CMR
INSTITUTE OF
TECHNOLOGY

USN | | | | | | | | | | |

Improvement Test

Sub:	Object Oriented Modeling and Design							Code:	10CS71
Date:	15 / 11 / 2016	Duration:	90 mins	Max Marks:	50	Sem:	VII	Branch:	CSE/ISE

Answer Any FIVE FULL Questions

		Marks	OBE	
			CO	RBT
1	What is system conception? List and explain questions that must be answered by a good system concept.	[10]	CO 1	L2
2	Explain the Object oriented Software Development process.	[10]	CO1	L2
3	Compare forward engineering and reverse engineering.	[10]	CO5	L4
4	Differentiate between waterfall model and iterative model.	[10]	CO1	L4
5	Explain the steps involved in implementation modeling.	[10]	CO 1	L2
6	Explain the different categories of patterns. Discuss the properties of Architectural patterns.	[10]	CO6	L2
7	Explain Forward –Receiver design pattern.	[10]	CO 6	L2

Course Outcomes		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1:	Explain the steps involved in the OOMD - Requirements, Analysis, Design.(SDLC)	0	0	1	1	0	0	0	0	0	1	0	0
CO2:	Model classes using UML notations covering - Events, Generalization, Association, Links,	2	1	2	2	3	0	0	0	1	0	0	0
CO3:	Model requirements using Use cases and Usecase scenarios.	0	1	0	0	3	0	0	0	0	1	0	0
CO4:	Model Object interaction using state diagram, sequence diagram, Activity Diagram	2	1	2	2	3	0	0	0	0	1	0	0
CO5:	Explain the object oriented methodology to handle legacy systems	2	0	0	0	0	0	0	0	0	0	0	0
CO6:	Explain the following Design Patterns with an example application - Forward-Receiver, Client-Dispatcher-Server, Publisher-Subscriber, Command Process, View handler.	0	0	3	0	0	1	0	0	0	0	0	0

Cognitive level	KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PO1 - *Engineering knowledge*; PO2 - *Problem analysis*; PO3 - *Design/development of solutions*; PO4 - *Conduct investigations of complex problems*; PO5 - *Modern tool usage*; PO6 - *The Engineer and society*; PO7- *Environment and sustainability*; PO8 – *Ethics*; PO9 - *Individual and team work*; PO10 - *Communication*; PO11 - *Project management and finance*; PO12 - *Life-long learning*

CMR INSTITUTEOF TECHNOLOGY

										USN

Improvement Test – Nov 2016 - SCHEME OF EVALUATION

Sub:	Object Oriented Modeling and Design				**Code:**	10CS71
Date:	15/11/2016	Duration:	90 mins	Max Marks: 50	**Sem:** VII	**Branch:** ISE/CSE

Note: : Answer any 5 questions Total marks: 50

1. **What is system conception? List and explain questions that must be answered by a good system concept (10M)**
   (Definition of System Conception) 1M
   (Mention any 4 points)
   Some ways to find new system concepts
   - New functionality
   - Streamlining
   - Simplification automate manual process
   - Integration
   - Analogies
   - Globalization

   (Explain Each it point carriers 1.5 mark) 1.5*6=9M

1. *Who is the application for?*
   a. Stakeholders of the system
2. *What problems will it solve?*
   a. Features
3. *Where will it be used?*
   a. Compliment the existing base, locally, distributed, customer base
4. *When is it needed?*
   a. Feasible time, required time
5. *Why is it needed?*
   a. Business case
6. *How will it work?*
   Brainstorm the feasibility of the problem

2. **Explain the Object oriented Software Development process.**
   (Definition of Object oriented Software Development process) 2M
   (Each point carriers 1 mark) 1*8=8M

1. System Conception
   a. Conceive an application and formulate tentative requirements
2. Analysis
   a. Deeply understand the requirements by constructing models
   b. To specify *what must be done.*

   Domain analysis focuses on real-world things whose semantics the application captures.
   Application analysis addresses the computer aspects of the application that are visible to users
3. *System design*
   a. Devise the architecture, Devise a high-level strategy — the architecture — for solving the application problem.
   b. The choice of architecture is based on the requirements as well as past experience.
4. *Class design*

toward computer concepts.

b. To choose algorithms to implement major system functions.

5. *Implementation*
   a. Translate the design into programming
6. *Testing*
   a. Ensure that the application is suitable for actual use and actually satisfies requirements
7. *Training*
   a. Help users master the new application
8. *Maintenance*
   a. Preserve the long term viability of the application

3. Compare forward engineering and reverse engineering **10M**

(Any 5 difference with explanation each carriers 2 marks) **2*5=10M**

Forward engineering

1. Given Requirements, develop an application
2. More certain .The developer has requirements and must deliver and implements them
3. Prescriptive .Developers told how to work.
4. More mature. Skilled staff readily available.
5. Time consuming
6. The model must be correct and complete or application fails

Reverse engineering

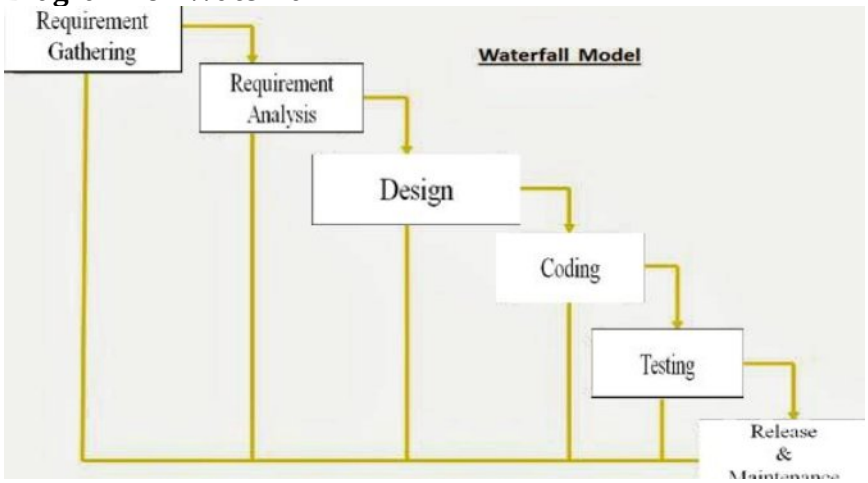
1. Given an application, Deduce tentative requirement.
2. Less Certain .An implementation can yield different requirements ,depending on the reverse engineer's interpretation.
3. Adaptive .The reverse engineer must find out what the developer actually did.
4. Less mature. Skilled staff are sparse
5. Can be performed 10 t0 100 times faster than forward engineering
6. The model can be imperfect. Salvaging partial information is still useful

4. Differentiate between waterfall model and iterative model. **10M**

Waterfall model **5M**

- Requirements are very well known.
- Product definition is stable.
- Technology is understood.
- New version of an existing product.

Diagram for Water fall



ADVANTAGES

- A waterfall model is easy to implementation.
- It helps to find errors earlier
- Easy to understand, easy to use.
- Works well when quality is more important than cost or schedule
- Documentation is produced at every stage of a waterfall model allowing people to understand what has been done.

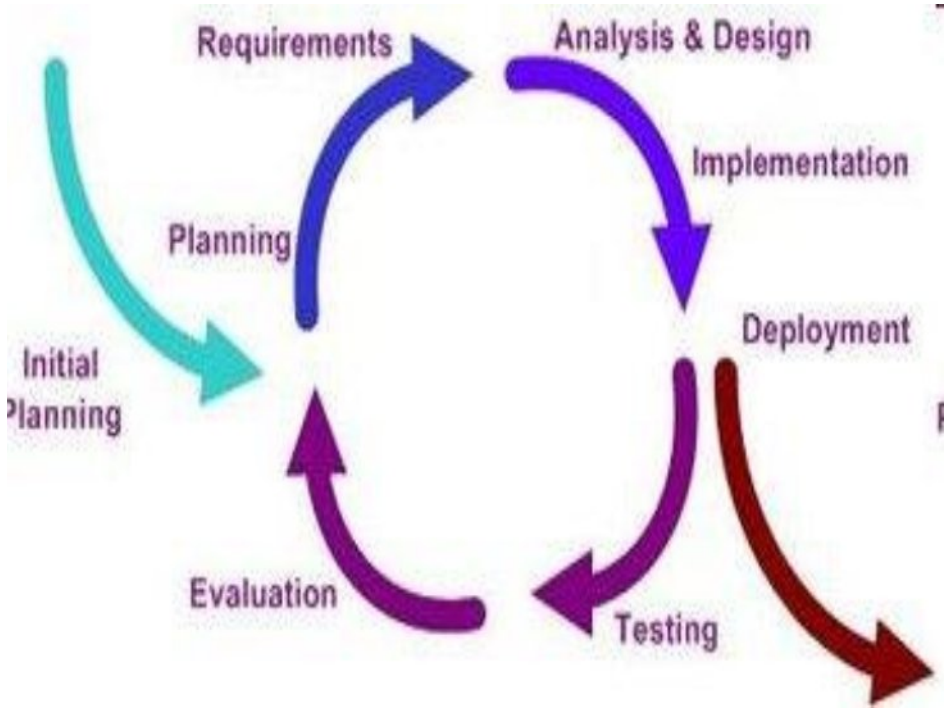
DISADVANTAGES

- It is only suitable for the small size projects.
- Constant testing of the design is needed.
- If requirements may change the Waterfall model may not work.
- Difficult to estimate time and cost for each stage of the development process.
- Adjust scope during the life cycle can kill a project.
- High amount of risk and uncertainty.
- This model is not suitable to handle dynamic changes in the requirements

Iterative model. **5M**

- First develop the nucleus of a system, then grow the scope of the system…
- There are multiple iterations as the system evolves to the final deliverable.
- Each iteration includes a full complement of stages:
  analysis, design, implementation, and testing

Diagram for Iterative model



5. Explain the steps involved in implementation modeling. **10M**

1. Overview of Implementation
2. Fine-tuning Classes
3. Fine-tuning Generalization
4. Realizing Associations

5. Testing
   (Above Mentioned points to be explained in details) **2*5=10M**

6. Explain the different categories of patterns. Discuss the properties of Architectural patterns. 10M
(Definition of pattern) **2M**

Types of patterns: 3M

Architectural patterns:
Fundamental structural organization is expressed for software systems that provide a set of predefined subsystems. Relationships are specified. These patterns include the rules and guidelines for organizingthe relationships.
Example :MVC

Design patterns:
Capture the static - dynamic roles & relationships in solutions that occur repeatedly.
Example: Observer Publisher subscriber

Idioms :
Restricted to a particular language.
Example: Counted Body

(Explain any 5 properties each carriers 1mark) 1*5=5M

The following points summarize the properties of patterns:

1) Addresses a recurring design problem that arises in specific design situations and presents a solution to it.

Example: Supporting variability in user interface. This problem may arise when developing software systems with human computer interaction.

2) Document existing, well-proven design experience. Patterns are not invented or created artificially. They distill and provide a means to reuse the design knowledge gained by experienced practitioners. Those familiar with an adequate set of patterns can apply them immediately to design problems without having to rediscover them.

3) Identify and specify abstractions that are above the level of single classes and instances or of components.A pattern describes several components, classes, objects. Patterns portray details, responsibilities, relationships and co-operation. All components together solve the problem that the pattern addresses. (more effectively than a single component).

4) Provide a common vocabulary and understanding for design principles. Pattern names if chosen carefully, become part of widespread design language. They facilitate effective discussion to design problems and solutions. They remove the need to explain a solution to a particular problem with a lengthy and complicated description.

5) Means of documenting software architectures. Patterns can describe the vision of the originator when designing a software system. This helps others to avoid violating this vision when extending and modifying the original architecture.

6) Support the construction of software with defined properties. Patterns provide a skeleton of functional behavior and therefore help to implement the functionality of the application. patterns also provide explicitly non-functional requirements for software systems such as reliability, testability and reusability.

supports the speed and quality of design. Understanding and applying well written patterns saves the time when compared to searching for the solutions. Not to say "better than own solutions", but at least provides a platform to evaluate and assess design alternatives. Patterns may not provide complete solution for new scenario and can be extended.

8) Help to manage software complexity. Every pattern describes a proven way to handle the problem it addresses on basis of the kinds of components needed, component roles, details that should be hidden, abstractions that should be visible and working parameters.

7. Explain Forward –Receiver design pattern **10M**

Defintion **1 M**

Forwarder-Receiver design pattern Provides transparent inter process communication for software systems with peer-to-peer interaction model. It introduces forwarders and receivers to decouple peers from the underlying communication mechanisms.

Example **1M**

The company DwarfWare offers applications for the management of the computer networks. System consists of agent processes written in Java that run on each available network node. These agents are responsible for observing and monitoring the resources. Routing tables get modified. Each agent is connected to remote agents in a peer-to-peer fashion, acting as client or server as required. As the infrastructure needs to support a wide variety of different hardware and software systems, the communication between the peers must not depend on a particular mechanism for inter-process communication.

Context – peer-to-peer communication **3M**

Problem forces –

- The system should allow the exchangeability of the communication mechanisms.
- The co-operation of components follows a peer-to-peer model, in which a sender only needs to know names of its receivers.
- The communication between peers should not have a major impact on performance.

Solution – peers may act as clients or servers. Therefore the details of the underlying IPC mechanisms for sending or receiving messages are hidden from peers by encapsulating all system-specific functionality into separate components. system specific functionalities are the mapping of names to physical locations, the establishment of communication channels and marshaling and unmarshaling messages.

Structure – Forwarders, receivers and peers are agents. Each peer knows the names of the remote peers with which it needs to communicate. It uses a forwarder to send a message to other peers and a receiver to receive the messages from other peers. The different types of messages that can be passed between the two peers are command messages, information messages and response messages. Command messages are related to implementation process. For instance, changing of routing tables of host machine is a command message. The payload i.e data on the network is communicated via the information messages. For instance, data on network resources and network events form information messages. Response messages deals with acknowledgement details such as, the arrival of a message.

Peer pattern

1) Continuously monitor network events and resources.
2) Listen for incoming messages from remote agents.
3) Each agent may connect to any other agent to exchange information and requests.
4) The network management infrastructure connects the network administrator's console with all other agents.
5) Administrators may send requests to network agents or retrieve messages from them by using available network administration tools.

Forwarder pattern
1) Sends messages across process boundaries.
2) Provides a general interface that is an abstraction of a particular IPC mechanism.
3) Includes functionality for marshaling and delivery of messages.
4) Contains a mapping from names to physical addresses.
5) Determines the physical location of the recipient by using its name to- addresses mapping.
6) In the transmitted message, the forwarder specifies its own peer, so that remote peer is able to send a response to the message originator.

Receiver pattern
1) Wait for incoming messages on behalf of their agent process.
2) As soon as the message arrives, they convert the received data stream into a general message format and forward the message to their agent process.

Class peer
Responsibility
_ Provides application services.
_ Communicates with other peers.
Collaborators
_ Forwarder
_ Receiver

Class Forwarder
Responsibility
_ Provides general interface for
sending messages.
_ Marshals and delivers messages
to remote receivers.
_ Maps names to physical
addresses.
Collaborators
_ Receiver

Class Receiver
Responsibility
_ Provides general interface for receiving
messages.
_ Receives and unmarshals messages
from remote forwarders.
Collaborators
_ Forwarder

Class Diagram for Forward –Receiver design pattern **2.5 M**

Sequence diagram for Forward –Receiver design pattern **2.5 M**