14CS/IT702

Hall Ticket Number:

IV/IV B.Tech (Regular / Supplementary) DEGREE EXAMINATION

January, 2021 — **Common to CSE & IT**

Seventh Semester — **Object Oriented Analysis And Design**

Time: Three Hours — **Maximum :** 60 Marks

Answer ALL Questions from PART-A. (1X12 = 12 Marks)

*Answer **ANY FOUR** questions from PART-B.* (4X12=48 Marks)

Part - A

1 Answer all questions (1X12=12 Marks)

a) Define actor?
An actor in the Unified Modeling Language (UML) "specifies a role played by a user or any other system that interacts with the subject." "An Actor models a type of role played by an entity that interacts with the subject but which is external to the subject."

b) Define scenario.
Scenario is a description of a specified sequence of actions. It depicts the behavior of objects undergoing a specific action series. The primary scenarios depict the essential sequences and the secondary scenarios depict the alternative sequences.

c) Define internal event.
Internal events are those that are passed among objects living inside the system. For example, a overflow exception generated by an object is an internal event. In UML, we can model four kinds of events namely: signals, calls, passing of time and change in state.

d) What is meant by messages?
Message is a named element that defines one specific kind of communication between lifelines of an interaction. The message specifies not only the kind of communication, but also the sender and the receiver. Sender and receiver are normally two occurrence specifications

e) Define swim lane.
A swimlane (or swimlane diagram) is used in process flow diagrams, or flowcharts, that visually distinguishes job sharing and responsibilities for sub-processes of a business process. Swimlanes may be arranged either horizontally or vertically.

f) Define logical design.
Logical design pertains to an abstract representation of the data flow, inputs, and outputs of the system.

g) Define signal event in UML.
A signal is an event that represents the specification of an asynchronous stimulus communicated between instances. A time event is an event that represents the passage of time. modeled by using the keyword 'after' followed by some expression that evaluates to a period of time which can be simple or complex

h) Differentiate a pattern and frame work.
Design pattern captures the static and dynamic structure and collaboration among key participants in software designs. They can be used across different domains. Framework: ... A design pattern is a type of pattern and is more like a concept, whereas a framework is something already coded to be used repetitively.

i) Define stake holder.
Stakeholder (in the context of software projects or systems) could be defined as a person, group or organization within or outside of the project or the system, sponsoring or having an interest in the project, or having a positive or negative influence in the project.

j) What are the three basic types of attributes?
Simple Attribute & Composite Attribute.
Single Valued Attribute & Multi-valued Attribute.
Stored Attribute & Derived Attribute.
Key Attribute & Non-key Attribute.

k) What is Reusability?
Reusability is the use of existing assets in some form within the software product development process. Assets are products and by products of the software development life cycle and include code, software components, test suites, designs and documentation

l) What is Multiplicity?

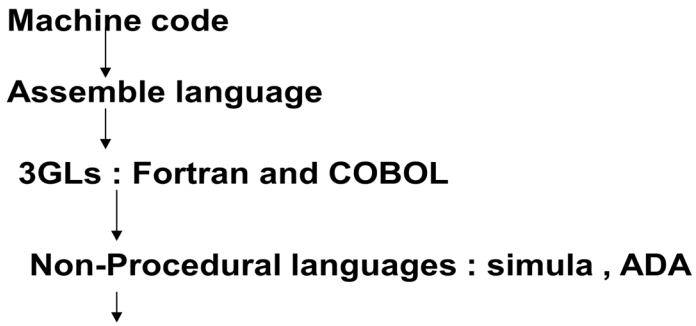
Multiplicity can be set for attributes, operations, and associations in a UML class diagram, and for associations in a use case diagram. The multiplicity is an indication of how many objects may participate in the given relationship or the allowable number of instances of the element.

Part – B

2 a) What is object-orientation? Explain briefly the concepts and origin of object orientation. 6M

Increasing Abstraction :

The earlier systems has a steady increase in the level of abstraction at which programmers have ability to operate . The increase in abstraction applies both to the activity of programming itself, and to the computer programs expected to perform. The following path shows increasing abstraction of programming.



Event-driven programming :

Early work on systems, simulation led directly to the OO paradigm of independent, collaborating objects that communicate via messages. A typical simulation task is to model the loading of vehicles onto a ship, in order to determine the safe way to do this. This simulation would be run many times under differing assumptions, for eg: the sequence of loading, the arrangement of vehicles on decks, the speed at which the vehicles are driven on to the ship…….etc.

this kind of task is difficult to do in 3GLs, designs of these languages are based on the underlying assumption that the structure controls the flow of execution. If program is written in 3GL, it must to have separate routines that test for , and respond to, a vast no of alternative conditions.

solution for this is, structure the program in a similar way to the problem situation it self: as set of independent s/w agents., each which represents the real-world system that is to be simulated.

In this way the complexity is solved from model of application domain and model of the s/w.

Modular s/w – In an OO system, classes have two kinds of definition:

From an External perspective, a class is defined in terms of its interface, which means that other objects need only know the services that offered by objects of that class and the signature used to request each service.

from an Internal perspective , a class is defined in terms of what it knows and what it can do- but only objects of that class need to know anything about this internal definition.

It says that OO system can be constructed so that the implementation of each part is largely independent of the implementation of the other parts., which is what modularity means.

Advantages :

It is easy to maintain a system built with Modular design, b'cas as changes to sub system affects very less on remaining system.

It is easy to upgrade a modular system.

It is easy to build a system which is reliable in use.

Each module provides useful and coherent package of functionality.

b) Draw a class diagram for each group of classes: school, playground, principal, school board, classroom book, student, teacher, cafeteria, restroom, computer, desk, chair, ruler, door, swing. Add associations and generalizations and show multiplicity wherever it is applicable. 6M

3 Explain use case realisation in detail with suitable example. 12M

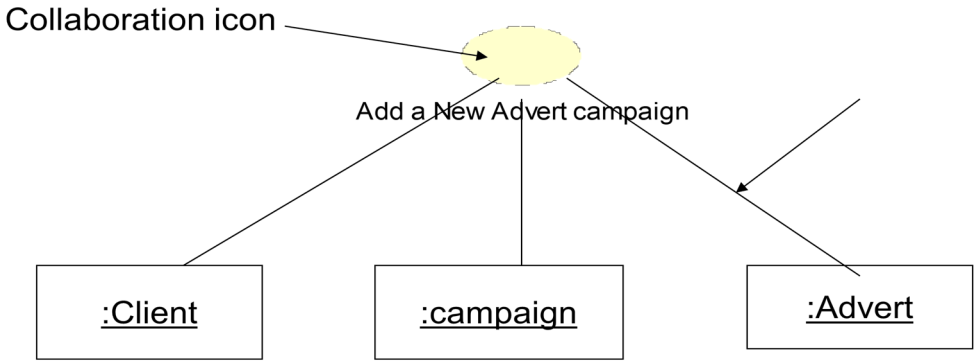
From use case to Sequence, collaboration , Object and class diagrams

To move from an initial use case to the implementation of software that entirely satisfies the requirements identified by the use case involves at least one iteration through all of the development activities, from requirements modelling to implementation.

In relation to a single use case, this activity is known as *use case realization. Conisider* the use case Add a new advert to a campaign, which is shown below.

Use case realization is nothing but an instance of a use case which involves the identification of a possible set of classes, together with an understanding of how those classes might interact to deliver the functionality of the use case. The set of classes is known as a *collaboration.*

The simplest representation of a collaboration as a set of classes is shown below



Here these three classes participate in the collaboration in other words, they can interact, when implemented as software, in such a way as to achieve the result described by the use case.
The another view of the following at the collaboration 'from the outside', which is used to show the relationship between a collaboration and the use case that it realizes.

Here we have similarity between the two icons: a collaboration appears to differ from a use case in this view only in being drawn with a dashed line instead of a solid one. However, the collaboration is not the same thing as the use case. Instead, it has a relationship with the use case that is shown here by the *dependency* arrow. Here, this means that the definitions of these classes must maintain a reference to the collaboration, which in turn must maintain a reference to the use case.

Means this type of representation doesn't includes , how they interact nor how they relate to other parts of the model, but this is just one view of a collaboration.

In detail this can be realized by using the following UML diagrams.

1. Sequence diagram
2. Collaboration diagram
3. Object diagram
4. Class diagram

4 a) Compare and contrast the sequence diagrams and collaboration diagrams 6M

Sequence Diagrams	Collaboration Diagrams
The sequence diagram represents the UML, which is used to visualize the sequence of calls in a system that is used to perform a specific functionality.	The collaboration diagram also comes under the UML representation which is used to visualize the organization of the objects and their interaction.
The sequence diagram are used to represent the sequence of messages that are flowing from one object to another.	The collaboration diagram are used to represent the structural organization of the system and the messages that are sent and received.
The sequence diagram is used when time sequence is main focus.	The collaboration diagram is used when object organization is main focus.

analysis activities. for depicting simpler interactions of the smaller number of objects.

b) What are the main differences between Algorithmic and Non-algorithmic techniques in operation specification? 6M

Non-algorithmic approaches

A non-algorithmic approach concentrates on describing the logic of an operation as a black box. In an object-oriented system this is generally preferred for two reasons.

First, classes are usually well-encapsulated, and thus only the designers and programmers responsible for a particular class need concern themselves with internal implementation details. Collaboration between different parts of the system is based on public interfaces between. Classes and sub-systems implemented as operation signatures (or message protocols). As long as the signatures are not changed, a change in the implementation of a class, including the way its operations work, has no effect on other parts of the system•

Second, the relatively even distribution of effort among the classes of an object-oriented system generally results in operations that are small and single-minded. Since the processing carried out by anyone operation is simple, it does not require a complex specification.

- Decision Tables
- Preconditions and Post conditions

Algorithmic approach

An *algorithm* describes the internal logic of a process or decision by breaking it down into small steps .The level of detail to which this is done varies greatly, depending on the information available at the time and on the reason for defining it.

An algorithm also specifies the sequence in which the steps are performed. In the field of computing and information systems, algorithms are used either as a *description* of the way in which a programmable task is currently carried out, or as a *prescription* for a program to automate the task. This dual meaning reflects the differing perspectives of analysis (understanding a problem and determining what must be done to achieve a solution) and design (the creative act of imagining a system to implement a solution).

An algorithmic technique is almost always used during method design, because a designer is concerned with the efficient implementation of requirements, and must therefore select the best algorithm available for the purpose. But algorithms can also be used with an analysis intention.

A major difference here is that the analyst no need to worry about efficiency, since the algorithm need only illustrate accurately the results of the operation.

- Control structures in Algorithm
- Structured English
- Pseudocode
- Activity Diagrams

5 a) Clearly explain the notation and model elements and to draw state chart diagram 6M

Following are the main purposes of using State chart diagrams −

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model the states of an object.

Statechart diagram is used to describe the states of different objects in its life cycle. Emphasis is placed on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately.

Statechart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.

Before drawing a Statechart diagram we should clarify the following points −

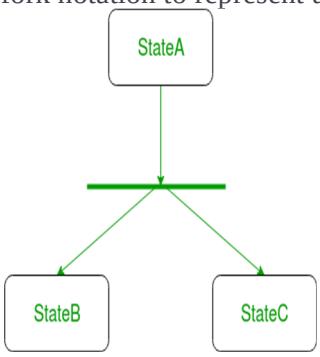
- Identify the important objects to be analyzed.
- Identify the states.
- Identify the events.

1. **Initial state –** We use a black filled circle represent the initial state of a System or a class.

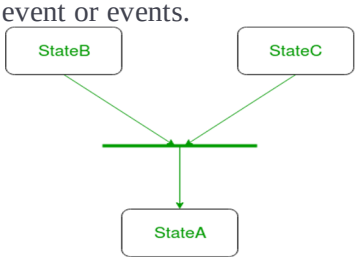
2. **Transition –** We use a solid arrow to represent the transition or change of control from one state to another. The arrow is labelled with the event which causes the change in state.

3. **State –** We use a rounded rectangle to represent a state. A state represents the conditions or circumstances of an object of a class at an instant of time.

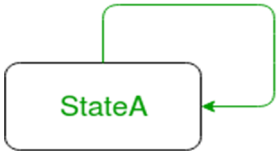
4. **Fork –** We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent state and outgoing arrows towards the newly created states. We use the fork notation to represent a state splitting into two or more concurrent states.



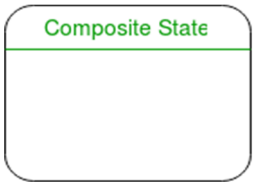
5. **Join –** We use a rounded solid rectangular bar to represent a Join notation with incoming arrows from the joining states and outgoing arrow towards the common goal state. We use the join notation when two or more states concurrently converge into one on the occurrence of an event or events.



6. **Self transition –** We use a solid arrow pointing back to the state itself to represent a self transition. There might be scenarios when the state of the object does not change upon the occurrence of an event. We use self transitions to represent such cases.



7. **Composite state –** We use a rounded rectangle to represent a composite state also.We represent a state with internal activities using a composite state.

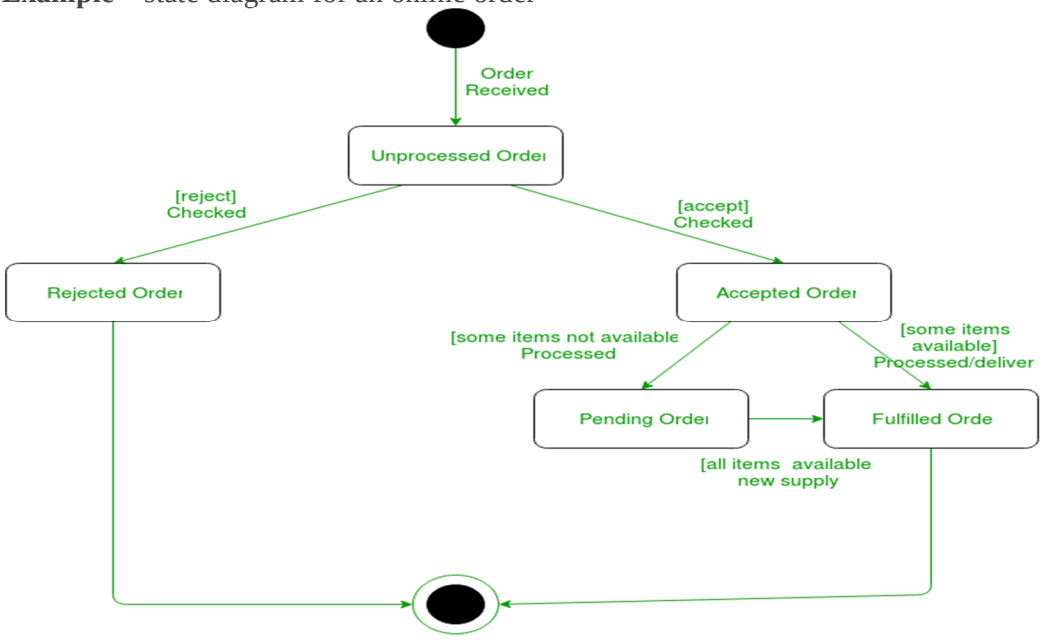


8. **Final state –** We use a filled circle within a circle notation to represent the final state in a state machine diagram.

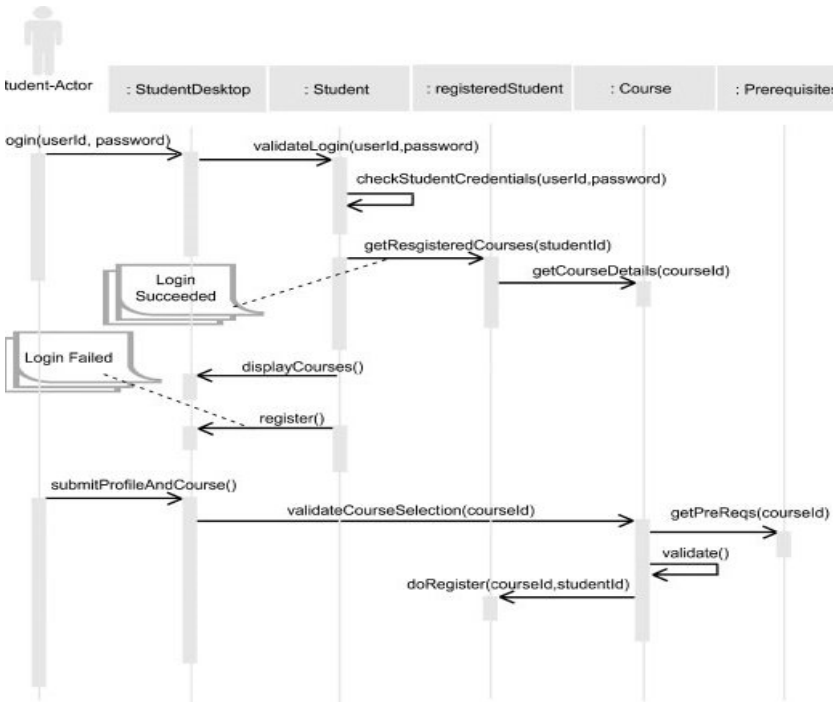
Steps to draw a state diagram –

1. Identify the initial state and the final terminating states.
2. Identify the possible states in which the object can exist (boundary values corresponding to different attributes guide us in identifying different states).
3. Label the events which trigger these transitions.

Example – state diagram for an online order –



b) Draw the sequence diagram for student course registration system. 6M



6 a) Discuss about qualities and Objectives of Good Analysis and Design? 6M

The cost of fIxing faults in a system 'increases as the system progresses through the system's development life cycle. If an error occurs in the analysis of a system, it is cheaper to fix it during the analysis phase than it is later when that error may have propagated through numerous aspects of the design and implementation. It is most expensive to fix it after the system has been deployed and the error may be reflected in many different parts of the system. The quality of the design is, therefore, dependent to a large extent on the quality of the analysis.

Some methodologies have explicit quality criteria that can be applied to the products of every stage of the life cycle, but these quality criteria typically check syntactic aspects of the products, that is whether the notation is correct in diagrams, rather than semantic aspects, that is whether the diagrams correctly represent the organization's requirements.

Objectives of Analysis

- ***correct scope,***
- ***completeness,***
- ***correct content and***
- ***consistency.***

The quality of the design will clearly be reflected in the quality of the finished system that is delivered to the clients. Moreover, in the same way as the quality of analysis affects the work of designers, the quality of the design has an impact on the work of the programmers who will write the program code in order to implement the system based on the design.

Some of the criteria given below for a good design will bring benefits to the developers, while some will provide benefits for the eventual users of the system.

Objectives of Design

- ***Functional.***
- ***Efficient***
- ***Economical***
- ***Reliable.***
- ***Secure***
- ***Flexible.***
- ***Manageable***
- ***Maintainable***
- ***Usable.***
- ***Reusable.***

b) Explain in detail about criteria for good design? 6M

Coupling and cohesion

- *interaction coupling*
- *Inheritance coupling*
- *Operation cohesion*
- *Class cohesion*
- *Specialization cohesion*

Liskov substitution principle

Design Clarity
Don’t over Design
Control inheritance
Keep messages and operations simple
Design Volatility
Evaluate by scenario
Design by delegation
Keep classes seperate

7 a) Explain about processor allocation. 6M

In the case of single-user system it is almost always appropriate for the complete system to operate on a single computer. The software for a multi-user information system may be installed on many computers that all use a common file-server. More complex applications sometimes require the use

of more than one type of computer, where each provides a specialized kind of processing capability for a specific sub-system.

An information system may also be partitioned over several processors either because sub-systems must operate concurrently, or because some parts of the application need to operate in different locations. Information systems that use the Internet or company intranets for their communications are being built more frequently. Such distributed information systems operate on diverse computers and operating systems.

The allocation of a system to multiple processors on different platforms involves the following steps.

- The application should be divided into sub-systems.
- Processing requirements for each sub-system should be estimated.
- Access criteria and location requirements should be determined.
- Concurrency requirements for the sub-systems should be identified.
- Each sub-system should be allocated to an operating platform-either general purpose (PC or workstation) or specialized (embedded micro-controller or specialist server).
- Communication requirements between sub-systems should be determined.
- The communications infrastructure should be specified.

The estimation of processing requirements requires careful consideration of such factors as event response times, the data throughput that is needed, the nature of the I/O that is required and any special algorithmic requirements

b) Explain about data management issues. 6M

Suitable data management approaches for an information system can vary from simple file storage and retrieval to sophisticated database management systems of various types. **In** some applications where data has to be accessed very rapidly, the data may be kept in main memory while the system executes. However, most data management is concerned with storing data, often large volumes, so that it may be accessed at a later stage either by the same system or by another.

Database management systems (DBMS) provide the following facilities that are useful in many applications.

- different views of the data by different users,
- control of multi-user access,
- distribution of the data over different platforms,
- security,
- enforcement of integrity constraints,
- access to data by various applications,
- data recovery,
- portability across platforms,
- data access via query languages and
- query optimization.

DBMS exhibits a significant performance overhead and the standard data access mechanisms may be inappropriate for specialized systems.

Relational DBMS is likely to be appropriate if there are large volumes of data with varying (perhaps ad hoc) access requirements.

Object-oriented DBMS is more likely to be suitable if specific transactions require fast access or if there is a need to store complex data structures and there is not a need to support a wide range of transaction types.

A third type of DBMS is emerging-the object-relational DBMS-that is similar to an object oriented DBMS in its support for complex data structures, but that also provides effective querying facilities. In some systems there may be different data management requirements for different sub-systems and it may be best then to use a mix of DBMS types.

8 a) Explain User Interface Design Patterns and What are the Five steps to prepare a Statechart to model a User Interface? 6M

Structural patterns address issues concerned with the way in which classes and objects are organized. Structural patterns offer effective ways of using object-oriented constructs such as inheritance, aggregation and composition to satisfy particular requirements. If there , a requirement for a particular aspect of the application to be extensible. In order to achieve this, the application should be designed with constructs that minimize the sideeffects of future change. Alternatively it may be necessary to provide the same interface for a series of objects of different classes also.

Eg : Composite Pattern

Composite Pattern

To apply Composite structural pattern in a design for the Agate case study , consider if further work is required to design a multimedia application that can store and play components of an advert.

Here an advert is made up of sound clips and video clips each of which may be played individually or as part of an advert. The classes SoundClip and VideoClip have attributes and operations in common and it is appropriate that these classes are subclassed from MediaClip as shown below as MediaClip Inheritance hierachy.

b) Explain Planning a Strategy for Reuse? 6M

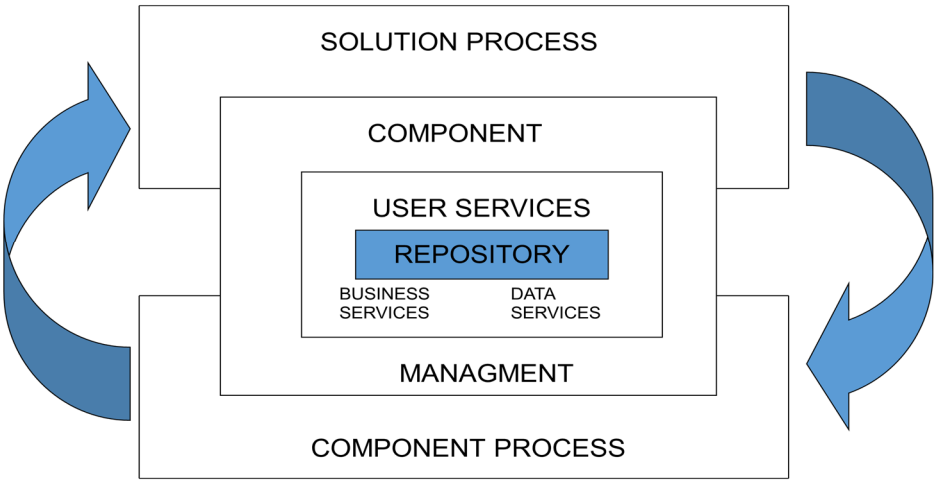
The SELECT Perspective

Allen and Frost describes the SELECT Perspective approach to the developers of reusable components. At the level of practical techniques, this includes guidelines for the modelling of business-oriented components and for wrapping legacy software in component wrappers. They distinguish between reuse at the level of component packages, which consist of executable components grouped together, and service packages, which are abstractions of components that group together business services.

The focus of this approach is to identify the services that belong together and classes that implement them. Service classes in a single package should have a high level of internal interdependency and minimal coupling to classes in other packages .

In order to develop reusable components while achieving the development of a system to meet users' needs, the Perspective approach breaks the development process into two parts:

1. The solution process
2. The component process.



- The solution process focuses on specific business needs and delivering services to meet the users' requirements. Its products have immediately definable business value. During the solution process, developers will draw on the component process in their search for reusable components that can be applied to the project.
- The component process focuses on developing reusable components in packages that group together families of classes to deliver generic business services. During the component process, the developers produce components that can be reused in the solution process. The component process also searches out opportunities to reuse services from existing legacy systems and legacy databases and from other packages of components.

9 a) Define Deployment Diagram? Draw Deployment diagram for Railway Reservation System? 6M

A UML deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).

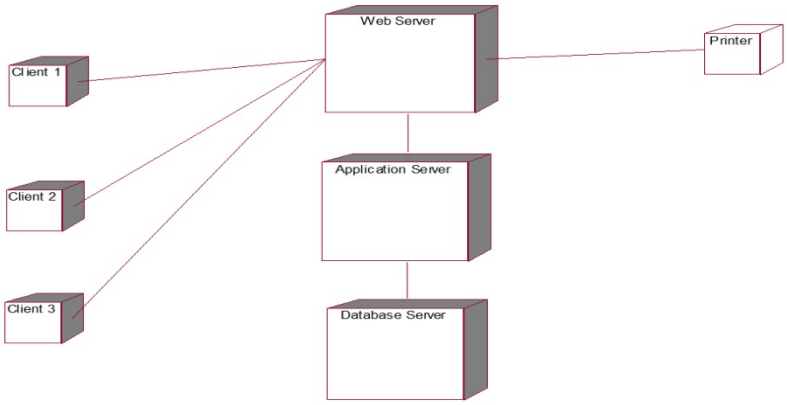
- What existing systems will the newly added system need to interact or integrate with?
- How robust does system need to be (e.g., redundant hardware in case of a system failure)?
- What and who will connect to or interact with system, and how will they do it
- What middleware, including the operating system and communications approaches and protocols, will system use?
- What hardware and software will users directly interact with (PCs, network computers,

browsers, etc.)?

- How will you monitor the system once deployed?
- How secure does the system needs to be (needs a firewall, physically secure hardware, etc.)?

Purpose

- They show the structure of the run-time system
- They capture the hardware that will be used to implement the system and the links between different items of hardware.
- They model physical hardware elements and the communication paths between them
- They can be used to plan the architecture of a system.
- They are also useful for Document the deployment of software components or nodes



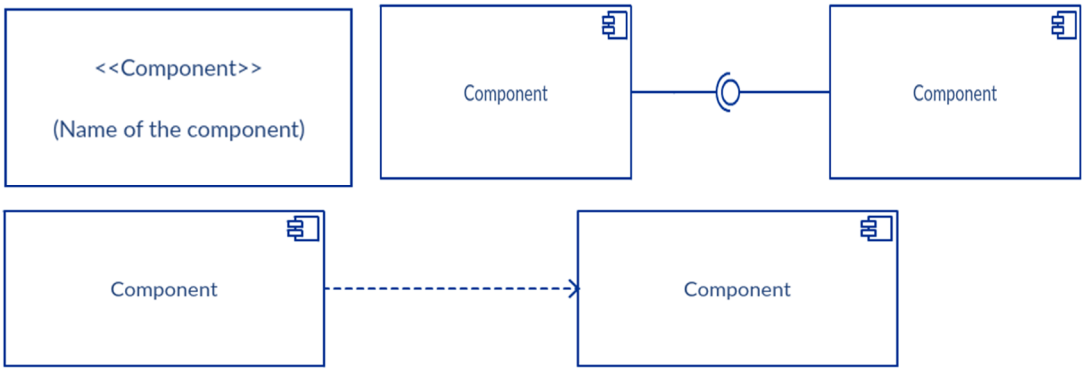
b) Define Component Diagram? Draw Component diagram for Library Management System. 6M

Component diagrams are used to visualize the organization of system components and the dependency relationships between them. They provide a high-level view of the components within a system.

The components can be a software component such as a database or user interface; or a hardware component such as a circuit, microchip or device; or a business unit such as supplier, payroll or shipping.

Component diagrams

Are used in Component-Based-Development to describe systems with Service-Oriented-Architecture
Show the structure of the code itself
Can be used to focus on the relationship between components while hiding specification detail
Help communicate and explain the functions of the system being built to stakeholders



LIBRARY MANAGEMENT SYSTEM