Software Testing: Tutorial 3

Category-Partition Testing

- **Prerequisites:** before the tutorial you should review section 11.2 of the Pezzè & Young.
- **Preparation:** read the following functional specification (exercise 11.4 of Pezzè & Young).

Your task is to: *Derive test specifications using the category partition method for the following Airport connection check function.*

> **Airport connection check:** The airport connection check is part of an (imaginary) travel reservation system. It is intended to check the validity of a single connection between two flights in an itinerary. It is described here at a fairly abstract level, as it might be described in a preliminary design before concrete interfaces have been worked out.
>
> **Specification Signature:** Valid Connection (Arriving Flight: flight, Departing Flight: flight) returns Validity Code
>
> Validity Code 0 (OK) is returned if Arriving Flight and Departing Flight make a valid connection (the arriving airport of the first is the departing airport of the second) and there is sufficient time between arrival and departure according to the information in the airport database described below.
>
> Otherwise, a validity code other than 0 is returned, indicating why the connection is not valid.

Data types

Flight: A "flight" is a structure consisting of

- A unique identifying flight code, three alphabetic characters followed by up to four digits. (The flight code is not used by the Valid Connection function.)
- The originating airport code (3 characters, alphabetic)
- The scheduled departure time of the flight (in universal time)
- The destination airport code (3 characters, alphabetic)
- The scheduled arrival time at the destination airport.

Validity Code: The validity code is one of a set of integer values with the following interpretations:

- **0:** The connection is valid.
- **10:** Invalid airport code (airport code not found in database)
- **15:** Invalid connection, too short: There is insufficient time between arrival of first flight and departure of second flight.
- **16:** Invalid connection, flights do not connect. The destination airport of Arriving Flight is not the same as the originating airport of Departing Flight.
- **20:** Another error has been recognized (e.g., the input arguments may be invalid, or an unanticipated error was encountered).

Airport Database: The Valid Connection function uses an internal, in-memory table of airports which is read from a configuration file at system initialization. Each record in the table contains the following information:

- **Three-letter airport code.** This is the key of the table and can be used for lookups.
- **Airport zone.** In most cases the airport zone is a two-letter country code, e.g., "US" for the United States. However, where passage from one country to another is possible without a passport, the airport zone represents the complete zone in which passport-free travel is allowed. For example, the code "EU" represents the European countries which are treated as if they were a single country for purposes of travel.
- **Domestic connect time.** This is an integer representing the minimum number of minutes that must be allowed for a domestic connection at the airport. A connection is "domestic" if the originating and destination airports of both flights are in the same airport zone.
- **International connect time.** This is an integer representing the minimum number of minutes that must be allowed for an international connection at the airport. The number -1 indicates that international connections are not permitted at the airport. A connection is "international" if any of the originating or destination airports are in different zones.

Activities

Having considered this specification you should carry out the following activities that will be facilitated by your tutor:

1. Individually, have a look again at the specification and write down a list of Independently Testable Functions (ITFs) in this specification. There might be more than one — however, carefully justify each ITF you have identified so you can argue for it.

   [take max 5 mins to do this]

2. Choose a partner you will work with in the tutorial and discuss your decision from activity 1 (if the tutorial has an odd number of students you will need to have one group of three). Agree on a final list of ITFs for this specification.

   [take max 5 mins for this activity]

3. Whole group discussion: read out your lists of ITFs: the tutor will write up distinct ITFs so you have a definitive list of ITFs — be selective, if the group does not find the justification convincing do not include a proposed ITF in the final list. *Hint: there are probably only 2 or 3 ITFs depending on how you look at it.*

   [take max 5 mins for this activity]

4. Select two of the ITFs and split the group to work on them separately. Each group should work in the following way:

5. Individually, each group member should work to identify the parameters and environment elements that are relevant to their ITF. Once you have identified these you should then work out characteristics of your parameters and environment elements.

   [allow around 10 mins for activities 4 and 5]

6. Together with your partner(s) agree your merged list, then agree the list with your group.

   [allow 5 mins for this activity].

7. Individually, identify value classes for each of your characteristics. Check these with your partner and agree a list.

   [allow 10 mins for this activity].

8. Together with your partner(s) work out how many test case specifications you have assuming no constraints between the value classes. Also, pay some attention to the expected result for a test. Then work together to try to identify situations where you can use constraints to reduce the number of possible test case specifications.

   [take a further 10 mins for this activity]

9. Finally, as a group try to merge your efforts to create a set of test case specifications for your chosen ITF.

   [allow 5 mins for this]