AGREEYA®

White Paper

Test Automation

"To be or not to be"

Testing is a form of Software Business Insurance. More coverage means less risk. And that's good news for everyone.

Author: Adtiya Jain

Overview

To be or not to be, that is the test. It's a dilemma that many test managers face when considering test automation. Test automation can be a mixed set of affairs for organizations in terms of ROI based on varying results from different companies. Given this set of mixed perceptions, many decision makers choose to avoid the long-term cost, and with it, the quality benefits that test automation can offer. Some consider test automation a "pill for every ill," greatly overestimating its role in their overall test strategy. Still others understand the boundaries of this "magic pill," take a marathon approach and build their entire test automation concept around the existing overall test strategy, reaping maximum benefits. But of course, nothing is perfect. Test automation, despite all of its glory, should only be used as a supportive role, and never as the primary – or only – activity in an organization's software improvement endeavors.

"Some consider test automation a "pill for every ill," greatly overestimating its role in their overall test strategy."

In this whitepaper, we address key concerns and challenges around test automation to help shed light on the truth behind this activity.

Myths about test automation that affect good decision making

- ### One can achieve 100% test automation.

The assumption or goal of 100% test automation is not just impractical -- it's impossible.

- ### Test automation is a one-time investment.

Often misconceived as a "fire and forget" tool, test automation involves and demands continuous investment in the essential three: people, process and tools.

- ### Test automation will simply do away with manual testing.

This assumption cannot be further from the truth. test automation at no point can replace the importance of manual testing or, for that matter, manual testers. Test automation combats repetitive, mundane tasks and helps speed up execution. Moreover, it is not a substitute for walk-throughs, inspection, good project management, coding standards and solid configuration.

- ### Immediate reduction in efforts.

With test automation, initial phases are often intensive, requiring a great deal of effort during setup. Once the scripts and the optimized test suite are ready, there is a considerable decrease in resources and effort required. Based on the maturity of the product, test automation requires and involves a certain degree of maintenance effort.

- ### 100% test coverage.

One cannot test anything and everything. The general philosophy is to primarily test what is important and repetitive.

- ### Anyone can automate.

When automating tests, cases are scripted; this involves coding and technological development knowledge.

- ### Running tests quickly and more often results in better software production.

It's not about speed and quantity, but generally, running and continuously re-evaluating tests are what result in better software.

What drives test automation? Is it cost or is it quality?

About 90% of organizations only undertake test automation as a response to their failure in completing testing in the assigned timeframe. Sometimes, despite best efforts, quality goals have not been met, resulting in multiple defects in production. Essentially, test automation then becomes a last-resort, giving management three alternatives to choose from:

(a) Hire more manual testers and grow the team organically.

(b) Grow the team inorganically by outsourcing some part of their test activities to a third-party.

"Adopting test automation without proper practices and process (both development & testing) in place will result in a negative ROI."

(c) Begin evaluating software test automation tools and strategy.

In the long-run, test automation not only helps organizations deliver quality software faster, but it enables them to do so at a fraction of the cost. This economy of scale is not possible with options (a) and (b).

Only 10% of organizations (those with matured test processes and practices) adopt test automation to help with cost reduction. Adopting test automation without proper practices and process (both development & testing) in place will result in a negative ROI.

What to automate and what not to automate?

Test automation is all about identifying the right things to automate – after all, 100% adoption is impossible. Ideal candidates for test automation include:

- **Sanity tests:** Basic functionality rarely changes, making sanity testing ideal for automation. This is particularly advantageous when handling multiple releases on a monthly basis.

- **Regression tests:** Save on costs and allow testers to solely focus on new functionality verification, avoiding repetitive tasks and eliminating risk in code change or bug fix issues.

- **Unit tests:** Inexpensive to write and maintain, unit test automation results in the highest ROI, providing value to the team multiple times per day.

- **Performance tests:** Manually performing controlled web application tests with hundreds or even thousands of users is quite the challenging feat. With test automation, users can simulate a large quantity of virtual users to check application load capacity. Management generally possesses very optimistic view of performance test automation – simply because they know that it is logically not feasible to do it manually.

- **Quick Tips:**

1. Automating an application whose UI changes frequently requires significant maintenance. By the time necessary changes have been made to the script, the UI will change again, rendering most efforts useless.

2. Manual testing is recommended when the lifespan of the application is short and not many releases are planned.

Advantages of test automation over manual testing

- **Improved accuracy:** Even the most seasoned and professional tester will make mistakes during monotonous manual testing. A tool performs the same task precisely every time.

- **Increased test coverage:** Complex test cases become easily executable with test automation, providing coverage that was previously impossible with manual tests.

- **24/7 test environment:** Test automation allows for scheduling unattended test runs 24/7 -- the equivalent of having three daily 8-hour shifts of manual testing.

- **Close alignment of test & development activities:** Test automation, when integrated with continuous-build software, provides a collaborative and continuous build-and-test environment. This is particularly of importance in an agile environment. As soon as a build/code is submitted by the developer, tests are run automatically, notifying the developer of any errors. This increases the efficiency and effectiveness of developers and testers alike.

- **Faster time to market:** Test automation provides testers the ability to execute tests 24/7 on multiple machines simultaneously, allowing complex and long regression testing to be executed in a fraction of time, and faster software delivery.

- **Increased reusability:** Done properly, test automation offers immense benefits to testers in terms of case reusability.

> "Test automation provides test engineers the space to be intuitive and allow them to focus their efforts on more rigorous tests."

There are many applications where basic functionality remains the same; take, for example, the case of e-commerce applications, which are 80% similar. The same test scripts, with little or no modification, can be used to test other applications, an option that is unavailable via manual testing.

Benefit Analysis

In this section, we will highlight the cost benefits that test automation has to offer in terms of real ROI. To showcase this, we have taken some assumptions (based on experience from several automation projects) which may vary by small percentage points from project to project.

- Number of test cases per test build: 500 (1 build every week)
- Average cost of a tool: $12,000 (single license)
- Hourly rate of a manual tester: $50
- Hourly rate of a test automation engineer: $60
- Manual Maintenance Effort required/week: 4 hours
- Machine cost: $400 (each)
- Time to automate a single test case (script development): 1 hour
- Maintenance Effort required/week: 8 hours
- Time to manually execute a single test case: 10 minutes
- Time to write a single test case manually: 30 minutes
- Resource training cost $10,000

**Considering two test cycles in a year, i.e. a test cycle of 6 months each. A build (500 test cases) is released every week. During the first year, we have only considered the last 6 months as the initial phase which will comprise of tool/process setup and tester training.*

Table 1

	1st Year (6 months/1* cycle)	
	Test Automation	Manual
Tool Cost	$134,000	NIL
Cost of scripting/test design	$30,000	12500
Cost of executing/running test	NIL	102000#
Cost of maintenance	$11,520 (24 weeks)	NIL
Total	$175,520	$114,500
ROI	-45.53%	

Investment on Test Automation = Tool Cost (10 licenses) + Training Cost (if training internal resources) + Machine Cost (10 machines)

 # The cost of manually executing/running tests will increase as a result of annual increases in test cycles and the number of weekly builds and/or the number of test cases corresponding with every build.

As seen in the table above, the difference and the cost advantage comes only in terms of test execution. With test automation, the testers will no longer be required to spend their valuable time on retesting something that has already been tested. Test automation provides test engineers the space to be intuitive and allow them to focus their efforts on more rigorous tests.

Table 2

	2nd Year (2 cycles)	
	Test Automation	Manual
Tool Cost	NIL	NIL
Cost of scripting/test design	NIL	NIL
Cost of executing/running test	NIL	$204,000#
Cost of maintenance	$23 040,	NIL
Total	$23 040,	$204,000
ROI	135%	

A note of caution, as time passes, new development techniques will become available and new sets of requirements will be requested. As a result, the application will incorporate a new set of components, which the tool may no longer be able to recognize. Consequentially, one will have to upgrade or replace the tool altogether, involving re-investments such as new licenses, tester training and script development. This could lead to a reduced ROI, potentially urging firms to revisit their automation strategy altogether.

Choosing the right automation tool

In the long run, selecting the right automation tool is the most important factor that will ultimately decide the success of your test automation project - poor choices unfortunately often result in project failure. During the culling process, current application

portfolios and future changes should be taken into consideration. The following are some basic features and factors to think about during the selection process:

- Ease of use
- Support for different types of testing
- Functionality
- Performance
- Automated scripting support using different scripting languages
- Results and reporting based on industry best practices
- Support for cross browser and cross platform testing
- Support for integration with 3rd party open & commercial tools (test management & continuous build environment)
- Object recognition
- Inbuilt exception handling or recovery mechanism
- Multi-user support
- Parallel execution
- Inbuilt data management

> "Too often, customers wind up paying more for products that sound new and exciting but were not required in the first place."

The list goes on, but ultimately, tool selection is one of biggest challenges to tackle before going for automation. It is very important to identify the requirements, explore various tools and its capabilities, set expectations and go for a Proof of Concept (PoC). While attractive pricing, branding and long lists of features are tempting, it is more important to first compare the tools and analyze its individual benefits before making the final decision. Too often, customers wind up paying more for products that sound new and exciting but were not required in the first place.

Test Automation Frameworks

Test automation tools are standard products which, depending on the users' needs, may or may not be a perfect fit. As with any other application, a need is felt to enhance or add new functionality to the underlying tool for ease of use. Test automation framework is a piece of software that sits on top of an automation tool, to provide additional or preferred functionality to the test automation engineer. Every test automation team/organization has their own style of working and standardized processes that they like to replicate with test automation. A test automation framework allows organizations/teams to modify or enhance the underlying tool, without actually making any physical changes to the tool, in order to suit their specific requirements and style of working. As an example, a test automation tool can only provide basic reporting and analysis, which a test manager may find to be inadequate. To overcome this, they may develop a program that based on the input received from the underlying tool can generate a more comprehensive report.

Various types of available automation framework options are:

- Keyword-driven
- Data-driven
- Hybrid
- BPT

The choice of a test automation framework, again, depends on the individual objectives that one would like to achieve vis-à-vis the underlying test automation tool and the overall test automation strategy.

A test automation framework provides certain added advantages with respect to test automation:

- Ease of use and reduced learning curve
- Faster test creation
- Easier maintenance
- Comprehensive reporting based on customized metricises

Conclusion

Test automation, if planned and used effectively, can be a powerful tool in an organization's inventory to help deliver quality software faster and at a fraction of cost.

But in the end, it's important to note that test automation should only be considered as a special set of software that works to verify the state of another piece of software. Used properly and for its anticipated purpose, test automation can lead to better results and overall success all around.

About AgreeYa

AgreeYa Solutions is a global provider of software, solutions, and services focused on deploying business-driven, technology-enabled solutions that create next-generation competitive advantages for customers. Headquartered in Folsom, California, AgreeYa is a growing and dynamic organization with 15 offices in 8 countries employing more than 1,100 professionals. Over the last 15 years, AgreeYa has worked with 200+ companies ranging from Fortune 100 firms to small and large businesses, delivering solutions for variety of industries including telecommunications, BFSI, healthcare, high-tech, manufacturing, utility and government. AgreeYa's software portfolio includes SocialXtend (intranet and enterprise social collaboration), VDIXtend (Desktop-on-Cloud), and Onvelop (enterprise mobility productivity suite). As part of its solutions and services offerings, AgreeYa provides intranet and enterprise collaboration on SharePoint, cloud and infrastructure, enterprise mobility, product engineering, application development and management, independent software testing, and staffing (IT and risk/compliance) solutions.

For more information visit us online: visit www.agreeya.com. or Follow us on twitter @AgreeYaGlobal

About the Author

For more than a decade, Aditya Jain has worked on a multitude of testing projects from inception to operation. Specifically, his vast experience includes significant contributions to the establishment of the Test Centre of Excellence (TCoE) for several large enterprises.

Possessing strong working knowledge of multiple test automation tools, including open source and commercial, Aditya has guided numerous clients in implementing 'NextGen' test automation solutions, helping them leverage the cost benefits of these tools.

Aditya holds a bachelor's degree in engineering from the College of Engineering, Roorkee.