Estimation. Practice creating test documentation

Assignment

Stefania Pruteanu

- Create a **high-level** test case to test the functionality of the mobile application of choice.
- Create 3 **low-level** test cases based on the high-level one.

I will pick the app **Airport Distance Map** : with the requirement
"Users should be able to see the total distance between two airports, placing the markers according to departure and destination."

1. HIGH LEVEL

Checklist-based technique :
Check if the distance between any 2 airports is displayed correctly, in miles.

2. LOW LEVEL

1#
Type Requirement based - **Dynamic - Technique** Black Box **- Level** System - **Scope** Functionality - **Subtechnique Pairwise**

```
FROM    M1      TO      M2      ======= AIRPORT =========       FLIGHT
        0               1.660           GEG OR MEM              OK
        1.660           0               GEG OR MEM              OK
        1.660           1.660           SAME AIRPORT            _
        0               0               SAME AIRPORT            _
```

M1, M2 - Departure/Destination points**: 0 or 1.660,** for Georgia **(GEG) or** Memphis **(MEM)** **(example)**

For EP:
We have all the States airports at a fixed distance, so individual partitions with the same results.

For BVA: is redundant, we will have the same result as "not possible" if we pick values higher or lower than the airport's marker (we can't fly to an airport that's not an airport).

For Decision table: is also redundant, since we don't have any other option than to "fly" to destination.

For Pairwise: because we have two combinations of departure and destination, we can have the same airport as a destination or as a departure: We either fly from MEM to GEG or from GEG to MEM.

This also includes the negative cases where to FLIGHT is not possible marked as " **_** "
IF the destination = departure (we can't fly to the same airport)

Test cases are here Airport distance map - Testsuite.pdf or in TestRail under my test suite.

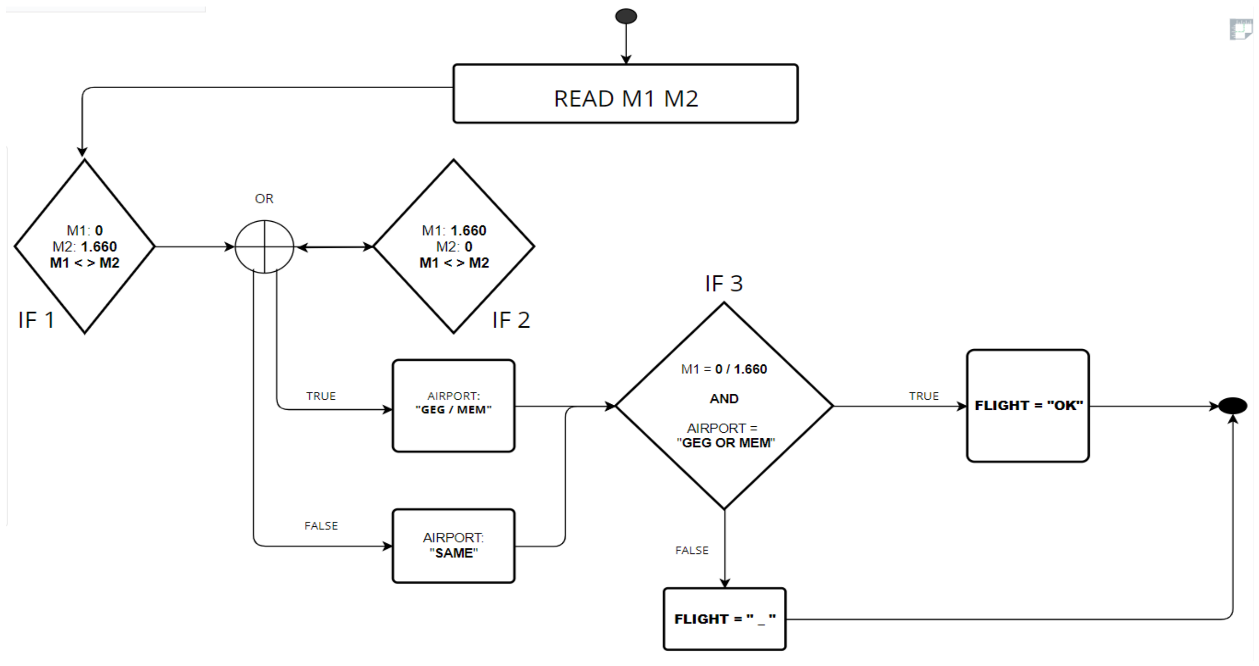
2#
Type Requirement based - **Dynamic - Technique** Whitebox **- Level** System - **Scope** Functionality - **Subechnique Path coverage**

Since I generated that pairwise table, with a code**,** I might try to make a test on it too:

```
FROM    M1      TO      M2      ======= AIRPORT =========       FLIGHT
        0               1.660           GEG OR MEM              OK
        1.660           0               GEG OR MEM              OK
        1.660           1.660           SAME AIRPORT            _
        0               0               SAME AIRPORT            _
```

```
IF
([M1] = 0 AND [M2] = 1.660) OR ([M1] = 1.660 AND [M2] = 0) AND ([M1] <> [M2])
THEN [AIRPORT] = "GEG OR MEM"
ELSE [AIRPORT] = "SAME AIRPORT";

IF ([M1] = 0 OR [M1] = 1.660) AND [AIRPORT] = "GEG OR MEM"
THEN [FLIGHT] = "OK"
ELSE [FLIGHT] = "_";
```



Airport distance map - flowchart - whitebox.pdf

Not an accurate code.., but here here are:
5 statements

- **READ M1 M2**
- AIRPORT = **SAME , GEG/MEM**,
- FLIGHT = **OK , _**

3 conditions - 2 are the same by OR, 1 is unique
12 decisions

READ has 1 path, **IF1 has 3** paths**, OR has 1** path, **IF2 has 2** paths, **IF3 has 2** paths, **FLIGHT has 2** paths
= 11 paths **(without entry point)**

TC1:
READ **M1: 0 & M2: 1.660** **— FLIGHT OK — 5/11** (from airport1 to airport2)

TC2:
READ **M1 : 1.660 & M2 : 0** **— FLIGHT OK — 7 /11** (from airport2 to airport1)

TC3:
READ **M1: 0** AND **M2 : 0** **|** **M1: 1.660** AND **M2: 1.660** **— FLIGHT " _ " — 7 /11** (from/to same airport)