International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869, Volume-1, Issue-5, July 2013

Method for Estimating Reusable Module in Object Oriented Program

Nehil Rao Nirmal, Avinash Dhole,

***Abstract*— Software Reuse increases the productivity and reduces the cost and improves the Quality of the software development. Reusability is one of the most important Quality Characteristic. Therefore it is necessary to measure the reusability of the module in order to realize the reuse of module effectively. By reuse the module of existing software has increased in past recent year which impact more on the software quality. Many measuring Reusability methods have been proposed for estimating the reusability of module.**

***Index Terms*—Reusability, Modularity, Coupling, Cohesion, Inheritance.**

I. Introduction

Reusability is the degree to which existing module or component used in Software system or new project. Software Reuse reduces the cost, improves the quality and increases the productivity of the software development. Reusability is one of the quality characteristic. Therefore it is necessary to measure the reusability of the module in order to realize the reuse of module effectively. The practice of reuse of existing software has increased in recent years which have a great impact on the software quality.

Any Module can be reused in software development process in two ways which are:

→ Reuse the module without modification: it is the extent of ease with which the existing module can be used in the new development.
→ Reuse the module with modification: it is the extent to which some part of the module with modification can be reused in the development.

If we reuse the oldest module which is already develop in new system it increase the productivity because effort of the programme is reduce and it correctness is also increase because the exist system already tested..

Manuscript received June 28, 2013

Nehil Rao Nirmal, Department of Computer Science & Engineering, Raipur Institute of Technology, Raipur, India

Avinash Dhole, Department of Computer Science & Engineering, Raipur Institute of Technology, Raipur, India

II. Why Reuse is Important?

Many Organization and department, by increasing the level of the software reuse, save the time and development cost taken to develop the software. U.S. Department saved 300 million $ by increasing the 1% software reuse.[1] Reusability measurement is providing the way to build and identify the reusable modules from existing program. Existing programs contain the knowledge and experience of the developers who are expert in particular application domain. So if we extract information from existing program which meet the needs of the software organization then it is beneficial for the organization.

These are the some example where reuse helps

- In Missile Systems Division (MSD) using the software reuse concept it increased the 50% productivity.[3]
- American Navy uses the reusable modules which reduce 26 % of labor required to develop and maintain the Restructured Naval Tactical Data Systems (RNTDS).[4]
- Magnavox saw if 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.[3]

III. Definition of Reuse Types

Bieman et al's [9] defined the several 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 "[5].

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 perhaps more appropriately internal reuse) as the "extent to which modules within a product are reused within the same product" [5]. 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: [2]

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

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 a modification of reuse is allowed. [6][8]

D) Verbatim Reuse: In Verbatim reuse means a modification of reuse is not allowed.[6][8]

E) Direct Reuse: Direct reuse is reuse without using the intermediate entity. One module directly calls another module.

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.

Three Perspectives for reuse:

Bieman [9] provided the three perspective view for identifying the reuse views. These are described as follows:

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.

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

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

IV. Approach for Identification of Reusable Module

For identification of reusable module consist following step and these are: Analyzing the source code and calculating the all metrics and displaying the result. These steps are shown in figure 1

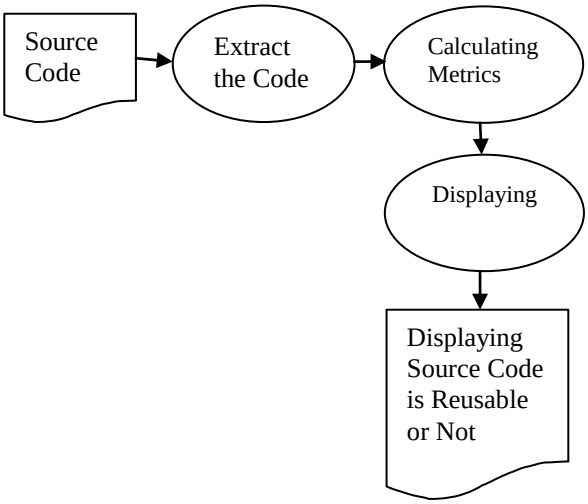

Figure-1

V. Parameters for Calculating Reusability

A. *D.I.T. (Depth of Inheritance)*

This metric is used for measuring the inheritance complexity for the programs, when programmer uses the inheritance in his program then this Metric can be used. DIT is the Maximum depth from the root node of tree to particular node. Here class is represented as a node. Deeper node in the tree has more no of the methods because they inherit the more classes in the tree and it makes the class more complex. Figure-2 shows the algorithm for D.I.T Calculator.

B. *Weighted Method per Class (W.M.C.)*

This metrics is used for calculating the structure complexity of the programs. WMC is sum of complexity of the all methods which is implemented in class. And method complexity is measured by using Cyclomatic Complexity. Suppose class is having the methods (m1, m2, and m3…mn) and complexity of the methods are (c1, c2, and c3…cn) then

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

Cyclomatic Complexity has foundation of the graph theory and is computed in one of the 3 ways. [8]

Number of regions in flow graph. Cyclomatic Complexity defined in flow graph as follow

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

Where E is the no. of the edge in the graph and N is the no. of the nodes in 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. An algorithm is given in figure-3.

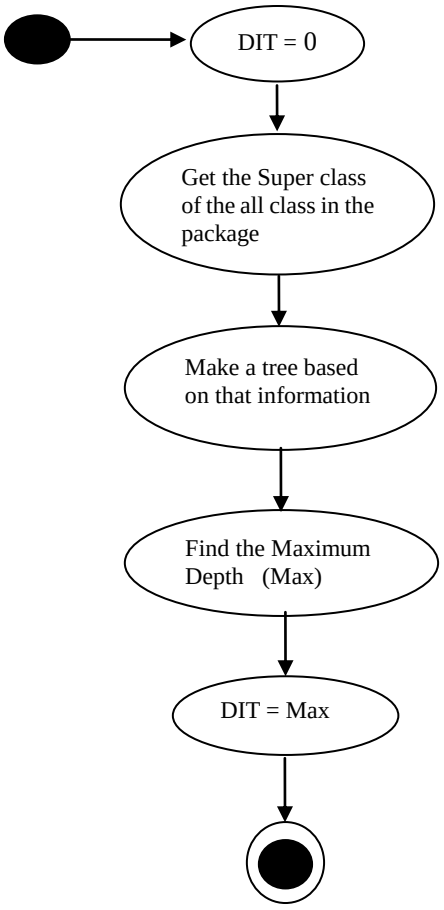

Figure-2 Algorithm for D.I.T. calculator

C. *Number of Children (N.O.C.)*

NOC is defined as Number of the Sub- Classes of the Particular class in hierarchy of the class. If class has more children then it requires more testing because subclass may misuse the super class.

D. Lines of Code (L.O.C.)

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

E. Comment Percentage (C.P.)

CP is calculated by number of comment line divided by Line of Code. High value of the CP increases the understandability and maintainability.

CP = Comment Line / LOC;

F. Public Interface Size (P.I.S.)

Public interface size is defined as a number of the public method present in the class. Which describe how much other class is using that class' method?

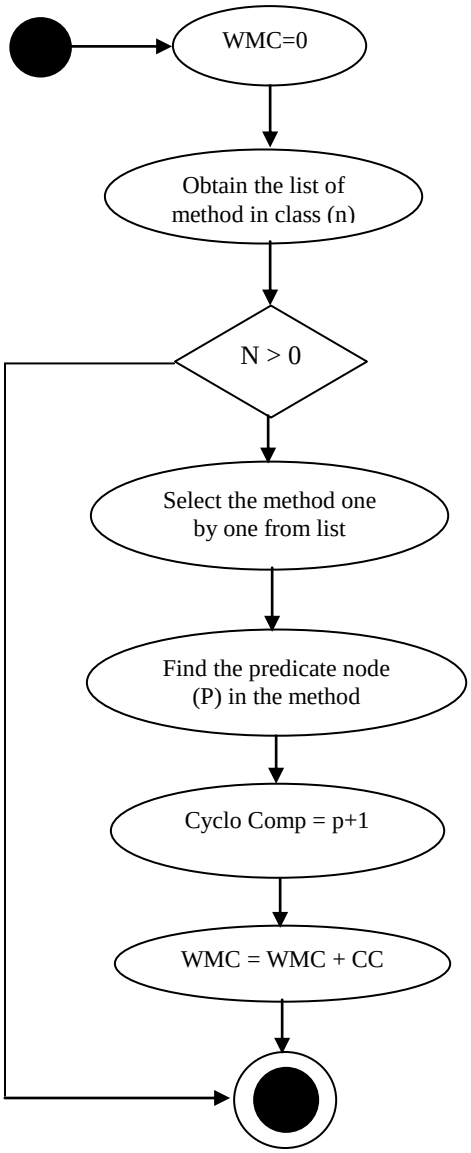

Figure-3 Algorithm for WMC Calculator

G. Module Coupling

Coupling can occur if

- A method of one class is invoked from another class
- An attribute of one class is modified / used by a method of another class
- An attribute is defined in terms of something defined in another class

We are using the Coupling Metrics for determine Coupling between the classes. This metrics is also determining the indirect Coupling between the classes. Suppose system having the Set of class C = {C1, C2, C3….Cn} and Mj = {M1, M2, M3 ….Mk} is the set of the methods which is having in class Cj.And Ri,j is defined as no of the method and instance variable in class Cj which is called by class Ci. [9]

We define Direct Coupling between class I to class j CoupD (i,j), as ratio of number of methods of class j called by class I to total no of the methods in the class i.Measure the coupling of the class is defined as

Class Coup = ∑ Coup (i,j) / (m*m-m);

H. Module Cohesion

We use the Cohesion Metrics for determining Cohesiveness of the class. This metrics also determines the indirect cohesion between the methods.
Suppose class have a set of methods M = {M1, M2, M3… Mn} and Vj = {V1, V2, V3… Vm} is the set of instance variable used by method Mj. [9]

We calculate the direct similarity of two methods Mi and Mj by using this formula.

$$SimD(i,j) = |V_i \cap V_j| / |V_i \cup V_j|$$

The cohesion of the class is defined as

$$ClassCoh = \sum \text{Sim}(i,j) / (m * m - m)$$

VI. Reusability Metrics

Identify the reusable module is one of the difficult task. We already explain factor on which reusability depend. Most important factor on which reusability depends are coupling and complexity. If the coupling is high of module and reusability of that module is low. And if the complexity of the module is high then reusability of that module is low. [6]

Formula for calculating the reusability of objected oriented program is described below.

$$Reusability = 0.25M + 0.25I + 0.25D + 0.25C$$

Where, M is Modularity of system
I is Interface Size of system
C is Complexity of system
D is Documentation of system

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

- Class size: if the size of the class is large then it is hard to understand and difficult to reuse.

- Complexity : if 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 understand and difficult to reuse.
- Dependencies: Dependency of the one module to several modules may also make reuse more difficult
- Complexity of interface: Complicated interface make reuse difficult.

VII. CONCLUSION

The Purpose of this method is 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.

REFERENCES

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

[2] [BB81] J.W. Bailey and V.R. Basili. "*A meta-model for software development resource expenditures"*. Proc. Fifth Int. Conf. Software Engineering.Pages107-116. 1981.

[3] [CDS86] S.D. Conte, H.E. Dunsmore, and V.Y. Shen, "*Software Engineering Metrics and Models"*. *Benjamin*"Cummings, Menlo Park, California 1986.

[4] [Boe81] B. W. Boehm. "*Software Engineering Economics"* .Prenntice Hall, Englewood Cliffs, NJ, 1981.

[5] [Fen91] Norman Fenton. "*Software Metrics A Rigorous Approach"* .Chapman & Hall, London, 1991.

[6] [Sel89] Richard W. Selby. "*Quantitative studies of software reuse"*. In Ted J. Biggersta and Alan J. Perlis, editors,

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

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

[9] [CK93] Shyam R. Chidamber, Chris F. Kemerer, "A *metrics suit for object oriented design"*,1993

AUTHOR PROFILE

Nehil Rao Nirmal received the B.E. degree in Information Technology Engineering from Pt.Ravi Shanker Shukla University Raipur in 2007. During 2007-2008, he stayed in State Wide Area Network Laboratory, in N.I.C. Raipur. In 2008 he joined as an assistant professor of Department of Information Technology in Raipur Institute of Technology. He is now pursuing Master of Technology in Computer Science & Engineering at Chhattisgarh Swami Vivekananda Technical University, Bhilai. He is also a research scholar/ Student of M.Tech in R.I.T., Raipur. Presently he is working on the principles of Software Engineering Metrics.

Avinash Dhole is The Head of Computer Science & Engineering Department in Raipur Institute of Technology, Raipur. He received his B.E. degree in Computer Science from Government Engineering College, Bilaspur. He joined as an assistant professor at Raipur Institute of Technology in 2005. During this period he obtained Master of Technology degree from C.S.V.T University, Bhilai in 2009. He has worked on various researches such as automata theory, software engineering and neural networks.