Journal of Software Engineering and Applications, **2012, 5, 671-676**
http://dx.doi.org/10.4236/jsea.2012.59079 Published Online September 2012 (http://www.SciRP.org/journal/jsea)

Impact of Coupling and Cohesion in Object-Oriented Technology

Vipin Saxena*, **Santosh Kumar**

Department of Computer Science, Babasaheb Bhimrao Ambedkar University, Lucknow, India.
Email: *vsax1@rediffmail.com

Received June 26th, 2012; revised July 30th, 2012; accepted August 15th, 2012

ABSTRACT

The interaction between the classes or within the classes shows the complexity of the design. For one smaller problem, there may be more than one software design but who will be the best; depends on the complexity level of software design. Therefore, coupling and cohesion which shows the interlinking of classes and strength of classes; control the complexity of the design. The best software object oriented design is based upon the low coupling and high cohesion level. In the present work, a real case study of Life Insurance policy for handicapped person is demonstrated through the UML Class Diagram; coupling and cohesion levels are measured and results are demonstrated in the form of tables.

Keywords: UML; Object-Orientation; Coupling; Cohesion and Class Diagram

1. Background and Related Work

The Unified Modeling Language *i.e.* known as UML is a very popular and powerful modeling language which provides a collection of modeling tools for drafting the software designs based on the object-oriented technology. It is one part of any software system which is to be developed by software programmer by using the object-oriented language. It s not a process oriented language but it provides only visual syntax for designing the UML models. Therefore, one can say that the UML is a standard modeling language in the software development field which shows the every aspects and behavior of the system. The software professionals and researchers have used it very widely to develop an object-oriented system in the current scenario. The object-oriented paradigm is generally used by the software professionals and researchers for the development of real complex software systems. This paradigm moves around the objects and classes; it requires the real analysis of the system to view the interconnections between objects with the classes and attributes with methods. The interconnectivity between objects and attributes is monitored by the coupling and cohesion technique. A coupling finds the connectivity between classes while cohesion gives the strength to the bond between attributes.

Let us briefly explain the literature related to the present work. Gui and Scott [1] have proposed an account of new measure of coupling and cohesion developed to assess the reusability of Java components. Vanderfeesten *et al.* [2] have proposed a heuristic model that offers guidance for the creation and evaluation of process designs in the administrative settings. Designers can use this heuristic to select the several alternatives for the process design that is strongly cohesive and weakly coupled. Meyers and Binkley [3] have presented a large-scale empirical study of slice-based cohesion and coupling metrics and presented the results of the study. Jeong *et al.* [4] have proposed the cohesion and coupling metrics namely Average Cohesiveness of States (ACOS) and Average Number of Similar States of States (ASSOS), to evaluate the understandability of state diagrams. Ensan and Du [5] have presented a set of semantic metrics for measuring cohesion and coupling in modular ontologies based on the semantic of modular ontologies. Ujhazi *et al.* [6] have presented two novel conceptual metrics for measuring coupling and cohesion in software systems; first one is Conceptual Coupling between Object Classes (CCBOC) while other one is Conceptual Lack of Cohesion on Methods (CLCOM). Husein and Oxley [7] introduced the coupling and cohesion metrics and implemented for the object-oriented software systems. Chowdhury and Zulkernine [8] have conducted an extensive case study on Mozilla Firefox to provide empirical evidences on how vulnerabilities are related to complexity, coupling, and cohesion. Chowdhury and Zulkernine [9] have also presented a framework to automatically predict vulnerabilities based on CCC metrics. Oh *et al.* [10] have proposed novel metrics to measure ontology modularity and for evaluated the cohesion and coupling based on the

*Corresponding author.

theory of software metrics. Gandhi and Bhatia [11] have proposed 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. Hui *et al.* [12] also proposed a novel method to realize dual-redundancy detection for motor rotor and joint positions by installing two resolvers onto motor shaft at the same time, which can also have improve position detection precision and noise immunity by difference principle, decrease joint size and simplify joint structure.

In the current scenario, many of the software companies are shifting their old traditional based software systems towards the object-oriented software systems. Therefore, it is necessary to study the linking between the various designed classes or linking with the attributes. Therefore, the present work is an attempt to find out the impact of coupling and cohesion on the object-oriented design. A real case study of the Insurance Policy for the handicapped person is considered and converted into the UML class diagram alongwith attributes and thereafter impact of coupling and cohesion is measured and computed results are recorded in the form of tables.

2. Coupling and Cohesion

The term coupling is used to measure the relative interdependency between various classes as one class has the link with another class. While on the other hand cohesion is defined as the strength of the attributes inside the class which means how the attributes are linked inside the class. Coupling is always correlated with cohesion in such a way as if coupling is high then cohesion is low and vice versa. One can say that a class is highly coupled or many dependent with other classes, if there are many connections and loosely coupled or some dependent with other classes if there is a less connections. The coupling is decided at the designing phase of the system, it depends on the interface complexity of the classes. Therefore, the coupling is a degree at which a class is connected with other classes in the system.

Let us now describe the cohesive class which can perform a single task within the software procedure. It requires little interaction with other procedures that are used in other parts of a program. Cohesion gives the strength to the bond between attributes of a class and it is a concept through which capture the intra-module with cohesion. Therefore, cohesion is used to determine how closely or tightly bound the internal attributes of a class to one another. Cohesion gives an idea to the designer about whether the different attributes of a class belong together in the same class. Thus, the coupling and cohesion are related with each other; therefore the **Figure 1** shows the general representation of coupling and cohesion.

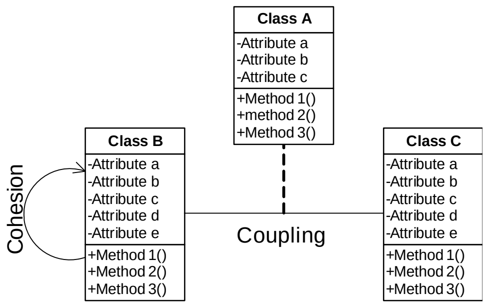

Figure 1. General representation of cohesion and coupling.

3. UML Class Diagram

UML class diagram shows the functionality of a system in a diagrammatic form, in which the classes are designed and combined for designing the software system. The dependent and independent classes are also designed in the form of classes and subclasses in the UML class diagram. In the current work, authors have taken a UML class diagram of opening the policy for handicapped in Life Insurance Corporation of INDIA as an example to demonstrate the approach. The class diagram shows the complete process of issuing a policy for handicapped. The independent classes and dependent classes are represented in the class diagram; the independent classes are shown along with the solid connecting lines while dependent classes are shown alongwith the dotted arrows as shown in the **Figure 2**.

Independent and Dependent Classes

Independent classes (IC's) are those classes which are not depending on other classes of the system. So, two classes are independent if one class can function without the presence of other class; these classes are easily solvable and modifiable separately. However, in any system all the classes are not independent but there are many classes which are dependent to other classes. The functionality of dependent classes (DC's) is affected when the changes are made in the attributes of the classes on which the classes are dependent. Thus, the dependent classes are those classes which use the attributes of other classes, therefore the DC's are more dependent to other classes if it use number of more attributes of other classes and less dependent if they use less attributes of other classes. The number of classes can be determined by the summation of all the IC's and DC's of the system. From the **Figure 2**, total number of classes (TC's):

$$\text{TC} = \text{Number of IC's} + \text{Number of DC's} \qquad (1)$$

4. Experimental Study

In the experimental study, the authors have used a metric [11] DCH (Degree of Cohesion) by exploring the two

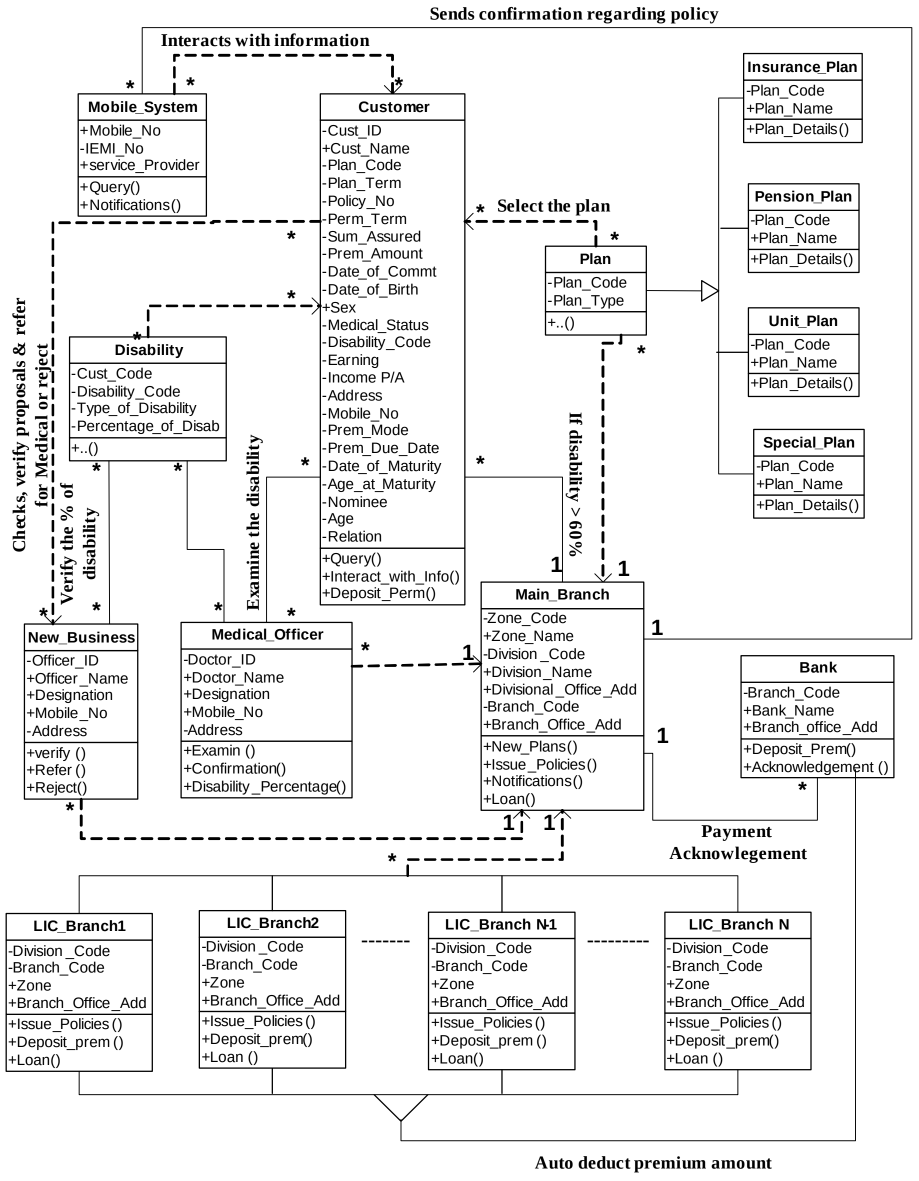

Figure 2. UML class Diagram for handicapped.

metrics MRC (Message Received Coupling) and DCP (Degree of Coupling) which helps to measure the functional strength of the classes of an object-oriented software system. DCH plays an important role in the field of software designing for measuring the dependency of the classes. The degree of coupling is calculated after calculating the number of DC's, IC's and the degree of coupling of the system. The metric has been implemented on the designed UML model for life insurance plan for Handicapped persons.

4.1. Degree of Coupling

The degree of coupling [11] is computed as the ratio of number of message received to the number of message passed. For finding the degree of coupling message received coupling (MRC) is used, it is a set of number of message received by a class. The degree of coupling is given

$$\text{Degree of Coupling }(\text{DC}) = \frac{\text{MRC}}{\text{MPC}} \qquad (2)$$

4.1.1. Message Received Coupling (MRC)

The MRC measures the complexity of message received by the classes, as MRC is the number of message received by a class form the other classes.

4.1.2. Message Passed Coupling (MPC)

The MPC is defined as the number of message passed among objects of the classes; it is the low level coupling that is achieved through the communication between the components.

For calculating the degree of coupling, the MCG (Message Calling Graph) is designed and represented below:

All the nodes of the graph are in the form of **A::M$_i$** where Mi is methods of class A, thus **A::M$_i$** is called by the **B::M$_j$**, **C::M$_k$**, **D::M$_l$** and **E::M$_m$**; where {i, j, k, l, m} = 1, 2, 3, 4, ··· the message calling graph is constructed as follow:

$$\begin{tikzcd} & B::M_j & \\ E::M_m & A::M_i \arrow[l] \arrow[u] \arrow[r] \arrow[d] & C::M_k \\ & D::M_l & \end{tikzcd}$$

On the basis of above, let us compute the values of MPC, MRC and DC for all the classes which are shown in the **Figure 3** and

Methods()	Methods Name
M$_1$ ()	Issue_Policy()
M$_2$ ()	Queries()
M$_3$ ()	Deposit_Prem()
M$_4$ ()	Notifications()
M$_5$ ()	Verifications()
M$_6$ ()	Examine()
M$_7$ ()	Confirmation()
M$_8$ ()	Rejection()
M$_9$ ()	Loan()

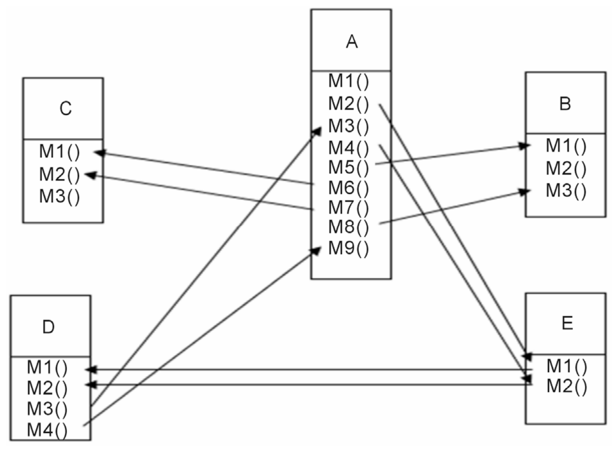

Figure 3. Method calling graph (MCG).

Class A has 9 Methods, but this class has not called any method, thus the MPC for this class is 8 and MRC is 0, therefore the Degree of Coupling can be computed as:

$$\text{Degree of Coupling } (\text{DC}) = \frac{\text{MRC}}{\text{MPC}}$$

$$\text{DC} = \frac{0}{8} = 0$$

Class B has 3 methods, so the message calling graph for this class is shown below:

$$\begin{tikzcd} B::M_1 & A::M_5 \arrow[l] \\ B::M_3 & A::M_8 \arrow[l] \end{tikzcd}$$

Thus, the MPC for class B is 0 *i.e.* the class B has not sent any statement to other classes and the MRC for class B is 2. Therefore the Degree of Coupling can be calculated as:

$$\text{DC} = \frac{2}{0} = \infty$$

Class C has 3 methods also but no any method passes, so the MCG for class C is as below:

$$\begin{tikzcd} C::M_1 & A::M_6 \arrow[l] \\ C::M_2 & A::M_7 \arrow[l] \end{tikzcd}$$

Thus, the MPC for class C is 0 *i.e.* the class C has not sent any statement to other classes and the MRC for class C is 2. Therefore, the Degree of Coupling can be calculated as:

$$\text{DC} = \frac{2}{0} = \infty$$

Class D has 4 methods, so the MCG for the class E is shown below:

$$\begin{tikzcd} D::M_1 & E::M_1 \arrow[l] \\ D::M_2 & E::M_2 \arrow[l] \\ D::M_3 & A::M_3 \arrow[l] \\ D::M_4 & A::M_9 \arrow[l] \end{tikzcd}$$

Thus the MPC for class D is 0 *i.e.* the class D has not sent any statement to other classes and the MRC for class D is 4. Therefore the Degree of Coupling can be calculated as:

$$\text{DC} = \frac{4}{0} = \infty$$

Class E has 2 methods, so the MCG for the class E is shown below:

$$\begin{tikzcd}
E::M_1 & A::M_2 \arrow[l] \\
E::M_2 & A::M_4 \arrow[l]
\end{tikzcd}$$

Thus, the MPC for class E is 2 *i.e.* the class E has passed two statements to the class D and the MRC for class E is 2. Therefore the Degree of Coupling can be calculated as:

$$\text{DC} = \frac{2}{2} = 1$$

The DC of each class is shown in the tabular form and shown as below (**Table 1**).

4.2. Degree of Cohesion

In DCH metric the functional strength [12] of the associated attributes within a class is measured which shows that how strongly a method is depending on the attributes of a class. In DCH, the strength of a function is depending on the number of attributes of a class which are used by the function. It is calculated at the attributes level, the ratio of the number of attributes used to the total number of attributes.

$$\text{Degree of Cohesion } (\text{DCH}) = \frac{\text{NAU}}{\text{TNA}} \qquad (3)$$

where NAU = Number of Attributes Used and TNA = Total Number of Attributes.

For computing the degree of cohesion, let us design an attribute calling graph (ACG) as shown in **Figure 4** which shows the connectivity between the classes and the number of attributes that are used by the methods of a class. The dependent classes are LIC_Branch (Class A), New_Business (Class B), Medical_Officer (Class C), Cuatomer (Class D), and Mobile_System (Class E) from the **Figure 2** that helps to design the proposed metric.

According to the attribute calling graph (ACG) the degree of cohesion (DCH) for each class is calculated which is shown in the **Table 1**.

5. Results and Discussions

From the above **Table 1** it is shown that the proposed model is loosely coupled while from the **Table 2** it is shown that the proposed model is highly cohesive due to the maximum number of attributes are used by the methods of the classes.

Table 1. Class level Metrics.

Class	Object-Oriented Matrices		
	MPC	MRC	DC
A	8	0	0/8
B	0	2	∞
C	0	2	∞
D	0	4	∞
E	2	2	1

Table 2. Attributes level Metrics.

Class	Object-Oriented Matrices		
	No. of attributes used	Total No. attributes	Degree of cohesion
A	4	4	4/4 = 1
B	4	6	4/6
C	4	6	4/6
D	7	24	7/24
E	2	3	2/3

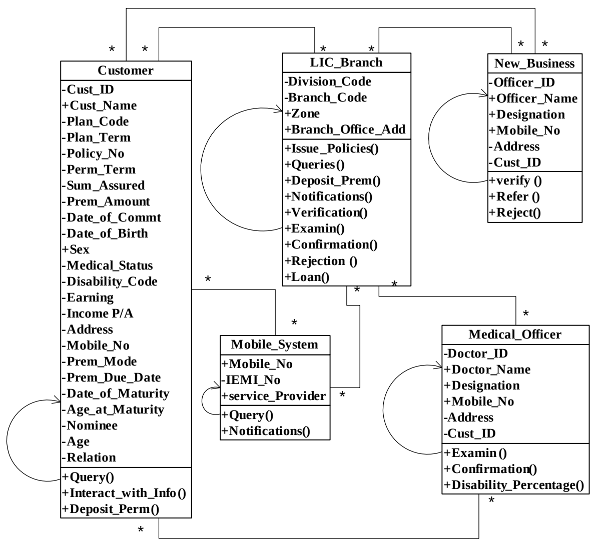

Figure 4. Attribute calling graph.

6. Conclusion

From the above work, it is concluded that UML is the powerful language and used to model the research problem. A performance metric is used to analyze the performance of the designed UML model. In the model, maximum numbers of the attributes are used by the numbers of the defined classes. The linking between the classes is demonstrated by the coupling while the strength of class called as attributes is represented by the cohesion. It is observed that the proposed model is highly cohesive although maximum numbers of the attributes are used in the designed UML class model.

7. Acknowledgements

Thanks are due to University Grant Commission New Delhi, for financial support to carry out the above research work.

REFERENCES

[1] G. Gui and P. D. Scott, "New Coupling and Cohesion Metrics for Evaluation of Software Component Reusability," *Proceedings of the* 9*th International Conference for Young Computer Scientists*, Zhangjiajie, 18-21 November 2008.

[2] I. Vanderfeesten, H. A. Reijers and W. M. P. van der Aalst, "Evaluating Workflow Process Designs Using Cohesion and Coupling Metrics," *Computers in Industry*, Vol. 59, No. 5, 2008, pp. 420-437. doi:10.1016/j.compind.2007.12.007

[3] T. M. Meyers and D. Binkley, "An Empirical Study of Slice-Based Cohesion and Coupling Metrics," *ACM Transactions on Software Engineering and Methodology*, Vol. 17, No. 1, 2007, Article No. 2.

[4] Y. J. Jeong, H. S. Chae and C. K. Chang, "Semantics Based Cohesion and Coupling Metrics for Evaluating Understandability of State Diagrams," *IEEE* 35*th Annual Computer Software and Applications Conference*, IEEE Computer Society, Washington, 18-22 July 2011.

[5] F. Ensan and W. Du, "A Metric Suite for Evaluating Cohesion and Coupling in Modular Ontologies," *Proceedings of the* 2010 *Conference on Modular Ontologies*: *Proceedings of the* 4*th International Workshop*, IOS Press, Amsterdam, 11 May 2010.

[6] B. Ujhazi, R. Ferenc, D. Poshyvanyk and T. Gyimothy, "New Conceptual Coupling and Cohesion Metrics for Object-Oriented Systems," *Proceedings of the* 2010 10*th IEEE Working Conference on Source Code Analysis and Manipulation*, Timişoara, 12-13 September 2010. doi:10.1109/SCAM.2010.14

[7] S. Husein and A. Oxley, "A Coupling and Cohesion Metrics Suite for Object-Oriented Software," *Proceedings of the* 2009 *International Conference on Computer Technology and Development*, Vol. 1, IEEE Computer Society, Washington, 2009. doi:10.1109/ICCTD.2009.209

[8] I. Chowdhury and M. Zulkernine, "Can Complexity, Coupling, and Cohesion Metrics Be Used as Early Indicators of Vulnerabilities?" *Proceedings of the* 2010 *ACM Symposium on Applied Computing*, ACM, New York, 2010, pp. 1963-1969.

[9] I. Chowdhury and M. Zulkernine, "Using Complexity, Coupling, and Cohesion Metrics as Early Indicators of Vulnerabilities," *Journal of Systems Architecture*, Vol. 57, No. 3, 2011, pp. 294-313. doi:10.1016/j.sysarc.2010.06.003

[10] S. Oh, H. Y. Yeom and J. Ahn, "Cohesion and Coupling Metrics for Ontology Modules," *Information Technology and Management*, Vol. 12, No. 2, 2011, pp. 81-96. doi:10.1007/s10799-011-0094-5

[11] P. Gandhi and P. K. Bhatia, "Optimization of Object-Oriented Design Using Coupling Metrics," *International Journal of Computer Applications*, Vol. 27, No. 10, 2011, pp. 41-44.

[12] H. Li, Y. He, Z. H. Jiang, Y. C. Huang and Q. Huang, "High-Cohesion and Low-Coupling Integrative Joint for Space Manipulator," *Advanced Intelligent Mechatronics*, *IEEE/ASME International Conference*, Singapore City, 14-17 July 2009, pp. 1463-1467.