International Journal of Computer Science and Engineering Open Access
Research Paper Volume-2, Issue-4 E-ISSN: 2347-2693

Hypotheses Verification for High Precision Cohesion Metric

Kayarvizhy N[1*], Kanmani S[2] and Rhymend Uthariaraj V[3]

[1*]*Department of Computer Science and Engineering, AMC Engineering College, India*
[2] *Department of Information Technology, Pondicherry Engineering College, India*
[3]*Ramanujan Computing Centre, India*

www.ijcseonline.org

Received: 03/03/2014 Revised: 19/03/2014 Accepted: 15/04/201x4 Published: 30/04/2014

***Abstract*—** Metrics have been used to measure many attributes of software. For object oriented software, cohesion indicates the level of binding of the class elements. A class with high cohesion is one of the desirable properties of a good object oriented design. A highly cohesive class is less prone to faults and is easy to develop and maintain. Several object oriented cohesion metrics have been proposed in the literature. In this paper, we propose a new cohesion metric, the High Precision Cohesion Metric (HPCM) to overcome the limitations of the existing cohesion metrics. We also propose seven hypotheses to investigate the relationship between HPCM and other object oriented metrics. The hypotheses are verified with data collected from 500 classes across twelve open source Java projects. We have used Pearson's coefficient to analyze the correlation between HPCM and the metrics in the hypotheses. To further bolster our results we have included p-value to confirm the statistical significance of the findings.

***Keywords*—**Object Oriented Metrics; Cohesion; High Precision

I. INTRODUCTION

Object oriented design and development continues to be the most widely used methodology for designing high quality software. Ensuring the quality of such systems is of prime interest. Eliminating defects that affect quality in early stages of software development life cycle, like design, saves cost and effort [1] [2] [3]. Object oriented design-level metrics and their associated quality prediction techniques attempt to ensure quality during the design and early coding phase. Many object oriented metrics have been proposed to compute an underlying property like cohesion, inheritance and coupling. Cohesion metrics indicate how well the class elements bind with each other. Since a class consists of two basic sets of elements, attributes and methods, all of the cohesion metrics revolve around the usage of these. A high cohesion value indicates that the class is well structured and provides the stated functionality with the help of well-knit attributes and methods [4]. Conversely a low cohesive value indicates a class that may need to be split or redesigned. A non-cohesive class is a maintenance nightmare and is prone to faults [5] [6] [7].

Several cohesion measures have been proposed in the literature. Chidamber and Kemerer [8] proposed the first cohesion metric. They presented an inverse metric, thus measuring the lack of cohesion (LCOM1). LCOM1 measures the number of pairs of methods that do not have any common attribute. Chidamber and Kemerer [9] revised their metric and came up with an altered version (LCOM2) which found the difference between the number of pairs of methods not sharing any attribute and those sharing at least one attribute. LCOM1 and LCOM2 do not have normalized values. Li and Henry [10] proposed their version of lack of cohesion (LCOM3) as the number of connected components in a graph with methods as node and edges between each pairs of methods with at least one common attribute. Hitz and Montazeri [11] enhanced the LCOM3 to include edges for method to method invocations as well and named it LCOM4. Henderson-Sellers [12] came up with LCOM5 which included the number of distinct attributes accessed by each method. LCOM3, LCOM4 and LCOM5 show high values of cohesion even with a single common attribute among methods. After LCOM5, the cohesion metrics proposed were direct cohesion metrics compared to the inverse nature of lack of cohesion metrics proposed till then. The first direct cohesion metric, Connectivity (Co) was introduced by Hitz and Montazeri [13]. Briand et al [21] proposed Coh which uses the concept of number of attributes used by each method. Bieman and Kang [14] introduced TCC (Tight class cohesion) and LCC (Loose class cohesion) which captured the methods connected through attributes directly and indirectly. Badri [15] introduced variations on TCC and LCC naming them DCD and DCI which includes method invocations also, directly or in the call trace respectively. TCC, LCC and their derivatives DCD and DCI also give skewed cohesion values when a single common attribute exists in all methods. The metrics that were proposed further were based on the similarity between methods based on the number of common attributes used between them. Bonja and Kidanmariam [16] defined Class Cohesion (CC) based on this similarity of methods. The concept was further enhanced by Fernandez and Pena [17] in their metric Sensitive Class Cohesion Metric (SCOM). Bansiya et al. [18] defined Cohesion among Methods in a class (CAMC) as a modified version of Co. Counsell et al

Corresponding Author: *Kayarvizhy N, kayarvizhy@gmail.com*

[19] also introduced Normalized Hamming Distance (NHD) and Scaled Normalized Hamming Distance (SNHD) based on how many attributes each method accesses. Al Dallal and Briand [20] came up with Low-level design Similarity based Class Cohesion (LSCC) which finds out how many methods access each attribute. SCOM, CAMC, NHD, SNHD and LSCC give high cohesion values in the presence of few common attributes across methods.

In this paper, we propose a new cohesion metric, the High Precision Cohesion Metric (HPCM) to address the limitations in the existing cohesion metrics. HPCM is designed to give precise cohesion values. We then analyze the proposed metric with hypotheses. This paper is organized as follows. Section 2 explains the High Precision Cohesion Metric (HPCM) in detail. In Section 3 we propose seven hypotheses capturing the relation between HPCM and other OO metrics. These hypotheses are verified empirically in Section 4. Section 5 lists the limitations and future work and Section 6 concludes the study followed by a list of references.

II. HIGH PRECISION COHESION METRIC

A. *Limitations of existing Cohesion Metrics*

LCOM1 and LCOM2 are not normalized in their value and hence it is difficult to compare their values between two classes and understand their relative cohesion. Their values are dependent on the number of methods in a class. Hence their values can grow to very high values. Consider two classes, c1 and c2 which have two and four methods each. The attributes accessed by the methods are listed here - c1m1 {a1}, c1m2 {a2}, c2m1 {a1}, c2m2 {a2}, c2m3 {a3}, c2m4 {a4}. Each method accesses just a single unique attribute in both the classes. But the values of LCOM1 and LCOM2 are very different for the classes. LCOM1(c1) = LCOM1(c2) = 1 and LCOM2(c1) = LCOM2(c2) = 6.

LCOM3 and LCOM4 are also not normalized in their value but the range is improved over LCOM1 and LCOM2. But LCOM3 and LCOM4 suffer from a different issue. Consider class c1 with the following method-attribute interactions - c1m1 {a1, a2, a4}, c1m2 {a3, a5, a4}, c1m3 {a6, a7, a4}, c1m4 {a7, a8, a4}, c1m5 {a9, a10, a4}, c1m6 {a11, a12, a4}. The methods are not cohesive since they all share just one common attribute between them and use their own unique set of attributes otherwise. However LCOM3 and LCOM4 give the class a perfect cohesion value of 1. The same limitation applies to TCC, LCC, DCD and DCI.

CC and SCOM cohesion metrics vastly improve on the idea of similarity. They do not consider just one common attribute sufficient for high level of cohesion. Instead these metrics account for the number of attributes that are shared between methods. However they have a limitation as highlighted here. Consider a class c1 with the following method-attribute interactions – c1m1 {a1}, c1m2 {a1}, c1m3 {a1}, c1m4 {a1}, c1m5 {a2, a3, a4, a5}. We have CC = SCOM = 0.6 for this class. However careful observation of the class would show that most of the attributes are not shared by majority of its methods. This is not realistically captured by CC and SCOM.

The motivation for a new metric is to have a fine grain value for cohesion. HPCM differentiates classes with similar but not identical cohesion. HPCM values are normalized on a scale of 0 to 1.

B. *Definition of HPCM*

The High Precision Cohesion Metric (HPCM) is based on two related concepts – Average Attribute Usage (AAU) and Link Strength (LS). The AAU is a separate metric which computes the average number of attributes used by each method of the class directly or indirectly through a method invocation. We consider only public, non-inherited methods of a class. Inherited attributes are also not considered for computation of AAU. Let 'c' be the class in consideration. Then $M_I(c)$is the set of non-inherited, overriding or newly implemented methods of c. Further $M_{pub}(c)$ is the set of public methods of c. Public non-inherited or overridden methods are given by $M_I(c) \cap M_{pub}(c)$. The total number of methods in our case is the cardinality of such a set. It is given by $|M_I(c) \cap M_{pub}(c)|$. The attributes referenced by a method is given by AR(m) where m is the method. Hence total attributes referenced is given by $\sum AR(m)$, for each m in the set $M_I(c) \cap M_{pub}(c)$. Hence the AAU is given in Equation (1).

$$AAU = \frac{\sum AR(m)}{|M_I(c) \cap M_{pub}(c)|} \quad (1)$$

Link Strength is a concept to capture the level of interaction between a pair of methods based on the number of attributes commonly used between them. Link Strength (LS) is based on AAU. LS for a pair of methods m1 and m2 is given in Equation (2).

$$CA_{m1m2} = |AR(m1) \cap AR(m2)|$$

$$LS_{m1m2} = \begin{cases} \frac{CA_{m1m2}}{AAU}, & if\ CA_{m1m2} \leq AAU \\ 1, & if\ CA_{m1m2} > AAU \end{cases} \quad (2)$$

HPCM is defined using Link Strength. We define HPCM as given in Equation (3)

$$M = |M_I(c) \cap M_{pub}(c)|$$

$$HPCM(c) = \frac{2 * \sum LS_{m_i m_j}}{M * (M-1)} \quad (3)$$

HPCM is an average of Link Strength of all method pairs in the class. To find the average we first find the total of all LS in the numerator. Here all public non inherited, overridden and newly implemented methods are considered, similar to AAU and their link strengths are summed up. The

denominator denotes the total available method pairs and is given by a simple formula of $n * (n - 1)/2$ where 'n' is the total number of methods. In our case the total number of methods is given by $M_I(c) \cap M_{pub}(c)$ and hence results in given denominator.

C. Evaluating HPCM

To evaluate HPCM, we have considered 500 classes from 12 open source projects written in Java. The projects considered were Apache Mahout, Apache Open Web Beans, Apache Sling, Apache Synapse, Apache Tobago, Apache Tomcat, Camel, Castor, Cayene, Eclipse, JDK 7, and Struts. The code for each of the project was downloaded from their respective websites [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36]. Classes were chosen such that there they were comparable in terms of their size ranging from a few methods to around twenty methods. Classes without attributes or methods were not considered. The bug information for the projects was retrieved from Sonar Nemo website [23]. Only major and critical defects were considered for the study. The cohesion metrics LCOM1, LCOM2, LCOM3, LCOM4, LCOM5, TCC, LCC, SCOM, CC and HPCM were computed using Automated Tool developed by the authors [24]. Studies have confirmed of a linear relationship between cohesion metrics and defects [10] [37] [38]. Univariate Linear regression was used to evaluate the fault prediction capability of the cohesion metrics. Linear Regression attempts to fit the data points (in our case the bug data and each cohesion metric) in a straight line $y = \alpha + \beta x$. Where 'α' is the constant term, 'β' is the coefficient, 'y' is the number of bugs in the class and 'x' is the cohesion metric. The Root Mean Square Error (RMSE) was used to measure the fitness of the resulting plot. A lower RMSE indicates a good fit whereas a high RMSE indicates poor fit. Table 1 lists the RMSE for the various cohesion metrics that were obtained in the linear regression analysis.

Table 1. HPCM Evaluation – RMSE

Metric	RMSE	Coefficient	Constant
LCOM1	0.509	0.001	.54
LCOM2	0.509	0.001	.54
LCOM3	0.481	0.049	.39
LCOM4	0.525	0.049	.46
LCOM5	0.433	0.721	.07
TCC	0.379	-1.19	.95
LCC	0.389	-.910	.92
SCOM	0.391	-1.19	.95
CC	0.410	-1.46	.89
HPCM	0.370	-1.32	.87

We find that RMSE of the metrics TCC, LCC, SCOM and CC indicate lower error and hence better fit compared to the traditional cohesion metrics. However HPCM gives the least RMSE indicating the best fit among the cohesion metrics for predicting bugs. This indicates that HPCM predicts the faults better compared to other cohesion metrics

III. HPCM HYPOTHESES

Our goal in this paper is to study specific aspects and characteristics of HPCM. In the following sections we propose seven hypotheses investigating the relationship between HPCM and other object oriented metrics. We limit the metrics to a class since HPCM is class cohesion metric.

A. HPCM vs. Size Metrics

Size metrics of a class captures how big the class is in terms of its attributes, methods and lines of code. Research efforts have gone it to measure the effect the size of a class has on other OO metrics [22]. A class with more number of methods tends towards having groups of methods, with each group concentrated on a different functionality. The same argument can be extended to attributes as well. Also an increase in attributes and methods leads to more lines of code. Hence a cohesive class should depict an inverse relation to all of these size metrics. We have considered three size metrics - lines of code (LOC), number of attributes (NOA) and number of methods (NOM) for our analysis. Our hypotheses are based on the understanding that bigger classes tend to be overloaded with more than one core functionality and hence lead to poor cohesion. The following hypotheses are proposed for analyzing the relationship between HPCM and size metrics.

Hypothesis 1. Lesser the number of attributes defined in a class, higher the value of HPCM of the class

Hypothesis 2. Lesser the number of methods defined in a class, higher the value of HPCM of the class

Hypothesis 3. Lesser the LOC of a class, higher the value of HPCM

B. HPCM vs. LCOM

LCOM proposed by Chidamber and Kemerer [8] is one of the earliest known metric for cohesion of a class. It is an inverse metric that captures the lack of cohesion. The value of LCOM is not range bound and tends to get very high for classes with more methods. Both HPCM and LCOM capture the underlying property of cohesion in a class. However HPCM is a precise and finer metric compared to the coarse LCOM. HPCM is also a direct cohesion metric. Based on this information we propose the below hypothesis

Hypothesis 4. Higher the value of HPCM, lower would be the value of LCOM of the class

C. HPCM vs. CBO

Coupling between Objects (CBO) was proposed by Chidamber and Kemerer [9] to measure the level of dependency between classes. A well designed class is a self-contained unit and should have minimum dependency on other classes to fulfill its functionality. On the other hand, a highly coupled class has a lot of interaction with other classes. CBO captures the depth of inter class relationship of

a class. With this basic premise we propose the next hypothesis below.

Hypothesis 5. Higher the value of HPCM, lower would be the value of CBO of the class

D. HPCM vs. DIT

The Depth of Inheritance Tree was proposed by Chidamber and Kemerer [9] to measure the level of inheritance of a class. It is the length of the maximal path from the node to the root of the tree. A class's cohesion and its depth of inheritance are unrelated properties. The depth of inheritance (DIT) of a class should not have any effect in the way the class's elements are bound together. With this assumption we propose the following hypothesis

Hypothesis 6. The DIT of a class should not have an effect on the HPCM of the class

IV. HYPOTHESES VERIFICATION

In this section we perform the empirical analyses for the hypotheses that were formulated in the previous section

A. Data Set

To evaluate the proposed hypotheses, we have considered the same dataset that was used in evaluating HPCM in section 2.3

B. Metrics

The metrics listed in Table 2 were computed to evaluate the proposed hypotheses. All the metrics were computed from the dataset using the Automated Tool developed by the authors [24]

Table 2. Metrics considered for the study

Metric	Detail
HPCM	High Precision Cohesion Metric
LCOM	Lack of Cohesion
LOC	Lines of Code
NOA	Number of Attributes
NOM	Number of Methods
CBO	Coupling Between Objects
DIT	Depth of Inheritance

C. Pearson Coefficient and p-value

We have used Pearson's coefficient to estimate the relationship between the variables in our hypotheses. Pearson's coefficient of correlation (r) is a widely used measure of correlation. It gives the extent and direction of relation between the two variables under consideration. Given variables X and Y and 'n' data points for them, Pearson's coefficient is given as in Equation (4)

$$r = \frac{\sum_{i=0}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=0}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=0}^{n}(Y_i - \bar{Y})^2}} \quad (4)$$

Pearson's coefficient can take a value from -1 to +1, both inclusive. The larger the value of 'r', ignoring sign, the stronger the association between the two variables and hence more accurately one can predict the value of one variable from the knowledge of other. A value of -1 or +1 suggests that the variables have perfect correlation. An 'r' value of 0 suggests the absence of correlation between the variables – there is no relationship between the two variables. The sign of 'r' gives information on the direction of the association. A positive value suggests that relatively high values of one variable are paired with relatively high values of another variable and low values are paired with low values of the second variable. A negative correlation means that relatively high values of the first variable are paired with relatively low values of the second variable and vice versa.

The p-value is the probability of getting the correlation by chance. When the p-value is low, the confidence in the obtained Pearson coefficient is high. Generally p-value less than 0.05 are considered statistically significant and we have taken the same cut-off in our experiment as well.

D. Results and Interpretation

The results of the empirical analysis are summarized in this section.

Projects	HPCM	LCOM	LOC	NOA	NOM	CBO	DIT
Mahout	0.461	24.18	133.12	4.72	7.66	8.38	0.59
OWB	0.489	29.81	163.63	4.31	7.95	7.5	0.86
Sling	0.416	18.66	125.59	4.61	8.38	4.07	0.48
Synapse	0.271	318.8	382.3	10.4	13.7	24.7	1.1
Tobago	0.355	69.57	141.76	6.09	7.86	6.24	0.57
Tomcat	0.189	176.6	378.58	8.9	13.03	12.4	0.73
Camel	0.279	156.0	160.0	5.15	10.46	14.9	0.36
Castor	0.205	50.76	149.76	4.57	9.73	6.66	0.61
Cayene	0.314	73.33	168.71	7.52	11.62	12	0.86
Eclipse	0.236	105.4	172.52	10.2	6.66	20.4	0.42
JDK7	0.330	35.15	185.87	7.33	7.37	2.38	1.86
Struts	0.292	51.45	138.8	4.45	6.27	7.63	0.36

Table 4 provides the correlation coefficient values for HPCM with size metrics. We find that HPCM is negatively correlated with all the size metrics as proposed in the hypotheses

Table 4. Pearson Coefficients for HPCM vs. Size Metrics

Projects	HPCM vs. LOC	HPCM vs. NOA	HPCM vs. NOM
Mahout	-0.28712	-0.37333	-0.15952
OWB	-0.37087	-0.45170	-0.36659
Sling	-0.16793	-0.31734	-0.30214
Synapse	-0.24568	-0.40196	-0.10998
Tobago	-0.49601	-0.37482	-0.35987
Tomcat	-0.21269	-0.28263	-0.13176
Camel	-0.52366	-0.46848	-0.20135

Castor	0.029200	-0.24433	-0.06038
Cayene	-0.14734	-0.42173	-0.24623
Eclipse	-0.42998	-0.08017	-0.23754
JDK7	-0.27492	-0.19197	-0.22159
Struts	0.315460	0.170510	-0.34170

Table 5 provides the correlation coefficient values for HPCM with LCOM, CBO and DIT. HPCM has negative correlation with LCOM and CBO as proposed earlier. HPCM does not show a clear correlation direction with DIT. Some projects show non-trivial positive correlation (like 0.44169 in Synapse) while others show a non-trivial negative correlation (-0.2933 in Sling). Hence there is no clear conclusion on the effect of DIT on HPCM

Table 5 – Pearson Coefficients for HPCM vs. LCOM, CBO and DIT

Projects	HPCM Vs LCOM	HPCM Vs CBO	HPCM Vs DIT
Mahout	-0.50256	-0.38886	0.3572
OWB	-0.43935	-0.16926	-0.0151
Sling	-0.37584	0.02919	-0.2933
Synapse	-0.38714	-0.28206	0.44169
Tobago	-0.51058	-0.70940	0.10882
Tomcat	-0.15733	-0.17478	0.08981
Camel	-0.48129	-0.51830	0.05387
Castor	-0.33969	-0.13932	0.40893
Cayene	-0.34952	-0.27688	-0.2077
Eclipse	-0.11025	-0.20351	-0.2022
JDK7	-0.44104	-0.14557	-0.0741
Struts	-0.39241	-0.08755	-0.0002

Table 6 gives the overall correlation results for the various metrics along with their statistical significance (p-value). We find that all the metrics except DIT are negatively correlated as proposed in the hypotheses. The correlation coefficient between HPCM and DIT is close to zero, indicating near absence of correlation. However since the correlation between HPCM and DIT was not conclusive in section 4.4. Additional experiments are suggested before drawing conclusions on hypothesis 6. Also in all the cases we find p-value less than 0.05 confirming the statistical significance of the coefficients. The p-value is less than 0.01 for 6 cases shows high statistical significance.

Table 6 – Overall Pearson Coefficients

Hypotheses	Pearson Coefficient	p-value
HPCM Vs LOC	-0.2662179	0.00000115
HPCM Vs NOA	-0.2768793	0.00000043
HPCM Vs NOM	-0.2307795	0.00002290
HPCM Vs LCOM	-0.2494641	0.00000501
HPCM Vs CBO	-0.2274472	0.00002965
HPCM Vs DIT	0.0044887	0.00468834

V. LIMITATIONS AND FUTURE WORK

The study is prone to certain limitations which can be investigated and addressed in related future work. Fault modeling based on one metric might not yield the best results. Faults could be contributed by other metrics too. For the scope of this study we have considered that metrics other than cohesion have a constant impact on faults. The relationship between HPCM and DIT is inconclusive. Some projects presented a positive correlation and some indicated negative correlation. This requires additional experiments to arrive at a conclusion. We have considered data from open source Java projects. This data set is representative of a small population of projects. A large real time object-oriented system from industry will serve as a better data set and can be considered for future study. The metric could be deployed in an industry setup and feedback from industry users can be used to further refine the metric. Future studies could also consider fault prediction models based on HPCM and other metrics to further understand their relationship and their contribution to the overall defect prevention

VI. CONCLUSIONS

In this paper we analyzed the limitations of existing cohesion metrics. A new cohesion metric, the High Precision Cohesion Metric has been proposed to address the limitations. The characteristics of HPCM have been investigated and validated with the help of seven hypotheses. The empirical study was done using data from 500 classes taken from 12 open source projects in Java. The hypothesis gives an insight on the relationship between HPCM with other object oriented metrics.

REFERENCES

[1] G. Concas, M. Marchest, G. Destefanis and R. Tonelli, An empirical study of software metrics for assessing the phases of an agile product, International Journal of Software Engineering and Knowledge Engineering, June 2012, vol. 22, no. 04, pp. 525-548

[2] E. Paikari, M. M. Ritcher and G. Ruhe, Defect Prevention using case based reasoning: An attribute weighing technique based upon sensitivity analysis in neural networks, International Journal of Software Engineering and Knowledge Engineering, Sep. 2012, vol. 22, no. 06, pp. 747-768

[3] M. Khoshgoftaar, K. Gao and A. Napolitano, An empirical study of feature ranking techniques for software quality prediction, International Journal of Software Engineering and Knowledge Engineering, Mar. 2012, vol. 22, no. 02, pp. 161-183

[4] C. Z. Zhou and Y. B. Xu, A novel approach to measuring class cohesion based on dependence analysis, Proceedings of the International Conference on Software Maintenance, 2002, pp. 377-384.

[5] L. C. Briand, C. Bunse and C. J. Daly, A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs, IEEE Transactions on Software Engineering, 2001, pp. 513-530.

[6] J. Bieman and L. Ott, Measuring functional cohesion, IEEE Transactions on Software Engineering, 1994, pp. 644-657.

[7] T. Mens and S. Demeyer, Future trends in software evolution metrics, Proceedings of IWPSE2001, ACM, 2002, pp. 83-86.
[8] S. R. Chidamber and C. F. Kemerer, Towards a metrics suite for object-oriented design, Proceedings of Conference on Object-Oriented Programming Systems, Languages and Applications, 1991, pp. 476-493.
[9] S. R. Chidamber and C. F. Kemerer, A metrics suite for object-oriented design, IEEE Transactions on Software Engineering, 1994, pp. 476-493.
[10] W. Li and S. Henry, Object-oriented metrics that predict maintainability, Journal of Systems and Software, 1993, pp. 111-122.
[11] M. Hitz and B. Montazeri, Chidamber and Kemerer metric suite - a measurement theory perspective, IEEE Transactions on Software Engineering, 1996, pp. 267-271.
[12] B. S. Henderson, Object-oriented Metrics: Measure of Complexity. New Jersey, Prentice Hall, 1996, pp. 142-147.
[13] M. Hitz and B. Montazeri, Measuring coupling and cohesion in object oriented systems, Proceedings of the Int. Symposium on Applied Corporate Computing, 1995, 25-27.
[14] M. M. Bieman, B. K. Kang and W. Melo, Cohesion and reuse in an object oriented system, Proceedings of the symposium on software reliability, 1995, pp. 259-262.
[15] L. Badri and M. Badri, A Proposal of a new class cohesion criterion, an empirical study, Journal of Object Technology, 2004
[16] C. Bonja and E. Kidanmariam, Metrics for class cohesion and similarity between methods, Proceedings of the 44th Annual ACM Southeast Regional Conference, 2006, pp. 91-95.
[17] L. Fernandez and R. Pena, A sensitive metric of class cohesion, International Journal of Information Theories and Applications, 2006, pp. 82-91.
[18] J. Bansiya, L. Etzkorn, C. Davis and W. Li, A class cohesion metric for object-oriented designs, Journal of Object Oriented Program, 1999, pp. 47-52.
[19] 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, 2006, pp. 15:123-149.
[20] A. J. Dallal and L. Briand, A Precise method-method interaction based cohesion metric for object oriented classes, Simula Research Laboratory, Simula Technical Report, 2009
[21] L. C. Briand, J. Daly, J. Wuest, A unified framework for cohesion measurement in object-oriented systems, Empirical Software Engineering, An International Journal, 1999, pp. 65-117
[22] K. E. Emam, The confounding effect of class size on the validity of object oriented metrics, IEEE Transactions on Software Engineering, 2001, pp. 630-650
[23] Sonar Bug Repository (2012) http://nemo.sonarsource.org, September
[24] N. Kayarvizhy and S. Kanmani, An Automated Tool for Computing Object Oriented Metrics using XML. Proceedings of International Conference on Advances in Computing and Communication ACC2011, Springer, 2011, pp. 69-79.
[25] The Apache Mahout project, http://mahout.apache.org.
[26] The Apache OpenWebBeans project, http://openwebbeans.apache.org.
[27] The Apache Sling project, http://sling.apache.org.
[28] The Apache Synapse project, http://synapse.apache.org.
[29] The Apache Tobago project, http://myfaces.apache.org/tobago/index.html.
[30] The Apache Tomcat project, http://tomcat.apache.org.
[31] The Apache Camel project, http://camel.apache.org.
[32] The Castor project, http://www.castor.org.
[33] The Apache Cayene project, http://cayenne.apache.org.
[34] The Eclipse project, http://www.eclipse.org.
[35] The JDK7 project, http://jdk7.java.net.
[36] The Apache Struts project, http://struts.apache.org.
[37] Basili, Victor R., Lionel C. Briand, and Walcélio L. Melo, A validation of object-oriented design metrics as quality indicators. IEEE Transactions on Software Engineering, 1996, pp. 751-761.
[38] Subramanyam, Ramanath, and Mayuram S. Krishnan, Empirical analysis of ck metrics for object-oriented design complexity: Implications for software defects. IEEE Transactions on Software Engineering, 2003, pp. 297-310.

AUTHORS PROFILE

Kayarvizhy N received her B.Tech and M.Tech degree from Pondicherry University, Puducherry in the year 2001 and 2004 respectively in computer science and engineering. She is working as Associate Professor in AMC Engineering College, Bangalore since 2006. She is pursuing her PhD at Anna University, Chennai. Her research interest includes object oriented software metrics, neural network models and swarm intelligence algorithms.

Kanmani S received her B.E in 1991 and M.E in 1992 in computer science and engineering from Bharathiar University, Coimbatore and PhD in information and communication engineering from Anna University, Chennai in 2006. Currently she is a Professor in Pondicherry Engineering College, Puducherry. Her research area includes software systems, software metrics, object oriented systems, algorithms and data structures. Dr. Kanmani's professional affiliations are with Indian Society for Technical Education (ISTE) and Computer Society of India (CSI).

Rhymend Uthariaraj V has done his M.E and PhD in Computer Science and Engineering, Anna University, Chennai. His areas of expertise include Network Security, Pervasive Computing, Distributed Computing, Operations Research, and Computer Algorithms. He has an overall experience of 27 years. He holds the position of Secretary, Tamil Nadu Engineering Admissions and Coordinator, AICTE-MCA QIP Program at Anna University, Chennai. He is the Director of Ramanujan Computing Centre, Anna University, Chennai and has professional affiliations with Indian Society for Technical Education (ISTE).

.