System Level Test Automation Design and User Guide

The system level test automation is a self-developed java application that executes the application with file input or command line input (need to be pre-written in text files than type in at run time) and checks the application output with criteria to decide if the application has correctly performed the tasks.

1. File Structure:

System test automation stores files in six folders:

- criteria: criteria files
- data: data input file, including properties files as application input, text files to store command line input, and a testSuite.properties file to define the automation test suite.
- report: stores test results and create test report.
- result: stores the application output
- src: the source files of the test automation
- testItem: the version of application under testing, a jar file

To make sure the file structure is correct, it is suggested check out all the SystemTest folder from repository.

2. Test automation design:

The system test automation contains 8 classes and 1 ready-made class contributed by other people. The 9 classes are:

- TestSuite.java
- TestMainipulator.java
- TestCase.java
- TestCaseFileInput.java
- TestCaseCommandLine.java
- TestComparator.java
- TestResult.java
- TestReport.java

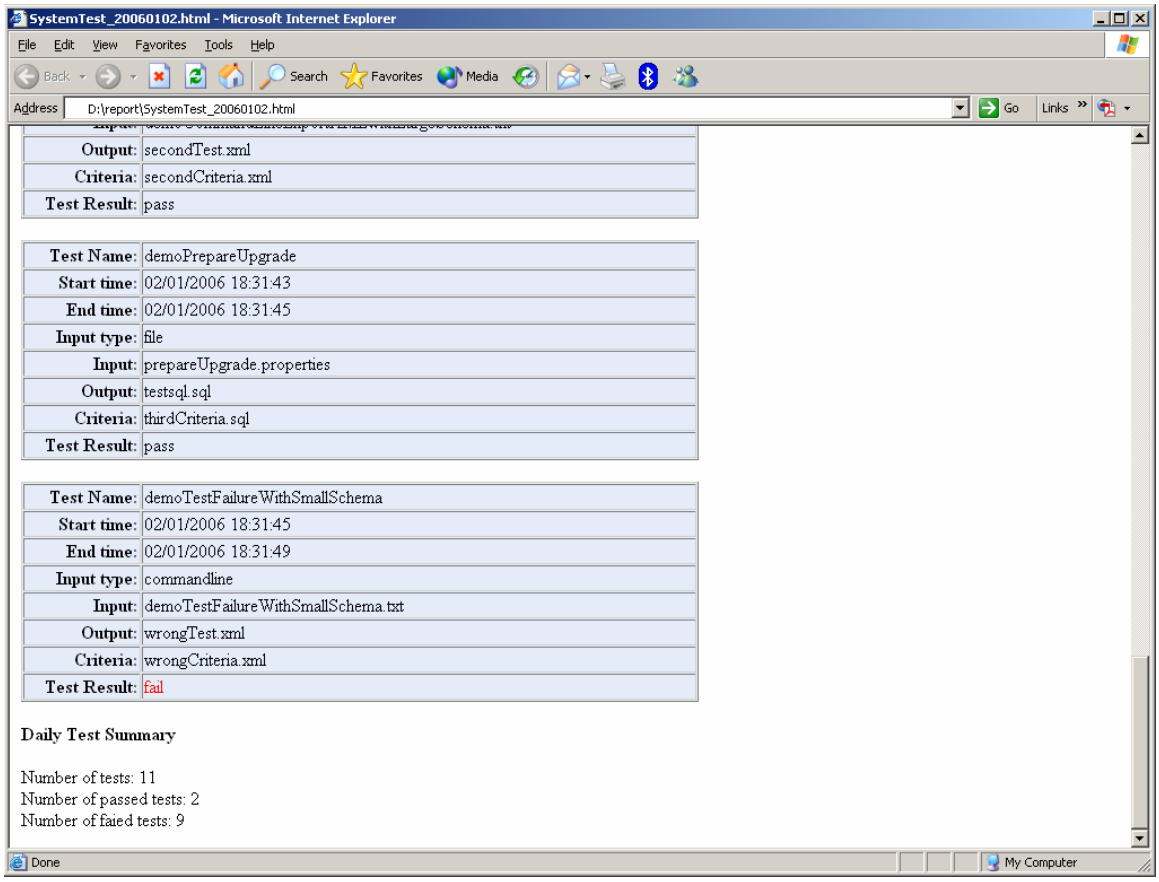
There names are self-descriptive. Among them, TestCaseFileInput and TestCaseCommandLine are sub-classes of TestCase.

The sequence chart is available from separate file.

3. How to use test automation

The automation setup and execution require following steps:

- Copy the item to be tested into testItem folder. Use name dbupgrader.jar
- Add all the test cases in testSuite.properties file. First, add the test case name to the test list, separating them by comma, then add properties of type, input, output and criteria for each test.
- For each test case, put the input into data folder, using the file name defined in the testSuite.properties file. Put the corresponding criteria file into criteria folder, also using the file name defined in testSuite.properties.
- The output of the application is stored in result folder, the names defined in testSuite.properties must be same with in application input. Before running the test, make sure there is not file under result folder with same names. This limitation is because the software under testing has its own way of handling output file names and test automation has to comply with it.
- Go to src folder, compile all files by javac *.java, run the test by calling java TestSuite.
- During test running, it first prints all the test cases it parses from properties file to console, and displays a simple log to console. After test execution, date are stored into dat file and a report in html is created and displayed, as follows:



4. Improvement plan

There are some improvements on the test automation under consideration. They are mainly:

- The test automation can be built into a jar file.
- Several items to be tested in one test execution, which can compare the quality of different software builds.
- More intelligent setup of TestSuite.properties file.
- Interactive way of testing command line input.
- Better error handling. Validation before running the tests.

These are not implemented yet because the test automation is supposed to be used by internal team, not distributed to customer. Error handling can be complex but we think careful usage of the software is more efficient than implementing too much error controls. However, these improvements can be taken into consideration in future.