International Journal of Advances in Engineering Research http://www.ijaer.com

(IJAER) 2012, Vol. No. 4, Issue No. V, November **ISSN: 2231-5152**

A STUDY OF OBJECT ORIENTED ANALYSIS AND DESIGN

GARJE RAKESH RAMESHRAO
RESEARCH SCHOLAR, DEPT. OF COMPUTER SCIENCE
CMJ UNIVERSITY, SHILLONG, MEGHALAYA

INTRODUCTION

Object-oriented Analysis and Design is a new way of approaching the solutions to problems related to the programming world. This concept is an evolutionary concept and offers a refreshing change from the obsolete programming concepts, such as modular programming (in modular programming, a program is divided into small units of code, called “modules”).

In this new approach, models are organized around real world concepts. As we know that in the earlier programming concepts, the stress was upon solving the problem. The flaw in that approach was, no reusability of code was considered. All such flaws have been removed in the new approach – “Object-oriented Analysis and Design”.

In the OOAD, the central concept is the object, which constitutes the attributes (properties) and the behavior (operations). Any object can be thought of as an entity comprising of data structure and behavior. OOAD is useful for understanding the problems, modeling enterprises, designing programs and databases and last of all, preparing documentation.

The term “Object-oriented” actually means that we organize the software as a collection of discrete and distinct objects that incorporate both data structure and behavior related to objects. This is contrasted with the approach followed in conventional programming techniques, in which the data structure and behavior are connected in a loose fashion. Object-oriented is an innovative approach of visualizing based on generalization and abstraction that exists in the real world.

OBJECTS AND CLASSES

- Objects: It is defined as a concept or thing with certain boundaries and meanings for the problem under consideration.

  They are basically employed for serving two purposes: -

  1. Promote understanding of the real world.
  2. Provide a practical basis for computer implementation.

- Classes: It describes a group of objects with similar properties (characteristics), common behavior, common associations with other objects and common logic. E.g. – person, animal, furniture are all object classes. Therefore, by grouping objects into classes one can abstract the problem, giving object modeling the power and ability to generate from few specific cases to a chain of similar cases.

- Object diagrams: These are entities that provide a formal graphical notation for modeling objects, classes and the relationships among one another. These are concise, easy to understand and are useful for abstract modeling and designing of actual programs. Object diagrams are basically of two types, that are: -

    1. Class diagram: A pattern or a structure that is used for describing many possible instances of a data. In other words, they describe object classes.

    2. Instance diagram: A pattern that describes how a particular set of objects relates to one another. It basically describes object instances.

- Attributes: An attribute is a data value that is held by the objects in a particular class, such as name, age and weight are attributes of person objects. Each attribute has a value for each object instance. E.g. – the attribute age has value "28" in object James Martin. Moreover, each attribute name is unique within a class and thus different classes may each have an attribute with the same name. For e.g. class person and Class Company may each have an attribute called "address". Also, each attribute name can be followed by optional details such as type and default value.

- Operations: An operation is a function that may be applied to the various objects in a class. All objects in a class share the same operations. E.g. – open, close, hide and resize are all operations on class WINDOW. Associated with each operation is a target object, so that the right object class is accessed and hence there is right implementation of the operation. The same operation may be applied to many different classes. Such type of operations is called, "Polymorphic" i.e., those that take different forms in different classes. An operation may have arguments in addition to its target object even though these do not affect the choice of method. Rather, they parameterize the operation.

  E.g. As shown in fig. 1, the class PERSON has attributes name and age operations change-job and change address.

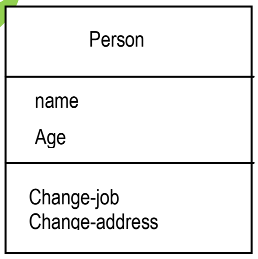


Figure 1

Similarly, FILE has a print operation while GEOMETRIC OBJECT has move, select and rotate operations (as shown in Fig. 2)

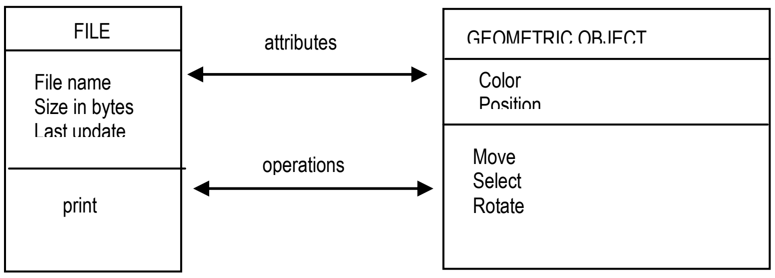
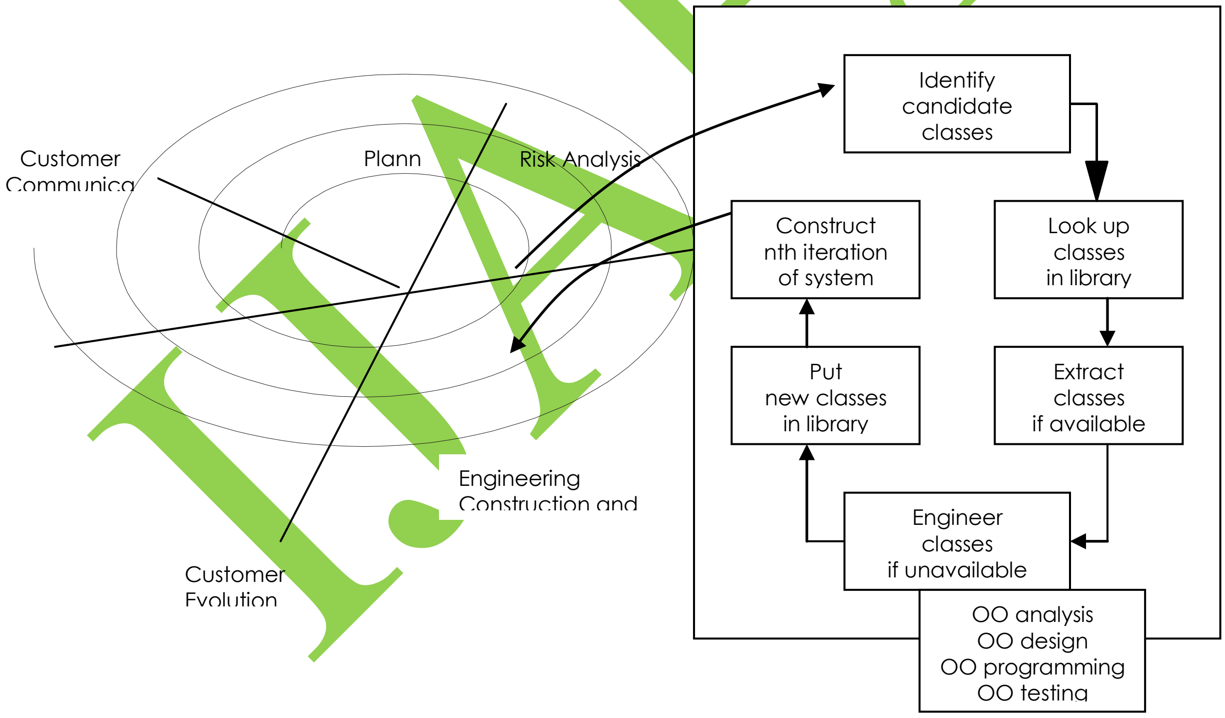


Figure 2

OBJECT-ORIENTED DEVELOPMENT

The term object-oriented is generally used to denote a programming approach that uses a number of object-oriented programming languages (e.g. Ada95, C++, Java). But actually the object-oriented paradigm encompasses a complete view of software engineering process.


Customer Communica
Plann
Risk Analysis
Engineering Construction and
Customer Evolution
Identify candidate classes
Look up classes in library
Extract classes if available
Engineer classes if unavailable
Put new classes in library
Construct nth iteration of system
OO analysis
OO design
OO programming
OO testing

The object-oriented development process starts with a customer communication phase. In this phase the main emphasis is on developing the framework of object-oriented project plan. As the name of this stage suggests, this phase involves communicating with the customers (clients) and ascertaining the needs. These needs are molded into classes. It is in this phase that the problem domain is defined and the basic problem classes are identified.

Then follows the planning and risk analysis. These stages establish a foundation for the object-oriented project plan. These stages are very important as in these stages, the planning relating to object-oriented analysis and design is undertaken and the risk analysis is done to gauge the risk factors involved.

The actual technical work associated with object-oriented analysis and design follows the plan shown in shaded boxes. It is in this stage that the actual development of objects and its members is undertaken. The main stress, in object-oriented analysis and design is upon 'reuse'. Instead of developing objects from scratch, they are looked up in the existing library of classes. The objects are also developed in such a way that promotes their reusability after completion of their development. First of all potential classes that define the needs of system are identified and these classes are called candidate classes. These classes are the framework or the raw material from which the whole system is developed.

Then the classes are looked up in library of existing object-oriented classes before they are built. If a class cannot be found in the library of existing classes, then through inheritance or polymorphism of object-oriented approach, classes are extracted. After that, the software engineer applies object-oriented analysis, object-oriented design, object-oriented programming and finally object-oriented testing to create the class and the objects derived from the class. These classes are put in the library so that they may be reused in future, whenever the need arises. This process is repeated until the system is completely developed.

ENTITY-RELATIONSHIP MODEL

The Entity-Relationship (ER) helps in implementing the real world by conceptualizing it in the form of entities, relationships (and attributes with each of them). This approach is the most common approach to information modeling.. ER diagrams are readily converted into a database implementation. The ERD is especially useful for application in which data and relationships, that govern data, are complex. *The ER model consists of the data, object and the relationship that connect data object to one another.* A data object is a representation of any composite information that must be understood by software. Attributes define the properties of data object and are used to name instances of data object, describe the instance, make reference to another instance in another table and act as "key" to identify a data object. For example, car ID. The relationships show how data objects are connected to each other.For e.e. a company *hires* employees.

CONCLUSION

Our world consists of objects. These objects exist everywhere, in nature, in human's artificial world, in our work place and in the products that we use. These objects can be created, destroyed, categorized, organized and manipulated. Perhaps, it is due to all pervasiveness of objects in our real world that an object-oriented view was proposed for the creation of computer software. This approach enables us to model our real world into the computer world in ways that helps us to understand and navigate it better than perhaps, other approaches.

Though this concept is very odd, an object-oriented approach to the development of software was first proposed in late 1960s. It took about twenty years to catch up and get popular with computer programmers. And in the decade of 1990, this approach gained new heights of popularity. Now more and more software product builders and a growing number of information systems and engineering professionals are resorting to this approach for programming.

An important question may arise in the reader's mind that why is object–oriented approach being preferred over the classical approach. The answer lies in actually implementing this object-oriented concept and then the obvious advantages of object-oriented programming stand apart from that of the classical approach. The object-oriented systems are easier to adapt and to scale. This means that assembling reusable subsystems, instead of developing it from scratch can easily create larger systems.

In this chapter, we will analyze the various steps and methods associated with object-oriented analysis and design. We will discuss the various stages of object-oriented development. Then we will understand in detail, the intricacies of system and object design. After that we will study a brief overview of existing methodologies of object-oriented analysis and design.

REFERENCES

1. Armstrong, *The Quarks of Object-Oriented Development*. In descending order of popularity, the 'quarks' are: Inheritance, Object, Class, Encapsulation, Method, Message Passing, Polymorphism, Abstraction
2. C. J. Date, Introduction to Database Systems, 6th-ed., Page 650
3. Kay, Alan. "The Early History of Smalltalk". Retrieved on 2007-09-13.
4. Kay, Alan. On the Meaning of "Object-Oriented Programming". Retrieved on 2008-07-26.
5. M.Trofimov, *OOOP - The Third "O" Solution: Open OOP*. First Class, OMG, 1993, Vol. 3, issue 3, p.14.
6. Stroustrup, The C++ Programming Language, 3rd-ed., p. 158
7. http://www.aaai.org/aitopics/html/aieffect.html
8. STLport: An Interview with A. Stepanov