ECE 322
Lab Report 3

Arun Woosaree
XXXXXXX

October 20, 2019

1 Introduction

The purpose of this lab was to serve as an introduction to cause-effect graphs and decision tables. The goal was to become familiar with the use of these graphs and rules in black box testing, and to put into practice generating test cases using these tools. We tested two GUI applications written in Java. The first application, InsuranceCalculator takes in 3 inputs, a gender (Male/Female), an age, and number of claims. A cause-effect graph (Appendix A) was derived from 5 rules which were given. From the cause-effect graph, a decision table is derived, and test cases were made using the decision table. Depending on the input, this application outputs either \$750, \$1000, \$1500, or \$3000. The second appplication, BoilerShutdown is also a Java GUI application. It takes in 10 inputs labelled from A-J, which are checkboxes. Depending on the inputs, it will output either OK or FAIL. In this case, a cause-effect graph was provided, however, it was unsimplified. By tracing back, a simplified cause-effect graph was created (Appendix B), from which a decision table was derived. Finally, from the decision table, test cases were created. The purpose of cause-effect graphs is to divide the software specification into workable pieces. Once a decision table is generated, the test cases are derived using boundary value analysis, and also looking at the columns of the decision table.

2 Task 1

For task one in this lab, the InsuranceCalculator application was tested using a cause-effect graph from which a decision table and test cases were made. InsuranceCalculator is a GUI program written in java, which takes in three inputs:

- gender: Male/Female
- age: integer

- claims: integer

Depending on the value of the inputs, the program will output either \$750, \$1000, \$1500, \$3000, or ERROR.

From the following rules, a cause-effect graph was generated, and can be found in Appendix A.

- IF sex = Male AND age < 25 AND claims = 0 THEN premium = \$1500
- IF sex = Male AND $25 \leq$ age < 65 AND claims = 0 THEN premium = \$1000
- IF sex = Female AND age < 65 AND claims = 0 THEN premium = \$750
- IF age ≥ 65 AND claims = 0 THEN premium = \$1500
- IF claims ≥ 1 THEN premium = \$3000

From the cause-effect graph found in Appendix A, the following decision table was made:

	1	2	3	4	5
Causes					
A	0	1	1	x	x
B	1	0	0	x	x
C	x	0	1	0	x
D	x	1	0	0	x
E	0	0	0	1	x
F	1	1	1	1	0
G	0	0	0	0	1
Effects					
Y1	1	x	x	x	x
Y2	x	1	x	x	x
Y3	x	x	1	1	x
Y4	x	x	x	x	1

From the decision table above, the following test cases were made:

Test ID	Sex	Age	Claims	Expected	Actual
1	F	64	0	\$750	\$750
2	M	25	0	\$1000	\$1000
3	M	24	0	\$1500	\$1500
4	F	66	0	\$1500	\$1500
5	F	54	1	\$3000	\$3000

Normally, failed test cases would be highlighted in red. However, none of the test cases failed in this case.

2.1 Discussion

While no failures in the software were caught, this is not neccessarily an indication of poor tests. It does appear that there is a good coverage, since there is one test case which tests the expected output for each rule defined in the software specification, and it does so with relatively few test cases. (Only 5 in this case). However, using other black-box testing methods learned earlier like dirty testing, EPC, or weak n x 1 strategies, some more scenarios are covered, which were not covered using the cause-effect and decision table tools. For example, none of the test cases tested negative valued inputs for the age or total claims fields, which allow negative inputs. In this case, the application behaves as expected, outputting an ERROR message, however, this scenario was not covered when using the techniques learned in this lab. Relative to the other black-box testing techniques learned, there are fewer, more efficient test cases from cause-effect graphs and decision tables. However, from a subjective standpoint, using this method demanded more effort for creating test cases versus other techniques previously learned. The black box testing techniques previously learned also seem to cover more scenarios that were missed using this technique.

3 Task 2

For task two in this lab, the BoilerShutdown application was tested using a simplified cause-effect graph from the one which was given, from which a decision table was made, and test cases were created. BoilerShutdown is a GUI application written in Java, which has 10 checkbox inputs labelled alphabetically from A to J. These are defined as:

- A - Water level meter failure during operation
- B - Steam rate meter failure during operation
- C - Water level out of range
- D - Instrumentation system failure
- E - Communication link failure
- F - Steam rate meter failure during startup
- G - Water level meter failure during startup
- H - Multiple pumps failure (more than one)
- I - Confirmed keystroke entry
- J - Confirmed "shut now" message

The given cause-effect graph for Task 2 was simplified, and can be found in Appendix B. By tracing back the graph, we were able to determine that inputs A, B, and C had no effect on whether the Boiler would shut down or not. Furthermore, We were able to eliminate almost all intermediate nodes, since it was found that the boiler effectively shuts down if any of the inputs D, E, F, G, H, I, or J = 1 and it does not shut down if all of the inputs from D-J = 0. From the new cause-effect graph, the following rules were derived:

- IF D OR E OR F OR G OR H OR I OR J THEN Boiler Shutdown
- IF NOT D AND NOT E AND NOT F AND Not G AND NOT H AND NOT I AND NOT J THEN Boiler not Shutdown

From the rules defined above, a decision table was made:

	1	2	3	4	5	6	7	8
Causes								
A	x	x	x	x	x	x	x	x
B	x	x	x	x	x	x	x	x
C	x	x	x	x	x	x	x	x
D	1	x	x	x	x	x	x	0
E	x	1	x	x	x	x	x	0
F	x	x	1	x	x	x	x	0
G	x	x	x	1	x	x	x	0
H	x	x	x	x	1	x	x	0
I	x	x	x	x	x	1	x	0
J	x	x	x	x	x	x	1	0
Effects								
Boiler shutdown	1	1	1	1	1	1	1	0
Boiler not shutdown	0	0	0	0	0	0	0	1

From the decision table above, the following test cases were made:

Testid	A	B	C	D	E	F	G	H	I	J	Expected	Actual
1	0	0	0	1	0	0	0	0	0	0	FAIL	FAIL
2	0	0	1	0	1	0	0	0	0	0	FAIL	FAIL
3	0	1	0	0	0	1	0	0	0	0	FAIL	FAIL
4	0	1	1	0	0	0	1	0	0	0	FAIL	FAIL
5	1	0	0	0	0	0	0	1	0	0	FAIL	FAIL
6	1	0	1	0	0	0	0	0	1	0	FAIL	FAIL
7	1	1	0	0	0	0	0	0	0	1	FAIL	FAIL
8	1	1	1	0	0	0	0	0	0	0	Ok	Ok

Normally, failed test cases would be highlighted in red. However, all test cases passed in this scenario.

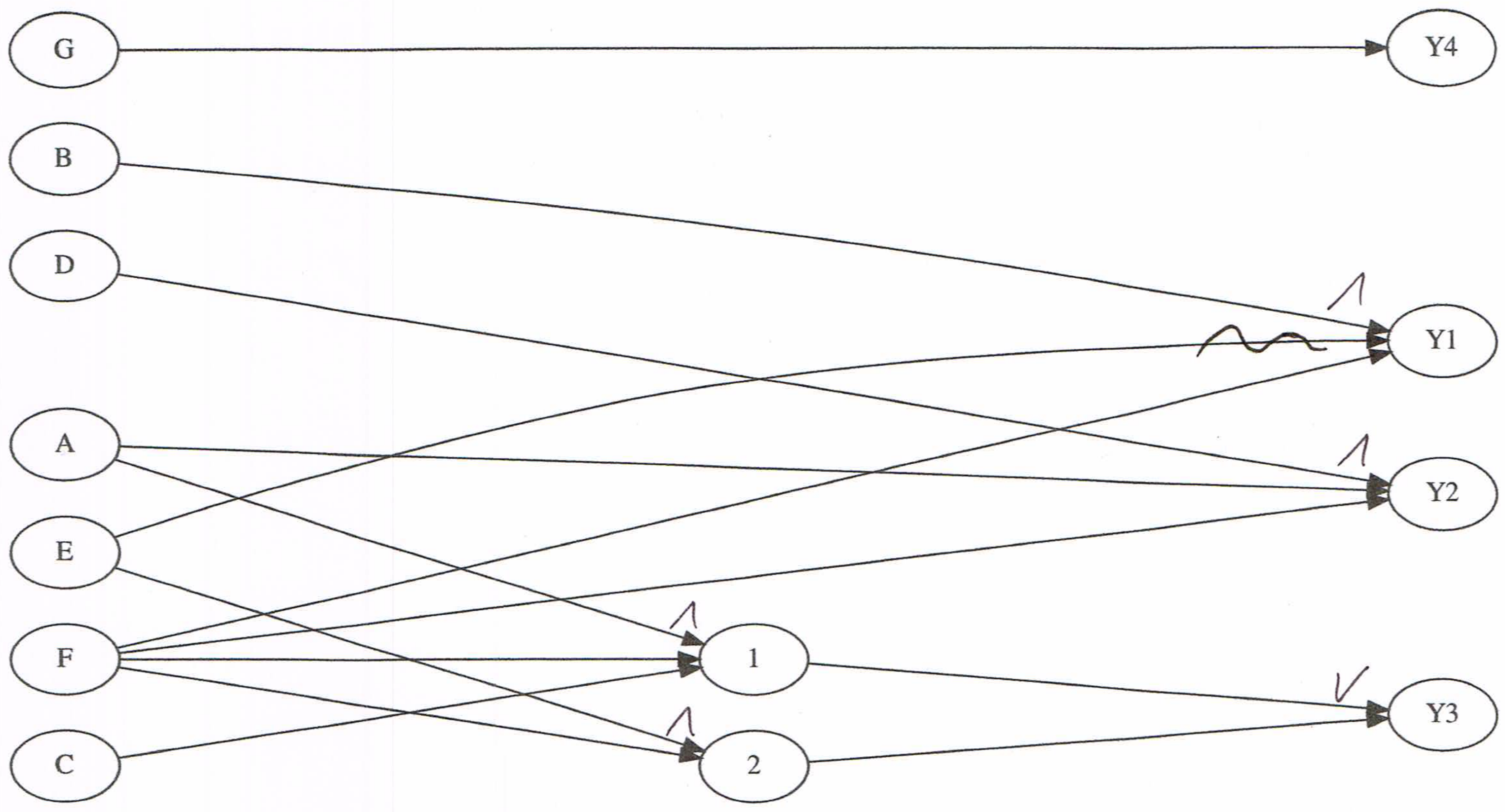
3.1 Discussion

While no failures in the software were caught, this is not neccessarily an indication of poor tests. It appears that there is good coverage, since there is one test case for each input that should cause the boiler to shut down,and an additional test case where the boiler should not shut down. In terms of number of test cases, this is definitely more efficient compared to previous methods learned like EPC, weak n x 1, and dirty testing. Only 8 test cases were made to test the functionality, while other methods would have resulted in far more test cases. Deriving test cases from the decision table for this piece of software seemed more appropriate than for the InsuranceCalculator program, since the program constrains the user to only enter valid inputs. This avoids scenarios like in Task 1 where the the test cases had a bunch of valid inputs, but missed the cases where an input was invalid. Compared to other black-box testing methods, the cause-effect diagram and decision tables seems most effective for testing this program, since there are not really any boundaries to check, and the inputs are always valid. In this case, we're only concerned about the inputs (causes) and their direct effect, and there are not really any edge cases to consider. Combinatorial testing, and testing all possible combinations are two other testing techniques that my be considered for testing this program.

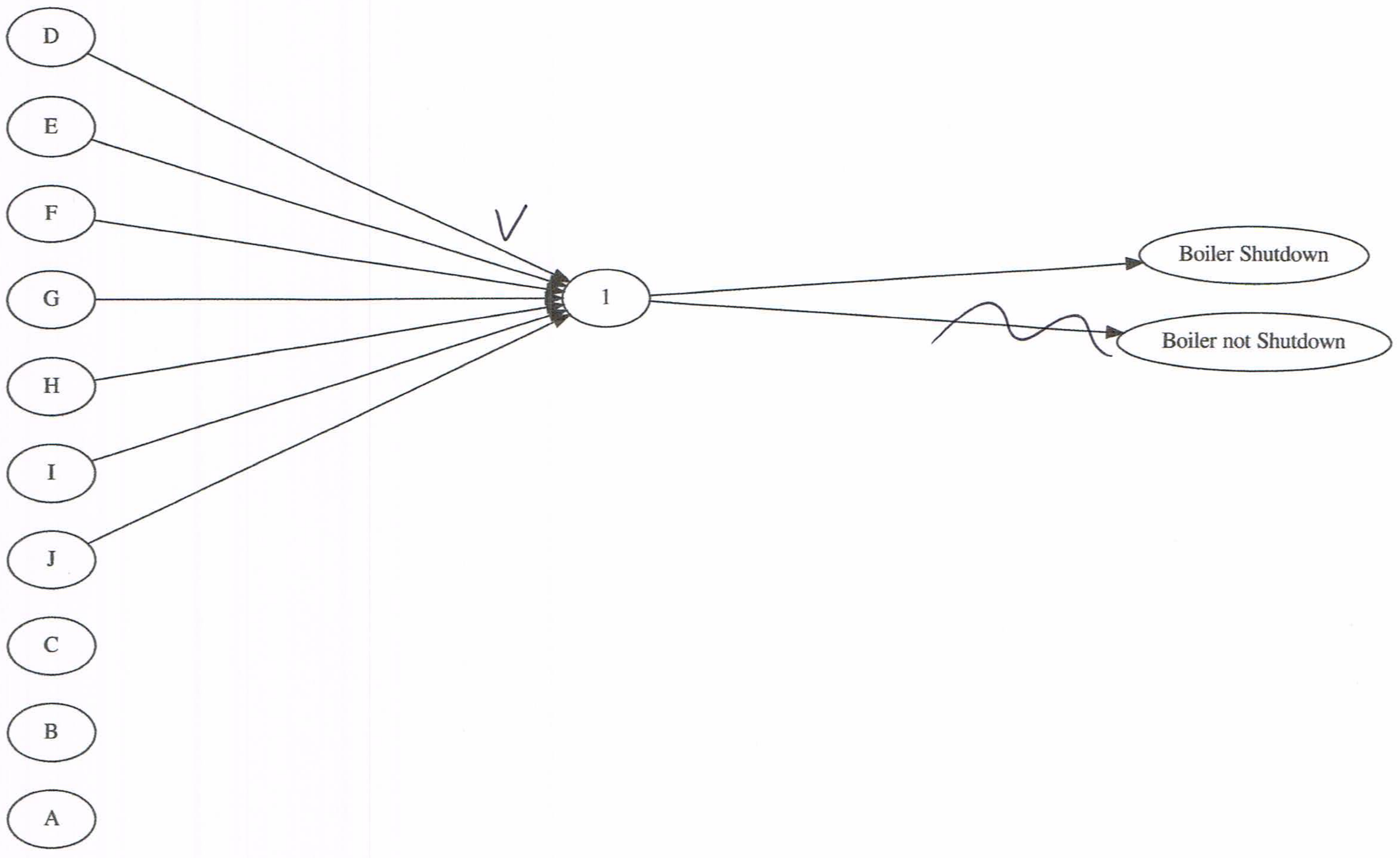
4 Conclusion

In this lab, we were introduced to the black box testing strategy of creating cause-effect graphs and decision tables to help come up with test cases. While it would not be fair to say that one testing strategy is universally better than the other, since it would be better to match each problem at hand to the appropriate testing strategy, compared to other testing methods previously learned like dirty testing, error guessing, EPC, and weak n x 1 strategy, this method seems to demand the most amount of thought and effort for creating the test cases. From a subjective standpoint, making cause effect graphs and deriving test cases from them is an okay testing technique, but not one of my favourites. While in terms of number of test cases, it is efficient in that it generates very few test cases while still having generally good coverage, the effort required to generate the test cases does not seem like it is worth it for most applications. Additionally, it seems to not cover some scenarios, like in Task 1 with the CarInsurance program, where invalid inputs were not tested. There are some scenarios, however, where the testing technique seems appropriate to use. For example, in Task 2 with the BoilerShutdown application, it was impossible to enter an invalid input, and we were strictly concerned with the inputs (causes), and their effects. By creating a simplified cause effect graph, we created few, but effective test cases. Overall, making cause-effect graphs and decision tables definitely has its place in black box testing. Like with any other testing technique, there are scenarios where it shines, and other cases where another testing technique would work better.

Appendix A: Cause Effect Graph for Car Insurance Premium



Appendix B: Simplified Cause Effect Graph for Boiler Shutdown


D
E
F
G
H
I
J
C
B
A
V
1
Boiler Shutdown
Boiler not Shutdown