

## UNIT-2

**Q1. Write short notes on the following,**

- (i) System architecture
- (ii) Software architecture.

**Answer :**

(i) **System Architecture**

System architecture can be defined as the collection of rules, conventions, customer requirements and specification that are required to develop the system.

(ii) **Software Architecture**

Software architecture can be defined as the specification of structure of the software under development along with data algorithms and interfaces required in it. It is considered as an essential part of software engineering due to the following reasons.

**Q2. What are the design principles of layered architectural style?**

**Answer :**

Following are the design principles of layered architectural style.

1. **High Cohesion**

Different functions are assigned to different layers which maximizes the cohesion and minimizes the coupling with the layer.

2. **Loose Coupling**

The interaction among layers is carried out in the form of events which leads to loose coupling between the layers.

3. **Abstraction**

The system details are kept hidden by developing the system at high level. Only the required amount of details necessary for understanding layer interaction and their responsibilities are made available.

4. **Reusability**

The lower layers are independent of their higher layers which makes them reusable in other environments.

5. **Well-defined Functional Layers**

The layers are assigned with well-defined functions where the functionality of top most layer depends on the functionalities of lower layers (i.e., business logic, data).

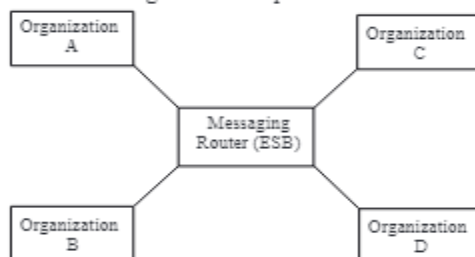
---

**Q3. Discuss in brief about enterprise service bus.**

**Answer :**

ESB is an architecture which acts as the communication system for number of software applications to interact with each other. A part from this, ESB also enables the communication among incompatible form of messages by converting one form of message to the other. This in turn helps in transferring the messages in a single, general form on to the network. Messaging router ( of MOA style using Hub and Spoke model) is an example of ESB.

This router first determines the web services in a directory which is based on UDDI. After this, it acts as a service provider to assist communication among the ports using endpoint website in favor of service consumer application. This, it allows all the service consumers to interact only over the ESB. Diagrammatic representation of ESB is show below.



**Figure: Messaging Router - ESB**

**Q4. What is service oriented architecture?****Answer :**

SOA is defined as the process of exchanging data through different applications.

(or)

The method of integrating the business processes by dividing large application into smaller modules (services).

(or)

SOA is also defined as a set of services communicating with each other involving some activity.

(or)

An architectural method to develop sophisticated systems through set of loosely-coupled interconnected blocks called services.

In SOA, services reflect the functionality. These services are provided to the service consumers with the help of interfaces. The interface details are broadcasted. It supports interaction to be carried out among processes instead of components or objects. For this reason, SOA is a loosely coupled architecture. Commonly used processes need to be developed as web services so that they can be easily shared among all the departments. These web services allow interoperability among the processes. The client and server may belong to distinct platforms. In such case, they require http protocol XML data format to communicate. However, they can be on one tier or remote physical server within the network.

---

**Q6. What is object oriented design approach?****Answer :**

In this approach, the building blocks of software is the object or class. Object is an entity whereas class describes the features of set of common objects exhibiting common state and behavior.

Here, the system is decomposed with respect to key abstractions prevailing in the problem domain. In previous approach, the problem was decomposed into steps such as get formatted update and add checksum, but in this approach, the objects are identified as master file and checksum.

**Q5. Describe about grid computing.****Answer :**

The concept of grid computing is different from that of cloud computing. Grid computing is the process that involves aggregation, sharing and selection of several distributed resources and later process access to them. Such resources include storage, computers, data sources and specific devices of different organization and so on. The only difference is that the distributed resources will be replaced with the web service based protocols. These protocols play the role of discovering the resources, accessing them, allocating them to tasks etc. But the entire thing will be viewed as a single virtual system.

---

**Q7. Discuss the differences between top down structure and object oriented design.****Answer :**

Top Down Structure		Object Oriented Design
(i)	In top down structure approach, functions acts as the basic units of system analysis and design.	In object oriented design approach, objects acts as the basic units of system analysis and design.
(ii)	It involves algorithmic decomposition.	It involves object oriented decomposition.
(iii)	It can solve complex problems moderately.	It can solve any type of complex problems.

**Q8. Write short notes on the following,**

- (i) Class diagrams
- (ii) Activity diagrams.

**Answer :**

Model Paper-II, Q4

**(i) Class Diagrams**

These diagrams reveal the static design view of a system and include few active classes which are used in capturing the static process of overall system. The class diagram consists of classes, interfaces and collaborations. The main purpose of class diagrams is to visualize, specify and document the structural modeling.

**(ii) Activity Diagram**

These diagrams are considered as effective while modeling functionalities of a given system. Hence these diagrams reflect activity(ies), the type of flows between these activities and finally the response of objects to these activities. In this way, it reflects dynamic aspect of a given system to be modeled while prioritizing the flow of control among objects participating in the system. The above list of diagrams are not sufficient to model a complex software project many tools use UML to extract different kinds of diagram that are sufficient enough to model complex projects.

---

**Q9. What is the importance of UML?**

**Answer :**

UML is one of the standard languages used for modeling or writing the blueprints of a software. The blueprints are very essential prior to the development of a successful software. To achieve this, UML initially visualizes, specifies, constructs and documents various issues related to building of software system. Based on these capabilities, UML is used for modeling systems that may extend from small enterprises to large scale development organizations and from small entity of computer-based software to hard real time embedded systems.

UML is termed to be an "Expressive language". The word "Expressive language" means that the language models the systems in an elucidating manner and can be applied directly without any modifications. UML is also termed as a simple language that is easy to understand and to use.

**Q14. What is meant by architectural style? Discuss in detail about client/server architectural style.**

**Answer :**

**Architectural Style**

An architectural style can be defined as the description group of systems with respect to the pattern of structural organization. It can also be define as the specification of vocabulary possessing the components, types of connectors and collection of rules regarding their connectivity. Components can be clients, servers, filters, databases, layers etc., whereas connectors can be event broad cast, procedure call, pipes and database protocols.

**Client/Server Architectural Style**

This style is based on two types of components called clients and servers. The client can request and receive the services offered by the server. This style restricts the clients to interact with other clients directly. The procedure followed for client/server communication is as follows, initially the client initiates the communication with server by sending a request for the required service. The server recieves the request, process it and then responds to the requested client. It is also called as 2-tier architectural style.

This style is one of the core concept of distributed computing. Web browser is an example of this architecture.

## **Layered Architectural Style**

This architectural style is based on hierarchical organization where every layer serves its higher layer and acts as a client for its lower layer. Every layer performs specific task assigned to it while the communication between the layers is carried out using interfaces. For instance, internet users use http as an interface to communicate over first layer (say presentation) where as JDBC is used as an interface when communication is between middle layer (say business logic) and backend (database) layer. Every layer is assigned with different set of responsibilities which increases maintainability and flexibility.

This architecture has the following two styles. They are as follows,

**(i) Strict Layering Style**

This architectural style allows the interaction of resources belonging to the same layer or the resources of the immediately below layer.

**(ii) Relax Layering Style**

This architectural style allows interaction of resources of same layer or the resources of any of the layers present below it.

The layers in this architectural style can be organized in different ways. For example, all the layers can be included in a single hardware machine (also known as one-tier) (or) each can reside on a different hardware machine (also known as n-tier).

## **Design Principles of Layered Architectural Style**

Following are the design principles of layered architectural style.

**1. High Cohesion**

Different functions are assigned to different layers which maximizes the cohesion and minimizes the coupling with the layer.

**2. Loose Coupling**

The interaction among layers is carried out in the form of events which leads to loose coupling between the

Layers
--------

**3. Abstraction**

The system details are kept hidden by developing the system at high level. Only the required amount of details necessary for understanding layer interaction and their responsibilities are made available.

**4. Reusability**

The lower layers are independent of their higher layers which makes them reusable in other environments.

**5. Well-defined Functional Layers**

The layers are assigned with well-defined functions where the functionality of top most layer depends on the functionalities of lower layers (i.e., business logic, data).

**Advantages of Layered Architectural Style**

The primary advantages of this style includes the following.

**1. Localization**

The affect of upgrades in technology is on a particular layer instead of all the layers.

**2. Performance**

The issues related to performance can be estimated as one layer can be distributed over different layers based on size, users and response time.

**3. Ease of Change Management**

Changes at the highest level (i.e., layer) can be managed easily as the details of this layer is abstracted.

**4. Reusability**

A single functionality can be easily shared among various channels. For instance, sharing of banking

**Q16. Discuss about separated presentation pattern.**

**Answer :**

**Separated Presentation Pattern**

It is a pattern which isolates the function of User Interface (UI) design from the logic behind the UI. An example of this pattern is Model View Controller (MVC).

**Model View Controller (MVC)**

The MVC design pattern separates user interface code into three classes, namely Model View Controller. These are also considered as objects.

MVC implements design patterns to solve the problems in the applications which contain business logic, data access code. This is because even a small change causes adverse effects since the components are interdependent. The MVC separates data access, data presentation and business logic.

---

1. **Model**

Model represents enterprise data and business rules which governs access and updates of data. It stores application logic and data for user interface. The figure clearly shows how it responds to state queries sent by the View and the state change addressed by Controller.

2. **View**

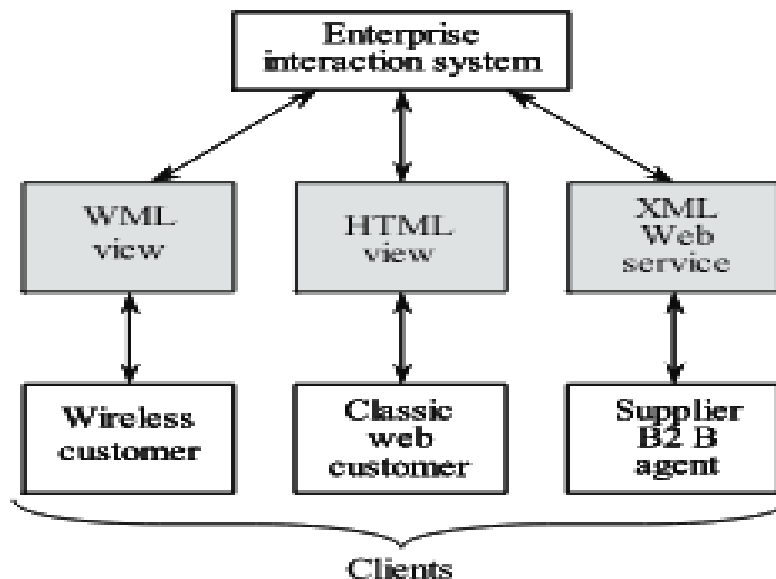
View concerns with presentation of enterprise data. It allows controller to select the view. Multiple views can be attached to model which provide different presentations where each view update itself and must forward all input events to controller.

3. **Controller**

The controller responds to the notifications from view and translates the input which further reflects changes in the model. Sometimes the controller also makes logical decisions itself. The controller selects the view for responses and also calls methods on the view to apply changes to the user interface such that they do not affect the model.

Model, View, Controller can be provided in a single MVC as shown in the below figure. The view passes user input to controller thereby the controller modifies view and model. In response to the updates provided from the controller, the model changes and even notifies the change to the view which in turn updates user interface.

must have a view.



Figure



**Q17. Discuss in detail about N-tier/3-tier architectural style.**

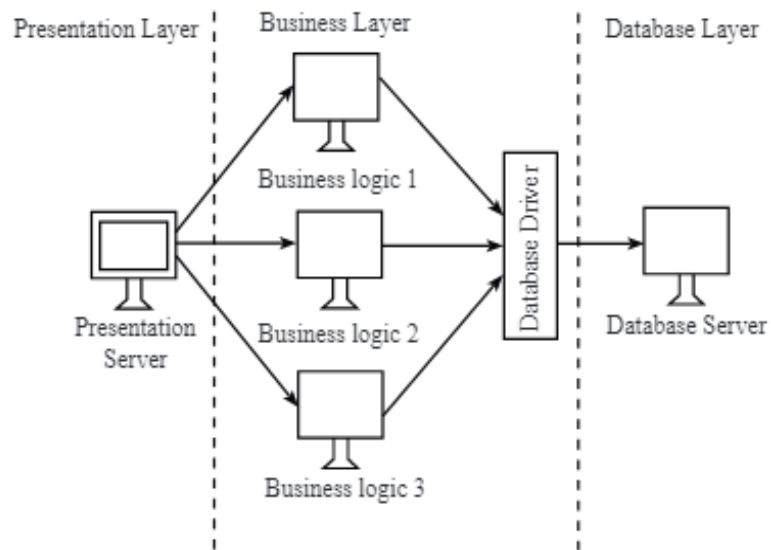
**Answer :**

**N-Tier/3-Tier Architectural Style**

N-tier/3-Tier architectural style refers to the architecture where program is divided into three or more layers and these layers are separately assigned to different physical computers. Basically, the layers in this architecture are,

- (i) Presentation layer/Client tier
- (ii) Application layer/Business tier
- (iii) Database layer/Data tier.

These layers interact with each other through platform specific protocols such as JDBC, http etc. Diagrammatic representation of 3-tier architectural style is shown below,



**Figure: 3-Tier Architecture**



## **Object Oriented Architecture**

It is an architectural style which divides the functionalities of an application or system into different independent and reusable objects. In other words, it divides a system functionality into different subsystems using object oriented concept. This style is used during application design stage. It represents the system in the form of different objects that work collaboratively rather than a set of procedures or sub-programs. The objects in this style are loosely coupled and interact with each other through interfaces or messages.

The object models developed using this style are capable of supporting engineering, scientific or business applications. Here, all the real world entities are represented as objects. For instance, the passengers or tickets in a ticketing application are considered as objects.

### **Principles of Object-oriented Architecture**

The following are key principles of object oriented architectural style.

#### **1. Encapsulation**

The internal details of the objects such as operations and variables are encapsulated. So, to avail the service, service consumers need to mention only the values with in the interfaces. These services are served by the objects. This prevents the internal details of the object from unauthorized modifications and access.

#### **2. Composition**

Different objects can be combined together to form a single, large object where unnecessary objects can be made hidden while other objects can be made available using interfaces.

#### **3. Abstraction**

Generalization can be performed on complicated operations at higher level to obtain its basic characteristics while avoiding unnecessary details.

#### **4. Inheritance**

The new object can inherit the functionality of an existing object with the similar functionality. The changes are reflected automatically to all the objects from base objects.

#### **5. Decoupling**

An object's abstract interface can be separated from the caller or consumer object. This helps the service consumer to opt desired implementation independently.

#### **6. Decomposition**

The components which are larger and complex can be decomposed into smaller ones to simplify the process of solving.

- (a) Enterprise Service Bus (ESB)
- (b) Internet Service Bus (ISB).

**Answer :**

**(a) Enterprise Service Bus (ESB)**

ESB is an architecture which acts as the communication system for number of software applications to interact with each other. A part from this, ESB also enables the communication among incompatible form of messages by converting one form of message to the other. This in turn helps in transferring the messages in a single, general form on to the network. Messaging router (of MOA style using Hub and Spoke model) is an example of ESB.

This router first determines the web services in a directory which is based on UDDI. After this, it acts as a service provider to assist communication among the ports using endpoint website infavor of service consumer application. Thus, it allows all the service consumers to interact only over the ESB. Diagrammatic representation of ESB is show below,



**Figure: Messaging Router - ESB**

**(b) Internet Service Bus (ISB)**

ISB is capable of connecting two devices, the devices with servers, two ESBs and two websites. It is similar to ESB but it is organized by the third party and charge the customers based on the service usage.

## **Component Based Architectural Style**

This architectural style considers the components that are developed by different developers individually as independent entities. Development of such components require some standards and contracts so that the system can work properly upon their integration. The main purpose of this style is to provide reusability while preventing the development of component from scratch.

The term components have been defined in different ways. Some of them are discussed below,

### **Definition 1**

It is an integral, deployable and replaceable part of software in binary form with a particular interface or API (Application Programming Interface). It does not require modification and redevelopment and also reduces the need for recompilation. This definition is the most efficient one among all the definitions.

## **Principles of Component-based Architectural Style**

The major principles of component based are,

### **1. Replaceable**

The components can be substituted by other related components without losing its functionality.

### **2. Reusable**

The components can be reused in different applications.

### **3. Independent**

The components design donot rely on the design of other components.

### **4. Encapsulated**

The internal details of the components are encapsulated since, the interfaces helps in making the components functionality available to the service consumers of the component.

### **5. Extensible**

The component functionality can be modified according to obtain new functionalities which makes the system extensible.

### **6. Context Independence**

The static-defined-data is used for developing context of the component. Thereby, making these component context independent.

## Cloud Computing Architecture Style

Cloud computing can be viewed as a model for distributing information technology. Here, the information which is to be accessed is stored in cloud storage and it gives the privilege to the user to access the information whenever and from wherever they want. Thereby, allowing the users to work remotely. In general, cloud computing is nothing but the use of computing resources such as hardware and software which are distributed as a service across the network. The service can be compared with internet connection on mobile for which users are charged based on the usage. It centralises the data storage, processing and bandwidth which in turn provides efficient computing process to the users.

Cloud computing is defined by US National Institute for standards and Technology (NIST) as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (eg. networks, servers, storage applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

Services offered by cloud computing are categorized as,

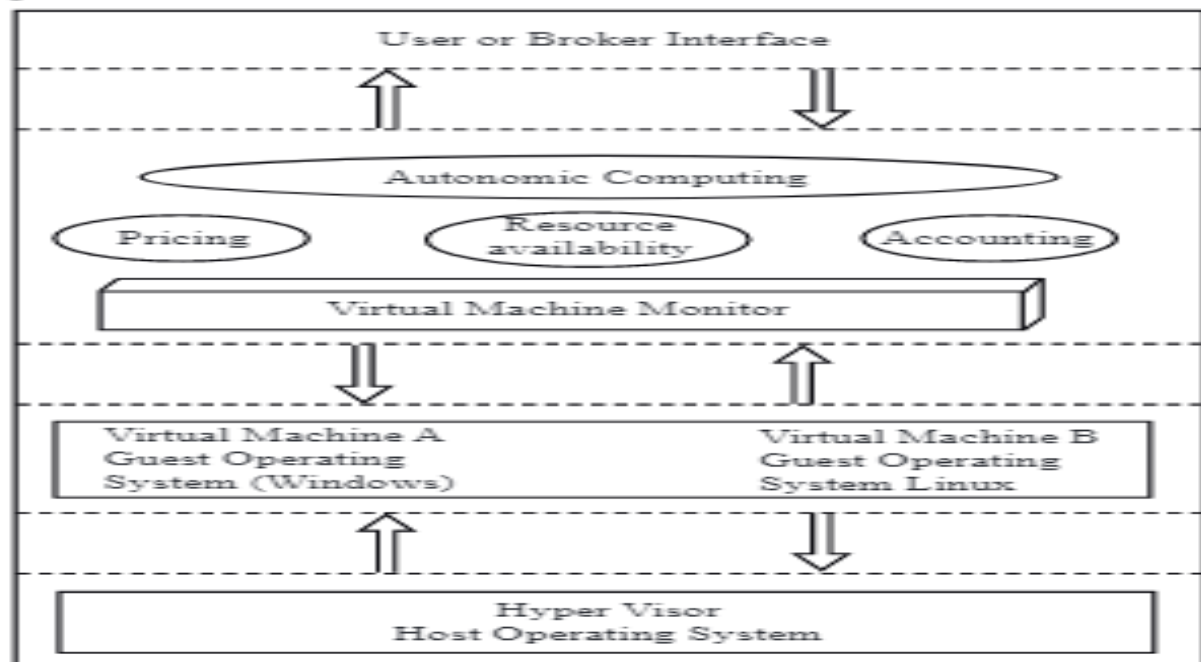
1. Software-as-a-Service (SaaS)
2. Platform-as-a-Service (PaaS)
3. Infrastructure-as-a-Service (IaaS).

### **(a) Grid Computing**

The concept of grid computing is different from that of cloud computing. Grid computing is the process that involves aggregation, sharing and selection of several distributed resources and later process access to them. Such resources include storage, computers, data sources and specific devices of different organization and so on. The only difference is that the distributed resources will be replaced with the web service based protocols. These protocols play the role of discovering the resources, accessing them, allocating them to tasks etc. But the entire thing will be viewed as a single virtual system.

**(b) Cloud Computing Platform**

The high level cloud computing platform contains four layers. First layer is user interface layer. Second layer deals with autonomic computing, monitoring of virtual machine and other activities. Third layer consist of virtual machine. The lowest layer carries Hypervisor software such as ESX, Xen, KVM, Hyper V. The diagrammatic representation of cloud computing platform is shown below,



**Figure: High-level Architecture of Cloud Computing Platform**

**(c) Cloud Platform in Market**

The Intel or AMD processors are divided into virtual partitions with the help of Hyper-V. It contains the following three components.

1. Hypervisor
2. Virtualization stack
3. Virtualized input/output model.

The Hypervisor component acts as a thin software layer which is placed directly on processor. Stack of virtual machines and operating systems are managed by using threads on host operating system. Virtualized I/O models such as live meeting, document sharing etc., are offered by microsoft share point portal services.

EUCALYPTUS (Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems) is an open source product. It helps in developing on-premise cloud platforms which are called modular and hypervisor independent.

Q27. Discuss in brief about core, configurable and customizable architecture.

**Answer :**

**Core, Configurable and Customizable Architecture**

This architecture enables the development of replicable applications considering the following paradigms,

- (i) Core
- (ii) Configuration
- (iii) Customization.

---

**Example**

Consider that an application has to be developed for a country to serve its functionalities to all its states. If this application need to offer 10 mandatory and 5 optional services from among 15 services, it must be developed centrally.

**(i) Core**

The functionalities of this application that are required by all the states is called core.

**(ii) Configuration**

The application made available to a state, must provide state specific interface (UI) that fulfills the requirements and the selection of the state. This is called as configuration. Some of the configurations that can be done on the above given example are,

- ❖ Enabling user interface in state-specific language.
- ❖ Enabling five optional services selected by the state.
- ❖ Enabling fonts, colours, layout of various types selected by the state.
- ❖ Enabling service delivery charges with respect to the state.

The main advantages of configuration include,

1. The application can be supported in a better way as it has to be managed at state level.
2. The quality can be controlled effectively leading to reliable and stable system.

**(iii) Customization**

The business processes may differ for different states. So, the program code need to be modified according to the state requirements. This is called as customization. It can result in increased cost, delay and efforts for maintenance.

## **Data Flow Model**

Data flow model depicts processing of data at each stage of system development. The model reflects the procedure used in data processing and the functional transformation that takes place on data. It is developed with the help of data flow diagrams.

It is the hierarchy of various DFDS. The highest level of the DFD is called as level 0 DFD or context diagram. It is easy to draw and interpret. It includes level 1, level 2 and other lower levels. Context diagram takes a major role in developing the DFD model of a system. Level 0, Level 1 contains only single DFD. Also, level 2 contains 7 and level 3 contains 49 DFDs so on. It consists of only one data dictionary and includes several data names and their respective definitions.



### Structured Analysis and Design (SAD)

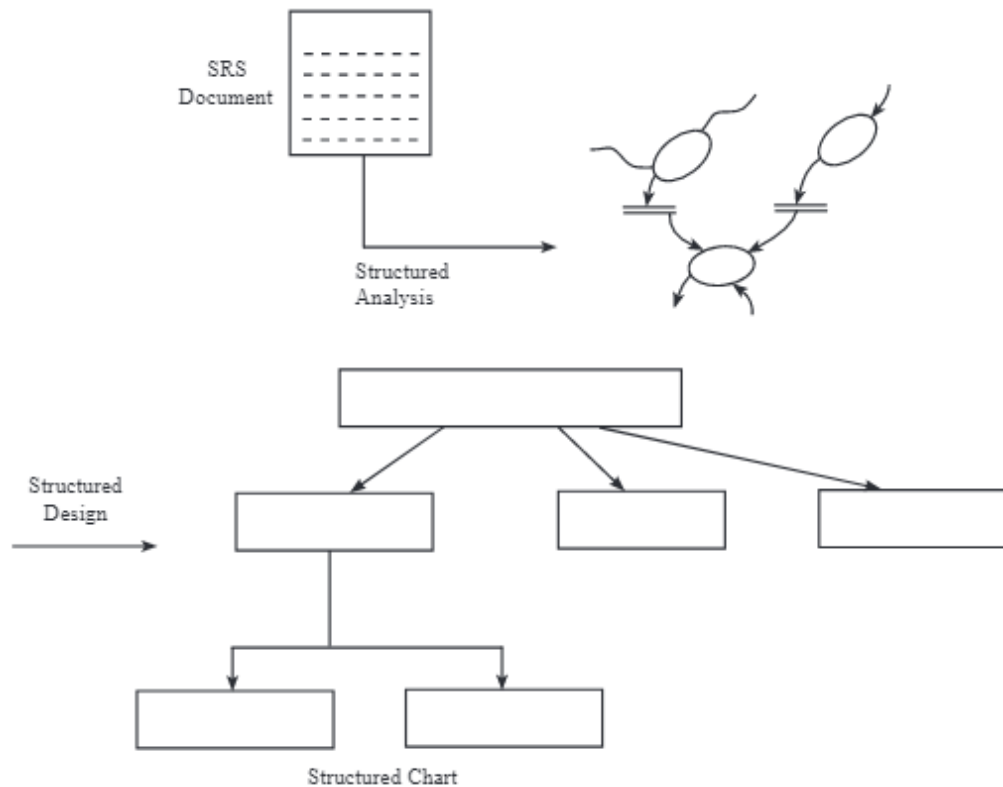
Structured analysis and design is a combination of Structured Analysis (SA) and Structured Design (SD).

#### (i) Structured Analysis (SA)

Structured analysis refers to the conversion of decomposed functions of textual format to a graphical or diagrammatic format DFDs. It considers the detailed structure of the system and represent these details according to the users point of view. Therefore, functions are labelled in such a way that they are understandable to the user. Every function in SA is considered as independent of others and hence decomposition is performed without considering its impact on other function.

#### (ii) Structured Design (SD)

This step improves and converts the functions data into a form which can be easily used in specific programming language.



### Construction of Context Diagram

Construction of context diagram involves analysing the following,

- (i) Accepting the input values for Rate of interest (R), Time (t) and Principal amount (P).
- (ii) Producing the output for calculated interest.

The formula for the calculation of simple rate of interest is,

$$\text{Interest} = \frac{\text{Principal amount (P)} \times \text{Time (t)} \times \text{Rate of interest (R)}}{100}$$

Therefore, the context diagram for calculation of interest will be,

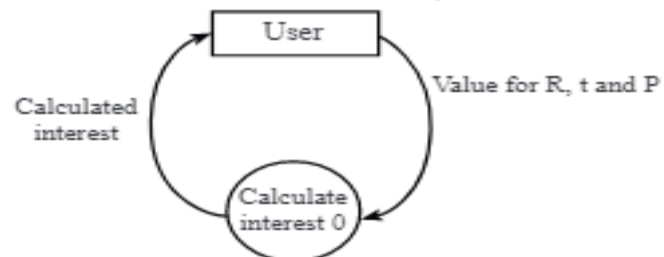


Figure: Context Diagram for Interest Calculation

### Construction of Level 1 DFD

The above context diagram is decomposed to form level-1 DFD. The decomposition is done as follows,

- (i) Reading of input values for P, R and t.
- (ii) Validating input values in order to assume that they are positive.
- (iii) Calculating simple interest using the formula.
- (iv) Displaying the output.

The level-1 DFD can be drawn using above functions are,

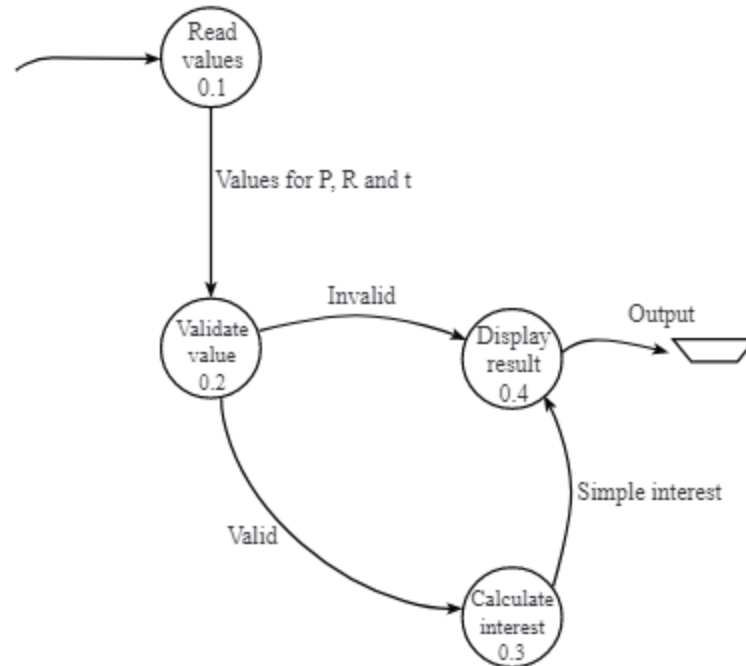


Figure: Level-1 DFD for Calculation of Simple Interest

**Q37. What is object oriented analysis? Discuss the four elements of object model.**

**Answer :**

**Object Oriented Analysis (OOA)**

It deals with the problem domain. The focus of the object that comes out of analysis is on the problem domain. These objects are referred as “semantic objects”, as they often describe few things or concepts in the problem. The task of analysis activity is to find out the classes in the system and their relationship and frequently represented by class diagrams. The models built during object-oriented analysis form the basis for object-oriented design.

**Elements of Object Model**

There are four major elements of object model, they are as follows,

1. Abstraction
2. Encapsulation
3. Modularity
4. Hierarchy.

**1. Abstraction**

For answer refer Unit-II, Page No. 71, Q.No. 35, Topic: Abstraction.

**2. Encapsulation**

Encapsulation is the mechanism which binds together the code and the data. This encapsulation prevents the code and data from being accessed by other code defined outside the class. Encapsulation can be defined as wrapping up of data and methods into a single unit known as a

## Top Down Structure Design Method

Top down structure design method is defined as the process of breaking an entire unit into small sub units till each subunit can be written using a program instruction. The arrangement of units follows a hierarchical structure i.e., larger unit appears at the top and smaller unit appears at the bottom. The flow of control is passing from top to bottom. Moreover, the organization of components is fixed, so each component can have only one entry point as input and only one exit point as output.

## Data Flow Diagrams

A Data flow diagram is the diagrammatic representation of various functions and the data items which are exchanged among different processes through an information system. It represents the system operations, its implementation and the tasks completed by system.

DFDs help in designing information-processing in an organization. It is also used for performing structured analysis.

### Symbols Used in DFD


Data Flow Diagrams (DFDs) use several symbols to explain the inputs, outputs, storage points and routes between every destination. The symbols used in DFD are as follows,

1. **Entity**

Entity is denoted by a rectangle i.e., .

It represents the source/destination entity which is external to the system.

2. **Data Flow**

Data flow is denoted by an arrow i.e., .

It represents the flow of data between processes, entities and data stores.

3. **Process**

Process is denoted by a circle i.e., .

It represents the process that performs data transform.

4. **Data Store**

Data store is denoted by two parallel lines i.e., .

It represents the storage location of data which can be either temporary or permanent.

### Example

Consider an example of Train Ticket Reservation system. Its corresponding data flow diagram using function oriented model is as follows,



Figure: Flow Diagram of Train Ticket Reservation System

This model typically describes about not only entities and data generated by them, but also about the data stored in them. In addition to this, it also describes about the relationships existing among entities. These models use natural language which helps the customers in retrieving their corresponding feedback and approval easily.

The development of datamodels is performed at three layers namely conceptual data model, logical data model and physical data model. The initial step in modeling is the development of Entity-Relation(E-R) diagram.

### E-R Diagram

The entity-relationship model is one of the most widely used data model while designing the database. This model was developed in 1976 by Peter Chan and is also referred to as high-level conceptual data model. The main purpose of developing E-R model is to support the user's view about the data and to hide the implementation details associated with database design.

ER-models are represented in ER-diagram, which are basically the graphical representations of entire logical structure of database. The following are the basic components of ER model.

- (i) Entities
- (ii) Attributes
- (iii) Relationships.



#### (i) Entities

An Entity is a real-world object that is distinguishable from other objects. In ER-diagram an entity is represented by a rectangle (entity box). The name of the entity is a noun and is written in the centre of rectangle. Whenever, ER-diagram is applied to relational model, an entity is mapped to relational table wherein each row represents an entity instance.

#### Example



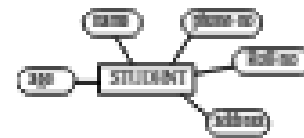
#### (ii) Attributes

A set of attributes associated with an entity describes the characteristic feature of respective entity. It can also be defined as the qualifier that provides additional information about the entity. Generally, it is an atomic unit of information associated with the named entity. This information helps in uniquely defining an entity.

In ER-diagram, attributes are represented by ellipse and the name of the attributes is written inside the ellipse. Each of the attribute is linked with the respective entity.

#### Example

The attributes associated with 'student' entity include Roll-no, name, age, address, phone-no. These attributes are represented as,



#### (iii) Relationships

Relationships describe associations between different entities. In ER-diagram, relationships are represented by diamond-shaped symbol and the name of relationship is written inside the diamond. The two sides of diamond are connected to their respective related entities. The different types of relationships that are defined in ER model includes, one-to-many, many-to-many and one-to-one. These relationships are represented by using any one of the following notations,

- (a) Chen notation
- (b) Crow's foot notation.

#### (a) Chen Notation

In this type of notation, the relationship is written next to the respective entity box.

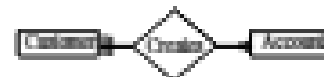
#### Example



#### (b) Crow's Foot Notation

In Crow's Foot Notation, three-pronged symbol is used for representing 'many' sides of relation and short-line segments for representing 'one' side of relation.

#### Example



### Layers of Data Model

The three layers of data model are given below,

#### 1. Conceptual Data Model

Conceptual data model is referred to as domain modelling. It describes about the entities and the relationship existing among the entities. The following figure illustrates the conceptual data model.



Figure: Conceptual Data Model

As shown in the above figure, only instances are described, but not the attributes.

## 2. Logical Data Model

In logical data model, entities of conceptual data model are upgraded by including the attributes of the entities. The following figure illustrates the logical data model.



Figure: Logical Data Model

## 3. Physical Data Model

In physical data model, entities and attributes identified at logical data model are upgraded by including data types of the attributes belonging to each and every entity.



Figure: Physical Data Model

## Q42. Discuss briefly about class diagram and associated class.

**Answer :**

### Class Diagram

These diagrams reveal the static design view of a system and include few active classes which are used in capturing the static process of overall system. The class diagram consists of classes, interfaces and collaborations. The main purpose of class diagrams is to visualize, specify and document the structural modeling.



Figure: Interaction Between Class 'Customer' and Class 'ATM' Associative Class

A new class which is emerged by interaction of two classes is referred to as associative class. For instance, consider the above class diagram wherein Customer → Debits Money → ATM. The attributes such as Debit card No, PIN NO, expiry date and so on cannot be placed in two classes (i.e., customer, ATM). Therefore, a new class 'Debit Card' is created and these attributes are stored in it. The associated class 'Debit Card' is shown below,



Figure: Associated Class 'Debit Card'

The various relationships possible among different objects of the classes are as follows,

- (i) Unidirectional association
- (ii) Bidirectional association
- (iii) Aggregation relationship
- (iv) Composition relationship
- (v) Inheritance relationship.

**(i) Unidirectional Association**

Unidirectional association is a structural relationship existing among two classes where in dependency is unidirectional (i.e., One direction). The following figure illustrates unidirectional association between two classes.

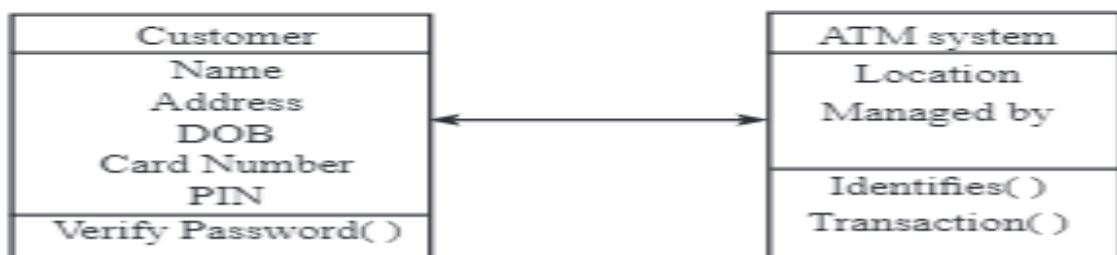


**Figure: Unidirectional Association**

**(ii) Bidirectional Association**

Bidirectional association is a structural relationship existing among two classes wherein dependency among them is in both the directions. In other words, one class knows about the relation of another class and vice versa.

The following figure illustrates Bidirectional relationship existing among two classes namely 'Customer' and ATM system.



**Figure: Bidirectional Association**

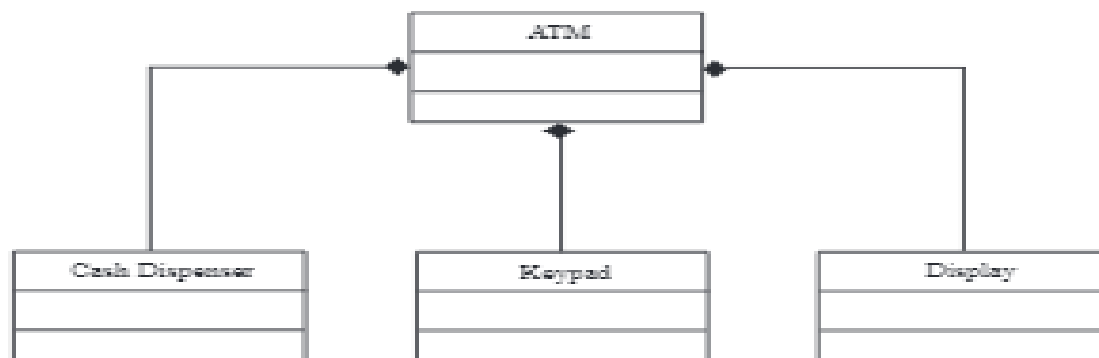


The following figure illustrates the aggregation relationship between two classes.



**Figure: Aggregation Relationship Between 'Customer' Class and 'DebitCard' Class.**

The following figure illustrates the composition relationship between two classes.



**Figure: Composition Relationship**

### Multiplicity

Multiplicity is used to indicate how many objects can be connected across an instance of an association in the form of explicit value or a range of values. This '**How many**' is often referred to as multiplicity of an associations role.

The following table illustrates multiple indicators along with their description.

Indicator	Description
0....1	Zero to one
1	Exactly one
*	Many
1...*	One to many
0...n or 1....n	Zero to n or one to n

### Example

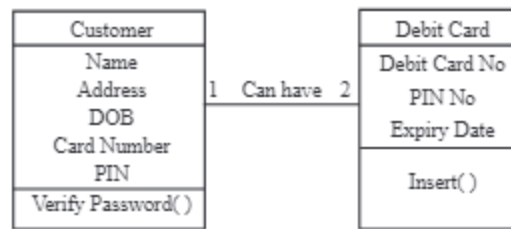


Figure: Multiplicity Indicator Between 'Customer' Class and 'Debit Card' Class

### (v) Inheritance Relationship

Inheritance relationship is also referred to as generalization relationship. It is a relationship between a more general element and more specific element. The more general element is called parent class or super class and the more specific element is called child or sub class. The parent describes a set of instances with common properties over the children.

The following figure illustrates generalization relationship.

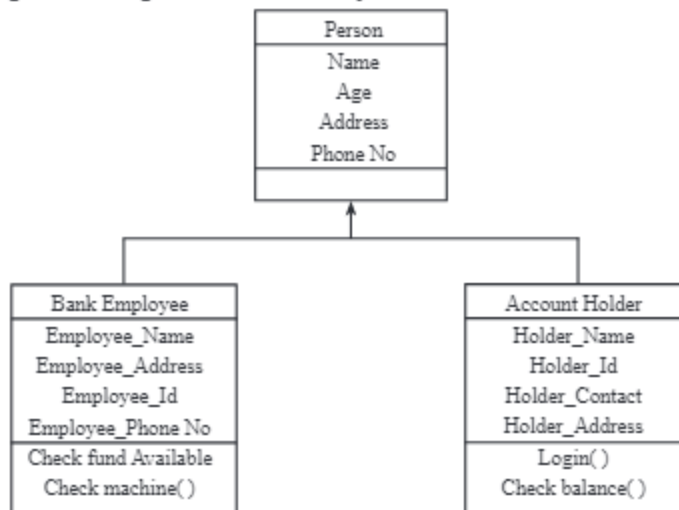


Figure: Inheritance Relationship

## Sequence Diagram

Sequence diagrams are used to model the behavioral aspects of the system. It is a type of interaction diagram that shows the interaction between a set of objects, the way they are linked to each other and also the messages exchanged between them. The objects interact among themselves with respect to time ordering. The sequence diagram depicts a graph with objects placed along the x-axis and message passing along y-axis. The y-axis proceeds downwards with respect to increasing time. Under each object, its life line (dashed line) is drawn along the y-axis, which indicates the existence of the particular object for a particular time period. Objects will be created or destroyed depending on the message stereotype received which can be either “create” or “destroy”.

The time duration during which a particular object is active is indicated by a vertical rectangular box called as “focus of control”.

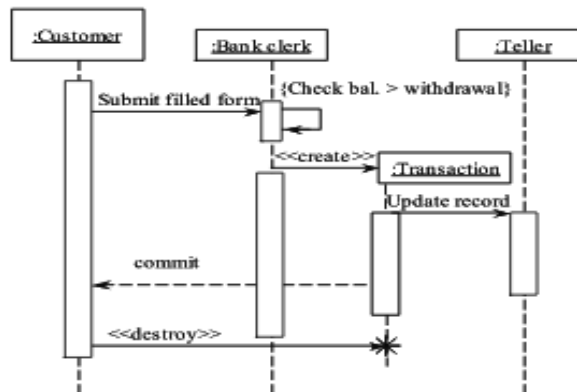


Figure: Modeling of Flow of Control by Time Ordering

## Basic Notations of Sequence Diagram

Sequence diagram basically involves the following.

1. Messages
2. Focus of Control and Activation Boxes
3. Returns
4. Messages to “self” or “this”
5. Instance creation
6. Object lifelines and object Destruction
7. Conditional messages
8. Mutually exclusive conditional messages
9. Messages to class objects

---

The Notation for these things is discussed in detail below:

1. **Messages:**

In this diagram, the messages are passed by using message expression on a link line between the objects. In addition to this, time ordering is also maintained from top to bottom. The diagrammatic representation is given below:

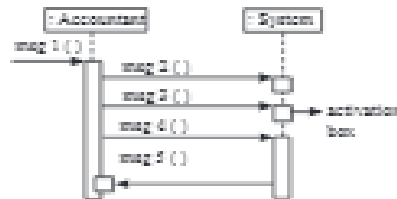


Figure: Message Notation in Sequences Diagram

2. **Focus of Control and Activation Boxes:**

Sequence diagram make use of activation boxes for focus of control. Focus of control can be defined as an operation on the stack having consistent blocking call. In the above given figure - "message notation in sequence diagram", the rectangular boxes below the instances are called activation boxes.

3. **Returns:**

Returns are indicated as a dashed open-headed line from the end of the activation box. Returns are messages containing the return information.

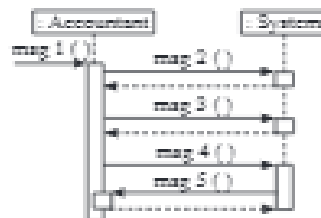


Figure: Returns

4. **Messages to "self" or "this"**

This diagram (sequence) makes use of nested activation box for indicating message to self from an object to itself. The figure below illustrates this representation.

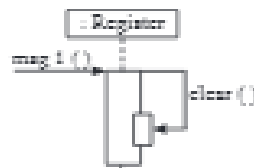


Figure: Showing Messages to Self

5. **Instance Creation**

The notation for creation instance in sequence diagram is shown in figure below,



Figure: Instance Creation