International Journal of Enterprise Computing and Business Systems
ISSN (Online) : 223-8849
http://www.ijecbs.com

Volume 1 Issue 2 July 2011

EVALUATING THE IMPACT OF DIFFERENT TYPES OF INHERITANCE ON THE OBJECT ORIENTED SOFTWARE METRICS

Arti Chhikara
Maharaja Agrasen College, Delhi, India.

R.S.Chhillar
Deptt. Of Computer Sc. And Applications, Rohtak, India.

Sujata Khatri
Deen Dyal Upadhyaya College, Delhi, India

Abstract: Object Oriented Metrics plays a pivotal role in the development of fault free software product. Object Oriented Metrics are mainly designed for Object Oriented Systems which are based on the principles of Localization, Abstraction, Encapsulation, Information Hiding and Inheritance. This research paper focuses on effects of inheritance on object oriented metrics. A Software company schema is taken and different designs are made showing different level of inheritance. Chidamber & Kemerer metric suit is calculated for each design and the results are evaluated.

Keywords: Inheritance, metrics, Object Oriented System, Software Product.

1. Introduction

Software metrics are essential to software engineering for measuring software complexity and quality, estimating cost and project effort to simply name a few. The traditional metrics like function point, software science and cyclomatic complexity have been well used in the procedural paradigm. However, they do not readily apply to aspects of the OO paradigm [10, 11].

It is important to distinguish between the design principles of object oriented approach and the design principles of functional oriented approach, in order to clarify many aspects of the object orientation and allow better quality and administration management [1, 2, 3, 5, 11].
Pressman [7] points at five situations, where the object oriented metrics can be configured.

- Localization: It relates to the tendency of information in being centralized.
- Encapsulation: Encapsulation means that objects include their data and attributes.
- Information Hiding: Information hiding means to hide object characteristics (data and attributes).
- Inheritance: This property allows the possibility of deriving a new class and giving it the attributes of a class or more (partially or as a whole).

International Journal of Enterprise Computing and Business Systems
ISSN (Online) : 223-8849
http://www.ijecbs.com

Volume 1 Issue 2 July 2011

- Object Abstraction Technique: This technique allows the designer to concentrate only on the basic and necessary details of certain parts of program.

The OO technology forces the growth of OO software metrics [8]. Several such metrics have been proposed. The metrics suite proposed by Chidamber and Kemerer is one of the best-known OO metrics. Chidamber and Kemerer (1994) introduced a CK metrics suite which consists of:

- Weight Methods per Class (WMC),
- Depth of Inheritance Tree (DIT),
- Number of Children (NOC),
- Coupling Between Object classes (CBO),
- Response For a Class (RFC) and
- Lack of Cohesion in Methods (LCOM)

In this research paper, I have evaluated the impact of different types of inheritance: single, hierarchical, Multilevel on the values of object oriented metrics and how this affects the design of a software product. A software company schema is considered and its different designs are made without using inheritance and with using different types of inheritance. The designs are converted into java language code. All the six object oriented metrics of CK metric suit is calculated for each design and the results are evaluated.

The rest of the paper is organized as follows. Section 2 presents the brief overview of the inheritance and object oriented metrics. Section 3 presents the different designs of Employee database Schema .Section 4 presents the results based on collected data. Section 5 presents the discussion and Section 6 presents the conclusion.

2. Inheritance and Object Oriented Metrics:

One of the important characteristic of the OO system is inheritance. Inheritance is the ability of one class to acquire the properties of another class. The basic idea behind inheritance was reusability of code i.e. we do not have to write the same code again and again. Once a behavior (method) is defined in a super class, that behavior is automatically inherited by all subclasses. Thus, you write a method only once and it can be used by all subclasses. Once a set of properties (fields) are defined in a super class, the same set of properties are inherited by all subclasses. A class and its children share common set of properties. A subclass only needs to implement the differences between itself and the parent Inheritance is a key feature of the OO paradigm. This mechanism supports the class hierarchy design and captures the IS-A relationship between a super class and its subclass. Class design is central to the development of OO systems. Because class design deals with functional requirements of the system, it is the highest priority in OOD (Object-Oriented Design). The use of inheritance is claimed to reduce the amount of software maintenance necessary and ease the burden of testing [3, 8, 9, 10] and the reuse of software through inheritance is claimed to produce more maintainable, understandable and reliable software [3].

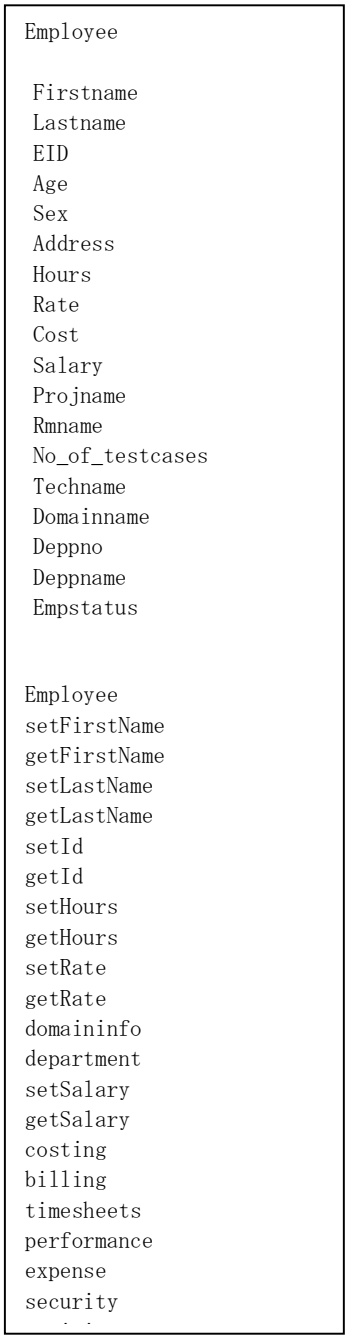
International Journal of Enterprise Computing and Business Systems
ISSN (Online) : 223-8849
http://www.ijecbs.com

Volume 1 Issue 2 July 2011

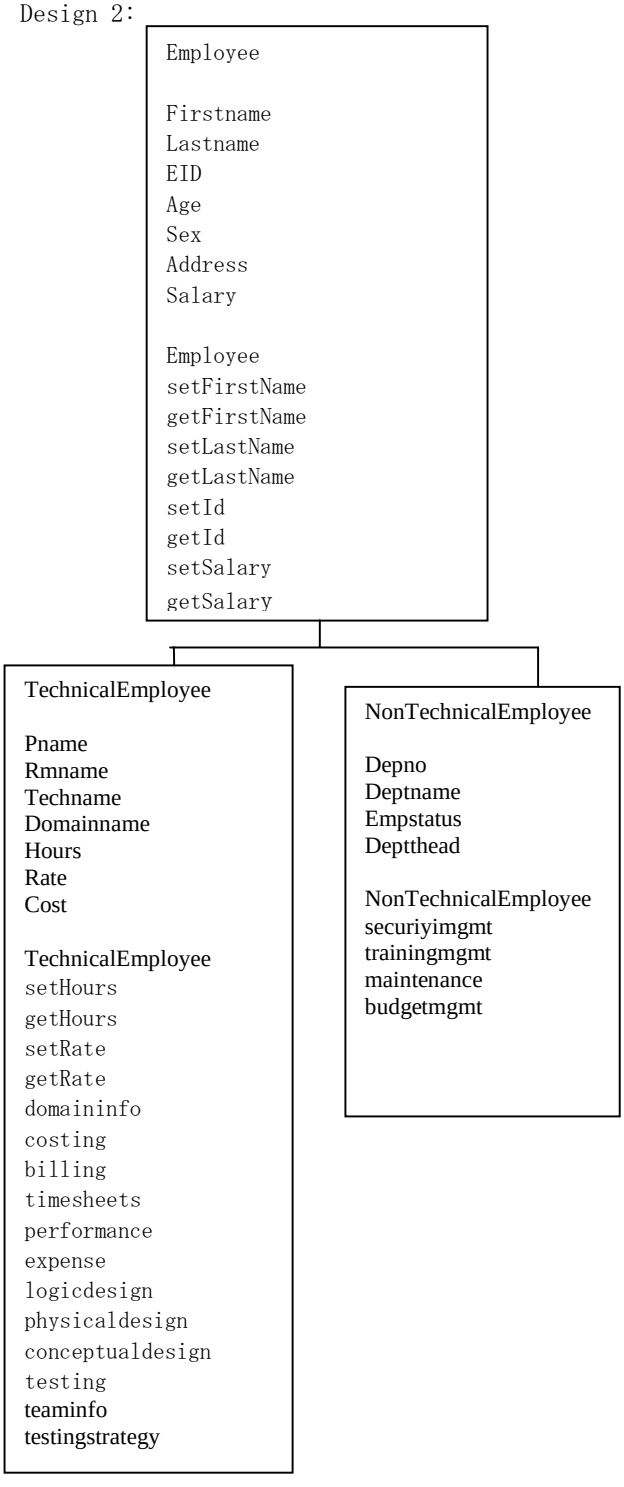
Because inheritance is an important characteristic in many Object Oriented Systems, many Object Oriented metrics focus on it. The inheritance metrics give us information about the inheritance tree of the system. Metrics such as Depth of Inheritance, Number of children, Number of parents, Class hierarchy nesting level are based on it.

3. Different Designs using Different levels of Inheritance:

Design 1:



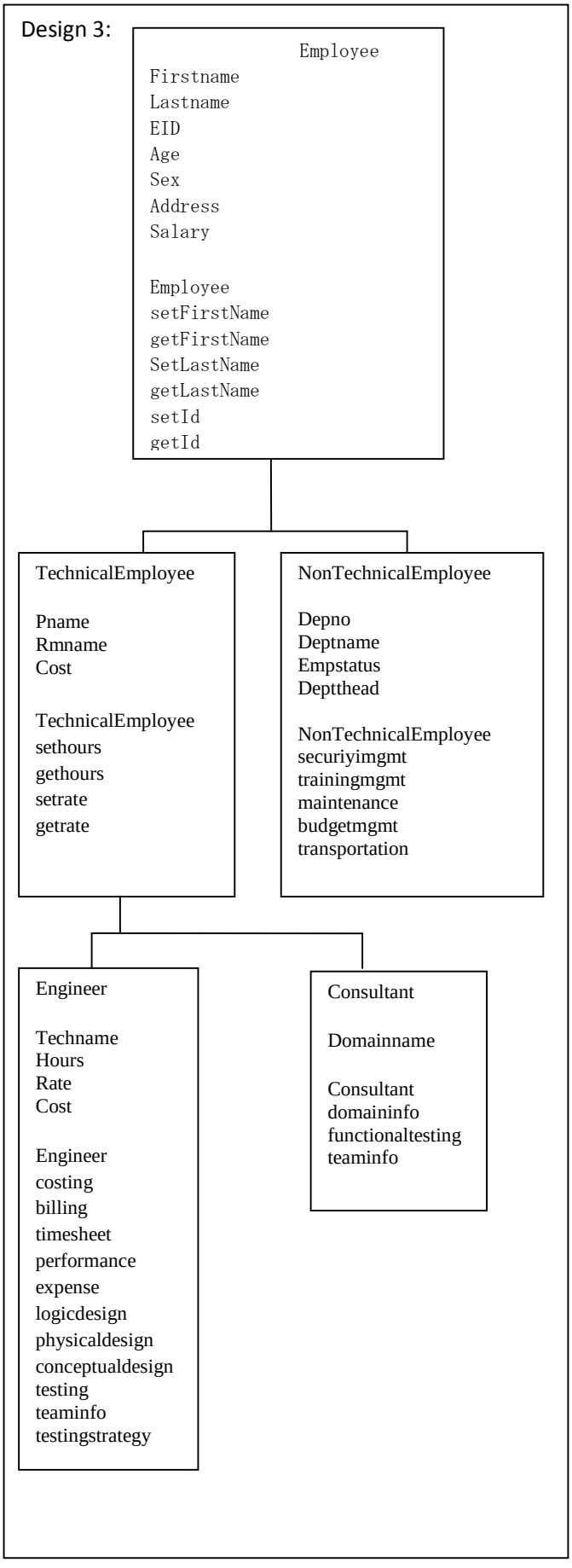
Design 2:



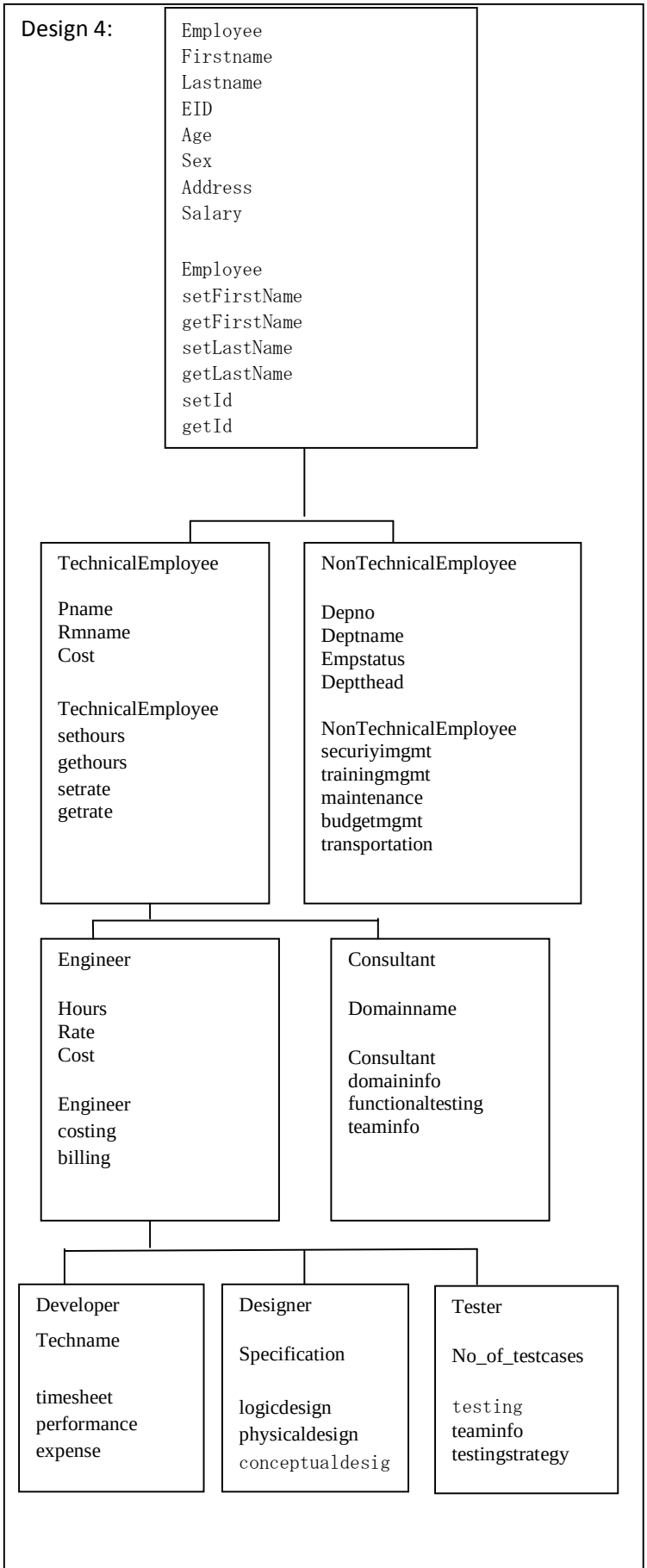
International Journal of Enterprise Computing and Business Systems
ISSN (Online) : 223-8849
http://www.ijecbs.com

Volume 1 Issue 2 July 2011

Design 3:



Design 4:



International Journal of Enterprise Computing and Business Systems
ISSN (Online) : 223-8849
http://www.ijecbs.com

Volume 1 Issue 2 July 2011

4. Results Obtained after evaluating the designs:

Design 1:

Class Name	WMC	DIT	RFC	NOC	CBO	LCOM
Employee	24	0	24	0	0	230

Employee class is too large. It should be decomposed.

Design 2:

Class Name	WMC	DIT	RFC	NOC	CBO	LCOM
Employee	9	0	9	2	0	2
TechnicalEmployee	17	1	18	0	0	130
NonTechnicalEmployee	6	1	7	0	0	0

TechnicalEmployee class is too large. It should be decomposed.

Design 3:

Class Name	WMC	DIT	RFC	NOC	CBO	LCOM
Employee	9	0	9	2	0	2
TechnicalEmployee	5	1	6	2	0	0
Engineer	12	2	13	0	0	10
Consultant	4	2	5	0	0	0
NonTechnicalEmployee	6	1	7	0	0	1

Engineer class is too large. It should be decomposed.

Design 4:

Class Name	WMC	DIT	RFC	NOC	CBO	LCOM
Employee	9	0	9	2	0	2
TechnicalEmployee	5	1	6	2	0	0
Engineer	3	2	4	3	0	0
Developer	3	3	4	0	0	0
Designer	3	3	4	0	0	0
Tester	3	3	4	0	0	0
Consultant	4	2	5	0	0	0
NonTechnicalEmployee	6	1	7	0	0	1

5. Results and Analysis:

When the four designs are evaluated and CK metrics suit is calculated for each design, following points are observed.

1. Design1 is considered as a poor design because Employee class is too large. There are large number of variables and methods in Employee class. The value for the metric WMC is high. As a result the functional complexity of the design is high and it contains too much information and also the value of LCOM metric is too high means methods are less cohesive. One solution to this problem is to decompose the class into several subclasses. It would make the design simpler and more understandable.

2. Design2 is better than Design1 but it suffers from many problems. TechnicalEmployee class is still very large. The value of LCOM metric for this class is very high. As a result this class needs to be decomposed. Simplying the class would make it less complex. However to increase the quality, more decomposition is required and inheritance should be used.

3. When Design3 is evaluated, it scored higher mark as compared to the previous designs. Almost all the factors are within the range but still there is a class Engineer with slightly high LCOM value. Therefore, this class needs to be further decomposed so that it can be more simplified and inheritance should be used more.

4. Design4 is considered appropriate. All its factors are within the range.

6. Conclusion:

This paper assesses the effect of inheritance on the object oriented metrics. Assessment shows that inheritance is a key factor of object oriented systems. When no inheritance is used in designing a software product, the values of object oriented metrics are not within the range and the resulting design is complex and tedious. Although locating the information in one class reduces the depth of inheritance and the number of children, it would however, increase the class size and program complexity. When inheritance is introduced, all the factors of object oriented metrics are within the range and the design not only becomes simpler and understandable but the quality of software product improves a lot.

References:

[1]Sami Mäkelä and Ville Leppänen.”Observation on Lack of Cohesion Metrics”. Proceedings of the International Conference on Computer Systems and Technologies - CompSysTech’06.

[2]Dr. M.P.Thapaliyal and Garima Verma.”Software Defects and Object Oriented Metrics” - An Empirical Analysis. International Journal of Computer Applications 9(5):41–44, November 2010.

[3]Chidamber, S. and Kemerer, C.” A Metrics Suite for Object Oriented Design”, IEEE Transactions on Software Engineering, vol. 20, no. 6, pp. 476-493,1994.

[4]Chidamber, S., Darcy, D., Kemerer, C.” Managerial use of Metrics for Object Oriented Software”: an Exploratory Analysis, IEEE Transaction on Software Engineering, vol. 24, no. 8, pp. 629-639,1998.

[5]Conte, S., Dunsmore, H., Shen, V. “Software Engineering Metrics and Models”, The Benjamin/Cummings Publishing Company, California, USA,1986

[6] Lionel C. Briand, John W. Daly, and Jürgen Wüst: “A Unified Framework for Coupling Measurement in Object-Oriented Systems”. Fraunhofer Institute for Experimental Software Engineering. Kaiserslautern, Germany, 1996.

[7]Pressman R.” Software Engineering: a Practitioner's Approach”: European Adaptation, 5th edition, McGraw-Hill, UK, 2000.

[8] Booch.G,” Object-Oriented Design and Application”, Benjamin/Cummings, Mento Park, CA, 1991.

[9] Dale and H. Van Der Zee, "Software Productivity Metrics -- Who Needs Them” Eurometrics '92 Proceedings, pp. 31 – 43, April 1992.

[10] N. Fenton, "Software Measurement: A Necessary Scientific Basis," IEEE Transactions on Software Engineering, Vol. 20, No. 3, pp. 199 – 206, March 2006.

[11] R. Fichman and C. Kermerer, "Object-Oriented and Conventional Analysis and Design Methodologies: Comparison and Critique," IEEE Computer, Vol. 25, No. 10, pp. 20 – 39, October 1992.