EFREI M1

SCV

Lab 1

Test VS Formal Verification

In the following we propose two approaches in order to implement a partial application that manages flights and resources of a small airport. The first approach consists in, first, designing then implementing the system using an appropriate programming language. The validation of the implementation will be done by testing and analyzing the execution through chosen scenarios. The second approach uses Petri nets to model the system and to check the expected properties before the implementation. In this part of the tutorial, you will use TINA (TIme petri Net Analyzer) toolbox (http ://www.laas.fr/tina/). This toolbox supplies several tools allowing to edit, to simulate and to analyze Petri nets.

Note : : Feel free to read the documentation of TINA and test the editor on simple examples before processing the modeling phase of the resource management system.

Question 1 :

Consider a resource management system in a small airport. There are 4 boarding rooms and 3 gateways. For flights on departure, the procedure is in two steps. First of all, you have to reserve a free boarding room to accommodate passengers. Then, while maintaining the reserved boarding room, it is necessary to reserve a free gateways to be able to embark the passengers. A boarding room/ gateway is no longer available for another flight. Finally, once the embarkation is complete, the boarding room and the gateway are released.

For flights on arrival, we consider two kinds of arrival :

— A normal arrival where it is necessary to reserve a gateway available to disembark passengers. Once the procedure is completed the gateway used is made available for other flights.

— An arrival with a quick stopover where a gateway is first reserved and then a boarding room is reserved in order to not stack passengers who are about to continue their travel.

1. Design-Implementation-Test
   (a) Write a program (you have the choice of the appropriate programming language) that allows to simulate the behavior of the flights on arrival and on departure with respect to the shared resources. You can associate to each departure a thread and write a main program that represents the resource management system.
   (b) How can you check the correctness of the program with respect to fairness (each flight will eventually get the required resources), and deadlock freeness (the system will never be in a state where none action is possible) ? Perform this verification by displaying messages on the screen and by checking some execution scenarios.
2. Formal Modeling-Formal Verification-Implementation
   (a) Model the resource management system
   (b) Express and check the two above properties formally using the TINA toolbox.

3. Compare the two previous approaches. If you implementation does not satisfy the requirements, fix the problem.