DM519 Concurrent Programming

Spring 2015 Exam Project

Department of Mathematics and Computer Science
University of Southern Denmark

February 22, 2015

Introduction
The purpose of the project for DM519 is to try in practice the use of models in the design and implementation of concurrent programs.

Please make sure to read this entire note before starting your work on the project. Pay close attention to the sections on deadlines, deliverables, and exam rules.

Exam Rules
This project is an exam. Thus, the project must be done individually, and no cooperation is allowed beyond what is explicitly stated in this document.

Deliverables

- A short project report in PDF format (5-10 pages excluding front page and appendix) has to be delivered. This report should document the result of has to contain at least the following 4 sections:
  - **Modelling:** design decisions, FSP model, structure diagram
  - **Analysis:** absence of deadlocks, safety, liveness
  - **Implementation:** threads vs monitors, relevant parts
  - **Testing:** correctness of implementation
- FSP model as .lts file
- Java source code as .java file

A preliminary version of the report describing at least the results of Tasks 0-4 as described below has to be handed in together with the preliminary FSP model as .lts file until the“pre-delivery” deadline given below.

The full version of the report, the final FSP model and the Java source code have to be handed in until the “final delivery” deadline

The deliverables have to be delivered using Blackboard’s SDU Assignment functionality. Delivering by e-mail or to the teacher is only considered acceptable in case Blackboard is down directly before the deadline.

Deadlines

pre-delivery: March 15, 2015, 23:59

final delivery: April 12, 2015, 23:59

The Problem

IMADA has won the bid for designing the new elevator system for the main elevator of Area 51 secret underground base. The classified specification is given on this page.

The secret base has a total of 4 floors:

G the ground floor with an exit into one of the airfield hangars

S secret floor housing the nuclear weapons control unit

T1 secret floor housing the experimental weapon development unit

T2 top-secret floor housing the alien conservation & experimentation unit

On each floor there is a button to call the elevator to that floor. Likewise, inside the elevator there are four buttons for G, S, T1, and T2. When any button is pressed, all other buttons are deactivated until the target floor is reached. Thus, a memory of the buttons pressed is not necessary for this elevator.

There are three levels of security clearance at Area 51 (lower levels are disallowed from the base and do not have to be considered): Confidential, Secret, and Top-Secret. Each agent arriving at the elevator calls the elevator to the current floor, enters the elevator, has their retina scanned, presses buttons in the elevator any number of times, and leaves the elevator.

The retina scanner is connected to a database, where the clearance level for each agent is stored. The elevator may not open its door at S, T1, or T2 if a person with the lowest clearance level Confidential is present. Likewise, the elevator may not open its door at T1 or T2 if a person with the clearance level Secret is present.

A possible model for an agent given a range of indices FLOORS for the floors and a range of indices LEVELS for the clearance levels could be:

```
AGENT = (call[FLOORS] -> enter -> scan[LEVELS] -> INSIDE),
INSIDE = (call[FLOORS] -> INSIDE | leave -> AGENT).
```

It is sufficient to allow only one agent to be in the elevator at any time. If you choose to allow more than one agent entering the elevator at the same time, make sure that the person with the lowest security clearance is considered for decisions where the elevator may open its doors.

Your Task

Your task is to implement the elevator system as a Java program and test it by letting agents of different security clearances use the system repeatedly.

To this end, you are all expected to perform the following tasks:

0. Read this description very carefully. You will probably find that some details are underspecified. You can make your own choices, but try to keep them meaningful.
1. First model the elevator, the buttons, the retina scanner, the agents, and their interactions without any restrictions. Let the elevator start empty at the ground floor.
2. Observe by executing the model that the elevator may move to a secret floor with a agent classified as Confidential and to the top-secret floor with a agent classified as Secret.
3. Add an elevator control process that ensures that the elevator does not open its doors when a person of a too low clearance level is in it.
4. Make a structure diagram of your system.
5. Add a safety property NO_SECURITY_BREACH formally verifying that a agent of a certain security clearance cannot leave the elevator on a floor with a higher security requirement. Verify this using the LTSA tool.
6. Check your model for deadlocks. If there are any, break them in a meaningful way.
7. Structure your model into an implementation. Which processes should be implemented as (active) threads and which as (passive) monitors?
8. Implement your model in Java. As usual, try to reuse action names as method names in order to make the connection between the model and the implementation obvious.

For the remainder of the project, you can choose between two lists of tasks: MORE MODELS and GRAPHICAL USER INTERFACE. You have to chose one of the lists. The following tasks are expected to be performed when you chose the MORE MODELS list:

9. Expand the elevator to keep track of the number of agents in it.
10. Exceeding the elevator's capacity $C = 2$ should blocks the buttons on the floor and the elevator until enough agents have left the elevator. Add a second controller to the elevator such that the elevator can no longer move when the capacity is exceeded.
11. Add a safety property NO_CAPACITY_EXCEEDED formally verifying that the capacity is not exceeded when moving. Verify this using the LTSA tool.
12. Add a liveness property ALL_EXIT formally verifying that any agent entering the elevator can eventually leave it at some floor. Verify this using the LTSA tool and adapt your model and implementation, if necessary.

Alternatively, you can choose the GRAPHICAL USER INTERFACE list, where you are expected to perform the following taks:

9. Using Java's Swing GUI classes (`javax.swing.*` and `java.awt.*`), implement a window that graphically represents the four floors as well as the elevator and its position.
10. Ensure that the agents testing the system are updating the graphical view with their actions. Hint: a `GridLayout` and `JLabel` may be very useful here.
11. Add buttons (e.g. by using `JButton`) to the floors and the elevators. For each floor and the elevator, add representations of the agents on it (e.g. also by using `JButton`). Buttons pressed should move the elevator. Agents pressed should move them into or out of the elevator.
12. Add a possibility to control the number of automated agent threads that are testing the system (e.g. using a `JSpinner`) and add a possibility to add a new agent of a given security clearance to a given floor (e.g. by a `JComboBox`).