least one of them operates on the other. Metrics employed a Graph (G) to construct coupling since it has nodes and edges. The node of a graph is mentioned about the component of vertices and the edges of the graph are the interface between the components. This directed graph has five nodes: A, B, C, D, and E, each of which is related to another node by edge interface contact. A graph's node and edge connectivity is primarily determined by five characteristics: starting point, end point, regular parameter, neutral parameter, crucial parameter for coupling metrics measurement in Fig. 4.

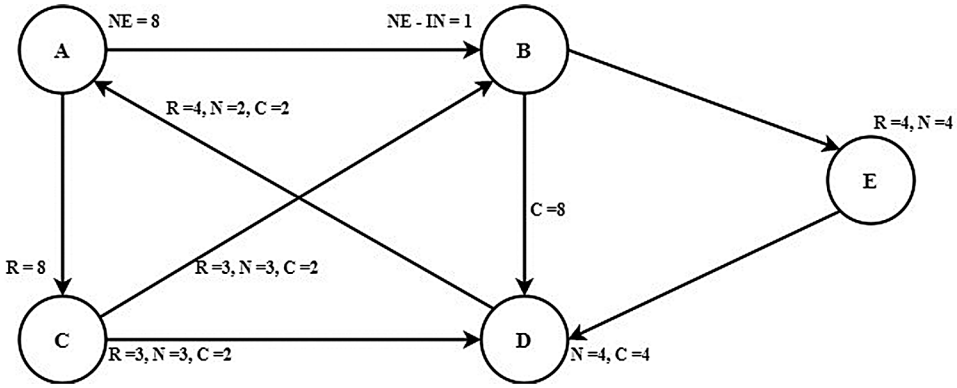



**Figure 4:** Coupling metric related to the node & edges graph

From the derived graph of node and edges are used the coupling intermix metrics. In this type of coupling metrics of intermix value represent =1 means there is an interface between the component and it represents = 0 means there is no interface between the component. The direct graph is used between the starting point and end point parameter if A to A = 0, A to B = 1 analysed the interface between the component in Tab. 2. In this graph matrix row and column equal to the total number of components or nodes. It can represent the edges of graph G by parallel arrays.

**Table 2:** Coupling graph matrix

| Graph matrix | A | B | C | D | E |
|---|---|---|---|---|---|
| **A** | 0 | 1 | 1 | 0 | 0 |
| **B** | 0 | 0 | 0 | 1 | 1 |
| **C** | 0 | 1 | 0 | 0 | 0 |
| **D** | 1 | 0 | 1 | 0 | 0 |
| **E** | 0 | 0 | 0 | 1 | 0 |

In the set of components used In (input) parameter and Out (Output) parameter to measure the coupling interface between the vertices and edges of the directed graph. The component used C = {C1, C2, C3 …Cn}. These parameters are further classified into (Regular, Neutral, Crucial) parameters in Tab. 3. In this method, each return value is considered as an input parameter and the passing of every argument is called an output parameter.

**Table 3:** Inside and outside parameter of the component between the coupling metrics

| Index | Start | End | Regular | Neutral | Crucial |
|---|---|---|---|---|---|
| 1 | A | B | 0 | 8 | 0 |
| 2 | A | C | 8 | 0 | 0 |
| 3 | B | D | 0 | 0 | 8 |
| 4 | B | E | 4 | 4 | 0 |
| 5 | C | B | 3 | 3 | 2 |
| 6 | D | A | 4 | 2 | 2 |
| 7 | D | C | 3 | 3 | 2 |
| 8 | E | D | 0 | 4 | 4 |

The passing of output parameter is mainly used for the array concepts for storing a large number of values on a single variable. The output parameter used five different representations starting point vertex, end point vertex, regular output parameter passed by the starting point interaction, neutral output parameter passed by the starting interaction, crucial output parameter passed by the starting interaction. The first row of the array represents that there is an interface between component A and component B which has eight parameters of regular, neutral, and crucial so an. So, will find the average of in & out components.

## 8 Measurement of Coupling and Cohesion Complexity Using the Average Component

### *8.1 Average Component In-Parameter (ACIP)*

The concept of this parameters, has been primarily used in the complexity measurement of interaction and group cohesiveness.

$$ACIP = \sum_{i=0}^{M} CIPi/m \tag{5}$$

**ACIP** = Average component in-parameter of coupling and cohesion.

**CIPi** = Component of Input Parameter along with 'i' term of minimum value with reference of maximum methods of 'm'.

***Component In-Parameter (CIPi)***

The eight simultaneous input data have been used to calculate the complex nature of components. In the element concatenation, the peak amount reflects the 'n' term, whereas the lowest signifies i = 0.

$$CIPi = \sum_{i=0}^{n} \begin{matrix} 0.10 \le Xi \le 30 \\ (input\ Parameter) \\ Value \\ 0\ \ No\ Parameter \end{matrix} \tag{6}$$

The component value of complexity calculation has taken into the two different parameters: Input value and Null value. The input parameter of Xi has set the limit of minimal functionality value 0.10 to maximum functionality value 0.30. If no input parameter has occurred, then it's moved into the value of 0 as a no parameter functionality.

For our experimental study has taken into the 8 different components observed the result from the graph matrix also calculated the total number of component value present on the input parameter in Tab. 4. This CIPi complexity value is used for the average calculation of the inside component from the vertex and edges.

**Table 4:** Calculate the component inside parameter value

| | | |
|---|---|---|
| **CIP 1** = 0.20 | **CIP 2** = 0.10 | **CIP 3** = 0.30 |
| **CIP 4 =** 0.20 | **CIP 5 =** 0.20 | **CIP 6 =** 0.10 |
| **CIP 7 =** 0.10 | **CIP 8 =** 0.30 | **CIPi = 1.5** |

### *8.2 Average Component Out-Parameter (ACOP)*

The idea of the ordinary element of endogenous variable, that is used to quantify the interconnection of the outside established sources.

$$ACOP = \sum_{i=0}^{M} COPi/m \tag{7}$$

ACOP = Average component out-parameter component.

COPi = Component of Output Parameter along with 'i' term of minimum value with reference of maximum methods of 'm'.

***Component Out-Parameter***

COPi has taken into account the complexity of the component in eight parallel interaction output measures. In the device summing up, the peak amount symbolises the 'n' term, although the margin requirement symbolises i=0.

$$COPi = \sum_{i=0}^{n} (ORi * Wr) + (ONi * Wn) + (OCi * Wc \tag{8}$$

OR = Output parameter of Regular vertex

ON = Output parameter of Neutral vertex

OC = Output parameter of Crucial vertex

Wr = Weight factor for regular type

Wn = Weight factor for neutral type

Wc = Weight factor for crucial type

The component value of complexity calculation has taken into the two different parameters: output parameter and weight factor. On the output, the parameter used the parallel array concept of regular vertex, neutral vertex, and crucial vertex. Then other parameters focus on the weighting factor used for interaction between regular type, neutral type, and crucial type in Tab. 5.

For our experimental study has taken into the 8 different components observed the result from the graph matrix also calculated the total number of component value interaction between the outside parameter. This COPi complexity value is used for the average calculation of outside components from the vertex and edges. In this concept, outside parameter vertex and weighting factor parameter value are used for measuring the interaction among coupling and cohesion complexity.

**Table 5:** Calculate the component outside parameter value

| | | |
|---|---|---|
| **COP1** = | 0*0.10 + 8*0.20 + 0*0.30 | = 1.6 |
| **COP2** = | 8*0.10 + 0*0.20 + 0*0.30 | = 0.8 |
| **COP3** = | 0*0.10 + 0*0.20 + 8*0.30 | = 2.4 |
| **COP4** = | 4*0.10 + 4*0.20 + 0*0.30 | = 1.2 |
| **COP5** = | 3*0.10 + 3*0.20 + 2*0.30 | = 1.5 |
| **COP6** = | 4*0.10 + 2*0.20 + 2*0.30 | = 1.4 |
| **COP7** = | 3*0.10 + 3*0.20 + 2*0.30 | = 1.5 |
| **COP8** = | 0*0.10 + 4*0.20 + 4*0.30 | = 2.0 |

### *8.3 Comparison of In & Out Parameter*

The outcome of the experimental analyses can be seen in this comparison Tab. 6, which includes the actual data set value which assess the complexity of output parameters.

**Table 6:** Calculate the component value and complexity level

| INTERFACE | COPi | ACOP | CiPi | ACIP | AIOBC |
|---|---|---|---|---|---|
| **I1** | 1.6 | 12.4 | 0.20 | 1.5 | 13.9 |
| **I2** | 0.8 | | 0.10 | | |
| **I3** | 2.4 | | 0.30 | | |
| **I4** | 1.2 | | 0.20 | | |
| **I5** | 1.5 | | 0.20 | | |
| **I6** | 1.4 | | 0.10 | | |
| **I7** | 1.5 | | 0.10 | | |
| **I8** | 2.0 | | 0.30 | | |

**I** = Interface value used for the component

**COPi** = Component out-parameter value using the three-parameter and weighting factor is observed for COPi and along with the average calculation of outside component interaction.

**CIPi** = Component in-parameter value using regular, neutral, crucial type observation along with the average calculation of inside component.

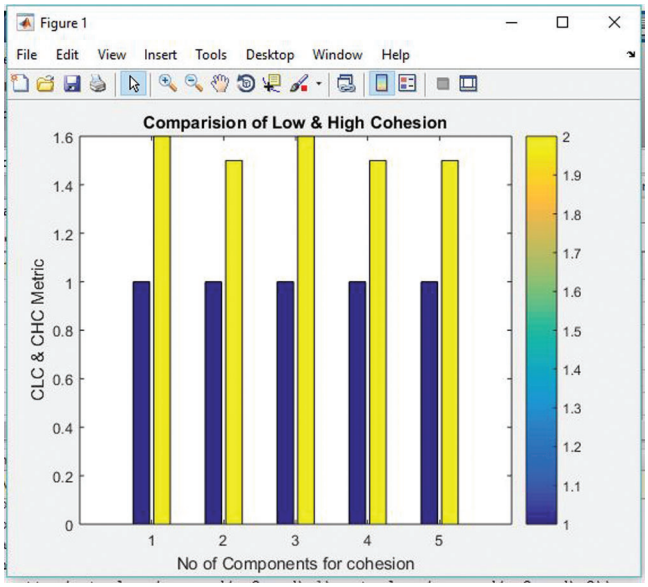
### *8.4 Average of In-Parameter and Out-Parameter of both component (AIOBC)*

The performance measures are using the nomenclature to estimate the inside and outside variables. It is used to assess the interface complexity of a component-based system.

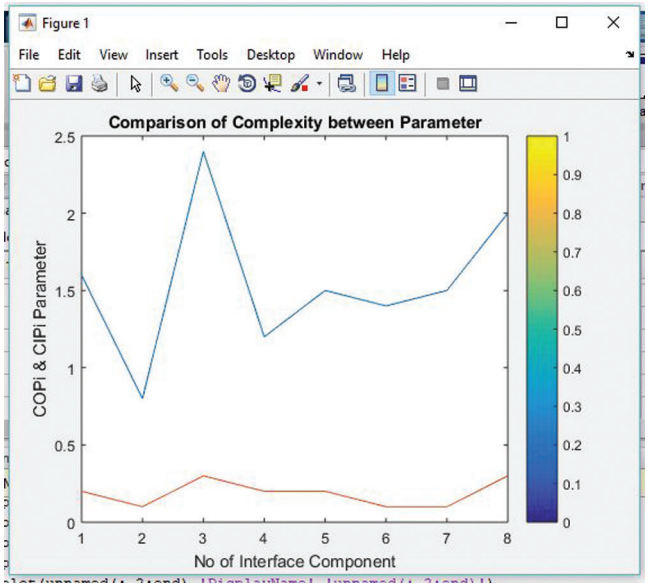
$$\mathbf{AIOBC} = \mathbf{ACIP} + \mathbf{ACOP} \tag{9}$$

$$\mathbf{AIOBC} = \mathbf{12.4} + \mathbf{1.5} = \mathbf{13.9}$$

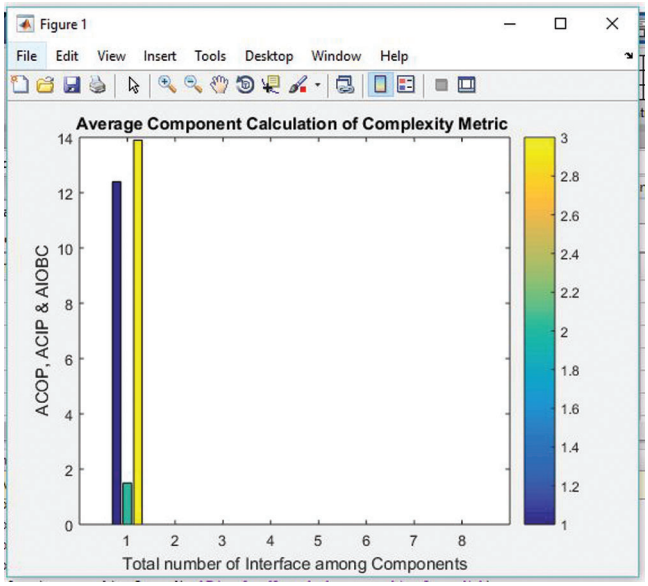
According to the above observed result which satisfies high level intradependence and low level interdependence among the software component with the proposed algorithm of optimal selection framework.




**Graph 1:** Calculated complexity of low cohesion and high cohesion with number of components




**Graph 2:** Calculated Complexity between component in-parameter & out-parameter comparison




**Graph 3:** Average calculation of complexity metric between the total numbers of components

## 9 Conclusion

This paper analyzed the low cohesion and high cohesion of complexity metric and coupling metric considered into the parameter of inside & outside interaction among the component. Interface modular density is also measured between the coupling and cohesion interaction among the vertex and edges. The proposed algorithm of HOO is also suitable for selecting the optimal selection of complexity among the component. The results are also compared with the previous graph, module, and matrix concepts. The MCCD, CLC, CHC, CDI, ACIP, ACOP, and AIOBC is also validated using the recently used properties. The proposed measures are based on the direct and indirect relations between classes and methods. This measurement will assist both these developers and middleware users in determining complex nature of cohesion. In the future rather than reusability, this metric can be used to establish the relations with some other factors in a component-based development environment like maintainability, adaptability.

**Acknowledgement:** We deeply acknowledge Taif University for Supporting this research through Taif University Researchers Supporting Project number (TURSP-2020/231), Taif University, Taif, Saudi Arabia.

**Funding Statement:** The authors received no specific funding for this study.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

[1] J. M. Bieman and B. K. Kang, "Measuring design-level cohesion," *IEEE Trans Software Engineering*, vol. 24, no. 2, pp. 111–124, 1998.

[2] J. M. Bieman and L. M. Ott, "Measuring functional cohesion," *IEEE Trans Software Engineering*, vol. 20, no. 8, pp. 644–658, 1994.

[3] L. Briand, S. Morasca and V. Basili, "Property-based software engineering measurement," *IEEE Trans Software Engineering*, vol. 22, no. 1, pp. 68–86, 1996.

[4] V. Cortellessa, F. Marinelli and P. Potena, "An optimization framework for build-or-buy decisions in software architecture," *Journal Computers and Operations Research*, vol. 35, no. 10, pp. 3090–3106, 2018.

[5] G. Priyalakshmi and R. Latha, "Evaluation of software reusability based on coupling and cohesion," *International Journal of Software Engineering and Knowledge Engineering*, vol. 28, no. 10, pp. 1455–1475, 2018.

[6] S. Almugrin, W. Albattah and A. Melton, "Using indirect coupling metrics to predict package maintainability and testability," *Journal of System and Software*, vol. 21, no. 1, pp. 298–310, 2016.

[7] F. B. Abreu and M. Goulao, "Coupling and cohesion as modularization drivers: Are we being over-persuaded?," in *Proc. of the 5th European Conf. on Software Maintenance and Reengineering*, Lisbon, Portugal, pp. 47–57, 2001.

[8] J. Chen, K. W. K. Yeap and S. D. Bruda, "A review of component coupling metrics for component based development," in *WCSE '09: Proc. of the 2009 WRI World Congress on Software Engineering*, Xiamen, China, pp. 65–69, 2001.

[9] D. Chaudhary and R. S. Chillar, "Component based software engineering systems: Process and metrics," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 1, no. 7, pp. 91–95, 2013.

[10] A. Aloysius and K. A. Maheswaran, "Review on component based software metrics," *International Journal of Fuzzy Mathematical Archive*, vol. 7, pp. 185–194, 2015.

[11] R. Sekar, A. J. V. D. Merwe, P. Kotze, M. M. Tanik and R. Paul, "Assessment of coupling and cohesion for component-based software by using shannon languages," *Journal of Integrated Design and Process Science*, vol. 8, no. 4, pp. 33–43, 2004.

[12] S. Mittal and P. K. Bhatia, "Predicting quantitative functional dependency metric based upon the interface complexity metric in component based software," *International Journal of Computer Application*, vol. 73, no. 2, pp. 21–28, 2013.

[13] R. S. Chhillar and P. Kajla, "New component composition metrics for component based software development," *International Journal of Computer Application*, vol. 60, no. 15, pp. 17–20, 2012.

[14] V. L. Narasimhan, P. T. Parthasarathy and M. Das, "Evaluation of suite of metrics for component based software engineering," in *Proc. of the Conference: InSITE 2009: Information Science and IT Education Conference*, Macon, United States, pp. 731–740, 2000.

[15] G. Gui and D. Scott, "Measuring software component reusability by coupling and cohesion metrics," *Journal of Computers*, vol. 4, no. 9, pp. 797–804, 2009.

[16] D. H. Chi, H. H. Lin and W. Kuo, "Software reliability and redundancy optimization," in *Proc. of the Annual Reliability and Maintainability Sym.*, Atlanta, GA, USA, pp. 41–45, 1989.

[17] U. Tiwari and S. Kumar, "Cyclomatic complexity metric for component based software," *ACM SIGSOFT Software Engineering*, vol. 39, no. 1, pp. 1–6, 2014.

[18] N. Kaur and A. Singh, "A complexity metrics for black box components," *International Journal of Soft Computing and Engineering*, vol. 3, no. 2, pp. 06–12, 2013.

[19] N. S. Gill and Balkishan, "Dependency and interaction oriented complexity metric of component based system," *ACM SIGSOFT Software Engineering*, vol. 33, no. 2, pp. 1–5, 2008.

[20] U. Chhillar and S. Bhasin, "A journey of software metrics: Traditional to aspect-oriented paradigm," in *5th National Conf. on Computing for National Development*, New Delhi, pp. 289–293, 2000.

[21] S. R. Chidamber and C. K. Kemerer, "Towards a metrics suite for object oriented design," in *Proc. of 6th ACM Conf. on Object Oriented Programming Systems Languages and Applications*, (Phoenix Arizona 1991), pp. 197–211, 1991.

[22] S. Patel and J. Kaur, "A study of component based software system metrics," *International Conference on Computing, Communication and Automation*, vol. 2, no. 1, pp. 824–828, 2016.

[23] J. Chen, H. Wang, Y. Zhou and S. Bruda, "Complexity metric for component based software systems," *"International Journal of Digital Content Technology and its Applications*, vol. 5, no. 3, pp. 235–244, 2011.