Software Testing, SS 2022

Prof. Dr. Reinhold Plösch, Mag. Rudolf Ramler

Assignment 4: Model-Based Testing

Deadline: **May 9, 2022, 23:55**

The goal of this exercise is to extend the knowledge about testing towards approaches and tools for automatically generating test cases using models.

Note: This assignment is based on the result from the previous assignment, where you developed a model for the class RingBuffer in form of a state machine. Use (and adjust/extend if necessary) the state diagram for RingBuffer from the last assignment.

- Include the (updated) state diagram as part of your submission for this assignment.

Assignment Part A: Model Implementation

The framework *ModelJUnit* is used for generating test cases from your state model implemented as Java class[1]. Implement an executable version of your model for *RingBuffer* named *RingBufferModel.* Use *ModelJUnit* to generate sequences exploring your model.

- Implement actions for all transitions related to RingBuffer methods, e.g., *enqueue, dequeue, peek*
- The model should reflect the states of the RingBuffer (e.g., *empty, filled or full*). Member variables (e.g., *counter*) can be used to keep track of how many elements are in the buffer and to determine the current state of the buffer in the function *getState().*
- Use guard functions to make sure that the actions dequeue is only called if the RingBuffer is in a state where this action is valid (e.g., *filled* or *full*)
- Add additional actions for negative scenarios like *dequeueFromEmptyBuffer* (and a matching guard function) to verify that the expected exceptions are thrown

Run the model via the associated JUnit test *RingBufferModelTest*. Copy the console output into a text file named *RingBufferModel_Output.txt* for submission.

Assignment Part B: Model with Adapter

Extend the *RingBufferModel* with an adapter to actually call the methods of the class *RingBuffer*. Name the resulting model *RingBufferModelWithAdapter.*

- The model should extend the existing actions with calls to the corresponding methods of the class RingBuffer, and the resulting response (i.e., returned values, thrown exceptions) should be compared to the expected response.
- Make use of the model's state variable (*counter*) to assert the correct response; use JUnit asserts
- In case of expected exceptions, make sure that (1) exceptions are actually thrown and (2) they are of the correct type and contain the correct message.
- Furthermore, the actions *size* and *isEmpty* should call the corresponding method of the class RingBuffer. They will not change the state of the RingBuffer. However, they allow asserting the response by a comparison to the counter variable whenever triggered.
- Write a unit test *RingBufferModelWithAdapterTest* to exercise *RingBufferModelWithAdapter*.

Run the test and measure the code coverage. Experiment with different lengths for the generated test sequences – set as parameter of *tester.generate()* – to achieve a maximum coverage of the called methods. Include the coverage report in the submission.

[1] For details see: M. Utting and B. Legeard: Practical Model-Based Testing: A Tools Approach. Elsevier,2007.

Submission & Presentation

Hand in the source code of your tests and any related files you created as zip archive and **upload it to the Moodle course platform before the deadline**.

Work in teams. Only one upload per team is required, i.e., one team member uploads the files to Moodle.

The work has to be **presented in the lecture on Thr, 12. May 2022**. Each team has to be prepared to present their results. The teams that present will be selected at the beginning of the lecture. No slides (ppt etc.) have to be prepared, just bring a laptop with a development environment including all the tools you used and the source code of your solution.