Neuro Object Oriented Programming Approach and Design

Prateek Chaudhary

Pursuing MCA, IP University

Abstract- Our object oriented programming approach have great ability to improve the programming behavior for modern system and software engineering but it does not give the proper interaction of real world .In real world , programming required powerful interlinking among properties and characteristics towards the various objects. Basically this approach of programming gives the better presentation of object with real world and provide the better relationship among the objects. I have explained the new concept of my neuro object oriented approach .This approach contains many new features like originty , new concept of inheritance , new concept of encapsulation , object relation with dimensions , originty relation with dimensions and time , category of NOOPA like high order thinking object and low order thinking object , differentiation model for achieving the various requirements from the user and a rotational model .

Index Terms- Originity, F-inheritance, NF – inheritance, Acquisition, Differentiation, Dimension, Weak coupled, Tightly coupled, State, Stationary state, Motion-able state, Global Entity Local entity, Main entity, Impulsive entity , Interface entity, Controlled entity, Modularity, Random Nature , Public permission, Private Permission

I. Introduction

TIt is the process of defining and designing the object on the basis of their unique characteristics and properties.

Example = consider a simple family relationship ,

```
                    let a family consist two children , mother and father
                      then
                        let a Class Father Family              let a Class Mother Family
                   {                                      {

                          Data types                             Data types
                              Functions                            Functions
                      .                                      .
                      .                                      .
                      .                                      .

                           }                                                }

                          let a Class One child               let a Class Second child
                   {                                     {

                          By default some characteristics        same like one class
                          from mother and father class
```

Characteristics of object means , How object reacts , How single object behaves with respect to the other objects on the basis of certain functions and with variables and followed the vice versa.

In this approach , the object generally describes the knowledge entities are expected to acquire or construct.

Characteristics consists special features of object like movement of object and state of object.

<u>Movement of Object</u>

" Reaction of object towards other object and interaction with other objects".

<u>State</u>

These are those points or persist properties of object when they occur under certain period of time.

<u>(=State of object are divided into two category=)</u>

<u>Motion able State</u>

When object are ready to interaction with other objects under some circumstances .

This is simply like linking of multiple objects through global class.

```
                                        +
                              Additional Properties of child also
                              including with the existing properties



                                    }                       }
```

Stationary State

When object does not ready to interaction with other objects under some circumstances.

This basically define the objects behavior of multiple objects with Local classes.
Example = consider a predefined limited algorithm behavior based Class

```
Class Car
  {
     .
     .
     .

  }
```

Object :

These are those type of unique entity which change their properties on the basis of other object that why these are divided into four main category.

Properties of Object

These are those type of behavior of objects that define object constraints with respect to surrounding.

Basically, its abstraction of important features of objects inside the system . properties contain almost all the object oriented programming concept

***Originity**
***Inheritance**
***Encapsulation**
***Modularity**
***Class**
***Polymorphism**
***Classification**
***Generalization**
***Specialization**
***Composition**
***Random nature**

II. ORIGINITY

Originity => refer to way , "How to find the objects from the real world under the certain period of time with the collection of limited resources.".

Its completely depends on the number of dimensions which is define for objects.

On the basis of the dimensions , we can easily find as much as large number of object within a certain time period.

originity of object is directly proportional to the n-number of dimensional and also proportional to time period required for finding the objects.

According to dimensional analysis on the basis of originity , " the originity of object is directly proportional to the number of dimension present in the system and inversely proportional to the time period required to finding the object with respect to surroundings ".

$$\textbf{Originity} \;\alpha\; \textbf{Dimension}$$
$$\textbf{Originity} \;\alpha\; \textbf{Time}$$
$$\textbf{Originity} \;\alpha\; \textbf{Dimension} * \textbf{Time}$$
$$\textbf{Originity} = \textbf{K Dimension} * \textbf{Time}$$

Where K , is proportionality constant and its value depends on complexity ratio around the objects

Complexity ratio depends on two main properties

# **MICRO Complexity ratio** => One , Two and Three Dimensions

# **MACRO Complexity ratio** => For more than Three Dimensions

*ORIGINITY is the aspect of created or invented works by which new object being created and also provides differentiated features by which cloning of object can be stopped.

The term originity is often applied as a compliment for objects creation and helps the object its properties processing.

III. OBJECT RELATIONSHIP WITH DIMENSIONS

According the Originity behavior , " dimensions of system is directly proportional to the number of objects with respect to their surroundings .

$$\textbf{Dimension} \;\alpha\; \textbf{Object}$$
$$\textbf{Dimension} = \textbf{K Object}$$

Where K , proportionality constant and its value depends on
If no dimension then , K = infinity
If One dimension then , K = minimum =1 to infinity

$$1 \le K \le \infty$$

If Two dimension then , K = 2

$$2 \le K \le \infty$$

General formula ,
For n – dimensions

$$n \le K \le \infty \qquad n \in N$$

N = natural numbers

Advantage of Originity

- Help in Object finding.
- Time saving process .
- Efficient and simple in nature.

IV. INHERITANCE

"The process of transferring the information from existing object class to the new one object class ."

It's a simple parents and child relationship in which child inherits the properties of their parents.

This is the most powerful feature of NOOP on which the parameters of existing object transfer to the another object class.
In NOOP, the inheritance are divided into two major form

F-Inheritance (Filter Inheritance) = These are those type of inheritance , in which it helps in removing the ambiguity occur during the transfer of information on two or more object classes.
This type of inheritance avoid the transfer of those properties , which cause ambiguity in classes.

Example = This can be done by removing the both variable from the two or more class and make a global variable which can use for both classes.

NF-Inheritance (Non Filter Inheritance) = These are those type of inheritance on which ambiguous features transfer from one existing object class to another existing object class.
This concept based on the old concept of normal inheritance.

(Way of transferring the information and data from one object class to other object class.)

- **Single Inheritance(same as previous)**
- **Multilevel Inheritance(same as previous)**
- **Multiple Inheritance(same as previous)**
- **Hierarchical Inheritance(same as previous)**
- **Hybrid Inheritance(same as previous)**
- **Star Inheritance(new concept)**

Single Inheritance = (Derived class)One or more than one class inherits the properties of base class.
Note : Base class act as a parents .
Derived class act as a Child.

Multilevel Inheritance = it's a continuous process of transferring the properties of one class to another class with n-1 level.
$n \leq 1 \leq \infty$

Multiple Inheritance = In this type of inheritance , any class inherits the properties of two or more classes.

Hierarchical Inheritance = In this type of inheritance , more than two classes inherits the properties of Base class.

Hybrid Inheritance = It's a combination of multiple and multilevel inheritance .

Star Inheritance = This is special type of inheritance which describe the **macro complexity ratio** .

Star inheritance is used for F-inheritance .

Strictly prohibited , for NF-inheritance(Reason star inheritance consists a large number of possibilities of occurrence of ambiguity) .

In star Inheritance , star class properties with their object act as a intense or power object which act as a base object class for all the other subclasses.

Advantage of using star inheritance

1.Make more interconnect-able class.

2.if we are using F-inheritance , then its very powerful and efficient because its uses the limited amount
Data types for variables.

3.Provide fast execution of code system.

4.Secure as compare to other terminology and worked very well if one class does not work (avoid the central class).

V. ENCAPSULATION

In class simply a grouping procedure of behavior , properties and characteristics .

It defines the behavior of class on the basis of their properties and characteristics .

In programming , properties and characteristics are like data types and functions.

Choosing of properties(data types and functions) are like behavior of object.

It reduces the complexity nature of data types with functions.

The need of encapsulation lead to two basic requirements

# The basic need to cleanly differentiate between the specification and the implementation of an operation .

# Due the need of modularity .

Specification = define object structure and help in implementation of object behavior.

It makes code system more readable and compact .

There are three view of Encapsulation
# Programming language view
# Database adaption view
#Combination of programming and database adaption view

In programming language view , an object have two part

- **Interface view**
- **Implementation view**

Interface part = The interface part is the specification of the set of operation that can be performed on the object . it is the only visible part of the object.

Implementation view =implementation part has a data type and a procedure part .The data part is the representation or state of the object and the procedure part describes , in some programming language the implementation of each operation.

Database adaption view = In this view , the object encapsulation both program and data.

In the database world , it is not clear whether the structural part of the type is or is not part of the interface .This is depends on the system , while in the programming language world , the data structure is clearly part of the implementation and not of the interface.

VI. `

Modularity is the procedure of dissociating the nature of objects into n – number of objects nature.

This dissociation of object is completely depends on the behavior of objects with their characteristics.

Basic idea behind the modularity is to break the complexity of object.

Dissociation helps in extraction of object properties.

Dissociation also helps in finding the weakly attached and tightly attached constraints of object under certain period of time.

Modularity are defined on the basis of coupling behavior of objects

Coupling define the interconnection among various objects .

Coupling are divided into two category

# Weak coupled = By finding , the weak common features of object , then we can easily create the differentiation in properties of objects.

Example = simple relationship of uncle(mama , chacha) relationship with his/her brother son/daughter and if it's a thinkable object then firstly check their surroundings group of objects.

If these are thinkable then apply the thinkable approach otherwise apply the properties of non- thinkable object properties.

# Tightly coupled = By finding , the most linkable properties of the object , this is very difficult for programmer to find the tightly properties of objects.

Example =simple parents and children relationship .

Main aim of modularity

Polymorphism = > (same as previous)
Generalization => (same as previous)
Specialization = > (same as previous)
Composition => (same as previous)

VII. RANDOM NATURE

Random nature =>refer to the phenomena by which object of other class can handle each and every properties and characteristics of every other object with limited resource's under the organic object oriented approach .

Basically , it act as a global object which have ability to perform each and every action of other object.

- Random nature of object makes its nature more complex for other object , because other object inherits its all properties and characteristics .
- Random nature of object divided into two form :
- **1. Global entity (Global object)**
- **2.local entity (Local object)**

Global object = are those type of object which contain all categories of object like

1. Main object
2. Interface object
3. Impulsive object
4. Controlled object

And perform each and every action of other object using unique state , properties and characteristics of other object.

It provides the convenient way to use the same variables and functions of the other object.

Global object are always public in nature when they access the data types and functions of other objects.
Global object act as a super object of all other object classes.

For instance =

Grand parents
!
Parents
!
children

all the unique and best feature of grand parents comes into parents and it may also be possible some or all feature of grand parents comes in children or sub child of children.

- Grand parents has ability to access all the properties of parents and children with some certain rule .but parents and children does not have right to give any permission to do any work to its parents or grand parents.
- Similarly , children does not have right to give permission to its parents .
- Global object always contain two type of access permission

- **1. Public permission**
- **2.Protected permission**

Public permission = In this permission , object are able to share their properties and behavior with other object classes.

Protected permission = In this permission , object only shared specified properties and behavior with other object under certain rule.

Random nature give us a functionality to handle every object class with free nature behavior .

Rules in random nature

- Global object must inherits all necessary characteristics of some other classes.
- If any properties of global object and child object property intersect , then property will be used or we'll give priority to global object.
- Data type values and variable will always get the prior position in object child object.

Advantage of using the random nature

1.improved performance => because global object handle all the functionality of child object so execution of different -2 process related to child object occur fastly.

2.Removal of ambiguity => because we following the rule of global object under random nature ,so ambiguity automatically reduces.

3.better inheritance

4.reduces the complexity of the local program.

References

[1] Help of Prof . Pallavee joshi and Prof . Priti khtri .
[2] Reference book : Ivar Jacobson OOAD book.

Authors

First Author – Prateek Chaudhary, (Pursuing MCA), IP University, April 2013