What is UML

- Unified Modeling Language

- A language for modellng software systems from requirements to specification

- The goal is to become a common language for creating models of object oriented computer software
Benefits of UML

• You know exactly what you are getting
• You will have lower development costs
• Your software will behave as you expect it to. Fewer surprises
• The right decisions are made before you are given poorly written code. Less overall costs
• We can develop more memory and processor efficient systems
• System maintenance costs will be lower. Less relearning takes place
• Working with a new developer will be easier.
• Communication with programmers and outside contractors will be more efficient
Types of UML Diagrams

• Use Case Diagram
  – Description of a system’s behavior from a user’s point of view

• Class Diagram
  – Models class structure and contents using design elements such as classes, packages, and objects
  – Displays relationships such as containment and inheritance
Types of UML Diagrams

• **Sequence Diagram**
  – Shows the time-based dynamics of the interaction between objects
  – Two dimensions; time and different objects

• **Collaboration Diagram**
  – Displays the interaction organized around the objects and their links to one another
  – Numbers are used to show the sequence of messages
Types of UML Diagrams

• State Diagram
  – Displays the sequences of states that an object of an interaction goes through during its life in response to received stimuli

• Activity Diagram
  – Displays a special state diagram where most of the states are action states and most of the transitions are triggered by completion of the actions in the source states
  – Like a flowchart
Types of UML Diagrams

• **Component Diagram**
  – Displays the high level packaged structure of the code itself
  – Dependencies among components are shown, including source code, binary code, and executable components

• **Deployment Diagram**
  – Displays the configuration of run-time processing elements and the software components, processes, and objects that live on them
Use Case Diagrams

• Use case diagrams show how a system’s users interact with it
  – i.e. the system’s requirements

• Use case diagrams represent:
  – **Actors**: things (often people) outside the system that interact with it
  – **Use cases**: tasks the system supports
  – **Associations** between the two
Use Case Diagram Example

- Patient
  - Make appointment

- Doctor
  - Perform medical tests
Use Case Diagrams

• Used in almost every project

• Helpful in exposing requirements analysis and planning the project

• During the initial stage of a project most use cases should be defined, but as the project continues more might become visible
Use Case Diagram Example

Evolution of a UML Use Case Diagram

Drive Car

... Becomes ...

Drive Car

<<uses>>

Brake

<<uses>>

Turn

... Which Becomes ...

Drive Car

<<uses>>

<<extends>>

Drive Ambulance

Brake

<<uses>>

<<extends>>

Turn

<<extends>>

Turn Left

<<extends>>

Turn Right
Initial Design:

Ticket Clerk

Reservation System
- Check in Passenger
- Add Reservation
- Cancel Reservation

Sub-Diagram:

Check in Passenger
- Weigh Luggage
- Assign Seat

To add detail (extension):

Assign Seat
- Assign Window Seat
- Assign Aisle Seat

Weigh Luggage

Check in Passenger
- <<uses>>
- <<uses>>

Assign Seat
- <<extends>>
- <<extends>>

Assign Window Seat
- <<extends>>

Assign Aisle Seat
- <<extends>>
Class Diagrams

• Widely used to describe the types of objects in a system and their relationships
• Model class structure and contents using design elements such as classes, packages and objects
• Describe three different perspectives when designing a system; conceptual, specification, and implementation
Class Diagrams

- Classes are composed of three components:

  - **Class**
    - The class name typically has the first alphabet capitalized. If your class has more than one word, and capitalize the first alphabet of both words and join the two. For e.g.: Student

  - **List of Attributes / Variables**
    - A list of attributes of your class goes in here. The syntax is: attribute : Type = "default value (if any)"
      - For e.g.: studentId : int OR
      - studentName : String

  - **List of methods**
    - A list of your methods goes in here. The syntax is: MethodName(List of parameters (if any)): Return type (if any)
      - For e.g.: String get Student Name(int studentId)
      - Notation: Hungarian Notation
Class Diagrams

• Used in nearly all Object Oriented software designs

• Used to describe the classes of the system and their relationships to each other
Class Diagrams

• Relationships Between Classes
  – Inheritance

  – Composition

  – Association
Class Diagram Example

**Person**
- Name
- Phone Number
- Email Address
- Purchase Parking Pass

**Address**
- Street
- City
- State
- Postal Code
- Country
- Validate
- Output As Label

**Student**
- Student Number
- Average Mark
- Is Eligible To Enroll
- Get Seminars Taken

**Professor**
- Salary
Class Diagrams

• Use of templates, interfaces, and types

• Can even specify body of methods
Sequence Diagrams

• Dynamic model view

• Details how operation are carried out, what messages are sent and when

• Two dimensions:
  – Time
  – Objects
Example Sequence Diagram

- **caller**
  - a: lift receiver
  - b: dial tone
  - c: dial digit
  - d: route
  - ringing tone
  - stop tone

- **exchange**
  - stop ringing

- **receiver**
  - phone rings
  - answer phone

The call is routed through the network.

{b.receiveTime - a.sendTime < 1 sec.}

{c.receiveTime - b.sendTime < 10 sec.}

{d.receiveTime - d.sendTime < 5 sec.}

At this point the parties can talk.
Collaboration Diagrams

• Give the same information as sequence diagrams but they focus on object roles instead of the times that messages are sent

• Object roles are vertices and messages are the connecting links

• Each message has a sequence number
State Diagrams

• Shows the possible states of the object and the transitions that cause a change in state

• Initial state is a dummy to start the action

• Final states are also dummy states that terminate the action
State Diagram Example

Proposed

Scheduled

Open For Enrollment

entry/ logSize()

Full

enroll student / addToWaitingList(); considerSplit()

Closed to Enrollment

entry/ notifyInstructor()
Example State Diagram
Activity Diagrams

• Similar to state diagrams
  – Activity diagram focuses on the flow of activities involved in a single process
  – State diagram focuses attention on an object undergoing a process

• Is essentially a fancy flowchart
Activity Diagram Example

1. Manage Course
   - Course exists
     - New course
     - Modify course
   - Remove course
2. Create Course
3. Modify Course
4. Remove Course
5. :Course [completed]
UML in Real Practice

• You don't typically use *all* the diagrams
  – You'll choose between them based on preference and particular situation

• You typically use *many* diagrams
  – A single use case may not capture all scenarios
  – If you are going to use statecharts, there are probably *lots* of objects with states
  – Each sequence/collaboration diagram only shows *one* interaction
Example: Student Registration System

• Not going to do all the diagrams
  – Not all types, not even all that completely specify the system
• But this is an application you know, so the examples may help make sense
Partial Use Case Diagram

- Apply for Admission
  - Enroll in the University
    - Enroll in a Course
    - Withdraw from a Course
  - Admissions
- Student
States of a Student

Apply [ Must be accepted first ]

Enrolled

EnrollInClass ( Add a Transcript )

Registered

Withdraw

AddCourse

Graduate [ All courses must be completed ]

Registered

Withdraw

AddCourse

Graduate [ All courses must be completed ]
Sequence Diagram: Registering for Course

getSectionsFor:

return sections

enrollInSection:

takenCourse: prerequisite

have prereq

enrolled

addStudent:

tenakedCourse: prerequisite

state of prereq

enrolled

enrolled