COMBINATORIAL TEST TOOL
PROJECT REPORT

Meera Aravind
Simon Eklund
Linus Eklund
Sara Ericsson
Henning Bergström
Adeel Khan
Juan Antonio López Muntaner

DVA313

Group - 2

14th January 2016

Contents

1. Background 1
2. Overview of Results Produced 2
3. List of Deliverables 3
4. Acceptance Test 3
5. Changes to Organization and Routines 4
6. Positive Experiences 4
7. Missing Functionality and Possible Future Improvements 5
8. Improvement Possibilities 6
9. Worked Hours 6
10. Distribution of Work and Responsibilities 7
11. Total Project Effort 8
Appendix 9

1. Background

Bombardier Transportation is a global leader in the rail industry. The company has several units in Västerås, including the head office of Bombardier in Sweden. In this project, we developed a combinatorial test tool that generates test cases for testing the Train Control Management System (TCMS) of Bombardier. TCMS is the centre of distributed system controlling different operations in train like heating, opening and closing of the doors, the braking system, collecting line voltage, upload of diagnostic data and controlling the train engines.

There are different levels of testing at Bombardier namely, Vehicle Integration testing, System Integration testing, Program Integration testing and Program testing. Our tool will be used in program testing which is the responsibility of developer/engineer.

Figure 1 depicts the basic functionality of the tool. The software runs as a GUI tool written in C# and runs on Windows. User either manually enters the variables and data types or imports an XML file into the GUI which automatically loads the variables. Based on the selected test technique and input intervals entered by the user, the tool should generate test cases. If random technique is selected, test cases should be generated as combination of random values automatically selected from each input interval. Number of test cases to be generated will be specified by the user in case of random technique. If base choice is selected, user must specify base value for each input. 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. The generated test cases will be saved by the user into a CSV file which will be used for testing TCMS.

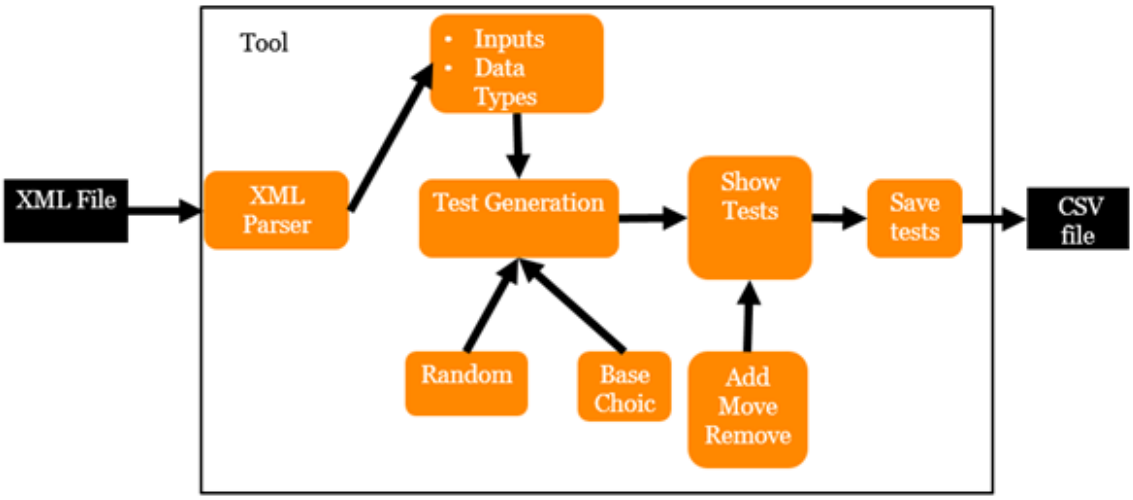


Figure 1. Basic functionality diagram

Currently these test values are entered manually by the users to test TCMS. Our tool shall reduce the workload on users as the values will be generated automatically and can be uploaded for testing TCMS from the saved CSV files. This also provides a possibility to run more tests which in turn increase the chances of detecting bugs.

2. Overview of Results Produced

We have been successful in producing a working combinatorial test tool which included all mandatory and few optional requirements and deliver it to the client within deadline. In addition to the product, all deliverables were prepared and submitted on time. Functionalities that have been implemented in the final product is listed below.

- Created tool written in C# and running on Windows
- The tool can parse XML files and display names and variable types read from the file on GUI
- The tool also facilitate user to directly enter variables and data types if not using XML file
- The tool supports data types INT, REAL and BOOL
- User can select input intervals(single or multiple). Below table shows the format of interval to be entered for different data types with some examples.

Data type	**Format**
INT	1_5;12_16
BOOL	0_1
REAL	1,2_2,3

- The user can select preferred test technique: Random or Base choice algorithms
- User can save tests to CSV file (Comma-separated values)
- User can add, move, edit and remove test cases
- Treeview for displaying XML files of selected folder has been implemented
- The user can also save/load project state (only one state)

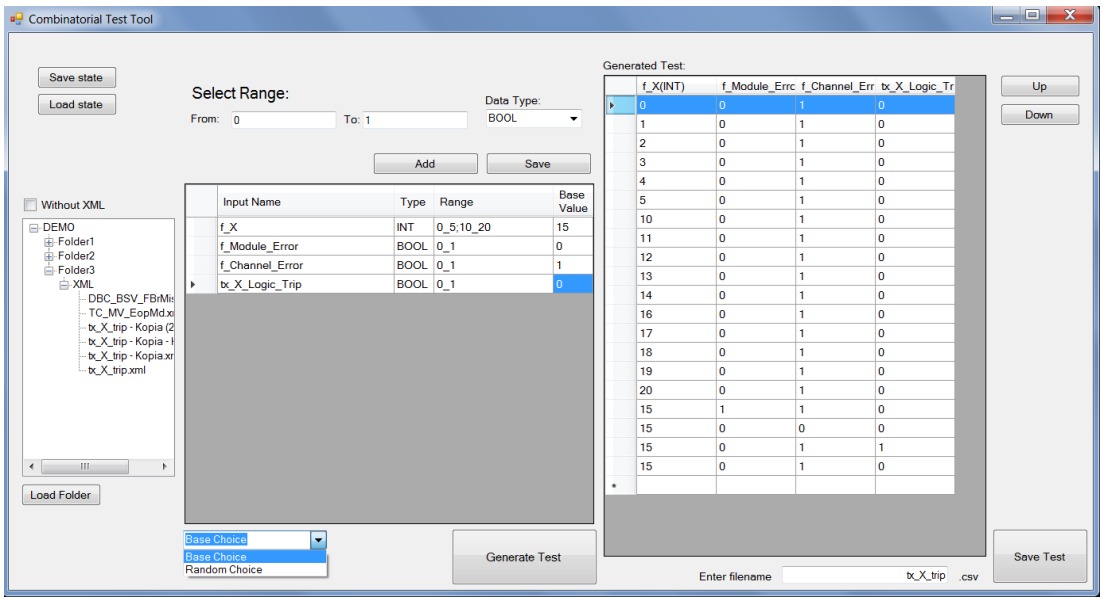


Figure 2. Combinatorial Test Tool

3. List of Deliverables

Below is the list of deliverables delivered as part of the project along with their completion date.

No.	Deliverables	Finished date
1.	Project plan	19/11/2015
2.	Presentation-Project plan and requirements	25/11/2015
3.	Design description (1st version)	03/12/2015
4.	Product (1st version)	03/12/2015
5.	Presentation-Preliminary design and implementation	09/12/2015
6.	Final presentation	13/01/2016
7.	Design description (Final version)	14/01/2016
8.	Product (Final version)	13/01/2016
9.	Project report	14/01/2016

4. Acceptance Test

To do the acceptance test a meeting with the client was scheduled. In this meeting the test was conducted live in front of the client. It was done in this way to give the client the best possible experience and also provide them with a good opportunity to ask questions and to test specific areas and requirements of the application. Detailed descriptions of tests are presented in Appendix.

Test 1 was using the treeview to open an XML file, and then there was single and multiple intervals added to the inputs and test were generated. This test was run both with the base choice algorithm and the random algorithm. Test 1 was successful.

Test 2 consisted of running the application without loading an XML file. This was an optional requirement for the application. In the same way as test one was conducted on both base choice and random algorithm so was this test. Test 2 was successful.

Test 3 aimed to test the optional requirement of modifying, adding, moving or removing rows of inputs in the generated test before saving it to a csv file. Test 3 was successful.

Test 4 tested that the function of saving the generated tests to a CSV file with the proper formatting in it was working. Test 4 was successful.

Test 5 tried to save the state of the application upon closing it, and then being able to load the saved state of the application once it was opened again. This test was only partially successful. When running the tool from the option of loading XML files from the treeview, everything worked fine. However, when working with the tool without loading an XML file and leaving one or more rows of the gridview that holds input names and intervals semi completed the save state function would crash the application.

These five tests served us to test all the required requirements and some optional ones that was implemented. After these tests were done we gave the client free hands to abuse the tool by coming up with tests of their own. When doing this two additional bugs were found.

Firstly, boolean values should not be able to take in any other values than 0 and 1. At this moment any integer value will be accepted by the application even though it provides an error message to user saying the input is incorrect. It will not crash from this but it will not provide the desired test values either.

Secondly a problem was found when entering a decreasing interval instead of the intended increasing ones. With the random algorithm this causes the application to crash, in base choice it will not make the application crash but it will not provide the desired test values.

The overall outcome of the acceptance test was a success. The application offers the desired features, but due to lack of time and a proper testing period, there are room for improvements to the product. Regardless, the client was happy with the results produced in the amount of time that was at hand.

5. Changes to Organization and Routines

A few changes of the roles have been made during the project. In the beginning of the project the role configuration manager was assigned to Henning, but later on changed to Juan, due to previous experience of GitHub. The role of document manager was discarded since it never seemed necessary to have this role when all the deliverable documents were shared in Google Docs, and the presentations for the weekly steering group meetings were managed by the member responsible for preparing the presentation.

The initial plan was to conduct three team meetings per week, though this was not possible to go through with each week because of differences in the members' schedules. The idea of having a member sending everyone the minutes of meeting along with the tasks assigned to each member was never implemented and instead this was discussed during the meetings.

6. Positive Experiences

One of the first issues that arose in the group was how communication should be handled throughout the course. The solution to this problem came in two parts. Firstly, a forum where fast and direct communication of small things could be handled. Here the Facebook chat room served the goal fine. Secondly, there was a need of more reliable form of communication that would eliminate as much confusion and misunderstanding as possible.

For this the only good way found was to agree on having face to face meetings, because they provide the opportunity for everyone to clearly state problems and ideas, and at the same time make sure that everyone understand what was meant. The way these two means of communication was combined gave a good and solid platform for both problem solving and idea sharing within the group.

Another issue that came to light early in the work was that no one in the group had any real experience in estimating how much time different tasks in the project might take. As a solution to this problem, planning poker (planitpoker.com was used for this project) was introduced. An estimation technique that consist of that everyone places a vote on how many hours they think a task will need to be completed, the result is then summarized and discussed, and a time estimation is set based on the different answers. This was a technique that suited us very well and provided us with a decently accurate estimation. Our time estimation was not in any way perfect but considering this is a new experience for every member of the group we did get a close enough estimation.

Lastly we would like to highlight the fact that this project has been a great experience overall. Not only has it been great to work with different people with different skills and from different cultures, it has also given us a deeper understanding of what might be expected from us and what we can expect from clients in the future.

7. Missing Functionality and Possible Future Improvements

Due to the lack of time and the additional functionalities that was needed in the end, some prioritizing was necessary. We focused on completing the main parts of the program, which were the graphical user interface, the XML parser, random & base choice algorithms and save to CSV. This also meant that some functionalities and error handling was left out.

Bug fixes & error handling:

- INT should only be able to have integers in the interval
- REAL should only be able to have float values in the interval
- BOOL should always be set to 0 - 1 as interval
- The user should not be able to input decreasing intervals
- Base values should be set according to the type (floats for REAL, integers for INT)
- Error handling for faulty interval format input (correct format should be value1_value2 with each interval separated by semicolon)

Improvements & additional features:

- Support for additional input types (UDINT, DOUBLE)
- Save states include all XML files (not just the currently open one)
- The option to have several states saved
- Suggest Initial value from XML file as Base value
- Run as DLL callable by other applications
- Additional algorithms
- The option to let the user add an algorithm
- Scalable window and content
- The option to choose save directory

- Root directory for treeview remains the same between sessions
- Code optimization

8. Improvement Possibilities

Right from the start there was some issues with how to keep track of, collect and display how much time different member had spent on different tasks in the project. The solution that we went with was that every member should plot in what task and how many hours they had worked on that task into a excel sheet. The idea was good in theory because it would provide us with a detailed chart of all the tasks and their time consumption. However, the one task of actually filling in the excel document was forgotten in the excitement and rush to deliver a working product, and we ended up with no real record of how much time went into which task. This is something that would have to be improved upon for future project. The solution of the excel sheet itself was not bad, just the execution of it.

Instead of testing the program during the development period it would be better to have a proper testing period to find all the bugs that would need to be fixed. This would improve the quality of the application since more errors would be found at an earlier state.

Another important improvement to be done is to change the way we have shared the code. In this project we have done it via email. This fact has caused a huge desynchronization between us. A good option to solve this would be to commit changes in a version control system such as GIT or SVN, thus every member of the project group would have an updated version of the code. Then afterwards, see what changes has been done and what parts remains the same or even remove changes that have been made.

Finally, to get a better and faster work flow in a project it would be a good idea to put a little more time into first researching exactly what it is that the project will contain. Once that is done, tasks should be distributed to project members based on their skills and strengths, not just randomly divided.

9. Worked Hours

Members	W.46	W.47	W.48	W.49	W.50	W.51	W.52	W.53	W.01	W.02	Total
Meera	18	22	20	21	19	5	0	0	3	20	**128**
Linus	18	18	22	17	19	7	3	0	5	20	**129**
Adeel	18	20	17	17	18	8	7	0	3	20	**128**
Simon	18	18	17	17	19	9	4	0	4	14	**120**
Henning	18	18	16	17	19	13	6	0	7	14	**128**
Juan	18	18	18	17	19	6	8	0	3	16	**123**
Sara	18	18	21	17	19	11	2	0	5	20	**131**

10. Distribution of Work and Responsibilities

Responsibilities:

- Meera - Project manager
- Simon - Client contact
- Juan - SVN/GIT manager

Division of work (Development & Implementation):

- Henning & Juan - XML Parser
- Simon & Linus - Graphical User Interface and error handling
- Sara, Meera & Adeel - Random & Base choice algorithms, save to CSV (comma-separated values) and edit of test cases (Add/Move/Remove)
- Linus - Treeview
- Simon - Use tool without XML
- Sara - Save project state

Division of work (Presentations):

- Project plan and requirements (Presentation) - Meera & Linus
- Preliminary design and implementation (Presentation) - Henning & Juan
- Final presentation and live demo - Simon, Sara & Adeel

Division of work (Documentation):

All members of the group took part in writing and controlling the documents. Although each member were responsible for writing different parts of the documents, everyone worked together when the final document was put together during a group meeting. During this group meeting all members also went through all parts and made improvements and changes which were needed.

Project plan:

- Sara: Introduction
- Meera: Project Group, Organization and Communication, Planned Effort Per Member, Deliverables, Deadlines and Milestones
- Linus: Quality Assurance, Description of Existing System
- Juan: High Level Description of System and Functionality
- Simon: Initial Project Backlog
- Adeel: Functional and Non-Functional Requirements

Design description:

- Adeel: Short Background
- Meera: High-level Description
- Linus: System Overview and Graphical User Interface
- Henning, Juan, Sara and Simon: Software Architecture and Detailed Software Design

Project report:

- Meera: Background Information, Overview of Results, Produced Deliverables
- Linus: Results of Acceptance Test, Positive Experiences

- Simon: Missing Functionalities and Improvements, Worked Hours, Distribution of Work and Responsibilities
- Sara: Changes to Organization and Routines
- Adeel: Total Project Effort
- Juan: Improvement Possibilities

11. Total Project Effort

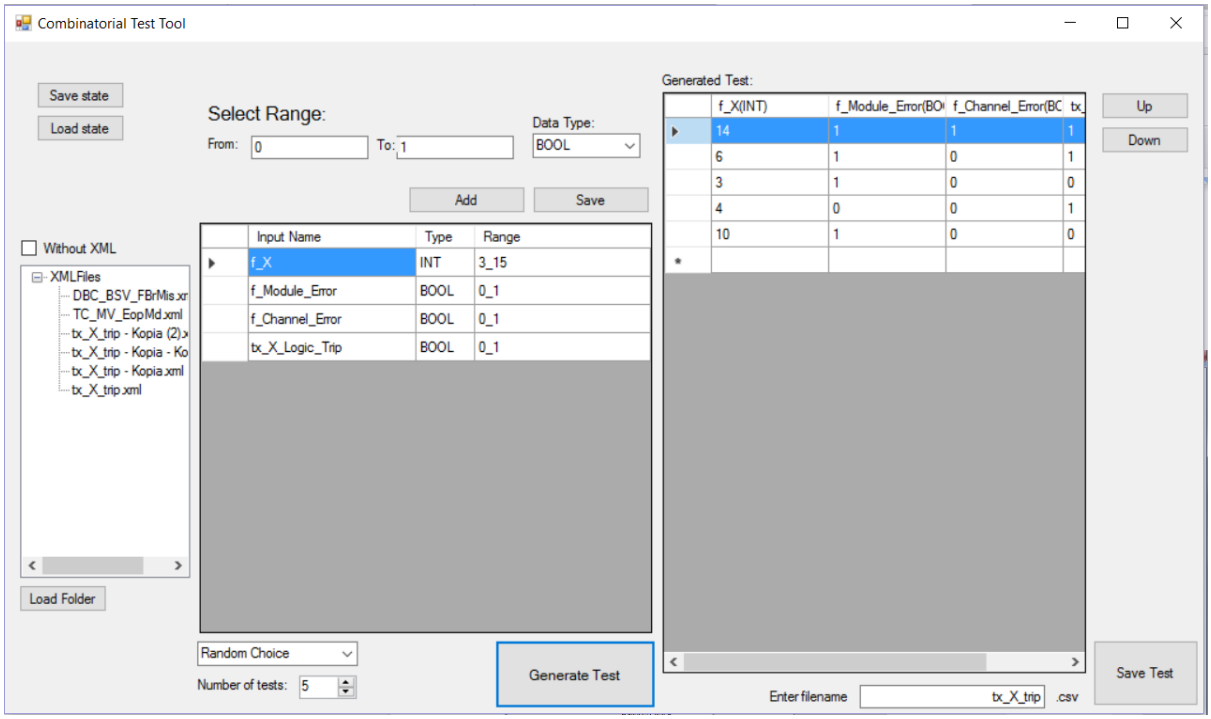
Table shows the approximate time that went into each part of the project. Due to lack of reliability in the time reports this table is showing an as close to reality as possible approximation of the time spent on different tasks.

Project Tasks	Total Person-Hours
Implementation and Testing	
Base Choice & Random Algorithm	140
XML Parser	126
Graphical User Interface	139
Treeview	22
Error Handling	23
Add, Move, Remove Test Cases	15
Use Tool Without XML	5
Save Test CSV	12
Save Project States	18
Documentations	
Project Plan	78
Design Document	76
Final Project Report	80
Monday Meeting Slides	79
Presentations	
Presentation-Project Plan and Requirements	25
Presentation-Preliminary Design and Implementation	23
Final Presentation	26

Appendix

Test 1: Generate test cases using XML file using single and multiple intervals.

a) Random technique

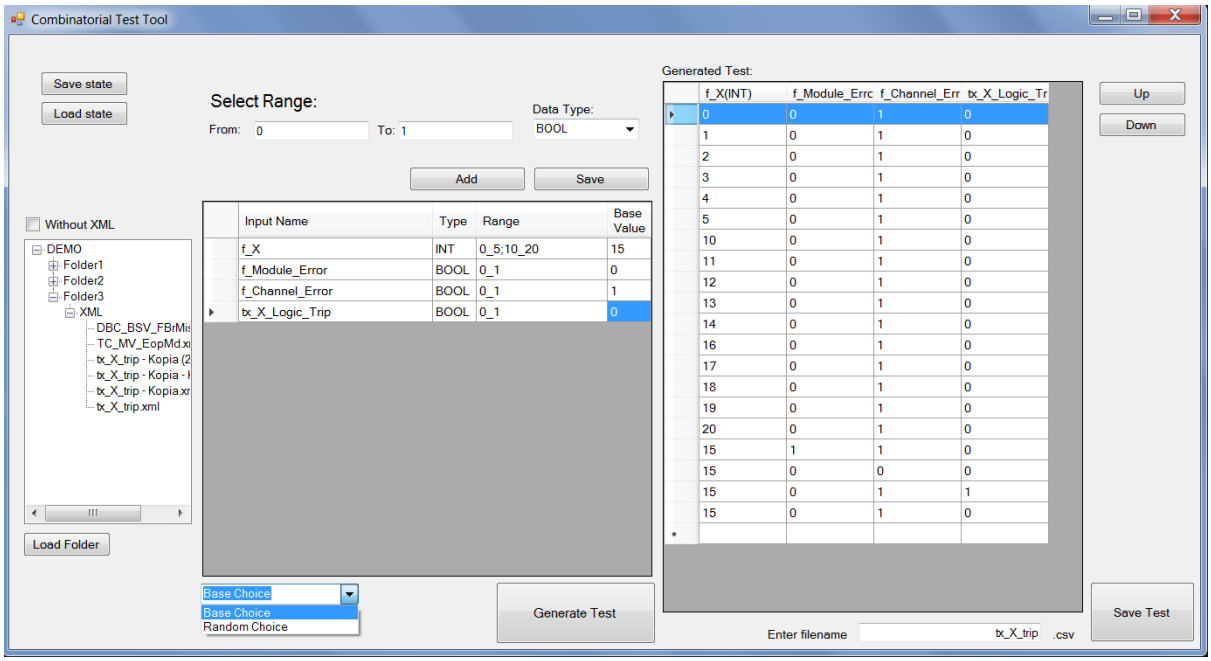


Steps:

1. Click on ‘Load Folder’.
2. Select a root folder containing XML files. XML files will be displayed in a treeview to the left end.
3. Select the XML file. Variable names and data types will be loaded onto the GUI from the XML file.
4. Enter the range of input values. User can enter single or multiple interval.
5. Select Random choice from the drop box and enter the number of test cases required.
6. Click on generate test and test cases will be displayed on the right end of GUI.

Test Result: Successful

b) Base Choice



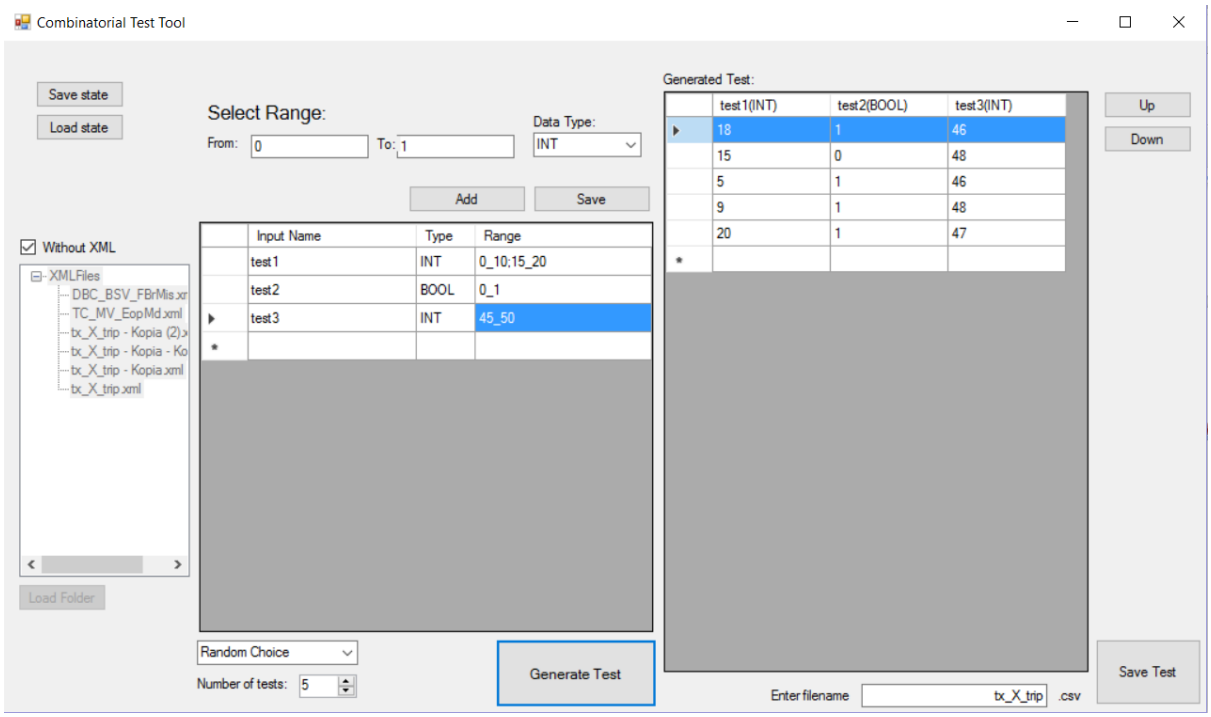
Steps:

1. Click on ‘Load Folder’.
2. Select the folder containing XML files. XML files will be displayed in the list box to the left end.
3. Select the XML file. Variable names and data types will be loaded onto the GUI from the XML file.
4. Enter the range of input values. User can enter single or multiple interval.
5. Select Base choice from the drop box and enter the Base value for each input.
6. Click on generate test and test cases will be displayed on the right end of GUI.

Test Result: Successful

Test 2: Generate test cases without XML file

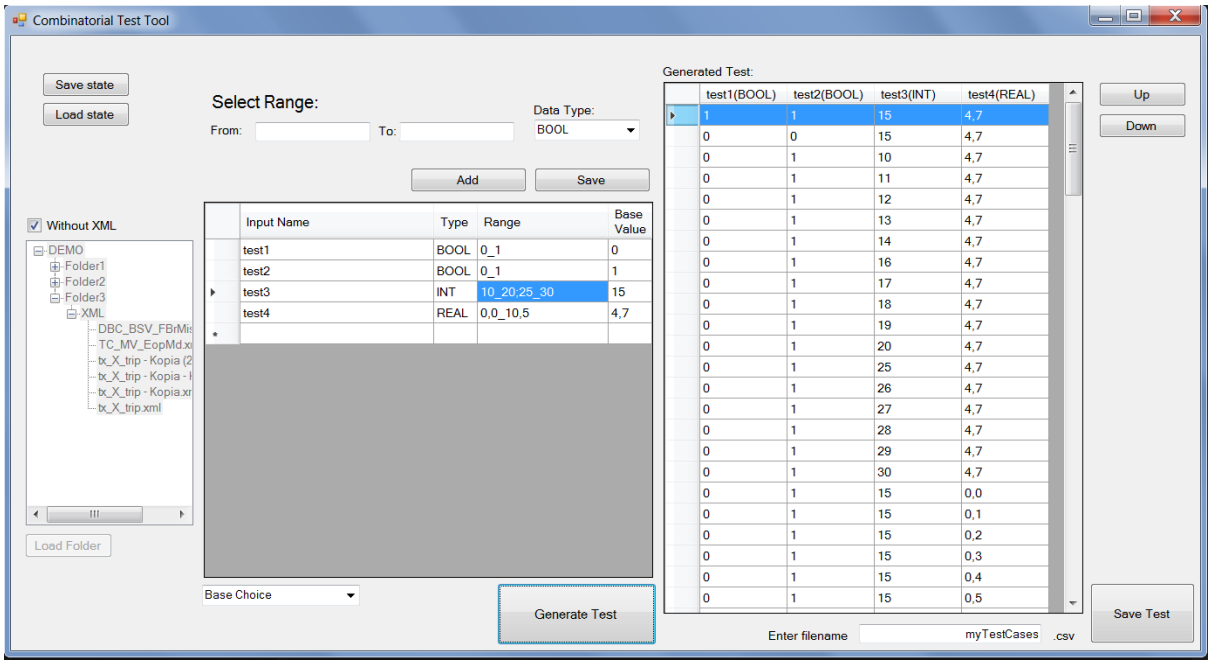
a) Random Technique



Steps:

1. Click on checkbox ‘Without XML’.
2. Enter variable names and data types in datagridview.
3. Enter the range of input values. User can enter single or multiple interval.
4. Select Random choice from the drop box and enter the number of test cases required.
5. Click on generate test and test cases will be displayed on the right end of GUI.

b) Base Choice

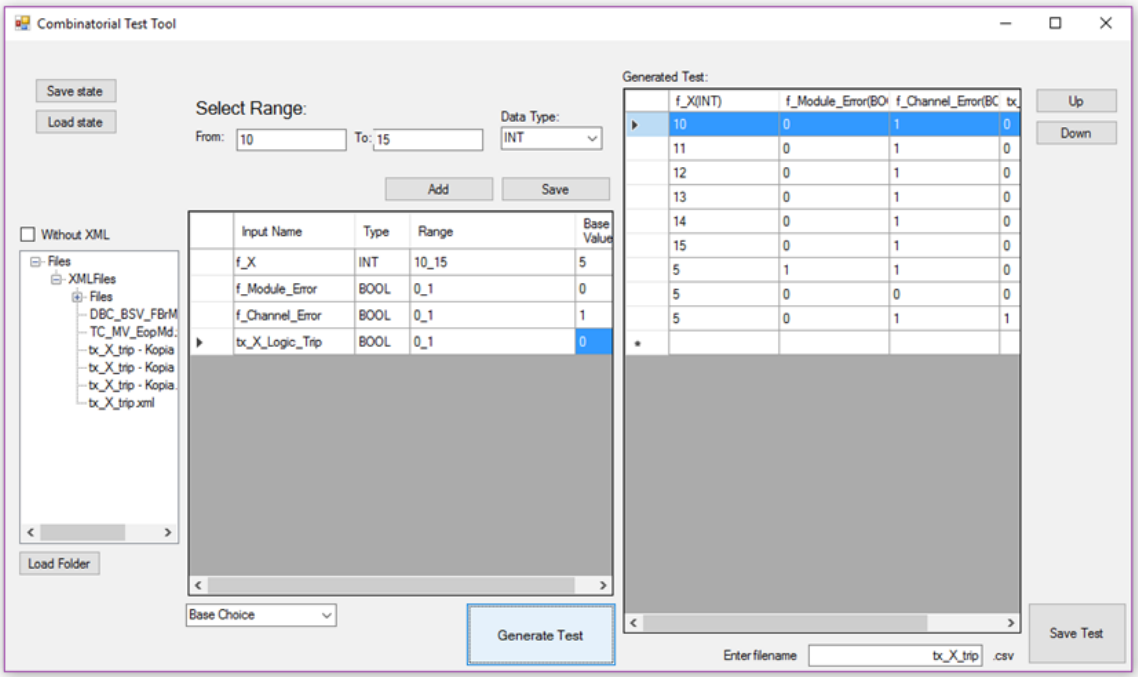


Steps:

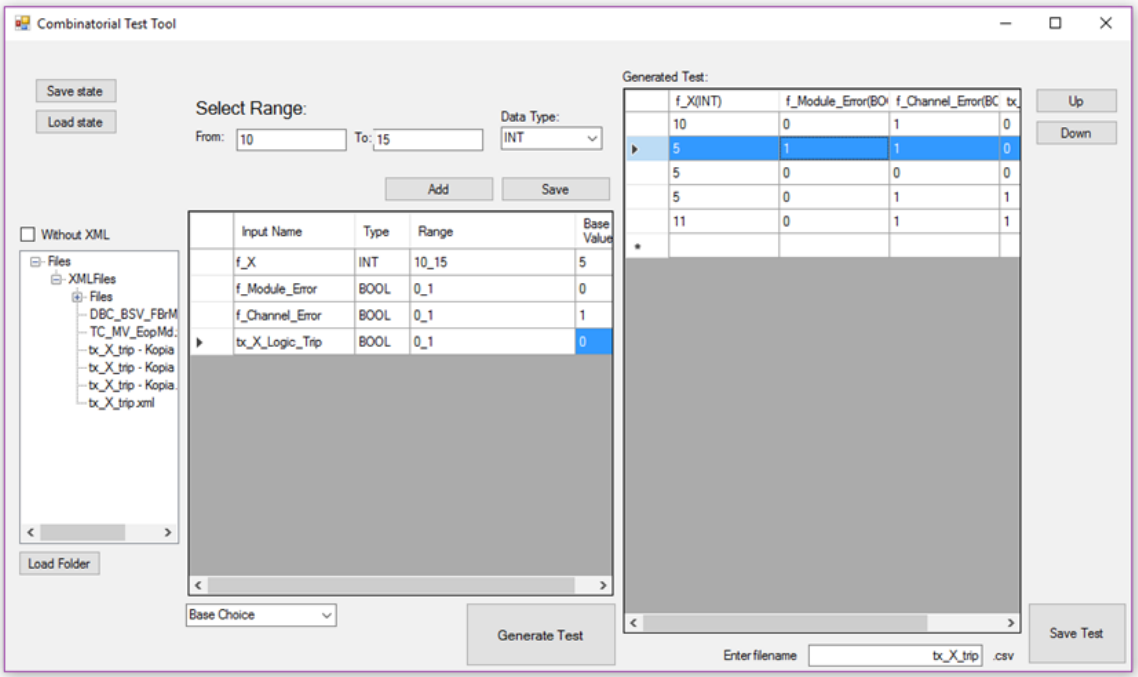
1. Click on checkbox ‘Without XML’.
2. Enter variable names and data types in datagridview.
3. Enter the range of input values. User can enter single or multiple interval.
4. Select Base choice from the drop box and enter the Base value for each input.
5. Click on generate test and test cases will be displayed on the right end of GUI.

Test Result: Successful

Test 3: Edit test cases (add, change, delete, remove)



Before modifying test cases



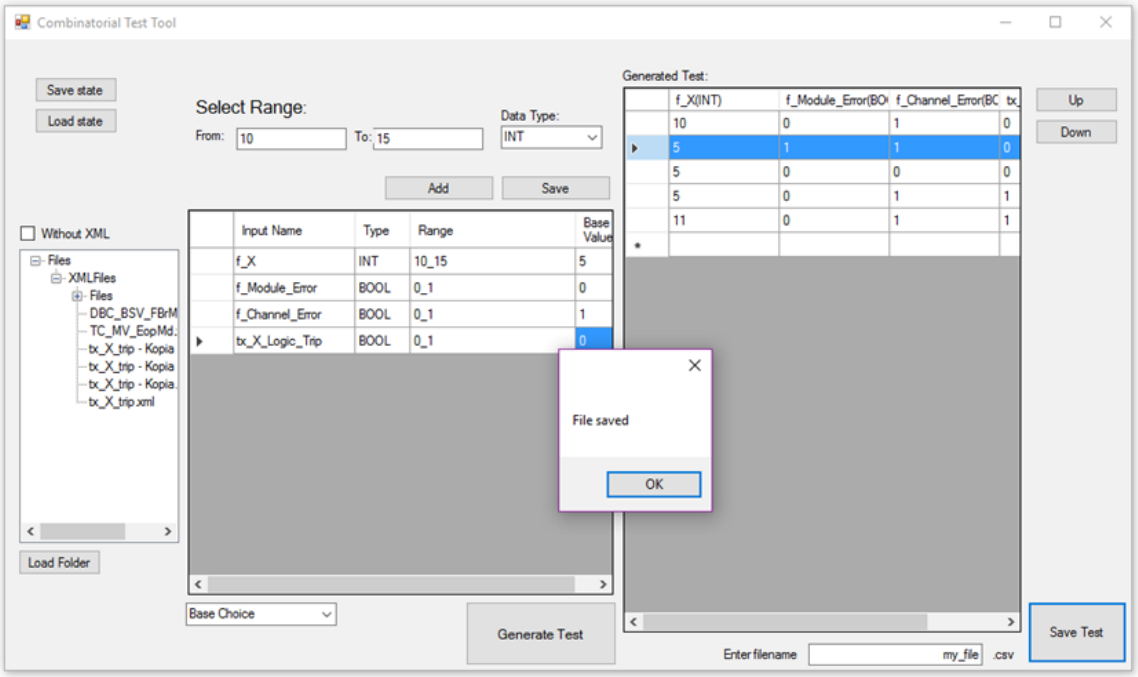
After modifying test cases

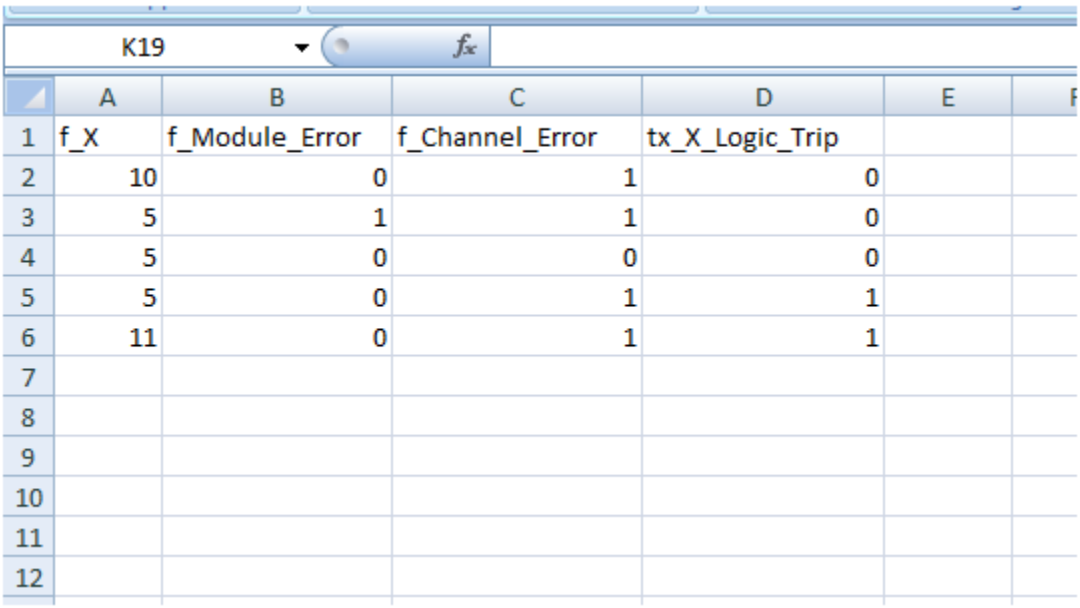
Steps:

1. Generate test cases as described in Test 1 or Test 2.
2. Select a line of test case and click on delete button of keyboard to delete the line.
3. Type new input values at the end of generated test cases to add new tests.
4. Select a line of test case, click on ‘Up’ button to move it upwards by a line and click on ‘Down’ button to move the test downwards by a line.
5. Alter any value in the generated test cases by directly typing in the required column.

Test Result: Successful

Test 4: Save test into CSV file





K19

	A	B	C	D	E
1	f_X	f_Module_Error	f_Channel_Error	tx_X_Logic_Trip	
2	10	0	1	0	
3	5	1	1	0	
4	5	0	0	0	
5	5	0	1	1	
6	11	0	1	1	
7					
8					
9					
10					
11					
12					

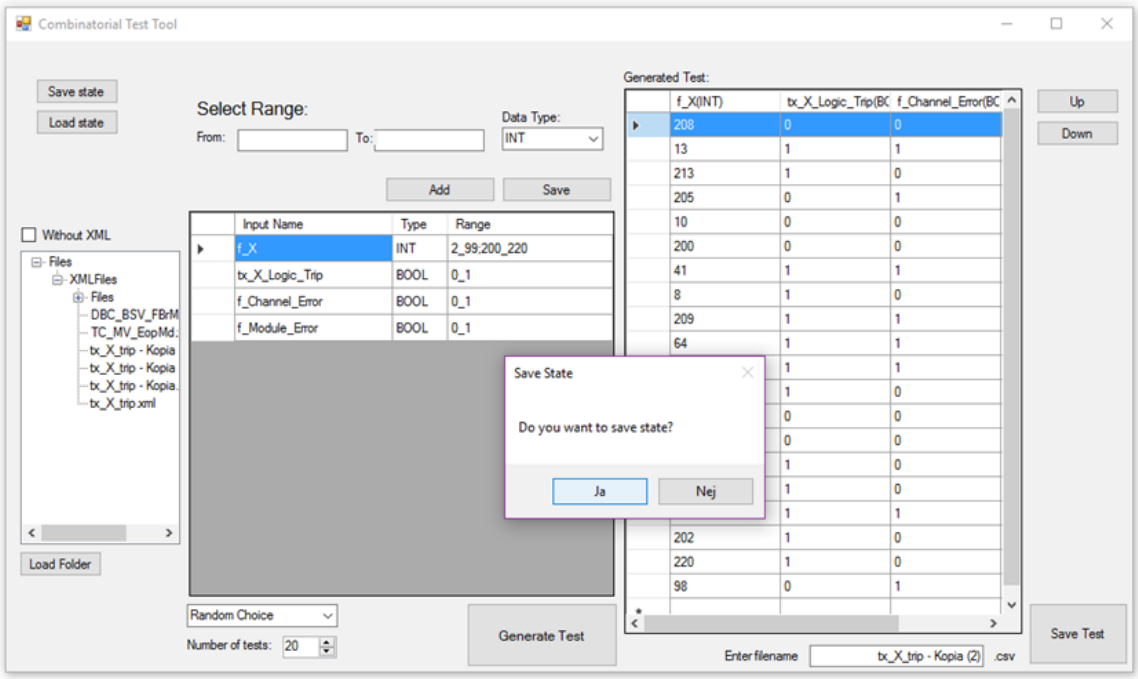
Steps:

1. Generate test cases as described in Test 1 or Test 2.
2. Enter filename in the text box provided at bottom right end on GUI. Default file name will be the name of XML file in case it is generated from XML file. If not generated from XML file, filename defaults as ‘myTestCases’.
3. Click on ‘Save Test’ and you will get a pop up message ‘File Saved’.

Test Result: Successful

Test 5: Save and load state

a) With XML file



Steps:

1. Generate test cases as described in Test 1.
2. Click on ‘Save state’. Or close the application which gives a pop up ‘Do you want to save state?’. Select ‘Yes’.
3. Reopen the application and click on ‘Load state’.
4. All saved values will be reloaded into the GUI.

Test Result: Successful

b) Without XML file

Steps:

1. Generate test cases as described in Test 2.
2. Click on ‘Save state’. Or close the application which gives a pop up ‘Do you want to save state?’. Select ‘Yes’.
3. Reopen the application and click on ‘Load state’.
4. All saved values will be reloaded into the GUI.

Test Result: Successful

c) Without XML file (partially filled)

Steps:

1. Select checkbox ‘Without XML’.
2. Enter variable name and data types. Do not enter intervals.
3. Click on ‘Save state’. Program crashes.

Test Result: Fail

Test 6: Enter decreasing interval

Steps:

1. Select XML file to be uploaded as described in Test 1.
2. Enter a decreasing interval for one of the variables.
3. Select ‘Random Choice’ and enter number of test cases greater that 0.
4. Click on ‘generate tests’. Program crashes.

Test Result: Fail

Test 7: Bool accepts numbers other that 0 and 1

Steps:

1. Select XML file (which has a variable with data type BOOL) to be uploaded as described in Test 1.
2. Enter input intervals for all variables.
3. Enter input interval for BOOL as 0 to 2.
4. Generate test cases using Random or Base technique.
5. Test cases are generated using value 0 to 2 for BOOL.

Test Result: Fail