DM519 Concurrent Programming

Spring 2013 Exam Project

Department of Mathematics and Computer Science
University of Southern Denmark

February 24, 2013

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 (4-8 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 0-3 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 8, 2013, 12:00

final delivery: April 5, 2013, 12:00

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

S1 secret floor housing the experimental weapon development unit

S2 secret floor housing the nuclear weapons control unit

T 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, S1, S2, and T. When a call button is pressed, all other call buttons are deactivated until the target floor is reached. Thus, a memory of the buttons pressed is not necessary for this elevator. Here is a possible model of one of these buttons.

```
// example FSP for an elevator call button
BUTTON = (call -> BUTTON).
```

The elevator has a retina scanner that scans the retina of each person entering and leaving and looks up their security clearance in a database. 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. The elevator may not move to S1, S2, or T if a person with the lowest clearance level "'Confidential" is present. Likewise, the elevator may not move to T if a person with the clearance level "Secret" is present.

Your Tasks

Your task is to implement the elevator system as a Java program with the ability to test its function by letting persons of different security clearances use the system.

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

0. Read this description very carefully. You will probably find that a lot of details are underspecified. You can make your own choices, but try to keep them meaningful.

1. First model the elevator, the floors, the call buttons, and their interactions without any restrictions. Let it start empty at the ground floor.

2. Observe that the elevator may move to a secret floor with a confidential person and to the top-secret floor with a secret person. Add a safety property SECURITY_BREACH formally verifying that a person of a certain security clearance cannot leave on a floor with a higher security requirement.

3. Add the CAMERA process and adapt the elevator control such that the elevator only moves to floors for which the person is cleared. Show that there cannot be a security breach any longer. Verify this using the LTSA tool.

4. Add a liveness property SECURE_EXIT formally verifying that a person of "Confidential" security clearance eventually can leave the elevator at the ground floor. Verify this using the LTSA tool and adapt your model, if necessary.

5. Check your model for deadlocks. If there are any, break them in a meaningful way.

6. Make a structure diagram of your system (without the safety and liveness properties).

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.

It is sufficient to allow only one person to be in the elevator at any time. If you choose to allow more than one person 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 move etc.