Collaboration of Object Oriented Programming and Software Development

Shrilaxmi Deshpande

Assistant professor IT department
MMK College of Commerce and Economics, Mumbai, India

---***---

Abstract - *Object oriented programming and design are very important in today's environment. It provides generalizations for many problems in addition to many benefits like reusability, decomposition of problems into small easily understandable objects. In this paper, both structured programming and object oriented programs are discussed along with features and object oriented software paradigms.*

***Key Words*: Object oriented, encapsulation, inheritance, polymorphism, events, concurrency, and persistence**

1. INTRODUCTION

A group of programs developed specific purpose is referred to as software. The software development life cycle is a process of building good software and in life cycle stages provides quality and correctness of good software [1]. One wrong step in lifecycle can create big mistake in the development of software. The concepts used in software development life cycles are mainly structured programming and object oriented programming.

Structured programming is also known as modular programming. In this approach functions are defined according to the algorithm to solve the program. A function is applied to some data to perform the actions on data. This approach may be called a data driven approach. It depends on the solution domain because the algorithm is closer to the coding of the program. It uses following principle:

- Operator-operand concept
- Function abstraction
- Separation of data and functions

It has given importance to developing of algorithm. Some time critical data having global access may result in miserable output. Data and functionalities are considered as two separate parts. But in real world problem is solved by using a responsibility-driven approach. In this approach the relationship between the user and programmer is emphasized. The natural way of problem solving has basic principles namely:

- Message passing
- Abstraction
- Encapsulation

These are achieved through object-oriented technology, which follows the natural way of problem solving. Data abstraction and data encapsulation help in abstract view of the solution with information hiding. Data is given the proper importance and action is initiated by message passing[2]. Data and functionalities are put together resulting in objects and a collection of interacting objects are used to solve the problem. Object oriented programming languages are developed based on object-oriented technology.

2. FEATURES OF OBJECT ORIENTED PROGRAMMING

The fundamental features of object-oriented programming are encapsulation, data abstraction, inheritance, polymorphism, extensibility, persistence, delegation, genericity, concurrency, event handling,, message passing[3].

2.1 Encapsulation

The process of combining code and manipulating into single unit is commonly referred to as encapsulation, which provides a layer of security around manipulated data, protecting in external interference and misuse.

2.2 Data Abstraction

Abstraction is a design technique that focuses on the essential attributes and behavior. It is a collection of essential attributes and behavior relevant to programming a given entity for specific problem domain, relative to perspective of the user.

2.3 Inheritance

This feature allows the extension and reuse of existing code, without having repeat or rewrite the code from scratch. Inheritance involves the creation of new classes, also called derived classes from base class. The new derived class inherits the members of the base class and also adds its own. It is useful in extension and specialization.

2.4 Multiple Inheritance

When class is derived through inheriting one or more base class is called multiple inheritances. Instances of classes using multiple inheritances have instance variables for each of the inherited base classes.

2.5 Polymorphism

Polymorphism allows an object to be processed differently by the data types and data classes. It is ability for different objects to respond to the same message in different ways. It allows single name or operator to be associated with different operations [4].

2.6 Delegation

It is an alternative to class inheritance. Delegation allows an object composition to be as powerful as inheritance. In delegation two objects are involved in handling request namely: methods can be delegated by one object to another but the receiver stays bound to the object doing the delegating.

2.7 Generecity

It is a technique for defining software components that have more than one interpretation depending on the data type of parameter. Thus, it allows the abstraction of data items without specifying their exact type. These generic data type are resolved at the time of their usage and are based on the data type of parameter.

2.8 Persistence

It is the concept by which an object outlives the life of the program, existing between executions. All database systems support persistence.

2.9 Concurrency

Concurrency is represented through threading, synchronization and scheduling. It allows additional complexity to the development of applications, allowing more flexibility in software applications.

2.10 Events

Event can be considered a kind of interrupt. These interrupt the normal flow of program execution. Objects can pass information and control from themselves to another object, which in turn can pass control to other objects, and so on.

3. SOFTWARE DEVELOPMENT AND OBJECT-ORIENTED PROGRAMMING PARADIGMS

Instantiation of an object is defined as the process of creating an object of a particular class. An object has state or properties, operations and identity. Properties maintain the internal state of an object. Operations provide the appropriate functionality to the object. Identity differentiates one object from the other. Object name is used to identify the object [5]. Unique identity is important and hence the property reflecting unique identity must be used in an object.

The properties of an object are important because the outcome of the functions depends on these properties. The functions control the properties of an object. They act and react to messages. The message may cause a change in the property of an object. Thus, the behavior of an object depends on the properties [6].

There are two types of programming languages namely object-based and object oriented. Object based languages incorporate features like encapsulation and object identity. The object oriented languages in turn incorporates all features of object-based along with inheritance and polymorphism. In object oriented programming language a module is a logical grouping of related declarations, such as objects or procedures. There are some important features of object-oriented programming and design like:

- Emphasis on data rather than algorithms.
- Procedural abstraction is done by data abstraction
- Data and associated operations are unified, grouping objects with common attributes, operations and semantics.

Object-oriented technology is built upon object models. Object is anything having crispy defined conceptual boundaries. Model is the description of specific view of real world problem domain showing those aspects, which are important to the observer of problem domain. Object oriented programming addresses the solution to the problem domain. Object model is defined by means of classes and objects. The development of programs using object model is known as object-oriented development.

4. MAPPING OF REAL WORLD ENTITY INTO OBJECT ORIENTED PROGRAMMING

The concepts of object-oriented technology must be represented in object-oriented programming languages. Only then, complex problems can be solved in the same manner as they are solved in real-world situations. OOP languages use classes and objects for representing the concepts of abstraction and encapsulation. The mapping of abstraction to a program is as shown in figure 1.

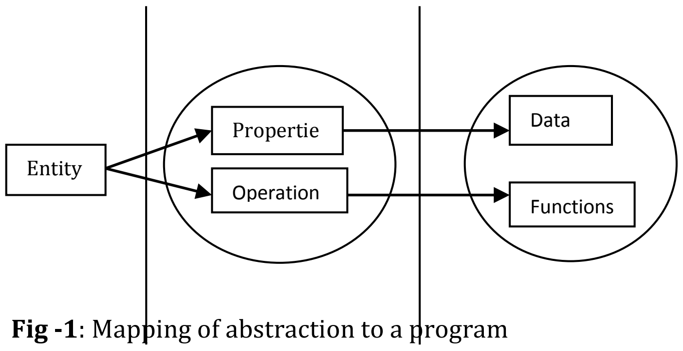

Fig -1: Mapping of abstraction to a program

The software structure that supports data abstraction is known as class. A class is a data type capturing the essence of an abstraction. It is characterized by a number of features. The class is a prototype or blue print or model that defines different features. A feature may be a data or an operation. Data are represented by instance variables or data variables in a class. The operations are also known as behaviors, or methods, or functions. They are represented by member functions of a class in C++ and methods in Java and C#.

A class is a data type and hence it cannot be directly manipulated. It describes a set of objects. For example, apple is a fruit implies that apple is an example of fruit. The term "fruit" is a type of food and apple is an instance of fruit. Likewise, a class is a type of data (data type) and object is an instance of class.

Similarly car represents a class (a model of vehicle) and there are a number of instances of car. Each instance of car is an object and the class car does not physically mean a car. An object is also known as class variable because it is created by the class data type. Actually, each object in an object-oriented system corresponds to a real-world thing, which may be a person, or a product, or an entity.

5. MODULARITY

The complexity of a program can be reduced by partitioning the program into individual modules. In object-oriented programming languages, classes and objects form the logical structure of a system. Modules serve as the physical containers in which the classes and objects are declared. Modularity is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules. A module is an indivisible unit of software that can be reused. The boundaries of modules are established to minimize the interfaces among different parts of the development organization. Modules are frequently used as an implementation technique for abstract data type. Abstract data type is a theoretical concept and module is an implementation technique. Each class is considered to be a module in OOP.

The responsibilities of classes are defined by means of their attributes and behavior. But a single object alone is not very useful. Higher order functionality and complex behavior are achieved through interaction of objects in different modules. Hence, interaction of objects is very important. Software objects interact and communicate with each other by sending messages to each other.

The activities are initiated by the transmission of a message to an object responsible for the action. The message encodes the request and the information is passed along with the message as parameters. There are three components to comprise a message:

- The receiver objects to whom the message is addressed.
- The name of the function performing the action.
- The parameters required by the function.

Interaction between objects is possible with the help of message passing. In the case of distributed applications, objects in different machines can also send and receive messages.

6. ADVANTAGES OF OBJECT-ORIENTED PROGRAMMING

The following are the advantages of software development using object-oriented programming:

- Software reuse is enhanced.
- Maintenance cost can be reduced.
- Data access is restricted providing better data security.
- Software is easily developed for complex problems.
- Improved performance and quality of software is achieved
- Data abstraction is possible.

5. LIMITATIONS OF OBJECT-ORIENTED PROGRAMMING

The following are the limitations of software development using object-oriented programming:

- The benefits of oop may realized after a long period
- Requires intensive testing procedures
- Solving problems using oop approach consumes more time than the time taken by structured programming approach

6. CONCLUSION

If there is complexity in software development, object-oriented programming is the best paradigm to solve the problem. Object concept helps to translate our thoughts to a program. It provides a way of solving a problem in the same way as a human being perceives a real world problem and finds out the solution. It is possible to construct large reusable components using object oriented techniques. Development of reusable components is rapidly growing in commercial software industries.

REFERENCES

[1] S. Barbey, M. Ammann, and A. Strohmeier. Open issues in testing Object Oriented software. In K. F. (Ed.), editor, ECSQ '94 (European Conference on Software Quality), pages 257–267, vdf Hochschulverlag AG an der ETH Z¨urich, Basel, Switzerland, October 1994

[2] G. Booch. Object Oriented Design. Benjamin/Cummings Publ., USA, 1991.

[3] R. Doong and P. Frankl. The astoot approach to testing object-oriented programs. ACM Transactions on Software Engineering and Methodology, 3(2):101–130, April 1994.

[4] R.-K. Doong and P. G. Frankl. Case Studies on Testing Object-Oriented Programs. In Proceedings of the Symposium on Testing, Analysis, and Verification (TAV4), pages 165–177, Victoria, CDN, Oct. 1991. ACM SIGSOFT, acm press.

[5] S. P. Fiedler. Object-Oriented Unit Testing. HP Journal, 40(3):69–74, April 1989.

[6] R. Fletcher and A. S. M. Sajeev. A framework for testing object-oriented software using formal specifications. In A. Strohmeier, editor, Reliable Software Technologies, number 1088 in Lecture Notes in Computer Science, pages 159–170. Springer, 1996.