CMR
INSTITUTE OF
TECHNOLOGY

USN	1	C								

Internal Assessment Test 3 – November 2019

Sub:	**Object Oriented Modeling and Design Patterns**							**Code:**	17MCA51
Date:	16-11-19	Duration:	90 mins	Max Marks:	50	**Sem:**	V A & B	**Branch:**	**MCA**

Note: Solution for OOMD – IAT3 Total Marks: 50

1. **Elaborate and explain the questions to be answered for a Good System Concept.**

A good system concept must answer the following questions:

I. Who is the application for?
- You should clearly understand who is the stake holders of the new system
- Two important kinds of stakeholders are
  1.financial Sponsors
  2.End Users
- Financial Sponsors: They are important because they are paying for the new system, and they expect the project to be on schedule and within budget.
- End users: They will ultimately determine the success of the new system by an increase in their productivity or effectiveness.

II. What problems will it solve?
- You must clearly bound to the size of the effort and establish its scope
- Determine features of the system by consulting various users of the system

III. Where it will be used?
- You should where the new system is used and determine if the system is mission-critical software for the organization, experimental software, or a new capability that you can deploy without disrupting the workflow.

IV. When it is needed?
- Two aspects of time are important.
- First the feasible time, the time in which the system can be developed within the constraint of cost and available resources
- Second the required time, when the system is needed to meet business goals

V. Why it is needed?
- Prepare a business case if it is not prepared for the new system.
- It contains the financial justifications for the new system, including the cost, tangible benefits, intangible benefits, risks and alternatives.

VI. How will it work?
- You should brainstorm about the feasibility of the problem.
- For large system you should consider the merits of different architectures.

- Here the purpose is not to choose the solution, but to increase the confidence that the problem can be solved reasonably.

2. **Discuss the steps to construct a domain state model, with an example of ATM**

Identify domain classes with states.
Finding states.
Finding events.
Build state diagrams.
Evaluate state diagrams.

Identifying Classes with States

Examine the list of domain classes for those that have a distinct life cycle. Look for classes that can be characterized by a progressive history or that exhibit cyclic behavior. Identify the significant states in the life cycle of an object.

Finding States

List the states for each class. Characterize the objects in each class-the attribute values that an object may have, the associations that it may participate in and their multiplicities, attributes and associations that are meaningful only in certain states, and so on. Don't focus on fine distinctions among states, particularly quantitative differences, such as small, medium, or large. States should be based on qualitative differences in behavior, attributes, or associations.

Finding Events

Once you have a preliminary set of states, find the events that cause transitions among states. Think about the stimuli that cause a state to change. In many cases, you can regard an event as completing a do-activity. For example, if a technical paper is in the state Under consideration, then the state terminates when a decision on the paper is reached. In this case, the decision can be positive (Accept paper) or negative (Reject paper).
ATM example. Important events include: close account, withdraw excess funds, repeated incorrect PIN, suspected fraud, and administrative action.

Building State Diagrams

The domain state model documents important classes that change state in the real world. Add transitions to show the change in state caused by the occurrence of an event when an object is in a particular state. If an event terminates a state, it will usually have a single transition from that state to another state. If an event initiates a target state, then consider where it can occur, and add transitions from those states to the target state.

Evaluating State Diagrams

Examine each state model. Are all the states connected? Pay particular attention to paths through it. If it represents a progressive class, is there a path from the initial state to the final state? Are the expected variations present? If it represents a cyclic class, is the main loop present? Are there any dead states that terminate the cycle?

3. **Describe the various steps to construct an application interaction model**

Determine the system boundary
Find actors

Find use cases
Find initial and final events
Prepare normal scenarios
Add variation and exception scenarios
Final external events
Prepare activity diagrams for complex use cases
Organize actors and use cases
Check against the domain class model.

4. **What is software control strategy? Explain the different types of it**

The analysis model shows interaction between objects. There are two kinds of control flows in a s/w system:

- External control
- Internal control

External control : concerns the flow of externally visible events among the objects in the system.
There are 3 kinds in this :

- Procedure-driven sequential
- Event driven sequential
- Concurrent.

Procedure-driven sequential :

- In this control resides within the program code.
- Procedure request external input and then wait for it; when input arrives , control resumes within the procedure that made the call.

Advantage:

- It is easy to implement with conventional languages

Disadvantage:

- It requires the concurrency inherent in objects to be mapped into a sequential flow of control. The designer must convert events into operations between objects.

Event-driven Control:

- In this system, control resides within a dispatcher or monitor that the language, subsystem, or OS provides.
- Developer attaches application procedure to events, and the dispatcher calls the procedures when the corresponding events occur. (Call back).
- Event-driven control is more difficult to implement with standard language than procedure-driven control.
- This system permits more flexible control than procedure-driven system.
- This system is helpful and preferred to procedure-driven system because mapping from events to program constructs is simpler and more powerful.

Concurrent Control :

Resides concurrently in several independent objects, each a separate task.
Such a system implements events directly as one-way message between objects.

A task can wait for input, but other tasks continue execution.
The OS resolves scheduling conflicts among tasks and usually supplies a queuing mechanism, so that events are not lost if a task is executing when they arrive.

Internal Control :
The developer expands on objects into lower-level operations on the same or other objects.

The difference between internal control and external interactions is external interactions inherently involve waiting for events, because objects are independent and cannot force other objects to respond;

Objects generate internal operations as part of the implementation algorithm, so their form of response is predicable.

5 **Write the various steps to design algorithms**

Steps to design algorithms.

Choose algorithms that minimize the cost of implementing operations
Select data structures appropriate to the algorithms
Define new internal classes and operations as necessary
Assign operations to appropriate classes

6 **Draw an informal diagram showing the organization of a system into sub-system with ATM as case study**

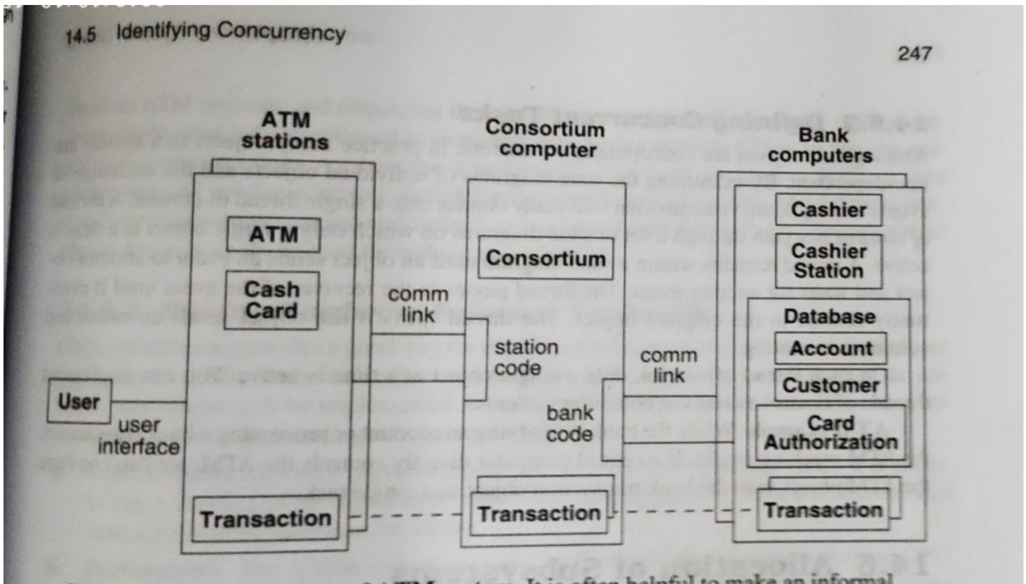

Figure 14.2 Architecture of ATM system. It is often helpful to make an informal diagram showing the organization of a system into subsystems.

7 **Describe in detail about the design optimization**

The analysis model captures the logical information about the system, while the design model adds details to support efficient information access. Before a design is implemented, it should be optimized to make the implementation more efficient. The aim of optimization is to minimize the cost in terms of time, space and other metrics.

However, design optimization should not be excess, as ease of implementation, maintainability and extensibility are also important concerns. It is often seen that a perfectly optimized design is more efficient but less readable and reusable. So, the designer must strike a balance between the two.

The various things that may be done for design optimization are-

- Add redundant associations
- Omit non-usable associations
- Optimization of algorithms
- Save derived attributes to avoid re-computation of complex expressions

Addition of Redundant Associations
During design optimization, it is checked if deriving new associations can reduce access costs. Though these redundant associations may not add any information, they may increase the efficiency of the overall model.

Omission of Non-Usable Associations
Presence of too many associations may render a system indecipherable and hence reduce the overall efficiency of the system. So, during optimization, all non-usable associations are removed.

Optimization of Algorithms
In object-oriented systems, optimization of data structure and algorithms are done in a collaborative manner. Once the class design is in place, the operations and the algorithms need to be optimized.
Optimization of algorithms is obtained by –

1. Rearrangement of the order of computational tasks
2. Reversal of execution order of loops from that laid down in the functional model
3. Removal of dead paths within the algorithm

Saving and Storing of Derived Attributes

Derived attributes are those attributes whose values are computed as function of other attributes (base attributes). Re-computation of the values of derived attributes every time they are needed is a time-consuming procedure. To avoid this, the values can be computed and stored in their forms.

However, this may pose update anomalies, i.e., a change in the values of base attributes with no corresponding change in the values of the derived attributes. To avoid this, the following steps are taken –

1. With each update of the base attribute value, the derived attribute is also recomputed.
2. All the derived attributes are re-computed and updated periodically in a group rather than after each update.

8 **Define System Design and the list the detailed decisions involved in system design**

Overview of System design
Estimating performance
Making a reuse plan
Breaking a system into subsystem
Identifying concurrency
Allocation of subsystems
Management of data storages
Handling global resources
Choosing a software control strategy
Handling boundary conditions

9 **Discuss the steps involved in Class design**

Overview of class design
Bridging the gap
Realizing use cases
Designing Algorithms
Recursing downward
Refactoring
Design optimization
Reification of behavior

10 **Explain the following**

1.Functional Layers
Functionality recursion takes the required high-level functionality and breaks it into lesser operations.

2.Mechanism Layer
Mechanism recursion means that we build the system out of layers of needed support mechanism.

3.Refactoring
The initial design of set of operations will contain inconsistencies, redundancies and inefficiencies**.** Refactoring is defined as changes to the internal structure of software to improve its design without altering its external functionality.

4.Reification
Reification is the promotion of something that is not an object into an object.