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

Sub:	Object Oriented Modeling and Design					Sub Code:	10CS71	Branch:CSE
Date:	7/11/2017	Duration:	90 min's	Max Marks:	50	Sem/Sec:	7^{th} / A ,B,C	

Answer any FIVE FULL Questions — MARKS

1 (a) Prepare a use case diagram for airline reservation system(ARS) and explain. [06]

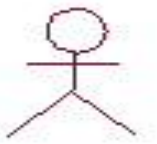
- Use Case diagrams show the various activities the users can perform on the system.
  - System is something that performs a function.
  - They model the dynamic aspects of the system.
- Provides a *user's* perspective of the system.

A use case is a model of the interaction between External users of a software product (actors) and The software product itself

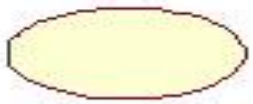
- More precisely, an actor is a user playing a specific role describing a set of user scenarios capturing user requirements contract between end user and software developers

- Use case diagrams are used to visualize, specify, construct, and document the (intended) behavior of the system, during requirements capture and analysis.
- Provide a way for developers, domain experts and end-users to communicate.
- Serve as basis for testing.
- Use case diagrams contain use cases, actors, and their relationships.

- **Actors:** A role that a user plays with respect to the system, including human users and other systems. e.g., inanimate physical objects (e.g. robot); an external system that needs some information from the current system.
- **Use case:** A set of scenarios that describing an interaction between a user and a system, including alternatives.



Actor



Use Case

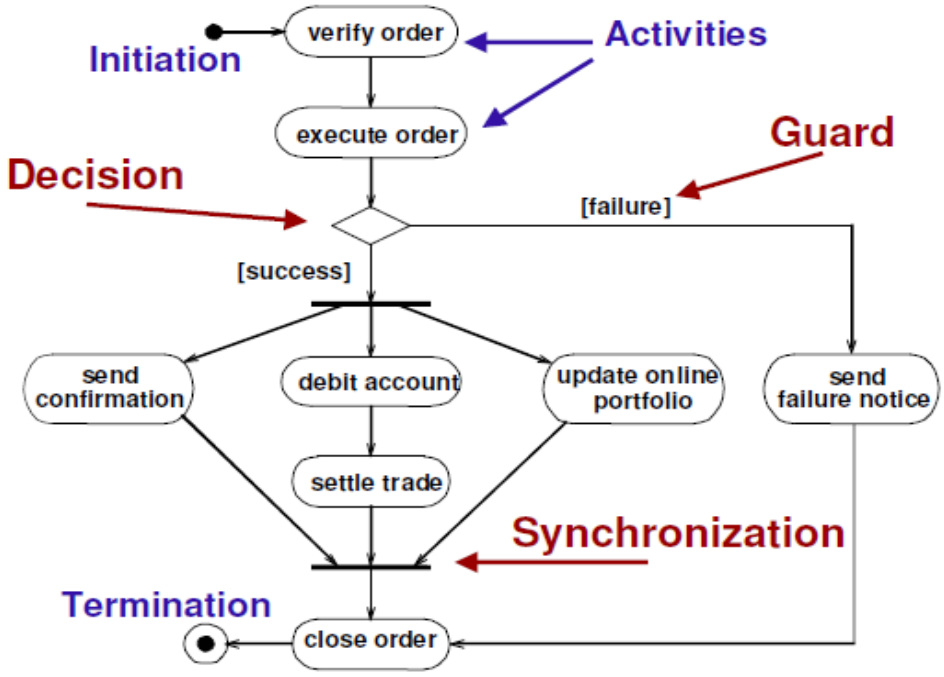
- **System boundary**: rectangle diagram representing the boundary between the actors and the system.
- **Association:** **C**ommunication between an actor and a use case;

(b) Explain activity diagram for an UML with an illustration showing stock trade processing. [04]

Definition of Activity 1M
Diagram With Explaination 3M

- Activity diagrams and use cases are logical model which describe the business domain's activities without suggesting how they are conduct.
- A diagram that emphasizes the flow of control from activity to activity in an object.
- Similar to the traditional program flowchart.
- Used to provide detail for complex algorithms.

- Primary activities and the relationships among the activities in a process.



2 (a) Draw and explain in detail about nested states for a phone line. [7]

Diagram 4M
Explaination 3M

A state may be represented as nested substates.

- In UML, substates are shown by nesting them in a superstate box.
- A substate inherits the transitions of its superstate.

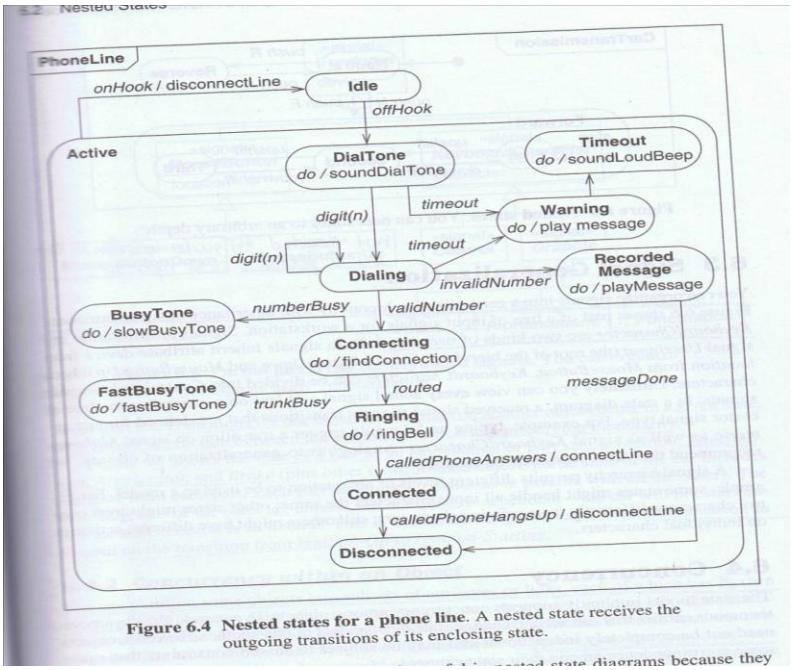


Figure 6.4 Nested states for a phone line. A nested state receives the outgoing transitions of its enclosing state.

- Composite State name (Ex: Active]
- Each of the nested state receives the outgoing transitions of its composite state

- Incoming transitions should be explicit into a nested state – cannot be to the outer Composite state
- Possible to depict transitions from a nested state to a state outside of the contour of nesting
- Entry and Exit activities can be associated at the composite state level
- When there is multiple level nesting, entry is executed outside in and exit inside out. Similar to nested sub-routine calls.

2b. What is Aggregation conncurrent? Illustrate with example

Definition 1M [3]

Diagram 2M

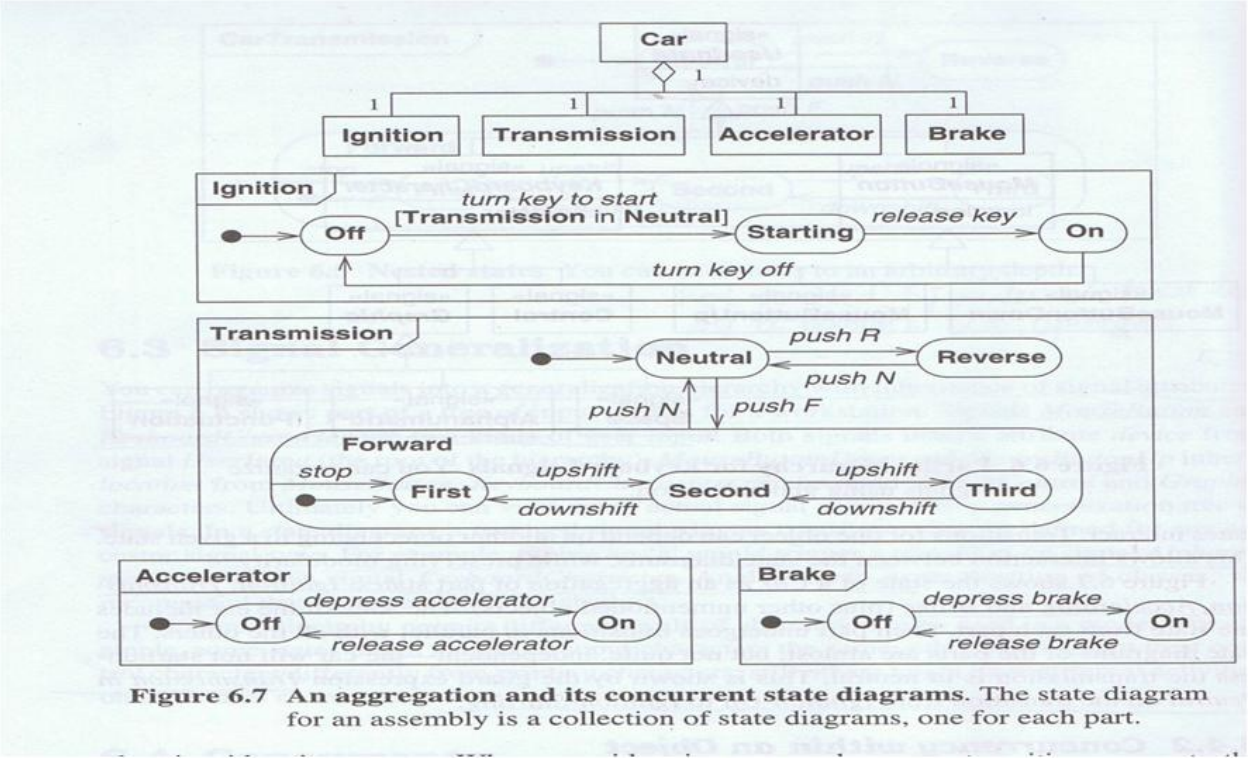


Figure 6.7 An aggregation and its concurrent state diagrams. The state diagram for an assembly is a collection of state diagrams, one for each part.

3 (a) With a neat sequence diagram, explain process transaction scenario [7]

- Sequence of events that occurs during one execution of a system.
- A scenario displayed as a list of text statements.
- A sequence diagram shows the participants in an interaction and the sequence of messages among them.
- A sequence diagram shows the interaction of a system with its actors to perform all or part of a use case.
- Each use case requires one or more sequence diagrams to describe its behaviour.
- Sequence diagrams, also known as event diagrams or event scenarios, illustrate how processes interact with each other by showing calls between different objects in a sequence.

These diagrams have two dimensions:

- The vertical lines show the sequence of messages and calls in chronological order
- Horizontal elements show object instances where the messages are relayed.

(b) List the steps involved in construction application interaction model. [3]

Mention 10 points (each point carries 0.3 M) 10*0.3=3 M

- **<u>Steps to construct model</u>**
- Determine the system boundary
- Find actors
- Find use cases
- Find initial and final events
- Prepare normal scenarios
- Add variation and exception scenarios

- Find external events
- Prepare activity diagrams for complex use cases.
- Organize actors and use cases

4 (a) Explain the steps followed in constructing application class model. [06]

Explanation of any 6 point where each carries 1M 6*1=6M

1. Bridging the gap
2. Realizing Use Cases
3. Designing Algorithms
4. Recursing Downward
5. Refactoring
6. Design Optimization
7. Reification of Behavior
8. Adjustment of Inheritance
9. Organizing a Class Design

5 (a) List and explain the steps involved in the design of algorithms [10]

Need of Algorthims 1M

Explain 3 steps of design ing Algorithms *3*3=9M*

- Formulate an ***algorithm for each operation 3*3=9M***
- The analysis specification tells *what the operation does for its clients*
- The algorithm show *how it is done*

Designing Algorithms- steps

i. Choose *algorithms that minimize the cost of implementing operations.*
ii. Select *data structures appropriate to the algorithms*
iii. Define new internal classes and operations as necessary.

Choosing algorithms (Choose algorithms that minimize the cost of implementing operations)

- When efficiency is not an issue, you should use simple algorithms.
- Typically, 20% of the operations consume 80% of execution time.
- Considerations for choosing alternative algorithms
- Flexibility
- Simple but inefficient
- Complex efficient

ATM Example

- Interactions between the consortium computer and bank computers could be complex.

ii. Choosing Data Structures (select data structures appropriate to the algorithm)

a. Algorithms require data structures on which to work.
b. They organize information in a form convenient for algorithms.
c. Such as arrays, lists, queues, stacks, set…etc.

iii. Defining New Internal Classes and Operations

a. To invent new, low-level operations during the decomposition of high-level operations.
b. The expansion of algorithms may lead you to create new classes of objects to hold intermediate results.

ATM Example:

i. Process transaction uses case involves a customer receipt.
ii. A *Receipt class is added.*

6 What is System design? Explain reuse concept of system design. [10]

• Two aspects of reuse:
i) Using existing things
ii)Creating reusable new things
• Reusable things include:
– Models
– Libraries
– Frameworks
– Patterns

Reusable Libraries

• A library is a collection of classes that are useful in many contexts.
• Qualities of "Good" class libraries:
– *Coherence* – well focused themes
– *Completeness* – provide complete behavior
– *Consistency* - polymorphic operations should have consistent names and signatures across classes
– *Efficiency* – provide alternative implementations of algorithms
– *Extensibility* – define subclasses for library classes
– *Genericity* – parameterized class definitions
• Problems limit the reuse ability:
– Argument validation
• Validate arguments by collection or by individual
– Error Handling
• Error codes or errors
– Control paradigms
• Event-driven or procedure-driven control
– Group operations
– Garbage collection
– Name collisions

Reusable Frameworks

• A framework is a skeletal structure of a program that must be elaborated to build a complete application.
• Frameworks class libraries are typically application specific and not suitable for general use.

Reusable Patterns

• A pattern is a proven solution to a general problem.
• There are patterns for analysis, architecture, design, and implementation.
• A pattern is more likely to be correct and robust than an untested, custom solution.
• Patterns are prototypical model fragments that distill some of the knowledge of experts.

Pattern vs. Framework

• A pattern is typically a small number of classes and relationships.
• A framework is much broader in scope and covers an entire subsystem or application.

7 (a) Differentiate between Forward Engineering and Reverse Engineering [5]

Each Point Carries 5*1=5M

Forward Engineering	Reverse Engineering
Given requirements, develop an application	Given an application, deduce tentative requirements
More certain	Less certain
Prescriptive	Adaptive
More mature	Less mature
Time consuming	10 to 100 times faster than forward engineering

(b) Define Sequence models? Explain the Different types of Sequence model [5]

Definition of Procedural Sequence Diagram 2M

Example 3M

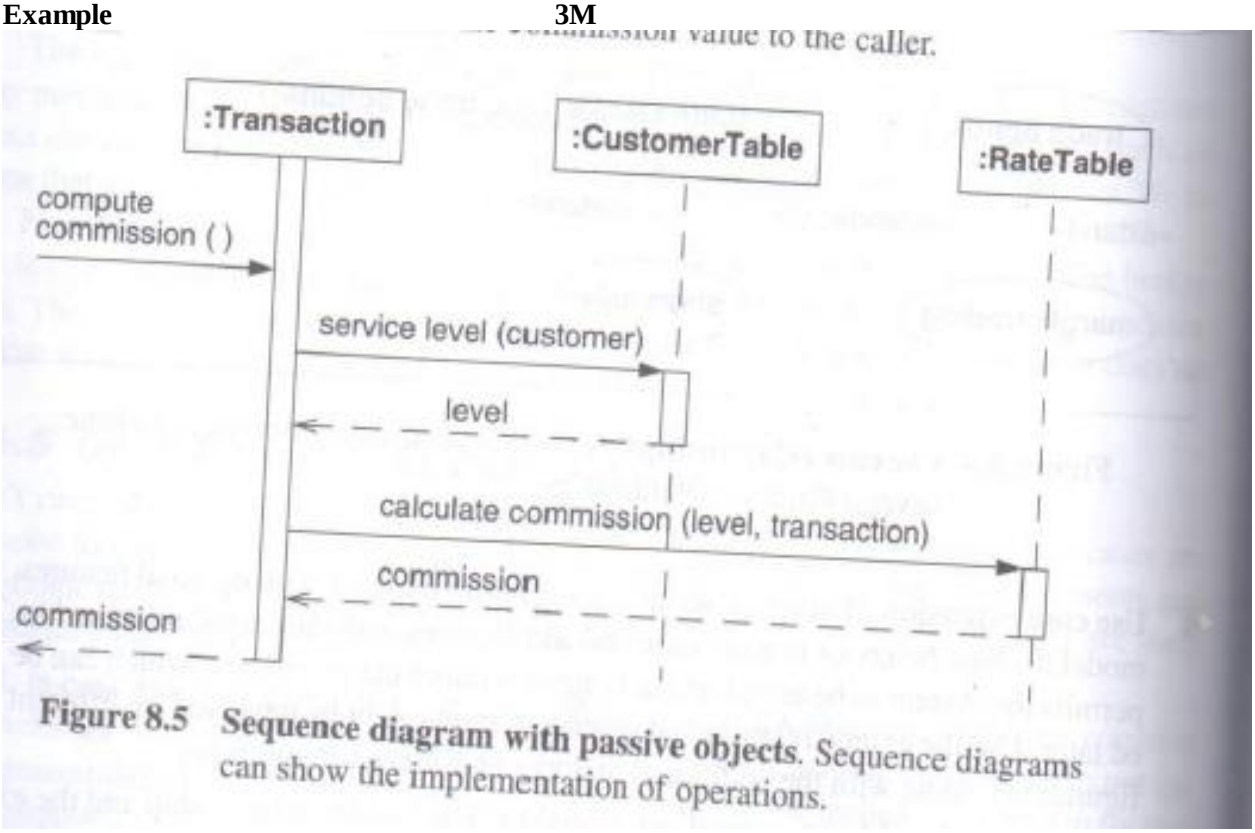


Figure 8.5 Sequence diagram with passive objects. Sequence diagrams can show the implementation of operations.

8 (a) Write a short note on [5+5]

i)Recursing downward

ii) include and extend w.r.t usecase

Definition of Include and notation 1.5M
Example 1 M

Include: a dotted line labeled <<include>> beginning at base use case and ending with an arrows pointing to the include use case. The include relationship occurs when a chunk of behavior is similar across more than one use case. Use "include" in stead of copying the description of that behavior.

<<include>>

– – – – – – – – – – – – – – – ▶

- The base use case explicitly incorporates the behavior of another use case at a location specified in the base.
- The included use case never stands alone. It only occurs as a part of some larger base that includes it.

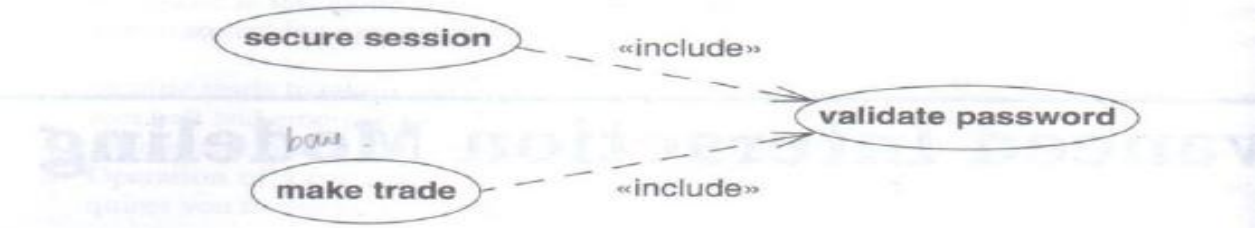


Figure 8.1 Use case inclusion. The *include* relationship lets a base use case incorporate behavior from another use case.

Definition of Extend and notation 1.5M
Example 1 M

Extend: a dotted line labeled <<extend>> with an arrow toward the base case. The extending use case may add behavior to the base use case. The base class declares "extension points".

<<extend>>

- ▶

- The base use case implicitly incorporates the behavior of another use case at certain points called extension points.
- The base use case may stand alone, but under certain conditions its behavior may be extended by the behavior of another use case.
- In UML modeling, you can use an extend relationship to specify that one use case (extension) extends the behavior of another use case (base)
- This type of relationship reveals details about a system or application that are typically hidden in a use case

1 Use Case Relationships

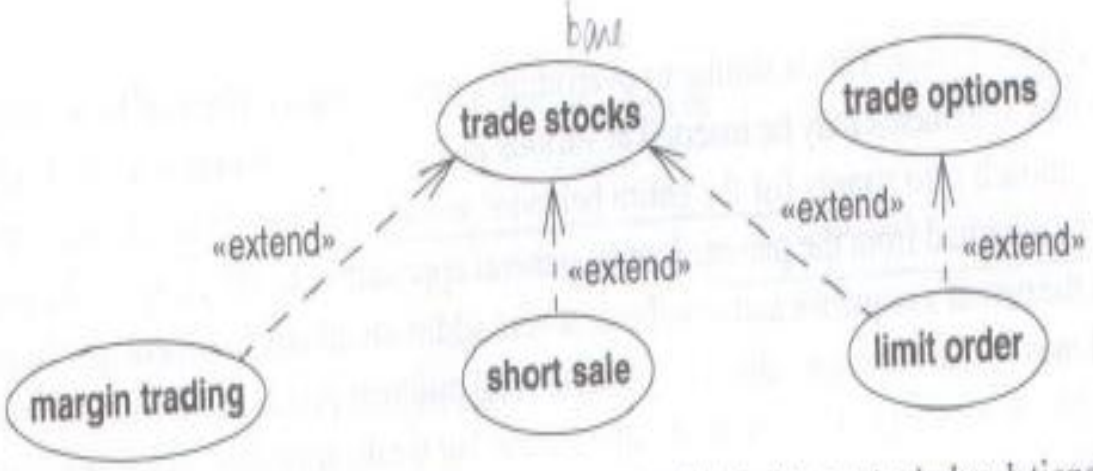
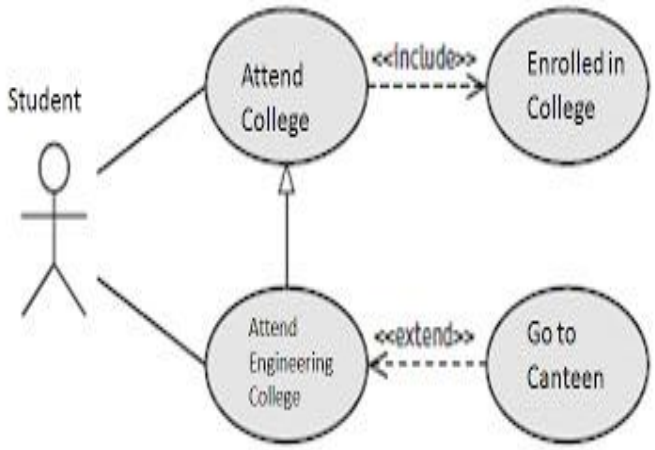


Figure 8.2 Use case extension. The *extend* relationship is like an *include* relationship looked at from the opposite direction. The extension adds itself to the base.



---------- ALL THE BEST----------