RESEARCH ARTICLE OPEN ACCESS

# A Evaluation of Software Re-Usability Using Software Metrics through Fuzzy Logic

Manoj Kumar Singh, Govind Kamboj, Avnish Kumar Sharma
M. Tech Scholar, Department of Computer Science and Engineering, Graphic Era University Dehradoon, India
Assistant Professor, Department of Computer Science and Engineering, Graphic Era University Dehradoon, India
Assistant Professor, Department of Master of Computer Application, Marathwada Institute of Technology, Bulandshahr, India

***Abstract***
flexibility and quality through reusability, replace-ability, efficient reusability and scalability are some additional benefits of Component based software development .Component Based Software Engineering (CBSE) is a paradigm that aims at constructing and designing systems using a pre-defined set of software components explicitly created for reuse. According to Clements, CBSE embodies the “the ‘buy, don’t build’ philosophy”. The idea of reusing software components has been present in software engineering for several decades. Although the software industry developed massively in recent decades, component reuse is still facing numerous challenges and lacking adoption by practitioners. One of the impediments preventing efficient and effective reuse is the difficulty to determine which artifacts are best suited to solve a particular problem in a given context and how easy it will be to reuse them there. So far, no clear framework is describing the reusability of software and structuring appropriate metrics that can be found in literature. Nevertheless, a good understanding of reusability as well as adequate and easy to use metrics for quantification of reusability are necessary to simplify and accelerate the adoption of component reuse in software development.
***Keywords—*** Software Reusability; Software Reusability Metrics; Component-Based Software Development.

## I. INTRODUCTION

A component is a reusable, self -contained piece of software with well specified interface that is independent of any application. The very important point which has to be kept in mind, while developing a component is the reusability aspect, regardless of whether or not an organization can identify what the future requirements of the component will be. Components can be placed on any network node, depending on application needs and regardless on the type of particular network structure. An extra effort must be paid for the additional functionality of the component beyond the current application’s need, to make the component more useful.

Object-oriented technology alone is not enough to cope with the rapidly changing requirements of present day applications. One of the reasons is that, although the Object-Oriented (O-O) methods encourage one to develop rich models that reflect the object of problem domain, this does not necessarily yield software architectures that can be easily adapted to changing requirements. Moreover, today’s applications are large, complex and are not integrated. Although they come packaged with a wide range of features but most features can neither be removed, upgraded independently or replaced nor can be used in other applications. In particular, O-O methods do not typically lead to designs that make a clear separation between computational and compositional aspects [15].

An application must have some additional characteristics like robustness, usability, flexibility, simple installation, reusability, portability, interoperable, proper documentation etc. to fight with the advancement in the technology and rapidly changing requirements. To improve the business performance, it is necessary to use the latest technologies available. Today Component Based Software Development (CBSD) is getting accepted in the company or an industry as a new effective development paradigm. It emphasizes the design and construction of software system using reusable components. CBSD is capable of reducing development cost and increasing the reliability of entire software system using components. The major advantages of CBSD are in-time and high quality solutions. Higher productivity, flexibility and quality through reusability, replace-ability, efficient reusability and scalability are some additional benefits of CBSD [5]. Component Based Software Engineering (CBSE) is a paradigm that aims at constructing and designing systems using a pre-defined set of software components explicitly created for reuse. According to Clements [19], CBSE embodies the “the ‘buy, don’t build’ philosophy”. He also says about CBSE that “in the same way that early subroutines liberated the programmer from thinking about details, CBSE shifts the emphasis from programming to composing software systems”. The focus of CBSE is reusing whole software component, not objects. An important motivation for many organizations for adopting CBSE as their software development paradigm is to reduce development cost. One of the main contributions that CBSE has to this objective is the reuse of software components in multiple systems. In this way a software component is developed only once and can save out development effort multiple times.

### *A. Component Based Software Engineering*

“CBSE is a process that aims to design and construct software systems using reusable software components”. If one is familiar with Object Oriented programming (OOP), it can be useful to think of CBSE in a similar way. In OOP, code is reused in the form of objects. These objects are often contained in vast libraries of reusable code. Frameworks take the process even further, providing more robust and disciplined systems of reuse. By obtaining and reusing parts of systems which have already been ‘tried and tested’, we can exploit the principal advantages of OOP techniques over procedural programming techniques. They enable programmers to create modules1 that do not need to be changed when a new type of object is added. A programmer can simply create a new object that inherits many of its features from existing objects. This makes object-oriented programs easier to modify. In the same way, in CBSE, by reusing an existing component you cut out a lot of the hard work with establishing the usefulness and in testing that component. Although some testing will still be required [9,8].

CBSE should, in theory, allow software systems to be more easily assembled, and less costly to build. Although this cannot be guaranteed, the limited experience of adopting this strategy has shown it to be true. The software systems built using CBSE are not only simpler and cheaper, but usually turn out to be more robust, adaptable and updateable. CBSE allows use of predictable architectural patterns and standard software architecture leading to a higher quality end result.

CBSE is an approach to software development that relies on reuse. It emerged from the failure of object-oriented development to support effective reuse. Single object classes are too detailed and specific. Components are more abstract than object classes and can be considered to be stand-alone service providers [13].

### *B. Software Metrics*

As the number of components available on the market increases, it is becoming more important to devise software metrics to quantify the various characteristics of components and their usage. Software metrics are intended to measure the software quality and performance characteristics quantitatively, encountered during the planning and execution of software development. These can serve as measures of software products for the purpose of comparison, cost estimation, fault prediction and forecasting.

Metrics can also be used in guiding decisions throughout the life cycle, determining whether software quality improvement initiatives are financially worthwhile [1].

A lot of research has been conducted on software metrics and their applications. Most of the metrics proposed in literature are based on the source code of the application.

However, these metrics cannot be applied on components and component-based systems (CBS) as the source code of the components is not available to application developers. Therefore, a different set of metrics is required to measure various aspects for CBS and their quality issues.

### *C. Reusability*

Reusability is the degree to which a component can be reused and reduces the software development cost by enabling less coding and more integration [2]. The reusability of assets is different in different contexts. However, there are some characteristics that generally contribute to the reusability of assets. Although many of these characteristics apply to assets in general, we focus in this section, we focus on components as assets. At a high level, we distinguish two aspects of reusability i.e. usability and usefulness [10].

Reusability = Usability + Usefulness
Usability is the degree to which an asset is ‘easy’ to use in the sense of the amount of effort that is needed to use an asset. Usability as such is independent of functionality of the component. Usefulness is the ‘frequency’ of suitability for use i.e. usefulness depends on the functionality, the generality and quality of a component [10].

### *D. Objectives of This Proposed Technique*

In this paper, various Software metrics and component based system techniques have been studied and analyzed.
The main objectives of this proposed work are as follows:

1) Software reuse is to minimize repetition of work, development time, cost and efforts and increase reliability of the systems. It also improves the reusability and portability of the system.
2) One of the main contributions that CBSE has to this objective is the reuse of software components in multiple systems. In this way a software component is developed only once and can save out development
3) Effort multiple times.
4) Increased reliability
5) Reduced process risk
6) Effective use of specialists
7) Standards compliance
In the present study, we have taken into account, the understanding of the components as well, along with other important factors required for evaluating the reusability of software components like:

- Customizability
- Interface complexity
- Portability
- Documentation
- Observability

## II. RELATED WORK

In case of component-based development, software reuse refers to the utilization of a software component with in a product, where the original motivation for constructing this component was not known. Here, reuse is seen as black-box reuse, where the application developer sees the interface, not the implementation of the component. The interface contains public methods, user documentation, requirements and restrictions of the component. If there is a change in the code of a black-box component, compiling and linking the component would propagate the change to the applications that reuse the component. As the users of the component trust its interface, changes should not affect the logical behaviour of the component [3].

As the number of components available in the market increases, it is becoming more important to devise software metrics to quantify the various characteristics of components. Among several quality characteristics, the reusability is particularly important when reusing components. Reusability can measure the degree of features that are reused in building applications [3]. In the present study, we have taken into account, the understanding of the components as well, along with other important factors required for estimating the reusability of software components.

## III. PROPOSED METHOD

We proposed a Fuzzy Logic based approach for estimating reusability for component based systems. It is often impossible to estimate software quality attributes directly. For example, attributes (say, reusability, etc.) are affected by many different factors, and there is no straightforward method to measure them.

Emphasis in the present work is to estimate reusability, if the changes require customization/replacement of the component or integration code has to be modified.

***A. Customizability*** is defined as the ability to modify a component as per the application requirement. Better customizability will lead to a component with better reusability in applications and thus help in maintaining the component in the later phases.
It may be measured on the basis of writable properties available in the component. The following formula is used to evaluate this criterion:

$$\textit{Customizability} = \frac{\text{No. of writable properties}}{\text{Total number of Properties}}$$

By using this metric, one can measure that how much an interface method can be customized. Therefore, it may be used to measure the reusability of the component. Customizability of a component may vary from 0 to 1.

***B. Interface complexity*** Components are black box in nature. The source code of these components is not available. Application may interact with these components only through their well - defined interfaces. Interface acts as a primary source for understanding, use and implementation and finally maintenance for the component. Therefore, the complexity of these interfaces plays a lead role while measuring the overall complexity of the component.
Complex interfaces will lead to the high efforts for understanding and customizing the components. Therefore for better reusability, interface complexity should be as low as possible. The following formula is used by us to evaluate this criterion

***Interface Complexity = 1 - (Number of interfaces not required / Total number of interfaces provided)***

More the unrequired interfaces more will be complexity and hence less will be reusability. The whole value is reciprocal because more complexity will lead to less reusability, thus the reciprocal value of complexity near to 1 will give us the reusability value for the component.

***C. Portability*** It is the ability of a component to be transferred from one environment to another with little modification, if required. It is typically concerned with reuse of component on new platforms.

The component should be easily and quickly portable to specified new environments if and when necessary, with minimized porting efforts and schedules. For better reusability, component should be highly portable, means; it should be supported by several platforms. Here, for the proposed work, portability may be defined as:

$$\textit{Portability} = \frac{\textit{No. of platforms the component can support}}{\textit{Total no. of platforms that may be required by CBSS}}$$

By using this metric, one can measure that how many platforms can be supported by the COTS component. Therefore, it may be used to measure the reusability of the component. Portability of a component may vary from 0 to 1.

***D. Observability*** It may defined as the ability to understand component's functional elements as per described in its manual. Functional elements may be referred as the interfaces, operations, or events that a component may support or require from other components to achieve its functionality, i.e., to implement its services. As the source code of the component is not available better observability will lead to a component with better reusability in applications and thus help in using the component in the later phases.

It may be measured on the basis of readable functional elements available in the component. The following formula is used to evaluate this metric by washizaki et. al. [9]:

$$\textit{Observability} = \frac{\textit{No. of readable functional properties}}{\textit{Total number of functional Properties}}$$

By using this metric, one can measure that how many functional elements can be understood. Therefore, it may be used to measure the reusability of the component. Observability of a component may vary from 0 to 1.

***E. Documentation Level*** As the source code of the COTS component is not available to the application developer; documentation is the only source from where he/she can understand the component. Documentation provides the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component.

Here, the term documentation of component refers to component manuals, demos, help system, and marketing information. A good quality document must include functional description, installation details, system administrator's guide, system reference manuals etc. It may also require non - functional details, like performance, security issues, and previous maintenance activities, if any. It will help in understanding the component and reusing and integrating it easily in the CBSS. It will also help in implementing the maintenance activities with less effort. For present work, we categorize documentation quality from low to High. The metric proposed by us that can be used to measure the documentation from this level scale can be described as follows:-

$$\textit{Documentation Level} = \sum_{i=1}^{n} \textit{Level of each documentation } D_i \textit{ provided} / n * 3$$

Where n is the total number of documents required by the CBSS to clear understanding of the COTS component. Documentation level metric value of a component may vary from 0 to 1.

The goal of our work is to develop a tool for estimating reusability of the software component. We consider that reusability is a measure of factors mentioned above. The values of these individual factors can be measured by using the appropriate metrics. Customizability metric is the ratio of writable properties to the total number of properties. Documentation and portability can be classified from very low to very high categories.

## IV. RESULT AND ANALYSIS

### *I. Implementation of Fuzzy system*

*I*mplementing a fuzzy system requires that the different categories of the different inputs be represented by fuzzy sets which, in turn, is represented by membership functions. The domain of membership function is fixed, usually the set of real numbers, and whose range is the span of positive numbers in the closed interval [0, 1]. There are total 11 membership functions available in Mat Lab. We considered Triangular Membership Functions (TMF) for our problem, because of its simplicity and heavy use by researchers for prediction models [18]. It is a three-point function, defined by minimum $\alpha$, maximum $\beta$ and modal value m i.e. TMF ($\alpha$, m, $\beta$), where ($\alpha \leq m \leq \beta$). This process is known as fuzzification. These membership functions are then processed in fuzzy domain by inference engine based on knowledge base (rule base and data base) supplied by domain experts and finally the process of converting back fuzzy numbers into single numerical values is called defuzzification [17].

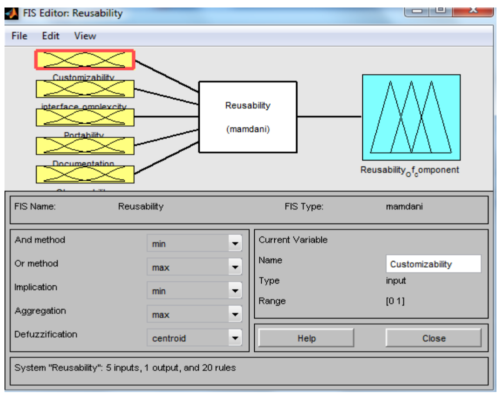
### II. Fuzzy Model

We propose that reusability of component-based system is a measure of five factors mentioned above. These combined factors can be used to measure the reusability, as it cannot be measured directly. The

proposed FL based model considers all five factors as inputs and provides a crisp value of reusability using the Rule Base. All inputs can be classified into fuzzy sets *viz.* Low, Medium and High.
The output reusability is classified as Very High, High, Medium, Low and Very Low. All possible combinations ($3^5$ i.e. 243) of inputs are considered to design the rule base. Each rule corresponds to one of the five outputs based on the expert opinions. Some of the proposed rules are shown as:

1) If Customizability of components is Low, Interaction Complexity among component is High, Observability is Low, Portability is Low, Document level is Low then it is very difficult to maintain the system i.e. Reusability will be Very Low.
2) If Customizability of components is Low, Interaction Complexity among component is High, Observability is Low, Portability is Medium and Document level is Medium then Reusability will be Low.
3) If Customizability of components is Medium, Interaction Complexity among component is High, Observability is Medium, Portability is Medium and Document level is Medium then Reusability will be Medium.
4) If Customizability of components is Medium, Interaction Complexity among component is Medium, Observability is Medium, Portability is Medium and Document Level is Medium then Reusability will be Medium.
5) If Customizability of components is Medium, Interaction Complexity among component is Low, Observability is High, Portability is Medium and Document Level is Medium then Reusability will be high.
6) If Customizability of components is High, Interaction Complexity among component is Low, Observability is High, Portability is Medium and Document Level is High then Reusability will be high.

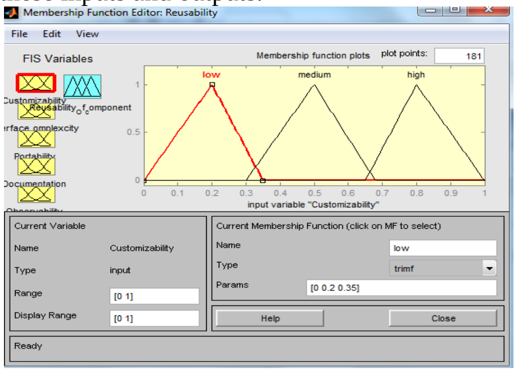
All 243 rules are inserted into the proposed model and a rule base is created. Depending on a particular set of inputs, a rule is fired. Using the rule viewer, output i.e. reusability is observed for a particular set of inputs using the Mat Lab Fuzzy tool box.



FIS Name: Reusability
FIS Type: mamdani
And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid
Current Variable
Name: Customizability
Type: input
Range: [0 1]
Help Close
System "Reusability": 5 inputs, 1 output, and 20 rules

Fig 4.1Fuzzification

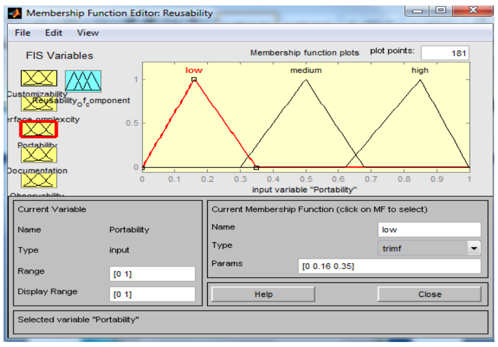
In this figure set the five inputs customizability, Interface complexity, Portability, documentation, observablity and one outputs reusability. Set the rules of these inputs and outputs.



plot points: 181
Current Variable
Name: Customizability
Type: input
Range: [0 1]
Display Range: [0 1]
Current Membership Function (click on MF to select)
Name: low
Type: trimf
Params: [0 0.2 0.35]
Help Close
Ready

Fig 4.2 Member Function of Customizability

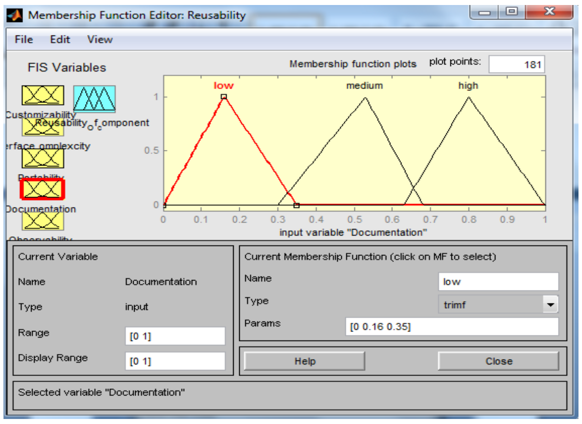
In this FIS passing the input parameters of customizability (low, medium, high) in between 0and 1.



plot points: 181
Current Variable
Name: Portability
Type: input
Range: [0 1]
Display Range: [0 1]
Current Membership Function (click on MF to select)
Name: low
Type: trimf
Params: [0 0.16 0.35]
Help Close
Selected variable "Portability"

Fig 4.3Member Function of Portability

In this FIS passing the input parameters of portability (low, medium, high) in between 0and 1.


Membership Function Editor: Reusability

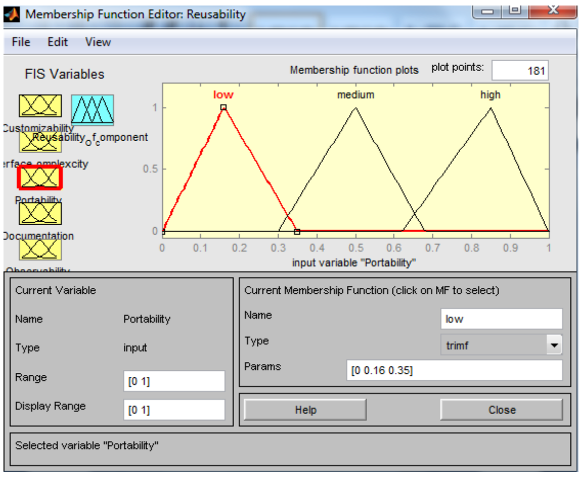
Current Variable
Name Documentation
Type input
Range [0 1]
Display Range [0 1]

Current Membership Function (click on MF to select)
Name low
Type trimf
Params [0 0.16 0.35]

Selected variable "Documentation"

Fig 4.4 Member Function of Documentation

In this FIS passing the input parameters of Documentation (low, medium, high) in between 0and 1.


Membership Function Editor: Reusability

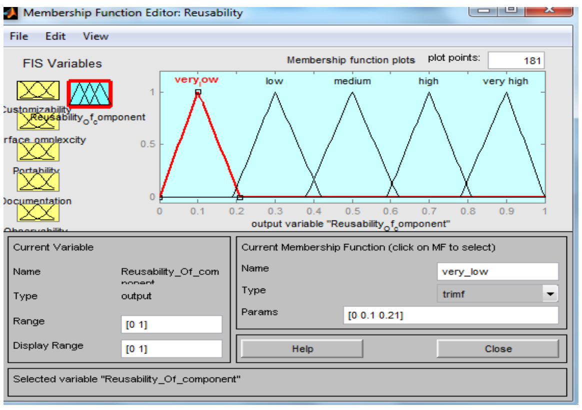
Current Variable
Name Portability
Type input
Range [0 1]
Display Range [0 1]

Current Membership Function (click on MF to select)
Name low
Type trimf
Params [0 0.16 0.35]

Selected variable "Portability"

Fig 4.5 Member Function of interface complexity

In this FIS passing the input parameters of interface complexity (low, medium, high) in between 0and 1.


Membership Function Editor: Reusability

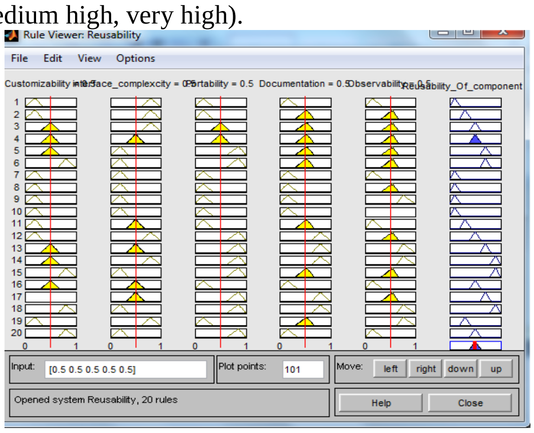
Current Variable
Name Reusability_Of_component
Type output
Range [0 1]
Display Range [0 1]

Current Membership Function (click on MF to select)
Name very_low
Type trimf
Params [0 0.1 0.21]

Selected variable "Reusability_Of_component"

Fig 4.6 Member Function of Reusability of Component

This FIS passing the output parameters of Reusability. Set the value of the reusability (very low, low, medium high, very high).


Rule Viewer: Reusability

Customizability = 0.5 Interface_complexcity = 0.5 Portability = 0.5 Documentation = 0.5 Observability = 0.5 Reusability_Of_component

Input: [0.5 0.5 0.5 0.5 0.5]
Plot points: 101

Opened system Reusability, 20 rules

Fig 4.7 Rules Viewer of Reusability of Component

Input values Customizability, Interface complexity, Portability, Documentation, observability and finding the value of reusability in this FIS.

## V. CONCLUSION AND SCOPE OF THE FUTURE WORK

In this work, we have proposed a Component-based software engineering (CBSE) (also known as component-based development (CBD)) is a branch of software engineering which emphasizes the separation of concerns in respect of the wide-ranging functionality available throughout a given software system. This practice aims to bring about an equally wide-ranging degree of benefits in both the short-term and the long-term for the software itself and for organizations that sponsor such software.

Software reuse is the process of creating software systems from existing software rather than building software systems from scratch. Reusable software components are designed to apply the power and benefit of reusable, interchangeable parts from other industries to the field of software construction. Reusable components add standardized interfaces and object introspection mechanisms to widgets allowing builder tools to query components about their properties and behaviour. Software components need not be visible in a running application; they only need to be visible when the application is constructed.

Reusability metrics can measure the degree of features that are reused in building applications. Even if there are some metrics defined for the reusability of object-oriented software (OOS), they cannot be used for CBSD because these metrics require analysis of source code. Building software systems with reusable components bring many advantages to Organizations. Reusability may have several direct or indirect factors like cost, efforts, and time. It may also have the issues like whether reusability is for the entire component or only for a selected service provided by that component. Reusability is the most important criteria for selecting a component for component-based systems. A highly reusable component will help in better understanding and low maintenance efforts for the application. Therefore, it is necessary to estimate the reusability of the component, before integrating it into the system. Present thesis adopts Fuzzy logic based approach to estimate the reusability of component. It also proposes to improve the Reusability factors. Fuzzy Logic based approach has several advantages over other methods including Neural Network and others. One major advantage is that it may also work without the data. So, Fuzzy logic will result in better understanding the reuse task.

## REFERENCES

[1] "Sonu Mittal, Pradeep kumar Bhatia" *Framework for Evaluating and Ranking the Reusability of COTS Components based upon Analytical Hierarchy Process*", IIJST 2013

[2] T. Karthikeyan, J. Geetha," *A Study and Critical Survey on Service Reusability Metrics* ",*I.J. Information Technology and Computer Science,* 2012, 5, 25-31 Published Online May 2012 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijitcs.2012.05.04

[3] Arun Sharma, Rajesh Kumar, and P. S. Grover, "*A Critical Survey of Reusability Aspects for Component-Based Systems*", World Academy of Science, Engineering and Technology 33 2007

[4] V. Subedha, S. Sridhar, "*Design of Dynamic Component Reuse and Reusability Metrics Library for Reusable Software Components in Context Level", International Journal of Computer Applications (0975 – 8887) Volume 40– No.9, February 2012*

[5] Vidushi Sharma and Prachi Baliyan," *Maintainability Analysis of Component Based Systems*", International Journal of Software Engineering and Its Applications Vol. 5 No. 3, July, 2011

[6] Arun Sharma, Rajesh Kumar, P S Grover, "*Managing Component-Based Systems With Reusable Components*", International Journal of Computer Science and Security, Volume 1 : Issue (2)

[7] Nasib S. Gill," *Reusability Issues in Component-Based Development*"., FEB 2011 ,IRCJA

[8] Danail Hristov, Oliver Hummel, Mahmudul Huq, Werner Janjic," *Structuring Software Reusability Metrics for Component-Based Software Development***",** ICSEA 2012: The Seventh International Conference on Software Engineering Advances

[9] Prakrit Trivedi, Rajeev Kumar ," *Software Metrics to Estimate Software Quality using Software Component Reusability",* IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 2, No 2, March 2012 ISSN (Online): 1694-0814 www.IJCSI.org

[10] DR. P. K. SURI, NEERAJ GARG," *Software Reuse Metrics: Measuring Component Independence and its applicability in Software Reuse*", IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.5, May 2009

[11] "MIGUEL GOULAO, FERNANDO BRITO," *An overview of metrics-based approaches to support software components reusability assessment* ", Quinta da Torre, 2829-516 Caparica, Portugal

[12] ''G.N.K.suresh babu and Dr.s.k.srivatsa ", *analysis and measures of software Reusability ",* IJRIC 2009, International Journal of Review in Computing.

[13] "Majdi Abdellatiefab, Abu Bakar Md Sultana, Abdul Azim Abd Ghania, Marzanah A.Jabara*" , Component-based Software System Dependency Metrics based on Component Information Flow Measurements"*, ICSEA 2011, The Sixth International Conference on Software Engineering Advances.

[14] ", V. Lakshmi Narasimhan, P. T. Parthasarathy, and M. Das", *Evaluation of a Suite of Metrics for Component Based Software Engineering (CBSE)*", Informing Science and information Technology vol 6 2009.

[15] ", Jeffrey S. Poulin", *Measuring Software Reusability", Third* International Conference on Software Reuse Brazil, 1994.

[16] "Mollaghasemi, M., Pet-Edwards, J., "*Technical briefing: making multiple objective decisions*", IEEE Computer Society Press, Los Alamitos, CA 2007

[17] "V. bhardwaj, "Estimating reusability of software components using Fuzzy logic", M.tech. Thesis, 2010.

[18] "Gill, N.S., "*Importance of Software Component Characteristics for Better software Reusability*", ACM SIGSOFT SEN, 2006. 31(1): p. 1-3.

[19] "A. Sharma, P.S. Grover, and R. Kumar, *"Reusability assessment for software components*", SIGSOFT Software Engineering Notes, vol.34, No.2, February 2009, pp.1-6.