IV/IV B.Tech (Regular) DEGREE EXAMINATION

Nov, 2022 **Information Technology**
Seventh Semester **Object Oriented Analysis and Design**
Time: Three hours **maximum:** 50 Marks

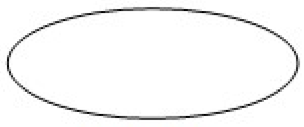
Answer question No.1 compulsorily. (1*10=10 Marks)
Answer one question from each unit (4*10=40 Marks)

a) Define Model. 1M

Object-oriented modeling and design is a way of thinking about problems using models organized around real world concepts. The fundamental construct is the object, which combines both data structure and behavior.

b) Define use case. 1M

Use case diagrams are a way to capture the system's functionality and requirements in UML diagrams. A use case represents a distinct functionality of a system, a component, a package, or a class. An actor is an entity that initiates the use case from outside the scope of a use case.



USECASE

c) What must a requirement model do? 1M

Class diagrams are the main building block in object-oriented modeling. They are used to show the different objects in a system, their attributes, their operations and the relationships among them. So instead of other diagrams we prefer class diagram in UML.

d) Define model consistency. 1M

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.

e) List the different types of Class Stereo types. 1M

- Entity
- Boundary
- Control

f) What do you meant by consistency checking? 1M

Here Consistency checks are an important task in the preparation of a complete set of models. This process highlights omissions and errors, and encourages the clarification of any ambiguity or incompleteness in the requirements.

1. Every event should appear as an incoming message for the appropriate object on
an interaction diagram.
2. Every action should correspond to the execution of an operation on the appropriate class, and perhaps also to the dispatch of a message to another object.

g) Write any two major elements of system design. 1M

- ➢ Defining the system architecture.
- ➢ Designing the components

h) What are the benefits of using Patterns? 1M

- ➢ Patterns provide a mechanism for the reuse of generic solutions for object-oriented and other approaches.
- ➢ Another benefit gained from patterns is that they offer a vocabulary for discussing the problem domain means whether it be analysis, design or some other aspect of information systems development at a higher level of abstraction than the class and object making it easier to consider micro-architectural issues.

i) State about deployment diagrams. 1M

- ➢ 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.

j) What is the main use of reuse strategy? 1M

In some organizations, reuse may just be about making use of reusable components from elsewhere, In others, reuse will be about the kind of organizational change.
The SELECT Perspective-The focus of this approach is to identify the services that belong together and classes that implement them.

UNIT-1

2. a) Discuss about fact finding techniques in Requirements capture. 5M

<u>Fact finding Techniques each –1M</u>

There are five main fact finding techniques that are used by analysts to investigate requirements.
1. Background Reading.
2. Interviewing.
3. Observation.
4. Document sampling.
5. Questionnaires.

Background Reading

If an analyst is assigned within the organization that is the subject of the fact gathering exercise, then he or she will already have a good under standing of the organization and its business objectives. If, however, he or she is going in as an outside consultant, then one of the first tasks is to try to gain an understanding of the organization. Background reading or research is part of that process.
The kind of documents that are suitable sources of information include the following: company reports, organization charts, policy manuals, job descriptions, Reports and documentation of existing systems.

Example

· Background reading is appropriate for projects where the analyst is not familiar with the organization being investigated.

· It is useful in the initial stages of investigation.

A d d a n e w m e m b e r o f s t a f f
a d d a n e w s t a f f g r a d e
c h a n g e t h e r a t e f o r a s t a f f
g r a d e
c h a n g e t h e g r a d e f o r a
m e m b e r o f s t a f f
s t a ff c o n t a c t
c a l c u l a t e s t a f f b o n u s e s

Interviewing

Interviewing is probably the most widely used fact finding technique; it is also the one that requires the most skill and sensitivity. Interviews can be used to gather information from management about their objectives for the organization and for the new information system, from staff about their existing jobs and their information needs, and from customers and members of the public as possible users of systems. While conducting an interview, the analyst can also use the opportunity to gather documents that the interviewee uses in his or her work.

Guidelines on Interviewing conducting an interview requires good planning, good interpersonal skills and an alert and responsive frame of mind. These guidelines are the important points while planning and conducting an interview.

At the start of the interview

Introduce yourself and the purpose of the interview. Arrive on time for interviews and stick to the planned timetable do not overrun. Ask the interviewee if he or she minds you taking notes or tape-recording the interview. Even if you tape record an interview, you are advised to take notes.

During the interview

Take responsibility for the agenda. You should control the direction of the interview. This should be done in a sensitive way. If the interviewee is getting away from the subject, bring them back to the point. If what they are telling you is important, then say that you will come back to it later and make a note to remind yourself to do so.

Example

Interviews are appropriate in most projects. They can provide information in depth about the existing system and about people's requirements for a new system.

Observation

Watching people carrying out their work in a natural setting can provide the analyst with a better understanding of the job than interviews, in which the interviewee will often concentrate on the normal aspects of the job and forget the exceptional situations and interruptions which can occur with the system and how to cope up with those problems. Observation also allows the analyst to see what information people use to carry out their job. This can tell you about the documents they refer to, whether they have to get up from their desks to get information, how well the existing system handles their needs. People who are not good at estimating quantitative data, such as how long they take to deal with certain tasks, and observation with a stopwatch can give the analyst lots of quantitative data, not just about typical times to perform a task but also about the statistical distribution of those times.

Example

· Observation is essential for gathering quantitative data about people's jobs.

· It can verify or disprove assertions made by interviewees, and is often useful in

· Situations where different interviewees have provided conflicting information about the way the system works.

· Observation may be the best way to follow items through some kind of process from start to finish.

Document sampling

First, the analyst will collect copies of blank and completed documents during the course of interviews and observation sessions. These will be used to determine the information that is used by people in their work, and the inputs to and outputs from processes which they carry out, either manually or using an existing computer system. From an existing system, the analyst may need to collect screen shots in order to understand the inputs and outputs of the existing system.

Second, the analyst may carry out a statistical analysis of documents in order to find out about patterns of data. For example, many documents such as order forms contain a header section and a number of lines of detail. The analyst may want to know the distribution of the number of lines in an order. This will help later in estimating volumes of data to be held in the system and in deciding how many lines should be displayed on screen at one time.

Example

The first type of document sampling is almost always appropriate. Paper-based documents give a good idea of what is happening in the current system. They also provide supporting evidence for the information gathered from interviews or observation. The statistical approach is appropriate in situations where large volumes of data are being processed, and particularly where error rates are high, and a reduction in errors is one of the criteria for usability.

Questionnaires

Questionnaires are a research instrument that can be applied to fact finding in system development projects. They consist of a series of written questions. The questionnaire designer usually limits the range of replies that respondents can make by giving them a choice of options.

YES/NO questions only give the respondent two options. If there are more options, the multiple choice type of question is often used when the answer is factual, whereas scaled questions are used if the answer involves an element of subjectivity. Some questions do not have a fixed number of responses, and must be left open-ended for the respondent to enter what they like.

Example

Multiple choice questions

Yes/ No questions

Feed back questions

b) State and explain about activity diagrams. **5M**

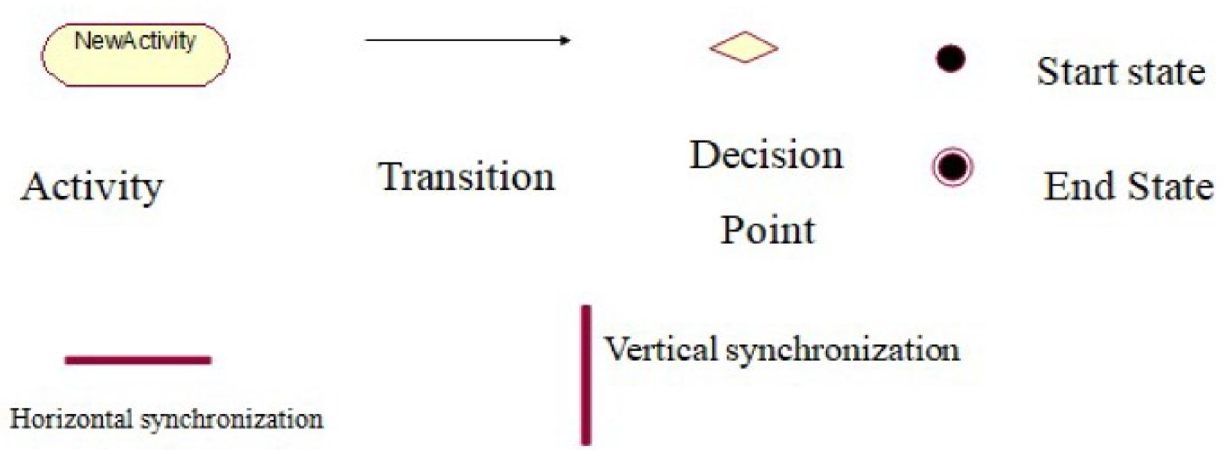
Purpose of activity---2M Notations ----3M

Purpose of Activity diagram:

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

Notation of Activity Diagrams



Activities :
An activity represents the performance of some behavior in the workflow.
Transitions :
Transitions are used to show the passing of the flow of control from activity to activity. They are typically triggered by the completion of the behavior in the originating activity.
Decision Points :
When modeling the workflow of a system it is often necessary to show where the flow of control branches based on a decision point.
Synchronization Bars
In a workflow there are typically some activities that may be done in parallel. A synchronization bar allows you to specify what activities may be done concurrently. Synchronization bars are also used to show joins in the workflow; that is, what activities must complete before processing may continue.
Swim lanes
Swim lanes may be used to partition an activity diagram. This typically is done to show what person or organization is responsible for the activities contained in the swim lane.
Initial and Final Activities
There are special symbols that are used to show the starting and final activities in a workflow. The starting activity is shown using a solid filled circle and the final activities are shown using a bull's eye.

3.a) Explain about requirement analysis. **5M**

Explaining Requirement Analysis ---5M

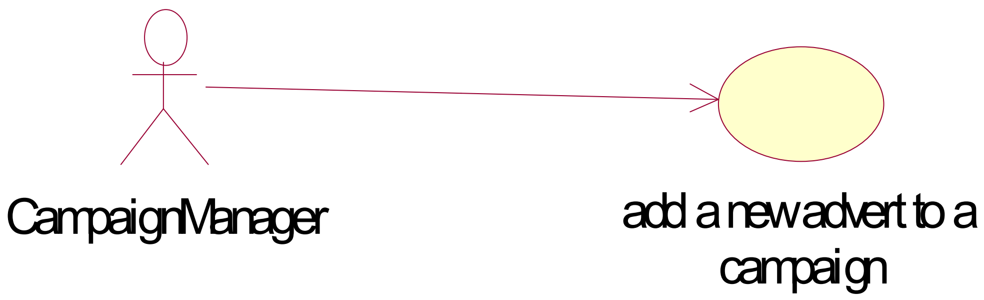
The most important factor for the success of an IS project is whether the software product satisfies its users' requirements. Models constructed from an analysis perspective focuses on determining these requirements. This means Requirement Model includes gathering and documenting facts and requests.

Use Case Realization:

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

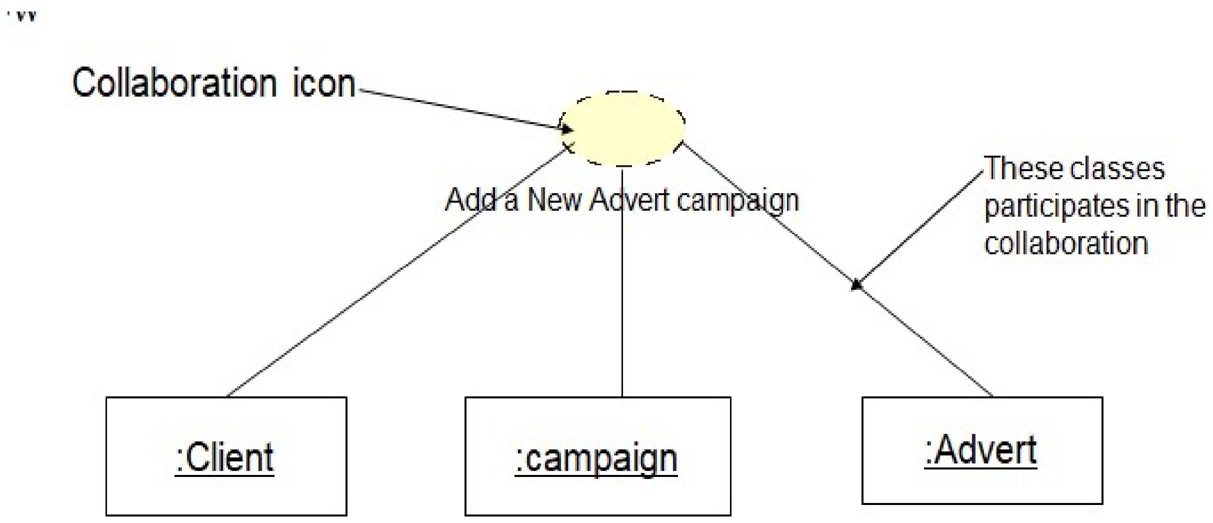
To move from an initial use case to the implementation of software that entirely satisfies the requirements identified by the use case involves at least one iteration through all of the development activities, from requirements modelling to implementation.

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 collaboration.

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

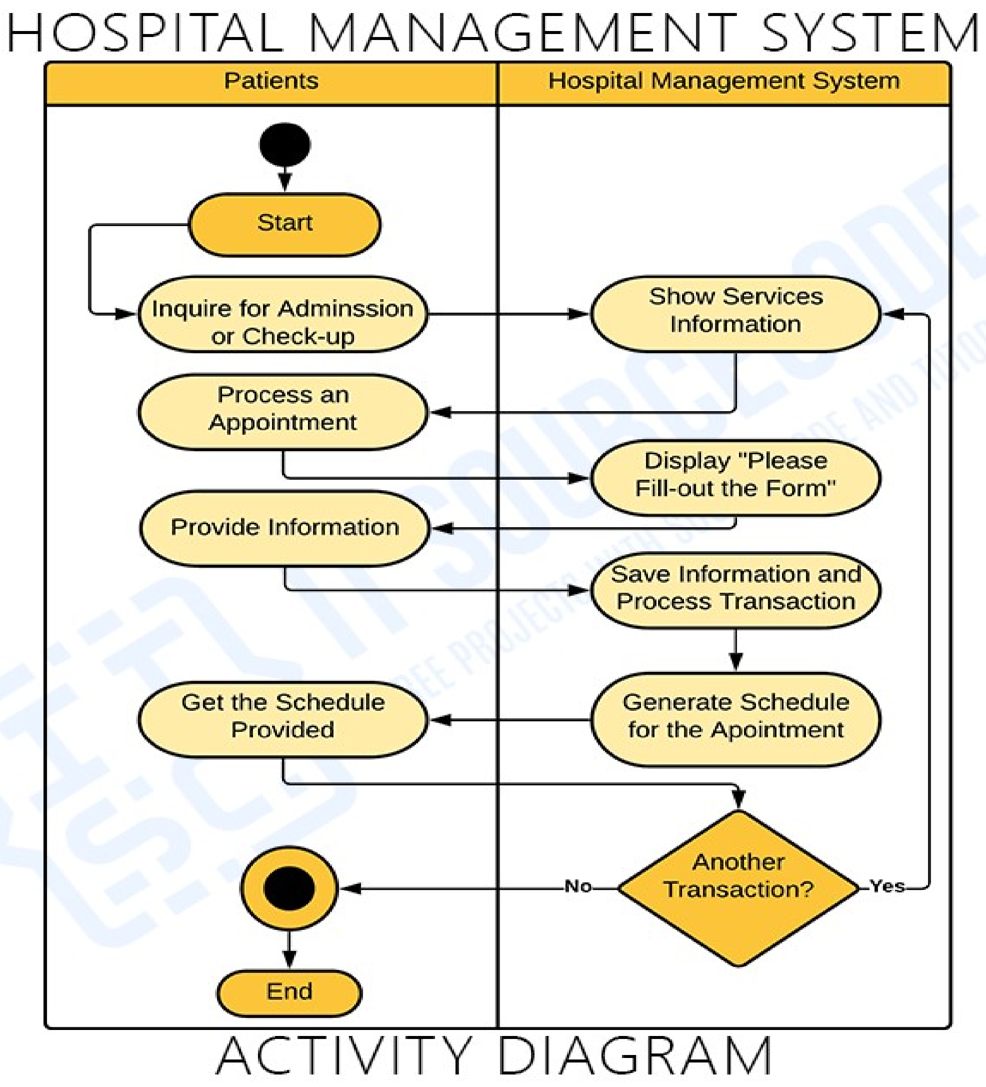


Collaboration icon
Add a New Advert campaign
These classes participates in the collaboration
:Client
:campaign
:Advert

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.

b) Draw an activity diagram, for hospital management system. 5M

Any related example can be considered----5M


HOSPITAL MANAGEMENT SYSTEM
Patients
Hospital Management System
Start
Inquire for Adminssion or Check-up
Show Services Information
Process an Appointment
Display "Please Fill-out the Form"
Provide Information
Save Information and Process Transaction
Get the Schedule Provided
Generate Schedule for the Apointment
Another Transaction?
No
Yes
End
ACTIVITY DIAGRAM

4.a) Describe about software development patterns 5M

Description steps about patterns----5M

Pattern : Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of a solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice. Coplien identifies the critical aspects of a pattern as follows.

- It solves a problem.
- It is a proven concept.
- The solution is not obvious.
- It describes a relationship.
- The pattern has a significant human component.

Architectural patterns – Responsibilities

1. Addresses some of the issues concerning the structural organization of software systems.
2. Architectural patterns also describes the structure and relationship of major components of a software system.
3. These patterns identifies subsystems, their responsibilities and their interrelationships.

Design patterns – Responsibilities
1. These patterns identify the interrelationships among a group of software components describing their responsibilities, collaborations and structural relationships.
2. Idioms describe how to implement particular aspects of a software system in a given programming language.
Analysis patterns : Responsibilities
1. These are defined as describing groups of concepts that represent common constructions in domain modelling. These patterns may be applicable in one domain or in many domains.
2. The use of analysis patterns is an advanced approach that is principally of use to experienced analysts,. They are closely related to design patterns also.
3. An analysis pattern is essentially a structure of classes and associations that is found to occur over and over again in many different modelling situations.

b) Demonstrate sequence diagram with ATM or LIBRARY example. 5M

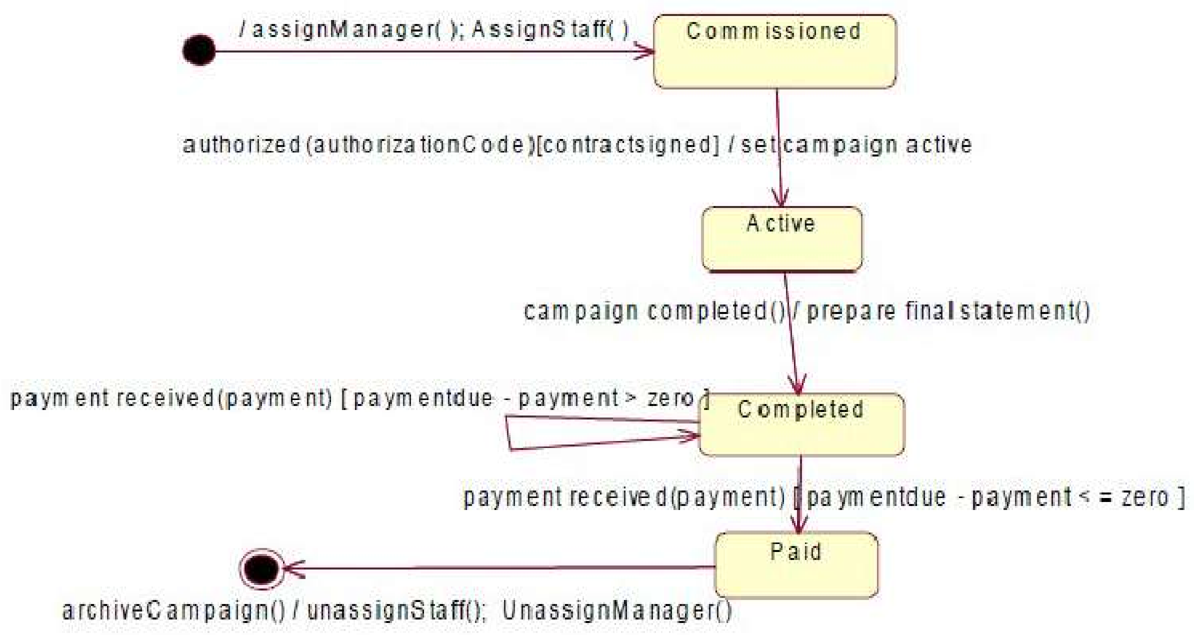
<u>Any related example can be considered----5M</u>

5.a) What do you know about state chart diagram? **5M**

Steps to prepare a state chart ----5M

Event
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.
State
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.
Transition
Movement from one state to another is called a transition, and is triggered by an event Preparation of State chart diagram
The preparation of a state chart from a set of interaction diagrams using this behavioural approach has the following sequence of steps.
1. Examine all interaction diagrams that involve each class that has heavy messaging.
2. Identify the incoming messages on each interaction diagram that may correspond to events. Also identify the possible resulting states.
3. Document these events and states on a state chart.
4. Elaborate the state chart as necessary to cater for additional interactions as these become evident, and add any exceptions.
5. Develop any nested state charts if already identified.
6. Review the state chart to ensure consistency with use cases. In particular, check that any constraints that are implied by the state chart are appropriate.
7. Iterate steps 4, 5 and 6 until the state chart captures the necessary level of detail.
8. Check the consistency of the state chart with the class diagram, with interaction diagrams and with any other state charts.

Example


/ assignManager( ); AssignStaff( )
Commissioned
authorized (authorizationCode)[contractsigned] / set campaign active
Active
campaign completed() / prepare final statement()
payment received(payment) [paymentdue - payment > zero]
Completed
payment received(payment) [paymentdue - payment < = zero]
Paid
archiveCampaign() / unassignStaff(); UnassignManager()

b) How to describe operation logic? **5M**

Explanation ---3 M examples---2M

The following are the reasons for the classification of operations based on various ways of describing their logic.

First, operations that have side-effects. Possible side-effects include the creation or destruction of object instances, setting or returning attribute values, forming or breaking links with other objects, carrying out calculations, sending messages or events to other objects or any combination of these. A complex operation may do several of these things, and, where the task is at all complex, an operation may also require the collaboration of several other objects. It is partly for this reason that we identify the pattern of object collaboration before specifying operations in detail.

Second, operations that do not have side-effects. These are pure queries; they request data but do not change anything within the system.

Like classes, operations may also have the property of being either {abstract} or {concrete} .Abstract operations have a form that consists at least of a signature, sometimes a full specification, but they will not be given an implementation (i.e. they will not have a method). 'Typically, abstract operations are located in the abstract superclasses of an inheritance hierarchy. They are always overridden by concrete methods in concrete subclasses.

A specification may be restricted to defining only external and visible effects of an operation, and for this we have two approaches for specifying operational logic.

1. Non- algorithmic approach

2.Algorithmic approach.

A specification may also define internal details, but this is effectively a design activity.

1 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.

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.

6.a) Demonstrate system design and detailed design. 5M

System design---2M Detail design---3M

System design -

During system design the designers make decisions that will affect the system as a whole. The most important aspect of this is the overall architecture of the system. Many modern systems use a client-server architecture in which the work of the system is divided between the clients and a server . This arises questions about how processes and objects will be distributed on different machines, and it is the role of the system designer or system architect to decide on this.

The design will have to be broken down into sub-systems and these sub-systems may be allocated to different processors. This introduces a requirement for communication between processors, and the systems designer will need to determine the mechanisms used to provide for this communication. Distributing systems over multiple processors also makes it possible for different sub-systems to be active simultaneously or concurrently.

Detailed Design –

This addresses the design of classes and the detailed working of system. These activities we need to consider under traditional and OO approaches.

Traditional detailed design -

Here, detailed design was seen as consisting of four main activities:

- Designing inputs - Designing inputs meant designing the layout of menus and data entry screens;

- Designing outputs - Designing outputs concerned with the layout of enquiry screens, reports and printed documents;
- Designing processes - Designing processes dealt with the choice of algorithms and ensuring that processes correctly reflected the decisions that the software needed to make;
- Designing files - Designing files dealt with the structure of files and records, the file organization and the access methods used to update and retrieve data from the files.

First, the work of Jackson provided a method for designers to design programs by using a technique to match the structure of the inputs and outputs with the structure of data to be read from or written to files.

Second , A criteria which is to be considered in breaking systems and programs down into modules to ensure that they are easy to develop and maintain. These criteria concern two issues: *cohesion* and *coupling.*

b) Explain in the importance about user interface design. 5M

<u>Importance about user interface design----5M</u>

Users of an information system need to interact with it in some way.

Whether they are users of tele-sales system entering orders made over the telephone by customers, or members of the public using a touch screen system to find tourist information, they will need to carry out the following secondary tasks:

- read and interpret information that instructs them how to use the system;
- issue commands to the system to indicate what they want to do;
- enter words and numbers into the system to provide it with data to work with;
- read and interpret the results that are produced by the system either on screen or as a printed report;
- respond to and correct errors ;

There are many different ways of designing and implementing the elements of the user interface that support the interaction with users by using formal and informal approaches.

The following are the factors to be considered while designing User Interface .

- the nature of the task that the user is carrying out,
- the type of user,
- the amount of training that the user will have undertaken,
- the frequency of use
- the hardware and software architecture of the system.

- These factors may be very different from system to systems which are listed in the following table. The following are the factors for the tele-sales system and a WAP tourist information system. Systems that are used by members of the public are very different from information systems used by staff.

7.a) How to document patterns? Explain. 5M

Pattern template explanation---5M

Pattern: Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of a solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.

Patterns can be documented using one of several alternative templates. The pattern template determines the style and structure of the pattern description, and these vary in the emphasis they place on different aspects of patterns. The differences between pattern templates may provide variations in the problem domain but there is no agreement as to the most appropriate template even within a particular problem domain. Generally a pattern description should include the following elements.

Name. A pattern should be given a meaningful name that reflects the knowledge embodied by the pattern. This may be a single word or a short phrase. These names become the vocabulary for discussing conceptual constructs in the domain of expertise.

Problem. This is a description of the problem that the pattern addresses means the intent of the pattern. It should identify and describe the objectives to be achieved, within a specified context

and constraining forces. For example, one problem might be concerned with producing a flexible design, another with the validation of data.

The problem can frequently be written as a question, for example 'How can a class be constructed that has only one instance and can be accessed globally within the application?' This question expresses the problem addressed by the Singleton pattern .

Context. The context of the pattern represents the circumstances or preconditions under which it

can occur. The context should provide sufficient detail to allow the applicability of the pattern

to be determined.

Forces. The forces embodied in a pattern are the constraints or issues that must be addressed by

the solution. These forces may interact with and conflict with each other, and possibly also with

the objectives described in the problem. They reflect the details of the pattern.

Solution. The solution is a description of the static and dynamic relationships among the components of the pattern. The structure, the participants and their collaborations are all described. A solution should resolve all the forces in the given context. A solution that does not resolve all the forces fails.

b) What are the approaches to user interface design. 5M

Three Approaches brief explanation----5M

Approaches to USERINTERFACE Design

There are many different ways of designing and implementing the elements of the user interface that support the interaction with users by using formal and informal approaches. The following

are the factors to be considered while designing User Interface.
 the nature of the task that the user is carrying out,
 the type of user,
 the amount of training that the user will have undertaken,
 the frequency of use
 the hardware and software architecture of the system.

1. Structured approaches
Structured approaches to user interface design have been developed in response to the growth in the use of structured approaches to systems analysis and design. Structured analysis and design methodologies have a number of characteristics. They are based on a model of the systems development life cycle, which is broken down into stages, each of which is further broken down into steps that are broken down into tasks. Specific analysis and design techniques are used, and the methodology specifies which techniques should be used in which step. Each step is described in terms of its inputs , the techniques applied and the deliverables that are produced as outputs.

Benefits of Structured approaches
 They make management of projects easier. The breakdown of the project into stages and steps makes planning and estimating easier, and thus assists management control of the project.
 They provide for standards in diagrams and documentation that improves understanding between the project staff in different roles means between analyst, designer and programmer.
 They improve the quality of delivered systems. Because the specification of the system is comprehensive, it is more likely to lead to a system that functions correctly.

2. Ethnographic approaches
The term ethnography is applied to a range of techniques used in sociology and anthropology and reflects a particular philosophy about how scientific enquiry should be carried out. In HCI this means that the professional charged with carrying out the user interface design spends time with the users immersed in their everyday working life. Only by spending time in this way can the real requirements of the users be understood and documented.
Ethnographic methods also concentrate on how different users interpret their experience of using systems subjectively, and it is this subjective interpretation that the HCI professional must understand rather than assuming that the system can be assessed objectively. Ethnographic approaches use a range of techniques to capture data: interviews, discussions, prototyping sessions and videos of users at work or using new systems. These data are analysed from different perspectives to gain insights into the behaviour of the users.
3. Scenario-based approaches.
Scenario-based design has been developed by John Carroll and others . It is less formal than the structured approaches but more clearly defined than most ethnographic approaches. Scenarios are step-by-step descriptions of a user's actions that can be used as a tool in requirements gathering, interface design and evaluation. Use cases are similar to scenarios.
Among these three approaches, scenario-based design fits best with use case modeling. Scenarios can be textual narrative describing a user's actions or they can be in the form of storyboards video mock-ups or even prototypes. Scenarios provide a means of communication that can be used by professionals and end-users to communicate about the design of the users' interaction with the system. They are simple enough that users can produce them without the need for the kind of training that

they would need to understand class diagrams, for example. Scenarios can be used with use cases. The use cases can provide a description of the typical interaction.

8.a) Define software testing. Explain about testing with an example. 5M

Software testing----3M Example---2M

SOFTWARE TESTING

Who carries out the testing?

One view of testing is that it is too important to be left to the programmers who have developed the software for the system. Here important fact is testing is carried out by someone whose assessment of the software will be objective and impartial. It is often difficult for programmers to see the faults in the program code that they have written. An alternative is provided by Extreme Programming (XP).

XP is an approach to rapid application development in which programmers are expected to write test harnesses for their programs before they write any code. Every piece of code can then be tested against its expected behaviour, and if a change is made can easily be retested.

he analysts who carried out the initial requirements analysis will be involved in testing the system as it is developed. The analysts will have an understanding of the business requirements for the system and will be able to measure the performance of the system against functional and non-functional requirements.

The systems analysts will use their knowledge of the system to draw up a test .This will specify what is to be tested, how it is to be tested, the criteria by which it is possible to decide whether a particular test has been passed or failed, and the order in which tests are to take place. Based on their knowledge of the requirements, the analysts will also draw up sets of test data values that are to be used.

What is tested?

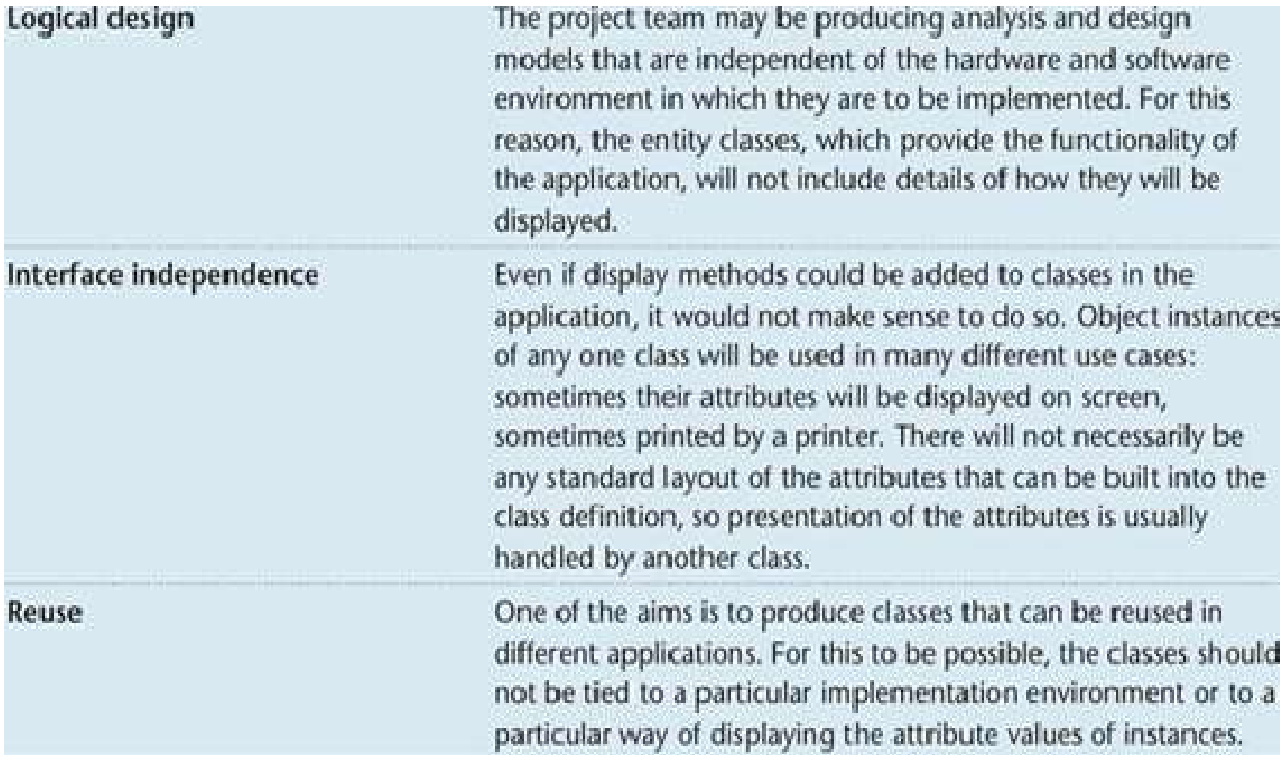
In testing any component of the system, find out whether its requirements have been met or not. One kind of testing needs to answer the following questions.

- ✓ Does it do what it's meant to do?
- ✓ Does it do it as fast as it's meant to do it?
- ✓ This type of testing for functional specifications of the system known as ***black box*** testing because the software is treated as a black box. Test put into it and it produces some output, but the testing does not investigate how processing is carried out. Black box testing tests the quality of performance of software. It is also necessary to check how well the software has been designed internally.
- ✓ The second type of testing known as *white box* testing in which we need answer for the following question.
- ✓ Is it not just a solution to the problem, but a *good* solution?

- ✓ because it tests the internal workings of the software whether the software works as specified. White box testing tests the quality construction of the software. In a project where reusable components are used, may not be possible to apply white box testing to these components, as they may provided as compiled object code.

b) Explain about the architecture of presentation layer **5M**

The three-tier architecture is a common way to separate out user interface classes from the business and application logic classes and from mechanisms for data storage. There are a number of reasons for doing this,



Logical design	The project team may be producing analysis and design models that are independent of the hardware and software environment in which they are to be implemented. For this reason, the entity classes, which provide the functionality of the application, will not include details of how they will be displayed.
Interface independence	Even if display methods could be added to classes in the application, it would not make sense to do so. Object instances of any one class will be used in many different use cases: sometimes their attributes will be displayed on screen, sometimes printed by a printer. There will not necessarily be any standard layout of the attributes that can be built into the class definition, so presentation of the attributes is usually handled by another class.
Reuse	One of the aims is to produce classes that can be reused in different applications. For this to be possible, the classes should not be tied to a particular implementation environment or to a particular way of displaying the attribute values of instances.

This is not to say that classes should contain no means of displaying their contents to the outside world. It is common practice to include in each class a print() (or a toString()) method that can be used to test the classes before the presentation layer has been developed. Taking a three-tier architectural approach does not necessarily mean that the different types of classes will end up running on different machines or even that they will be completely separate. It is useful to distinguish between the logical architecture of the system and the physical architecture. The physical architecture may combine layers of the logical architecture on a single physical platform or it may split logical layers across physical systems. If you are designing an AJAX application, some of the responsibilities for control will be located in the JavaScript classes, together with the boundary classes, while other control responsibilities may be located in classes on a server together with entity classes. In a distributed system, the entity classes may exist on different servers and the control classes would pull the data together from these different sources in order to deliver it to the boundary classes.
For the Agate system, we are going to keep the boundary, control and entity classes separate. The boundary classes will run on the users' machines, while the control classes will be located on servers, and the entity classes will initially be on local servers but may later be distributed in different offices.

1. prototyping the user interface
2. designing the user interface classes
3. modelling the interaction involved in the interface
4. modelling the control of the interface using state machines.

9.a) How do you implement component diagram explain with an example? 5M

Component diagram implementation---3M Example Diagram---2M

In a large project there will be many files that make up the system. These files will have dependencies on one another. The nature of these dependencies will depend on the language or languages used for the development and may exist at compile-time, at link-time or at run-time. There are also dependencies between source code files and the executable files or byte code files that are derived from them by compilation. Component diagrams are one of the two types of implementation diagram in UML.

Component diagrams show these dependencies between software components in the system. Stereotypes can be used to show dependencies that are specific to particular languages also.

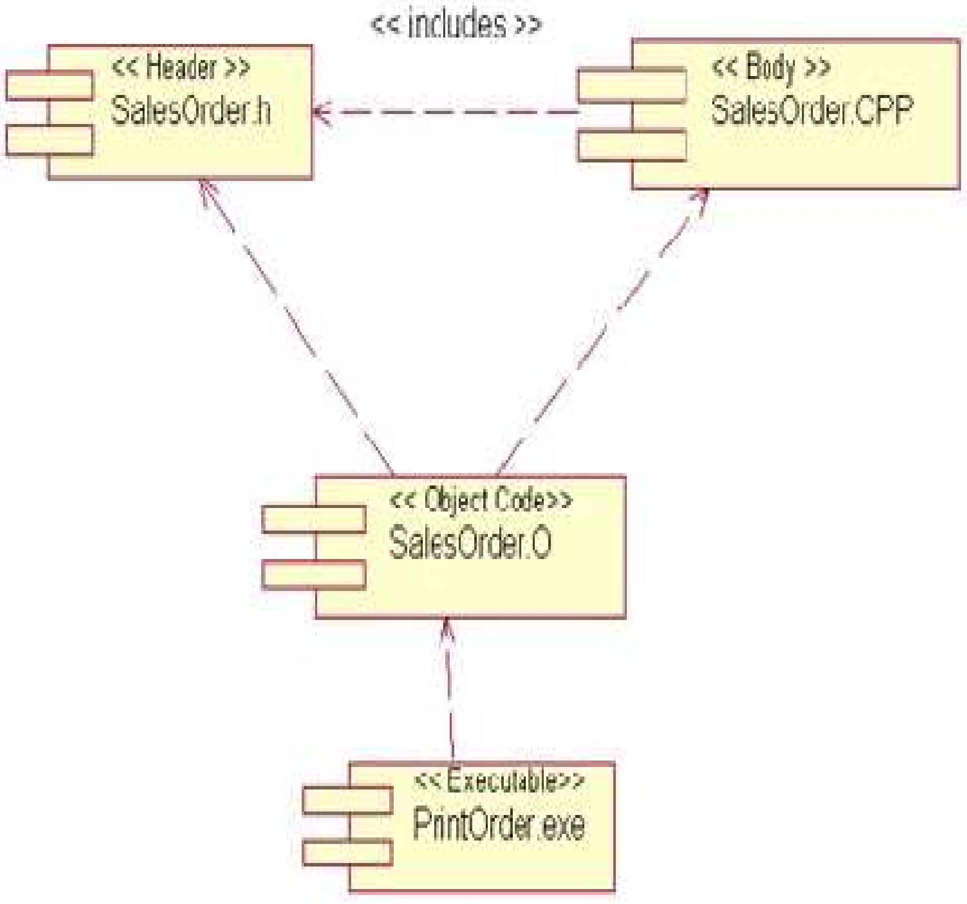
A component diagram shows the allocation of classes and objects to components in the physical design of a system. A component diagram may represent all or part of the component architecture of a system along with dependency relationships.

The dependency relationship indicates that one entity in a component diagram uses the services or facilities of another.

 Dependencies in the component diagram represent compilation dependencies.

 The dependency relationship may also be used to show calling dependencies among Components, using dependency arrows from components to interfaces on other components.

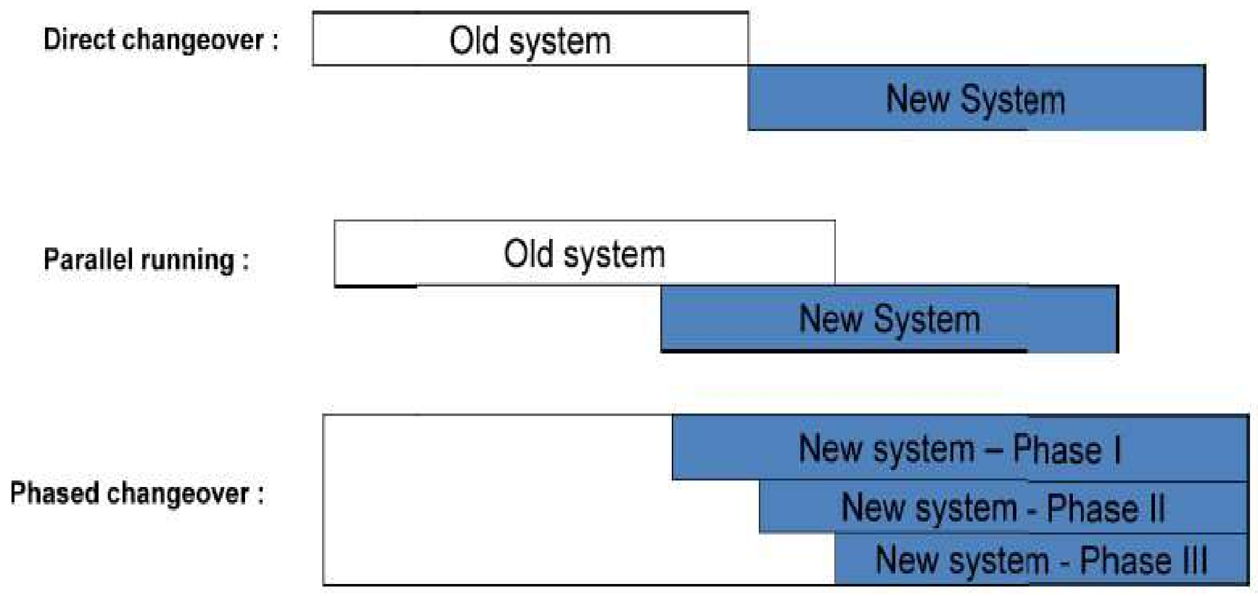
The following figure shows a component diagram that represents the dependency of a C++ source code file on the associated header file, the dependency of the object file on both and the dependency of an executable on the object file. Stereotypes can be used to show the types of different components.



b) Discuss in detail about implementation stratagies. **5M**

<u>Four Strategies each ---1M Diagram---1M</u>

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.



Direct changeover means that on an agreed date users stop using the old system and start using the new system. Direct changeover is usually timed to happen over a week some time for data conversion and implementation of the new system. Direct changeover is suitable for small-scale systems and other systems where there is a low risk of failure the implementation of established package software.
Parallel running allows the existing system to continue to run alongside the new system.
Parallel running should be used in situations where there is a high level of risk associated with the project and the system is central to the business operations of the organization.
The advantages and disadvantages of this approach are:
+ There is a fallback if there are problems with the new system;
+ The outputs of the old and new systems can be compare there is a high cost as the client must pay for two systems during the overlap period, and this includes the staffing necessary to maintain information in the old system as well as the new.
-There is a cost associated with comparing the outputs of the two systems;
- Users may not be committed to the new system as it is easier to stick with the familiar system.
In a phased changeover, the system is introduced in stages. The nature of the stages depends on the sub-systems within the software, but introduction into one department at a time may be appropriate. Phased changeover is suitable for large systems in which the sub-systems are not heavily dependent on one another.

The advantages and disadvantages are:
+ Attention can be paid to each individual sub-system as it is introduced;
+ if the right sub-systems can be chosen for the first stages then a fast return on investment can be obtained from those sub-systems;
+ Thorough testing of each stage can be carried out as it is introduced;
- Disaffection and rumour can spread through the organization ahead of the implementation if there are problems with the early phases;
-There can be a long wait before the business benefits of later stages are achieved.

Scheme prepared by   Signature of the HOD, IT Dept.

Paper Evaluators:

S.No	Name Of the College	Name of the Faculty	Signature