A review of

OO

analysis and design

HW 3 due next time

Update to HW3:

   Please read chapter 2 of Arlow and Neustadt at home
IFSC 7310: Part II
Here is what this Course is not…

• It’s not about programming in Java
  – …or about Object Oriented Programming
  – It is about Object Oriented Analysis & Design

• Information system development:
  – It’s more than programming (or buying stuff)

• It’s based on analysis and design, & teams
  – For example, see figures…
Relative Costs of Development Process Phases
(from Boehm via Schach via Turner, slightly modified)
Relative cost to fix a fault that could have been fixed in requirements
Figure 26.1: The Cost of Change

Source: http://www.osl.iu.edu/~lums/swc/www/xp.html
Object-Oriented Analysis and Design: in Context

• See life cycle figures…
Source:
http://buyrpics.com/shop/images/0407070061.JPG
Source:
http://pumahosa.home.att.net/home.html
A “pure” waterfall model (source: http://cnx.org/content/m14619/latest/)
Agile development interpreted in the waterfall model

Source: http://duncanpierce.org/node/180
How would you improve some arrows?

Source:
http://www.jodypaul.com/SWE/LCM/waterfall.html
Source:
Source: http://www.acidaes.com/SWM.htm. “Unlike the Pure Waterfall Model where complete documentation is provided to the team handling the next phase, Sashimi Waterfall Model suggests personnel continuity between phases.”
The classic waterfall development model

1. Requirements/analysis
2. Design
3. Coding
4. Testing
5. Maintenance
Source:
http://ozark.hendrix.edu/~burch/csbsju/cs/160/notes/36/1.html
Source: http://www.outside-in-development.com/outside_in_process/agile_outside_in_process.html
Source: http://www.albany.edu/acc/courses/fall97/acc681/ch7.html

The waterfall model (Systems Development Life Cycle)
Figure 26.2: The Waterfall Model
(Source: http://www.osl.iu.edu/~lums/swc/www/xp.html)
Somerville’s waterfall model

1. Requirements analysis and definition
2. System and software design
3. Implementing and unit testing
4. System testing
5. Operation and maintenance
Ian Sommerville’s (2000) Waterfall model

www.cise.ufl.edu/~pjd/courses/3031/notes/SDPModels.ppt
Source:
Fountain Model
Object-Oriented Analysis and Design: in Context

• Recall life cycle figures…

• Unified Process:
  – Applies a waterfall or fountain-like approach to each new functionality
  – Why do you imagine clients would like it?
An alternative........
Don’t bother taking this course!
Just use...the **Build-and-Fix Model!!** (after Schach)

(repeat many slow, expensive times!!)

- Build something that runs
- Fix it up until client is happy
- Operations mode
What is **Analysis**? Design? Object-Oriented Analysis and Design?

- **Analysis**: “any detailed examination”
  
  — Webster’s New World College Dictionary (4th ed.)

- In IS development, you examine **problems**, to understand them
  
  — (You don’t **examine** solutions, you **build** them)

- Therefore **analysis** refers to
  
  — Examining the **problem** to be **solved**
What is Analysis? **Design**? Object-Oriented Analysis and Design?

- **Design:**
  - description of the structure of a solution

- In creating IS systems, you design IT solutions
  - (analyze the problem, design the solution)

- Therefore design refers to
  - determining the structure of an IT system
What is **Analysis? Design?**

Object-Oriented Analysis and Design?

- **Analysis**: “Do the right thing”
- **Design**: “Do the thing right”
  - (Larman 2\textsuperscript{nd} & 3\textsuperscript{rd} p. 7)
- **Analysis**: What the system will do
- **Design**: How the system will do it

- Which is more related to the concept of “specs”?
What is Analysis? Design?

Object-Oriented Analysis and Design?

• OOA&D:
  – Object-Oriented Analysis and Design

• Object-oriented analysis:
  – Specs which include the problem’s object-like parts
  – Can be done regardless of implementation language

• Object-oriented design:
  – Determine actual objects in the solution
    • I.e., their interactions and contents

• The analysis/design boundary is far from razor sharp

• “analysis” & “design”
  – They help understand things (thus avoiding the “build-and-fix” model)
  – Which goes with understanding the problem?
  – Which goes with understanding the solution?
A Definition of “Object”

• An object is
  – A “data abstraction”:
    • A bundle of
      – Data, and
      – Methods (modules containing executable statements)
    • The bundle is conceptually coherent
    • It presents an interface to the outside
      – Outside, no need to understand the rest
      – Why is that good?
  – Inheritance is possible
  – Thus, object=data abstraction + inheritance
A Definition of “Object”

• An object is a bundle of
  • Data, and methods
  • The bundle is conceptually coherent
    – It presents an interface to the outside

• Inheritance is possible

• Consider objects of class TrafficLight

• How does this relate to TrafficLights?
Example of OOA&D

• Player rolls 2 dice: a 7 wins, other #s lose
  – (Legal notice: IFSC 7310 takes no position for or against gambling! It’s just from a book! Larman 2nd/3rd sect. 1.5)

• First, define “use cases”
• Next, define a “domain model”
• Third, define “interaction diagrams”
• Fourth, define “class diagrams”
Defining Use Case

- A use case is a **system operation scenario**
  - Use cases are “written stories” (Larman)
  - They are “text documents” (Larman p. 71 (2nd), 89 (3rd))
- **Use** – utilization, purpose, functionality…
- **Case** – example, instance, situation…
  - (suggesting 9 phrases whose meaning = “use case”)
- Systems that do lots of things
  - will have lots of use cases!
- Is the following a use case?
3.53.3 Example

Figure 3-44  Use Case Diagram
3.53.3 Example

These are not exactly use cases (which are documents)

Rather this is a use case diagram.

(Figure source: OMG Unified Modeling Language Specification version 1.3, www.omg.org)

Figure 3-44 Use Case Diagram
Not only that, this use case diagram actually shows not a use case, but four separate use cases! (what are they?) (Figure source: OMG Unified Modeling Language Specification version 1.3, www.omg.org)
First: Define Use Cases

• Recall the example:
Player rolls 2 dice: 7 wins, other #s lose

• Create a use case now on paper…
  – Then we’ll discuss a bit
Back to Defining the Use Cases

• You can describe the dice game with…
  – …just 1 use case

• Player rolls dice, winning if they add to 7, else losing
  – Two use cases also work well
  – Thirty-six use cases is a bit extreme
Second: From Use Cases to a Domain Model

Strategy:

→ Define use cases

→ Define domain model

→ Define interaction diagrams

→ Define design class diagrams

Objective: decompose domain into object-like pieces

Try it now…
Domain Model Example

• This is an “entity-relationship” diagram
• See Figure next slide
• Note
  – name of entity,
  – relationship description (1 word),
  – relationship numerical property,
  – attribute names
Entity-relationship diagram

(after Larman Fig. 1.3, p. 8 (2nd ed.), 9 (3rd))
Third: Define Interaction Diagrams

- Given the entities...
  - Show their possible interactions
  - Interactions among objects use
    - Messages
  - A message is just the name for a method call
- I’ll draw the entities
  - (computer, die 1, & die2)
- You draw the interactions
- See solution next slide
- Note: assume a computerized dice game

Strategy:

→ Define use cases
→ Define domain model
→ Define interaction diagrams
→ Define design class diagrams
Fourth: Define Design
Class Diagrams

- This is a basic **design** for a system…
  - Show classes in boxes
  - List the data members and methods of each class
  - Show how many objects of each class interact with how many of each other class

- Try it now…
- Then we’ll see what text has (Figure 1.5)

**Strategy:**

- Define use cases
- Define domain model
- Define interaction diagrams
- Define design class diagrams