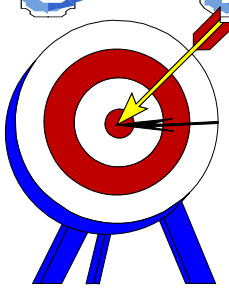
Object-Oriented Systems Development:
Using the Unified Modeling Language

Chapter 5:
Unified Modeling Language



Goals

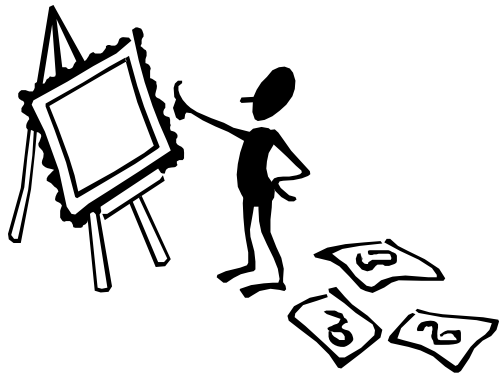
- **Modeling.**
- **Unified modeling language.**
  - **Class diagram.**
  - **Use case diagram.**
  - **Interaction diagrams.**
    - **Sequence diagram.**
    - **Collaboration diagram.**

Goals (Con't)

- **Statechart diagram.**
- **Activity diagram.**
- **Implementation diagrams.**
  - **Component diagram.**
  - **Deployment diagram.**

Introduction

- **A *model* is an abstract representation of a system, constructed to understand the system prior to building or modifying it.**
- **Most of the modeling techniques involve graphical languages.**

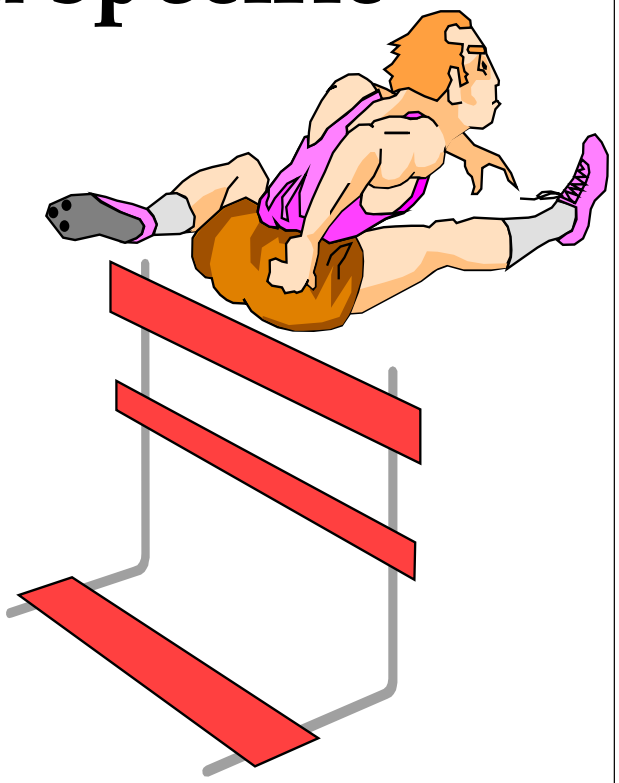


Static or Dynamic Models

- **Models can represent**
  - **– static or**
  - **– dynamic situations.**

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.**
- **The classes' structure and their relationships to each other frozen in time are examples of static models.**

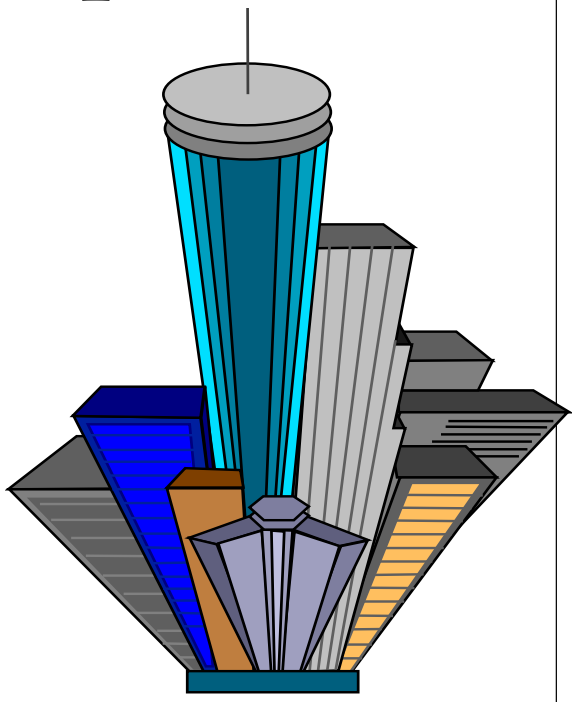


Dynamic Model

- **Is a collection of procedures or behaviors that, taken together, reflect the behavior of a system over time.**
- **For example, an order interacts with inventory to determine product availability.**

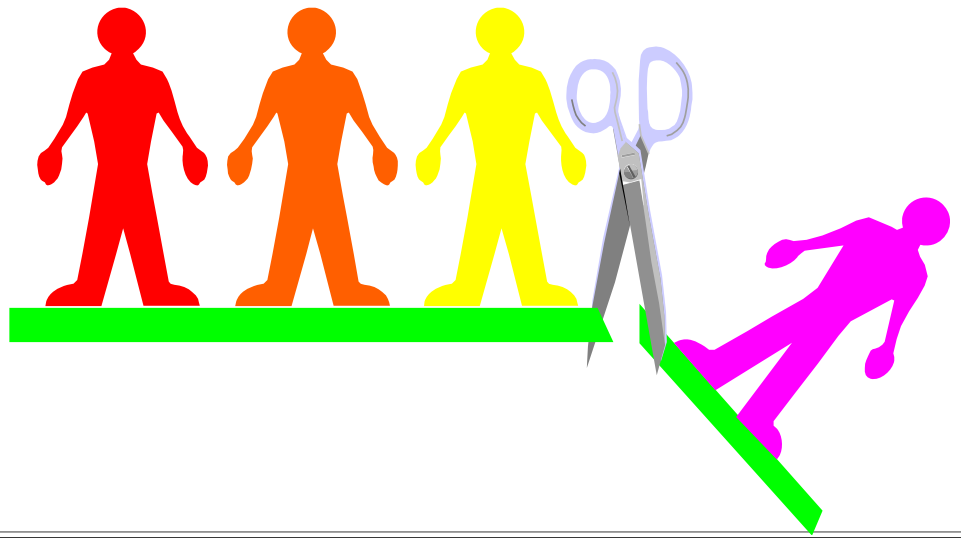
Why Modeling?

- **Turban cites the following advantages:**
- **Models make it easier to express complex ideas.**
- **For example, an architect builds a model to communicate ideas more easily to clients.**



Advantages of Modeling (Con't)

- **Models reduce complexity by separating those aspects that are unimportant from those that are important.**

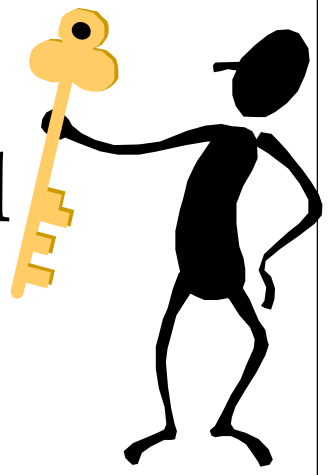


Advantages of Modeling (Con't)

- **Models enhance learning.**
- **The cost of the modeling analysis is much lower than the cost of similar experimentation conducted with a real system.**
- **Manipulation of the model (changing variables) is much easier than manipulating a real system.**

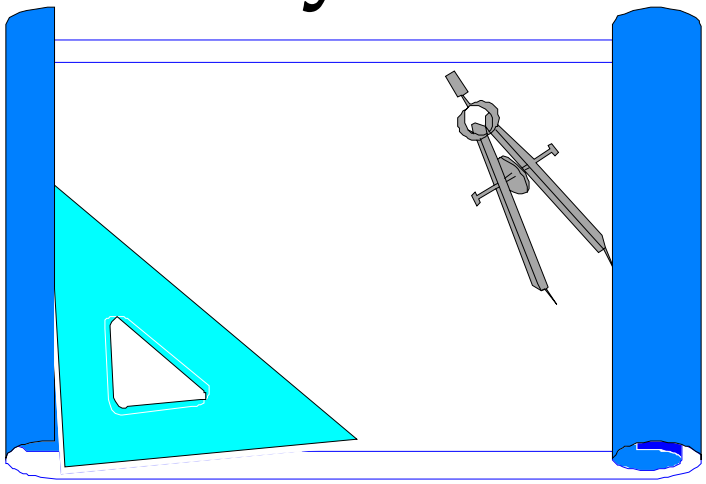
Modeling Key Ideas

- **A model is rarely correct on the first try.**
- **Always seek the advice and criticism of others.**
- **Avoid excess model revisions, as they can distort the essence of your model. Let simplicity and elegance guide you through the process.**



The Unified Modeling Language (UML)

- **The unified modeling language (UML) is a language for specifying, constructing, visualizing, and documenting the software system and its components.**



UML Diagrams

The UML defines nine graphical diagrams:

1. **Class diagram (static)**
2. **Use-case diagram**
3. **Behavior diagrams (dynamic):**
   - **3.1. Interaction diagram:**
     - **3.1.1. Sequence diagram**
     - **3.1.2. Collaboration diagram**

UML Diagrams

- 3.2. Statechart diagram
- 3.3. Activity diagram

4. Implementation diagram:

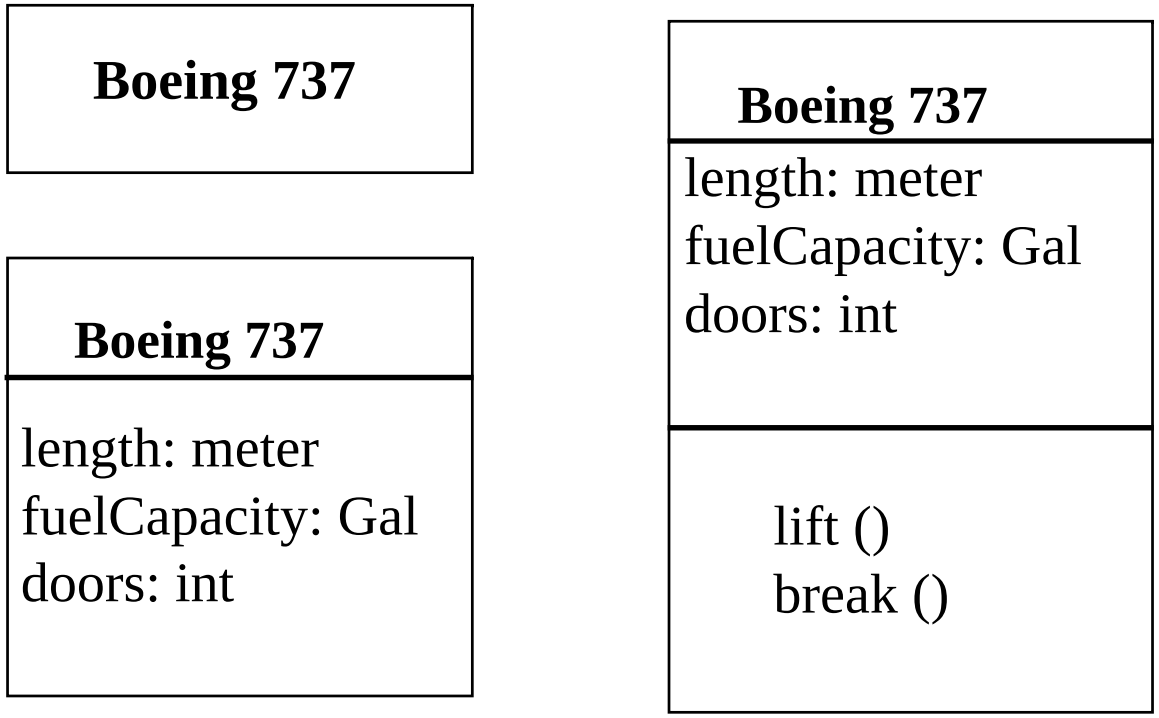
- 4.1. Component diagram
- 4.2. Deployment diagram

UML Class Diagram,

- **The UML class diagram is the main static analysis diagram.**
- **Class diagrams show the static structure of the model.**
- **Class diagram is collection of static modeling elements, such as classes and their relationships.**

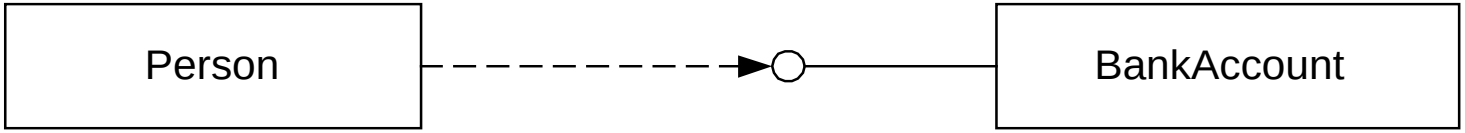
Class Notation

- **In class notation, either or both the attributes and operation compartments may be suppressed.**



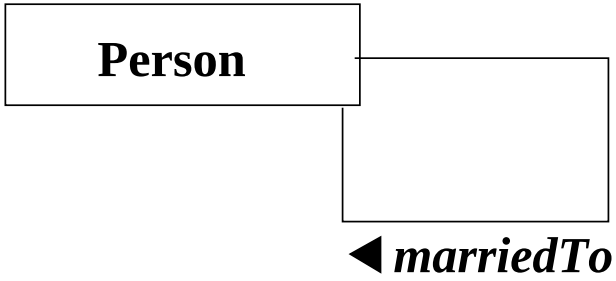
Class Interface Notation

- **Class interface notation is used to describe the externally visible behavior of a class.**
- **For example, an operation with a public visibility.**



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.**

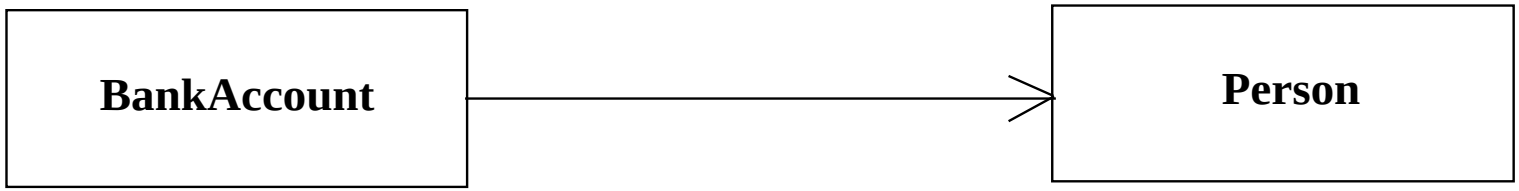


Association Role

- **A simple association—the technical term for it is *binary association*—is drawn as a solid line connecting two class symbols.**
- **The end of an association, where it connects to a class, shows the association role.**

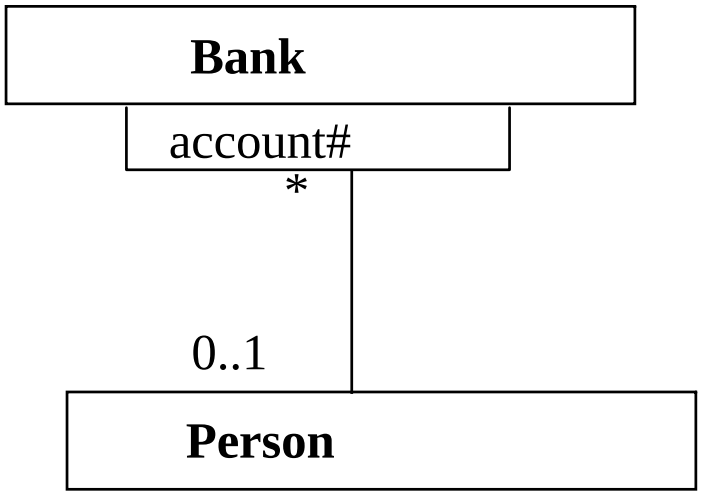
UML Association Notation

- **In the UML, association is represented by an open arrow.**



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.**

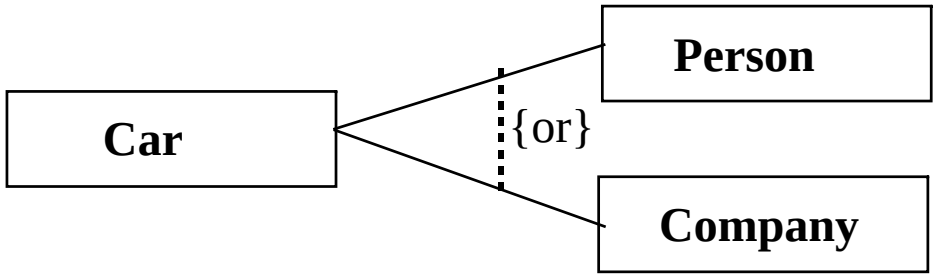


Multiplicity

- **Multiplicity specifies the range of allowable associated classes.**
- **It is given for roles within associations, parts within compositions, repetitions, and other purposes.**
- **lower bound .. upper bound.**

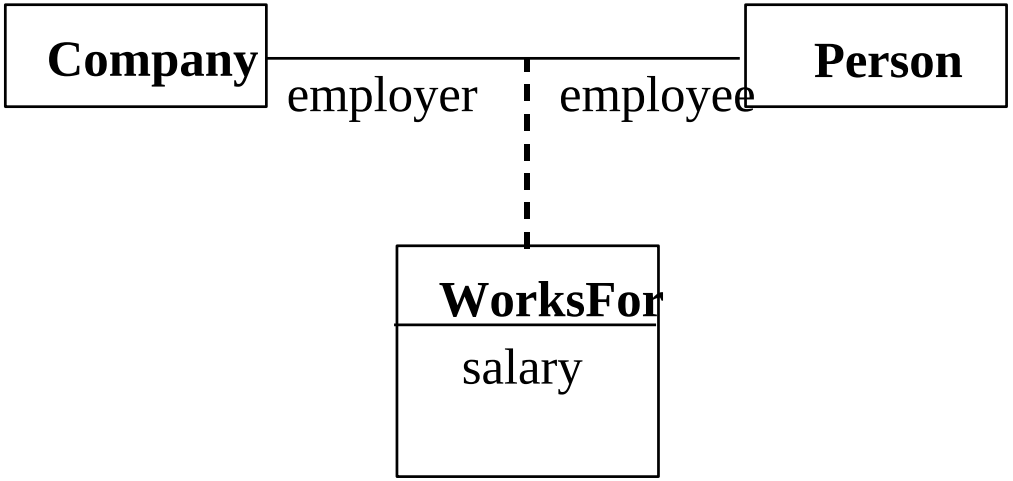
OR Association

- **An *OR association* indicates a situation in which only one of several potential associations may be substantiated at one time for any single object.**



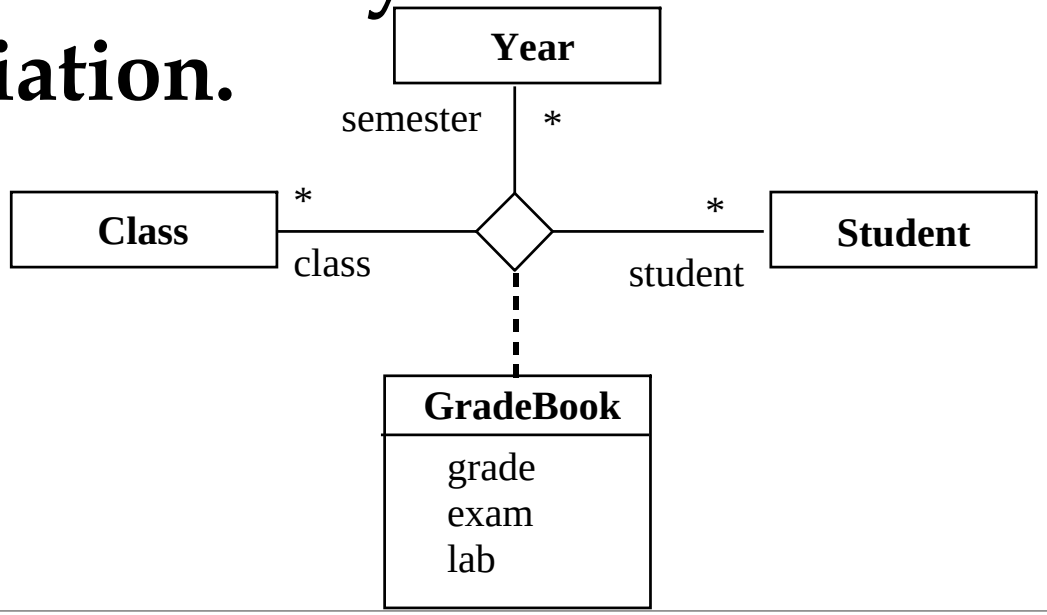
Association Class

- **An *association class* is an association that also has class properties.**
- **An association class is shown as a class symbol attached by a dashed line to an association path.**



N-Ary Association

- An *n-ary association* is an association among more than two classes.
- Since n-ary association is more difficult to understand, it is better to convert an n-ary association to binary association.

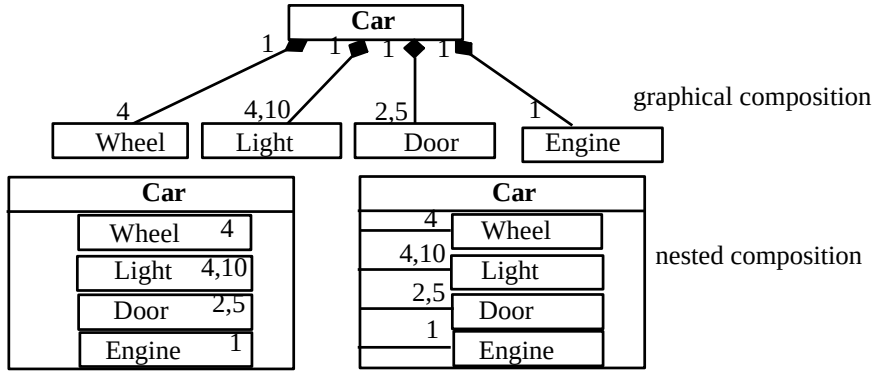


Aggregation

- **Aggregation is a form of association.**
- **A hollow diamond is attached to the end of the path to indicate aggregation.**

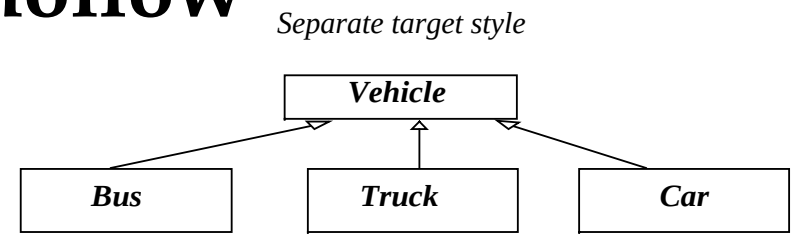
Composition

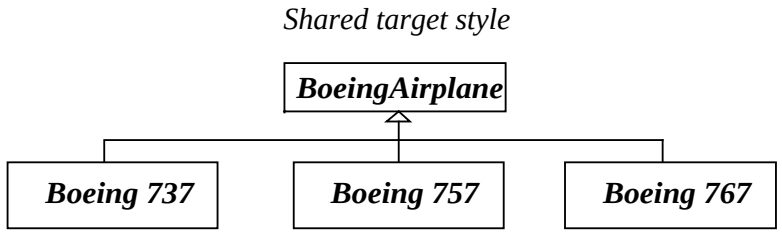
- **Composition, also known as the *a-part-of*, is a form of aggregation with strong ownership to represent the component of a complex object.**
- **The UML notation for composition is a solid diamond at the end of a path.**



Generalization

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



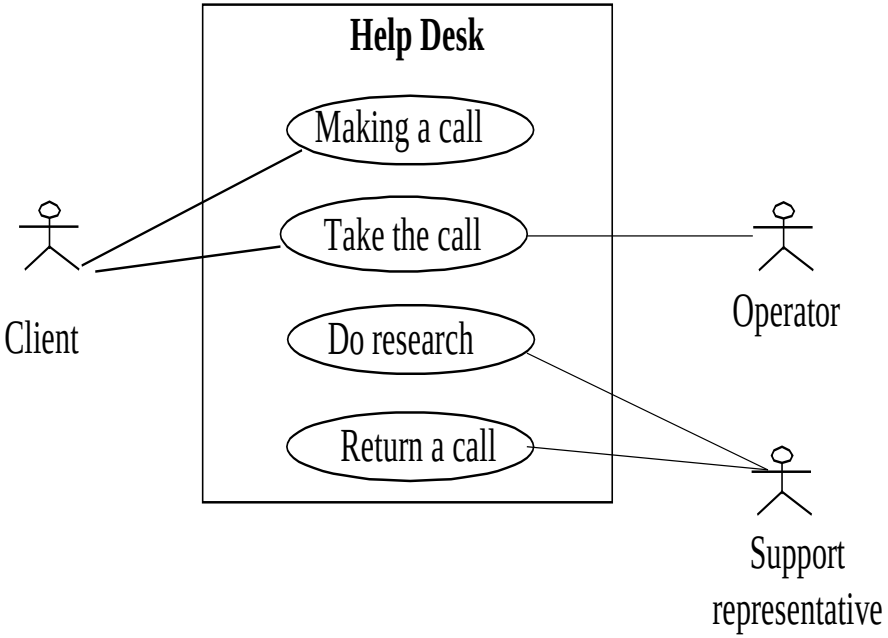


Use-Case Diagram

- **The description of a use case defines what happens in the system when the use case is performed.**
- **In essence, the use-case model defines the outside (actors) and inside (use case) of the system's behavior.**

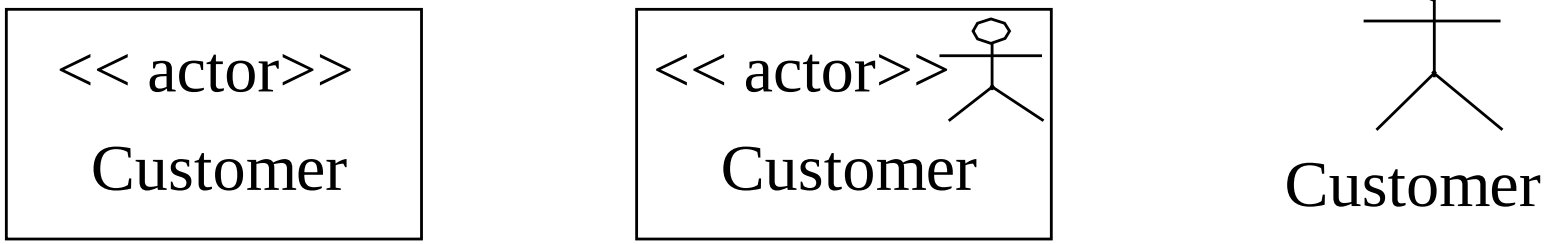
Use-Case Diagram (Con't)

- **A use-case diagram is a graph of actors, 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.**



Actor Notations

- **The three representations of an actor are equivalent.**

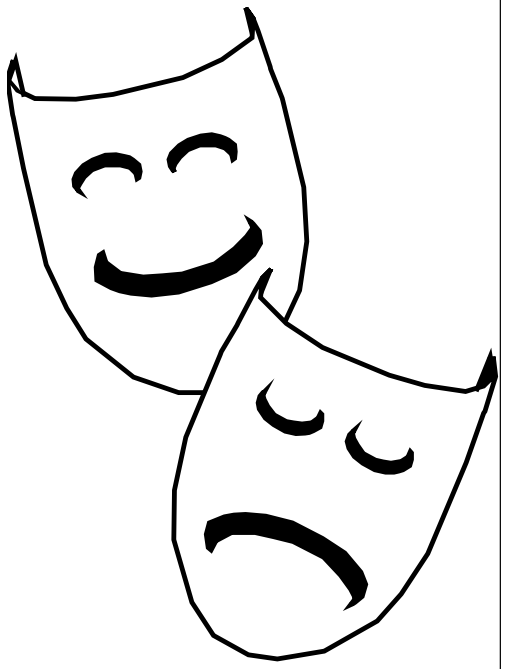


Use-Case Diagram (Con't)

- **These relationships are shown in a use-case diagram:**
  - ***Communication.***
  - ***Uses.***
  - ***Extends.***

Behavior or Dynamic Diagrams

- **Interaction diagrams:**
  - **Sequence diagrams**
  - **Collaboration diagrams**
- **Statechart diagrams**
- **Activity diagrams**

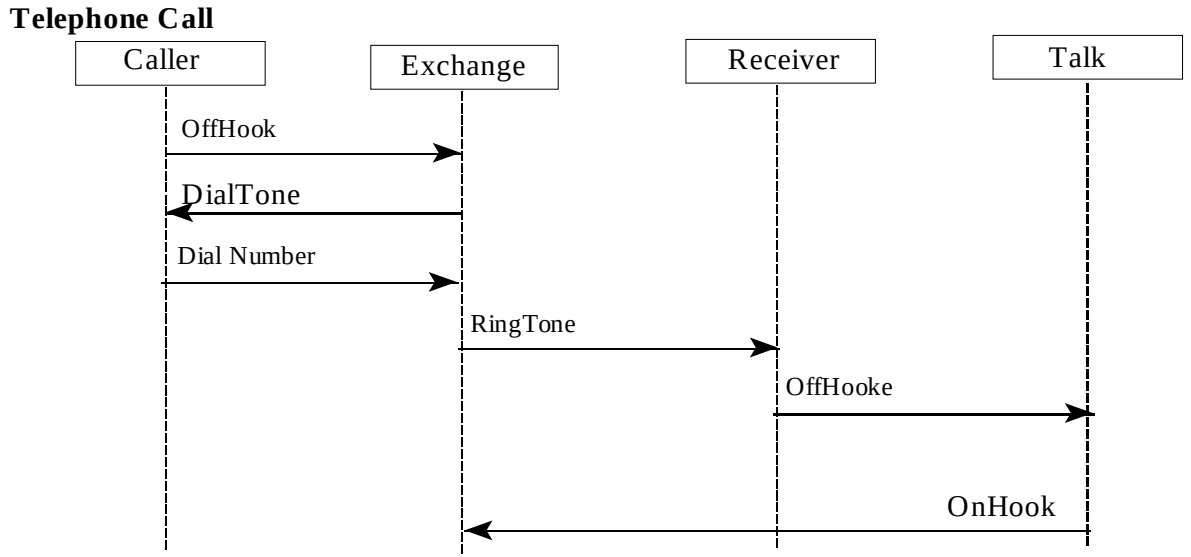


UML Interaction Diagrams

- **Interaction diagrams describe how groups of objects collaborate to get the job done.**
- **Interaction diagrams capture the behavior of a single use case, showing the pattern of interaction among objects.**

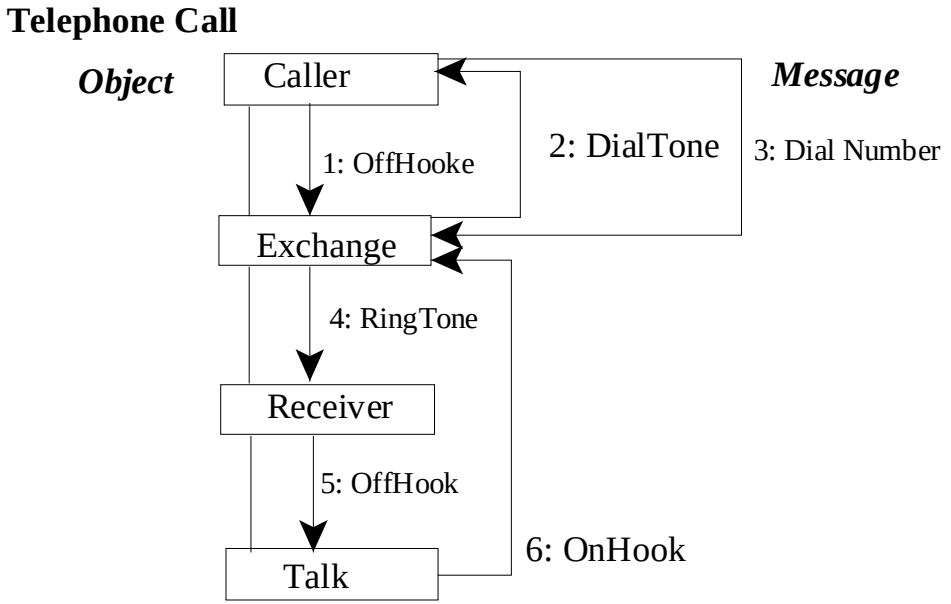
UML Sequence Diagram

- ***Sequence diagrams*** **are an easy and intuitive way of describing the behavior of a system.**
- **A sequence diagram shows an interaction arranged in a time sequence.**

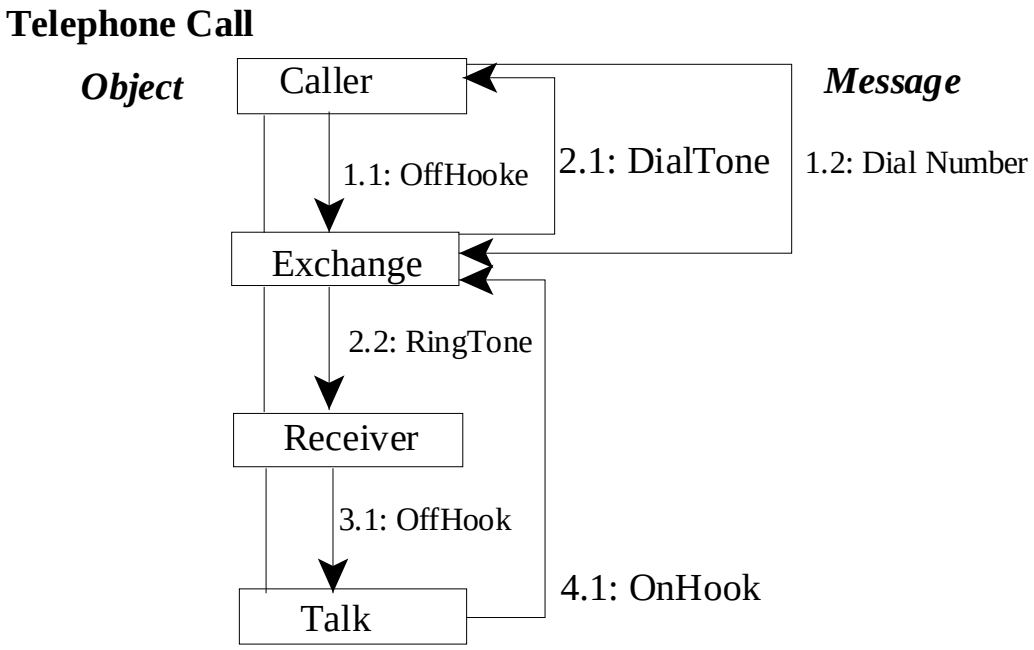


UML Collaboration Diagram

- **A collaboration diagram represents a set of objects related in a particular context, and the exchange of their messages to achieve a desired outcome.**

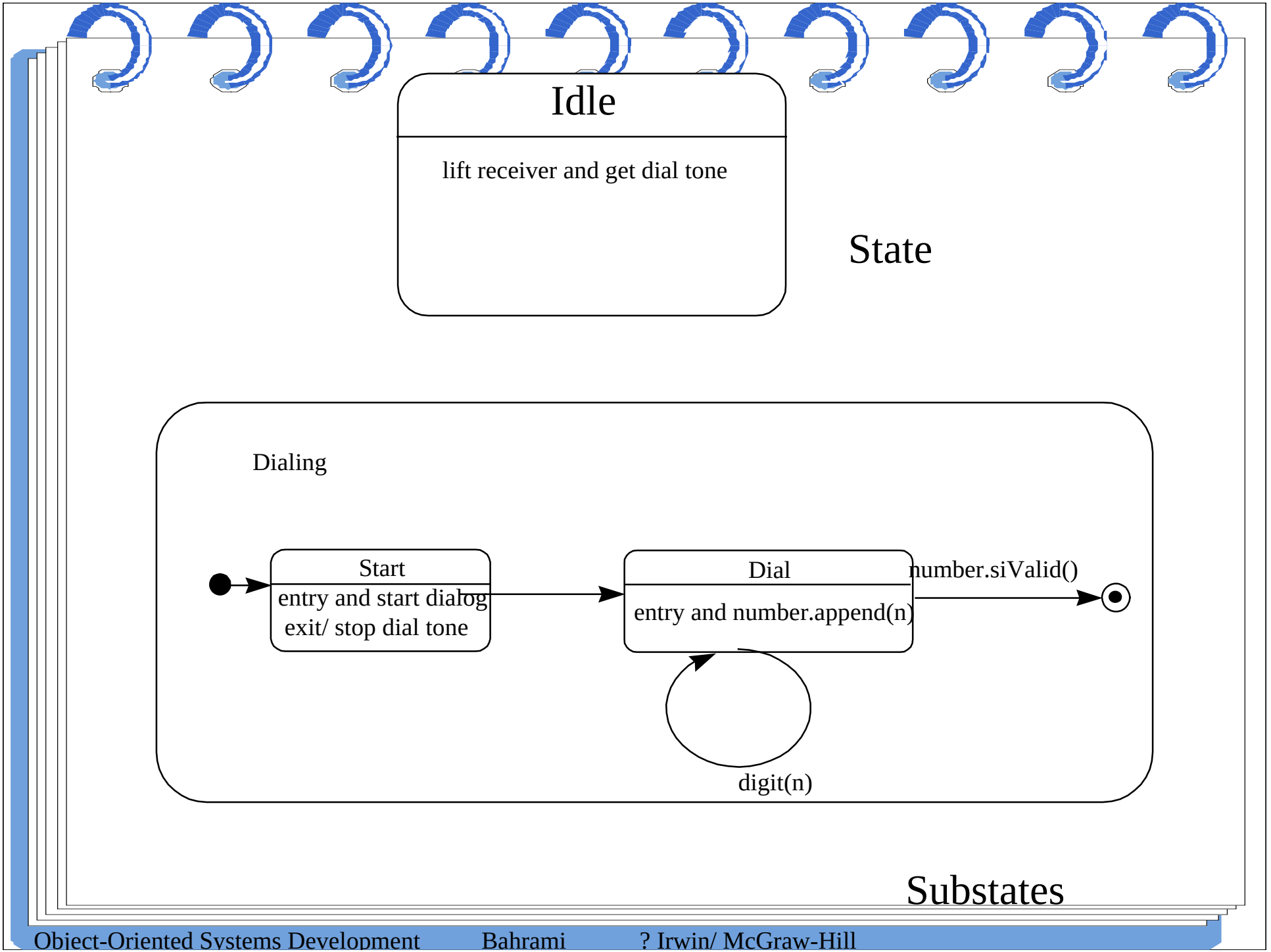


UML Collaboration Diagram (Con't)

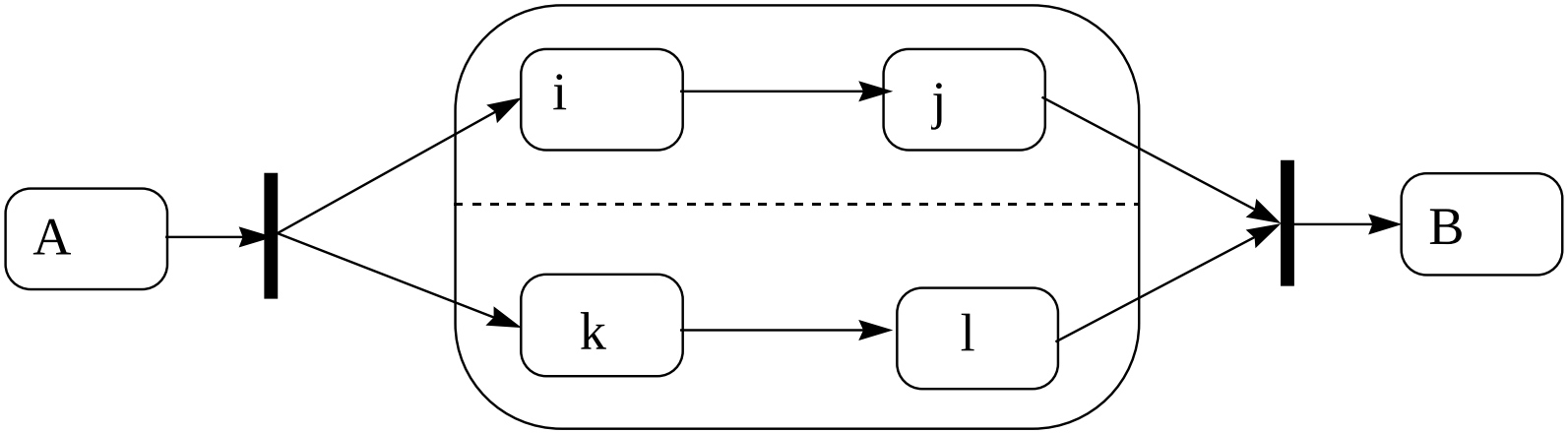


UML Statechart Diagram

- **A statechart 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.**


Idle
lift receiver and get dial tone
State
Dialing
Start
entry and start dialog
exit/ stop dial tone
Dial
entry and number.append(n)
number.siValid()
digit(n)
Substates

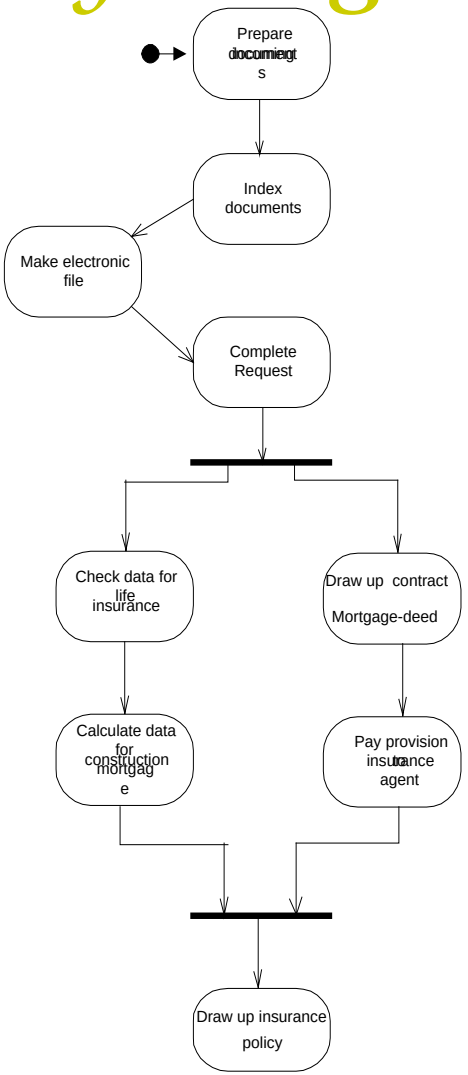
Complex Transition



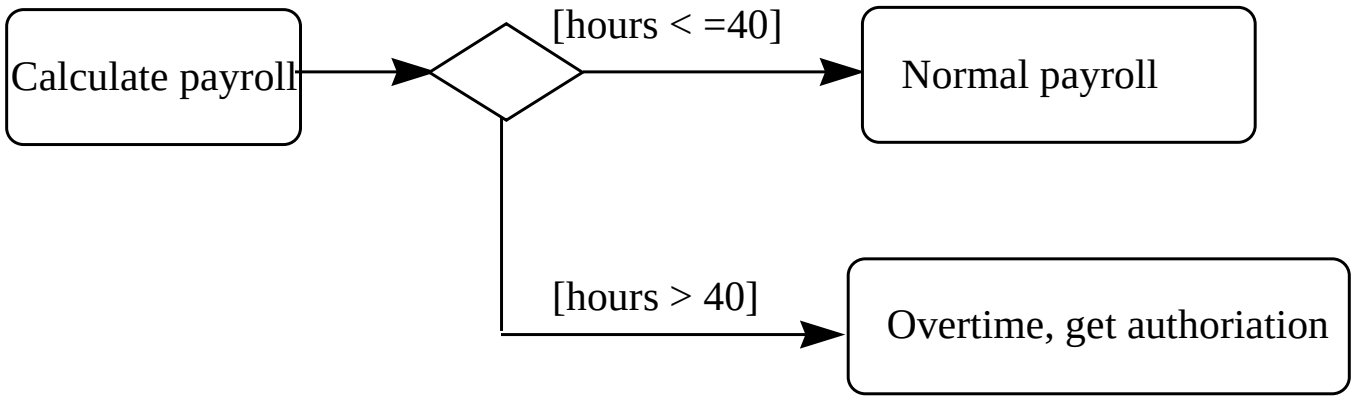
UML Activity Diagram

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

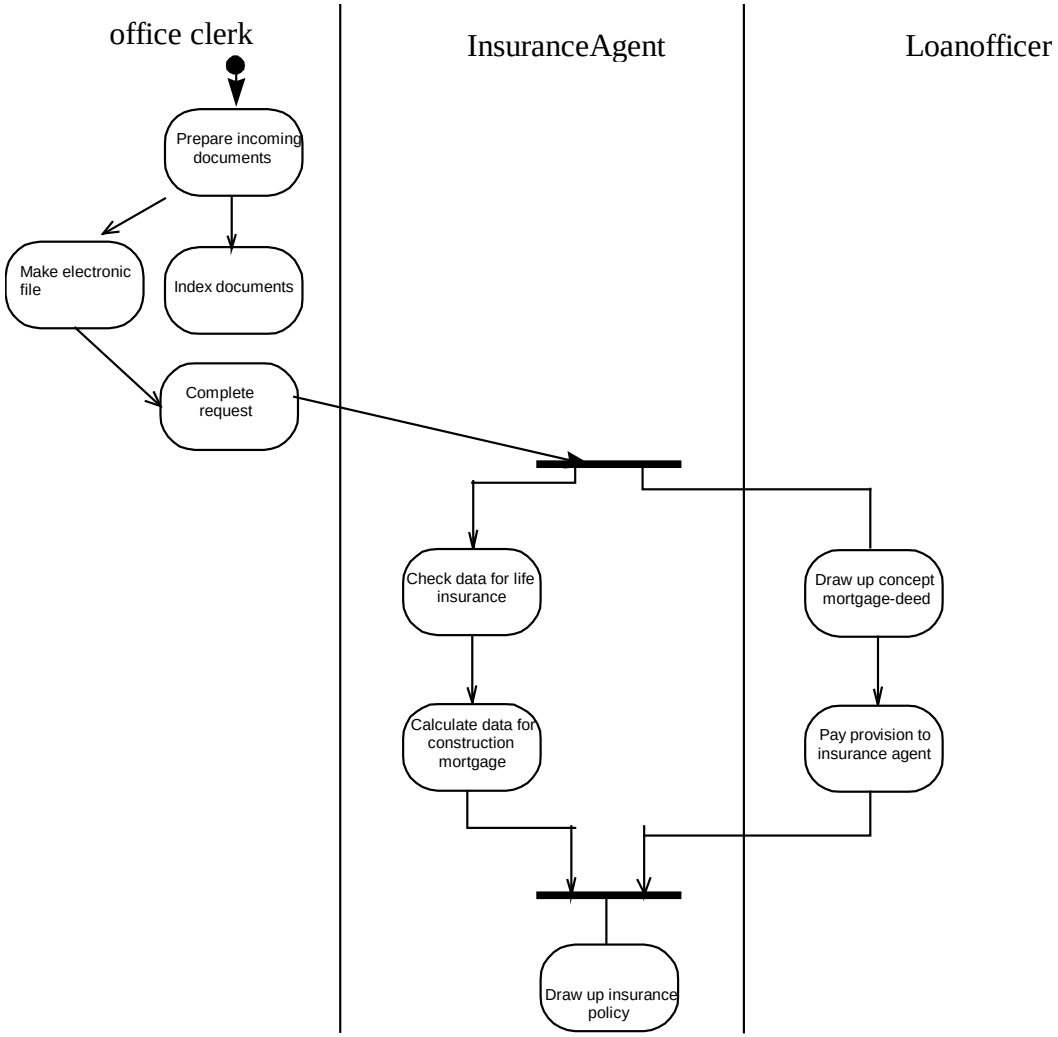
UML Activity Diagram (Con't)



UML Activity Decision



Swimlanes.


office clerk
InsuranceAgent
Loanofficer
Prepare incoming documents
Make electronic file
Index documents
Complete request
Check data for life insurance
Calculate data for construction mortgage
Draw up concept mortgage-deed
Pay provision to insurance agent
Draw up insurance policy

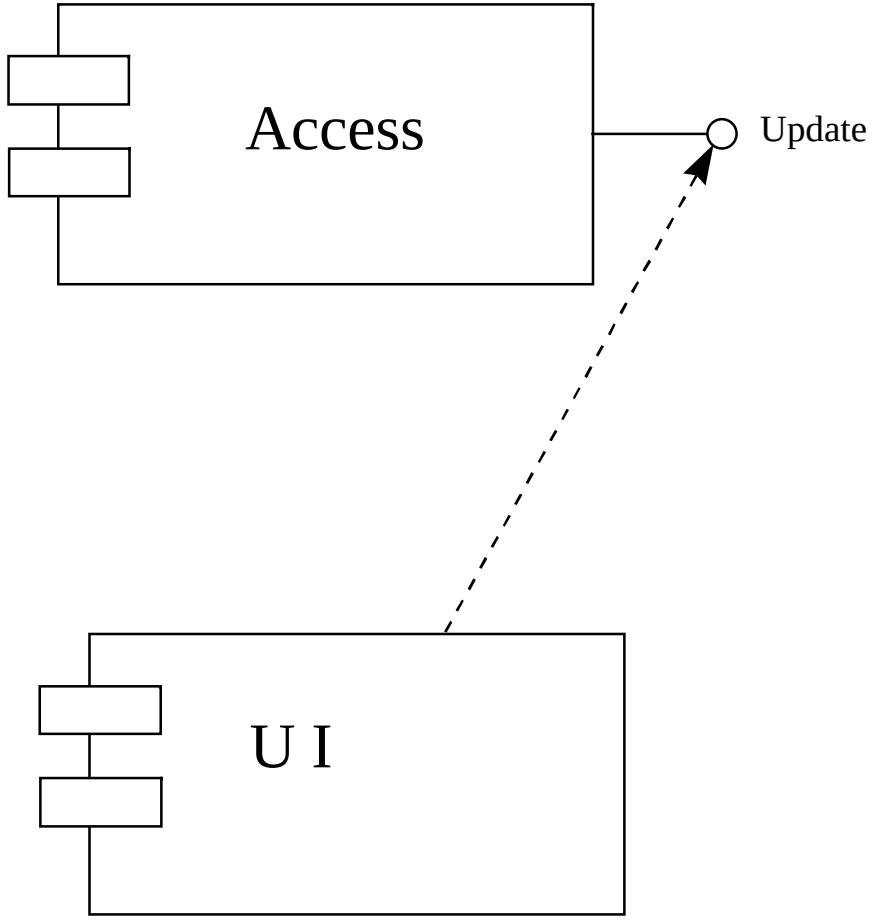
Implementation diagrams

- **These diagrams show the implementation phase of systems development.**
- **Such as the source code structure and the run-time implementation structure.**

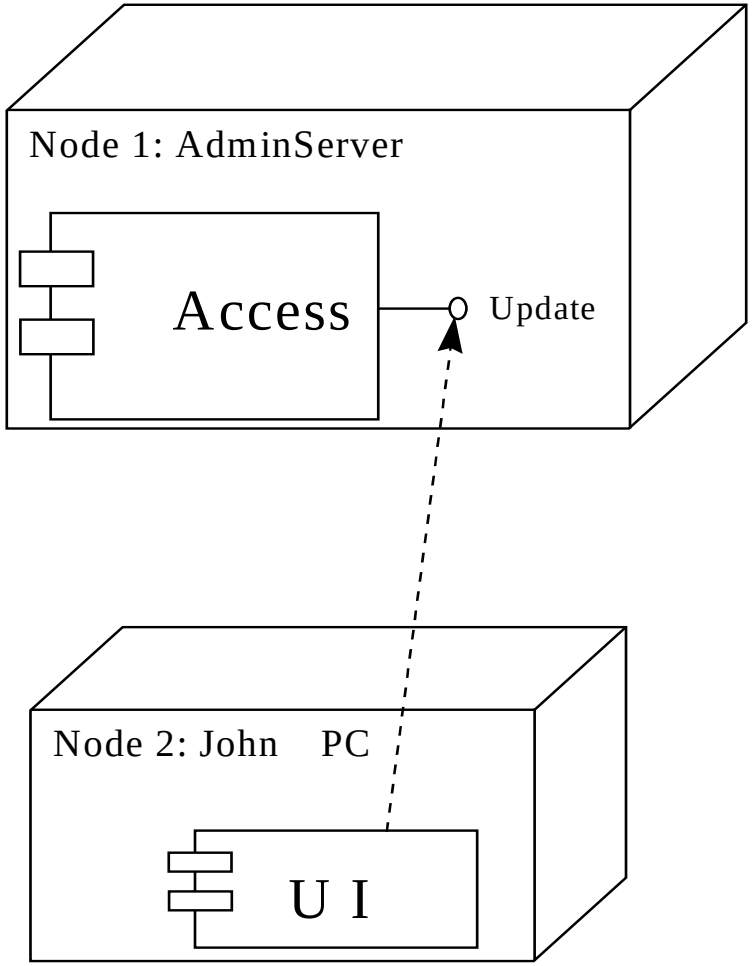
Implementation diagrams (Con't)

- **There are two types of implementation diagrams:**
  - **Component diagrams show the structure of the code itself.**
  - **Deployment diagrams show the structure of the run-time system.**

Component diagrams



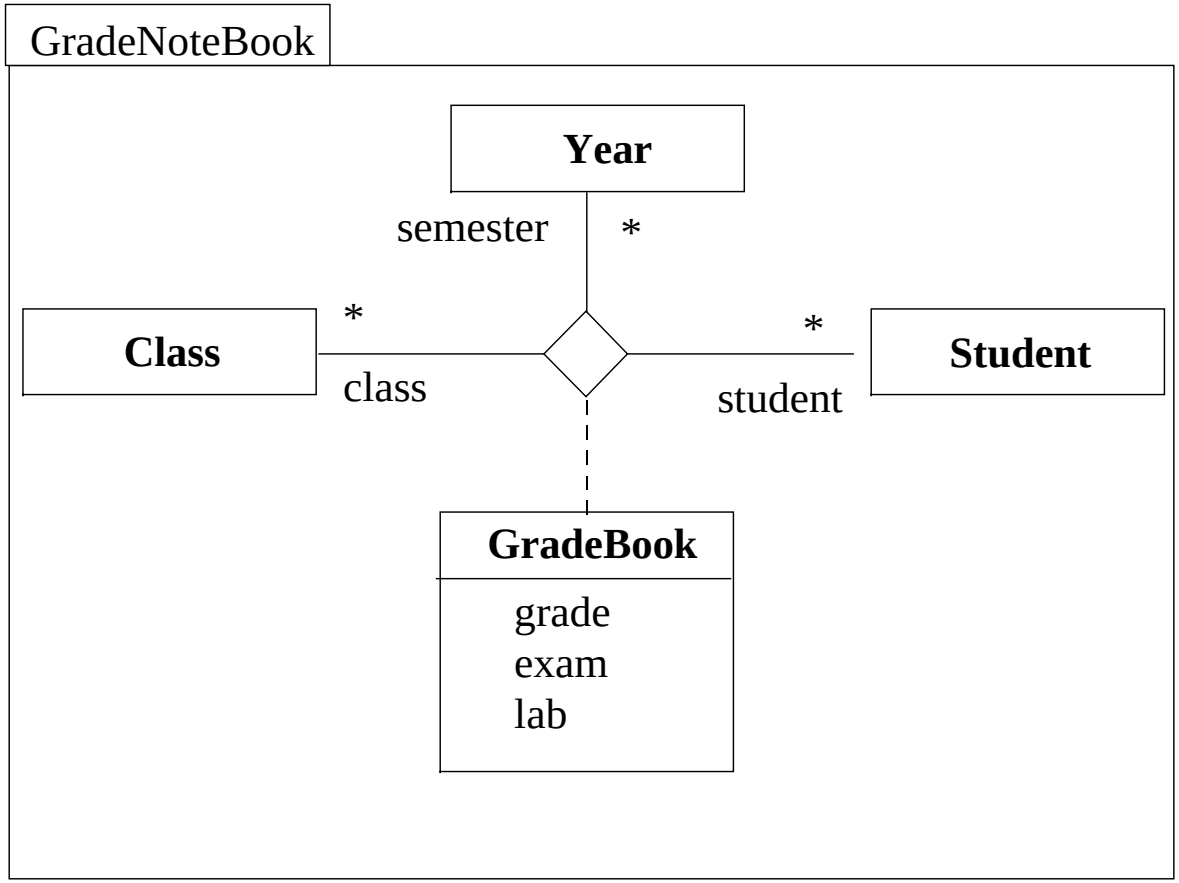
Deployment Diagram



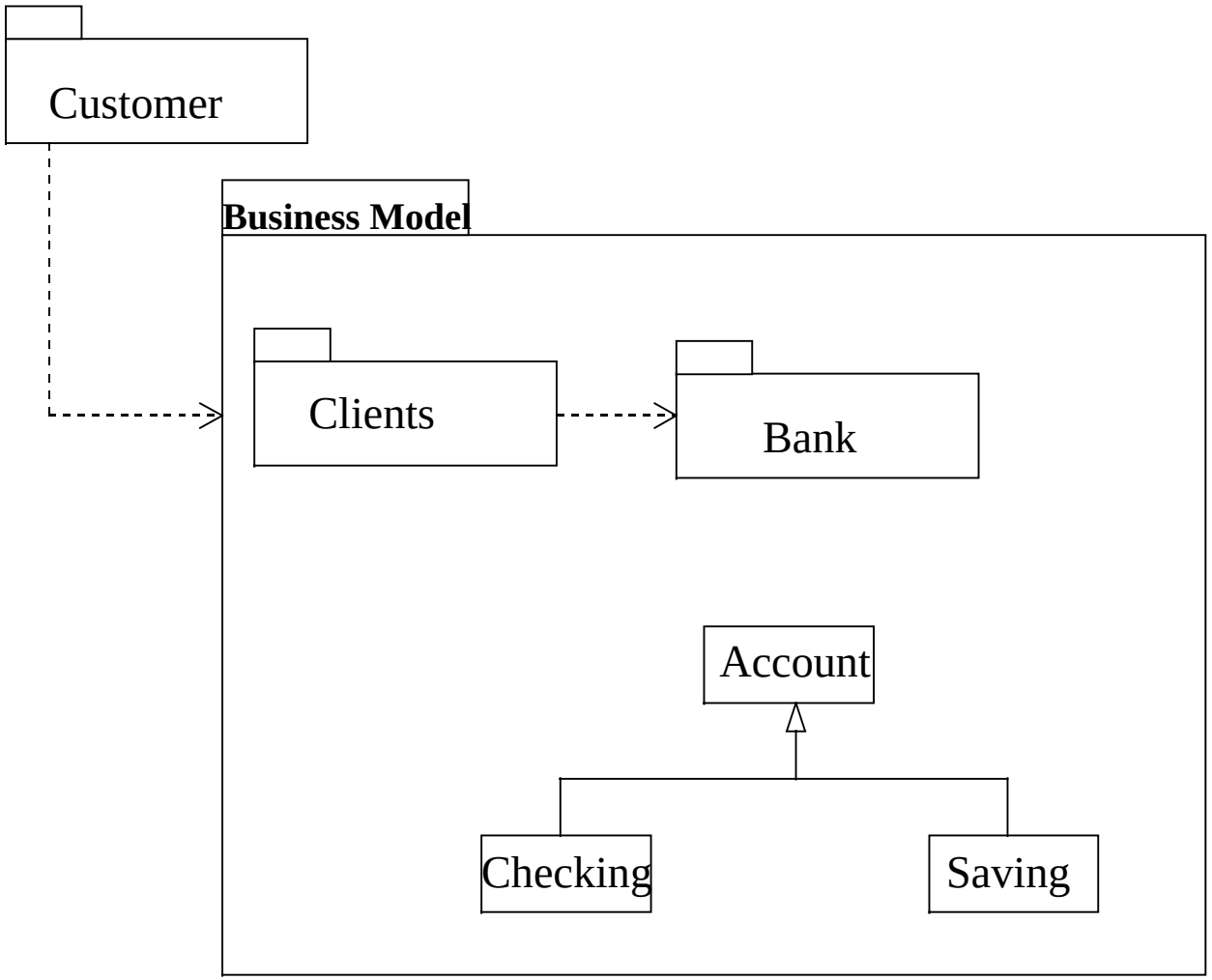
Model Management: Package

- **A package is a grouping of model elements.**
- **Packages themselves may contain other packages.**
- **A package may contain both subordinate packages and ordinary model elements.**

A Package and Its Contents

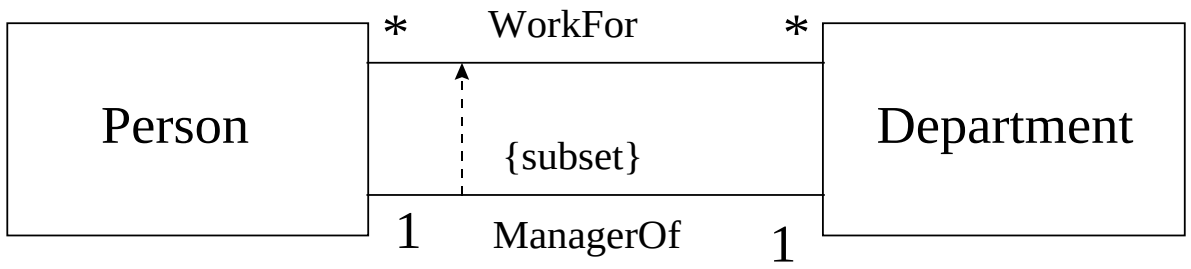


A Package and Its Dependencies



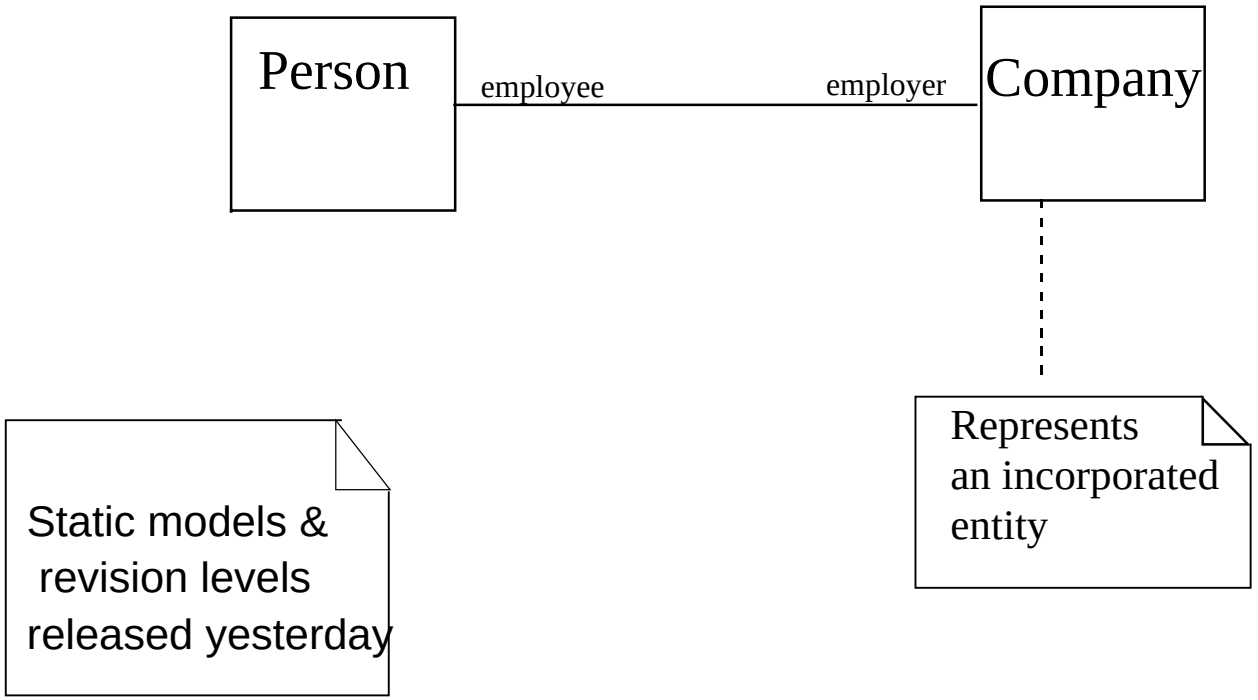
Model Constraints and Comments

- **Constraints are assumptions or relationships among model elements specifying conditions and propositions that must be maintained as true otherwise the system described by the model would be invalid.**



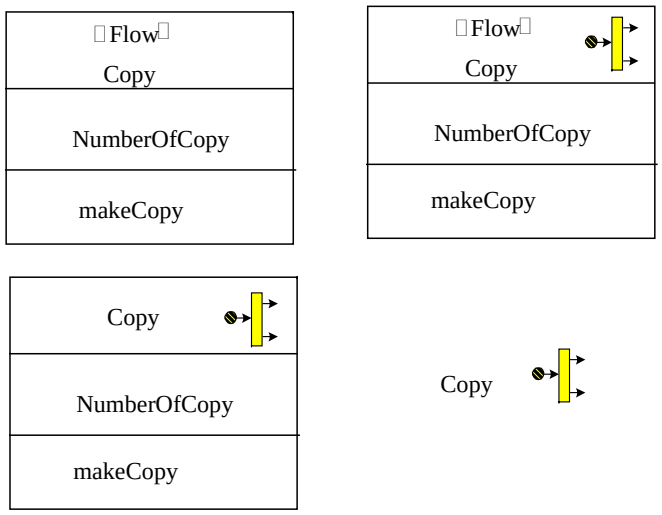
Note

- **A *note* is a graphic symbol containing textual information; it also could contain embedded images.**



Stereotype

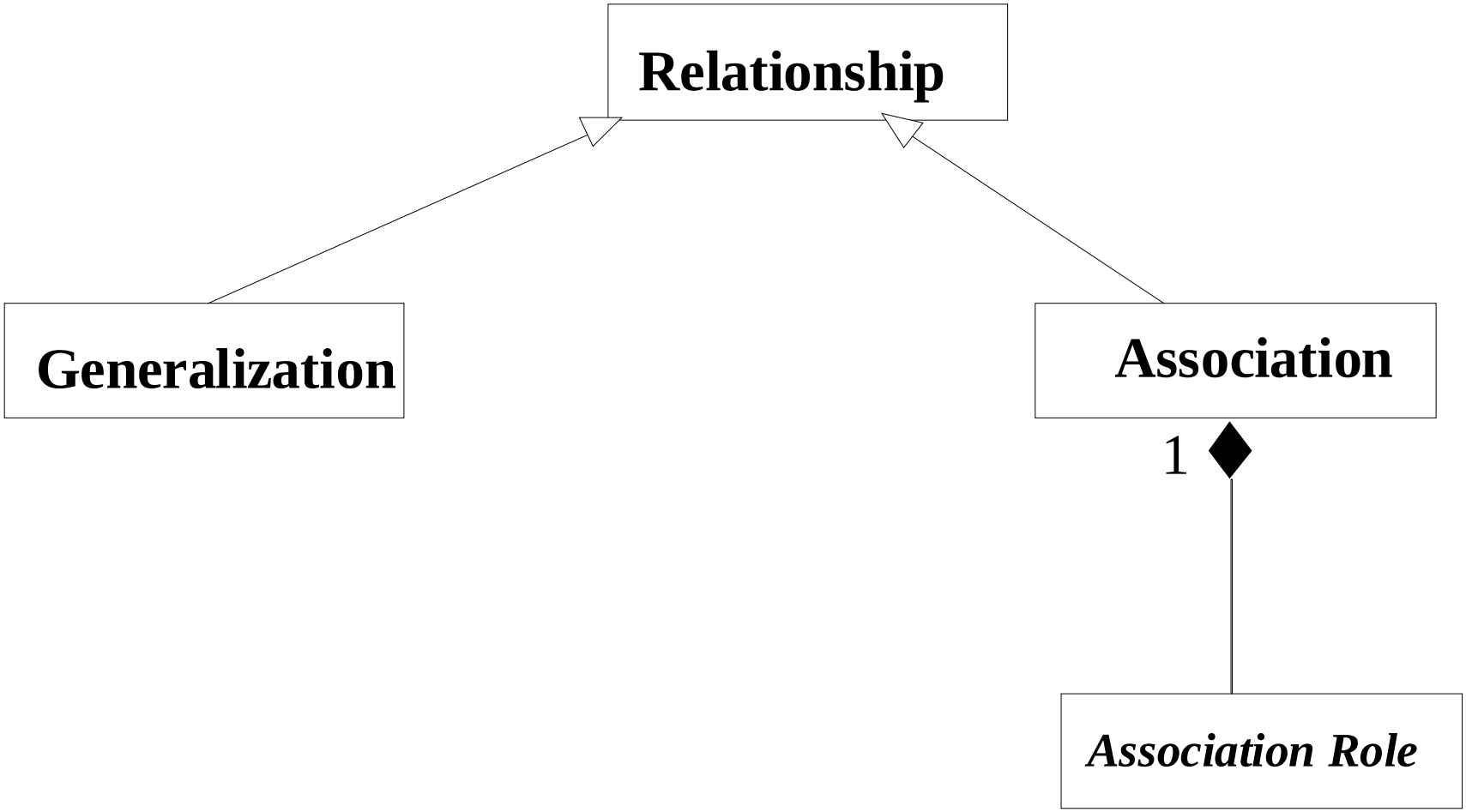
- ***Stereotypes* represent a built-in extendibility mechanism of the UML.**
- **User-defined extensions of the UML are enabled through the use of stereotypes and constraints.**



UML Meta-Model

- **A meta-model is a model of modeling elements.**
- **The purpose of the UML meta-model is to provide a single, common, and definitive statement of the syntax and semantics of the elements of the UML.**

UML Meta-Model (Con't)



Summary

- **A model is a simplified representation of reality.**
- **The unified modeling language (UML) was developed by Booch, Jacobson, and Rumbaugh and encompasses the unification of their modeling notations.**

Summary (Con't)

- **UML consists of the following diagrams:**
  - **Class diagram.**
  - **Use case diagram.**
  - **Sequence diagram.**
  - **Collaboration diagram.**

Summary (Con't)

- **Statechart diagram.**
- **Activity diagram.**
- **Component diagram.**
- **Deployment diagram.**