Volume 2, No. 6, Nov-Dec 2011
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

An Approach for Reusability Measurement of Object-Oriented Code Based Upon CK Metrics

Vicky Bumbak*, Pradeep Kumar Bhatia
M.TECH CSE, Associate Professor, DEPT. OF CSE
G. J. University of science and technology,
Hisar ,India
Vicky5835@gmail.com, pk_bhatia2002@yahoo.com

Abstract – This study focus on a set of object-oriented metrics that can be used to measure the reusability of object-oriented code. This research address a new model set of metrics for object-oriented code. Reusability is the process to reduce the cost and improve the quality of software. Reusability may be achieved by concept of Generic Programming through C++ Templates. There are various techniques to measuring reusability of object-oriented code. Chaidmber and Kemere (CK) metrics are proposed for measuring reusability of object-oriented code. This paper introduces new model for reusability measurement of object-oriented code with the help of metrics Number of Template Children (NTC), Depth of Template Tree (DTT), Method Template Inheritance Factor (MTIF), and Attribute Template Inheritance Factor (ATIF).

Keywords: Object-Oriented code, Reusability, CK metrics, Generic Programming, Templates.

I. INTRODUCTION

It is widely accepted that object- oriented development requires a different way of thinking than traditional structured development [1] and software projects are shifting to object oriented design. The main advantage of object oriented design is its modularity and reusability. Object-oriented metrics are used to measure properties of object oriented designs [2]. Method reflects how a problem is broken into segments and the capabilities other classes except of a given class. There are very few metrics in the literature for measuring the complexity of object-Oriented. Proposed Metric for an Object- Oriented code, which are capable to evaluate the System complexity of operation in method. Most of the metrics for object-oriented design [3].

Recent work in the field has also addressed the need for research to better understand the determinants of software quality and other project outcomes such as productivity and cycle-time in OO software development [4]. There are two approaches for reuse of code: develop the reusable code from scratch or identify and extract the reusable code from already developed code. The organizations that has experience in developing software, but not yet used the software reuse concept, there exists extra cost to develop the reusable components from scratch to build and strengthen their reusable software reservoir [5]. A software metric is a measure of some property of a software artifact or its specification. One of the most difficult topics in software engineering is the assessment of the non-functional parameters of a system. Performance, maintainability, reusability, security, adaptability, etc., are examples of nonfunctional requirements. While some of the above mentioned characteristics of a software artifact, like performance and security, can be measured without investing important effort, assessing the adaptability and the reusability is a tedious task [4]. OO software settings. In addition, many commercial tools are now available to automatically collect some or all of these metrics (see the sidebar, "Commercial Tools for OO Metrics Collection"). A variety of studies have documented relationships between OO metrics and managerial-performance variables including effort, reusability, defects and faults, maintainability, and cost savings [6]. In this paper, set of two metrics is proposed to measure amount of reusability included in the form of templates by the designer. These metrics are then analytically analyzed against CK metrics proposed set of six axioms. Standard projects are used for application of these metrics and suggest the ways in which project managers can use these metrics. Further, the amount of lines of code (LOC) reduced in projects using templates is also shown. The paper is organized gives introduction to Generic Programming with Templates [7].

A. Object Oriented Design

Object-oriented design is concerned with developing an object-oriented module of a software system to apply the identified requirements. Designer will use OOD because it is a faster development process, module based architecture, contains high reusable features, increases design quality and so on. "Object-oriented design is a method of design encompassing the process of object-oriented decomposing and a notation for depicting both logical and physical as well as static and dynamic models of the system under design" [8]. Objects are the basic units of object oriented design. Identity, states and behaviors are the main characteristics of any object. A class is a collection of objects which have common behaviors. "A class represents a template for several objects and describes how these objects are structured internally. Objects of the same class have the same definition both for their operation and for their information structure" [9]. There are several essential themes in object oriented design. These themes are mostly support object oriented design in the context of measuring. These are discussing in next sub section.

B. CK Metrics Suite

Metrics set proposed by Chidamber and Kemerer contains six OO design metrics. These metrics are based on Bunge's ontology as the theoretical basis and analytically evaluated against Weyuker's measurement principles. All

CONFERENCE PAPER
International Conference on Issues & Challenges in Networking, Intelligence & Computing Technologies
Organized by Krishna Institute of Engineering and Technology (KIET) Ghaziabad, India

these six metrics captures the concept of inheritance, coupling and cohesion [10].

a. ***Weighted Method per Class (WMC) assesses*** complexity of a class through aggregating a complexity measure of its methods. Complexity of a class can for example be calculated by the cyclomatic complexities of its methods. High value of WMC indicates the class is more complex than that of low values. Consider a class Ci, with methods M1….Mn that are defined in the class. Let C1….Cn be the complexity of the methods [11]. Then

$$WMC = \sum_{i=1}^{n} Ci$$

b. ***Depth of Inheritance Tree (DIT)*** Inheritance is when a class shares the behaviour of another class. When a subclass inherits from one super class then it is called as single inheritance and when a subclass inherits from more than one upper class then it is called as multiple inheritance. Depth of inheritance of the class is the DIT metric for the class. In cases involving multiple inheritances, the DIT will be the maximum length from the node to the root of the tree [10].

c. ***Number of Children (NOC)*** This metric measures how many sub-classes are going to inherit the methods of the parent class. (NOC) of a class is the number of immediate subclasses subordinated to a class in the class hierarchy. Since inheritance is a measure of reuse, the reuse is proportional to NOC [4].

d. ***Coupling Between Object (CBO)*** The idea of this metrics is that an object is coupled to another object if two object act upon each other. A class is coupled with another if the methods of one class use the methods or attributes of the other class. An increase of CBO indicates the reusability of a class will decrease. Thus, the CBO values for each class should be kept as low as possible.

e. ***Response for a Class (RFC)*** (RFC) is the count of all the methods that can be invoked in response to a message to an object of the class or to some method in the class. The larger the number of methods that can be invoked from a class through messages, the greater the complexity of the class [4].

f. ***Lack of Cohesion of Methods (LCOM)*** Number of method pairs whose similarity is 0 minus the count of method pairs whose similarity is not zero. The larger the number of similar methods in a class the more cohesive the class is. Cohesiveness of methods within a class is desirable, since it promotes encapsulation and lack of cohesion implies classes should probably be split into two or more subclasses.

II. GENERIC PROGRAMMING WITH TEMPLATES

Generic programming focuses on representing families of domain concepts. There is no universally accepted definition for generic programming. We refer to generic programming as the ability to write reusable, independent programming units that can be plugged together by writing glue code [13]. "Generic programming is in some ways more flexible than object-oriented programming. In particular, it does not depend on hierarchies. For example, there is no hierarchical relationship between an int and a string." An important feature of C++ called templates strengthens this benefit of object-oriented programming. Templates are very useful when implementing generic constructs like vectors, stacks, lists, queues, which can be used with different data types. Templates can be classified into two types: Class Templates and Function Templates [7].

A. Function Templates:

Function templates are special functions that can operate with *generic types*. This allows us to create a function template whose functionality can be adapted to more than one type or class without repeating the entire code for each type.

In C++ this can be achieved using *template parameters*. A template parameter is a special kind of parameter that can be used to pass a type as argument: just like regular function parameters can be used to pass values to a function, template parameters allow to pass also types to a function. These function templates can use these parameters as if they were any other regular type.

```
// addition as a template function.
template <type name T>
T plus (T arg1, T arg2) {
Return (arg1 + arg2);
}
// sample usage
int i = plus (10, 20);
float fval = 20.0f;
float f = plus (10.0f, fval);
```

Figure 1: Source Code for Function Templates.

B. Class Templates:

The limitation of classes to hold objects of any particular data type can be overcome

```
template < class M1, class M2, class M3>
class sample
M1 w;
M2 y;
M3 z;
……..
/* when objects of templates class are created using the
following statements Sample <int, float, char>s */
/* the compiler creates the following class sample with three
data members one is of int type, second is float type and third
is of char type */
Class sample
 {
Int w;
float y;
char z;
……..  };
```

Figure 2: Source Code for Class Templates

by declaring that class as class templates. Thus classes, which differ only often there is a need for functions, which have to be used frequently with different data types. The limitation of such functions is that they operate only on a particular data type, which can be overcome by using function template or generic function [7].

III. EXISTING METRICS

A. Number of Template Children (NTC):

The metric NTC can be defined as number of immediate sub-classes of a template class.

```
Template < class T>
Class M1
{
……
};
Template <class S>
Class M2: public M1 <S>
{
…….
};
Class M3
{
….
};
```

Figure 3: Source Code for calculating metric NTC

In this example there is one class M2 which inherits from a template class M1 therefore Number of Template Children (NTC) is 1. The more the value of metric Number of Children (NTC), more reusable software components are include in the projects.

B. Depth of Template Tree (DTT):

The metric DTT can be defined as maximum inheritance path from the class to the root template class.

```
Template <class T>
Class M1
{…}
Class M2: public M1
{….};
Class M3: public M2
{….};
Class M4
{….};
```

Figure 4: Source Code for calculating metric DTT

In this example class M2 inherits from class M1 and class M3 inherits from class M2 thus if we start the root node at level 0 the Depth of Template Tree (DTT) will be 2. The greater the metric Depth of Template Tree (DTT) value greater is the reusability since generic programming is form of reuse.

C. Method Template Inheritance Factor (MTIF):

MTIF is defined as the ratio of the sum of the methods inherited from template classes of the system under consideration to the total number of available methods for all classes.

$$MTIF = \frac{\sum_{i=1}^{n} Wt(Ci)}{\sum_{i=1}^{n} Wa(Ci)} \times NO$$

n = Total number of classes
NO = Number of Objects of Template Classes
WiCi = Number of methods declared in class i
WtCi = Number of the methods inherited form template class i
Wa(Ci) WiCi + WtCi Total no of methods invoked

```
Template <class T>
Class M1
{
T large (Ta, Tb)
{
….
}
};
Class M2: public M1<S>
{
S sum (S c, S d)
{
…..
}
};
```

Figure 5: Source Code for calculating metric MTIF

No of template inherited method = 0+1=1
No of methods declared in each class =1+1=2
Total=3
If we create two objects of class M2
MTIF= (1*2)/3=0.6

The greater the value of metric method Template Inheritance Factor (MTIF) will result in the increased code reusability.

D. Attribute Template Inheritance Factor (ATIF):-

ATIF is defined as the ratio of the sum of attributes inherited from template classes of the system under consideration to the total number of available attributes for all classes.

$$ATIF = \frac{\sum_{i=1}^{n} Xt(Ci)}{\sum_{i=1}^{n} Xa(Ci)} \times NO$$

n = Total number of classes
NO = Number of Objects of Template Classes
XiCi = Number of attributes declared in class i

XtCi = Number of the attributes inherited form template class i
Xa(Ci) XiCi + XtCi Total no of methods invoked

```
Template <class T>
Class M1
{
T a;
T b;
};
Template <class S>
Class M2: public M1<s>
{
S c;
S d;
};
```

Figure 6: Source Code for calculating metric ATIF

No of template inherited attributes = 0+2=2
No of attributes declared in each class=2+2=4
Total =6
If we create two objects in class M2
ATIF= (2*2)/6=0.6
The more the value of metric Attribute Template Inheritance Factor (ATIF) more will be the code reusability.

IV. PURPOSED MODEL

Reusability metric (RM) = NTC + DTT + MTIF + ATIF

V. CONCLUSIONS

Reuse not only saves time and cost, if done correctly, can also improve the quality of software, making it more maintainable. This paper has presented a reusability metrics for object-oriented code. The metrics is based on Chidamber and Kemerer (CK metrics) taken from well used and accepted object-oriented code. The reusability metric can be used to provide a value of reusability for Generic Programming through C++ Templates. This study focus on a set of object-oriented metrics that can be used to measure the quality of an object-oriented code. Object-oriented metrics lead to a number of inherent benefits that provides advantages at both the management and technical level. Purposed metrics presented in this paper have been found to be very useful to find the extent of reusability included in the code in the form of class and function templates.

Reusability metric (RM) purposed in this paper helps us to quantative measurement of reusability of object-oriented code. More value and RM indicates more reusability adopted in the object-oriented code.

V. REFERENCES

[1]. Hironori Washizaki, Hirokazu Yamamoto_and Yoshiaki Fukazawa "A Metrics Suite for Measuring Reusability of Software Components" in proceedings of the ninth international software metrics symposium, pp. 211,2003

[2]. Bellin, D.Manish Tyagi, Maurice Tyler: "Object- Oriented Metrics: An Overview" In Proceedings of the 1994 conference of the Centre for Advanced Studies on Collaborative research,1994

[3]. Sanjay Misra "An Object Oriented Complexity Metric Based on Cognitive Weights" in proceedings of 6th international conference on cognitive informatics, pp. 134-139, 2007

[4]. Octavian Paul ROTARU Marian DOBRE "Reusability Metrics for Software Components" in the 3rd ACS/IEEE International Conference on Computer Systems and Applications, pp. 24, 2005

[5]. Parvinder S. Sandhu, Harpreet Kaur, and Amanpreet Singh "Modeling of Reusability of Object Oriented Software System" World Academy of Science, Engineering and Technology, pp. 162-165, 2009

[6]. David P. Darcy and Chris F. Kemerer Nancy Eickel mann Motorola nancy "OO Metrics in Practice" in Software, IEEE, pp.17-19, nov.-dec.2005.

[7]. K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra "Software Reuse Metrics for Object-Oriented Systems" in proceedings of the third ACIS international conference on software engineering research, management and application, pp. 48-54, 2005

[8]. G. Booch . "Object-Oriented Analysis and Design with Applications" 2nd ed, ISBN 0-8053-5340-2, 1998

[9]. Jacobson, I., Christeerson, M., Jonsson, P.and Overgaard G: "Object- Oriented Software Engineering: A Uses-Case Driven Approach", ISBN-10: 0201544350, 1992

[10]. Mr. U. L. Kulkarni, Mr. Y. R. Arde, Ms. Vrushali G. "Validation of CK metrics for Object Oriented Design Measurement" in 3rd International Conference on Emerging Trends in Engineering and Technology (ICETET), pp.646-651, 2010

[11]. Shyam. R. Chidamber , Chris. F. Kemerer, "A Metrics Suite for Object Oriented Design, IEEE Transactions on Software Engineering" in IEEE Transactions on Software Engineering , Vol .20 , pp. 476 - 493, June 1994.

[12]. C. Neelamegam, Dr. M. Punithavalli "A Survey - Object Oriented Quality Metrics" global jourbal of computer science and technology, pp. 183-186, 2009

[13]. http://www.joyofprogramming.com

CONFERENCE PAPER
International Conference on Issues & Challenges in Networking, Intelligence & Computing Technologies
Organized by Krishna Institute of Engineering and Technology (KIET) Ghaziabad, India