

## Announcements

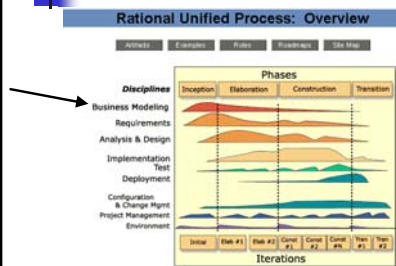
- The next HW has been updated
- It now states:
- **HOMEWORK #9:** using what you know from other classes about planning, make a plan for your own use (not to hand in) and then implement a user interface for the Use Case Diagram Assistant System. Screen shot(s) *and printed source code* due next Tuesday 10/19/04. Your interface *should contain event handler stubs*. These might contain, for example, `System.out.println()` calls.
- Questions?

## Domain Modeling Introduction

- Reference: Chapter 10
- As context:
  - Implementation is discussed in chap. 20 (p. 301)
  - Elaboration iteration 2 starts in chap. 21 (p. 319)
  - Textbook emphasizes OOA/D
    - (not so much implementation)

## Domain Modeling in the RUP

(Fig. 2.4 is similar)



Requirements is a subset of analysis (p.6)

Business modeling is a prerequisite for design

## Domain Vs. Software Modeling

- The domain is the real world environment
- A domain model is a description of that environment
- A software system is something different!
- Classes, objects, etc. are *not* in the real world
  - But, the real world *motivates* classes, objects, etc.
  - *Analyzing* the world helps *design* the software later
- Thus, we spend 4 chapters discussing the world

## Domain Modeling Introduction

- Let's model (i.e. abstractly describe) the domain
- A Domain Model (caps) is a UP artifact
- A domain model (small) may or may not be
- We will use UML (Universal Modeling Language)
- Domain models show:
  - Domain entities ("conceptual classes") – chap. 10
  - Relationships among conceptual classes – chap. 11
  - "Attributes of conceptual classes" (p. 128) – chap. 12
  - Example: this course

## Example Domain Model

•See Fig. 10.1

•Let's apply to the 486 example...

## POS Domain Model Example

- The notation is UML
- The same idea could be said using text, glossary, whatever
- Note the *abstractness*
  - Many details are missing
- Note that software components can include other things (databases, windows, methods)
- See Figs. 10.2 and 10.3

## Which of these could be in a registration domain model?

- Student class schedule
- Student
- Course (e.g. CPRE 486)
- Registration start screen
- Time overlap checker
- Server

## Domain Model as Problem Decomposition

- Complex problems (like software system development projects)
  - ...require decomposing the problem into pieces
- Problem pieces help design solution pieces
- Object-oriented problem decomposition
  - ...uses entities (to become classes and objects)
- Structured (pre-object-oriented) decomposition
  - ...uses processes (to become functions)

## Finding Object-Oriented Domain Entities

- Method 1:
  - "Conceptual Class Category List"
    - Long phrase – what are the adjectives? Nouns?
      - (I.e. how does it break down into pieces?)
- Method 2:
  - Look for nouns
    - Nouns may be found in use cases
    - Nouns are candidate domain entities
- Let's try these for the **POS system**

## Example: POS Main Success Scenario

- See Category list (Table 10.1)
- See Use Case scenario (p. 50)
- Let's think of some possible conceptual classes (domain entities)...

## POS Example: the Receipt as a Conceptual Class

- Should it be one?

("answer" next slide)

## POS Example: the Receipt as a Conceptual Class

- Should it be one?
- It *is* a part of the domain
- It is *not* used in the main success scenario
- So don't use it in iteration 1
- *Do* use it in a later iteration

## Conceptual Classes – Entities Vs. Attributes

- Domain models (e.g. using UML Class Diagrams)
  - ...contain both (recall Figure 10.1)
  - ...so what should be a *entity* and what should be an *attribute*?
- Larman: "If we do not think of some conceptual class X as a number or text in the real world, X is probably a conceptual class, not an attribute."
- Better: "If it is a characteristic of something else, make it an attribute"
- Example:
  - A *switch* has *location*, *max. ratings*, and whether it is *on or off*
  - The *electricity* is *flowing or not*
  - Which of these is/are conceptual class(es)?
  - Which of these is/are attribute(s)?
    - (attributes of what?)
- Simple data type suggests attribute

## Concepts Vs. Attributes II

- For the domain entities identified earlier, which should be *conceptual classes* and which should be *attributes*?
- For *attributes*, what *conceptual classes* should they be attributes *of*?

## Conceptual Class Names

- Consider alternate names:
  - "POST" (from "POS Terminal")
  - "Register" (as in cash register)
- A POST is one possible kind of register
  - Abstractness is good
  - So which term is better?
- Can any of our other classes usefully be given more abstract names?

## Conceptual Class Names II

- Easily understood, familiar names are good
  - (Jump ahead to section 10.9 for full discussion)
- Could any of our names be changed?

## Conceptual Classes, in concept

- Conceptual class – is domain entity
- Design class – is in the design
- Implementation class – is in the code
- Class – a conceptual class, a design class, or an implementation class...depending



Next time: Associations in  
Domain Models (Chapt. 11)