Webology (ISSN: 1735-188X)
Volume 18, Number 6, 2021

Investigating Role Of Functional, Coupling, And Constraint Complexity Metrics In Component Based Software Engineering

PARUL[1] , **RAJENDER SINGH**[2]

[1](Research scholar) Department of computer science & Application Maharshi Dayanand University, Rohtak, Haryana, India.

[2](Professor) Department of computer science & Application, Maharshi Dayanand University, Rohtak, Haryana, India.

Abstract: Component-based software metrics are used to measure the quality and risk of component-based software. Measurement begins with an assessment of the task's most critical components. Split them into sub-characteristics or characteristics thereafter. As a consequence, these more subtle sub-features begin to emerge in characteristics. This definition-based metric property is used to get the required metrics. Prior studies have studied software metrics based on component composition. Too little research has been done on component-based software metrics so far. Studies like this have a very specific focus. Programming modules were compared in terms of usefulness, new features and the difficulty and complexity of configuring the modules. In the suggested research, formulas for calculating the functional, coupling, and constraint complexity metrics were studied.

Keyword: CBSE, CBD, Software component, Software reusability, Software metric.

[1] INTRODUCTION

Component-based architecture allows for the reuse of classes (components required to construct an application). The spiral concept is used in a variety of ways in this design. We may thank evolution for this concept. As a result, iterative software design is a viable option. Component-based development (CBD) simplifies the design and development of computer systems by using reusable software components. CBD has shifted its focus to software system design.

Component Based Software Engineering (CBSE)

Component-based software engineering (CBSE), sometimes known as components-based development (CBD), is a branch of software engineering that focuses on decoupling the numerous components that make up an agiven software system. This is a reuse-based technique for putting

together systems of loosely connected components. In both the short and long term, this technique strives to bring about a broad range of advantages for both the software itself and the sponsoring corporations.

There is a strong connection between components and service-oriented architecture. Adding new features to an existing component is one of the benefits of using a web service. Web services and service-oriented architectures, for example, are examples of this (SOA).

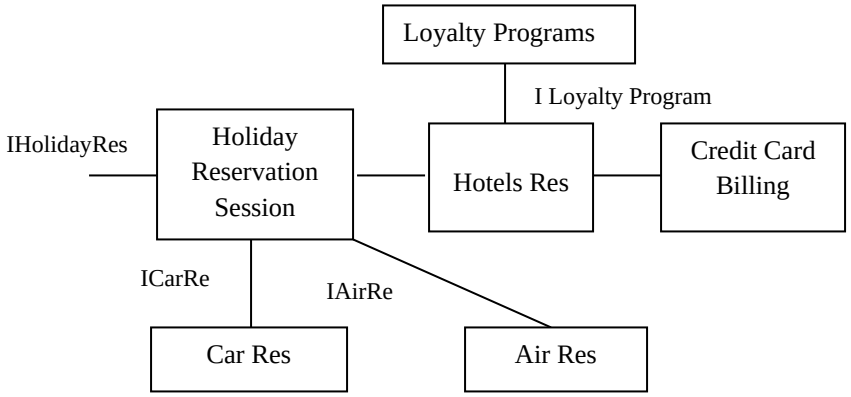

Fig 1 Component Based Software Engineering (CBSE)

Software Component

Software packages, online services, web resources, and modules are the most frequent discrete components of software (or data). The system's processes are divided into distinct components to guarantee that all data and functions inside each component are semantically connected (just as with the contents of classes). Be a consequence of this mindset, components are typically referred to as modular and cohesive.

Objects (rather than classes), collections of objects (as in object-oriented programming), and some type of binary or textual interface description language are all popular forms of software components that may operate independently of each other on a computer (IDL). The component's functioning is not affected by changes to the code.

Component model

Specifying the characteristics of components as well as how to put them together is the purpose of a component model.

There have been many different component models proposed by researchers and practitioners over the last several decades. This section offers a breakdown of the existing models of components.

The EJB, COM,.NET, X-MAN, and CORBA models from the EJB family of JavaBean implementations are examples of component models.

Software metric

Software metrics come in a wide variety of forms and may be used to assess the overall quality of a software system or process. Despite the fact that metrics are functions and measures are the values that are the consequence of applying metrics, the terms are commonly used interchangeably. Since quantitative metrics are essential in many domains, computer science practitioners and theoreticians are always trying to apply similar methodologies to software development. For a number of purposes, including budget and schedule planning, cost estimates, quality assurance, and software debugging and optimization, we need measures that are precise and repeatable. We also need to know how to best allocate work to our staff..

Software reusability

To be reusable, assets like code, components, test suites, designs, and documentation must have been previously generated in the context of software engineering or computer science. Reusability encompasses the whole process of creating, packing, distributing, installing, configuring, deploying, maintaining, and upgrading. Software that seems to be reusable from a design viewpoint may not really be reusable if these issues are addressed. Reusability is the ability of a piece of software to be reused in the future.

[2] LITERATURE REVIEW

Earlier this year, Sonal Geholt [1] published a comparison of several complexity measures created by various writers. The complexity of component-based software is measured using a variety of metrics, including instance variables, instance methods, control flow, and interface techniques, among others. The comparison is based on a number of quality parameters, such as maintainability, Integrity, complexity, testability, customizability, and so on.

In 2014, Chander Diwaker [2] resulted in increased production, greater quality, reduced development time and costs. Metrics for software quality play a significant role in assessing and enhancing the overall quality of software products. Software development and deployment methodologies are guided by these metrics. To achieve quality and manage risk in a component-based system, metrics are used to assess the elements that influence risk and quality.

Component reuse and cost-savings advantages of component reusability were highlighted by Lovepreet Kaur [3] in a paper published in 2015. Developers that want to create reusable software products or salvage useable components from old software might follow these guidelines. CED-adopting companies may also reap the benefits of these principles for quality and productivity development.

[4] Various aspects of CBSE with its associated metrics were outlined by Kiran Narang in 2018. Using metrics, a developer is able to determine the risk factor and lower it prior to the building of a system. They require metrics to evaluate the quality, reusability, functionality, portability, and understandability of each component they choose from the market. Many scholars have established many measures for CBSE. Only a handful of them have any practical use.

Component-based metrics were the focus of Sakshi Patel's [5] research in 2016. On the basis of both functional and non-functional features of software, this article compares several component-based metrics and discusses how this new method differs from any previous strategy for software development.. Component-based metrics are designed to promote reusability and save development costs and time. Quality and risk management are assessed using these measures.

Using black box components, Sachin Kumar [6] suggested a method in 2014 for determining the coupling complexity of software. On the basis of the interconnections between components, the suggested metric is presented. Traditional metrics, on the other hand, don't apply to a black box component since the component's source code is unavailable. Components' black box nature makes it impossible to gauge software's complexity.

Software matrixes were developed in 2012 by Prakriti Trivedi [7] to verify the relationship between software components and applications. [7] After employing this component, the quality of the programme depends on how strong this relationship is. Finally, the aggregate metrics will provide the ultimate result in terms of the component's application's boundless. When employing these components, the most important question is whether or whether they are advantageous or not. The same question is being addressed in this planned effort.

It was in 2012 that Majdi Abdellatief [8] published a thorough mapping analysis of different metrics that had been developed to quantify the quality of CBSS and its components. [8] Seventeen ideas may be used to assess CBSSs as a whole, while fourteen proposals could be used to assess particular components without the others. Software components that were measured are evaluated and explained in detail. A small number of the measures that have been put out are well defined. The original studies' quality evaluation found several flaws and provided advice for how to strengthen and broaden the acceptability of metrics. Although it is difficult to quantify a CBSS and its components, this remains a problem. So much work must be put into developing a better method of evaluating in the future.

According to Majdi Abdellatief [9], structural design of Component-based software systems was one of Majdi's primary concerns in 2012. (CBSS). Based on the notion of Component Information Flow, two sets of metrics, namely Component Information Flow Metrics and Component Coupling Metrics, are provided. We also explore the reasons for and potential usage of system- and component-level metrics. The suggested measures seem to be quite intuitive, according on preliminary findings from our ongoing empirical study.

This paradigm was first proposed by Danail Hristov [10] in 2012, with the goal of organizing our collection of reusability indicators for component-based software development. The lack of a documented paradigm for describing software reusability and establishing relevant metrics has been frustrating. However, the adoption of component reuse in software development will be simplified and accelerated if a thorough understanding of reusability and appropriate and simple metrics for quantifying reusability are provided.

Table 1 Literature survey

S no.	Author / year	Title	Methodology	Objectives
1	Sonal Gehlot / 2019	Complexity Metrics for Component Based Software — A Comparative Study	CBSE Matrics	To perform comparative analysis of CBSE metric
2	Chander Diwaker / 2019	Metrics Used In Component Based Software Engineering	CBSE matrix	To consider the metrics used in software engineering
3	Lovepreet Kaur / 2015	Quality Enhancement in Reusable Issues in Component – Based Development	Software Matrics	To improve quality for reusable components.
4	Kiran Narang / 2018	Comparative Analysis of Component Based Software Engineering Metrics	Software Matrics	To perform comparative analysis of CBSE metrics
5	Sakshi Patel / 2016	A Study of Component Based Software System	Software system	Considering need and scope of CBSE
6	Sachin Kumar / 2014	Coupling Metric to Measure the Complexity of Component Based Software through Interfaces	CBSE Matrics	Considering coupling metric to check the complexity of CBSE
7	Prakriti Trivedi / 2012	Software Metrics to Estimate Software Quality using Software Component Reusability	Software Metrics	Calculating quality of software with support of component reusability.
8	Majdi Abdellatief / 2012	A mapping study to investigate component-based software system metrics	Software Metrics	To study need of component based metrics

9	Majdi Abdellatief / 2012	Component-based Software System Dependency Metrics based on Component Information Flow Measurements	CBSE Matrics	Considering dependency of metrics over component information flow
10	Danail Hristov / 2012	Structuring Software Reusability Metrics for Component-Based Software Development	Software Metrics	To structure the metrics for reusability

[3] CBSE MATRIX

Component based software metrics are those metrics which will measure the quality and manage the risk of component based software. To build an efficient metrics, first identify the main characteristics of work. After that divide them or break down into sub-characteristics [15]. Then these refined sub characteristics are appearing in attributes. These attributes based on metric definitions are used to get required metrics.

A. Metric Suite

This metrics structure is represented in the form of tree. In component based software engineering there are two types of metrics: 1) Non-functional Metrics 2) Functional Metrics[16-32].

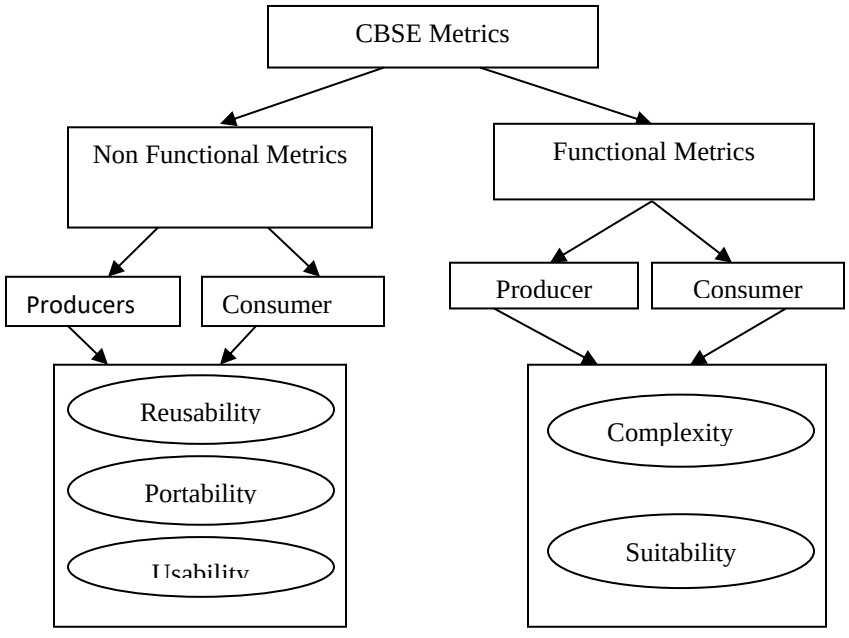

Fig 2 CBSE Matrices representation

In this we are having various metrics based on various dimensions.

i) Suitability Metrics: The degree to which components fulfill the confine requirement. The component suitability is the nature that can be determined after the component gets installed [16]. For suitability there are two types of metrics based on different perspective[32-48]:

Required Functionality (RF): It comes under producer perspective. In this only required functionality need to be checked that must be satisfied.

$$\mathrm{RF} = \frac{\text{No.of useful functionality components that are provided}}{\text{Total count of functionalities required by the CBSE}}$$

Increase in the value of RF will increase suitability of component.

Extra Functionality (EF): It comes under consumer perspective. In this extra functionalities are needed to be checked.$\mathrm{EF} = \frac{\text{No.of extra functionalities given by the Components}}{\text{Total no.of functionalities necessary by component based system}}$

As the value of EF decrease the suitability will increase because increase in EF will increase the unwanted functionalities.

ii) Complexity: complexity of software depends upon its complexity attributes such as coupling, cohesion etc. The quality of software components, its interfaces and specifications are computed by complexity metrics. The more demand of quality will automatically increase the complexity of component. In this one metrics is based on producer perspective and two for consumer's perspective.

Component coupling (COC): It comes under producer perspective. In this internal structure of component is checked i.e. classes and relationship between them.

$$\mathrm{COC} = \frac{\text{No. of other components sharing attribute or methods}}{\text{Total No. of possible sharing pairs in the component} - \text{based application}}$$

Interface complexity: It comes under consumer perspective. The quality characteristics such as usability, portability, performance and reusability are evaluated by complexity metrics [5]. More complex interface from user point of view will create testing and debugging problem. In this there are two metrics:

Constraints complexity (CTC):

$$\mathrm{CTC} = \frac{\text{No. of constraints}}{\text{No. of properties and operation in an Interface}}$$

Configuration Complexity (CFC):

$$\text{CFC} = \frac{\text{No. of configuration}}{\text{No. of context of use of the components}}$$

iii) Component coupling complexity metrics for black box component:

$$\text{CCCM (BB)} = \text{FICM (BB)} + \text{FOCM (BB)}$$

Where FICM (BB) is Fan-in complexity Metric used to compute the coupling complexity due to received information from additional component and FOCM is fan-out complexity metrics which is used to compute the complexity because of leaving information

iv) Reusability: The degree to which a component can be used reused by software and some given application. It is the quality of software to improve productivity. There are various sub factors of reusability as shown in figure.

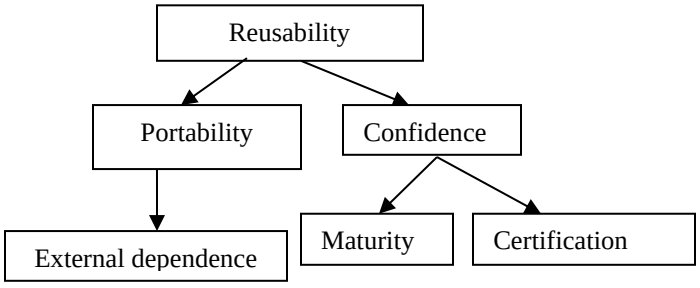

Figure 3 Component Reusability Tree

Portability: In this external dependency is evaluated.

$$\text{ED} = \frac{\text{Portability: In this external dependency is evaluated.}}{\text{Total No. of methods (Read/ Write)}}$$

Confidence: In this maturity level of reusable component is calculated

$$\text{Mat} = \text{DF} + \text{CR}$$

DF= No. of faults detected
CR= No. of changes Requests

[4] PROPOSED MODEL

In proposed model two different programming modules have been considered to compare the required functionality, extra functionality, component coupling, constraints complexity, configuration complexity. In proposed work the equations used to find the functionality, coupling, constraint complexity configuration complexity metric have been considered.

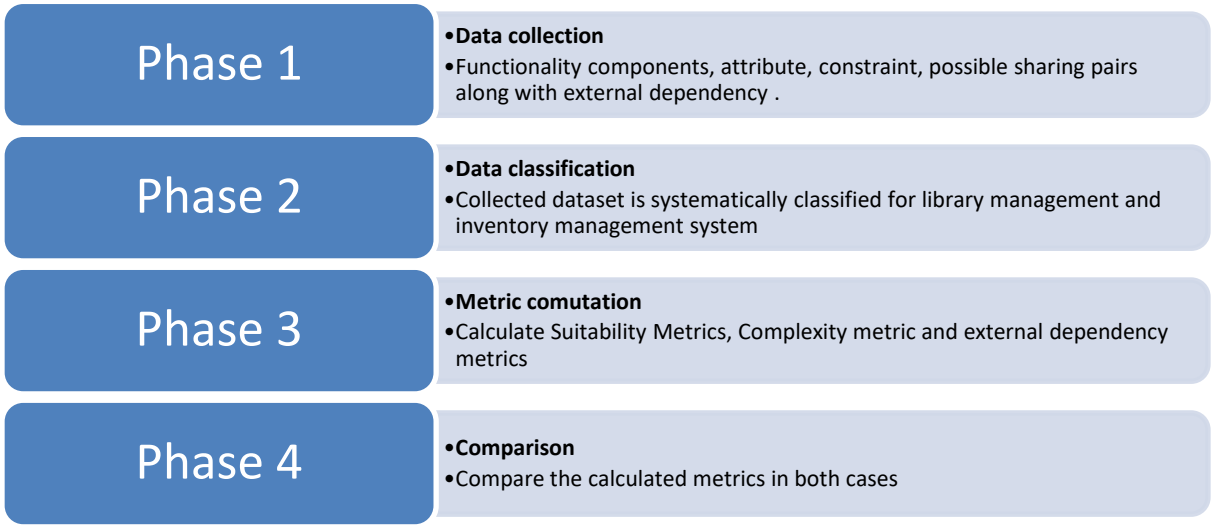

Fig 4 Process flow of proposed work

Table 2 Chart of functionality components, attribute, constraint, possible sharing pairs along with external dependency in case of library management and inventory management system.

	Library management Programming module	Inventory management Programming module
Count of useful functionality components that are provided	30	45
Total count of functionalities required by the CBSE	50	60
Count of extra functionalities given by the Components	10	9
Total count of functionalities necessary by component based system	15	10
Count of other components sharing attribute or methods	14	18
Total count of possible sharing pairs in the component-based application	20	25
Count of constraints	24	35
Count of properties and operation in an Interface	40	45
Count of configuration	5	6
Count of context of use of the components	10	10

Portability of external dependency	17	23
Total count of methods (Read/ Write)	80	85

[5] RESLUT AND DISCUSION

This section has focused on finding the suitability, complexity, external dependency for library management and inventory management.

5.1 SUITABILITY METRIC

Simulation of Suitability metric for Library management Programming module

$$\text{RF (a)} = \frac{\text{No.of useful functionality components that are provided}}{\text{Total count of functionalities required by the CBSE}}$$

RF=30/50=0.6

$$\text{EF (a)} = \frac{\text{No. of extra functionalities given by the Components}}{\text{Total no. of functionalities necessary by component based system}}$$

EF=10/15=0.667

Simulation of Suitability metric for Inventory management Programming module

$$\text{RF (b)} = \frac{\text{No.of useful functionality components that are provided}}{\text{Total count of functionalities required by the CBSE}}$$

RF= 45/60=0.75

$$\text{EF (b)} = \frac{\text{No. of extra functionalities given by the Components}}{\text{Total no. of functionalities necessary by component based system}}$$

EF=9/10=0.9

5.2 COMPLEXITY METRIC

Simulation of Complexity metric **for** Library management Programming module

Component coupling (COC):

$$\text{COC (a)} = \frac{\text{No.of other components sharing attribute or methods}}{\text{Total No.of possible sharing pairs in the component-based application}}$$

COC=14/20=0.7

Constraints complexity (CTC):

$$\text{CTC (a)} = \frac{\text{No. of constraints}}{\text{No. of properties and operation in an Interface}}$$

CTC=24/40=0.6

Configuration Complexity (CFC):

$$\text{CFC (a)} = \frac{\text{No. of configuration}}{\text{No. of context of use of the components}}$$

CFC=5/10=0.5

Simulation of Complexity metric **for Inventory** management Programming module

Component coupling (COC):

$$\text{COC (b)} = \frac{\text{No.of other components sharing attribute or methods}}{\text{Total No.of possible sharing pairs in the component}-\text{based application}}$$

COC=18/25=0.72

Constraints complexity (CTC):

$$\text{CTC (b)} = \frac{\text{No. of constraints}}{\text{No. of properties and operation in an Interface}}$$

CTC=35/45=0.778

Configuration Complexity (CFC):

$$\text{CFC(b)} = \frac{\text{No. of configuration}}{\text{No. of context of use of the components}}$$

CFC=6/10=0.6

5.3 EXTERNAL DEPENDENCY

External dependency for Library management Programming module

$$\text{ED(a)} = \frac{\text{Portability: In this external dependency is evaluated.}}{\text{Total No. of methods (Read/ Write)}}$$

ED=17/80=0.2125

External dependency for Inventory management Programming module

$$\text{ED(b)} = \frac{\text{Portability: In this external dependency is evaluated.}}{\text{Total No. of methods (Read/ Write)}}$$

ED=23/85=0.27

5.4 COMPARATIVE ANALYSIS

This section has presented comparative analysis of required functionality, extra functionality,

Table 3 Comparative analysis of programming Module for RF

Programming Module	Required Functionality (RF)
Library Management	0.60
Inventory Management	0.75

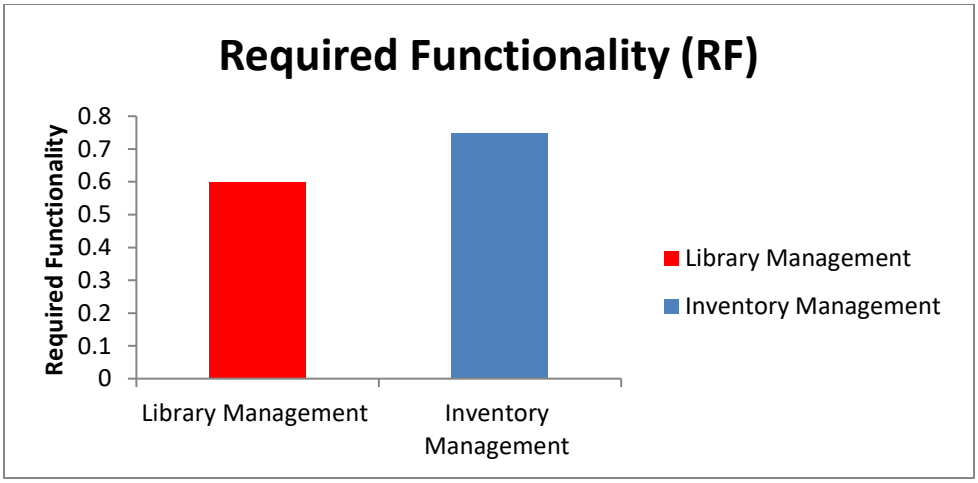

Fig 5 Comparative analysis of programming Module for Required functionality

Table 4 Comparative analysis of programming Module for EF

Programming module	Extra Functionality (EF)
Library Management	0.67
Inventory Management	0.90

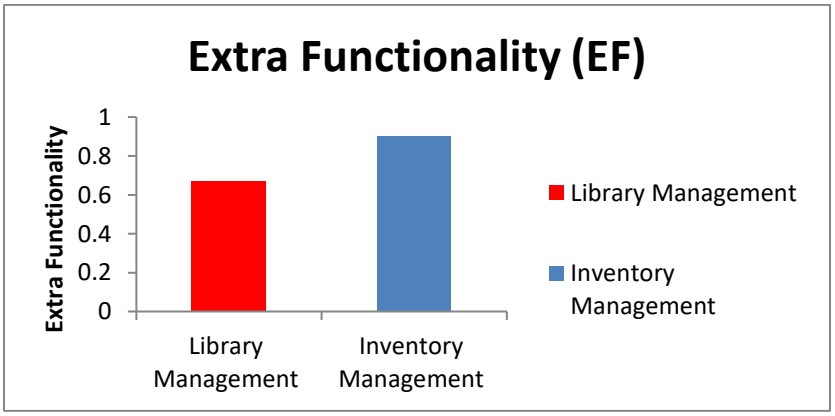

Fig 6 Comparative analysis of programming Module for Extra Functionality

Table 5 Comparative analysis of programming Module for Component coupling

Programming module	Component Coupling (COC)
Library Management	0.70
Inventory Management	0.72

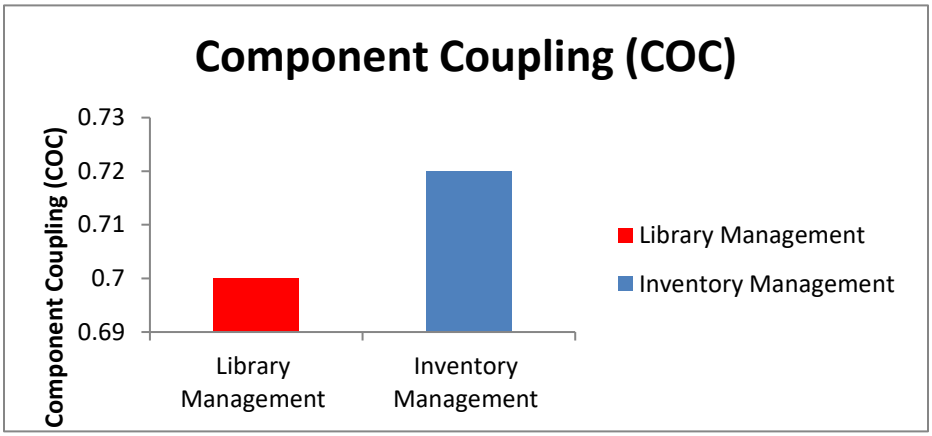

Fig 7 Comparative analysis of programming Module for COC

Table 6 Comparative analysis of programming Module for CTC

Programming Module	Constraints Complexity (CTC)
Library Management	0.60
Inventory Management	0.78

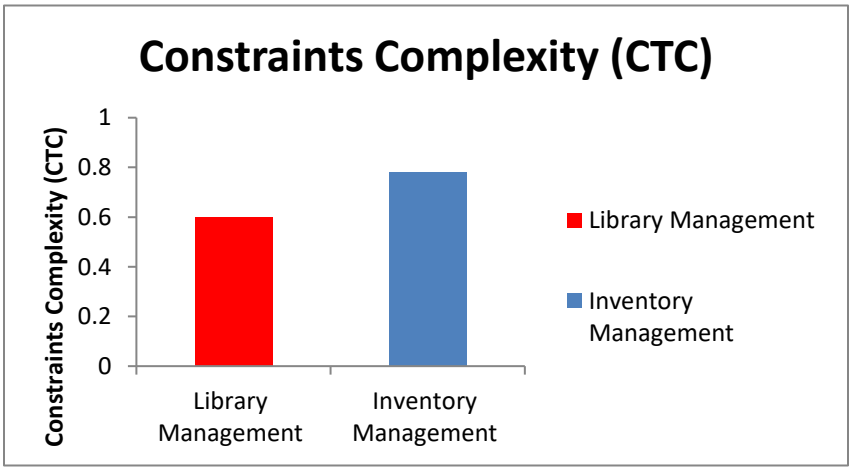

Fig 8 Comparative analysis of programming Module for CTC

Table 7 Comparative analysis of programming Module for CFC

Programming Module	Configuration Complexity (CFC)
Library Management	0.50
Inventory Management	0.60

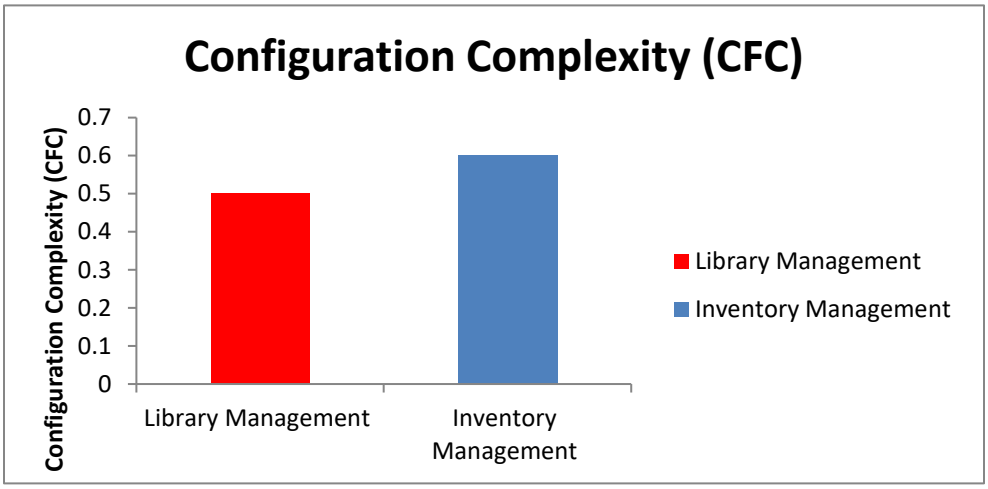

Fig 9 Comparative analysis of programming Module for EF

Table 8 Comparative analysis of programming Module for ED

Programming Module	External Dependency (ED)
Library Management	0.21
Inventory Management	0.27

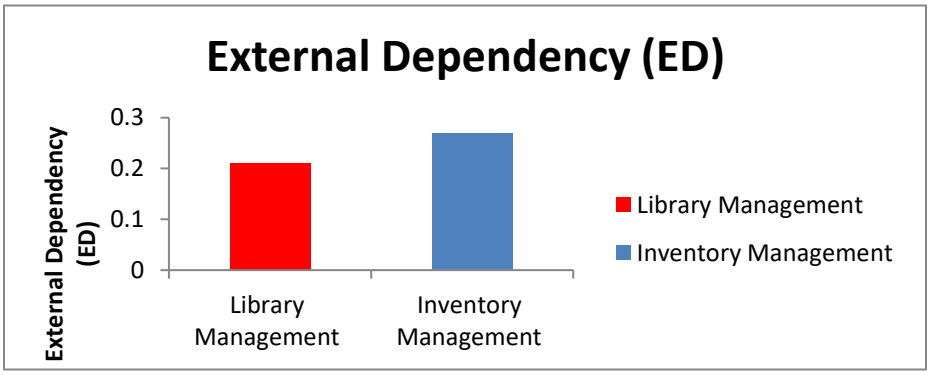

Fig 10 Comparative analysis of programming Module for ED

[6] CONCLUSION

In proposed work, we are representing the simulation process at different metrics. At Suitability metric, simulation of RF is 0.60 and EF is 0.67 for Library management Programming module and RF is 0.75 and EF is 0.9 for Inventory management Programming module. At Complexity Metric, Component coupling (COC) is 0.70, Constraints complexity (CTC)is 0.60 and Configuration Complexity (CFC) is 0.72 for Library management Programming module and Component coupling (COC) is 0.778, Constraints complexity (CTC)is 0.60 and Configuration Complexity (CFC) is 0.60 for Inventory management Programming module. At External Dependency, for Library management Programming module is 0.2125 and for Inventory management Programming module is 0.27.

REFFERENCES

1. Sonal Gehlot, Pooja Rana, Rajender Singh, (2019), Complexity Metrics for Component Based Software — A Comparative Study. doi: 10.17706/jcp.14.6 389-396
2. Chander Diwaker , Sonam Rani2 , Pradeep Tomar , (2014), Metrics Used In Component Based Software Engineering. IJITKM Special Issue (ICFTEM-2014) May 2014 pp. 46-50 (ISSN 0973-4414).
3. Lovepreet Kaur, (2015), Quality Enhancement in Reusable Issues in Component – Based Development. ISSN: 2393-9028 (PRINT) | ISSN: 2348-2281 (ONLINE), IJRECE VOL. 3 ISSUE 1 JAN-MAR 2015.
4. Kiran Narang, Dr. Puneet Goswami, (2018), Comparative Analysis of Component Based Software Engineering Metrics, DOI: 10.1109/CONFLUENCE.2018.8443016.
5. Sakshi Patel, Jagdeep Kaur, (2016), A Study of Component Based Software System. International Conference on Computing, Communication and Automation (ICCCA2016).
6. Sachin Kumar, Pradeep Tomar, Reetika Nagar, Suchita Yadav,"Coupling Metric to Measure the Complexity of Component Based Software through Interfaces", April 2014
7. Prakriti Trivedi, Rajeev Kumar, "Software Metrics to Estimate Software Quality using Software Component Reusability", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 2, No 2, March 2012.
8. Majdi Abdellatief∗, Abu Bakar Md Sultan, Abdul Azim Abdul Ghani1, Marzanah A. Jabar, "A mapping study to investigate component-based software system metrics", jo u rn al homepage: www.elsevier.com, 5 October 2012.
9. Majdi Abdellatiefab, Abu BakarMd Sultana, Abdul AzimAbdGhania, Marzanah A.Jabara, "Component-based Software System Dependency Metrics based on Component Information Flow Measurements" ICSEA: The Sixth International Conference on Software Engineering Advances, 2012.
10. Danail Hristov, Oliver Hummel, Mahmudul Huq, Werner Janjic, "Structuring Software Reusability Metrics for Component-Based Software Development", ICSEA: The Seventh International Conference on Software Engineering Advances, 2012.

11. N. Gehlot and J.Kaur, "Dynamic Inheritance Coupling Metric-Design and Analysis for Assessing Reusability", Int. J. Software Engineering Technology and Applications, Vol.1, No.1, PP. 118-133, 2015

12. V. Subedha, S. Sridhar, "Design of Dynamic Component Reuse and Reusability Metrics Library for Reusable Software Components in Context Level", February 2012.

13. Pooja Rana Rajender Singh, "A Study of Component Based Complexity Metrics", November 2014.

14. V. L. Narasimhan and B. Hendradjaya, "A New Suite of Metrics for the Integration of Software Components", 2007

15. P. Edith Linda, V. Manju Bashini, S. Gomathi, "Metrics for Component Based Measurement Tools", International Journal of Scientific & Engineering Research Volume 2, Issue 5, May-2011.

16. Vinay Tiwari, Dr. R.K. Pandey, "Open Source Software and Reliability Metrics", International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 10, December 2012.

17. A.Aloysius and K.Maheswaran, "A Review on Component Based Software Metrics", 22 January 2015

18. K.P. Srinivasan1 and T. Devi, "Software Metrics Validation Methodologies In Software Engineering", International Journal of Software Engineering & Applications (IJSEA), Vol.5, No.6, November 2014.

19. Gurdev Singh, Dilbag Singh, Vikram Singh, "A Study of Software Metrics", International Journal of Computational Engineering & Management, Jan 2011, Vol. 11, pp. 22-27.

20. Abhikriti Narwal, "Empirical Evaluation of Metrics for Component Based Software Systems", International Journal of Latest Research in Science and Technology, Dec 2012, Vol 1, Issue 4, pp. 373-378.

21. Sidhu Pravneet, "Quality metrics Implementation in Component based Software Engineering using AI Back Propagation Algorithm Software Component", International Journal of Engineering and Management Sciences, 2012, Vol. 3(2), pp. 109-114.

22. Kshirsagar, P. R., Chippalkatti, P. P., & Karve, S. M. (2018) Optimum Spread for Generalized Regression Neural Network using Particle Swarm Intelligence, Jour of Adv Research in Dynamical & Control Systems, Vol. 10, 04-Special Issue, 2018

23. Taranjeet Kaur, Rupinder Kaur, "Comparison of various Lacks of Cohesion Metrics", International Journal of Engineering and Advanced Technology, Feb 2013, Vol. 2, Issue 3, pp. 252-254.

24. Divya Chaudhary, Prof. Rajender Singh Chillar, "Component Base Software Engineering Systems: Process and Metrics", International Journal of Advanced Research in Computer Science and Software Engineering, July 2013, Vol. 3, Issue 7, pp. 91-95.

25. Umesh Kumar Tiwari and Santosh Kumar, (2021), Component-Based Software Engineering Methods and Metrics. 2021 Taylor & Francis Group, LLC CRC Press is an imprint of Taylor & Francis Group, LLC.

26. Kshirsagar, P. R., Chippalkatti, P. P., & Karve, S. M. (2018). Performance optimization of neural network using GA incorporated PSO. Journal of Advanced Research in Dynamical and Control Systems, 10(4).
27. S. Sundaramurthy, C. Saravanabhavan, and P. Kshirsagar, "Prediction and classification of rheumatoid arthritis using ensemble machine learning approaches," in 2020 International Conference on Decision Aid Sciences and Application (DASA), pp. 17–21, Sakheer, Bahrain, 2020.
28. Kshirsagar, Pravin R., et al. "Automation Monitoring With Sensors For Detecting Covid Using Backpropagation Algorithm." KSII Transactions on Internet and Information Systems (TIIS) 15.7 (2021): 2414-2433.
29. Jude, A.B., Singh, D., Islam, S. et al. An Artificial Intelligence Based Predictive Approach for Smart Waste Management. Wireless PersCommun (2021). https://doi.org/10.1007/s11277-021-08803-7.
30. B. Prabhu Kavin, Sagar Karki, S. Hemalatha, Deepmala Singh, R. Vijayalakshmi, M. Thangamani, Sulaima Lebbe Abdul Haleem, Deepa Jose, Vineet Tirth, Pravin R. Kshirsagar, Amsalu Gosu Adigo, "Machine Learning-Based Secure Data Acquisition for Fake Accounts Detection in Future Mobile Communication Networks", Wireless Communications and Mobile Computing, vol. 2022, Article ID 6356152, 10 pages, 2022. https://doi.org/10.1155/2022/6356152
31. Pravin Kshirsagar et.al (2016), "Brain Tumor classification and Detection using Neural Network", DOI: 10.13140/RG.2.2.26169.72805
32. Pravin Kshirsagar et. al., "OPERATIONAL COLLECTION STRATEGY FOR MONITORING SMART WASTE MANAGEMENT SYSTEM USING SHORTEST PATH ALGORITHM", Journal of Environmental Protection and Ecology, Vol. 22, Issue 2, pp. 566-577,2021
33. M. A. BERLIN et. al., "NOVEL HYBRID ARTIFICIAL INTELLIGENCE-BASED ALGORITHM TO DETERMINE THE EFFECTS OF AIR POLLUTION ON HUMAN ELECTROENCEPHALOGRAM SIGNALS", Journal of Environmental Protection and Ecology 22, No 5, 1825–1835 (2021)
34. M. ABUL HASAN et. al., "INTERNET OF THINGS AND IT'S APPLICATION IN INDUSTRY 4.0 FOR SMART WASTE MANAGEMENT" Journal of Environmental Protection and Ecology 22, No 6, 2368–2378 (2021).
35. Oza S. et al. (2020) IoT: The Future for Quality of Services. In: Kumar A., Mozar S. (eds) ICCCE 2019. Lecture Notes in Electrical Engineering, vol 570. Springer, Singapore. https://doi.org/10.1007/978-981-13-8715-9_35
36. Sathawane, N.K.S., Kshirsagar, P.: Prediction and Analysis of ECG Signal Behaviour using Soft Computing, International Journal of Research in Engineering & Technology, Vol. 2, Issue 5, pp. 199, May (2014)
37. P. Kshirsagar and S. Akojwar, "Classification & Detection of Neurological Disorders using ICA & AR as Feature Extractor", Int. J. Ser. Eng. Sci. IJSES, vol. 1, no. 1, Jan. 2015.

38. Pravin Kshirsagar, Dr.Sudhir Akojwar, “Classification and Prediction of Epilepsy using FFBPNN with PSO”, IEEE International Conference on Communication Networks, 2015.
39. P. Kshirsagar, S. Akojwar, Nidhi D. Bajaj , “A hybridised neural network and optimisation algorithms for prediction and classification of neurological disorders” International Journal of Biomedical Engineering and Technology ,vol. 28,Issue 4,Pp. 307-321,2018.
40. P. Kshirsagar and S. Akojwar, "Novel approach for classification and prediction of non linear chaotic databases," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016, pp. 514-518, doi: 10.1109/ICEEOT.2016.7755667,2016
41. Kshirsagar, P.R., Akojwar, S.G., Dhanoriya, R, “ Classification of ECG-signals using artificial neural networks”, In: Proceedings of International Conference on Intelligent Technologies and Engineering Systems, Lecture Notes in Electrical Engineering, vol. 345. Springer, Cham (2014).
42. P. Kshirsagar and S. Akojwar, “Optimization of BPNN parameters using PSO for EEG signals,” ICCASP/ICMMD-2016. Advances in Intelligent Systems Research. Vol. 137, Pp. 385-394,2016
43. Pravin Kshirsagar, Nagaraj Balakrishnan & Arpit Deepak Yadav “Modelling of optimised neural network for classification and prediction of benchmark datasets” , Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization, 8:4, 426-435, DOI: 10.1080/21681163.2019.1711457,2020
44. Dr. Sudhir Akojwar, Pravin Kshirsagar, Vijetalaxmi Pai “Feature Extraction of EEG Signals using Wavelet and Principal Component analysis”, National Conference on Research Trends In Electronics, Computer Science & Information Technology and Doctoral Research Meet, Feb 21st & 22nd ,2014.
45. S. Akojwar and P. Kshirsagar, “A Novel Probabilistic-PSO Based Learning Algorithm for Optimization of Neural Networks for Benchmark Problems”, Wseas Transactions on Electronics, Vol. 7, pp. 79-84, 2016.
46. Sudhir G. Akojwar, Pravin R. Kshirsagar, “ Performance Evolution of Optimization Techniques for Mathematical Benchmark Functions”. International Journal of Computers, **1**, 231-236,2016.
47. Pravin Kshirsagar And Sudhir Akojwar “Hybrid Heuristic Optimization for Benchmark Datasets”, International Journal of Computer Applications (0975 – 8887) Volume 146 – No.7, July 2016
48. Kshirsagar, P., Akojwar, S.: ‘ Classification of human emotions using EEG signals’, Int. J. Comput. Sci., 2016, 146, (7), pp. 17– 23