Analysis of Reusability of Object-Oriented System using CK Metrics

Brij Mohan Goel
Research Scholar,
Deptt. of CSE
SGVU, Jaipur-302025, India

Pradeep Kumar Bhatia
Deptt. of CSE.,
G J University of Science & Technology,
Hisar-125001 (Haryana), India

ABSTRACT

In the object-oriented environment, one of the most important aspects having strong influence on the quality of resulting software system is the design complexity. The OO model offers the technology to create components that can be used for general programming. Design complexity has been imagining to play a strong role in the quality of the resulting software system in OO development environments. This paper gives the design of CK suit of metrics and evaluation to these metrics so that these metrics should reflect accurate and precise results for object oriented based systems. Moreover, a set of new metrics are proposed that can find the impact on reusability of a class by using the combination of one CK metric with another metric.

Keywords

Reusability, CK Metric, Object - Oriented.

1. INTRODUCTION

Object oriented systems continue to share a major portion of software development and customer base for these systems is on the rise. This is because there are many advantages in taking the object oriented concept. The weakness though is that most object oriented systems tend to be quite complex. Hence, the quality of such systems takes priority and lots of time, money and effort is spent in ensuring it [1]. One such method that predicts quality of a software system is by evaluating impact on reusability of class of the software through the use of metrics. The introduction and subsequent use of metrics as a means to evaluate the software quality has had deep and useful impact on the overall system.

In this paper, an attempt is made to use object oriented metrics as a predictor for software complexity of the underlying system. The study consists of calculating and analyzing object oriented metrics on object oriented systems developed using C++.

The following section represents a review of related work. Section III discusses the brief description of the six class based CK Metrics. Following that Section IV proposed a set of three new metrics Metrics1, Metrics2, Metrics3. Following that Section V describes the analysis of the results. Following that summarizes the study undertaken and conclusions and future work and references.

2. RELATED WORK

A significant number of object oriented metrics have been developed. For example, metrics proposed by Abreu [2], CK metrics [3], Li and Henry [4] metrics, MOOD metrics [5], Lorenz and Kidd [6] metrics etc. CK metrics are the most popular among them. Another comprehensive set of metrics is MOOD metrics. Here one of the first suites of OO design measure was proposed by Chidamber and Kemerer (CK) [7], [8] will be discussed. The authors of this suite of metrics claim that these measures can aid users in understanding design complexity, in detecting design flaws and in predicting certain project outcomes and external software qualities such as reusability, software defects, testing, and maintenance effort. Use of the CK set of metrics and other complementary measures are gradually growing in industry acceptance [9]. CK metrics suite [8] is one of the object-oriented design complexity measurement systems which support the measurement of the external quality parameter which may evolve in software package.

3. OVERVIEW OF CK METRICS

Brief description of the six CK metrics suite for OO Design [10, 11] is the deepest research in OO metrics investigation:

3.1 Weighted Methods per Class (WMC)

It is defined as the sum of the complexities of all methods of a class.

- The number of methods and the complexity of methods involved is a predictor of how much time and effort is required to develop and maintain the class.
- The larger the number of methods in a class the greater the potential impact on children, since children will inherit all the methods defined in the class.
- Classes with large numbers of methods are likely to be more application specific, limiting the possibility of reuse.

3.2 Depth of Inheritance Tree (DIT)

It is defined as the maximum length from the node to the root of the tree.

- The deeper a class is in the hierarchy, the greater the number of methods it is likely to inherit, making it more complex to predict its behavior.
- Deeper trees constitute greater design complexity, since more methods and classes are involved.
- The deeper a particular class is in the hierarchy, the greater the potential reuse of inherited methods.

3.3 Number of Children (NOC)

It is defined as the number of immediate subclasses.

- The greater the number of children, the greater the reuse, since inheritance is a form of reuse.
- The greater the number of children, the greater the likelihood of improper abstraction of the parent class. If a class has a large number of children, it may be a case of misuse of sub classing.
- The number of children gives an idea of the potential influence a class has on the design. If a class has a large number of children, it may require more testing of the methods in that class.

3.4 Coupling between object classes (CBO)

It is defined as the count of the classes to which this class is coupled. Coupling is defined as : Two classes are coupled when methods declared in one class use methods or instance variables of the other class. [Chidamber and Kemerer 1994]

- Excessive coupling between object classes is detrimental to modular design and prevents reuse. The more independent a class is, the easier it is to reuse it in another application.
- In order to improve modularity and promote encapsulation, inter-object class couples should be kept to a minimum. The larger the number of couples, the higher the sensitivity to changes in other parts of the design, and therefore maintenance is more difficult.
- A measure of coupling is useful to determine how complex the testing of various parts of a design is likely to be. The higher the inter-object class coupling, the more rigorous the testing needs to be.

3.5 Response for a Class (RFC)

It is defined as number of methods in the set of all methods that can be invoked in response to a message sent to an object of a class.

- If a large number of methods can be invoked in response to a message, the testing and debugging of the class becomes more complicated since it requires a greater level of understanding on the part of the tester.
- The larger the number of methods that can be invoked from a class, the greater the complexity of the class.
- A worst case value for possible responses will assist in appropriate allocation of testing time.

3.6 Lack of Cohesion in Methods (LCOM*)*

It is defined as the number of different methods within a class that reference a given instance variable.

- A highly cohesive module should stand alone; high cohesion indicates good class subdivision.
- High cohesion implies simplicity and high reusability.
- Cohesiveness of methods within a class is desirable, since it promotes encapsulation. As a drawback, a highly cohesive class has high coupling between the methods of class, which in turn indicates high testing effort for that class.
- Lack of cohesion implies classes should probably be split into two or more subclasses.
- Low cohesion increases complexity, thereby increasing the likelihood of errors during the development process.

Object-oriented methodologies require significant effort early in project life cycle to identify objects and classes, attributes and operations, relationships between objects and classes, encapsulation, inheritance, and polymorphism require designers to carefully structure the design and consider the interaction between objects. Accordingly, much effort will be saved rather than rewriting the code and helps producing high quality software. In the current work, CK suite is utilized for several reasons: CK suite covers all aspects of object oriented (reusability, encapsulation and polymorphism). It was chosen by SATC (Software Assurance Technology Center) at NASA Goddard Space Flight Center [12], [13] and still used widely till now. Much effort was devoted for empirically validating [14], [15], [16] the original CK metrics and linking them to Object Oriented Design (OOD) quality attributes. Most of the other metrics are built upon the original CK metrics suite. It is easy to lift CK metrics from the code level to the model level [17]. CK suite could be kinked to economic variables (productivity, rework effort, and design) to assess practicing managers [18]. CK suite proves to be useful in predicting class fault proneness [19]. CK metrics are the most referenced among all other metrics [20].

4. PROPOSED METRICS

In this section, a set of new metrics are proposed to measure reusability of an OO codes.

4.1 Metric1 (DIT+NOC)

- ➢ Deeper a particular class is in the hierarchy, the greater the potential for reuse of inherited methods [21]. 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.
- ➢ A moderate value for NOC indicates scope for reuse [21]. Up to particular threshold value NOC has positive impact on reusability of a class.

- ✓ Therefore the reusability of a class increases with the increase in combination of DIT and NOC of a class. So DIT + NOC have positive impact on reusability of a class.

Let Metric1 = DIT (Depth of Inheritance) + NOC (Number of Children) [22].

4.2 Metric2 (CBO+LCOM)

- ➢ Excessive coupling indicates weakness of class encapsulation and may inhibit reuse [21]. It indicates that coupling has negative impact on reusability of a class.
- ➢ High LCOM increases complexity, thereby increasing likelihood of errors during the development process. The class should probably split into two or more smaller classes. It indicates that cohesion has negative impact on reusability of a class.

- ✓ Therefore CBO +LCOM have negative impact on reusability of a class.

Let Metric2 = CBO (Coupling between Objects) + LCOM (Lack of Cohesion of Methods) [22].

4.3 Metric3 (WMC+RFC*)*

- ➢ The large no. of methods in a class, the greater the potential impact on children. Classes with large no. of methods are likely to be more application specific, limiting the possibility of reuse. So WMC has negative impact or reusability of a class.
- ➢ The larger the no. of methods that can be invoked from a class through message, the greater the complexity of the class. So RFC has negative impact or reusability of a class.

- ✓ Therefore the reusability of a class decreases with the increase in combination of WMC and RFC of a class. So WMC+RFC have negative impact on reusability of a class.

Let Metric3 = WMC (Weighted Methods per Class) + RFC (Response for a Class) [22].

5. ANALYSIS RESULTS

The proposed set of metrics Metric1, Metric2 & Metric3 is applied to C++ program in Fig. 1 to measure the impact of reusability of a class.

```
#include<iostream.h>
#include<conio.h>
class GF
{
int a;
public:
GF(int x)
 {
 a=x;
 }
int geta( )
{
 return a;
}
};
class F: public GF
{
int b;
public:
F(int x, int y):GF(y)
{
 b=x;
}
int getb( )
{
 return b;
}
};
class S : public F
{
int c;
public:
S(int x, int y, int z):F(y,z)
{
 c=x;
}
void show( )
{
cout << geta( ) <<"    " << getb( ) << "    " ;
cout << c << "\n";
}
};
main( )
{
clrscr( );
S ob(1,2,3);
ob.show( );
cout << ob.geta( ) <<"    " << ob.getb( ) << "\n";
getch( );
return 1;
}
Output:
3          2          1
3          2
```

Parameterized constructor in base class GF

Parameterized constructor using in derived class F and executing from Base class GF

Parameterized constructor using in sub derived class S and executing from Base class GF.

Fig. 1

Table 1. VALUES OF PROPOSED METRICS

Metric / Class			Metric1			Metric2			Metric3
	DIT	**NOC**	**DIT+NOC**	**CBO**	**LCOM**	**CBO+LCOM**	**WMC**	**RFC**	**WMC+RFC**
GF (Grand Father	0	1	1	1	1	2	2	2	4
F (Father)	1	1	2	1	1	2	2	3	5
S (Son)	2	0	2	1	1	2	2	5	7

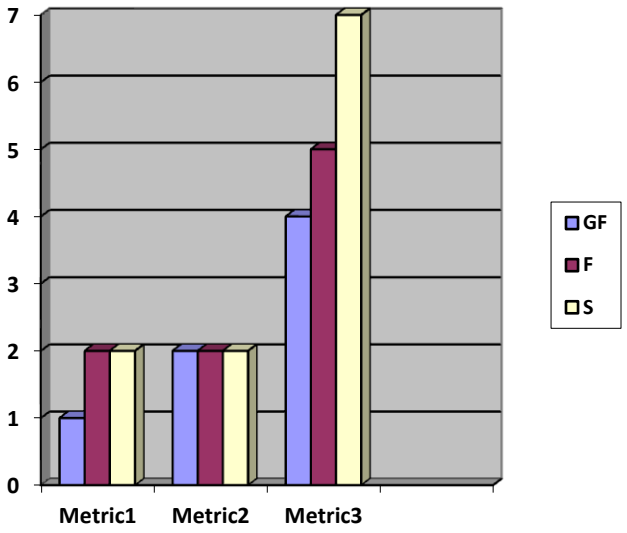

Fig. 2

As according to Table1 and Figure2, the values of Metric1 are very less as compare to in combination of Metric2 and Metric3. So we can say that object oriented system using parameterized constructor based upon C++ program is reusable up to some extent, because greater values of Metric2 and Metric3 means negative impact on reusability of class. So object oriented system using parameterized constructor based upon C++ program has negative impact on reusability of class.

6. CONCLUSION AND FUTRUE WORK

The organizations implement systematic software reuse programs in an effort to improve productivity and designing. Reusability increases with increase of DIT and NOC, reusability decreases with increase of CBO and LCOM, reusability decreases with increase of WMC and RFC. In this paper, an approach to measure the reusability of object oriented program based upon CK metrics is proposed. Since reusability is an attribute of software design and can analyze software design by measuring software reusability. Hence, this approach is important to measure reusability of class diagram.

The most obvious extension of this work is to analyze the degree to which these metrics correlate with managerial performance indicators such as testing, maintenance effort and quality.

This study can be followed up with another which includes the model necessary to map the metrics to software quality. Another future study prospect would be to have the data set as projects with identical requirements done in different object oriented languages. This would help us to ascertain that the metrics are capable of predicting the quality of software across the object oriented language.

7. REFERENCES

[1] Kayarvizhy, N., Kanmani, S. : " Analysis of Quality of Object Oriented Systems using Object Oriented Metrics", Electronics Computer Technology (ICECT), 2011 3rd International Conference on, April 8-10, 2011. Kanyakumari: IEEE Computer Society Press, 2011.

[2] Abreu, Fernando B. ,Carapuca, Rogerio.: "Candidate Metrics for Object-Oriented Software within a Taxonomy Framework.", Journal of systems software 26, 1(July 1994).

[3] Chidamber, Shyam , Kemerer, Chris F. "A Metrics Suite for Object-Oriented Design." M.I.T. Sloan School of Management E53-315, 1993.

[4] Li,Wei , Henry, Salley.: "Maintenance Metrics for the Object OrientedParadigm", First International Software Metrics Symposium. Baltimore,Maryland, May 21-22, 1993. Los Alamitos, California: IEEE Computer Society Press, 1993.

[5] Abreu, Fernando B: "The MOOD Metrics Set," Proc. ECOOP'95Workshop on Metrics, 1995.

[6] Lorenz, Mark & Kidd Jeff: "Object-Oriented Software Metrics", Prentice Hall, 1994.

[7] Chidamber S. R., and Kemerer C.F., "Towards a Metrics Suite for Object-oriented Design," Proc. Conf. Object-oriented Programming Systems, Languages, and Applications (OOPSLA'91), vol. 26, no. 11, pp. 197-211, 1991.

[8] Chidamber S. and Kemerer C.: "A Metrics Suite for Object-oriented Design", IEEE Transactions on Software Engineering, vol. 20, no. 6, pp. 476-493, June 1994.

[9] Subramanyam R., Krishnan M.S., "Empirical analysis of CK metrics for object-oriented design complexity: implications for software defects Software Engineering", IEEE Transactions on Publication Date: April 2003 Volume: 29, Issue: 4 On page(s): 297- 310.

[10] Shatnawi R., " A quantitative investigation of the acceptable risk levels of object-oriented metrics in open-source systems", IEEE Transactions on Software

Engineering, Vol. 36, No.2, pp. 223-224 March/April 2010.

[11] Camargo Cruz Ana Erika, "Chidamber & Kemrer Suite of Metrics", Japan Advanced Institute of Science and Technology School of Information, May 2008.

[12] Rosenberg, L. H. and Hyatt, L., "Applying and interpreting object oriented metrics," in Proceedings of Software Technology Conference, Utah, April 1998.

[13] Rosenberg, L. H. and Lawrence, E. H., "Software Quality Metrics for Object- Oriented Environments," Unisys Technology Conference, Virginia,1996.

[14] Chidamber, S. R. and Kemerer, C. F., "A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering," vol. 20, pp. 476-493, 1994.

[15] Succi, G, Pedrycz, W., Stefanvic, M., and Miller, J., "Practical assessment of the models for identification of defect-prone classes in object-oriented commercial systems using design metrics," The Journal of Systems and Software, vol. 65, pp. 1–12, 2003.

[16] Basili, V. L., Brianc, L., and Melo., W. L., "A Validation of Object- Oriented Metrics as Quality Indicators," IEEE Transactions Software Engineering, vol. 22,pp. 751-761, 1996.

[17] McQuillan, J. A. and Power, J. F., "On the application of software metrics to UML models," Springer Lecture Notes in Computer Science, vol. 4364, pp. 217-226. 2007.

[18] Chidamber, S.R., Darcy, D.P. and Kemerer, C.F., " Managerial use of Metrics for Object Oriented Software: an Exploratory Analysis," IEEE Transaction on Software Engineering, vol. 24, pp. 629-639, 1998.

[19] Benlarbi, S., Eman, K. El, Goel, N., Rai, S.,"Thresholds for Object-Oriented Measures," Proceedings of ISSRE2000, San Jose, CA, USA, pp. 24-37, 2000.

[20] Briand, L., Arisholm, E., Counsell, S., Houdek, F. and Thevenod-Fosse, P., "Empirical Studies of Object Oriented Artifacts, Methods, And Processes: State of the Art and Future Direction," In Empirical Software Engineering,vol. 4, pp. 387-404, 1999.

[21] Liang,V., and Colemon, C., "Principal Components of Orthogonal Object Oriented Metrics", Software Assurance Technology Center, White Paper SATC-323-08-14, NASA Goddard Space Flight Center, Greenbelt, Maryland 20771.

[22] Goel, B.M., Bhatia, P.K, "Investigation of Reusability Metrics for Object–Oriented Designing", Proceeding of NCETCIT - May, 2012, GVM IT&M, Sonipat, Har, India., pp. 104-110, 2012.

APPENDIX

Constructor in Drive Classes: When we have a parameterized constructor in base class then it is mandatory for drive class to create a parameterized constructor in its class and invoke the base class constructor.