COMBINATORIAL TEST TOOL
PROJECT PLAN

DVA313

Group - 2

10th December 2015

Contents

1. Introduction

The project is about creating a combinatorial test tool for the Train Control Management System (TCMS) of the client Bombardier. TCMS manages and controls whole operations in a train like the braking system, heating, opening and closing of doors, controlling train engines, air-conditioning etc. This tool will help developers test their programs by automatic generation of test cases.

The test tool, which should be written in C# and running on Windows, should provide a GUI which lets the user import an XML file and then choose between two techniques, random or base choice, to generate tests. Random generates random values for each input within a given interval chosen by the user, which also allows the user to decide the number of tests that should be created. Base choice lets the user assign a base value and an interval to each input and then tests are generated by letting every input be covered at least once while the other values are held constant. The generated tests should then be displayed to the user and exported to a CSV file.

In addition to the requirements above, the project includes some optional parts. The first one is the possibility of editing the generated tests by letting the user add, remove or move tests. The other optional part is to run the program as a DLL callable by other applications. The requirements remain the same except that the set of input values should be provided to another application.

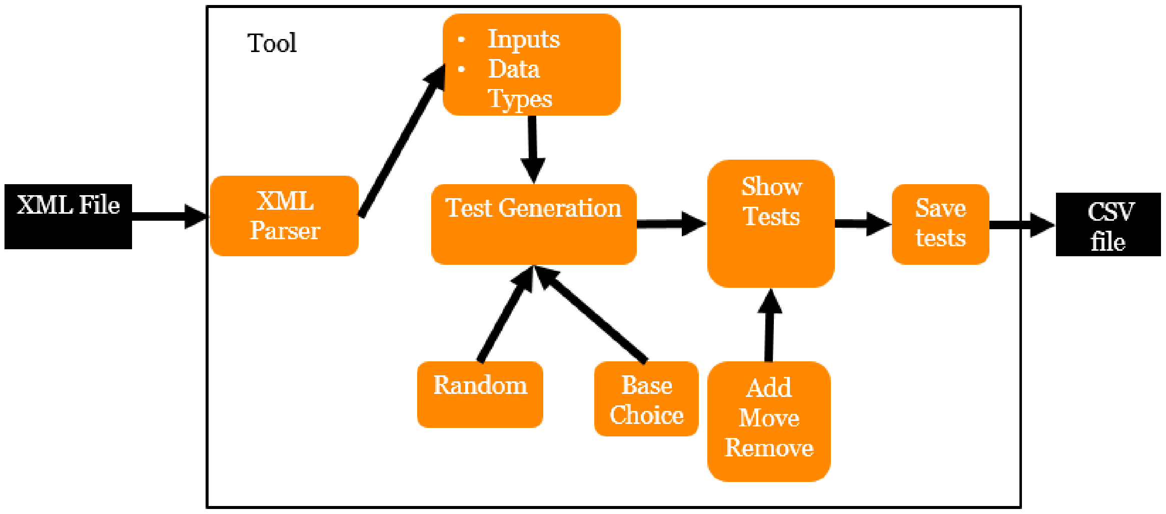

Figure 1. Basic functionality diagram.

2. Project Organization

Organization of the project is detailed below.

2.1 Project group:

No:	Team Members	Contact	Roles
1.	Adeel Khan	adeel6993@gmail.com	Document Manager, Developer - Algorithm
2.	Henning Bergström	hbm13001@student.mdh.se	Developer - Parsing
3.	Juan Antonio López Muntaner	jlr15002@student.mdh.se	Configuration Manager, Developer - Parsing
4.	Linus Eklund	led13001@student.mdh.se	Developer - GUI
5.	Meera Aravind	mad15006@student.mdh.se	Project Manager, Developer- Algorithm
6.	Sara Ericsson	sen13009@student.mdh.se	Developer - Algorithm
7.	Simon Eklund	sed13001@student.mdh.se	Client contact, Developer - GUI

2.2 Organization and communication:

Internal team meetings will be conducted thrice per week (Monday, Wednesday and Friday) where each member discusses the tasks assigned to them and the progress made. The past activities will be evaluated by the team and also tasks to be completed before next meeting will be discussed and distributed among team members. One of the team members will send the minutes of meeting including the tasks assigned to each member, which should be completed prior to next meeting, after the internal group meetings. Documentation work is updated in a single shared google document so that all team members are aware of the portion of work completed. The tasks worked upon by each member will be updated in a shared excel document along with the time spent on it. Coding will be performed individually or in pairs based on tasks assigned and integrated with rest of the development. Configuration manager will be responsible for GIT management including uploading of files into Github and informing steering group about same.

2.3 Planned effort per member for each week in the project:

The estimated effort per member, also considering the vacation plans, is illustrated in the table below:

MEMBERS	W46	W47	W48	W49	W50	W51	W52	W53	W01	W02	Total
Meera Aravind	20	20	22	20	18	10	10	10	15	5	150
Linus Eklund	20	20	20	20	20	10	10	10	10	10	150
Adeel Khan	20	20	20	20	20	20	10	5	10	5	150
Simon Eklund	20	20	20	20	20	20	10	5	10	5	150
Henning Bergström	20	20	20	20	20	20	10	5	10	5	150
Juan Antonio López Muntaner	20	20	20	20	20	20	10	5	10	5	150
Sara Ericsson	20	20	20	20	20	20	10	5	10	5	150

2.4 Deliverables, deadlines and milestones:

A list of important deliverables and milestones is shown below along with their deadlines:

NO:	DELIVERABLES & MILESTONES	DEADLINES
1.	**Deliverables**	
1.1	Project Plan	19/11/2015
1.2	Presentation-Project Plan and requirements	25/11/2015
1.3	Design description (1st version)	03/12/2015
1.4	Product (1st version)	03/12/2015
1.5	Presentation-Preliminary design and implementation	09/12/2015
1.6	Final presentation	13/01/2016
1.7	Design description (Final version)	14/01/2016

1.8	Product (Final version)	14/01/2016
1.9	Project report	14/01/2016
2.	**Milestones**	
2.1	Completion of requirement specification	17/11/2015
2.2	Completion of design	24/11/2015
2.3	Completion of implementation	08/12/2015
2.4	Completion of preliminary testing	10/12/2015
2.5	Final product Delivery	07/11/2015

2.5 Quality assurance:

There will be internal testing by the group on everything that is going into the final artefact (including code, documents and presentation slides). All parts of the development will be done with at least two persons working on them with the hope of providing a two-step quality assurance. At first, the artefacts have to be accepted by the people working on them, followed by approval from the whole group, ranging from design choices to test of functionality. The final product will be tested with XML-files provided by the client, to verify that the program is doing what it should and runs in a stable manner.

3. Description of the system to be developed

3.1 High level description of the domain and functionality:

The client needs to test his software used in TCMS in order to get a functional execution without any problems, because if the software fails people could be harmed. To achieve this, it is necessary that software should be tested properly using different input values. The tool to be developed will generate a number of test cases which can be used as input to the client’s software. From the generated list of test cases, the client will be able to choose the desired inputs for the software to be tested. The tool will also allow the user to save the displayed test cases into a csv file.

The use case diagram of the test tool is shown below:

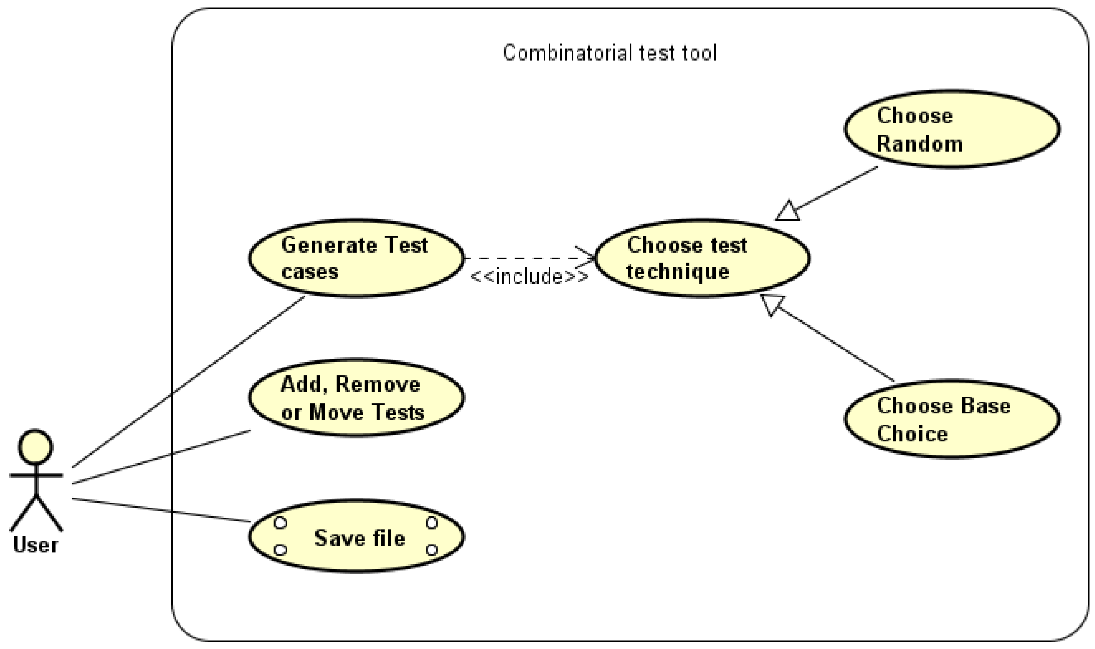

Figure 2. Use case diagram.

1. Generate Test cases:
   1) User uploads XML file.
   2) User selects test technique (Random or Base Choice technique).
   3) User enter intervals of inputs.
   4) Base choice value for each input is entered in case of Base Choice technique. Number of tests is entered in case of random technique.
   5) User generates test cases
2. Add, remove or move tests:
   1) User generates test cases.
   2) Use adds, removes and reorder test cases.
3. Save file
   1) User generates test cases.
   2) User modifies test cases.
   3) User saves test cases.

3.2 Description of the existing system:

The final artefact from the project will fit in between the technician that is plotting the values for the test and the existing testing program. The artefact is created to take some workload off the technicians and to provide the possibility to run randomly generated test sequences during times where technicians cannot run tests. The software subjected to the tests is controlling braking systems and door systems among other functions on trains. It is therefore crucial that as many bugs and faults as possible are found by the tests. The ability to do random testing 24 hours a day provides the possibility to find bugs that would never have been found by the technicians just by entering the most suspected values.

3.3 Initial Project Backlog:

The number of person hours was estimated using planning poker method. The tool used was PlanITPoker (www.planitpoker.com).

ID	**ITEM**	**DESCRIPTION**	**PRIORITY**	**ESTIMATED TIME (PERSON HOURS)**
1.	**Project Plan**	The document should present some basic information about the project, how the work will be organized and the initial set of requirements	HIGH	30
2.	**Project plan and requirements (Presentation)**	The presentation should describe how the project is organized, the project plan & performed activities to capture the customer's requirements	HIGH	10
3.	**Design Description**	The document should capture important design decisions we have made in the project, and should provide a good basis for understanding the implementation	HIGH	30
4.	**GUI**	Implement a Graphical User Interface to make the program user friendly	HIGH	100
4.1.	Design	-	-	-
4.2.	Implementation	-	-	-
5.	**XML Parser**	Implement the functionality to insert a XML file to the application and retrieve variables (names, types and values)	HIGH	120
6.	**Random Algorithm**	Allow the user to choose random values for each input for the test	HIGH	50
6.1.	Choose number of tests	-	-	-
6.2.	Choose value interval	-	-	-
7.	**Base Choice Algorithm**	Allow the user to choose an option to generate a test that covers each value input at least once	HIGH	70

7.1.	Choose Base Choice value	-	-	-
7.2.	Choose value interval	-	-	-
8.	**Preliminary design and implementation (Presentation)**	The presentation should show that the design and implementation work is going in the right direction	HIGH	20
9.	**Save Tests**	Implement the option to save test for later uses	HIGH	16
10	**Final presentation**	A presentation about the project work and the final product	HIGH	12
11.	**Project Report**	The document should summarize the outcomes of the project, both in terms of results produced and experiences from the project work	HIGH	100
12.	**Manage Tests**	Allow the user to manage tests in a variety of ways	MEDIUM	50
12.1.	Add Test	-	-	-
12.2.	Move Test	-	-	-
12.3.	Remove Test	-	-	-
13.	**Run program as a DLL callable by other applications**	The program should be able to run as a DLL callable by other applications, allowing the application to set the input values. Same requirements as in the case of GUI	LOW	30

3.4 Additional Functional Requirements:

1. The user should not be able to input files other than XML files.
2. The user should not be able to input XML files with incorrect format.

3.5 Non-Functional Requirements:

1. The application should be able to run at the same time as other applications, i.e. the application should not consume too much memory or use up too much of the CPU.