Int J Syst Assur Eng Manag
https://doi.org/10.1007/s13198-024-02331-w

ORIGINAL ARTICLE

Cohesion measurements between variables and methods using component-based software systems

Shipra[1]

Received: 31 August 2022 / Revised: 14 March 2024 / Accepted: 30 March 2024

Abstract The practice of leveraging previously created software components to progress new software is identified as component-based software engineering (CBSE). Good software engineering design is the foundation of CBSE principles. The black box approach that underpins CBSE hides the execution of components in nature, and the components communicate with one another using strictly delineated interfaces. Component platforms are shared, which lowers the price of creation. To ascertain a system's complexity, various software metrics are employed. For superiority in software intricacy, coupling would be minimal, and cohesiveness must be high. It is predetermined that coupling should be low and cohesion should be increased for refinement in software complexity. We are identifying the combination of different software systems and improving the methods for doing so with our approach. Proposed: Cohm (cohesion of methods) and Cohv (cohesion of variables) are two cohesion metrics that have been proposed. The cohesiveness metrics in this study have been analytically and empirically evaluated, and a comparison has been made between them. Additionally, an effort was made to give the outcomes of an empirical estimation based on the case study. The *T*-test is used to determine the consequences of the metrics, and Python is used to validate the metrics. Python or R programming and the Matlab tool are used to determine the relationship between various variables and metrics. Findings: The consequence of the current investigation is very encouraging and might be used to estimate the involvedness of the parts. The proportional analysis of the proposed metrics and various cohesion metrics reveals that the suggested metrics are more cohesive than the present metrics, increasing the likelihood that they can be reused when creating new applications.

Keywords Cohesion · Cohm (cohesion of methods) · Cohv (cohesion of variables) · CBSE metrics · Black box · Testing

✉ Shipra
shipra.frames@gmail.com

1 Department of Mathematics, Pt. NRS Govt. College, Rohtak, India

1 Introduction

Software components are preconfigured building elements with predetermined functions that can connect through industry-standard message interfaces. In contrast to software substances, mechanisms are larger units that indicate a better task or process level. A section can be used as a black box and has an outward description that is separate from its internal workings. Software development employing pre-built or existent software machines based on the definition of those apparatuses is known as CBSE. Measurements and their metrics are crucial for managing the software engineering process. Software metrics are measurable measurements employed to assess various traits and aspects of a software life cycle or the software organisation itself. Programme metrics are essential for evaluating and forecasting a variety of software qualities and complexity, including maintainability, testability, reusability, etc. Between all of these characteristics, the complexity aspect impacts every other part of the software, according to Gill and Balkishan (Weyuker 1988). Software metrics are crucial for forecasting, planning, carrying out, overseeing, controlling, and evaluating procedures and goods. Because technologies change over time, it is necessary to leverage ideas like constituent reusability, constituent interaction, and disappointment rate to create a novel product quickly. CBSE is a subset of software engineering that relies heavily on its constituent parts' dependencies, exchanges, and reuse. In CBSE, the capacity of the reusable component to provide fresh output with few errors and satisfy client needs depends on reliability (Biemen and Kang 1995). When assessing reusability in CBSE, component compatibility and reliability are crucial. CBS is a modern approach in software engineering that emphasises combining parts into sophisticated software organisations with the rapid growth of section technology. This method has various benefits, including quicker delivery, quality, lower maintenance costs, reusability, and productivity speed to market. Reliability may be anticipated by considering the dependability of each form of a part and the interconnection methodology among elements (Chen and Zhou 2011). Reusing existing mechanisms takes less time than creating a novel component. As a consequence, employing the component-based software engineering technique allows organisations to be constructed more quickly. Through lowering software expansion costs and raising software efficiency, businesses may become more competitive. Prior to making adjustments or developing the organisation, it offers an accurate view of the existing setup. Finished the dependency relationship, it unveils software configuration issues and reveals the implementation's difficulties without forcing us to study every line of code. The failure forecasting approaches used in CBS's reliability forecasting incorporate a quantitative evaluation of system dependability. The main goals of software metrics are cost reduction, quality improvement, control and keeping an eye on the timetable, minimising testing initiatives, and efficiently utilising reusable building blocks or pieces. The paper is divided into several parts; Sect. 2 reviews relevant research on a few fundamental cohesion metrics in the literature. Section 3 Limitations, Sect. 4 delivers a description of the scenario, Sect. 5 Materials and Methods, Sect. 6 discusses the problem at hand. Results and discussion are accessible in Sect. 6, and Sect. 7 concludes with a discussion of future instructions.

2 Related work

The asset of the relationship between elements and substances in a module is measured by cohesion. In other terms, it refers to how closely each module's instructions and components correspond to a certain purpose. Suit Chidamber, e.l. Metrics are LCOM (Lack of Cohesion in Methods) (Chidamber and Kemerer 1991). Later, it was changed to LCOM2 Chidambere (Chidamber and Kemerer 1994). LCOM2 is not employed in the observed investigation since it cannot discriminate between two software packages by giving them a cohesive score of zero. LCOM and LCOM2 don't take the invocation manner into account. In 2000, Li proposed RLCOM (Li et al. 2001). By separating the entire number of method pairs by the number of non-similar technique pairs, LCOM is extended. Cohesion measures were proposed by Hitz and Montazeri (1995). (LCOM3). It is a more effective form of LCOM. An unordered graph is used to illustrate how a class's functions are related to one another. A class's methods are its nodes. If two techniques share at least one parameter, there should be an edge. COM, LCOM3, and LCOM are, in reality, indicators of a lack of consistency; they really should be highlighted. TCC, which measures tight class cohesion, measures cohesion rather than lack of it. Bieman and Kang proposed TCC (Tight Class Cohesion) (1995). These measures take into account shared characteristics that methods will employ, as well as inter-method activation. All variables used in technique n would also be used by procedure m if procedure m is called procedure n. If two techniques use similar properties by referencing or invoking each other, they are said to be related. The similarity of approaches is viewed as an infinitive relation through these cohesion measures. LCOM3 and TCC take into account tangential connections between techniques. LCOM3 and TCC diagnose Similar to indirect and direct cohesiveness, Gandhi and Guie (2012), Gui, and Scott (2008) discuss the complexity factors of the component, techniques, and aggregate components (Biemen and Kang 1995; Chen and Zhou 2011; Chidamber and Kemerer 1991, 1994; Weyuker 1988; Gill and Balkishan 2008; Gandhi and Kumar 2012; Gui, and Scott 2008; Hitz and Montazeri 1995; Jianguo and Hui 2011). The author concludes that the sophisticated constituent requires extra time to run and is challenging to preserve and reuse after validation. Michael et al. (2015) have introduced BICM (Bounded Interface Complexity Metrics), which is an expansion of ICM. It is constrained; therefore, it might not always expand with size. According to the analysis of this statistic, connection size has no bearing on it (Tabrez et al. 2022; Singha et al. 2018a, 2018b, 2018c, d, 2022; Zubair and Singha 2021, 2020; Sultana et al. 2022; Arvind, & Ratan, R. 2020). The BICM can be used to assess a component's portability, independence, and self-completion. For component-based software systems, Kartika Yadav and Tomer (2014) (Singha et al. 2018c) introduced dual metrics: cohesion in class (CIC) and cohesion between components (CBM). These indicators are beneficial for raising the standard of CBSS design. In CBS, these metrics are being used to identify classes and components that are badly designed (Jianguo and Hui 2011; Li et al. 2001; Kaur and Singh 2013; Rana and Singh 2016; Singh and Chhillar, and Kajla, P. 2012; Singh et al. 2012; Sengupta and Kanjilal 2011; Sharma et al. 2009; Mittal and Bhatia 2013; Mwangi and Michael

2015; Tiwari and Kumar 2014). Two metrics—cohesion of variables within a component (COVC) and cohesion of methods in a component—were suggested by Rana and Rajender Singh (2016). (COMC). These statistics demonstrate the link between the variables utilised in various methodologies (Arvind & Ratan 2020; Bhat et al. 2022; Al-Taani and Al-Sayadi 2022; Kumar and Rath 2017; Azadeh et al. 2017; Ubaid et al. 2020; Jain and Raj 2018; Sreenivasula Reddy et al. 2022; Gadekar et al. 2022; Faiz and Daniel 2022). According to the authors, the difficulty of the component depends on the type and quantity of the parameters (Chhillar and Bhasin 2011; Singha et al. 2018a) (Din et al. 2023; Rostami et al. 2022; Zhang et al. 2019; Wechsler 2023; Liu 2021; D'Aniello et al. 2018; Taimoor et al. 2023; Samriya et al. 2023).

3 Limitations

- Most of the intrinsic metrics discussed above take into account direct connection, cohesion among classes, and explicit similarity among methods. Incorporating indirect linkages between procedures has been proposed as an addition by one of the cohesion metrics, LCOM3. It does not allow for a numerical specification of, directly or indirectly, cohesiveness and considers both in the same manner.
- The complexity of a module will grow with its size, given that ICM expands with the size of the component interface. This indicates that even if the new, enhanced component has substantially more identity, it will be graded poorly due to its increased complexity. The examination of BICM shows that it is self-governing in terms of boundary size. Still, it is necessary to assess it based on the entire system rather than just one element.
- Utilising metrics and tools that quantify them is one technique to assess the cohesiveness and coupling of a code base. LCOM4 (Lack of Cohesion of Methods) and CBO (Coupling Between Objects) are two examples of tests that may be used for the same thing. However, these metrics are ambiguous as to whether the computation should take into account only exiting dependencies or both incoming and outgoing dependents.

4 Problem description

The purpose of software engineering is to make high-quality software that is very inexpensive to maintain. At various phases of software development, the quality of the software is evaluated. Additionally, the design level can be assessed. The design of a component in a system that uses components has two views: internal and outward. Building customs is more of a concern for component developers. The component's value will automatically rise if the build quality of the component is poor. The number of lines of code must be raised, and more work must be put into updating the component to make it usable. High component reuse and low component reliance are results of good design. Metrics and their dimensions are crucial for managing the software engineering procedure. Software metrics are measurable measurements employed to assess various traits and aspects of a software expansion cycle or the software organisation itself. Programme metrics are essential for evaluating and forecasting a variety of software qualities, including maintainability, testability, complexity, etc. Out of all these, the complexity factor impacts every additional software attribute, according to Gill and Balkishan (2008). Software quality is crucial for forecasting, planning, carrying out, overseeing, controlling, and evaluating procedures and goods. Software metrics' main goals are to lower costs, advance quality, regulate and monitor the schedule, minimise testing requirements, and support efficient usage of reusable construction blocks. Cohesion metrics have been suggested throughout this research to evaluate the component's effectiveness. Our primary attention is on parameters and variables, such as its procedures, components' internal characteristics, etc.

5 Materials and methods

The intensity of a relationship between components inside a component is measured by cohesion. All of the commands in a cohesive component are directed towards carrying out a single, common goal. All the cohesive constituent needs to do is accept the statistics sent to it, process them, and then send the results to its superordinate component. The resemblance of a component's methods is described by cohesion. It measures the degree to which a component's multiple functions are connected.

5.1 Cohesion metrics

The cohesion of a component demonstrates how several properties relate to one another. The component's strength is shown. The cohesiveness component is an autonomous component that can be reused. If the component is very coherent, the likelihood of reuse rises.

5.2 Cohesion of variables (CoV)

A component's cohesion of factors refers to the regularity of its variables. The constituent is cohesive if the set of specified variables is focused on carrying out a particular purpose. The term "cohesion of variables" relates to how frequently variables are used in a component relative to the overall set of variables.

CoV$=\frac{\sum_{i=1}^{n} f(B_i)}{U}$, where U is called the Number of variables overall in the component.

$f(B_i)$ Is the frequency of use for each attribute in the component.

5.3 Cohesion of methods (CoM)

The term "cohesion of techniques" describes how closely variables are employed in procedures related to each other. This measure considers how different techniques communicate with one another inside a component to determine the component's strength. By counting the ways using the same sort of variables and separating by the number of approaches, this measurement determines the cohesiveness of methods in a component.

CoM $=\frac{\sum_{i=1}^{n} f(A_i)}{n^2+n+1}$, here $f(A_i)$ = number of techniques using the same kind of variables.

$V = n^2 + n + 1$ = a component's overall methods count.

6 Results and discussion

Experimentation is run on component-based software that is built in Java by Python to validate proposed complexity measures. Numerous Python components with varying numbers of object instances and methods can be found in this product.

6.1 Cohesion of variables

The occurrence of variables used in the element divided by the total number of variables is known as the cohesion of variables. There are ten (10) mechanisms in the sample, numbered B1 to B10. There are a few instance variables and methods within every element. The incidence of the variables is shown in Table 1. Some components' variable frequencies are the same, while others are different. Those frequencies are used to compute the CohV value.

6.2 Cohesion of methods (CoM)

The connectedness of a component's procedures and local variables is referred to as the cohesion of its methods. This metric takes into account the interplay of the techniques in a component. There are ten (10) elements in the illustration from B1 to B10. Every element has examples of variables and processes. Table 2 displays the number of methods being used with identical types of parameters.

Two cohesion metrics, Cohesion between Methods (CBM) and Cohesion in Class (CIC), were suggested by Tomar and Yadav (2018c). Cohesion in a class refers to how frequently the class's methods use its attributes (variables) in a component (Singha et al. 2018c). The relatedness of class members is referred to as cohesion between methods (Singha et al. 2018c).

6.3 Cohesion in class (CIC)

CIC$=\frac{\sum_{i=1}^{N} f(c_i)}{TM}$ N = Total Number of class attributes. $f(c_i)=$ frequency of use of each attribute by class methods for each attribute.TM = total Number of class methods.

6.4 Cohesion between methods (CM)

CM$=\frac{\sum_{i=0}^{a} f(c_i).M_i}{am(m-1)}$ $f(c_i).M_i$ = Total of the techniques that use the same kind of attributes.M = Number of class methodsa = Number of factors

To verify these measures, an empirical investigation based on Python components must be done. The identical Python must be used. CM and CIC must be determined for each Python component. The frequency of attributes

Table 1 CoV and the frequency of variables

U	F(B$_i$)	CoV	Components
4	7	3	B1
4	7	3	B2
5	9	3.25	B3
5	9	3.25	B4
17	46	3.56	B5
5	9	3.25	B6
3	6	3.50	B7
5	9	3.25	B8
4	6	3	B9
4	7	3	B10

Table 2 Demonstrates CoM values

V	F(A$_i$)	CoM	Components
13	7	0.6	B1
13	7	0.6	B2
31	9	0.4	B3
31	9	0.4	B4
65	46	0.006	B5
31	9	0.4	B6
5	6	3.0	B7
5	9	0.80	B8
4	6	0.6	B9
4	7	3.0	B10

Table 3 Displays the CIC value and the frequency of the characteristics

TM	F(c_i)	CIC	Components
5	7	2.5	B1
5	7	2.5	B2
7	9	2.5	B3
7	9	2.5	B4
31	45	3.0	B5
7	9	3.0	B6
3	6	3.0	B7
5	9	3.0	B8
4	6	2.5	B9
4	7	2.5	B10

and CIC values aimed at each component are displayed in Table 3. Table 4 displays the total number of methods utilised for the same type of attribute and the CBM value for each component. CM, CIC, and CoM statistical tools should be used to determine the statistical significance of the CoV results. These measures will be subjected to the T-test. The inferential analysis is the T-test. It is employed to ascertain whether the resources of the dual groups differ meaningfully (Fig. 1). Table 5 and Fig. 1 represents the standard deviation and mean of CIC and CoV. It is used to determine whether there is a significant difference between the means of two groups.

When paired sample t-tests are used on the data (results in Table 6), it is discovered that CoV's cohesiveness is greater than CIC [Tomer and Yadav]. The average value shows that the cohesiveness values of the given metrics are greater than those of the CIC that Yadav and Tomer have suggested (Singha et al. 2018c). The *T*-test value is 5.0654, and at a 99 percent degree of confidence, the same would be substantial. This indicates that CoV has a more significant cohesiveness value (Fig. 2).

When paired sample *t*-tests are used on this data (results in Table 6), it is discovered that CoM is much more cohesive than CM [Tomer and Yadav] (Singha et al. 2018c).

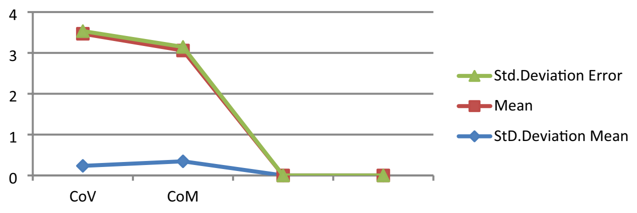

Fig. 1 Std. deviation and mean of CIC and CoV

Table 5 Std. Deviation and Mean of CIC and CoV

	Std. deviation	Mean	N	Std. deviation error
Cov	0.2345	3.2345	14	0.0654
CoM	0.3456	2.7134	14	0.2345

Table 6 *T*-Test on paired samples

	Paired differences 95%Confidenceintervalofthedifference	T	*DF*	Sig. (2-tailed)
Pair1 Cohv-CIC	0.8765	5.0654	12	99.000

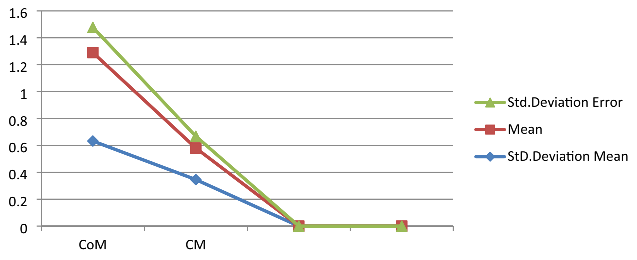

Fig. 2 Comparison between CM and CoM graph

Comparison between CM and CoM Graph (Result in Fig. 2 and Table 7) are applied on the data and found that the cohesiveness of proposed metrics CohM (Cohesion of Methods)

Table 4 Depicts the CBM value

TM	*FCI.M$_i$*	CM	Components	m	m − 1	$am(m-1)$,wherea = 1
5	7	0.50	B1	4	3	12
5	7	0.50	B2	4	3	12
7	9	0.50	B3	9	8	72
7	9	0.50	B4	9	8	72
31	45	0.007	B5	5	4	20
7	9	0.004	B6	5	4	20
3	6	0.003	B7	6	5	30
5	9	0.60	B8	6	5	30
4	6	0.60	B9	7	6	42
4	7	0.50	B10	7	6	42

Table 7 CoM and CM's standard deviation and median

	Std. deviation	Mean	N	Std. deviation error
CoM	0.63245	0.6576	14	0.18765
CM	0.34567	0.23456	14	0.087645

Table 8 Paired sample *T*-test

	T	*df*	Sig. (2- tailed)
Pair 1 CoM-CM	5.0654	12	99.000

Table 9 Mean and Std. deviation of CohM and CBM

	Mean	N	Std. deviation	Std. error mean
CohM	0.5537	13	0.61020	0.16205
CBM	0.25639	13	0.345075	0.093935

is more than CBM (Cohesion between methods). The mean value is reflecting that cohesion value of proposed metrics (CohM) is higher than the value of CBM which is proposed by Yadav and Tomer (Singha et al. 2018c). Table 7 and Fig. 2 denote CoM and CM's standard deviation and median. The paired sample *T*-test value is given in Table 8. The mean illustrates that the cohesiveness advantages of the proposed metrics (CoM) are higher than the value of the CM that Tomer and Yadav have proposed (Singha et al. 2018c). The 99 percent threshold of confidence *T*-test value is 5.0654, but the same is significant. It implies that CoM's value is higher and more significant in the recommended metrics (CoM). This analysis shows that, when compared to CIC and CM [Tomer and Yadav] (Singha et al. 2018c), the proposed metrics (CoV and CoM) are important. The suggested measures CIC and CBM were proposed by CoM and CVC, Tomar and Yadav (Singha et al. 2018c). These measures are based on the elements that are utilised in various ways by the components. CVC and CoM have been updated, and thus the cohesion of variables within a coefficient COM within an element is suggested (Rana and Singh 2016). These two measures are also affected by many variables and methodologies, but the authors classify the data into standard, moderate, and critical categories and take weights into account to normalize. The frequency of various types of variables that bind or enhance a component is represented by the COVC. The connectedness of a constituent's methods and instance variables is called the cohesion of methods. This measure considers how a component's methods communicate with one another (Rana and Singh 2016). Mean and Std. Deviation of CohM and CBM (Result in Table 9) are applied on the data and found that the cohesiveness of proposed metrics

Table 10 Paired sample *T* test

	T	Df	Sig. (2-tailed)
Pair 1 CohM-CBM	4.838	11	0.000

CohM (Cohesion of Methods) is more than CBM (Cohesion between methods). The mean value is reflecting that cohesion value of proposed metrics (CohM) is higher than the value of CBM which is proposed by Yadav and Tomer (Singha et al. 2018c). The *T*-test value is 4.838 and the same is significant at 99% level of confidence. It means that the value of CohM (cohesion of methods) is higher and significant in proposed metrics (CohM) in Table 10.

6.5 Cohesion of variables in a component (COVC)

$$COVC = \sum_{i=0}^{N} \frac{FV}{TV}$$

$$FV = \sum_{i=0}^{N} \{ [f(vi) * wi] + [f(vmi).wmi] + [f(vci) * wci] \}$$

hereFV = frequency of a component's instance variables.TV = Number of instance variables in a component as a whole.$f(vi) = the$ standard variable occurrence rates.$f(vmi)$ = regularity with which moderate variables occur.$f(vci)$=the frequency with which important factors arise.

The weight factors for the standard, moderate, and critical types of variables are, individually, *wi*, *wmi*, and *wci*.

The ten-part Python project will be used for the empirical analysis. Table 9 displays the COVC value as well as the frequency of various types of variables. The ten-part Python project will be used for the empirical analysis, conducted based on Python components, regarding each Java Beans component.

The inference is made that a component uses a moderate variable frequently, which increases its reusability when creating a brand-new application. The measures from Rajender Singh and Rana (Rana and Singh 2016) are subjected to correlation analysis. This leads to the conclusion that both the kind and frequency of the variables affect the component's complexity (coupling or cohesiveness). The outcome demonstrates that these characteristics have an impact on the component's complexity in Table 11. Although the recommended complexity seems rational and matches intuitive perception, it is not the sole factor in determining how complicated a CBSE is in total. One of our upcoming initiatives will involve conducting more empirical studies using genuine CBSS systems to apply our suggested measures. It will be possible to investigate the link between the suggested metric values and a number of CBS quality parameters using data from projects that the industry has already implemented. For standard-type variables, the Pearson correlation value (Table 12) is − 0.654. It implies that the value

Table 11 Displays the COVC value as well as the frequency of various types of variables

Component	$f(vi)$	$f(vmi)$	$f(vci)$	FV	TV	COVC
B1	3	3	3	1.3	4	0.404
B2	3	3	3	1.5	4	0.456
B3	5	3	4	1.7	4	0.453
B4	5	3	5	2.3	4	0.456
B5	3	17	6	2.5	5	0.567
B6	3	4	7	2.6	6	0.456
B7	0	4	5	2.8	2	0.567
B8	4	5	5	3	4	0.678
B9	5	2	3	3.4	5	0.654
B10	6	2	2	5	3	0.123

Table 12 Demonstrates the Pearson correlation between cohesiveness measurements and the frequency of several types of variables

Various variables	Correlation by pearson	Significance	N
Critical	0.675	0.0654	14
Standard	− 0.654	0.0134	14
Moderate	0.5678	0.0567	14

of COVC is reduced by 0.654 per unit if we increase the standard sort variables in a constituent. The moderate-type variable has a correlation value of 0.5678. It implies that the value of COVC grew by 0.5678 per unit if we raised the moderate-type variables in a component. For crucial-type variables, the Pearson correlation is also 0.675. This suggests that the value of COVC grew by 0.577 per unit if the frequency of essential type variables was increased in a component.

7 Conclusion and future work

Since moderate instance variables are used frequently within a component, there is a significant likelihood that the component can be reused when creating a brand-new application. To explore the relationship between the cohesiveness measure and the regularity of different kinds of variables (critical, moderate, and standard), the Pearson correlation approach is functional to the metrics developed by Rajender Singh and Rana (2016). For standard-type variables, the Pearson correlation value (Table No. 12) is − 0.654. It implies that the value of COVC is reduced by 0.654 per unit if we increase the standard sort variables in a constituent. The moderate-type variable has a correlation value of 0.5678. It implies that the value of COVC grew by 0.5678 per unit if we raised the moderate-type variables in a component. For crucial-type variables, the Pearson correlation is also 0.675. This suggests that the value of COVC grew by 0.577 per unit if the frequency of essential type variables was increased in a component. As a result, it is advised that researchers use fewer essential type variables and more moderate and critical sort variables. We already recognise that a component needs to have a high cohesion value and a low coupling value to be independent. Conclusion: To get the greatest outcomes for the component's strengthening, which results in the component's reusability for creating a new application, the use of moderate illustration variables within a section ought to be high. Projects completed using component-based systems are frequently delivered on time and within budget (Rajmohan and Ramasubramanian 2023; Pakrooh and Bohlooli 2021; Pawar et al. 2019; Li and Mao 2017; Upadhya 2023; Annepu and Rajesh 2020; Edla et al. 2020; Alimi et al. 2019; Miura and Suzuki 2003; Alam 2022). To assess the projects' complexity, metrics are created. An experimental assessment of the current and proposed metrics has been done on this Python. Cohesion measures have undergone extensive analysis and comparison between the proposed measures and various cohesiveness metrics. Using a statistical tool, CM and CIC's proposed metrics should be compared to CoM and CoV. The data is subjected to a t-test, which reveals that the proposed measures, CoM and CoV, have higher levels of cohesion than CM and CIC. The Pearson Correlation approach is used to display the relationship amid the cohesiveness measure and the regularity of various variables, functions that employ different variables, and COVC and COM are upgraded versions of CoV and CoM. The analysis of COVC and COM shows that by using modest wildcards, the component's strength, cohesion, and likelihood of being reused for creating new applications would all be high. The complexity of the constituent depends on the type and regularity of the variables, according to the case study's findings. The outcome demonstrates that these characteristics impact the component's complexity. The suggested cohesion complexity seems rational and fits intuitive perception. However, it is not the only factor determining how complicated a CBSE is. The primary drawbacks of the suggested technique are related to component trustworthiness since components are black-box programme units and users may not have access to the component's source code. Therefore, it is unwise to trust the components. Trade-offs between ideal criteria and available components are a constant in the system definition and design process, which may be another restriction. The use of moderate local variables within a portion and methodologies using moderate instance variables in an element must be on the higher side to have the best possible results for the reinforcing of the constituent, which in turn wires the reusability of the constituent for developing a novel application. Soft computing techniques and MATLAB may be utilised in future works to optimise the output of given measures (Ezugwu et al. 2022; Agushaka et al. 2022, 2023; Hu et al. 2023; Zare et al. 2023; Abualigah et al. 2023).

Author contributions SS is the only author of the paper, and she confirms that she was the only one to conceptualize the study, evaluate the findings, and prepare the draft manuscript. The final draft of the paper was authorized by the author after they had evaluated the findings.

Funding No funding.

Declarations

Conflict of interest The author declare that they have no conflict of interest.

Human and animals rights For this type of study, informed consent is not obligatory.

References

Abualigah L, Ekinci S, Izci D, Zitar RA (2023) Modified elite opposition-based artificial hummingbird algorithm for designing FOPID controlled cruise control system. Intell Autom Soft Comp. https://doi.org/10.32604/iasc.2023.040291

Agushaka JO, Ezugwu AE, Abualigah L (2022) Dwarf mongoose optimization algorithm. Comput Methods Appl Mech Eng 391:114570

Agushaka JO, Ezugwu AE, Abualigah L (2023) Gazelle optimization algorithm: a novel nature-inspired metaheuristic optimizer. Neural Comput Appl 35(5):4099–4131

Alam T (2022) Blockchain-enabled deep reinforcement learning approach for performance optimization on the internet of things. Wireless Pers Commun 126(2):995–1011

Alimi IA, Teixeira AL, Monteiro PP (2019) Effects of correlated multivariate FSO channel on outage performance of space-air-ground integrated network (SAGIN). Wireless Pers Commun 106(1):7–25

Al-Taani AT, Al-Sayadi SH (2022) Extractive text summarization of arabic multi-document using fuzzy C-means and Latent Dirichlet Allocation. Int J Syst Assur Eng Manag 15:713–726

Annepu V, Rajesh A (2020) Implementation of an efficient artificial bee colony algorithm for node localization in unmanned aerial vehicle assisted wireless sensor networks. Wireless Pers Commun 114:2663–2680

Arvind, Ratan R (2020) Identifying traffic of same keys in cryptographic communications using fuzzy decision criteria and bit-plane measures. Int J Syst Assur Eng Manag 11(2):466–480

Azadeh A, Jebreili S, Chang E, Saberi M, Hussain OK (2017) An integrated fuzzy algorithm approach to factory floor design incorporating environmental quality and health impact. Int J Syst Assur Eng Manag 8:2071–2082

Bhat J, Saqib M, Moon AH (2022) Fuzzy extractor and chaos enhanced elliptic curve cryptography for image encryption and authentication. Int J Syst Assur Eng Manag. https://doi.org/10.1007/s13198-021-01330-5

Biemen JM, Kang BY (1995) Cohesion and reuse in an object-oriented system. In: Proceeding on ACM symposium on software reusability (SSR'95), pp 259–262

Chen J, Wang H, Zhou Y et al (2011) Complexity metrics for component-based software systems. Int J Digital Content Technol Appl 5(3):235–244

Chhillar U, Bhasin S (2011) A journey of software metrics: traditional to aspect-oriented paradigm. In: 5th National conference on computing for nation development, 289–293

Chidamber SR, Kemerer CK (1991) Towards a metrics suite for object oriented design. In: Proceedings of 6th ACM conference on object oriented programming, systems, languages and applications (OOPSLA'91), 197–211

Chidamber SR, Kemerer CKA (1994) Metrics suite for object oriented design. IEEE Trans Software Eng 20:476–493

D'Aniello G, Gaeta A, Gaeta M, Tomasiello S (2018) Self-regulated learning with approximate reasoning and situation awareness. J Ambient Intell Humaniz Comput 9:151–164

Din AFU, Mir I, Gul F, Akhtar S (2023) Development of reinforced learning based non-linear controller for unmanned aerial vehicle. J Ambient Intell Humaniz Comput 14(4):4005–4022

Edla DR, Tripathi D, Kuppili V, Dharavath R (2020) Multilevel automated security system for prevention of accidents at unmanned railway level crossings. Wireless Pers Commun 111:1707–1721

Ezugwu AE, Agushaka JO, Abualigah L, Mirjalili S, Gandomi AH (2022) Prairie dog optimization algorithm. Neural Comput Appl 34(22):20017–20065

Faiz M, Daniel AK (2022) A multi-criteria cloud selection model based on fuzzy logic technique for QoS. Int J Syst Assur Eng Manag 15:687–704

Gadekar R, Sarkar B, Gadekar A (2022) Model development for assessing inhibitors impacting Industry 4.0 implementation in Indian manufacturing industries: an integrated ISM-Fuzzy MICMAC approach. Int J Syst Assur Eng Manag 15(2):646–671

Gandhi P, Kumar BP (2012) Analytical analysis of generic reusability Weyuker's. Properties in international journal of computer science issues (IJCSI), 1–9

Gill NS, Balkishan (2008) Dependency and interaction oriented complexity metrics of component-based systems. ACM SIGSOFT Softw Eng Notes 33(2):1–5

Gui G, Scott PD (2008) New coupling and cohesion metrics for evaluation of software component reusability. In: 9th International conference for young computer scientists, IEEE https://doi.org/10.1109/ICYCS.2008.270

Hitz M, Montazeri B (1995) Measuring coupling and cohesion in object-oriented systems. In: Proceedings of the international symposium on applied corporate computing, pp 1–8

Hu G, Zheng Y, Abualigah L, Hussien AG (2023) DETDO: An adaptive hybrid dandelion optimizer for engineering optimization. Adv Eng Inform 57:102004

Jain V, Raj T (2018) An adaptive neuro-fuzzy inference system for makespan estimation of flexible manufacturing system assembly shop: a case study. Int J Syst Assur Eng Manag 9:1302–1314

Jianguo C, Hui W (2011) Complexity metrics for component-based software systems. Int J Digital Cont Technol Appl 5(3):235–244

Kaur N, Singh A (2013) A complexity metrics for black box components. Int J Soft Comput Eng 3(2):179–184

Kumar L, Rath SK (2017) Software maintainability prediction using hybrid neural network and fuzzy logic approach with parallel computing concept. Int J Syst Assur Eng Manag 8:1487–1502

Li X, Liu Z, Pan B, Xing B (2001) A measurement tool for object-oriented software and measurement experiments with IT. In: Proceeding on IWSM, 2000. (Lecture Notes in Computer Science 2006, Springer, Berlin and Heidelberg, 2001), 44–54

Li L, Mao Y (2017) Autonomously coordinating multiple unmanned vehicles for data communication between two stations. Wireless Pers Commun 97:3793–3810

Liu W (2021) Slam algorithm for multi-robot communication in unknown environment based on particle filter. J Amb Intell Human Comput. https://doi.org/10.1007/s12652-021-03020-3

Mittal S, Bhatia PK (2013) Predicting quantitative functional dependency metric based upon the interface complexity metric in component-based software. Int J Comput Appl 73(2):1–10

Miura R, Suzuki M (2003) Preliminary flight test program on telecom and broadcasting using high altitude platform stations. Wireless Pers Commun 24:341–361

Mwangi T, Michael A (2015) Empirical evaluation of complexity metrics for component-based systems. J Theor Appl Inf Technol 73(2):275–282

Pakrooh R, Bohlooli A (2021) A survey on unmanned aerial vehicles-assisted internet of things: a service-oriented classification. Wireless Pers Commun 119:1541–1575

Pawar P, Yadav SM, Trivedi A (2019) Performance study of dual unmanned aerial vehicles with underlaid device-to-device communications. Wireless Pers Commun 105:1111–1132

Rajmohan S, Ramasubramanian N (2023) Improved Symbiotic organisms search for path planning of unmanned combat aerial vehicles. J Ambient Intell Humaniz Comput 14(4):4289–4311

Rana P, Singh R (2016) A design of cohesion and coupling metrics for component-based software systems. Int J Comput Appl 146(4):23–27

Rostami M, Farajollahi A, Parvin H (2022) Deep learning-based face detection and recognition on drones. J Ambient Intell Human Comput 15(1):373–387

Samriya JK, Kumar M, Tiwari R (2023) Energy-aware aco-dnn optimization model for intrusion detection of unmanned aerial vehicle (uavs). J Ambient Intell Humaniz Comput 14(8):10947–10962

Sengupta S, Kanjilal A (2011) Measuring complexity of component-based architecture: a graph-based approach. ACM SIGSOFT Softw Eng Notes 36(1):1–10

Sharma A, Grover PS, Kumar R (2009) Dependency analysis for component-based software systems. ACM SIGSOFT Softw Eng Notes 34(4):1–6

Singh R, Chhillar, Kajla P (2012) New component composition metrics for component-based software development. Int J Comput Appl 60(15):51–56

Singh R, Chhillar, Ahlawat P, Kumari U (2012) Measuring complexity of component-based system using weighted assignment technique. In: 2nd International conference on information communication and management (ICICM 2012), 17–23

Singha AK, Kumar A, Kushwaha PK (2018) Classification of brain tumors using deep Encoder along with regression techniques. EPH-Int J Sci Eng 1(1):444–449

Singha AK, Kumar A, Kushwaha PK (2018) Patient cohort approaches to data science using biomedical field. EPH-Int J Sci Eng 1(1):457–462

Singha AK, Kumar A, Kushwaha PK (2018) Recognition of human layered structure using Gradient decent model. EPH-Int J Sci Eng 1(1):450–456

Singha AK, Pathak N, Sharma N, Gandhar A, Urooj S, Zubair S, Sultana J, Nagalaxmi G (2022) An experimental approach to diagnose covid-19 using optimized CNN. Intell Autom Soft Comput 34(2):1066–1080

Singha AK, Kumar A, Kushwaha PK (2018) Speed prediction of wind using artificial neural network. EPH-Int J Sci Eng, pp. 463–469

Sreenivasula Reddy T, Sathya R, Nuka M (2022) Intuitionistic fuzzy rough sets and fruit fly algorithm for association rule mining. Int J Syst Assur Eng Manag 13(4):2029–2039

Sultana J, Singha AK, Siddiqui ST, Nagalaxmi G, Sriram AK, Pathak N (2022) COVID-19 pandemic prediction and forecasting using machine learning classifiers. Intell Autom Soft Comput 32(2):1007–10243

Tabrez SS, Ahmad MO, Khamruddin M, Gupta AK, Singha AK (2022) Blockchain and IoT for educational certificates generation and verification. In: 2022 2nd international conference on computing and information technology (ICCIT), IEEE, pp. 298–303

Taimoor M, Lu X, Maqsood H, Sheng C (2023) A novel fault diagnosis in sensors of quadrotor unmanned aerial vehicle. J Ambient Intell Humaniz Comput 14(10):14081–14099

Tiwari U, Kumar S (2014) Cyclomatic complexity metric for component-based software. ACM SIGSOFT Softw Eng Notes 39(1):1–10

Ubaid AM, Aghdeab SH, Abdulameer AG, Al-Juboori LA, Dweiri FT (2020) Multidimensional optimization of electrical discharge machining for high-speed steel (AISI M2) using Taguchi-fuzzy approach. Int J Syst Assur Eng Manag 11(6):1021–1045

Upadhya A (2023) On the reliability of interference limited unmanned aerial vehicles. Wireless Pers Commun 129(1):119–131

Wechsler H (2023) Immunity and security using holism, ambient intelligence, triangulation, and stigmergy: sensitivity analysis confronts fake news and COVID-19 using open set transduction. J Ambient Intell Humaniz Comput 14(4):3057–3074

Weyuker EJ (1988) Evaluating software complexity measures. IEEE Trans Softw Eng 14(9):1357–1365

Yadav K, Tomar P (2014) Design of metrics for component-based software system at design level. Int J Eng Tech Res 2(4):285–289

Zare M, Ghasemi M, Zahedi A, Golalipour K, Mohammadi SK, Mirjalili S, Abualigah L (2023) A global best-guided firefly algorithm for engineering problems. J Bionic Eng. https://doi.org/10.1007/s42235-023-00386-2

Zhang K, Qu T, Zhou D, Thürer M, Liu Y, Nie D et al (2019) IoT-enabled dynamic lean control mechanism for typical production systems. J Ambient Intell Human Comput 10:1009–1023

Zubair S, Singha AK (2020) Parameter optimization in convolutional neural networks using gradient descent. Microservices in big data analytics, Springer, Singapore, pp. 87–94

Zubair S, Singha AK (2021) Network in sequential form: combine tree structure components into recurrent neural network. In: IOP conference series: materials science and engineering. Vol 1017(1), p. 012004