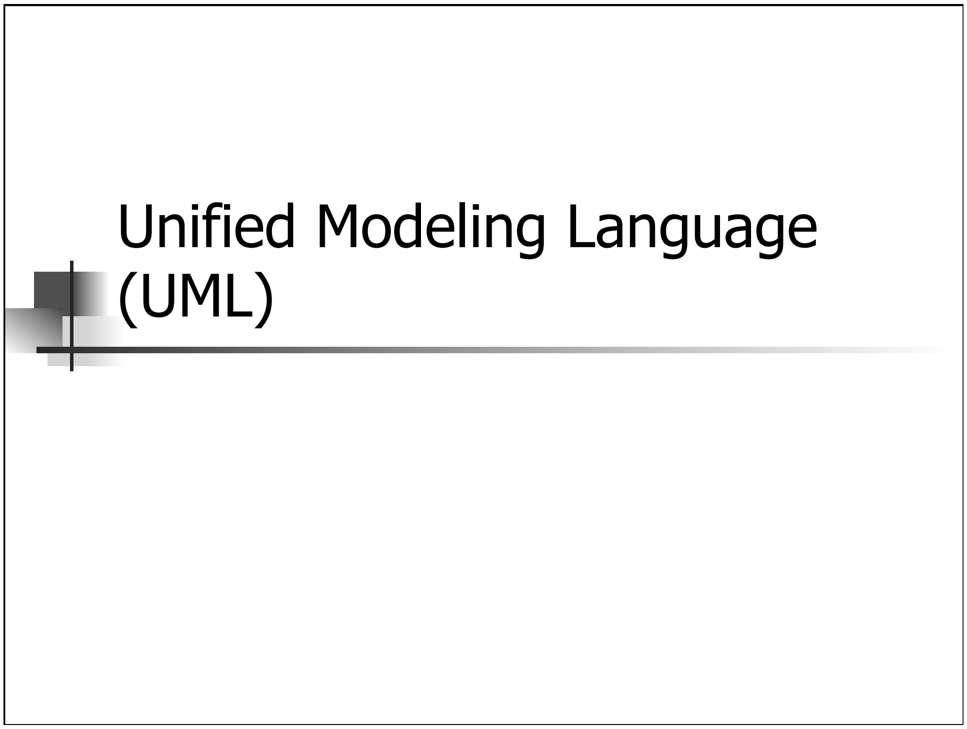
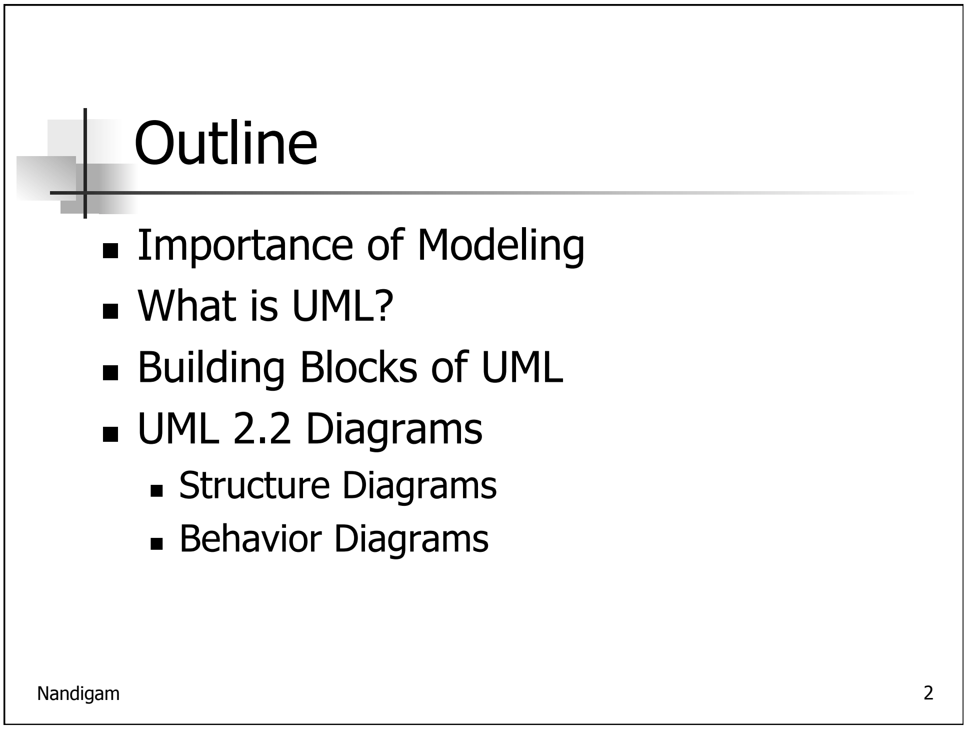
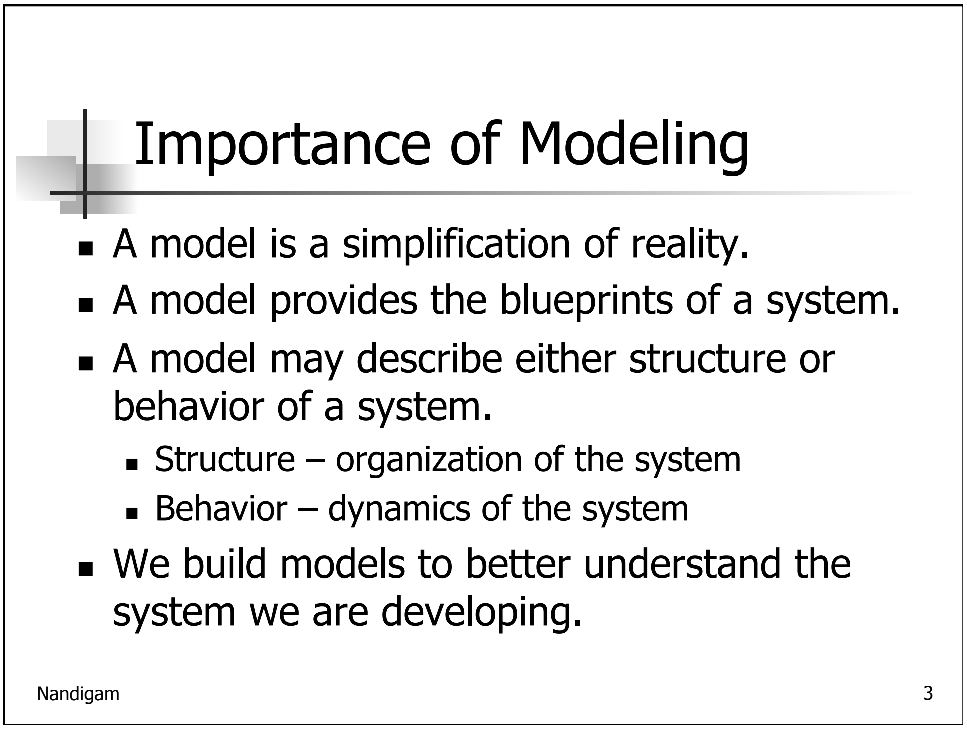


Unified Modeling Language (UML)



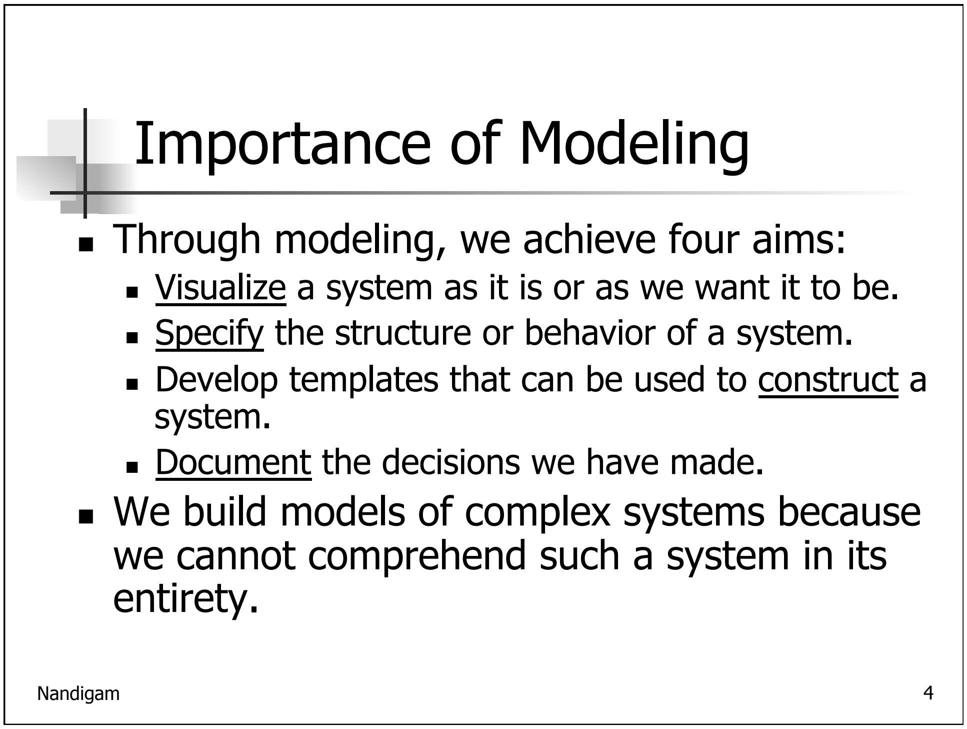
Outline

- Importance of Modeling
- What is UML?
- Building Blocks of UML
- UML 2.2 Diagrams
  - Structure Diagrams
  - Behavior Diagrams



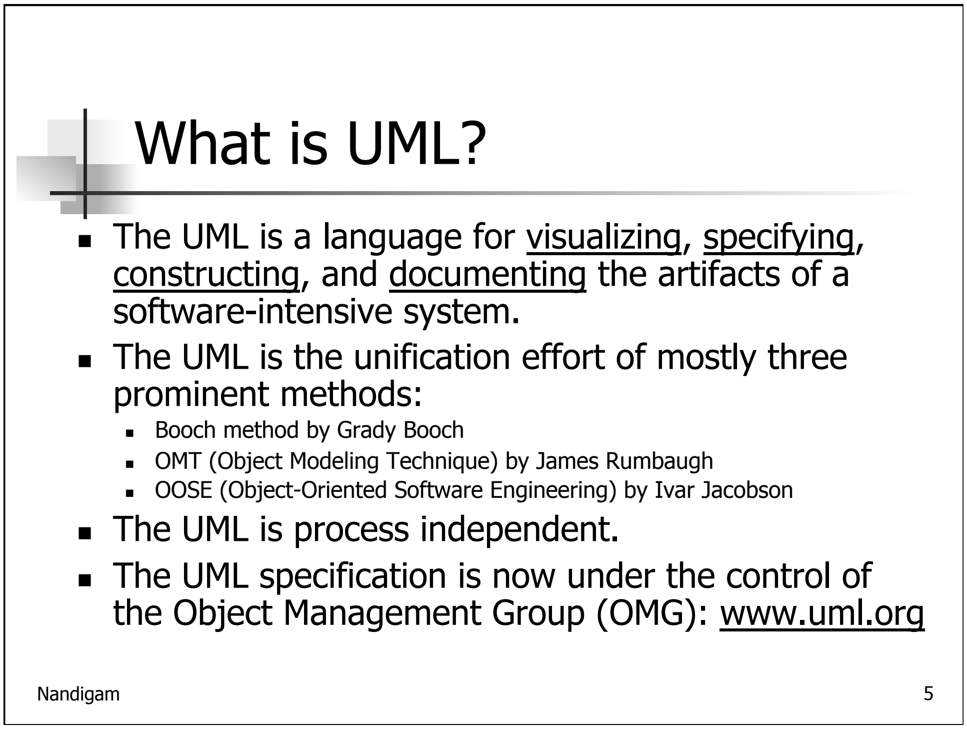
Importance of Modeling

- A model is a simplification of reality.
- A model provides the blueprints of a system.
- A model may describe either structure or behavior of a system.
  - Structure – organization of the system
  - Behavior – dynamics of the system
- We build models to better understand the system we are developing.



Importance of Modeling

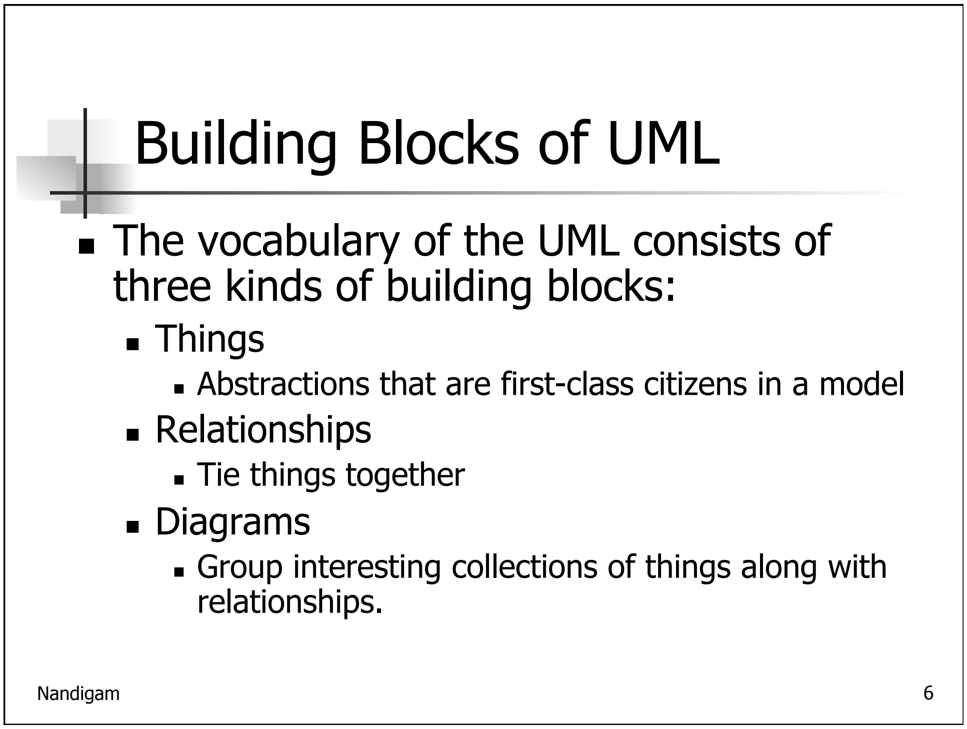
- Through modeling, we achieve four aims:
  - Visualize a system as it is or as we want it to be.
  - Specify the structure or behavior of a system.
  - Develop templates that can be used to construct a system.
  - Document the decisions we have made.
- We build models of complex systems because we cannot comprehend such a system in its entirety.



What is UML?

- The UML is a language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system.
- The UML is the unification effort of mostly three prominent methods:
  - Booch method by Grady Booch
  - OMT (Object Modeling Technique) by James Rumbaugh
  - OOSE (Object-Oriented Software Engineering) by Ivar Jacobson
- The UML is process independent.
- The UML specification is now under the control of the Object Management Group (OMG): www.uml.org

Nandigam 5



Building Blocks of UML

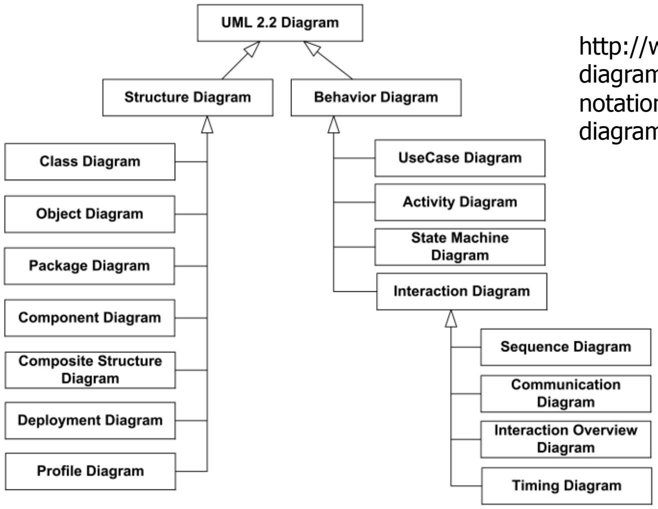
- The vocabulary of the UML consists of three kinds of building blocks:
  - Things
    - Abstractions that are first-class citizens in a model
  - Relationships
    - Tie things together
  - Diagrams
    - Group interesting collections of things along with relationships.

Nandigam 6

UML 2.2 Diagrams

- UML 2.2 defines fourteen types of diagrams, divided into two categories.
  - Structure diagrams
  - Behavior diagrams
- Structure diagrams show the static structure of the system and its parts at different abstraction and implementation levels and relationships between parts of the system.
- Behavior diagrams show the dynamic behavior of objects (and other things) in the system.

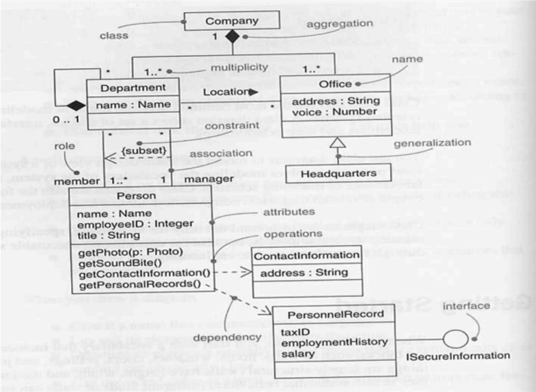
Diagrams in UML 2.2



http://www.uml-diagrams.org/notation/uml-22-diagrams.png

Class Diagrams

- A class diagram shows static model elements of a design such as classes and types, their contents, and their relationships.



Relationships in UML

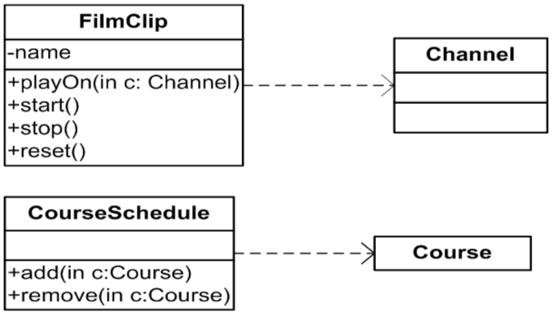
- A relationship is a connection between things.
- Graphically, a relationship is rendered as a path, with different kinds of lines to distinguish the different relationships.
- Kinds of relationships in UML
  - Dependency
  - Generalization
  - Association
    - Aggregation
    - Composition
  - Realization

Dependency Relationship

- A dependency is a using relationship between two things.
- It specifies that a change in specification of one thing may affect another thing that uses it, but no necessarily the reverse.
- Apply dependencies when you want to show one thing using another.
- Typical use of dependencies – one class uses another class as an argument in the signature of an operation or a local variable with a method.

Dependency Relationship

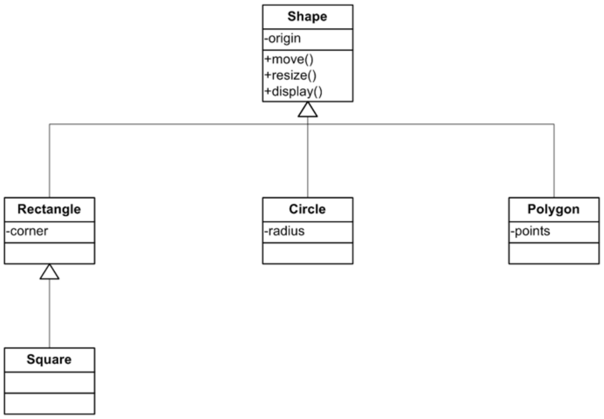
- Graphically, a dependency is rendered as a dashed directed line, directing to the thing being depended on.



Generalization Relationship

- A generalization is a relationship between a general thing (called the superclass or parent) and a more specific kind of that thing (called the subclass or child).
- Generalization is sometimes called an "is-a" or "is-a-kind-of" relationship.
- In generalization, the child is substitutable for the parent.
- You can also create generalizations among other things – packages, use cases, actors, etc.
- Graphically, generalization is rendered as a solid directed line with a large open arrowhead pointing to the parent.

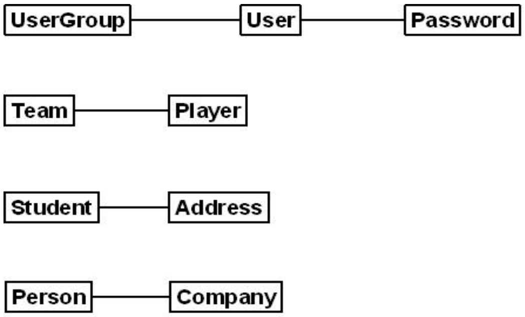
Generalization Relationship

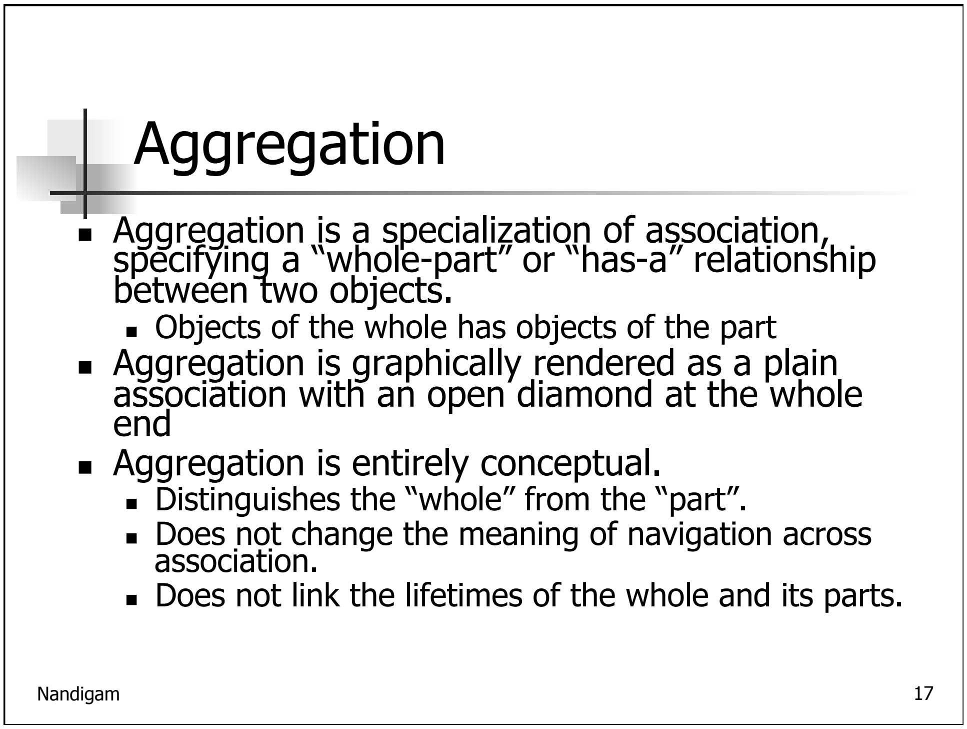


Association Relationship

- An association is a structural relationship that specifies that objects of one thing are connected to objects of another thing.
- Basic adornments that apply to an association
  - Name
  - Role at each end of the association
  - Multiplicity at each end of the association
  - Navigation (bidirectional assumed if not specified)
  - Aggregation (simple or composite)
- Graphically, an association is rendered as a solid line connecting the same or different classes.

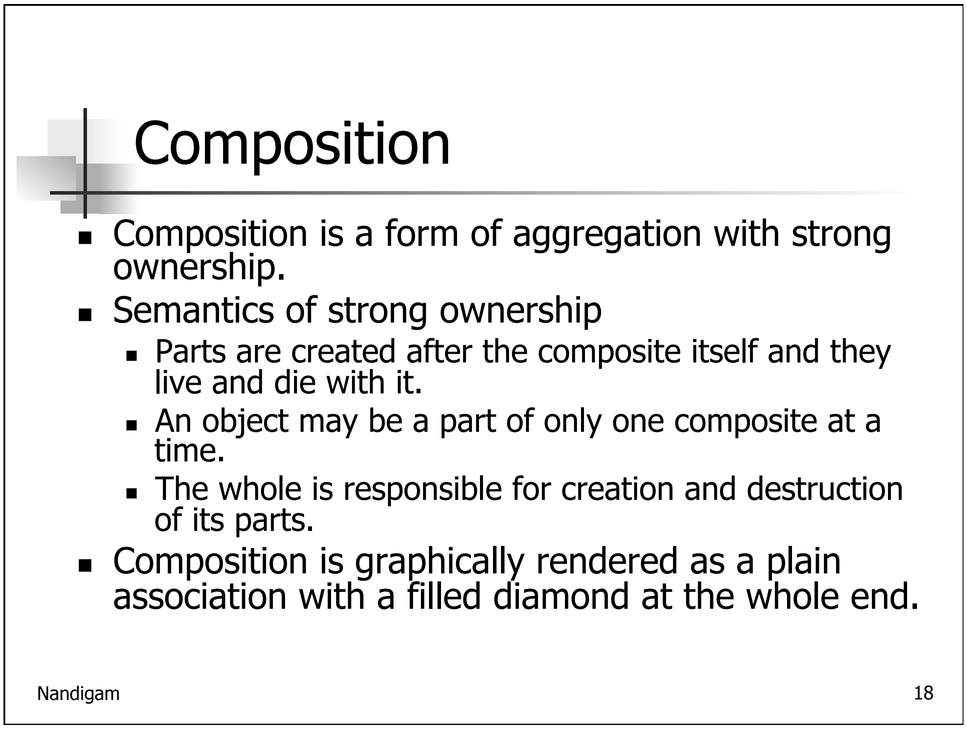
Association Relationship





Aggregation

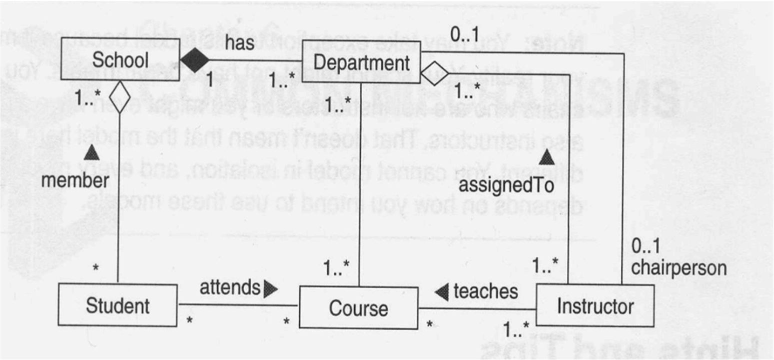
- Aggregation is a specialization of association, specifying a “whole-part” or “has-a” relationship between two objects.
  - Objects of the whole has objects of the part
- Aggregation is graphically rendered as a plain association with an open diamond at the whole end
- Aggregation is entirely conceptual.
  - Distinguishes the “whole” from the “part”.
  - Does not change the meaning of navigation across association.
  - Does not link the lifetimes of the whole and its parts.



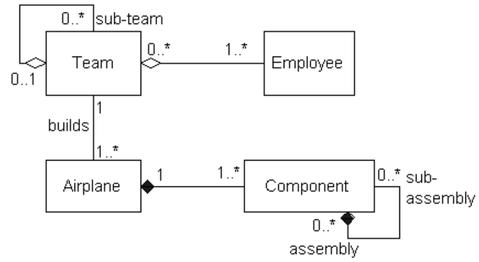
Composition

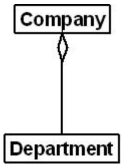
- Composition is a form of aggregation with strong ownership.
- Semantics of strong ownership
  - Parts are created after the composite itself and they live and die with it.
  - An object may be a part of only one composite at a time.
  - The whole is responsible for creation and destruction of its parts.
- Composition is graphically rendered as a plain association with a filled diamond at the whole end.

Associations – Examples

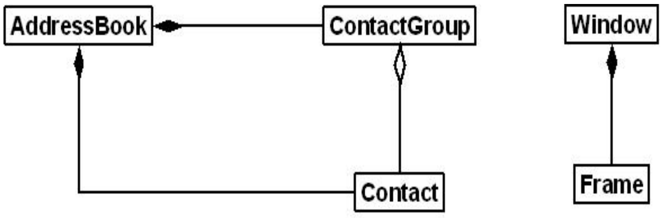


Associations – Examples



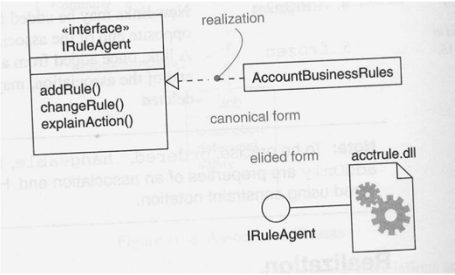


Associations – Examples



Realization Relationship

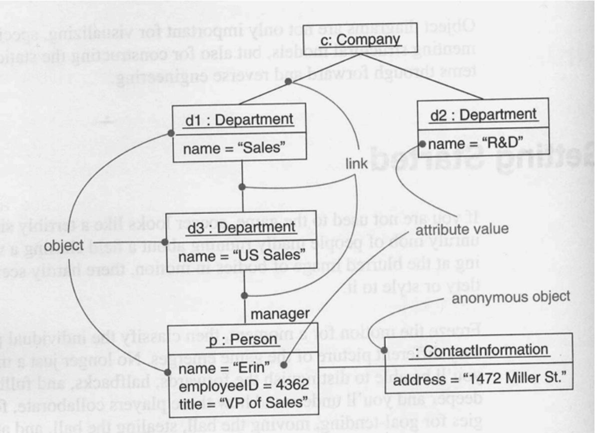
- A realization is a semantic relationship between classifiers in which one classifier specifies a contract that another classifier guarantees to carry out.
- Graphically, a realization is rendered as a dashed directed line with a large open arrowhead pointing to the classifier that specifies the contract.



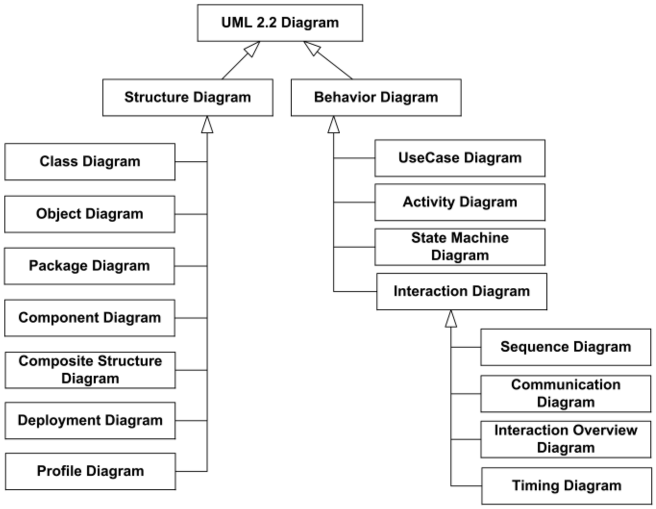
Object Diagrams

- An object diagram shows a set of objects and their relationships at a point in time.
- An object diagram covers a set of instances of the things found in a class diagram. It is essentially an instance of a class diagram.
- An object diagram expresses the static part of an interaction, consisting of objects that collaborate, but without any of the messages passed among them.
- An object diagram provides a snapshot of the objects in a system at a given moment in time.
- Typically used to model object structures.

Object Diagrams



Diagrams in UML 2.2



UML 2.2 Diagram

Structure Diagram

Behavior Diagram

Class Diagram

Object Diagram

Package Diagram

Component Diagram

Composite Structure Diagram

Deployment Diagram

Profile Diagram

UseCase Diagram

Activity Diagram

State Machine Diagram

Interaction Diagram

Sequence Diagram

Communication Diagram

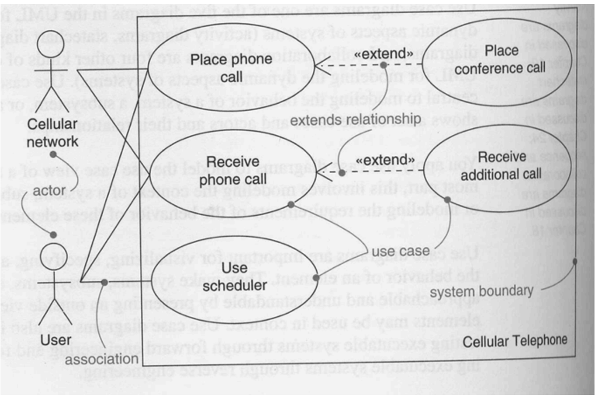
Interaction Overview Diagram

Timing Diagram

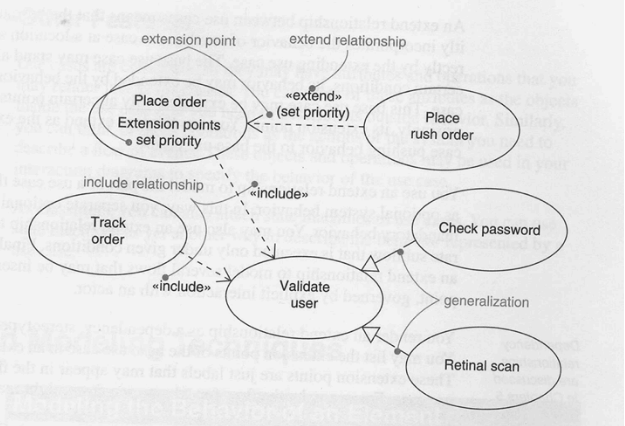
Use Case Diagrams

- A use case diagram shows a set of use cases and actors and their relationships.
- Use case and use case diagrams are typically applied in one of two ways
  - To model the context of a system
    - Draw a line around the whole system with use case inside the box.
    - Actors lie outside the system and interact with the system
  - To model the requirements of a system
    - Use cases specify the desired behavior of the system

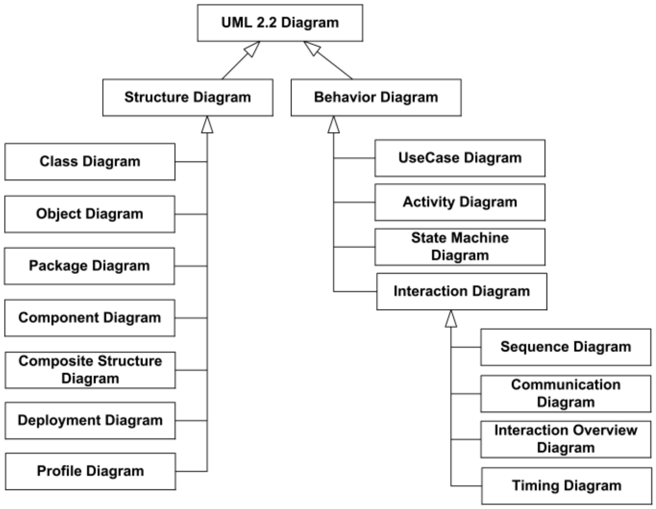
Use Case Diagrams



Use Case Diagrams



Diagrams in UML 2.2

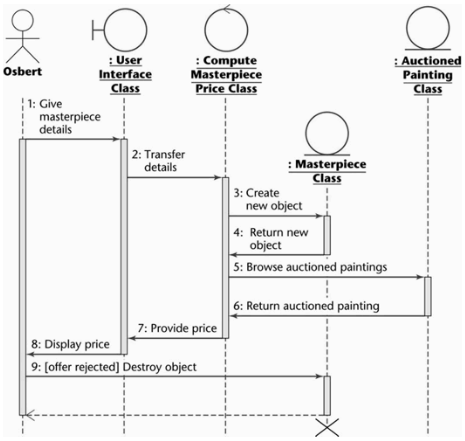


- UML 2.2 Diagram
  - Structure Diagram
    - Class Diagram
    - Object Diagram
    - Package Diagram
    - Component Diagram
    - Composite Structure Diagram
    - Deployment Diagram
    - Profile Diagram
  - Behavior Diagram
    - UseCase Diagram
    - Activity Diagram
    - State Machine Diagram
    - Interaction Diagram
      - Sequence Diagram
      - Communication Diagram
      - Interaction Overview Diagram
      - Timing Diagram

Sequence Diagrams

- A sequence diagram emphasizes the time ordering of messages.
- Graphically, a sequence diagram is a table that shows objects arranged along the x axis and messages, ordered in increasing time, along the y axis.
- Two features that distinguish sequence diagrams from collaboration diagrams
  - Object lifeline – a vertical dashed line that represents the existence of an object over a period of time.
  - Focus of control – a tall, thin rectangle that shows the period of time during which an object is performing an action.

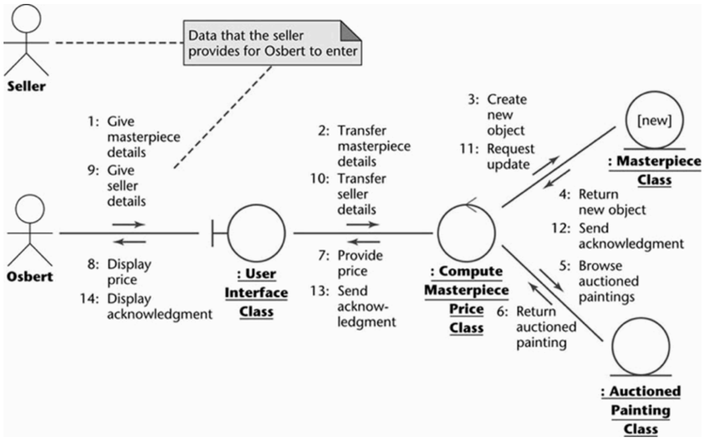
Sequence Diagrams


Osbert
: User Interface Class
: Compute Masterpiece Price Class
: Auctioned Painting Class
: Masterpiece Class
1: Give masterpiece details
2: Transfer details
3: Create new object
4: Return new object
5: Browse auctioned paintings
6: Return auctioned painting
7: Provide price
8: Display price
9: [offer rejected] Destroy object

Communication Diagrams

- A communication (aka collaboration) diagram emphasizes the organization of objects that participate in an interaction.
- Two features that distinguish communication diagrams from sequence diagrams
  - Path – To show how one object is linked to another, a path stereotype can be attached to the far end of a link.
  - Sequence number – To indicate the time order of a message, each message is prefixed with a number.
- Sequence and communication diagrams are semantically equivalent.
  - You can take a diagram in one form and convert it to the other without any loss of information.

Communication Diagrams


Seller
Data that the seller provides for Osbert to enter
1: Give masterpiece details
9: Give seller details
Osbert
8: Display price
14: Display acknowledgment
: User Interface Class
2: Transfer masterpiece details
10: Transfer seller details
7: Provide price
13: Send acknow-ledgment
: Compute Masterpiece Price Class
3: Create new object
11: Request update
[new]
: Masterpiece Class
4: Return new object
12: Send acknowledgment
5: Browse auctioned paintings
6: Return auctioned painting
: Auctioned Painting Class