Experiment Results

Ali Can Zeybek, Furkan Samet Akıncı

I. Introduction

In this report we implemented and tested two given programs. Decision table testing, equivalence testing and boundary testing done for each and compared both programs with each other in respect to 3 test techniques.

II. Experiment 1

In this program there are 3 input variables as ‘number_of_items‘ , ‘total_cost‘ and ‘delivery_time‘. First two variables are numberic while the third parameter is an enum type.

A. Decision Table Testing

Decision tree testing for program 1 generated 12 different test cases, it is expected since for inputs ‘number_of_items‘ and ‘total_cost‘ generate two possible paths of execution,

$$\text{Branching due to } (\#items) = \begin{cases} \text{True}, & \text{if } number\_of\_items \leq 30, \\ \text{False}, & \text{if } number\_of\_items > 30. \end{cases}$$

$$\text{Branching due to } (total\_cost) = \begin{cases} \text{True}, & \text{if } number\_of\_items \leq 100, \\ \text{False}, & \text{if } number\_of\_items > 100. \end{cases}$$

while ‘delivery_time‘ can generate 3 values as ‘NEXT_DAY‘, ‘SECOND_DAY‘ and ‘THIS_WEEK‘. Hence, 2 x 2 x 3 = 12.

number_of_items	total_cost	delivery_time	Expected Outcome
≤ 3	≤ 100	NEXT_DAY	25
≤ 3	≤ 100	SECOND_DAY	10
≤ 3	≤ 100	THIS_WEEK	number_of_items * 1.50
≤ 3	> 100	NEXT_DAY	35.00
≤ 3	> 100	SECOND_DAY	15.00
≤ 3	> 100	THIS_WEEK	10.00
> 3	≤ 100	NEXT_DAY	number_of_items * 6.00
> 3	≤ 100	SECOND_DAY	number_of_items * 2.50
> 3	≤ 100	THIS_WEEK	0.00
> 3	> 100	NEXT_DAY	number_of_items * 7.50
> 3	> 100	SECOND_DAY	number_of_items * 3.50
> 3	> 100	THIS_WEEK	number_of_items * 2.50

TABLE I: Test cases generated for decision table testing.

The total test case generation and implementation took 55 minutes. While running 12 test cases took 0.008 seconds [1], running the full ‘mvn test‘ command took 4.59 seconds [2] [3]

B. Equivalence Testing

For equivalance testing valid and invalid partitions of input variable domains are selected.

1) Equivalence Partitions: Following are the valid and invalid partitions of program 1.

[1]Test execution times are reported by mvn

[2]‘time mvn test‘ command used for timekeeping

[3]Tested setup has 32G ram Ryzen 7 5800X running Debian 12 and openjdk 17.0.13

Input	Partition Type	Partition Condition	Selected Value
number_of_items	Valid	≤ 3	3
	Valid	> 3	4
	Invalid	≤ 0	-1
total_cost	Valid	≤ 100	100
	Valid	> 100	101
	Invalid	< 0	-2
delivery_time	Valid	NEXT_DAY, SECOND_DAY, THIS_WEEK	SECOND_DAY
	Invalid	Null or unsupported value	null

TABLE II: Partitioning for input values with conditions and selected values.

Total Cost	Number of Items	Shipping Time	Expected Outcome
100.0	3	NEXT_DAY	25.0
150.0	3	SECOND_DAY	15.0
100.0	4	THIS_WEEK	0.0
150.0	4	NEXT_DAY	30.0
100.0	-1	NEXT_DAY	Exception
-50.0	3	SECOND_DAY	Exception
100.0	3	null	Exception

TABLE III: Test cases generated for equivalence testing

The total test case generation and implementation took 40 minutes. While running 7 test cases took 0.028 seconds , running the full ‘mvn test‘ command took 6.49 seconds.

C. Boundary Value Testing

For boundary value testing ‘delivery_time‘ does not eligable since enum’s do not have boundaries. For other two variables, values just before, exactly and just above valid input range and logical branching boundaries are selected. Those values are -1,0,1 for both ‘number_of_items‘ and ‘total_cost‘ while branching boundary values are 3,4 and 100,101 respectively as can seen in Table IV

Input	Boundaries
number_of_items	-1
	0
	1
	3
	4
total_cost	-1
	0
	1
	100
	101
delivery_time	n/a

TABLE IV: Boundary values for input parameters.

1) Test Cases: Using above boundaries following test scenarios generated.

The total test case generation and implementation took 24 minutes. While running 9 test cases took 0.026 seconds , running the full ‘mvn test‘ command took 7.01 seconds.

number_of_items	total_cost	delivery_time	Partition Type	Expected Outcome
3	100	NEXT_DAY	All Valid	25.00
3	101	SECOND_DAY	All Valid	15.00
4	100	THIS_WEEK	All Valid	0.00
4	101	NEXT_DAY	All Valid	30.00
-1	100	NEXT_DAY	Invalid Items	Exception
3	-50	SECOND_DAY	Invalid Cost	Exception
3	100	null	Invalid Delivery	Exception

TABLE V: Test cases generated for boundary value testing

III. Experiment 2

A. Decision Table Testing

The decision table enumerates various combinations of conditions (employee status, special authorization, auditor status, hour of day, and weekend) and shows the expected outcome. The conditions chosen are representative samples from the decision space.

#	isEmployee	hasSpecialAuth	isAuditor	hourOfDay	isWeekend	Expected Result
1	False	False	False	9	False	Deny (false)
2	True	False	False	9	False	Allow (true)
3	True	True	False	8	False	Allow (true)
4	True	False	False	17	False	Deny (false)
5	True	False	False	9	True	Deny (false)
6	False	False	True	9	True	Allow (true)
7	False	False	True	8	False	Deny (false)
8	True	True	False	17	True	Allow (true)

TABLE VI: Test cases generated for for decision table testing

The total test case generation and implementation took around 1 hour and 30 minutes. While running above cases took 5709 nanoseconds.

B. Equivalence Testing

Here, input domains are divided into equivalence classes. We pick one representative test from each class to reduce the total number of tests while still ensuring coverage of all logical categories.

#	Conditions (Representative)	Expected Result
1	Not Employee, Not Auditor, Hour=10 (within 9–16), Weekday=false (not weekend)	Deny (false)
2	Employee, No Special Auth, Not Auditor, Hour=10 (within 9–16), Weekday=false	Allow (true)
3	Employee, Special Auth, Any Hour (e.g., 17), Weekend=true	Allow (true)
4	Auditor, Hour=16 (within 9–16), Weekend=true	Allow (true)
5	Auditor, Hour=8 (before 9), Weekday=false	Deny (false)

TABLE VII: Test cases generated for for equivalence testing

The total test case generation and implementation took around 45 minutes. While running above cases took 3917 nanoseconds.

Equivalence Classes Considered:

1) Employee vs. Non-Employee
2) Auditor vs. Non-Auditor
3) Special Authorization vs. No Special Authorization
4) Inside vs. Outside Working Hours (9–16)
5) Weekend vs. Weekday

C. Boundary Value Testing

Boundary values are chosen around the critical time limits. For this scenario, critical hours are 8 (just before 9), 9 (start of working hours), 16 (end of working hours), and 17 (just after 16).

#	Conditions (Focus on Hour Boundaries)	Expected Result
1	Employee, No Special Auth, Hour=8 (just before start), Weekday=false	Deny (false)
2	Employee, No Special Auth, Hour=9 (start), Weekday=false	Allow (true)
3	Employee, No Special Auth, Hour=16 (end), Weekday=false	Allow (true)
4	Employee, No Special Auth, Hour=17 (just after end), Weekday=false	Deny (false)
5	Auditor, Hour=8 (before start), Weekend or Weekday (e.g., false)	Deny (false)
6	Auditor, Hour=9 (start), Weekend=true	Allow (true)
7	Employee, Special Auth, Hour=8 (any boundary), Weekend=true	Allow (true)

TABLE VIII: Test cases generated for for boundary value testing

The total test case generation and implementation took around 1 hour and 15 minutes. While running above cases took 5876 nanoseconds.

Boundary Values Considered:

1) Hours: 8, 9, 16, 17
2) Weekend vs. Weekday, Employee vs. Auditor, Special Auth variations

IV. Conclusion

In Project 2 (Access Control scenario), the decision conditions were related to employee roles, special authorization, and time constraints. In contrast, Project 1 focuses on numerical thresholds (Purchase Amount, Number of Items) and cost calculations. While Project 2 tested boolean logic and access permissions, Project 1 tests arithmetic computations and fee structures.

Test Complexity:

- Project 2: More logical/boolean conditions.
- Project 1: More arithmetic and boundary-value checks.

Number of Test Cases:

- Project 2: Focused on covering all logic paths (e.g., 8 decision table tests).
- Project 1: Has a straightforward table with distinct numeric thresholds, potentially leading to more test cases because of multiple numeric partitions and delivery speed options (12 scenarios as outlined above).

Coverage Strategy:

- Project 2: Ensured every logical combination was tested.
- Project 1: Ensures each numerical range and delivery option is covered, testing all formula variations.

ADDITIONAL RESOURCES

All of this code and reports and report codes can be fount at projects git page.