Unit III

CHAPTER 3

Advanced State Modeling and Interaction Modeling

University Prescribed Syllabus

Advanced State Modeling: Nested state diagrams; Nested states; Signal generalization; Concurrency; A sample state model; Relation of class and state models; Practical tips. Interaction Modeling: Use case models; Sequence models; Activity models. Use case relationships; Procedural sequence models; Special constructs for activity models.

Specific Instructional Objectives :

At the end of this lesson the student will be able to :

- Identify nested states.
- To draw state diagram and nested state diagram.
- Explain signals and signal generalization.
- Describe what a concurrency is?
- Differentiate amongst class model and state model.
- Describe use case model.
- Identify actors and their responsibilities, use cases.
- To draw use case diagram.
- Design and explain use case relationships.
- Design and explain sequence model.
- Explain procedural sequence model.
- Explain activity model.
- Describe special constructs for activity model.
- Design and explain a sequence diagram.
- Design and explain a activity diagram.

3.1.2 Concurrent Substates

- If we have to show two or more state machines that executes concurrently, then we can make use of concurrent substates in order to represent the concurrency amongst two or more state machines.
- Use of sequential substates is too common in state modeling, but in extraordinary cases, we need to specify concurrent substates to show parallel flow of execution in two nested state machines.
- A concurrent substate does not have an initial and final state. But, the sequential substates that comprise concurrent substates may have an initial and final state.
- Execution of two or more concurrent substates continues in parallel. Each nested substate attains its final state ultimately. If one of the substate achieves its final state before the other substate then, control of that particular concurrent substate pauses at its final state only and do not continues further. As soon as second concurrent substate reaches at its final state, both of the substates then combines into one flow.
- Fig. 3.1.3 shows concurrent substates.

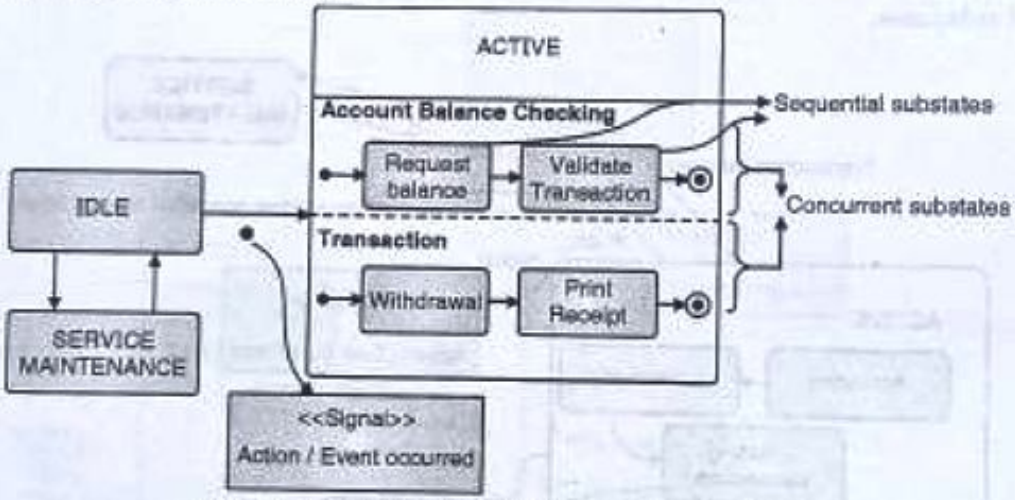

Fig. 3.1.3 : Nested State Diagram with concurrent substates

- In the Fig. 3.1.3, account balance checking and transaction are two concurrent substates which comprises of two sequential substates. Account balance checking consists of request balance state and validate transaction substates that executes sequentially. Transaction is the other concurrent substate that again consists of two sequential substates, withdrawal and print receipt.

3.1.3 Signal Generalization

- In advanced state model, we can arrange signals in generalization.
- The signal generalization mechanism offers use of different levels of abstraction in the state model.
- Each and every signal in the state model can be viewed as a leaf node or child node in the generalization hierarchy.

3.1.4 Concurrency

- The advanced state model indirectly supports and permits concurrency between several objects involved within the software system scenario.
- For achieving concurrency in state model, each and every object should share their features, operations and constraints with other objects in the system.
- Interdependency between several objects refers to concurrency in state model.

3.1.5 State Diagram Case Study : ATM Machine

GQ. Draw a state diagram for ATM system.

- As an example, we are going to study state diagram for an ATM Machine.
- At the outset, welcome message is displayed on the ATM Machine screen and ATM is in an idle state at this stage. When customer presses ENTER button on the screen, ATM requests for entering a card into the given slot.
- At this stage, ATM navigates from idle state to waiting state, since after insertion of a card, ATM asks customer for entering valid PIN on the screen.
- After successfully login into the system, customer should enter his/her choice for making transaction as per his/her requirement.
- In case of unsuccessful login (i.e., if entered PIN is invalid); then customer should remove card from the ATM slot and should again follow the same procedure for login into the system.
- Customer should then proceed for transaction by following instructions given on the ATM screen. Customer should then select his/her type of account and should enter amount to be withdrawn.
- After this, ATM machine is responsible for processing transaction initiated by a customer within a particular time instance. Within this transaction, ATM machine contacts to the bank's central server for checking the exact balance amount in the customer's account and validates that particular transaction initiated by customer.
- ATM machine then gives out the amount requested by a customer along with the printed receipt of the transaction. These are the basic activities involved in the transactions made via ATM machine.
- Fig. 3.1.4 depicts state diagram for ATM machine example at its initial phase, working phase and terminating phase respectively.

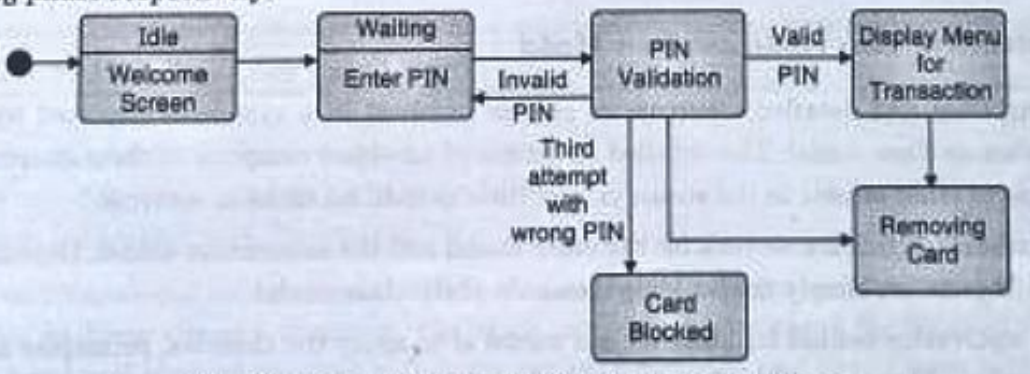

Fig. 3.1.4 : State transitions in ATM Machine at initial phase

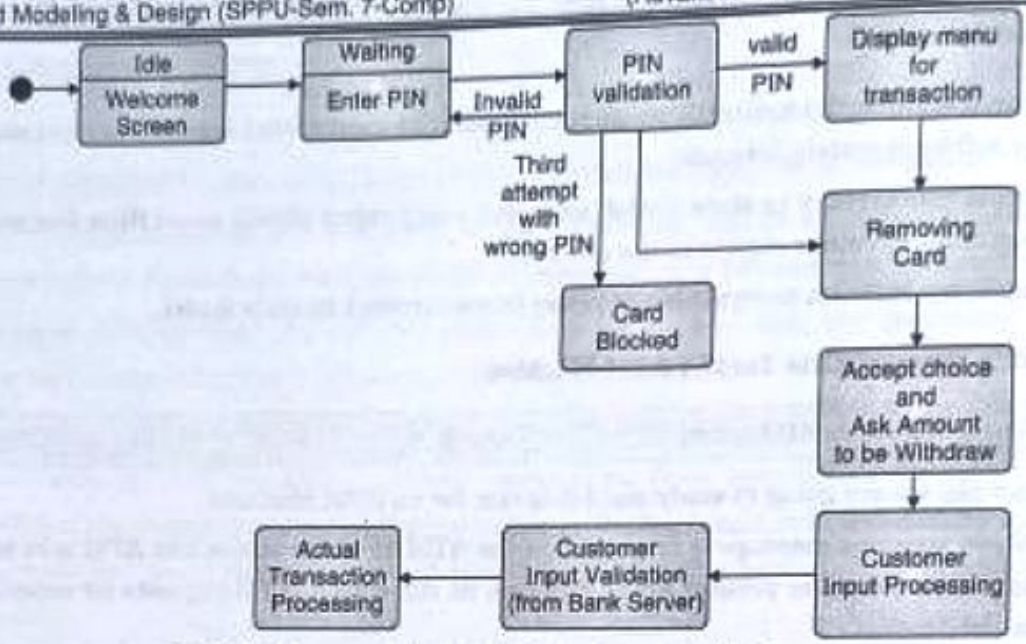

Fig. 3.1.5 : State transitions in ATM Machine at working phase

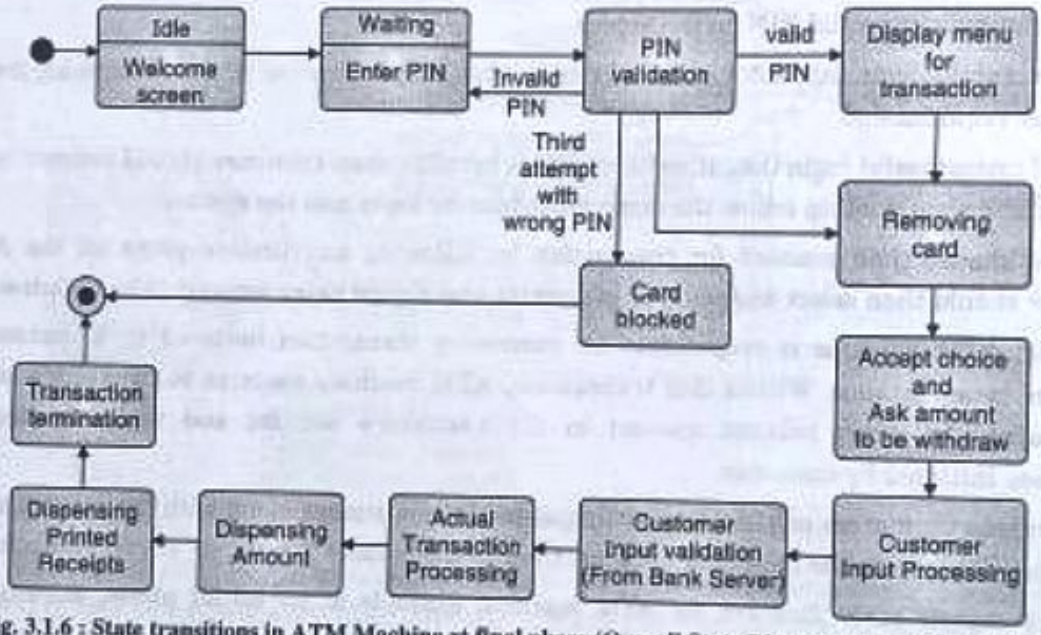

Fig. 3.1.6 : State transitions in ATM Machine at final phase (Overall State Diagram for ATM Machine)

3.1.6 Relation of Class Model and State Model

- The organization and detailed structure of objects involved in a system is depicted by means of the model known as class model. The detailed structure of an object comprise of their characteristics, their interactions to other objects in the scenario, and their individual tasks or services.
- The class model offers a framework for the state model and the interaction model. Objects are instances of classes. Objects are simply the building elements of the class model.
- The main motivation behind building a class model is to apply the theories, principles and conceptions from the real world that are significant and vital to the proposed software system.

- The class model must comprise of the things familiar to end users of the proposed system. Basically, the class model is articulated by means of class diagram.
- Objects from the class model are described with respect to sequencing and timing of operations in case of the state model. The state model describes those aspects of objects concerned with time and the sequencing of operations.
- The state model simply organizes and systemizes the states and the event involved in the proposed software system. The state model is dedicated for detailed description of sequence of tasks and services involved within the system operations and do not deals with the contents like details of operations and services, how different services and operations are executed during implementation and execution of the proposed system, etc. The state model is basically articulated by means of a state diagram.
- The state diagram represents the sequences of states and events that are tolerable in a proposed system for individual class of objects. The state diagram talks about the other models too.
- Tasks or jobs of a particular object from the class model are equivalent to the events and activities from the state diagram. References between state diagrams become interactions in the interaction model.

Guidelines for designing a State Diagram

- A state diagram should be simple enough and all the states, events and transitions involved in the software system scenario should be named from the vocabulary of the system for better understanding of a software system archetype.
- It should not contain any unnecessary transactions, events or states.
- Active classes and concurrent substates should be used in order to represent sequential flow and concurrency amongst the active objects in the software subsystem archetype.
- When something is happening into the system, a state or an event should be named with the help of the situation, condition or a particular point of time. Each state should be named with unique names for unique identification of the state in the system archetype.
- Actions and events involved in the scenario should be categorized in proper manner. Actions are the effects of the state transitions and events are the causes of the state transitions.
- Every state should have exit state, i.e., it should be conceivable to exit from each state.
- Actions and conditions should be used in proper situations and it is not at all compulsory to declare actions and conditions in the state archetype.

3.2 Interaction Modeling : Use Case Model

GQ. What do you mean by use case model? Describe in brief.

GQ. Explain the following terms with respect to use case model :
(a) Use case (b) Actor (c) System boundary

- The elementary functional requirements of the software system are explained with the help of use case modeling. As we have already discussed, the state modeling is dedicated for detailed description of sequence of tasks and services involved within the system operations and do not deals with the contents

like details of operations and services, how different services and operations are executed during implementation and execution of the proposed system, etc.

- But, initially software system designers should have a basic idea about the input given to the software system and output generated from the software system. So, use case models are used in primary phase of the software system design in order to define input to and output from the proposed software system.
- The use case model depicts functional requirement of the proposed system by means of the use cases and the actors.
- Use case model supports in the design of the software system by presenting intended behavior of the proposed system without any description about the implementation of the system.
- Let us discuss the three basic components of the use case model : use cases, actors and relationships in subsequent sections.

3.2.1 Use Cases

GQ. Write a short note on : Use case.
GQ. Draw and explain use case scenario in brief. Give suitable example.
GQ. List and explain the elements of use case scenario.

- In UML, the system requirements and functionality of the system are depicted with the help of use cases.
- Use cases are meant for specification of the interaction between the system itself and end users of the system which are termed as actors in UML.
- Basically, use case offers a detailed description of how the system is used.
- The set of activities and events in some proper sequence specifying the interaction amongst a system and its end users is known as a scenario.
- We might have to handle several scenarios in the execution of the system depending on the situations or scenarios involved within the proper execution of the system.
- For instance, in case of an ATM machine, access will be given to authorize customer only by authenticating the particular customer through his/her card and PIN. If login is successful, then customer will be in position to perform the transaction. But, if the login is failed, no access will be given to the customer and customer will not be able to perform the transaction through an ATM machine. So, both of these scenarios are different which are the things that might happen.
- All of the scenarios mentioned in the above example are different but are equivalent since the customer has the same goal in all of the scenarios i.e., to perform the transaction. This goal is only the main component behind definition of use cases.
- A use case is defined as a set of scenarios that collectively work to achieve a common user goal. It outlines a sequence of interactions amongst one or more actors and the system itself.
- In the above example, transaction processing can be one of the use case that strives for the successful transaction processing that is the goal of the customer.
- Each use case comes with its primary actor who is responsible for certain tasks involved in the scenario. More details of the actor are discussed in subsequent section of this chapter. Each use case should clearly mention the exact interaction between the actors and the system itself.
- Furthermore, use case guides us about what the system exactly does in response to the actor's activity and it is not devoted for detailing how system does it.

- At all times, a use case starts with input from a respective actor. Thus, an actor is responsible for giving input the system and the system is dedicated for giving response to the actor.
- A simple use case encompasses a single interaction between the actor and the system. But, a single use case can handle set of interactions between a particular actor and the system. More than one actor can be a part of the complex use cases.
- Graphically, a use case is shown with the help of an oval shape containing the use case name inside its body. Each use case should be named uniquely and use case name should describe the desired functionality of that particular use case.
- We can describe the use case in more detail by making partition inside the oval shape and details of the respective use case can be given in the second partition.
- The contents of the second partition are called as extension points. Use case can be depicted as a classifier also. Fig. 3.2.1 shows a simple use case notations in UML : (a) A simple use case (b) Use case with extension points and (c) Use case as a classifier.

(a) A simple use case (b) Use case with extension points (c) Use case as a classifier

Fig. 3.2.1 : Use Case notations in UML

- More fundamentally, any use case explanation consists of five entities:
  - The name of the use case.
  - The name of the actor.
  - A brief explanation of the use case.
  - A brief explanation of sequence of activities and events involved in a use case.
  - Preconditions and postconditions necessary for successful execution of the use case.
- More detailing of a particular use case can be done with the help of following use case scenario template :

Use case ID :	
Use case name :	
Use case description :	
Actors :	
Preconditions :	
Main sequence description :	
Alternative sequence description :	
Nonfunctional requirements :	
Postconditions :	
Frequency of use :	

Fig. 3.2.2 : Use Case Scenario Template

- As shown in the Fig. 3.2.2, software designers should describe each and every use case involved in the scenario of the system archetype during use case modeling.
  - ***Use case ID*** : Use case ID acts as an absolute unique identifier for unique identification of particular use case in the software system scenario. Even though use case name is used for unique identification of a particular use case, it may change over time. So, it is moderately necessary to have a unique identifier for the individual use case.
  - ***Use case name*** : A unique name should be given to each use case and it should specify the functionality of the use case. Use case name should be a verb since use cases specifies behavior of the system. Use case name should be descriptive but short.
  - ***Use case description*** : Use case should be described in brief in this section. This section should summarize the objectives of the use case.
  - ***Actors*** : actors can be classified as primary actors and secondary actors from the use case perspective. Primary actors are the main actors and they initiate the use case while secondary actors cooperates with use case after initiation of that particular use case.
  - ***Preconditions*** : Preconditions are simply nothing but the limitations that should be attained before triggering the use case. Preconditions do not permit actors for initiation of the use case unless and until all preconditions are satisfied.
  - ***Main sequence description*** : Actual flow of events and activities is described in this section.
  - ***Alternative sequence description*** : The use case may have alternative flow of execution along with the main flow. The alternative flow do not return to the main flow repeatedly. Possible alternatives to the main flow gives us the alternative flow for the execution of the use case.
  - ***Nonfunctional requirements*** : Security requirements and Performance requirements can be declared in this section for better understanding of the use case scenario.
  - ***Postconditions*** : Postconditions are the constraints that should be achieved after successful execution of a particular use case.
  - ***Frequency of use*** : As its name suggests, this section describes the frequency of use of a particular use case during single successful execution of the software system.
- For better understanding of the use case scenario template, refer Fig. 3.2.3 showing the use case scenario template for login into the ATM machine system for transaction.

NOTES

Use case ID:	01
Use case name:	Login into the ATM Machine.
Use case description:	After successful account opening within the system, the user will be provided with the unique account PIN with the help of which, user can get entry into the system for making transactions through ATM machine.
Actors:	Customer
Preconditions:	Customer should open an account in the bank and should avail the facility of using ATM. From ATM machines' perspective, ATM machine should have sufficient free memory available in order to launch the task.
Main sequence description:	Click the ENTRY button on the ATM machine screen. Enter PIN number for getting logged in into the system. After authentication and validation from server side, an authorized access will be given to the customer.
Alternative sequence description:	NIL
Nonfunctional requirements:	Unique Account PIN should be provided to the customer which will be used by the customer while performing transaction through the ATM machine.
Postconditions:	Authorized access to the system will be provided to the customer after authentication and validation from the server side.
Frequency of use:	01

Fig. 3.2.3 : Table for use Case Scenario Template for "Login into the ATM Machine"

3.2.2 Actors

- An actor is the external user of the system who is responsible for communication and coordination with the software system.
- It is not always necessary to have a human being as an actor within the use case scenario. We may have any other element or an external system as an actor.
- An actor have different representations in UML as shown in Fig. 3.2.4.

Fig. 3.2.4 : Actor notation in UML

- It is first of all necessary to recognize the role of an actor for better understanding of an actor.
- Actors are always ***external*** to the system.
- For identification of actors of a particular system, following questions should be taken into consideration :
  - Who are the end users of the system?
  - Who are the installers of the system?
  - Who provides information to the system?

- Is there any other cooperating or interacting system available in the scenario?
- Who maintains the system?

- From the business point of view, each actor should be named uniquely.
- Always, there is direct communication and coordination amongst actors and the system.
- We may have time as an actor in case we have to model the things that happen at a particular point of time in the system scenario.

3.2.3 System Boundary (Subject)

- It is essential to define boundaries of the system before moving towards the implementation and execution of the system. That is, software designers should define the things which are external to the system and the things which are the part of the system.
- So, the things which are the part of the system come inside the system boundary which is also known as a subject while the things which are external to the system are located outside the boundary of the system.
- The system boundary is graphically represented by means of a rectangular box, labeled with the name of the system. Since, actors are external to the system, they are drawn outside the system boundary. Use cases are drawn inside the system boundary.

3.2.4 Use case Relationships

- The possible relationships amongst actors and use cases in the use case diagram are as follows :
  - ***Use case/Actor Association*** : This relationship specifies that, the actor initiates the use case.
  - ***Use case Generalization*** : It indicates the generalization relationship between a specific use case and a general use case.
  - ***Actor Generalization*** : It indicates the generalization relationship between a specific actor and a general actor.
  - ***<<include>>*** : One use case can include its behavior from the other use case with the help of this relationship.
  - ***<<extend>>*** : One use case can extend its behavior with one or more other use cases with the help of this relationship.
- Let us have a brief discussion on use case relationships.

3.2.4.1 Use case/Actor Association

- Use case/Actor Association indicates that, the actor initiates a particular use case.
- Normally, one actor can be associated with one or more use cases.
- It actually depicts that, a specific output or result is supplied to the actor.
- An association between a use case and an actor is graphically shown as a solid line between them.

Fig. 3.2.5 : Use case/Actor Association

3.2.4.2 Use case Generalization

GQ. What is use case generalization in use case model? Illustrate with appropriate example.

- Use case generalization indicates the generalization relationship between a specific use case and a general use case.
- It should be used only in the simplification of the use case model.
- A general use case can inherit properties from their individual parent use case as well as change (override) properties.
- A general use case is simply a child use case and it automatically inherits all of the characteristics of its parent use case.
- Fig. 3.2.6 shows the UML notation for use case generalization.

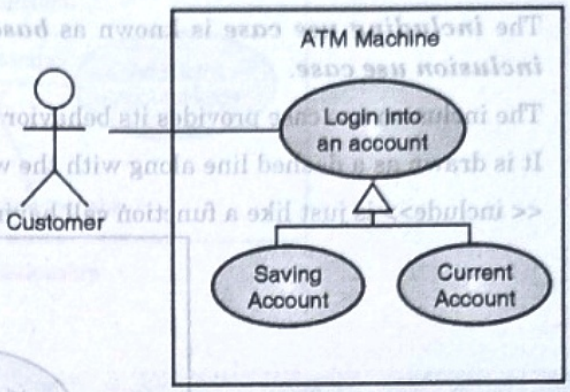

Fig. 3.2.6 : Use case Generalization

3.2.4.3 Actor Generalization

GQ. What is actor generalization in use case model? Illustrate with appropriate example.

- Actor generalization indicates the generalization relationship between a specific actor and a general actor.
- Just like use case generalization, this relationship should be used in the simplification of the use case model.
- Refer Fig. 3.2.7 for UML notation of actor generalization.

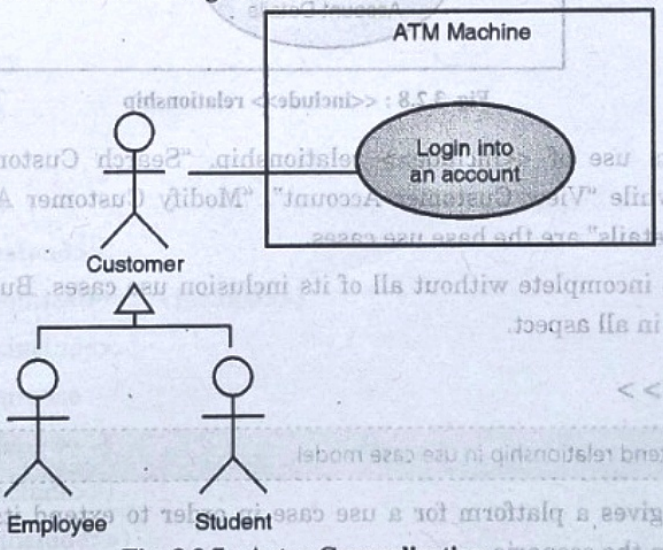

Fig. 3.2.7 : Actor Generalization

3.2.4.4 <<include>>

GQ. Explain the use of include relationship in use case model.

- Include relationship gives us the facility to include the behavior of one use case into the flow of the other use case involved within the scenario.
- The ***including use case*** is known as ***base use case*** while ***included use case*** is considered as the ***inclusion use case***.
- The inclusion use case provides its behavior to the base use case.
- It is drawn as a dashed line along with the word <<include>> on the top of the line.
- << include>> is just like a function call having similar semantics and syntax as that of the function call.

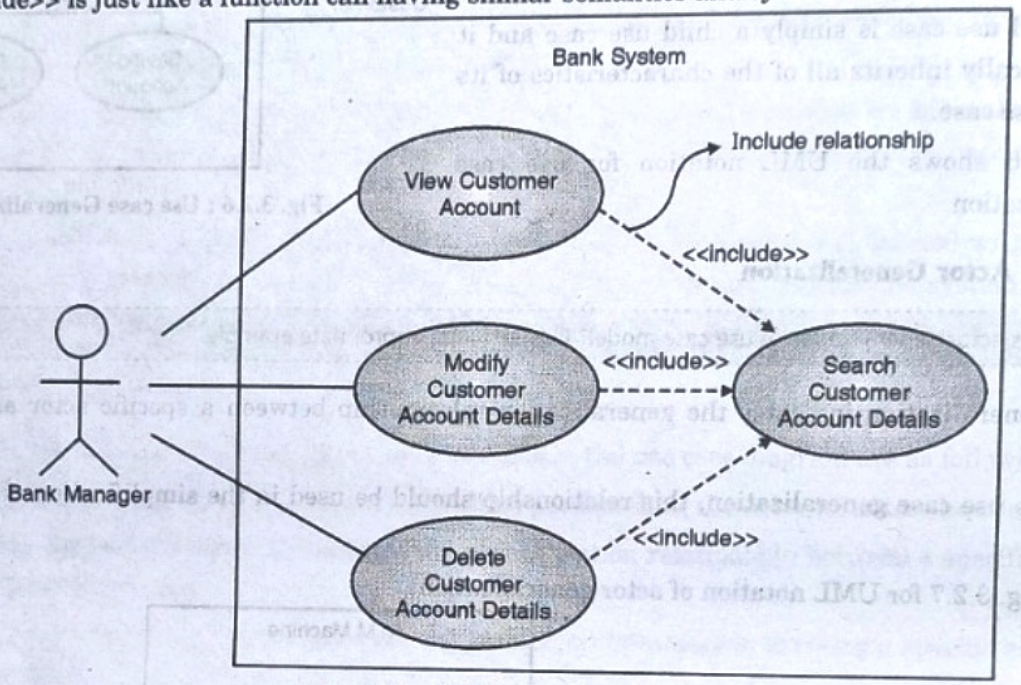

Fig. 3.2.8 : <<include>> relationship

- Fig. 3.2.8 represents use of <<include>> relationship. "Search Customer Account Details" is an inclusion use case while "View Customer Account", "Modify Customer Account Details" and "Delete Customer Account Details" are the base use cases.
- The base use case is incomplete without all of its inclusion use cases. But inclusion use cases may or may not be complete in all aspect.

3.2.4.5 <<extend>>

GQ. Explain the use of extend relationship in use case model.

- Extend relationship gives a platform for a use case in order to extend its behavior with one or more other use cases within the scenario.
- New behavior can be inserted into the existing use case with the help of <<extend>> relationship.
- The <<extend>> relationship should specify one or more extension points in the base use case.

- If certain specified conditions are satisfied, we can make use of this relationship to add some additional functionality in the base use cases.
- Fig. 3.2.9 shows <<extend>> relationship between "Open Bank Account" and "Add Joint Account Holder" use cases.

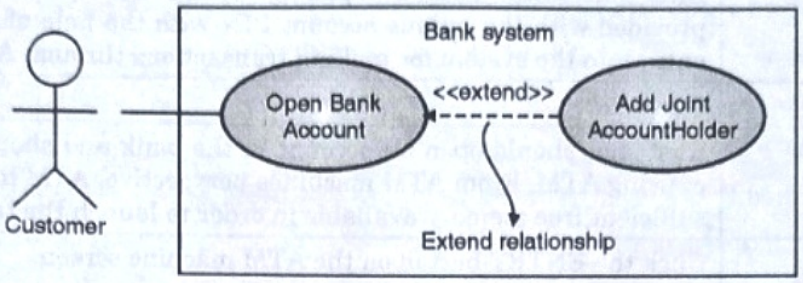

Fig. 3.2.9 : <<extend>> relationship

3.2.5 Use Case Diagram Case Study : ATM Machine

GQ. Draw and explain use case diagram for ATM System. Explain at least two use cases with the help of use case scenario.

- For ATM machine, following are the necessary actors and their respective use cases.
- Actors involved in the ATM scenario are :
  - Customer
  - Administrator (Third Party)
  - Bank
- Use cases involved in the ATM scenario are :
  - Login
  - Transaction
    - Balance Enquiry
    - Transfer Funds
    - Deposit Funds
    - Withdraw Cash
    - ATM Help (<<extend>>)
    - Customer Authentication (<<include>>)
    - Invalid PIN (<<include>>)
  - ATM Machine Maintenance
    - Upgrades (<<include>>)
    - Reporting (<<include>>)
    - Shut Down (<<include>>)
    - Diagnostics (<<include>>)
  - ATM Machine Repair
    - Diagnostics (<<include>>)

• Overall use case scenarios are described below for better understanding :

Use case ID:	01
Use case name:	Login.
Use case description:	After successful account opening within the system, the user will be provided with the unique account PIN with the help of which, user can get entry into the system for making transactions through ATM machine.
Actors:	Customer
Preconditions:	Customer should open an account in the bank and should avail the facility of using ATM. From ATM machines perspective, ATM machine should have sufficient free memory available in order to launch the task.
Main sequence description:	Click the ENTRY button on the ATM machine screen. Enter PIN number for getting logged in into the system. After authentication and validation from server side, an authorized access will be given to the customer.
Alternative sequence description:	NIL
Nonfunctional requirements:	Unique Account PIN should be provided to the customer which will be used by the customer while performing transaction through the ATM machine.
Postconditions:	Authorized access to the system will be provided to the customer after authentication and validation from the server side.
Frequency of use:	01

Fig. 3.2.10 : Use case Scenario – Login

Use case ID:	02
Use case name:	Transaction.
Use case description:	After successful login into the system, the user will be provided with the menu displayed on the ATM Machine screen for making transactions through ATM machine.
Actors:	Customer
Preconditions:	Customer should enter valid PIN for getting authorized access to the system.
Main sequence description:	Customer should select correct choice from the available options in the transaction menu. Furthermore, customer should fill correct necessary details related to transaction.
Alternative sequence description:	NIL
Postconditions:	Authorized access to the system will be provided to the customer after authentication and validation from the server side. Transactions can be of four types: Balance enquiry, Transfer funds, Deposit funds and Withdraw money. On the basis of the choice selected by a customer, required transaction will be successfully executed and customer will be provided with expected result. (For example; in case of money withdrawal, customer will be provided with amount and printed receipt of transaction.)
Frequency of use:	01

Fig. 3.2.11 : Use case Scenario - Transaction

- Fig. 3.2.12 depicts detailed use case diagram for an ATM machine.

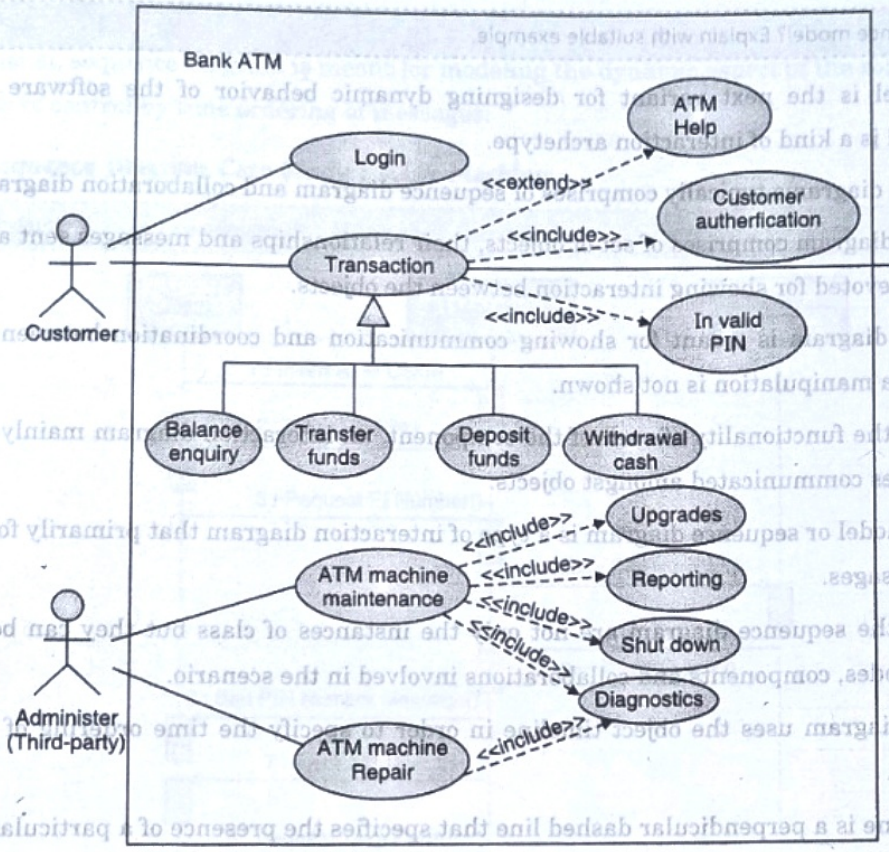

Fig. 3.2.12 : Use case diagram for ATM Machine

Guidelines for designing a Use Case Diagram

- All use cases involved in the scenario should be named uniquely and they should unitedly depict the overall behavior of the system.
- For better understanding, only important use cases should be shown.
- Only related actors should be shown in the diagram.
- System name should be given in such a way that, it should specify its purpose.
- Arrange the actors and use cases in the diagram in some proper sequence so that, roles of the actors and their behavior can be represented in right manner and it will also help to avoid lines that cross.
- Try to avoid showing too many relationships between use cases and respective actors.

3.3 Sequence Model and Procedural Sequence Model

GQ. What is sequence model? Explain with suitable example.

- Sequence model is the next variant for designing dynamic behavior of the software system. The sequence model is a kind of interaction archetype.
- The interaction diagrams typically comprises of sequence diagram and collaboration diagram.
- An interaction diagram comprises of set of objects, their relationships and messages sent amongst them and is mainly devoted for showing interaction between the objects.
- An interaction diagram is meant for showing communication and coordination between two or more objects but, data manipulation is not shown.
- To understand the functionality of each of the component, an interaction diagram mainly emphases on specific messages communicated amongst objects.
- The sequence model or sequence diagram is a type of interaction diagram that primarily focuses on time ordering of messages.
- The objects in the sequence diagram are not only the instances of class but they can be instances of elements like nodes, components and collaborations involved in the scenario.
- The sequence diagram uses the object timeline in order to specify the time ordering of the messages between objects.
- An object timeline is a perpendicular dashed line that specifies the presence of a particular object over a particular period of time for which an object is active in the system execution.
- All of the objects are arranges at the top of the diagram horizontally.
- The lifeline of each object involved is drawn from top to bottom of the diagram.
- Focus of control is another important element of a sequence diagram which is graphically represented as a rectangle that shows the time period during which an object is active or performing inside the system.
- As far as procedural sequence model is concerned, we should categorize objects into active objects and passive objects. Active objects have their own focus of control and are always activated.
- Also, we can depict conditions in a procedural sequence model.
- Implementation details of the proposed software system should be given in procedural sequence model for better understanding purpose.
- Also, in case of a procedural sequence model; active objects and passive objects should be distinguished; since, active objects are having their solitary focus of control and they are active in nature; while, passive objects are passive and they don't have their own focus of control.

- Other they additional features should be added as per the typical requirements of the proposed software system scenario.
- In conclusion, sequence diagram is meant for modeling the dynamic aspect of the software system and it gives flow of control by time ordering of messages.

3.3.1 Sequence Diagram Case Study : ATM Machine

GQ. Draw a detailed sequence diagram for ATM system scenario.

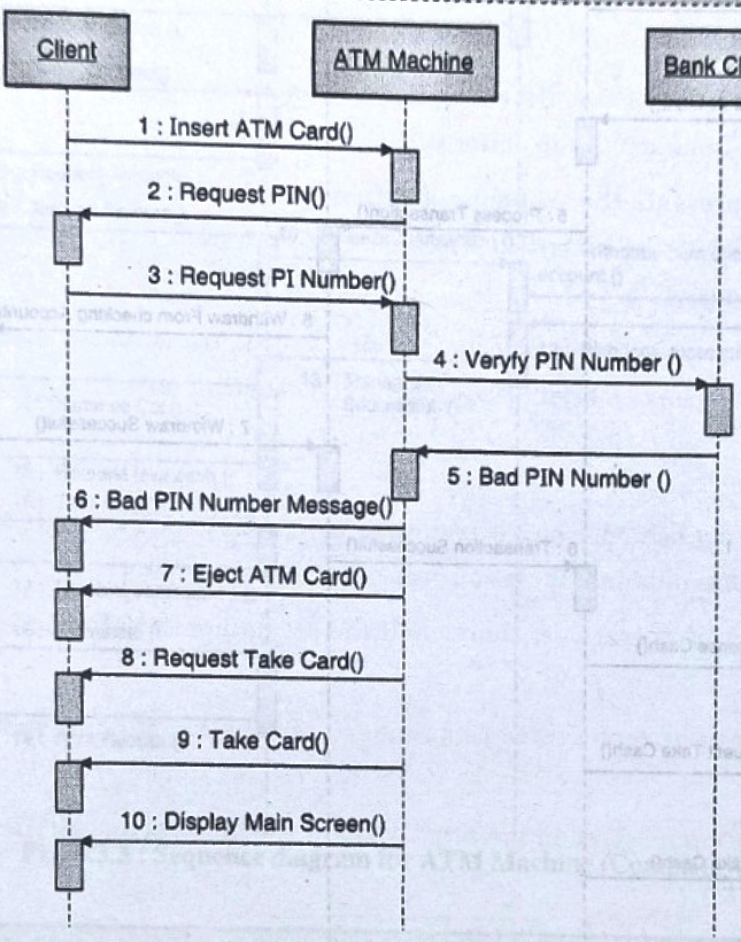

Fig. 3.3.1 : Sequence diagram for ATM Machine - Entering Invalid PIN

NOTES

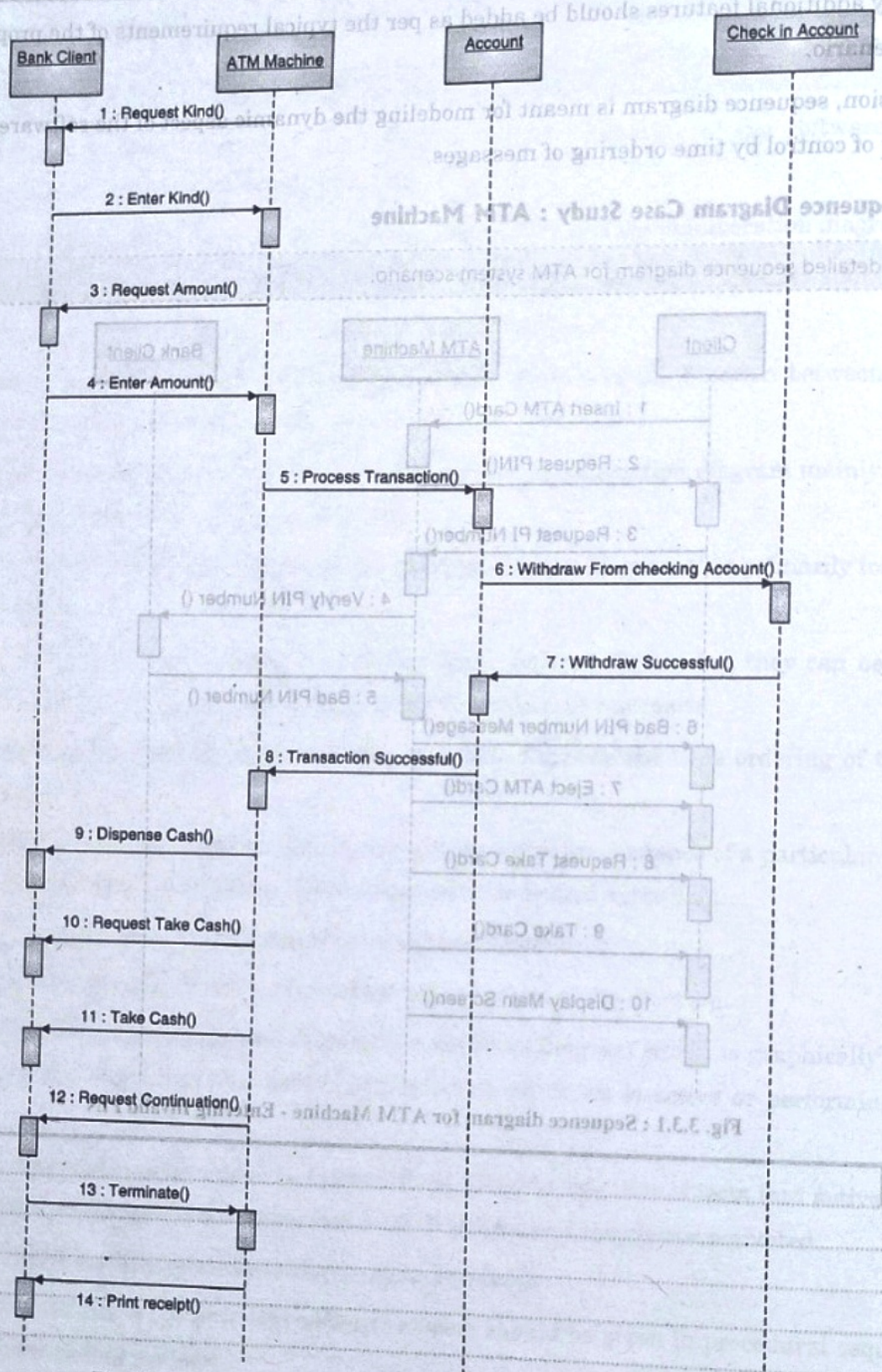

Fig. 3.3.2 : Sequence diagram for ATM Machine - Cash Withdrawal

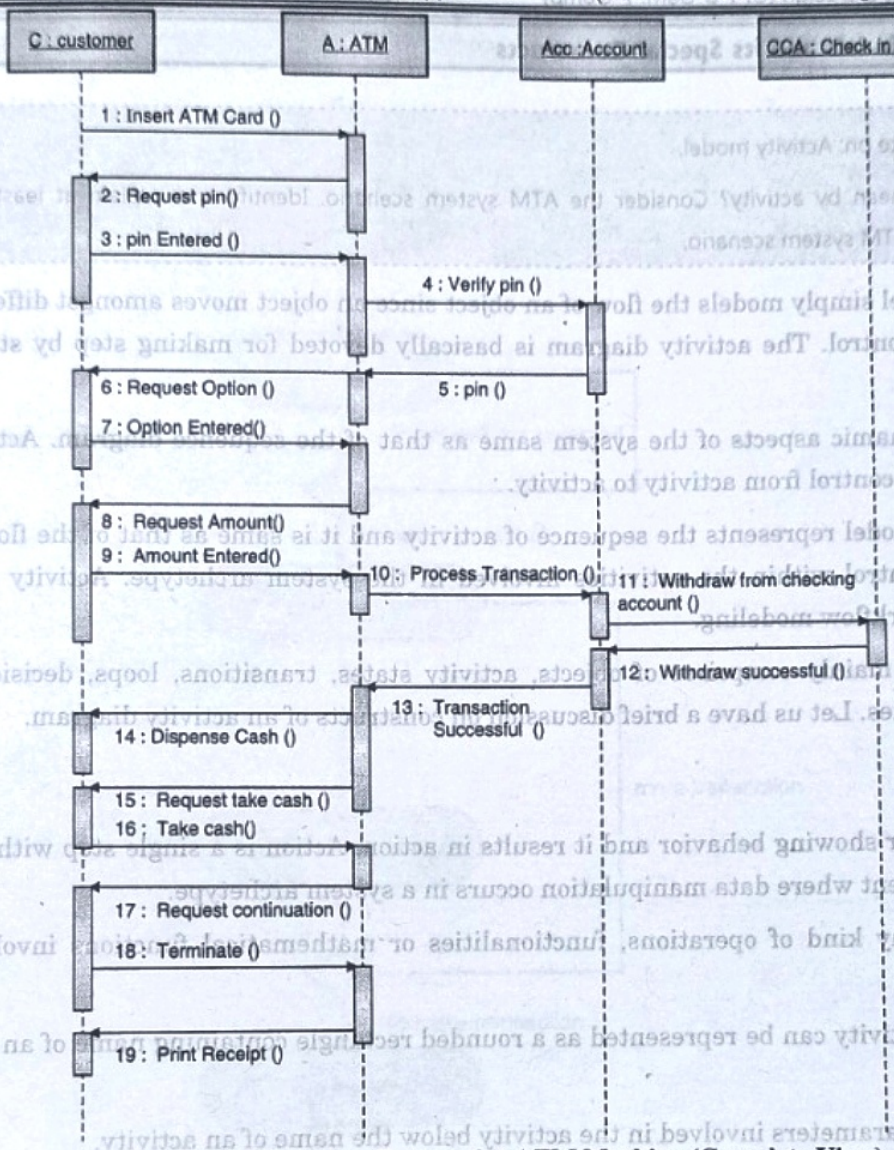

Fig. 3.3.3 : Sequence diagram for ATM Machine (Complete View)

Guidelines for designing a Sequence Diagram

- A sequence diagram should be used when we have to depict the flow of control within a system by time ordering of messages amongst objects.
- A sequence diagram should contain only important objects, nodes, components, collaborations or actors for better understanding.
- The lifeline for each of the object that is actively involved in the system execution should be set.
- Packages should be use in case of a large collection of sequence diagrams and each sequence diagram in this case should be named uniquely in order to differentiate it from other sequence diagrams.

3.4 Activity Models and its Special Constructs

GQ. Write a short note on: Activity model.

GQ. What do you mean by activity? Consider the ATM system scenario. Identify and explain at least five activities involved in the ATM system scenario.

- The activity model simply models the flow of an object since an object moves amongst different states in the flow of the control. The activity diagram is basically devoted for making step by step executable systems.
- It models the dynamic aspects of the system same as that of the sequence diagram. Activity diagram focuses on flow of control from activity to activity.
- Mainly activity model represents the sequence of activity and it is same as that of the flowchart which shows flow of control within the activities involved in the system archetype. Activity diagrams are widely used in workflow modeling.
- Activity diagram mainly comprises of objects, activity states, transitions, loops, decision nodes and concurrent activities. Let us have a brief discussion on constructs of an activity diagram.

3.4.1 Activity

- Activity is used for showing behavior and it results in action. Action is a single step within an activity and is the component where data manipulation occurs in a system archetype.
- Actions can be any kind of operations, functionalities or mathematical functions involved within a system execution.
- Graphically, an activity can be represented as a rounded rectangle containing name of an activity in its body.
- We can show the parameters involved in the activity below the name of an activity.
- For instance refer Fig. 3.4.1.

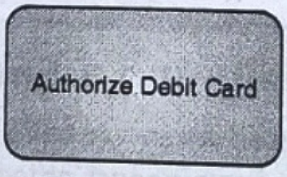

Fig. 3.4.1 : Simple Activity

NOTES

3.4.2 Activity Diagram Case Study : ATM Machine

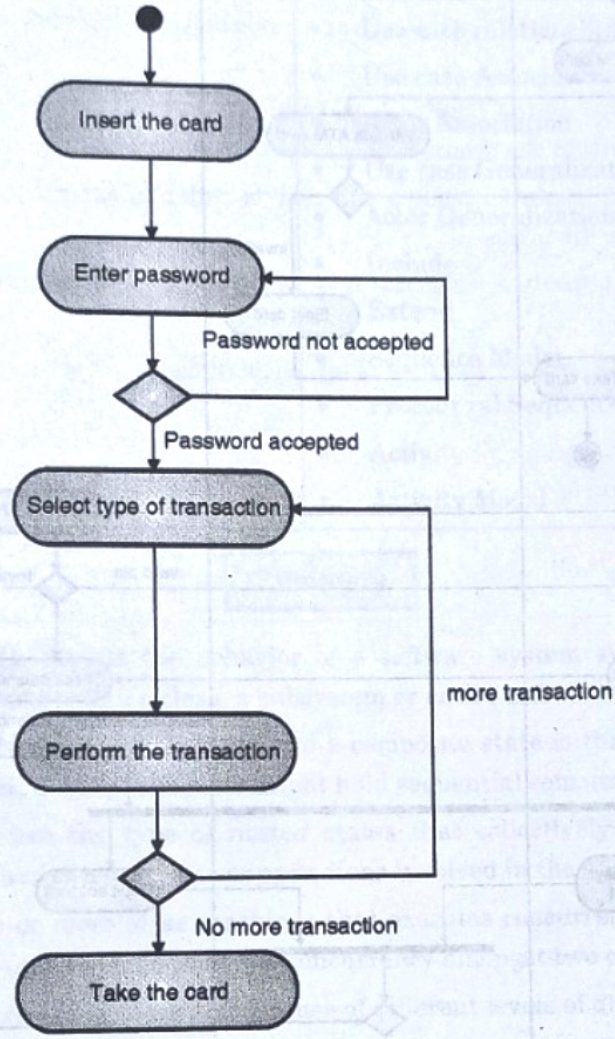

Fig. 3.4.2 : Simple Activity diagram for ATM Machine

Guidelines for designing Activity Diagram

- Activity diagram is intended for representing dynamic behavior of a system.
- For better understanding, only essential components should be taken into account and should be shown in the diagram.
- Name of activity diagram should mention its intention appropriately.
- All activities involved in the software system scenario should be arranged in some sequence.
- Activity modeling should comprise of primary flow, secondary considerations and object flows.

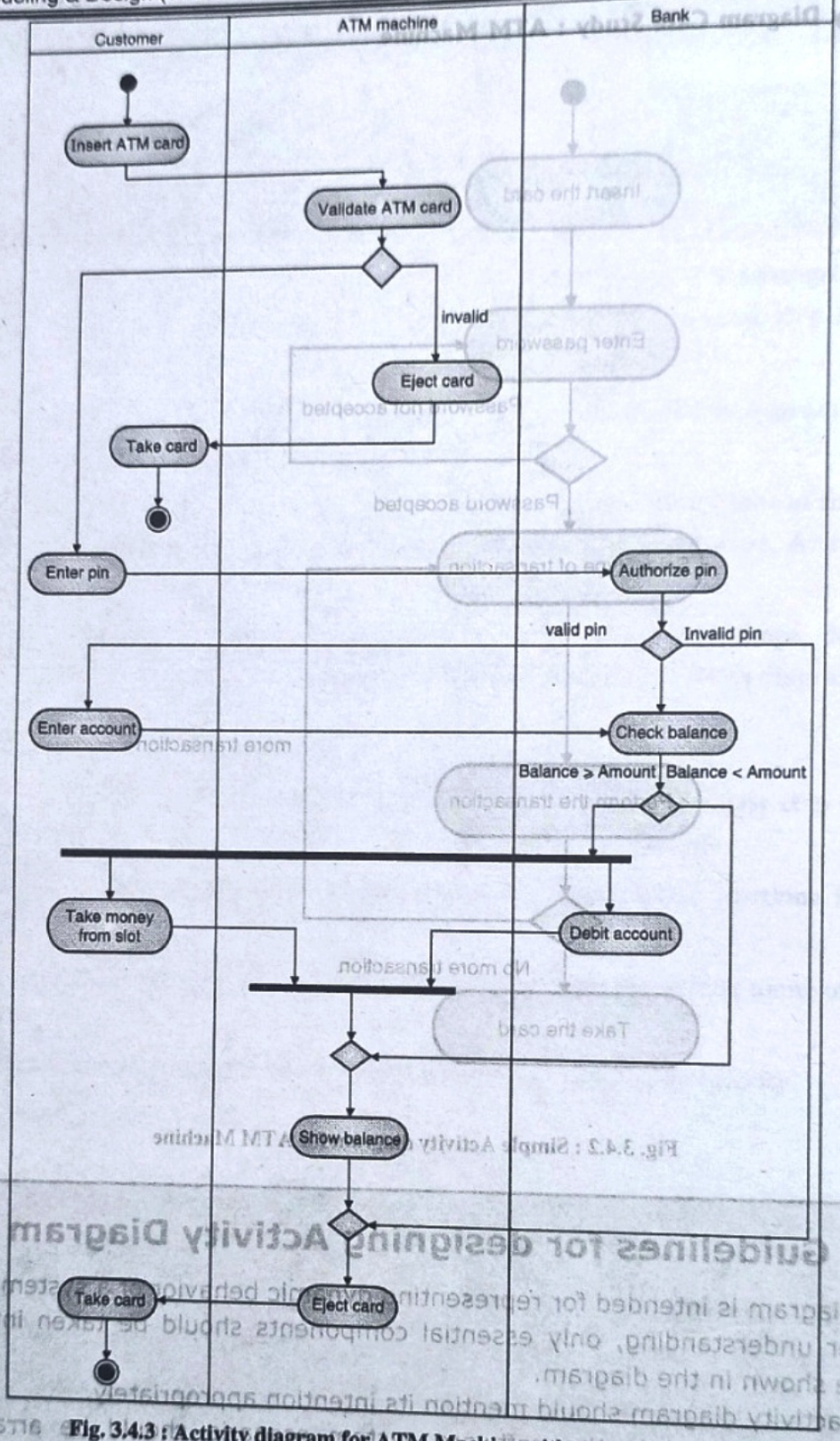

Fig. 3.4.3 : Activity diagram for ATM Machine with special constructs

Key Concepts

- Nested State
- Nested State Diagram
- Sequential substate
- Concurrent substate
- Class model
- State model
- Use case model
- Use case
- Actor
- System Boundary
- Use case relationships
- Use case Association
- Actor Association
- Use case Generalization
- Actor Generalization
- Include
- Extend
- Sequence Model
- Procedural Sequence Model
- Activity
- Activity Model

Summary

- ***State diagram*** in UML depicts the behavior of a software system and can be used to model the behavior of different elements like a class, a subsystem or entire software system.
- The basic difference between a simple state and a composite state is that, there is no substructure in case of a simple state but, a composite state might hold sequential concurrent substates.
- ***Sequential substates*** are the type of nested states that collectively defines certain scenario and depicts a proper flow of activities, actions or operations involved in the scenario.
- If we have to show two or more state machines that executes concurrently, then we can make use of concurrent substates in order to represent the concurrency amongst two or more state machines.
- The ***signal generalization*** mechanism offers use of different levels of abstraction in the state model.
- For achieving ***concurrency*** in state model, each and every object should share their features, operations and constraints with other objects in the system.
- The ***use case model*** depicts functional requirement of the proposed system by means of the use cases and the actors.
- ***Use case model*** supports in the design of the software system by presenting intended behavior of the proposed system without any description about the implementation of the system.
- ***Use cases*** are meant for specification of the interaction between the system itself and end users of the system which are termed as actors in UML.
- The set of activities and events in some proper sequence specifying the interaction amongst a system and its end users is known as a ***scenario***.
- An ***actor*** is the external user of the system who is responsible for communication and coordination with the software system.
- Actors are always ***external*** to the system in use case diagram.

- ***Use case generalization*** indicates the generalization relationship between a specific use case and a general use case.
- ***Actor generalization*** indicates the generalization relationship between a specific actor and a general actor.
- ***Include*** relationship gives us the facility to include the behavior of one use case into the flow of the other use case involved within the scenario.
- ***Extend*** relationship gives a platform for a use case in order to extend its behavior with one or more other use cases within the scenario.
- The ***sequence model*** or ***sequence diagram*** is a type of interaction diagram that primarily focuses on time ordering of messages.
- The ***activity model*** simply models the flow of an object since an object moves amongst different states in the flow of the control.
- ***Activity*** is used for showing behavior and it results in action.
- ***Action*** is a single step within an activity and is the component where data manipulation occurs in a system archetype.

Chapter Ends...

❑❑❑