IJCSI International Journal of Computer Science Issues, Volume 13, Issue 2, March 2016
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org http://dx.doi.org/10.20943/01201602.1218

Scoped Class Cohesion Metric for Software Process Assessment

Raphael Wanjiku[1], George Okeyo[2] and Wilson Cheruiyot[3]

[1]Computer Information Systems Department, Africa Nazarene University
Nairobi, Kenya
[2,3]Computing Department, Jomo Kenyatta University of Agriculture and Technology
Nairobi, Kenya

Abstract

Class Cohesion is an important software quality that can be used to improve software development process and the software product: process merit assessment and dependable software product. Many Class cohesion metrics measuring the relationship between methods and attributes have been developed and extensively researched. However, the use of relationships among attributes in measuring class cohesion from class scopes has been ignored and the effects of local variables on class cohesion need to be factored in the measurements. This research paper presents a new class cohesion metric that uses attributes relationships within class scopes with data collected using the SCCM software tool that was developed for the purpose this study. The results give higher metric values showing the importance of scoped relationships among these class members while giving a simpler and better interpretation of class cohesion through class attributes interaction.

Keywords: *Class, Cohesion, Attribute, Method, SCCM, Scoping*.

1. Introduction

The quality of a software product can be traced from its process and the set metrics that measure its effectiveness in fulfilling customers' requirements and adherence to acceptable development standards. One of these software metrics is cohesion. Cohesion refers to the degree of relatedness among modules of a software product [21] [22] [23]. Cohesion measures the usage of a module and its elements within another module in terms of imported or exported functionality. Cohesion has been a subject of study for almost four decades with Yourdon and Constantine [38] classifying measures on an ordinal scale for component cohesion to normalized Hamming Distance metrics by Counsell, Swift and Crampton [10].

In object oriented systems, cohesion is measured in terms of the degree to which methods and attributes of a class belong together. High class cohesion in object oriented systems manifests a well-designed class [16]. According to Briand, Daly and Wust [6], high cohesion within a module makes it easier to develop, facilitates comprehension [15], helps in identification of modules that require reconstruction [30], enhances maintenance, testing [1] and components reusability, improves process merit Assessment [31] and reduces fault-proneness ensuring components independence with less complexities [32].

Class scope refers to the visibility of variables and their usage within the class [45]. Scoping of class elements controls access of data in various parts of a program and a metric that address attributes interactions [33] within scopes would help in understanding this control [28].

2. Related Work

This section discusses the various cohesion metrics that can be used in evaluating a class' cohesiveness.
Chidamber and Kemerer proposed the Lack of Cohesion Methods (LCOM 1) and LCOM 2 [8] [9] that measure lack of class cohesion [2] through lack of attribute commonality in methods. As outlined by Sharma and Srinivasan [36], these are inverse cohesion measures [24] and a class with zero value indicates that none of its methods use any of the attributes, therefore lacking cohesion [41].

Li and Henry [26] further extended LCOM1 and LCOM2 with LCOM3 Metric that introduces the use of undirected graph [13]. Each class method is represented as a graph node (vertice) and any shared instance attribute(s) is represented as an edge. The total class cohesion is the number of connected graph components. This concept was advanced by Hitz and Montazeri [18] to LCOM4 metric where a class X has a set of instance attributes I (x) and a set of methods M (x). A undirected graph G (v, e) is used where M(x) represents vertices. The graph edges [37] are formed when two vertices access the same instance attribute [7]. LCOM4 is measured as the number of connected components of G(x) and recommends that large classes should be divided into smaller, more cohesive classes if LCOM4>1.

Henderson-Sellers [17] proposed the last version of the LCOM metrics: the LCOM5 metric. LCOM5 outlines that a given class has a cohesion measure (LCOM5) zero (0) if

every method references all its attributes (perfect cohesion). A one (1) is given, if every class method references only one attribute. This metric uses a normalized range of 0 to 1 and the measure varies as a percentage of the perfect cohesion.

There are other metrics that have borrowed concepts from the LCOM metrics;
The RLCOM metric proposed by Li [27] works by use of pairs of methods; The Coh Metric proposed by Briand, Daly and Wust [6] uses distinct types for each method in a class although it excludes members' and normalizes the range (from 0 to 1) [20]; Bieman and Kang's [4] tight class cohesion(TCC) and loose class cohesion(LCC) metrics. The TCC measures the percentage of pairs of public methods in a class with no common attribute usage and its relative number of directly connected methods (those sharing at least one attribute) [12].The LCC measures the percentage of pairs of public methods in a class with transitive closure of common attribute usage and its relative number of indirectly connected methods (two methods that share at least one attribute directly or transitively).

Badri [3] also proposed the DC_D and DC_I metrics that add method invocations [20]. The DC_D (Degree of Cohesion Direct) measures the fraction of the directly connected pairs of methods where two methods are directly connected if they are directly connected to an attribute or if they directly or transitively invoke the same method. The DC_I (Degree of Cohesion Indirect) measures the fraction of the directly and transitively connected pairs of methods where the two methods are transitively connected if they are directly or indirectly connected to an attribute or if the two methods directly or transitively invoke the same method [29]. Bonja and Kidanmariam [5] proposed the Class Cohesion (CC) Metric that measures the degree of similarity between methods pairs whereas Dallal [46] proposed the distance design-based direct class cohesion (D3C2) that uses a direct attribute type (DAT) matrix that measures the interaction between methods caused by sharing attributes.

3.Methodology

Software development process assessment is the beginning of a great software product improvement [42].The assessment of class design and development ensures that highly cohesive classes are achieved at fair costs without compromising on the quality of the software product. Class cohesion assessment is normally done to ensure standard practices have been followed and to make recommendations for process improvement [43] [44].

3.1 Scoped Class Cohesion Metric (SCCM) Software Tool

In this study, the SCCM software tool was developed to assist in the calculation of the metric values. The software has been developed in JavaScript and HTML5 with four separate JavaScript files for each language (Java, JavaScript, PHP and C++) implementation and is accessible on a web browser interface as shown in Fig.1 below.

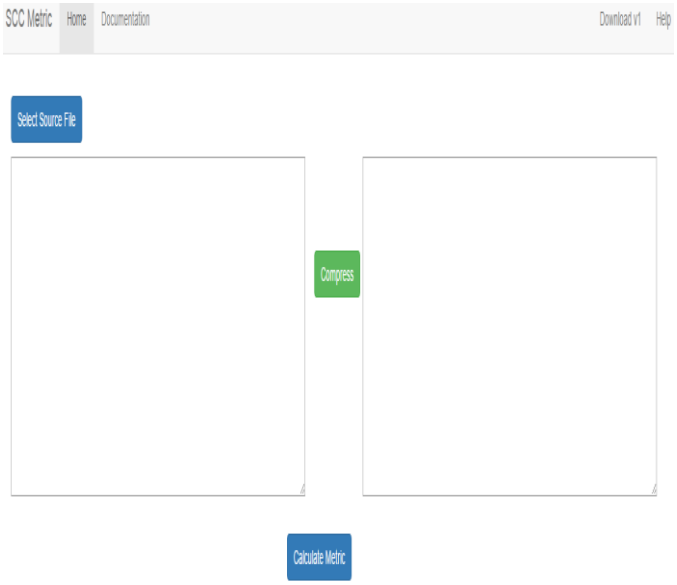

Fig. 1 SCCM interface

The software works by allowing a user to select a valid source code file from a storage location, the user then compresses the code in order to remove white spaces and comments that do not form part of the tokenized source code and then calculates the metric values which are output on the web console.

3.2 Experimental Setup

The data used in the calculation of the SCCM is acquired from a source-code rich online repository (https://www.github.com) and from a total of ten standard classes per cluster (4clusters-PHP, Java, C++ and JavaScript classes) from ten different object oriented systems. The languages were selected because they are in the top ten lists of most currently used OOP languages by developers [19].

The metric values used by the SCCM use a rational scale with a minimum value of natural 0 and a maximum value of 1[20].The variables used in the experiment are: public

attributes, private attributes, local attributes, public methods, private methods, direct and indirect occurrences of attributes and methods. In addition, the inherited methods were factored in whereas the constructors and the destructors were omitted so that artificial cohesion is not introduced [40].

In order to identify the effective metric among the two, descriptive statistics have been used for data interpretation. In measuring the central tendency; the geometric mean is used to approve the effective metric among SCCM and COH whereas the relationship between the metric values and its various constituents uses Pearson's coefficient. The COH metric has been considered since its formulation is closely related to the SCCM and for reliability purposes although it ignores scoping and use of local variables.

3.3 SCCM Metric

The following parameters were used in the development of the metric;

PM - public methods
PRM - private and or protected methods
PA - public attributes
PRA - private and local attributes
PO - public occurrences (of both PA and PRA in public methods and the invocations of any class methods)
PRO - private occurrences (both PA and PRA in private methods and the invocations of any class methods).
TPC - Expected total public cohesion

$$\mathbf{TPC} = (\mathbf{PA} + \mathbf{PRA}) * \mathbf{PM} \quad (1)$$

TPRC - Expected total private cohesion

$$\mathbf{TPRC} = (\mathbf{PA} + \mathbf{PRA}) * \mathbf{PRM} \quad (2)$$

PC - Observed total public cohesion

$$\mathbf{PC} = \frac{\mathbf{PO}}{\mathbf{TPC}} \quad (3)$$

PRC - Observed total private cohesion

$$\mathbf{PRC} = \frac{\mathbf{PRO}}{\mathbf{TPRC}} \quad (4)$$

TC - Total class cohesion

$$\mathbf{TC} = \mathbf{PC} + \mathbf{PRC} \quad (5)$$

4.Results and Discussion

This section introduces the acquired results and discussion from the experiment performed. Table 1 shows the scanned raw values from several systems collected using the SCCM software.

Table 1: SCCM and COH values from the PHP Cluster

PHP SYSTEMS										
SYSTEM	**SCCM**	**COH**	**PM**	**PRM**	**PA**	**PRA**	**PO**	**PRO**	**LV**	**LVUSAGE**
Shopping Cart	0.5	0.5	5	5	0	1	19	0	6	19244
Configuration	0.754	0.754	10	15	0	7	156	0	17	1462080
CSS	0.233	0.233	5	5	6	0	22	12	7	112944
Game1	0.333	0.25	4	0	0	3	11	0	1	0
Game2	0.6	0.6	3	2	2	0	6	4	11	30562
Board	0.167	0.167	8	0	0	3	21	0	4	1560
CRUD	0.523	0.262	13	0	0	5	46	0	12	10610
Registration-login	0.4	0.4	5	0	2	0	13	0	8	6940
DSN	0.25	0.2	4	1	0	1	7	0	1	40
TicTacToe	0.6	0.6	3	2	2	0	6	4	11	30562
Pearson's Coefficients			0.197	0.571	-0.191	0.339	0.56	-0.168	0.844	0.414

The geometric means were computed and recorded as shown in Table 2 comparing between SCCM and COH metric values.

Table 2: C++ systems cluster geometric means

C++ SYSTEMS			
SYSTEM	SCCM	COH	GEOMETRIC MEAN
Escape pod	0.533	0.333	0.421
Stationary	0.292	0.292	0.292
Player game	0.419	0.419	0.419
Widget	0.325	0.175	0.238
Banking System	0.447	0.447	0.447
HRM System	0.652	0.202	0.363
Snake Game	0.228	0.229	0.229
Person	0.512	0.184	0.307
LeanclubLib	0.375	0.375	0.375
ProgramLib	0.089	0.071	0.079

Fig. 2 below shows the influence of public methods within a given class.

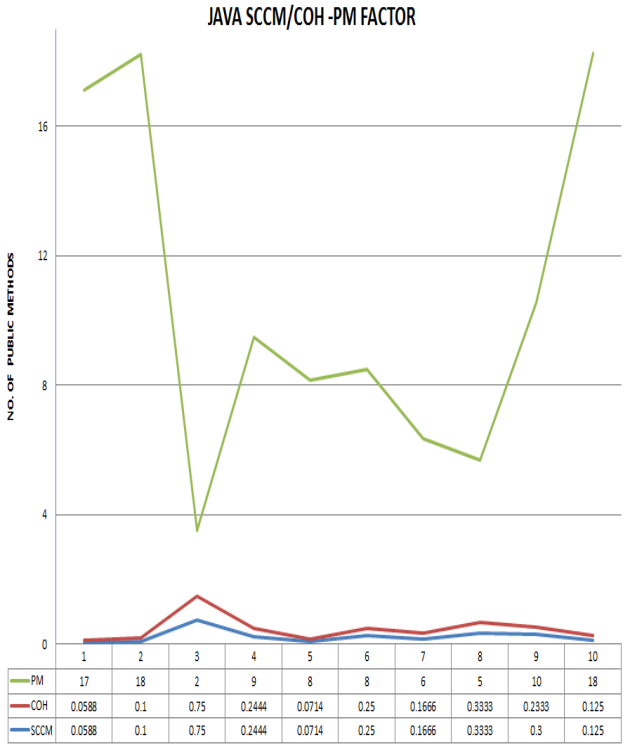

Fig. 2 SCCM/COH PM factor

The effect of both private and public attributes of the classes is also shown in Table 3 giving Pearson's coefficient values in regard to SCCM.

Table 3: PA and PRA Pearson's Coefficients on SCCM values

JAVASCRIPT SYSTEMS			
SYSTEM	SCCM	PA	PRA
AlertifyJs	0.403	5	22
AncestryJs	0.875	0	4
Board Game	0.6155	8	14
Class12Lib	0.3043	3	10
Metaclass	0.925	4	0
JsClass9Lib	0.1561	6	50
Jssl8Lib	0.189	18	25
ProtoJS	0.3238	0	7
PersonaJs	0.6667	3	3
LanguageJs	0.4375	1	3
		-0.4001	-0.6863

Fig. 3 below shows the effects of total number of variables in a class.

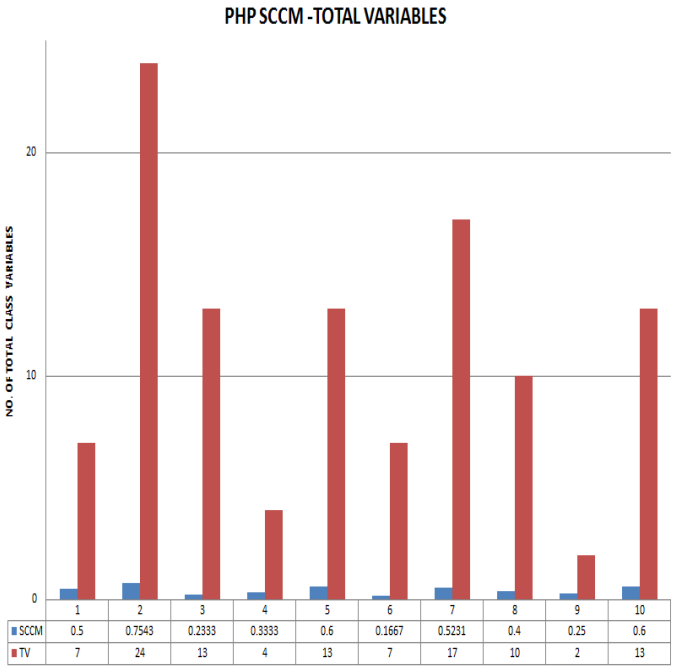

Fig. 3 Total variables influence on SCCM

In all the four sampled clusters, the values of SCCM were found to be higher than those of COH which is also evident in comparison to the geometric means. This results from the accounting of method calls by SCCM which are not factored in by the COH metric.

Classes from systems with higher public attributes gave lower SCCM values showing a negative correlation

between the number of public attributes and the cohesion values. Within the SCCM metric, private attributes were noted to contribute to class cohesion negatively and they also non-influenced the metric if the class is inherited. This is because private members cannot be accessed outside a class [14]. However, the calculation of the SCCM values does not account the use constructors and destructors because they artificially increase the cohesion value [11]. Inherited attributes and methods (directly or indirectly) were factored in to cater for inheritance- which is a major concept in object oriented software development) [35]. Classes with lower numbers of both public and private attributes gave higher SCCM values and as the total number of attributes increased, the values of SCCM also decreased.

From the analysed data, the presence of low local variables and their usage is associated with low SCCM values. This emphasizes the importance of local variables and their usage in calculating class cohesion value which is also reflected by the negative Pearson's correlation value with the SCCM metric stating that they are equally important just like the public and private attributes.

Classes with higher number of public methods were also noted to have least SCCM values whereas those classes that possess private methods and variables were found to be more cohesive than those that did not utilize both scoped variables and methods.

In all the classes studied, we observed that large classes have the least SCCM values making them good candidates for inspection. These large classes are also associated with large number of methods which may increase the likelihood of errors during the development process making error tracking a strenuous process [34]. These class lengths could also lead to maintainability issues and subsequently effecting any changes in the development and testing processes. The values of SCCM also tend to decrease with increase in the number of public occurrences.

A highly cohesive class gives a high quality software product. This is characterized with reduced fault proneness and complexity whereas at the same time enhancing a developer's understanding and reliability of the code[39].A highly cohesive class also helps in supporting low coupling between the modules and ensures code reusability which is core facet in object oriented systems**.** This sharply contrasts with lowly cohesive classes that are associated with poor class design and costly testing efforts' [25].

The values from both COH and SCCM are closely related although different approaches have been followed. However, with the consideration of the new methodology that integrates the scoping of the occurrences' and adding up of local variables and their usage, new metric gains have been achieved by the SCCM compared to its derivative: COH metric.

5.Conclusions and Future work

In this paper, a new way of evaluating class cohesiveness has been introduced based on scoping elements that make up a class. An easier way of collecting the data has also been provided which could also be employed on the COH metric with light adjustments. From the analyzed data, it has also been found out that local variables also play a critical role similarly to the public and private variables in enhancing data control and that large classes or classes with many members do not necessary mean they are cohesive compared to smaller classes. It is therefore important for developers to introduce them when necessary if at all understanding, easier maintenance, better testing and good class design is to be achieved in the long run.

The future scope of this work can be extended by:

- Creating a source code parser for the four clustered languages instead of just JavaScript.
- Evaluating cohesion of a class from the methods cohesion perspective using local variables.
- Analysis of the COH metric using SCCM software with a few adjustments on the tool.

References

[1] L. Badri, B. Mourad and T. Fadel. An Empirical Analysis of Lack of Cohesion Metrics for Predicting Testability of Classes. International Journal of Software Engineering and Its Applications, 5(2), 2011, pp. 69-86.

[2] I. Baig. Measuring Cohesion and Coupling of Object-Oriented Systems: Derivation and mutual study of cohesion and coupling. m.s. thesis. School of Engineering, Blekinge Institute of Technology Sweden, 2005.

[3] L. Badri and M. Badri. A Proposal of a new class cohesion criterion: an empirical study. Journal of Object Technology, 3 (4), 2004.

[4] J. Bieman and B. Kang. Cohesion and reuse in an object-oriented system. Proceedings of the 1995 Symposium on Software Reusability, Seattle, Washington, United States, 1995, pp.259–262.

[5] C. Bonja and E. Kidanmariam. Metrics for class cohesion and similarity between methods. Proceedings of the 44th Annual ACM Southeast Regional Conference, Melbourne, Florida, 2006, pp. 91-95.

[6] L. C. Briand, J. W. Daly and J. Wust. A Unified Framework for Cohesion Measurement in Object-Oriented Systems. Software Metrics Symposium, Proceedings, Fourth International, 1997, pp. 43-53.

[7] S. M. Chandrika, E. S. Babu and N. Srikanth. Conceptual Cohesion of Classes in Object Oriented Systems.

International Journal of Computer Science and Telecommunications, 2(4), 2011, pp.38-44.

[8] S. R. Chidamber and C. F. Kemerer. Towards a metrics suite for object-oriented design. Object-Oriented Programming Systems, Languages and Applications (OOPSLA), 26, 1991, pp.197–211.

[9] S. R. Chidamber and C.F. Kemerer, C.F. A metrics suite for object oriented design. IEEE Transactions on Software Engineering, 20, 1994, pp.476–493.

[10] S. Counsell, S. Swift and J. Crampton. The interpretation and utility of three cohesion metrics for object-oriented design, ACM Transactions on Software Engineering and Methodology (TOSEM), 15(2), 2006, pp. 123-149.

[11] A. J. Dallal. Validating Object-Oriented Class Cohesion Metrics Mathematically. Recent Advances in Software Engineering, Parallel and Distributed Systems, 2011.

[12] A. J. Dallal and L. C. Briand. An object-oriented high-level design-based class cohesion metric. Information and Software Technology, 2010, pp. 1346–1361.

[13] A. J. Dallal, A Design-Based Cohesion Metric for Object-Oriented Classes. International Journal of Computer Science and Engineering, 1(2), 2010, pp. 195-200.

[14] Dammers.Why do we need private variables. http://programmers.stackexchange.com/questions/143736/why-do-we-need-private-variables. [Cited: 23rd December,2015]

[15] R. Dasari and G. Vasanthakumari. Fault Prediction in Object-Oriented Systems Based on C^3 (Conceptual Cohesion of Classes).International Journal of Modern Engineering Research (IJMER), 1(1), 2011, pp. 113-119.

[16] G. Gui and P. Scott. Measuring Software Component Reusability by Coupling and Cohesion Metrics. Journal of Computers, 4(9), 2009.

[17] B. Henderson-Sellers. Object-Oriented Metrics Measures of Complexity. Prentice-Hall, Inc., Upper Saddle River, NJ, 1996.

[18] M. Hitz and B. Montazeri. Measuring coupling and cohesion in object oriented systems. Proceedings of the International Symposium on Applied Corporate Computing, 1995, pp. 25–27.

[19] R. Hiscott.10 Programming Languages You Should Learn Right Now. http://mashable.com/2014/01/21/learn-programming-languages/. [Cited: 22nd December,2015]

[20] S. M. Ibrahim, S. A. Salem, A. I. Manal and M. Eladawy. Identification of Nominated Classes for Software Refactoring Using Object-Oriented Cohesion Metrics. International Journal of Computer Science Issues (IJCSI), 9(2), 2012, pp. 68-76.

[21] K. Kaur and H. Singh, H. An investigation of Design Level Class Cohesion Metrics. The International Arab Journal of Information Technology, 9(1), 2012.

[22] R. Kaur and T. Kaur. Comparison of various lacks of Cohesion Metrics. International Journal of Computer Trends and Technology (IJCTT). 4(5), 2013.

[23] A. Kaur and P. Kaur. Class Cohesion Metrics in Object Oriented Systems. IJSWS, 2(3), 2013, pp.78-82.

[24] N. Kayarvizhy, S. Kanmani and R. V. Uthariaraj. High Precision Cohesion Metric. WSEAS TRANSACTIONS on INFORMATION SCIENCE and APPLICATIONS. 10(3), 2013, pp. 79-89.

[25] S. Khatri, S. Chhilar and A. Sangwan. Analysis of Factors Affecting Testing in Object oriented systems. International Journal on Computer Science and Engineering.Vol.3 No.3, March, 2011.

[26] W. Li, and S. M. Henry, S.M. Maintenance metrics for the object oriented paradigm. Proceedings of 1st International Software Metrics Symposium, Baltimore, 1993, pp. 52–60.

[27] X. Li, B. Pan and B. Xing. A Measurement Tool for Object Oriented Software and Measurement Experiments with It. Proc. IWSM 2000. (Lecture Notes in Computer Science 2006, Springer-Verlag, Berlin, Heidelberg, 2001), 44-54.

[28] D. Marshall. Scope of Variables (webpage). http://www.cs.cf.ac.uk/Dave/PERL/node52.html. [Cited: 20th February,2016]

[29] I. Marsic. Class Cohesion Metrics. Retrieved from http://www.ece.rutgers.edu/~marsic/books/SE/instructor/slides/lec-16%20Metrics-Cohesion.ppt. 2013.[Cited: 20th February,2016]

[30] M. Meyers and B. David. An Empirical Study of Slice-Based Cohesion and Coupling Metrics. ACM Transactions on Software Maintenance, V (N), 2007, pp.1-25.

[31] K. Patidar, R. Gupta and G. Chandel. Coupling and Cohesion Measures in Object Oriented Programming. International Journal of Advanced Research in Computer Science and Software Engineering, 3(3), 2013.

[32] R. Pena and L. Fernandez. A sensitive Metric of Class Cohesion. International Journal "information Theories & Applications", 13, 2006.

[33] D. G. Ratna, S. V. Appaji, P. L. N. Raju and A. N. L. Kumar. Efficient Implementation of Fault Prediction in Object-Oriented Systems. International Journal of Computer Science & Communication Networks, 1(3), 2011, pp. 310-317.

[34] L. Rosenberg and L. Hyatt. "Software Quality Metrics for Object- Oriented System Environments", Software assurance Technology Center, Technical Report SATC-TR-95-1001,NASA Goddard Space Flight Center, Greenbelt, Maryland 20771.

[35] L. Samel, S. Ibrahim and E. Mohammed. International Journal of Computer Science Issues. Identification of Nominated Classes for Software Refactoring Using Object-Oriented Cohesion Metrics. Volume 9, Issue 2, March 2012.

[36] S. Sharma and S. Srinivasan. A review of Coupling and Cohesion metrics in Object Oriented Environment. International Journal of Computer Science & Engineering Technology (IJCSET), 4(8), 2013, pp. 1105-1111.

[37] S. Yadav, S. Sunil and S. Uttpal. A Review of Object-Oriented Coupling and Cohesion Metrics. International Journal of Computer Science Trends and Technology (IJCST), 2(5), 2014.

[38] E. Yourdon and L. Constantine. Structured Design. Yourdon Press, 1978.

[39] A. Yadav and R. A. Khan. Journal of Information and Operations Management ISSN: 0976–7754 & E-ISSN: 0976–7762 , Volume 3, Issue 1, 2012, pp. 191-193

[40] J. A. Dallal, "Improving Object-Oriented Lack-of-Cohesion Metric by Excluding Special Methods", In Proceedings of the 10th WSEAS International Conference on Software Engineering Parallel and Distributed Systems, 2011, pp. 124- 129.

[41] S. P. Sreeja and R. Sridaran. "A survey on different approaches of determining cohesion based object oriented metrics"international journal of engineering research and development, Vol 4, 2012.
[42] Software process improvement and assessment https://www.asaquality.ee/services/software-process-improvement. [Cited: 23rd February,2016]
[43] D. Davis. Software Process Assessment using the Software Engineering Institute's CMM® Based Appraisal for Internal Process Improvement. From http://www.davissys.com/PDF/cbaipiov.pdf [Cited: 20th December,2016]
[44] P. Marko. SPICE-International Standard for Software Process Assessment. Seminar on Quality Models for Software Engineering. Helsinki. https://www.cs.helsinki.fi/u/paakki/Pyhajarvi.pdf. [Cited:23rd February,2016]
[45] Dave. Scope of Variables. https://www.cs.cf.ac.uk/Dave/PERL/node52.html. [Cited 24th February, 2016]
[46] J. A. Dallal, A design-based cohesion metric for object-oriented classes, International Journal of Computer Science and Engineering, 2007b, 1(3), pp. 195-200.