

Test Report

Contents

Summary of the testing	2
What was tested.....	2
Why you picked particular techniques	2
2.1 Black-box	2
2.1.1 EP, BVA, and combinations.....	2
2.2 White-box.....	2
Any interesting problems you encountered	3
Any interesting results	3
Total number of tests run, passed, and failed	3
Code Coverage with eclemma.....	4

Summary of the testing

This report contains the summary of the testing conducted on a system that contains the files: Lights.java, Part.java, ManualControl .java and TrafficLight.java. The system is used to simulate the traffic lights, i.e. how certain lights change their color, given color change on the other lights. Also, there is a function: *estimatedLifetime* that is used to measure the lifetime of a bulb, given the input parameters.

What was tested

Testing was conducted on a system that contains the files: Lights.java, Part.java, ManualControl.java and TrafficLight.java. BBT, WBT, OOT and System Testing were used to test the system. BBT and WBT were used to test the performance of the function: *estimatedLifetime*, while System Testing using the Abbot Libraries was used to test the performance of the ManualControl.java that contains the graphical simulator for the application. OOT was used to test the classes: Part, Lights and TrafficLight.

Why you picked particular techniques

2.1 Black-box

2.1.1 EP, BVA, and combinations

Black box testing was done based on the specifications. Taking into account the specifications, different detectable partitions were created for each input parameter like brightness, leds and offCurrentFactor. More detailed information is contained in the file LightTest.java.

In the boundary value analysis it was a little tricky for variable offCurrentFactor since it is a double value and is very difficult(impossible) to find which is the next smaller or greater number than that particular value. In this case a precision (approximation) of 0.001 was considered.

When organizing the truth table, I tried not to decrease the number of different causes since all the possible combinations there would give more or less an estimation of the tests needed to cover the whole black box testing. The total number is 30 since some additional tests were added from the boundary value analysis which in some cases can cover the same combination 2 or more times.

2.2 White-box

White-box testing techniques such as branch and statement coverage were based on the CFG. Because of the fact that it is not possible to get 100 % code coverage, also some nodes (in this case 3 out of 27) cannot be covered because there is no combination of values which can force the program to execute the statements. In the case of branch coverage as well is the same problem, since some statements cannot be executed, as well branches connecting the statements are not going to be executed (5 out of 38).

Because all possible branches, statements and combinations were already tested, it is thought that there is no need of adding anymore tests, because condition coverage, decision coverage, multiple coverage, d-u pair testing, and path coverage will not cover anything that was previously already covered.

Any interesting problems you encountered

To be mentioned is the fact that it cannot be achieved 100 percent coverage since the condition if (loop==2) is never achieved because value of done will always be true, since for both leds is equal to false or leds = true, value of done will be always true. Knowing this and using the tool EcIEmma expected 82 percent code coverage can be expected in this case.

In the case of the method estimatedLifetime code coverage cannot be 100 % since the if condition if(loop == 2) is never achieved because variable leds as a boolean variable can take only value true and false and in both cases, value of variable done will be equal to true, before ever reaching the condition if loop == 2. This result is also shown by the tool EcIEmma which shows a 94.7 % percentage of code coverage.

Any interesting results

The majority of the tests conducted passed, even when extreme values were given.

Total number of tests run, passed, and failed

	Number of tests	Passed	Failed
BBT and WBT Unit Testing of Lights.estimatedLifetime	34	33	1
OOT of Part	3	3	0
OOT of Lights	3	3	0
OOT of TrafficLight	5	5	0
System Testing	7	7	0

Code Coverage with eclemma

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
LightsTest.java	62.0 %	707	434	1,141
TrafficLight.java	0.0 %	0	318	318
TrafficLightOOTest.java	0.0 %	0	80	80
LightsOOTest.java	0.0 %	0	46	46
Lights.java	80.6 %	154	37	191
PartOOTest.java	0.0 %	0	32	32
unitTestingRunner.java	94.7 %	288	16	304
unitTestingRunner	94.7 %	288	16	304
Part.java	60.0 %	9	6	15

Figure 1 - BBT & WBT unit testing

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
LightsTest.java	0.0 %	0	1,141	1,141
TestWithAbbotSystemTesting.java	44.2 %	516	652	1,168
unitTestingRunner.java	0.0 %	0	304	304
TrafficLight.java	55.7 %	177	141	318
Lights.java	27.7 %	53	138	191
TrafficLightOOTest.java	0.0 %	0	80	80
LightsOOTest.java	0.0 %	0	46	46
PartOOTest.java	0.0 %	0	32	32
ManualControl.java	97.5 %	549	14	563
Part.java	60.0 %	9	6	15

Figure 2 - System Testing

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
LightsTest.java	0.0 %	0	1,141	1,141
ManualControl.java	0.0 %	0	563	563
TrafficLight.java	0.0 %	0	318	318
unitTestingRunner.java	0.0 %	0	304	304
Lights.java	0.0 %	0	191	191
TrafficLightOOTest.java	0.0 %	0	80	80
LightsOOTest.java	0.0 %	0	46	46
Part.java	100.0 %	15	0	15
PartOOTest.java	100.0 %	32	0	32

Figure 3 - PartOOTest










LightsOOTest (15-Dec-2012 00:15:17)					
Element		Coverage	Covered Instructio...	Missed Instructions	Total Instructions
▶ LightsTest.java		0.0 %	0	1,141	1,141
▶ ManualControl.java		0.0 %	0	563	563
▶ TrafficLight.java		0.0 %	0	318	318
▶ unitTestingRunner.java		0.0 %	0	304	304
▶ Lights.java		27.7 %	53	138	191
▶ TrafficLightOOTest.java		0.0 %	0	80	80
▶ PartOOTest.java		0.0 %	0	32	32
▶ LightsOOTest.java		100.0 %	46	0	46
▶ Part.java		100.0 %	15	0	15

Figure 4 – LightsOOTest










TrafficLightOOTest (15-Dec-2012 00:18:31)					
Element		Coverage	Covered Instructio...	Missed Instructions	Total Instructions
▶ LightsTest.java		0.0 %	0	1,141	1,141
▶ ManualControl.java		0.0 %	0	563	563
▶ unitTestingRunner.java		0.0 %	0	304	304
▶ TrafficLight.java		46.2 %	147	171	318
▶ Lights.java		24.1 %	46	145	191
▶ LightsOOTest.java		0.0 %	0	46	46
▶ PartOOTest.java		0.0 %	0	32	32
▶ Part.java		100.0 %	15	0	15
▶ TrafficLightOOTest.java		100.0 %	80	0	80

Figure 5 - TrafficLightOOTest