

# Divide and conquer

Technical Consultant **Rupert Wills** of Compuware describes how more control of data-driven test automation can be gained by separating the object information from the test scripts



*The life of a Test Consultant* is a varied and interesting one. Last month it was a large bank looking for a complete end-to-end solution, this week it's a small software house experiencing difficulties with data migration. From Terminal emulators for mainframe to web, no one assignment is the same. The only common theme that runs across all environments is that of data driven testing. At one stage or another there will be a requirement to load data into a form. The classic method for this is via connection to the good old spreadsheet.

You would normally record the navigation to the form of interest, enter the data, save and then navigate back to the starting point. Connect the script to the spreadsheet, variant the recorded data and put it in a loop. Job done – or so I thought!

Whilst scoping a recent assignment I stumbled across a more exciting concept. The customer, a mature tester, with experience of test tools from most vendors, stated his requirements to me thus: "I only want two test scripts and I want them to do everything for me." I assumed that "everything" did not include making the tea, and asked him to elaborate.

The first script should take the form of a wizard, he said, and should allow him to set up and maintain projects. For each project the wizard should enable him to navigate through the SUT (Software Under Test), creating object data that is external to and independent from the test tool.

He then required a spreadsheet that, via connection to the data store, allowed him to pick an object and specify the action to perform against it. In its simplest form the spreadsheet would have only two columns: Object and Action. For example:

```
'EditBox UserName', "Rupert"  
'EditBox Password', "password"  
'Button OK', 'Click'
```

The second script simply reads the spreadsheet, locates the object and performs the action specified in each row. So what was the rationale behind his requirement for two test scripts?

He had been burnt before by a tool that stored object information within its test scripts. If one of the properties of an object changed it was a complete nightmare for him to search through all his scripts to correct the problem. Why should he generate and maintain both a large repository of scripts and a stack of spreadsheets? His test team consisted of manual testers. They were quite knowledgeable with spreadsheets and pivot tables, but had no previous experience with automation test tools.

The question was where to start? The first thing was to find the right tool for the job. To be able to meet the requirement of only two test scripts the items selected had to be capable of the following:

## Script 1

- active capture of the properties of objects as the mouse moves over them
- read/write connectivity to a database using ODBC allowing the tester to add, delete or amend records
- user forms agile enough to create the wizard required to guide the user through the first stage.

## Script 2

- able to read the data within the spreadsheet and act accordingly. Attach to the required control or window and perform the required action(s).

## Spreadsheet package

- allows you to write a macro to connect a column to a field within a database table and present those records in the form of a combination list.

## So how does it work?

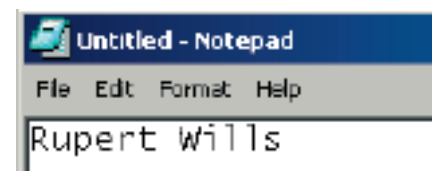
First, I will describe briefly how most Capture/Replay tools function.

Any automation tool worth its salt will, as you record a script, gather data on the objects presented by the SUT. What does this mean? Well for simplicity reasons we shall consider Notepad.

If you record opening Notepad and typing

in your name, your automation tool should capture the following:

- 1 clicking on the Start button
- 2 selection of the 'Run' command
- 3 typing 'Notepad' into the Run command combo box and clicking the 'OK' button
- 4 attachment or focusing of the Notepad window
- 5 typing 'Rupert Wills' into Notepad's editor window.



Your tool should now know all there is to know about the following:

- the tray window where the Start button lives
- the Run window where you enter your command into the combo box
- the Notepad window into which you typed.

On replay the tool will attach to the relevant windows or objects and perform your original actions. Attaching is achieved by finding the window or object by some of its properties. Those selected as significant (as below) will be used to identify the Notepad window correctly.

Significant	Property	Value
<input checked="" type="checkbox"/>	Application	NOTEPAD.EXE
<input checked="" type="checkbox"/>	Caption	Untitled - Notepad
<input type="checkbox"/>	ClassName	Notepad
<input type="checkbox"/>	ExtendedStyle	8H110
<input type="checkbox"/>	Height	537
<input type="checkbox"/>	Left	194
<input type="checkbox"/>	Style	8H14c0000
<input type="checkbox"/>	Top	259
<input checked="" type="checkbox"/>	TypeName	Window
<input type="checkbox"/>	Width	768

Here the tool will first look for an application called NOTEPAD.EXE. It must be a type

# Are your applications leaking money?



**64%**

of IT Executives say Yes...  
having experienced  
material revenue loss  
as a direct result of  
application failure\*

Compuware's ASQ solution can prevent application failure by building quality in from the start. To learn more about how we can help save you time and money simply visit

[www.compuware.co.uk/products/qa](http://www.compuware.co.uk/products/qa) and request your FREE Patricia Seybold Group White Paper - 'Your Applications are Leaking Money'.

Alternatively you can call us on 01753 444 444

In addition you can view our latest webinar 'Quality First, Strengthening the Quality Process for Successful Application Deployment' by visiting [www.compuware.co.uk/webinars/asq2](http://www.compuware.co.uk/webinars/asq2)

**COMPUWARE®**



[www.compuware.co.uk](http://www.compuware.co.uk)

'Window' and the window must have the caption 'Untitled - Notepad'. Using the 'Caption' rather than the XY co-ordinates (Left and Height) has obvious advantages.

### Generation of the Data Store

Some of you may be relieved to read that I am not going to cover the ins and outs of how to connect to a database via ODBC or how to generate the forms for the wizard. However, how do we grab the object data and what data do we put into the data store? We will address these two issues in reverse order.

### Data in the Data Store

Simply contains the object's 'Type', the unique property 'AttachProperties' that Script 2 will use to find it again, and 'DescriptiveName' the tester's meaningful or descriptive name.

ControlRef	Type	AttachProperties	DescriptiveName
1	Run	Notepad	Start Notepad.exe
2	Window	Caption=Untitled - Notepad	Untitled - Notepad Main Window
3	EditBox	Index=1	Untitled - Notepad Editor
[AutoNumber]			

### Object Data

As previously described, your tool should know the properties of the objects presented by the SUT. By returning the object that is under the mouse pointer as it moves across the screen we enable the tester to identify the object of interest. Once the object has been identified we pull out the properties that we (or rather Script 2) will need to correctly re-identify the object at replay.

At this stage we need to process each object by its type, ie

- a 'Window' can be identified by its 'Caption'.
- an 'EditBox' by its 'Name' or 'Label'.
- a 'RadioButton' by its 'Caption'.

For Notepad, we store:

```
'Type' = 'Window'
'AttachProperties' = 'Caption=
'Untitled - Notepad'
```

All that is required of the tester is to give the Notepad window a meaningful name ('DescriptiveName').

### The spreadsheet

Ideally when the Script 1 wizard creates a new project for the SUT it also creates a new spreadsheet. The automatic creation will include the macro required to connect a column of the spreadsheet to the data store for the project.

As we don't live in an ideal world, creation and connection may be a manual process. You won't want to do this every time so just create one example and copy it for each new project.

At this stage we can also introduce some test management. The first sheet of the workbook can contain the test plan, this plan will drive Script 2.

A	B
Project	
Task 1	Execute Task 1
Task 2	Execute Task 2
Task 1	Execute Task 1 again

Now Sheet 1 ('Plan') is our test plan or running order. It declares the project and the tests to be executed. Sheet 2 and Sheet 3 ('Task 1' and 'Task 2') are tests created by connection to the data store. Connection is made to the 'Project' table and then to the Field 'DescriptiveName' in the form of a combo box. Once connection is made, the tester simply selects the required object from the 'DescriptiveName' and completes the action.

Here again we have room for improvement.

	A	B	C
1	1	NotePad Editor	Rupert Wills
2	2	NotePad Editor	Rupert Wills(?)
3			

Command 1 will type 'Rupert Wills' into notepad's editor, where Command 2, with the addition of '(?)' will test to see if 'Rupert Wills' exists in notepad's editor.

Script 2 requires connection to both the Data Store and the spreadsheet.

The 'Plan' sheet is read to locate the correct project and running order. The Project is used to load all of the object data and 'DescriptiveNames'.

Script 1 matches the 'DescriptiveName' to the 'Objects Type and Properties'. The test tool then uses its replay engine to find and attach to the required object and then to perform the action stated in 'Task 1'.

Beyond attaching to and actioning objects and controls, you will need Script 2 to perform certain repeatable functions.

'Start Notepad.exe' is matched to 'Run' and 'Notepad'. The tool accesses the Start-Run Function and executes Notepad.

'Rupert Wills(?)' will be seen as a validation point. The '(?)' will tell Script 2 to locate the object and retrieve its value. The value is then compared to 'Rupert Wills' and Pass/Fail data recorded. The results can be customisable within Script 2.

### So what are the advantages of adopting such an approach to automation?

- There are only two scripts
- The data store is created easily through a wizard and is fully maintainable;
- Tests can be easily created by non-technical testers, and are easy to read;
- Validation points can be created easily;
- Test execution can be planned and controlled;
- If your requirements tool has a database, the 'spreadsheet' and 'Script 2' could be expanded to achieve requirements based testing.

PT

