iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** IV **Month of publication:** April 2017

DOI: http://doi.org/10.22214/ijraset.2017.4057

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Brief Review of Various Metrics to Achieve Reusability and Changeability of a Software

Yasrub Shah[1], Er. Sonali Goel[2]
[1]M.Tech (4th Sem), [2]Lecturer, Department of Computer Science
Haryana Engineering College Jagadhri, Kurushetra University, Haryana, India

Abstract: ***Component Based Software Engineering (CBSE) is one of the trending techniques that specifies upon the design and construction of large and complex software systems. The main objective is to minimize the changeability, complexity, time and error factors but at the same time to achieve reusability. The success of CBSD projects can be ensured by use of various software metrics. Although CBSE is increasingly adopted technique, but however to keep it less complex is still a challenging issue. This paper aims to analyze the various metric methods used to achieve the reusability and changeability of software.***
Keywords: Component Based Software Engineering (CBSE); Component Dependency Graph (CDG); Component Based Software System (CBSS); Software Reusuability; Software Changeability.

I. INTRODUCTION (INTRODUCTION TO CBSE)

Component-Based Software Engineering (CBSE) is a technique that focuses upon the design and construction of computer–based systems using reusable software components. This principle represents an element of "buy, rather than to build" that transfers the importance from programming software to composing software systems (Pressman, 2001). It is also a loom for developing software that relies on software reuse and it emerged from the limitations of object-oriented development to support valuable reuse. It is not possible to assess the behavior and the stability of an application unless it is tested fully. The quality of the application is high when it yields the desired results, which is stable, adaptable and leads to reduce in the cost of maintenance. If a change has been introduced in a component, which has been integrated in an application, to assess the stability of application, the brunt of the change on the whole application has to be determined by the developer [1].The object oriented programs are used in the IT world which is used for secured transactions and mainly convenient. Even though they are widely used, the complex and confusing structure may arise if the program is not properly measured and not properly planned. To overcome this problem various proposed varieties of metric tools which have to be used to measure the criteria of Object oriented programming are developed .Some metric tools measure the particular criteria of the object oriented programs because of which a problem can arise. To overcome the individuality of those tools, migration of them is necessary. To provide the user friendly environment, the main aim is to search some important tools and making them as a single add-on .Each individual tool will measure some constraint to assess the java program. But a single tool will not assess all the constraints. Thus there exist needs of collection of tools to measure the java programs which will gratify all the constraints to be measured [2]. For continuous success of this developmental approach, the evaluation of CBSSs as well as the individual components is an essential research area. To measure the quality of CBSS attributes helps us to better comprehend, evaluate, and control the quality of CBSSs and to isolate weaknesses over the entire software life cycle[3].If a component can be expressed as an independent part of the module/application that can be replaced easily and it provides a distinct function in such a way that it should not affect the working of other modules, that is, they should not have any dependency on the replaced component [7]. The reuse of a component is misconceived several times, so it must be made clear to what a component reuse accounts for and to what a component reuse does not accounts for[5]. A software component is a self-contained piece of software that provides clear purpose, has open interfaces and offers plug-and-play services. It can be defined as a reusable software element such as a function, file, module, class or subsystem **[**8**].** Component Dependency Graph of a CBS is defined as G=(S,D,s,t), is a directed graph, where S is a non empty set of vertices each represents a component in the system,D is a set of dependency edge among the two vertices each represents a direct dependency between components, s is a starting node, t is a terminating node. Figure 1 describes the direct dependency where{ D=(A,B),(B,D),(C,D),(C,B),(C,A),(E,B),(E,D)}.

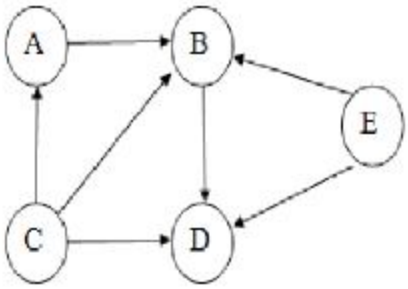


Figure 1: Component dependency graph [6].

A. A New Way to Analyze Dependency in Component Based System (CBS)

This approach contains the following steps

1) Draw a Component Dependency Graph (CDG) of a Component Based System (CBS).
2) A lot weights to every edge of Component Dependency Graph.
3) Analyze the minimum spanning tree for CDG by any one of the existing algorithms (Prim's algorithm or Kruskal's Algorithm).
4) Dependency of the individual component is the minimum weight of that component.

First we evaluate the dependency of each component using Minimum Spanning Tree (MST) in component based system and then evaluate the dependency of each component using Analytical Hierarchal Process. Lastly, we calculate the Correlation Coefficient of the two techniques which shows that the technique is valid [6].

B. Metrics

Software metrics are used to compute the software quality to check whether it satisfies the requirements. Metrics are defined as "Quantifiable measures that could be used to compute the features of a software system or the software development process." Software metrics are essential to plan, predict, monitor, control, evaluate, products and processes. The main objective of the software metrics is to reduce costs, improve quality, Control/ Monitor schedule, small testing effort, many reusable fragments, to better comprehend the quality of the product and the program [2].

C. Existing Metrics

Number of software metrics linked to software complexity and quality assurances has been developed in the past and are still being proposed [6].

D. Metrics for Structured and Object Oriented Systems

Several conventional metrics were designed for structured systems among them developers often found that Wang [9], McCabe's Cyclomatic complexity metric, Halstead's complexity metric and Kafura's and Henry's fan-in, fan-out are most frequently used metrics [10,11,12]. For object oriented systems Chidamber and Kemerer metrics [13] is a foundation of all metrics, Misra [14] recommended Complexity Metric of OOP's Based on Cognitive Weights and many researchers like Arockiam et al. [16,17], Misra et. al [14,15] proposed the various level of metrics of object oriented programs based on their perspective including cognitive phase.

E. Metrics for Component Based Systems

Many researchers like Vernazza et al. expanded the CK metric [18], Salman's [19] considered components, connectors, interfaces, and composition trees as main attributes that found out the structural complexity of a component based system. Bertoa et al. [20] projected the metrics for software components to access their usability, Sharma et al. proposed interface complexity metric for software components by bearing in mind the interface methods and their associated features, arguments types and return types [21].

II. LITERATURE REVIEW

V-Lakshmi, P.T.Parthasarathy, and M .Das (2009) discussed that the Component-Based Software Engineering (CBSE) has shown remarkable prospect in speedy production having superior quality of large software systems, and importance on collapsing of the engineered systems into functional or logical components across components with well defined interfaces which are used for communication. The various metric evaluations which draw so many conclusions include testability, modularity, reusability and

stability of the underlying components. The inferences are argued to be helpful for CBSE-based software development, integration and maintenance [1].

P-Edith Linda, V-Manju Bashini,S-Gomathi (2011) discussed how to integrate the different object oriented metric tools and make them available as a single add-on[2].

Majdi Abdel latief, Abu Bakar Md Sultan, Abdul Azim Abdul Ghani1, Marzanah A. Jabar (2013) discussed that a component-based software system (CBSS) is a software system that has been deployed independently and developed by integrating components. The main aim is to focus on approaches and elements that are used to evaluate the quality of CBSS and components from a consumer point of view as well as to understand, classify and evaluate existing research in component-based metrics [3].

Adnan Khan, Khalid Khan, Muhammad Amir and M. N. A. Khan (2014) discussed that a Component-based development facilitates software reusability, testing and high quality and allow integrating and developing products. The software development cost and time is reduced by use of reusability approach, which speeds up software development by using already developed components [4].

Anshul Kalia, Sumesh Sood (2014) discussed that there are several ways to define reusable software components. The reusable software components own a distinct functionality that does not influence the functionality of other components. It has also been specified accurately that for what the component reuse stands for and for what the component reuse does not stands for. It is required to characterize the components for better reuse. The components can be distinguished on several features that facilitates with the better usage, better retrieval, better understanding and better cataloguing. One can get the assurance of choosing the right component and the ways in which a component can be reused through component classification.[5].

A.Aloysius and K.Maheswaran(2015) discussed that in the technological world every day number of software's are developed and made available in the market, however measuring the complexity as well as quality of the software is still a challenging issue. Component based software is emerging field and now-a- days, most of the software are developed by using the technique of component based software development (CBSD).By use of this technique ,factors like complexity, time and error were reduced so that reusability is achieved. However, the success of the CBSD projects can be ensured only from the metrics that are previously proposed [6].

Table I Comparative study of various metrics for reusability and changeability of software.

Author	Details of various methods				
	Parameters	Year	Tools/Methods	Findings	References
V. Lakshmi Narasimhan , P. T. Parthasarathy, and M.Das.	Complexity, reusability, testablity, modularity, stablity.	2009	Depend(Depend2007)and Metrics(Metrics1.3.6 2007)	1. It generates design quality metrics for each java package 2. It provides graphical visualization 3. Package is operating system independent.	1.
P.Edith Linda, V. Manju Bashini,S.Gomathi.	Software reusability, small testing effort.	2011	JHAWK,LOCC,CO DE COUNTER.	1.Objectoriented paradigms are measured in the program. 2. It will generate the charts and will produce the reports about the program.	2.

Author	Details of various methods				
	Parameters	Year	Tools/Methods	Findings	References
Majdi Abdel latief, Abu Bakar Md Sultan Abdul Azim Abdul Ghani Marzanah A. Jabar	Maintainability, testability, performance and reliability	2013	Systematic mapping review	1.It allows us to identify the relationship between the researchers and the practitioners. 2. It helps practitioners to remain up-to-date with the state-of-the-art.	3.
Adnan Khan, Khalid Khan, Muhammad Amir and M. N. A. Khan	Reliablity,cost efficiency, reusblity, Cohesion reduction	2014	CBSE techniques.	1. It helps to meet the requirements of the customers to deliver the product at a very low cost. 2 It reduces development time.	4.
AnshuKalia, SumeshSood	Characterization of reusable component, storage space required, authentication controls.	2014	Uncontrolled vocabulary and automatic indexing of software components	1. It helps in easy retrieval of reusable components from component repository.	5.
A.Aloysius, K.Maheswaran	Complexity, quality aspect using reusability, dependency and complexity of black box.	2015	CBSD technique.	1. It helps to reduce complexity, time and error factors. 2. It helps to achieve reusablity.	6.

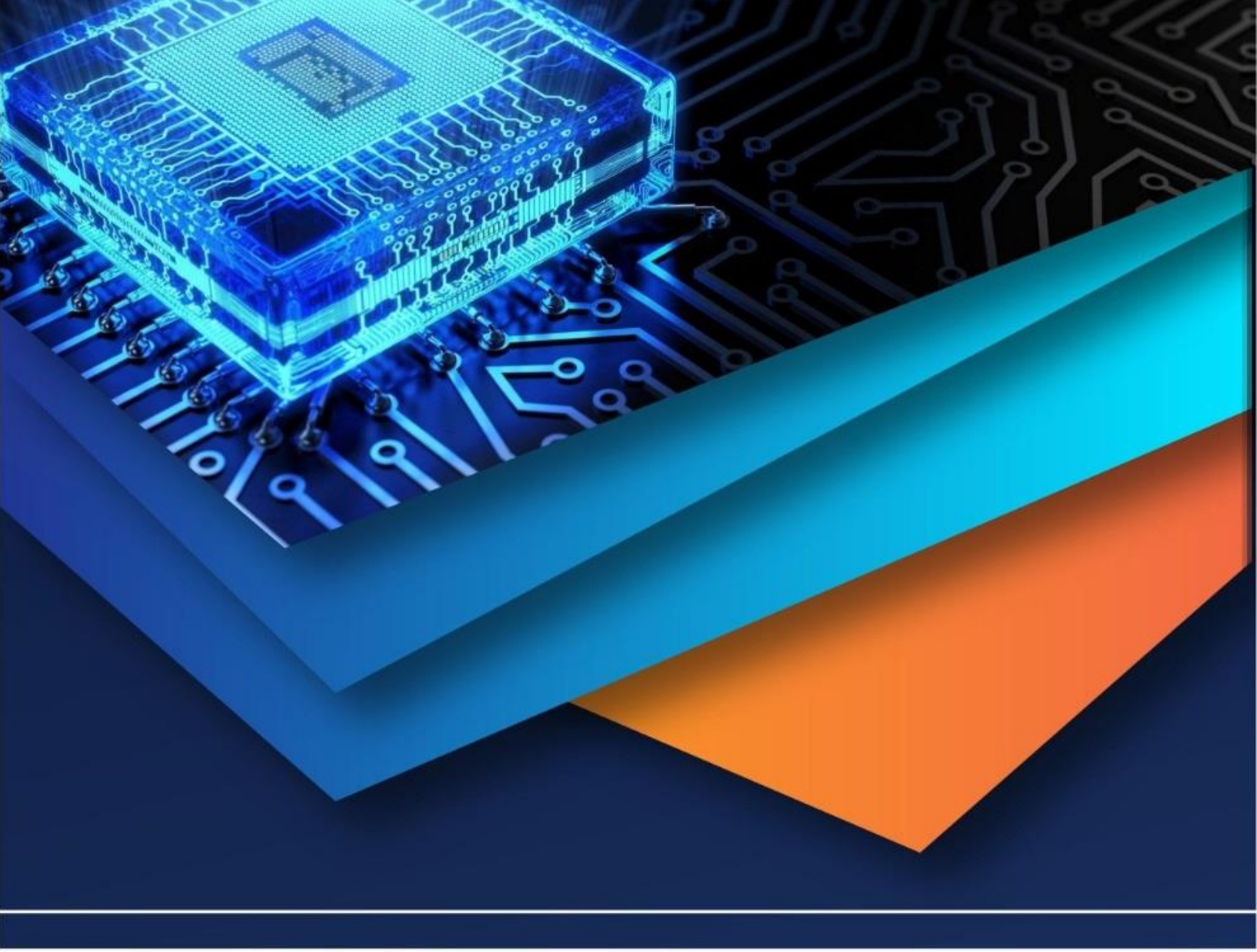
III. CONCLUSION

In this paper we have studied the concept of reusability and changeability and the various metrics to define them. jDepend (Depend 2007)and Metrics(Metric1.3.62007 help to generate design quality metrics for each java package and also provides graphic visualization. JHAWK, LOCC, CODE COUNTER, help to measure object oriented paradigms in the program and also helps to generate the charts. systematic mapping review identifies the relationship between researchers and practitioners and also helps to remain up to date with the state of art. Reliablity,cost efficiency,cohesion reduction using CBSE technique results in meeting the requirements of the customers to deliver the product at a very low cost and reduces the development time. Characterization of reusable components, storage space required and authentication controls using uncontrolled vocabulary and automatic indexing of software components results in easy retrieval of reusuable components from component repository. Complexity, quality aspect using reusability, dependency and complexity of black box, reduces the complexity, time and error factors. In this way, it helps to achieve reusability.

REFERENCES

[1] V. Lakshmi Narasimhan, P. T. Parthasarathy, and M.Das “Evaluation of a Suite of Metrics for Component Based Software Engineering (CBSE)” Issues in Informing Science and Information Technology Volume 6, 2009
[2] 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 ISSN 2229-5518.
[3] Majdi Abdel latief, Abu Bakar Md Sultan, Abdul Azim Abdul Ghani1, Marzanah A. Jabar, “A mapping study to investigate component-based software system

metrics",The Journal of Systems and Software 86 (2013) 587– 603.
[4] Adnan Khan, Khalid Khan, Muhammad Amir and M. N. A. Khan, "A Component-Based Framework for Software Reusability", International Journal of Software Engineering and Its Applications Vol. 8, No. 10 (2014), pp. 13-24 http://dx.doi.org/10.14257/ijseia.2014.8.10.02
[5] Anshul Kalia, Sumesh Sood, " Characterization Of Reusable Software Components For Better Reuse", IJRET: International Journal of Research in Engineering and Technology eISSN: 2319-1163 | pISSN: 2321-7308
[6] A.Aloysius1 and K.Maheswar2, "A Review on Component Based Software Metrics, Intern. J. Fuzzy Mathematical Archive" Vol. 7, No. 2, 2015, 185-194 ISSN: 2320 –3242 (P), 2320 –3250 (online) Published on 22 January 2015.
[7] Yacoub S. et. al, "Characterizing a Software Component", In Proceedings of the 2nd Workshop on Component – Based Software Engineering, in conjunction with ICSE'99, 1999
[8] L.F.Capretz, "A new component-based software life cycle model, Journal of Computer Science", 1(1) (2005) 76-82
[9] Y.Wang and J.Shao," A new measure of software complexity based on cognitive weights", Can. J. Elect. Comput. Eng., 28(2) (2003) 69-74.
[10] McCabe, "A Complexity Measure, IEEE Trans. on Software Engg.", SE-2 (4) (1976) 308-320.
[11] Halstead, "Elements of Software Science", New York: Elsevier North Holland, 1977.
[12] D.Kafura and S. Henry, "Software quality metrics based on interconnectivity, Journal of Systems and Software", 2(2) (1981) 121-131.
[13] S.R.Chidamber and C.F.Kemerer, "A metrics suite for object oriented design, IEEE Transactions on Software Engineering", 20(6) (1994) 476-49.
[14] S.Mishra, "An object oriented complexity metric based on cognitive weights", Proc. 6th IEEE International Conference on Cognitive Informatics (ICCI'07), 2007.
[15] S.Misra, I.Akman and M.Koyuncu, "An inheritance complexity metric for object oriented code:A cognitive approachcode: A cognitive approach, Indian Academy of Sciences", 36(3) (2011) 317–337.
[16] L.Arockiam, A.Aloysius and J.Charles Selvaraj, "Extended weighted class complexity: a new of software complexity for objected oriented systems,Proceedings of International Conference on Semantic E-business and Enterprise computing (SEEC)", pp. 77-80, 2009.
[17] L.Arockiam and A.Aloysius, "On validating class level cognitive complexity metrics, CiiT International Journal of Software Engineering and Technology", 2(3) (2010) 152-157.
[18] T.Vernazza and G.Granatella, "Defining metrics for software components, D Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics", XI (2000) 16-23.
[19] N.Salman, "Complexity metrics as predictors of maintainability and integrability of software components, Journal of Arts and Sciences", 2 (2006) 39-50.
[20] M.F.Bertoa, J.M.Troya and A.Vallecillo, "Measuring the usability of software components, Journal of Systems and Software", 79(3) (2006) 427-439.
[21] A.Sharma, R.Kumar and P.S.Grover, "Evaluation of complexity for software components, International Journal of Software Engineering and Knowledge Engineering", 19(5) (2008) 919-931.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY