Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 04, April-2016

Design and Analysis of Aspect Oriented Metric CW CAE using Cognitive Approach

G. Arockia Sahaya Sheela
Dept. of Computer Science
Holy Cross College Trichy
TamilNadu India.

A. Aloysius
Dept. of Computer Science
St. Joseph's College Trichy
TamilNadu India.

K. R. Martin
Dept. of Computer Science
St. Joseph's College Trichy
TamilNadu India.

***Abstract* - Aspect Oriented Systems (AOS) in order to evaluate their quality, gain importance as the paradigm continues to increase in popularity. Aspect Oriented Programming (AOP) emphasizes the creation of aspects, which are modules that centralize distributed functionality. AOP is one of the most promising solutions to the problem of creating clean, well encapsulated objects without extraneous functionality. Consequently, several Aspect- oriented metrics have been proposed to evaluate different aspects of these systems. This paper presents a new cognitive complexity metric namely "Design and Analysis of Aspect Oriented Metric CWCAE using Cognitive Approach" in Aspect Oriented System. This paper addresses the Cognitive Weighted Coupling on Advice Execution (CWCAE) metric to measure the different type of joint points.**

Keywords: Aspect Oriented Systems (AOS), Aspect Oriented Programming (AOP), Cognitive Approach, Advice, metric, Join Point, Coupling.

I. Introduction

Software engineering is the study and application of engineering to the design, development, and maintenance of software. Software metric is a measure of some property of a piece of software or its specifications. Metrics attempt to measure a particular aspect of a software 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.

In computing, aspect-oriented programming is a programming paradigm that aims to increase modularity (a grouping of related code) by allowing the separation of cross-cutting concerns. AOP forms a basis for aspect-oriented software development. Out of the available AOP languages, AspectJ is the most popular and mostly used in research areas. AspectJ is a simple general purpose extension to Java that provides, through the definition of new constructors, support for modular implementation of crosscutting concerns. AspectJ has been successfully used to cleanly modularize implementations of crosscutting concerns such as synchronization, consistency checking, protocol management and others.

The aspect is the modular unit of crosscutting implementation. Each aspect encapsulates functionality that crosscuts other classes in an AspectJ program. A central concept in the composition of an aspect with other classes is called a join point. A join point is a well-defined point in the execution of a program, such as a call to a method, an access to an attribute, an object initialization, an exception handler etc.

AspectJ has no Cognitive Weighted Coupling on Advice Execution (CWCAE) metric to measure the different type of Join Points proposed by various researchers. So, there is a need for cognitive weighted CAE for the Aspect level measurement. Hence our main goal is to define a Cognitive Weighted Coupling on Advice Execution (CWCAE) metric to measure the Complexity of various types of Joint Points.

II. Literature Review

Several metrics have been proposed for AOP systems by researchers. One of the metric proposed by Ceccato et.al [9] and KotrappaSirbi et.al [7] is CAE. Coupling on Advice Execution (CAE) is a number of aspects containing advices possibly triggered by the execution of operations in a given module. Such kind of coupling is absent in Object Oriented (OO) systems.

Bartsch and Harrison [11] suggested, all join points that can cause advice to be executed. AspectJ supports more types of join points that can also cause the execution of advice, such as object initialization join points, exception handler join points, call join points and advice execution join points. A valid measure of coupling on advice execution needs to count all of these join point coupling mechanisms.

WJP metric proposed by Parthipan, SenthilVelan, ChitraBabu [1]. The WJP per class or aspect is the sum of cognitive weights of types of join points shadow in classes or aspects. The cognitive weight assigned to the identified designators is based on its cognitive complexity. The drawback of the WJP metric is that they didn't prove their metric according to the statistical approach and data are not accurate. Because of empirical data collection, the data doesn't satisfy the Fenton et al. [8] properties.

The motivation of proposed metric is discussed in section 5, Empirical Metric Data Collection & Evaluation Criteria 6, the experimentation of a new metric and the case study is described in section 8, a comparative study of CWCAE with CAE in section 9 and Section 10 presents the conclusion and future work.

III. ASPECTJ

AspectJ is an implementation of AOP for the Java language built as an extension to the language. A compiler

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 04, April-2016

and a set of JAR files take common Java code and AspectJ aspects and compile them into standard Java byte-code, which can be executed on any Java-compliant machine. Followings are some of the concepts in AspectJ.

- Join point—A predictable point in the execution of an application.
- Pointcut—A structure designed to identify and select join points within an AspectJ program.
- Advice—Code to be executed when a join point is reached in the application code.
- Inter-type declarations—A powerful mechanism to add attributes and methods to previously established classes.
- Aspect—A structure analogous to a Java class that encapsulates join points, pointcuts, advice, and inter-type declarations.

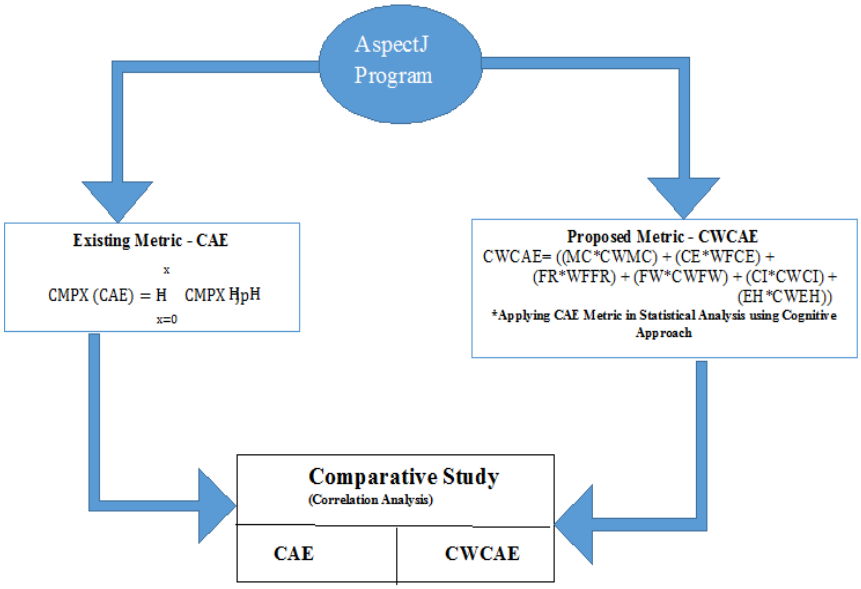
IV. Metric Analysis

A. Existing work

Coupling on Advice Execution (CAE) is a number of aspects containing advices possibly triggered by the execution of operations in a given module. If the behavior of an operation can be altered by an aspect advice, due to a pointcut intercepting it, there is an (implicit) dependence of the operation from the advice. Thus, the given module is coupled with the aspect containing the advice and a change of the latter might impact the former. Such kind of coupling is absent in Object Oriented (OO) systems.

B. Proposed work

Several metrics have been proposed for AOP systems by researchers. One of the metric proposed by Ceccato et.al [9] and KotrappaSirbi et.al [7] is CAE. CAE Ananthi et.al [12] counts the number of aspects containing advices possibly triggered by the execution of methods, advices or method intertype declarations, attribute and attribute intertype declarations in a given class or aspect. AspectJ supports more types of join points that can also cause the execution of advice, such as object initialization join points, exception handler join points, call join points and advice execution join points. This metric does not considered the various types of join point. The proposed metric called Cognitive Weighted CAE (CWCAE), considers the cognitive complexity of the different types of joint points.



Assessment Framework

V. EXPERIMENTAL DESIGN

This section discusses the chosen metric for the analysis, Cognitive Approach used to gather data.

A. Calibration

In this section, an experiment is conducted to assign cognitive weight to the various types of join point. A comprehension test has been conducted for a group of students to find out the time taken to understand complexity of aspect oriented program with respect to different types of join point. The group of students selected had sufficient exposure in analysing the aspect oriented programs, as they had undergone courses in AspectJ language. 30 students from Rural, 30 students from Urban 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. Five different programs were administered in each case, totally fifteen different mean timings were recorded. Average time was calculated for each program from the individual time taken by students which is shown in Table 1.

The average comprehension time, for programs are listed in table1. These programs are based on Aspect Oriented Programming. The mean time is also calculated for each category of the programs and is tabulated.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 04, April-2016

Table 1 Categorized mean comprehension time

Programs	Average Comprehension Time (In Minutes)					
	Method Call(MC) / Constructor Call(CC)	Method Execution(ME) / Constructor Execution(CE)	Field Read Access(FR)	Field Write Access (FW)	Class Initialization (CI)	Exception Handler Execution (EH)
P1	15	18	25	30	34	37
P2	14	19	23	28	33	39
P3	14	18	24	27	34	39
P4	12	17	23	27	31	36
P5	13	16	22	25	33	36
Mean value	13.6	17.6	23.4	27.4	33	37.4

VI. COGNITIVE WEIGHTED COUPLING ON ADVICE EXECUTION

The proposed metric called Cognitive Weighted Coupling on Advice Execution (CWCAE), which considers the cognitive complexity of the different types of joint points such as object initialization join points, exception handler join points, call join points and advice execution join points. The existing CAE metric proposed by Ceccato et.al [9] and KotrappaSirbi et.al [7] counts the number of aspects containing advices possibly triggered by the execution of methods or advice. This metric does not consider the various types of joint point. CWCAE can be calculated by using the Equation as follows,

CWCAE = ((MC*WFMC) + (ME*WFME) + (CC*WFCC) + (CE*WFCE) + (CI*WFCI) + (FR*WFFR) + (FW*WFFW) + (EH*WFEH)) ------- 1

Where,

MC is the count of Method Call Joint Point

ME is the count of Method Execution Joint Point

CC is the count of Constructor Call Joint Point

CE is the count of Constructor Execution Joint Point

CI is the count of Class Initialization Joint Point

FR is the count of Field Read Access Joint Point

FW is the count of Field Write Access Joint Point

EH is the count of Exception Handler Execution Joint Point

The Weighting Factor of each type of Joint Point is calibrated using the method discussed in the Empirical Metric Data Collection. The weight value is calculated based on the mean time and mean correlation time, to normalize the mean value to get appropriate weight value. Average mean value of each type of joint point is divided by corresponding mean correlation time. Finally weight value is calculated by dividing the values by 20 to reduce the range of values. The finalize weight values are given as follows,

Table 4 Weight Value of Each type of Advice

Joint Point	Weight Value
WFMC	1
WFME	1
WFCC	1.4
WFCE	1.4
WFFR	1.9
WFFW	2.3
WFCI	3
WFEH	3.7

Where,

WFMC is the Weighting Factor of Method Call Joint Point
WFME is the Weighting Factor of Method Execution Joint Point
WFCC is the Weighting Factor of Constructor Call Joint Point

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 04, April-2016

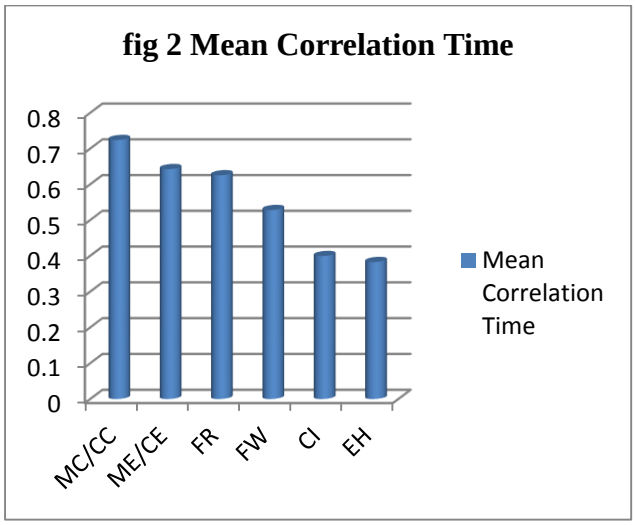
WFCE is the Weighting Factor of Constructor Execution Joint Point
WFCI is the Weighting Factor of Class Initialization Joint Point
WFFR is the Weighting Factor of Field Read Access Joint Point
WFFW is the Weighting Factor of Field Write Access Joint Point
WFEH is the Weighting Factor of Exception Handler Execution Joint Point

VII. STATISTICAL ANALYSIS

For statistical analysis, CAE metric is selected for AO software. This metric is used to find the complexity of various types of advice using Cognitive Approach. The relationship among the join points are evaluated and analyzed statistically. For each join point, mean was selected as a measure of correlation between other join points. Table 2 illustrates statistical computation of different types of join points.

Table 2 Correlation between Comprehensions time of different joint points

	MC / CC	ME / CE	FR	FW	CI	EH
MC / CC	1	0.8077	0.9231	0.6940	0.5718	0.3468
ME / CE	0.8077	1	0.5385	0.6940	0.0673	0.7473
FR	0.9231	0.5385	1	0.4627	0.7736	0.0534
FW	0.6940	0.6940	0.4627	1	0.0759	0.2408
CI	0.5718	0.0673	0.7736	0.0759	1	-0.0934
EH	0.3468	0.7473	0.0534	0.2408	-0.0934	1
Mean Correlation Value	0.7239	0.6425	0.6252	0.5279	0.3992	0.3825



Types of join points are compared on the basis of mean and correlation. One join point, mean was selected as a measure of correlation between other join points and used for evaluation. If the value of this correlation is high, it shows better indicator of complexity of the classes or aspects.

VIII. THEORETICAL ANALYSIS & DATA COLLECTION PROPERTIES

Fenton et al. [8] defined some properties which were used for the data collection process and are described as follows:

- *Accuracy*: The higher the difference between the actual data and measured data and the lower is the accuracy and vice-versa. The difference between CWCAE and CAE is lower so the accuracy is higher.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 04, April-2016

- *Replicability*: Means that the analysis can be done at different times by different people using the same setting. Data are taken from rural and urban PG students at different time.
- *Correctness*: According to the metrics definition data was collected.
- *Precision*: Data is expressed by number of decimal places. Less decimal place shows a lower accuracy. If the decimal place of the data is high (i.e. 0.5502), it shows a higher accuracy.
- *Consistency:* It counts the differences with the metric values when collected using different tools by different people.

The following section explains how CWCBO is calculated by means of a case study.

IX. ILLUSTRATION

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

Program:

A. Java program

```
public class Stack
{
static final int DEFAULT_CAPACITY=5;
private Object [] theArray;
privateinttopOfStack;
public Stack()
{
theArray = new Object[DEFAULT_CAPACITY];
topOfStack=-1;
}
public void push(Object x)
{
if (topOfStack+1 == theArray.length)
doubleArray();
topOfStack++;
theArray[topOfStack]=x;
}
public void pop() throws Exception
{
if (isEmpty())
throw new Exception("Stack pop");
topOfStack--;
}
public Object top() throws Exception
{
if (isEmpty())
throw new Exception("Stack top");
returntheArray[topOfStack];
}
publicbooleanisEmpty()
{
returntopOfStack==-1;
}
public void clear()
 {
topOfStack=-1;
}
publicintgetSize()
{
return topOfStack+1;
}
}
public static void main(String args[])
{
Stack stack = new Stack();
stack.push(new Integer(4));
try
{
System.out.println(stack.top());
}
catch(Exception e)
{
System.exit(1);
}
stack.push(new Integer(5));
stack.push(new Integer(6));
try
{
System.out.println(stack.top());
stack.pop();
}
catch(Exception e) { System.exit(1); }
System.out.println("Empty? : " + stack.isEmpty());
}
}
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 04, April-2016

B. AspectJ program

```
aspect  PointEX
{
pointcut field() : call(* Stack.push(..));
before() : field()
{
Stack stack = (Stack)thisJoinPoint.getTarget();
System.out.println(thisJoinPoint.toLongString() +
" Stack Size:" + stack.getSize());
}
pointcut field() : execution(Stack.new(..));
before() : field()
{
Stack stack = (Stack)thisJoinPoint.getTarget();
}
pointcut field() : set(private Object [] Stack.theArray);
before() : field()
 {
System.out.println("Attribute theArray set");
}
pointcut field() : get(private intStack.topOfStack);
before() : field()
{
System.out.println("Attribute topOfStack read");
}
pointcut field() : handler(Exception);
before() : field(s)
 {
System.out.println("Exception Thrown");
}
pointcut field() : staticinitialization(Stack);
before() : field()
 {
System.out.println(thisJoinPoint.getSignature());
}
}
```

CAE

$$\text{CMPX (CAE)} = \sum_{x=0}^{x} \text{CMPX (jp)}$$

x=6 so,

CMPX(CAE) = 6

CWCAE

CWCAE= ((MC*CWMC) + (CE*WFCE) + (FR*WFFR) + (FW*CWFW) + (CI*CWCI) + (EH*CWEH))

CWCAE= ((1*1) + (1*1.4) + (1*1.9) + (1*2.3) + (1*3) + (1*3.7))

CWCAE= 1 + 1.4 + 1.9 +2. 3 +3 +3.7 = 13.5

Table 5 Joint Point Complexity metric value for the above program

Program#	CAE	CWCAE
1	6	13.5

X. COMPARATIVE STUDY

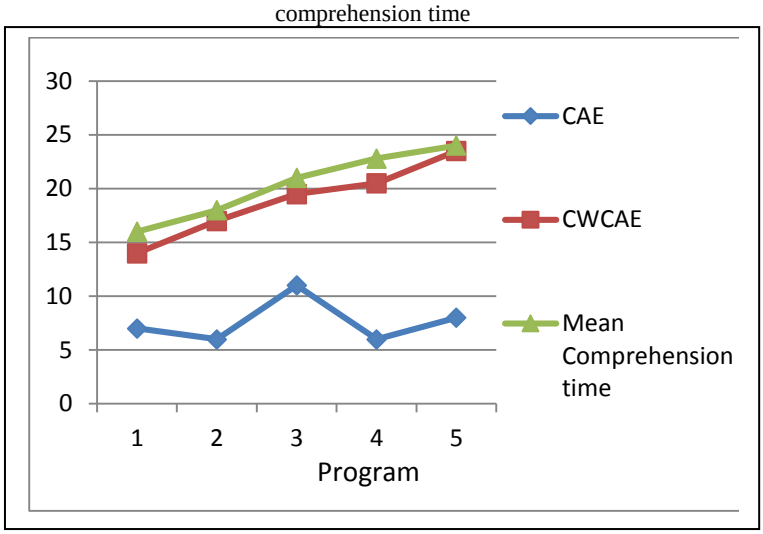
A comparative study has been made with most widely accepted metric proposed by Ceccato et.al [9] and KotrappaSirbi et.al [7] is CAE. CAE defines total number of the aspects containing advices possibly triggered by the execution of methods or advice. The current CWCAE metric is one step ahead of existing CAE metric, because it includes the complexity that arises due to the various types of Join Points. Another advantage of CWCAE metric is that, it takes cognitive weights into consideration and data collection satisfies the Fenton et.al [8] properties. In order to compare the proposed metric a comprehension test was conducted for rural and urban post graduate students. Sixty students participated in the test; the students were given five different programs in AspectJ for the comprehension test. The test 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 6, a comparison has been demonstrated with CAE, CWCAE of the comprehension test result.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 04, April-2016

Table 6 Complexity metric values and mean comprehension time

Program#	Existing Metric Value (CAE)	Proposed Metric Value (CWCAE)	Mean Comprehension Time
1	7	14	16
2	6	17	18
3	11	19.5	21
4	6	20.5	22.8
5	8	23.5	24

Fig 3 Complexity metric values Vs mean comprehension time



The CAE complexity of the class is calculated by computing Method Call(MC), Method Execution(ME), Constructor Call(CC), Constructor Execution(CE), Class Initialization(CI), Field Read Access(FR), Field Write Access(FW) and Exception Handler Execution(EH). This is better indicator than CAE. The weight of each type of Join Point is calculated by using cognitive weights and weighting factor of type of the Join Point similar to that suggested by Wang et al.[6] It is found that the resulting value of CWCAE is larger than the CAE. This is because, in CAE, the weight of each advice is assumed to be one. However, including cognitive weights for calculation of the CWCAE is more realistic because it considers different types of Joint Point. The results are shown in the Table 6. A correlation analysis was performed between CAE Vs Comprehension Time with r = 0.221981 and CWCAE Vs Comprehension time with r = 0.980778. CWCAE is more positively correlated than CAE. From the table 6, it is observed that CWCAE value is larger than CAE value which concludes that CWCAE is a better indicator of complexity of the classes with various types of Join Point in Advice Execution.

XI. CONCLUSION AND FUTURE WORK

A CWCAE metric for measuring the class level complexity has been formulated. The complexity of the class includes the Advice Execution complexity of the class. CWCAE includes the cognitive complexity due to different types of Joint Point. CWCAE has proven that, complexity of the class getting affected, is based on the cognitive weights of the various types of Joint Point. The assigned cognitive weight of the various types of Join Point is validated using the comprehension test and found that the cognitive load to understand the EH > CI > FW > FW > FR > CE, ME > CC, MC. The metric is evaluated through a statistical analysis, case study and a comparative study, and proved to be a better indicator of the class level complexity. Newer metrics may also be proposed and validated for assessing the cognitive complexity of another types of join point.

XII. REFERENCE

[1] Parthipan, SenthilVelan and ChitraBabu , "Design Level Metrics to Measure the Complexity Across Versions of AO Software", IEEE , 2014.
[2] A. Aloysius and G. Arockia Sahaya Sheela, **"Aspect Oriented Programming Metrics – A Survey"**, in the "International Journal of Emerging Trends in Computing and Communication Technology (IJETCCT)", Volume 1, No 3, August 2015.
[3] Chidamber S.R., Kemerer, C.F., "A metrics suit for object oriented design", IEEE, Trans. Software Engineering, vol.20, pp.476-498, 1994.
[4] G. Arockia Sahaya Sheela and A. Aloysius, "Analysis of Measuring the complexity of Advice using a Cognitive Approach", "International Journal of Applied Engineering Research (IJAER)", Vol. 10, No.82, 2015.
[5] Aloysius. A., "Coupling Complexity Metric: A Cognitive Approach", Modern Education and Computer Science, vol.9, pp.29-35, 2012.
[6] J. Shao and Y. Wang, "A new measure of software complexity based on cognitive weights.",Canadian Journal of Electrical and Computer Engineering, 2003.
[7] Kotrappa Sirbi and Prakash Jayanth Kulkarni, "Metrics for Aspect Oriented Programming-An Empirical Study", International Journal of Computer Applications, pp.17-23, 2010.
[8] N.Fenton, S.P.fleeger, "Software Metrics: A Rigorous and Practical Approach". PWS Publishing Company, 1997.
[9] Ceccato, M., and Tonella, P.: „Measuring the Effects of Software Aspectization", Proc. Workshop on Aspect Reverse Engineering (WARE 2004), Delft, The Netherlands, November 2004 (Cd-rom)
[10] Joseph D.Gradecki, Nicholas Lesiecki, "Mastering AspectJ – Aspect-Oriented Programming in Java",Wiley Publishing, Inc., 2003.
[11] Bartsch, M., Harrison, R, "An Evaluation of Coupling Measures for AspectJ", LATE Workshop AOSD, 2006.
[12] Ananthi Sheshasaayee, Roby Jose, "A Theoretical Framework for the Maintainability Model of Aspect Oriented Systems", Elsevier, SCSE, 2015.
[13] https://eclipse.org/aspectj/doc/next/progguide/semanticsadvice.html
[14] https://eclipse.org/aspectj/doc/released/progguide/startingaspectj.html

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 04, April-2016

G.Arockia Sahaya Sheela is working as Assistant Professor in Department of Computer Science, Holy Cross College (Autonomous), Tiruchirappalli, Tamil Nadu, India. She has obtained the Master of Computer Science degree in 2005 and Master of Philosophy degree in 2007 from Bharathidasan University, Trichy. She has 11 years of experience in teaching Computer Science. Her research areas are Sensor Networks and Software Metrics. She has published many research articles in the National/ International Conferences, and Journals. She is currently pursuing Doctor of Philosophy Program and her current area of research is the Cognitive Complexity of Aspect-Oriented Software Metrics.

A. Aloysius is working as Assistant Professor in Department of Computer Science, St. Joseph's College, Trichy, Tamil Nadu, India. He has got the Master of Computer Science degree in 1996, Master of Philosophy degree in 2004, and Doctor of Philosophy in Computer Science degree in 2013 from Bharathidasan University, Trichy. He has 15 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. His current area of research is Cognitive Aspects in Software Design, Big Data, and Cloud Computing.

K R Martin is working as Assistant Professor in Department of Computer Science, St. Joseph's College (Autonomous), Tiruchirappalli, Tamil Nadu, India. He has obtained the Master of Computer Applications degree in 1999, Master of Philosophy degree in 2007 and Master of Business Administration degree in 2011 from Bharathidasan University, Trichy. He has 17 years of experience in teaching Computer Science. His research areas are Software Engineering and Software Metrics. He has published many research articles in the National/International Conferences and Journals. He is currently pursuing Doctor of Philosophy Program and his current area of research is the Cognitive Complexity of Aspect-oriented Software Metrics.