Module 3: Assignment (INL5)

This document provides the instructions for the Module 3 assignment INL5 on combinatorial testing in DVA-434. The assignment, when completed, corresponds to 0.5 credits of the 7.5 credits in the course examination. Once you have completed the assignment, please send your report to daniel.sundmark@mdh.se

Objective and Platform

The goal of this exercise is to experiment with combinatorial issues when testing software using any freely available tool (some are free for a trial purpose only but enough to complete this assignment).

Note that there are several tools available for combinatorial testing, and we do not particularly promote any tool.

Some example tools are:

https://inductive.no/pairwiser/

https://github.com/cornutum/tcases

After you have selected a tool, please go through the available tutorials that will help you solve tasks in this exercise.

Overview

The task is to create tests for a component that handles a car specification in a car booking service. Depending on the property values, a different car will be booked and placed in front of your house.

The parameters and corresponding values are as follows:

1. CarType: Truck, Jeep, Mini-van, Mini
2. Color: Blue, Lilac, Yellow
3. Engine: Diesel, Gasoline
4. PassengerSeatAirbag: Yes, No
5. PassengerSeatSideAirbag: Yes, No
6. SideAirbagBack: Left, Right, None
7. Seatbelts: Yes, No
8. PassengerSeatChildSeat: Yes, No

Task 1: Create a test plan

The first task is to create a new test plan and enter above parameters along with their values.

1. Create a paired "2-way interactions" test suite. How many tests did this require?
2. What is the theoretical number of tests to exhaustively cover all combinations?

You tool should generate a set of test cases with input values. In a real example you would have to provide some kind of oracle (expected result of a test) that tells whether the chosen car is present in the database or not.

You do not need to implement this though but take a look at how this is done in the tool for a few steps in your test plan.

Task 2: Removing impossible test cases (adding constraints)
One can argue for removing combinations that are illegal or impossible, to make sure that we do not test in vain. In our case, the traffic regulations say that you cannot put a child seat in the passenger seat if there exists an airbag. Furthermore, a *Truck* (in our context) never has a back seat, which implies that we cannot have any *SideAirbagBack*.

Check the manual on how to add constraints and add those to your test plan.

1. How many tests do you get for 2-way interactions after adding above constraints? Please save your 2-way interactions tests in an Excel file. Also try out the different X-way interactions
2. Is the result what you would have expected for 2-way interactions? Please also comment briefly as to why did you get that result.
3. Compare the list of tests after adding constraints with the first list. Please comment on the changes you find in both the lists.
4. Discuss with a colleague some situations where it is actually necessary to test invalid combinations.

Task 3: Enforcing pairs (adding requirements)
On the other hand, some combinations may be mandatory to test. Since we know that most of our customers want a *yellow truck* with *passenger seat airbag*, we want to make sure that we always include a test for this combination.

1. How many tests did you get now and why?
2. Do you test your requirement(s)?

Task 4: Adding parameters to see combinatorial growth
Now add a parameter and one value and check how many more possible pairwise tests we get.

Add a few more values to the parameter and check again after adding each value

Add a few parameters with different number of values until you have a feeling of what affects the number of 2-way interaction tests most. A new parameter or another possible value?