Practice Midterm Exam: Software Engineering

Prof. Bruegge
WS 2001/2002
Out: December 13, 16:00, S1128
Due: December 14, 11:15, S1128

Last name	
First name	
Matriklnr.	
Hauptfach	
Semester	
Date of birth	

1. Modeling with UML

Hint: read the exercise completely before drawing your diagram.

a) 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.

b) Add multiplicity to the class diagram.

c) Extend the class diagram 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

d) Note that the `Part`, `Chapter`, and `Section` classes all include a title and a number attribute. Add an abstract class and an inheritance relationship to factor out these two attributes into the abstract class.

2. Project communication

You are a member of the user interface team. You are responsible for designing and implementing forms collecting information about users of the system (e.g., first name, last name, address, E-mail address, level of expertise). The information collected by these forms is stored by a storage subsystem and used by the reporting subsystem. You are not sure which fields of these forms are required information and which are optional.

How do you find out?

3. Requirements elicitation

a) Consider your watch as a system and set the time 2 minutes ahead. Write down each interaction between you and your watch as a scenario. Record all interactions, including any feedback the watch provides you.

b) Consider the scenario you wrote in a). Identify the actor of the scenario. Next, write the corresponding use case `SetTime`. Use the template provided in the book. Include all flow of events, and include setting the time forward, backward, setting hours, minutes, and seconds.

4. Analysis

Hint: read the exercise completely before drawing your diagram.

Consider the object model in Figure 1:

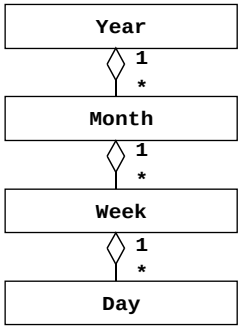

Figure 1 A naive model of the calendar (UML class diagram).

a) Given your knowledge of the calendar, list all the problems with this model. Modify it to correct each of them.

b) Using association multiplicity only, can you modify this object model such that a developer unfamiliar with the Gregorian calendar could deduce the number of days in each month? Identify additional classes if necessary.

5. Analysis II

a) 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.

b) 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.

c) Arrange the objects listed in a) & b) horizontally on a sequence diagram, the boundary objects to the left, then the control object you identified, and finally, the entity objects. Draw the sequence of interactions resulting from dropping the file into a folder. For now, ignore the exceptional cases.

d) Examining the sequence diagram produced c), identify the associations between these objects.

e) Identify the attributes of each object that are relevant to this scenario (copying a file from a floppy disk to a hard disk). Also consider the exception cases "There is already a file with that name in the folder" and "There is no more space on disk."

6. System Design

a) Decomposing a system into subsystems reduces the complexity developers have to deal with by simplifying the parts and increasing their coherence. Decomposing a system into simpler parts usually results into increasing a different kind of complexity: Simpler parts also means a larger number of parts and interfaces. If coherence is the guiding principle driving developers to decompose a system into small parts, which competing principle drives them to keep the total number of parts small?

b) In the lecture, we classified design goals into five categories: performance, dependability, cost, maintenance, and end user. Assign one or more categories to each of the following goals:

- Users must be given a feedback within 1 second after they issue any command.
- The `TicketDistributor` must be able to issue train tickets, even in the event of a network failure.
- The housing of the `TicketDistributor` must allow for new buttons to be installed in the event the number of different fares increases.
- The `AutomatedTellerMachine` must withstand dictionary attacks (i.e., users attempting to discover a identification number by systematic trial).

The user interface of the system should prevent users from issuing commands in the wrong order.