

TESTING SOFTWARE COMBINED WITH CONVENTIONAL AUTOMATED SOFTWARE QUALITY (ASQ) PRODUCTS

FIELD OF INVENTION

5 The invention generally relates to automated software quality and performance testing.

BACKGROUND OF THE INVENTION

10 Software testing is an important, expensive and time-consuming activity in the development life cycle of a software product. As defined in this disclosure, a test is a program executed to validate the behavior of another program. A test can include a set of one or more test cases. A test case refers to an individual test condition.

15 Test cases are broken down into one or more test scripts. Reuse refers to the ability to share libraries of common test cases among multiple tests. Automation refers to the removal of human interaction with the testing process and placing it under computer or program control.

20 In general, reuse of common artifacts can provide a significant gain in productivity for software testing. In addition, because software testing involves running a system under a variety of configurations and circumstances, automation of execution-related activities offers another potential source of savings in the testing process. Through reuse and automation, it is reasonable to expect reduction of time and/or resources (i.e., hardware and people) needed to perform the testing compared to testing the system manually.

25 Persons with detailed knowledge of the system can create manual test cases. In various operating environments, this person can be a system user, a business analyst, or a manual tester. Test cases can be passed to an automation engineer familiar with testing tools and scripting language(s). Automation engineers typically do not have detailed knowledge of the tested system. In real situations, automated test tools frequently become shelf-ware because of the cost of maintaining the automated scripts. Automated test tools
30 also become shelf-ware because of a lack of knowledge by the automation engineers required to modify existing scripts every time the system changes.

In view of the problems mentioned above, there is a need in the art for software with a simple interface in the form of a few key-words in which automated scripts can be developed and maintained by less-skilled testers or users of the system compared to automation engineers. Another need in the industry is for a simple interface between a user and a specific automation tool. By separating automation scripts from a specific automation tool, the intellectual value invested in creating and maintaining these scripts is preserved from system and technology changes.

SUMMARY OF THE INVENTION

The invention relates to a method and software program called Octane used for automation of web application testing process. The method involves creating test cases using a predefined set of keywords and data values describing desired actions to perform. As one option, test cases can contain data values representing expected results that can be compared with results generated by system.

After all test cases have been created and saved, anyone with proper access can create their own automation scripts by grouping available test cases into one or more tests. Several tests can be grouped together as a test group. A person can execute all tests and test groups by a push of button in an unattended mode. When all tests have been run, the results can be displayed and the automated test will be finished.

According to one exemplary and preferred embodiment, the inventive Octane software is built on the top of conventional automated quality and performance testing software or known in the industry as automated software quality software (“ASQ software”), such as the Mercury Interactive automation tool called WinRunner. WinRunner is used as an execution engine to manipulate web based objects that are part of the tested system. If WinRunner was replaced with another automation tool, the Octane software interface layer usually requires modifications to communicate with the new tool, however, there would be no need to modify existing automation scripts.

If an alternative ASQ software is used, the inventive Octane software may require some conversion such that its code shares a language similar to the code of the ASQ software. For example, if the Quick Test brand ASQ software is used, then the task

sequencing language (TSL) scripting language designed for WinRunner brand software would need conversion to visual basic.

Other parameters that vary between types of ASQ software include Gui files. Gui files may be specific to one particular ASQ software such as the WinRunner brand testing tool. Therefore, if another tool is used, then the method by which the new testing tool recognizes objects on a webpage would likely require modification.

In addition, there are general option settings that need modification to accommodate any new tool options for a particular ASQ software. Since the general option settings are read via a caller file, it may be necessary to modify information in this script. General option settings are usually feature specific to certain ASQ software applications such as the WinRunner brand software, but the settings defined in the inventive Octane's caller script can override the general option settings of the ASQ software.

The inventive Octane software can further comprise other files that are associated with the ASQ software: A library function script, a DBConnection script, a reporting script, a loglib script, and an errors script.

The libraryfunctions script can be accessed by a perform script. The perform script can use various functions in the libraryfunctions script. Basically, the libraryfunctions script comprises a module of functions. Examples of some functions include, but are not limited to, the following: a close browser function, a security alert function, and a random number generator function.

The close browser function is accessed when a user specifies a "yes" value in a browser column as will be discussed below in the detailed description. A security alert function can be called upon in a window case statement inside a perform script. When a window keyword is called, an ASQ software can look for an Internet Explorer security pop-up. If the window displays, then the "yes" button is automatically pressed.

The random number generator function can be called if a user specifies "yes" in a borroweruniqueId column of a group file. If "yes" is present, then a random nine-digit number can be generated. When a test is executed by the ASQ software, the nine-digit random number can be entered into one of a Social Security number, an email, a verify email, a verify password field, and a password field itself.

The DBConnection can be executed whenever a user specifies a database keyword in a test file. All the functions in this compiled module can make calls to the database and return a value. This return value can be passed back into a perform script.

5 DETAILED DESCRIPTION OF THE PREFERRED EMOIDIMENTS

Referring now to the drawings, in which like numerals represent like elements throughout the several figures, aspects of the present invention and the exemplary operating environment will be described. Referring now to Figure 1, this figure illustrates exemplary computer file directories in which portions of the inventive Octane software can reside.

The Octane software can comprise the following file directories: caller script directory 1; start script directory 2; perform script directory 3, a nav directory 4; a logs directory 5; a suites directory 6; a lib directory 7, and an obj directory 8.

The caller script directory 1 can comprise an initialize environment set-up script. The start script directory 2 can read the suite file information and the data to the perform script in the perform script directory 3.

The perform script directory 3 can read both group and test file information. All keywords reside in the script of the script directory 3. The nav directory 4 comprises all of the test and group files.

The logs directory 5 stores output after each execution by the ASQ software. The output is usually report information stored into a separate files with a date marked on each file. The suites directory 6 files comprises the suite files of the inventive software.

The lib directory 7 contains separate modules called upon by the perform script. Further details of the lib directory will be described below in connection with Figure 2.

The obj directory 8 contains the object files that the ASQ software uses to identify objects on a web page.

Referring now to Figure 2, this figure illustrates the contents of the lib directory 7. This directory 7 comprises the following exemplary sub-directories: a DBConnecton subdirectory 9, an errors subdirectory 10, a libraryfunctions subdirectory 11, a reporting subdirectory 12, and a loglib subdirectory 13.

The DBConnection subdirectory 9 can comprise all of the necessary database functions. The errors subdirectory 10 can contain all of the necessary functions to be used with the valid keywords. The libraryfunctions subdirectory 11 can contain all of the necessary functions to be used with the general keywords.

5 The reporting subdirectory 12 can contain all of the necessary functions to be used for reporting. The loglib subdirectory 13 can contain all of the necessary functions to be used for the log files.

10 Successfully running a test with the Octane software requires at least three files to be present. The three files follow a hierarchical set-up. This means that if one file is missing, the Octane software will not be able to execute the test.

The three files required for running a test are the following: files in the suite directory 6, group files 400 (see Figure 4) and test files 500 (see Figure 5). The files in the suite directory 6 call the group files 400 and the group files 400 call the test files 500. These files can be stored anywhere on a computer or network. It is necessary that the
15 correct location in the TestDir column 16 in the suite file 300 (see Figure 3) be provided to successfully run a test.

Referring now to Figure 3, this figure illustrates information that can be contained in a suite file 300 found in the suite file directory 6. A suite file 300 can be made from any spreadsheet type of software. According to one exemplary embodiment, Micorsoft
20 Excel brand spreadsheet software can be used to set up a Suite file directory 6. The following four columns are created in sequence as follows: group column 14, perform column 14, TestDir column 16, and ObjDir column 17.

The group column 14 comprises a name of a group file. Next, the perform column 14 can comprise either a “1” or “0”. A value of “1” causes the group file
25 identified in the group column 14 to be executed by the ASQ software. A value of “0” causes the ASQ software to skip this row in the suite file 300.

In the TestDir column 16, a test directory location can be entered. In one exemplary embodiment, a full path name of the test location can be included in this column. In the ObjDir column 17, an object directory location can be provided.
30 According to one exemplary embodiment, a full path name usually must be provided in

this column. The object directory can comprise a location where ASQ software GUI object files reside.

Referring now to Figure 4, this figure illustrates information that can be contained in a group file 400. A group file 400 can be made from any spreadsheet type of software.

5 According to one exemplary embodiment, Microsoft Excel brand spreadsheet software can be used to set up a group file 400.

The following thirteen columns are created in the following order: navigate column 18, perform column 19, URL column 20, GUI File column 21, closebrowser column 22, NewBrowser column 23, TestName column 24, TestDescription column 25,
10 ServerName column 26, PortNumber column 27, BorrowerUniqueId column 28, SecondServerName column 29, and SecondPortNumber column 30.

In the navigate column 18, a name of a test file without an extension can be entered. If there is a second test, its name can be entered into a second row in the navigate column 18. Additional test names can be added in subsequent rows if more tests exist.
15 The test names should be spelled correctly.

In the newbrowser column 23, a “Y” or “N” can be entered. A “yes” causes the ASQ software to open a newbrowser at the start of a testing process. Conversely, a newbrowser will not be opened if a “no” is entered in the newbrowser column 23.

In the testname column 24, a name of a test can be entered. According to one
20 exemplary embodiment, the testname column 24 can be an optional field.

In the testdescription column 25, a description of the test can be entered. According to one exemplary embodiment, this testdescription column 25 can be an optional field.

In the servername column 26, an IP address of a database may be entered if the
25 test relates to a database. In the portnumber column 27, if there is an IP address in the servername column 26, then a portnumber can be entered into this column.

In the borroweruniqueId column 28, a “Y” or “N” can be entered. A “yes” entry can generate a random Social Security number, email, and password value and enter the data into any fields that are being tested. According to one exemplary embodiment, this
30 borroweruniqueId column 28 can be used for borrower web testing. This borroweruniqueId column 28 can be left blank if a borrower website is not being tested.

In the secondservername column 29, the IP address of a second database can be entered if the test being executed is a database comparison type test. In the secondportnumber column 30, a portnumber can be entered if there is an IP address in the secondservername column 29.

5 Referring now to Figure 5, this figure illustrates a test file 500 according to one exemplary embodiment of the invention. The test file 500 can be made from any spreadsheet type of software. According to one exemplary embodiment, Microsoft Excel brand spreadsheet software can be used to create the test file 500. Four columns are created in this order: an action column 31, a description column 32, a data column 33,
10 and a testind column 34.

In the action column 31, a name of a keyword can be entered into this column. Further details about keywords are discussed below with reference to Figure 10.

In the description column 32, the object or window name can be entered. In the data column 33, data can be entered.

15 In the testind column 34, a value of "1" should be placed in this column if the step in a particular row validates a test case. In other words, when a value of "1" is entered into the testind column 34, the system will tally or keep track of tests that are ran against a particular object or window that is identified in the description column 32 for a particular row. The testind column 34 should be left blank (without any value) if tracking
20 of testing for a particular object or window that is identified in the description column 32 of a particular row is not desired. In Figure 5, all rows of the testind column 34 have been left blank and therefore, the tracking of testing or the results of tests for all objects or windows in the description column 32 will not occur.

Referring now to Figure 6, this figure illustrates an exemplary database test file
25 600. The database test file 600 can be made from any spreadsheet type of software. According to one exemplary embodiment, Microsoft Excel Brand spreadsheet software can be used to create a database test file 600. Eight columns are created in the following order: an action column 35, a description column 36, a data column 37, a testind column 38, a fieldtocompare1 column 39, a fieldtocompare2 column 40, a secondSQL column
30 41, and a perform column 42.

In the action column 35, a database keyword can be entered. In the description column 36, a description of the test can be entered into this column. In the data column 37, a firstSQL statement can be entered for execution by the ASQ software.

5 In the testind column 38, a “1” can be entered if a test case will be validated by the ASQ software. In the fieldtocompare1 column, a first database field name can be entered. In the fieldtocompare2 column 40, a second database field name can also be entered.

10 In the secondSQL column 41, a secondSQL statement to be executed by the ASQ software can be entered. In the perform column 42, a “1” or a “0” can be entered into this column. A value of “1” causes the ASQ software to execute the test while a “0” in this column 42 will cause the ASQ software to not run a test.

Running A Test

15 To start the inventive Octane software, according to one exemplary embodiment, an icon associated with the Octane software can be double clicked on a desktop (see Figure 11). Once this icon is double clicked, prompts will be provided to start the automated testing process.

20 Once testing is complete, a report summary displays indicating if a test has passed or failed. A hyperlink can be presented on each report allowing a user to drill down into a more detailed summary report.

Results for a Test

25 Referring now specifically to Figure 7, this figure illustrates a suite summary report 43 according to one exemplary embodiment of the invention. This report 43 can provide a user with a name of the suite tested and if the suite passed or failed. The suite summary report 43 can also provide a status column 710, a total column 715, a pass column 720, and a fail column 25.

30 The Name Column 703 of the Suite Report 43 can include names that are derived from the name of the group file entered in the Group Column 14 of the Suite File 300 illustrated in Figure 3. The Status Column 710 can comprise status based on all of the tests run in a particular group file. The status of “pass” can be given only if all of the

tests in the group file pass. The number of pass and fails are totaled up in the perform script and reported by passing the information to the reporting module that prints while the test is being executed by the ASQ Software.

5 The Total Column 715 can comprise a total number of test cases in a group file. This can include both pass and fail statuses. In each test file, there is a Testind Column 34. The user needs to place a value of “one” into this Testind Column 34 to be included in a total test case count.

10 A Pass Column 720 comprises a total number of test cases for each test in the group file having a status of pass. The Fail Column 725 comprises a number of test cases from the group file that have failed. If any of the test cases failed in the group then the status of the suite summary will usually be set to fail. The Detailed Log File 705 can comprise a path file location to the log files for this test suite. This information can be generated from the loglib script.

15 Referring now to Figure 8, this figure illustrates a group summary report 44 according to one exemplary embodiment of the present invention. This group summary report 44 can allow a user to view the name 805 of tests that have been executed and if a group test has passed or failed. Similar to the suite summary report 43, the group summary report 44 can comprise a status column 810, a total column 815, a pass column 820, and a fail column 825.

20 The Name Column 805 lists the name of the test file that is entered into the Group File 400 of Figure 4. Referring briefly back to Figure 4, each test file listed in the Navigate Column 18 in the Group File 400 has data printed to the report file regardless of whether a test is run or not for a particular test file. This information is generated from both the perform and reporting scripts.

25 Referring back to Figure 8, the Status Column 810 indicates if a test has passed, failed, or if the test has not been run. The status of “pass” is provided only if all the test steps pass for a particular test. The Total Column 815 indicates a number of test cases including pass or fail for a particular test file. In the Test File 500 of Figure 5, there is a Testind Column 34. A user would need to place a value of “one” into the Testind
30 Column 34 of Figure 5 for this test to be included in the total test case count.

Referring again back to Figure 8, the Pass Column 820 indicates a number of test taken from a test file having a status of pass. Similarly, the Fail Column 825 indicates the number of tests cases taken from a test file having a status of fail.

5 A link (not shown in Figure 8) to a detailed log file similar to the other summary reports can be provided in the Group Report 44. This link can allow the user to view the log file for each test run. All the information displayed in the Group Report 44 is independent of the ASQ Software and corresponds and is generated by the Inventive Octane Software. The number of pass and fails are totaled up in a perform script and reported by passing the information to a reporting module and printed out in a report
10 while a test run is being executed.

Referring now to Figure 9, this figure illustrates a test summary report 45 according to one exemplary embodiment of the invention. This test summary report 45 can provide a user with detailed information about each step that was executed for a test.

For instance, each keyword 905, description and data 910, as well as a data setup status 915 and a test status 920 can be provided. The test summary report 45 can further
15 comprise a hyperlink 46 to a log file for each report. The log file can provide a user with detailed information about a test. The summary report 45 can also provide a file name 47 that indicates where a user can find the summary report 45.

The Test Report 45 is calculated by perform 3 and reporting scripts 12 of the
20 Inventive Octane Software. A number of pass and fails can be totaled up in the perform script 3 and reported by passing the information to the reporting script 12 and printed out in the report at the end of the test.

The Test Name 925 of the Test Report 45 provides the name of the test from the Testname Column 24 in the Group File 400 of Figure 4. The Description Field 930 of the
25 Test Report 45 provides the information taken from the Testdescription Column 25 of the Group File 400 of Figure 4.

The Filename Field 47 corresponds to the test file name entered in the Navigate Column 18 of the Group File 400 of Figure 4. The Detailed Log File Field 46 comprises a path for a location of both the report and log files and can reside in a log directory.

The Keyword Action Column 905 corresponds to the keyword name executed in a test script. This information corresponds to the Action Column 31 of the Test File 500 in Figure 5.

5 The Data Column 910 comprises sequential query language (SQL) that is generated from the Data Column 33 of the Test File 500 of Figure 5. When a keyword from the Keyword Action Column is executed, results can be returned and evaluated. The results originated from the Performs Script 3 of Figure 1 and from the DBConnection Script 9 of Figure 2.

10 Referring back to Figure 9, the Test Status Data Column 920 comprises information that is derived from the value in the Testind Column 34 of the Test File 500 of Figure 5. If a "1" is entered into Testind Column 34 and if this step was successful, then a status of pass is entered into Test Status Column 920 of Figure 9. All of the detailed results described in the test report are generated by the inventive Octane Software.

15 Referring now to Figure 10, the keywords in listed in Table 1000 originate from the Perform Script 3 of Figure 1. Each keyword can reside inside a case statement. The Action Column 31 of the Test File 500 in Figure 5 can contain the keyword names that are listed in the Keyword Name Column 1005 in Figure 10. A keyword in the Action Column 31 of a Test File 500 in Figure 5 can call a corresponding keyword in a case statement that causes the ASQ Software to execute the desired action.

20

An Example Of A Test That Is Executed With The Inventive Octane Software

Caller Script 1

25 The caller script file 1 loads all of the directory locations for logging, objects, base path and environment settings. The values of these settings are stored in an initialization file. The initialization file can comprise a Microsoft Excel file stored in the main directory. The values in this file can be modified at any time without having to modify computer code for the inventive Octane software.

30 Once the caller script 1 reads in the initialization file, each of these values are stored into a variable for later use. Included in this information are the following: a user ID, a password, log level settings, and a base path location. The caller script file 1 then

passes the variables to the start script. In addition, the caller file 1 prompts the user for their name and suite file location before running the ASQ software.

Start Script 2

5 Once the variables are passed to the start script 2, the first suite file 300 is opened and a row count is performed. The first item to be evaluated is the perform column 15 of the suite file 300. The perform column 15 dictates if a test is executed or not by the ASQ software. If a value of “1” is entered into a perform column 15, then the data in the first row are stored into variables. These variables can comprise the group name from the
10 group column 14, the test directory column 16, and the object directory 17 of the suite file 300. These variables are then passed to the perform script 3.

Perform Script 3

15 Once the variables from the suite file 300 are passed to the perform script, the appropriate group file 400 based on the group name identified from column 14 of the suite file 300 list is opened from the test file location. A row count is performed on the group file 400 and the first row can be evaluated. The first variable to be evaluated is the perform column 19. If a “1” is present in column 19, then the other variable in the
20 columns are stored into corresponding variables. If a “0” is present, then the ASQ software evaluates the next row of the group file 400. For each column in the group file, a variable is provided to store a respective value.

25 After the values are stored, they are evaluated. For instance, if an IP address exists in the server name variable, then the Octane software connects to the database using the IP address and port number indicated in the group file 400 from the server name column 26 and the port number column 27.

30 If errors are not present in the group file 400, then the first test file in the list is opened based on the value retrieved from the navigate column 18 of the group file 400. The data stored in the perform variable is evaluated first. The data for the perform variable is taken from the perform column 19 of the group file 400. If the value is equal to “1”, then the other variables in the row are evaluated. If the value is equal to “0”, then the ASQ software ignores the row and continues to the next row of data in the suite 400.

The action variable from the action column 31 of the test file 500 is evaluated next to see what keyword to execute. The keywords can comprise a large case statement. Its selection can be based on the information provided in the action column 31 of the test file 500. Based on the keyword used, the description and data variables were other variables are evaluated inside the case statement.

In addition, external libraries are called from inside the case statement. For instance, if a database keyword is called, then the DBConnection library 9 can be called in order to execute database queries. If any of the verify keywords are used, then the error library 10 is called upon to look information up in the error tables.

At the end of the first test file, the testind values of the testind column 34 of the test file 500 are added up and written to report files. A pass/fail status is given on the outcome of the testind information. If any of the steps in the test fail, then the entire test is given a fail status. The test file can then close. If there is another test entered in the group file, then this file is opened and evaluated. The process can be repeated for each file listed in a group file 400 as illustrated in Figure 4.

Logging Information

If logging information is set to 100% in the initialization file, then all variable output from the start and perform scripts 3 are sent to the log files 5. This includes error messages, variable values, and keyword information. Each keyword in the perform script 3 has built in error checking. If the keyword being evaluated runs correctly, then the information is sent to the log files. If there is an error, then the errors are also written to the log file 5.

The perform script 3 can evaluate the return value with either a status of “pass” or “fail.”

The reporting script 12 can create all of the reporting information presented to the user at the completion of testing. This script 12 can be responsible for the look and feel of the reports.

The loglib script 13 can comprise the following functions: outputting a line number of the function, getting the current system date, creating a log directory if it is not present, opening a log file, closing a log file, returning the latest error log file and

pathname and converting the air code numbers to more informative text messages. The perform script 3 can use the loglib 13 script to provide the user with an end-depth description as to why a test may have failed.

5 The errors script 10 can comprise the following functions: finding errors on a page, comparing a literal string entered with text on the page, getting error messages text using the error numbers specified in the test file, and searching a database table for a field name specified in a test file. All the aforementioned functions return information back to the perform script 3 to be evaluated using a “pass” or “fail” status.

10 The ASQ Software provides an initial command line startup process. In addition, the ASQ Software executes or runs the inventive Octane scripts. The built-in task sequencing language (TSL) provided by the ASQ Software allows the capability to access a database as well as reading data from the test files discussed above. Each case statement in the perform script must access a GUI File in order to locate objects on the application being tested. The GUI File is an intermediate between the test files, the code,
15 and the application.

The ASQ Software can also provide a GUI map editor where all of the object files are housed. The GUI map can contain objects in which each has a physical and logical description. The logical description can be used when entering the object named into the test files.

20

Execution of a Test Suite

The following is a description of a step-by-step list of what happens when a test suite is executed by the ASQ Software. Referring now to Figure 11, an Octane Icon 1100 is illustrated. Clicking on this Octane Icon 1100 can startup the ASQ Software via a
25 command line startup, in which the caller script begins and starts the testing process. The ASQ Software can provide the start-up command line found in a Target Field 1205 of Figure 12. The ASQ Software can prompt a user for their name with a Dialogue Box 1300 as illustrated in Figure 13. The ASQ Software can also prompt a user for a suite file location as illustrated in Figure 14. The Dialogue Box 1300 and suite file location of
30 Figure 14 are part of the inventive Octane Software.

The ASQ Software can read in the initialized file from the caller script and passes the information to a start script 2. Both of these scripts 1, 2 are part of the inventive Octane Software.

5 The ASQ Software can read in the suite file information from the start script 2 and passes the information to the perform script 3. Both of these scripts are part of the inventive Octane Software.

10 The ASQ Software can read the first row in the Group File 400 as illustrated in Figure 4. The test file in the first row in the Navigate Column 18 can be opened. The GUI File indicated in the first row of the Group File in the GUI File Column 21 can be opened and the testing process can began. Each step in the test file calls a keyword. These keywords can be located in a case statement within the perform script.

15 These functions are part of Octane. Depending on the keyboard and the action performed on, the application refers to objects in the GUI File. The keywords access the GUI File in order to have the ability to enter information into radio buttons, select list boxes, and enter data in the text fields. The GUI File can be a function of the ASQ Software. The coding within each keyword making the press of the radio button, can select and enter actions to solely a function of the Octane Software.

20 The reporting function of the inventive Octane Software can run simultaneously with ASQ Software and can printout a “pass” or “fail” for each step in a test file. The same method of reporting is done for both the group and suite files. Once the end of a test file is reached, the file closes. At this point, the process repeats until the end of the test lists in a group file is reached. All the aforementioned functions are part of the inventive Octane Software.

25 Example Test File 500 - Run

30 The following is an example of setting up a Test File 500 that tests the front-end of a software application: With a new Microsoft Excel file, data can be entered into a first row of a spreadsheet such as illustrated in Figure 5. That is, a users enters the data for the Action Column 31, the Description Column 32, a Data Column 33, and a Testind Column 34. For this example, a user can test an Exemplary Web Page 1500A that is illustrated in Figure 15.

Referring briefly to Figure 17, a test file 1500B can be created that corresponds with the Web Page 1500A illustrated in Figure 15. All of the keywords in the action column 31A of the test file 500B of Figure 17 correspond to the available Keywords 1005 set forth in Table 1000 of Figure 10. In other words, a user can only enter one of the predefined keyword name types listed in Column 1005 of Table 1000 of Figure 10.

Referring back to Figure 14, the “select” Action 31A1 of the second element in the Action Column 31A corresponds to the Dropdown Menu 1105 of Figure 11 that can be selected. The Stateofautopurchase 32B1 of the Description Column 32B describes the type of information that is inserted into the Dropdown Menu 1505 of Figure 15.

The “North Carolina” Data 33B1 of the Data Column 33B in Figure 17 corresponds to Exemplary Data that can be entered into the Dropdown Menu 1505 of Figure 15 during the execution or run of the Test File 500B.

Referring back to Figure 15, the Web Page 1500A is usually established before objects can be pressed, selected, or entered. This usually must happen each time a new window is displayed. The keyword window goes into the Action Column 31A of the Test File 500B of Figure 17. In order to find out a name of the window, the GUI File (object file) usually must be opened. The GUI File can comprise a collections of objects about the application to be automated. This allows the ASQ Software to read the object in the file and find the matching object on the Web Page. The GUI File is typically part of the ASQ Software.

Referring now to Figure 16, this figure illustrates exemplary GUI File 1600. This exemplary GUI File 1600 comprises fields such as social security numbers, amounts to be borrowed, and phone numbers that will be tested by the inventive Octane software in combination with the ASQ software.

Referring back to Figure 17, in the first row of the Action Column 31A, the data of “window” 31A1 can be entered. In the first row of the Description Column 32B, the data of “auto” can be entered to further describe the Web Page 1500A being tested. In the Data Column 33B, the data of “autoredesign” 33B1 can be entered.

The description above is a small example of what the inventive Octane Software can accomplish. The description above corresponds to the inventive Octane Software except for the use of the GUI File of the ASQ Software to identify objects on the Web

Page 1500A. The description of the Web Page 1600 in Figure 16 is an example of a completed page using the Test File 500B of Figure 17 to run an automated test on the Exemplary Web Page 1500A of Figure 15.

5 Referring now to Figure 18, the figure illustrates Exemplary Web Page 1500B with information inserted during a test from the Test File 500B of Figure 17. The data of “North Carolina” 33B2 in Figure 1500B corresponds to the same data presented in the Excel Spreadsheet of the Test File 500B of Figure 17.