Review Paper on Inheritance and issues in Object Oriented Languages

Suvarnalata Hiremath¹, C M Tavade²

¹ Associate Professor, Dept. of CS&E, BKEC, Basavakalyan, India,
² Professor, Dept of E&CE, BKIT, Bhalki, India,

Abstract

The objective of this research paper is to review concept of inheritance in object oriented languages. The review paper begins upon the survey of inheritance and reusability of object oriented language. Inheritance plays an important role for code reusability. Since object oriented has been widely acclaimed as the technology that will support creation of reusable software, particularly because of the inheritance feature. Then discuss a new approach of inheritance mechanism, which overcomes the encapsulation issues and other issues derived from inheritance, which compromises severely reusability in object oriented language and also we explore the connection between inheritance and code reusability.

KEYWORDS: Object Oriented programming Language, Inheritance, Software Reuse and Encapsulation.

I. INTRODUCTION

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 maintenance, 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 behaviour 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.

In object oriented language, encapsulation and inheritance play an important role for code reuse and maintainability. Encapsulation is the method of combining the data and functions inside a class. The features of encapsulation are supported using classes in most object-oriented programming languages, although other alternatives also exist. Data encapsulation, sometimes referred to as data hiding, is the mechanism whereby the implementation details of a class are kept hidden from the user. The user can only perform a restricted set of operations on the hidden members of the class by executing special functions commonly called methods. The actions performed by the methods are determined by the designer of the class, who must be careful not to make the methods either overly flexible or too restrictive. This idea of hiding the details away from the user and providing a restricted, clearly defined
interface is the underlying theme behind the concept of an abstract data type.

Data Abstraction is a useful form of modular programming. The behaviour of an Abstract data object is fully defined by a set of Abstract operations defined on the object; the user of an object does not need to understand how these operations are implemented or how the object is represented[2].

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.

This paper is based upon the survey of inheritance and reusability of object oriented language. This paper examines the open issues of object oriented language and connection between inheritance and code reuse in Object Oriented Language. We begin by reviewing the concepts of inheritance and reusability and then review the concepts of encapsulation and demonstrate how inheritance and code reusability had a connection.

1.1 Literature Survey

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

Li XuanDong and Zheng GuoLiang, [1] described the separation of subtyping 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.

Alan Snyder [2] Introduced 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 and examines the relationship between inheritance and encapsulation and develops requirements for full support of encapsulation with inheritance.

OLE LEHRMANN MADSEN [3] discussed a number of open issues within object-oriented programming. The term object-oriented programming is meant to cover: analysis, design implementation, data modeling in databases, and distribution. The issues being discussed in this paper are: modeling versus reuse as the main benefit of object orientation; the need for a language independent conceptual framework; abstraction mechanisms for supporting object-oriented programming including classification and composition, single and multiple inheritance, inner versus super, genericity versus virtual classes and issues related to typing; class-based versus prototype-based languages; and concurrency.

Wolfgang Weck and Clemens Szyperski [4] suggested to abolish inheritance and possibly subtyping, at least in its present form. A concrete research plan involves definition of an experimental programming language, various design patterns will be expressed using this language and the language could be implemented and used to construct some real world frameworks.

Alan Snyder [5] identified two areas where most object oriented language are deficient in their support for encapsulation. One area is encapsulation of instance variables and other area is the visibility of inheritance itself.

Liu Yijing Ali, H. ; Qiu Zongyan [6] proposed a novel specification and verification framework, which supports abstraction and offers modularity via a set of scope and inheritance rules, and a concept called emph{specification predicate}.


1.2 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 modular design and software reuse.

II. Objective

The main objective of this work is to examine the interaction between inheritance and encapsulation and explore various ways in which implementation details can be exposed via inheritance. This work uses a different technique which allows us much flexibility as possible as desirable.

III. Methodology and Expected Outcomes

Object Oriented Programming is a practical and useful programming methodology that encourages modular design and software reuse. Object oriented Language makes the promises of reduced maintenance, code reusability, improved reliability and flexibility and easier maintenance through better data encapsulation .To achieve these gains, different methodology can be used as delegates, framework, merging instance variables, design patterns and overriding.

IV. CONCLUSION

In this paper we discussed how encapsulation and inheritance play an important part for code reusability and maintainability and demonstrate how inheritance and code reusability had a connection. A new approach of inheritance mechanism, which overcomes the encapsulation issues and other issues derived from inheritance, which compromises severely reusability in object oriented language. 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.


[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 University of Technology c.szyperski@qut.edu.au.


[8] Practical Use of Encapsulation in Object-Oriented Programming: Mats Skoglund Department of Computer and System Sciences Stockholm University/Royal Institute of Technology Kista, Sweden matte@dsv.su.se