International Journal of Research In Science & Engineering e-ISSN: 2394-8299
Volume: 3 Issue:1 January 2017 p-ISSN: 2394-8280

An Approach to Analyse Software Reusability of Object Oriented Code

Amit Gupta[1], Dr. Pankaj Dashore[2.]
M-Tech Scholar, Sanghvi Innovative Academy, Indore1
Professor, Sanghvi Innovative Academy, Indore2

Amitgupta5997@gmail.com [1], Pankaj.dashore@sims-indore.com

Abstract: **as in today's world, time is playing a crucial role while developing the applications. Reusing something will definitely increase the productivity of the application that is needed to be developed. Reusing existing components not only save the time but also the efforts made by the development team and also the use of environment and other resources. It also helps us to reduce the cost of the product. In this paper, we have implemented a project which takes input as a object oriented code and helps us to identify whether the some part or whole code is reusable or not.**
Keywords: **software reuse, reusability, metrics, CK metrics, Cyclomatic complexity**

I. INTRODUCTION

Reusability is the 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 components also. This is proving that code specifics to application requirements can also be reused in developing projects related with same requirements. The main aim of this paper proposed a way for reusable module. A process takes source code (Object Oriented Code) as input that will helped to take the decision approximately that the given code in reusable or not. This tool will help to identify the reusability of any object oriented code, which helps in various organizations and industries that they can choose the most reusable module from existing number of modules. The reusability is one of the most important factors to improve the productivity and quality of the product with a very less cost. This chapter includes the motivation, problem definition, approaches and scope of the report. It describes the basic theme of the report and provides overall idea about research.

II. PROBLEM STATEMENT

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 feature attributes exists from which one can be identify the software quality. Reusability is the very efficient and simple way to boost up the productivity, because a lot of time and effort already have expended for the developed product. Old software component that is well tested and designed already, so that one should reuse the being software. Many times modules are not developed for reusable extends to highly product development time and cost. Thus, one should explore that which existing component or module is more suitable for reuse, and try to reuse. Thesis work proposed a reusability tool for calculating reusability for the object oriented program. By using reusability analysis tool one can be resolve that whether the existing component is suitable for reuse or not.

III. PROPOSED APPROACH

General Steps For Identification of Reusable Module [14] [21]

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 all the metrics, these metrics are necessary to evaluate factors on which reusability depend.

Calculating the Metrics: In this phase we calculate all metrics which described 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.

Display: In above phase we calculated the reusability of the source code. And in this phase we display source code is reusable or not.

e-ISSN: 2394-8299
p-ISSN: 2394-8280

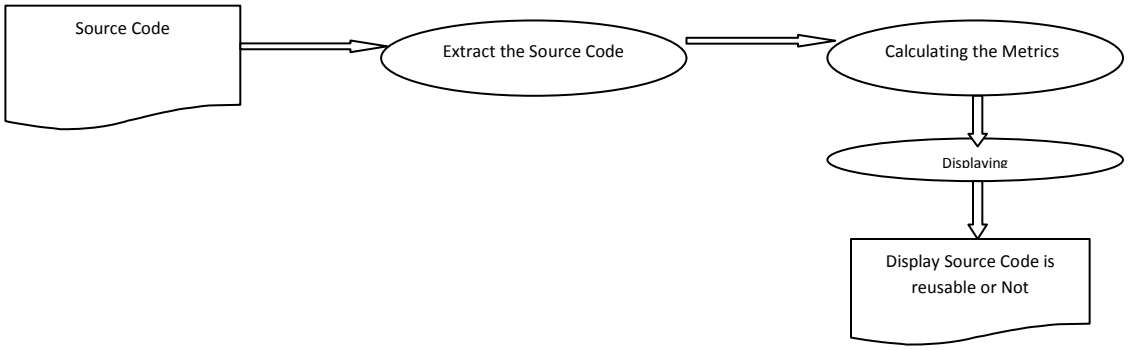

Figure 1: Step follows for identified the reusable module

Proposed Approach

Our methodology is to derive procedure to measure the reusability of OO code based on following principle [19]:

- Deeper a particular class is in the hierarchy, the greater the potential for reuse of inherited methods. [19] It states that reusability of a class increases with increase in DIT of a class. So DIT has positive impact on reusability of a class [19] [20].
- A moderate value for NOC indicates scope for reuse. [19] Up to particular threshold value NOC has positive impact on reusability of a class.
- Excessive coupling indicates weakness of class encapsulation and may inhibit reuse [19]. It indicates that coupling has negative impact on reusability of a class.

Reusability of a class = a*(DIT) + b*(NOC) – c*(CBO)

Where a, b, c are empirical constants.

For convenience, we take a = b =1 and c = 0.5

Reusability of any OO code is same as class having highest reusability [19].

IV. RESULT AND IMPLEMENTATION

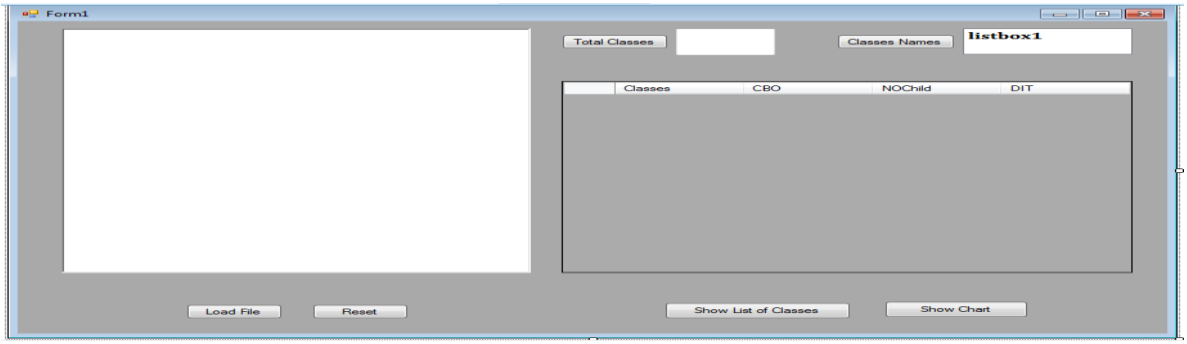

Figure 2 Interface of the System

As we have used C#.Net language to implement the tool, we use various in built controls like textbox, button, list box etc. for the same. By clicking on 'Load File' button, an Open Dialog Box window will open and we can select any file (.cs, .cpp or .txt) and the text of that selected file will appear in textbox. After that we can calculate number of classes in the program by clicking on 'Total Classes' button, then by clicking on 'Classes Names' button we got the list of all classes present in the program. After that by clicking on 'Show List of

e-ISSN: 2394-8299
p-ISSN: 2394-8280

Classes' button we got a list of classes along with there calculated metrics, then by 'Show Chart' button a chart open which shows the result of the code we have selected in textbox.

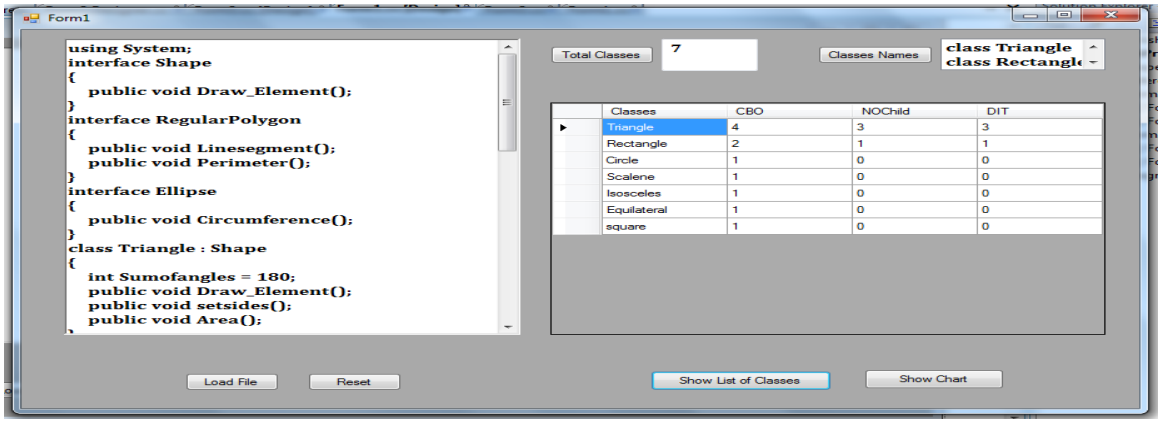

Figure 3 Interface of Shape Class

The above interface shows the example of a shape class program. We select a shape class program from opendialogbox window, then by clicking on 'Total Classes' button we got the count 7 it means that there are 7 classes exists in the program. After that we got the name of classes by clicking 'Classes Name', then displaying the class list along with there calculated metrics, then following result can be generated as shown in figure. 5.4.

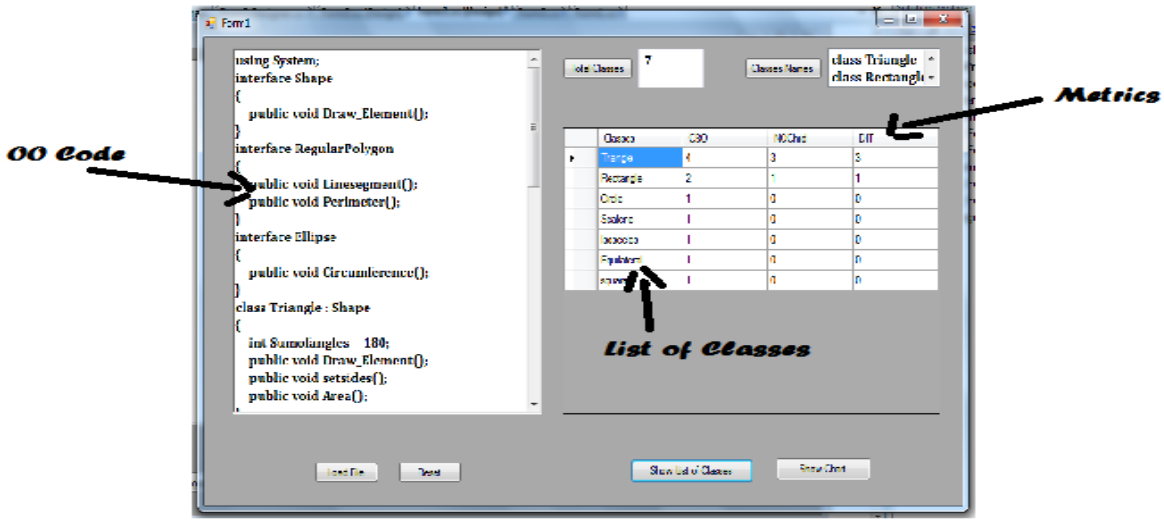

Figure 4 Interface with its elements

Above figure demonstrates the components used in the interface. Like a textbox is used to display the OO code, a grid view to show the list of metrics and classes. And various buttons to perform there own functionalities as discussed above.

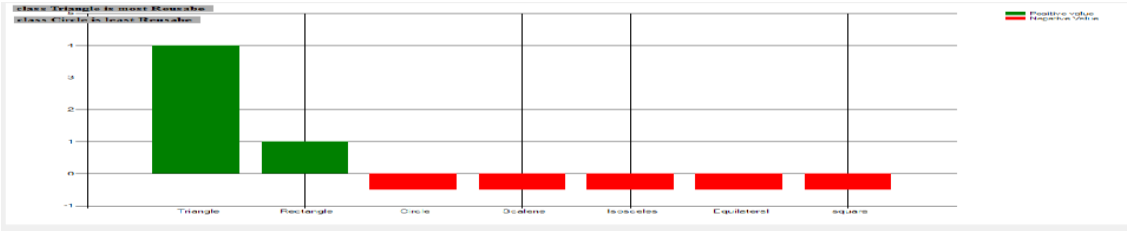

Figure 5 Result of Shape Program

e-ISSN: 2394-8299
p-ISSN: 2394-8280

Above figure shows the result of Shape class selected in Figure 5.2. The triangle class have highest bar so Triangle class is most reusable and Circle class shows minimum bar so it is least reusable.

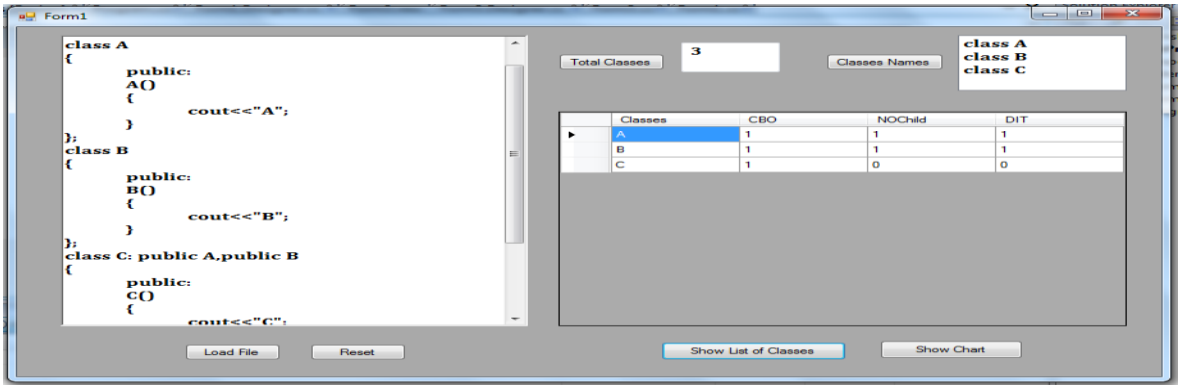

Figure 6 Interface for Multiple Inheritance

The above interface shows the example of a Multiple Inheritance program. We select a Multiple Inheritance program from opendialogbox window, then by clicking on 'Total Classes' button we got the count 3 it means that there are 3 classes exists in the program. After that we got the name of classes by clicking 'Classes Name', then displaying the class list along with there calculated metrics, then following result can be generated as shown in figure. 5.6.

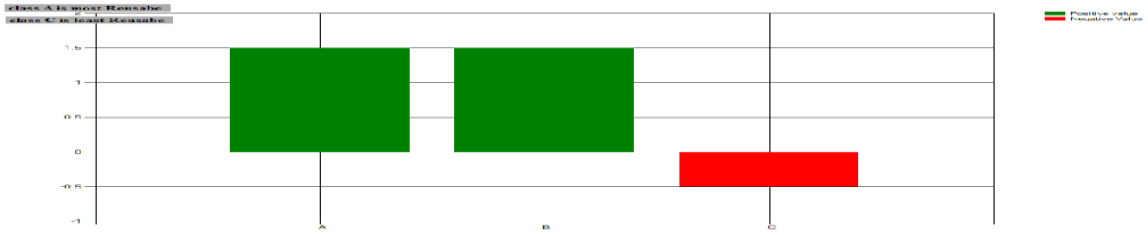

Figure 7 Result of Multiple Inheritance

Above figure shows the result of Multiple Inheritance program selected in Figure 5.5. The class A have highest bar so A class is most reusable and C class shows minimum bar so it is least reusable.

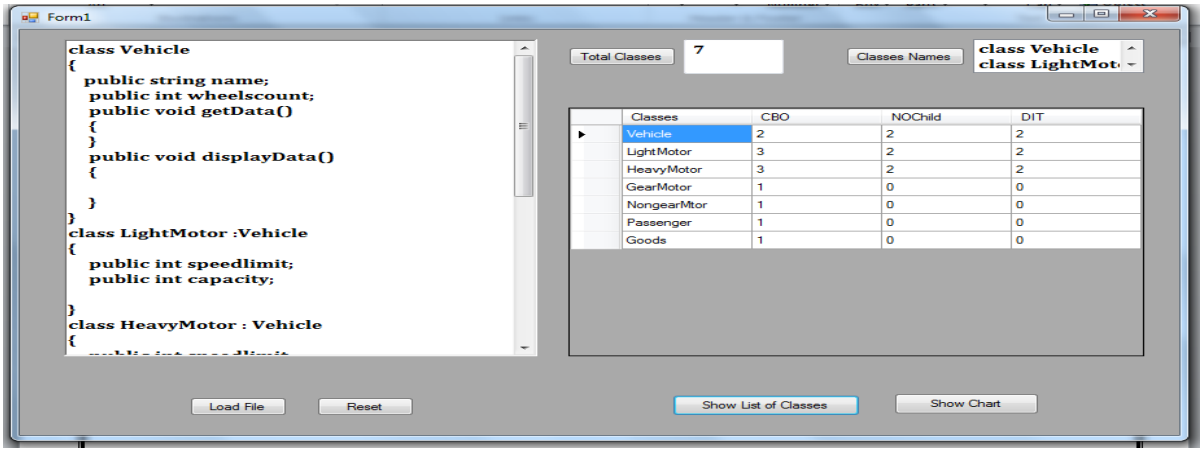

Figure 8 Interface for VehicleClass

The above interface shows the example of a vehicle class program. We select a vehicle class program from opendialogbox window, then by clicking on 'Total Classes' button we got the count 7 it means that there are 7

e-ISSN: 2394-8299
p-ISSN: 2394-8280

classes exists in the program. After that we got the name of classes by clicking 'Classes Name', then displaying the class list along with there calculated metrics, then following result can be generated as shown in figure. 5.8.

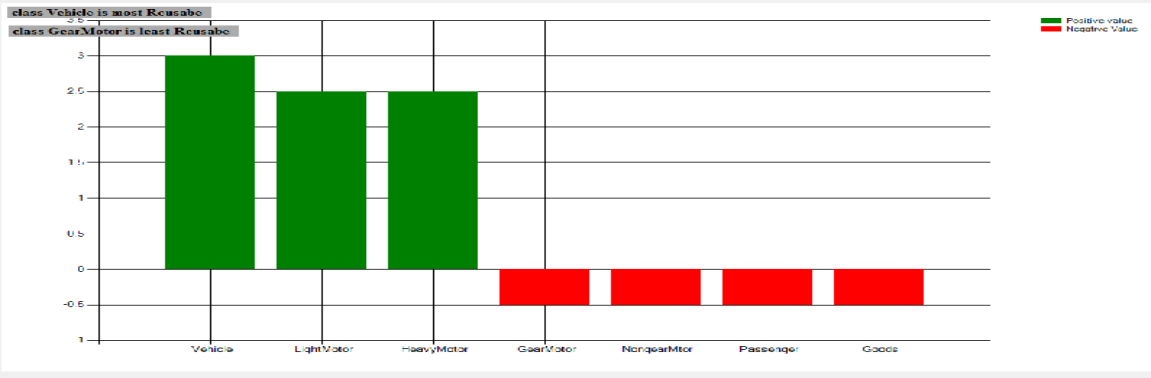

Figure 9 Result of Vehicle Class

Above figure shows the result of Vehicle class selected in Figure 5.7. The Vehicle class have highest bar so Vehicle class is most reusable and Gear motor class shows minimum bar so it is least reusable.

V. 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,