Waivers

- No 340?
  - no waiver unless you can show me real UML models
- No 378?
  - I'll accept it as a co-req
- I need your name and sid to fill in the form
  - I haven't had many emails
  - John H
  - Ann D
- I expected an email message from the "three amigos" by now.

Sept 17 Anecdotes/Housekeeping

- How many programmed during summer (again)
  - I cannot believe the results of my little poll!
- Customer hands over design. What do you do first?
  - Should you assume design is perfect?
- Anecdote:
  - regexp test scaffold. Scaffold does only what testing requires.
  - two team metaphor. GUI team and model team.
- Code generation?
- A1 issues:
  - try and follow the design. Nothing is perfect.
  - my goof: remove references to "check out" use case.
  - iconix goof: back to home page not login page after add account
  - iconix goof: really need "add item to shopping cart".
  - Otherwise you miss an important method on ShoppingCart

UML

- Unified Modeling Language
  - In early 90s, there were many competing graphical notations all used for OOA.
  - Three of the major players got together in Booch's company
    - Rational Software Corporation
      - Booch, Rumbaugh, Jacobson
    - Merged their ideas to produce
      - UML (public domain)
      - Associated tools (mainly Rational Rose)
      - Rational Software Process (public domain)

Linkage between Domain, Design

- The transition between Analysis (Domain) and OO Design has been the stumbling block.
- Many authors have contributed and many agree to some level.
- Use cases are used to capture the sequence of operations the system must support. Related to requirements.
  - Identify many domain classes
- I like Doug Rosenberg's ICONIX approach.
  - Robustness diagrams identify classes that support the user interface (boundary) and control logic of a system
    - Often find missed domain classes. Hence name.
    - In many cases crude notion of the UI is required to proceed.
    - http://www.iconixsw.com/uml_for_e-commerce.ppt
    - say around slide 10-17

Last Lecture

Books on UML

- You must acquire reference materials on UML
  - Some of these lecture materials prepared from
    - UML In A Nutshell (O'Reilly) by Sinan Si Alhir
  - Also
    - The Unified Modeling Language User Guide
      - Booch et. al
    - UML Distilled (second edition) ← Cheaper, THIN!
      - Martin Fowler.
  - Also
    - Reference materials off the Web
- Object Modeling books:
  - Object Oriented Analysis and Design
    - Booch et.al.
  - Designing Object-Oriented Software
    - Wirfs-Brock et. al.
  - Object-Oriented Modeling and Design
    - Rumbaugh et. al.
  - Object-Oriented Analysis
    - Coad and Yourdon
-

Uses for UML

- OOA
  - A visual language for, in the problem domain,
    - capturing knowledge about a subject
    - expressing knowledge for the purposes of communication
- ..Difficult transition.. Domain Space
- OOD
  - A visual language for, in the solution space,
    - capturing design ideas
    - communicating design ideas Design Space
  -
- Related, but distinct usages
- Must supplement both with written explanations and transient diagrams such as robustness, sequence.

UML Definition

- OMG-endorsed standard (Object Management Group)
  - UML Semantics Document
    - "inside-view"
    - specifies semantics of constructs
  - UML Notation Guide
    - "outside-view"
    - specifies notation for expressing constructs
  - Object Constraint Language specification document
    - definition of a (textual) language for expressing logical constraints
  - Zillions of people working on formalizing UML
    - Meanwhile it's just bubbles!

This Course and UML

- You will use UML for assignments
  - Unfortunately, many of following slides are in OMT, as is the Design Patterns book.
- UML
  - Has its warts
  - Good enough when augmented by written explanation
  - NOT FORMAL. Just bubbles, really.
- Cover only the most useful subset of UML
  - Mainly class/object/use case/sequence charts.

The World Out There

- The real world is impenetrably complex
  - e.g., a complete model of you would include DNA, behaviour specifications, total history, parents' history, influences, …
  - for a particular problem, abstracting you as
    - last name
    - first name
    - student number
    - course
    - final grade
  - may be enough.
- The Object-Oriented paradigm is one method for simplifying the world.

UML is For

- For Problems
  - Specifying
  - Visualizing
  - Promoting Understanding
  - Documenting
  -
- For Problem Solving
  - Capturing Attempts
  - Communicating Attempts
  - Leveraging Knowledge

For Solutions
Specifying
Visualizing
Evaluating
Constructing
Documenting

Primarily communication aid between people!

Objects [Rumbaugh]

- An object is
  - A concept, abstraction, or thing
  - with crisp boundaries and
  - meaning for the problem at hand
- Objects
  - promote understanding of the real world
  - provide a practical basis for computer implementation
- Decomposition of a problem into objects depends on
  - Judgment
  - The nature of the problem being solved
    - Not only the domain: two analyses of the same domain will turn out differently depending upon the kind of programs we wish to produce

Guidance of domain experts essential

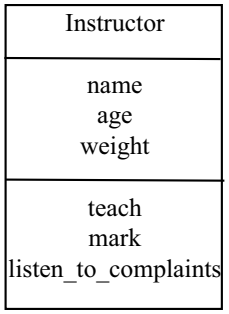
Parts of UML

- Class Diagrams
  - models
- Object Diagrams
  - example models
- Use Case Diagrams
  - document who can do what in a system
- Sequence Diagrams
  - shows interactions between objects used to implement a use case
- Collaboration Diagrams
  - same as above, different style
- Statechart Diagrams
  - possible states and responses of a class and what transitions them
- Activity Diagrams
  - describe the behaviour of a class in response to internal processing
- Component Diagrams
  - Organization of and dependencies amongst software implementation components
- Deployment Diagrams
  - Describe the mapping of software implementation components onto processing nodes

UML is GIANT
but no robustness..

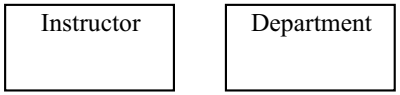
Operations

- A function or a transformation that may be applied to or by objects in a class.
  - Not often used (not often terribly useful) in an OOA



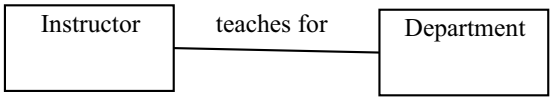
Classes

- A class describes a group of objects with similar properties.
  - ***Class***: Instructor
    - ***Object***: David Penny
    - ***Object***: Matthew Zaleski
  - ***Class***: Department
    - ***Object***: Department of Computer Science
    - ***Object***: Department of Electrical Engineering
  -



Links and Associations

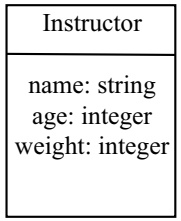
- The means for establishing relationships among objects and classes.
  - **link**: a connection between two object instances
  - **association**: a collection of links with common structure and semantics.



By default, read association names left to right and top to bottom (override with or)

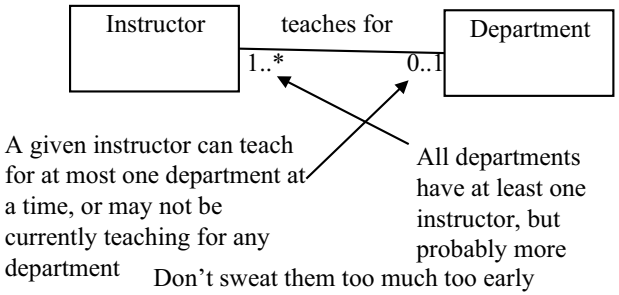
Attributes

- Data values held by the objects of a class



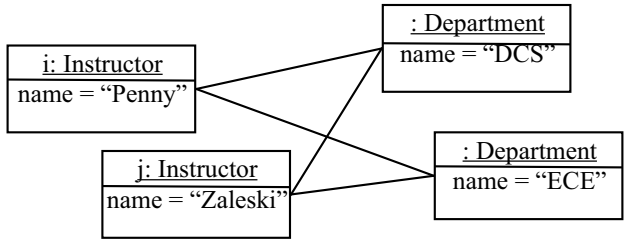
Multiplicities Carry Important Messages

- Used to indicate the number of potential instances involved in the association when the other associated class is fixed.

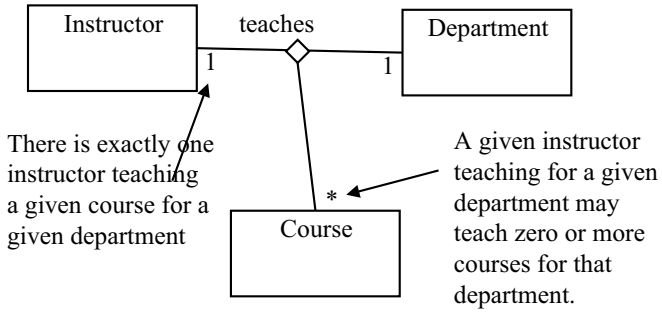


Object Diagrams

- Models instances of things contained in class diagrams.
- Shows a set of objects and their links at a point in time
- Useful preparatory to deciding on class structures.
- Useful in order to better explain more complex class diagrams by giving instance examples.



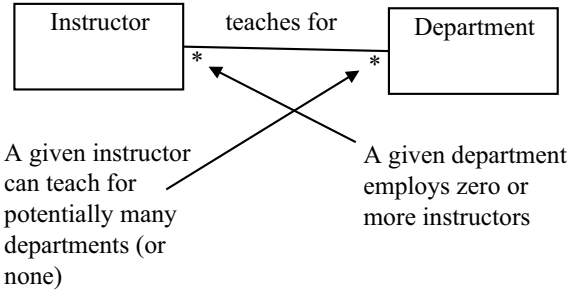
N-Ary Associations



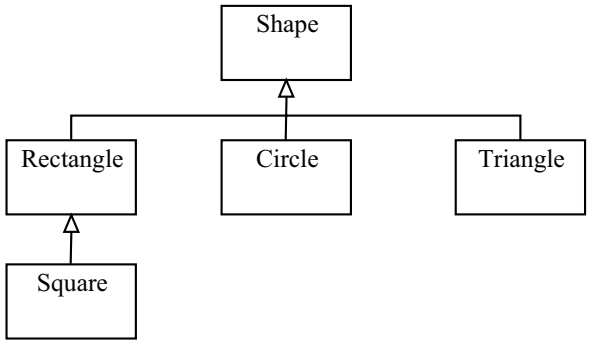
Try to avoid them!

Multiplicity

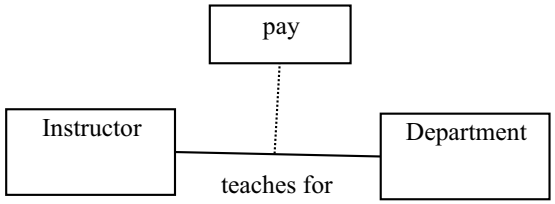
- Used to indicate the number of potential instances involved in the association when the other associated classes are fixed.



Generalization (a.k.a. Inheritance, is-a)

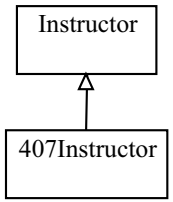


Attributes on Associations

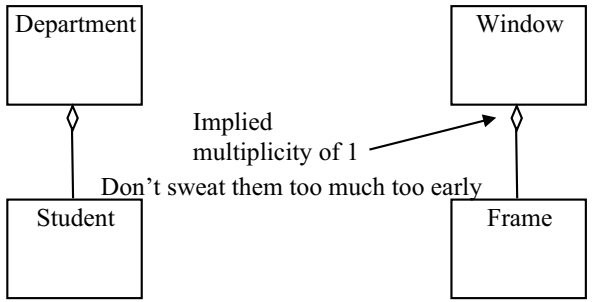


Inheritance can be a trap..

- Analysis shown below may not be a good choice,
  - instances of class 407Instructor (me) may teach different things next term.
- Point is meant to be that just because a concept *can* be viewed as a generalization of another doesn't mean you should use inheritance. It has to makes sense as objects change.



Aggregation Indicators (Part-Of)



Aggregation
(no associated semantics)

Composition
(strong ownership,
coincident lifetime)

Example

- We are asked to build a system for keeping track of the time our workers spend working on customer projects.
- We divide projects into activities, and the activities into tasks. A task is assigned to a worker, who could be a salaried worker or an hourly worker.
- Each task requires a certain skill, and resources have various skills at various level of expertise.

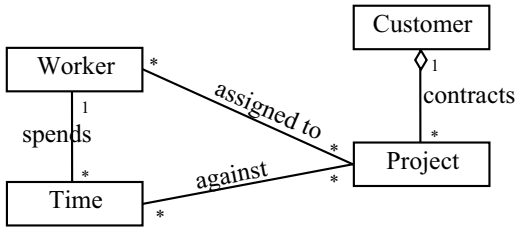
Steps

- Analyze the written requirements
  - Extract nouns: make them classes
  - Extract verbs: make them associations
  - Draw the OOA UML class diagrams
  - Determine attributes
  - Draw object diagrams to clarify class diagrams
- Determine the system's use cases
  - Identify Actors
  - Identify use case
  - Relate use cases
- Draw sequence diagrams
  - One per use case
  - Use to assign responsibilities to classes
- Add methods to OOA classes

Preliminary Design: Robustness Diagrams

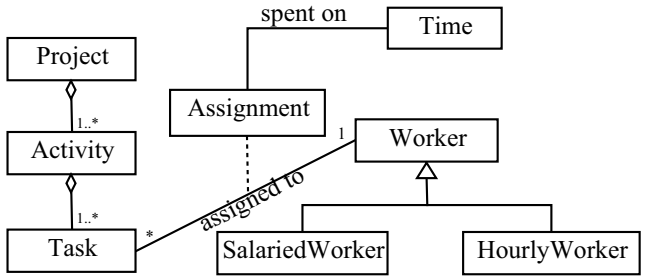
Example

- We are asked to build a system for keeping track of the time our workers spend working on customer projects.

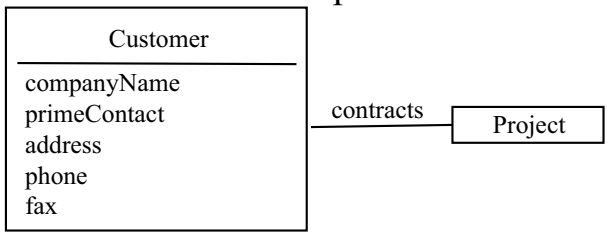


Example

- We divide projects into activities, and the activities into tasks. A task is assigned to a worker, who could be a salaried worker or an hourly worker.



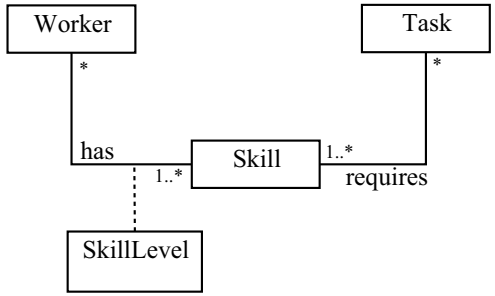
Example



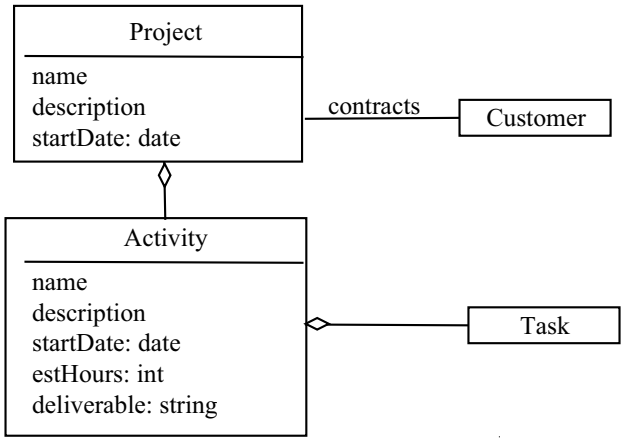
N.B.
- Project has no attribute in Customer
- association is enough
- no database id for Customer shown
- in an OOA, only include an id if visible to users
- may include such things during database design or OOD

Example

- Each task requires a certain skill, and workers have various skills at various level of expertise.



Example

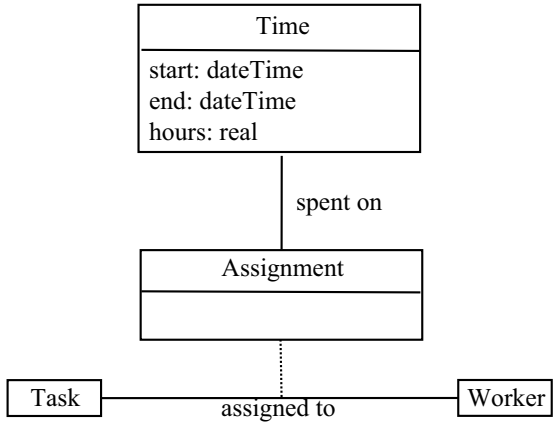


Steps

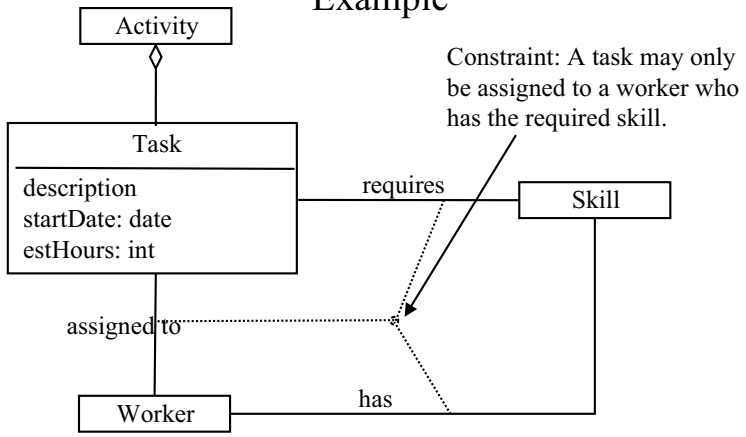
- Analyze the written requirements
  - Extract nouns: make them classes
  - Extract verbs: make them associations
  - Draw the OOA UML class diagrams
  - **Determine attributes**
  - Draw object diagrams to clarify class diagrams
- Determine the system's use cases
  - Identify Actors
  - Identify use case
  - Relate use cases
- Draw sequence diagrams
  - One per use case
  - Use to assign responsibilities to classes
- Add methods to OOA classes

Preliminary Design: Robustness Diagrams

Example



Example

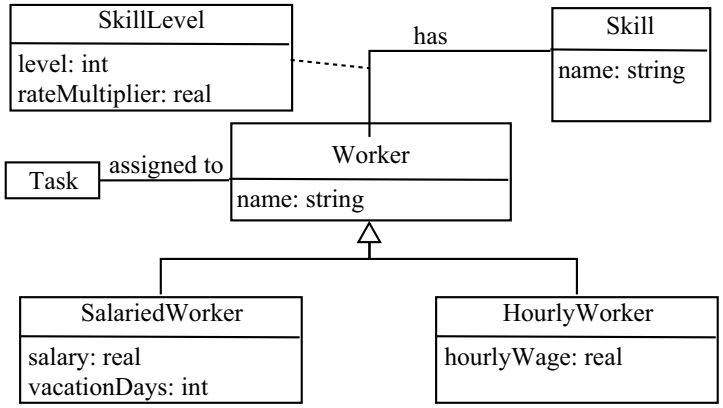


Steps

- Analyze the written requirements
  - Extract nouns: make them classes
  - Extract verbs: make them associations
  - Draw the OOA UML class diagrams
  - Determine attributes
  - Draw object diagrams to clarify class diagrams
- Determine the system's use cases
  - Identify Actors
  - Identify use case
  - Relate use cases
- Draw sequence diagrams
  - One per use case
  - Use to assign responsibilities to classes
- Add methods to OOA classes

Preliminary Design: Robustness Diagrams

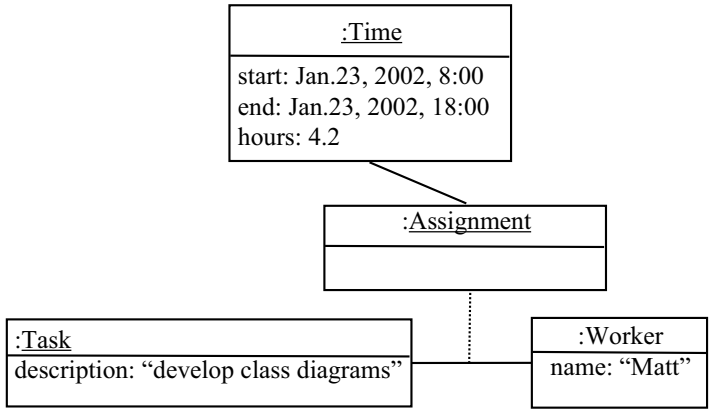
Example



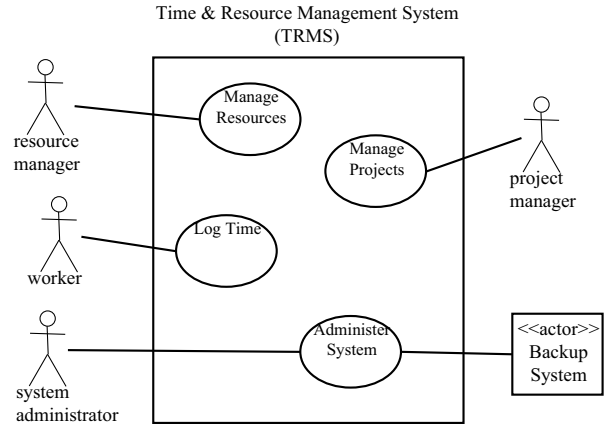
Use Cases

- Actors:
  - Represent users of a system
    - human users
    - other systems
- Use cases
  - Represent functionality or services provided by a system to its users

Object Diagrams



Use Case Diagrams



Steps

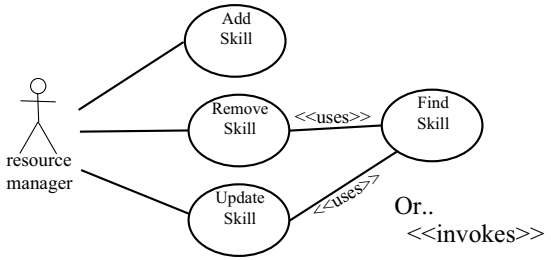
- Analyze the written requirements
  - Extract nouns: make them classes
  - Extract verbs: make them associations
  - Draw the OOA UML class diagrams
  - Draw object diagrams to clarify class diagrams
  - Determine attributes
- Determine the system's use cases
  - Identify Actors
  - Identify use case
  - Relate use cases
- Draw sequence diagrams
  - One per use case
  - Use to assign responsibilities to classes
- Add methods to OOA classes

Preliminary Design: Robustness Diagrams

Add Skill Use case

- The system displays the “New Skill Editor” page, which the clerk fills in. When satisfied the clerk presses the “Submit” button and the system processes the new Skill and returns to the “Skill Manager” page.
- Alternative courses
  - clerk changes mind and cancels edit
  - new skill fails edit checks

Resource Manager Use Cases

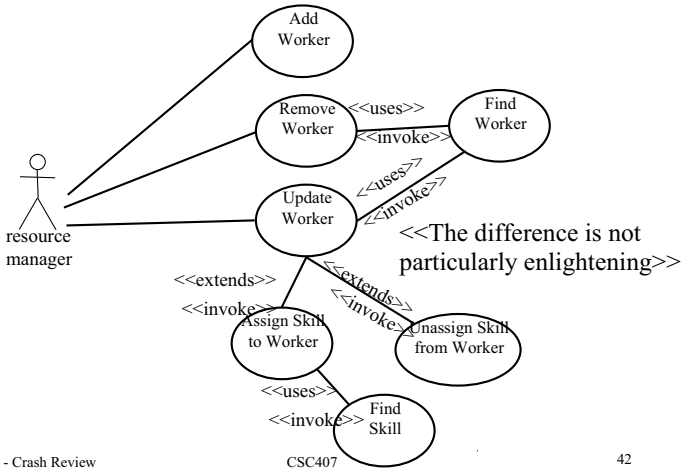


Steps

- Analyze the written requirements
  - Extract nouns: make them classes
  - Extract verbs: make them associations
  - Draw the OOA UML class diagrams
  - Draw object diagrams to clarify class diagrams
  - Determine attributes
- Determine the system’s use cases
  - Identify Actors
  - Identify use case
  - Relate use cases
- Draw sequence diagrams
  - One per use case
  - Use to assign responsibilities to classes
- Add methods to OOA classes

Preliminary Design: Robustness Diagrams

More Resource Manager Use Cases



Robustness Diagrams

- Not part of UML, really.
- Jacobson saw the need for a preliminary design stage in which use cases “drive” or “simulate” the system interacting with the just completed domain model.
  - Flushes out omissions (hence robustness?)
  - Finds new objects and classes that are not in the domain but will obviously be required in design
    - Boundary classes
    - Controller classes
- General idea is that robustness diagrams are a transitional diagram between the first cut at a domain model and the real software design that occurs during the construction of a sequence diagram.

Preliminary Design

- Sanity check model, use case text
- Completeness Check on use cases, especially alternate courses
- Discover more domain classes
- Start uncovering Design classes
  - Hard to progress without some notion of user interface.
  - Perhaps this is where lessons learned building a GUI prototype feed back in to design process?
  - I try to have screen shots available at this point.
- Crossover point.
  - Still a lot of information in use cases that needs to be considered.

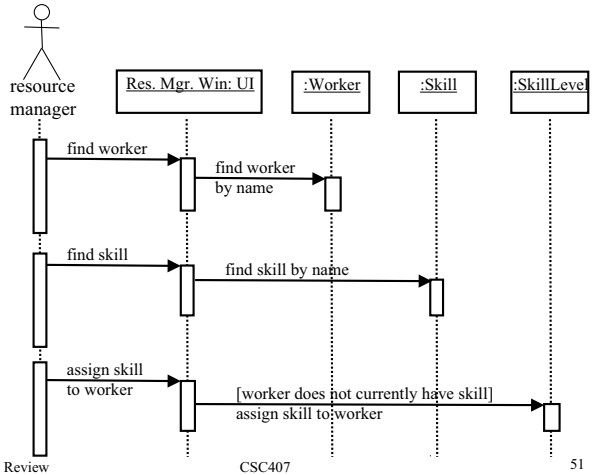
Robustness Diagrams

- Definition, such as it is:
  - Entity object are domain.
  - Boundary Objects are used by Actors.
  - Controllers are glue. Largely placeholders for methods.
- Classification implies some limitations:
  - Actors talk only to Boundary objects
  - Boundary objects talk to Controllers and Actors
  - Entity objects talk only to Controllers
  - Controllers send messages to Boundary objects and Entity objects but not Actors.

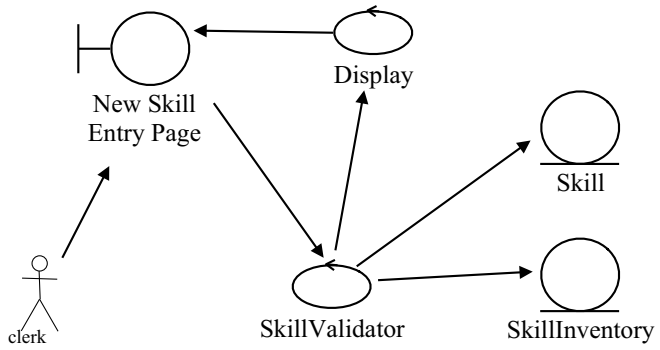
Add Skill Use case

- The system displays the “New Skill Editor” page, which the clerk fills in. When satisfied the clerk presses the “Submit” button and the system processes the new Skill and returns.
  - Alternate Course: If the data entered into the new skill page fails to edit check the system returns the user to the new skill editor but with a message inserted into the editor’s message area.
- Whoa! “processes the new skill”? Missing stuff!
- The system displays the “New Skill Editor” page, which the clerk fills in. When satisfied the clerk presses the “Submit” button and the system adds the new Skill to the Skills Inventory and returns.

Sequence Diagram – Assign Skill to Worker Use Case



Robustness Diagram



Steps

- Analyze the written requirements
  - Extract nouns: make them classes
  - Extract verbs: make them associations
  - Draw the OOA UML class diagrams
  - Draw object diagrams to clarify class diagrams
  - Determine attributes
- Determine the system's use cases
  - Identify Actors
  - Identify use case
  - Relate use cases
- Draw sequence diagrams
  - One per use case
  - Use to assign responsibilities to classes
- Add methods to OOA classes

Preliminary Design: Robustness Diagrams

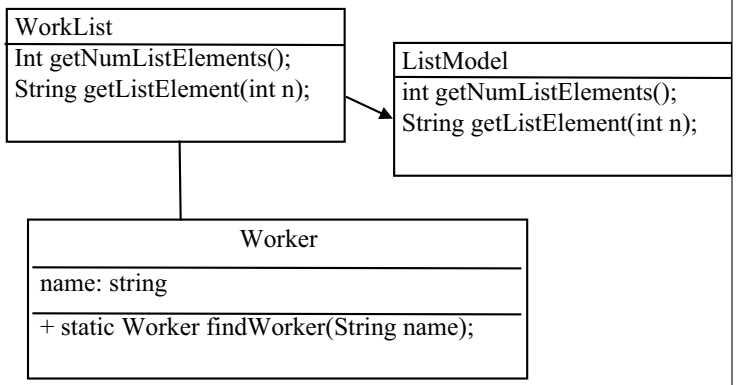
Steps

- Analyze the written requirements
  - Extract nouns: make them classes
  - Extract verbs: make them associations
  - Draw the OOA UML class diagrams
  - Draw object diagrams to clarify class diagrams
  - Determine attributes
- Determine the system's use cases
  - Identify Actors
  - Identify use case
  - Relate use cases
- Draw sequence diagrams
  - One per use case
  - Use to assign responsibilities to classes
- Add methods to OOA classes

Preliminary Design: Robustness Diagrams

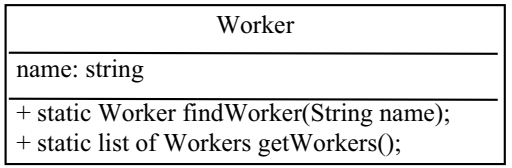
In Design

- Bring methods closer to implementation



Add Methods

- Read sequence diagrams to identify necessary methods



In Design

- Bring methods closer to implementation