IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A TECHNIQUE TO STUDY SOFTWARE REUSABILITY USING OBJECT ORIENTED MATRICES

Jyotsna Raj*, Deepak Agrawal, Ashish Agrawal
Computer Science & Engineering, Acropolis Institute of Technology & Research, Indore (M.P.)-452003.

ABSTRACT

Reusability is one best direction to increase developing productivity and maintainability of application. We have to first search and focus upon the good tested software component and reusable. Application is being developed by one programmer can be reuse by other programmer. The data, design for the application and code of one project or application can be reused with the projects/applications having the same requirement. The objective of this paper proposed a way for reusable module. An process that takes source code as a input and helped us to take decision whether the particular code of one project can be reuse or not or how much part of the code we can reuse.

KEYWORDS: software reuse, reusability, metrics, CK metrics, cyclomatic complexity.

INTRODUCTION

As in today's era, everyone is interested to increase the productivity and efficiency of the work is being done by the organizations. It is possible only when, we use some concept like reusability. Many of the well known organizations are using the software of reusability and saving the time in software product development. Not only the code that is developed is reusable but the design of the product, requirement gathering for developing the product can also be reuse. Dimension/Software matrices may help us not only to learn how to develop reusable components but it also helps us to identify how much amount of code we can reuse from the existing code. Existing applications that is already have been developed will involve knowledge and experience for particular application domain and it actually fulfils all the needs of the organization. We can develop future applications vary efficiently, if could extract or get some of the reusable modules from those existing applications. That will not only save the time and effort but also helps us to guide the project or applications in more positive direction and also increase the performance of the software applications. One more thing that we can achieve from these things is that we can reduced the cost of the application that we want to develop and provide the good quality product to the end user.

REUSABILITY

Not all code of the application or program is reusable, so for measuring what part of the code is reusable we have an matrices called reusability. The matrices called reusability will able to identify what portion of code can be reuse so that we can save the cost of development and time to develop the application. 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. We can use the concept of reusability in missile system.
b. American Navy utilizes the reusable modules.

As we know that to develop the product we follow the standard phases of software development life cycle. We determined the reusable components from requirement gathering or feasibility stage to last stage i.e. testing 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/ developing phase
b. Requirement/feasibility phase
c. Architecture/design documentation

d. Test plans/ Testing
e. Specifications gathering
f. Design the application

PREVIOUS WORK

Michael a cusumano suggested that reusability also needs to be seen as a *administrative and organizational* problem. In this paper, the discussion was especially on techniques and tools used for the development of quality software's. Which in the mid- 1980s was regularly delivering software systems with nearly 50% reused code. Finally, the paper concluded that Reusability, and the promising reuses approaches, actually fell across a range.

Young Lee suggested a proposed quality model for object oriented software called as Reconfigurable Automated Metrics for Object (RAMOOS). The quality model points the maintainability and reusability aspects of software which can be successfully predicted from the source code. The modal works during the time period of software development.

James M.biemen suggested that derivation for the measurement of software reuse and introduces some definition, attributes and abstractions. In this paper the focus is upon the importance of the standpoint of the observer when analyzing, measuring and profiling reuse using measurement theory. The author has given measurement theory for measuring the concept of reusability. He also compares the measurement theory with the traditional object oriented approach.

Dandashi F. Suggested that how productivity and maintainability play an important role in software quality assurance. the author tried to develop some sample application using some technology and try to find a way that help to enhance the productivity of the software.

Selby 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. 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. 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.

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. Understand ability depending on the structure complexity, documentation level of the programs and size.

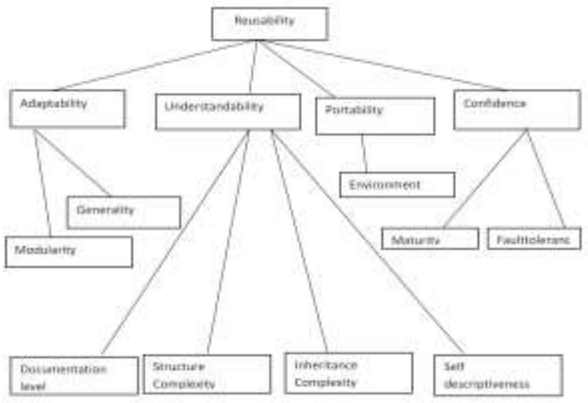

Figure 1 Factor in which Reusability Depends

A. Understand ability:

Understand ability is the level in which the meaning of the software system or module should be authorizing to the developer or user. Understand ability depend along the following element, these are Documentation level, size and complexity. When module are well documented then understand ability 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. Understand ability 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].

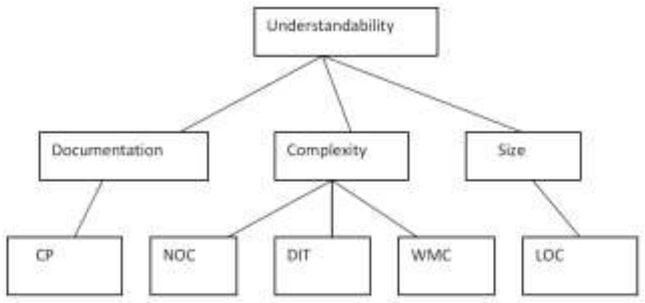

Fig 2 Factor and Metrics in which Understandability depends

- **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 understand ability.

  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.
- *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.

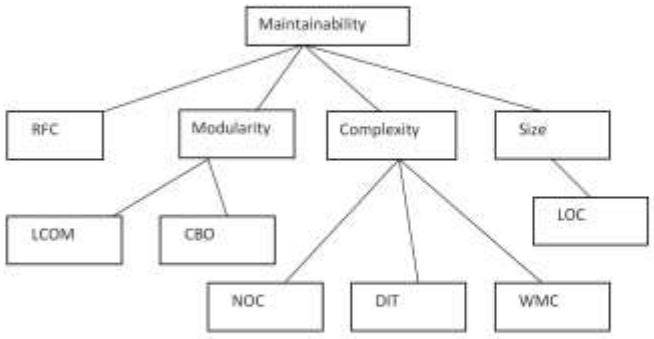

Figure 3 Factor and Metrics in which Maintainability depends

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.

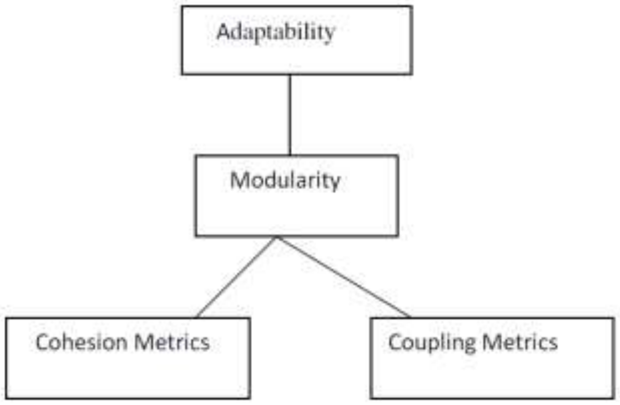

Figure. 4 Factor and Metrics on which Adaptability depends

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].

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.

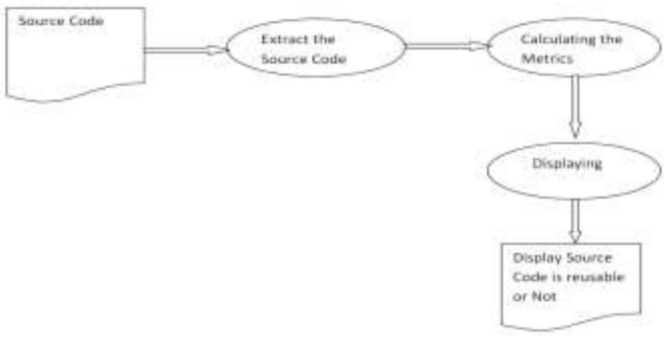

Figure 5 Steps follow for identified reusable module

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.

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,