A Framework for Automated Assessment of Changeability and Reusability

Isha Rani[1],Pinki Tanwar[2]

[1]Student, dept. of c.s.e, JMIT,Kurukshetra Univesity,Radaur,India
[2]Assistant professor, dept. of c.s.e, JMIT,Kurukshetra Univesity,Radaur,India

---

ABSTRACT-*Software reuse is the method of implementing or updating software systems using accessible software components. A good software reuse procedure facilitates the increase of productivity, reliability and quality. the diminish of costs and implementation time. There is an urgent need to understand how these software components can be implemented as plug and play devices and changeability of software components can be understand in some tangible framework. Therefore, in this research work we are solving this issue by building a framework which helps to measure degree of reusability by using clustering methods (machine learning).*

Keywords: Component, software component, software development with reuse, properties of software metrices , k means++ algorithm.

1. INTRODUCTION

1.1 COMPONENT- Components are the gathering of numerous pre programmed tools which are used as the add-on page which is to make use of those tools. There are a variety of tools accessible to calculate the Java source code. Those tools are developed a number of tools will calculate some parameters to be measured in java program. Initially the tools which are used to determine the java object oriented programs are searched and analyzed individually to make it as a component. The component based tools will be execute independently but available in a same position and some tools will offer a chart for the results when the program is executed.[1]

1.2 SOFTWARE COMPONENT

Software Component is a cover up of software implementations which define well clear interfaces. Software components can be a portion of code, function, unit or class, scheme or software itself and when these components get included they form an whole application .[2]
Some basic properties of the software components are:[1]
A software component can be a code block, module, function, class, control or the project or software itself.A software component can be end product or it can be extendable. The software component can be language dependent or language independent.A software component is the unit of interfacing that conceptually specifies it's internal and the external interfacing with main application.A software component can be online or the offline product or code. A software component can also be a deliverable software object
As a software component is not an individual expression it is the essential concept that gives the software reusability in some way. several kind of inner or interfacing in software in the form of individual components are represented in the form of software components. Each of the software language describes most of software components in dissimilar way. [1]

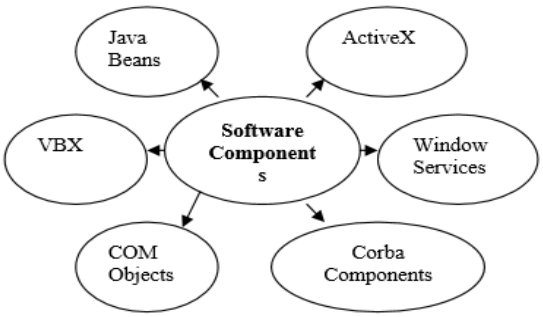

Fig1: Software Components

Software Reusability -Software reuse is the method of implementing or updating software systems using accessible software components. A good software reuse procedure facilitates the increase of productivity, reliability and quality. the diminish of costs and implementation time. An initial investment is required to create a software reuse process, but that speculation pays for itself in a few reuses .In short, the development of a reuse process and repository manufacture a base of knowledge that pick up in quality behind every reuse, minimize the sum of development work required for future projects, and ultimately reducing the risk of new projects that are based on warehouse knowledge .[1]

Why Reuse?
Reuse has been established to offer many rewards. When we reuse code, components and other artifacts, our objective are to :

- Reduce the cost of developing the product
- Reduce time to market..
- Improve the predictability of the development process
- Improve the productivity of the development teams.
- Increase the quality and reliability of the product
- When reuse is mentioned, we often consider only of code reuse.

Software development with reuse: Software development with reuse is an method which tries to exploit the reuse of existing software components. advantage of this approach is that overall development costs of the software are reduced . Cost reduction is only one potential profit of software reuse. Systematic reuse the development offers for further advantages:

- System reliability is increased: Reused components in working systems must be more dependable than new components. These components have been tested in diversity of operational systems environment and have therefore been uncovered to realistic operating conditions.
- Overall process risk is reduced: If we use a function which is by now exists, there is less doubt in the cost of reusing that in the cost of development. For project management this is essential factor as it reduce insecurity in project cost exclusion. If relatively huge components such as sub systems are reused then this become true.
- Effective use defined by specialists: Application specialists doing the same work on diverse project environment instead these specialists can build up reusable components which encapsulate their knowledge.
- Organizational standards can be embodies in reusable components: We can reuse a number of standards such as user interface standard which can be executed as a set of standard components.

Software reuse is the method of implementing or update software systems using existing software resources. A good software reuse procedure facilitates the increase of quality productivity and reliability, and the development of costs and implementation time. An original investment is required to start a software reuse process, but that investment pays for itself in a few reuses.[1]

METRICS - Software metrics are used to calculate the software quality to ensure its requirements. Metrics are described as "Quantifiable measures that could be used to calculate characteristics of a software system or the software development method." Software metrics are necessary to plan, monitor, predict, control, evaluate, products and processes. The main aim of the software metrics is to reduce costs, Control /Monitor schedule, it prove quality small testing effort, many reusable fragments, to better recognize the quality of the product and the program.[3]

METRICS CATEGORIZATION–Metrics can be classify into three Kinds, measures and size.
Two types of software metrics they can be Product metrics: quantify characteristics of the product being developed to calculate the, reliability, size and Process metrics: quantify characteristics of the development being used to develop the software to approximation the efficiency of fault detection.
Types of methods are Direct Measures (internal attributes):
To calculate the Cost, effort, speed , LOC ,memory and Indirect Measures (external attributes): to calculate the Functionality, complexity, quality, reliability, efficiency, maintainability.
Size-oriented metrics are KLOC - 1000 Lines Of Code, LOC -Lines Of Code, LOC – Statement Lines of Code (ignore whitespace).[3]

PROPERTIES OF A SOFTWARE COMPLEXITY METRIC - A software complexity metric is applicable if it succeeds in fulfilling defined properties. numerous researchers have tried to explain a set of properties that a good software complexity metric should satisfy.
Property 1: Nonnegativity: A complexity metric value can not be in a negative number. For a few complexity metrics it is essential to be even stricter, since a value of zero will not for all time be accepted. Interpretation guidelines: The importance of a complexity metric worth for a software manufactured article that provides some functionality to be identical to zero is that the artifact is the least-complex probable design that can provide that functionality. A lower complexity value, for two functionally identical designs, is preferred over a superior value since lower complexity is believed to be connected with less development, testing, and maintenance efforts.

Property 2: Scalability: A software complexity metric should offer a scale of values. Comparison among diverse alternatives should be possible. For any two software artifacts it should be possible to evaluate and then make managerial decision according to the metrics values. For any two functionally-identical components C1 and C2, if Complexity(C1) > Complexity(C2) then C2 is preferred over C1 assuming that keep all other parameters constant. This is due to the reality that C2 will need, less integration, less testing and less maintenance efforts. Also, metrics must offer enough information to help managers make business decisions and compare diverse alternatives.

Property 3: The complexity of single software unit S collected of two software components may not be less than the calculation of the complexities of the individual components.

Complexity(S) >= Complexity(C1) + Complexity(C2)

According to the metrics explaind in , the complexity of a component oriented software system is a function of the complexities of individual components that formulate it up, and an added complexity will appear as a result of new connections that may exist among the components. In the best case, when a system is composed of two components and no new added interactions among the components are available, the system's complexity will be same to the sum of the individual component complexities.

Property 4: If a component C is decomposed into two or more components C1, C2, .., Cn then the sum of complexities of the resulting components is no extra than the overall complexity of the original component. Complexity(C1) + Complexity(C2) + … + Complexity(Cn) <= Complexity(C). The motive for this is that, according to observation of the three-level component-oriented software complexity, there is usually an added complexity whenever two components are composed. The complexity usually results from the communications amid these components. So, when the component is decomposed these relations will disappear and only the component's intrinsic complexity will remain.

Property 5: The complexity value of one component does not have a straight relation to its functionality for any two components C1 and C2, if Complexity(C1) > Complexity(C2) then it is not essential that C1 provides more functionality than C2. The same functionality can be obtained by dissimilar designs and then implementation. The complexity procedures described in this article are those that allow software developers and/or managers to take decisions and contrast/compare diverse alternative solutions to the same difficulty. Of course, any added functionality may initiate an added complexity. So, a complexity metric does not believe evaluating functionality of the system or offer any information about the system size.

Property 6: The complexity value is openly influenced by structure. Two dissimilar structures for the same functionality can result in two dissimilar complexity values. A complexity measure of the system can have diverse values for dissimilar alternative architectures of the same functionality.[4]

COMPONENT GENERAL METRICS

Some of the significant metrics applicable for the examination of components quality in design stage are:

- Inferences from the LCOM metric: A high value for the LCOM metric implies that there usability of that component is high and the component is relatively less complex
- Inferences from the WMC metric: if value of the WMC is more, reusability is considered low.
- Inferences from the NOC metric: A high value for the NOC metric implies that the component is highly reusable
- Inferences from the DIT metric: If the value of DIT is high, reusability is high and complexity is high
- Inferences from the CPD Metric: From the theoretical analysis, if a high value for the CPD metric implies that the reusability decreases[2]

K-MEANS ++ ALGORITHM- K-means clustering algorithm is responsive to the initial value, namely dissimilar initial values may lead to dissimilar clustering results . expected at the deficiency of K-means algorithm, K-means++ algorithm is proposed. beginning the input data sets, this algorithm firstly randomly choose a point as the first initial clustering center, then according to the D2 weighting technique to decide the next point as the initial clustering center, until you decide K initial clustering centers. The algorithm can choose widely distributed initial cluster centers, and get enhanced clustering results K-means clustering algorithm is responsive to the initial value, namely dissimilar initial values may lead to dissimilar clustering results . [5]

The idea of K-means++ algorithm- K-means algorithm is firstly to randomly choose K sample spot as the K initial cluster centers from the clustered data sets. Then it analyze the distance the rest of the sample spot respectively to K initial clustering centers, select the nearest sample, and then allocate it to the corresponding cluster, and the cluster center iterative bring up to date until it satisfies the least squared error function or does not alter until the cluster center. And the K-means++ clustering algorithm is to improve the defect of instability brought by the K-means algorithm randomly choose initial cluster centers. The work of K-means++ algorithm is reflected in the choice of the initial cluster centers, the essential principle is to make the distance among the initial centers as large as possible .

The basic steps of the algorithm are as follows:

1) From the input data sets we randomly choose a sample point as the first original clustering center.

2) For each sample spot of the remaining data collection, it will measure the distance to the nearest cluster center point D2 (x) and the D2(x) is stored in an array, and then add up these distances Sum (D 2(x)).

3) Then, referring to Sum(D2(x)), randomly choose a random value, weight to compute the next initial clustering centers. The algorithm implementation is,

take a fall in the Sum (D2(x)) the Random values in the Random, and then use the Random -= D 2(x), until it <= 0, this point is the initial clustering center.
4) Repeat 2), 3) until the K initial cluster centers are selected.
5) The input K initial clustering center, like as the input of K-means algorithm.[5]

2. RELATED WORK

The research work performed in this field by many researchers and the work is presented as follows:
Nael Salman et al (2006)[4]The work presented in this paper introduce a set of metrics for component oriented software systems. it focuses mainly on the complexity that results mainly from factors connected to system structure and connectivity. Also, a set of properties that a component-oriented complexity metric should possess are defined. The metrics have been evaluated using the properties clear in this paper.

K. A. Abdul Nazeer et al(2009)[6]-appearance of modern technique for scientific data gathering has resulted in large scale accumulation of data pertaining to different fields. Conventional database querying techniques are inadequate to take out useful information from huge data banks. Cluster examination is one of the major data analysis technique and the k-means clustering algorithm is widely used for several practical applications. But the original k-means algorithm is computationally exclusive and the quality of the resulting clusters heavily depends on the selection of initial centroids. some methods have been proposed in the literature for improving the presentation of the k-means clustering algorithm. This paper proposes a system for making the algorithm more effectual and efficient, so as to get better clustering with decrease complexity.

Gholam Reza Shahmohammadi et al(2010)[7]The selection of software architecture style is an significant result of design stage, and has a significant impact on different system quality attributes. To establish software architecture based on architectural style selection, the software functionalities have to be distributed amid the components of software architecture. In this paper, a technique based on the clustering of use cases is proposed to find out software components and their responsibilities. To choose a proper clustering method, first the proposed technique is performed on a number of software systems using dissimilar clustering methods, and the results are established by expert opinion, and the best scheme is recommended. By sensitivity analysis, the outcome of features on accuracy of clustering is evaluated. Finally, to decide the appropriate number of clusters (i.e. the number of software components), metrics of the internal cohesion of clusters and the coupling amid them are used. Advantages of the proposed technique include; 1) no need for weighting the features, 2) sensitivity investigation of the effect of features on clustering accuracy, and 3) presentation of a clear method to recognize software components and their responsibilities.

Jianguo Chen et al(2011)[8]In latest years, the software engineering community has put considerable attempts into the design and development of component-based software system (CBSS) in order to handle the software increasing complexity and to exploit the reuse of code. This paper presents diverse of such efforts by investigating the improved measurement tools and method, i.e., through the useful software metrics. Upon the research on the classical valuation measures for software systems, the author argue the traditional metrics are not appropriate for CBSS. Therefore author present an account of novel software measures for component by adequate cohesion, coupling and interface metrics. The complexity metrics merge with three metrics on the CBSS level is also investigated. The advantages of author's methods are discussed as well during a case study in this paper

Prakriti Trivedi et al (2012)[1]Today the majority of the applications residential using a number of codes, existing libraries, open sources etc. As a strategy is accessed in course, it is represented as the software module Such as in .net ActiveX controls and java beans are the software mechanism. These components are complete to use programming rules or controls that excel the system growth. A component based software organization defines the perception of software reusability. While by means of these mechanism the main question occur is whether to use such components is helpful or not. In this planned work we are tiresome to present the reply for the similar question. In this occupation we are presenting a position of software matrix that will verify the interconnection among the software element and the application. How well-built this relative defines the software value after using this software part. The generally metrics will revisit the final product in terms of the unlimited of the part with application.

Divya Chaudhary et al(2013)[9]Component based software engineering is one of the main advancement in the ground of software engineering. It is a process that highlight the design and construction of computer based systems using reusable software components. It offers the methodology of developing a large software systems. It carry both the Commercial-off-the-shelf and in-house components. This paper talk about the component based software engineering fundamentals. It also emphasizes the process involved. This paper surveys the different current metrics for component based software

engineering systems namely for cost, complexity etc. in detail. The metrics help in enhance the quality and risk management in the component based system.

Suchita Yadav et al(2014)[10]In Component-Based Software Engineering (CBSE), it is essential to calculate the reusability of components in order to understand the reuse of components effectively because reusability is an efficient way to recover productivity in CBSD; it is required to calculate the reusability of components. This study will suggest a modified reusability metrics suite and reusability assessment model by using reuse, complexity and adaptability factors. while the complexity of a software component decide that how easy it is to get used to the component in the new context of use, Reuse of the component can also be used to deduce how usable and how easy to adapt it, adaptability define that how easy it is to adapt a component to the context of the developer.

Muhammad Husnain Zafar et al(2015)[11]Software reuse is the procedure of implementing or updating software systems using accessible software components. A good software reuse process facilitates the raise of quality, productivity and reliability. It reduce the cost and implementation time as compared to build up new system. even with its many benefits the author cannot achieve its full benefits. The cause behind this is that software reuse is often done in an relaxed and haphazard way. If done systematically, then author can achieve its full benefits. This research proposes a technique through which author will classify the reusable components in proper way to get the full benefits of reusability. The author define the reusable components according to their clusters. Clusters can be made on the source of parameters present with components. The writer design an algorithm for assigning clusters to the reusable components.

3. PROPOSED WORK

Problem formulation-When a car is manufactured, thousands of components are developed to get the final complete car. Most of the components of the cars are developed in such a way that these components are interchangeable and can be used in any another car if needed and if one component is not functioning properly it can be interchanged with other component by plugging in and out with other components, so that changeability of the components is increased and maintenance can be made easier. There is an urgent need to understand how these software components can be implemented as plug and play devices and changeability of software components can be understand in some tangible framework. Therefore, in this research work we are solving this issue by building a framework which helps to measure degree of changeability by using clustering methods (machine learning) , which would be best suited for our problem definition and better from previous research works.

Objectives-

Develop representative dataset of projects(object-oriented). Classify parameters which impact the changeability& reusability of software components. Develop a framework and to calculate the reusability, changeability metrics. Using machine learning algorithm to develop automated assessment of changeability& reusability. estimation of proposed model using recall and precision

4. RESULTS AND ANALYSIS

To build up a cost- effective and quality products is an significant and challenging feature of software development. Component-based software development can help developers to produce well-organized software within the time and budget constraints. The idea of component-based software engineering (CBSE) is set on the development of self-determining and loosely coupled components of the system, by avoiding unrelated dependency amid system components.

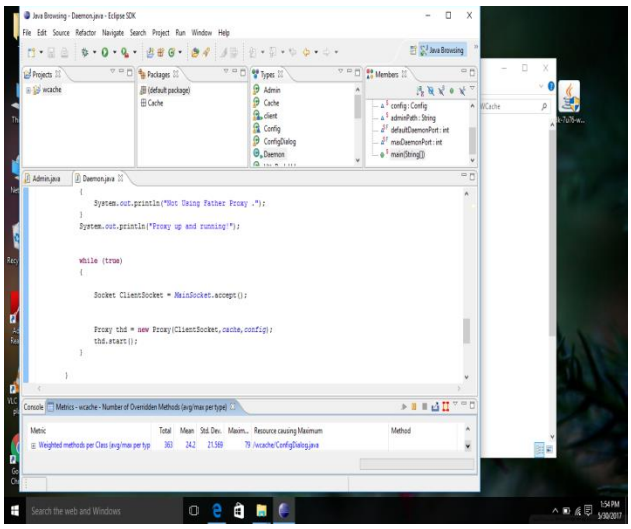

Figure4.1:calculating metrices using and eclips.

The figure shows that different metrices values are calculate

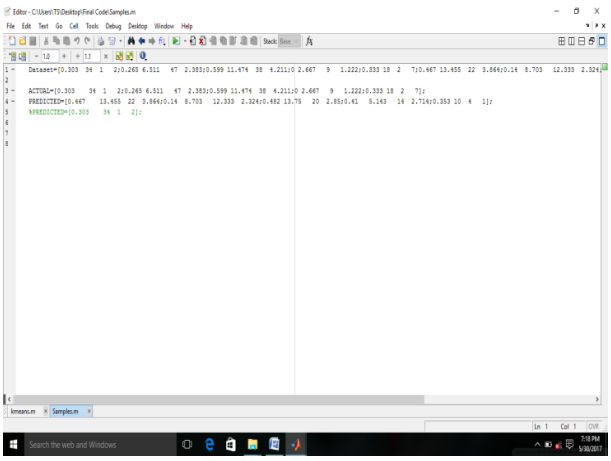

Figure4.2:shows values for data set ,actual values and predicted values.

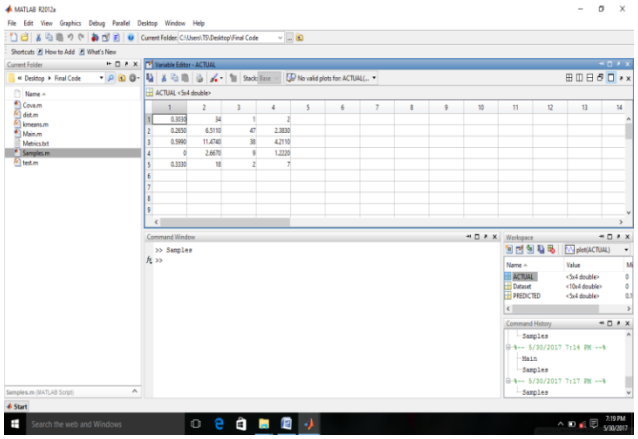

Figure4.3:this figure shows actual training values for sample programe.

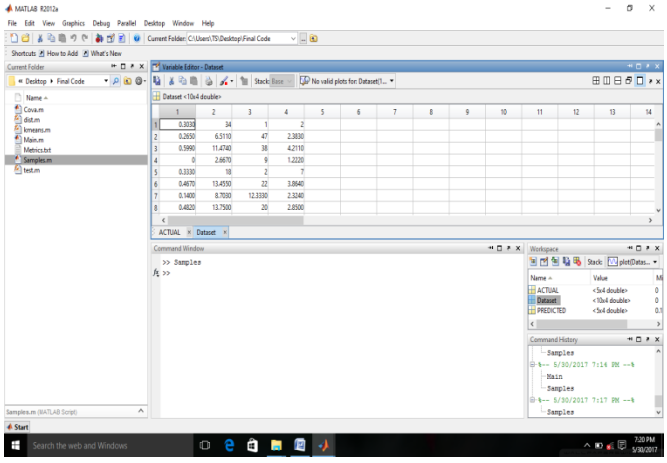

Figure4.4: shows actual datasets for sample program.

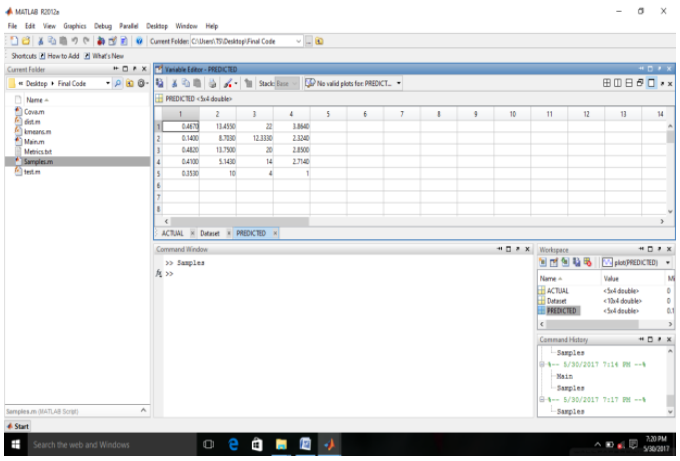

Figure 4.5: shows the pridicted values to be tested for sample programe.

The figures defines the ten data set values and five training values for the actual dataset and five tested values for pridicted data set.

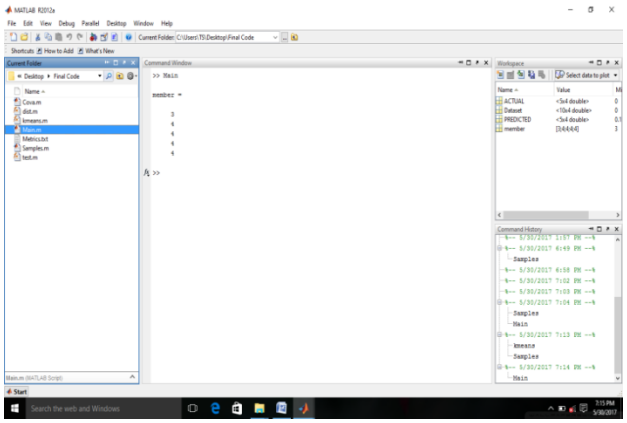

Figure4.6:shows first project to be tested and it belongs to third cluster and other four project belongs to fourth cluster

5. CONCLUSION AND FUTURE SCOPE

The proposed model can calculate the reusability of software components by using k means and clustering methods. The proposed approach applied on 10 projects, first five project define training values and other five project define predicted values to be tested. It also defines which project belongs to which group of cluster. This information can be taken by proposed work. The existing work can be extended by considering more metrics like Sloc (source line of code), Rco (rate of component observability), Emi (existence of meta information) can be consider for better performance and evalution of proposed work.

6. REFERENCES

1) Prakriti Trivedi " 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.
2) Shweta Bhambri1, Sheetal Chhabra2" Estimation of Software Reusability Based on Clustering" DOI 10.4010/2016.1682 ISSN 2321 3361 © 2016 IJESC.
3) P. Edith Linda "Metrics for Component Based Measurement Tools" International Journal of Scientific & Engineering Research Volume 2, Issue 5, May-2011 1ISSN 2229-5518.
4) Nael SALMAN' Complexity Metrics AS Predictors of Maintainability and Integrability of Software components' Çankaya Üniversitesi Fen-Edebiyat Fakültesi, Journal of Arts and Sciences Say›: 5, May 2006.
5) ZHANG Min " Improved research to k-means initial cluster centers" 2015 Ninth International Conference on Frontier of Computer Science and Technology.
6) K. A. Abdul Nazeer' Improving the Accuracy and Efficiency of the k-means Clustering Algorithm 'Proceedings of the World Congress on Engineering 2009 Vol I WCE 2009, July 1 - 3, 2009, London, U.K.
7) Gholam Reza Shahmohammadia' Identification of System Software Components Using Clustering Approach' JOURNAL OF OBJECT TECHNOLOGY Published by ETH Zurich, Chair of Software Engineering © Jot, 2010 Online at http://www.jot.fm.
8) Jianguo Chen 'Complexity Metrics for Component-based Software Systems 'International Journal of Digital Content Technology and its Applications. Volume 5, Number 3, March 2011.
9) Divya Chaudhary' International Journal of Advanced Research in Computer Science and Software Engineering' Volume 3, Issue 7, July 2013 ISSN: 2277 128X. Suchita Yadav 'Metrics Suite for Accessing the Reusability of Component Based Software' IJARCSSEll Rights Reserved Volume 4, Issue 5, May 2014 ISSN: 2277 128X.
10) Muhammad Husnain Zafar 'Classification of Reusable Components Based on Clustering'I.J. Intelligent Systems and Applications, 2015, 10, 55-62 Published Online September 2015 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijisa.2015.10.07.