



Object-Oriented Analysis and Design

Exam Winter Semester 2007/2008

February 19th, 2008
Software Systems Institute
TUHH

Name: _____

Student Number: _____

Rules:

1. The exam is closed book. That means that the only things you are allowed to have on your desk or use during the exam are pens and the exam itself.
2. All phones off. A switched on phone is considered cheating.
3. Keep your eyes on your own work.
4. Cheating will cause you to fail this exam.

Additional information:

- The exam will take 90 minutes.
- Each problem is associated with a number of points. This is also the amount of time in minutes we expect you to take for the answer (the total sum is 75).
- Please put your student identification as well as a passport/official id card on the table. We need to check these.
- Don't forget to put your name and student number on each page.
- If you draw into existing diagrams, please think *before* you draw to keep your answer readable.
- We took the utmost care to make the English and German version semantically identical. In case of doubt, you may inspect both versions on the front desk.
- We have more paper, should you need some, ask

Do not start reading this exam until instructed to do so

Task 1: Development Process.

(a) Describe three iterative phases of the Unified Process. (6 pts)

(b) In the context of the Unified Process, explain the following statements:
“Construction builds the system in a series of iterations. Each iteration is a project in itself.” (4 pts)

(continue your answer on the reverse of this sheet if necessary)

Task 2: Activity Diagrams.

(a) Describe the process depicted in Figure 1, paying attention to the temporal ordering of events and activities. (5 pts)

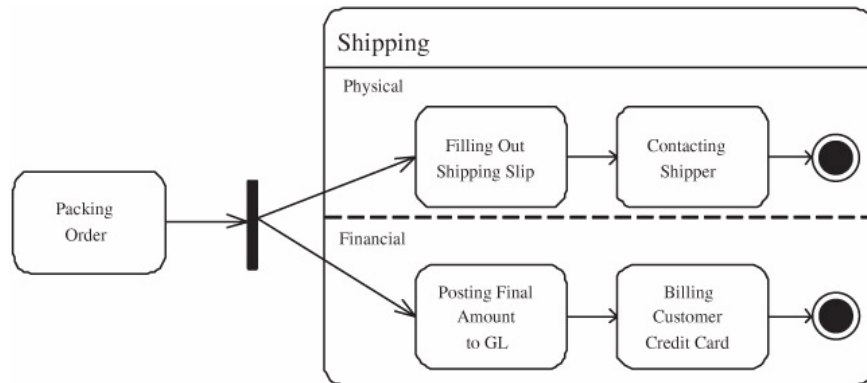


Figure 1

Task 3: Diagrams

Please indicate whether the following statements are true or false. (Points are discounted for wrong answers, so if in doubt you may consider leaving fields blank instead of randomly ticking a checkbox) (6 pts)

	<i>true</i>	<i>false</i>
A collaboration diagram can depict interactions that are not possible to show with statecharts.	<input type="radio"/>	<input type="radio"/>
Objects that do not react to any events have a statechart with a single state.	<input type="radio"/>	<input type="radio"/>
A class diagram can be subclassed by a statechart to model behavior.	<input type="radio"/>	<input type="radio"/>
For each use case there should be at least one activity diagram.	<input type="radio"/>	<input type="radio"/>
A deployment diagram shows whether a system is distributed or not.	<input type="radio"/>	<input type="radio"/>
Both conceptual-level and implementation-level class diagrams are possible.	<input type="radio"/>	<input type="radio"/>

(continue your answer on the reverse of this sheet if necessary)

Task 4: Statecharts.

The control software for two robots is being designed. These robots should play the PacMan game, and therefore there are two kinds of robots: PacMan and Ghost. The objective of PacMan is to travel around all paths in a maze, collecting coins as they are found. The objective of a Ghost is to travel around the maze and find PacMan.

Each robot has limited sensory capabilities, as follows:

- There is a front sensor that can report whether walls exist (left, right, ahead) within one distance unit. Based on this information, the robot may decide where to turn next.
- There is a proximity sensor (ultrasonic) that can report in line-of-sight (ahead of the robot) whether a coin or Ghost is located. The range of visibility is five distance units, and the precision is up to one distance unit. This sensor does not report whether the object is moving, or in which direction. However, two consecutive readings showing a sensed object in different positions may be used to derive whether the detected object is moving (one should also take into account whether PacMan is moving).

Additionally,

- Each robot has a timer, which allows the scheduling of events (for example, “in two seconds, raise event X”). The timer ticks 50 times a second, and this is signaled by means of an event.

The actions that PacMan can perform are the following:

- Rotate 90° (left, right)
- Move one distance unit (forwards, backwards)

You may compose the above actions into more complex ones (e.g., `turnAround()` consists of two rotate-left actions). Moreover, you may assume that utility functions for arithmetic, date and time, etc. are available in Java libraries.

Your tasks are:

- (a) draw a UML class diagram to represent the data structures involved in this software. As a minimum, the diagram should contain classes for the maze, PacMan, and Ghost. You may model additional classes you consider necessary for PacMan to reach its objective (for example, classes for objects manipulated in PacMan’s memory). Class depictions should include attributes and operations. (10 pts)
- (b) draw the statechart that specifies the behavior of PacMan. Such behavior should allow it to achieve its objective. The events to process are only those that can be reported by the sensors, as described above. (15 pts)

Task 5: Alloy

An application manages binary trees, i.e. trees where nodes have at most two children (labeled `left` and `right`). These trees are not necessarily complete: some nodes may lack one or both children. The trees are also not necessarily ordered (a subtree `T` is ordered iff each value in a node in its left branch is \leq the value at the root of `T` and each value in a node in its right branch is \geq the value at the root of `T`).

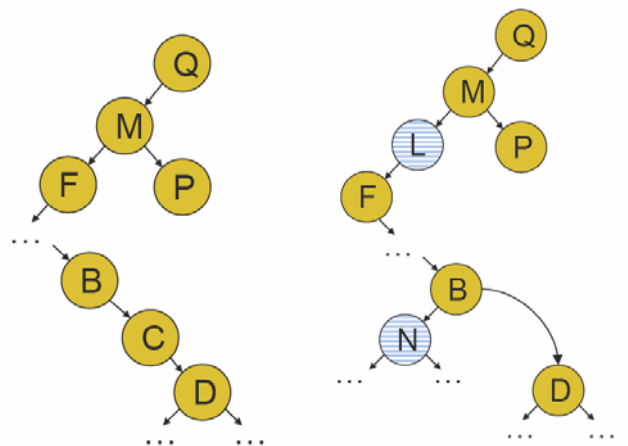
Remembering that in Alloy there are only relations over uninterpreted symbols (“atoms”), formulate the following in Alloy:

(a) a relation to represent binary trees, the values of whose nodes come from a set `V`, over which a total order `LE` has been defined (5 pts)

(b) a predicate indicating whether the subtree received as argument is ordered (5 pts)

(c) predicates that allow updating binary trees (one predicate to insert a node and another to remove a node from a tree). As usual in Alloy, such predicate are written with one argument representing the tree in the pre-state and another argument for the tree in the post-state. Each such predicate establishes the input-output relationship between pre- and post-state, based on additional arguments (if necessary). For illustration, three representative updates are shown in Figure 2, namely the insertion of two nodes (`L` as left child of `M`, and `N` as left child of `B`) and the deletion of one node (`C`). (12 pts)

In case you’re unsure about some aspect of the Alloy textual notation, you may answer using English, using the set and relational terminology that can be found in books on, for example, Discrete Mathematics. However, you may not make up concepts not present in the Alloy formalism: doing so does not count as a valid answer.



(a) Tree before update (b) After insertion of L, N and deletion of C

Figure 2

(continue your answer on the reverse of this sheet if necessary)

Task 6: Change and revision systems.

Describe in detail the operation of a change and revision system.

(7 pts)