International Journal of Technology Research and Management
ISSN (Online): 2348-9006
Vol 3 Issue 7 July 2016

An Approach to Find Software Reusability Based on Object-Oriented Metrics

Mr. Pavan Devade[1], Prof. Pritesh Jain[2]
Patel College of Science and Technology, PG scholar, CSE Department, RGPV University, Indore, Madhya Pradesh, India[1]
Patel College of Science and Technology, Reader, CSE Department, RGPV University, Indore, Madhya Pradesh, India[2]
pawan1devade@gmail.com[1], pritesh.arihant@gmail.com[2]

Abstract: *The Software reusability has considerable effect on software quality and productivity, software quality increases with reuse of software components and source code directly relates to cost and quality. Software Reusability is the likelihood of a segment of source code that can be used again to add new functionalities with slight or no modification in software.. Software programming is a complex task, mainly due to the complexity involved in the process. Reusability is the ability to implement or combine independent software components in large units. The metrics that evaluate object oriented metrics are: classes, methods, inheritance. Very few metrics are presented for object oriented source code. In this Dissertation, a measurement has been done for measuring the Reusability of source code using Object Oriented Metrics. The aim of this proposed work is to identify and analyze the object oriented programming to increase reusability and improve software quality through object oriented metrics. Object oriented metrics have been applied on programs of inheritance the metrics values are evaluated. This thesis presents a measurement to measure Reusability in object oriented code using Object Oriented Metrics.*

Keywords: *Reusability, coupling matrices, DIT, WMC, LOC.*

1. INTRODUCTION

To extend the development of any application's productivity its matainabilty and Reusabilty is one of the best options. Firstly a good tested software component with reusable features must be found.

A Programmer's developed application can be useful as a component to others, thereby demonstrating that code specifics to application requirement can be reused to develop projects associated with similar necessities.

Measurement may help us to build reusable components along with reusable components among the wealth of existing programs. Existing programs contain the knowledge and experience obtained from working in the specific application domain and meeting the organization's software needs. If we could pull out this information effectively, we could retrieve a valuable resource upon which to build future applications. Reusability saves time for developing the software which ultimately reduces the cost of application development. This also improves the software performance. And objective of the any system organization is giving the product with good quality and reduced the low cost [1].

Software reusability has considerable effect on software quality, software quality increases with reuse of software components and source code directly relates to cost and quality, but software quality cannot be improved unless it is measured. OO measurement is being used to evaluate and predict the quality of the software [2].

Reusability software developed for calculating the Reusability of the object oriented programs. This developed software will be calculate reusability of existing software component artifacts following terms of portability, understandability, maintainability, adaptability and characteristics of quality that would be utilize. And also this reusability software will be help to decision take support almost which is particular reusable components should be reused.

Coupling: The Coupling is a act of adding two things with together. In software development, coupling points to the degree to which software components are dependent upon. For instance, in a tightly-coupled architecture, its associated components and each all components must be present in order for code to be executed or compiled [9].

In a loosely-coupled architecture, components can remain autonomous and that allows middleware software for managing the communication between them. Coupling: Degree of dependence among components High coupling makes modifying parts of the system difficult, e.g., the modification of any component affects all the components to which the component is connected [9].

Cohesion: Cohesion is refers to the degree to which the elements of a module belong together. Thus, Cohesion is the measure of how each piece strongly-related of functionality which is expressed by the source code of a software module. The Module with cohesion refers on the lightly bound the internal elements of the module are to each another. The coupling between the modules is lower when the cohesion is greater. The internal Cohesion of a module is calculated in terms of the strength of the hiding of the elements inside the modules itself [9].

2. SOFTWARE METRICS

The Software Metrics is a 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. 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 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 component-based systems and their quality issues [2].

Object Oriented Metrics:

Weighted Method per Class (WMC)
Number of Children (NOC)
Depth of Inheritance Tree **(**DIT)
Line of Code (LOC)
Response for a Class (RFC)
Coupling Between Object (CBO)
Comment Percentage (CP)
Lack of Cohesion in Methods (LCOM)

Table: 2.1 Object Oriented Metrics

WMC	Weighted Method per Class
NOC	Number of Children
DIT	Depth of Inheritance Tree
LOC	Lines of Code
RFC	Response for a Class
LCOM	Lack of Cohesion in Methods
CBO	Coupling Between Object
CP	Comment Percentage

Metric 1: Weighted Methods per Class (WMC)

WMC measures the complexity of any individual class. There are two main approaches are used and to the calculate the WMC metric. The first uses the sum of the complexity of each method contained in the class. In a second approach assigns a complexity of 1 for each method in the class and then sums the result. This is equivalent to using the number of methods per class as a measure for WMC.The number of methods and complexity of methods involved is a direct predictor of how much time and effort is required to develop and maintain the class [10].

Weighted Method per Class (WMC) metrics is applied towards calculating the structure complexity of the programs. Method complexity is measured by using Cyclomatic Complexity and WMC is sum of complexity of the all methods which is applied in class. Let consider class is getting the methods (m1, m2, and m3…mn) and complexity of the methods are (c1, c2, and c3…cn) then [3].

$$WMC = c1+c2+c3+\ldots. +cn;$$

Cyclomatic Complexity causes base of the graph theory and is computed in one of the 3 directions. Number of regions in flow graph.

Cyclomatic Complexity find out in flow graph as follow

$$C\ (G) = E - N + 2;$$

Where N is the no of the nodes in a graph and E is the no of the edge in the graph. Cyclomatic Complexity defined in flow graph as follow [4]

$$C\ (G) = P+1;$$

Where 'P' is a number of predicate nodes in a graph. Statement where we are taking some decision are called predicate node.

Metric 2: Depth of Inheritance Tree (DIT)

Depth of inheritance (DIT) metric is applied for measuring the inheritance complexity for the programs, when programmers usages the inheritance in their program then this Metric can be utilized [4].

Depth of inheritance (DIT) is the Maximum depth from the root node of tree to special node. Here class is represented as a node. Deeper node in the tree accepts more number of methods because they inherit and the more classes in the tree and it make the class more complex.

Metric 3: Number of children (NOC)

The Number of children (NOC) is applied when there are many numbers of the Sub- Classes of the Particular class in hierarchy of the class exist. If children of a class are more, it requires more test because super class may be misused [3].

Public Interface Size:

If the number of the public method is delivered in the class than Public interface size is determined, which describes how much other class is using that class method.

Metric 4: Coupling between object classes (CBO)

Coupling between object classes (CBO) for a class is a count of the no. of other classes to which it is coupled. It describe to the notion that an object is coupled to other object if one among them acts on the other, i.e., methods of one use methods or instance variables of another. As described earlier, since objects of the same class have the same properties, two classes are coupled while methods declared in one class use methods or instance variables defined by the other class [1][3].

Metric 5: Response for a Class (RFC)

The response set of a class is a set of methods that may probably be executed in response to a message received by the object of that class [7]. The cardinality of the set is a measure of the attributes of objects within the class. Since it specifically includes methods which are called from outside the class, which is also a measure of the effective communication between the class and other classes.

RFC = | RS | where RS is the response set for the class.

Metric 6: Lack of Cohesion in Methods (LCOM)

The larger the number of same type of methods, the more cohesive the class, which is consistent with traditional notions of cohesion that measure the interrelatedness among portions of a program. If none of the methods of a class display any instance behavior, i.e. they do not use any instance variables, they have no similarity and the LCOM value for the class will be zero.

The LCOM value gives a measure of the relative disparate nature of methods in the class. A smaller number of disjoint pairs (elements of set P) implies similarity of methods is greater. LCOM is tied to the instance variables and methods of a class, and so it is a measure of the attributes of an object class.

Metric 7: Comment Percentage (CP)

Comment Percentage (CP): Comment Percentage(CP) is computed by number of comment line separated along Line of Code. High evaluate of the Comment Percentage (CP) increases the maintainability and understandability [3].

CP = Comment Line / LOC;

Metric 8: Line of Code (LOC)

Lines of Code (LOC): Lines of Code (LOC) metrics applied for measuring the size of the program by considering the no of lines in program. Lines of Code (LOC) counts to all lines like as source line and the number of statements, the number of blank lines and number of comment lines.

3. METHODOLOGY

We collect the all metrics determined for extract reusability of object oriented software that are suggested in the literature. And effort to develop a software that would be implements them. To calculate reusability automatically of object oriented programs. Using the object oriented programming for developing the system. This Software is allowing the flexibility. Because Software concentrate on the quality factors like understandability maintainability, portability, interface size, adaptability, reliability which affects the reusability attributes. We can easily add it and utilize the functionality of the module [1].

Reusability is depends on followings like adaptability, portability, understandability maintainability and reliability. We are dealing with the C# .NET programs that way portability is not issue for us. Complexity is of two types i.e. Inheritance complexity and structure complexity [2], and we are treating with static code, so we are not consider the reliability an influence which is the reusability, since reliability is measured in terms of the average time and error which is calculated, on the execution of the program. Understandability depends on the structure complexity, documentation level of the programs and size [3].

Figure 3.1 shows the factors in which reusability are depend. Adaptability, portability, maintainability reliability and understandability these are the main factors use to calculate the reusability of source code. This are the factors in which the reusability of software is depend [4].

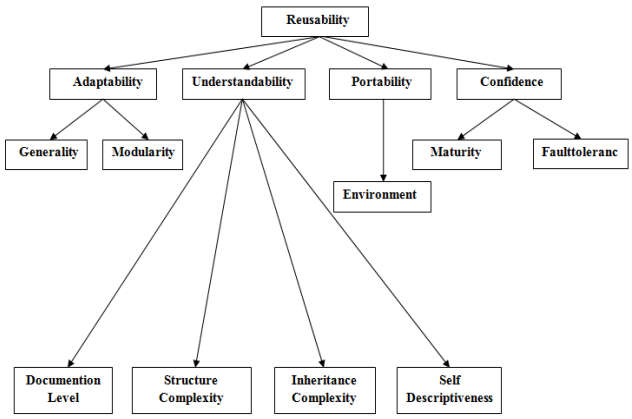

Figure 3.1: shows the Factor and Metrics in which Reusability depends

3.1. Understandability

Understandability is that stage in which the sense of the software system or module should be allowed to the developer or user.

Understandability depends on the following elements, which are Documentation level, size and complexity. understandability of the module is high when modules are well documented i.e. new developer understand module code easily because module having more comment lines, since what cause function do describe in the starting of the purpose. Understandability is also depends on the size of the module. When size of the module is high and then itself difficult to understand [4][8].

3.2. Maintainability

The degree to which the module or system of the software can be modified easily in order to adding quality attributes, fix bugs, or for adjustment of the operating environment change, increase efficiency of the system [8].

Maintainability depends on the following the factor like modularity, RFC, size, complexity. If module comprises more complex data structure in his program and more decision statement, we can say module is more complex. If the Complexity of the module is higher than module is difficult to maintain. Modularity is measure by using the coupling metrics, cohesion metrics and Maintainability highly depends on the modularity.

3.3. Adaptability

Adaptability determines as how easily software satisfies requirement or and user requires of the new environments from being system and system constraints. Now suddenly business environment or business require is changed, thus handling this situation adaptability is one of the important component or weapon. Business market situate is change frequently so our software system should be adaptable to satisfy this requirement. It doesn't intend whatever software. We build up from OOP is always adaptable [8].

4. PROPOSED METHODOLOGY

One of the difficult task it to identify the reusable module. We have already explain component on which reusability depend. Most important element on which reusability depends are complexity and coupling. When the coupling and complexity are high of module and reusability of that module is low. Formula for calculating the reusability of objected oriented program is described below [1].

$$\text{Reusability of a class} = a*(DIT) + b*(NOC) - c*(CBO)$$

Where a, b, c are empirical constants [1]. DIT is depth of inheritance tree, NOC is number of child and CBO is coupling between object. DIT and NOC is positive impact on reusability and CBO has negative impact on reusability.

5. APPROACH FOR IDENTIFICATION OF REUSABLE MODULE

In few of the organizations have applied systematic reuse programs, which have resulted in-house libraries of reusable components. Other different organizations have supported their reuse with own techniques and software to recover components [4].

Step follow to identify the reusability

- Extract the source code.
- Calculating the Metrics.
- Display.

Extract the source code: In this stage we analyzed the source code and pull out valuable information and store it in memory, which is needed for calculating the all metrics.

Calculating the Metrics: In this phase we calculate, object oriented metrics. And result of the all metrics is store in memory. All the metrics are concern by object oriented system.

Display: We give them some weighted to each metrics and finally we determine the reusability of the sources code, Figure 5.1 show the Step follow for identified the reusable module [3][4].

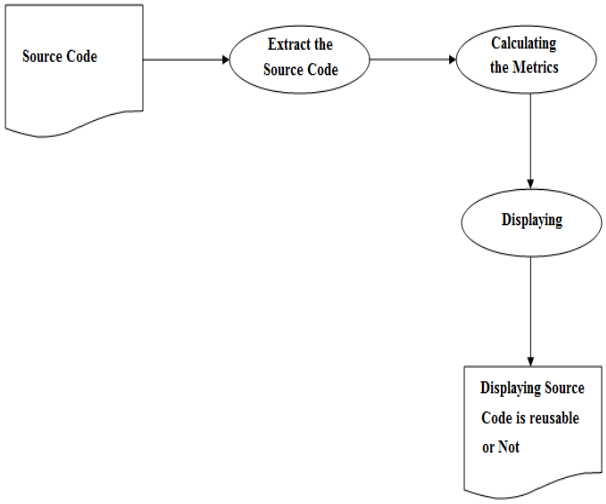

Figure 5.1: show Step follow for identified the reusable module

Why Reuse ?

It has been proven to offer many rewards. When we reuse code, components and other artifacts, our goals are to [5]:

- Reduce the time to market.
- Reduction of cost of developing product.
- It improves the output of the development terms.
- It improves the predictability of the development.
- Process Increase the reliability and quality of the product.
- When reuse is mentioned, we often think only of code reuse.

Software reuse is the process of implementing or updating software systems using existing software asset. A good software reuse process facilitates the increase of productivity, quality, and reliability, and the process of costs and implementation time. An initial investment is require to start a software reuse process, but that investment pays for itself in a few reuses [6].

6. DESIGN & IMPLEMENTATION DETAILS

6.1. GUI of the system

Figure 6.1 show the main Graphical User Interface of the system. This is the index form of the system:

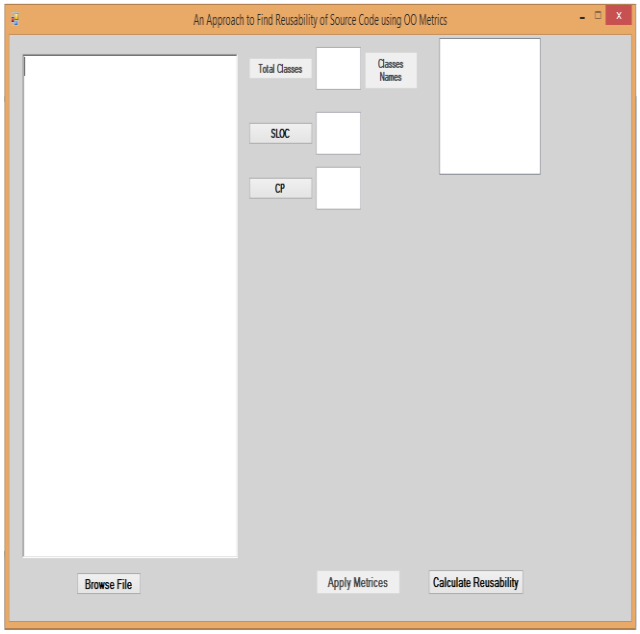

Figure 6.1: GUI of the System

Figure 6.1 show the main GUI of the system. This is the GUI form developed in C# programming language. In this form some GUI control are used. In this GUI form browse file, apply metrics; calculate reusability etc is button controls. And one Rich Textbox is used for show the selected source code, some text boxes are used for display the value and one list box are used for the show name of all classes. GUI programming is user friendly language.

7. RESULTS

7.1. Systems for Test

Now select the C# source code file for the calculate metrics. Following systems were taken for test:

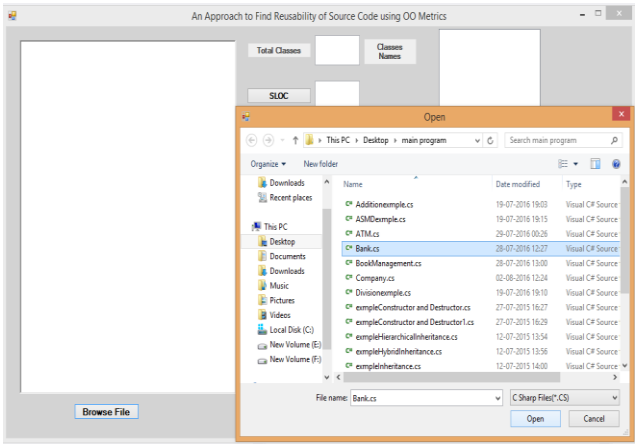

Figure 7.1.1: Browse the Source Code as Input

Figure 7.1.1 show the select the source code as input for calculating the reusability.

7.2. Calculating Reusability of Bank Class

For calculating reusability of class we browse the source code of Bank class as input. And then extract the data from source code that are taken in input and apply the software reusability metrics on source code and show the reusability of classes.

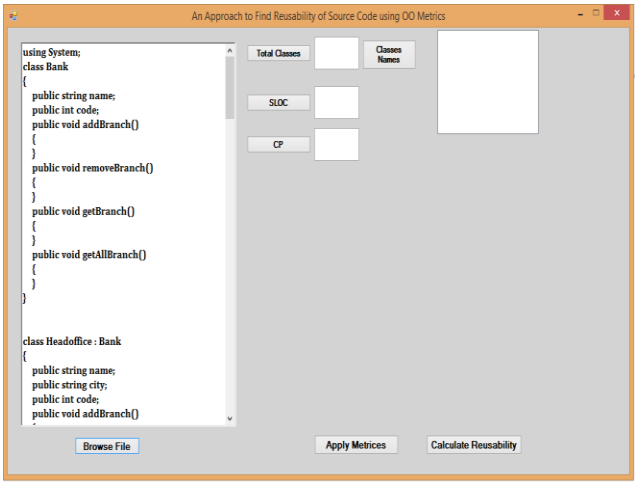

Figure 7.2.1: Browse the Bank Source Code as Input

Now after loading source code file we can calculate total number of classes, in this C# programming source code:

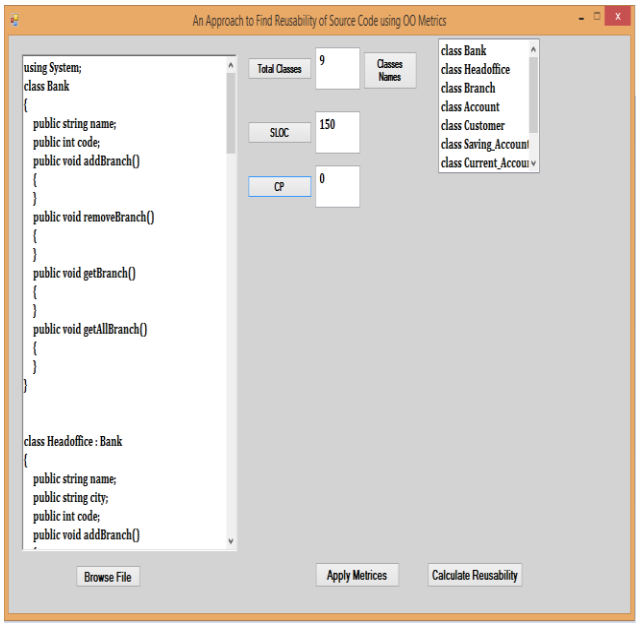

Figure 7.2.1.1: Browse the Bank Source Code as Input

Now after loading source code file we can calculate total number of classes, in this C# programming source code:

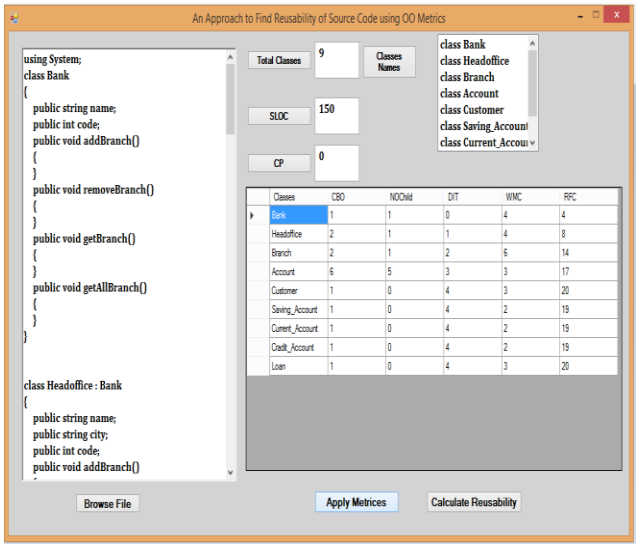

Figure 7.2.1.3: calculating the metrics for Bank

Now we can perform final comparison and the result will be appearing. The maximum reusability in source code is 5.

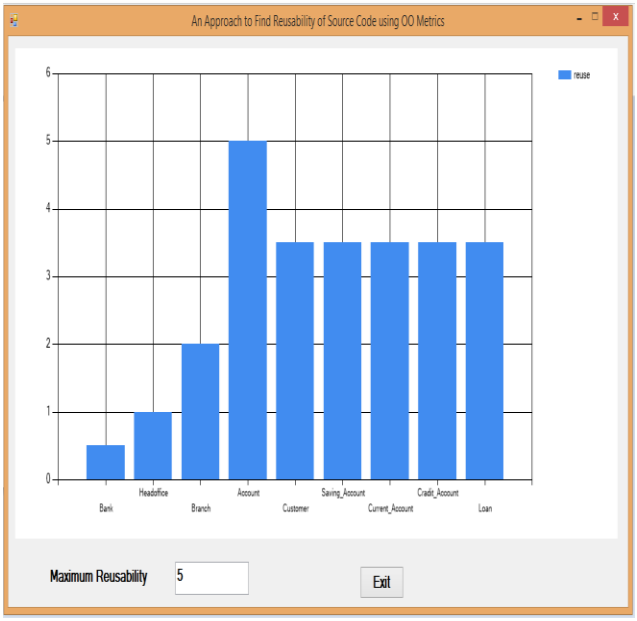

Figure 7.2.1.4 Reusability value for Bank

Figure 7.2.1.4: show the reusability of each classes of source code.

8. CONCLUSION

The purpose of this thesis is to find the way to evaluate reusability of object oriented programs. Reusability is one of the quality attribute which is of key feature in object oriented software development. As this method leads to increase in developer productivity, minimising development cost and also time to market. The work presented in this thesis can be effectively used to evaluate the reusability of any give object oriented software module.

Proposed approach estimates the reusability on the basis of different metrics based on inheritance, number of children and coupling. This work can be used in any organization or industry to found the most reusable module from number of modules to improve their productivity and quality.

When the class cohesion is increased and coupling between object is reduced, the reusability of source code will also get increased. When the coupling measures are reduced and cohesion measures are increased, the classes can function more independently. In future work we can apply various other Object Oriented metrics (like coupling factor, lack of cohesion etc) to identify or to compare better reusability and quality results for given source code. This software or system only checks the reusability in C# source code. In future we are developed a software those are check the reusability in all Object Oriented Code. As in our Interface, the code shown in text box have to be compiled by language compiler before using, we didn't provide any compile or error checking services in our implementation.

REFERENCES

[1] "An Approach to Measure Software Reusability of OO Design", Selim Kebir, Abdelhak-Djamel Seriai, Allaoua Chaoui, Joint Working Conference on Software Architecture & 6th European Conference on Software Architecture, 978-0-7695-4827-2/12 $26.00 © 2012 IEEE, DOI 10.1109/WICSA-ECSA.212.26, 2012

[2] "A Novel Coupling Measure difference between Inheritance and Interface to find better OOP Paradigm using C#". WICT, 2011 978-1-4673-0125-1c 2011 IEEE, Narendra, Ravindra Gupta, 2011.

[3] "Significance of Software Metrics to Quantify Design and Code Quality", S.Arun Kumar T.Arun Kumar P.Swarnalatha, International Journal of Computer Applications (0975 – 8887), Volume 11– No.9, December 2010.

[4] "An Approach For Calculation Of Reusability Metrics Of Object Oriented Program", Avinash Dhole and Nehil Rao Nirmal, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 2 Issue 6, June – 2013.

[5] "A Method for Assessing the Reusability of Object-oriented Code Using a Validated Set of Automated Measurements", Dandashi F ACM 2002 pp 997-1003, 2002.

[6] "An Empirical Study on Object- Oriented Metrics", Tang, M., Kao, M., and Chen, M., EEE Transactions on Software Engineering, 0-7695-0403-5, 1999.

[7] "Software Quality Metrics for Object Oriented System Environment" by SATC (Software Analysis Technology Center) SATC-TR-95-1001, June 1995.

[8] "An Approach to Analysis Software Reusability", International Journal of Advanced Research in Computer Science, Ashish Agrawal, Volume 3, No. 3, May-June 2012,

[9] http://www.indiawebdevelopers.com/articles/reusability.asp.

[10] http://www.boddunan.com.articles/education/19-engineering/156-cohesion-and tstypes.html.