Optimization of Object-Oriented Design using Coupling Metrics

Parul Gandhi
Department of Computer Science & Business Administration
Manav Rachna International University
Faridabad, 121001, India

Pradeep Kumar Bhatia
Department of Computer Science & Engineering
G. J. University of Science and Technology
Hisar, 125001, India

ABSTRACT
In spite of large acceptance of object-oriented paradigm, many programmers don't have a firm grip on the design principles and the intimate mechanisms of object-orientation and this result into to a lot of poor designed large scale OO systems. Coupling in the software is one of the most vibrant internal quality attribute to measure the design performance. In this paper, we propose Message Received Coupling (MRC) and Degree of Coupling (DC) metrics for the automatic detection of a set of design problems along with an algorithm to apply these metrics to redesign an object-oriented source code, if necessary. We also design a Method Calling Graph (MCG) that helps in calculating the value of proposed metrics. The revised set of metrics helps the developers to decide whether a design needs to be changed or left in its original form.

General Terms
Analysis, Metrics, Object-Oriented Design.

Keywords
Object oriented metrics, Coupling, Design optimization.

1. INTRODUCTION
Popularity of object-oriented paradigm requires acute analysis of object oriented software to accurately monitor the internal software quality attributes such as coupling, cohesion, size and complexity etc. According to Biggerstaff this paradigm has a good balance between power and generality [10]. Design is the backbone of any software system. Object-Oriented (OO) paradigm includes a set of mechanisms such as inheritance, encapsulation, and polymorphism and message-passing that plays an important role in designing of object-oriented code. Coupling has been defined as one of the most basic qualitative attribute to measure the performance of software at design or implementation phase [8, 9]. Good software design should have minimized coupling interaction [7, 12, 13]. Many object-oriented metrics have been proposed to assess the design of a software system [1, 2, 4].

The remaining section of this paper is organized as follows. Section 2 discusses the limitations of existing metrics. Section 3 introduces proposed coupling metrics. In Section 4 we introduce a new method that assist in doing the analysis of proposed metrics. Section 5 shows the experimental result which indicates why our metrics are better than the already existing metric. Section 6 discusses the new redesigning algorithm develop for automatic detection of design problem. Section 7 offers conclusions and directions for future research.

2. LIMITATIONS OF EXISTING METRICS
Numerous metrics have been proposed by various researchers to measure class coupling, these metrics fail to control coupling interaction in various parts of object-oriented system. Chidamber and Kemerer [6] and Li and Henry [11] have proposed a set of object-oriented metrics including Coupling Between Objects (CBO) and Message Pass Coupling (MPC). Informally, the CBO metric aims to measure the amount of interconnectivity between a given class and other classes in the system and MPC gives an indication of how many messages are passed among objects of the classes. The CBO metrics defined by Chidamber and Kemerer have later been formalized by Briand et al. [5]. MPC (Message Pass Coupling) addresses the external methods which is the "number of send statements defined in a class". If a message invokes numerous methods as a response, the class becomes more complicated and more testing and debugging is required.

In this paper we propose Message Received Coupling (MRC) and Degree of Coupling (DC) and also provide example to show that MRC and DC are more adequate to check the design performance of an object-oriented system as compare to MPC metric.

3. PROPOSED METRICS
We propose two metrics MRC and DC that helps to detect the flaws in design of object-oriented software at an early stage. In the MPC metric the redesigning of the software depends on only the number of send statements from a particular object regardless of the status of incoming call of that object which also plays an important role in the designing of software. The proposed metrics will help overcome the above said. These metrics are:

3.1 Message Received Coupling (MRC)
can be defined as the number of statements received by a class. It also used to measure the complexity of messages received among classes.

MRC= number of received statement in a class

3.2 Degree of Coupling (DC)
It will be calculated at class level. It can be defined as the ratio of number of incoming statements to a class to the ratio of

outgoing statements from a class. This metric helps in finding and correcting the defects in the design of object-oriented software in a better way as compare to Message Pass Coupling (MPC).

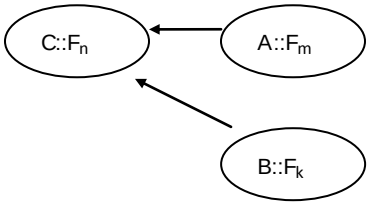
$$DC = \frac{MRC}{MPC}$$

Experimental results shows that much deviation in the value of this metric from the interval less than 0.5 and greater than 2 indicates the prioritize redesigning of that class.

4. ANALYSIS OF PROPOSED METRICS

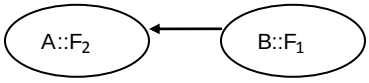
We design Method Calling Graph (MCG) that helps to evaluate the proposed metrics. Method Calling Graph (MCG) traverses all possible paths upon methods called in a class by another class. We construct on MCGs for each method in a class that is called by other methods

The nodes of graph are of the form A::fi where fi is a method in class A. If C::Fn is called by A::Fm and also by B::Fk then we construct it as follows:



We now compute MPC, MRC and DC value for all the classes in figure 1.

Class A has 10 methods. MCG for class A is



MPC for class A is 9 as 9 send statements are from class A. MRC count for this class is 1 and DC can be calculated as the ratio of MRC to MPC, therefore DC of class A is 1/9 i.e. 0.11

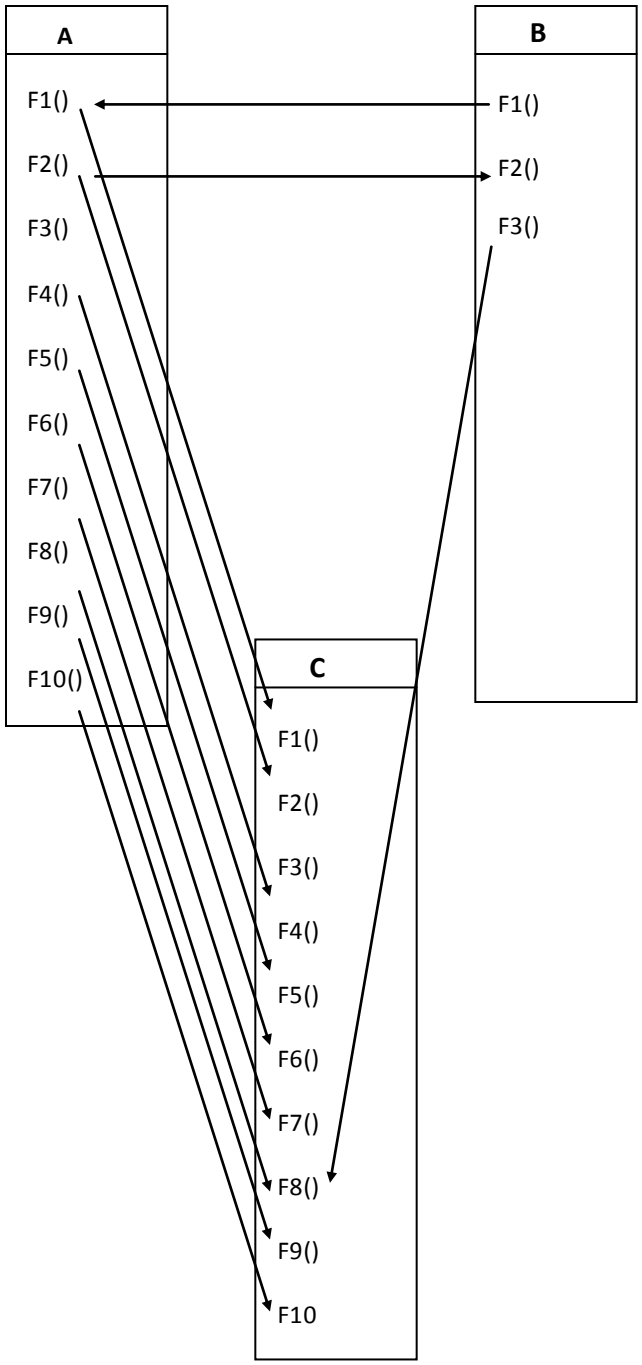
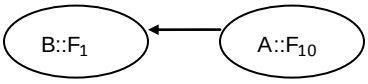


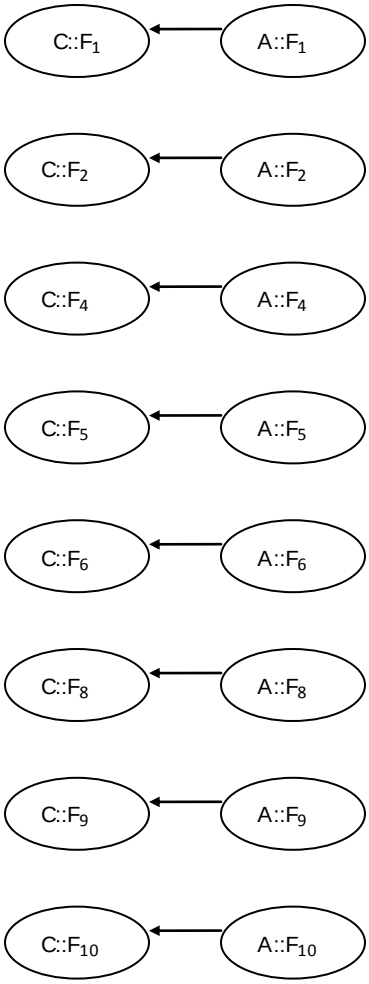
Fig 1: Method Invocation Chart

Class B has 3 methods. MCG for class B is



MPC for class B is 2 as 2 send statements are from class B. MRC count for this class is 1.

Class C has 10 methods. MCGs for class C is



MPC for class C is 0 as 0 send statements are from class C. MRC count for this class is 9 and DC can be calculated as the ratio of MRC to MPC, therefore DC of class C is 9/0 i.e. ∞

5. EXPERIMENTAL RESULTS

Table 1. Class Level Metrics

Class	Object-Oriented Metrics		
	MPC	**MRC**	**DC**
A	9	1	0.11
B	2	1	0.5
C	0	9	∞

DC responds better to change in coupling than MPC. A larger number of MPC indicates increased coupling between this class and other classes in the system. This makes the classes more dependants on each other which increases the overall complexity of the system and makes the class more difficult to change.

In the example discussed above the MPC value for class A is 9 whereas for class C MPC value is 0. MPC showed the designer to redesign Class A and there is no requirement to change class C. MPC in this case could provide partial guidance regarding the need to redesign class A, whereas DC showed that the classes having values less that 0.5 and greater than 2 needs to be redesigned, therefore both class A and C needs to be redesigned to maintain the balance in the designing of object-oriented software. Thus DC responds better which classes needs to be redesigned than MPC.

6. REDESIGN ALGORITHM

1. $\exists$ Class A, Generate MCG corresponding to each method in a class

2. Based on MCG Created, compute MRC for the class by

$$\text{MRC (A)} = \sum_{i=1}^{n} MCG_i$$

3. We now compute DC for the class

By

$$\text{DC (A)} = \frac{MRC(A)}{MPC(A)}$$

Where MPC is Message Pass Coupling metric value for the corresponding class.

4. If DC of a class is < 0.5 or >2 Then designer must decide which method should be moved without affecting the inheritance property.

Else

No action is required.

7. CONCLUSION

In this paper we have introduced new set of coupling metrics MRC and DC. The results show that these metric gives better indication about redesigning of source code as compare to existing metric MPC. We have also designed MCG that helps in calculating the value for these metrics and also design an algorithm that provides a way how to apply these metrics for the automatic detection of design flaws at an early stage of software development life cycle.

8. ACKNOWLEDGMENTS

I express my sincere gratitude and acknowledgement towards Dr. Pradeep Kumar Bhatia, Associate Professor, who guided me. It was his constant support and inspiration without which my efforts would not have taken this shape. I sincerely thank him for this, and seek his support for all my future endeavors.

9. REFERENCES

[1] Ghassan Alkadi and Ihssan Alkadi, "Applying A Revised RFC Metric to Redesign AN OO Design," Aerospace Conference, IEEE Proceeding, 2001.

[2] Simon Allier, St´ephane Vaucher, Bruno Dufour, and Houari Sahraoui, "Deriving Coupling Metrics from Call Graphs", IEEE International Workshop on Source Code Analysis and Manipulation, pp. 43-52, September 2010.

[3] Mandeep Kaur, Parul Batra & Akhil Khare,"Static Analysis And Run-Time Coupling Metrics", International Journal of Information Technology and Knowledge Management, Volume 3, No. 2, pp. 707-710, July-December 2010.

[4] Denys Poshyvanyk, Andrian Marcus, The Conceptual Coupling Metrics for Object-Oriented Systems", 22nd IEEE International Conference on Software Maintenance, 2006

[5] L. Briand, J. Daly, and J. W¨ust, "A unified framework for coupling measurement in object-oriented systems," IEEE Trans. on Software Engineering, vol. 25, no. 1, pp. 91–121, jan/feb 1999.

[6] S. Chidamber and C. Kemerer, "A metrics suite for object oriented design," IEEE Trans. on Software Engineering, vol. 20, no. 6, pp. 476–493, June 1994.

[7] Huan Li, "A Novel Coupling Metric for Object-Oriented Software Systems", IEEE International Symposium on Knowledge Acquisition and Modeling Workshop, pp. 609-612, 2008

[8] L.C. Briand, J.W. Daly, and J.K. Wust, "A Unified Framework for Coupling Measurement in Object-Oriented Systems", IEEE

[9] Transactions on Software Engineering, vol. 25, no. 1, pp. 91–121, Jan 1999.

[10] Troy, D.A., Zweben, S.H. "Measuring the Quality of Structured Designs", Journal of Systems and Software, pp. 113-120, 1981.

[11] Biggerstaff, T., and C. Richter, "Reusability Framework, Assessment, and Directions," IEEE Software, pp.41-49, March 1987.

[12] W. Li and S. Henry, "Object oriented metrics that predictmaintainability.", Journal of Systems and Software, pp. 111–122, Nov 1993.

[13] S.L. Pfleeger and J.M. Atlee, "Software Engineering: Theory and Practice" 3rd ed. Pearson Prentice Hall, 2006.

[14] G. Myers, "Software Reliability: Principles and Practices", Wiley, 1976.