Assignment #1

Practice with UML Models

CS 4354 Fall 2012
Instructor: Jill Seaman

Due: in class **Wednesday, 9/12/2012**
Submit a "hard copy" (probably hand-written, optionally computer-generated).

Problems from Chapter 2:

Note: some questions have been modified.

2-1 Consider an ATM system. Identify at least three different actors that interact with this system.

2-3 What is the difference between a scenario and a use case? When do you use each construct (who is the intended audience)?

2-4 Draw a use case diagram for a ticket distributor for a train system. The system includes two actors: a traveler, who purchases different types of tickets, and a central computer system, which maintains a reference database for the tariff. Use cases should include: BuyOneWayTicket, BuyWeeklyCard, BuyMonthlyCard, UpdateTariff. Also include the following exceptional cases: Time-Out (i.e., traveler took too long to insert the right amount), TransactionAborted (i.e., traveler selected the cancel button without completing the transaction), DistributorOutOfChange, and DistributorOutOfPaper.
Note: A Tariff is a list or schedule of prices for such things as rail service, ... (Wikipedia)

2-6+7 Draw a class diagram representing a book defined by the following statement: "A book is composed of a number of parts, which in turn are composed of a number of chapters. Chapters are composed of sections." Focus only on classes and relationships. Add multiplicity to the class diagram you produced.

2-9 Extend the class diagram of Exercise 2–6+7 to include the following attributes:

- a book includes a publisher, publication date, and an ISBN
- a part includes a title and a number
- a chapter includes a title, a number, and an abstract
- a section includes a title and a number

2-10 Consider the class diagram of Exercise 2–9. Note that the Part, Chapter, and Section classes all include a title and a number attribute. Add an abstract class and a generalization relationship to factor out these two attributes into the abstract class.

2-11 Draw a class diagram representing the relationship between parents and children. Take into account that a person can have both a parent and a child. Annotate associations with roles and (precise) multiplicities.

2-13 (modified) Draw a sequence diagram for the warehouseOnFire scenario of Figure 2-21 (also in the lecture). Include the objects bob, alice, john, FRIEND, and instances of other classes you may need. Use the class diagram in Figure 2-22 (also in the lecture, the next diagram) as a reference. Also assume that EmergencyReport has these operations: specifyIncident(), requestResource(), and commit(), Incident has this operation: allocate(). You can add more as necessary.

2-15 Consider the process of ordering a pizza over the phone. Draw an activity diagram representing each step of the process, from the moment you pick up the phone to the point where you start eating the pizza. Do not represent any exceptions. Include activities that others need to perform. You should have about 10 actions (give or take a few). Don't forget about swim lanes. Also use concurrency for actions that do not need to be in a specific sequence.

2-16 Add exception handling to the activity diagram you developed in Exercise 2–15. Consider at least three exceptions (e.g., delivery person wrote down wrong address, deliver person brings wrong pizza, store is out of your favorite toppings).

2-17 (modified) Consider the software development activities which we described in Section 1.4 of the book (also in the Intro lecture, slide 10). Draw an activity diagram depicting these activities, assuming they are executed strictly sequentially. Draw a second activity diagram depicting the same activities occurring incrementally (i.e., one part of the system is analyzed, designed, implemented, and tested completely before the next part of the system is developed). Draw a third activity diagram depicting the first three steps as executed sequentially, but during System design various components are designed (assume there are three for simplicity). Then the remaining activities are done sequentially on each component by separate teams in parallel.