# The Unified Modeling Language



## The Unified Modeling Language

The Unified Modeling Language (UML) is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct, and document the artifacts of a software-intensive system.

- Grady Booch
- James Rumbaugh (OMT)
- Ivar Jacobson (OOSE)



## Building Blocks of UML

- Things
- Relationships
- Diagrams



## Things

- Structural things
  - *classes, interfaces, collaborations, use cases, active classes, components, nodes.*
- Behavioral things
  - *interactions, state machines.*
- Grouping things
  - *packages.*
- Annotational things
  - *notes.*

## Relationships

- Dependency
- Association
- Generalization
- Realization



## Diagrams

- 1. Class diagram
- 2. Object diagram
- 3. Use case diagram
- 4. Sequence diagram
- 5. Collaboration diagram
- 6. Statechart diagram
- 7. Activity diagram
- 8. Component diagram
- 9. Deployment diagram



## Structural Things

Structural things are the nouns of UML models. These are the mostly static parts of a model, representing elements that are either conceptual or physical.



## Class

- Class
  - A class is a description of a set of objects that share the same attributes, operations, relationships, and semantics.
  - Attribute
    - An attribute is a named property of a class that describes a range of values that instances of the property may hold.
  - Operation
    - An operation is the implementation of a service that can be requested from any object of the class to affect behavior.

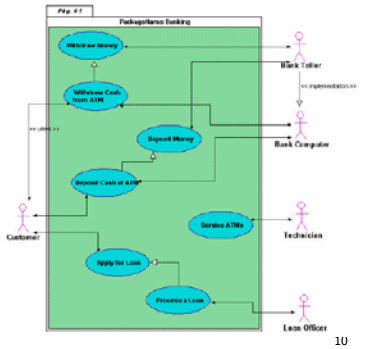
## Use Case

- Use case
  - A use case specifies the behavior of a system or a part of a system and is a description of a set of sequences of actions, including variants, that a system performs to yield an observable result of value to an actor.
  - Actor
    - An actor represents a coherent set of roles that users of use cases play when interacting with these use cases.



## Use Case Diagram

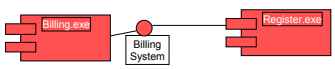
- Use case diagram
  - A use case diagram shows a set of use cases and actors and their relationships.






## Interface

- Interface
  - An interface is a collection of operations that specify a service of a class or component.

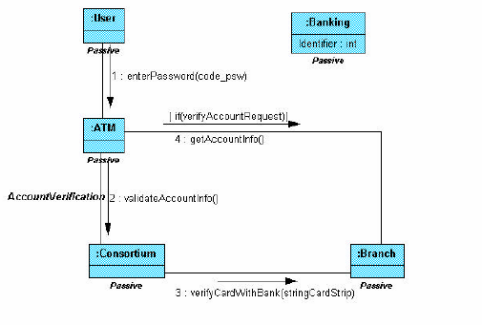





## Collaboration

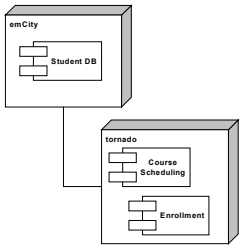
- Collaboration
  - Cross between an symbol diagram and a sequence diagram (interaction).
  - Describes a specific scenario
  - Numbered arrows show the movement of messages during the course of a scenario
- **When to use a Collaboration Diagram**
  - When you prefer to show a spatial organization of symbols and interaction rather than concentrating on the sequence of the interaction

## Collaboration Diagram






- Node
  - A node is a physical element that exists at run time and represents a computational resource.






## Behavioral Things

Behavioral things are the dynamic parts of UML models. These are the verbs of a model, representing behavior over time and space.



## Behavioral Things (cont'd)

- Interaction
  - An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose.
- State machine
  - A state machine is a behavior that specifies the sequences of states an object or an interaction goes through during its lifetime in response to events, together with its response to those events.

## Grouping and Annotational Things

Grouping things are the organizational parts of UML models.

- Package
  - A package is a general purpose mechanism for organizing elements into groups.

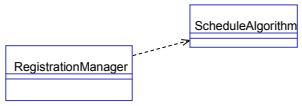
Annotational things are the explanatory parts of UML models.

- Note
  - A note is simply a symbol for rendering constraints and comments attached to an element or a collection of elements.



## Dependency

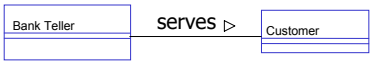
- Dependency
  - A dependency is a using relationship that states that a change in specification of one thing may affect another thing that uses it, but not necessarily the reverse.






## Association

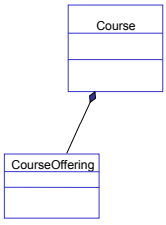
- Association
  - An association is a structural relationship that specifies that objects of one thing are connected to objects of another.





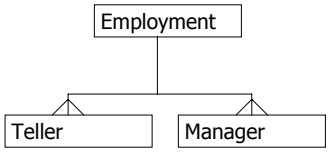

## Aggregation

- Aggregation
  - An aggregation is a special form of association that specifies a whole-part relationship between the aggregate (the whole) and a component (the part).



## Generalization

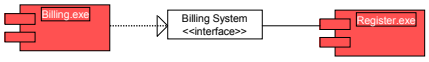
- Generalization
  - A generalization is a relationship between a general thing and a more specific kind of that thing. Sometimes it is called an "is-a-kind-of" relationship.






## Realization

- Realization
  - A realization is a semantic relationship between classifiers, wherein, one classifier specifies a contract that another classifier guarantees to carry out.






## Class and Object Diagram

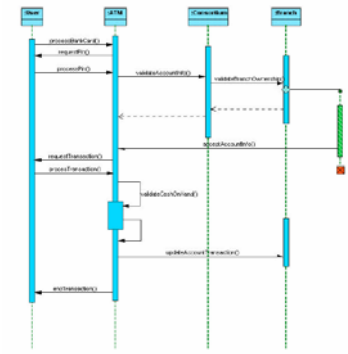
- Class diagram
  - A class diagram shows a set of classes, interfaces, and collaborations and their relationships.
- Object diagram
  - An object diagram shows a set of objects and their relationships.



## Sequence Diagram

- **Sequence diagram**
  - A sequence diagram is an interaction diagram that emphasizes the time-ordering of messages.
  - This diagram is a model describing how groups of objects collaborate in some behavior over time.
  - The diagram captures the behavior of a single use case.
  - It shows objects and the messages that are passed between these objects in the use case.
- **When to use a sequence diagram**
  - A good design can have lots of small methods in different classes. Because of this it can be difficult to figure out the overall sequence of behavior. This diagram is simple and visually logical, so it is easy to see the sequence of the flow of control.
  - A sequence diagram also clearly shows concurrent processes and activations.

# Sequence Diagram



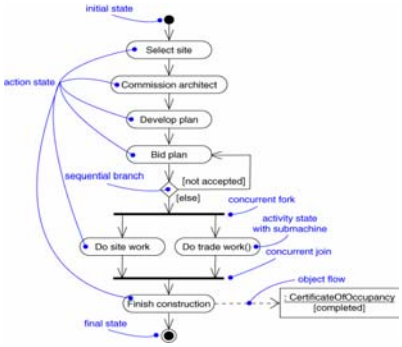


# State Chart Diagram

- Statechart diagram
  - A statechart diagram shows a state machine, consisting of states, transitions, events, and activities.
  - Provides a very detailed picture of how a specific symbols changes states.
  - A state refers to the value associated with a specific attribute of an object and to any actions or side
  - effects that occur when the attribute's value changes
- **When to use a State Diagram**
  - Used when you are working on real-time process control applications or systems that involve concurrent processing
  - When you want to show the behavior of a class over several use cases



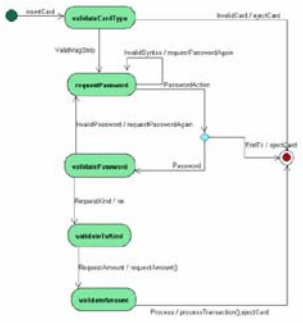
# State Chart Diagram





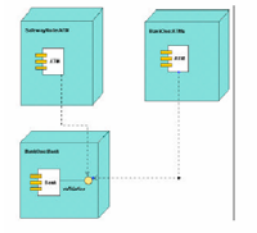

# Activity Diagram

- Activity diagram
  - An activity diagram is a special kind of a statechart diagram that shows the flow from activity to activity within a system.



# Deployment Diagram

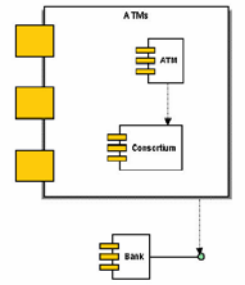
- A Deployment diagram shows the configuration of run-time processing elements and the software components, processes, and objects. Software component instances represent run-time manifestations of code units. Components that do not exist as run-time entities do not appear on these diagrams. These components should be shown on component diagrams.





# Component Diagram

- A component diagram shows the dependencies among software components, including source code, binary code and executable components. Some components exist at compile time, some exist at link time, and some exist at run time; some exist at more than one time.





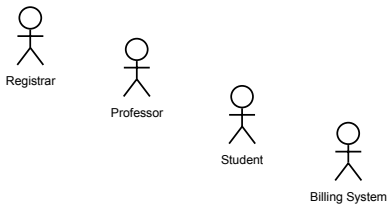

# Example

- An University wants to computerize their registration system
  - The Registrar sets up the curriculum for a semester
    - One course may have multiple course offerings
  - Students select 4 primary courses and 2 alternate courses
  - Once a student registers for a semester, the billing system is notified so the student may be billed for the semester
  - Students may use the system to add/drop courses for a period of time after registration
  - Professors use the system to receive their course offering rosters
  - Users of the registration system are assigned passwords which are used at logon validation



# Actors

- An actor is someone or some thing that must interact with the system under development

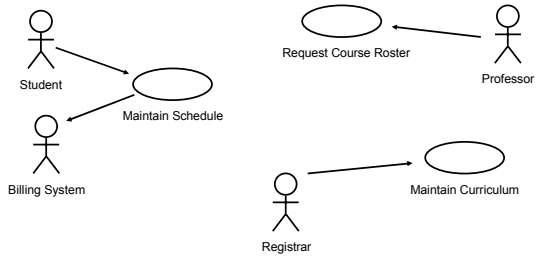


# Use Cases

- use case is a pattern of behavior the system exhibits
  - Each use case is a sequence of related transactions performed by an actor and the system in a dialogue
- Actors are examined to determine their needs
  - Registrar -- maintain the curriculum
  - Professor -- request roster
  - Student -- maintain schedule
  - Billing System -- receive billing information from registration

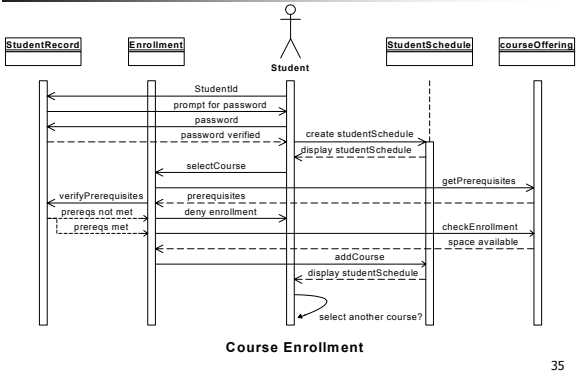



# Use Case Diagram





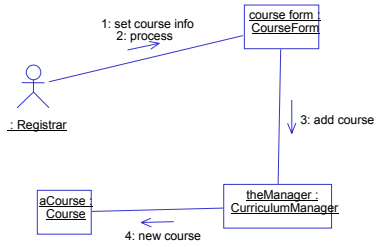

# Sequence Diagram



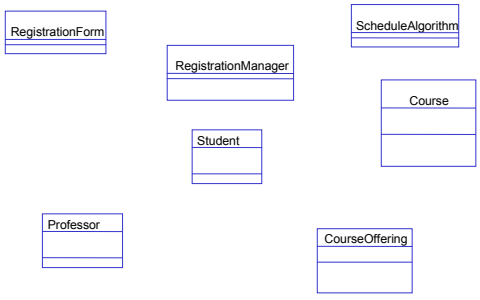

**Course Enrollment**



# Collaboration Diagram

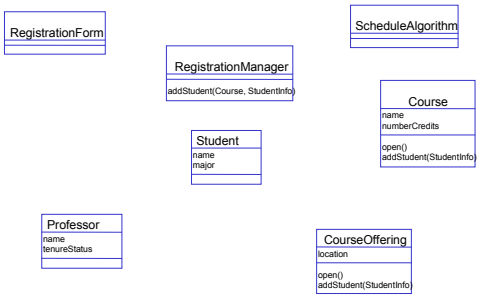


# Classes



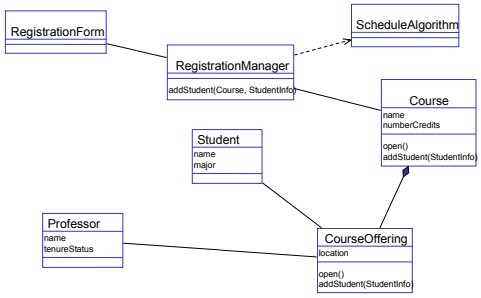



# Classes



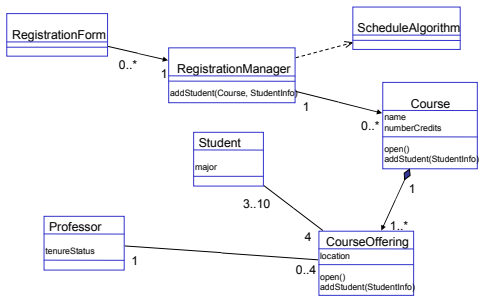



# Relationships

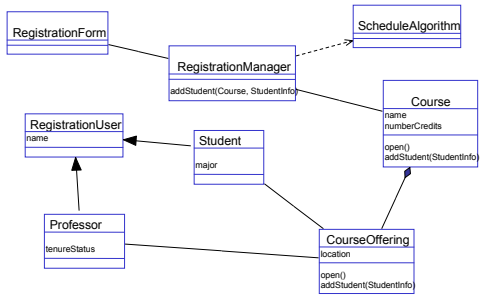





# Multiplicity and Navigation

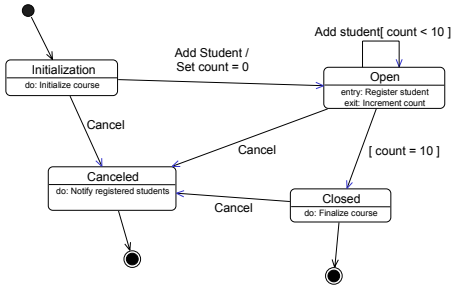


# Inheritance



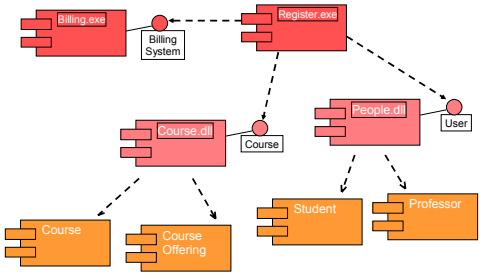



# State Transition Diagram






# Component Diagram






# Deployment Diagram