ramme: BESE
rse: Object Oriented Software Development

ndidates are required to give their answer in their words as for as practicable.
e figures in the margin indicate full marks.
**ttempt all the questions**

a) Describe the Object-oriented Analysis and Object-oriented Design process with example.

b) Differentiate process and product engineering. Justify process engineering ultimately is also a product engineering.

a) Change management is very important in Iterative and Incremental Development. Why and how is it done?

b) Define Use case realization. What do you mean by use case driven software development process?

a) Define software quality. Describe the user, developer, customer and manager perspective of software quality.

b) Define design pattern. Differentiate programming paradigm and design pattern.

4. a) Write about structure and documentation of pattern

b) Explain Creational Patterns, describing one of its type.

a) What are the difficulties and risks while using design pattern? Discuss in brief.

b) Describe about Client Server and Distributed Architecture. Are they similar? Justify. What is the role of distributed architecture in today's world of automation? Explain.

6. a) Compare and contrast design and architectural pattern.

b) Describe MVC architectural pattern with the design principles it helps to adhere.

7. Write short notes on (any two).
a) Message Oriented Architecture
b) Architecture centric process
c) Player Role Pattern
d) Extreme Programming (XP)

e : Sudha Shrestha
No. : 2014SE41

Subject : OOSD
Date : 14-07-2074

73

Q no. 1a)
Ans→

Object oriented analysis (OOA) is the systematic evaluation of the requirements of a system or project. object oriented analysis is the first step in the object ~~orient~~ oriented software development. It is the process of developing a model that has all the requirements of a system. Object oriented Design (OOD) is the process of creating the design model of the system. OOD is not the implementation rather it is the description of how a system is supposed to be. OOD is also the process of converting the analysis model to design model. Object oriented analysis gives the information about "what" is to be done and object oriented design gives the information about how is it is to be done. Object oriented analysis and design are the processes that is to be done before object oriented

programming.
Let us take an example of the dice game:

In object oriented analysis we create the use case, domain ~~diagram~~ model etc.

Use case: A player rolls two dice. If the total is seven then he wins else loses.

Use case diagram: shows user & system interaction.

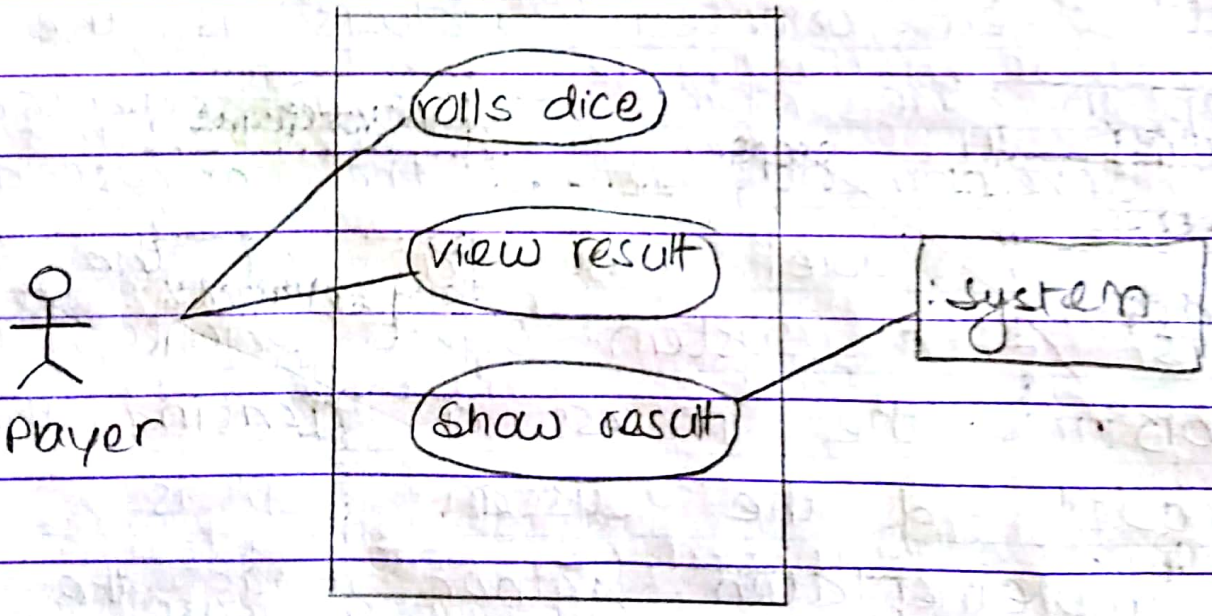
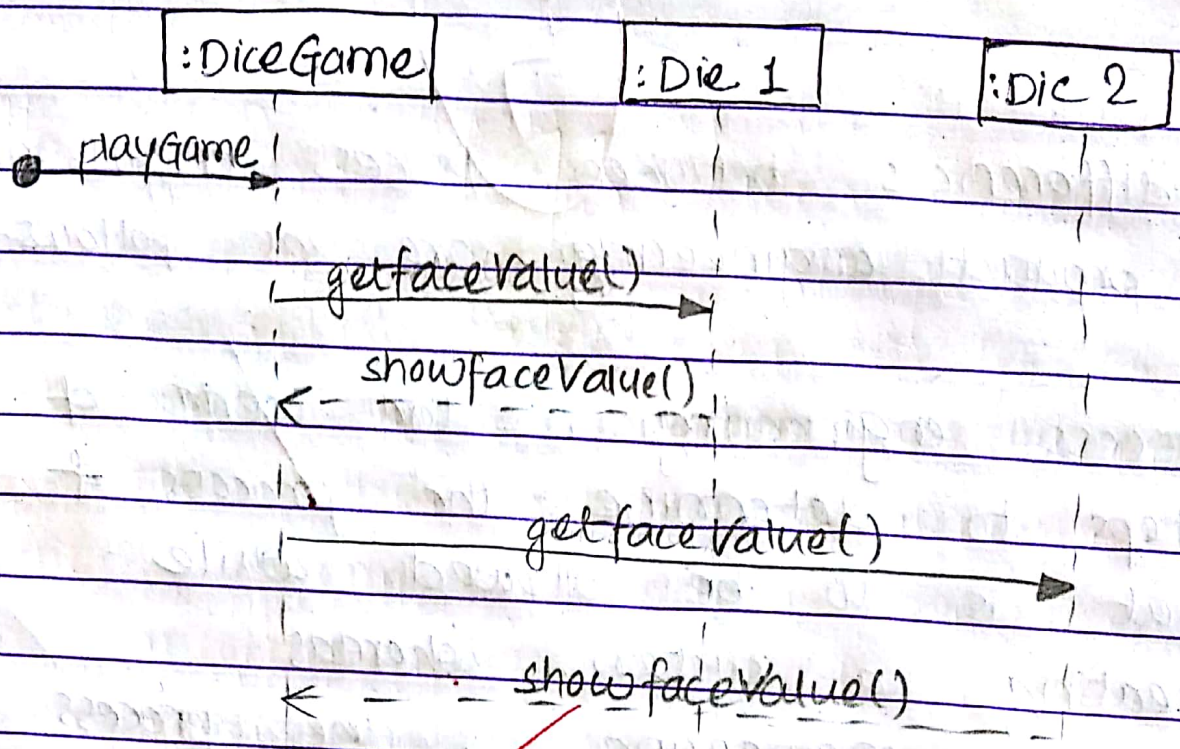
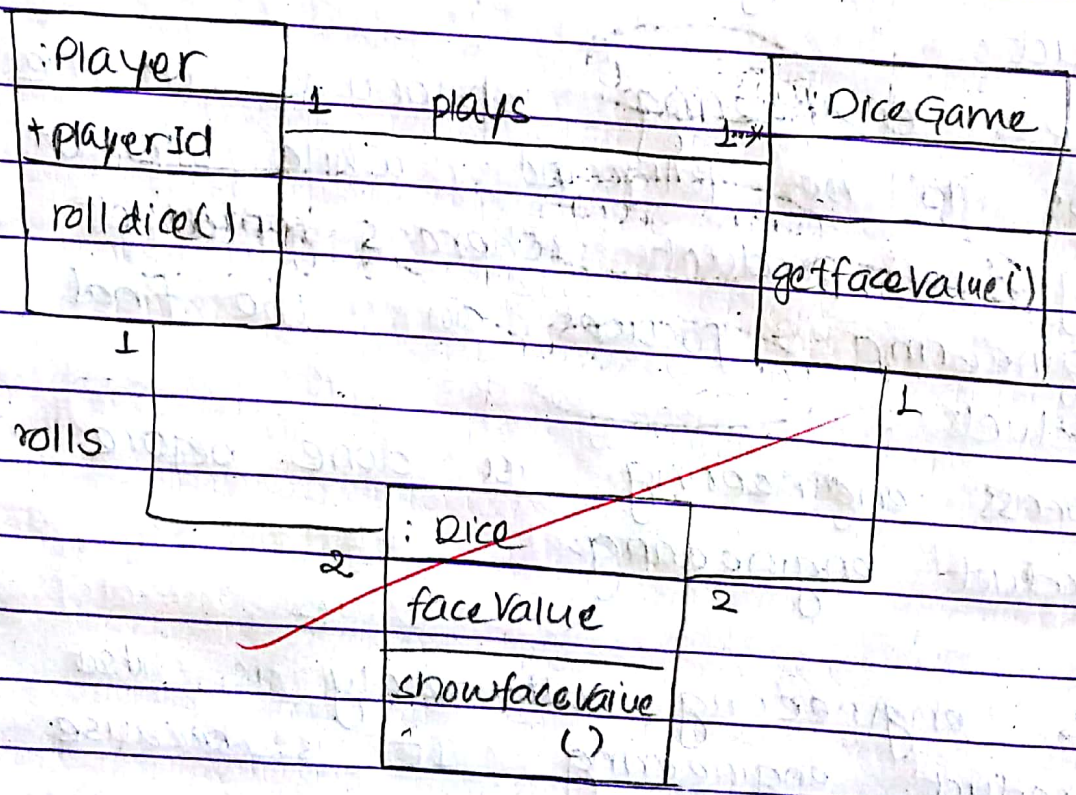



fig: use case diagram for user-system interaction.

Interaction diagram:




Design Class diagram:




In this way, object oriented analysis and design is carried out.

Q no 1b)

Ans →

5

The differences between process engineering and product engineering are as follows.

1. Process engineering is the series of steps to determine the process that is to be followed while creating a system, whereas Product engineering is the process of following the design principles and methodologies to create a product.
2. Process engineering is focuses on the steps to be followed while developing a product whereas product engineering focuses on the final product.
3. Process engineering is done before product engineering.

Process engineering ultimately is also a product engineering. It is because the steps that are taken in process engineering is also directed towards creating the product. The process that is followed is the evaluation of requirements, specifications and finding out the

artifacts of a system which is eventually required while doing product engineering. Process engineering is similar to the analysis phase and product engineering is similar to the design phase. The model that is created at the end of process engineering is required used by in the product engineering to create a final model that meets all the requirements of that particular system.

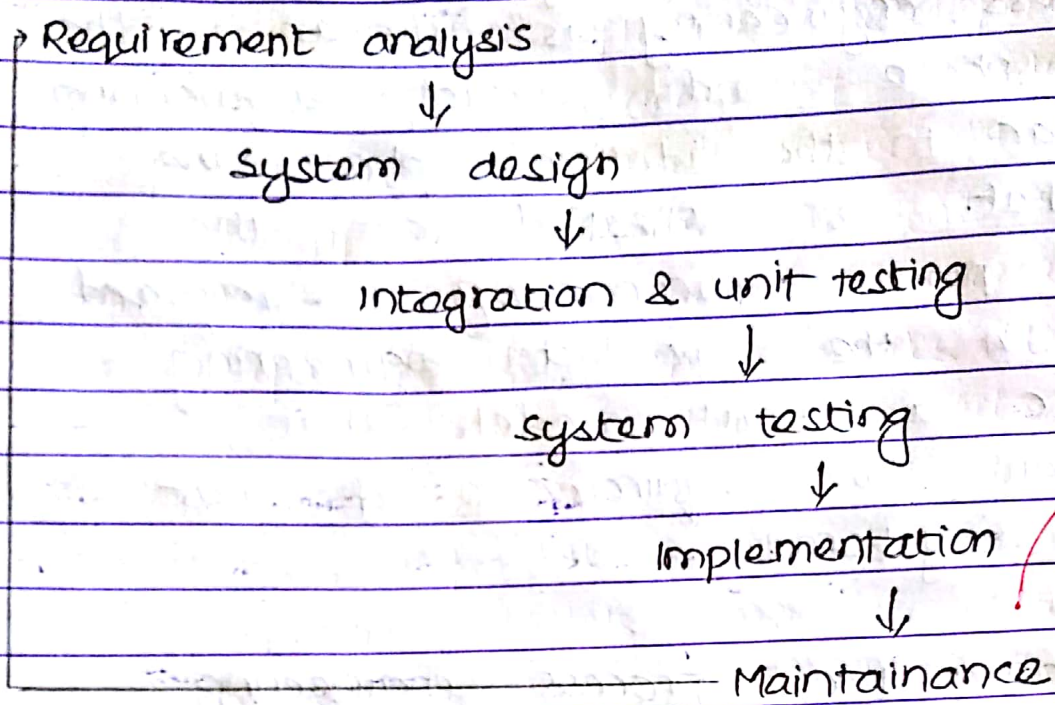
Hence, as the process engineering is also focused towards the development of the product, process engineering ultimately is also a product engineering.

Q.no. 2a)

Ans →

Iterative and Incremental development is the ~~mof~~ model of software development in which the ~~pf~~ system is made by a series of steps which undergoes several iterations before the final ~~deploy~~ product is formed. In iterative and incremental development, the system model goes through

series of steps like :




As shown in the above diagram, when all the steps are completed, the system completes one iteration and then again next iteration takes place. change management is a very important aspect in iterative and incremental development. Many changes might take place ~~while~~ while developing a software. For example, the client may want to add some features or delete some features, or the system might not be convenient or compatible. Many other changes might occur as well. so, change management is a must. It is handled in after each iteration in iterative

and incremental development. After each iteration is completed feedbacks are taken from the customers and the users and if changes are to be made, then it is done in the next iteration. When certain changes are to be made then the risk factors are first analyzed and tested if the risks can be managed or not then the changes are applied.

In this way, change management is done in iterative and incremental development. As the system forms incrementally in every iterations, changes are made in each iteration through series of steps and managed.

Q.no. 2b)

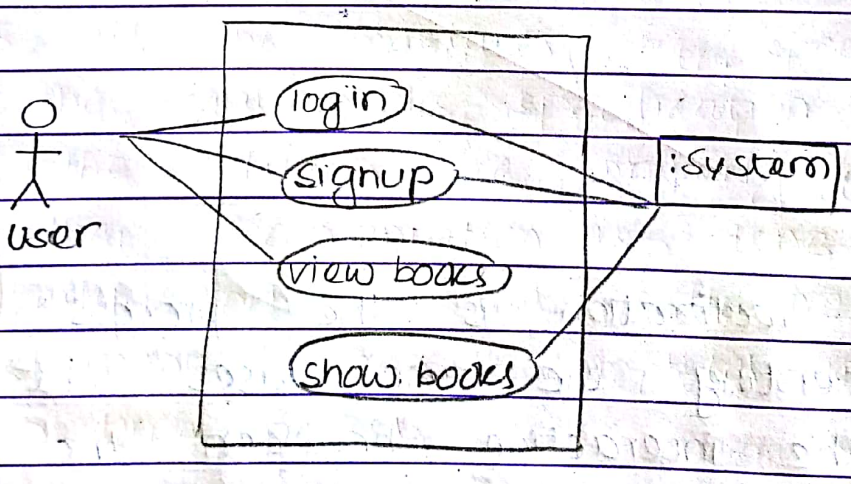
Ans →

Use case realization is the process of documenting the use cases that shows the interaction between the user and the system so, that other artifacts are also created taking reference from the use-cases. Use cases shows certain the way that the user and the

6

system interact with each other.

Use case driven software development process is the process of developing a software on the basis of the description provided in the use case. The actors, their functionalities, the requirements, the works that are to be done by the actors and system are given in the use case. So, by taking it as a reference, other design models are formulated which shows how the system will give the proper functionalities to the users such that it meets all the requirements.
For example:




eg: Fig: example of a use case diagram.

By looking at the use case diagram,

a developer can find out what the user does and what the system is supposed to do. In addition to this, there are fully dressed use cases which provide more information to the developer that shows the scenarios of how the series of actions take place. In this way, a use case driven software development process takes place.

Qno. 3a)

Ans→

Software quality is the measure of the extent to which the software meets the users requirements as well as provides feasibility, scalability and is easy to use. Software quality can mean different to a different group of people: for example: the software quality that a user expects in a software may be different to the software quality that a developer thinks.

The software quality from the perspective of a user is its easyness to use, operate, the design of the system, the user interface etc.

From a developers perspective, the software quality is good if it meets all the software requirements provided by the customer, if the changes can be handled easily and if the software can operate in different new situations and errors are handled properly.

From a customers perspective, a software quality is good if the software meets all their requirements, if it is easy to handle and if the end users are satisfied by the software.

From a managers point of view, a software is of good quality if it has all the features as specified in the requirements specification, if all the artifacts are properly made, if the documentation is done properly and if the software development process completes in the given time period.

These are the different perspectives of user, developer, customer and manager, in terms of software quality.

q.no.3b)

Ans→

6

Design pattern is the solutions of the repeatible situations while developing a software. There might be many situations or problems that occur repeatedly while the software development process is carried out. Design pattern is the description of how the problems are to be handled. Design pattern is not the model that can be implemented directly. There has to be certain changes and adjustments in the pattern to create the implementation model as the design pattern only provides the information about how the situation or problem is to be handled. A software project can be completed even without implementing a design pattern but by the use of design pattern, it will complete in less time with less errors and problems.

The differences between programming paradigm and design pattern are as follows.

→ Programming paradigm is the way in which the programming is to be

done whereas design pattern is the solution to the repeatible problems while creating the design ~~pattern~~ model of a software.

- Programming paradigm is used to create the implementation or the deployment model of the software whereas the design pattern is used to create the design model is formed.

- Programming paradigm is applied after the design model is completed whereas design pattern is applied while creating the design model. It means, programming paradigm may depend upon the design pattern used.

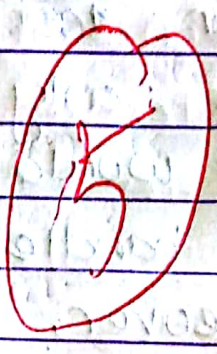
- The design pattern is the solution to the problems i.e. it provides solutions that is already tested in several test cases, which means there is always one particular solution to certain problem but, while applying the programming paradigm it completely depends on the developer as programming can be done in one way or another.
- Examples of programming paradigm: object oriented, procedural & of design pattern: creator, divide & conquer, low coupling etc.

that every actor is involved in their own role and tasks will be done properly.

Q.no: 4a)

Ans:-

Structure of a pattern:



| Name of pattern: |
| --- |
| Problem: |
| Solutions: |

The structure of a pattern is as shown in the above diagram. At first the name of the pattern is written. For example: Low coupling, Creator, High cohesion etc.
Then, the problem is written and then the solutions i.e. what steps are to be taken in order to solve the particular problem.

Q.no. 4b)

Ans →

Creational patterns are those design patterns which can be applied while creating a model. These patterns are those patterns which helps in the way classes and objects are created. For example: A class is created by inheriting from another class then design patterns can be used such that there are less errors.

An example:

| Name: ~~Creator~~ Low Coupling |
| --- |
| Problem: The relationship between classes may create errors if changes are made to one. |
| Solution: Create less dependencies as possible between the classes. |

These are the differences between programming paradigm and design pattern.

Q.no.5a)

Ans→

Design patterns are the solutions to the repeatible problems that might occur while creating the design model of a software. Design patterns are created after a series of test on test cases. Design pattern makes it easier to develop the design model of a system.
But there might occur some difficulties and risks while using design patterns. Some of the difficulties and risks are:

- If we apply the design pattern that is not appropriate for solving our problem, then it will in turn create other additional problems to the software design process. Which means that we must be careful while choosing the correct and suitable design pattern

for our project.

- While using a design pattern, all the rules must be followed to prevent creating additional problems.

- The rules of the design pattern may be diffult to be applied to the classes and objects of our project.

- The design pattern applied to some classes may ~~be~~ make it incompatible to some functionalities.

These are some of the difficulties and risks while using a design pattern.

Q.no. 55)

Ans→

Client - Server architecture is that architecture that has two main parts that communicate between each other i.e. the client and the server. The client requests some services to the server and usos them whereas the server doesnot request any service from

the server is not centralized.

There is a great role of distributed architecture in today's world of automation. The companies may be distributed is sereral geographical regions. It may be inside a town, country or even in different countries. So, a centralized server is not posible for providing serices to different areas. Hence, distributed architecture is required. Moreover, the data is more secure and communication is more secure in case of distributed architecture because even if one server encounters problem, other servers will still function properly, so, there is a great role of distributed architecture in today's world of automation.

the client. The client only recieves services ~~ff~~ and information from server but cannot send whereas the server can only provide services to the client and cannot receive any service from client.

Distributed architecture is that architecture in which the servers are distributed in ~~sere~~ different regions and are connected to each other. The other clients and components are connected to the different servers.

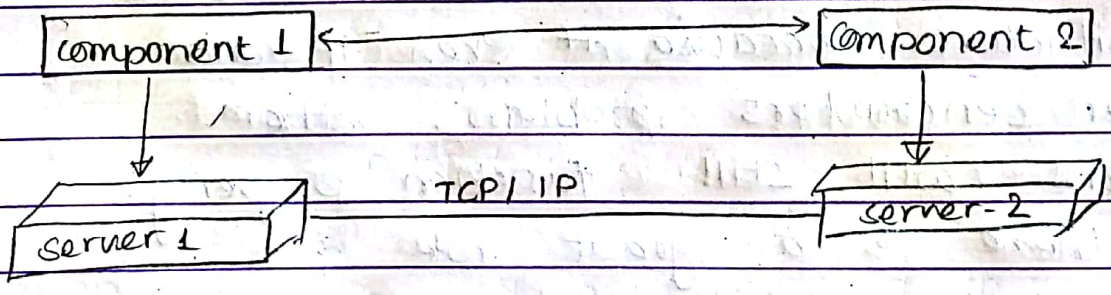



Fig: Distributed system architecture

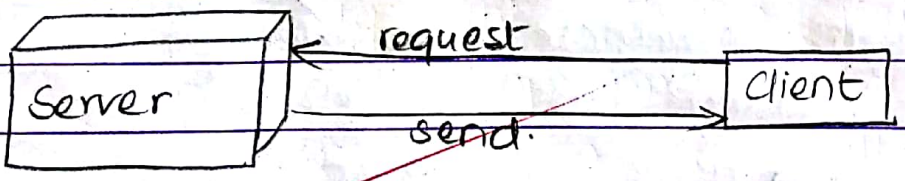



Fig: Client server Architecture.

In case of client - server architecture, there is a centralized server for clients whereas in distributed architecture,

Q.no. 6a)

Ans →

Design patterns are those patterns that provide the solutions to repeatible problems whereas Architectural patterns are those patterns that determines how the system is made and how the interactions are to be done in the system.

Architecture must be good for the design model to be good. The design patterns are applied to create designs of a system whereas architecture pattern is used to create the frame-work for the design.

Examples of architectural patterns are: Model view controller Architecture, message oriented architecture etc. whereas examples of design pattern are: low coupling, high cohesion, abstraction etc.

Architectural patterns determine how the actors in the system interact but design pattern donot.

Q.no.6b)

Ans→

MVC architectural pattern stands for (Model View controller) architecture pattern. The model - view controller architectural pattern divides the system into three parts having their own functionalities. The MVC pattern provides great flexibility and abstraction in the software development process. Many of the programming languages like Java, Ruby, Python uses the MVC pattern.

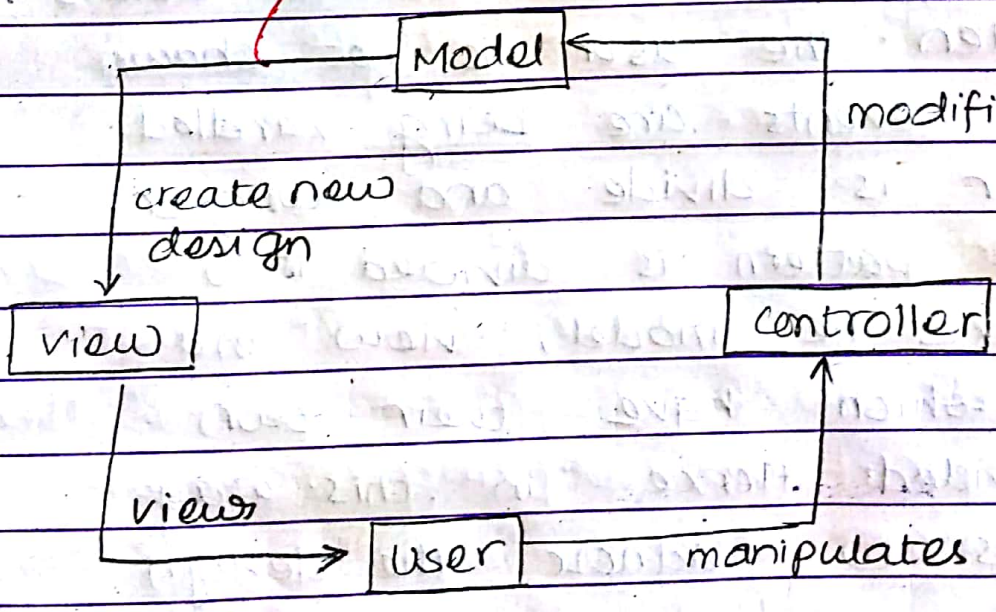



Fig:- Model - view - Controller Pattern

In MVC, at first the user modifies the controller to perform different tasks and handles them. Then, the

controllor manipulates the model of the system. Then, the view changes according to the model of the system and finally, the user views the final view of the system. The controller controls the performance of the system, the model determines how the system is to be built and the view is the design that is shown to the user.

MVC pattern also follows some design principles. The first one is abstraction. The view is only shown to the user and the internal details are hidden. The user is not shown how the events are being handled. The other is divide and conquer. The MVC pattern is divided into three parts i.e. model, view and controller which have their own tasks divided. Hence in this way MVC helps to adhere the design principles.

Q.no: 7 b)

Ans→

Architecture centric process:

Architecture centric processes are those processes that follows certain architecture pattern. The architecture determines how the system is made, how the system interaction is handled, then the architecture centric processes handle the architecture in same way as mentioned in the architecture.
For example: In message oriented architecture, the interactions are handled through message passing and there is a broker in between the sender and receiver for the communication.

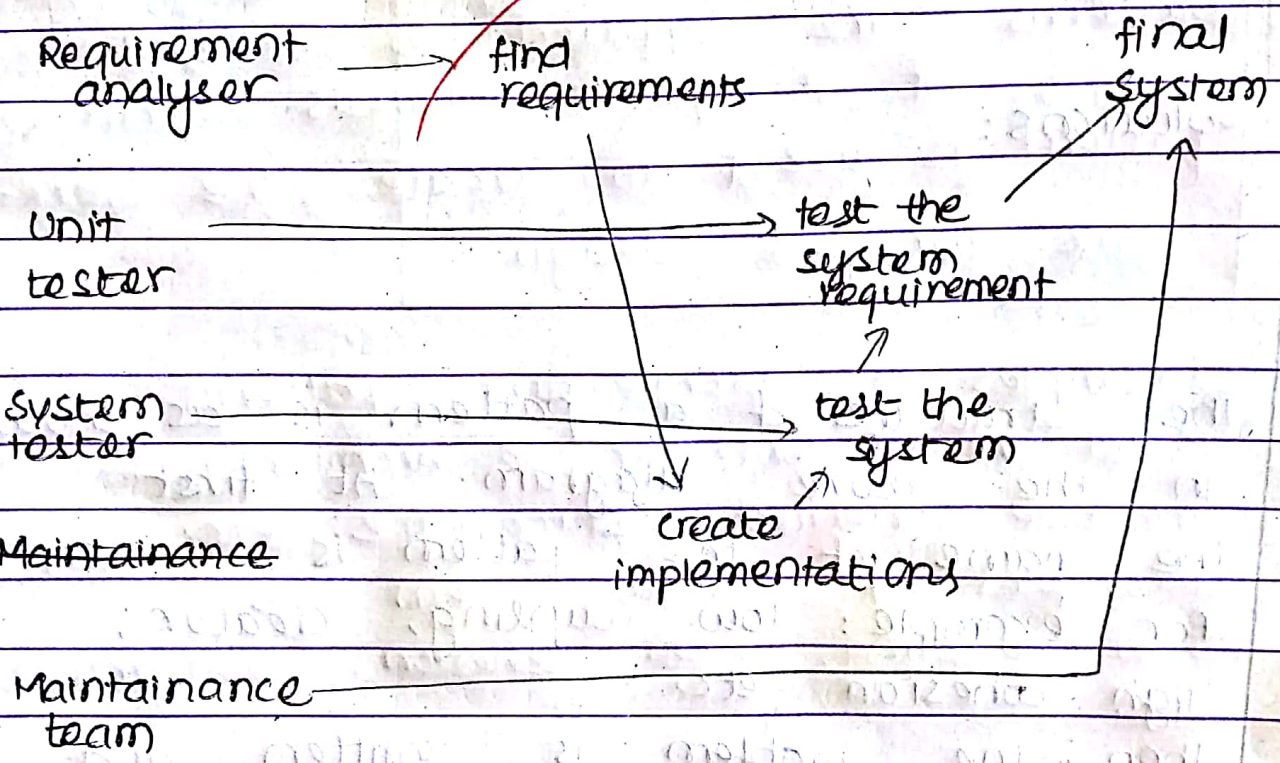
In this way, architecture centric processes takes place.

Q.no.7c)

Player Role Pattern:

Player Role Pattern is the design pattern which states that one actor should be involved in one role only as far as possible. It would be difficult for one class to handle different roles by itself moreover, it will difficult create errors in the system.

For example:




As shown in the figure above, the requirement of the system is done by one actor, unit testing is done by another, system testing, maintainance is done by separate people so