*I.J. Information Technology and Computer Science,* **2012, 9, 29-35**
Published Online August 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2012.09.04

# Coupling Complexity Metric: A Cognitive Approach

**A. Aloysius**
Assistant Professor in Computer Science,St. Joseph's College, Tiruchirappalli
aloysius1972@gmail.com

**L. Arockiam**
Associate Professor in Computer Science, St. Joseph's College, Tiruchirappalli
larockiam@yahoo.co.in

***Abstract*** — Analyzing object – oriented systems in order to evaluate their quality gains its importance as the paradigm continues to increase in popularity. Consequently, several object- oriented metrics have been proposed to evaluate different aspects of these systems such as class coupling. This paper presents a new cognitive complexity metric namely cognitive weighted coupling between objects for measuring coupling in object- oriented systems. In this metric, five types of coupling that may exist between classes: control coupling, global data coupling, internal data coupling, data coupling and lexical content coupling are consider in computing CWCBO.

***Index Terms***— Software Metrics; Control Coupling, Global Data Coupling, Internal Data Coupling, Data Coupling, Lexical Content Coupling, Cognitive Weighed Coupling Between Objects (CWCBO)

## I. Introduction

Software engineering is a difficult and complex task. Software metrics are one way to predict quality within a system, pointing to problem areas that can be addressed prior to software release. Metrics attempt to measure a particular aspect of a software system. These aspects can range from traditional measurements such as the number of lines of code to the relationships created between components in a system. There are several approaches to estimate complexity of software but none of them have been accepted as a true measure of complexity of a class [1]. Object oriented perspective is one of the most significant ways to quantify reliability of software by controlling object oriented constructs. Object oriented design provides a novel approach for problem solving using models around real world entities. Most of the software projects are shifting towards object oriented design because of only that design phase of a software development life cycle is the only phase in which structure of a software is made available. From last two decades lots of metrics has been proposed ranges from cohesion to coupling in object oriented software. However, coupling has a negative impact on software reliability [2]. To minimize complexity of software it is necessary to control coupling of object oriented software. Coupling is closely related with reliability. To improve reliability of software, coupling should be taken into consideration to minimize complexity of software [3]. A lot of coupling metrics to measure the coupling between two classes or objects but there is no cognitive weighted coupling metric to measure the different type of coupling proposed by various researchers. So, there is a need for cognitive weighted coupling metric for the class level coupling measurement. Hence our main goal is to define a cognitive weighted coupling metric to measure the coupling at the various levels.

## II. Literature Review

Several metrics have been proposed for OO systems by researchers. A metric suite proposed by Chindamber and Kemerer (C&K) is one of the best known suites of OO metrics. The six metrics proposed by CK are Weighted Method per Class (WMC), Depth of Inheritance Tree (DIT), Response For a Class (RFC), Number Of Children (NOC), Lack of Cohesion of Methods (LOCM) and Coupling Between Objects (CBO) [4] [5]. Parvinder Singh Sandhu and Dr. Hardeep Singh [6] have proposed a research that gives the evaluation of CK suite of metrics and suggests the refinements and extensions to these metrics so that these metrics should reflect accurate and precise results for OO based systems. Raed Shatnawi [7] has proposed a research that identifies the threshold values for CBO, RFC and WMC at two levels of risks using a quantitative methodology based on the logistic regression curve. These threshold values can be used to identify the most error-prone classes. Hitz and Montazeri [8] argue that coupling between two classes should be multi-faceted rather than being a singular relation. In other words, there should be many aspects taken into account when measuring the coupling relationship between classes within a system. Briand et al. [7] [9] [10] identify eighteen distinct aspects of coupling with each focusing on a different type of relationship. These relationships are finer-grained than previous approaches where they tend to only pay attention to method-method, class-method, class-attribute, etc. Li and Henry [11] propose two additions to the existing CK suite of metrics. Message Passing Coupling (MPC) is the number of messages (method invocations) a class sends to other classes.

CBO has been shown to be correlated to class quality (defect or error-proneness of a class) [6] [12] [13] [14] [15] and [16]. First, Gui and Scott argue that some metrics like CBO treat coupling between a pair of classes as a binary relation-either they have one or not. There is no distinction between a strong and weak relation. Second, the metrics do not consider the various type of coupling complexity of the classes that were proposed by Edward Berard [17]. CBO is explained in section 3, the motivation of proposed metric is discussed in section 4, Calibration of Types of coupling is discussed in section 5, The proposed metric CWCBO is explained in section 6, the experimentation of a new metric and the case study is described in section 7, a comparative study of CWCBO with CBO in section 8 and Section 9 presents the conclusion and future work.

## III. Coupling between Objects (CBO)

Chidamber and Kemerer (CK) introduced a metric suite to measure testability, maintenance, and reusability of a class but without any empirical validation. CK define Coupling Between Objects (CBO) for a class to be the count of the number of other classes to which it is directly coupled. This number represents an object's fan-out to external objects. The metric's basis is in the fact that if an object is coupled to another it uses another's methods or instance variables.

Stevens et al. [18] introduced the concept of coupling into structured design. He defined coupling to be "the measure of the strength of association established by a connection from one module to another." This infers that highly coupled classes are not desired as it is considered bad design and can lead to difficulty understanding classes. As their degree of coupling increases so does the complexity of the class. This results with the module becoming increasingly dependent on external classes to implement its functionality and is bound to reflect any changes the external classes may undergo in future maintenance.

Coupling Between Objects (CBO) for a class is a count of the number of other classes to which it is coupled. This definition is flexible in three ways.

- Which direction a class is coupled to another
- How a class is actually coupled to another
- The value to give a coupling relationship to distinguish its strength from another coupling

CBO count only outward coupling to foreign classes. The way two classes are coupled will follow the same definition as before in this section. The value that will be given to the coupling will be defaulted to one, but this research will experiment with various other values as well. These variations will be the novel part of the proposed metric.

Edward Berard [17] has proposed various types of coupling which are defined as follows:

The following section discus the motivation derived from the literature reviewed.

| | |
|---|---|
| **Control Coupling :** | Passing control flags between modules so that one module controls the sequencing of the processing steps in another module. |
| **Global Data Coupling :** | Two or more modules share the same global data structures. |
| **Internal Data Coupling :** | One module directly modifies local data of another module. |
| **Data Coupling :** | Output from one module is the input to another Using parameter lists to pass items between routines |
| **Lexical Content Coupling :** | Some or all of the contents of one module are included in the contents of another. |

## IV. Motivation

Creating software is complex and increasingly expensive to develop [19]. The maintenance phase of software is by far the most costly part in the software life cycle [20]. Being able to reduce potential defects as well as increasing ease of maintenance through software metrics creates a huge interest in the applicability of metrics. The two metrics (CWCBO and CBO) offer varying degrees of aspects measured within a software system. CBO is a measurement which can be interpreted to show the reusability of a component and its proneness to change in the future. This proneness to change is caused through its extensive coupling throughout the system. If one object is modified where the coupled object relied on the preexisting behavior previously, then there is a subtle defect that is potentially introduced.

The CWCBO includes various types of coupling. The various types coupling are Control Coupling (CC), Global Data Coupling (GDC), Internal Data Coupling (IDC), Data Coupling (DC) and Lexical Content Coupling (LCC) [17].The cognitive weight of these types of couplings is calibrated by means of psychological experiments conducted to students of masters and bachelors degree. With the new metric there is hope that this metric will reflect the real complexity of OO system. This results in a measure that will be able to indicate an object that is highly coupled to methods of another class. This can lead developers to rethink particular components that can be re-factored into more maintainable modules or indicate the complexity of reusing a component to get another system.

## V. Calibration

In this section, an experiment is conducted to assign cognitive weight to the various type of couplings discussed in section 3. A comprehension test has been conducted for a group of students to find out the time taken to understand complexity of object oriented program with respect to different types of coupling. The group of students selected had sufficient exposure in analyzing the object oriented programs, as they had undergone courses in Java language. 30 students who scored 65% and above in the semester Examination were selected to participate in the comprehension test.

The time taken by students to comprehend the programs was recorded after the completion of each program. The time taken for comprehension of all these programs was noted and the mean time to comprehend was calculated. Two different programs have been administered in each case, totally ten different mean timings were recorded. Average time was calculated for each program from the individual time taken by students which shows in Figure 1.

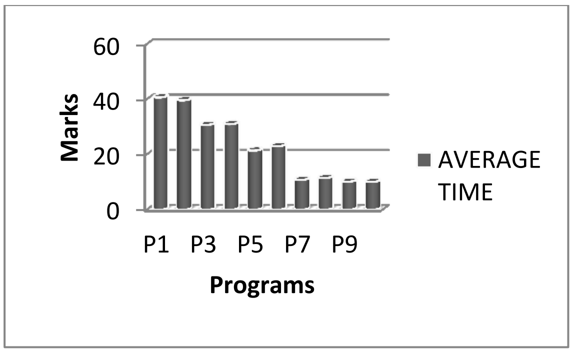

Fig. 1: Average comprehension time for each program

Table 1: Categorized mean comprehension time

| Programs | Average Comprehension Time | Category | Average Comprehension Time |
|---|---|---|---|
| 1 | 40.7 | LCC | 40.18333 |
| 2 | 39.66667 | | |
| 3 | 30.76667 | DC | 30.88333 |
| 4 | 31 | | |
| 5 | 21.43333 | IDC | 22.21667 |
| 6 | 23 | | |
| 7 | 10.8 | GDC | 11.13333 |
| 8 | 11.46667 | | |
| 9 | 10.16667 | CC | 10.11667 |
| 10 | 10.06667 | | |

In Table 1, the average comprehension times, for programs are listed. These programs are based on object oriented programming. The mean time is also calculated for each category of the programs and is tabulated. From the above table, it’s clear that, the mean time of LCC is higher than which in turn is higher than DFC, that implies the cognitive load to understand the LCC is greater than DC, EDC, GDC, CC.

## VI. Cognitive Weighted Coupling Between Object (CWCBO)

The proposed metric called Cognitive Weighted Coupling Between Objects (CWCBO), which considers the cognitive complexity of the different types of coupling such as data coupling, control coupling, global coupling and interface coupling. "Unnecessary object coupling needlessly decreases the reusability of the coupled objects", "Unnecessary object coupling also increases the chances of system corruption when changes are made to one or more of the coupled objects". The exiting CBO metric proposed by C.K uses the count of number of objects the current classes coupled. Each couple is assign a weight 1. This metric does not considered the various types of coupling. CWCBO can be calculated by using the Equation as follows,

$$cwcbo = ((\mathrm{CC} * \mathrm{WFCC}) + (\mathrm{GDC} * \mathrm{WFGDC}) + (\mathrm{IDC} * \mathrm{WFIDC}) + (\mathrm{DC} * \mathrm{WFDC}) + (\mathrm{LCC} * \mathrm{WFLCC}) \quad (1)$$

Where

| | |
|---|---|
| CC | is the total number of modules that contains Control Coupling |
| GDC | is the count of Global Data Coupling |
| IDC | is the count of Internal Data Coupling |
| DC | is the count of Data Coupling |
| LCC | is count of Lexical Content Coupling |

The Weighting Factor of each type of coupling is calibrated using the method discuss in the previous section and the values are given as follows,

| | |
|---|---|
| WFCC | is the Weighting Factor of Control Coupling |
| WFGDC | is the Weighting Factor of Global Data Coupling and its weight is given as 1 |
| WFIDC | is the Weighting Factor of Internal Data Coupling and its weight is given as 2 |

| | |
|---|---|
| WFDC | is the Weighting Factor of Data Coupling and its weight is given as 3 |
| WFLCC | is the Weighting Factor of Lexical Content Coupling and its weight is given as 4 |

If there are many classes namely CWCBO is the some of all CWCBO for individual classes. The following section explains how CWCBO is calculated by means of a case study.

## VII. Case Study

The proposed complexity metric given by Eq 1 is evaluated with the following program.

**Program:**

```
import java.io.*;
class bank
{

DataInputStream in=new
DataInputStream(System.in);
int accno,amtde;
String name,acctype;
            //GLOBAL
DATA  COUPLING//(Sharing global
variables)
void getdata()throws IOException
{
System.out.println("Enter the Account No:");
accno=Integer.parseInt(in.readLine());
System.out.println("Enter the Account
Type:");
acctype=in.readLine();
System.out.println("Enter the Customer
Name:");
name=in.readLine();
System.out.println("Enter the Initial
Deposit:");
amtde=Integer.parseInt(in.readLine());
}
void display()
{
System.out.println("******************");
System.out.println(" Account No:"+accno);
System.out.println("Account  Type:"+acctype);
System.out.println("Customer Name:"+name);
System.out.println("Initial Deposit:"+amtde);
}}
class withdraw extends bank
{
int wd;
void getin()throws IOException
{
System.out.println("Enter the amount to be
withdrawn:");
wd=Integer.parseInt(in.readLine());
}
void show()
{
if(wd<=amtde)           //DATA COUPLING//
(passing variables again for use )
 {
amtde-=wd;
System.out.println("Balance after
withdrawal:"+amtde);
System.out.println(" ******************");
}
else   //CONTROL COUPLING//(using true
or false values)
{
System.out.println("You cannot withdraw this
amount");
System.out.println("
*********************");
}}}
class deposit extends bank
{
int dt;
void get()throws IOException
{
System.out.println("Enter the amount to be
deposited:");
dt=Integer.parseInt(in.readLine());
}
void print()
{
amtde+=dt;                //LEXICAL
CONTENT COUPLING//(Same content)
System.out.println("Balance after
deposit:"+amtde);
System.out.println("*******************"
);
}}
class bankacc
{
public static void main(String args[])throws
IOException
{
DataInputStream in=new
DataInputStream(System.in);
System.out.println("Enter the choice:");
int op=Integer.parseInt(in.readLine());
switch(op)
{
case 1:
      withdraw w=new withdraw();
       w.getdata();
       w.getin();
       w.display();
       w.show();
       break;
case 2:
      deposit d=new deposit();
     d.getdata();   // INTERNAL DATA
COUPLING/   (Modifying same method
with   different     objects)
      d.get();
      d.display();
```

```
            d.print();
            break;
      default:
            System.out.println("Enter the choice 1 or
      2");
            break;
      }}}
```

**CBO:**

CBO=CC+GDC+IDC+DC+LCC

CBO=1+1+1+1+1

CBO=5

**CWCBO:**

$CWCBO= (CC*WF_{CC}) + (GDC*WF_{GDC}) + (IDC*WF_{IDC}) + (DC*WF_{DC}) + (LCC*WF_{LCC})$

CWCBO = (1*1) + (1*1) + (1*2) + (1*3) + (1*4)

CWCBO=1+1+2+3+4=11

Table 2: Coupling Complexity metric value for the above program

| Program # | CBO | CWCBO |
|---|---|---|
| 1 | 5 | 11 |

## VIII. Analytical Evaluation of CWCBO

Several researchers have recommended properties that software metrics should possess to increase their usefulness. For example, Basili and Reiter [21] suggest that metrics should be sensitive to externally observable differences in the development environment, and must also correspond to intuitive notions about the characteristic differences between the software artifacts being measured. Weyuker [22] has developed a formal list of properties for software metrics and has evaluated a number of existing software metrics using these properties. These properties include notions of monotonicity, interaction, non-coarseness, non-uniqueness and permutation. He developed nine properties.

In this section, the new metric CWCBO is analyzed and evaluated against the properties of metrics defined by Weyuker as discussed in the previous section. This analytical evaluation explains how the proposed metric satisfies or not satisfies those properties.

- ***Non-coarseness:*** Not all class can have the same CWCBO since the number of class that it interacts with varies for different class. Hence this property is satisfied.
- ***Granularity:*** Since the number of class of any large scale system is always finite, the number of class having the same CWCBO is also finite. Hence this property is satisfied.
- ***Non-uniqueness (Notion of Equivalence):*** A class can have the same number of interactions with the rest of the class as another class thus having the equal value of CWCBO (s1, s2) can be equal to CWCBO (s3, s2), since both s1 and s3 can have the same level of interactions with s2. Therefore this property is satisfied.
- ***Design Details are Important:*** Inter-class coupling occurs when methods of one class use methods of another class, i.e., coupling depends on the manner in which methods are designed and not on the functionality provided by the class. Therefore this property is satisfied.
- ***Monotonicity:*** Let A and B be two class with CWCBO (A) = p and CWCBO (B) = q. If A, and B are combined, the resulting class will have p + q – r couples, where r is the number of couples reduced due to the combination i.e., CWCBO (A+B) = CWCBO (A) + CWCBO (B) – (CWCBO (A, B) + CWCBO (B, A)). If A and B are highly coupled i.e., r is very high, CWCBO (A+B) may be less than CWCBO (A) or CWCBO (B). Hence this property is not satisfied for CSL. Whereas for any three class, A, B and C, if A and B are combined, then CB CWCBO S (A+B, C) ≥ CWCBO (A, C) and CWCBO (A+B, C) ≥ CWCBO (B, C). Thus this property holds good for CWCBO.
- ***Non-Equivalence of Interaction:*** For all class A, B and C, let CWCBO (A) = CWCBO (B). CWCBO (A + C) = CWCBO (A) + CWCBO (C) – CWCBO (A, C) and CWCBO (B+C) = CWCBO (B) + CWCBO (C) – CWCBO (B, C). Since CWCBO (A, C) and CWCBO (B, C) may not be equal, CWCBO (A+C) is not necessarily equal to CWCBO (B+C). This means that interaction between A and C can be different than interaction between B and C resulting in different complexity values for A+C and B+C. Thus this property is satisfied.
- ***Permutation:*** This property holds good only for structured programming, since order of methods need not be the same as the order of execution in object-oriented systems.
- ***Renaming:*** CWCBO does not depend on the name of the class and depends only on the details of the implementation. Hence renaming a class does not affect the CWCBO. Hence this property holds good.
- ***Interaction Increases Complexity:*** Let A, B and C be any three class with CWCBO (A) = p and CWCBO (B) = q. If A, and B are combined, the resulting subsystem will have p + q – r couples, where r is the number of couples reduced due to the combination. That is CWCBO (A+B) = CWCBO (A) + CWCBO (B) – (CWCBO (A, B) + CWCBO (B, A)). Since CWCBO (A, B) and CWCBO (B, A) are non-negative, SC (A) + CWCBO (B) > SC (A+B). But, CWCBO (A, C) + CWCBO (B, C) ≤

CWCBO (AB, C) is possible for some cases. Hence this property is not satisfied for SC and satisfied only for CWCBO

The CWCBO metric satisfies almost all of the Weyuker's properties. Though it does not satisfy two of those properties, these two properties clearly shows that not being satisfied actually decreases the complexity. Hence, the metric is proven to be a valid metric for object oriented system.

## IX. Comparative Study

A comparative study has been made with most widely accepted CK metric suite [5] and found that CBO metrics proposed by CK did not provide the total complexity of the class by considering the cognitive complexity due to message Coupling Between Object of that class. This differentiates CWCBO from the CK metrics. The current CWCBO metric is one step ahead of CK's CBO, because it includes the complexity that arises due to the various types of Coupling Between Object. Another advantage of CWCBO metric is that, it takes cognitive weights into consideration. In order to compare the proposed metric a comprehension test was conducted to bachelors and master degree students. There were sixty students who participated in the test; the students were given five different programs in java for the comprehension test. Thetest was to find out the output of the given programs. The time taken to complete the test in minutes is recorded. The average time taken by all the students is calculated. In the following Table 3, a comparison has been demonstrated with CBO, CWCBO and the comprehension test result.

Table 3: Complexity metric values and mean comprehension time

| Program # | CBO | CWCBO | Mean Comprehension Time |
|---|---|---|---|
| 1 | 16 | 16 | 19.6 |
| 2 | 7 | 13 | 13.5 |
| 3 | 12 | 12 | 15.7 |
| 4 | 14 | 20 | 20.5 |
| 5 | 12 | 27 | 22.2 |

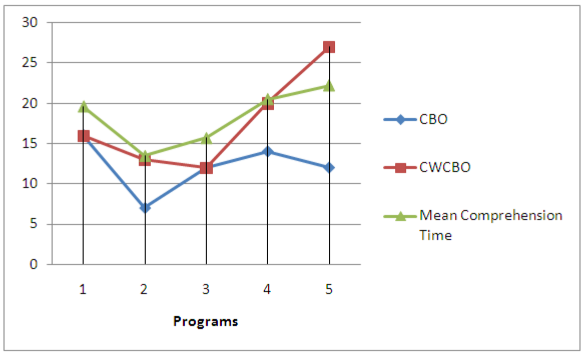

Fig.2: Complexity metric values Vs mean comprehension time

The coupling complexity of the class is calculated by computing Control Coupling (CC), Global Data Coupling (GDC), Internal Data Coupling (IDC), Data Coupling (DC) and Lexical Content Coupling(LCC). This is better indicator than the CK's CBO. The weight of each type of coupling is calculated by using cognitive weights and weighting factor of type of the coupling similar to which is suggested by Wang et al. It is found that the resulting value of CWCBO is larger than the CBO. This is because, in CBO, the weight of each coupling is assumed to be one. However, including cognitive weights for calculation of the CWCBO is more realistic because it provides for the complexity of the internal architecture of Coupling Between Object. The results are shown in the Table 3. A correlation analysis was performed between CBO Vs Comprehension Time with r = 0.699243 and CWCBO Vs Comprehension time with r = 0.872378. CWCBO has more positively correlated than CBO. From the table 3, it is observed that CWCBO value is larger than CBO value which concludes Chat CWCBO is a better indicator of complexity of the classes with various types of coupling.

## X. Conclusion and Future Work

A CWCBO metric for measuring the class level complexity has been formulated. The complexity of the class includes the coupling complexity of the class. CWCBO includes the cognitive complexity due to different types of coupling. CWCBO has proven that, complexity of the class getting affected, which is based on the cognitive weights of the various types of coupling. The assigned cognitive weight of the various types of coupling is validated using the comprehension test and found that the cognitive load to understand the LCC is larger than CC, GDC and IDC. The metric is evaluated through a case study and a comparative study, and proved to be a better indicator of the class level complexity. A tool is to be developed for calculating the CWCBO value and to compare it with CK metrics. Newer metrics may also be proposed and validated for assessing the cognitive complexity of other object oriented features.

## Reference

[1] Xu, B.Randell. J, C.M.F. Rubira, and J. R.Stroud, "Towards an Object Oriented Approach to Software Fault Tolerance",IEEE, ISBN:!0-792-38069-X, pp. 226-232, 1995.

[2] Yadav. A and Khan. R .A, "Measuring Design Complexity–An Inherited Method Perspective", SIGSOFT Software Engineering Notes, 24 No.4,pp: 1-5, july 2009.

[3] Yadav. A, and Khan. R. A, "Complexity:A Reliability Factor", IEEE International Advance Computing Conference (IACC-2009), Patiala, India, pp.2375-2378,6-7 March 2009.

[4] Mc Quillan. J. A and Power. J. F, "On the application of software metrics to UML model," Lecture Notes in Computer Science, Vol. 4364, 2007, pp.217-226.
[5] Chidamber. S. R and Kemerer. C. F, "A Metric Suite for Object-Oriented Design", IEEE Trans. on Software Engineering, 1994, pp.476-493.
[6] Harrison. R, Counsell. S and Nithi. R, "Coupling metrics for object-oriented design," In Proceedings for the Fifth International Software Metrics Symposium, 1998. pages 150-157
[7] Raed Shatnawi "An Investigation of CK Metrics Thresholds" ISSRE Supplementary Conference Proceedings, 2006, pp.12-13.
[8] Hitz. M and Montazeri. B, " Measuring coupling and cohesion in objectoriented systems," In Proceedings of the International Symposium on Applied Corporate Computing, Monterrey, Mexico., 1995.
[9] Briand, L.C., J.W. Daly, and J.K. Wust, A Unified Framework for Coupling in Object-Oriented Systems. IEEE Transactions on Software Engineering, v.25(1): p. 91-121, 1999.
[10] Briand. L, Wiist. J and Lounis. H, "Using coupling measurement for impact analysis in object-oriented systems," In Proceedings of the 19th International Conference on Software Maintenance, Oxford, UK, , 1999, pages 475-482
[11] Li. W. and Henry. S, "Object-oriented metrics that predict maintainability," Journal of Systems and Software , 1993, 23:111-122
[12] Basili. V. R, Briand. L. C, and Melo. W. L, "A validation of object-oriented design metrics as quality indicators," IEEE Transactions on Software Engineering, 1996, 22(10):751-761,
[13] Wilkie. F. and Kitchenham. B, "Coupling measures and change ripples in c++ application software", Journal of Systems and Software, 2000, 52(2-3):157-164
[14] Wilkie. F. and Kitchenham. B, "An investigation of coupling, reuse, and maintenance in a commercial c++ application", Information and Software Technology, 2001, 43(13):801-812
[15] Olague. H. M, Etzkorn. L. H, Gholston. S and Quattlebaum. S, "Empirical validation of three software metric suites to predict fault-proneness of objectoriented classes developed using highly iterative or agile software development processes," IEEE Transactions on Software Engineering, 2007, 33(6):402-419
[16] Gyimthy. T, Ferenc. R and Siket. I, "Empirical validation of object-oriented metrics on open source software for fault prediction," IEEE Transactions on Software 8Engineering, 2005, 31(10):897-910
[17] Edward Berard. V "Essays on object-oriented software engineering (vol. 1)" Berard Software Engineering, Prentice-Hall, 1993, ISBN:0-13-288895-5, 1993
[18] Stevens. W, Myers. G and Constantine. L, "Structured design," IBM Systems Journal, 1974, 13(2):115-139
[19] Kemerer. C. F, "An empirical validation of software cost estimation models," Communications of the ACM, 1987, 30(5):416-429
[20] Pearse. T and Oman. P, "Maintainability measurements on industrial source code maintenance activities," In ICSM '95: Proceedings of the International Conference on Software Maintenance, 1995, page 295
[21] Basili, V. and R. Reiter, Evaluating automatable measures of software Models. in IEEE Workshop on Quantitative Software Models, NY, p. 107-116, 1979,
[22] Weyuker, E., Evaluating software complexity measures. IEEE Transactions on Software Engineering, v.14: p. 1357-1365, 1988.

Mr. **A. Aloysius** is working as Assistant Professor in Department of Computer Science, St.Joseph's College (Autonomous), Tiruchirappalli, Tamil Nadu, India. He has 12 years of experience in teaching and research. He has published many research articles in the National / International conferences and journals. He has also presented 2 research articles in the International Conferences on Computational Intelligence and Cognitive Informatics in Indonesia. He has acted as a chair person for many national and international conferences. He is currently pursuing doctor of philosophy programme and his current area of research is cognitive aspects in software design.

**Dr. L. Arockiam** is working as Associate Professor in the Department of Computer Science, St.Joseph's College (Autonomous), Tiruchirappalli, Tamil Nadu, India. He has 23 years of experience in teaching and 15 years of experience in research. He has published 109 research articles in the International / National Conferences and Journals. He has also presented 2 research articles in the Software Measurement European Forum in Rome. He has chaired many technical sessions and delivered invited talks in National and International Conferences. He has authored a book on "Success through Soft Skills". His research interests are: Software Measurement, Cognitive Aspects in Programming, Web Mining and Mobile Networks.