I.J. Information Technology and Computer Science, **2015, 02, 12-20**
Published Online January 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2015.02.02

A Framework for Assessing the Software Reusability using Fuzzy Logic Approach for Aspect Oriented Software

Pradeep Kumar Singh[1], **Om Prakash Sangwan**[2]
Amity University Uttar Pradesh, Noida, India[1]; Gautam Buddha University, Gr. Noida, India[2]
Email: pradeep_84cs@yahoo.com, sangwan_op@yahoo.co.in

Amar Pal Singh[3], **Amrendra Pratap**[3]
Amity University Uttar Pradesh, Department of CSE, Noida, India[3]
Email: singhamarpal48@gmail.com, amrendra.bt11@gmail.com

Abstract**—**Software reusability is very important and crucial attribute to evaluate the system software. Due to incremental growth of software development, the software reusability comes under attention of many researcher and practitioner. It is pretty easier to reuse the software than developing the new software. Software reusability reduces the development time, cost and effort of software product. Software reusability define the depth to which a module can be reused again with very little or no modification. However the prediction of this quality attribute is cumbersome process. Aspect oriented software development is new approach that introduce the concerns to overcome the issues with modular programming and object oriented programming. However many researcher worked on accessing the software reusability on object oriented system but the software reusability of aspect oriented system is not completely explored. This paper explores the various metric that affects the reusability of aspect oriented software and estimate it using fuzzy logic approach.

Index Terms**—**Aspect Oriented Programming (AOP), Aspect Oriented Software Development (AOSD), Software Reusability**,** Cohesion, Coupling, Separation of Concern (SOC), Size and Complexity, Fuzzy Logic.

I. Introduction

Software reusability is software quality attribute in which software or its module is reused with very little and no modification. Software reuse is the development of existing software system by using their existing features or components. The goal of software reusability is to provide higher quality products, less development time, higher scheduling accuracy and Reliability. There are numerous software development methodologies to support reusability with different characteristics.

A. Module Oriented Approach (MOA)

Module oriented approach (MOA) is very dominant approach to develop the software. MOA is basically depends on perception of procedure call. Procedure can be also known as subroutine, method and function. It is appropriate choice than simple sequential or unstructured languages which involves complexity and requires significant amount of reusability.

A few significant characteristics and restrictions of Module Oriented Approach are listed below:

i. It provides reusability of code.
ii. MOA is robustly modular than structure.
iii. Development by writing or replaces the code.
iv. Inability to provide data binding with operations.

To overcome these limitations of MOA, Object Oriented Approach (OOA) is introduced.

B. Object Oriented Appraoch (OOA)

OOA is applicable for real world problems. OOA decompose problem into objects that abstract behavior and data into a single unit. It introduces the concept of inheritance, modularity, polymorphism and encapsulation. [1, 2].

A few significant characteristics and restrictions of Object Oriented Approach are listed below:

i. OOA approach mainly prominence on data relatively than procedure.
ii. Allows message passing and functions through interaction of objects.
iii. It provides the concept of classes, encapsulation and hides implementation details.
iv. OOA provides an easy way to add new data and functions, whenever required.
v. Ability to provide polymorphism and inheritances.

In practice with larger projects, it has been observed that OOA has difficulty to separate concerns such as readability, security, modifiability. To overcome the limitations of object oriented approach (OOA), Aspect Oriented (AO) approach is introduced.

C. Component Based Software Development (CBSD)

Nowadays CBSD is receiving as a new effective development paradigm in industry. This approach has wide acceptance in terms of cost effectiveness [3]. Software development in CBSD is a process of composition of diverse software components into integrated form. The main issue in CBSD is the reclamation and selection of appropriate software components from a huge variety of reusable components.

The major advantages of CBSB are as follow:

i. This technique provides in-time and high quality solutions for new software development.
ii. This approach also offers high productivity, flexibility and quality through reusability, replace-ability, efficient maintainability, and scale ability.
iii. This approach also reduces development time for new system.

D. Aspect Oriented Software Development (AOSD)

AOSD is reasonably forthcoming technique which combines concerns that crosscut the modularity from conventional programming approaches such as Module Oriented Approach (MOA) and Object Oriented Approach (OOA) [4, 5]. There are many languages that belongs to family aspect oriented systems such as AspectJ (Java Extension), AspectC (C Extension), AspectC++ (C++ Extension), AspectXML (XML Extension), CaserJ and HyperJ (used by IBM) [6, 7].

AspectJ is most popular one and few significant characteristics of AspectJ are listed below
i. It is most popular languages that used in AOP.
ii. Supports following characteristics such as joint points, pointcuts, advice and introduction [8, 9].
iii. Defines new constructor, maintain in favor of modular accomplishment for crosscutting concerns.
iv. It provides synchronization, consistency checking, protocol management and others.
v. The concern that crosscuts the modularity of traditional programming is known as cross-cutting concerns for examples cross-cutting includes logging, tracing, resource pooling, etc.

This paper is organized into five sections. Introduction followed by related work is mentioned in second part. Second section discusses the various reusability models given by famous researcher and their metrics that affect the reusability of software. The third section of this paper unfolds the brief description of metrics that affect the reusability of software. The fourth section of this paper evaluates the software reusability by using fuzzy logic approach. Finally, last section presents conclusion and future aspects of the proposed work.

II. Related Work

Nowadays, real world systems evolve so fast that they are capable to congregate challenges among the customer constraint as well as operational atmosphere. Design of software concern is well modularized to meet all the non-functional requirement of a system. AOSD still faces several problems; quantitative assessment of some important characteristics like modularity, complexity and the overall quality of aspect oriented technology is fairly unexploded. The related work will discussed the work done so far by many researchers in the area of object oriented approach and aspect oriented along with their metrics for assessing software reusability [5].

ISO developed a quality model that was further known as ISO/IEC 9126 Model [10]. According to ISO, quality of software is consisted into six attributes namely functionality, reliability, usability, efficiency, maintainability, portability. This model is an enhancement of earlier work which is specified by McCall, Bohem, and FURPS.

In Dromey's Quality Model, he integrates two quality attributes namely reusability and process maturity in ISO/IEC 1926 quality model [11].

Kumar *et al.* projected first quality model known as Aspect-Oriented Software Quality Model (AOSQUAMO) [12]. This model is an enhancement of ISO/IEC 9126 Quality Model. In this model Complexity, Reusability, Modularity and Code-Reducibility are included in diverse characteristics of ISO/IEC 9126 Model.

Castillo *et al.* proposed a conceptual quality model to elucidate the AOSD evolving technologies like aspect, composition concern, and quality requirement for software system [13]. REASQ Model is another name of this model which integrates ISO/IEC 9126 Model and ISO/IEC 25030 model in addition to articulated in UML.

Kumar P. introduced new attribute to AOSQUAMO Model i.e. evolvability [14]. The sub- characteristics of Evolvability is sustainability, extensibility, Configurability and Design Stability. This model is known as AOSQ Model.

Price *et al.* offered a new dimension for assessing the reusability measurements [15]. This dimension facilitates the large scale object oriented structure reuse. The researcher must have good understanding of application domain and type of system that they expect to devlop in future.

Barnard *et al.* projected a new reusability metric for OO software which is based on empirical evidence taken from well used accepted programming libraries in classes are assumed to be highly reusable [16].

Dandashi *et al.* reported a measurement in support of reusability of C++ code by actions of non-functional quality attributes like Completeness, Understandability, Adaptability and Maintainability [17].

A framework is projected by Sant'Anna *et al.* for estimating the reusability and maintainability of AO software [18]. They established a relationship between reusability and maintainability internal metrics such as coupling, cohesion, crosscutting concern and size. This concern driven architecture proposed by Sant'anna *et al.* [18] also help in measuring the SoC, Cohesion, Coupling and Complexity for AO system.

Cunha *et al.* discussed the high level concurrency patterns and mechanisms coded in AspectJ [19]. They explored some advantages of reusability, modularity and understandability. They accomplished that above mentioned characteristics will progress the AO approach while applying with java program.

Zhang *et al.* proposed AO approach with connectors in reusable design and implementation of connectors [20]. They uncover the affect of crosscutting concerns in reusability of connectors.

Aljasseret *et al.* proposed the extension of AspectJ programming languages known as ParaAJ [21]. This language has ability to parameterize aspects. This language is upcoming step to modularize aspects and enhancement in reusability.

Zhao proposed a coupling metrics which is based on no. of dependencies among aspects and classes like module-class, attribute-class, aspect-inheritance and module-method dependencies [22].

Ceccato *et al.* presented the coupling metrics for Aspect Oriented Programming languages [23]. They present a framework for aspects, classes, modules and term operation as class methods and aspect advices/introductions.

Bortsch *et al.* extended the framework from OO systems (Briand *et al.*) [24]. In their extended framework they contain a specific definition of six different coupling schemes such as locus of impact types of connection, firmness of server, granularity, straight or undirected connections, instantiation along with inheritance.

A unified framework for coupling is also reported by Bartolomei *et al.* projected for AO system [25]. They extended their work from the existing framework of Briand's frameworks (Briand *et al.* [26] ; Arisholm *et al.* [27]). In their work, two AOP languages are considered namely AspectJ and CaesarJ and showed that the how these two languages can be instantiated on Java.

In [28], Kumar *et al.* projected a new framework of coupling metrics by extending the work of Bartolomei *et al.*[25], Briand *et al.*[26] and Arisholm *et al.* [27] frameworks. They recognized the various types of connection between couplings such as attribute type, parameter type, attribute reference, operation invocation, inheritance and high level association.

Zhao *et al.* projected a cohesion measure which is based on dependency graphs [29]. They defined number of dependencies among aspect and classes and cohesion as the degree of relatedness among modules and attributes.

Gelinas *et al.* define a metrics to measure the cohesion in AO system [30]. This metrics is known as "ACoh Metrics". Two aspect cohesion criteria are defined in ACoh metrics namely "Modules-Modules Connection Criterion" and "Modules-Data Connection Criterion".

On the basis of Table 1, we can conclude that most of the researchers have worked upon mainly cohesion and coupling metrics but Sant'Anna *et al.* introduced a new metrics that majorly affects reusability i.e. "Separation of Concern (SoC)". SoC is the main metrics that differentiate between aspect oriented software and object oriented software. The size and complexity metrics is least used by researchers but this metrics affects maximum the reusability for aspect oriented software. So we have considered four main metrics namely Separation of Concern (SoC), Cohesion, Coupling, Size and Complexity for the proposed work to develop reusability model for AO software using Fuzzy Logic approach. All these four metrics that affect the reusability of AOP are discussed in next section in detail.

Table 1. Metrics used by Various Researcher and Software Practitioners to determine Software Reusability [5]

Researchers	Year	Coupling	Cohesion	Size & Complexity	SoC	Major Findings
Chiamder & Kemerer [32]	1994	X	X	X		Projected Lack of Cohesion metric for AO System.
Bieman & Kang [33]	1995		X			Proposed the frame work for Cohesion Measures in AOP.
Hender- Sellers [34]	1996		X			Extends the Chiamder & Kemerer framework criticizes the LCOM measure in AOP.
Briand *et al.* [16]	1999	X				Contributed in OO system for Coupling Measurement.
Sant'Anna *et al.* [18]	2003	X	X	X	X	Projected LCO and LCC metrics for AO system.
Dospisil *et al.* [35,36]	2003			X		Projected a metric based on the theory of entropy.
Zhao *et al.* [29]	2004	X	X			Projected number of dependencies among aspect and classes.
Ceccato *et al.* [23]	2004	X				They present a framework for aspects, classes, modules and term operation as class functions and aspect advices/introductions
Arishom *et al.* [27]	2004	X				Dynamic Coupling dimension for OO Software.
Gelinas *et al.* [30]	2006		X			Projected metric ACoh to measure aspect cohesion
Bortsh & Harison [24]	2006	X				Enhancement of framework for coupling for OO systems proposed by Briand *et al*.
Bartomei *et al.* [25]	2006	X				Measured two AOP languages, AspectJ and CaesarJ for coupling measure.
Sicilia et al. [37]	2006			X		Design issues of (AOD) extensions on OJB libraries using fuzzy logic.
Patakiet *et al.* [38]	2006			X		Projected metrics that can compute complexity of MO, OO and AO parts of programs implemented in AspectJ.
Zhang & Jacobson [39]	2006	X		X		Projected size and complexity metrics using cyclomatic number, size etc.
Xia *et al.* [40]	2008			X		A new way of conveying complexity weight values to FP metric.
Aljasser *et al.* [21]	2009				X	New language is proposed called ParaAJ which is an extension of AspectJ
Mickelson [41]	2009			X		attentive on size metrics like NOC, module and LOC.
Coady & Kiczales [42]	2003				X	Studied runtime overheads as well as the position of unknown concerns in OS code

III. Metrics Affecting the Software Reusability

In this section various metrics which affects the software reusability have been taken into consideration. There are mainly four metrics (a) SoC (b) Coupling (c) Cohesion (d) Coupling, that majorly affects the software reusability for AO software and all these have been taken into account by Singh *et al.* [50] also. All the considered metrics are discussed in more detail as follows:

A. Separation of Concern (SoC)

SoC stands for Separation of Concern [18]. It makes available the capability to recognize, summarize and control relevant particular concern [48]. The metrics to evaluate SoC are as follows in AOP:

i. Concern Diffusion over Components (CDC)

The CDO metric calculates no. of major components, formal parameter, return types, throws declarations and local variables, or call their method.

ii. Concern Diffusion over Operations (CDO)

The CDO metric calculates the amount of chief operations to add the accomplishment of concerns. It also calculates no. of method as well as constructor.

iii. Concern Diffusion over LOC (CDLOC)

The CDLOC metrics counts no. of conversion points designed for every concern all the way to the LOC [43].

B. Coupling Metrics

The coupling is process of defining the potency of interconnection among the modules of system software [18]. High coupling means high interconnection between program modules that depends on each other [44]. There are following coupling metric that mainly affects software reusability for aspect oriented system.

i. Coupling between Components (CBC)

The CDO metric calculates the no. of classes that apply in attribute declaration. This metrics also calculate formal parameter, throws declaration and local variable.

ii. Depth of Inheritance Tree (DIT)

This metric defines highest extent from node to the origin (root) of the tree. This metrics also defines hierarchy for aspect or class.

C. Cohesion Metrics

This metric defines the closeness of the relationship between the internal modules of program [18 44]. There are number of cohesion metric that mainly affects software reusability of aspect oriented system as given below

i. Lack of Cohesion in Operations (LCOO)

This metrics defines lack of cohesion of module. Sant'Anna *et al.* [18] projected this LCOO metric. It is an extension of Lack of Cohesion in Methods (LCOM) metric which is framed via Chidamber and Kemerer [32].

D. Size Metrics

This metrics calculates the dimension of any software system [18].The following size metrics are given below [47]:

i. Vocabulary Size (VS)

VS count no. of classes and aspects into a system. In VS metric module instances are not counted.

ii. Lines of Code (LOC)

LOC is conventional measure that calculates the size of software. It calculates LOC which excludes blank lines as well as comments.

iii. Number of Attributes (NOA)

NOA calculates no. of attributes for every class and aspect. NOA exclude the inherited attribute.

iv. Weighted Operations per Component (WOC)

This metric determine the complication (complexity) of a component in provisions for operation.

IV. Proposed Fuzzy Model for Aspect Oriented Software Reusability

In this section of paper, the fuzzy logic approach has been discussed for the assessment of software reusability for aspect oriented software.

A. Fuzzy Inference System

Fuzzy inference system is proficient with map inputs during input membership functions and associated parameters as shown in Fig.1. The parameters associated with the membership functions and corresponding associated output parameters are used to understand the final output of fuzzy system. The FIS structure was generated from the Matlab Fuzzy logic Toolbox [45].

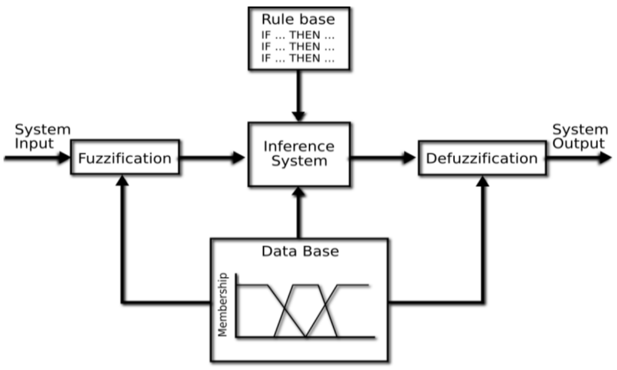


Fig. 1. Fuzzy Inference System

B. Dependent and Independent Variables

The dependent variable is output variable i.e. is reusability, which comes after fuzzification method. The fuzzy set for every output variable will set for defuzzification [46]. The lists of independent variables are four main metrics that majorly affects the reusability. These four metrics are namely Separation of Concerns (SoC), Coupling, Cohesion, and Size metrics.

Measurement values of all metrics always normalized between number 0 and 1. The result for all metrics will be use as an input to the fuzzy inference system (FIS) for measuring the reusability of aspect oriented software. Mamdani fuzzy system is commonly used and same has been used to evaluate software reusability as shown in Fig.2.

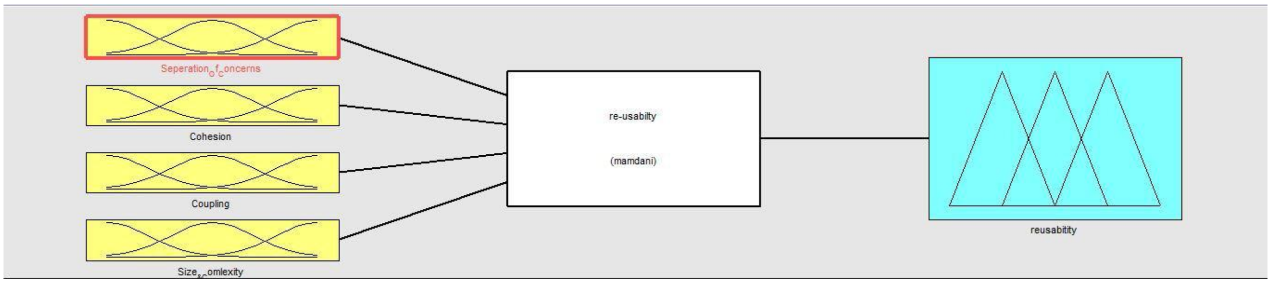


Fig. 2. FIS Structure: Reusability Model

Table 2 summarizes the system information as follow:

Table 2. System Information in MATLAB

Name	"Reusability"
Type	"Mamdani"
No. of Inputs	4
No. of Outputs	1

C. Membership Functions

Membership function enables users to exhibit as well as amend every membership functions that are linked by every input and output variables for the complete FIS. Fig.3 and Fig.4 shows the membership function for input and output variable.

In this work, the scale of membership functions for input as Low (L), Medium (M) and High (H).

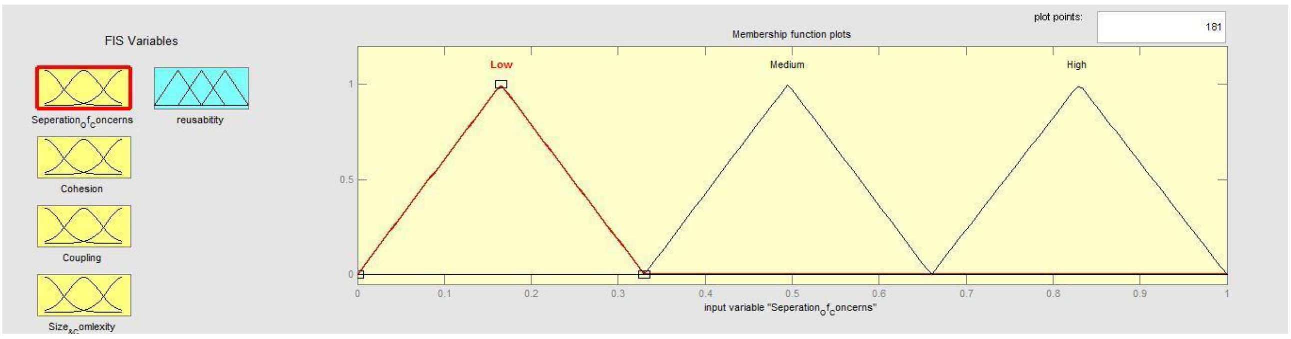


Fig. 3. Membership Function for Input Variable

The output variable is scaled into Very Low (VL), Low (L), Medium (M), High (H) and Very High (VH).

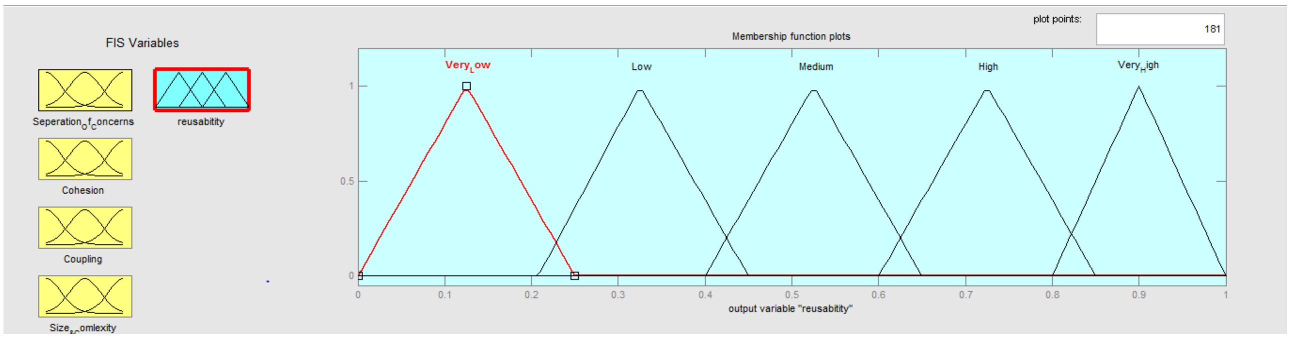


Fig. 4. Membership Function for Output Variable

D. Knowledge Base for Model

Fig. 5 shows the knowledge base model for fuzzy inference system. This knowledge model is known as rule editor. It constructs the rule using graphical rule editor Interface. In this paper, it has four measurement parameters and used three scales. So, total no. of rules is 3^4 which are 81 rules. Hence from this combination it has eighty one if-then rules that constructed and inserted in Rule Editor. Some sample rules have been shown in Table 3.

The triangular (trimf) membership functions are scaled into following:

For input variables

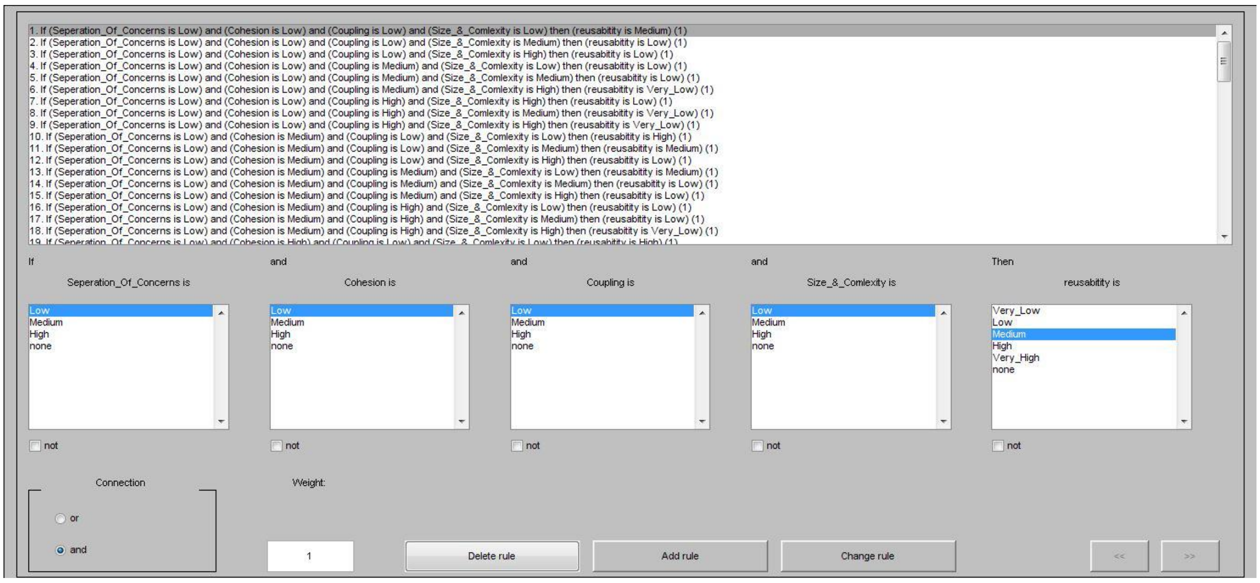
- Low (0-0.33).
- Medium (0.3-0.66).
- High (0.6-1.0).

The output variable is scaled into following:

- Very Low (0-0.2).
- Low (0.18-0.42).
- Medium (038-0.62).
- High (0.58-0.82).
- Very high (0.78-1.0).

Table 3. Some sample Rules

Rules #	Inputs				Output (Reusability)
	SoC	Cohesion	Coupling	Size & Complexity	
1	Low	Low	Low	Medium	Low
2	Low	Medium	High	High	Very Low
3	Low	Medium	Medium	Low	Medium
4	High	High	High	Medium	High
5	High	High	Low	Medium	Very High
6	High	High	Low	Low	Very High



```
1. If (Seperation_Of_Concerns is Low) and (Cohesion is Low) and (Coupling is Low) and (Size_&_Comlexity is Low) then (reusabitity is Medium) (1)
2. If (Seperation_Of_Concerns is Low) and (Cohesion is Low) and (Coupling is Low) and (Size_&_Comlexity is Medium) then (reusabitity is Low) (1)
3. If (Seperation_Of_Concerns is Low) and (Cohesion is Low) and (Coupling is Low) and (Size_&_Comlexity is High) then (reusabitity is Low) (1)
4. If (Seperation_Of_Concerns is Low) and (Cohesion is Low) and (Coupling is Medium) and (Size_&_Comlexity is Low) then (reusabitity is Low) (1)
5. If (Seperation_Of_Concerns is Low) and (Cohesion is Low) and (Coupling is Medium) and (Size_&_Comlexity is Medium) then (reusabitity is Low) (1)
6. If (Seperation_Of_Concerns is Low) and (Cohesion is Low) and (Coupling is Medium) and (Size_&_Comlexity is High) then (reusabitity is Very_Low) (1)
7. If (Seperation_Of_Concerns is Low) and (Cohesion is Low) and (Coupling is High) and (Size_&_Comlexity is High) then (reusabitity is Low) (1)
8. If (Seperation_Of_Concerns is Low) and (Cohesion is Low) and (Coupling is High) and (Size_&_Comlexity is Medium) then (reusabitity is Very_Low) (1)
9. If (Seperation_Of_Concerns is Low) and (Cohesion is Low) and (Coupling is High) and (Size_&_Comlexity is High) then (reusabitity is Very_Low) (1)
10. If (Seperation_Of_Concerns is Low) and (Cohesion is Medium) and (Coupling is Low) and (Size_&_Comlexity is Low) then (reusabitity is High) (1)
11. If (Seperation_Of_Concerns is Low) and (Cohesion is Medium) and (Coupling is Low) and (Size_&_Comlexity is Medium) then (reusabitity is Medium) (1)
12. If (Seperation_Of_Concerns is Low) and (Cohesion is Medium) and (Coupling is Low) and (Size_&_Comlexity is High) then (reusabitity is Low) (1)
13. If (Seperation_Of_Concerns is Low) and (Cohesion is Medium) and (Coupling is Medium) and (Size_&_Comlexity is Low) then (reusabitity is Medium) (1)
14. If (Seperation_Of_Concerns is Low) and (Cohesion is Medium) and (Coupling is Medium) and (Size_&_Comlexity is Medium) then (reusabitity is Low) (1)
15. If (Seperation_Of_Concerns is Low) and (Cohesion is Medium) and (Coupling is Medium) and (Size_&_Comlexity is High) then (reusabitity is Low) (1)
16. If (Seperation_Of_Concerns is Low) and (Cohesion is Medium) and (Coupling is High) and (Size_&_Comlexity is Low) then (reusabitity is Low) (1)
17. If (Seperation_Of_Concerns is Low) and (Cohesion is Medium) and (Coupling is High) and (Size_&_Comlexity is Medium) then (reusabitity is Low) (1)
18. If (Seperation_Of_Concerns is Low) and (Cohesion is Medium) and (Coupling is High) and (Size_&_Comlexity is High) then (reusabitity is Very_Low) (1)
19. If (Seperation_Of_Concerns is Low) and (Cohesion is High) and (Coupling is Low) and (Size_&_Comlexity is Low) then (reusabitity is High) (1)
```

Fig. 5. Rule Editor

E. Rule Viewer

Fig. 6 shows the rule viewer of proposed fuzzy model for AO software. The rule viewer is consisting of five columns. The first to four columns represents the input variable and the "if" part of each rule. Every rule is a row of plots, and each column is a variable that is used. The fifth column represents the membership function for output variable and the "then" part of each rule. The fifth column also represents the cumulative weighted conclusion for the proposed FIS. The bold vertical line on the plot is defuzzified output.

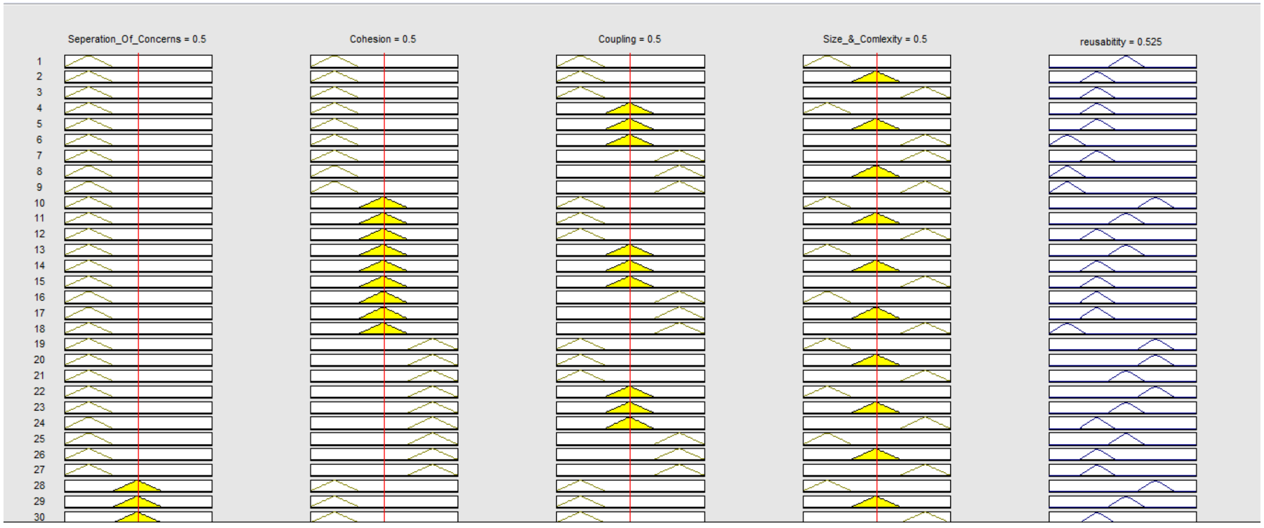


Fig. 6. Rule viewer

F. Surface viewer

Fig. 7, 8 and 9 show the surface viewer for proposed fuzzy model by considering two metrics at a instance. In the surface viewer users are able to produce a 3D output surface for two inputs. Fig. 6 shows a three-dimensional output surface for the set of two metrics (Separation of Concern and Cohesion) for reusability evaluation.

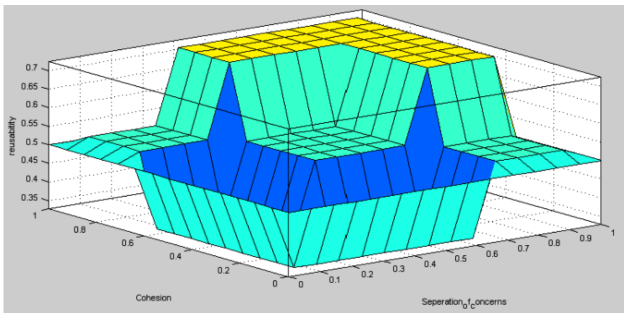


Fig. 7. Surface Viewer for Separation of Concern and Cohesion.

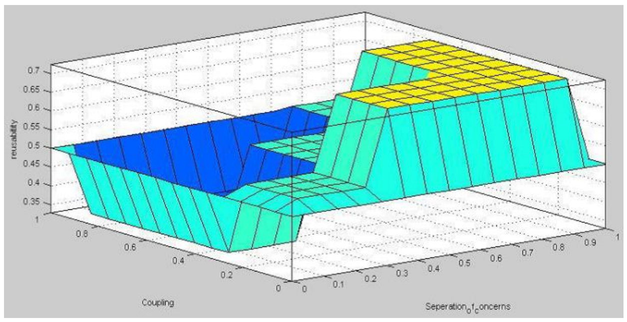


Fig. 8. Surface Viewer for Coupling and Separation of Concern

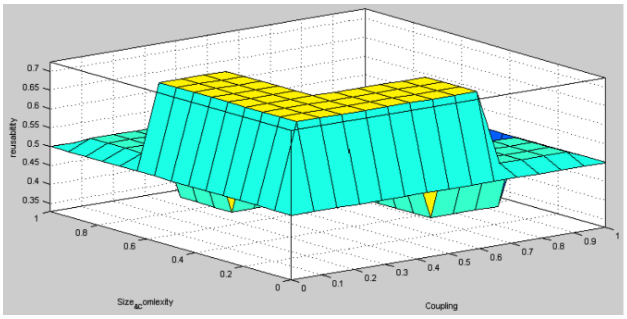


Fig. 9. Surface Viewer for coupling and Size and complexity.

G. Experimental Results

Suppose the subsequent crisp value of inputs is fed to the proposed fuzzy model: Separation_of_concern = 0.17, Cohesion= 0.44, Coupling= 0.60 and Size & Complexity = 0.22. These inputs value are provided to the fuzzification process and get the output as 0.52 (Medium). The fuzzification on given values evaluate the reusability of AO software using proposed fuzzy system by using software metrics and shown in Fig. 10.

Experimental results based on firing some rules on the proposed fuzzy based AO reusability system are reported in Table 4.

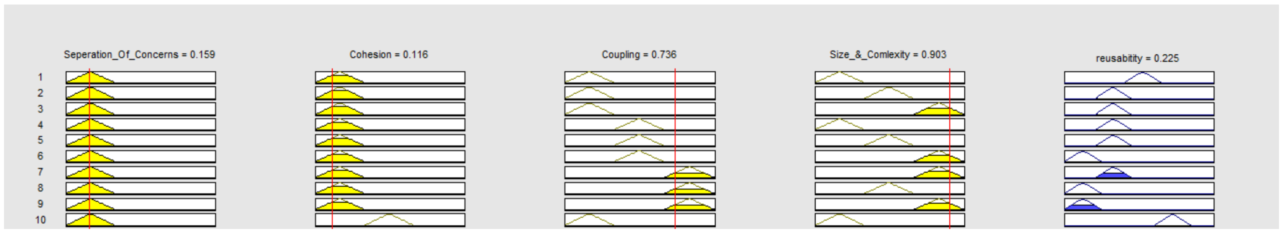


Fig. 10. Software Reusability for a given Input Set

Table 4. Experimental results for some rules

Rules #	Inputs				Output (Reusability)	Results
	SoC	Cohesion	Coupling	Size & Complexity		
1	L(0.12)	L(0.13)	L(0.08)	M(0.48)	L(0.32)	Worst Applicable
2	L(0.14)	M(0.48)	H(0.85)	H(0.83)	VL(0.12)	Not Applicable
3	L(0.17)	M(0.44)	M(0.60)	L(0.22)	M(0.52)	Somewhat Applicable
4	H(0.78)	H(0.69)	L(0.30)	M(0.35)	VH(0.9)	Highly Applicable
5	H(0.87)	H(0.88)	M(0.58)	M(0.57)	H(0.7)	Applicable

H. Defuzzification

It is a process of mapping from a space of fuzzy control actions defined over an output universe of discourse into a space of crisp (non-fuzzy) control actions. There are various names for this method such as center-of-mass, center-of-area, or center-of-gravity method.

Centre of Gravity (COG) has large applicability for defuzzification. The defuzzified output for X^* is by (1) and defuzzified output is shown in Fig.11 and validated for the rule number 3 from Table 4.

$$X^* = \sum A\,\bar{x} \,/ \sum A \quad (1)$$

$$X^* = 0.5029$$

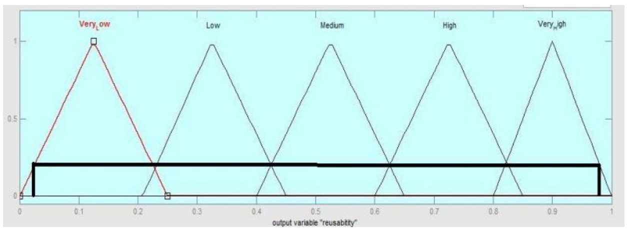


Fig. 11. Defuzzified Output

V. Conclusion and Future Scope

In this paper fuzzy logic approach is used to access the reusability of AO software. It is proved that application of fuzzy logic approach has shown their applicability other than traditional statistical techniques. The input variables in fuzzy model are software design metrics that derived from literature review. The four main metrics have been discovered from literature reviewed namely separation of concern (SoC), cohesion, coupling, size and complexity. On the basis of input variables, 81 rules are designed. These rules are based on expert's knowledge and judgment. So, it is concluded that proposed model based on fuzzy will helps the software developer's to select the best quality of software in terms of reusability among AO software.

In future, the dynamic metrics can also be considered for accessing the reusability of AO software systems. A maintainability model to show the relationship between the maintainability and metrics (static and dynamic metrics) for AO software has been reported by us in our previous work [31]. Due to unavailability of dynamic metrics measurement tools, we have taken into account only static metrics. Apart from fuzzy logic technique, other soft computing techniques such as Artificial Neural Network (ANN), Adaptive Neuro Fuzzy Inference System (ANFIS) and Support Vector Machine (SVM) can also be taken into consideration for the assessment of software reusability.

Acknowledgment

We would like to thank the faculty of Amity University for helping us in refining the objective and Amity University for providing us research environment and facilities. We also like to extend our thanks to Dr. Arun Sharma, Professor, CSE, K.I.E.T. Ghaziabad for his valuable suggestions.

References

[1] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, "Object-Oriented Modeling and Design", Prentice-Hall, New York, 1991.

[2] I. Jacobson, M. Christerson, P. Johnson & G. Overgaard, "Object Oriented Software Engineering: A Use Case Approach", Addison Wesley, 1992

[3] C. Szyperzaki, "Component Software: Beyond Object-Oriented Programming", Addison-Wesley", 2001.

[4] A. Kumar, R. Kumar, P.S. Grover, "A Comparative Study of Aspect-Oriented Methodology with Module-Oriented and Object-Oriented Methodologies", ICFAI Journal of Information Technology, Volume 2, No 4, pp. 7-15, December 2006.

[5] P. K. Singh, Parag Mittal, Lakshay Batra and Utkarsh Mittal, "A Perception on Programming Methodologies for Software Development", IJCA Proceedings on 4th International IT Summit Confluence 2013 - The Next Generation Information Technology Summit Confluence 2013(2), Pages 1-6, January 2014.

[6] I. Aracic, V. Gasiunas, M. Mezini, K. Ostermann, "Overview of CaesarJ Transactions on Aspect-Oriented Software Development", LNCS, 3880, pp. 135-173, 2006.

[7] B.R. Pekilis, "Multi-Dimensional Separation of Concerns and IBM Hyper/J", Technical Research Report, January 22, 2002.

[8] T. Elrad, M. Aksits, G. Kiczales, K. Lieberherr, H. Ossher, "Discussing Aspects of AOP", Communications of the ACM, 44(10), pp. 33–38, 2001.

[9] J.D. Gradecki, N. Lesiecki, "Mastering AspectJ: Aspect-Oriented Programming in Java", Wiley, 2003.

[10] ISO9126 Information Technology, "Software Product Evaluation – Quality characteristics and guidelines for their use", International Organization for Standardization, Geneva, 1992.

[11] R. G. Dromey, "A Model for Software Product Quality," IEEE Transactions on Software Engineering, Volume 21, Number 2, pp. 146 - 162, February 1995.

[12] A. Kumar, P. S. Grover, R. Kumar, "A Quantitative Evaluation of Aspect-Oriented Software Quality Model", ACM SIGSOFT Software Engineering Notes Volume 34, Number 5, pp. 1 - 9, September 2009.

[13] I. Castillo, F. Losavio, A. Matteo, J. Boegh, "REquirements, Aspects and Software Quality: the REASQ model", Journal of Object Technology, Volume 9, Number 4, pp. 69 - 91, 2010.

[14] P. Kumar, "Aspect-Oriented Software Quality Model: The AOSQ Model", Advanced Computing: An International Journal, Vol.3, No.2, March 2012.

[15] M.W. Price, S.A. Demurjian, "Analyzing and Measuring Reusability in Object-Oriented Design", In the Proceedings of the 12th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, Atlanta, Georgia, US, pp. 22-33, October 05-09, 1997.

[16] J. Barnard, "A New Reusability Metric for Object-Oriented Software", Software Quality Journal, Vol. 7, Issue 1, pp. 35–50, 1998.

[17] F. Dandashi, "A Method for Assessing the Reusability of Object-Oriented Code Using a Validated Set of Automated Measurements", In Proceedings of the ACM Symposium on Applied Computing, Madrid, Spain, pp. 997-1003, 2002.

[18] C. Sant'Anna, A. Garcia, C. Chavez, C. Lucena, and A. Von Staa, "On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework", 23rd Brazilian Symposium on Software Engineering, Manaus, Brazil, October 2003.

[19] C.A. Cunha, J.L. Sobral, M.P. Monteiro, "Reusable aspect-oriented implementations of concurrency patterns and mechanisms", In Proceedings of the 5th international Conference on Aspect-Oriented Software Development (Bonn, Germany, March 20 - 24, 2006), ACM, New York, NY, pp. 134-145, 2006.

[20] J. Zhang, H. Li, X. Cai, "Research on Reusability of Software Connector Based on AOP", In the IEEE Proceedings of International Conference on Computer Science and Information Technology, pp. 113-117, 2008.

[21] K. Aljasser, P. Schachte, "ParaAJ: toward Reusable and Maintainable Aspect Oriented Programs", In Proceedings of Thirty-Second Australasian Computer Science Conference , Wellington, New Zealand, CRPIT, 91, Mans, B., Ed. ACS, pp. 53-62, 2009.

[22] J. Zhao, "Measuring Coupling in Aspect-Oriented Systems", In: 10th International Software Metrics Symposium (Metrics 04), 2004.

[23] M. Ceccato, P. Tonella, "Measuring the Effects of Software Aspectization", In: Proceedings of the 1st

Workshop on Aspect Reverse Engineering, ACM Press, 2004.

[24] M. Bartsch, R. Harrison, "An Evaluation of Coupling Measures for AspectJ", Presented at the LATE Workshop at the Aspect-Oriented Software Development Conference (AOSD). Bonn, Germany. 2006.

[25] T. Bartolomei, A. Garcia, C. Sant'Anna, E. Figueiredo, "Towards a Unified Coupling Framework for measuring Aspect-Oriented Programs", In 3rd International Workshop on Software Quality Assurance Portland, Oregon, USA, , ACM Press, November 6, 2006

[26] L.C. Briand, J.W. Daly, J. Wust, "A Unified Framework for Coupling Measurement in Object-Oriented Systems", IEEE Transactions on Software Engineering, 25(1), pp. 91-120, 1999.

[27] E. Arisholm, L.C. Briand, A. Føyen, "Dynamic Coupling Measurement for Object-Oriented Software", IEEE Transactions on Software Engineering, 30(8), pp. 491-506, 2004.

[28] A. Kumar, R. Kumar, P.S. Grover, "Generalized Coupling Measure for Aspect-Oriented Systems", ACM SIGSOFT Software Engineering Notes, 34(3), pp. 1-6, 2009.

[29] J. Zhao, B. Xu, "Measuring Aspect Cohesion", In: Proceedings of International Conference on Fundamental Approaches to Software Engineering, March 29-31, LNCS 2984, Springer-Verlag, Barcelona, Spain, pp.54-68, 2004.

[30] J.F. Gelinas, M. Badri, L. Badri, "A Cohesion Measure for Aspects" Journal of Object Technology, 5(7), pp. 97–114, 2006.

[31] P.K.Singh, O.P.Sangwan, "Aspect Oriented Software Metrics Based Maintainability Assessment: Framework and Model", published in proceeding's of Confluence-2013, The Next Generation Information Technology Submit, 26th -27th September, Amity University, Noida, India, 2013.

[32] S.R. Chidamber, C.F. Kemerer, "A Metrics Suite for Object- Oriented Design", IEEE Transactions on Software Engineering, 20(6), pp. 476–493, 1994.

[33] J.M. Bieman, B.K. Kang, "Cohesion and Reuse in an Object-Oriented System", In Proc. ACM Symp. Software Reusability (SSR'94), pp. 259-262. 1995.

[34] B. Henderson-Sellers, "Software Metrics", Prentice Hall, Hemel Hempstead, UK, 1996.

[35] J. Dospisil, "Measuring Code Complexity in Projects Designed with AspectJ" Informing Science InSITE- "Where Parallels Intersects", pp. 185-197, 2003.

[36] Jana Dospisil, "Measuring Code Complexity in Projects Designed with Aspect/J™", Informing Science + IT Education (InSITE) Conference, Finland, June 2003.

[37] M.A. Sicilia, E. Garc ǹa-Barriocana, "Extending Object Database Interfaces with Fuzziness Through Aspect-Oriented Design", ACM SIGMOD Record, 35(2), pp. 4–9, 2006.

[38] N. Pataki, A. Sipos, Z. Porkolab, "Measuring the Complexity of Aspect-Oriented Programs with Multiparadigm Metric", ECOOP Doctoral Symposium and PhD Students Workshop, 2006.

[39] C. Zhang, H. A. Jacobsen, "Quantifying Aspects in Middleware Platforms", Department of Electrical and Computer Engineering and Department of Computer Science, University of Toronto, 2000.

[40] W. Xia, L.F. Capretz, D. Ho, F. Ahmed, "A new Calibration for Function Point complexity weights: Information and Software Technology", 50(7-8), pp. 670-683, 2008.

[41] Magnus Mickelsson, "Aspect-Oriented Programming compared to Object-Oriented Programming when implementing a distributed, web based application", Department of Information Technology, Uppsala University, 2002.

[42] Y. Coady, G. Kiczales, "Back to the Future: A Retroactive Study of Aspect Evolution in Operating System Code", University of British Columbia, 2003.

[43] A.F.Garcia, C.N. Santpana, G.C. Christina, "Agents and Objects: An Empirical Study on Software Engineering". Technical Report 06-03, Computer Science Department, PUC-Rio, February 2003.

[44] I. Sommerville, "Software Engineering", 6.ed. Harlow, England, Addison-Wesley, 2001.

[45] MATLAB Toolbox, http://www.mathworks.com "MatLab Toolbox for ANN, FIS, ANFIS".

[46] Fuzzy Logic Toolbox, User's Guide version 2, The Math Works Inc., SA, July 2002. SIGSOFT Software Engineering Notes, pp. 6, Volume 34, July 2009.

[47] N. Fenton, S.L. Pfleeger, "Software Metrics: A Rigorous and Practical Approach", 2.ed. London: PWS, 1997.

[48] P. Tarr, *et al.* "N Degrees of Separation: Multi-Dimensional Separation of Concerns", Proceedings of the 21st International Conference on Software Engineering, May 1999.

[49] S. Rajasekaran, G.A. Vijayalakshmi Pai, "Neural Networks, Fuzzy Logic, and Genetic Algorithms Synthesis and Application", PHI learning, Eastern Economy Edition.

[50] P. K. Singh, O. P. Sangwan, A. Pratap, A. P. Singh, "A Quantitative Evaluation of Reusability for Aspect Oriented Software using Multi-criteria Decision Making Approach" published in World Applied Sciences Journal, Volume 30, Issue 12, Pages 1966-76, 2014.

Author's Profile

Pradeep Kumar Singh, Mr. Pradeep Kumar Singh is M.Tech (CSE) from GGSIPU Delhi and pursuing his PhD. from Gautam Buddha University, Greater Noida, India. Currently, he is working as Assistant Professor in Department of Computer Science and Engineering of Amity University Uttar Pradesh, Noida, India. He is member of ACM, CSI and many professional bodies. He has published 10 papers in International Conferences and Journals of repute. His major area of Interest includes Software Engineering, Object Oriented Software Engineering, Aspect Oriented Software Engineering, Software Testing and Soft Computing Techniques.

Dr. Om Prakash Sangwan is M.Tech (CSE) and PhD. in Computer Science & Engineering. Currently, he is working as Assistant Professor in Department of Computer Science and Engineering of Gautam Buddha University, Greater Noida. Uttar Pradesh, India. He is Senior Member of ACM, CSI, IEEE and many professional bodies. He has filled two Patents and published 35 papers in International Conferences and Journals of repute. His major area of Interest includes Software Engineering, Object Oriented Software Engineering, Aspect Oriented Software Engineering, Soft Computing Techniques and Artificial Intelligence.

Amar Pal Singh is a M.Tech.(CSE) student from Amity University Uttar Pradesh, Noida, India. His interests include Software Engineering, Soft Computing Techniques. He has published four papers in International Journal and Conferences.

Amrendra Singh is a student of M.Tech. (CSE) from Amity University Uttar Pradesh, Noida, India. His interests include Software Engineering, Soft Computing Techniques. He has published three papers in International Conference and Journals.