14CS/IT702

Hall Ticket Number:

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

October, 2018 **Common to CSE & IT**

Seventh Semester **Object Oriented Analysis and Design**

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

Answer Question No.1 compulsorily. (1X12 = 12 Marks)

Answer ONE question from each unit. (4X12=48 Marks)

1. Answer all questions (1X12=12 Marks)

a) Define stereotype.
A stereotype is one of three types of extensibility mechanisms in the Unified Modelling Language (UML), the other two being tags and constraints.

b) Justify how messages are different from function calls.
Calling a function directly.
Access an object and seek among it's properties for the method to call.

c) Distinguish between abstract class and concrete class
The only real difference is that a concrete class can be instantiated because it provides (or inherits) the implementation for all of its methods. An abstract class cannot be instantiated because at least one method has not been implemented. Abstract classes are meant to be extended.

d) Define Association class?
An association class is essentially a class attached to an association; the association itself is modelled as a class.

e) Why is CRC useful?
CRC card is an index card used to represent
i) A class
ii) The responsibilities of the class and
iii) The interaction between classes.

f) Define include and extend dependency relationships.
Include-mandatory
Extend-optional

g) what is reusability.
Reusability is an attribute of software quality. By measuring reusability we can measure software quality. The authors have proposed a new metric to measure there usability of interfaces in object oriented programming

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

i) Differentiate logical and physical design?
A logical design is a conceptual, abstract design. You do not deal with the physical implementation details yet; you deal only with defining the types of information that you need. The process of logical design involves arranging data into a series of logical relationships called entities and attributes.

j) Differentiate action and activity?
An activity is the specification of a parameterized sequence of behaviour.

An action represents a single step within an activity.

Constraints can be attached to an action.

k) Define actor?
An actor in the Unified Modelling 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

l) What is aggregation?
UML Aggregation. Shared aggregation (aggregation) is a binary association between a property and one or more composite objects which group together a set of instances. It is a "weak" form of aggregation when part instance is independent of the composite.

UNIT I

2. a) What is modelling? Explain importance of modelling? 6M

MODEL : It is a simplification of reality. They are very useful in the following ways.
- A model is a quicker and easier to build.
- A model can be used in simulations to learn more about things in representation.
- A model gives clear understanding of a problem.
- A model is an abstraction, mean we can choose which are to be represented, and which are to be suppressed.
- A model can represent real or imaginary things from any domain.
- A useful model has just the right amount of detail and structure.

In UML there are a number of concepts that are used to describe systems and the ways in which they can be broken down and modelled. A *system* is the overall thing that is being modelled, such as the Agate system for dealing with clients and their advertising campaigns.

A *sub-system* is a part of a system, consisting of related elements, example the Campaigns sub-system of the Agate system. A *model* is an abstraction system or sub-system from a particular perspective or *view*. An example would be use case view of the Campaigns sub-system, which would be represented by a model containing use case diagrams, among other things. A model is complete and consistent at the level of abstraction that has been chosen. Different views of a system can presented in different models, and a *diagram* is a graphical representation of a set of elements in the model of the system.

b) Define activity diagram? Draw activity diagram for issue book. 6M

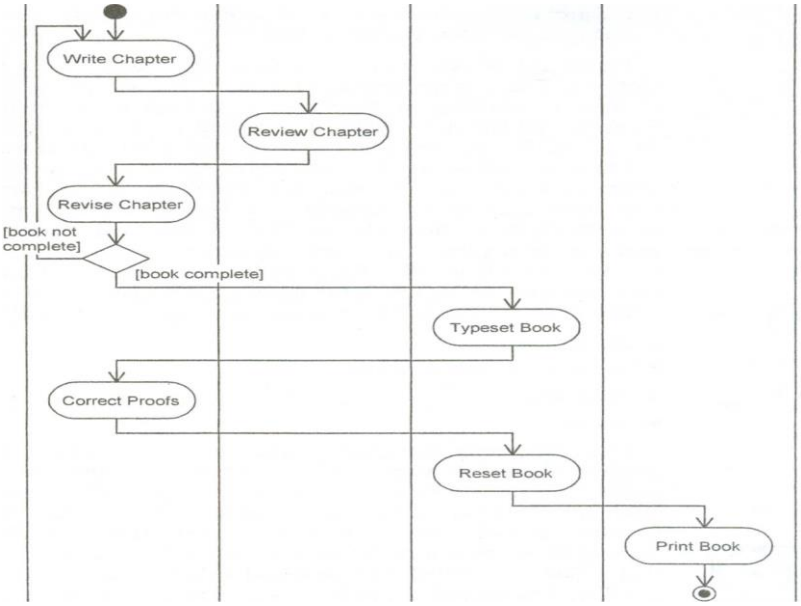
Activity diagrams can be used to model different aspects of a system. At a high level, they can be used to model business activities in an existing or potential system. For this purpose they may be used early in the system development lifecycle.

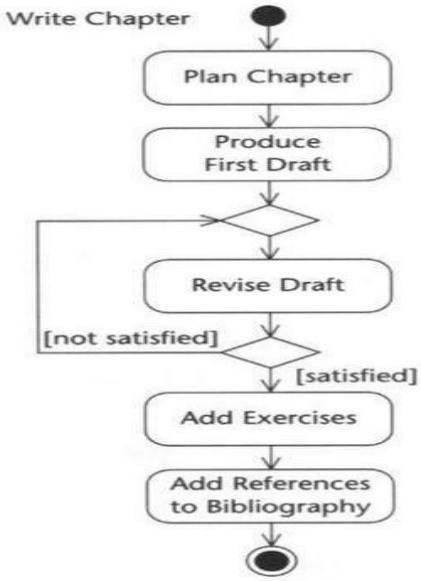
They can be used to model a system function represented by a use case, possibly using object flows to show which objects are involved in a use case. This would be done during the stage of the life cycle when requirements are being elaborated.

They can also be used at a low level to model the detail of how a particular operation is carried out, and are likely to be used for this purpose late in analysis or during the design stage of a project. Activity diagrams are also used within the Unified Software Development Process (USDP) - to model the way in which the activities of the USDP are organized and relate to one another in the software development lifecycle.

Mainly, activity diagrams are used for the following purposes:
- to model a task (in business modelling for instance);
- to describe a system function that is represented by a use case;
- in operation specifications, to describe the logic of an operation;
- in USDP to model the activities that make up the lifecycle.

Activity diagrams are really most useful to model business activities in the early stages of a project. For modelling operations, interaction sequence diagrams are closer to the spirit of object-orientation. However, there may be occasions when the analyst wants to model the activities that must be carried out, but has not yet identified the objects or classes that are involved or assigned responsibilities to them. In such circumstances, activity diagrams may be an appropriate tool to use.





(OR)

3. a) Draw the class diagram for the following scenario: A customer, characterized by his/her name and phone number, may purchase reservations of tickets for a performance of a show. A reservation of tickets, annotated with the reservation date, can be either a reservation by subscription, in which case it is characterized by a subscription series number, or an individual reservation. A subscription series comprehends at least 3 and at most 6 tickets; an individual reservation at most one ticket. Every ticket is part of a subscription series or an individual reservation, but not both. Customers may have many reservations, but each reservation is owned by exactly one customer. Tickets may be available or not, and one may sell or exchange them. A ticket is associated with one specific seat in a specific performance, given by date and time, of a show, which is characterized by its name. A show may have several performances. 6M

Basic Notation
Relationships among them (association, dependency, generalization, association class)
Class, Abstract class, interfaces etc...

b) A midterm exam is prepared by the instructor and taken by each of the students in the class. In special cases where the student misses the test the student has to take the makeup test. Draw a use case diagram for this situation. 6M

Actor, usecases, associations, dependencies (include,extends), generalization, system boundary.

UNIT II

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

State Chart diagram :

The statechart is a versatile technique, and can be used within an object-oriented approach for other purposes than the modelling of object life cycles. A statechart diagram shows the states of a single object, the events or messages that cause a transition from one state to another , and the actions that result from a state change. As in Activity diagram , statechart diagram also contains special symbols for start state and stop state.

States and Events :

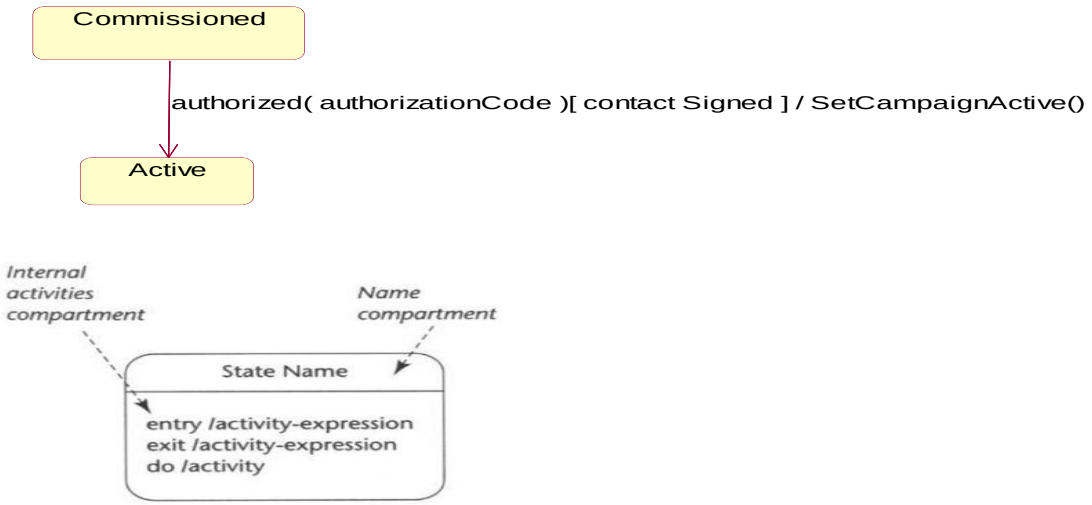
All objects will have a state in a system. The current state of an object is a result of the events that have occurred to the object, and is determined by the current value of the object's attributes and the links that it has with other objects. Some attributes and links of an object are significant for the determination of its state while others are not.

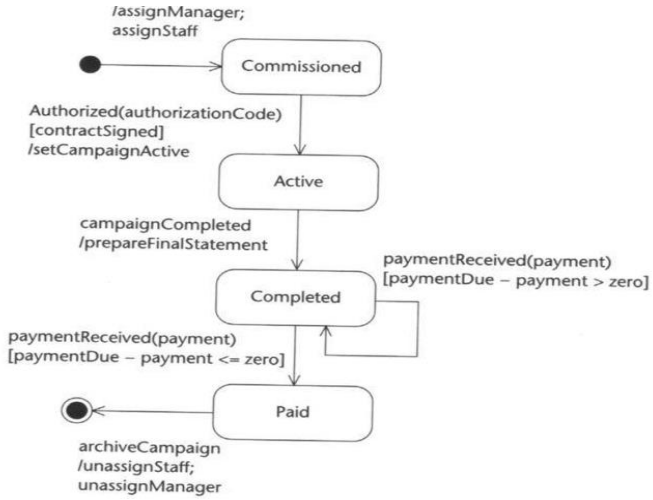
A state is a condition during the life of an object or an interaction during which it satisfies some condition, performs some action or waits for some event. Conceptually, an object remains in a state for an interval of time. The possible states that an object can occupy are limited by its class. Objects of some classes have only one possible state.

Movement from one state to another is called a *transition*, and is triggered by an *event.* When its triggering event occurs a transition is said to *fire.* A transition is shown as a solid arrow from the source state to the target state.

An event is an occurrence of a stimulus that can trigger a state change and that is relevant to the object or to an application.

Events can be grouped into several general types. A *change event* occurs when a condition becomes true. This is usually described as a Boolean expression, which means that it can take only one of two values: true or false. Change events are annotated by the keyword *when* followed by the Boolean expression in parenthesis. This form of conditional event is different from a guard condition that is only evaluated at the moment that its associated event fires.



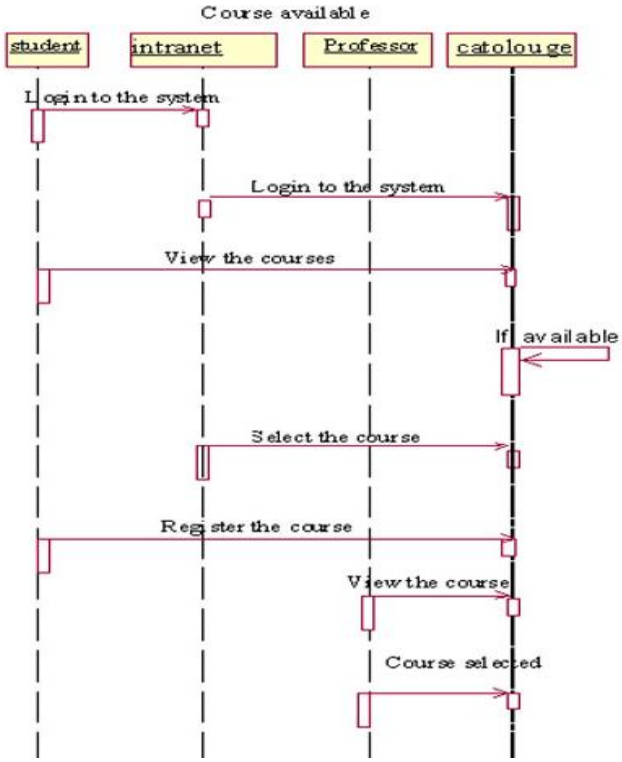


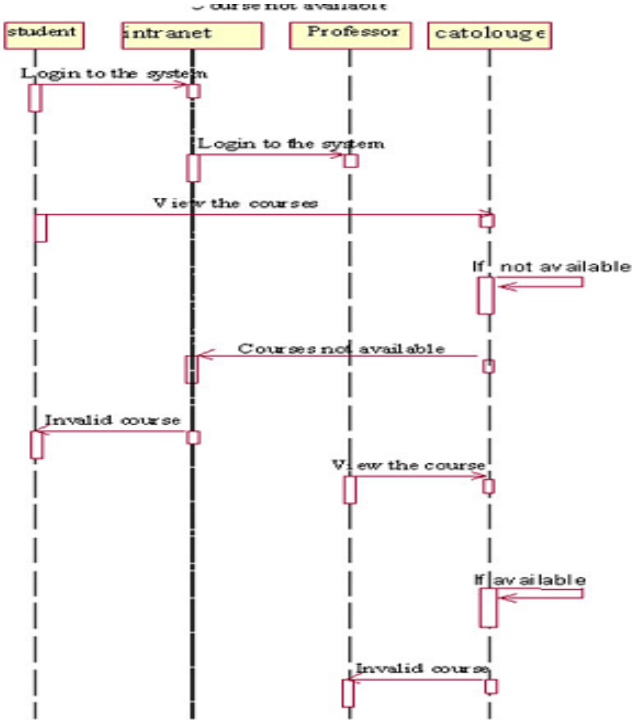
b) Explain the stereotypes in analysis class diagram with examples. 6M

Boundary classes
Entity Classes
Control Classes

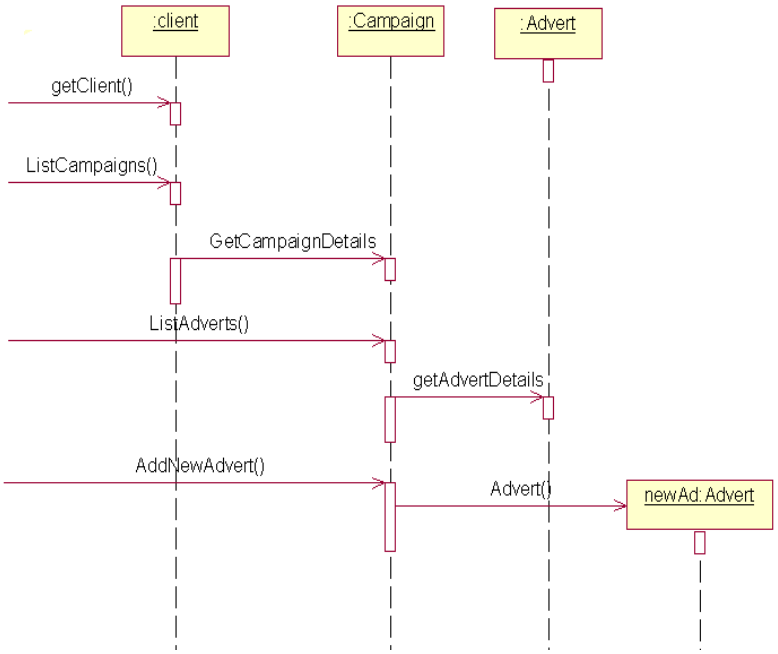
(OR)

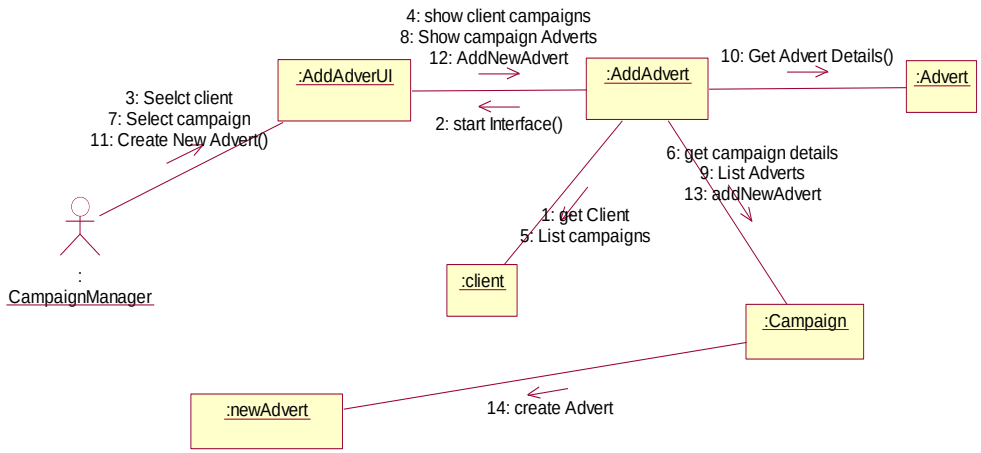
5. a) Draw the sequence diagram for student course registration system. 6M





b) Differentiate sequence and collaboration diagram and explain with an example? 6M





Sequence Diagram	Collaboration Diagram
The sequence diagram is a UML representation to visualize the sequence of calls in a system to perform a specific functionality.	The collaboration diagram is a UML representation to visualize the organization of the objects and their interaction
The sequence diagram represents the sequence	The collaboration diagram represents the

of messages flowing from one object to another.	structural organization of the system and the messages sent and received.
If the time sequence is important, the sequence diagram can be used.	If the object organization is important, then the collaboration diagram can be used.

UNIT III

6. a) Define pattern? Explain about types of Design patterns and with examples? 6M

Creational patterns---2M Structural patterns---2M Behavioural patterns---2M

A **pattern** (or design **pattern**) is a written document that describes a general solution to a design problem that recurs repeatedly in many projects. Software designers adapt the **pattern** solution to their specific project.

Types of Design Patterns

Patterns are classified according to their scope and purpose into the following three main categories.

1. *Creational patterns*
2. *Structural patterns*
3. B*ehavioural patterns.*

The scope of a pattern may be primarily at either the class level or at the object level. Patterns that are principally concerned with objects describe relationships that may change at run-time and hence are more dynamic. Patterns that relate primarily to classes tend to be static and identify relationships between classes and their subclasses that are defined at compile-time.

Changeability involves several different aspects - maintainability, extensibility, restructuring and portability.

Maintainability is concerned with the ease with which errors in the information system can be corrected.

Extensibility addresses the inclusion of new features and the replacement of existing components with new improved versions. It also involves the removal of unwanted features.

Restructuring focuses on the reorganization of software components and their relationships to provide increased flexibility.

Portability deals with modifying the system so that it may execute in different operating environments, such as different operating systems or different hardware

Creational patterns

A creational design pattern is concerned with the construction of object instances. In general, creational patterns separate the operation of an application from how its objects are created. This decoupling of object creation from the operation of the application gives the designer considerable flexibility in configuring all aspects of object creation. This configuration may be dynamic or static .

For example, when dynamic configuration is appropriate, an object-oriented system may use composition to make a complex object by aggregating simpler component objects. Depending upon circumstances different components may be used to construct the composite and, irrespective of its components, the composite will fulfill the same purpose in the application.

Creating composite objects is not simply a matter of creating a single entity but also involves creating all the component objects. The separation of the creation of a composite object from its use within the application provides design flexibility. By changing the method of construction of a composite object, alternative implementations may be introduced without affecting the current use.

Eg : Singleton Pattern

Structural patterns

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 side effects 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

Behavioural patterns

Behavioural patterns addresses the problems that arise when assigning responsibilities to classes and when designing algorithms. Behavioural patterns specifies static relationships between objects and classes and how the objects of one class communicates with another. Behavioural patterns may use inheritance structures to spread behaviour across the subclasses or they may use aggregation and composition to build complex behaviour from simpler components.

Eg: The State pattern uses both of these techniques.

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

Qualities

Correct scope. The scope of a system determines what is included in that system and what is excluded. It is important, first that the required scope of the system is clearly understood, documented and agreed with the clients, and second that every-thing that is in the analysis models *does* fall within the scope of the system.

Completeness. Just as there is a requirement that everything that is in the analysis models is within the scope of the system, so everything that is within the scope of the system should be documented in the analysis models. Everything that is known about the system from the requirements capture should be documented and included in appropriate diagrams. Often the completeness of the analysis is dependent on the skills and experience of the analyst. Knowing what questions to ask in order to list out requirements comes with time and experience. However, analysis patterns and strategies, can help the less experienced analyst to identify likely issues.

Correct content. The analysis documentation should be correct and accurate in what it describes. This applies to textual information, diagrams and also to quantitative features of the non-functional requirements. Examples include correct descriptions of attributes and any operations that are known at this stage, correct representation of associations between classes, particularly the multiplicity of associations, and accurate information about volumes of data.

Consistency. Where the analysis documentation includes different models that refer to the same things (use cases, classes, attributes or operations) the same name should be used consistently for the same thing. Errors of consistency can result in errors being made by designers, for example, creating two attributes with different names that are used in different parts of the system but should be the same attribute.

(OR)

7. a) Explain in detail about criteria for good design? 6M

Coupling and cohesion
Interaction Coupling
Inheritance Coupling
Operation Cohesion
Class Cohesion
Specialization Cohesion
Design Clarity
Don't Over-Design.
Control Inheritance Hierarchies
Keep Messages and Operations Simple.
Design Volatility.
Evaluate by Scenario.
Design by Delegation.
Keep Classes Separate.

b) what are the differences between system design and detailed design? 6M

System design

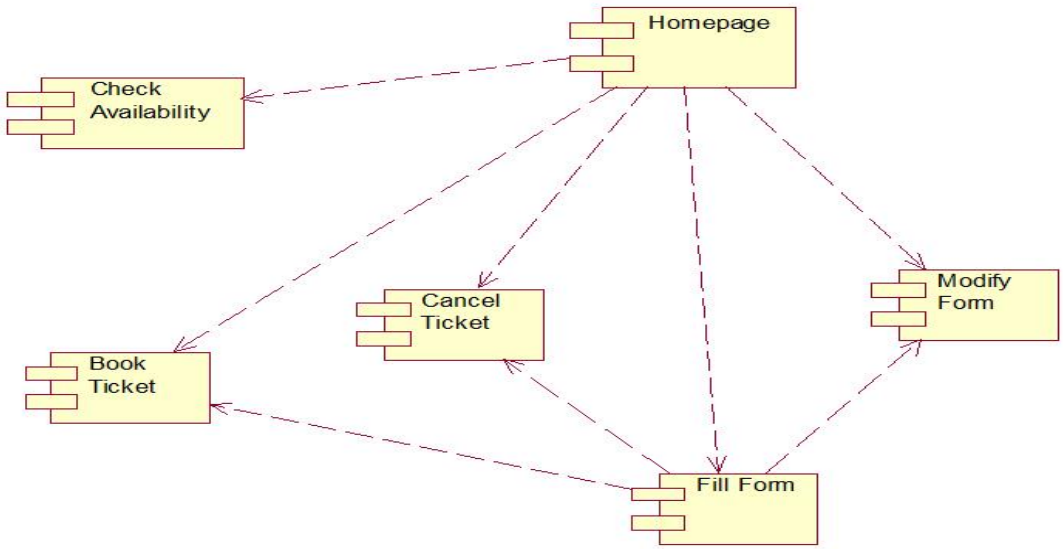
Sub-system and major components are identified
Any inherent concurrency is identified
Sub systems are allocated to processors
A data management strategy is selected

Code development standards are specified
A strategy and standard s for human computer interaction are chosen
The total aspects of the application are planned
Test plans are produced
Priorities are set for design trade offs
Implementation requirements are identified.

Detailed design
Subsystems
Layering and partitioning
Partitioned subsystems
Architectures for distributed systems.

UNIT IV

8. a) Define Component Diagram? Draw Component diagram for Railway Reservation System?. 6M



Component diagrams are different in terms of nature and behaviour. Component diagrams are used to model the physical aspects of a system.
Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

The purpose of the component diagram can be summarized as

- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.

Before drawing a component diagram, the following artifacts are to be identified clearly −

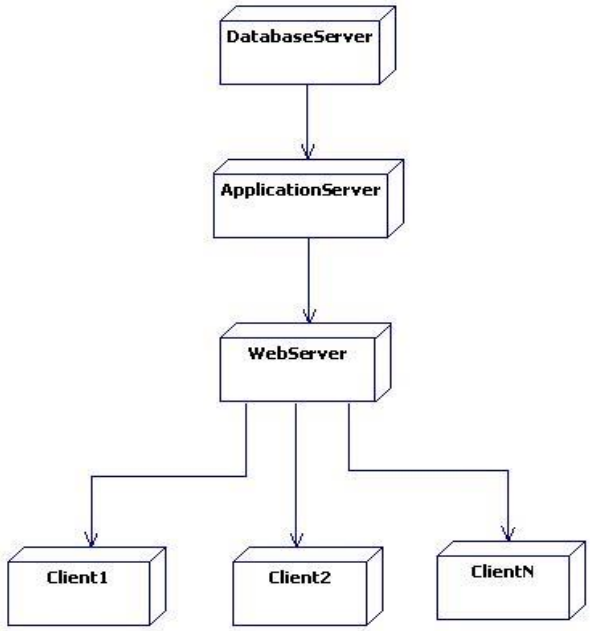
- Files used in the system.
- Libraries and other artifacts relevant to the application.
- Relationships among the artifacts.

After identifying the artifacts, the following points need to be kept in mind.

- Use a meaningful name to identify the component for which the diagram is to be drawn.

- Prepare a mental layout before producing the using tools.
- Use notes for clarifying important points.

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



Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed.

Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

UML is mainly designed to focus on the software artifacts of a system. However, these two diagrams are special diagrams used to focus on software and hardware components.

Most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on the hardware topology of a system. Deployment diagrams are used by the system engineers.

The purpose of deployment diagrams can be described as −

- Visualize the hardware topology of a system.
- Describe the hardware components used to deploy software components.
- Describe the runtime processing nodes.

Deployment diagrams are useful for system engineers. An efficient deployment diagram is very important as it controls the following parameters −

- Performance
- Scalability
- Maintainability
- Portability

Before drawing a deployment diagram, the following artifacts should be identified −

- Nodes
- Relationships among nodes

(OR)

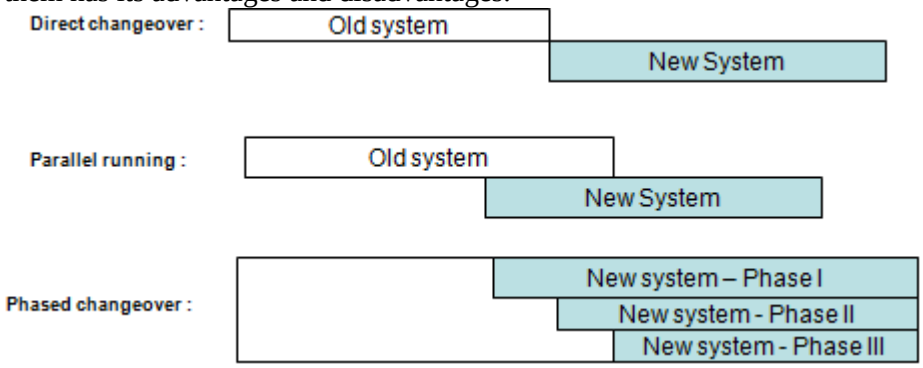
9. a) Explain about different types of implementation strategies. 6M

There are four main strategies for switching over to the new system:

1. Direct changeover;

2. Parallel running;
3. Phased changeover;
4. Pilot project.

The following figure shows three of these changeover strategies in diagram form. Each of them has its advantages and disadvantages.



b) Explain the architecture of presentation layer in designing boundary classes. 6M

Model
View
Controller
Logical Design
Interface design
Reuse
Prototyping the user interface
Designing the classes
Modelling the interaction involved in the interface and
Modelling the control of the interface using state charts