

The Essence of Test Automation

White Paper - Version 1.0

Abstract

Software test automation can dramatically reduce costs and speed up time-to-market, but its full potential cannot be achieved without a carefully planned, scalable and maintainable process. Without the right strategy, expensive test automation tools may become “shelf ware.”

This paper will first discuss the key benefits of software test automation, and examine the most common implementation tactics. It will then analyze the key reasons why test automation efforts often fail to meet their potential. Finally, it will show how using a keyword-based test automation strategy enables organizations to avoid those problems.

Action Based Testing, featuring the latest improvements to the keyword-based strategy from the original architect of the keyword method, and the Test Architect toolset will be presented as real-world examples of how the full potential of test automation can be realized.

Improved Quality of Automated Tests

The test designers follow a top-down approach which ensures that there is a clearly stated purpose for every test. The first step is to determine how the overall test automation effort will be broken down into individual test modules. Some common ways of grouping tests include:

- Different functional areas of the application.
- Different types of tests (positive, negative, requirements-based, end-to-end, scenario-based, etc.).
- Different quality attributes being tested (business processes, UI consistency, performance, etc.).

Once the test modules have been identified, the next step is to define test requirements for each module. Test requirements are critical because they force test developers to consider what is being tested in each module, and to explicitly document it.

Once the test requirements are defined, they serve both as a roadmap for developing the test cases in the module and as documentation for the purpose of the tests. Each test case is associated to one or more test requirements, and each test requirement should be addressed by one or more test cases.

By explicitly stating the test requirements, it is possible to easily determine the purpose of a test, and to identify if a test does not sufficiently meet those test requirements. Test requirements can be quickly checked to determine if the test needs maintenance or even retirement. Test developers can be precise and concise in their test creation, creating enough tests to meet their stated requirements without introducing unwanted redundancy.

After explicitly defining the test requirements, the Test designers can start implementing the test cases using either predefined actions or by defining new actions. Test designers can define their tests as high-level business processes, which allow the tests to be more readable than tests defined using low-level interface interactions.

The Benefits of Software Test Automation

Most software development and testing organizations are well aware of the benefits of test automation. A quick glance at the Web sites of any test automation tool vendor will point out a number of the key benefits of test automation. Some of these benefits include:

Reduced test execution time and cost: Automated tests take less time to fully execute than manual tests, and can generally run unattended. A tester simply starts the test, and then analyzes the results when completed.

Increased test coverage on each testing cycle: Automated tests enable testing teams to execute large volumes of tests against each build of their application, achieving a level of coverage that would not be possible with manual testing. This can help teams uncover bugs in existing functionality much more quickly than through manual testing. Test automation enables teams to test more features in each cycle (breadth), and to test those features with a larger set of inputs (depth).

Increased value of manual testing effort: So long as applications are meant for human end users, test automation will never entirely replace the need for human testers. No matter how sophisticated test automation tools become, they will never be as good as human testers at finding bugs in an application. Human testers will instantly notice subtle bugs that are almost never detected by test automation, particularly usability bugs. Automated test tools cannot 'follow their instincts' to uncover bugs using exploratory and ad-hoc testing techniques. By freeing manual testers from having to execute repetitive, mundane tests, test automation enables them to focus on using their creativity, knowledge, and instincts to discover more important bugs.

Investment in Automation

Automation is not cheap. Not only does it take time and effort to evaluate and select a test tool: it takes money to buy it and time to learn to use it; and more time to actually implement it; and the most time of all to maintain the test library as the application changes. In fact, experience shows that it takes between five and ten times as long to develop and debug an automated test as it does to execute it manually.

How to make the investment to get the return, and in most cases you will execute the same test at least ten times over the life of the application or even within the same release cycle. The problem is that the goal of test automation should not be to reduce either test resources or cycle time. The goal should be to reduce the risk and cost of software failure by increasing test coverage and the primary goal is to reduce the cost of testing, then we can propose a foolproof way of cutting 100 percent of your testing budget without even buying a tool.

The Real Payoff

If you want to justify test automation, don't look at your test budget, look at the cost of failure for the system you are testing. What does it cost the company or its customers in time, resources, and money if defects escape to production? What is it worth to deliver on time with a quality product? If you do this analysis, you may find that test automation justifies a bigger test budget, not a smaller one. What Do You Lose With Automation? Creating an automated test is usually more time-consuming (expensive) than running it once manually. The cost differential varies, depending on the product and the automation style.

- If the product is being tested through a GUI (graphical user interface), and your automation style is to write scripts (essentially simple programs) that drive the GUI, an automated test may be several times as expensive as a manual test.
- If you use a GUI capture/replay tool that tracks your interactions with the product and builds a script from them, automation is relatively cheaper. It is not as cheap as manual testing, though, when you consider the cost of recapturing a test from the beginning after you make a mistake, the time spent organizing and documenting all the files that make up the test suite, the aggravation of finding and working around bugs in the tool, and so forth. Those small "in the noise" costs can add up surprisingly quickly.

- If you're testing a compiler, automation might be only a little more expensive than manual testing, because most of the effort will go into writing test programs for the compiler to compile. Those programs have to be written whether or not they're saved for reuse.

Suppose your environment is very congenial to automation, and an automated test is only 10% more expensive than a manual test. (I would say this is rare.) That still means that, once you've automated ten tests, there's one manual test - one unique execution of the product - that is never exercised until a customer tries it. If automation is more expensive, those ten automated tests might prevent ten or twenty or even more manual tests from ever being run. What bugs might those tests have found?

So the first test automation question is this: The answers will vary widely, depending on your project. Suppose you're a tester on a telecom system, one where quality is very important and the testing budget is adequate. Your answer might be "If I automate this test, I'll probably lose three manual tests. But I've done a pretty complete job of test design, and I really think those additional tests would only be trivial variations of existing tests. Strictly speaking, they'd be different executions, but I really doubt they'd find serious new bugs." For you, the cost of automation is low.

Or you might be a testing version 1.0 of a shrink-wrap product whose product direction and code base has changed wildly in the last few months. Your answer might be "Ha! I don't even have time to try all the obvious tests once. In the time I would spend automating this test, I guarantee I could find at least one completely new bug." For you, the cost of automation is high.

My measure of cost - bugs probably foregone - may seem somewhat odd. People usually measure the cost of automation as the time spent doing it. I use this measure because the point of automating a test is to find more bugs by rerunning it. Bugs are the value of automation, so the cost should be measured the same way.

Conclusion

As with other areas of software development, the true potential of software test automation is realized only within a framework that provides a scalable structure. Keyword-based test automation has become the dominant approach precisely because it provides the best way to achieve this goal

Action Based Testing offers the latest in keyword-based testing from the original architect of the keyword concept. Test design, test automation and test execution are all performed within a spreadsheet environment, guided by a method focused on an elegant structure of reusable high-level actions. The result is a truly scalable, maintainable, reliable software test automation process that allows all contributors in a testing organization to focus on what they do best.

About STC

STC ThirdEye Technology (India) Pvt Ltd is India's largest Independent software testing organization providing End-to-End testing Services.

We build and operate dedicated India-based testing centers for our customers with the latest computing and data communication technologies, and deliver our services, with high standards of security and confidentiality. Consistent qualities of deliverables under compressed time schedules enable us to get repeat business.

We help Fortune 500 ERP, BFSI, Healthcare, Gaming and Telecom solution providers We are ISO 9001:2000 certified organization. For more details, please visit us at www.stcthirdeye.com

Disclaimer

The Whitepaper series presents reports on subjects in the sphere of activities of STC ThirdEye Technology (India) Pvt Ltd that are to be considered in the interest of wider public. These papers are part of the ongoing studies and authors will be glad to receive your comments.

The views expressed in these papers are to be regarded as those of the author and should not be interpreted as reflecting the views of the management of STC ThirdEye Technology (India) Pvt Ltd.

STC ThirdEye Technology (India) Pvt Ltd assumes no responsibility for any actions taken by Anybody based on the information provided in this paper.