OBJECT ORIENTED ANALYSIS AND DESIGN

UNIT-II: Object-Oriented Methodologies- Rumbaugh Object modeling technique-The Booch methodology-The Jacobson methodologies- The Unified Approach- UML- Class Diagrams-Dynamic Modeling.

TEXT BOOK

Ali Bahrami, “Object Oriented Systems Development”, TATA McGraw-Hill Edition, 2008.

PREPARED BY

Dr. D. Devakumari

Content

→Object-Oriented Methodologies

→Rumbaugh Object modeling technique

→The Booch methodology

→The Jacobson methodologies

→ The Unified Approach

→UML

→Class Diagrams

→Dynamic Modeling.

OBJECT-ORIENTED METHODOLOGIES

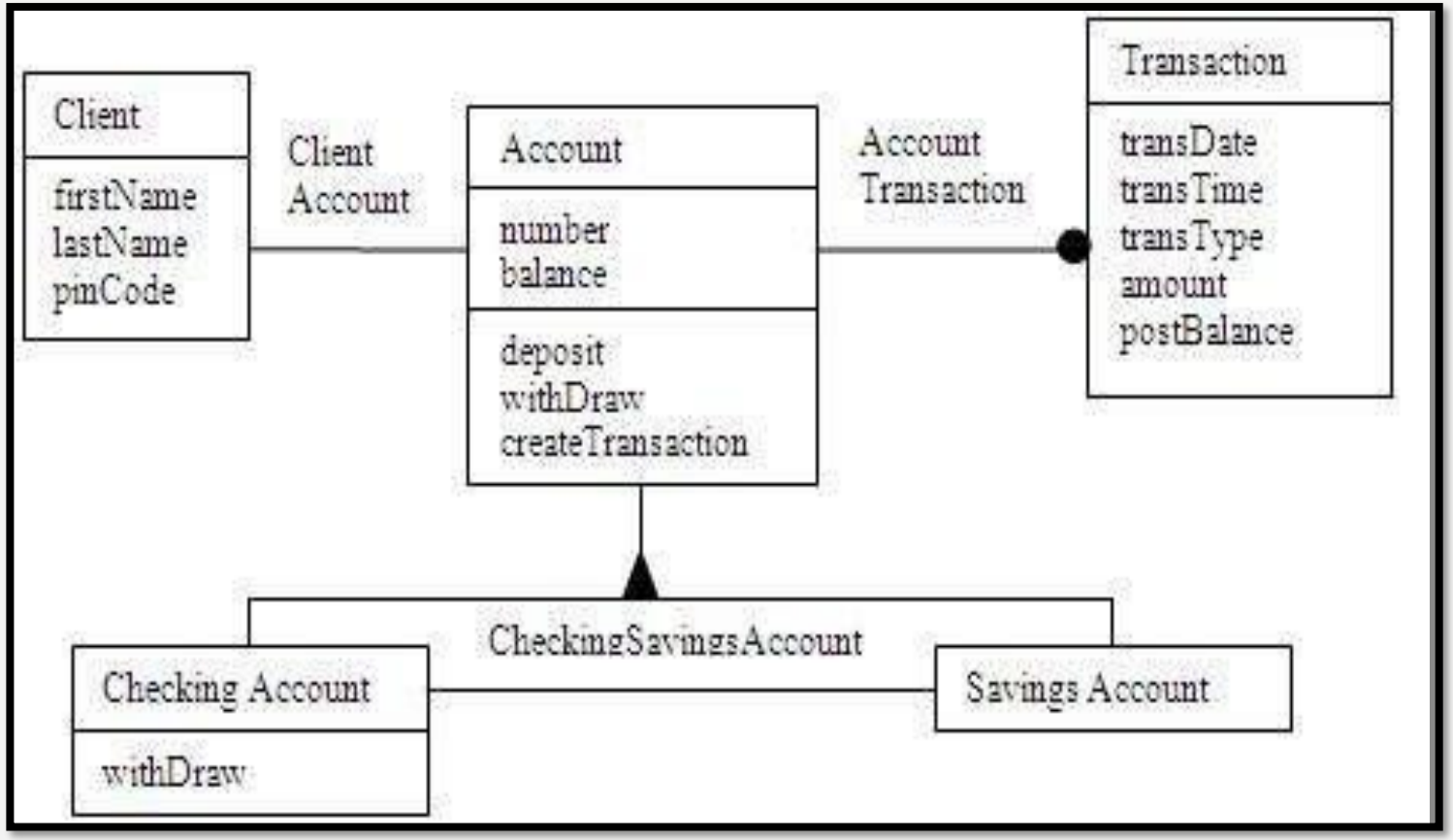
- The Rumbaugh et al. OMT.
- The Booch Methodology.
- Jacobson's Methodologies.
- Unified approach (UA).

RUMBAUGH OBJECT MODELING TECHNIQUE

- Four phases of OMT (can be performed iteratively)
  - **Analysis**: objects, dynamic and functional models
  - **System Design**: Basic architecture of the system.
  - **Object Design**: static, dynamic and functional models of objects.
  - **Implementation**: reusable, extendible and robust code.
- Three different parts of OMT modelling
  - An object model - object model & data dictionary
  - A dynamic model - state diagrams & event flow diagrams
  - A functional model - data flow & constraints

Object Model

- Classes interconnected by association lines
- Classes- a set of individual objects
- Association lines- relationship among classes (i.e., objects of one class to objects of another class)

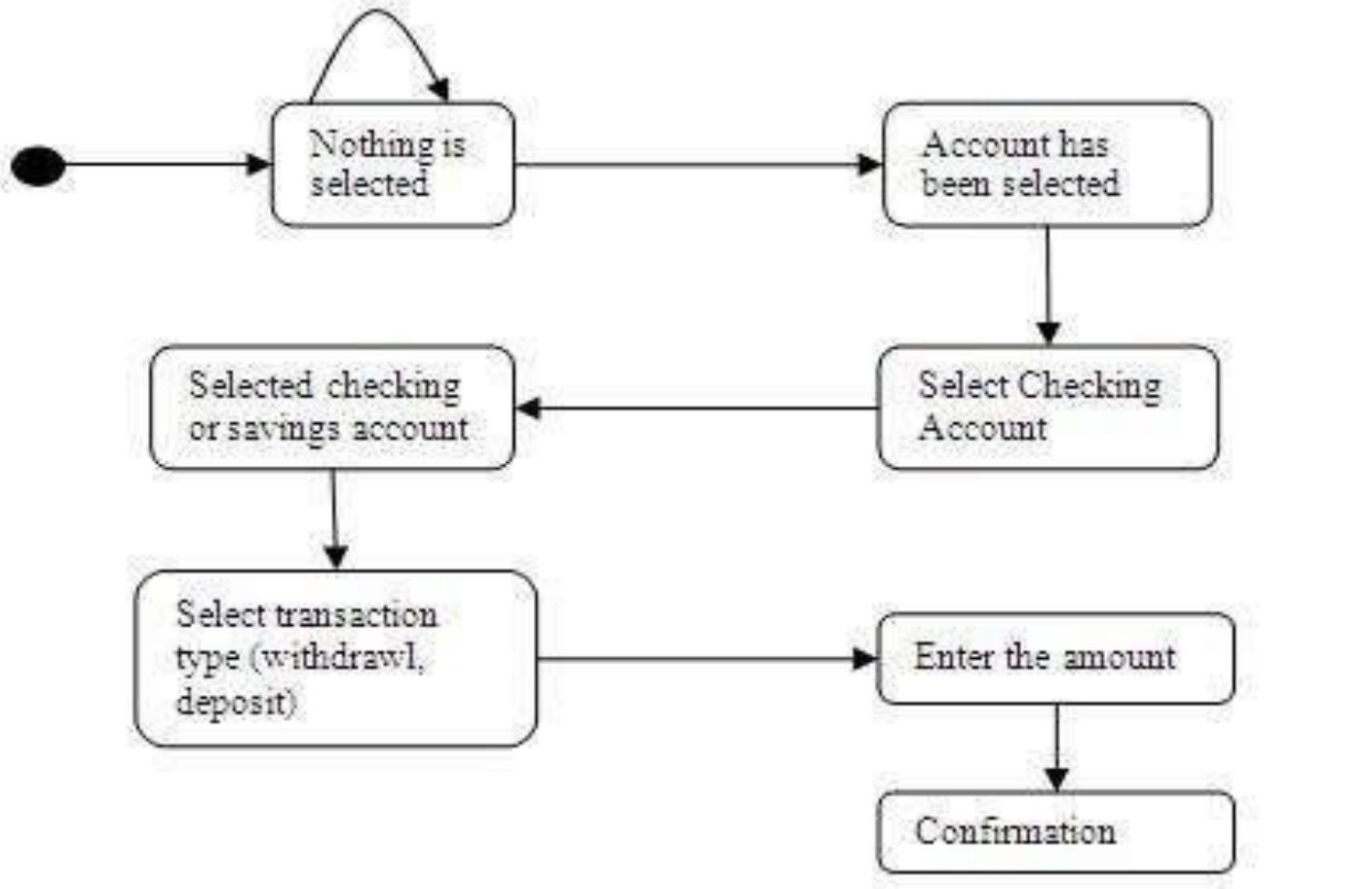


The OMT object model of a bank system. The boxes represent classes and the filled triangle represents specialization.

Association between Account and Transaction is one to many; since one account can have many transactions, the filled circle represents many (zero or more). The relationship between Client and Account classes is one to one: A client can have only one account and account can belong to only one person (in this model joint accounts are not allowed).

OMT Dynamic Model

- States, transitions, events and actions
- OMT state transition diagram-network of states and events



State transition diagram for the bank application user interface. The round boxes represent states and the arrows represent transitions.

OMT Functional Model

- DFD- (Data Flow Diagram)
- Shows flow of data between different processes in a business.
- Simple and intuitive method for describing business processes without focusing on the details of computer systems.

Data Flow Diagram

Four primary symbols

- **Process**- any function being performed 
- **Data Flow-** Direction of data element movement 
- **Data Store –** Location where data is stored 
- **External Entity**-Source or Destination of a data element 

THE BOOCH METHODOLOGY

Diagrams of Booch method

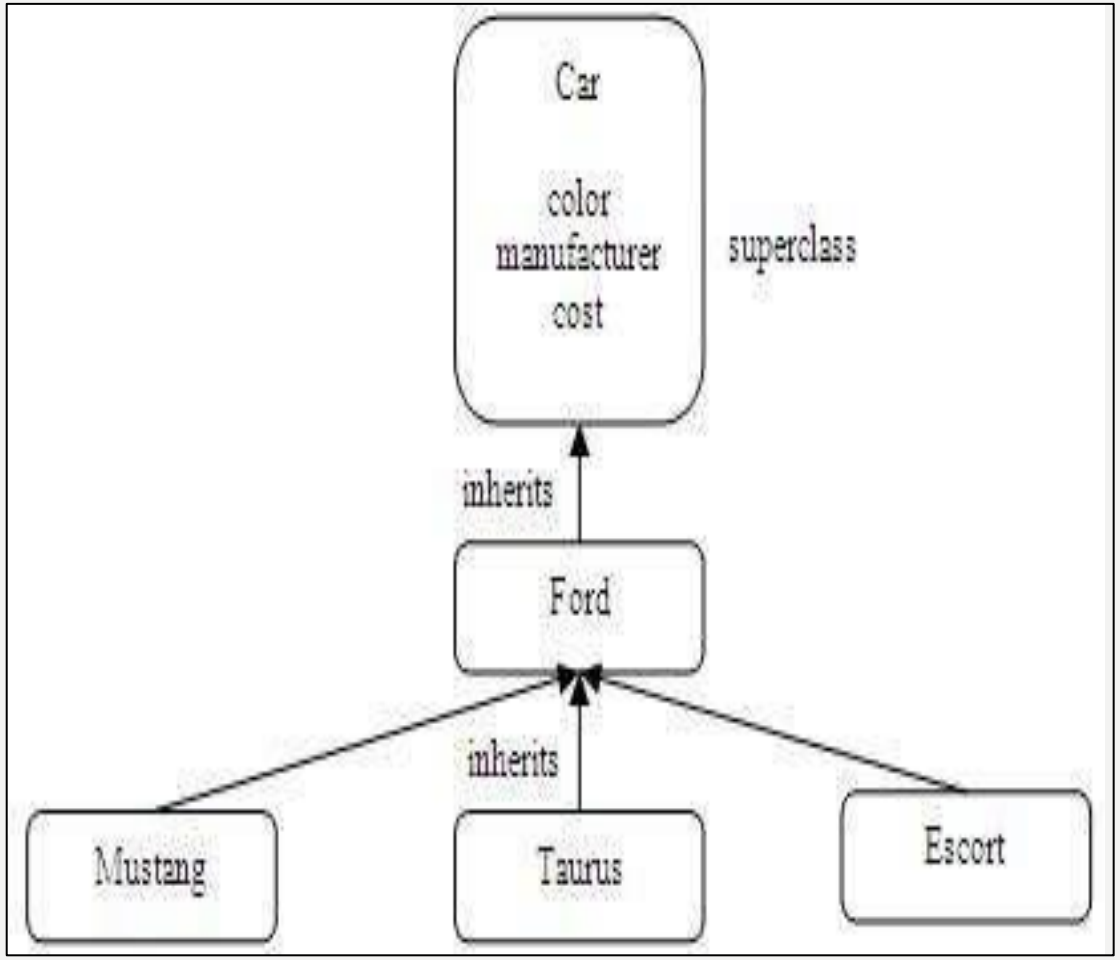
- **Class diagrams describe** roles and responsibilities of objects
- **Object diagrams** describe the desired behaviour of the system in terms of scenarios
- **State transition diagrams** state of a class based on a stimulus
- **Module diagrams** to map out where each class & object should be declared
- **Process diagrams** to determine to which processor to allocate a process
- **Interaction diagrams** describes behaviour of the system in terms of scenarios.

Booch method prescribes:

- Macro Development Process
- Micro Development Process

Macro Development Process

- Steps for macro development process
  1. Conceptualization
  2. Analysis & Development of the model
  3. Design or create the system architecture
  4. Evolution or implementation
  5. Maintenance



Object modeling using Booch notation. The arrows represent specialization; for example, class Taurus is subclass of the class Ford.

Micro Development Process

- Each macro process has its own micro development process

Steps:

- Identify classes & objects

- Identify class & objects semantics

- Identify class & object relationship

- Identify class & objects interface and implementation

THE JACOBSON METHODOLOGIES

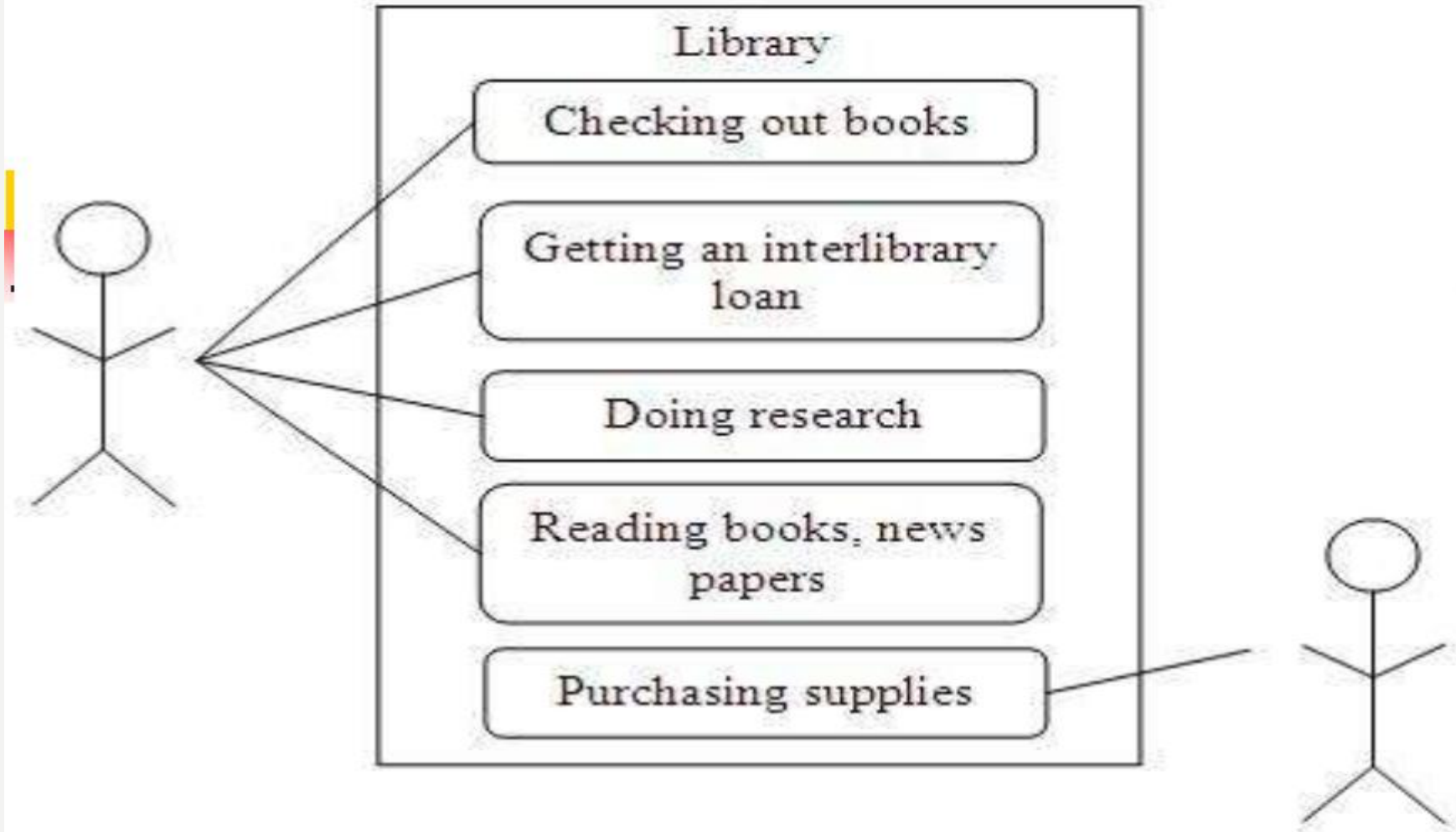
- ***Use Cases.***
- ***Object Oriented Software Engineering.***
- ***Object Oriented Business Engineering.***

Use Cases

- Understanding system requirements
- Interaction between Users and Systems

The use case description must contain

  - How and when the use case begins and ends.
  - The Interaction between the use case and its actors, including when the interaction occurs and what is exchanged.
  - How and when the use case will need data stored in the system.
  - Exception to the flow of events
  - How and when concepts of the problem domain are handled.



Some uses of a library. As you can see, these are external views of the library system from the actor such as a member. The simpler the use case, the more effective it will be. It is unwise to capture all of the details right at the start; you can do that later.

OOSE - Object Oriented Software Engineering.

- Objectory is built around several different models:
  - **Use case model**. The use-case model defines the outside (actors) and inside (use case) of the systems behaviour.
  - **Domain object model**. The objects of the "real" world are mapped into the domain object model.
  - **Analysis object model**. The analysis object model presents how the source code (implementation) should be carried out and written.
  - **Implementation model**. The implementation model represents the implementation of the system.
  - **Test model**. The test model constitutes the test plans, specifications, and reports.

OOBE - Object Oriented Business Engineering

- OOBE is object modeling at the enterprise level.
  - Analysis phase
  - Design and Implementation phase
  - Testing phase
    - E.g. Unit testing, integration and system testing.

UML – UNIFIED MODELING LANGUAGE

- A model is an abstract representation of a system , constructed to understanding the system prior to building or modifying it.

Static Model

- A static model can be viewed as a snapshot of a system's parameters at rest or at a specific point in time.

Dynamic Model

- A dynamic model, in contrast to a static model, can be viewed as a collection of procedures or behaviours that, taken together. Reflect the behaviour of a system over time.

INTRODUCTION TO THE UNIFIED MODELLING LANGUAGE

- The unified modelling language is a language for specifying, constructing, visualizing, and documenting the software system and its components.
- The UML is a graphical language with sets of rules and semantics.
- The rules and semantics of a model are expressed in English, in a form known as object constraint language (OCL).
- OCL is a specification language that uses simple logic for specifying the properties of a system.

The primary goals in the design of the UML were as follows:

1. Provide users a ready-to-use, expressive visual modelling to language so they can develop and exchange meaning models.
2. Provide extensibility and specialization mechanism to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basic for understanding the modelling language.
5. Encourage the growth of the OO tools market.
6. Support higher-level development concepts.
7. Integrate best practices and methodologies.

UML DIAGRAMS

The UML define nine graphical diagrams:

1. Class diagram (Static)
2. Use –Case Diagram
3. Behaviour diagram (dynamic):
    1. Interaction diagram
        1. Sequence diagram
        2. Collaboration diagram
    2. State chart diagram
    3. Activity diagram
4. Implementation Diagram:
    1. Component Diagram
    2. Deployment diagram

UML - CLASS DIAGRAMS

- The UML class diagram, also referred to as object modelling is the main static analysis diagram.
- A class diagram is a collection of static modelling elements, such as classes and their relationships, connected as a graph to each other and to their contents : for Example the things that exit (such as classes), their internal structures, and their relationship to other classes.

1. **Class Notation: Static Structure**
2. **Object Diagram:** Class diagrams can contain objects, so a class diagram with objects and no classes is an object diagram.
3. **Class Interface Notation**
4. **Binary association Notation**
5. **Association Role**
6. **Qualifier**
7. **Multiplicity**
8. **OR Association**
9. **Association Class**
10. **N-Ary Association**
11. **Aggregation and composition (a-part-of)**
12. **Generalization**

CLASS NOTATION: STATIC STRUCTURE

- A class is drawn as a rectangle with three components separated by horizontal lines. The top name compartment holds the class name, other general properties of the class, such as attributes, are in the middle compartment, and the bottom compartment holds a list of operations.

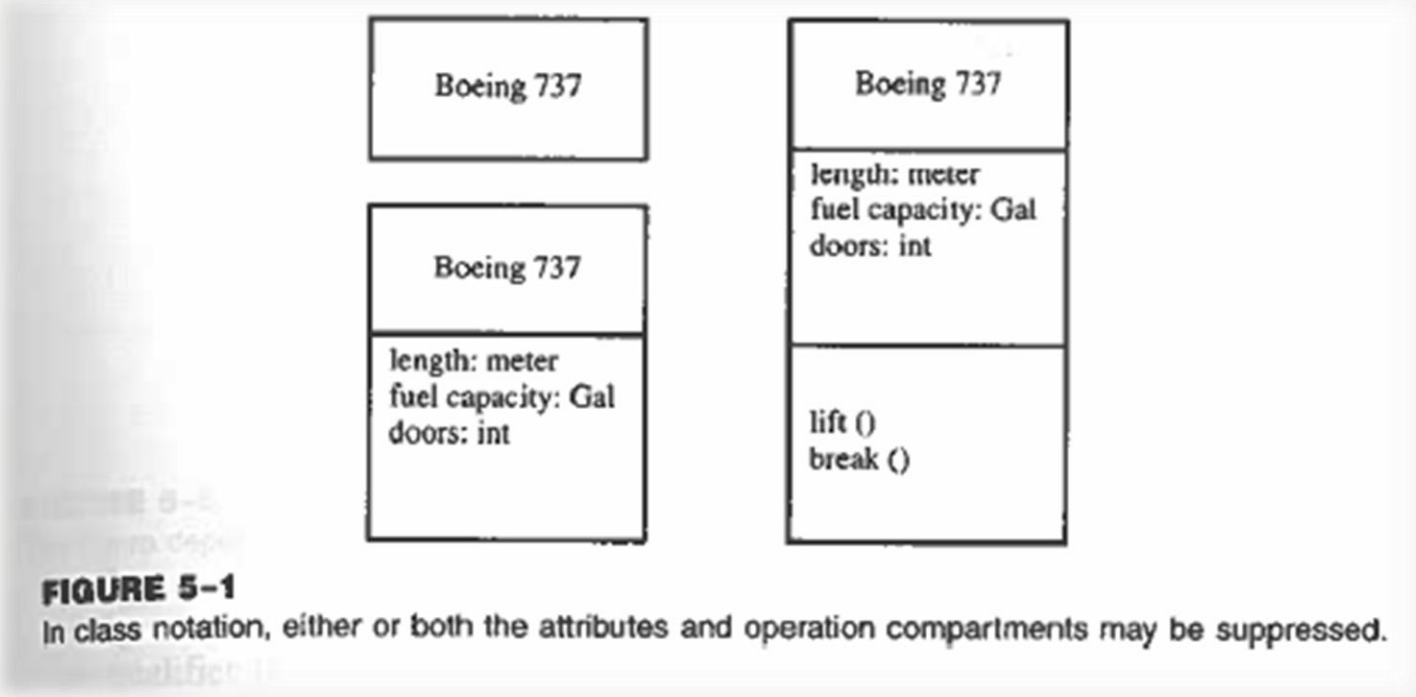


FIGURE 5–1
In class notation, either or both the attributes and operation compartments may be suppressed.

- Either or both the attribute and operation compartments may be suppressed.

CLASS INTERFACE NOTATION

- The UML notation for an interface is a small circle with the name of the interface connected to the class. (fig: 5.2)

- A class that requires the operations in the interface may be attached to the circle by a dashed arrow.

BINARY ASSOCIATION NOTATION

- A binary association is drawn as a solid path connecting two classes, or both ends may be connected to the same class.

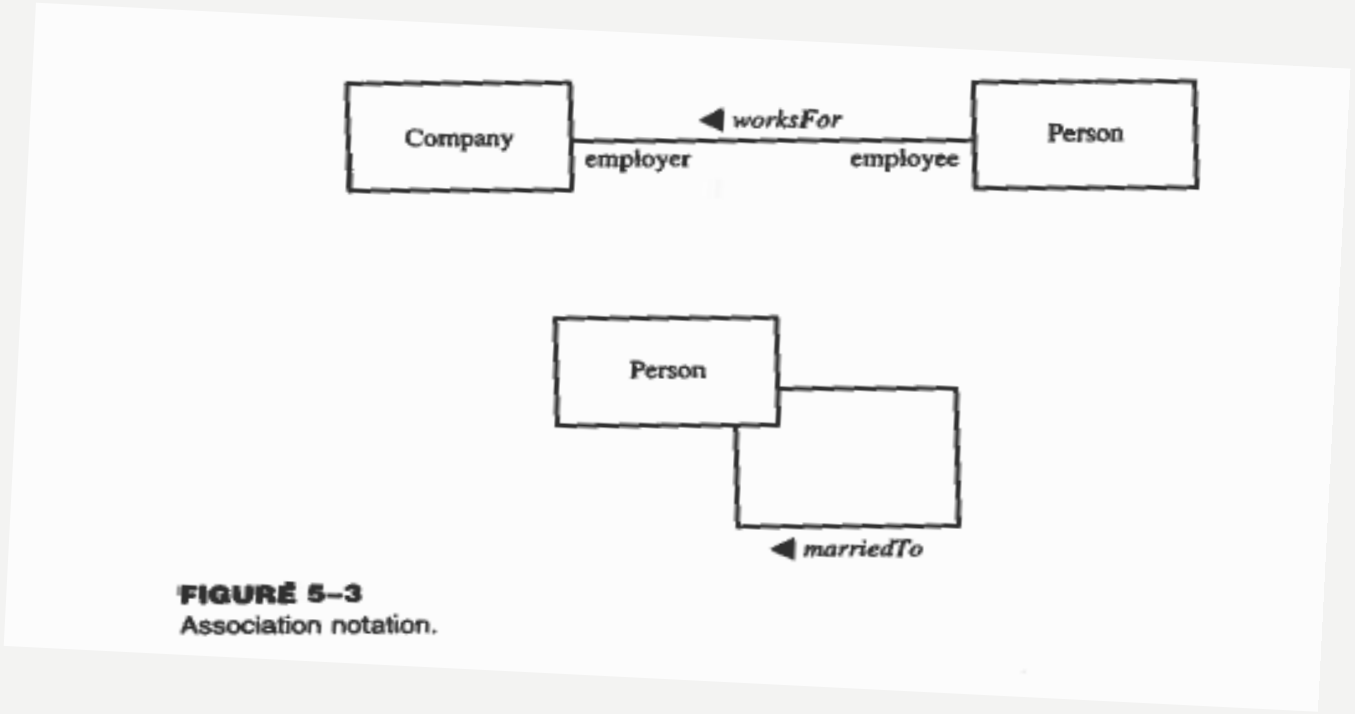


FIGURE 5–3
Association notation.

- An association may have an association name. Furthermore, the association name may have an optional black triangle in it, the point of the triangle indicating the direction in which to read the name. The end of an association, where it connects to a class, is called the association role.

ASSOCIATION ROLE

- An arrow may be attached to the end of the path to indicate that navigation is supported in the direction of the class pointed to.
- An arrow may be attached to neither, one, or both ends of the path.
- In particular, arrow could be shown whenever navigation is supported in a given direction.
- In UML, association is represented by an open arrow, as represented.

FIGURE 5–4
Association notation.

QUALIFIER

- A qualifier is an association attribute. For example, a person object may be associated to a bank object. An attribute of this association is the account#.
- The account# is the qualifier of this association.

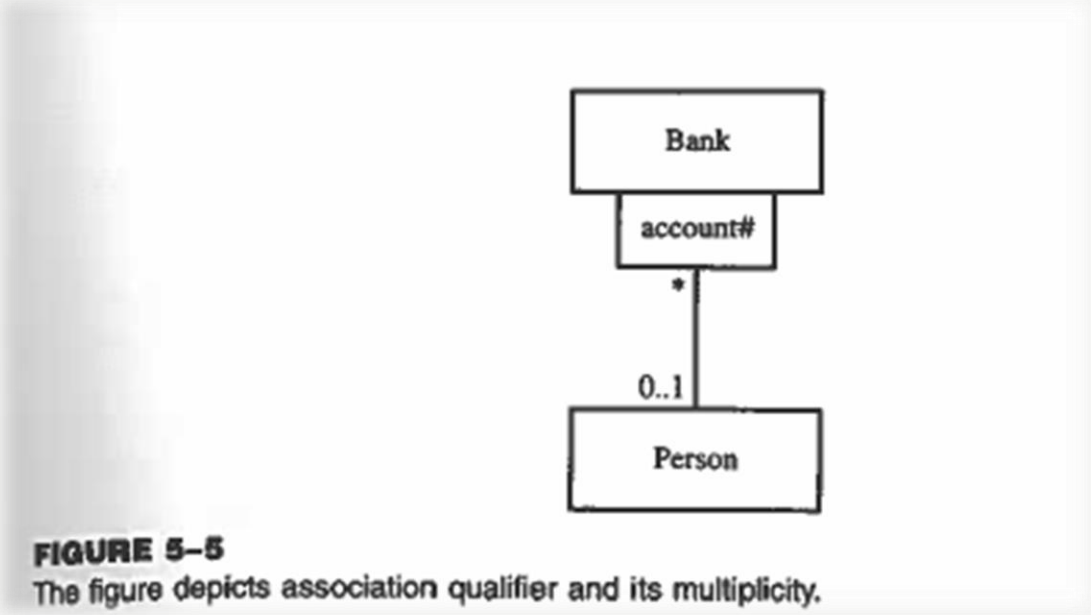


FIGURE 5–5
The figure depicts association qualifier and its multiplicity.

- A qualifier is shown as a small rectangle attached to the end of an association path, between the final path segment and the symbol of the class to which it connects.

MULTIPLICITY

- Multiplicity specifies the range of allowable associated classes.
- Lower bound.. upper bound.

0...1

0..*

1..3, 7..10, 15, 19..*

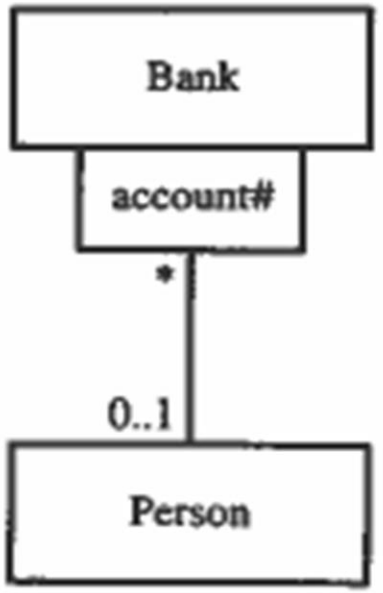


FIGURE 5–5
The figure depicts association qualifier and its multiplicity.

OR ASSOCIATION

- An OR association indicates a situation in which only one of several potential associations may be instantiated at one time for any single object.
- This is shown as a dashed line connecting two or more associations.

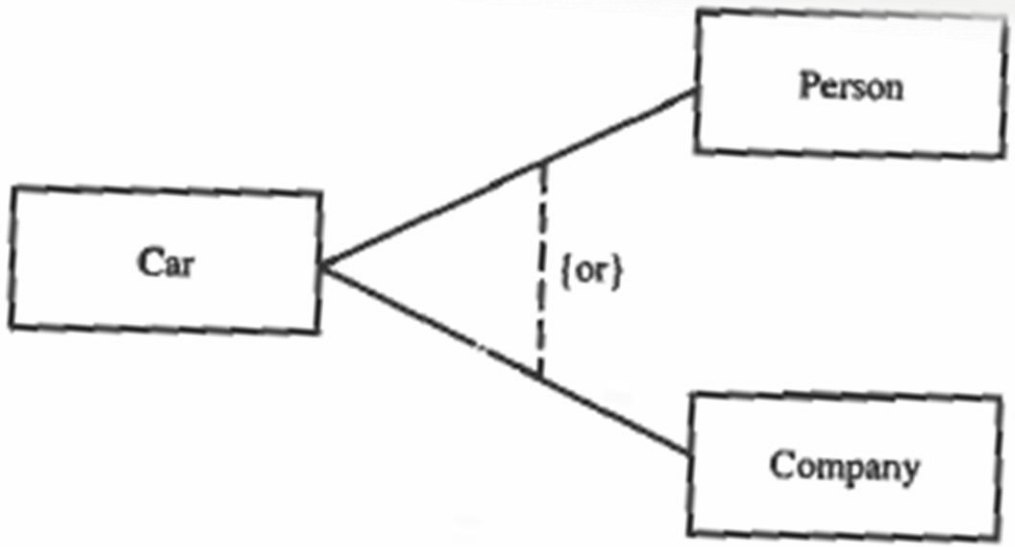
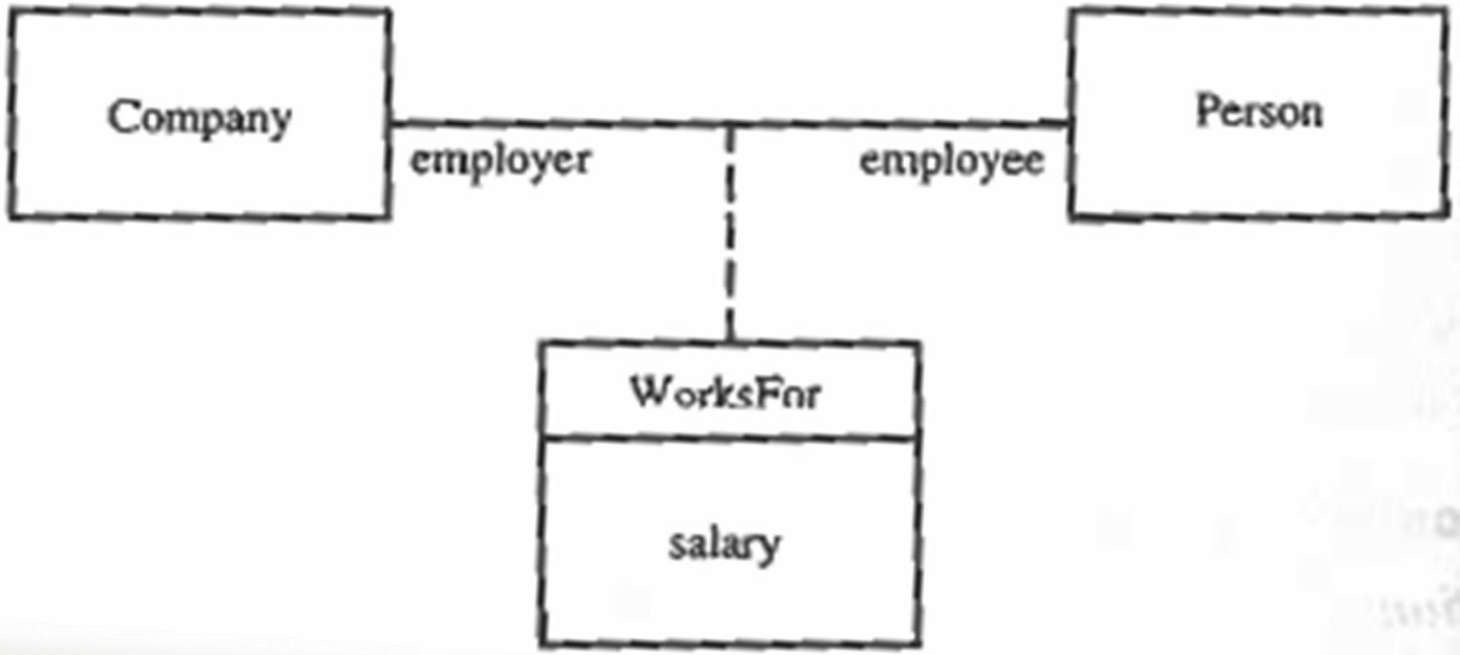


FIGURE 5–6
An OR association notation. A car may associate with a person or a company.

ASSOCIATION CLASS

- An association path. The name in the class symbol and the name string attached to the association path are the same.

FIGURE 5-7
Association class.



N-ARY ASSOCIATION

- An n-ary association is an association among more than two classes.
- An n-ary association is shown as a large diamond with a path form the diamond to each participant class.

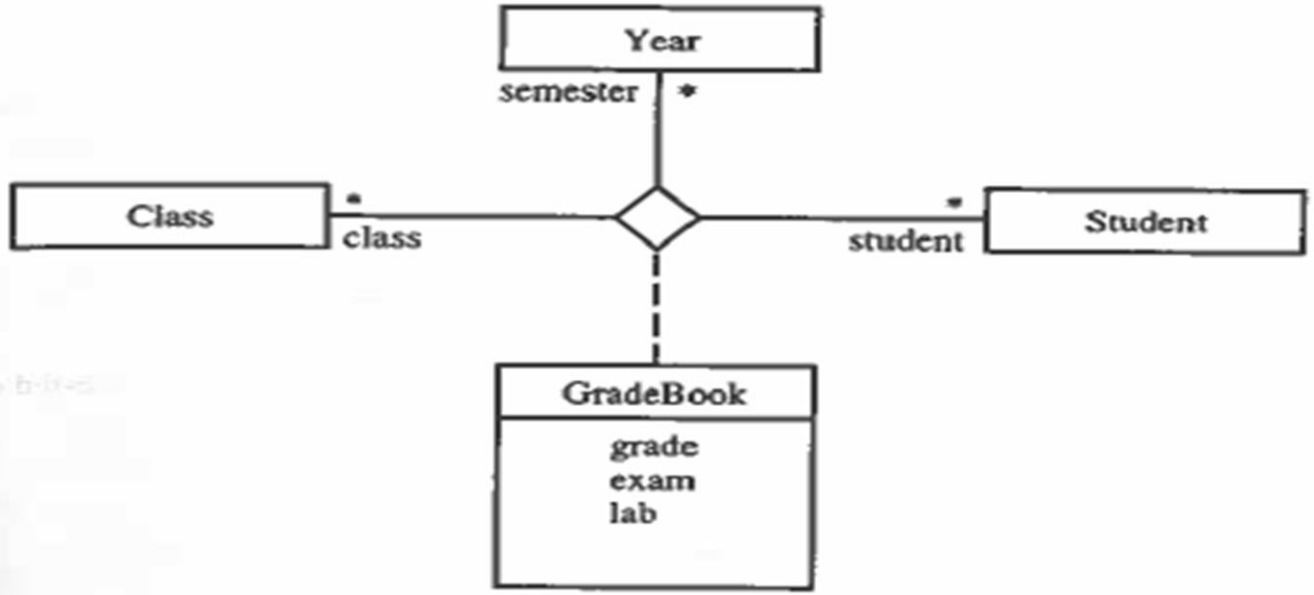


FIGURE 5–8
An n-ary (ternary) association that shows association among class, year, and student classes. The association class GradeBook which contains the attributes of the associations such as grade, exam, and lab.

AGGREGATION AND COMPOSITION (A-PART-OF)

- Aggregation is a form of association. A hollow diamond is attached to the end of the path to indicate aggregation.
- However, the diamond may not be attached to both ends of a line, and it need not be presented at all.

FIGURE 5–9
Association path.

- Composition also is referred to as a part-whole relationship. The UML notation for composition is a solid diamond at the end of a path.

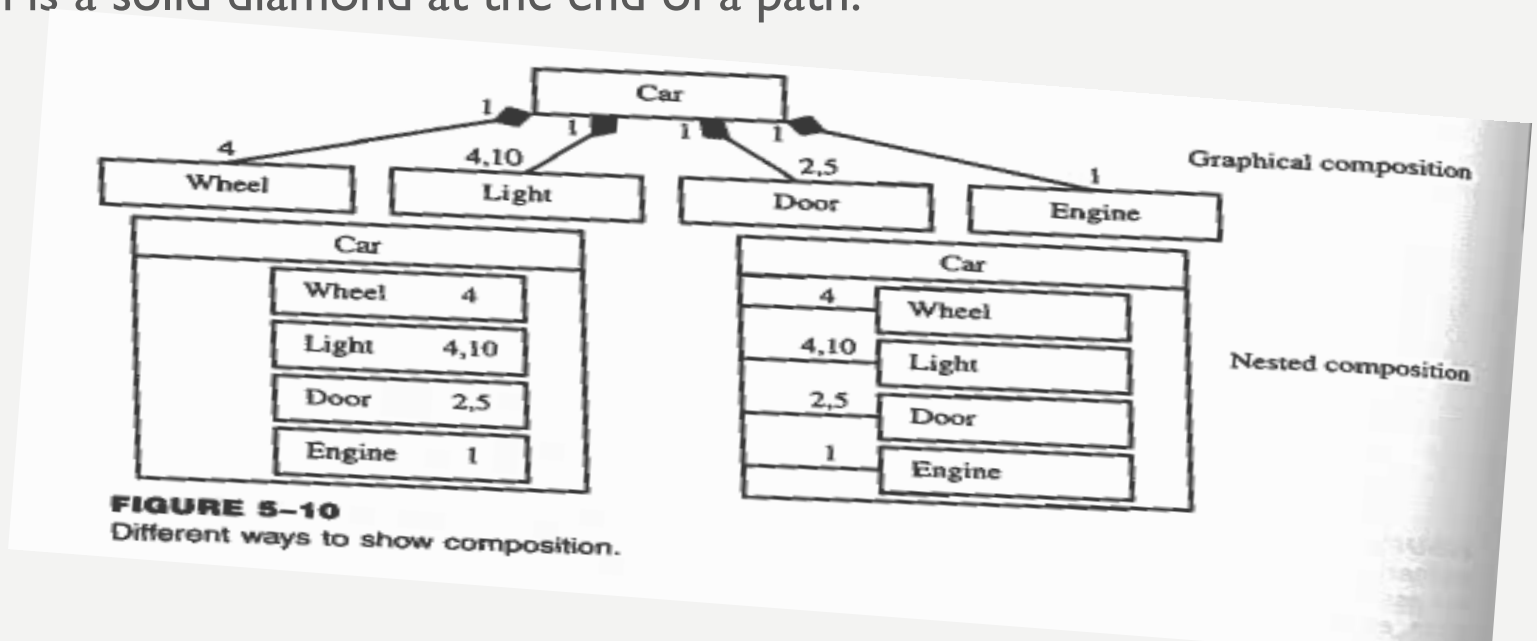
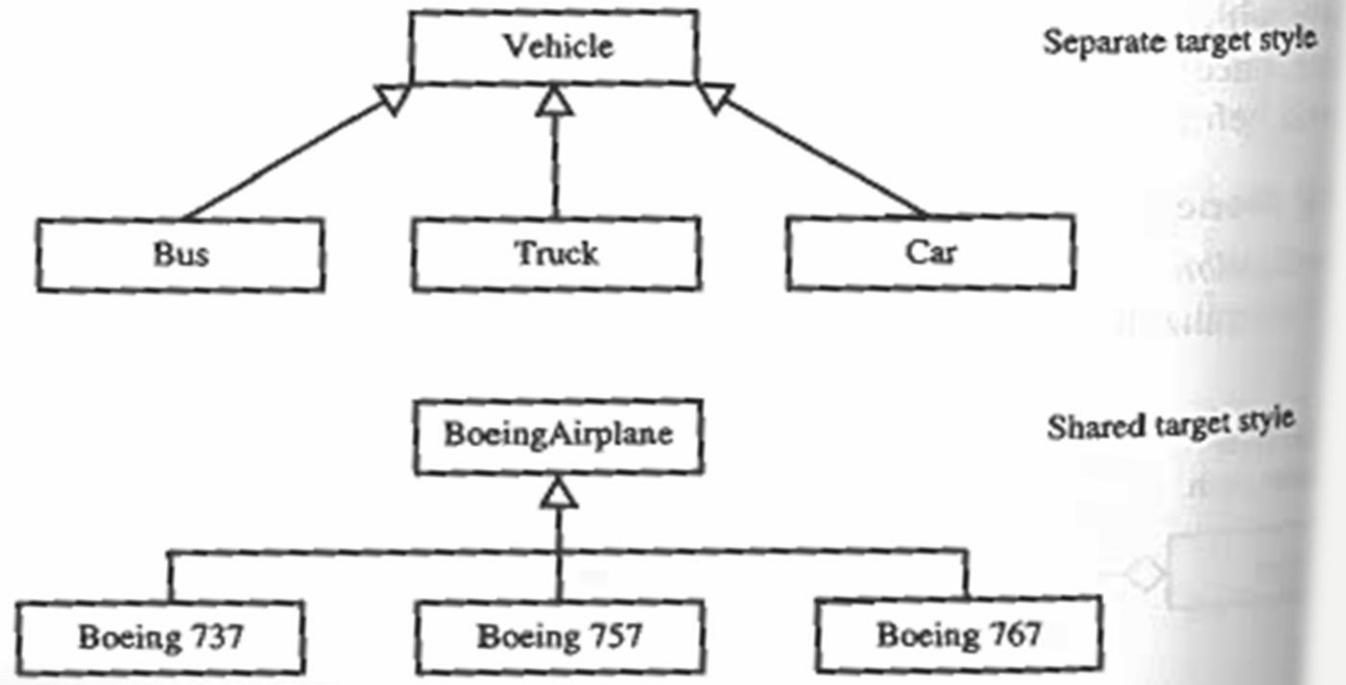


FIGURE 5–10
Different ways to show composition.

GENERALIZATION

- Generalization is the relationship between a more general class and a more specific class.
- Generalization is displayed as a directed line with a closed, hollow arrowhead at the superclass end.

FIGURE 5–11
Generalization notation.



USE- CASE DIAGRAM

- A use-case diagram is a graph of actor, a set of use cases enclosed by a system boundary, communication (participation) associations between the actors and the use cases, and generalization among the use cases.

Diagram use cases for a help desk.

- A use case diagram shows the relationship among the actors and use cases within a system.
- A client makes a call that is taken by an operator, who determines the nature of the problem.
- Some calls can be answered immediately; other calls require research and a return call.
- A use case is shown as an ellipse containing the name of the use case.
- The name of the use case can be placed below or inside the ellipse. Actors' names and use case names should followed the capitalization and punctuation guidelines of the model.
- An actor is shown as a class rectangle with the label <<actor>>, or the label and a stick figure, or just the stick figure with the name of the actor below the figure.

These relationships are shown in a use-case diagram:

1. Communication. The actor is said to "communicate" with the use case.
2. Uses. A uses relationship between use cases is shown by a generalization arrow from the use case.
3. Extends. The extends relationship is used you have one use case that is similar to another use case but does a bit more. In essence, it is like a subclass.

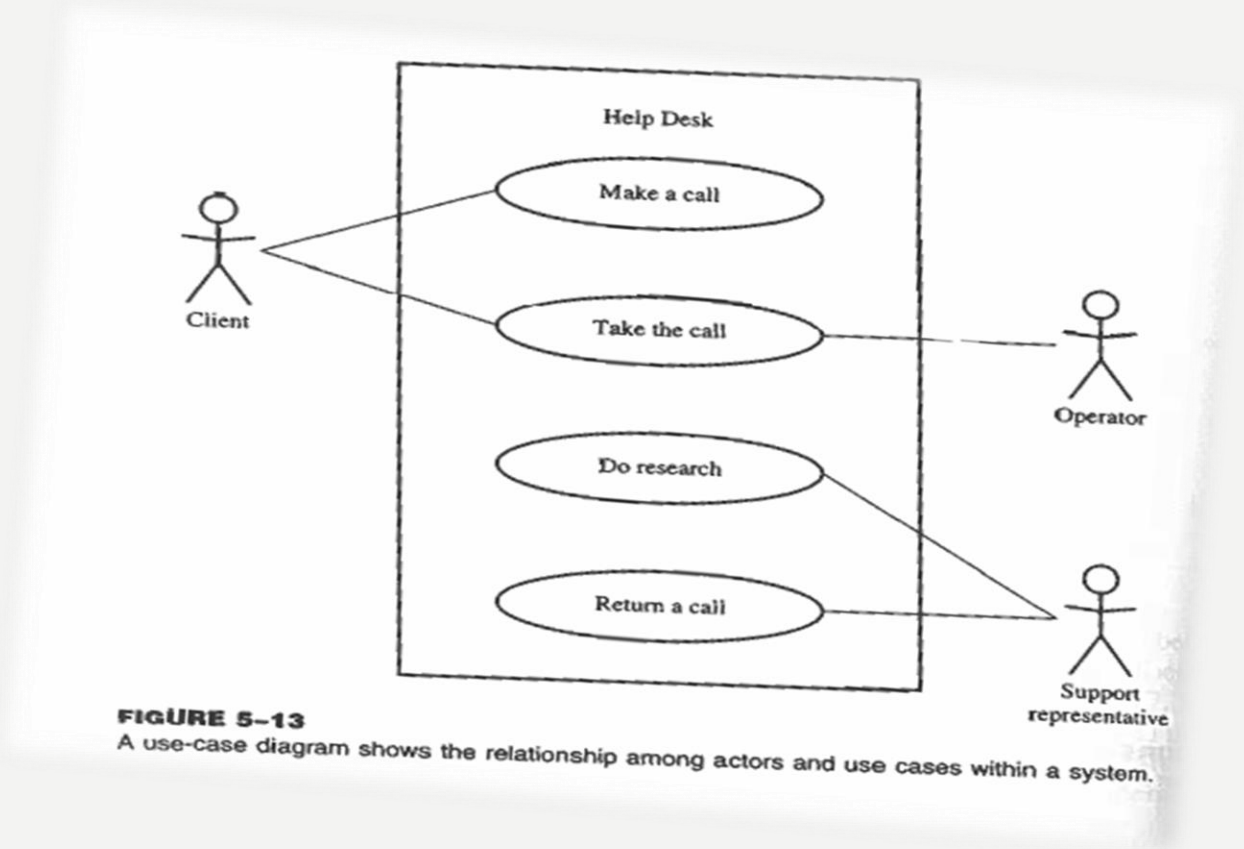


FIGURE 5–13
A use-case diagram shows the relationship among actors and use cases within a system.

UML DYNAMIC MODELLING (BEHAVIOUR DIAGRAMS)

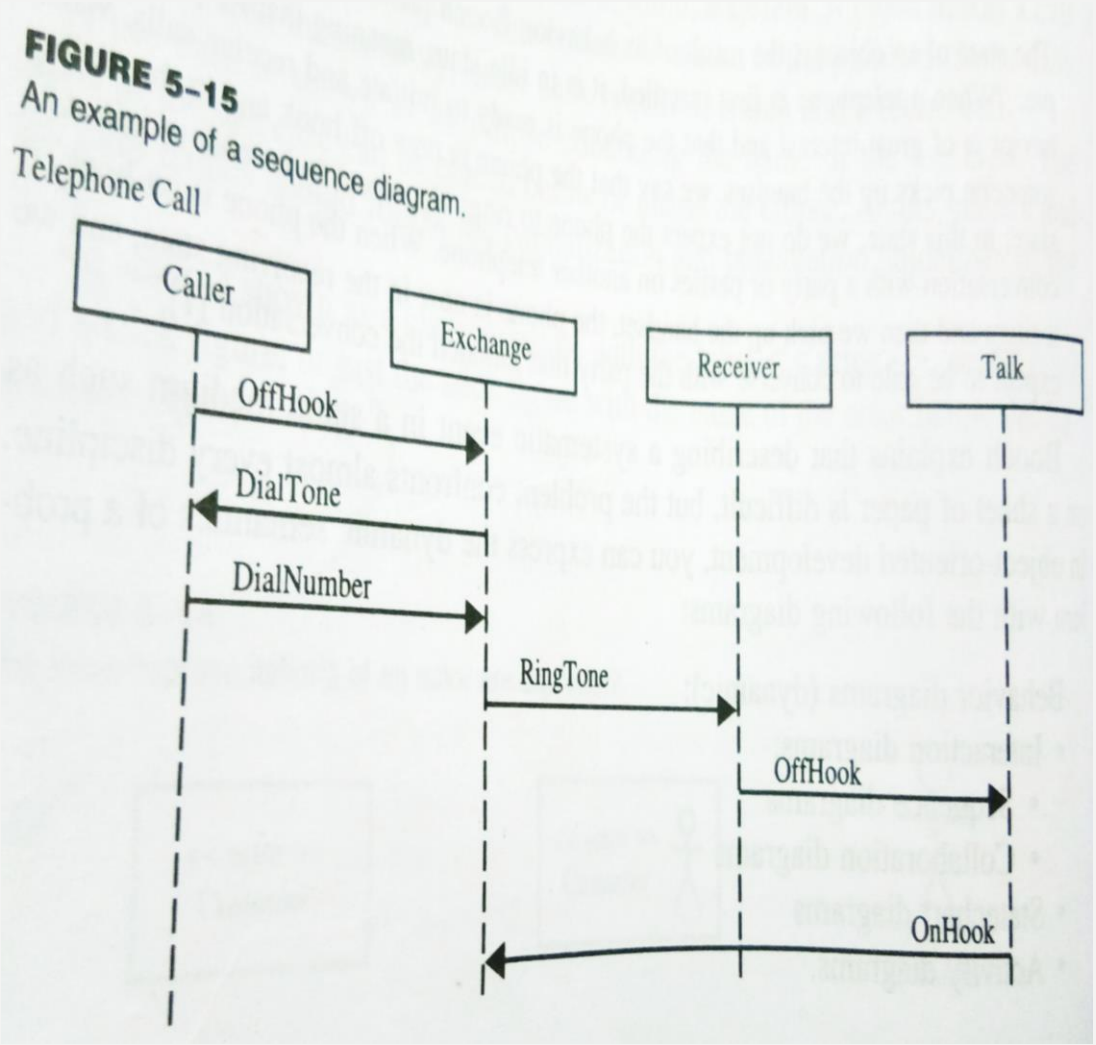
- Interaction diagrams capture the behaviour of a single use case, showing the pattern of interaction among objects.

There are two kinds of interaction models: sequence diagrams and collaboration diagrams.

1. **UML sequence diagram sequence diagrams**
2. **UML collaboration diagram**
3. **UML State chart Diagram**
4. **UML Activity Diagram**

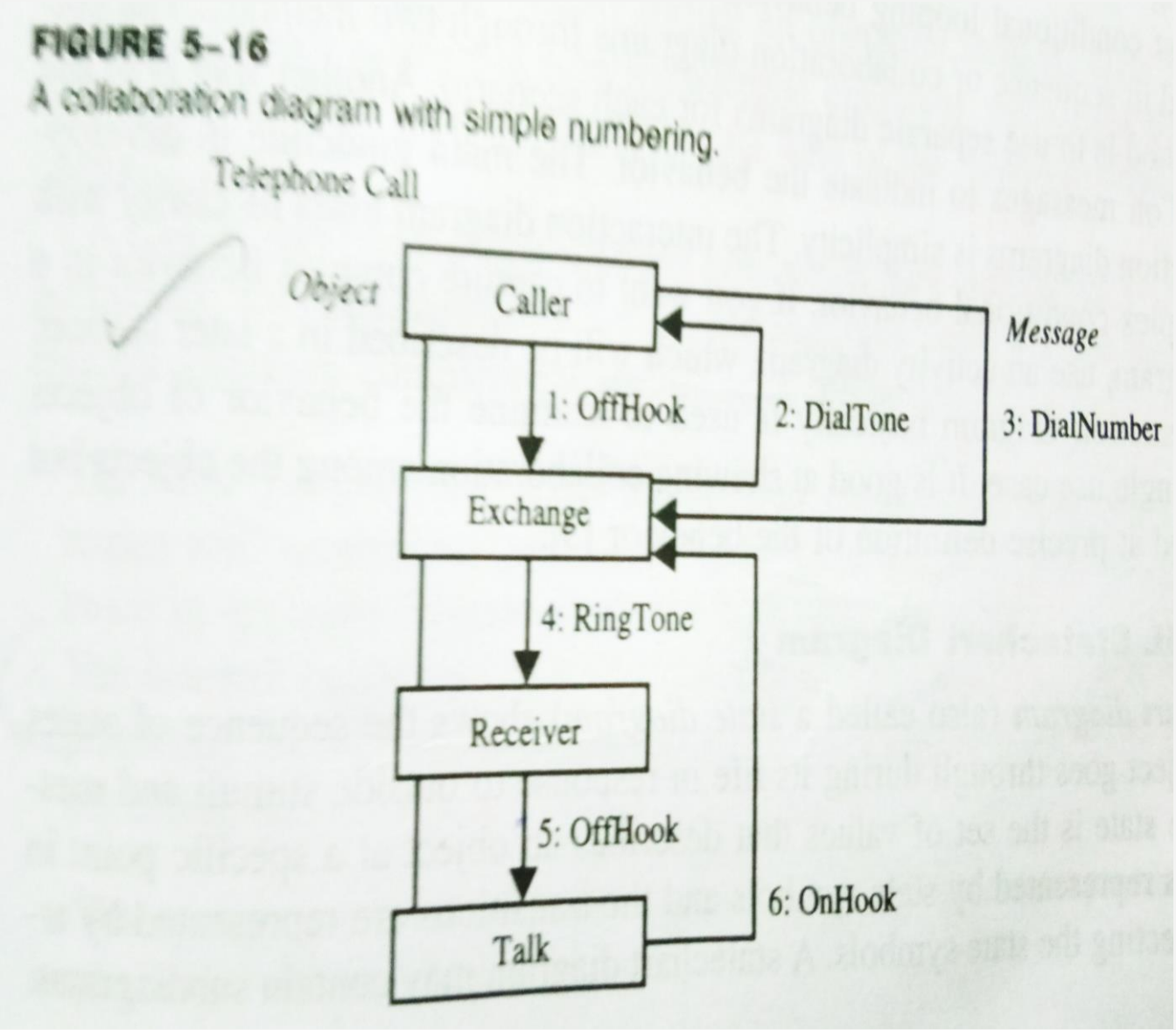
UML SEQUENCE DIAGRAM SEQUENCE DIAGRAMS

- A sequence diagram shows an interaction arranged in a time sequence.
- A sequence diagram has two dimension: the vertical dimension represents time, the horizontal dimension represented different objects.
- The vertical line is called the object's lifeline. The lifeline represents the object's existence during the interaction.
- This form was first popularized by Jacobson. An object is shown as a box at the top of a dashed vertical line.
- The sequence diagram is very simple and has immediate visual appeal.



UML COLLABORATION DIAGRAM

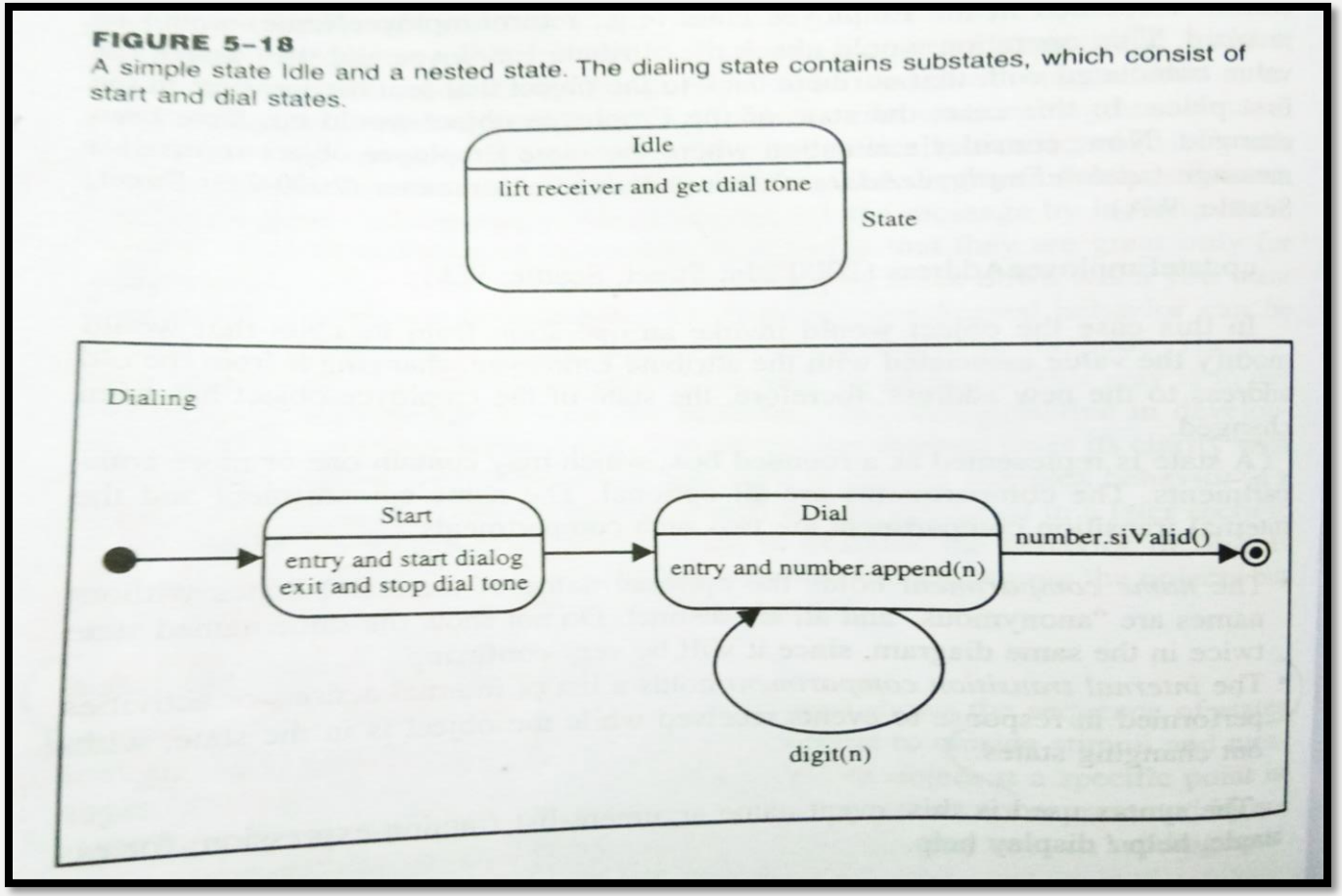
- A collaboration diagram represents a collaboration, which is a set of object related in a particular context, and interaction, which is a set of message exchanged among the objects within the collaboration to achieve a desired outcome.
- In a collaboration diagram, the sequence is indicated by numbering the messages.



UML STATE CHART DIAGRAM

- A state chart diagram (also called a state diagram) shows the sequence of states that an object goes through during its life in response to outside stimuli and messages.
- The purpose of the state diagram is to understand the algorithm involved in performed a method.
- A token (shown by a solid black dot) represents an activity symbol.
- When an activity symbol appears within a state symbol, it indicates the execution of an operation.
- An outgoing solid arrow attached to a state chart symbol indicates a transition triggered by the completion of the activity.
- A state is represented as a rounded box, which may contain one or more compartments.
- The compartments are all optional.
- The name compartment and the internal transition compartments are two such compartments:
  - The name compartment holds the optional name of the state.
  - The internal transition compartment holds a list of internal actions or activities performed in response to events received while the object is in the state, without changing states.
- The transition can be simple or complex. A simple transition is a relationship between two states indicating that an object in the first state will enter the second state and perform certain actions when a specific event occurs; if the specified condition are satisfied, the transition is said to "fire". Events are processed one at a time. An event that triggers no transition is simply ignored.
- A complex transition may have multiple source and target states.
- It represents a synchronization or a splitting of control into concurrent threads.

FIGURE 5–18
A simple state Idle and a nested state. The dialing state contains substates, which consist of start and dial states.



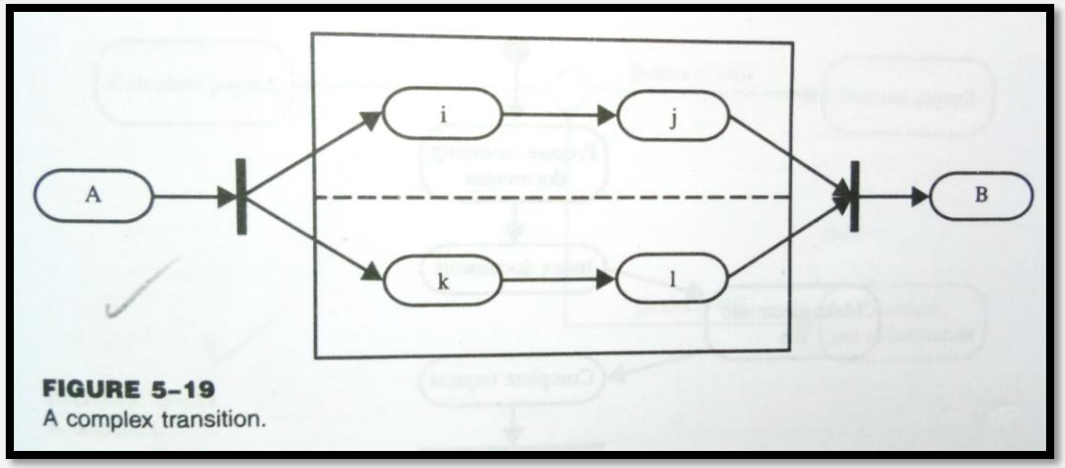


FIGURE 5–19
A complex transition.

UML ACTIVITY DIAGRAM

- An activity diagram is a variation or special case of a state machine, in which the state are activities representing the performance of operations and the transitions are triggered by the completion of the operations.

Component diagram component diagrams model the physical components (such as source code, executable program, user interface) in a design.

- Another way of looking at components is the concept of packages.
- A component diagram is a graph of the design's components connected by dependency relationships.
- A component is represented by the boxed figure shown in fig 5.23. Dependency is shown as a dashed arrow.

Deployment diagram deployment diagrams show the configuration of run-time processing elements and the software components, processes, and objects that live in them.

- A deployment diagram is a graph of nodes connected by communication association.
- Components are connected to other components by dashed-arrow dependencies, usually through interfaces, which indicate one component uses the services of another.
- Each node or processing element in the system is represented by a three-dimensional box.
- Connections between the nodes (or platforms) themselves are shown by solid lines (fig 5.24).

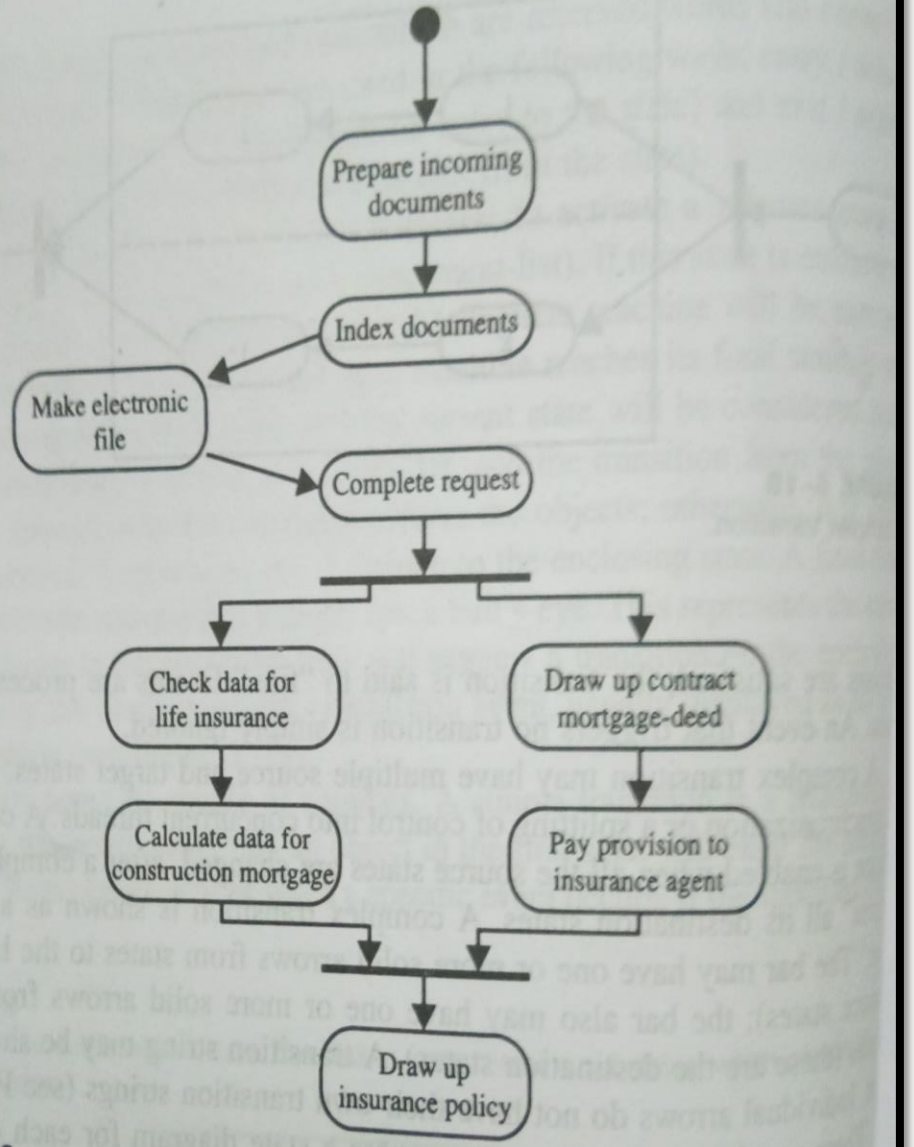


FIGURE 5–20
An activity diagram for processing mortgage requests (Loan: Processing Mortgage Request)

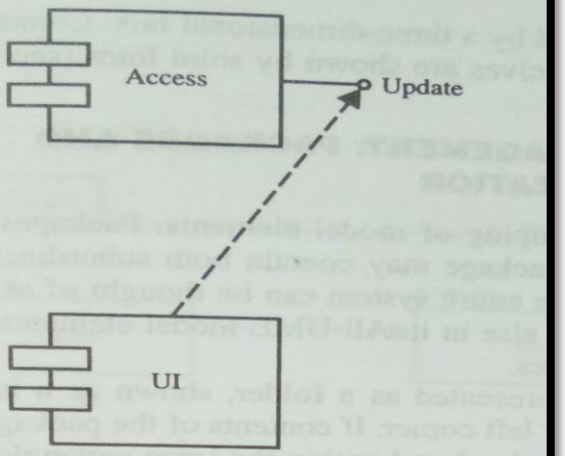


FIGURE 5–23
A component diagram.

FIGURE 5–24
The basic UML notation for a deployment diagram.