CMR
INSTITUTE OF
TECHNOLOGY

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

Internal Assesment Test - II

Sub:	SOFTWARE ENGINEERING							Code:	10IS51
Date:	02 / 11 / 2016	Duration:	90 mins	Max Marks:	50	Sem:	5	Branch:	CSE/ISE

Answer Any FIVE FULL Questions

		Marks	OBE	
			CO	RBT
1	List and explain different types of system models.	[10]	CO3	L2
2	Draw an activity chart showing the project schedule referring the table below. Find the critical path.	[10]	CO6	L3
3	Explain System Organization. Discuss the repository model and list out its advantages and disadvantages.	[10]	CO3	L2
4	Draw and explain the state model for Weather station system	[10]	CO3	L2
5 (a)	Explain with example object and class in OO Design.	[04]	CO3	L2
(b)	Define concurrent objects. Explain the implementation of concurrent objects.	[06]	CO3	L2
6	What is program dynamics? Explain Lehman's laws.	[02+08]	CO5	L2
7	Explain the principles of agile methods for software development. What are the shortcomings of agile methods?	[08+02]	CO1	L2
8	Explain different types of software maintenance. Discuss the reasons for high maintenance cost.	[10]	CO5	L2

Task	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
Duration(days)	8	15	15	10	10	5	20	25	15	15	7	10
Dependencies			T_1 (m_1)		T_2T_4 (m_2)	T_1T_2 (m_3)	T_1 (m_1)	T_4 (m_5)	T_3T_6 (m_4)	T_5T_7 (m_7)	T_9 (m_6)	T_{11} (m_8)

Course Outcomes		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1:	Select a process model for a particular project from waterfall model, evolutionary development, component based software engineering, incremental delivery, spiral development, rational unified process, agile development and rapid application development	3	2	3	-	2	1	2	-	2	2	2	3
CO2:	Design Software Requirement Document.	3	2	3	2	-	2	2	-	3	3	2	3
CO3:	Create a design document for a given requirement.	3	2	3	-	2	-	2	-	3	3	2	3
CO4:	Implement software testing strategies.	3	2	3	-	2	-	2	-	2	2	2	3
CO5:	Explain the activities involved in software evolution.	2	2	3	-	1	-	2	-	3	2	2	3
CO6:	Perform activities involved in project management like project planning, scheduling, risk management and cost estimation.	3	2	3	-	2	2	2	1	3	3	3	3

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 INSTITUTE OF TECHNOLOGY

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

Second Internal Assessment Test – Novemebr 2016

Sub:	Software Engineering			Code:	10IS51
Date: 02/11/2016	Duration: 90 mins	Max Marks: 50	Sem: V	Branch:	CSE/IS

Note: : Answer any 5 questions Total marks: 50

1. List and explain different types of system models. (10M)

System modelling helps the analyst to understand the functionality of the system and models are used to communicate with customers.

Different models present the system from different perspectives

- External perspective showing the system's context or environment;
- Behavioural perspective showing the behaviour of the system;
- Structural perspective showing the system or data architecture.

Model types: (2M)

- Data processing model showing how the data is processed at different stages.
- Composition model showing how entities are composed of other entities.
- Architectural model showing principal sub-systems.
- Classification model showing how entities have common characteristics.
- Stimulus/response model showing the system's reaction to events.

CONTEXT MODELS (1M)

- Context models are used to illustrate the operational context of a system - they show what lies outside the system boundaries.
- Social and organisational concerns may affect the decision on where to position system boundaries.
- Architectural models show the system and its relationship with other systems.

PROCESS MODELS (1M)

- Process models show the overall process and the processes that are supported by the system.
- Data flow models may be used to show the processes and the flow of information from one process to another.

BEHAVIOURAL MODELS (1M)

- Behavioural models are used to describe the overall behaviour of a system.
- Two types of behavioural model are:
  Data processing models that show how data is processed as it moves through the system;
  State machine models that show the systems response to events.
- These models show different perspectives so both of them are required to describe the system's behaviour.

DATA PROCESSING MODELS (2M)

- Data flow diagrams (DFDs) may be used to model the system's data processing.
- These show the processing steps as data flows through a system.
- DFDs are an intrinsic part of many analysis methods.
- Simple and intuitive notation that customers can understand.
- Show end-to-end processing of data.

DATA FLOW DIAGRAMS

- DFDs model the system from a functional perspective.
- Tracking and documenting how the data associated with a process is helpful to develop an overall understanding of the system.
- Data flow diagrams may also be used in showing the data exchange between a system and other systems in its environment.

STATE MACHINE MODELS

- These model the behaviour of the system in response to external and internal events.
- They show the system's responses to stimuli so are often used for modelling real-time systems.
- State machine models show system states as nodes and events as arcs between these nodes. When an event occurs, the system moves from one state to another.
- Statecharts are an integral part of the UML and are used to represent state machine models.

SEMANTIC DATA MODELS (1M)

- Used to describe the logical structure of data processed by the system.
- An entity-relation-attribute model sets out the entities in the system, the relationships between these entities and the entity attributes
- Widely used in database design. Can readily be implemented using relational databases.
- No specific notation provided in the UML but objects and associations can be used.

OBJECT MODELS (1M)

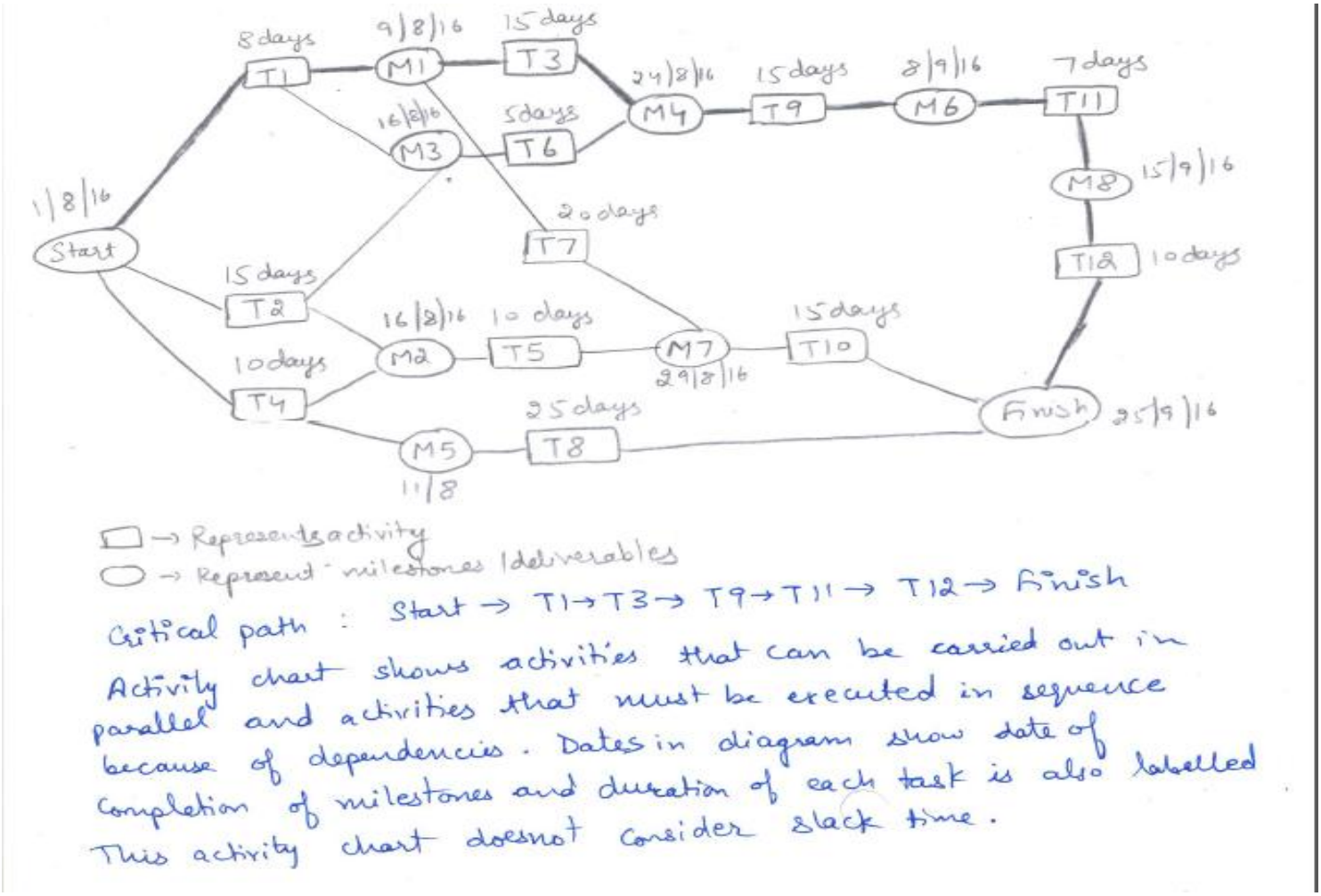
- Object models describe the system in terms of object classes and their associations.
- An object class is an abstraction over a set of objects with common attributes and the services (operations) provided by each object.
- Various object models may be produced
  Inheritance models;
  Aggregation models;
  Interaction models.

OBJECT BEHAVIOUR MODELS (1M)

- A behavioural model shows the interactions between objects to produce some particular system behaviour that is specified as a use-case.
- Sequence diagrams (or collaboration diagrams) in the UML are used to model interaction between objects.

2. Draw an activity chart showing the project schedule referring the table below. Find the critical path. (10M)

Task	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
Duration(days)	8	15	15	10	10	5	20	25	15	15	7	10
Dependencies			T_1 (m_1)		T_2T_4 (m_2)	T_1T_2 (m_3)	T_1 (m_1)	T_4 (m_5)	T_3T_6 (m_4)	T_5T_7 (m_7)	T_9 (m_6)	T_{11} (m_8)



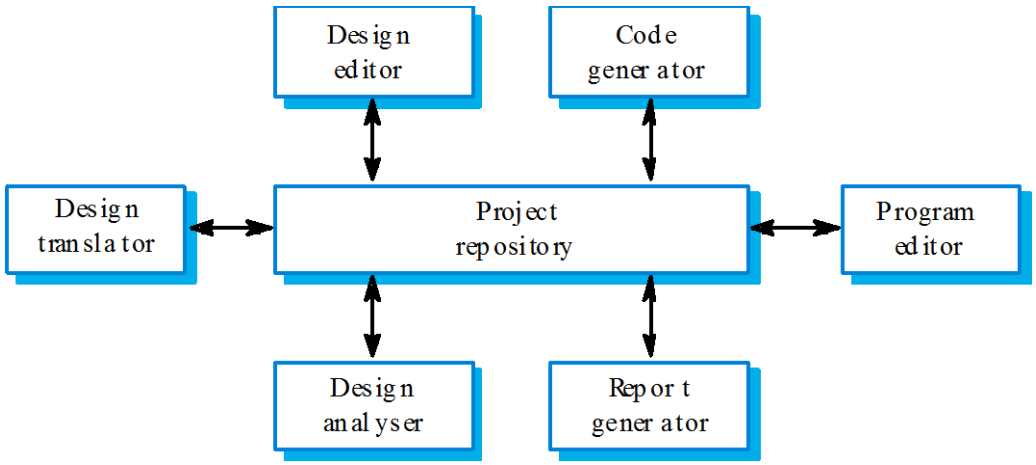
(8M diagram+2M Critical Path)

3. Explain System Organization. Discuss the repository model and list out its advantages and disadvantages. (10M)

System Organization (2M)

- Reflects the basic strategy that is used to structure a system.
- Three organisational styles are widely used:
    - A shared data repository style;
    - A shared services and servers style;
    - An abstract machine or layered style.

THE REPOSITORY MODEL (4M Explanation and diagram)



- Sub-systems must exchange data. This may be done in two ways:
  Shared data is held in a central database or repository and may be accessed by all sub-systems;
  Each sub-system maintains its own database and passes data explicitly to other sub-systems.

- When large amounts of data are to be shared, the repository model of sharing is most commonly used.

Advantages (2M)

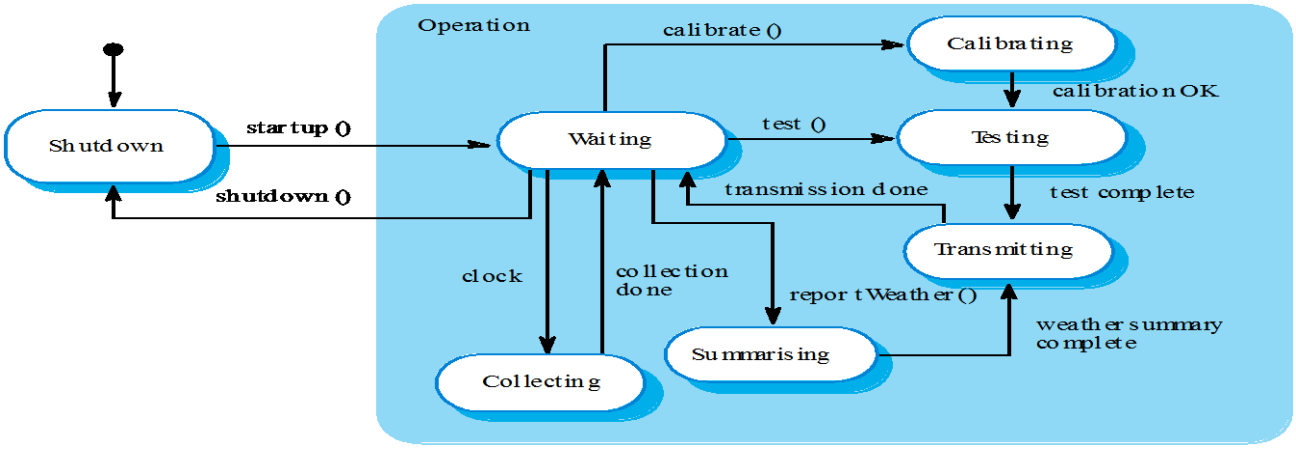
- Efficient way to share large amounts of data;
- Sub-systems need not be concerned with how data is produced Centralised management e.g. backup, security, etc.
- Sharing model is published as the repository schema.

Disadvantages (2M)

- Sub-systems must agree on a repository data model. Inevitably a compromise;
- Data evolution is difficult and expensive;
- No scope for specific management policies;
- Difficult to distribute efficiently.

4. Draw and explain the state model for Weather station system. (10M)

(7M diagram)



(3M Explanation)

Show how objects respond to different service requests and the state transitions triggered by these requests

- If object state is Shutdown then it responds to a Startup() message;
- In the waiting state the object is waiting for further messages;
- If reportWeather () then system moves to summarising state;
- If calibrate () the system moves to a calibrating state;
- A collecting state is entered when a clock signal is received.

5. (a) Explain with example object and class in OO Design. (4M)

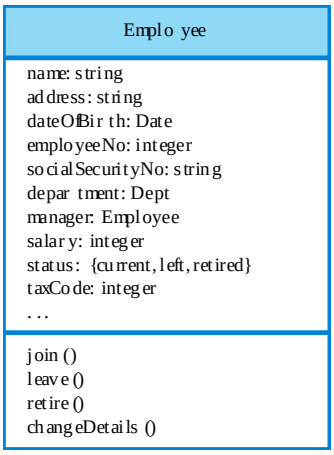
(3M explanation)

- Objects are entities in a software system which represent instances of real-world and system entities.
- Object classes are templates for objects. They may be used to create objects.
- Object classes may inherit attributes and services from other object classes.

An **object** is an entity that has a state and a defined set of operations which operate on that state. The state is represented as a set of object attributes. The operations associated with the object provide services to other objects (clients) which request these services when some computation is required. Objects are created according to some **object class** definition. An object class definition serves as a template for objects. It includes declarations of all the attributes and services which should be associated with an object of that class.

Example: (1M)

Employee object class:



(b) Define concurrent objects. Explain the implementation of concurrent objects. (6M)

(3M Explanation)

The nature of objects as self-contained entities make them suitable for concurrent implementation.

The message-passing model of object communication can be implemented directly if objects are running on separate processors in a distributed system.

Servers.

- The object is implemented as a parallel process (server) with entry points corresponding to object operations. If no calls are made to it, the object suspends itself and waits for further requests for service.

Active objects

- Objects are implemented as parallel processes and the internal object state may be changed by the object itself and not simply by external calls.

(3M Implementation)

```
class Transponder extends Thread {

        Position currentPosition ;
        Coords c1, c2 ;
        Satellite sat1, sat2 ;
        Navigator theNavigator ;

        public Position givePosition ()
        {
                return currentPosition ;
        }

        public void run ()
        {
                while (true)
                {
                        c1 = sat1.position () ;
                        c2 = sat2.position () ;
                        currentPosition = theNavigator.compute (c1, c2) ;
                }

        }

} //Transponder
```

Active objects may have their attributes modified by operations but may also update them autonomously using internal operations.

A Transponder object broadcasts an aircraft's position. The position may be updated using a satellite positioning system. The object periodically update the position by triangulation from satellites.

6. What is program dynamics? Explain Lehman's laws. (2M+8M)

(2M program dynamics)

Program evolution dynamics is the study of the processes of system change.

After major empirical studies, Lehman and Belady proposed that there were a number of 'laws' which applied to all systems as they evolved.

There are sensible observations rather than laws. They are applicable to large systems developed by large organisations. Perhaps less applicable in other cases.

LEHMAN'S LAWS **(1x8=8M)**

Law	Description
Continuing change	A program that is used in a real-world environment necessarily must change or become progressively less useful in that environment.
Increasing complexity	As an evolving program changes, its structure tends to become more complex. Extra resources must be devoted to preserving and simplifying the structure.
Large program evolution	Program evolution is a self-regulating process. System attributes such as size, time between releases and the number of reported errors is approximately invariant for each system release.
Organisational stability	Over a program's lifetime, its rate of development is approximately constant and independent of the resources devoted to system development.
Conservation of familiarity	Over the lifetime of a system, the incremental change in each release is approximately constant.
Continuing growth	The functionality offered by systems has to continually increase to maintain user satisfaction.
Declining quality	The quality of systems will appear to be declining unless they are adapted to changes in their operational environment.
Feedback system	Evolution processes incorporate multi-agent, multi-loop feedback systems and you have to treat them as feedback systems to achieve significant product improvement.

7. Explain the principles of agile methods for software development. What are the shortcomings of agile methods? (8M+2M)

PRINCIPLES OF AGILE (Principles-8M)

Principle	Description
Customer involvement	The customer should be closely involved throughout the development process. Their role is provide and prioritise new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognised and exploited. The team should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and design the system so that it can accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process used. Wherever possible, actively work to eliminate complexity from the system.

SHORTCOMINGS OF AGILE METHODS: (2M)

- It can be difficult to keep the interest of customers who are involved in the process.

- Team members may be unsuited to the intense involvement that characterises agile methods.
- Prioritising changes can be difficult where there are multiple stakeholders.
- Maintaining simplicity requires extra work.
- Contracts may be a problem as with other approaches to iterative development.

8. Explain different types of software maintenance. Discuss the reasons for high maintenance cost. (10M)

TYPES OF MAINTENANCE (4M)

Corrective: Maintenance to repair software faults

- Changing a system to correct deficiencies in the way meets its requirements.

Adaptive: Maintenance to adapt software to a different operating environment

- Changing a system so that it operates in a different environment (computer, OS, etc.) from its initial implementation.

Perfective: Maintenance to add to or modify the system's functionality

- Modifying the system to satisfy new requirements.

MAINTENANCE COST FACTORS: (6M)

Team stability

Maintenance costs are reduced if the same staff are involved with them for some time.

Contractual responsibility

The developers of a system may have no contractual responsibility for maintenance so there is no incentive to design for future change.

Staff skills

Maintenance staff are often inexperienced and have limited domain knowledge.

Program age and structure

As programs age, their structure is degraded and they become harder to understand and change.