Case Study on Automation testing

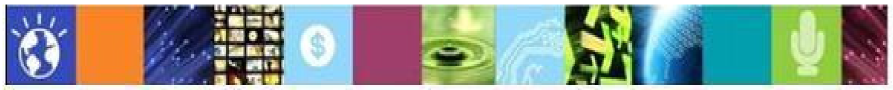
Overview

The Enervista suite of applications were software interfaces for electrical devices manufactured by the company and was basically about setting different set of values, controlling the parameters, and communicating with the device, In short these applications were graphical user interfaces to the electrical devices. The software has therefore numerous fields which had to be set and different combinations of actions that need to be performed on the application. This basically involved a large testing cycle considering the frequency of releases for the application, each application in the suite had 2-3 releases every year amounting to a large testing cycle, automation of the applications seemed a very good choice.

Objectives & Challenges

The modular approach of automation seemed very out of place here taking into account the multiple applications involved as this might require generating test scripts for each applications which would result in huge effort, less generic and less reusable scripts, and of course more maintenance over the life cycle. The feasibility study revealed that the application had lots of third party controls which were not recognized by the automation tool, neither was any automation support built into the application that would allow the tool to communicate to these third party controls.

The biggest challenge was automating the tree view and the grids that were not recognized by the automation tool. It was also important to generate very generic test scripts that could be used across different applications, this was required to eliminate any chances of redundancy in the framework. The most important requirement for the framework was to have an easy-to-use interface and a highly informative report to be generated at the end of test execution, thus allowing those who do not have exposure to the test automation tool to use the framework and interpret the results of the test execution.



Our Solution

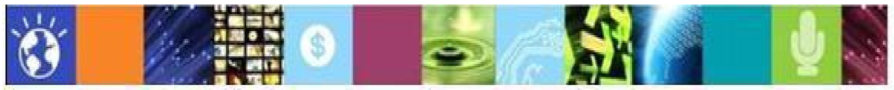
The feasibility study made us to think out of the box to automate the application and we decided to employ QuEdge's proprietary Optical Character Recognition (OCR) mechanism and the Sub-Image matching to facilitate the automation process. We tested the OCR and the Sub-Image match for effectiveness, concreteness, and reliability and was found to cater for automating most of the sections of the application.

We decided to use excel as the interface to the framework that would be built in SilkTest and to design a website that would display the results of the test execution even when the test execution is in progress.

Implementation

We designed a keyword Driven Framework in SilkTest that had an excel interface to administer test case management, test execution and test distribution. Keyword Driven Framework was used as the keywords could be reused across applications and in spite of the initial effort, and the effort in plugging in new projects in to the framework would be very less. The framework is also designed to work across multiple operating systems and over previous versions of SilkTest.

Keywords were developed that would work on multiple applications that share the same functionality and all the keywords built into the framework work for all the applications for which the functionality is applicable. A group of logical keywords make a test case and since these keywords are incorporated into the excel interface, thus allowing the end user to create or manage the test cases.



Benefits of the Framework

1. Excel Interface for test case management and execution: The test cases are built and executed from an Excel interface enabling it easy to use and intuitive even for resources without the knowledge of the automation tool used.

2. Selective test execution: Test cases can be executed as a batch and can be executed based on the attributes (for e.g. Category. Priority), and not to mention the ability to execute a specific test case.

3. Multiple machine execution: The test cases can be executed on a target machine, or individual test cases can be assigned to different machines.

4. Reporting Excel & Sql Database: The framework allows logging the results either in excel file or to an Sql database. This results are descriptive enough to give details on the actions performed in each test case.

4. Web based Reporting: The framework comes with Web based reporting wherein the website reads the data from the excel results or SQL database. The results can be viewed across the network through the network and options are provided to administer the results database through the website. Together with the description and status of the actions performed screenshots are associated with each step to give show the state of the application under test when an action is performed.

5. Auto Build Pickup via Microsoft Outlook: The tests can be launched via email with option of picking the build from either an FTP site or a shared drive. When the email is received, the build is picked up to update the installation and then execute the test case without any manual intervention.

6. Automation supported across three operating systems: The automation framework that works across 12 different applications equally works across three operating systems: Windows 2k, Window Xp, Vista.

The automation effort was planned for 6 milestones including one Milestone for Knowledge Transfer. The out-of-the-box approached we embraced on this effort allowed us to verify graphs and waveforms which would have impossible with the automation tool alone. The automation effort has covered areas that would have difficult to test manually considering the different combinations that are possible and thereby ensuring at least 50% lesser testing cycle per release of the application.