Tutorial for Manual Drive

OSMO Tester

MBT tool

v3.3

Teemu Kanstrén

Table of Contents

[Introduction 3](#_Toc372289293)

[Creating specific tests manually 3](#_Toc372289294)

[Conclusions 8](#_Toc372289295)

[References 8](#_Toc372289296)

# Introduction

This tutorial describes the manual test generation with OSMO Tester. The reader should be familiar with the information presented in the OSMO Tester basic tutorial and data modeling tutorial.

The reader is also expected to have basic knowledge of Java programming and ability to use their own favourite IDE such as Eclipse, IntelliJ, or Netbeans. The code shown in this tutorial is available in the OSMO Tester examples package.

# Creating specific tests manually

Previously in the data tutorial we created a model that prints “HELLO” and “WORLD” and uses ValueSet and ValueRange data model objects. In this tutorial we use the model program from the data tutorial as a basis and show how to manually create specific test cases from this model. As a reminder, Listing 1 shows the model program that was developed.

public class HelloModel {

private int helloCount = 0;

private int worldCount = 0;

private ValueSet<String> names = new ValueSet<>("teemu", "bob");

private ValueSet<String> worlds = new ValueSet<>("mars", "venus");

private ValueSet<Integer> sizes = new ValueSet<>(1,2,6);

private ValueRange<Double> ranges = new ValueRange<>(0.1d, 5.2d);

@BeforeTest

public void startTest() {

helloCount = 0;

worldCount = 0;

System.out.println("TEST START");

}

@AfterTest

public void endTest() {

System.out.println("TEST END");

}

@Guard("hello")

public boolean thisNameReallyIsIrrelevant() {

return helloCount == worldCount;

}

@TestStep("hello")

public void sayHello() {

System.out.println("HELLO "+names.balanced()+" ("+sizes.random()+")");

helloCount++;

}

@Guard("world")

public boolean thisNameIsIrrelevant() {

return helloCount > worldCount;

}

@TestStep("world")

public void sayWorld() {

System.out.println("WORLD "+worlds.next()+" ("+ranges.next()+")");

worldCount++;

}

}

Listing 1. The model program from the basic tutorial.

Similarly, Listing 2 shows the configuration we set up to run the model program.

public class Main {

public static void main(String[] args) {

OSMOTester tester = new OSMOTester(new HelloModel());

tester.addTestEndCondition(new Length(5));

tester.addSuiteEndCondition(new Length(2));

tester.generate(52);

}

}

Listing 2. Running the model program.

And as a final reminder, the output from running this model program is shown in Figure 1.

TEST START

HELLO bob (6)

WORLD venus (3.818798374856044)

HELLO teemu (2)

WORLD mars (3.3202641696335067)

HELLO teemu (2)

TEST END

TEST START

HELLO bob (6)

WORLD venus (0.3211659051330242)

HELLO bob (6)

WORLD venus (1.0997927720325893)

HELLO teemu (1)

TEST END

generated 2 tests.

Figure 1. Example output.

So far the models have been used as a basis by the OSMO Tester for automatically generating test cases based on the defined test algorithm configurations. This is commonly what model-based testing (MBT) is defined as. However, sometimes it is also nice to be able to take the model and manually click through different paths to see what exactly is happening. In OSMO Tester this is called a “manual drive” of the model.

How do we get the manual drive to use? Simply replace the test generation algorithm with the ManualDrive algorithms. This is shown in Listing 3. Notice that we do not set end conditions as the algorithm will provide its own, and setting one by ourselves will just mess it up.

public class Main {

public static void main(String[] args) {

OSMOTester tester = new OSMOTester(new HelloModel());

**tester.setAlgorithm(new ManualAlgorithm());**

tester.generate(52);

}

}

Listing 3. Running the model program.

Now, when you run this you will see the GUI pictured in Figure 2. Upper left corner shows the log of the test steps and data values you have chosen. In the bottom left corner you see the list of available test steps at this time. The only thing on this list is “hello” since “world” is only allowed after “hello”. Thus the GUI will always reflect what is legal for generation according to your model. Practically, it executes your model program one step at a time as you choose. Top right corner shows the last values for the variables the generator is tracking.

So click on “hello” in the lower left corner. What you will see is the GUI shown in Figure 3. This is asking you to specify a value for the “names” variable in the model program as the “hello” test step starts by asking a value to be generated for the “names” variable. With manual drive the user becomes the generator and all the OSMO modeling objects will ask the user for the input. Note that this only works with the ValueSet, ValueRange, ValueRangeSet, and Text objects included with OSMO Tester. This should not be a major constraint since most data can be modeled in this way assuming some modeling skills. The ones that are not modeled like this will just use the generated values from the model.

Now the GUI for ValueSet shown in Figure 3 contains the options defined for the “names” variable. That would be “teemu” and “bob”. In this case, we choose “teemu” and press “OK”. The other choices are “Skip” and “Auto”. Using these causes the algorithm to generate automatically values in the same way that would be done if not manual drive is used. Skip causes one value to be generated for the variable and next time the popup will be shown again. Auto causes this and all future instances of the variable to be automatically generated and the popup will not be shown the next time(s).

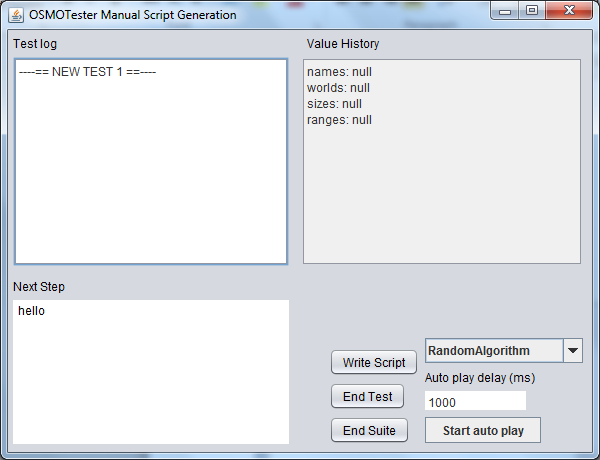


Figure 2. Manual drive GUI.

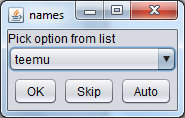


Figure 3. ValueSet GUI.

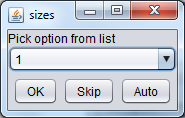


Figure 4. ValueSet GUI for “sizes”.

Now that we chose the value “teemu” and pressed OK, we should see a new popup requesting the next value. Since the “hello” step also generates a value for this, it is also requested before next step is chosen. As it is also a ValueSet, a similar GUI is shown to choose the value. This time we click “Skip”.

Now the manual drive GUI looks like Figure 5.

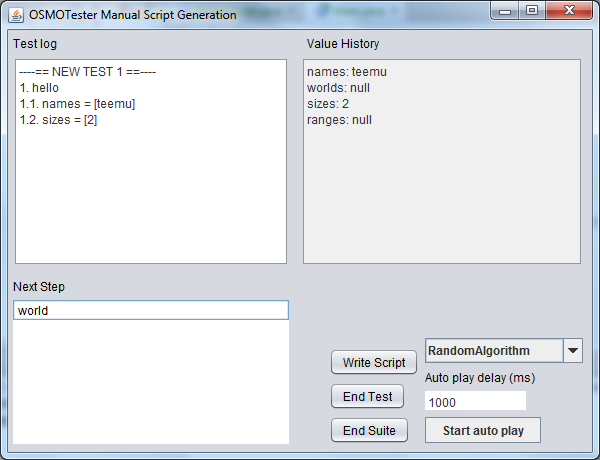


Figure 5. GUI after the first steps.

The log now shows that we have started the first test and chosen the first step, which is “hello”. For this step we have given the variable “names” value “teemu” and the second variable “sizes” got the value “2” from the automated algorithm choice. The metrics show that we have overall one step in our test cases and that is “hello”. The list of possible steps is now updated to show only “world” since that is the only enabled step once “hello” is taken once. Now we click on “world” to execute that step.

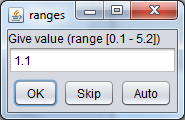


Figure 6. ValueRange GUI.

The “world” step requires values for variables “worlds” and “ranges”. The world GUI is similar to the name selection as both are ValueSet objects. The ValueRange GUI is shown in Figure 6. After these choices, the overall GUI should now look like Figure 7.

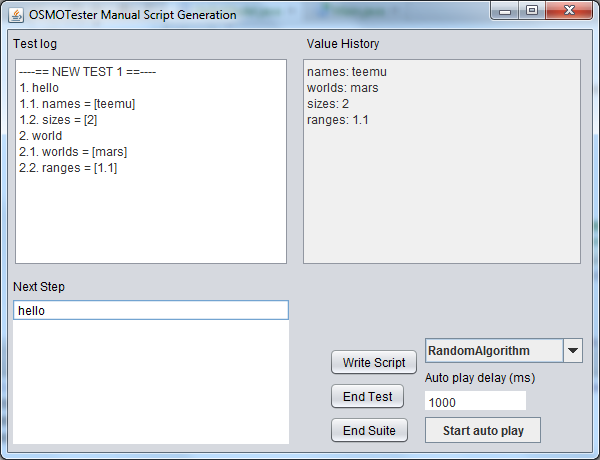


Figure 7. Yet another GUI screenshot.

Now we look at the controls on the bottom right corner. From these, “end test” and “end suite” do what the title says, mainly noting that they cause associated model elements to be executed (AfterTest, AfterSuite, BeforeTest, BeforeSuite as needed and defined). It is also possible to “start auto play”, which will choose steps according to the algorithm chosen in the algorithm box. The delay can be modified while running the auto play to change how fast the steps are taken. You can then observe in the GUI as the generation progresses.

Typically the script is generated (offline) or test is executed (online) while the model program is executed. Additionally, a script in OSMO Tester format can also be written to disk using “write script”. The result is written to a text file called “osmo-output/manual-tests.html” in your working directory. This is written using the TraceReportWriter component, producing a HTML trace of generated test cases similar to other generation options. Note that this same writer can also be used to track automatically generated test cases if desired.

# Conclusions

This tutorial showed the how to use the manual modeling mechanisms to manually explore the test model with OSMO Tester. A related part that you might find interesting is writing scenarios to create manually guide test execution. Check the manual for instructions on this.

# References

OSMOTester home page: <http://code.google.com/p/osmo/>