WHITE PAPER

PATH TO SUCCESSFUL FUNCTIONAL TEST AUTOMATION

Author: Amiya Pattanaik

Is the TOGAF (**T**he **O**pen **G**roup **A**rchitecture **F**ramework) certified, AWS Certified Solutions Architect – Associate. He has worked for technology companies like BMC Software, Apple Inc., Mastercard, Cognizant, TechMahindra etc. He is an active contributor to the open source community, such as WebdriverIO, NPM, Github and has developed several test automation frameworks for UI, Mobile, REST/SOAP API, DataBase testing using open source and commercial tools which has been widely used by open source community group across globe.

Executive Summary

Organizations today face several Quality Engineering challenges. They must optimize the quality of increasingly complex software applications – and do so more quickly and cost-effectively than ever before – in order to deliver winning solutions that yield a high return-on-investment (ROI) and drive competitive advantage.

There is no question that test automation can allow organizations to produce higher-quality software while leveraging existing resources. However, successful organizations realize that an automated functional testing solution is only part of the answer. To optimize software quality, speed time-to-market and heighten ROI, these companies think beyond an automated tool. They view automated testing as a much larger strategic endeavor – one that calls for an initial investment in planning a quality-driven software development process, training quality-related personnel and most importantly developing the right test framework/harness before they can achieve the benefits of automation.

This white paper provides practical insight into the lessons learned by those who have successfully automated the functional testing process. It answers questions such as: • What is the strategic role of functional testing in today's enterprise? • What are the value-added benefits associated with automating the functional testing process? • What is the best approach to ensure the success of your automation effort? • What should you look for in an automated functional testing solution?

What is functional testing?

Functional testing, or black box testing, is a Quality Assurance (QA) process used to verify that an application's end-user functionality (i.e., the ability to log in, complete a transaction, etc.) works accurately, reliably, predictably and securely. Functional testing can involve either manual or automated methods. Either way, it entails running a series of tests that emulate the interaction between the user and the application in order to verify whether or not the application does what it was designed to do.

Why automate functional testing?

While manual testing is appropriate in some cases, it is a time-consuming and tedious process that is inefficient and conflicts with today's shorter application development cycles. As a result, it impedes the ability to thoroughly test an application – enabling critical bugs to slip through undetected. And if an application needs to run on multiple platforms, the manual testing effort grows in parallel with the number of platforms to be tested. What's more, manual tests are prone to human error and inconsistencies that can skew test results.

Automated testing creates new efficiencies that accelerate the testing cycle and promote software quality. By automating regression tests and other repetitive tasks, for example, QA personnel are free to expand their quality efforts. That means they can increase test coverage by extending automation to parts of the application that may not have been thoroughly tested in prior releases. Automated testing further optimizes software quality and testing efficiency by delivering the following long-term advantages:

• **Reusability** – Automated testing enables QA to meet tight release schedules by reusing existing tests instead of starting from scratch with each new testing effort. Experience proves that reusable tests are run more frequently, enabling personnel to find and fix more errors earlier in the development process. This reusability benefit also enables QA to build libraries of repeatable test assets – in effect, transforming each test into intellectual property with long-term value.

• **Predictability & consistency** – With automation, QA can rerun a test with the utmost consistency. This is especially critical when development creates a new build. By running regression tests, QA can quickly verify that all pre-existing functionality still works in the new version and provide early feedback to development. Consistency also applies to the testing process itself. With a repeatable process for documenting test results, QA can reproduce and verify errors – speeding the resolution process.

• **Productivity** – Automated testing creates a high-productivity environment in which organizations can increase testing capacity without additional resources. With automation, for example, QA organizations can run tests unattended on a 24/7 basis, and test an application across multiple platforms, browsers and environments simultaneously. This frees up personnel to concentrate on other quality issues. The resulting productivity gains have the dual effect of shortening test cycles and increasing opportunities to optimize software quality. Collectively these automation advantages enable QA to accurately assess quality levels, make sound decisions regarding release readiness and minimize deployment risk.

How do you successfully automate the functional testing?

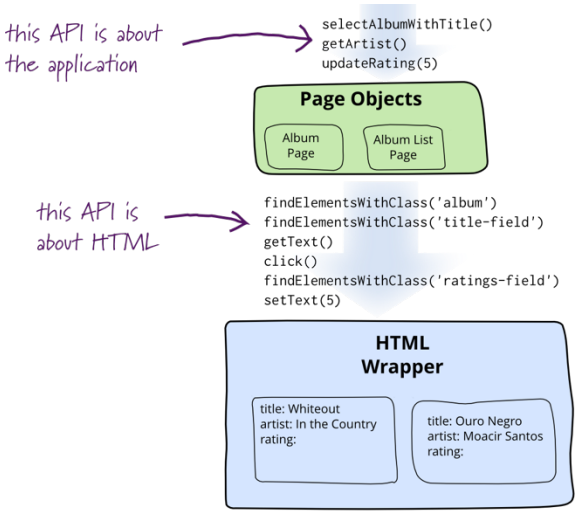
1. Determine what applications should be automated

The best candidates for automation are enterprise applications that will require multiple releases throughout their useful life due to new, expanded or changed functionality. Applications that must produce a consistent set of results using relatively stable data are also well-suited for automation. These application characteristics enable the quality effort to fully leverage the reusability and predictability benefits of automated functional testing.

2. Choose your automated testing approach

Several methodologies exist for creating automated functional tests.

- **Test modularity** - is an approach that divides the application under test into script components or modules or page objects. Using the scripting language of the automated testing solution, QA builds an abstraction layer in front of each component – in effect, hiding it from the rest of the application. This improves the maintainability and scalability of automated test suites by protecting the rest of the application from changes within an individual component. For example, A page object wraps an HTML page, or fragment, with an application-specific API, allowing you to manipulate page elements without digging around in the HTML.



- **Test library architecture** - is another scripting-based framework that divides an application into modules which are used to build tests. Unlike test modularity, however, the test library architecture framework describes these modules in procedures and functions rather than scripts – enabling even greater modularity, maintainability and reusability.

- **Keyword-driven testing** - is an application-independent framework that uses data tables and easy-to-understand "keywords" to describe the actions to be performed on the application under test. The data tables and keywords are independent of both the automated testing solution that executes them and the test scripts that drive the application and its data.

- **Record/playback testing** - eliminates the need for scripting in order to capture a test. It starts by recording the inputs of manual interactions with the application under test. These recorded inputs are used to generate automated test scripts that can be replayed and executed later. While record/playback is the fastest and easiest approach to test automation, the tests are neither maintainable nor reusable. Any change in the application means re-recording the entire sequence of steps – negating any time savings. Record/playback produces the lowest ROI, test asset reusability and resource productivity of any form of automation.

3. Develop your application test plan

The application test plan is a document that describes the scope, approach, resources, coverage and schedule of all the automated and manual activities involved in testing an application. Specifically, the test plan identifies all application features to be tested, the testing tasks, the individuals responsible for performing each task, the test environment, the chosen approach to test design, the various platforms and environments to be tested, and the test metrics for results reporting.

4. Create and deploy your automated tests

The application test plan serves as the roadmap for creating automated tests. Based on this plan, QA develops each test using an automated testing solution that supports the chosen approach to automation (i.e., test modularity / page object, test library architecture, keyword-driven testing, data-driven testing or record/playback). Once the tests are mapped to requirements, defined and created, the automation is ready to start working for you.

What are the best practices of functional test automation?

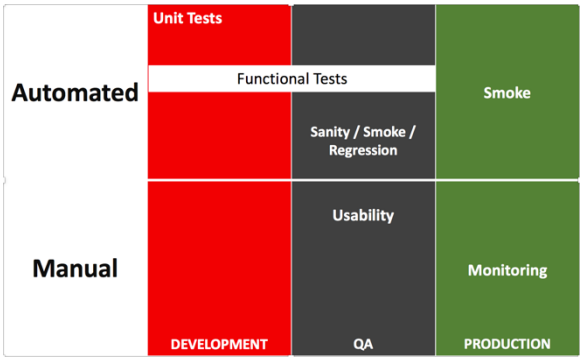
Treat your automation project as a software development effort

Just like software development, test automation must be carefully designed, documented and reviewed. Successful organizations view quality holistically across the entire software application lifecycle and apply best practices when implementing test automation. For example, they move testing up early in the development process – even as early as requirements definition. They use modular code that is easy to maintain. In addition, they build test assets that can be reused in later quality phases and future versions of the application.

Implementing test automation is not a short-term project. Those who succeed understand that test automation is a full-time effort that requires a significant upfront investment in time and resources. So, they set management's expectations accordingly. They also realize that an automated solution is no substitute for expertise.

Which test cases should be automated?

Every test has its own place in the testing process. The majority of test can be automated, but there are a few tests that need to be performed manually. This graphic gives you an overlook of what should be tested when during an apps' life cycle.



A test case or use case scenario is a simulated situation in which a user performs determinate actions when using a particular app. Some typical test cases can be automated, and it would be a very good idea to do so to save money, time and keep users as happy as possible! Depending on your app, you may want to write specific test scripts for certain features and actions.

Nonetheless there a few criteria to decide which tests to automate and these are test automation best practices.

1. Automate the most frequent test-cases

There may already be a number of tests that you're manually running as routine tests to check basic functionality on different devices and after every minor update. Those tests might be a good start for test automation since they are executed repeatedly and running them manually might lead to errors, such as forgetting a step, so that the results might not be reliable.
Above all performing repetitive test is time-consuming and extremely boring, so don't let the talent of your testers get wasted on something a script can do better.

2. Automate test cases that are easy to automate

Automated test cases that are easy to automate to have positive results on your ROI: it will take a little time to write effective scripts that will save time and money in the long run.

3. Automate test cases that have predictable results

You can automate test with conditional actions, which have predictable results e.g. if you click on the icon, the app must be launched. Those kinds of tests, which state the obvious are not obvious at all. There are apps which pass every in-app test, but when a user tries to launch it, nothing happens.

4. Automate the most tedious manual test cases

Manual testing can be tricky at times, that's when automated testing get his big spotlight. Software testing is always tricky but it becomes even trickier when it comes to mobile app testing. Every device is different and every manufacturer changes Android a little bit to personalize it, so manual testing can become tedious and complicated. We can automate those tests so to get the exact and right information that we need to know about our app.

5. Automate test cases that run on several different hardware or software platforms and configuration

With today's device fragmentation, testing on as many devices as possible is of the most important things to do before making the debut in the app market. So, writing tests that can be run on different devices, from different manufacturers, on different OSs and OS versions will save you a great amount of time.

Ensure collaboration to optimize software quality

Quality is not just the domain of the QA tester; it is a goal that must be owned by the entire company. By creating a collaborative environment with domain experts and software developers during pre-deployment and with operations throughout post-deployment, QA can capture all the relevant input needed to optimize software quality and ensure that applications meet business requirements.

One way in which successful QA organizations promote such collaboration is by creating the position of Automation Architect to fill the role of a test developer. The Automation Architect collaborates with domain experts to understand what business functions the application must perform and how the application must work in the eyes of its users. The Automation Architect can then design the test automation framework accordingly, ensuring that the application will meet its business requirements and end-user needs for functionality, quality and reliability.

Another way to foster collaboration is to seek a software test automation vendor with an integrated platform that expands the quality process beyond functional testing to encompass the entire software application lifecycle. Such a platform infuses quality each step of the way – from initial design, to deployment and into production. It also promotes cross-functional communication by enabling all stakeholders to access and share the quality-related information captured within the platform repository.

Manage constantly changing applications

Change is inevitable – and extensive – in software applications as new business objectives, user

requirements and computing environments emerge. In fact, the software maintenance required to keep pace with changing applications accounts for more than 70% of all development costs..

Successful companies mitigate these costly maintenance activities by building maintainability into test automation. For example, they create modular test components that isolate and encapsulate application functionality. When a change in functionality occurs, they only have to modify the effected component(s) instead of rescripting the entire test – enabling them to quickly and easily modify tests to reflect changing application needs.

Ease of maintenance and reusability are two major areas in which test automation drives ROI. The best way to build these two attributes into test automation is to adopt an automation solution with an object-oriented architecture. Such an architecture allows the use of any chosen approach to automation (i.e., test modularity, test library architecture, keyword-driven testing, data-driven testing or record/playback). Most importantly, an object-oriented architecture provides a dramatic reduction in the time and effort necessary to modify tests in order to reflect new or expanded functionality. As a result, organizations can keep up with application changes and maintain a high level of reusability among test assets from build to build.

What should you look for in an automated functional testing solution?

A test automation environment designed for maximum productivity

To keep pace with today's short product release cycles, QA organizations need an automated solution that enables them to build modular and reusable test assets. In addition to accelerating the testing process, reusable test cases optimize software quality by increasing the predictability of test runs.

High adaptability to cope with ever - changing business requirements

New business requirements can emerge literally overnight – creating an immediate need to revamp an application. But will you be able to revamp all of the associated test cases in time? Look for an automated solution that minimizes the burden of heavy maintenance. The inherent reusability and easy maintenance of an object-oriented testing solution will provide the ability to change test scripts quickly, along with the portability and extensibility to embrace different technologies.

Quality optimization throughout the application lifecycle

Though functional testing may be an initial priority, quality does not begin and end with functional testing. Ultimately, quality needs to extend to other areas such as unit testing, performance and load testing, test management and application performance management once the application is in production.

Maximizing test throughput

The increasing need to get to market faster also means there is less time to test. Test frameworks which provides parallel testing, multi browser parallel testing, multi-platform can be used to optimized on test execution speed which becomes a deciding factor in case of limited testing time. These optimizations allow you maximize test coverage and increase confidence in quality.

Ease of maintenance

The fragility of test scripts is one of the critical factors when Applications Under Test (AUT) are modified during development. Choose the automation scripting language which is in line with the application's implementation language. For example, JavaScript for browser based application, this way you can reuse or plugin some of the existing methods directly into your automation code i.e. you are not reinventing the wheel again.

Summary

With the right planning and effort, right framework and right scripting language, automated functional testing can optimize software quality by verifying the accuracy and reliability of an application's end-user functionality in pre-production. Automated testing also creates new efficiencies which ensure that even complex enterprise applications are deployed on time and on budget.

References

Effective Software Test Automation: by Kanglin Li, Mengqi Wu, Oreilly Publication
Clean Code: A Handbook of Agile Software Craftsmanship by Robert C. Martin
Head First Design Patterns: A Brain-Friendly Guide by Eric Freeman and Elisabeth Robson
Test automation best practices by Sauce Labs: https://saucelabs.com/blog/test-automation-best-practices
Test Automation Tips and Best Practices: https://www.testingexcellence.com/test-automation-tips-best-practices