IDENTIFYING COUPLING METRICS AND IMPACT ON SOFTWARE QUALITY

Vinay Singh [#1], Vandana Bhattacherjee [*2]

[#] Department of IT, Usha Martin Academy
Ranchi, India
[1] mailtovsingh@yahoo.co.in
[*]Department of CS & E, Birla Institute of Technology
Ranchi, India
[2] vbhattacharya@bitmesra.ac.in

***Abstract*— Coupling in software has been linked with maintainability and existing metrics are used as predictors of external software quality attribute such as fault-proneness, impact analysis, ripple effect of changes, changeability etc. Measurement helps in establishing these factors and hence the quality of a work product. Metric helps in deriving indicators from a collection of measures that can be used for analyzing and improving the quality baselines. Many coupling measures for object oriented software have been proposed each of them capturing specific dimensions of coupling. Coupling is considered by many to be an important concept in measuring design quality. Much of the existing work concentrates on direct coupling, that is, form of coupling that exists between entities that are directly related to each other. A form of coupling that has so far received little attention is indirect coupling, which is coupling between entities that are indirectly related. This paper identifies different coupling metrics both direct and indirect coupling metrics and study the impact on software quality.**

Keyword- Object Oriented, coupling, indirect coupling, transitive closure, software defect

I. INTRODUCTION

Reliability, availability and maintainability are some of the direct measures of product quality. These are sometimes argued as more purposeful than the defect measures as these indicate the usefulness of a product from the customer viewpoint.

Coupling was introduced by Constantine and others in 1960s as heuristic for designing modules. He observed that programs that were easier to implement and change were those composed of simple independent modules [5][18] and coupling was a way to determine how independent a proposed module was from the others in the system. Constantine and his colleagues' main concern was in reducing the cost of "debugging", which they observed was the dominant cost of software. One consequence of reducing the coupling between modules is the belief that it will reduce the likelihood that changes to one module will impact another.

From the above discussion, we conclude that the concept of coupling can help us with the management of changes or modifiability in software. We do not rule out other quality attributes, but for the remainder of this paper we will concentrate on this one. Thus, we will attempt to avoid the presence of unnecessary coupling which is an indication of a design that is more difficult than it should be.

The concept of coupling is considered by many to be important in investigating design quality. While our understanding of coupling is improving, most research has been applied only to direct coupling that is, coupling between modules that have some direct relationship. There has been comparatively little investigation into indirect coupling that is, coupling between modules that have no direct relationship. Indirect coupling in this paper demonstrate there are forms of indirect coupling that cannot be represented as the transitive closure of direct coupling, these forms are an important source of problems.

The data was made available through the students projects and metric data program at NASA [11]. We use software measurement data of NASA projects at class-level data for KC1. The KC1 project is a single large ground system that consists of 43 KLOC (thousand line of code) of C++ code. The data set contains 2107 modules of which 325 modules have one or more faults.

The objective of this study may be described as follows

- To find the different form of coupling metric identified in object-oriented classes
- Study on quality attributes functionality, user friendliness and effectiveness
- Relationship between coupling between objects and Number of defects[17]

Rest of the paper is organised as follows. The related work is presented in section 2. Proposed direct and indirect coupling is presented in section 3. Algorithm to identify coupling is defined in section 4. Section 5 studies the impact on software quality and the correlation of coupling between objects and number of defects. We present conclusion and future scope in section 6.

II. RELATED WORK

The problem of accessing the design quality of OO system has been of interest to many researchers [7][8][9].The complexity of coupling and cohesion are defined as quality measures for Object Oriented (OO) systems [2][3][4][14][15][16].

Chidamber and Kemerer were among the first to consider coupling for object oriented software [6][7]. Bernard provides a comprehensive survey of coupling in general and object oriented coupling in particular [1]. Briand et al. provide a framework with which to consider the many different forms of coupling that have been discussed in literature [5].

In almost all cases, the coupling metrics discussed are described in terms of features evident in the source code and referred to as direct coupling. There is very little discussion of coupling that might be caused by some non evident relationship between classes or indirect coupling. For example, Briand et al. describe nearly 30 coupling metrics of which only 2 measure some form of indirect coupling.

III. PROPOSED WORK

IEEE glossary [12] of software engineering defines modularity as the degree to which a computer program is composed of discrete components such that the change to one component has minimal impact on other component. Dependency between components is considered to be direct if the existence of the dependence is determinable by just looking at the source of the dependent component.

In this paper we propose a work of indirect coupling based on chaining method. Our metric is based on the idea that the longer the chain connecting two modules, the more hidden the dependencies are. Consequently it becomes more difficult to detect such indirect coupling. To some extent, indirect coupling can be detected by the transitive closure but there may be circumstances that instead of transitive closure the indirect coupling may still exist and it is hidden.

For example if one class (class A) calls a method of another class (class B) and that class(B) sends a message to another class for example (class C) for initializing an instance variable of class C, then class A is indirectly coupled with class C.

. We now give the definition of the Indirect Coupling metric ICM and analogously, Direct Coupling metric DCM for comparison.

The Coupling Metrics can be measured by the equations:

$$ICM = \frac{\sum IC_i}{\sum C_i}$$

Where IC_i is the number of classes to which a class C_i is indirectly coupled, and the summation is over all the classes.

$$DCM = \frac{\sum DC_i}{\sum C_i}$$

Where DC_i is the number of classes to which a class C_i is directly coupled, and the summation is over all the classes

IV. Algorithm to identify Coupling (Direct Coupling, Indirect Coupling from Transitive Rule, and Indirect coupling More than Transitive Rule)

The following algorithm is used to detect different form of coupling.

```
Identify Coupling (C)
{
  /*
   $C_i$ - Is the Number of classes in a program
   Dcm   - Is used to count the number of Direct Coupling
   Ict - Is used to count the number of indirect coupling by transitive rule
   Icmt - Is used to count the indirect coupling more than the transitive nature
*/
Step 1:          /*Initialization */
         Dcm=0, Ict=0,    Icmt=0
Step2:           /*Parse the input program class by class */
        for $C_i$=1 to N
Step 3:          /*count the number of Direct coupling */
        if Obj($C_i$+1) is a direct instance declared in class $C_i$ of another class $C_i$+1 then
                Dcm= Dcm+1
         End if
Step 4:  /*Indirect Coupling from Transitive rule */
        If   Obj($C_i$+1) is an instance declared in class $C_i$  of another class $C_i$+1 and Obj ($C_i$+2) is an
instance declared in class $C_i$+1 of another class $C_i$+2 then there is an indirect coupling from transitive rule
$C_i$ ->$C_i$+2  then
        Ict =Ict+1
        End if

Step 5:  /*Finding the chain */
        When a method in class $C_i$ called a method of  $C_i$+1  as an argument in which the method defined
in class $C_i$ sets the value for method defined in class $C_i$+2 . The set value is accessed by the successor of
the class $C_i$+2 which is the return type object of the method of class $C_i$.
Step 6: /* Relationship between $C_i$+1 with the successor of $C_i$+2 */
        If method in $C_i$+1 is fail to set the value, then the successor of $C_i$+2  will effected, hence there is a
relationship between $C_i$+1 and the successor of $C_i$+2 that cannot be detected from transitive rule, then
          Icmt= Icmt + 1
 Step 7: End of for loop
Step 8: End
```

ILLUSTRATION:

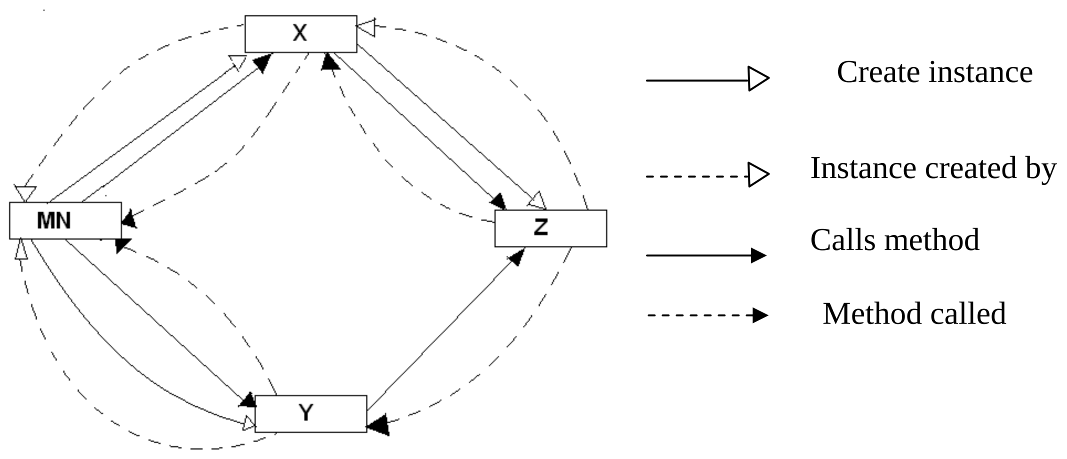
The following example demonstrates indirect coupling

```
Class MN
 {
  Public static void main(String args[])
{
     X x = new X();
    Y y = new Y();
     y.send(x.f1());
 }
}
class Z
{
       String message;
    void set(String message)
    {
        this.message =message;
    }
    String get()
   {
     return message;
       }
```

```
class X
{
    Z f1()

    {
    Z z = new Z();
    z.set("Initialize  Z");
       return z;
   }  }
class Y
{
    void send(Z z)
    {
System.out.println(z.get());
    }
  }
```

Figure 1 Calling relationships of given example



The notations for Figure 1 have been adapted from [18]. From the figure it can be seen that a different kind of relationship exists between X and Y. If we delete the statement z.set ("Initialize Z") from the class X it will result in a null pointer Exception thrown by Y because it tries to dereference z.get (). In other words there is a change to X that affects Y, which signifies dependency and further more an indirect one since it cannot be found by the transitive closure of direct dependencies.

The various metric values for the given illustration are presented in Table 1. Metric values for certain test data taken from Students Projects in C++ are given in Table 2.

Table 1
Metric values for the illustration of Figure 1

Serial No.	**No of class(C_i)**	**Direct Coupling(DC_i)**	**Indirect Coupling(IC_i)**	**DCM**	**ICM**
1	4	4	2	1	0.5

Table 2
Metric values for test data

Test data	No of class(Ci)	Direct Coupling(DC)	Indirect Coupling(IC)	DCM	ICM
PROJECT 1	10	4	2	0.4	0.2
PROJECT 2	15	6	4	0.4	0.26
PROJECT 3	20	12	6	0.6	0.3
PROJECT 4	25	9	7	0.36	0.28

V. IMPACT ON SOFTWARE QUALITY

In this section we study the impact of indirect coupling on software quality of the said C++ projects. For this we considered six quality attributes: Functionality, User friendliness, Ease of change in field width, ease of change in the order of menu display, Error handling effectiveness and Login Validation. The functionality metric was calculated by dividing the number of forms by the total number of functions. User friendliness was measured by the relative number of forms which were clearly displayed. Ease of change in the field width in a menu or forms was measured on an ordinal scale as 1 for Poor and 5 for Excellent. Ease of change in order of menu display was similarly measured. Error handling effectiveness was measured as the number of forms divided by the number of total field validations used. Finally Login Validation was a Boolean value depending on whether or not the Login form was present in the project.

Table 3
Measuring the quality attributes of software projects

METRIC	PROJECT 1	PROJECT 2	PROJECT 3	PROJECT 4
1. FUNCTIONALITY = ∑FORMS/∑FUNCTIONS	0.72	0.9	0.65	0.933
2. USER FRIENDLINESS=∑MENU/∑FORMS	0.375	0.166	0.2	0.642
EASE OF CHANGE IN FIELD IN A MENU OR FORMS(VALUE BETWEEN (1-5)	2	4	2	1
4. EASE OF CHANGE IN ORDER OF MENU DISPLAY(VALUE BETWEEN 1-5)	3	3	3	2
5. ERROR HANDLING EFFECTIVENESS= ∑forms/∑ field validations	0.66	0.62	0.937	0.848

Project 3 has the highest value of ICM metric, and it can be seen that the first quality metric values for this project are either lowest or second lowest. This holds for the fourth metric value also. The last two metrics are not affected much by coupling hence the impact is not visible.

We have also studied the KC1 project of NASA. After analysing the different attributes with the number of defects we attains that the coupling between objects causes defects.

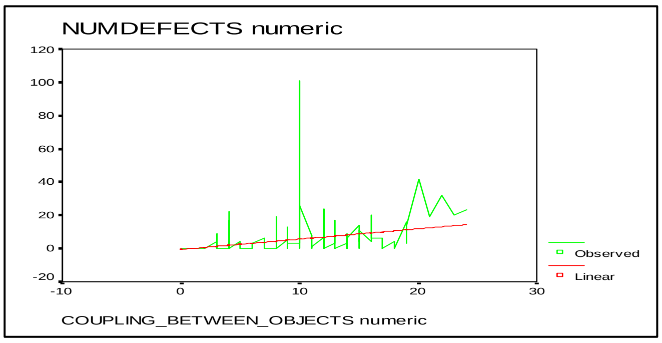


Figure 2 Linear relationships between CBO and num-defects [17]

Figure 2 shows that observed fluctuations are very high at certain point. It shows very high amount of variance with respect to actual required flow. This means CBO actually affects NUM DEFECT at points around 10 and (20-30) which is very large as compared with other results. Excluding such at all other points we see similar level of fluctuations which shows a mutual dependency between CBO and num-defects over most of the data points. This reaffirms that there is a strong relationship between coupling between object and number of defects indicates poor quality.

VI. CONCLUSION AND FUTURE WORK

In this paper the proposal for a metric to measure different form of coupling within classes is given. We have explored the potential impact of indirect coupling on quality and studied its effect on existing applications through initial empirical study. Our metric is based on the idea that the longer the chain connecting two modules, the more hidden the dependencies are. Consequently it becomes more difficult to detect such indirect coupling. To some extent, indirect coupling can be detected by the transitive closure but there may be circumstances that instead of transitive closure the indirect coupling may still exist and it is hidden. The work presented here is limited to a small number of projects, yet the impact of indirect coupling can be seen on quality attributes. The relationship between CBO and number of defects will arrive between (22 to 101) at 95% confidence level. We believe that the finding of this paper shall be foundation for some advanced work on coupling. It also remains to be seen that whether inclusion of more data over a brief time period changes the above results to any extent. We are currently in the process of validating the proposed metrics by our own software metric tool so that the measurement can be done on a bigger data set and with a wider set of quality metrics.

REFERENCES

[1] Bernard,E.V. Essays on object oriented software engineering prentice Hall,1993.

[2] Bhattacherjee,V. & K.Rajnish, “Class Complexity-A Case Study”, Proceedings of First International Conference on Emerging Application of Information Technology(EAIT-2006), Elsevier publication, Science City, Kolkata, India, 2006, pp. 253-258.

[3] Bhattacherjee,V.. Singh,V.,& Rajnish,K ” A new Dynamic cohesion metric using object oriented design”,proceedings of International conference on Advance Computing Technologies“ Hyderabad India(2008)

[4] Bhattacherjee,V.,Singh,V.,& Rajnish,K ”Dynamic cohesion metric using object oriented design”,proceedings of International conference on Compute communication and control “ Hoogly India(2009).

[5] Briand,L.C., Daly,J.W., & Wust,J.K. A Unified framework for coupling measurement in object oriented system. IEEE Transaction on Software Engineering ,(25(1): 91-121, January/February 1999

[6] Chidamber,S.R. & Kemerer,C.F.. Towards a metric suit for object oriented design.In proceeding of 6th ACM Conference on Object Oriented Programming system language and application(OOPSLA),pages 197-211,1991.

[7] Chidamber & Kemerer A Metric Suite for Object Oriented Design IEEE Transaction on Software Engineering 1994.

[8] Hitz,M., & Motazeri. “Measuring Coupling and Cohesion in Object Oriented System” in Proc. International Symposium on Applied Corporate Computing,Monterrey,Mexico,Oct. 1995

[9] Hitz.M, & B.Montazeri, Correspondence, Chidamber and Kemerer’s Metrics Suite: “A Measurement Theory Perspective”, IEEE Trans. on Software Engineering, 22, 4(1996), 267-271

[10] Hong Yul Yang, Ewan Tempero,Rebecca Berrigan “ Detecting Indirect Coupling” proceeding of the 2005 Australian Software Engineering Conference(ASWEC ‘05’).

[11] http://promise.site.uottawa.ca/SERepository

[12] IEEE standard glossary of software engineering terminology, 1990

[13] Lorenz,M. & Kidd,J.“Object-Oriented Software Metrics”: A Practical Guide, 1994.

[14] Rajnish,K & Bhattacherjee,V., “Maintenance of Metrics through class Inheritance hierarchy”, proceedings of International conference on Challenges and Opportunities in IT Industry”, PCTE, Ludhiana, 2005, pp.83.

[15] Rajnish,K & Bhattacherjee,V.,” A New Metric for Class Inheritance Hierarchy: An Illustration”, proceedings of National Conference on Emerging Principles and Practices of Computer Science & Information Technology”, GNDEC, Ludhiana, 2006, pp 321-325.

[16] Rajnish,K & Bhattacherjee,V.. ”Complexity of Class and Development Time: A Study”, Journal of Theoretical and Applied Information Technology (JATIT-2K6), Asian Research Publication Network, Islamabad, Pakistan, Vol. 3, No.1, June-Dec-2006, pp. 63-70.

[17] Bhattacherjee V, Singh V & Bhattacharya S “An Analysis of Dependency of Coupling on Software Defects” ACM Sig Soft Software Engineering Note. January2012,Volume37,Number1,DOI:10.1145/2088883.2088899,http://doi.acm.org/10.1145/ 2088883.2088899

[18] Yourdon,E. & Constantine,L.L. Structured Design: Fudamental of a discipline of computer program and system design prentice hall , 1979