Volume 5, Issue 6, June 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

An Efficient Tool for Calculating Reusability of Object Oriented Programs

Shruti*
Computer Science,
BITS Bhopal, India

Pankaj Richhariya
Computer Science,
BITS Bhopal, India

Ankit Anand
Computer Science,
ABESIT Ghaziabad, India

Abstract— Software Reuse increases the productivity and it has the positive impact on the quality. The reusability of the software reduces the cost and improves the Quality of the software. Reusability is considered as a one 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 propose for estimating the reusability of module .Reusing the oldest module which is already developed in new system, hence it increase the productivity because the effort of the programmer is reduced and it correctness is also increases because the exits system already tested.

Keywords— Adaptability, cohesion metrics, coupling metrics, maintainability, module, understandability.

I. INTRODUCTION

Introduction

Reusability is basically a method in which a software module or other work product can be used in one or more computing program or software system. It saves a lot of time and cost, if we use reusable component from a existing system to develop a new software. It also increases the performance of the software And Aim of the any software organization is giving the product with good quality and low cost As soon as possible. Unfortunately, working with existing software poses several significant problems due to its inconsistent quality, style, documentation, and design. We need an economical way to find domain knowledge and useful artifacts within these programs.

Problem Definition

In the era of rapid production everyone is interested to increase the productivity and reduce the cost of developing the software and providing the better quality of software. There are several types of quality attribute available from which one can identify the quality of the software. One way to increase the productivity is to reuse the existing module because a lot of effort and time have already spent for developing the old software that is already well tested and designed so one should use this existing software. Problem is that sometimes modules that are not developed for reuse if are reused, leads to high development cost and time. Thus, one should first explore which existing module or component is more suitable for reuse and then try to reuse it.

II. REUSABILITY

Reusability: Reusability can be simply defined as a 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. If anyone uses the reusable component from existing system, it saves a lot of time and effort for developing the software and subsequently reduces cost of the software development also. On the other hand reuse increases the performance of the software, since reusable software components are already tested for quality and optimized for performance, reusing them in verbatim increases the performance significantly. Aim of any software organization is to give the product with good quality at low cost as earlier as possible. So reusability concept is being used to achieve above mentioned goals.

Why Reuses Is Important

By increasing the level of the software reuse it saves the time and development cost taken to develop the software by many organization, such as U.S. Department saved 300 million $ by increasing the 1% software reuse. 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.

What Can Be Reused

In Software development life cycle Reusability concept is not limited to only coding phase. We used it from requirement phase to last stage of the software development. There are various phase in software development where reusability is used: Code, Requirements, Architecture/design, documentation, Test Plans, Specifications, Design, Manuals, Templates, Design decisions.

Definition of reuses Types

Bieman et al's defined the several types of reuse and defined the three types of reuse in the three perspectives. These are explained as follows:

Public Reuse:

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

Public reuse = length (E) / length (p);

E is the code developed externally.

P is the new system including E.

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".

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.

Leveraged Reuse: In Leveraged reuse means modifications of reuse is allowed.

Verbatim Reuse: In Verbatim reuse means modifications of reuse is not allowed.

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

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 92 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.

III. ARCHITECTURE USED

Proposed system uses plug-in architecture. This Architecture is providing the flexibility. If we want to add new metrics, we can directly add them without modifying the core modules and can use the functionalities of the new modules. The architectural elements are described below followed by description of each component specified:

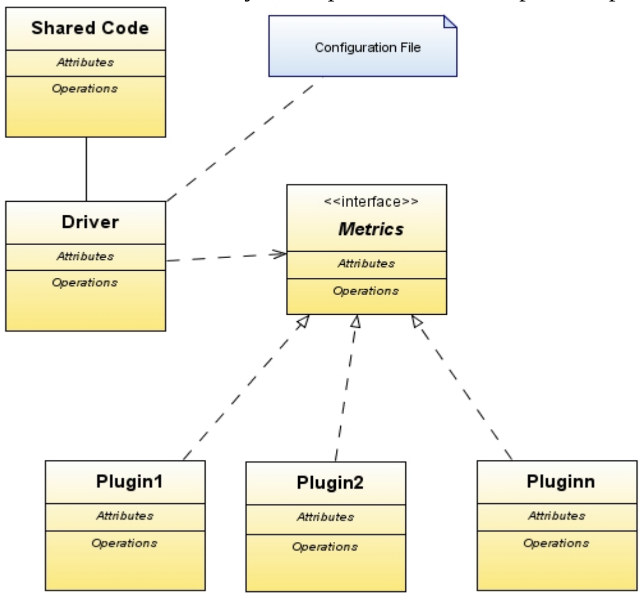


Figure 1 Architecture of the System

IV. METHODOLOGY

We use plug-in Architecture for developing the system. This Architecture is providing flexibility**.** Source code would be the input for the tool. By analysing this source code statically we will extract the information about modules. After extracting useful information from the static analysis the tool will automatically calculate reusability metrics. Based on the value of these metrics, one can decide about the reusability of the object oriented module. Reusability is depending on the portability, adaptability, understandability, maintainability and reliability. Here we are dealing with the java program that way portability is not issue for us. And we are dealing with static code therefore we are not considering the reliability as an affect which affects the reusability, because reliability is measured in terms of mean time and fault which is measured during the execution of the program. Understandability depends on the complexity, size and documentation level of the programs. Complexity is of two types' structure complexity and inheritance complexity.

Coupling Metrics

Coupling can occur if

i. A method of one class is invoked from another class
ii. An attribute of one class is modified / used by a method of another class
iii. 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. 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

$$\text{Class Coup} = \sum \text{Coup (i,j)} / (\text{m*m-m});$$

Cohesion Metrics

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. 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

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

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

$$\text{Re}usability = 0.25M + 0.25I + 0.25D + 0.25C$$

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

Approach for Identification of Reusable Module

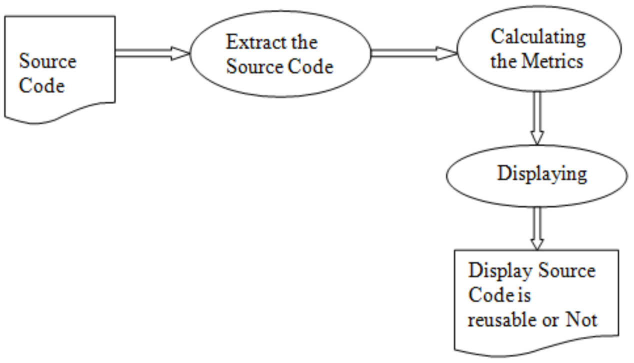
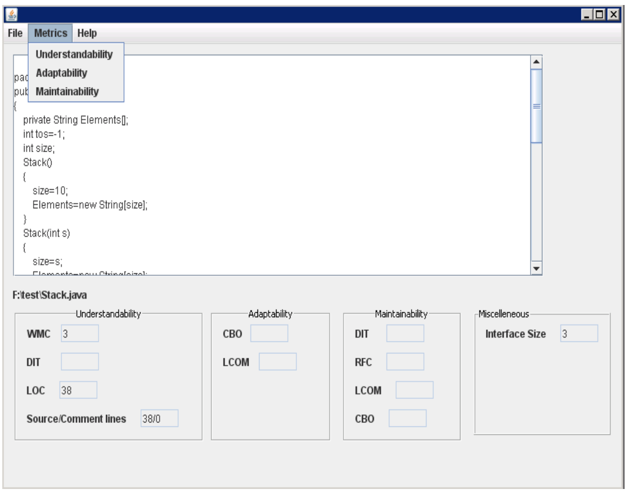


Figure 2: Step follows for identified the reusable module

GUI of the System

From File menu we load the file (whose reusability user want to calculates), we will provide the facility to the user of the tool, to specify an aspect of reusability i.e. (understandability, adaptability, maintainability other quality characteristics that would be useful) and metrics corresponding to this aspect will be calculated and displayed to the user.



```
pac
pub
{
  private String Elements[];
  int tos=-1;
  int size;
  Stack()
  {
    size=10;
    Elements=new String[size];
  }
  Stack(int s)
  {
    size=s;
```

F:\test\Stack.java

Understandability		Adaptability		Maintainability		Miscelleneous	
WMC	3	CBO		DIT		Interface Size	3
DIT		LCOM		RFC			
LOC	38			LCOM			
Source/Comment lines	38/0			CBO			

Figure 3: GUI of the System

V. RESULTS

We have taken some system as case and run our tool on the given code. Following systems were taken for test:
Sugr (Static UML generation from Requirements)
Code Generation using UML and DFD
Following figures show the result of experiment:

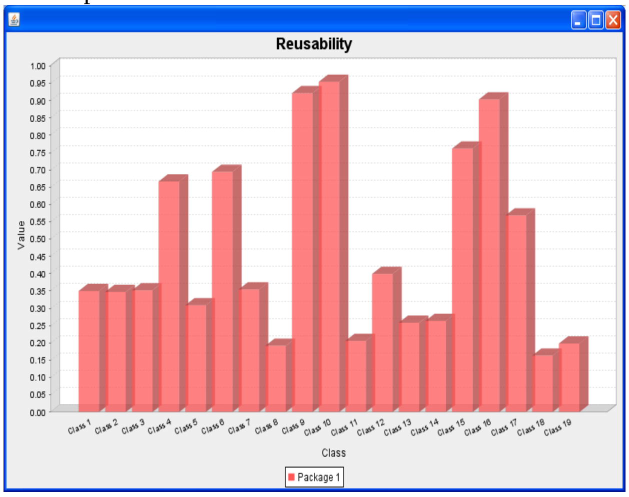
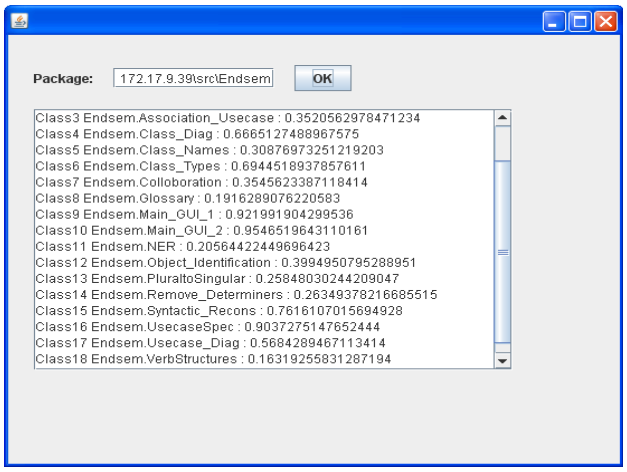


Figure 3: Result from endsem package of SUGR tool



```
Class3 Endsem.Association_Usecase : 0.3520562978471234
Class4 Endsem.Class_Diag : 0.6665127488967575
Class5 Endsem.Class_Names : 0.30876973251219203
Class6 Endsem.Class_Types : 0.6944518937857611
Class7 Endsem.Colloboration : 0.3545623387118414
Class8 Endsem.Glossary : 0.1916289076220583
Class9 Endsem.Main_GUI_1 : 0.921991904299536
Class10 Endsem.Main_GUI_2 : 0.9546519643110161
Class11 Endsem.NER : 0.20564422449696423
Class12 Endsem.Object_Identification : 0.3994950795288951
Class13 Endsem.PluraltoSingular : 0.25848030244209047
Class14 Endsem.Remove_Determiners : 0.26349378216685515
Class15 Endsem.Syntactic_Recons : 0.7616107015694928
Class16 Endsem.UsecaseSpec : 0.9037275147652444
Class17 Endsem.Usecase_Diag : 0.5684289467113414
Class18 Endsem.VerbStructures : 0.16319255831287194
```

Figure 4: Reusability Values

The sugr Package: endsem contains 19 classes some of them are GUI classes and some contain business logic. The highest reusability recorded in the package was 0.954.
The Package: codegen contains 11 classes some of them are GUI classes and some contain business logic. The highest reusability recorded in the package was 1.0.

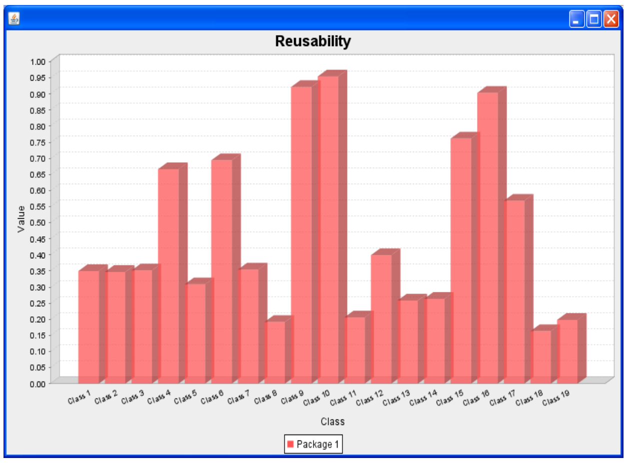
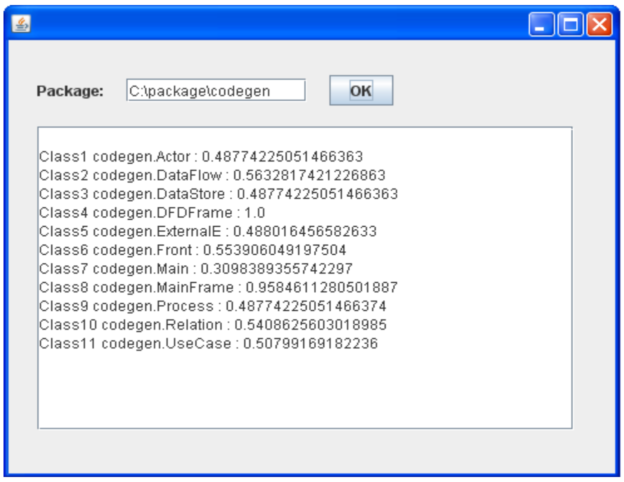


Figure 5: Reusability for codegen package



```
Class1 codegen.Actor : 0.48774225051466363
Class2 codegen.DataFlow : 0.5632817421226863
Class3 codegen.DataStore : 0.48774225051466363
Class4 codegen.DFDFrame : 1.0
Class5 codegen.ExternalE : 0.488016456582633
Class6 codegen.Front : 0.553906049197504
Class7 codegen.Main : 0.3098389355742297
Class8 codegen.MainFrame : 0.9584611280501887
Class9 codegen.Process : 0.48774225051466374
Class10 codegen.Relation : 0.5408625603018985
Class11 codegen.UseCase : 0.50799169182236
```

Figure 6: Reusability values for codegen package

VI. 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 thesis can be effectively used to calculate the reusability of any object oriented software module.

REFERENCE

[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
[10] [Bieman92] James M. Bieman "*Deriving Measures of Software Reuse in Object Oriented Systems"* Springer-Verlag 1992 pp 79-82.