COMBINATORIAL TEST TOOL
DESIGN DESCRIPTION

DVA313

Group - 2

14th January 2016

Contents

1. Introduction

Most software failures are induced by a single factor faults or combination of factor faults. Combinatorial testing tool helps in detection of failures that can arise due to certain combination of inputs.

Some benefits of having a combinatorial testing tool are:

- Improved quality of the product
- Defects are found at an earlier stage, when it is cheaper to fix them
- Makes testing of the system more efficient

Our project is to develop this combinatorial test tool for Bombardier's Train Control Management System (TCMS). Bombardier in Sweden is focused on development and manufacturing of trains and railway equipment. The Train Control Management system handles most of the train's operations like heating, opening and closing the door, the braking system, collecting line voltage, upload of diagnostic data and controlling the train engines.

There are different levels of testing in Bombardier:

- Vehicle: Vehicle integration Testing
- System: System Integration testing
- Software: Program Integration testing and Program testing

Program testing is the responsibility of developer or engineer and our tool will be used in program testing.

The test tool should be written in C# language, running on Windows and also provide a GUI. User imports an XML file into the GUI and choose either random or base choice technique to generate different input combinations. The generated test cases will be saved by the user into a CSV file which will be used for testing TCMS.

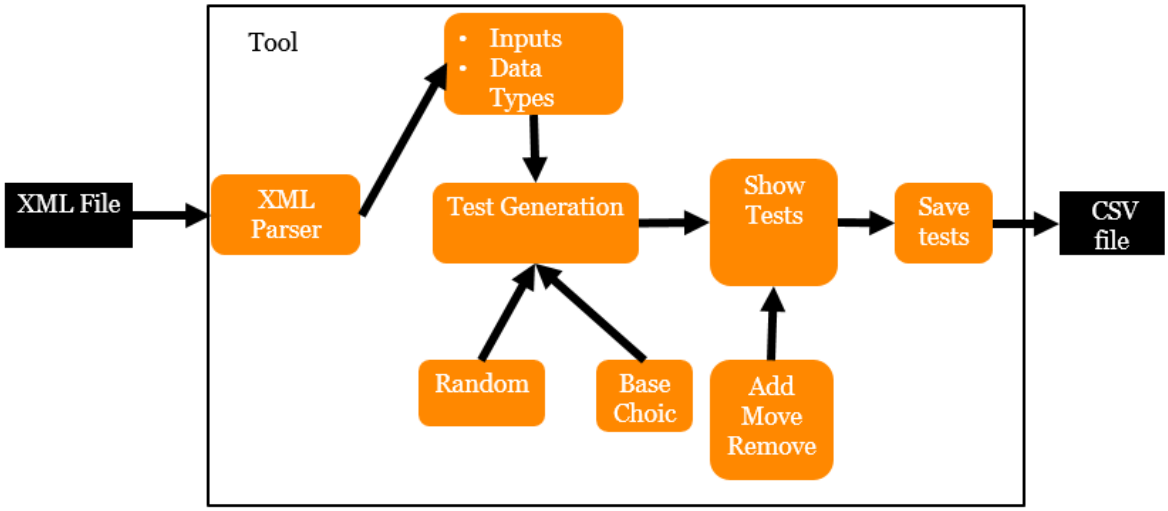

Figure 1. Basic functionality diagram

2. System overview

Inside Bombardier's trains they have a software system called TCMS. This system controls operations such as braking, heat level of engines, and opening and closing of doors etc. Since TCMS is controlling such crucial parts of their trains, it is of high concern to test it extensively. Currently this is done by technicians that manually plot test values into a test program.

The purpose of our program is to reduce the workload on the technicians and at the same time provide the possibility to run more tests on TCMS. The computer generated XML-files that holds the names and datatypes of each variable serves as the input to the test tool. The CSV file generated from the tool will be used as input for testing the software of TCMS.

By doing this they can generate many more tests and with random values that would take too long for the technicians to manually plot into the test program. With a higher number of tests runs the chances of finding critical bugs in TCMS increases.

3. High level description of the functionality

The combinatorial test tool should be able to generate test cases based on the variables and data types provided as input. The input can either be loaded from an XML file selected by the user or manually entered into the GUI. The tool should allow users to choose whether the test cases should be generated based on the random technique or the base choice technique.

If random technique is selected then the users should be able to enter the number of test cases as well as the interval of each input value. The interval can be single or multiple. The test cases are generated as combination of random values selected automatically from the provided intervals. If base choice is selected then the users will input a base value and an interval for each input. Based on the selections, test cases should be generated. A base test is formed by using the base value for each input. Succeeding tests are generated by combining the non-base choice value of each input with base choice value of other inputs.

Users should be able to save the test cases into a CSV file which will be used for testing the TCMS software. The tool may also allow users to add, remove and reorder test cases. The users may also have an option to save the last displayed data and reload it on GUI when it is opened again.

4. Software Architecture

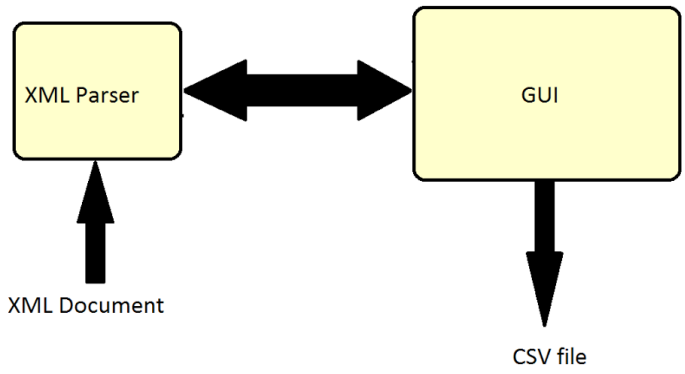

Figure 2. Software Architecture

The architecture of our Combinatorial Test Tool consists of two main independent entities: the XML Parser and the GUI. The Parser retrieves a document and sends the relevant data from it to the GUI where it is presented. From the GUI, the user can then generate tests, modify the values of the given variables, add interval ranges and save the generated tests to a CSV file. More detailed descriptions will be shown in the “Detailed Software Design” section.

5. Detailed software design

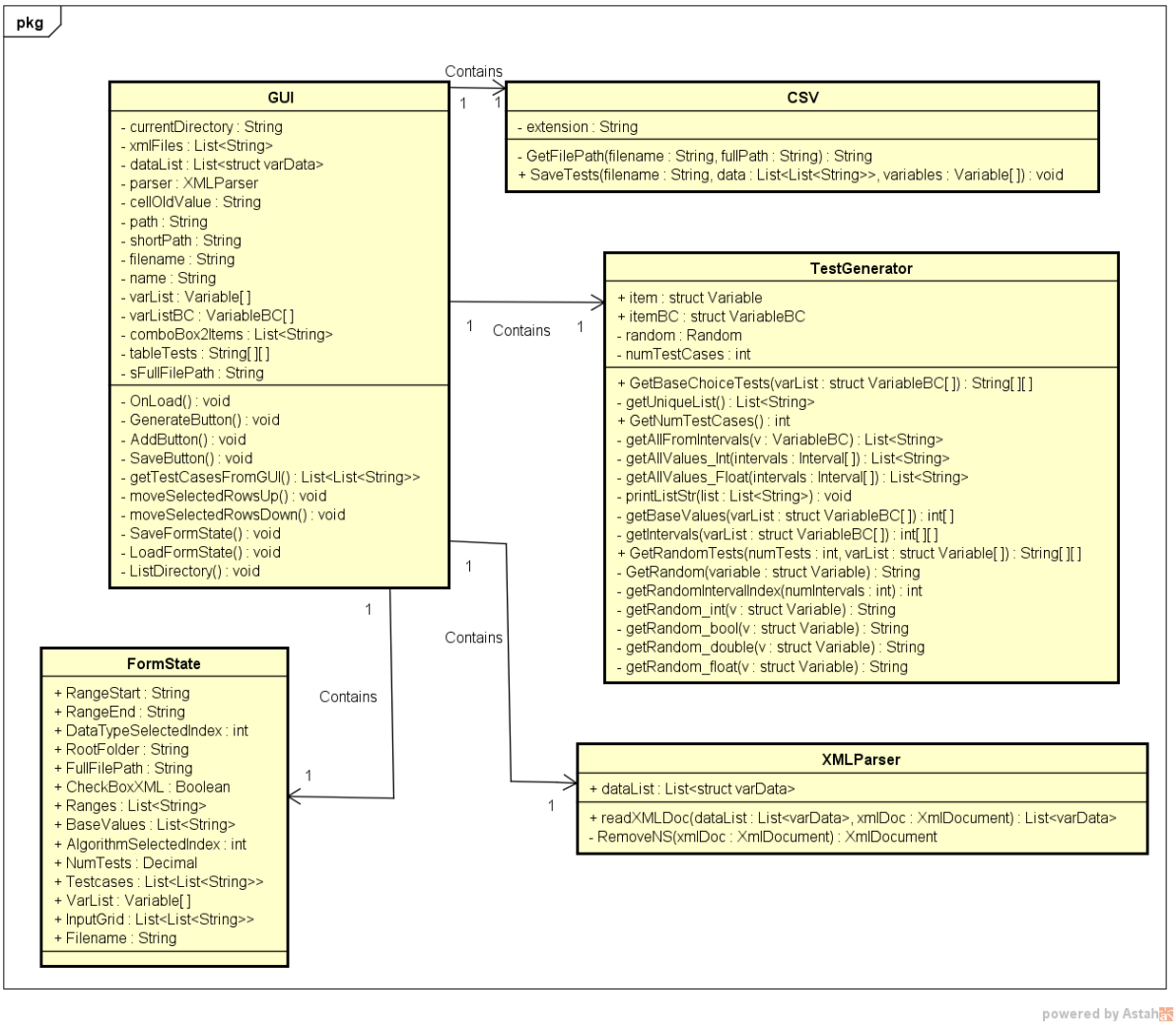

Figure 3. Class diagram of the tool

The software consists of a GUI, XML Parser, TestGenerator, CSV and FormState. The GUI handles all the graphical aspects and forward the data between the different entities. It presents the different Data Inputs gotten from the XML Parser that should be changed and used in the test generation. The Test Generator provides the GUI with data to be shown to the user.

The XML Parser gets the XML document from a directory provided by the GUI. It then extracts the input variables from the document and saves them in a list of structs named varData (that contains the variable name and type).

In order to generate the tests, the TestGenerator module gets the extracted variables from the list of structs. Based on the chosen algorithm (base choice or random) given by the GUI, appropriate values are assigned to each variable depending on the given interval also specified by the GUI.

The CSV takes the filename, data from the TestGenerator presented in the GUI and saves the test in a file of CSV format.

The FormState is used when saving the state of the project. The FormState should contain all the necessary information to be able to recreate the state, and is saved as an XML file. This XML file is then used when loading the saved state of the application.

Figure 4 shows the activity diagram of the tool, with the activities divided across four different swimlanes - GUI, XML parser, Testgenerator and Save to file. Activity ‘modify tests’ in the diagram includes adding, changing, deleting and reordering of test cases.

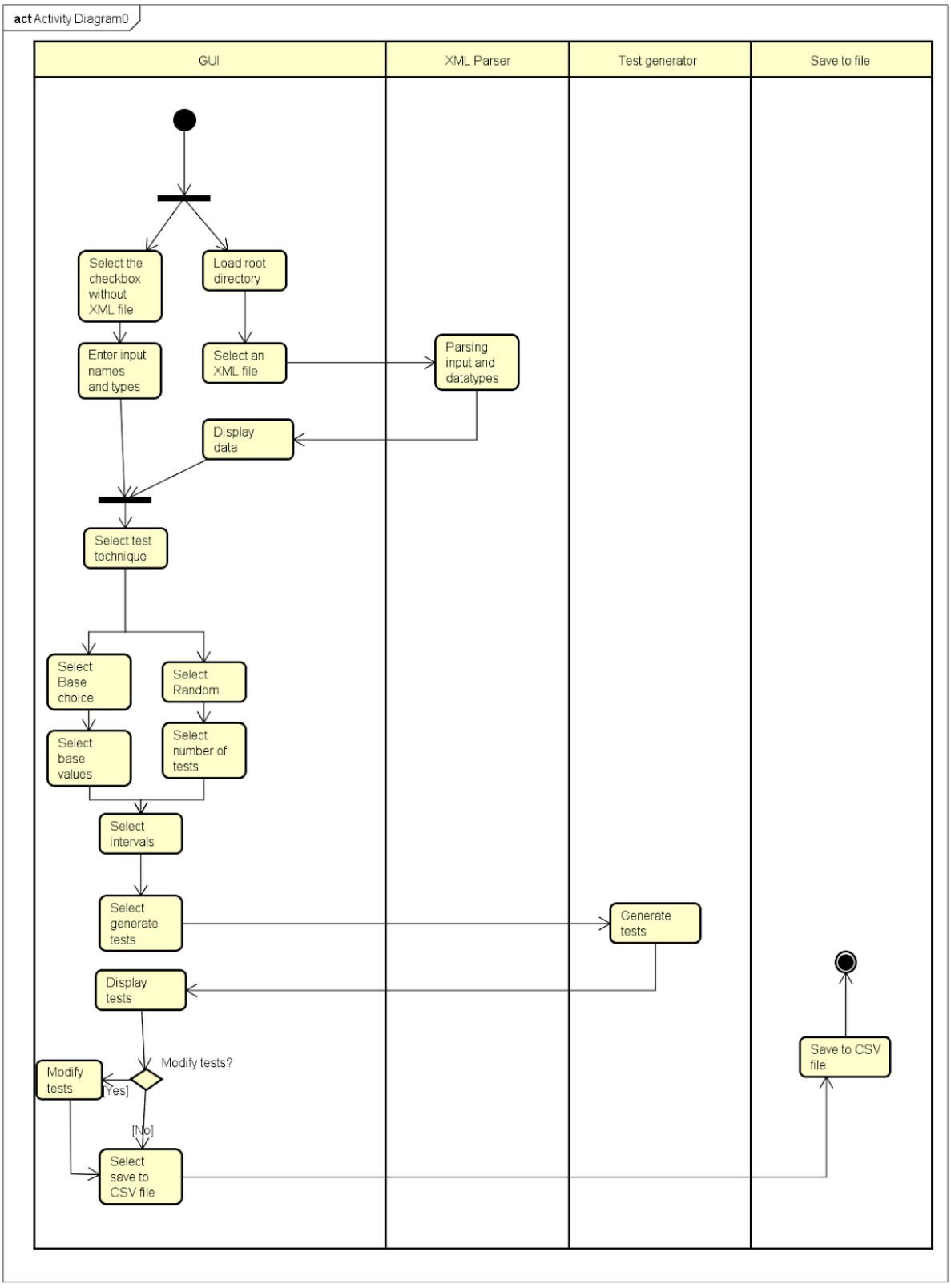

Figure 4. Activity diagram

6. GUI design

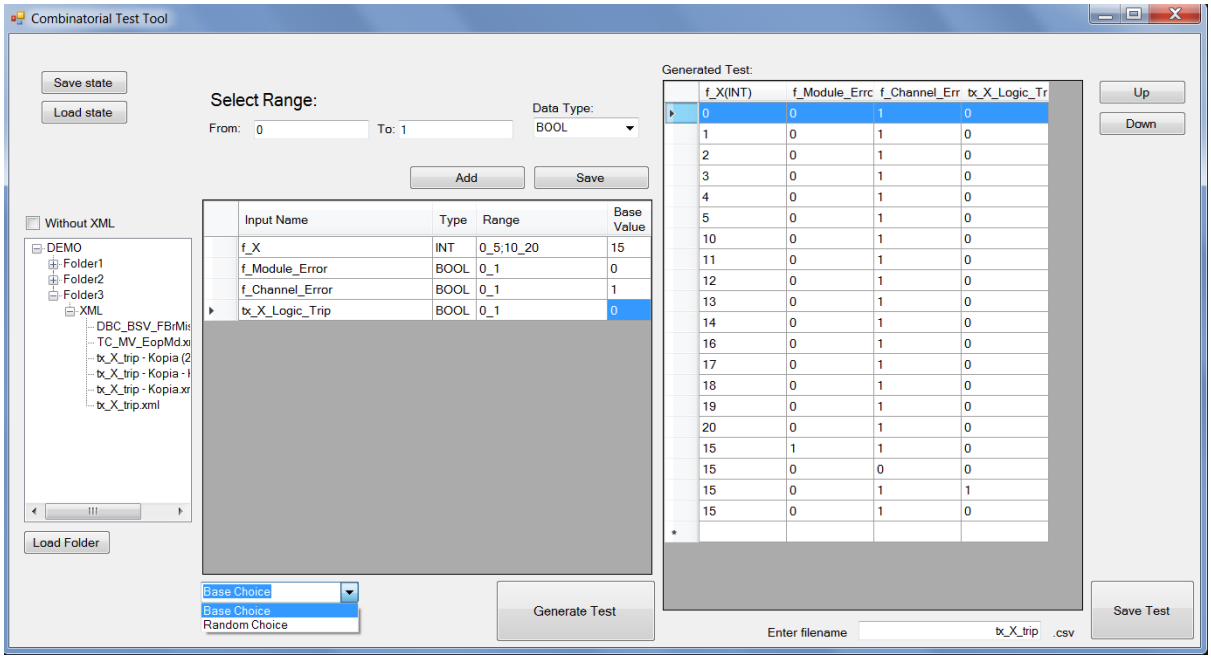

Figure 5. GUI design, using an XML file to generate tests

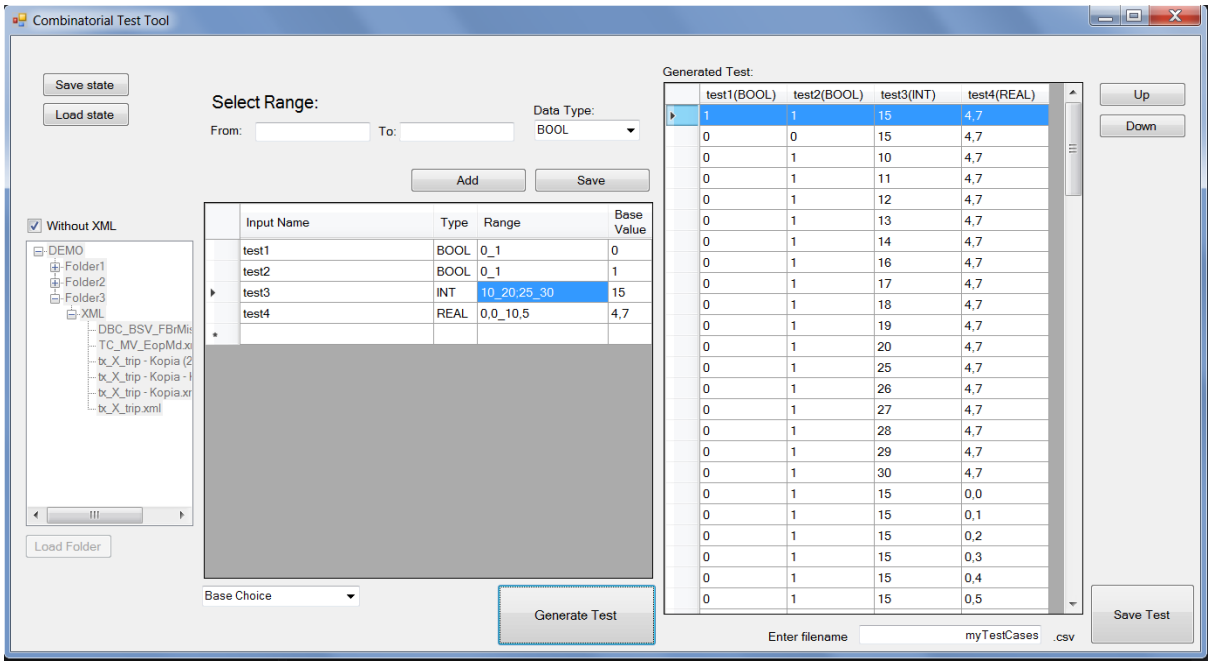

Figure 6. GUI design, without loading an XML file

To generate a test the user will first have to load which folder he wants to set as the root folder for the treeview. The content of this folder and its subfolders will be displayed in the treeview to the far left of the tool. It is now possible to switch between folders and files in the treeview to get the desired XML file to generate tests from. As soon as an XML file is selected its contents is displayed in the datagridview in the middle of the tool. Base choice is always the default test technique. If the user selects random from the dropdown menu instead of base choice, the base value column in the datagridview will disappear, and a selection of what number of tests to be run will appear under the algorithm drop-down box. It is now possible to choose the interval range for each input, using either the select range fields to set an interval for all inputs of that type, or to directly type the interval or values that

you want on a specific input into the datagridview. Once the user is satisfied with all values and intervals for each input, the “Generate Test” button will generate test cases based on those intervals and inputs. This newly generated test is displayed in the datagridview on the right side of the screen. Pressing the “Save Test” button will create a CSV file with the generated test cases with the name in the textbox below the generated test. A filename is suggested based on the name of the selected XML file, but the user may change the filename before saving.

The tool also provides the option to generate tests without selecting an XML file. This functionality is provided to make it possible for developers to generate test cases for programs (XML files) which have not been implemented yet. This is done when the user selects the checkbox above the view containing the treeview (see figure 6). In the left datagridview an empty row will be displayed, which lets the user enter new inputs, with an input name and a datatype, which would have automatically been provided from the XML file if it was used instead.

There is also a possibility to move, remove or add test cases manually before saving to a CSV file. The user can add, remove or edit the tests directly in the datagridview containing the generated test cases. To move one or more test cases at once, the user can use the “Up” or “Down” buttons.

Once the user decides to close the application by clicking the upper right exit button, a popup window will be displayed, asking the user to choose between saving the state of the project or not. If the user decides to save the state, then the next time the user runs the application he can press the “Load state” button to reload how the project looked like before. The state of the application can also be saved by pressing the “Save state” button.