

# INF 111 / CSE 121: Software Tools and Methods

Lecture Notes for Fall Quarter, 2007  
Michele Rousseau  
Set 15

(Some notes adapted from Susan E. Sim & UML Distilled)

---

---

---

---

---

---

---

---

## Announcements

- **Quiz #3- Next Friday 11/9**
  - All readings assigned since the last quiz
  - Plus the readings not covered on the last quiz
    - Ch 2 from "The Mythical Man-Month"
    - Van Vliet Ch. 4
  - Lectures from 10/31 – 11/7
- **Updated Assignment #2 – zip file**
- **Reminder**
  - Read: Van Vliet Ch. 12
    - Other info on UML that might be useful:
      - [http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML\\_tutorial/](http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/)
      - Argo UML Info:
        - <http://argouml.kennis.org/>
    - Some books on UML:
      - Fowler (2004). UML Distilled: Third Edition: A Brief Guide to the Standard Object Modeling Language, Addison-Wesley, 2004
      - Larman (2005). Applying UML and Patterns, Third Edition, Prentice Hall PTR, 2005

Topic 15

2

---

---

---

---

---

---

---

---

## Previously in INF 111...

- **Configuration Management**
  - For more info read Ch 19 on SCCS

Topic 15

3

---

---

---


---

---

---

---

---



## Today's Lecture

- Object Oriented Analysis & Design (OOAD) using...
  - UML – Part 1
    - Overview
    - More details in discussion

Topic 15 4

---

---

---


---

---

---

---

---



## Brooks on Invisibility of Software

"Software is *invisible* and *unvisualizable*. Geometric abstractions are powerful tools."

"As soon as we attempt to diagram software structure, we find it to constitute one, but several general directed graphs, superimposed on upon another. The several graphs may represent the flow of control, the flow of data, patterns of dependency, time sequence, name-space relationships. *These are usually not even planar, much less hierarchical. Indeed, one of the ways of establishing conceptual control over such structure is to enforce link cutting until one or more of the graphs becomes hierarchical.*"

Topic 15 5

---

---

---


---

---

---

---

---



## Unified Modeling Language

Let's break it down:

**Unified**

- In 1994,
  - Two important methodologists Rumbaugh and Booch decided to unify their approaches in 1994
- In 1995, another methodologist, Jacobson, joined the team
  - His work focused on use cases
- In 1997,
  - the Object Management Group (OMG) started to standardize UML

Topic 15 6

---

---

---

---

---

---

---

---

## Models

**Models** are abstract representations

- Contain essential characteristics and omit non-essential details
- “Essential” depends on the problem domain
  - There are no perfect representations
- Models can be representations of the world
  - Domain models
  - Requirements
- Models can be representations of software
  - Specifications
  - Design
  - Systems

Topic 15 7

---

---

---

---

---

---

---

---

---

---

## Why make models?

- Systems are complex and hard to understand
  - The world, organizations, relationships, software
- Models can make certain aspects more clearly visible than in the real system
- What can you do with models?
  - Express your ideas and *communicate* with other engineers
  - Reason about the system
    - ▣ detect errors
    - ▣ predict qualities
  - Generate parts of the real system
    - ▣ Code
    - ▣ Schemas

Can reverse engineer a system to make a model

Topic 15 8

---

---

---

---

---

---

---

---

---

---

## What constitutes a good model?

- A model should...
  - Provide abstraction
  - Render the problem in a format amenable to reasoning
  - use a standard notation
  - be understandable by clients and users
  - lead software engineers to have insights about the system
  - make the problem solvable computationally
    - ▣ Be good enough

Topic 15 9

---

---

---

---

---

---

---

---

---

---

## Remember: It's only a model

There will always be:

- Phenomena in the application domain that are not in the model (abstraction)
- Details in the application that are not in the model (abstraction)
  - **Just what you need**
- A model is never perfect
  - **"If the map and the terrain disagree, believe the terrain"**

Topic 15

10

---

---

---

---

---

---

---

---

## Modeling Languages

### Natural language

- Extremely expressive and flexible
- Very poor at capturing the semantics of the model
- Better used for elicitation, and to annotate models for communication

### Semi-formal notation

- Captures structure and some semantics
- Can perform (some) automated reasoning, consistency checking, animation, etc.
- Mostly visual - for rapid communication with a variety of stakeholders

**Examples:** diagrams, tables, structured English, etc.

### Formal notation

- very precise semantics, extensive reasoning possible
- Can automate reasoning, consistency checking, completeness checking, simulation, etc..
- Every detailed models )

Topic 15

11

---

---

---

---

---

---

---

---

## Unified Modeling Language (UML)

### **UML is a ...**

- *semi-formal graphical (visual) modeling language*
- *Object Modeling Language (OML)*
- *A way to communicate details...*
  - *Code*
  - *Architecture*

- **Uml is descriptive → tries not to be prescriptive**

Topic 15

12

---

---

---

---

---

---

---

---

### 3 Common Way to Use UML

- **Sketch** - *Quick Communication*
- **Blueprint** – *Complete Specification*
- **Programming Language**

Topic 15 13

---

---

---

---

---

---

---

---

### UML as a Sketch

- **Helps communicate** some aspect of the system
  - Forward & reverse engineering
- “Rough out” issues in the code
- **Not all of the code – just parts that you are working on immediately**
  - Selective communication → NOT complete specification
- **Short discussion with a team (10 min – 1 day)**
- **Quick and Collaborative**
- **Informal**

Topic 15 14

---

---

---

---

---

---

---

---

### UML as a Blueprint

- **Complete specification**
  - Forward & reverse engineering
- **Detailed design**
  - All design decisions laid out
  - Simplifies programming
- **Usually done by senior developer**
- **More formally documented**
- **CASE tools**
  - Forward Engineering
    - ▣ Support diagramming
    - ▣ Repository to store information
  - Reverse Engineering
    - ▣ Read source code → Generate Diagrams

Topic 15 15

---

---

---

---

---

---

---

---

## UML as a Programming Language

- UML Diagrams compiled into exe code
  - Automatic code generation
  - Sophisticated tool support

Topic 15 16

---

---

---

---

---

---

---

---

## Types of UML Diagrams

<h3>Structure</h3> <p>(6 types)</p> <ul style="list-style-type: none"> <li>Class diagrams</li> <li>Object diagram</li> <li>Package diagram</li> <li>Composite structure diagram</li> <li>Component diagram</li> <li>Deployment Diagram</li> </ul>	<h3>Behavior</h3> <p>(4 types)</p> <ul style="list-style-type: none"> <li>Activity diagram</li> <li>Use Case diagram</li> <li>State machine diagram</li> <li>Interaction diagrams           <ul style="list-style-type: none"> <li>Sequence diagram</li> <li>Communication diagram</li> <li>Interaction overview diagram</li> <li>Timing diagram</li> </ul> </li> </ul>
---	---

If the appropriate diagram is not part of UML  
*use it anyways*

Topic 15 17

---

---

---

---

---

---

---

---

## UML & the Software Process (Requirements)

- Use Cases
  - Describe how people interact with the system
- Class Diagram
  - Drawn from a conceptual perspective
  - Can build up a rigorous vocab of the domain
- Activity Diagram
  - Shows the workflow of the org.
    - Shows how s/w and human activities interact
  - Context for Use Cases
  - Details of complex Use Cases
- State Diagram
  - Shows states and events that change the state
    - Can be useful with interesting life cycles

Communication is key  
Customers may not be familiar with S/W techniques

Topic 18 18

---

---

---

---

---

---

---

---

## UML & the S/W Process (Design)

- **Class Diagrams**
  - From a software perspective
    - Show classes & how they interrelate
- **Sequence Diagrams**
  - For Common Scenarios
    - Pick most significant scenarios from Use Cases
    - Use CRC cards or sequence diagrams to determine how the software should behave
      - Class, Responsibilities, Collaborators (CRC) cards are index cards used to represent
        - » the responsibilities of classes
        - » interaction between the classes
- **Package Diagrams**
  - Show large-scale organization of the system
- **State Diagrams**
  - Used for classes with complex lifecycles
- **Deployment Diagrams**
  - Show the physical layout of the software

All of these can be used for design

Topic 15 19

---

---

---

---

---

---

---

---

---

---

## Class Diagrams

“A **Class Diagram** describes the types of objects in the system and the various kinds of static relationships that exist among them”

<b>Class Name</b>
Attributes (Name:type)
Operations / Methods (Name: Parameters)

Makes it easier to see the big picture

- Know what a class does at a glance

Topic 15 20

---

---

---

---

---

---

---

---

---

---

## Class Diagrams (terminology)

- **Properties →**
  - A structural feature of a class (fields in a class)
  - Can be represented 2 ways: Attributes and associations
- **Attribute →**
  - Describes a property as a line of text within the class box
  - Attribute name corresponds to the name of a field in a programming language
  - **Visibility Marker →**
    - Denotes whether an attribute is...
      - Public (+) or Private (-)
- **Associations →**
  - Describes a property as a solid line between 2 classes
  - Source to the target class

Attributes and Associations →  
Different notations for the same thing

Topic 15 21

---

---

---

---

---

---

---

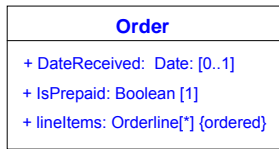
---

---

---

## Example: Properties as Attributes

### Placing an Order



Topic 15

22

---

---

---

---

---

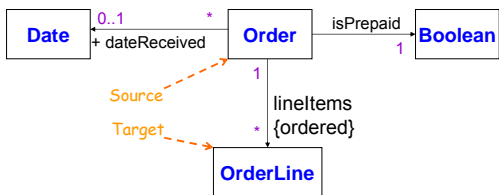
---

---

---

## Properties as Associations

### Placing an order:



Topic 15

23

---

---

---

---

---

---

---

---

## Attributes & Associations

- Same properties → different notations
- When do you use which?
  - Attributes for more simple properties (such as Booleans or Dates)
  - Associations for more significant properties (such as Orders or Customers)
- Associations show more – such as multiplicities (covered in discussion)

Topic 15

24

---

---

---

---

---

---

---

---