

13

laInvokeArgs optional, STRING sPropListName optional, BOOLEAN bReturnDynamic optional)

The complete code for the function listed in the file entitled sqfuncs.inc, which is located on the Computer Program Listing Appendix disc. SQ_GetProperties takes six arguments. The first argument, wParent, is the identifier of the window for which to capture verification data. The second argument, sPropSetFile, is the name of the file that contains the names of the properties to capture for each class of object in the application under test. The third argument, laInvokeArgs, is a list of arguments to pass to the invoke method of the parent window. The fourth argument, sPropListName is the name of the property set within the file sPropSetFile to use for the current window being captured. The last argument, bReturnDynamic, is a BOOLEAN which specifies whether to capture properties for child objects of wParent that have not been declared.

SQ_GetProperties calls the invoke method of the window specified by the wParent argument. It accesses the property set stored in sPropListName in the file specified by sPropSetFile. This property set provides the list of properties to capture for each object class. An example of a property set file is shown in FIG. 14.

Using this property set, SQ_GetProperties would capture the caption of a BrowserChild, the caption of a WebPage, the state of a Checkbox, etc. For each child object of wParent, SQ_GetProperties captures the properties and stores them in a file where they can be copied by the user into the declaration of the window being tested.

The automated software test system also includes a smoke test package, which contains a specialized smoke test engine and functions that allow users to build smoke tests for client-server and web based applications without having to write any scripting code. The smoke test package can be implemented using the following prototype for the SQ_SmokeTest function:

```
testcase SQ_SmokeTest (WINDOW wWindowToVerify,
LIST OF ANYTYPE laArgs optional)
```

The complete code for the function listed in the file entitled sqfuncs.inc, which is located on the Computer Program Listing Appendix disc. SQ_SmokeTest takes two arguments. The first argument, wWindowToVerify, is the name of the window object to test. The second argument, laArgs, is a list of arguments to pass to the invoke method of the window object to test. The smoke test performs three tasks. It invokes the window to be tested by calling its invoke method (step 120), calls the result capture function (SQ_VerifyProperties—step 122) and then calls the window object's close method if it is defined (step 124). The smoke test can be called from a test plan, main function or any other test case driver where a list of windows to test can be specified.

To build a smoke test, the user provides only simple invoke and close methods and expected results, all of which can be recorded. The user begins by opening a smoke test template and locates a line that includes the words "List of Windows to be included in the smoketest". Under that heading, the user lists the names of the windows to be tested. The user then begins to create invoke methods using the SilkTest record declarations feature to capture the window declarations for each window to be included in the smoke test. He or she can then write or record an invoke method for each window. Some invoke methods require only a single 4Test statement. For example:

```
VOID Invoke ( ) MyApp.File.Open.Pick( ) return
```

Other invoke methods require data to be input to the application to create a context for invoking a window. To

14

complete this step, the smoke test package includes a function called SQ_InputFormData. It is provided so that no additional 4Test code needs to be written to create the application context. An example of an Invoke method using SQ_InputFormData follows.

```
LIST OF FIELD_DATA lfdSueSmith={. . . }{"Name",
" Susan Smith"}{"Address", {"10 Main Street", "NYC,
NY", "02233"}}{"AgeGroup", "34-39"}{"NewUser",
TRUE}{"Submit", NULL}
```

```
VOID Invoke ( ) MyApp.File.Open.Pick( ) SQ_Input-
FormData (this, lfdSueSmith) return
```

The user then creates close methods. Some windows will not require a close method because the built-in 4Test close method may suffice. If closing a window does require a custom close method, the user can use the same steps as described above to write or record one. The datasets used by the smoke test package for expected results are recorded using SilkTest's verify window.

The present invention has now been described in connection with a number of specific embodiments thereof. However, numerous modifications which are contemplated as falling within the scope of the present invention should now be apparent to those skilled in the art. It is therefore intended that the scope of the present invention be limited only by the scope of the claims appended hereto. In addition, the order of presentation of the claims should not be construed to limit the scope of any particular term in the claims.

What is claimed is:

1. An automated software testing method, comprising the steps of:

receiving a user-created stimulus data set including series of pairs that each include a user interface object identifier and a desired action identifier for that object, wherein the pairs are for testing user interface objects of a software application under test, identified by the identifiers in the data set,

automatically converting the data set received in the step of receiving into a series of test instructions for the user interface objects, and

wherein the set of test instructions is operative to be run against the user interface objects of the software application under test.

2. The method of claim 1 wherein the step of receiving a data set receives a stimulus data set that includes a series of pairs that each include a software object identifier and a desired input value for that object.

3. The method of claim 1 further including a step of automatically generating the data set based on a state of the application under test.

4. The method of claim 3 wherein the step of automatically generating generates a stimulus data set by acquiring information submitted to the application by a user.

5. The method of claim 4 further including a step of acquiring a response data set based on the interaction with the user.

6. The method of claim 3 wherein the step of automatically generating generates a response data set by detecting responses from a state of the application.

7. The method of claim 1 wherein the step of automatically converting determines methods to generate based on object class and data type.

8. The method of claim 1 further including a step of providing a data set template to the user to fill in before the step of receiving the data set.

9. The method of claim 1 further including a step of prescribing a four-phase model on a test that includes the