**UNIT II**

**OOAD - Object Oriented Analysis**

* In the system analysis or object-oriented analysis phase of software development, the system requirements are determined, the classes are identified and the relationships among classes are identified.
* The three analysis techniques that are used in conjunction with each other for object-oriented analysis are object modelling, dynamic modelling, and functional modelling.

## Object Modelling

Object modelling develops the static structure of the software system in terms of objects. It identifies the objects, the classes into which the objects can be grouped into and the relationships between the objects. It also identifies the main attributes and operations that characterize each class.

The process of object modelling can be visualized in the following steps −

* Identify objects and group into classes
* Identify the relationships among classes
* Create user object model diagram
* Define user object attributes
* Define the operations that should be performed on the classes
* Review glossary

## Dynamic Modelling

* After the static behavior of the system is analyzed, its behavior with respect to time and external changes needs to be examined. This is the purpose of dynamic modelling.
* Dynamic Modelling can be defined as “a way of describing how an individual object responds to events, either internal events triggered by other objects, or external events triggered by the outside world”.

The process of dynamic modelling can be visualized in the following steps −

* Identify states of each object
* Identify events and analyze the applicability of actions
* Construct dynamic model diagram, comprising of state transition diagrams
* Express each state in terms of object attributes
* Validate the state–transition diagrams drawn

## Functional Modelling

Functional Modelling is the final component of object-oriented analysis. The functional model shows the processes that are performed within an object and how the data changes as it moves between methods. It specifies the meaning of the operations of object modelling and the actions of dynamic modelling. The functional model corresponds to the data flow diagram of traditional structured analysis.

The process of functional modelling can be visualized in the following steps −

* Identify all the inputs and outputs
* Construct data flow diagrams showing functional dependencies
* State the purpose of each function
* Identify constraints
* Specify optimization criteria

## Structured Analysis vs. Object Oriented Analysis

The Structured Analysis/Structured Design (SASD) approach is the traditional approach of software development based upon the waterfall model. The phases of development of a system using SASD are −

* Feasibility Study
* Requirement Analysis and Specification
* System Design
* Implementation
* Post-implementation Review

Now, we will look at the relative advantages and disadvantages of structured analysis approach and object-oriented analysis approach.

**Use Cases**

* A *use case* is a description of a set of sequences of actions, including variants, that a system performs to yield an observable result of value to an actor.
* Graphically, a use case is rendered as an ellipse.

**Names**

* Every use case must have a name that distinguishes it from other use cases. A *name* is a textual string.
* That name alone is known as a *simple name;* a *path name* is the use case name prefixed by the name of the package in which that use case lives.
* **Note**
* A use case name may be text consisting of any number of letters, numbers, and most punctuation marks (except for marks such as the colon, which is used to separate a class name and the name of its enclosing package) and may continue over several lines
* **Use Cases and Actors**

An actor represents a coherent set of roles that users of use cases play when interacting with these use cases.

### Use Cases and Flow of Events

* A use case describes *what* a system (or a subsystem, class, or interface) does but it does not specify *how* it does it. When you model, it's important that you keep clear the separation of concerns between this outside and inside view.
* **Main flow of events:**
* The use case starts when the system prompts the *Customer* for a PIN number. The *Customer* can now enter a PIN number via the keypad. The *Customer* commits the entry by pressing the Enter button. The system then checks this PIN number to see if it is valid. If the PIN number is valid, the systemacknowledges the entry, thus ending the use case.
* **Exceptional flow of events:**
* The *Customer* can cancel a transaction at any time by pressing the Cancel button, thus restarting the use case. No changes are made to the *Customer*'s account.
* **Exceptional flow of events:**
* The *Customer* can clear a PIN number anytime before committing it and reenter a new PIN number.
* **Exceptional flow of events:**
* If the *Customer* enters an invalid PIN number, the use case restarts. If this happens three times in a row, the system cancels the entire transaction, preventing the *Customer* from interacting with the ATM for 60 seconds.

**Use Cases and Scenarios**

* Scenarios are to use cases as instances are to classes, meaning

that a scenario is basically one instance of a use case.

* **Use Cases and Collaborations**
* A use case captures the intended behavior of the system (or subsystem, class, or interface) you are developing, without having to specify how that behavior is implemented. That's an important separation because the analysis of a system (which specifies behavior) should, as much as possible, not be influenced by implementation issues (which specify how that behavior is to be carried out)

The Concept of Classification

Classification is the process of checking to see if an object belongs to a category or a class and it is regarded as a basic attribute of human nature. • A class is a specification of structure, behavior, and the description of an object.

Approaches for Identifying Classes

•The noun phrase approach.

•The common class patterns approach.

•The use-case driven approach.

## UML Relationship

**Relationships in UML** are used to represent a connection between structural, behavioral, or grouping things. It is also called a link that describes how two or more things can relate to each other during the execution of a system.

Type of UML Relationship are Association, Dependency , Generalization , and Realization.

* **Association**

It is a set of links that connects elements of the UML model. It also defines how many objects are taking part in that relation.

* **Dependency**

In a dependency relationship, as the name suggests, two or more elements are dependent on each other. In this kind of a relationship, if we make a change to a particular element, then it is likely possible that all the other elements will also get affected by the change.

* **Generalization**

It is also called a parent-child relationship. In generalization, one element is a specialization of another general component. It may be substituted for it. It is mostly used to represent inheritance.

* **Realization**

In a realization relationship of UML, one entity denotes some responsibility which is not implemented by itself and the other entity that implements them. This relationship is mostly found in the case of **interfaces.**