REUSABILITY AND MAINTAINABILITY IN OBJECT ORIENTED LANGUAGES

Suvarnalata Hiremath[1], C M Tavade[2]
[1]Associate Professor, Dept. of CS&E, BKEC,
Basavakalyan, India,
[2]Professor, Dept of E&TC, SIT-COE, Yadrav, India

Abstract

In object-oriented languages, inheritance plays an important part for software reusability and maintainability. The separation of sub typing and inheritance makes inheritance a more flexible mechanism reusing code. Object-oriented programming has been widely acclaimed as the technology that will support the creation of reusable software, particularly because of the "inheritance" facility. In this paper, we explore the importance of reusability and maintainability in object oriented language.
KEYWORDS: Object Oriented programming Language, Inheritance, Software reuse and maintainability.

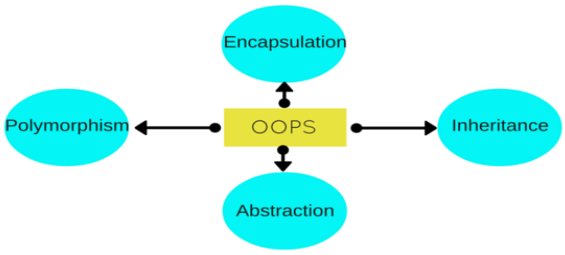
I. INTRODUCTION

Object-Oriented Programming (OOP) is the term used to describe a programming approach based on objects and classes. The object-oriented paradigm allows us to organise software as a collection of objects that consist of both data and behaviour. This is in contrast to conventional functional programming practice that only loosely connects data and behaviour.

The object-oriented programming approach encourages:

- Modularisation: where the application can be decomposed into modules.
- Software re-use: where an application can be composed from existing and new modules.

Object Oriented programming is a programming style that is associated with the concept of Class, Objects and various other concepts revolving around these two, like Inheritance, Polymorphism, Abstraction, Encapsulation etc.



Object Oriented Programming is a practical and useful programming methodology that encourages modular design and software reuse. Object oriented Language make the promises of reduced maintainance,code reusability, improved reliability and flexibility and easier maintenance through better data encapsulation. To achieve these gains, object oriented language introduce the concepts of objects, classes, data abstraction and encapsulation, inheritance and polymorphism.

In object oriented Language, the objects are well-defined data structures coupled with a set of operations, these operations are called behavior and are only visible part of an object and only these operations can manipulate the objects. Each object itself is an instance of one class and is represented by a collection of instance variables, as defined by the class [1]. Each class also defines a set of named operations called methods.A method defines the behaviour of the objects that are created from the class.

Inheritance is a technique that allows new class from older. The new class is subclass and old class is base/parent class. The subclass inherits all features of parent class. The subclass can add new features of their own in it. Inheritance

is used in several different ways. A subclass can be modified to provide different or additional behaviour from its super class. Multiple inheritance allows for a class to inherit traits from multiple classes and is usually considered a dangerous design mechanism.

II Literature Survey

As per the literature survey there has been a lot of research done on reusability and maintainability in object oriented language over the years.

Li XuanDong and Zheng GuoLiang[1] In object-oriented languages, encapsulation and in-heritance play an important part for software reusability and maintainability. The separation of sub typing and inheritance makes inheritance a more flexible mechanism reusing code. The main enhanced flexibility is that it is allowed for subclasses to redefine inherited methods to change their specifications. However, it results in an encapsulation issue derived from the semantics of inheritance, which compromises severely reusability and maintainability in object-oriented languages. In this paper, we present a modified inheritance mechanism, which overcomes this encapsulation issue, and give its denotation semantics. This modified in- heritance mechanism has been introduced in NDOOP, an object-oriented extension of Pascal we are developing.

G. Butler L. Li I.A. Tjandra[2] Software design is a difficult creative task learnt from long experience. Reusable object-oriented design aims to describe and classify designs and design fragments so that designers may learn from other peoples' experience. Thus, it provides leverage for the design process.

Shuguang Hong Barbara Koelzer[3] Software reusability has been regarded as one of the most important areas for improving software development
productivity and quality in the 1990's. The object-oriented approach to information systems development has promised to achieve large-scale software reuse. Object oriented analysis and design methodologies have aimed at the realization of the promised benefits; however, the degree to which the methodologies support reuse and deliver on this promise is an interesting, open question.

Parul Gandhi Pradeep Kumar Bhatia[4] Object-oriented metrics plays an import role in ensuring the desired quality and have widely been applied to practical software projects. The benefits of object-oriented software development increasing leading to development of new measurement techniques. Assessing the reusability is more and more of a necessity. Reusability is the key element to reduce the cost and improve the quality of the software. Generic programming helps us to achieve the concept of reusability through C++ Templates which helps in developing reusable software modules and also identify effectiveness of this reuse strategy

The advantage of defining metrics for templates is the possibility to measure the reusability of software component and to identify the most effective reuse strategy. The need for such metrics is particularly useful when an organization is adopting a new technology, for which established practices have yet to be developed. Many researchers have done research on reusability metrics

Alan Snyder[5] Object-oriented programming is a practical and useful programming methodology that encourages modular design and software reuse. Most object-oriented programming languages support data Abstraction by preventing an object from being manipulated except via its defined external operations. In most languages, however, the introduction of inheritance severely compromises the benefits of this encapsulation. Furthermore, the use of inheritance itself is globally visible in most languages, so that changes to the inheritance hierarchy cannot be made safely. This paper examines the relationship between inheritance and encapsulation and develops requirements for full support of encapsulation with inheritance.

1.1 Outcome of the literature survey

As per above papers we conclude that work done on the inheritance features to maintain more flexible mechanism for reusing code and encourages maintainability in object orientation languages.

1.2 Objective

The main objective of this work is to examine the duplicate and to avoid duplication and to capture commonality in undertaking classes of inherently similar task . Code that is organized so that it is easy to find and fix errors and to improve performance. Set of programming idioms with other programmer which improves communications.

III. Need of Reusability in OOL

One of the promises which object oriented languages holds is that it enhances software reusability. Indeed, software components designed in Object oriented languages are easier to be reused than those designed in conventional programming.

The first thing here we discuss about is Code Reuse. Code Reuse is probably one of the most important part why we use Object Oriented Programming languages in general.

The concept itself looks very easy. You have written a class and some parts of your class like some Methods or Attributes should be reused in another class

Reusability is not an OOP term, it is about reusing existing code to solve problems. This way you don't have to write the same thing twice. Writing the same code twice obviously means that you will spend more time on writing code, but it has another disadvantage. If you want to modify the code, for example fix a bug in the algorithm or extend it with a new parameter, then you have to modify it in two or more different locations. Which means even more typing, and more opportunity to add new bugs to the code. So reusability is a must if you want an maintainable code.

Reusability in OOP achieves through the features of C++ where it possible to extend or reuse the properties of parent class or super class or base class in a subclass and in addition to that, adding extra more features or data members in the subclass or child class or derived class. This whole set of mechanism is known as ***Inheritance***.

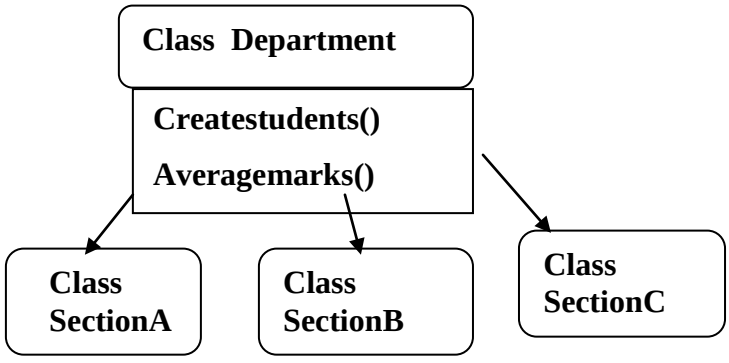
This helps in to use the same code over and again without writing it multiple times which saves time, coding effort and code reduction.

IV. Working with Reusability and Maintainability in OOL

The object orientation programming languages classes can be reused in several ways. Once a class has been written and tested, it can be adapted by another programmer to suit their requirements. This is basically done by creating new classes, reusing the properties of the existing ones. The mechanism of deriving a new class from an old one is called inheritance. The old class is referred to as the base class and the new one is called the derived class or subclass. A derived class includes all features of the generic base class and then adds qualities specific to the derived class.

Re usability is when a class inherits or derives another class, it can access all the functionality of the inherited class.

- **Re usability enhanced reliability**. The base class code will be already tested and debugged.
- **Re usability saves the programmer time and effort**.
- As the existing code is reused, it leads to **less development and maintenance costs.**



Using Reusability , we have to write the functions only one time instead of three times as we have inherited rest of the three classes from base class(Department).So ,maintainability becomes easy and low cost.

V. CONCLUSION

In this paper we discussed how reusability and maintainability has a relationship and an important part for code reuse and also to

examine the duplicate and to avoid duplication and to capture commonality in undertaking classes of inherently similar task On the contrary, the basic idea of our work is that the designers should know nothing about the implementation of a class so that they can define a class by inheritance and maintain a class correctly and easily.

REFERENCES

[1] A Modified Inheritance Mechanism Enhancing Reusability and Maintainability in Object-Oriented Languages Li XuanDong and Zheng GuoLiang Department of Computer Science Nanjing University, Nanjing Jiangsu, P.R.China 210093

[2] Encapsulation and Inheritance in Object-Oriented Programming Languages :Alan Snyder

[3] Open Issues in Object-Oriented Programming: OLE LEHRMANN MADSEN Computer Science Dept., Aarhus University, Ny Munkegade , DK-8000 Aarhus C, Denmark (e-mail: olmadsen@daimi.aau.dk)

[4] Do We Need Inheritance?: Wolfgang Weck ETH Zürich Weck@inf.ethz.ch and Clemens Szyperski Queensland University of Technology c.szyperski@qut.edu.au

[5] Inheritance and development of Encapsulated software components: Alan Snyder.

[6] Inheritance and Modularity in Specification and Verification of OO Programs: Liu Yijing ; Dept. of Inf., Peking Univ., Beijing, China ; Ali, H. ; Qiu Zongyan.

[7]Object-oriented programming concepts: Inheritance **by Michelle Yaiser**