



Introduction to UML

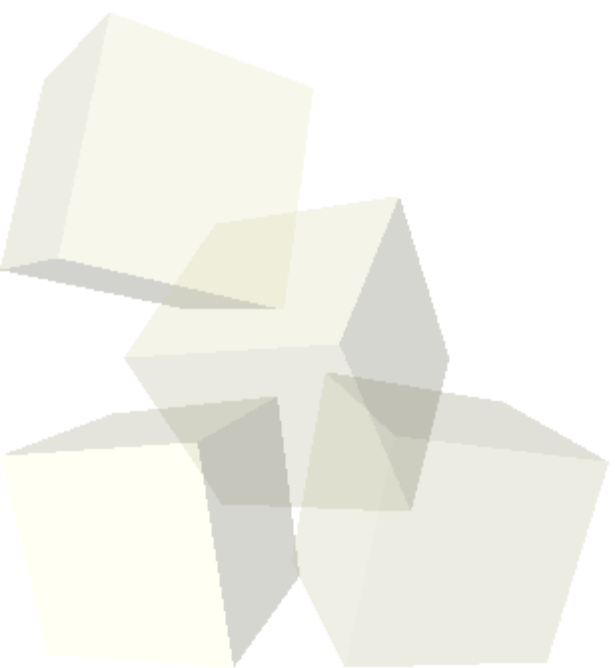
1/28/2008





Why UML?

- What is the purpose of UML? Why was it created? Why should you know it?
- What do you know currently about UML? In what situations have you used it?





What is UML

- UML stands for Unified Modeling Language. The name itself tells us something about it and its history.
- It is a modeling language. This implies that it is a formal method of modeling programs. In practice, it can go beyond just the original modeling use though.
- Why is it unified? Prior to UML people had developed a large number of different modeling languages that did nearly the same thing in different ways.



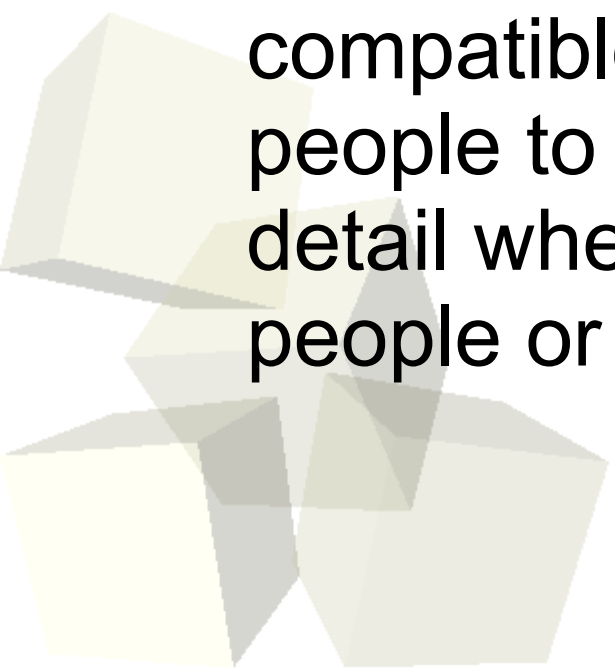
Beyond Modeling

- While UML was originally created to allow us to use formal drawing to model programs and aid with their analysis and design, it has other uses as well.
- UML is also a communication tool. We can all understand code by reading it, but reading the code for a large project is very difficult. UML gets key information across easier.
- UML can work as an architecture tool that allows us to see problems at high levels.
- UML 2.x was designed to allow code generation.



Easy to Draw

- One of the big keys to UML is that all the elements of it are easy to draw. You don't have to be a great artist to create UML diagrams
- You could also say the UML is “whiteboard compatible”. This implies that it is easy for people to sketch UML to varying degrees of detail when communicating ideas to other people or trying to develop ideas.





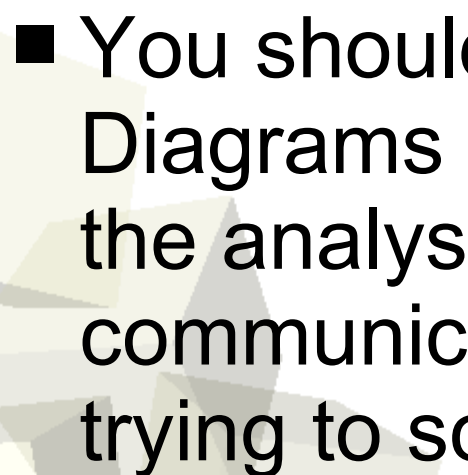
Multiple Diagrams

- Anyone who has taken PAD2 has seen UML class diagrams. However, UML is much more than just class diagrams.
- Version 2.0 of UML includes 13 different types of diagrams
 - ◆ Use Case, Activity, Class, Object, Sequence, Communication, Timing, Interaction, Composite, Component, Package, State Machine, and Deployment





Things You Already Know

- Each of you should already know about Class Diagrams and what they show us. These are the staple of design work and allow us to visualize class hierarchies and see dependencies in our code.
 - You should also all know about Use Case Diagrams and the fact that they are used in the analysis process to help define and communicate exactly what problem we are trying to solve.
- 



Things You Need To Learn

- Obviously you could learn all the different UML diagrams and that wouldn't be a bad thing. In practice you need to know the ones you need, but having an idea about all of them helps you determine which ones you need.
- More importantly, you need to practice with whatever ones you use so that you can get a feeling for them. All the diagrams can look pointless or confusing if you don't understand them and use them well.



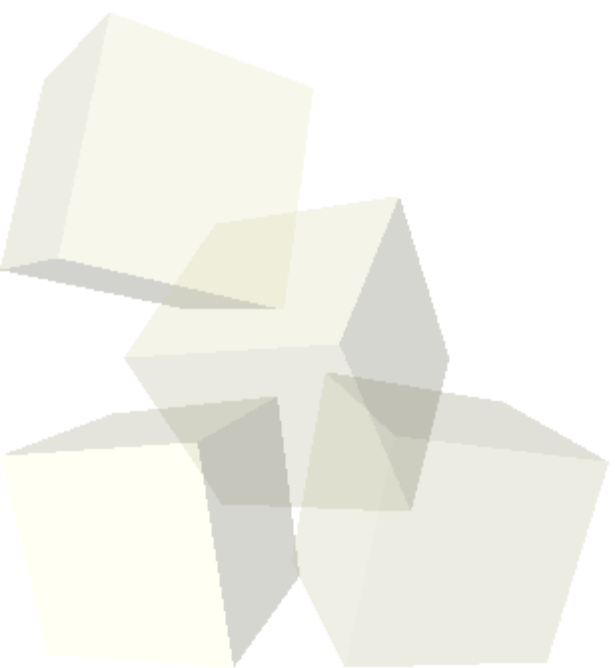
Tools

- There are lots of tools for doing UML diagrams. Different tools provide different support. Unfortunately, this is one area where good tools rarely come cheap.
 - ◆ ArgoUML is free and open source, but doesn't yet support UML 2.x. NetBeans now has UML.
 - ◆ Together and Rational Rose are industry standards with hefty price tags.
 - ◆ Poseidon is widely used, no longer completely free.
 - ◆ Visual Paradigm has a free Community Edition as does EclipseUML from Omondo.



Class Diagrams

- Let's review the basics of class diagrams and what they can tell us.





Use Case Diagrams

- In a real software development situation, the diagram that you start with should not be a class diagram, but a use case diagram. These are used in the analysis phase of developing a project. They help you determine what exactly your product does.
- The diagram consists of actors outside the system, the system, use cases, and connections showing relationships.
- These diagrams must be accompanied by use case descriptions.