Object-Oriented Analysis and Design

Momin Mukherjee

International Journal of Advanced Engineering and Management
Vol. 1, No.1, PP. 18-24, 2016
Received on: 28.11.2016
Revised on: 20.12.2016
www.ijoaem.com

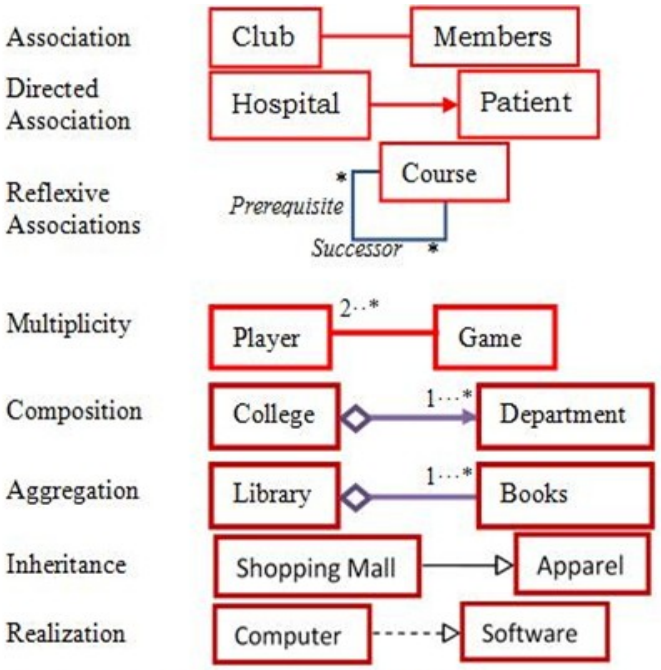
Object-Oriented Analysis and Design

Authors: ***M. Mukherjee***
International Journal of Advanced Engineering and Management, India
e-mail: editorijoaem@gmail.com

Subject Category: Management
Sub Category: *System Management*

Editor: ***Sahadev Roy***

Volume 1 Issue 1 December 2016



Popular symbolic representation in UML class diagrams

Keywords: *Activity diagram;*
Activities performed in object-oriented analysis;
Object-oriented analysis;
Object-oriented design;
UML Diagram.

IJOAEM/3/1/1/3

Object-Oriented Analysis and Design

M Mukherjee

Abstract

This study mainly focuses on how object-oriented analysis makes compatible with newly develop or other existing business computing application in a better way. This study also focuses on the modelling of the exact procedure or near to the exact procedure within its application domain which may model by using different objects class. Objects are basically structured into different classes of objects which are generally related to behaviours and characteristics. These methodologies may use different generalization, classification, and different aggregation as a structure object assemblies for the target actions like services or activities which are related to the objects. There are numerous misconceptions related to object oriented analysis which are required to address when we consider the use of any object-oriented method. In this paper, try to represent different advantages and various application of the UML. The platform presented here is a comprehensive range of the different UML templates with all other required information.

1. Introduction

Object-oriented analysis and design (OOAD) is a globally accepted technical process for manipulative an application specific, business or system model, and simple graphical diagram for analyzing and product quality improvement by applying the object-oriented prototype method [1]. Object Oriented Analysis (OOA) is basically collected works of concurring or cascading system modelling, incorporate various requirements and pre and post analysis methodology for software systems. These methodologies are primarily influenced by different object-oriented programming, data modelling and systematic interconnections [2]. Fundamental idea behind OOA is a streamline software design and development by considering all model as a discreet objects, classes, methods and by linking these one can designs and implement all kind of business requirements [3]. It mainly focused on the system development and analysis. This study also focus on how it make compatible with newly develop or other existing business computing application in better way. In the late 80‘s and 90‘s there were many different object-oriented methods and modelling techniques used what limited sharing of models across projects (reduced reusability) and hampered communication between team members and users. The main objective of any model is to build a distinct, globally accepted standard procedure for object-oriented systems software [4]. Unified Modelling Language (UML) is one of the standard widely accepted languages, generally is used for modelling any system considering as objects for better analysis [5]. Such techniques mainly focus on the modelling of the exact procedure or near to the exact procedure within its application domain which may model by using different objects class. These methodologies may used different generalization, classification and different aggregation as a structure object assemblies for the target actions like services or activities which are related with the objects. It can be implemented by MATLAB also [6]. The major difficulty of preventing object oriented code and reuse code by different attackers is a major challenge [7]. State changes may be affected the actions which are performed by that objects. There are numerous misconceptions related to object oriented analysis which are required to address when we consider the use of any object-oriented method.

2. Some Familiar Terms Related to OOA

a) Object

It may sensed by one or more by our sensing organs like vision, touched, etc. or by which users able to store data and correlate its behaviour.

b) ***Attributes***

This is basically all the information and different types of data which are used to describe the object characteristics of interest.

c) ***Behaviour***

It is basically refers to as a methodology, procedure, operation or service that the object able to correspond to any functions which may be operate on the object‘s attributes or data.

d) ***Inheritance***

It may consider as general model or methods which may be reused by another object class intrinsically.

e) Encapsulation

It is basically binding or packaging of a number of items and operation together into a single unit which is also known as to as the information biding. Different attributes and characteristics of the target object, both are both are considered as the part of the target object and also packed together. The only possible technique to change or access any attributes by changing specified behaviours of that object‘s which may consider as different object.

f) ***Class***

It is basically a collection of different objects or objects set which are share their behaviour and attributes. It is also known as object class.

g) ***Generalization/specialization***

It is basically a process or technique for widely reuses of inheritance, where the common attributes and characteristics of several object classes are merging or clubbing together to form a general class. Sometimes it referred as generalization class. The particular or set of methods and attributes of the mother object class are known as specialization.

h) ***Object/class relationship***

In general business organization may exists different relations among one or more objects or classes. Many developers consider the class diagrams are complicated ones.

i) ***Multiplicity***

It described how different instances of the particular object or class may associate with particular instance of the other object or class.

j) ***Aggregation***

It may describe as a special kind of relationship, which shows that some objects or classes are prepared by using other objects or classes. By recognizing proper aggregation and interactions, one can able to separate any complicated object by assigning different characteristics and attributes to the individual class or objects. There are two types of aggregation relationships: (i) composition aggregate relationships among object classes all part-objects make up and live in the whole-object, (ii) shared aggregation relationships, which imply that parts may be shared with others.

k) ***Message***

The message required to pass mainly when one of the objects requested for information from the one or more of different objects for some particular action or a set of actions [8].

l) ***Polymorphism***

It generally used for ―many forms‖ which are mainly applied any object-oriented process. The same attribute, behaviour or characteristics are accomplished by other objects or classes. What is important in message sending is that the requested object or slave object previously knows what type of services to request come and from where.

3. UML diagrams

UML generally offers a graphical model of a system by using different groups based on its functionality. All UML diagram or models prepared by the development expert are based on the various perspectives of the data and information system [9]. The different types of UML diagrams and their objects are briefly discussed here:

(i) Use Case Diagrams: By using various type of graph, the structure illustrate the connections among the internal systems and also different external systems along with end users [10]. It also graphically explains where this model use and the various ways the end user may interact with the target system.

(ii) Class Diagrams: They represent the fundamental structure of the system‘s objectives. This may show how object classes is smartly composed and maintain both inter and intra relationships between different object classes purposefully.

(iii) Object Diagrams: It is quite analogous to the class diagrams but instead of describe the object classes it replicate the actual object characteristics.

Sequence Diagrams: It illustrates graphically how different objects may interact or communicate between them via passing various messages in the time of execution for any use case type operations. It also describe how messages are transmitted and how it received maintaining the predefine sequence.

(v) Collaboration Diagrams: It is analogous to the sequence diagrams only the difference it is not follow timing and progression of messages. It may used to represent the inter communication between different objects in the network.

(vi) Activity Diagrams: It is basically used for graphically modelling illustrates the sequential dataflow of various activities may be business progression or any use case.

(vii) State Diagrams: It is used for modelling the various dynamic behaviour of any particular object or class. It may also describe a particular object‘s life cycle. The different states of the object may be considered as events which cause state transaction.

(viii) Component Diagrams: This diagram frequently used for graphically describe any system physical architecture.

Deployment Diagrams: It is used for represent the physical architectures of the both software and hardware of any particular system. Different popular symbolic notations are used in UML class diagrams are shown in Fig. 1

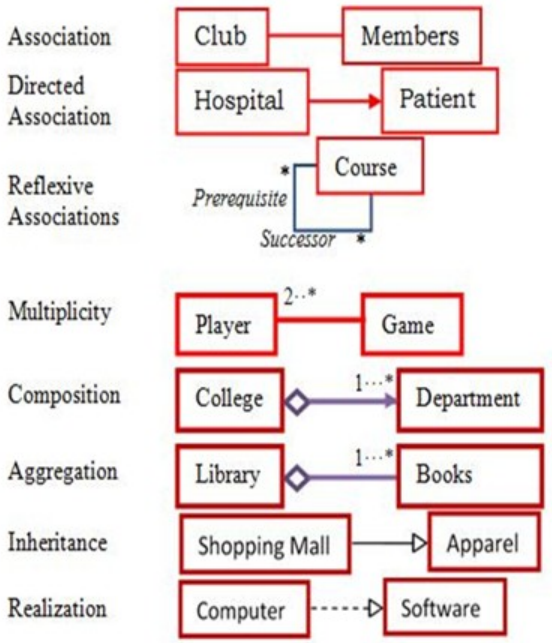


Figure 1. *Popular symbolic notations are used in UML class diagrams.*

4. Activities Performed in Object-Oriented Analysis

There are four general activities are essential for any object-oriented analysis which are briefly discussed here.

a) *Functions Modelling Techniques*

These processes are usually used for modelling of the functional characteristics of the target systems [11]. The Use Case modelling basically is the common method of any kind of system‘s functions modelling related to any business events. It basically correlated between all events creators, and also how the system provides responds for any particular events. Usually the Use Case behaviourally correlated all sequence of scenario or steps among the both automated and manual procedure for implementation a particular business task. This may triggered or initiated by any external users or by any systems, popularly known as Actors. The Actor may be representing anything which are required to interact or communicate with the system for information exchange. In many business events related information systems are automatically triggered by the particular date and time which is known as temporal event.

The Use Cases structure have few advantages like (i) it help to identify different objects easily and also their different responsibilities and high-level relationships, (ii) it may provide a clear view of the system performance, (iii) it may use an helpful tool for validating different requirements, (iii) it may use as an effective communication tool and as a user‘s guide or manual.

Steps involved in use case modelling are (i) identify different factors and all type of use cases, (ii) required to construct a diagram using simple graphically model by using Use Case. It simply illustrate different system scope and its boundaries, which characterized the required relationships among the factors and the Use Cases for the all business subsystems, (iii) document the use case course of the events – only general information about the business event (typical and alternative courses), which is called requirements use case, (iv) define the analysis use cases – more information regarding each use case, which specifies the systems functionality in detail but without any

implementation details. When the use case contains complex functionality we can simplify it by extracting the more complicated process for their own use cases, which are called extension use cases; an extension of the Use Case which may be call upon only by it is extend use case. Sometimes there may be use cases performing steps of identical functionality. They can be than extracted into separate use case of its own, which is familiar as abstract use case. It represents a form of ―reuse‖ and may be further used for further Use Case structure which may required its particular characteristics or functionality.

b) ***Finding and identifying the business objects***

Steps involved in identifying and finding business objects for object modelling: (i) find the potential objects – the best way is to review and reanalysis each and every use case to locate nouns which correspond to all business events or entities, (ii) chose the planned objects and prepare a list contains all potential business entities. The list time to time must be cleaned up by removing: unclear nouns which focus on really actions or attributes, nouns which is outside the scope of the desired system, nouns without any unique activities or any external roles and also all synonyms.

c) Organizing the objects and identifying their relationships

A class diagram is mainly used for graphically illustrate all the identified objects and also their associations and relationships. In these graphical diagrams, user also able to include multiplicity, associations, aggregation of generalized and specialized relationships.

Steps generally use to construct any class diagrams are discussed here:

(i) Identify associations and multiplicity: association between any two objects or classes are required to know about the each other; to help insure that all possible relationships are identified we can create an object/class matrix.

(ii) Identify generalization/specialization relationships: we should look for all one-to-one multiplicity relationships between objects because they may be gen/spec relationships as well as for objects that have common attributes and behaviours.

(iii) Identify aggregation relationships: we must remember that aggregation relationships do not imply inheritance. The object, which is the part of another object does not inherit attributes or behaviour commencing the whole object), but they propagate behaviours. The behaviours which are applied to the entire system are automatically applied to the each and every part of the sub-systems and sub-modules.

(iv) Prepare the class diagram as per requirement.

d) ***Modelling the behaviour of the objects***

All object have a state, which is the significance attributes at any time instance. An object may change it state when something insists to changes or when it‘s any attributes change. These changes in any state may triggered by an external or internal events [12]. Any **state diagram** is the models of a single object life cycle. It may illustrate the various states of an object may have. Events causes ‘the state change of the target object by some predefine rules that direct the object‘s state transition. State diagrams are not essentially required for all objects or class. Typically a state diagram may assemble only for individuals ‘objects that undoubtedly have particular states and may have composite behaviour.

5. Discussion

Using object-oriented analysis, it may possible improved reliability and flexibility, code reusability, reduced maintenance cost and also real world modelling. Though, practically few users realize that the OOA different benefits may not as persuasive as original. The OOA codes may reuse in comparison with other programming language. Basically code reusability depends on how the system is defined and also a subjective thing. This method does have the capability to minimize few major operating expense related with the model or structure, such as system may essentially required maintenance and also continues development of the general programming template and code data structures. We discuss here few common benefits of this approach:

a) Real-world modelling

Objects are basically structured into different classes of unique objects and generally correlated or associated with all behaviours and characteristics. Object-oriented systems are able to model any type of business organization or service provider as like as the real world. It generally models the whole system in a more complete fashion in comparisons with other traditional approach.

b) Maintenance cost reduction

One of the desire goals of any object-oriented system developments are must have a larger life span at

reasonable or very low maintenance costs. Since almost all of the processes are encapsulated, the behaviours can reuse and it may incorporate into new characteristics or behaviours.

c) Code Reusability

Code reusability is another desired advantage of any programmer. When any new object or class is designed, the characteristics and data attributes of the target class is automatically inherited. The new created object may also able to inherit all the super-classes data and behaviours in which it is participate. When an end user design a new widget, the new design object may acts as "wigitty", also it has new characteristics or behaviours which are quite similar with parents system.

d) Reliability and flexibility improvement

In general object-oriented systems are assured its reliability than other traditional systems, above all due to new behaviours or characteristics possible to generate from the existing objects or classes. These can be randomly and dynamically called and also accessed. It is also possible to create a new object at any time as per requirement or as per demand. The new objects can be able inherit all type of data and their attributes from one, or many other class or objects. All such behaviours and characteristics may be obtained or inherited from the parents 'class or super-classes, and any or collective novel characteristics may be enriched by adding without affecting or changing the existing systems or its any functions. Object-oriented Development is not considered as a technology and object; it is not yet completely accepted by all the major vendors globally. Although many organizations are try to promote object-oriented systems. Another challenge is less number of qualified programmers and DBA's available in this field. When one tries to investigate the common recognition of the object-oriented systems in any organization or any commercial marketplace, it discovers that most of the managers see that an technology approach as computer applications, but they not train their staffs this methods.

6. Conclusion

For any systems designer or programmer, building or analyzing class diagrams are essential after the construction of different blocks of OOA. As presented in this paper, all type of class diagram relations are very simple and easy to realize. As a trial and error methods, design class graphical diagrams in very simple possible ways may allows quite simple to understand and easily realized by programmer and also end user. For this target, designer must put label to all classes or objects and different interaction as elaborative as much feasible. Lastly, the object model or class models develop as like as the real time systems it must indicate the system change. This may imply that we don't require putting all small detail in the first draft design. Generally all the model classes or objects are interlink or interfaces and all required relationships which are essential for the system or graphical application design may ultimately materialize as this advance process moves forward. For the developer job easier, one can able to tally the online diagram, various application and examples offered in the different web site. Additionally, different platform also supports various type of collaboration and able to integrate into others to keep the new diagrams proper documented and updated for all. This tool must greatly development initiatives and also helpful for any organization to meet their targets. Object-oriented development is one of the best solutions for any interactive static as well as dynamic environments. Many large-scale object-oriented organizations are still under development and much simple information are still required for systems applications.

References

[1] G Castagna, "*Object-Oriented Programming A Unified Foundation*." Springer Science & Business Media, 2012.

[2] A Igarashi and N Kobayashi, "Resource Usage Analysis," *ACM Trans.on Programming Languages and Systems,* vol. 27, no. 2, pp. 264-313, 2005.

[3] S J Gay, V T Vasconcelos, A Ravara, N Gesbert, and A Z Caldeira, "Modular Session Types for Distributed Object-Oriented Programming," *ACM Sigplan Notices*, vol. 45, no. 1, pp. 299-312, 2010.

[4] N Thota and R Whitfield, "Holistic Approach to Learning and Teaching Introductory Object-Oriented Programming," *Computer Science Education*, vol. 20, no. 2, pp. 103-127, 2010.

[5] Jim Conallen, *Building Web Applications with UML*.: Pearson, 2002.

[7] Richard D Neidinger, "Introduction to Automatic Differentiation and MATLAB Object-Oriented Programming," *SIAM Review*, vol. 52, no. 3, pp. 545-563, 2010.

[8] F Schuster et al., "Counterfeit Object-Oriented Programming: On the Difficulty of Preventing Code

Reuse Attacks in C++ Applications," in *IEEE Symposium on Security and Privacy*, 2015, pp. 745-762.

[9] S Chaki, S K Rajamani, and J Rehof, "Types as Models: Model Checking Message-Passing Programs," *POPL, ACM SIGPLAN Notices*, vol. 37, no. 1, pp. 45-57, 2002.

[10] M Dezani-Ciancaglini, D Mostrous, N Yoshida, and S Drossopolou, "Session Types for Object-Oriented Languages," *ECOOP, Springer LNCS*, vol. 4067, pp. 328–352, 2006.

[11] H G Baker, "Use-onceVariables and Linear Objects: Storage Management, Reflection and Multi-Threading," *ACM SIGPLAN Notices*, vol. 30, no. 1, pp. 45-52, 1995.

[12] K Honda, N Yoshida, and M Carbone, "Multiparty Asynchronous Session Types," *POPL, ACM SIGPLAN Notices*, vol. 43, no. 1, pp. 273–284, 2008.

[13] M Fahndrich and R DeLine, "Type states for Objects," *ESOP, Springer LNCS*, vol. 3086, pp. 465-490, 2004.