

Requirements-Driven Test Automation Process ZAProcess – Implementing TestDirector 8.0 & QuickTest Pro 6.5

"When preparing a patient for surgery, the most difficult part is not the surgery itself, but to explain to the patient the complexity of the surgery, and to convince him or her to have the discipline to follow the after-surgery procedures in order to get better..."

... Brain Surgeon

Problem

As test automation becomes industry standard, more QA managers and Technology Executives are becoming obsessed with it, allocating large budgets and very often getting disappointed by not realizing the expected return on investment. Then they ask themselves "Why? Why did we spent so much money over the years investing in the latest state of the art test automation tools and hiring the best test automation consultants, and we still don't have this scripts working for us?"

The answer to this question is very simple – test automation may be a very dangerous process when it used inappropriately, with a lack of tool knowledge, or poor planning. You may lose yourself in the maintenance of automated test scripts (TestScripts), and lose focus of why you automate. The goal of automating is not just to develop TestScripts that can run overnight and execute some reports, but to develop a robust framework that can generate automated testing coverage of application requirements and result in a tremendous Return On Investment.

Test automation becomes extremely powerful when it is properly prepared and is used as an accessory to the overall testing process, and not as a stand-alone entity.

What to do?

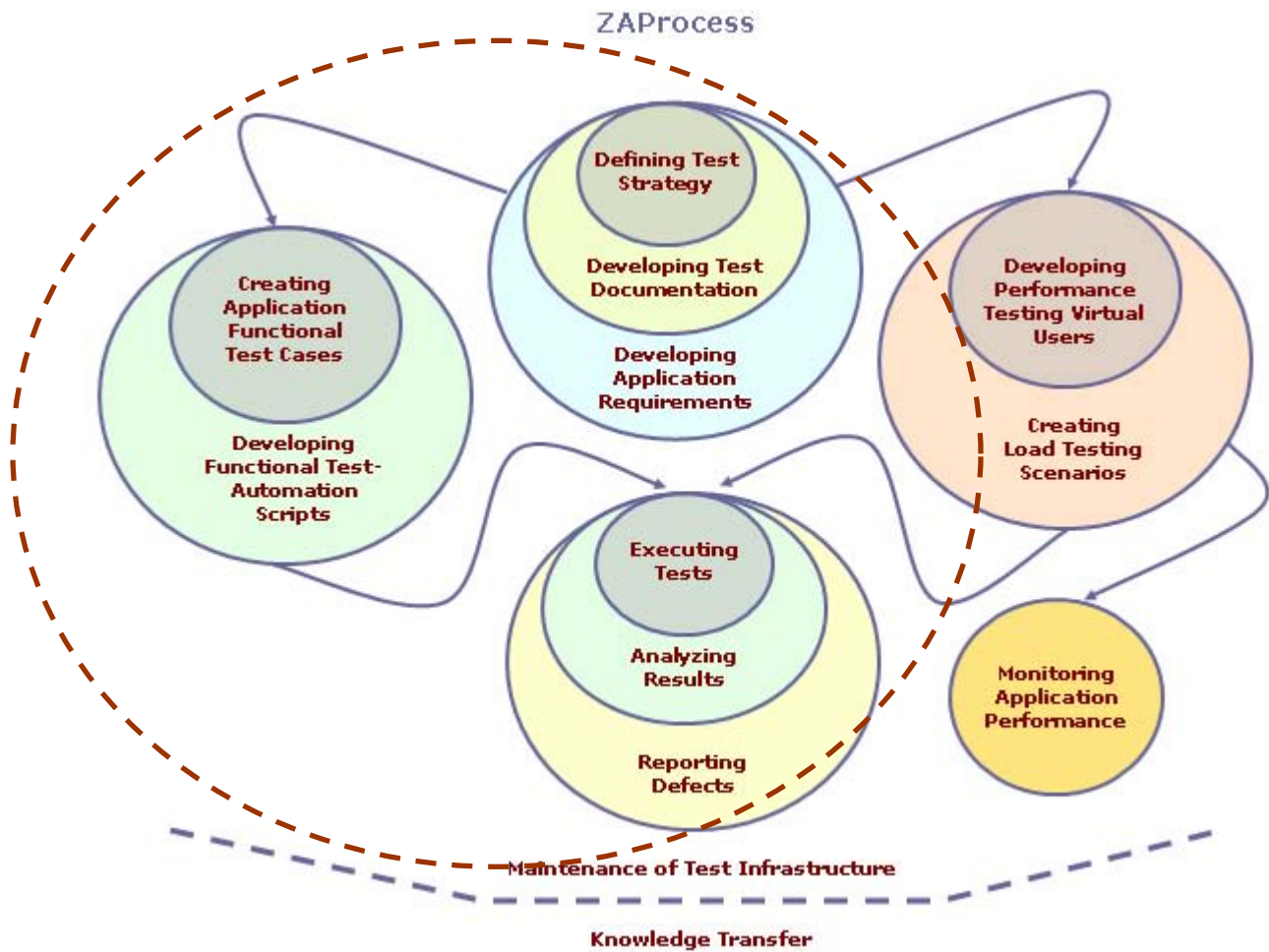
At ZapTechnologies.Com we are specializing in the management of software testing processes, automated functional and performance software testing, utilizing products and technologies by Mercury Interactive Corporation, exclusively. In our consulting practices, we start our Special Forces approach to testing by encouraging our clients to look at the testing process at the global scope, and prepare their testing artifacts for automation in order to develop a repeatable and robust test-automation framework.

Let's consider the test process as an example of the ZAProcess – process of implementation of QA practices. Since Mercury has become an industry standard for testing and application management, we think many of you using these products in your organizations will easily relate to this process.

ZAProcess

The ZAProcess includes test management, functional and performance testing and application monitoring. ZAProcess is oriented on testing application requirements, and is fully requirement-driven.

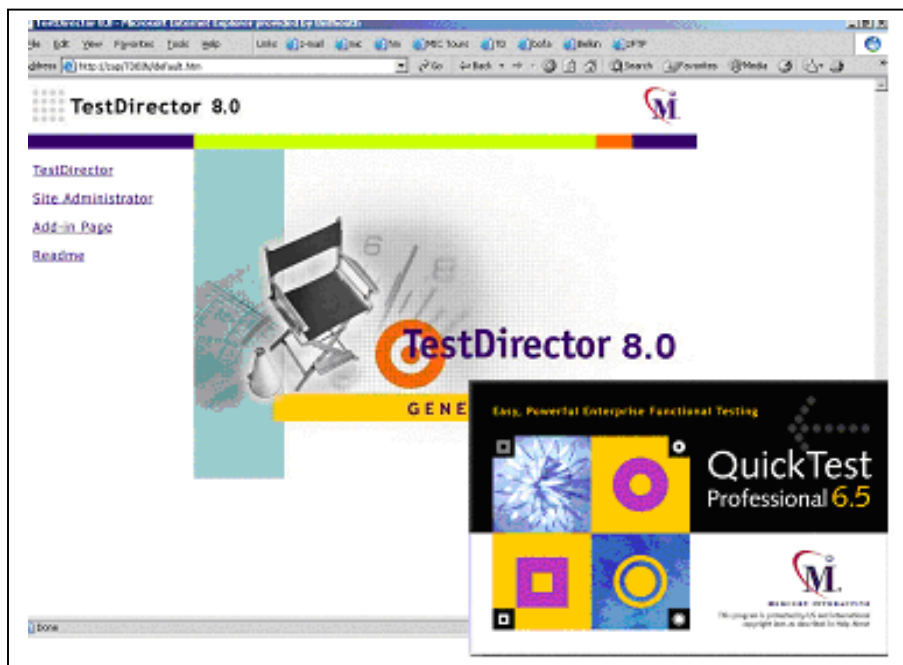
Today we are going to consider the functional testing phases of ZAProcess. The overall fundamentals of the ZAProcess requirement-driven testing approach are: *Defining Requirements > Developing Manual TestCases > Creating Requirement Coverage > Developing and Executing TestScripts and Analyzing Results > Managing Requirements and Associated Test Cases.*



How to do it

In order to implement ZAProcess we are going to consider following tools – TestDirector 8.0 as a test management tool and QuickTest Pro 6.5 as a functional test automation tool.

Let's review these phases, keeping in mind that we are going to use TestDirector and QuickTest Pro in the TestSuite architecture, meaning that we are going to use TestDirector as our only test management environment utilizing its components only...

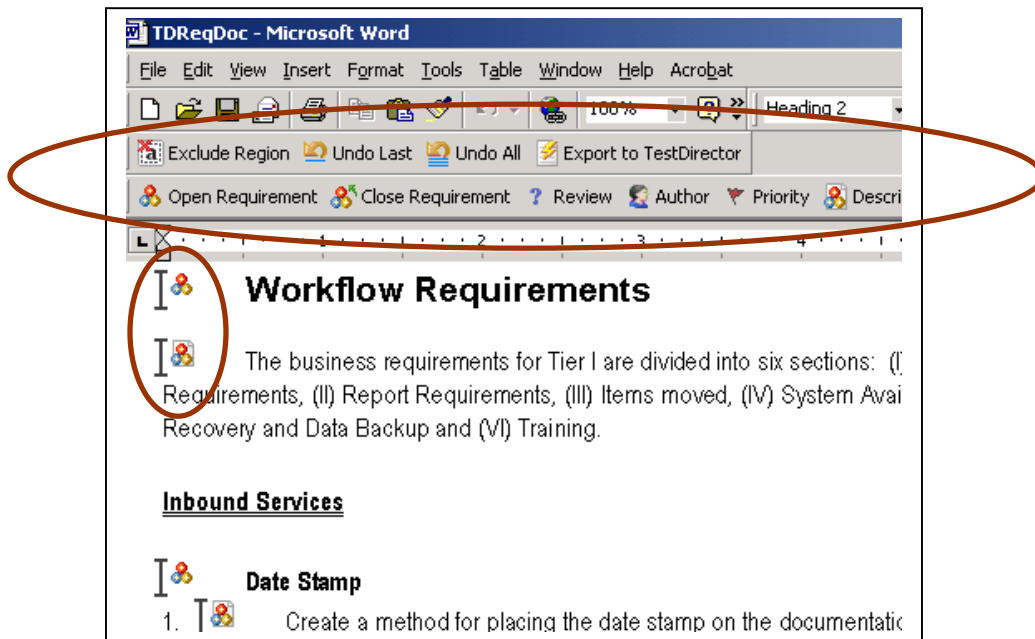


Defining Requirements of the Application Under Test

If you are small, midsize or large organization, you probably have QA Analysts or Subject Matter Experts (SME), who have solid knowledge of the application under test, functionalities and business process, and are responsible for developing manual test cases (TestCases) based on various sources of application documentation using MSWord or Excel.

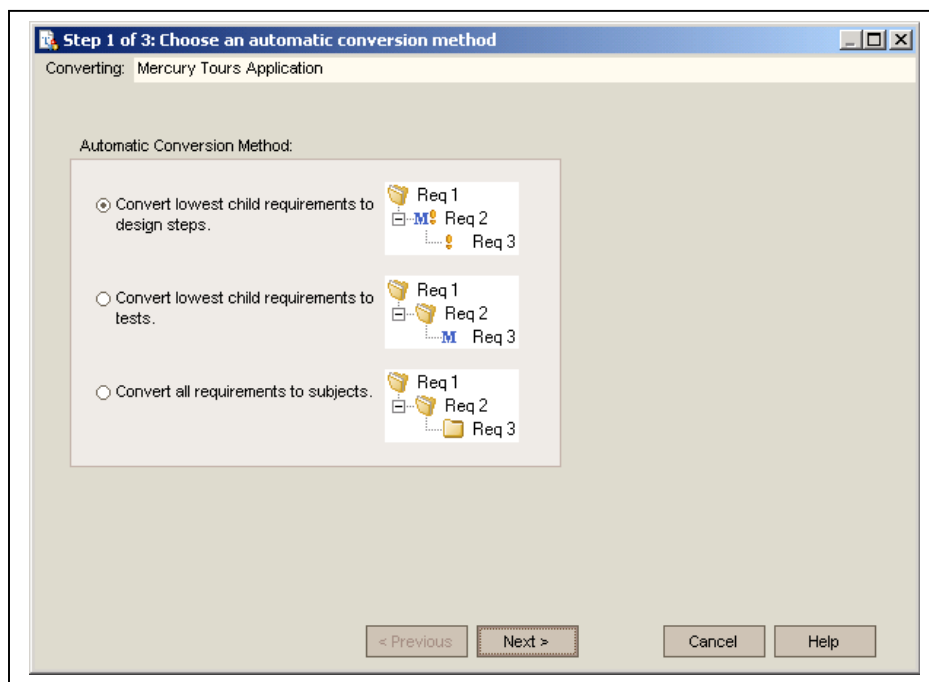
Let's take these sources and convert them into application requirements, populating the Requirements module of TestDirector. Sure, you can use the common way of doing that, using TestDirector dialogs and populating them in the nodes and sub-nodes with appropriate descriptions. If you have had experience doing that, you probably will agree with me that this is a slightly time consuming process, especially

if you are new to TestDirector and used to working with Word and Excel, and you probably will not like the idea of using TestDirector as your testing cockpit. In this case, we offer MSWord and Excel add-ins for TestDirector. These add-ins allow you to export your MSWord or Excel based documents into the Requirements module of TestDirector simply by placing the appropriate TestDirector related icons into your MSWord document, or creating a map with appropriate columns in your Excel spreadsheet including the hierarchy of the requirements, name of the author, status, descriptions, etc.



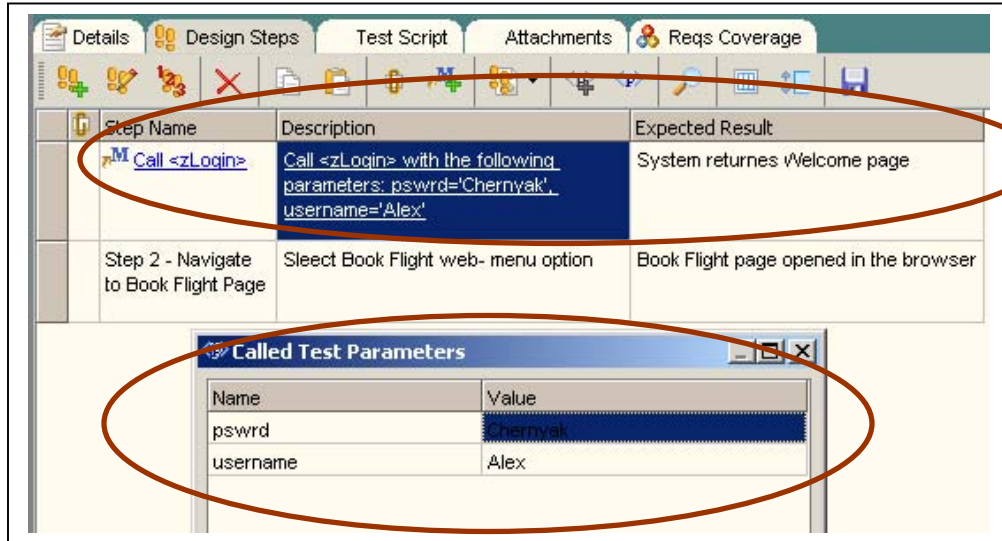
Creating TestCases

Once you have outlined the requirements of the application under test (AUT), you will develop manual test cases (TestCases) to test these requirements. The easiest way to do that is to use the 'Convert To Test' utility in the Requirements module. This is a wizard that allows you to convert nodes and sub-nodes of the requirements tree into the test tree, TestCases, and design steps based on your test scenarios. This helps you to outline the fundamentals of your test architecture.



In our practice, we are very often coming across manual test cases that were developed in a procedural manner, where each test case has number of precondition and post condition test steps, which repeat across test cases. Utilization of TestDirector 8.0 allows you to create an event-driven test architecture using Templates. Templates are test cases that include repeatable steps and can be called from other TestCases.

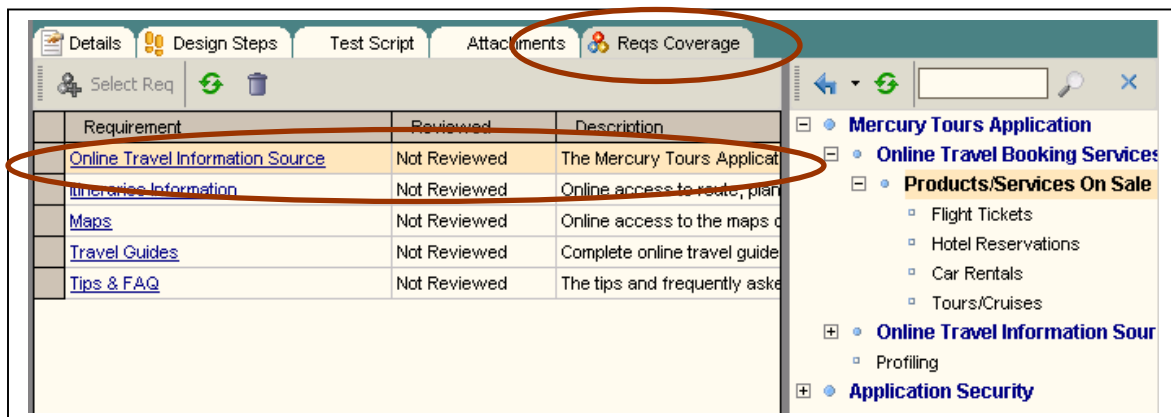
For example, in order to perform a certain test case, you need to log into the AUT. Instead of inserting steps of a login procedure in every TestCase, you can create a test case Login that includes login steps, save that as Template, and call it as a first step in every TestCase. You can also define Parameters of dynamic values of this Template, like "user name" and "password", which can vary in different TestCases.



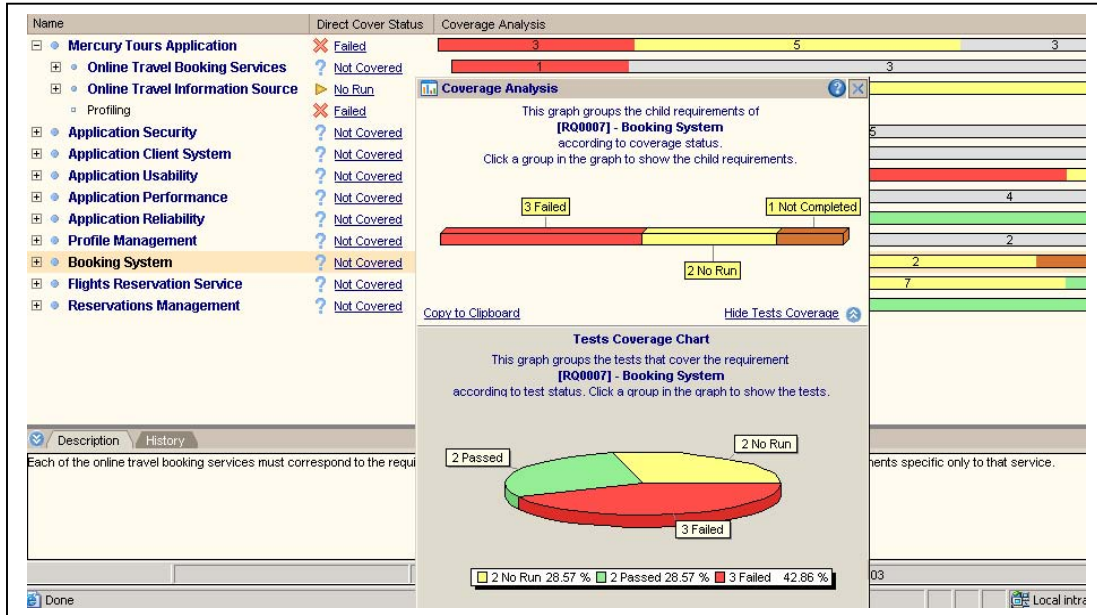
Developing TestCases in an event-driven manner like this, allows you to develop a TestCase-Independent architecture, where each TestCase may include pre- and post-conditions, in order to allow an independent test execution.

Creating Requirement Coverage

Each TestCase has to test one or number of application requirements defined in Requirements module. And, in order to track this Requirement - TestCase relationship you need to create Requirements Coverage. After creating a TestCase, use the 'Select Req' utility in the 'Reqs Coverage' tab of the TestCase, and select the desired Requirements that are covered in the TestCase.



The Requirements module in TestDirector also allows you to have a graphical representation of the TestCases coverage per given requirement, with TestCase execution results, by selecting 'Coverage Analysis View' option.



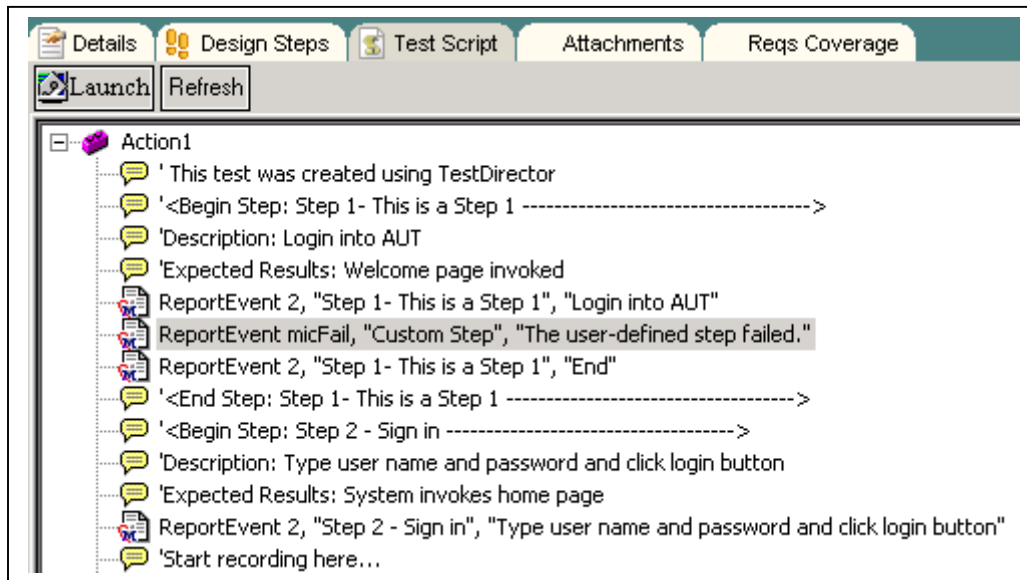
Developing and Executing TestScripts and Analyzing Results

Only after the above steps are performed, you can consider your testing process requirement-driven, and your test artifacts are ready for automation.

Test automation is a very complex development process and it would take a whole separate article to describe all of its best practices. Here, we just want to go over test automation as a part of the whole process and to show you how to leverage that as an accessory of your testing lifecycle.

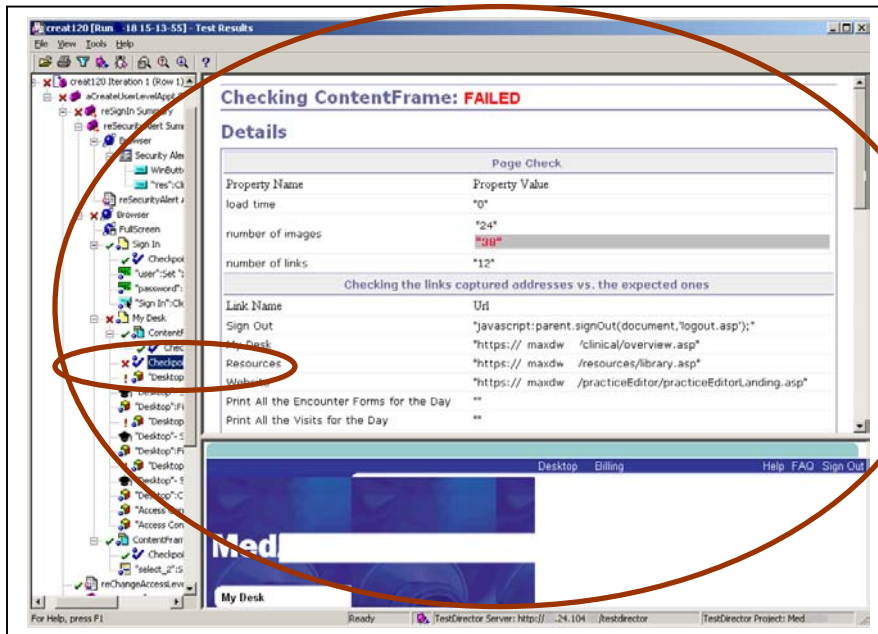
I would just like to mention that in order to use QuickTest Pro in the TestSuite architecture where all TestScripts will be stored and executed from TestDirector interface, you need to at least: First - connect QuickTest to a TestDirector Project, using the TestDirector Connection utility in QuickTest Pro. Second, store your QuickTest utility components like Object Repository files (For WinRunner users they known as GUI Map files), Data Tables, External Libraries, etc. as attachments in the connected TestDirector Project.

Using QuickTest Pro 6.5 you can convert your TestCases into a TestScript template, which will include your actual and expected results as comments of the QuickTest source code. You can do that with your Templates as well, which can be converted into Reusable Actions – TestScripts that can be used in a similar manner across your test-automation architecture.

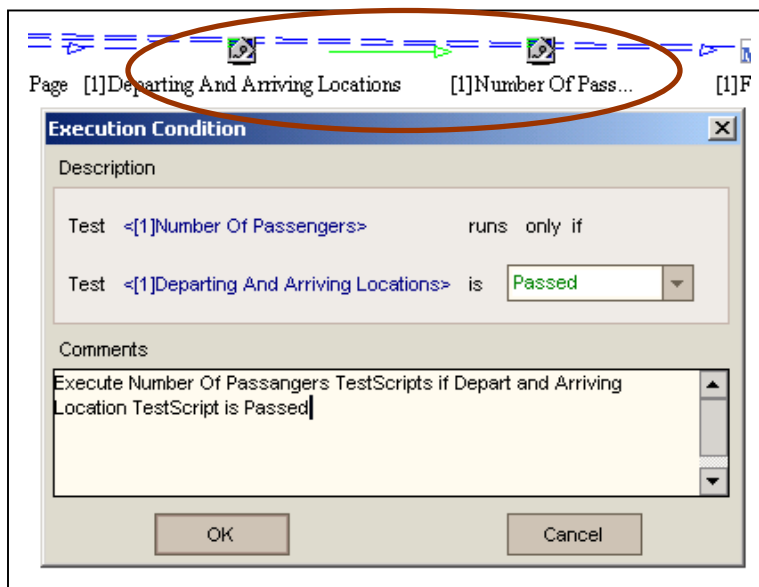


Test automation can be developed on any level based on schedule, skills and experience of your test automation group. In any case, any test automation practice results in execution TestScripts, and generates reports of a test execution.

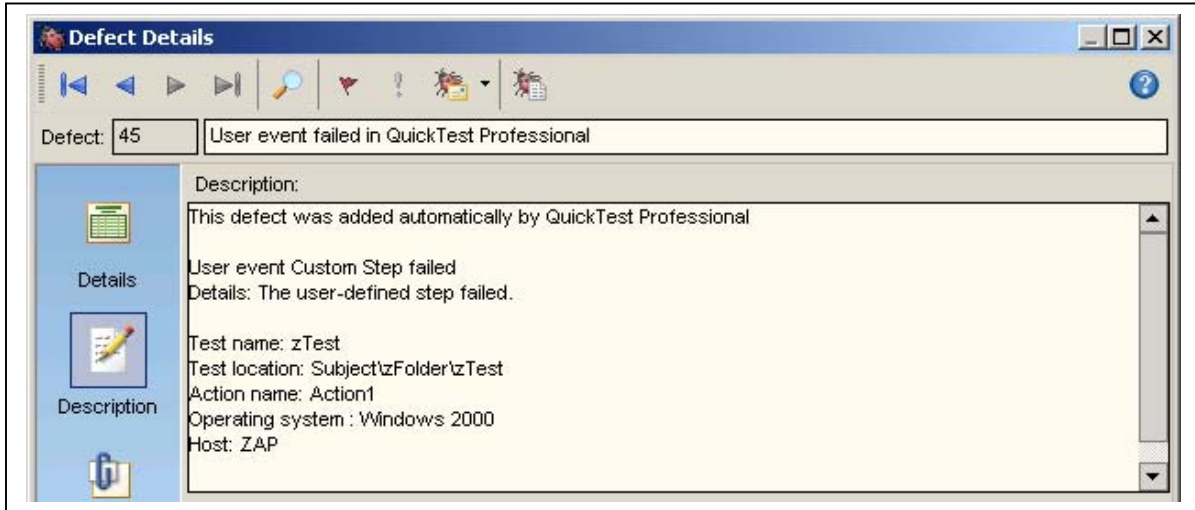
If you are an intermediate QuickTest user and do not have the skills and time to develop a sophisticated test automation architecture, creating your testing process and following the above steps, you can configure QuickTest Pro 6.5 to create global scope checkpoints automatically as you navigate to another page or screen in your application. That gives you an opportunity as a QA Analyst or SME to create icon-based (no scary source code) sanity TestScripts, just by performing the test steps developed by you in the previously created TestCase. You don't need to create extensive reporting of your TestScripts - QuickTest Pro generates descriptive reporting per each step in your TestScript by default. And even more – in the case of failure of the test step, QuickTest will take screenshot of the web page or application screen where the failure occurred, and will include it into the test execution report.



Execute your TestScripts in Test Sets organized in small modules in a nested hierarchy. You can define an Execution Flow of the Test Set, execution conditions, recovery scenarios, and notification rules.



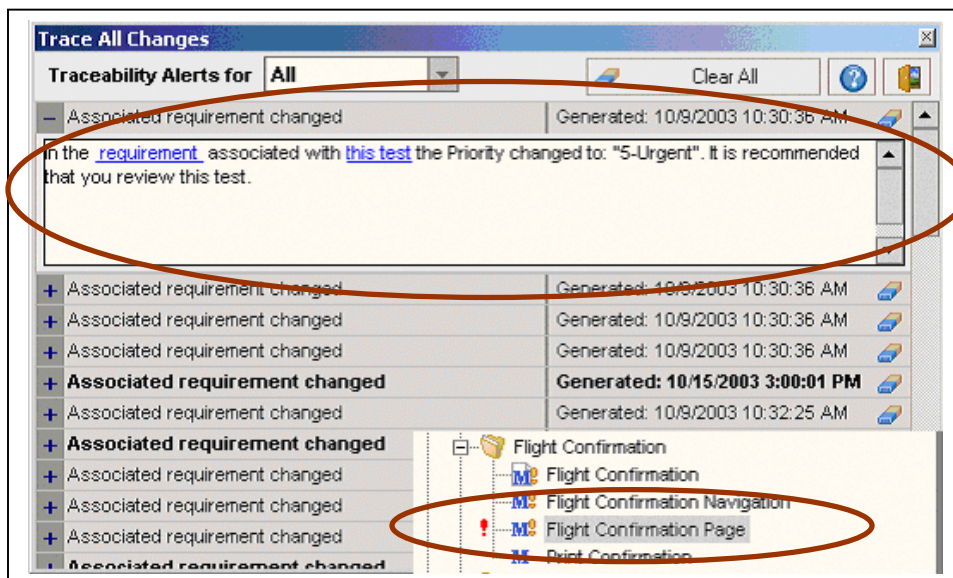
You don't even need to manually report defects found during TestScript execution – QuickTest Pro 6.5 can submit all the defects into the Defect module of TestDirector automatically, and you or your Test Coordinator in the QA department can manage them accordingly.



Managing Requirements and Associated Test Cases

You can use the Traceability settings of TestDirector to manage and monitor your requirements and associated test cases.

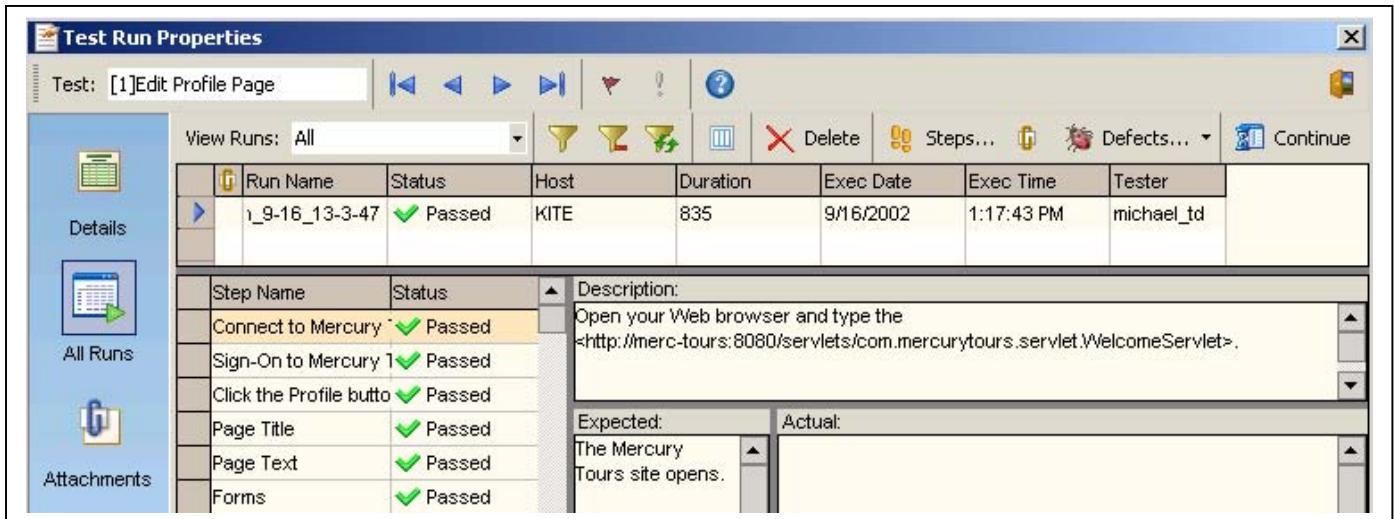
For example, one of your requirements was updated; using the Traceability utility allows you to alert the test designer(s) responsible for any test cases covering the requirement to update these TestCases or TestScripts. This allows you to properly maintain your automated TestScripts and make sure they up to date with the requirements.



Summary

- Organize your testing process requirement-driven, which should have phases: *Defining Requirements of the Application Under Test > Developing Manual TestCases > Creating Requirement Coverage > Developing and Executing TestScripts and Analyzing Results > Managing Requirements and Associated Test Cases.*
- Extensively utilize build-in functionalities of test automation tools. This will save your TestScript development time.
- When using TestDirector, utilize most of its functionalities while building Test Sets for TestScript execution. This will prevent you from developing various utility, drivers and conditional scripts in your test automation architecture.
- Test automation has to be considering as a part of overall testing process, resulting in testing application *requirements*, and it has to work for you and not vice versa. Preparation is the key of successful test automation. Do not hesitate to invest your efforts into outlining your application requirements, and taking modular approach to developing manual test cases - especially ones you are planning to automate.

If you properly prepare your test artifacts for automation, and will use that in the requirement-driven process, you will see tremendous Return On Investment (ROI) of Test Automation.



Alex Chernyak
ZapTechnologies.Com