*International Journal of Research in Advent Technology, Vol.7, No.5S, May 2019*
*E-ISSN: 2321-9637*
*Available online at www.ijrat.org*

# A New Proposal with a Cohesion Metric for Finding Complexity of COTS Components

Dr. Arvind Kumar[1], Ms. Jyoti sharma[2]
[1]*Assistant Professor, Department of Computer Science and Engineering, SRM University, Delhi-NCR, Sonepat, Haryana INDIA, k.arvind33@gmail.com*
[2]*Assistant Professor, Department of Computer Science and Engineering, MAIT, Rohini, Delhi Kaushik.jyoti27@gmail.com*

**Abstract—**In today's world we are living with a high technological environment surrounded with full of software's. The software's which we are using must be of good quality and of less complexity. The complexity of the software is highly dependent on the design of the software that we are using and the design is directly proportional to concept of coupling and cohesion. In this paper, we are giving the usage of already developed cohesion metric in finding the complexity of commercialized websites or software's by calculating the complexity of individual components.

**Keywords—**component, CBSD (Component based software engineering), COTS

## 1. INTRODUCTION

CBSD is basically the development of the software from the reusable number of components. The design and formation of software in this approach is totally based on the number of reusable components. The CBSE idea is not new. It was given by Mcllory more than forty years ago. CBSE came in to existence due to some of the problems faced by the object oriented paradigm. If any of the development process starts from scratch, then it is more time consuming as compared to the CBSE where the components are the reusable entities and the development process is not starting from the scratch.

There was also a contribution in automated testing of components as well [25]. Kaur reported that Software complexity cannot be removed completely but it can be controlled. But, for controlling the software complexity effectively, software complexity metrics are required to measure it. So many researchers have proposed the various metrics for evaluating and predicting software complexity. This journal describes various metrics which may be applied for measuring component complexity [26]. Sirohi have investigated that when a software system was developed to ensure the quality, reliability, robustness and functionality of system testing is necessary. This paper was all about the study of the component based software engineering and simultaneously some of the testing techniques. Some of the testing techniques have been covered in this paper so that in future any kind of testing techniques can be used according to the requirement of the situation [27].

## 2. PROBLEM

A Various number of metrics are available for the calculation of complexity of the software but the metrics are not capable of finding the complexity of all types of components. The calculation of complexity of each of the component is considered to be important for the future considerations of the component in development or up gradation of the software. The main problem is to select the components having which are less complex, helps in achieving the high performance and which improves the overall quality of system.

## 3. RELATED WORK

Some of the researchers have proposed the algorithm for the efficient retrieval of component from the component repositories using the Ontology based component retrieval method and also reviewed the twenty-eight years of CBSE. A many number of papers for the detailed literature survey for the CBSE has been considered by the many number of researchers. Gao and his co team members investigated the concept of traceable components, demonstrated that how the various properties of components are investigated to make a component to be traceable so that it can be easily identified and used during the development process. The solution given by him is considered to be very important as it were supporting the third party software components and check various component behaviors [17].
After that the class level metrics has been designed and its scope includes the complicated component based system and the model on which the whole procedure is based is state based model. Under this chidamber metric suite the six metrics has been proposed. For the validation of these metrics the hypothetical examples were used [18].In 1993 the revised metric set has been proposed [19].In the proposed set they have considered the facts and the ancestor classes as well. In 1999 the design level class cohesion metrics came in to existence in which the metrics are based on path based model [20].
The complexity metric suite is designed in 2001 in which the metrics are based on static model. In 2003 the washizaki metric set for the java components have been designed in

which the component observability and component customizability has been rated. In 2016 the suitability metric based on state based model and having scope of complicated component based system has been designed for measuring the functionality of the component given by the components with a ratio of functionality of the component required to deliver a system perfectly.

**Importance of A Metric In Calculation of The Complexity of Software**

A wide range of metrics, models are available for calculating the complexity and reliability, maintainability of the system and as well as for the CBS. But for consideration of many numbers of components, a different approach is required. Determining the component reliability and complexity before integrating it in to the system are very important factors to be calculated. However the calculation of the complexity of the component after integrating in to software is also considered being important for the further up gradations. This research paper is a small step to check or to calculate the complexity of the component by using the already developed metric based on the concept of coupling and cohesion and the reusable process approach. Although there are many published articles addressing the software complexity and reliability metrics in component based programs, very few papers address the calculation of complexity of component related to commercial website. Coupling represents the degree of interdependence between two software components. The importance of a metric in calculating the complexity of software can be determined by following number of points:-

1. The components with the good reusability ratings can be identified by using the value calculated with the help of defined metric.
2. Great performance of software can be achieved by using the metric related to component complexity calculation.

**General working of pc3m metric with replacable number of parameters**

A wide range of metrics, models are available for calculating the complexity and reliability, maintainability of the system and as well as for the CBS. An already developed metric known as PC3M (Package Cohesion Component complexity metric) is developed to measure the complexity of the software [2]. The metric can be defined as:-

**DUM:-**It is the set of all direct connections between classes and methods.

**NDIUC: -**It is the number of used direct or indirect connections in the case study and it can be calculated as n (n-1)/2 where n is the total number of packages in the case study. The formula of n (n-1)/2 has been taken from the concept of TCC (Tight class cohesion) and LCC (Loose Class Cohesion) metric

Now both the DUM and IUDM are working to support the functionality of the component. In the case of COTS Components

We are not capable of calculating the DUM and IUDM parameters, but the prediction of size of software can be estimated with the SLOC (Source Line of Code) parameter.so the SLOC will be acting as s replaceable parameter here.

The selection of components must be related to the cost of the particular component. Hence, the SLOC must be divided by cost of the appropriate components.

**PC3M**:- Package cohesion component complexity metric.

$$\textbf{PC3M} = \left\{ \begin{array}{ll} 0 & \text{if}(n=0) \\ 1 & \end{array} \right\}$$

Where n is the number of elements (classes, methods)

Other cases:-If n=0, there is no element so no possible relation therefore computed value of PC3M is also 0.

If n=1 means a single element is existing so the relation existing will also be single, hence the value of PC3M is also 1[2].

After dividing the SLOC by the cost of the particular component the functionality of the components must be taken in to concern and the low result values components corresponding to the high functionality values must be chosen for the development of the component based software development.

**Usage Of Pc3m Metric In Finding Complexity Of Cots Components**

For finding the complexity of the COTS components using the replaceable set of parameters in package cohesion component complexity metric, we must first consider the real data set as[21]:-

Table 1: Data Set for COTS Components

| Module | Component | COST | Version2 | | | | | COST | Version3 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | REL | ET | DT | SLOC | Functionality | | REL | ET | DT | SLOC | Functionality |
| 1 | 1 | 10 | 0.92 | 5.9 | 7 | 875 | 0.9 | 9 | 0.91 | 6.4 | 8 | 900 | 0.85 |
| | 2 | 8 | 0.9 | 7.3 | 9 | 925 | 0.83 | 7 | 0.89 | 7.9 | 10 | 1050 | 0.75 |
| | 3 | 9 | 0.91 | 6.4 | 8 | 900 | 0.87 | 8 | 0.9 | 7.3 | 9 | 925 | 0.81 |
| 2 | 1 | 12 | 0.95 | 4.8 | 5 | 800 | 0.93 | 11 | 0.93 | 5.3 | 6 | 850 | 0.92 |
| | 2 | 11 | 0.93 | 5.3 | 6 | 850 | 0.925 | 12 | 0.95 | 4.8 | 5 | 800 | 0.935 |
| 3 | 1 | 10 | 0.92 | 5.9 | 7 | 875 | 0.91 | 9 | 0.91 | 6.4 | 8 | 900 | 0.86 |
| | 2 | 9 | 0.91 | 6.4 | 8 | 900 | 0.88 | 10 | 0.92 | 5.9 | 7 | 875 | 0.905 |
| 4 | 1 | 9 | 0.91 | 6.4 | 8 | 900 | 0.845 | 8 | 0.9 | 7.3 | 9 | 925 | 0.82 |
| | 2 | 9 | 0.92 | 5.9 | 7 | 875 | 0.9 | 11 | 0.93 | 5.3 | 6 | 850 | 0.923 |
| | 3 | 11 | 0.93 | 5.3 | 6 | 850 | 0.928 | 10 | 0.92 | 5.9 | 7 | 875 | 0.915 |
| 5 | 1 | 11 | 0.93 | 5.3 | 6 | 850 | 0.92 | 12 | 0.95 | 4.8 | 5 | 800 | 0.935 |
| | 2 | 12 | 0.95 | 4.8 | 5 | 800 | 0.938 | 13 | 0.96 | 4.2 | 4 | 725 | 0.94 |
| 6 | 1 | 7 | 0.89 | 7.9 | 10 | 1050 | 0.77 | 8 | 0.9 | 7.3 | 9 | 925 | 0.82 |
| | 2 | 6 | 0.87 | 8.2 | 11 | 1100 | 0.7 | 5 | 0.85 | 9.3 | 12 | 1300 | 0.65 |
| | 3 | 8 | 0.9 | 7.3 | 9 | 925 | 0.81 | 7 | 0.89 | 7.9 | 10 | 1050 | 0.75 |
| 7 | 1 | 11 | 0.93 | 5.3 | 6 | 850 | 0.926 | 7 | 0.89 | 7.9 | 10 | 1050 | 0.75 |
| | 2 | 12 | 0.95 | 4.8 | 5 | 800 | 0.925 | 6 | 0.87 | 8.2 | 11 | 1100 | 0.75 |
| | 3 | 9 | 0.91 | 6.4 | 8 | 900 | 0.88 | 9 | 0.91 | 6.4 | 8 | 900 | 0.86 |
| 8 | 1 | 8 | 0.9 | 7.3 | 9 | 925 | 0.815 | 9 | 0.91 | 6.4 | 8 | 900 | 0.875 |
| | 2 | 9 | 0.91 | 6.4 | 8 | 900 | 0.86 | 8 | 0.9 | 7.3 | 9 | 925 | 0.805 |
| | 3 | 7 | 0.89 | 7.9 | 10 | 1050 | 0.78 | 8 | 0.9 | 7.3 | 9 | 925 | 0.8 |
| 9 | 1 | 10 | 0.92 | 5.9 | 7 | 875 | 0.915 | 9 | 0.91 | 6.4 | 8 | 900 | 0.89 |
| | 2 | 11 | 0.93 | 5.3 | 6 | 850 | 0.92 | 12 | 0.95 | 4.8 | 5 | 800 | 0.936 |

Table: 2 Data Set for In-House developed Components

| Module | comp | COST | ET | DT | SLOC | Functionality |
|---|---|---|---|---|---|---|
| 1 | y1 | 9 | 5.9 | 8 | 925 | 0.9 |
| 2 | y2 | 8 | 7.3 | 10 | 935 | 0.85 |
| 3 | y3 | 9 | 5.9 | 8 | 925 | 0.92 |
| 4 | y4 | 10 | 5.3 | 7 | 900 | 0.94 |
| 5 | y5 | 11 | 5.3 | 7 | 860 | 0.95 |
| 6 | y6 | 11 | 5.2 | 7 | 860 | 0.955 |
| 7 | y7 | 11 | 8.2 | 7 | 860 | 0.945 |
| 8 | y8 | 8 | 6.4 | 9 | 935 | 0.87 |
| 9 | y9 | 10 | 5.9 | 8 | 900 | 0.935 |

## 4. RESULTS

The component code for testing of package cohesion component complexity metric has been taken from the website bitbucket.org with the availability of source code. This is a commercial website where the source code of the components used for making the website is already given with the specified module. Considering the code for the component 1 like consumers.py.

```
import jsonfrom channels
import Groupfrom channels.sessions
import channel_session

# Connected to websocket.connect
@channel_session
def ws_connect(message, key):
   # Accept connection
   message.reply_channel.send({"accept": True})

# Connected to websocket.receive
@channel_session
def ws_message(message, key):
   reply = json.dumps({'status':'ok'})
   message.reply_channel.send({"text": reply})

# Connected to websocket.disconnect
@channel_session
def ws_disconnect(message, key):
   message.reply_channel.send({'accept': False})
```

Now applying the already developed replaceable cohesion metric on the above real data set of components in which the SLOC and cost is given.

Number of packages involved=3
Number of classes involved=3
Number of methods involved=2

DUM=3
NDIUC=n(n-1)/2
3(3-1)/2
3
PC3M = DUM/NDIUC

PC3M =1.00
But by using the new replaceable cohesion metric the value comes out to be as:-

PC3M=SLOC/COST
PC3M 879/9
PC3M = 97.2
Similarly for the other case
PC3M = 925/7
PC3M = 132.14
Therefore high value of PC3M will lead to low complexity of software and this metric with the replaceable number of parameters will play an important role in determining the complexity of the software.
From the above calculation it is concluded that the component named as manage.py is having the low value of complexity so that component needs no alteration in further up gradations. But the component named as consumers.py is having the high complexity value so this module or this component needs modification in further consideration for software development. So the component reusability ratings can be given as:-

TABLE I

| S.No | Name of Component | Reusability Ratings |
|---|---|---|
| 1 | Consumers.py | 2 |
| 2 | Manage.py | 1 |

In the same way the reusability ratings can also be given to values calculated using SLOC parameter.The reusability ratings of component consumers.py are having the rating of 2 because the component complexity value is high for this component. Therefore the reusability ratings are inversely proportional to component complexity value.

## 5. CONCLUSION AND FUTURE WORK

The calculation of component complexity considered to be important for the further upgradation or the selection of components from the component repository. The future work involves the calculation of complexity of components for the COTS Components.

## REFERENCES

[1] P. Goswami and J. Sharma "Package Based Cohesion measurement in component based software development", International Journal of Engineering and Technology, vol. 9, pp. 3172-3182, 2017.

[2] A. Sharma, R. Kumar, and P. S. Grover, "Evaluation of complexity for software components", International Journal of Software Engineering and knowledge Engineering, vol. 19, pp. 919-931, 2008.

[3] S. Parthipan, S. S. Velan, and C. Babu, "Design level metrics to measure the complexity across versions of AO Software", Proceedings of International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), pp. 1708-1713, 2014.

[4] S. Dhawan, and kiran, "Software metrics-A tool for measuring complexity", International Journal of Software and Web Sciences, vol. 2, pp. 4-7, 2012.

[5] A. Dixit, and P. C. Saxena, "Umbrella: A new component-based software development model", International Conference on Computer Engineering and Applications, vol. 2, pp. 61-65, 2011.

[6] T. J. McCabe, "A complexity measure", IEEE Transactions of Software Engineering, vol. 4, pp. 308-320, 1976.

[7] Gill, K. Geoffrey, Kemerer, and F. Chris, "Cyclomatic complexity metrics revisited: An Empirical Study of Software Development and Maintenance", Cambridge, mass: centre for information systems research, sloan school of management, Massachusetts Institute of Technology, CSIR WP No. 218, 1990.

[8] A. Yadav, and R. A. Khan, "Development of encapsulated class complexity metric", Procedia Technology, vol.4, pp.754-760, 2012.

[9] P. Niranjan, and C.V G. Rao, "A Model software reuse repository with an intelligent classification and retrieval technique", Scientific and Academic Publishing, vol. 1, pp. 15-21, 2011.

[10] N. Haghpanah, M. S. Moaven, J. Habibi, M. Kargar, and S. H. Yeganeh, "Approximation algorithms for software component selection Problem", Proc. of Asia Pacific Software Engineering Conference, pp. 159-166, 2007.

[11] L. Yu, K. Chen, and S. Ramaswamy, "Multiple parameter coupling metrics for layered component based software", Software Quality Journal, vol. 17, pp. 5-24, 2009.

[12] J. A. Whittaker, "Software invisible users", IEEE Software, vol. 18, pp. 84-88, 2001.

[13] E D. wei, "The software complexity model and metrics for object oriented", China English Academic Journals, pp. 16-18, 2007.

[14] J. Gao, and M. C. Shih, "A Component testability model for verification and measurement", Proceedings of the Annual International Computer Software and Application Conferences (COMPSAC '05), vol. 2, pp. 1-9, 2005.

[15] C. Jones, Applied software measurement: Global Analysis of productivity and quality, McGraw-Hill, New York, 2008.

[16] K. P. Srinivasan, "Unique fundamentals of software measurement and software metrics in software engineering", International Journal of Computer Science and Information Technology, vol. 7, pp. 29-43, 2015.

[17] J. Gao, and E. Y. Zhu "Monitoring software components and component based software", San Jose State University, pp. 402-412, 2000.

[18] S. R. Chidamber, and C. F. Kemerer, "A Metric suite for object oriented design", IEEE Transactions on Software Engineering, vol. 20, pp. 476-449, 1994.

[19] D. Kafura, and S. Henry, "Software quality metrics based on interconnectivity", Journal of Systems and Software, vol. 2, pp. 121-131, 1981.

[20] J. Bansiya, L. Etzkorn, C. Davis, and W. Li, "A class cohesion metric for object oriented designs", Journal of Object- Oriented Program, vol. 11, pp. 47–52, 1999

[21] R.Kaur and S.Arora, "Fuzzy multi criteria approach for component selection of fault tolerant system under consensus recovery block scheme" ICACTA, vol.45,pp. 842-841, 2015.