Chapter - 05 Project Implementation

5.1 Software used

Implementation of the test automation framework is mainly done with two open source software projects. Apache JMeter (http://jakarta.apache.org/jmeter/) is used as the Test Automation Tool. Apache Maven (http://maven.apache.org/) is used as the project/content management tool.

Badboy freeware tool to record JMeter specific scripts and few JAVA classes used to perform some background activities. Apart from that, IntelliJ IDEA as IDE, XmlSpy for manipulating xml and xsl files, Adobe Photoshop for design graphic stuff in reports, Macromedia Dreamweaver for CSS, JavaScript editor and report designer.

5.2 Test case bundle implementation

The implementation of the test case bundle is shows in Figure 5-1. Folder called “mecp-test” contains all the source files.

In “commons” folder, it contains the XSL files which are using to transform final test results XML into a HTML page. All images relevant to the report also included in the same folder.

“mcommodity”, “morder”, “minvoice” and “mtravel” are few products which added to test automation framework. So it means, every new product or project which we want add to the test automation framework, should be added with a new folder and correct folder hierarchy in that folder. Figure 5-1 shows the correct folder hierarchy. There are 3 major file types included in each project or product test cases. In the “coretests” folder, it contains JMeter specific script files (with “jmx” extension). The way to create and prepare those files will be describe in the user manual. “properties” folder contains parameter files relevant to each jmx (JMeter script) file. This parameterization expects great flexibility and reusability of JMeter scripts. “inputs” folder contains the input files which can be used to input, number of parameters to test cases or files need to upload with test cases.

“src” folder contains JAVA source files which are using by the Test Plugin. These files used to perform some file manipulation and background processes in the framework.

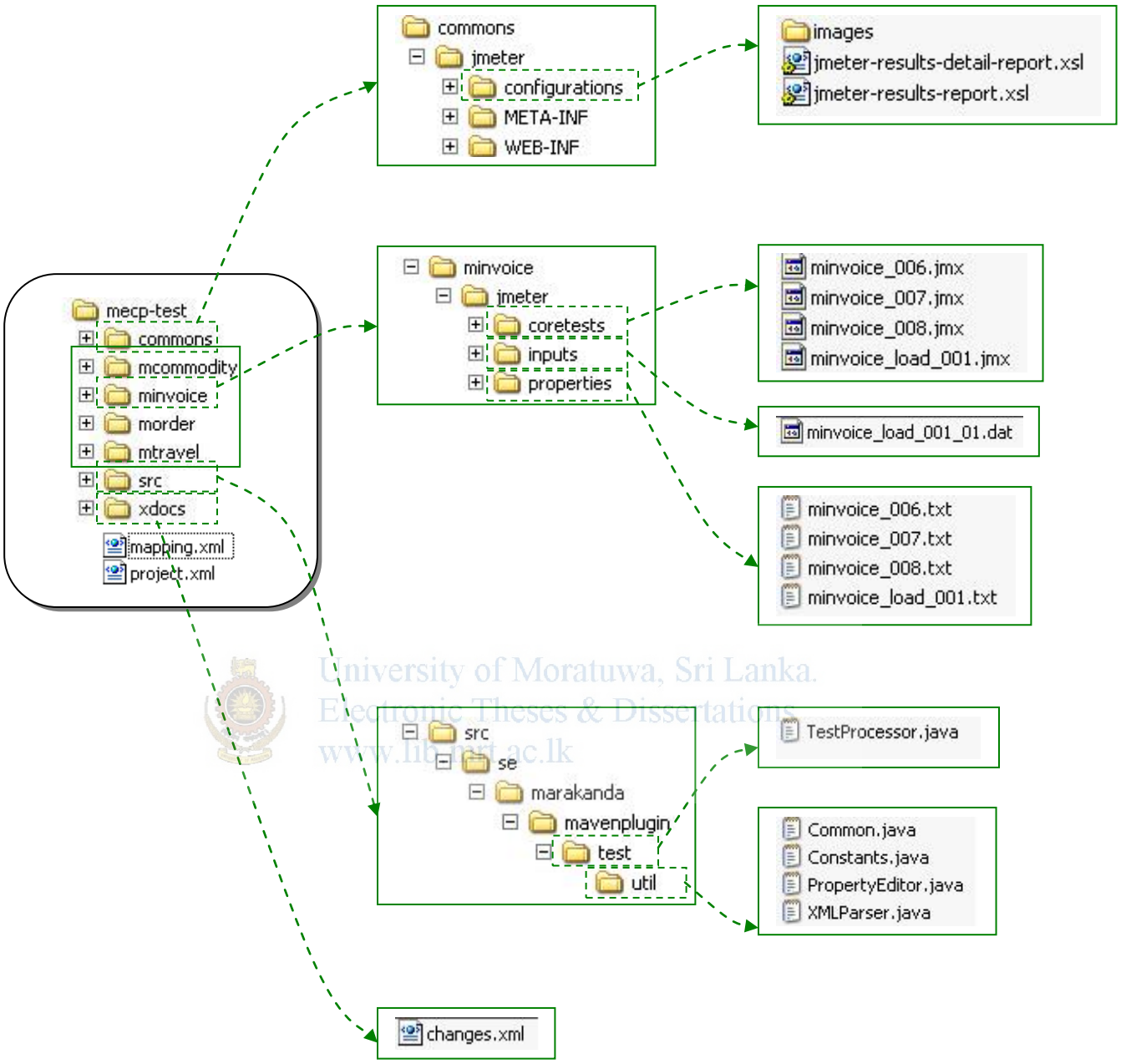

Figure 5-1 Test Case Bundle Implementation

The most important part of this test case bundle is “mapping.xml”. It contains all the data regarding the test cases. Simply it handles the integration between all the projects/products and test cases. As Figure 5-2 describes, this file can have many <module>s. This module

is equal to a project or a product. Under each <module> there can be many <testcase>s. Under the <testcase>, <caseid> holds the value of test case. So the value contains <caseid> must be equal with name of both jmx (JMeter script) file and property file. All the input files comes under the test case should specify in <inputs> as a <input>. There can be many input files for one test case. Above scenario clearly shows the Figure 5-2.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <mapping>
  - <modules>
    - <module>
        <moduleId>mtravel</moduleId>
        <active>true</active>
      - <testcases>
        - <testcase>
            <caseId>mtravel_001</caseId>
            <active>true</active>
            <type>regression</type>
            <inputs />
            <description>Search Travel Bills</description>
          </testcase>
        + <testcase>
        + <testcase>
        + <testcase>
        + <testcase>
        - <testcase>
            <caseId>mtravel_load_001</caseId>
            <active>true</active>
            <type>load</type>
          - <inputs>
              <input>mtravel_load_001_01</input>
            </inputs>
            <description>Save Definitive - Expense only travel</description>
          </testcase>
        </testcases>
      </module>
    + <module>
    + <module>
    </modules>
</mapping>
```


Figure 5-2 Mapping.xml

5.3 Test Plugin Implementation

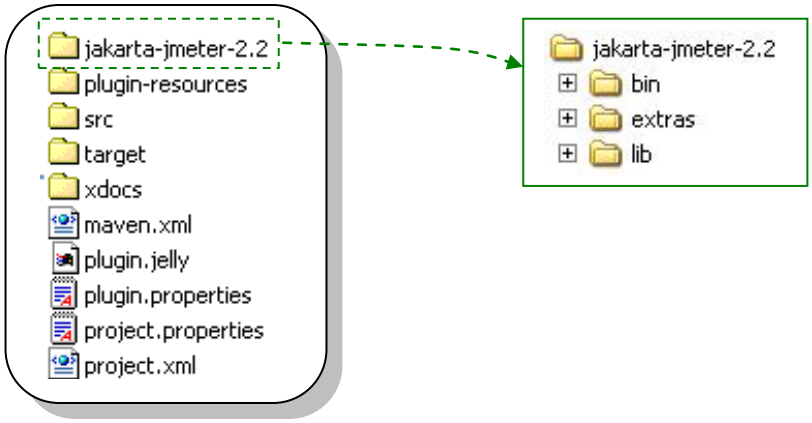

Figure 5-3 Test Plugin Implementation

The most important content of this plugin implementation is JMeter source. It contains in the "Jakarta-jmeter-2.2" folder. Test automation framework use JMeter source to execute jmx files in "Non-GUI" mode and write results into an xml file.

This maven Test Plugin has few maven-goals to do relevant processes of Test Automation Framework. All these things are coded in the "plugin.jelly" file. A scripting language called "jelly scripts" used to implement this maven plugin.

"plugin.jelly" code sample

```
<project xmlns:j="jelly:core"
        xmlns:license="license"
        xmlns:util="jelly:util"
        xmlns:ant="jelly:ant"
        xmlns:artifact="artifact"
        xmlns:maven="jelly:maven"
        xmlns:interaction="jelly:interaction"
        xmlns:velocity="jelly:velocity"
        xmlns:caller="caller"
        xmlns:doc="doc">


   <!--
=============================================================================
==================-->
```

```
   <!-- Clear tmp dir for the test env -->
   <!--
=============================================================================
==================-->
   <goal name="mecp:test-clean">
      <j:set var="snapshotBuild" value="${mecp.test.usesnapshot}"/>
      <!-- Create temp directory to store prepared test plans and results -->
      <ant:delete dir="${mecp.test.outputdir}/results" failonerror="false"/>
      <ant:delete dir="${mecp.test.outputdir}/classes" failonerror="false"/>
      <ant:delete dir="${mecp.test.outputdir}/tmp" failonerror="false"/>
      <j:if test="${snapshotBuild == 'true'}">
         <ant:delete dir="${maven.multiproject.basedir}/mecp-test/target/tmp"
failonerror="false"/>
      </j:if>
      <ant:delete dir="${maven.jboss.deploy.dir}/testresults.war" failonerror="false"/>
   </goal>
```

5.4 Summery

This chapter discusses on project implementation part. It discusses the four separate modules in details and hardware and software implementations of the project.