Team / Design discussion
UML Overview
Project #4 discussion

- Classroom discussion about design aspects.
What is UML?

- A modeling language standardized by the OMG (Object Management Group), and widely used in OO analysis and design
  - A modeling language is a visual language for representing software blueprints
  - Latest version is 2.0

- UML provides a standard notation, semantics for a set of OO abstractions
UML is a Modeling Language

- Modeling often refers to a (graphical) representation of software systems at higher levels of abstraction than code
  - UML has some “code-like” features.
- Modeling supports analysis and design of software
  - Modeling typically is used from early parts up to implementation
  - Sometimes during/after implementation for “reverse engineering”
- Modeling is used when building complex software systems
  - Something simple you don’t need it
    » Carpentry on something simple, can be done without blueprints
  - Once you get complicated enough
    » Then you need to fall back onto abstraction to help you understand it
Modeling

- Modeling is a growing, evolving area
- Modeling tools a part of every major IDE
  - Eclipse, JBuilder, Visual Studio. Net, etc.
  - Modeling is a multi-billion market
  - Projected to undergo more growth in the next 5-7 years
- UML 2.0 now considered by many to be “most successful” OMG standard ever
  - Well, it gets the most noise, anyway 😊
  - Even MS sees modeling as important
    » But naturally they have a different take than the rest of industry
Modeling

- Modeling is a “potential paradigm shift” in software development
  - Now: Models are a view on code
    » Often used as a “sketch”
  - Later(?): Code and models mix (code generation, toolkits)
  - Future(??): Domain Specific Modeling Languages, Executable Models

- Lots of people believe that modeling could become a large force in software development
Modeling Can Also Be Used For Exploration

- Models are all about abstraction
- Sometimes it is exactly what you want to help you understand something
- Otherwise you just root around in a bunch of source code
- Example:
What the...

- This is a section of a class library from SCIRun
  - 2D Core for SCI Run environment
  - SCI – Scientific Computing Institute
  - SCIRun – their main tool for doing SC analysis

- Model was created by reverse engineering C++
Okay, What’s the Point?

- Okay, you are in SCI working on SCIRun. You have to get to know the code.
- Would you:
  - A: Browse though the files to look at the code, using the directory structure as a guide
  - B: Browse though class models, being able to click on a class to view code
- With modeling, you can choose B to start with and always do A if you have to.
- SCIRun may have 100s of diagrams to look over
  - But, it’s 100,000s lines of code!
UML: History

- When OO was evolving, multiple graphical notations existed for OO design methods
  - Booch, OMT, Objectory, Shaller-Mellor, CRC
  - This is not good. Standard notation good.
  - Lots of, err, “debate” in the area.

- Three creators of popular OO notations ended up at Rational Software (early 90s)
  - The Three Amigos: Booch, Rumbaugh (OMT), Jacobsen (Objectory)
What is the UML?

- The UML is the standard language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system.
**UML: The History**

- Modeling designers came together to create what they termed the “Unified Modeling Language”
  - Created a initial version, 0.8 to 1.0 (mid 90s)
- Decided then to submit UML to OMG to standardize version 1.1
  - Big step forward. Many companies came together, contributed.
- Long, tough process (finished in 97).
The UML standard was refined and evolved somewhat in minor versions
  - UML 1.4 somewhat different from 1.1, for example. (97-2000)

But, major issues remained to be addressed.
  - Now that there was experience with using it, there was feedback and ideas to be analyzed.
  - “Time to do some more damage!”
**UML: Version 2.0**

- Major revision of the standard was started
  - Major internal changes, fewer external changes.
    - Our own Nathan Dykman was the HP representative in the UML 2.0 standards work
  - UML 2.0 standardized in 2005
  - Full UML 2.0 tools are available, but no tools for checking compatibility

- Need 2.2 Visio Templates
  - These are ok, but designed to DRAW with
  - No code generation
  - No checking that you are making illegal diagrams
UML: In a Nutshell

- A standard notation for models (diagrams) for various aspects of software systems.
- Two major types of models:
  - Static Models (Structure)
    » Use Cases, Classes, Composite Structure, Component, Deployment
  - Dynamic Models (Behavior)
    » Interaction (Sequence), State, Activity, Protocol
UML: Under The Shell

- UML is described largely by a metamodel.
  - A model of a model
- UML is extensible (Stereotypes, Profiles)
  - Refining existing concepts, adding new ones (lots of controversy on this point)
  - UML tries to adapt to your way of doing modeling
  - Extensibility in UML 2.0 is a hot issue
Dynamic Models

• To review, there are two major types of models in UML

• Static models
  – Use Case, Class, Component, Deployment
    » Composite structure (mix of static/dynamic)

• Dynamic models
  – Sequence, State, Activity
Dynamic Models

- Dynamic modeling is often overlooked in the design process
  - Why is this, do you think?

- Using dynamic models comes down to balance:
  - Too little: Critical dynamic details may be missed
  - Too much: Overwhelming detail

- Dynamic models are best used in two ways
  - Exploratory: Does this design do what we want, how could it do it?
  - Descriptive: This is a tricky aspect of the system, this model (should) help explain what is going on
Dynamic Modeling Traps

- Unfortunately, dynamic models are easy to overuse
  - For example, requiring a state diagram for every class
    » Not surprising, most of these diagrams aren’t helpful
  - Or activity diagram, or sequence diagram, etc.

- Therefore, some avoid them completely
  - Missing out where they are useful.
**Example Class Diagram**

- **Class Name**: Vehicle
- **Attributes**:
  - TopSpeed : int
  - Go()  
  - Stop()  
- **Operations**:
- **Generalization**
- **Composite**
- **Association**

**Vehicle**

- **Car**:
  - seats : int
  - Start()  

**Wheel**

- **Person**:
  - driver
  - passengers

- **Relationships**: 4 wheels, 1 driver, * passengers
This UML Diagram shows us that:

- We have four classes, and their structure and relationships to each other.

There are other features we don’t show, but we will get into later

- Template Classes, Interfaces, Association Classes
- Aggregation, Dependency, Realization
- Covered in Chap. 3 in UML Distilled.
Example Sequence Diagram

1: [1..*] \AddItemToCart\  

2: \IsAvailable\  

3: [Inventory = 0] \Order\  

4: \CheckOut\  

5: \ShipTo\  

6: \ShipItem\
Sequence Diagram

- Again, if we know UML notation, we now this is a interaction of class instances
  - We will go into more detail into what this diagram means later.
- But, the idea is that it gets an idea across
  - In this case, a sequence of messages between entities.
  - Chapter 4 in UML Distilled.
Example Activity Diagram

- Start Car
  - [GasLight=On] Get Gas
  - Drive to work
    - Park Car
      - Drink Coffee
      - Walk to Office Door
        - Work
Activity Diagram

- Again, if we know UML, we know we have a set of activities, and how they relate to each other
  - Activity Diagrams are a good way of showing flow of various things
  - Do you see a problem in this diagram?
  - Covered in Chapter 11 in UML Distilled.
Package Diagram

Vehicle

Car
- seats : int
+ Start( )
Package Diagram

• A Package just groups elements into a hierarchy
  – Packages can contain other packages, elements, etc
  – Packages can import or extend other packages
    » Import is the main relationship right now
  – Much like namespaces
About These Examples

- These are commonly used diagrams
  - Component Diagrams common, but much like Class Diagrams
- For most, the easiest to “grasp” at first
  - Oddly, Sequence and Activity Diagrams are some of the most complex models in UML
    » State Diagrams quite complex as well
The Core of UML

- The main features of UML:
  - It provides a standard set of elements
    » With a standard, graphical syntax.
  - It defines how those elements can relate to other elements (in a strict manner)
  - The elements have a semantic meaning that is often useful to software designers
  - It allows for elements to be extended and for new elements to be added (Extensibility)
An important part of UML is extensibility

- The ability to specialize certain elements via “stereotyping” (Lightweight extension)
- The ability to add new elements via “metamodelling” (Heavyweight extension)

Example follows
UML Extensibility Example
Extensibility

- We have three stereotypes of a class here
  - Boundary
  - Entity
  - Control

- We can attach additional data and constraints to stereotypes
  - For example, we could say that Boundary classes cannot be associated with Entity Classes
UML Stereotypes

- Stereotypes are best thought of as "decorators"
  - GOF pattern: Dynamic addition of state/behavior to a class
- They indicate that the element has additional (but compatible) semantic meaning
  - A <<boundary>> class is a special kind of UML class
- UML 2.0 changed the nature of extensibility a great deal
What To Take Away

- **Process** is how to organize the development of software
  - Lots of different ways of doing process, all benefits and disadvantages
- **Modeling** provides a way to abstract about software
- **UML** is a visual modeling language
  - Standardized by the OMG
  - UML has Static and Dynamic Models
  - What are the “core” diagrams that are used the most?
  - UML is extensible
- **Next up**: read chapter 9 of UML Distilled