Lab Tutorial 7

Model-based specification, verification and validation

While reviewing the fundamental principles of verification and validation in model-based system development, this lab provides practical instructions on how to systematically analyze and test design solutions using the tools of Simulink/Matlab. These tools are: 1. Verification and Validation Simulink Toolbox and 2. Design Verifier Simulink Toolbox. From a methodology point of view, as a support for quality assurance, this related verification and validation tasks are normally performed prior to automatic generation of object code.

1. Preparation

1.1 Prerequisites

Before you start the main part of the lab, it is recommended that you refresh your knowledge in Stateflow in Simulink. Please, visit the following page:

http://se.mathworks.com/help/stateflow/getting-started.html

and perform this tutorials first (for fuel control system):

http://se.mathworks.com/help/simulink/examples/modeling-a-fault-tolerant-fuel-control-system.html?prodcode=SF&language=en

- What is the state flow and what is the difference between the state flow and timed automata? How can you model timing transitions in a state flow?

1.2. Practical setup

This lab exercise is based on a tutorial provided by Mathworks that you will have to download from the following web page:

https://www.kth.se/social/files/5731f2b8f276541b6a7685cf/vvtWorkshopProject-2.zip

Before you start, unzip the downloaded zip file from Matlab, make all the folders and subfolders part of the path, and run m-scripts from DataDictionary directory. Then, you can start by running “vvtWorkshop.prj”. Matlab will open in a browser the tutorial description.

3. Practical Accomplishment and Tasks

The lab exercise covers a verification and validation process composed of 8 sequential steps. Some steps will have to be conducted differently from what is explained in the original tutorial by Mathworks. For reporting accomplishment of the lab, you have to demonstrate and explain all these steps. We provide suggestions below.

Please note that you will need to use Matlab 2015b for Steps 0-2 and Step 7. For Steps 3-6, you will have to use Matlab 2016a (Trial). This is because Matlab 2015b does not presently have a license for Design Verifier and Matlab 2016a is unable to generate code.

3.1 Step 1: Requirements Traceability

Demonstrate that you have accomplished all the steps, including code generation.

- *What is the purpose to provide requirement traceability down to the code level?*
- *What if you have a piece of code (or a model block) that is not linked to any requirement? What can it mean?*
- *Name a few static code metrics that can be used in model-based design?*

3.2 Step 2: Modeling Standards

Design your own check based on one of those 3 "custom checks" provided (you can simply modify a "font check", for example). Demonstrate that you managed to "fix" check violations and accomplished all the steps in this part of tutorial.

- *What are those standards: ISO 26262, IEC 61508, EN 50128, Do-178C and Do-331? What is the difference between standards and Guidelines?*
- *What is the difference between MISRA C and MAAB Style Guidelines?*

3.3 Step 3: Detecting Design Errors

From now on, you have to switch to Matlab 2016b. You can use the same folder. Don't forget to re-run configuration scripts and add folders and subfolders in vvtWorkshops to the Matlab 2016a path.

Note that, when running the tutorial, you won't get any error as it is expected. All the checks will pass. To make a "bug", you should manually change the value in the **press_thresh** parameter to **1.5** (you can do it directly in the Workspace window).

Try to introduce more errors by, for example, changing one of "0" in the State Flow to "-1". You will observe multiple violations.

You will need to accomplish tutorial steps and *explain "bug" taxonomy.*

- *What kinds of "bugs" (design errors) are possible?*

- *Which ones are the hardest to detect and why the software support is nee[illegible]?*

3.4 Step 4: Testing by Simulation

In Step 4, you will perform testing by simulation.

You will need to show that you have completed all the steps and achieved 100% model coverage for at least 3 modules (by manually adding more functional tests).

Note: `createCompareReportsFromSDI('Step_04_logic_harness')` command will probably not work for you (in the “Automatic Verification of Simulation Results using Simulation Data Inspector section”) and you can skip it.

Answer on the following questions:

- *What is MC/DC coverage and why this is different from Path coverage? Which safety standard explicitly demands MC/DC coverage? Can the Path coverage be 100%? Can MC/DC coverage be 100%?*
- *What are the other two coverage metrics? What is the difference between them?*
- *What is the assertion? What is useful about assertions? Can you think about an alternative option?*

3.5 Step 5: Test Case Generation

Accomplish all the tutorial steps and demonstrate the final test report to the lab assistant. Note that you may experience some issues with Matlab utilities and that it may require a bit of patience to complete this step from your side.

- *Are all the “bugs” covered?*

3.6 Step 6: Property Proving

This step is about property proving. When you are done with all the tasks in this step, reason with your colleague about:

“Is this really a violation, considering it is only for one time step? Because of how we constructed the property or requirement, the answer is: Yes. The bigger question is, is our requirement complete and accurately describing the intended behavior? Based on the answer to that, we can either fix the algorithm, or update the property (based on the modified requirement), for example, to allow such condition to exist for a specified number of time steps.”

- *What will you do in this case?*
- *What is the property proving? What types of model checking exist?*
- *Can you prove also timing properties?*

3.7 Step 7: Code Verification

This is the last step of the tutorial that will teach you performing SIL and PIL testing.

Here, you have to change back to Matlab 2015b and, in addition, to re-load the tutorial (you can, for example, rename the directory of your tutorial and unzip the tutorial file again with Matlab 2015b). This is because Matlab 2016a modifies the files and they are no longer readable with Matlab 2015b.

To make this step of the tutorial work, you will need to align configurations of the model-under-test and the test wrapper. This is suggested to change hardware configuration to your Arduino board used in Lab 1 and Lab 2. You can even try to download the code on the board (although it is optional).

The questions are as follows:

- *Why the wrapper and the model should have nearly the same configuration?*
- *What is the difference between SIL and PIL testing?*
- *To make it HIL, which modifications can be necessary?*

4. Summary

In this lab assignment, you have learned to use verification and validation toolboxes for Matlab Simulink and got familiar with coverage metrics and testing methods, including property proving. Verification and validation is an important part of any development process and it is sometimes claimed that about 80% development cost is spent on testing. Early testing e.g. on model and requirement levels can help to reduce this cost. In Agile, for example, Test-Driven Development (TDD) is a must and each development task is always followed by a testing task. It is one of the reasons why Agile become more and more common even for traditional development processes in automotive and other industries.

- *What is Agile development?*
- *What is Test Driven Development (TDD)?*