# An Approach to Analysis Software Reusability

| Ashish Agrawal* | Sachin Patel |
|---|---|
| Information Technology | Information Technology |
| PCST Indore, India | PCST Indore, India |
| Ashish_agrawal13@yahoo.com | Er.sachinpcst@gmail.com |

*Abstract:* Reusability is an only one best direction to increase developing productivity and maintainability of application. One must first search for good tested software component and reusable. Developed Application software by one programmer can be shown useful for others also component. This is proving that code specifics to application requirements can be also reused in develop projects related with same requirements. The main aim of this paper proposed a way for reusable module. An process that takes source code as a input that will helped to take the decision approximately which particular software, reusable artefacts should be reused or not.

## I. INTRODUCTION

The degree to which is a software Reusability module or early product work can be using in more than one system computing or software program. Many believe software reusability provides the key to tremendous benefits and saving in software development product. The U.S. Department of Defence only could be save $300 million annually by increasing its level of reuse by as little as 1% [l].Measurement may help us not only to learn how to build reusable components but also to identify reusable components among the wealth of existing programs. Existing programs contain the knowledge and experience gained from working in the particular application domain and meeting the organization's software needs. If we could extract this information efficiently, we could gain a valuable resource upon which to build future applications. If any Software system using the reusable component from existing System. Reusability save the lot of time for developing the software and also reduced the cost of the software development. It is also increases the software performance. And objective of the any system organization is giving the product with good quality and reduced the low cost.

Today everyone is interested to increase the productivity, and reducing the cost of developing products, and better quality of software providing. There are various types of quality attributes available from which one can be identify the software quality. Reusability is way to increase the productivity, which is to reusable the existing component. Because a lot of time and effort already have expended for the developing product. Old software component that is well tested and designed already [15]. So that one should reuse the being software. That problem is many times module are not developed for reusable, extends to highly product development time and cost. Thus, one should explore system which existing component or module is more suitable for reuse, and try to reuse.

## II. REUSABILITY

Reusability measurement is allowing for identify the reusable modules from being program and way to build. Some Department and Organization, along the increasing level of software reuse, development cost taken to develop the software and save the cost and time. Department of U.S. saved 300 million $ by increasing the 1% reusability software. Being software programs contain the knowledge's and experiences of the developers who are good in particular application domain area. So we take out information from being program which comfort to require of the software system then it is beneficial for the organization.

There are many examples where reusability helps

a. In Missile Systems Division (MSD) using the computer software reuse concept it increased the 50% productivity.
b. American Navy utilizes the reusable modules which reduce 26 % of labour required to develop and maintain the Restructured Naval Tactical Data Systems (RNTDS).
c. Magnavox saw when we are using the reusable modules to develop the Force Fusion System Prototype (FFSP); it reduces the 20% of the development time of estimated time for developing the new system.

We determined from requirement stage to last stage of the software development .In Software development life cycle Reuse concept is not determined to only coding stage. These are several phases where software development is reused:

a. Code
b. Requirement
c. Architecture/design documentation
d. Test plans
e. Specifications
f. Design
g. Manuals
h. Templates
i. Design decisions
.

## III. PREVIOUS WORK

The measurement of the reusability will help developers to control the current level of the reuse and providing the metrics for identifies one of the important quality properties reusability. Conte [14], Boehm [12], Bailey [2], and Fenton [3] describe reusability measuring that are based on comparisons between the size of the newly written code in particular software product and the length or size of reusability code. Change of the reusability code is not considered by these researchers. Conte reuse measure is estimating coding effort. Reusability reduces the coding effort. These reductions impact the effort estimation techniques. In standardized way Boehm And bailey use the size of reused code to adjust cost predictors And Fenton also develop a method which measuring the reuse based on the dependency in an associated call graph. All these above measurement develop by the researchers not supporting the modification. Selby[16] providing the approach which supporting the modification of reused code Selby classifying the module into a particular category based on the percentage of reused Code is modified in make new module. These categories are:

a. Complete new module.
b. Reused Module with grater then equal to 25 percentage changes.
c. Reused Module with less than equal to 25 percentage change.
d. Module that are reused without change.

All these measurement are based on only one Attributes program size or program length. Selby add one another attributes modification. [6] Reuse Measure Derivation As indicated in the title, this paper gave a description of how to derive measures of software reuse in object oriented systems [5]. The method used to derive measures of software reuse, is derived from measurement theory and was described as.

a. Define considerable and identify and nonrational, realized attributes of software reusability, we must qualitatively understand what we want to evaluate.
b. Determine precisely the attributes to be measured and the documents. The measurement must be able to specify and indications of the measurement and the aim with higher precision.
c. Development of formal methods with higher level of abstractions which describe the attributes. Formal designs for definition to produce quantitative evaluate to study these abstractions without any ambiguity in them.
d. Devising the relationship of the model with the attributes with this quantitative measurement. These relationships must be in consistency with the specification of attributes and quantitative values.

### A. Assessing Module Reusability:

In these paper Propose the Conceptual model for estimating the reuse of the module. Reusability [7] determines the reusability of the module as product of its applicability and functionality. In this paper define model of Reuse as function of Applicability and Functionality.

### B. Functionality:

Functionality of the module is determined as number of situation in which this module is Use, based on the specification. Important Problems will get a high functionality, while a module that covers a few specific, rare problems will get a low functionality.

### C. Applicability:

Applicability measuring the number of situations in which a module can be reused. When the module can only be applied in 50% of the situations where its specified features are needed, then its applicability is 50%. The applicability of a module is 100% when any situation that calls for features provided by the module, the module can actually be used. There are many reasons why a module's applicability might be less than 100%, including:

a. Technical limitations, such as programming languages and platforms.
b. Incompatibilities by details of the interface.
c. Incompatibilities by external dependencies.
d. Architectural mismatches.

## IV. DEFINITION OF REUSES TYPES

Bieman et al's [6] determined the various types of reuse and defined the three types of reuse in the three perspectives. These are explained as follows:

### a. Public Reuse:

Fenton define the public reuses as "the proportion of a product which was constructed externally "[3].

Public reuse = length (E) / length (p);
E is the code developed externally.
P is the new system including E.

### b. Private Reuse:

Fenton defines private reuse (or possibly more appropriately internal reuse) as the "degree to which modules between products are reused within the same product" [3].

Fenton uses the call graph which represents the flow connection of the module .In these graphs node represents the module and they are connected through edges. If one node calls another node then edge displays the connection between them. Fenton provides the formula for calculating the private reuse in call graph as follows

$$R (G) = e - n + 1;$$

e is total no of edges in graph. And n is the number of the nodes in graph.

### c. Leveraged Reuse:

In Leveraged reuse means modifications of reuse is allowed.

### d. Direct Reuse:

Direct reuse is reuse without using the intermediate entity. One module directly calls another module.

### e. Verbatim Reuse:

In Verbatim reuse means modifications of reuse is not allowed.

### f.    Indirect Reuse:

Indirect reuse is reuse through an intermediate entity. When first module calls second module and second module calls the third module then first module indirectly calls the third module.

### A.   Three Perspectives for Reuse:

Bieman [6] provided the three perspective view for identifying the reuse views. These are described as follows:

a.   **Server Perspective:** perspective of the library component known as a server perspective. Server reuse of the any class will characterize how one class is using the other class.

b.   **Client Perspective:** Client perspective means how one particular entity is using the other entity i.e. how the new system using the existing system.

c.   **System Perspective:** it is the combination of the both server perspective and client Perspective.

## V.  PROPERTIES OF SOFTWARE WHICH AFFECT REUSABILITY

There are some software attributes which affect the reuse. The relationship between these attribute and the reusability are explained as follows:

a.   **Complexity**: when the complexity of the class is high or for developing the class developer uses a complicated structure then that type of class is difficult to reuse and difficult to understand.

b.   **Complexity of interface**: Complicated interface make reuse difficult.

c.   **Class size**: when the size of the class is large then it is difficult to understand and difficult to reuse.

d.   **Dependencies:** Dependency of the single module to various modules may also make reuse more difficult.

Reusability is depending about the, adaptability, portability, maintainability reliability and understandability [4]. We are dealing with the java program that way portability is not issue for us. Complexity is of two types' structure complexity and inheritance complexity. And we are treating with static code therefore we are not considering the reliability an affect which is the reusability, since reliability is measure in terms of the average time and error which is measured, on the execution of the program. Understandability depending on the structure complexity, documentation level of the programs and size.
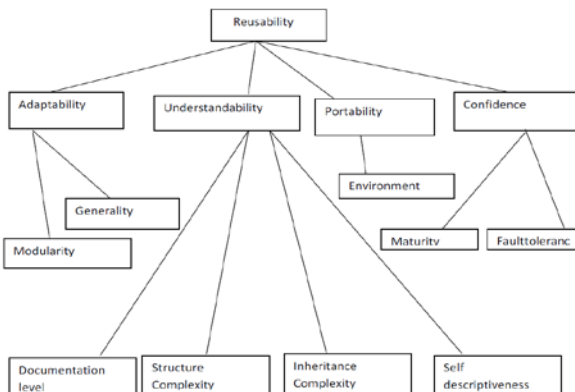


Figure 1 Factor in which Reusability Depends

### A.   Understand ability:

Understandability is the level in which the meaning of the software system or module should be authorizing to the developer or user. Understandability depend along the following element, these are Documentation level, size and complexity. When module are well documented then understandability of the module is high i.e. module having more comment line so new developer understand module code easily, since what cause function do describe in the starting of the purpose. Understandability is also depending along the size of the module. When size of the module is high and then itself difficult to understand. If the Complexity of the module is high then module is difficult to understand. We tell module is more complex when module holds more composite data structure in his program and more decision statement Complexity is two types beginning is structure complexity it was easily measure along WMC metrics and second is inheritance complexity; this complexity is measurement by using the DIT and NOC metrics. When the program using the inheritance concept or cover the class then these metrics is used for measuring the inheritance complexity [9].
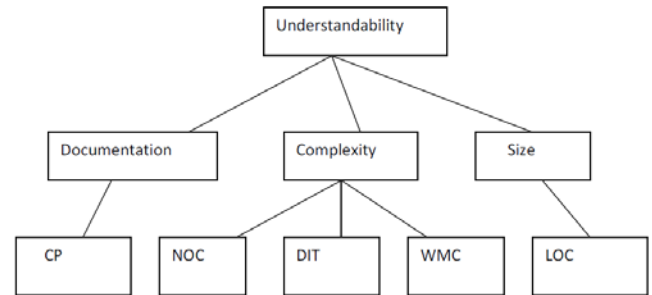


Fig 2 Factor and Metrics in which Understandability depends

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

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

$$CP = Comment Line / LOC;$$

c.   **Weighted Method per Class (WMC):** This metrics is applied towards calculating the structure complexity of the programs [13]. Method complexity is measured by using Cyclomatic Complexity and WMC is sum of complexity of the all methods which is applied in class. Suppose class is getting the methods ($m_1$, $m_2$, and $m_3$…$m_n$) and complexity of the methods are ($c_1$, $c_2$, and $c_3$…$c_n$) then

$$WMC = c_1+c_2+c_3+…. +c_n;$$

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

Cyclomatic Complexity determined in flow graph as follow

$$C (G) = E – N +2;$$

Where N is the no of the nodes in graph and E is the no of the edge in the graph.

Cyclomatic Complexity defined in flow graph as follow

C (G) = P+1;

Where 'P' is number of predicate nodes in the graph.

Statement where we are taking some decision are called predicate node.

d. **Depth of Inheritance Tree (DIT):** This metric is applied for measuring the inheritance complexity for the programs, when programmer usages the inheritance in his program then this Metric can be utilized.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 no of the methods because they inherit and the more classes in the tree and it make the class more complex [13].

e. **Number of Children (NOC):** NOC is applied when there are many numbers of the Sub- Classes of the Particular class in hierarchy of the class exist. When children of a class are more then it requires more testing because super class may be misused [13].

f. **Public Interface Size:** Public interface size is determined when a number of the public method deliver in the class. Which describe how much other class is using that class' method?

### B. Maintainability :

The degree to which the system or module of the software can be modified easily in order to fix bugs, adding quality attributes or for adjustment of the operating environment change, increase efficiency of the system.  Maintainability depends on the following factor, modularity these are size, complexity [8, 9].  We say module is more complex if module comprises more decision statement and more complex data structure in his program. When the Complexity of the module is high then module is difficult to maintain. Modularity is measure by using the coupling metrics and cohesion metrics and Maintainability heavily depends on the modularity. Modularity is idea for managing the complexity of complex system by dividing it into various small modules and this module is communicates by using interface. We handle the complexity by using the concept of the modularity. By using the concept of modularity we find clear and simple system. When the module is high cohesive then it is easy to maintain. If module having the more no of the local method then it difficult to maintain. And if size of the module is high then it difficult to maintain if the coupling of any module with other module is low then this module is easily modified and easy to maintain.

Maintainability is also depending on the cohesiveness of the module. If we write code for any module and interested in measuring the quality of module then the best criteria for measuring the quality of the code is measure based on quality factor maintainability. Maintainable code is more flexible, code is not maintainable because module performing several functionality and invoking other module [10]. Any module is more maintainable if we add easily new functionality and exchange existing functionality of the module. Metrics victimized for calculating maintainability of the program, Shown in the Figure 3.
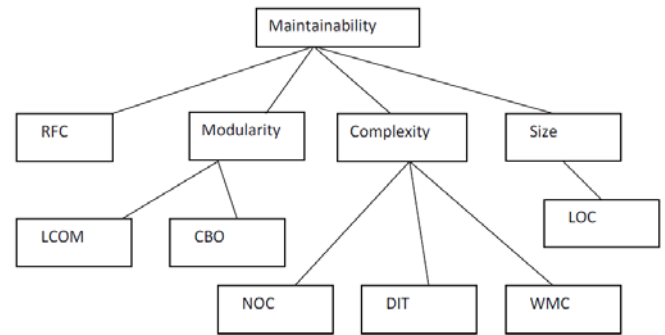


Figure 3 Factor and Metrics in which Maintainability depends

### C. 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 situation 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. In object oriented concept applying the ability to build adaptable software [8]. When the coupling of module is low and cohesion is high that signifies module is easily adjust in new environment from old environment. For make adaptable module we concentration on cohesion and coupling of the module. Metrics used for calculating adaptability of the program, Shown in the Figure 4.
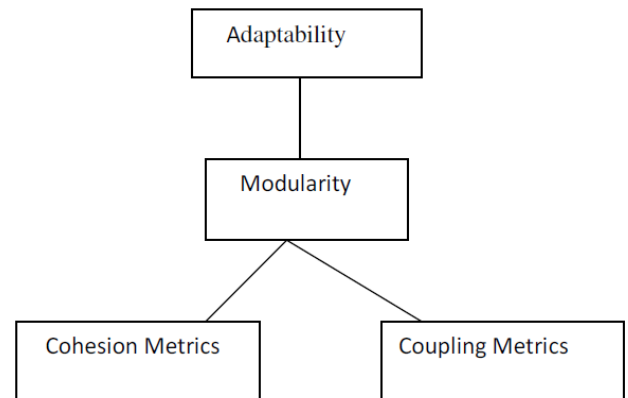


Figure. 4 Factor and Metrics on which Adaptability depends

a. **Coupling:** Coupling is also called dependency. Coupling is the level to in which one module is depend on the other module signifies. It is using the functionality of the other module. Coupling is one important component which helps you to determine the quality of the design or software. Design of the designer or good programmer is to archive low coupling [11].

b. **Cohesion:** Cohesive signifies that a certain class performs a set of closely related to actions. A lack of cohesion means that a class is performing and various unrelated tasks. Principle of object oriented say increase reduces the coupling between modules and cohesion of the module. It is beside beneficial for architecture point of view.

Cohesiveness means each functions in the class perform one affair. When this happened in the class then new designer can well understood what class performs [11].

## VI. APPROACH FOR IDENTIFICATION OF REUSABLE MODULE

a. ***Extract the source code***: In this phase we analyzed the source code and extract useful information and store it in memory, which is necessary for calculating the all metrics, these metrics are necessary for evaluate factor on which reusability depend.

b. ***Calculating the Metrics***: In this phase we calculate, all metrics which describes in above section, for implementing these metrics, we used information gathering from extract phase. And result of the all metrics is store in memory. All metrics are concern with object oriented system.

c. ***Display:*** In this phase we calculate the reusability of the source code. We give the some weighted to each metrics and finally we determine the reusability of the sources code. And display source code is reusable or not.
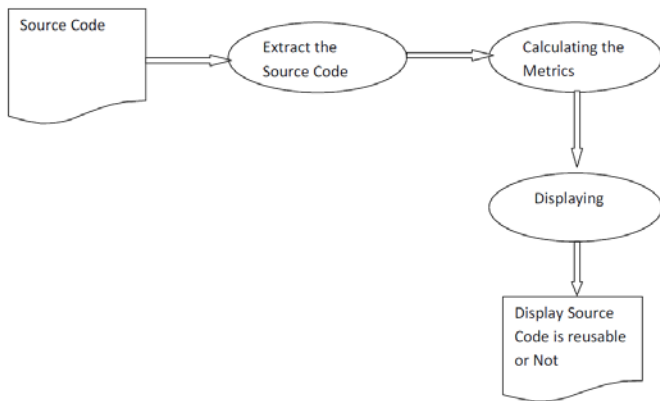


Figure 5 Steps follow for identified reusable module

## VII. CONCLUSION

The purpose of this paper is to finding the approach and way to calculate reusability of object oriented programs. Reusability is one of the quality attribute and it is of prime importance in object oriented software development as reusability leads to increase in developer productivity, reduce development cost as well as reduce time to market. The work presented in this paper can be effectively used to calculate the reusability of any object oriented software module.

## VIII.REFERENCES

[1]    Software Reuse Plans BringPaybacks," Computeworld, Vol. 27, KO. 49, pp.73-76. Anthes, Gary I I.,

[2]    J.W. Bailey and V.R. Basili. "A s meta-model for oftware development resource expenditures". Proc. Fifth Int. Conf. Software Engineering.Pages107-116. 1981

[3]    Norman Fenton. "Software Metrics A Rigorous Approach" .Chapman & Hall, London, 1991

[4]    Software Reusability Vol II Applications and Experiences, Addison Wesley, 1989.

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

[6]    James M. Bieman "Deriving Measures of Software Reuse in Object Oriented Systems" Springer-Verlag 1992 pp 79-82.

[7]    Chris Luer, "Assessing Module Reusability", First International Workshop on Assessment of Contemporary Modularization techniques (ACoM'07).

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

[9]    Young Lee and Kai H. Chang, "Reusability and Maintainability Metrics for object oriented software", ACM 2002 pp 88 – 94.

[10]   Jeffrey S. Poulin "Measuring Software Reusability", IEEE 1994 pp 126- 138.

[11]   http://en.wikipedia.org/wiki/Coupling_(computer science)

[12]   B. W. Boehm. "Software Engineering Economics" .Prenntice Hall, Englewood Cliffs, NJ, 1981.

[13]   Shyam R. Chidamber, Chris F. Kemerer, "A metrics suit for object oriented design",1993

[14]   S.D. Conte, H.E. Dunsmore, and V.Y. Shen, "Software Engineering Metrics and Models". Benjamin"Cummings, Menlo Park, California 1986.

[15]   M. Burgin. H. K. Lee. N. Debnath, "Software Technological Roles, Usability, and Reusability, Dept. of Math". California Univ., Los Angeles, 2004.

[16]   Richard W. Selby. "Quantitative studies of software reuse". In Ted J. Biggersta and Alan J. Perlis, editors,