

Formation Metrics of Object Oriented Reusability Using Commutative and Associative Properties

P. A. Selvaraj

*Associate Professor,
Department of Computer Applications,
Kongu Engineering College, Erode, Tamilnadu
kps.raj2001@gmail.com*

Dr. P. Thangaraj

*Professor and Head,
Department of Computer Science and Engineering,
Bannari Amman Institute of Technology, Sathyamangalam, Tamilnadu
ctptr@yahoo.co.in*

Abstract

Software Engineering is the basement of the various software development processes. In that, reusability is the backbone. Modern software developers utilize the various features of the object orientation for the product development. Object Orientation makes the software development process easier by using the real world objects and the relationships between the attributes which makes the development easier. This paper deals with the impact of the object orientation along with code reusability in the process of software development. The identification of the reusable component module from the existing system and analyze whether the module is suitable for the future system by using the properties such as association among the modules, similarity between the attributes and parameters, etc., The analysis of the various modules available in the projects are made and utilize it for future software product development along with agile technologies.

Keywords: Software engineering, Object Oriented Software Engineering, Software Project, Line of code (LOC), Reusability

Introduction

Software engineering is the process of developing the required software based on the need given by the user. It consists of sequence of steps to be followed in order to get the desired product. Software development processes normally focus on keep away from errors, identifying and correcting software faults that do occur, and predicting

software reliability after development [1]. In that, reusability along with the object oriented technique plays a major role in the development of the software product. The steps involved are analysis, design, implementation and finally the testing [2].

Object oriented software development is the analysis of the identification of the attributes and its functions using the real world entities which makes the development process easier [3][5]. Reusability is the resulting solution from the object oriented technique. The process of identifying the reusable components is with the help of pattern matching and similarity of function available in the system which helps to reuse the function from one project to another project. Reusability measurement is allowing for identify the reusable modules from being program and way to build along the increasing level of software reuse, development cost taken to develop the software and save the cost and time.

Programmers some time use redundant declarations without sufficient knowledge of programming language. In such cases, a compiler may be unable to check consistency between declarations and their uses. In order to avoid these multiple declarations in a system, the components which are reusable are applied in different situations so as to remove the redundant features in programming languages. The reusable components can also be termed “software clones” which are used for the purpose of multiple reuses to reduce development time [4].

Motivation and Background Issues

Reusability is only one best way to increase productivity of any software industry and it also increase the maintainability of any software. One must first look for good tested and reusable software component. Developed Application software by one programmer can be helpful for other programmers and components also useful [6]. This is often proving that code specifics to application requirements can be also reused in projects having same requirements. The primary aim of this paper suggested a way for reusable module. A procedures that takes program code as an input that will help to take the decision approximately which particular software code, reusable artifacts should be reused or not.

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

- 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.
- Complexity of interface: Complicated interface make reuse difficult.
- Class size: when the size of the class is large then it is difficult to understand and difficult to reuse.
- 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. 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. Recognizing the reusability depends on the structure complexity, level of the programs and size.

Recognition of the Reusability of the modules depends on the complexity involved in the development process of the software. The structure involves the relationship between the attributes and the functions involved in the system and the dependencies between the functions.

The program size i.e., lines of code (LOC) plays the main role in predicting the dependencies among the functions. This analysis is intended to identify risk in the perspective of a programmer's knowledge.. Every developer has his own technique to develop a program. The style of program writing of a programmer is entirely different from others. Idea of choosing the best level of programming among a crew of programming techniques based on three levels of program writing [7]. They are:

1. Low level program - Low level program involves simple programming with limited number of variables. Example: simple mathematical calculations.
2. Medium level program - Medium level Program consists of variables at multilevel systems. Example: Use of Inheritance to access the variable from base class.
3. High level program - High level Program involves complex data variables in the system which depicts the importance of the variable and its scope within the system. Example: Functions used for online quiz system.

The 'level' reveals the complexity levels of the program and not the types of software. The calculation metrics such as Line of Code (LOC), Program Volume, Compilation Detail and Error are used for analyzing the skill levels of a program. They are detailed as below.

LOC (N): LOC represents Line of Code. LOC is the total number of lines present in a program. If the LOC is large, then it is generally assumed that the complexity of the program is also large.

PROGRAM VOLUME (V): Program volume can be represented below.

$$V=N \log n$$

N - Total number of lines present in a program

n - Sum of the total number of operators and operands present in a program.

If program volume is larger, it seems to have much complexity in reusing the code because there may be many operators and operands.

Revolution of Object Oriented Reusability

The object-oriented programming (OOP) paradigm [8] eases the development of reusable software components by providing the programmer encapsulation, polymorphism, and hierarchy. These techniques make producing a single reusable component easier, allow easy customization of existing components, and allow the creation of components which are more powerful since they are more generic.

The techniques important for reuse identified in this section can serve as the foundation for the design of an object oriented specification language that supports code reuse.

Member functions in object oriented languages, and functions in general in certain sequential programming languages, may be polymorphic. The degree of polymorphism allowed varies, but there are three general flavors: parametric polymorphism, function overloading, and subtyping. These features significantly improve the reuse of an object by easily allowing it to be extended beyond its initial applications.

Despite the success of the object-oriented programming in producing reusable code, certain kinds of reuse patterns are still hard to achieve. The inheritance mechanism allows a user to augment the behavior of an object orientation. For example adding logging behavior to an object-oriented system would typically involve modifying every object in the system. Object-Oriented Programming (OOP) languages are designed to facilitate component based reuse in traditional programming languages.

It also identifies reusable components and risks associated with them, before any transformation of components to new proposed projects. The following issues need to be considered for analysis:

- Check whether the user requirement in the earlier project is in the shape of reusable component with suitable version.
- Is there any weakness or possible failures in that component?
- Which inputs or situations may cause the failure?
- What is the impact of such failure?
- Ask questions repeatedly to identify the risk.
- Analyze the Domain and sub domain expertise before taking the reusable components from the earlier system.

These various sources of information are required to identify the risks and must be documented by the developer about the system.

The parameters are to be considered during the comparison of arguments that are passed to any function and their types. They include the number of arguments for the function and the number of variables used by the function. The data type of the parameters in the function should be the same. However the names of the parameters could be different. Functions should be checked with various parameters such as 'return' type, Variable Declaration, Flow of control, Function argument type, etc.

Mathematical Representation of The Reusable System

A Function from a set A to a set B is a relation from A into B such that each element of A is related exactly to one element of set B. The set A is called the domain of the function and the set B is called the co-domain [9].

One-to-One Function Representation:

Let function 'f' from A to A is called One to One function if different elements of A are assigned to different elements of B. In this system, one can compare two similar

systems, which perform similar functionality and the presence of code clones is identified. Consider the following two systems which consist of functions that perform similar operations.

Example

<pre>int sum=0 void foo(Iterator iter) { for(item=first(iter)has (iter)item= next(iter)) { sum=sum+value(item) }</pre>	more	<pre>int bar(Iterator iter) { int sum=0 for(item=first(iter)has more(iter)item=next(iter)) { sum=sum+value(item) }}</pre>
--	------	---

Commutative Property (one to one function):

A function $f: X \times X \rightarrow X$ is said to be commutative if for every $x, y \in X$, $f(x, y) = f(y, x)$ which satisfies to be a commutative, then it is called effective level, $f(x, y) \neq f(y, x)$. If it does not satisfy the commutative property, then it is said to be a failure level.

On-to Function Representation:

Let 'f' be a function of A to B. If $f(A) = B$, (i.e.) if every member of B has the pre image of A, then 'f' is On-to. Consider the example below which contains the variables which are mapped to other variables.

Example

<pre>int sum=0 void add(int a, int b,int c) { sum=a+b+c cout<<sum }</pre>	<pre>void add(int a, int b,int c) { int sum=0 int d=a+b sum=c+d cout<<sum }</pre>
---	---

Associative Property (onto function): A function $f: X \times X \rightarrow X$ is said to be associative, if for every $x, y, z \in X$, $f(f(x, y), z) = f(x, f(y, z))$ it should satisfy the associative property. If an effective level, $f(f(x, y), z) \neq f(x, f(y, z))$, does not satisfy the associative property and it is said to be a failure level.

In the above example, three variables of a function are considered which are mapped to two arguments of another function which performs a similar operation.

Result and Discussion

The use of the reusable components is depicted with the help of the modular function and its usage. A Function is created for the purpose of newly proposed project and this function can be integrated with an existing project. The steps involved in the process for determining the reusability are narrated below.

Sequence of Steps For The Solutions:

- Step 1: Find out the functions in the new projects that may be adapted with reusable functions of the old projects. (Table 5.1)
- Step 2: Identify reusable functions of the old projects for code reusability. (Table 5.2)
- Step 3: Segregate functions depending upon applications.
- Step4: Group task dependent function components for appropriate adaptability.

Various Function selections that are taken from software projects can be checked to find whether they have exact matches or parameter matches for reusability function components of the new software. Software bugs can be minimized with reusable components and is applicable to all similar systems. The bugs may lead to various possible upcoming errors in the future. Reusable Software codes are also helpful in analyzing the various possible functionalities of projects shown in the table 5.1 and 5.2 respectively. Analyzation of packages results in the reuse of various existing functions of a system. Errors can be greatly reduced by using reusable software codes. The usage levels of the various reusable components in the projects are shown in Figure 5.1.

Table 5.1: Testing Factors of Reusability Levels

Test Factors	Original Function Level	Source file	Duplicate Function Level	Usage Level
Similar Property	Function (int,int)	Swap.cpp(swap two integer)	Function (int, int)	1
		Swap1.cpp(swap the integer and float)	Function (int, float)	0
		Swap2.cpp(swap the two float)	Function(float,float)	1
		Swap3.cpp(swap the float and integer)	Function (float, int)	0
Linking	Function (a,b)	Add.cpp	Function (a,b)	1
		Add.cpp	Function (i,j)	1
Compilation	Zero Error	Source file	Zero Error	1
		Source file	Error Occurrence	0

0 – Minimum 1 – Maximum

Table 5.2: Reusability Module Selection form Old Projects

Project Selection	Function Selections	Predefine Function Display		Reusable Modules
projects\ railway.cpp	void pnr() void helpline() void chart()	equal_to	void equal_to()	No
		divides	void divided()	Yes
		greater_than	void greater_equal()	Yes
		greater	void greater()	Yes
		getch	void getche()	Yes
		println	void println()	Yes
projects\ banking.cpp	void display_account() void transaction()	equal_to	void equal_to()	Yes
		divides	void divided()	Yes
		greater	void greater()	Yes
		Getch()	Void getch()	No
projects\ departmental. cpp	void bill() void password() void date() void time()	equal_to	void equal_to()	Yes
		divides	void divided()	Yes
		Print	Void print()	Yes

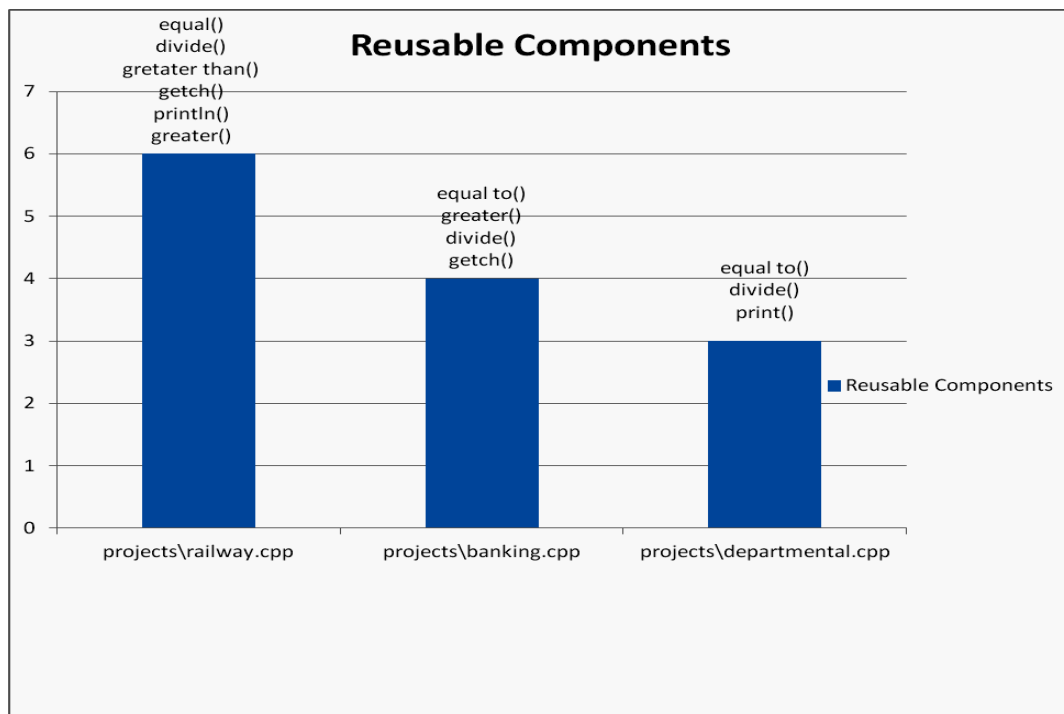


Figure 5.1: Usage Level of the Reusable Components in the Projects

References

- [1] Aman Kumar Sharma, Arvind Kalia, Hardeep Singh, “Metrics identification for measuring object oriented software quality”, International journal of soft computing and engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-5,2012.
- [2] Amandeep kaur et al., “Empirical analysis of CK & MOOD metric suit”, International journal of innovation, management and technology, Vol. 1, No. 5, ISSN: 2010-0248,2010.
- [3] Elnaz Namazi, Mahdi Bazargani, “A Complexity Measure Based Object-Oriented Software Metrics”, ARPN Journal of Systems and Software, VOL. 4, NO. 5, August 2014.
- [4] Laura Carvajal, Ana M. Moreno, Mari´a-Isabel Sa´nchez-Segura, and Ahmed Seffah, “Usability through Software Design”, IEEE Transactions on Software Engineering, Vol. 39, No. 11, November 2013.
- [5] Meryem Lamrani, Younes El Amrani, Aziz Ettouhami, “A formal definition of metrics for object oriented design: MOOD metrics”, University Mohammed V Agdal, Computer science department PB 1014 Rabat, Morocco journal of theoretical and applied information technology, Vol. 49 No.1,2013.
- [6] Nagesh Paliwal , Vivek Shrivastava , Ketki Tiwari, “An Approach to Find Reusability of Software Using Object Oriented Metrics”, International Journal of Innovative Research in Science, Engineering and Technology Vol. 3, Issue 3, March 2014.
- [7] Roger s. pressman, “Software engineering a practitioner’s approach”, Seventh edition, Newyork, Tata Mc Graw Hill, 2010.
- [8] Shweta Srivastava, Dr.Majhar Khaliq, “Object Oriented Design for Testability: A Systematic Review”, International Journal of Advance Research in Computer Science and Management Studies, Volume 2, Issue 10, October 2014.
- [9] Trembley J.P, Manohar.R, “Discrete Mathematical Structures with Applications to Computer Science”, Tata Mc Graw Hill, 1988.