Chapter – 7

Object Oriented Concepts and Principles

Object Oriented Paradigm:

The object oriented paradigm characterizes any problem domain as a set of objects that have specific attributes and behaviors. The objects in turn are characterized by classes and sub-classes.

The object oriented process moves through an evolutionary spiral that starts with customer communication. At this stage, the problem domain is defined and the basic problem classes are identified, planning is done to define resources, timelines and all other project related information.

The process involves:

a. Identify candidate classes
b. Lookup classes in library
c. Extract classes if available
d. Engineer/ develop classes if unavailable
   - Analysis
   - Design
   - Programming
   - Testing
e. Put new classes in library
f. Construct n^{th} iteration of system

Object Oriented Concepts:

1. Object:

An object is a real-world element in an object–oriented environment that may have a physical or a conceptual existence. Each object has:

- ☑ Identity that distinguishes it from other objects in the system.
- ☑ State that determines the characteristic properties of an object as well as the values of the properties that the object holds.
- ☑ Behavior that represents externally visible activities performed by an object in terms of changes in its state.

Objects can be modelled according to the needs of the application. An object may have a physical existence, like a customer, a car, etc.; or an intangible conceptual existence, like a project, a process, etc.

2. Class:

A class represents a collection of objects having same characteristic properties that exhibit common behavior. It gives the blueprint or description of the objects that can be created from it.

Creation of an object as a member of a class is called instantiation. Thus, object is an instance of a class.

The constituents of a class are:

- ☑ A set of attributes for the objects that are to be instantiated from the class. Generally, different objects of a class have some difference in the values of the attributes. Attributes are often referred as class data.
- ☑ A set of operations that portray the behavior of the objects of the class. Operations are also referred as functions or methods.

3. Encapsulation:

Encapsulation is the process of binding both attributes and methods together within a class. Through encapsulation, the internal details of a class can be hidden from outside. It permits the elements of the class to be accessed from outside only through the interface provided by the class.

4. Data Hiding:

Typically, a class is designed such that its data (attributes) can be accessed only by its class methods and insulated from direct outside access. This process of insulating an object's data is called data hiding or information hiding.

5. Message Passing:

Any application requires a number of objects interacting in a harmonious manner. Objects in a system may communicate with each other using message passing. Suppose a system has two objects: obj1 and obj2. The object obj1 sends a message to object obj2, if obj1 wants obj2 to execute one of its methods.

The features of message passing are:

- ☑ Message passing between two objects is generally unidirectional.
- ☑ Message passing enables all interactions between objects.
- ☑ Message passing essentially involves invoking class methods.
- ☑ Objects in different processes can be involved in message passing.

6. Inheritance:

Inheritance is the mechanism that permits new classes to be created out of existing classes by extending and refining its capabilities. The existing classes are called the base classes/parent classes/super-classes, and the new classes are called the derived classes/child classes/subclasses. The subclass can inherit or derive the attributes and methods of the super-class provided that the super-class allows so. Besides, the subclass may add its own attributes and methods and may modify any of the super-class methods. Inheritance defines an "is – a" relationship.

Types of Inheritance:

- ☑ **Single Inheritance:** A subclass derives from a single super-class.
- ☑ **Multiple Inheritance:** A subclass derives from more than one super-classes.

- ☑ Multilevel Inheritance: A subclass derives from a super-class which in turn is derived from another class and so on.
- ☑ **Hierarchical Inheritance:** A class has a number of subclasses each of which may have subsequent subclasses, continuing for a number of levels, so as to form a tree structure.
- ☑ **Hybrid Inheritance:** A combination of multiple and multilevel inheritance so as to form a lattice structure.

7. Polymorphism:

Polymorphism is originally a Greek word that means the ability to take multiple forms. In object-oriented paradigm, polymorphism implies using operations in different ways, depending upon the instance they are operating upon. Polymorphism allows objects with different internal structures to have a common external interface. Polymorphism is particularly effective while implementing inheritance.

Benefits of Object Model:

The benefits of using the object model are:

- ☑ It helps in faster development of software.
- ☑ It is easy to maintain. Suppose a module develops an error, then a programmer can fix that particular module, while the other parts of the software are still up and running.
- ☑ It supports relatively hassle-free upgrades.
- ☑ It enables reuse of objects, designs, and functions.
- ☑ It reduces development risks, particularly in integration of complex systems.

Identify Elements of an Object Model:

If we look around a room, there is a set of physical objects that can be easily identified, classified, and defined (in terms of attributes and operations). But when we "look around" the problem space of a software application, the objects may be more difficult to comprehend.

We can begin to identify objects by examining the problem statement or performing a "grammatical parse" on the processing narrative for the system to be built. Objects are determined by underlining each noun or noun clause and entering it in a simple table. Synonyms should be noted. If the object is required to implement a solution, then it is part of the solution space; otherwise, if an object is necessary only to describe a solution, it is part of the problem space. What should we look for once all of the nouns have been isolated? Objects manifest themselves in one of the ways represented below.

Objects can be:

- ☑ External entities (e.g., other systems, devices, people) that produce or consume information to be used by a computer-based system.
- ☑ Things (e.g., reports, displays, letters, signals) that are part of the information domain for the problem.
- ☑ Occurrences or events (e.g., a property transfer or the completion of a series of robot movements) that occur within the context of system operation.

- ☑ Roles (e.g., manager, engineer, salesperson) played by people who interact with the system.
- ☑ Organizational units (e.g., division, group, and team) that are relevant to an application.
- ☑ Places (e.g., manufacturing floor or loading dock) that establish the context of the problem and the overall function of the system.
- ☑ Structures (e.g., sensors, four-wheeled vehicles, or computers) that define a class of objects or in the extreme, related classes of objects.

It is also important to note what objects are not. In general, an object should never have an "imperative procedural name". For example, if the developers of software for a medical imaging system defined an object with the name image inversion, they would be making a subtle mistake. The image obtained from the software could, of course, be an object (it is a thing that is part of the information domain). Inversion of the image is an operation that is applied to the object. It is likely that inversion would be defined as an operation for the object image, but it would not be defined as a separate object to connote "image inversion." As Cashman states: "the intent of object-orientation is to encapsulate, but still keep separate, data and operations on the data."

MANAGEMENT OF OBJECT-ORIENTED SOFTWARE PROJECTS:

Project Management is the discipline of planning, organizing, securing and managing resources to bring about the successful completion of specific project goals and objectives. It is sometimes conflated with program management, however technically that is actually a higher level construction: a group of related and somehow interdependent

The primary challenge of project management is to achieve all of the engineering project goals and objectives while honoring the preconceived project constraints. Typical constraints are scope, time, and budget.

The secondary and more ambitious challenge is to optimize the allocation and integration of inputs necessary to meet pre-defined objectives.

In object oriented analysis we concentrated on identifying the objects that represented actual data within the business design. These objects are called entity objects. Entity objects usually correspond to items in real life and contain information, known as attributes, that describes the different instances of the entity. They also encapsulate those behaviors that maintain its information or attributes.

An entity object is said to be persistent, meaning the object typically lives on after the execution of a method. Two additional types of objects will be introduced during design. New objects will be introduced to represent a means through which the user will interface with the system. These objects are called interface objects. It is through the interface objects that the users communicate with the system. The use case functionality that describes the user directly interacting with the system should be placed in interface objects.

It translates the user's input into information that the system can understand and use to process the business event. Other types of objects that are introduced are objects that hold application or business rule logic. These objects are called control objects. They serve as the traffic cop

containing the application logic or business rules of the event for managing or directing the interaction between the objects.

1. Object Responsibility:

In design we focus on identifying the behaviors a system must support and, in turn, design the methods to perform those behaviors. Along with behaviors, we determine the responsibilities an object must have. An object responsibility is the obligation that an object has to provide a service when requested, thus corroborating with other objects to satisfy the request if required.

2. Object Framework:

An object framework is a set of related, interacting objects that provide a well-defined set of services for accomplishing a task. Component A component is a group of objects packaged together into one unit.

Management Spectrum:

The management spectrum describes the management of a software project or how to make a project successful. It focuses on the four P's; people, product, process and project. Here, the manager of the project has to control all these P's to have a smooth flow in the project progress and to reach the goal.

The four P's of management spectrum has been described briefly in below.

1. People:

People of a project includes from manager to developer, from customer to end user. But mainly people of a project highlight the developers. It is so important to have highly skilled and motivated developers that the Software Engineering Institute has developed a People Management Capability Maturity Model (PM-CMM), "to enhance the readiness of software organizations to undertake increasingly complex applications by helping to attract, grow, motivate, deploy, and retain the talent needed to improve their software development capability". Organizations that achieve high levels of maturity in the people management area have a higher likelihood of implementing effective software engineering practices.

Different people involves are:

a. Players
   - Senior Managers
   - Project (Technical) Managers
   - Practitioners
   - Customers
   - End user
b. Team Leader
c. Software Team

2. Product:

Product is any software that has to be developed. To develop successfully, product objectives and scope should be established, alternative solutions should be considered, and technical and management constraints should be identified. Without this information, it is impossible to define reasonable and accurate estimates of the cost, an effective assessment of risk, a realistic breakdown of project tasks or a manageable project schedule that provides a meaningful indication of progress.

3. Process:

A software process provides the framework from which a comprehensive plan for software development can be established. A number of different tasks sets tasks, milestones, work products, and quality assurance points enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team. Finally, umbrella activities overlay the process model. Umbrella activities are independent of any one framework activity and occur throughout the process.

4. Project:

Here, the manager has to do some job. The project includes all and everything of the total development process and to avoid project failure the manager has to take some steps, has to be concerned about some common warnings etc.