Test Automation

Implementing the Keyword Driven Framework

CLIENT OVERVIEW

The client is a pioneer in online options trading via the internet. The web based site helps in educating customers about options trading, evaluating options using their state-of-the-art proprietary tools, and executing their trading decisions quickly and accurately at a reasonable price.

BUSINESS NEED

The web site is constantly changing due to new enhancements, break/fix and regular maintenance releases. Time to market is critical in this business space which adds pressure on the IT staff to deliver quality releases quickly and efficiently. The company operates various modified web sites to execute business from multiple countries, which add to the complexity of the system and environment. The quality of the system is critical as this could impact millions of dollars in transactions during a given day. A known fact is that 100% test coverage is not possible. The business need was to develop a strong regression based automation strategy that would mitigate the risk of deploying releases to the production environments.

CHALLENGE and REQUIREMENTS

The client had an existing automation effort underway that wasn't being utilized due to the maintenance heavy burden on the team. The main challenge was to two fold, to develop the keyword driven framework using WinRunner and then to port their existing automated scripts, which covered many functional areas of the web site, to the new keyword driven framework. This was to be accomplished with one consultant and 2.5 month duration.

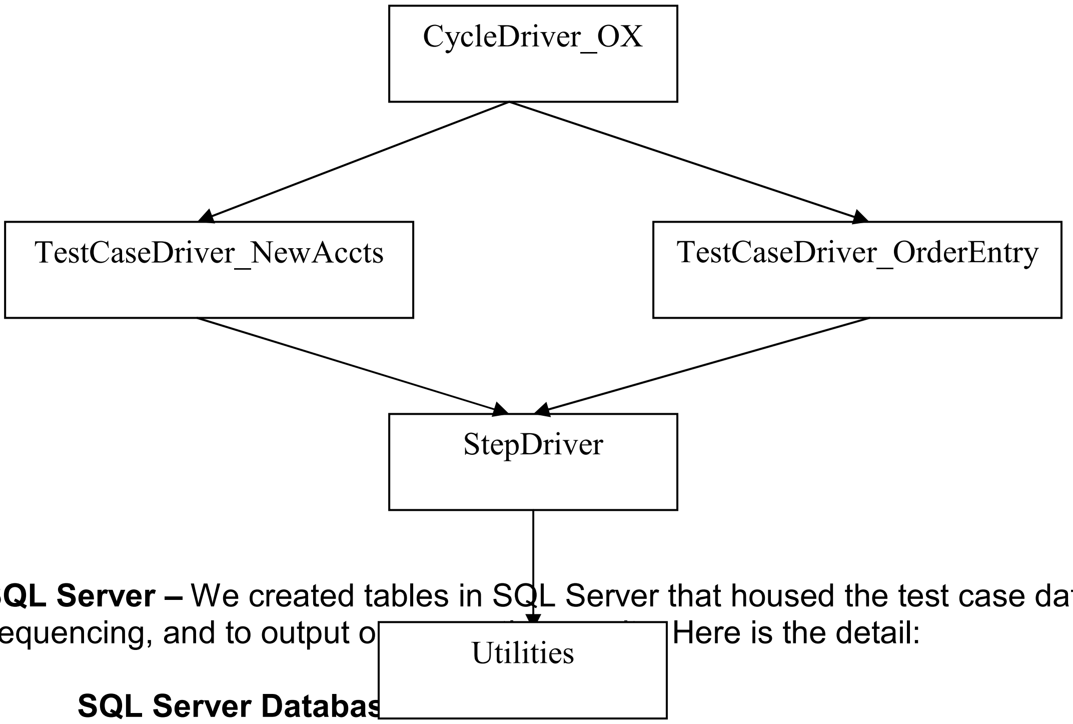
The requirements were laid out as follows:

- DSR will be able to utilize existing code from other client projects to get the Client project complete on time and within budget.
- DSR will be utilizing code / GUI maps / data schema (and data scripts) that have already been created by the Client when building the new keyword driven framework.
- The framework will be built to utilize Mercury WinRunner (version 7.5 or higher).
- The Client will not need to purchase any additional software.
- DSR will train the Client QA Automation employee in the new framework and this person will assist DSR in creating the framework.
- The Client will be involved in and have input on the design of the Keyword Driven framework.

- At the completion there will not be an ongoing licensing fee charged to the Client for use of the framework.

SOLUTION

The tools used were WinRunner for the automated code, excel for the test cases, and SQL Server DB's for the test case data. This solution comprised of the following components



SQL Server – We created tables in SQL Server that housed the test case data, test case sequencing, and to output o Here is the detail:

SQL Server Databas
In the 39DB1 database there's a table called QA_Automation. This table drives the automation framework, the individual test cases and the reporting of results. The following is a list of the tables and their function.

New_Account_Test_Cycle_Driver-Controls what group of tests to be executed. The 'Flag' needs to be set to 'Y' and the 'UserName' needs to be the person logging into 'bird' or 'eqautomation1' in order for the tests to be run properly.

New_Account_Test_Case_Driver –For each 'GroupID' in cycle driver there are number of corresponding spreadsheets in this table. These spreadsheets need to be in the order of execution using the 'TestCaseSEQ' column.

New_Account_Test_Case_Data –This table contains the test input data to the new accounts module.

QA_Test_Cases_Input-This table contains the test input data to the order entry and xml.api modules.

QA_Login_Info –Contains data related to order entry and xml.api

QA_Failed_TestCases-This is where all the results will be stored for new accounts,order entry and xml.api for any 'failed' tests only.

WinRunner Utility or Action Scripts -

Utility Scripts

Within each utility script there are several "Actions" a user can choose from to execute on a particular component or object. To see a description of an action, choose an action and mouse over the comment. The following are the most common utility scripts that are being used in the Client application today.

window-The window utility's main purpose is to set focus on a particular window. Common utility/action combinations:

window.Set: Specifies the window to receive specified input.
window.Exists: Verifies a window exists.
window.FindText: Finds specified text on a webpage.

button-This utility script performs actions on button objects. Common utility/action combinations:

button.Press: Presses on a button.
button.Set: Checks a radio button or check button to ON.
button.Enabled: Checks if a button is enabled.
button.Disabled: Checks if a button is disabled.

list-This utility performs actions on list objects. Common utility/action combinations:

list.Select: Selects a list item from a list box.
list.GetVal: Retrieves a value from a list box.

edit-This utility performs actions on edit/text objects. Common utility/action combinations:

edit.Set: Sets the value of a text field.
edit.GetVal: Retrieves the value from a text box.

object-This utility performs actions on objects. Common utility/action combinations:

object.ClickImage: Clicks on an image

noobject-This utility script has the most actions associated with it. The purpose of this noobject script is to compile multiple steps into one single step. Common utility/action combinations:

noobject.WebInvoke: Invokes IE and navigates to the specified URL.

noobject.DBCheck: Goes out to the application database and compares the values against the test data from the qa_automation database.

“Special” Actions
Within a given spreadsheet (TraderBackGroundInfo.xls is a good example to look at) one may come across ‘Validate’ and ‘SkipRow’ actions. The Validate action validates edit/text fields for invalid input (null field, special characters etc.) pressing ‘save and continue’ between the invalid inputs and looking at what the expected message should be off the popup message window. The SkipRow action can skip a row or multiple of rows depending on what the test case input data is within a given spreadsheet.

Validate-Validates an edit/text field with negative test data and compares the expected message with the actual message from the popup window.

SkipRow-Can skip a row or multiple of rows within a given spreadsheet depending on the test case data.

WinRunner Batch Scripts or Drivers – We had a base set of drivers from our previous projects to start with. Here is a complete description the drivers created for the client.

CycleDriver
This is considered the “main” driver where the execution of automated tests takes place first. Executing this script will call the other driver scripts mentioned below. This script also reads the new_account_test_cycle_driver table to see what appropriate tests need to run via the ‘GroupID,’ ‘Flag,’ and ‘UserName’ fields.

TestCaseDriver
The CycleDriver calls this script to see what data to execute with (new_account_test_case_data) and also, to see the order of execution of the spreadsheets ordered by the ‘TestCaseSeq’ in the new_account_test_case_driver table.

TestCaseDriver
This driver is exactly the same as the test case driver for new accounts, but the only difference is the table it points to for entering orders. The CycleDriver calls this script to see what data to execute with (qa_test_cases_input) and also, to see the order of execution of the spreadsheets ordered by the ‘TestCaseSeq’ in the new_account_test_case_driver table. The “GroupID” in qa_test_cases_input is represented by the ‘test_case_descr’ field.

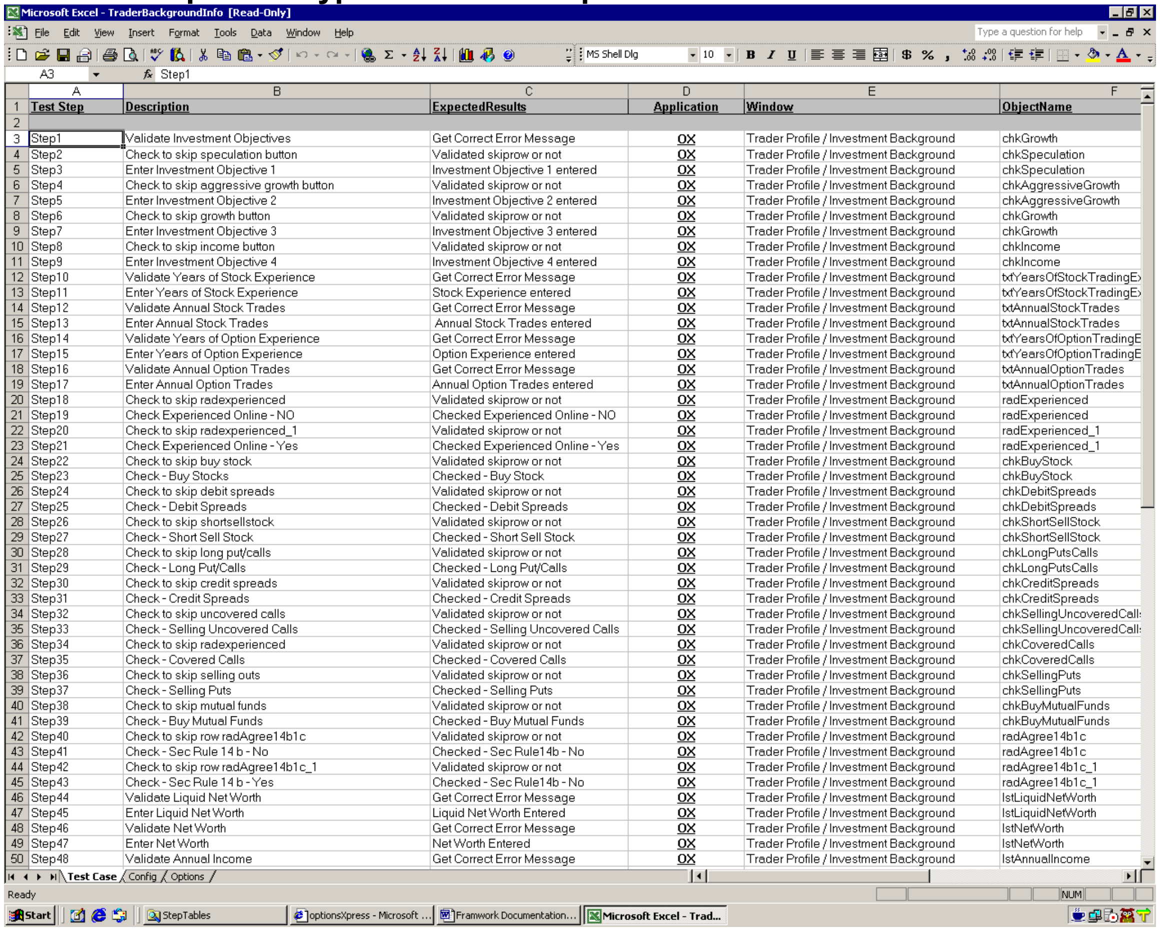
StepDriver
Both test case drivers call this script to see what “Utility” to execute. In a particular test spreadsheet there is a column called ‘Component.’ This component will call the appropriate utility script to execute for that particular step. Therefore, if that component

field is populated with the keyword “object,” the object utility script will be called and executed. The Spreadsheet section below goes into more detail on the different columns and their purposes.

Excel Spreadsheet

The test case spreadsheet is what drives the whole keyword framework. It serves two major purposes. One, it documents all the test cases step by step. If a new hire comes in, they can pretty much follow the test case verbatim with little direction from a senior analyst. Second, the automated test is created at the same time. An automation engineer will no longer have to create and code an automated test because the test case spreadsheet takes care of that already. Within a spreadsheet, the ‘Test Case’ worksheet is where all the time will be spent. The other two worksheets have to do with maintenance the automation engineer will use only. Details on the other two worksheets will be addressed later in the document. After creating a test case spreadsheet, it is very important to save it on the ‘Test Case’ worksheet. If not, WinRunner will not recognize what is being executed and the test run will fail.

Example of a typical test case spreadsheet

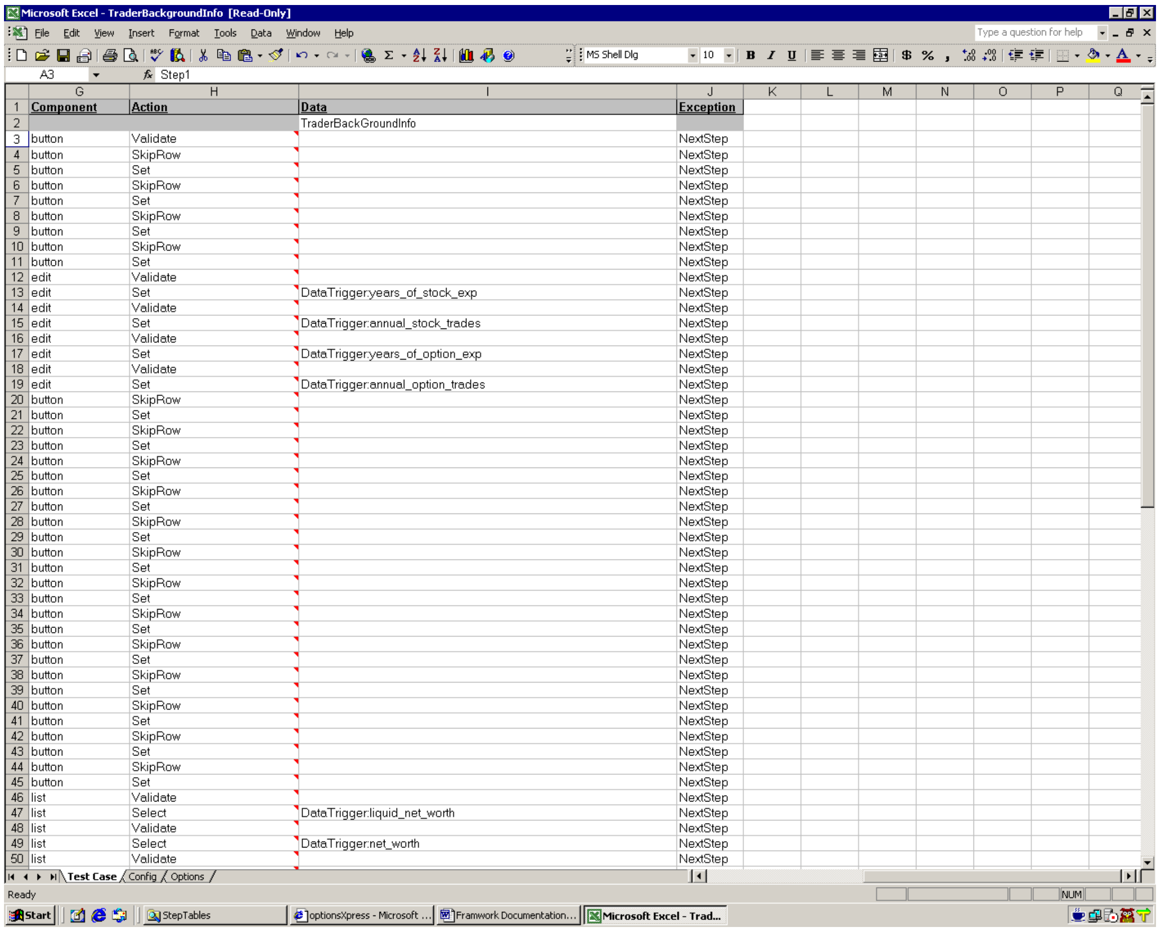

Microsoft Excel - TraderBackgroundInfo [Read-Only]

A3 Step1

	A	B	C	D	E	F
1	Test Step	Description	ExpectedResults	Application	Window	ObjectName
2						
3	Step1	Validate Investment Objectives	Get Correct Error Message	OX	Trader Profile / Investment Background	chkGrowth
4	Step2	Check to skip speculation button	Validated skiprow or not	OX	Trader Profile / Investment Background	chkSpeculation
5	Step3	Enter Investment Objective 1	Investment Objective 1 entered	OX	Trader Profile / Investment Background	chkSpeculation
6	Step4	Check to skip aggressive growth button	Validated skiprow or not	OX	Trader Profile / Investment Background	chkAggressiveGrowth
7	Step5	Enter Investment Objective 2	Investment Objective 2 entered	OX	Trader Profile / Investment Background	chkAggressiveGrowth
8	Step6	Check to skip growth button	Validated skiprow or not	OX	Trader Profile / Investment Background	chkGrowth
9	Step7	Enter Investment Objective 3	Investment Objective 3 entered	OX	Trader Profile / Investment Background	chkGrowth
10	Step8	Check to skip income button	Validated skiprow or not	OX	Trader Profile / Investment Background	chkIncome
11	Step9	Enter Investment Objective 4	Investment Objective 4 entered	OX	Trader Profile / Investment Background	chkIncome
12	Step10	Validate Years of Stock Experience	Get Correct Error Message	OX	Trader Profile / Investment Background	txtYearsOfStockTradingE
13	Step11	Enter Years of Stock Experience	Stock Experience entered	OX	Trader Profile / Investment Background	txtYearsOfStockTradingE
14	Step12	Validate Annual Stock Trades	Get Correct Error Message	OX	Trader Profile / Investment Background	txtAnnualStockTrades
15	Step13	Enter Annual Stock Trades	Annual Stock Trades entered	OX	Trader Profile / Investment Background	txtAnnualStockTrades
16	Step14	Validate Years of Option Experience	Get Correct Error Message	OX	Trader Profile / Investment Background	txtYearsOfOptionTradingE
17	Step15	Enter Years of Option Experience	Option Experience entered	OX	Trader Profile / Investment Background	txtYearsOfOptionTradingE
18	Step16	Validate Annual Option Trades	Get Correct Error Message	OX	Trader Profile / Investment Background	txtAnnualOptionTrades
19	Step17	Enter Annual Option Trades	Annual Option Trades entered	OX	Trader Profile / Investment Background	txtAnnualOptionTrades
20	Step18	Check to skip radexperienced	Validated skiprow or not	OX	Trader Profile / Investment Background	radExperienced
21	Step19	Check Experienced Online - NO	Checked Experienced Online - NO	OX	Trader Profile / Investment Background	radExperienced
22	Step20	Check to skip radexperienced_1	Validated skiprow or not	OX	Trader Profile / Investment Background	radExperienced_1
23	Step21	Check Experienced Online - Yes	Checked Experienced Online - Yes	OX	Trader Profile / Investment Background	radExperienced_1
24	Step22	Check to skip buy stock	Validated skiprow or not	OX	Trader Profile / Investment Background	chkBuyStock
25	Step23	Check - Buy Stocks	Checked - Buy Stock	OX	Trader Profile / Investment Background	chkBuyStock
26	Step24	Check to skip debit spreads	Validated skiprow or not	OX	Trader Profile / Investment Background	chkDebitSpreads
27	Step25	Check - Debit Spreads	Checked - Debit Spreads	OX	Trader Profile / Investment Background	chkDebitSpreads
28	Step26	Check to skip shortsellstock	Validated skiprow or not	OX	Trader Profile / Investment Background	chkShortSellStock
29	Step27	Check - Short Sell Stock	Checked - Short Sell Stock	OX	Trader Profile / Investment Background	chkShortSellStock
30	Step28	Check to skip long put/calls	Validated skiprow or not	OX	Trader Profile / Investment Background	chkLongPutsCalls
31	Step29	Check - Long Put/Calls	Checked - Long Put/Calls	OX	Trader Profile / Investment Background	chkLongPutsCalls
32	Step30	Check to skip credit spreads	Validated skiprow or not	OX	Trader Profile / Investment Background	chkCreditSpreads
33	Step31	Check - Credit Spreads	Checked - Credit Spreads	OX	Trader Profile / Investment Background	chkCreditSpreads
34	Step32	Check to skip uncovered calls	Validated skiprow or not	OX	Trader Profile / Investment Background	chkSellingUncoveredCall
35	Step33	Check - Selling Uncovered Calls	Checked - Selling Uncovered Calls	OX	Trader Profile / Investment Background	chkSellingUncoveredCall
36	Step34	Check to skip radexperienced	Validated skiprow or not	OX	Trader Profile / Investment Background	chkCoveredCalls
37	Step35	Check - Covered Calls	Checked - Covered Calls	OX	Trader Profile / Investment Background	chkCoveredCalls
38	Step36	Check to skip selling outs	Validated skiprow or not	OX	Trader Profile / Investment Background	chkSellingPuts
39	Step37	Check - Selling Puts	Checked - Selling Puts	OX	Trader Profile / Investment Background	chkSellingPuts
40	Step38	Check to skip mutual funds	Validated skiprow or not	OX	Trader Profile / Investment Background	chkBuyMutualFunds
41	Step39	Check - Buy Mutual Funds	Checked - Buy Mutual Funds	OX	Trader Profile / Investment Background	chkBuyMutualFunds
42	Step40	Check to skip row radAgree14b1c	Validated skiprow or not	OX	Trader Profile / Investment Background	radAgree14b1c
43	Step41	Check - Sec Rule 14 b - No	Checked - Sec Rule14b - No	OX	Trader Profile / Investment Background	radAgree14b1c
44	Step42	Check to skip row radAgree14b1c_1	Validated skiprow or not	OX	Trader Profile / Investment Background	radAgree14b1c_1
45	Step43	Check - Sec Rule 14 b - Yes	Checked - Sec Rule14b - No	OX	Trader Profile / Investment Background	radAgree14b1c_1
46	Step44	Validate Liquid Net Worth	Get Correct Error Message	OX	Trader Profile / Investment Background	lstLiquidNetWorth
47	Step45	Enter Liquid Net Worth	Liquid Net Worth Entered	OX	Trader Profile / Investment Background	lstLiquidNetWorth
48	Step46	Validate Net Worth	Get Correct Error Message	OX	Trader Profile / Investment Background	lstNetWorth
49	Step47	Enter Net Worth	Net Worth Entered	OX	Trader Profile / Investment Background	lstNetWorth
50	Step48	Validate Annual Income	Get Correct Error Message	OX	Trader Profile / Investment Background	lstAnnualIncome

Test Case
Config
Options
Ready
NUM
Start
StepTables
optionsXpress - Microsoft ...
Framwork Documentation...
Microsoft Excel - Trad...

Figure 1: Columns 1-6 of a Typical Test Case


Microsoft Excel - TraderBackgroundInfo [Read-Only]
File Edit View Insert Format Tools Data Window Help
Type a question for help
MS Shell Dlg
10
A3
Step1
Test Case
Config
Options
Ready
NUM
Start
StepTables
optionsXpress - Microsoft ...
Framwork Documentation...
Microsoft Excel - Trad...

	G	H	I	J
1	Component	Action	Data	Exception
2			TraderBackGroundInfo	
3	button	Validate		NextStep
4	button	SkipRow		NextStep
5	button	Set		NextStep
6	button	SkipRow		NextStep
7	button	Set		NextStep
8	button	SkipRow		NextStep
9	button	Set		NextStep
10	button	SkipRow		NextStep
11	button	Set		NextStep
12	edit	Validate		NextStep
13	edit	Set	DataTrigger:years_of_stock_exp	NextStep
14	edit	Validate		NextStep
15	edit	Set	DataTrigger:annual_stock_trades	NextStep
16	edit	Validate		NextStep
17	edit	Set	DataTrigger:years_of_option_exp	NextStep
18	edit	Validate		NextStep
19	edit	Set	DataTrigger:annual_option_trades	NextStep
20	button	SkipRow		NextStep
21	button	Set		NextStep
22	button	SkipRow		NextStep
23	button	Set		NextStep
24	button	SkipRow		NextStep
25	button	Set		NextStep
26	button	SkipRow		NextStep
27	button	Set		NextStep
28	button	SkipRow		NextStep
29	button	Set		NextStep
30	button	SkipRow		NextStep
31	button	Set		NextStep
32	button	SkipRow		NextStep
33	button	Set		NextStep
34	button	SkipRow		NextStep
35	button	Set		NextStep
36	button	SkipRow		NextStep
37	button	Set		NextStep
38	button	SkipRow		NextStep
39	button	Set		NextStep
40	button	SkipRow		NextStep
41	button	Set		NextStep
42	button	SkipRow		NextStep
43	button	Set		NextStep
44	button	SkipRow		NextStep
45	button	Set		NextStep
46	list	Validate		NextStep
47	list	Select	DataTrigger:liquid_net_worth	NextStep
48	list	Validate		NextStep
49	list	Select	DataTrigger:net_worth	NextStep
50	list	Validate		NextStep

Figure 2: Columns 7-10 of a Typical Test Case

From Figures 1 and 2, there are 10 columns total that are filled out. A macro is tied into these spreadsheets to give the appropriate selections via dropdown lists. Therefore, it is important that the rows of data be filled from the top down, left to right. Also, be sure that the first row of data starts at row 3. Below is a description of each column and their function.

Test Step-The test step number.

Description-The description of the step.

ExpectedResults-What the expected result is for that step.

Application-The application under test. It will always be 'OX.'

Window-All the window names associated with the 'OX' application. As new windows are created, please let the automation engineer know so he/she can upload them to the macro. If not, the window column will not be pre-populated.

ObjectName-The associated objects related to the window chosen.

Component-This calls the appropriate component script depending on the object chosen.

Action-Each component script comes with several actions to choose from. Mouse over the comment to see a description of what that action does.

Data-This is where an 'action' may need a parameter passed in for it to work properly. To distinguish between a database value versus a hard coded value, a database value will have the following naming convention 'DataTrigger:<column_name>.' For example, in the Order Entry module if price_amt_1 (QA_test_cases_input table) needs to be entered in an edit field, the data column would contain 'DataTrigger:price_amt_1.'

Exception-If a row fails for any reason there are 3 options that can be done:
Stop:Aborts the entire test run entirely.
NextStep:Continues on with the next step in the spreadsheet.
NextTest:Aborts the current test run and goes onto the next test.

Validations
The validations.xls table is not a test case spreadsheet, it's a lookup table to validate exceptions. In new accounts, it is used with the components: edit, button and list. Those components in combination, with the action: validate. Validations.xls columns include Window, ObjectName, Data and ExpResponse. This is located in the StepTables folder. In order to turn validations on be sure new_account_test_case_data.validate column has a value of 1.

TestCaseDataColumnNamesOrderEntry
This is a lookup table for all the database column names from the new_account_test_data table for the new account module. If any new database columns are added this spreadsheet needs to be updated. This is located in the StepTables folder.

TestCaseDataColumnNamesNewAccts
This is a lookup table for all the database column names from the qa_test_cases_input for the order entry and xml.api modules. If any new database columns are added this spreadsheet needs to be updated. This is located in the StepTables folder.

BENEFITS

- Substantially improved the Quality of the SLDC by allowing unattended automation on key functional areas of their web site.
- Development of scripts is faster. We were able to migrate all their existing automation to the new framework within 3 weeks.
- Provided the client the ability to hire only 1 automation engineer to maintain the framework as it was very maintenance friendly.

- Manual testers don't require tool expertise thus providing the client the ability to use the manual testers in the creation of automated test cases by using the front end excel spreadsheets.
- Allowed the client to increase test coverage while reducing overall maintenance.
- Provided the client with the ability to continually build and maintain automated test scripts in a cost effective manner.