Software Engineering test - April 24, 2012

1. 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`.
2. Consider a file system with a graphical user interface, such as Macintosh's Finder, Microsoft's Windows Explorer, or Linux's KDE. The following objects were identified from a use case describing how to copy a file from a floppy disk to a hard disk: `File`, `Icon`, `TrashCan`, `Folder`, `Disk`, `Pointer`. Specify which are entity objects, which are boundary objects, and which are control objects. Which is the rationale of classifying objects in these three categories? How are represented the classes corresponding to each category in a class diagram?
3. Assuming the same file system as before, consider a scenario consisting of selecting a file on a floppy, dragging it to Folder and releasing the mouse. Identify and define at least one control object associated with this scenario.
4. Please name and briefly describe 3 modeling concepts, that do not have a 1 to 1 correspondence in programming languages. Mention how these concepts can be mapped into appropriate constructs in programming languages.
5. Describe the behavior specified in Figure 1, and explain if some information is missing or not. In case of an affirmative answer, include the missing information. Name the UML concepts used in this diagram and explain the difference between the two kind of states.

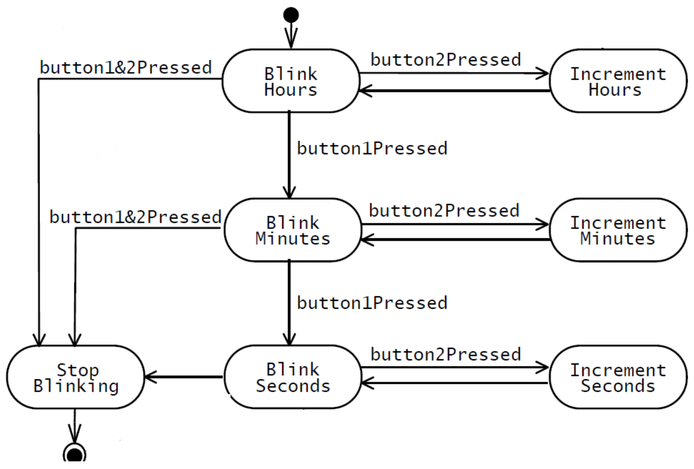

Figure 1 - UML State transition diagram

1. The following class diagrams and associated descriptions were presented as possible solution to the naive model of the Gregorian Calendar mentioned in Bruegge book:

The problems with Figure 5-24 on page 165 are related with the multiplicity of the associations. Weeks can straddle month boundaries. Moreover, the multiplicity on other associations can be tightened up: years are always composed of exactly twelve months, months do not straddle year boundaries, and weeks are always composed of seven days. Figure above depicts a possible revised model for this exercise.

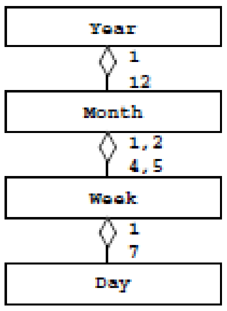

Figure 2 - Figure 5-4 A naive model of the Gregorian calendar (UML class diagram).

Consider the object model of Figure 5-24 on page 165 in the book. Using association multiplicity only, can you modify the model such that a developer unfamiliar with the Gregorian calendar could deduce the number of days in each month? Identify additional classes if necessary.

The purpose of this exercise is to show the limitation of association multiplicities. There is no complete solution to this problem. A partial solution indicating the number of days in each month is depicted in Figure 5-5. We created four abstract classes for each of the possible month lengths, 11 classes for each of the nonvariable months and two classes for the month of February. The part that cannot be resolved with association multiplicities is the definition of a leap year and that the number of days in February depends on whether the year is leap or not. In practice, this problem can solved by using informal or OCL constraints, described in more detail in Chapter 7, Object Design.

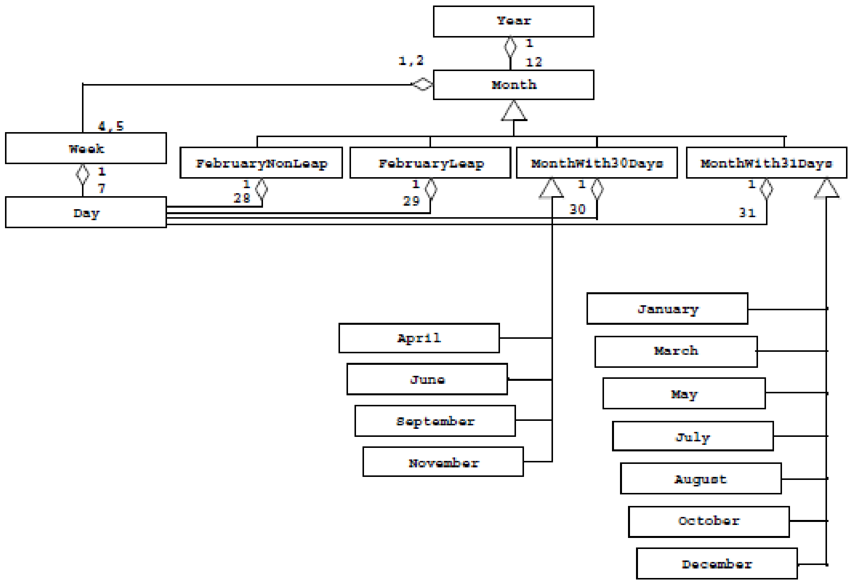

Figure 3 - A detailed solution for the Gregorian calendar

As mentioned above, even the model described in Figure 2, does not satisfy all the requirements. Which are in your opinion, the drawbacks of the model described in Figure 3? Using OCL, please describe a simpler model, and specify associated constraints and the observer that computes the week of the year corresponding to the current day.