SEG2105. Introduction to Software Engineering

Assignment 4 (2.5 %)

Fall 2016

- **Assignment due: December 4th 2015 (by midnight).**
- You may work individually or in groups of two.
- You must use a software tool to draw the UML diagrams (VisualParadigm, LucidChart, Umple, etc).
- You need to submit a report with all your answers. Don't submit any tool-related files.

Notes:

- Feel free to ask questions if you don't understand the following system description or write down your assumptions.
- **Use the Discussion Board to ask questions related to this assignment.**
- Do not leave this assignment to the last day; you would undoubtedly do badly if you do.

PART I (25 points)

A. [10 marks] **E152 -** b, c

B. [15 marks] **Create a state diagram for the states of a specific flight. Refer to the Airline example in the book, page 209.**
The most common sequence is as follows: A flight starts in planned state. It is next assigned a plane, and after that boarding starts when the captain declares the plane is ready for passengers. After boarding is complete, the gate is closed and the captain awaits permission to depart. The plane is then said to be 'underway'. Upon arrival at the gate of the destination airport, the flight is in 'disembarking' state until all the passengers are out. At that point the flight is declared completed. However some other possible things can happen.
A flight can be underway, but never take off and be sent back to the terminal again (e.g. if there is bad weather and the pilot is never given permission to take off). A flight can even take off and then return to the original airport for some reason.
The flight can have more than one leg. In this case after passengers have got out, some passengers may remain in the plane. When the crew is ready they will then let new passengers board.
A flight can be cancelled in most states. However, if there are passengers on board, the flight must 'complete' first, i.e. it must go to a gate and the passengers must all disembark before it is declared cancelled.
Passengers may all be asked to leave the plane, even if it has not left the gate. This might be because it is about to be cancelled, or because the airline needs to change the plane (e.g. because a mechanical problem has been discovered).
The plane can be changed any time passengers are not on board (unless the flight is already complete).
In extreme cases, passengers may be asked to leave the plane even if not at a gate (e.g. after an emergency landing on the runway).

PART 2 (50 points)

A. [20 marks] **Create a UML class diagram for the Decision-Making system described below.** Make sure you include correct multiplicity. Show all attributes and associations plus at least five important operations. If generalizations are necessary, show them too. Marks will be given for effort, even if you don't have a perfect solution. However, marks will be lost for the common types of mistakes we talked about in class (e.g. poor generalizations, wrong multiplicity, etc).
B. [10 marks] Improve your class diagram by applying two of the **patterns** discussed in Chapter 6. Describe **only** the changes to be made in your initial class diagram.
C. [20 marks] Design a **state diagram** describing the states a *Decision* instance can be in. You must use the following states and events. Feel free to add more events and/or states, if needed.

Initial state

- Under construction

Other non-final states

- On agenda (when it is ready to be considered and is on the agenda of one of the committees)
- Under reconsideration (when a higher-level committee has not accepted it and has pushed it back down to a lower level for further work)

Final states:

- FinalApproved (when approved by the highest-level committee that needs to approve it)
- Rejected
- Dropped
- Split

Events:

- Put on agenda
- ApproveAndMove
- ApproveFinal
- ReturnToLowerLevel
- Reject

A Decision System – SYSTEM DESCRIPTION

Many organization have decisions to make that are made by various committees in a hierarchical manner. Committees exist at various levels. For example, there might be a departmental committee that makes a decision, and passes the decision up to a division council for further approval, and that then passes the result to the board of directors for final approval.

Some committees might consist of a single individual, e.g. a department head or director.

The system you will model needs to be able to handle any kind of organization; the committees or individuals will have different names in each organization. In a university, for example, there might be many levels: Department curriculum committee, department council, faculty curriculum committee, faculty council, undergraduate council and Senate.

Decisions are of various types, with each type having an importance levels. Based on its importance, a decision only needs to be approved up to a certain level in the hierarchy. For example, the decision about which courses to offer in a given year might only need to be approved by the department curriculum committee, but a new program of studies might need to be approved by the Senate (and every level from department curriculum committee up to Senate). In a company, an expense less than $1000 might need approval by a department head. Expenses of over $1000000 might need approval by the Board of Directors.

Each decision has a 'short summary' that describes the decision. Examples might be: 'Purchase Photocopier model AB309 from Xerox for $2570' or 'Create course SEG2107 with title: "Model driven development", and description: 'Model driven development using Umple', and prerequisites: "SEG2105"'

Short summaries can have multiple versions (as changes are made).

Each decision has associated with it a set of documents. Each document has an id, a filename, a type (MS Word, Excel, Pdf etc.) and a set of versions.

When a committee approves a decision, it does so by approving the short summary, and possibly modifying the set of documents by adding new versions of existing documents or adding new documents.

A committee might turn down a decision based on one version of a document and request a list of change requests that are simple blocks of text. Users or other committees may add comments to the change requests and may create new versions of documents and an updated version of the short summary. A decision may therefore go up the hierarchy, be rejected, come back down for revisions, go back up again, back down again, and eventually be approved at the highest needed level, or rejected or dropped at any level.

It is also possible for a committee to split a decision into separate decisions, or merge two decisions together, or approve one decision conditional on approval of one or more other decisions.

Committees have members. Each member has a start and end date on the committee. Each document has a set of authors. Each version has a date and a set of authors.