

Assignment 3

(E)FSMs

Model-Based Testing
DIT848/GU and TDA260/Chalmers

Spring 2014

1 Introduction

The purpose of this assignment is to exercise writing your own extended finite-state machine models for software testing.

2 Submitting your work

You have to submit your work through the Fire system <http://xdat09.ce.chalmers.se/mbt/>. Upload your source code and txt or pdf file describing your answers.

The deadline for this assignment is **Wednesday 7 May 2014**.

3 An FSM for the Fire system

In this first exercise, you need to create a finite-state machine model to test the Fire system (the system you have been using to submit assignments). Your tests will be limited to the student part of the system. Your model should test the following specification:

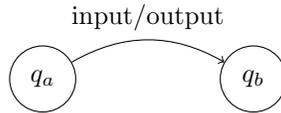
- The student needs to log in before any other action can be performed.
- Once logged in, the student sees the "Laboration overview" page where she can choose a lab.
- A lab can be either submitted or non-submitted. A submitted lab must be withdrawn before a new submission can be made.
- Before submitting, the student can upload an arbitrary number of files which will be included in the submission.
- The student can log out at any moment.

Here is, in plain English, an example of a test case that should be generatable from your model:

1. log in the system using the given username and password
2. click on "Lab 1"
3. upload a file
4. upload a file
5. logout
6. log in again
7. click on Lab 1
8. submit
9. withdraw
10. logout

For this first exercise, you can assume: that you are given a username/password for testing (i.e. no need to register in the system); that there is only one lab with title "Lab 1" ; that the lab has no deadline.

The model should be an FSM (no EFSM!), more precisely, it should be a Mealy Machine, i.e. each transition should be labeled with an input and an expected output like:



You will need to draw the automaton and define the input and output alphabets. For instance, the following is a very simple model that only tests login in and out of the system:

Input alphabet:	Output alphabet:	Automaton:
<p>login Fill in the given user-name and password and click the login button</p> <p>logout Click the “logout” link in the drop-down menu in the top right corner.</p>	<p>”text...” The text inside the quotes should be present in the new page</p>	

Your submission should include: the input alphabet, the output alphabet & a schema of your automaton

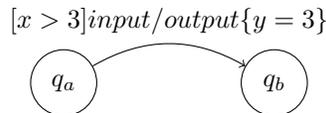
4 Extending to an EFSM

Here you need to extend your finite-state machine model to an extended finite-state machine. In addition to all the action above, this second model should also handle deleting a (non-submitted) file.

Your model should still be able to upload an arbitrary number of files but should also be able to delete files before submitting (but shouldn't try to delete files when there is no files to delete!) As an additional requirement, your model should never submit an empty lab (without any files).

Your submission should include: Any new input/output symbol you need (you don't need to repeat the whole set, just the additions), the variables used and their initial values, a schema of the *complete* EFSM model.

Here is an example of an EFSM transition where $[x > 3]$ is the guard, *input/output* is the same as in the FSM (the action to perform and the expected result) and $\{y = 3\}$ describes how the variables should be updated when the transition is taken.



4.1 The car computer

In this last exercise, we will model a wireless communicating system between a car on board computer and a smart phone. The smart phone is used to open and close the car, and receives statistics from the car computer, automatically transferred to the mobile phone when this is detected on a given radius of proximity (after having opened the car with the phone).

Due to security concerns this transmission is not done to any mobile phone in the range of the wireless communication, but only to the phones of the owners (which have been previously registered in a database in the car computer). It is assumed that the car computer already has those phones registered in the database.

Draw an Extended Finite-State Machine (EFSM) for the car computer according to the following specifications:

1. The car computer keeps waiting on a stand-by state till a mobile phone is detected.

2. If the mobile phone is not in the database then it is ignored.
3. If the mobile phone is already registered (it is in the database) then a connection is established and a window asking for a code is shown in the screen of the phone.
4. The user of the phone can only have 3 failed attempts to enter a correct code; if the correct code is entered then the car is opened; after failing 3 times the phone is blocked.
5. After the car is opened (due to the input of the correct code on the phone, not when opened with a normal key) then certain predefined data is automatically uploaded into the mobile phone.
6. After the data is transmitted there is a timeout for the engine: it must start within 15 min, otherwise the car is automatically closed and the car computer goes to standby.
7. If the engine starts within 15 min then the car computer goes to standby.

Give 2 test cases that can be extracted from the above EFSM, and two that cannot be extracted from it (in the latter case provide test cases that you think might be useful to further test your system, though they cannot be extracted from the EFSM as it is too abstract).

Your submission should include: a drawing of the EFSM, the list of variables and their initial values, an explanation of the labels used in the model & a short description of the 4 test cases.