CSE 416

UML Overview
Use Case Requirements

2

Reference

- Class diagrams

  en.wikipedia.org/wiki/Use_case

3

Session Objectives

- Understand the purpose of UML in the design and development of a system
- Understand the use of use case descriptions to identify the detailed functionality of a system
- Begin to transform top-level requirements into use cases

4

What is Modeling?

- Modeling consists of building an abstraction of reality
- Abstractions are simplifications because:
  - They ignore irrelevant details and
  - They only represent the relevant details
- What is relevant or irrelevant depends on the purpose of the model, the audience, and other factors

This is a very difficult decision

5

Why Model Software?

- Software is getting increasingly more complex
  - Some versions of Windows > 40M lines of code

Could you comprehend 40M LOC?

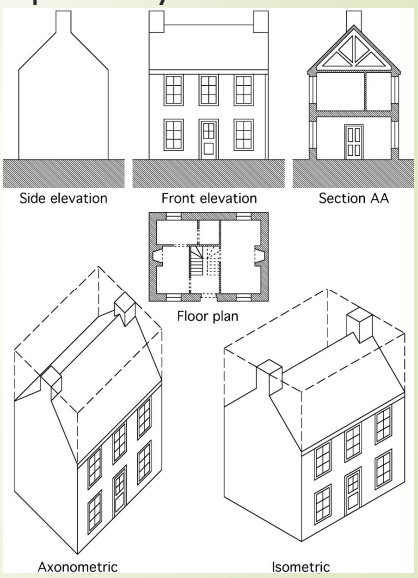
- Modifying a model of a system is much, much easier than modifying software
- We need simpler representations for complex systems
  - Modeling is a way for dealing with complexity

Remember, one course goal is to think first, code second

6

How Do We Deal With Complexity?

- Break it down into simpler parts
- Example- design specifications for a building
- Helps in
  - getting user/peer feedback
  - Getting approval
  - Avoiding construction problems



7

Systems, Models and Views

- A ***model*** is an abstraction describing a system or a subset of a system
- A ***view*** depicts selected aspects of a model
- A ***notation*** is a set of graphical or textual rules for depicting views
- Views and models of a single system may overlap each other

Unlike DB design, we often just generate different views, which together constitute a model

8

What is UML?

- UML (Unified Modeling Language)
  - A standard for modeling object-oriented software.
  - Derived from the convergence of notations from three leading OO approaches:
    - OMT (James Rumbaugh)
    - OOSE (Ivar Jacobson)
    - Booch (Grady Booch)
- Supported by several CASE tools
  - Visio
  - Workbench
  - Visual Paradigm
  - Lucidchart

You can model 80% of most problems by using about 20 of % UML (maybe 90/10)

9

UML Approach for CSE416

- Use cases
  - Describe the functional behavior of the system as seen by the user
  - Great for decomposing a system into buildable units
- Sequence diagrams
  - Describe the dynamic behavior between actors and the system and between objects of the system
  - Helps to define the objects that are needed to implement a use-case
- Class diagrams
  - Describe the static structure of the system: Objects, Attributes, Associations
  - Can be revised based on discoveries made from sequence diagrams

You can model 80% of most problems by using about 20 of % UML (maybe 90/10)

10

Other UML Notations

- UML provides other notations used less often
- Implementation diagrams
  - Component diagrams
  - Deployment diagrams
  - State-chart diagrams (essentially a finite state automaton)
  - Activity diagrams (essentially a flow chart)

We will use activity diagrams to model the behavior of the Python parts of the system

11

UML Core Conventions

- Rectangles are classes or instances
- Ovals are functions or use cases
- Instances are denoted with colon notation

```
myWatch:SimpleWatch
:SimpleWatch
joe:Firefighter
```

A consistent code and design style is essential for group communication

- Diagrams are graphs
  - Nodes are entities
  - Arcs are relationships between entities

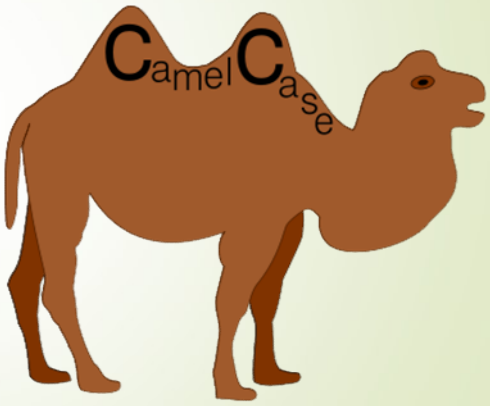
Note the camel case notation for the OO/Java components

12

CamelCase

Consistent with the Java components of your system

- A compound word begins each element with a capital letter
  - Upper camel case (UCC)
  - Lower camel case (LCC) – first letter not capitalized
- Examples
  - UCC – "CamelCase"
  - LCC – "camelCase"



13

Naming Conventions for OO/Java

- Camel case for classes (upper cc) and attributes (lower cc)
- Classes– singular
- Attributes– singular (plural for collections)
- Avoid acronyms and abbreviations except where well known
  (e.g., PI for Principal Investigator)

Conventions apply very early in the process

Names should describe the application domain, not the implementation approach

Naming conventions are part of the "teamwork" approach to CSE416

14

Use Case

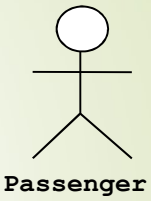
- Used during requirements elicitation to represent external behavior
- Represents an interaction sequence for a type of functionality
- A use case consists of:
  - Unique name
  - Participating actors
  - Entry conditions
  - Trigger
  - Flow of events (scenario)

A use case represents a class of functionality provided by the system as an event flow

15

Actors

- An actor models an external entity which communicates with the system
- It can be a:
  - User,
  - External system, or
  - Physical environment
- An actor has a unique name
- Example:
  - User of your system



Similar to a role

16

Example

- Example of a textual use case
- Design issues:
  - No overlap in use cases (instead think of preconditions)
  - Look for use cases that cover multiple roles (with exceptions that differentiate the roles)
  - Proper size (not too many steps or too few steps)

Use Case Sample – CSE308

Use-case:	Making super districts
Primary actor:	General User
Goal in context: *one sentence description*	User combines individual congressional districts into one super district.
Preconditions:	Currently on the main page.
Trigger:	Click the option for Super Districts in the "tests" dropdown.
Scenario: *step by step actions taken by the system to achieve goal in context*	1. To indicate the change in mode, all the districts change from being different colors, to the same dull color, but the boundaries are clear. Exact colors are contained in the colors.pdf document. 2. Toolbar shows a "Super District Mode" banner and 3 Buttons: "Undo", "Reset", and "Make Super District" 3. Display guidelines to user about redistricting based on parts of the House Bill. The text is updated dynamically, and is placed in some blank area of the screen. (text provided with a link to an external document that summarizes the House Bill points we want to use). 4. The user clicks any of the districts currently on the map, and that district changes to a different color from the Super district color table that does not conflict in color with any of the districts adjacent to it. 5. From the last clicked district, adjacent districts that can be clicked will be highlighted. The user can click on one of them. 6. Either use an Undo button or a Reset, to undo the last click that was done, or restart making a super district. 7. Once a user is done clicking districts, the user clicks the "Make Super District" button and the super district is run through verification tests (Use Case ID something). 8. If verified to be valid, then the super district is officially created. The fill color for the super-district is the next available color in the fill color table. 9. The "Create Super-District button is enabled again (the user can click to make a new super district again.)
Exceptions: *Special cases that involve deviations from scenario above*	1. When users enter a state with 5 or fewer districts, then the banner shows a message informing the user that he/she cannot make super districts in this state, instead of the default banner. 2. After clicking "Make Super District", if not valid, then the all the clicked districts are still there and user can undo.
Priority:	Build 3

17

Use Case: Summary

- Use case documentation
  - represents external behavior
  - are useful as an index into the use cases
  - Includes text and diagrams
  - Should be complete (all use cases need to be described)

We use text use-case (not diagrams)

18

UML Summary

- UML provides a wide variety of notations for representing many aspects of software development
  - Powerful, but complex language
  - Can be misused to generate unreadable models
  - Can be misunderstood when using too many exotic features

UML should be used to the extent that it improves communications concerning the system to be built

19

Have You Satisfied the Objectives?

- Understand the purpose of UML in the design and development of a system
- Understand the use of use case descriptions to identify the detailed functionality of a system
- Begin to transform top-level requirements into use cases