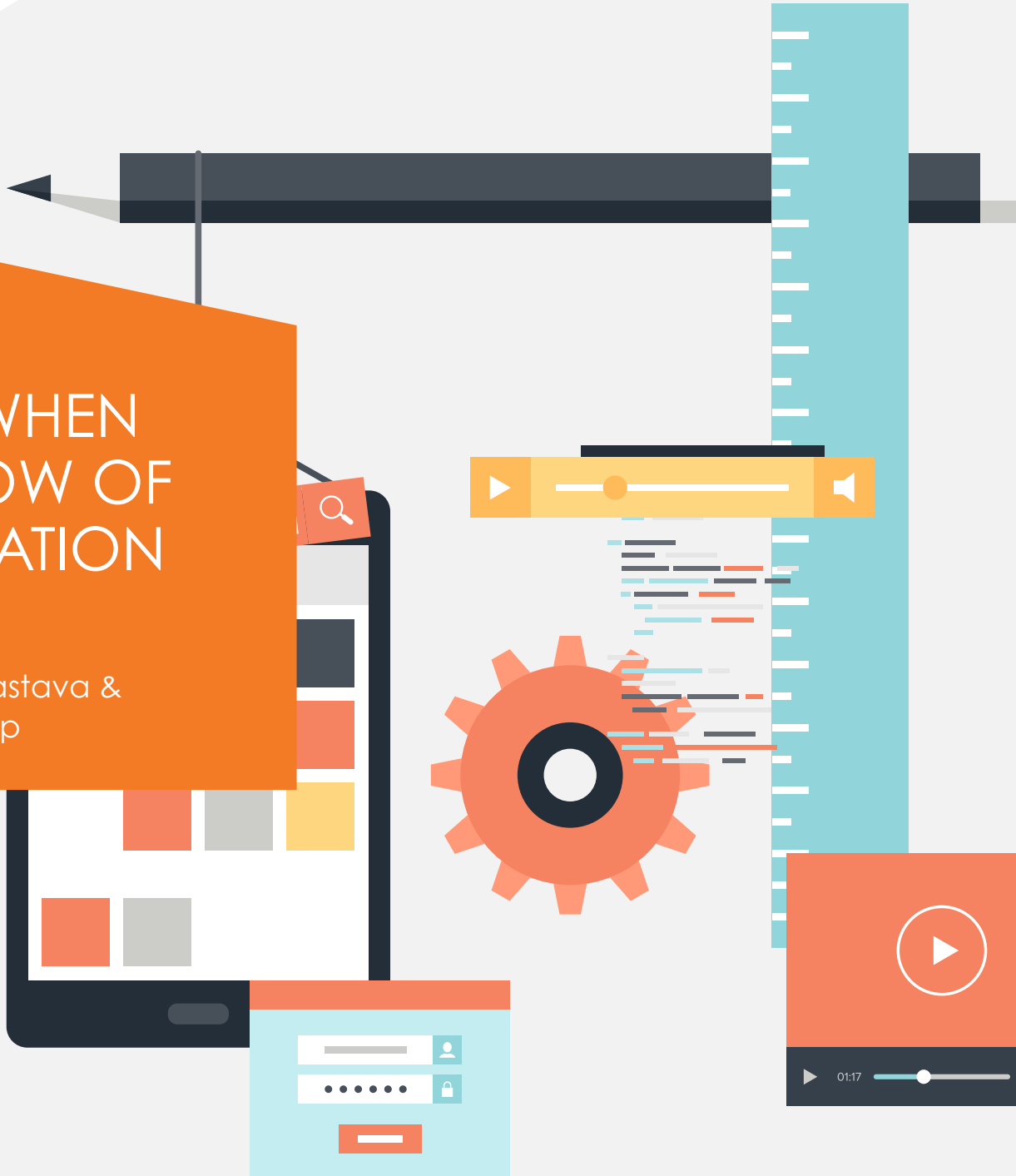
TO THE NEW

THE WHAT, WHEN AND HOW OF AUTOMATION TESTING

By Sarthak Srivastava &
Keshav Kashyap



May 2016

CONTENT

1. INTRODUCTION

There are a number of challenges faced in delivering a quality software product after comprehensive testing. A few of them are: executing large number of test cases, testing diverse legacy applications, managing time constraints in software development life cycle and, of course, delivering a first-rate quality product. To deal with all the above situations, we have to adopt a robust automation testing framework that can handle all the issues with an appropriate strategy and approach. As easy as it sounds, it is a challenge to identify the right approach for test automation.

Often it happens that the automated testing done on software applications does not produce cost-effective results, as the adopted testing approach might be an inappropriate one. This paper sketches a systematic approach taken while designing/ adopting an automation framework by identifying when and how to start automation in a testing project. This paper also describes the best practices for efficient and effective techniques of test automation, which helps in better quality of software in a short span of time as compared to manual testing. This paper neither recommends nor advocates test automation in general, but aims to highlight the benefits and risks involved in test automation, points out the differences between manual and automated testing, defines criteria for selection and guidance for validation of right test automation tools, right mode of automation tools and best-fitted automation framework.

2. WHAT IS AUTOMATION TESTING?

Every tester, in his/her daily job routine, faces the laborious task of executing a large set of test cases before every release or while doing regression testing. To add more on to this painstaking activity, these test cases are supposed to be executed across various platforms and numerous devices. Factors such as confirmation bias and human error come into play. At some point, a person could execute a test manually and miss an important step, thereby failing to identify a bug in the application being tested. That is when Automated Testing saves the day!

Automated Testing saves a lot of time allowing teams to speed up the overall testing process. This is accomplished by creating intelligent scripts that can be used to test the expected behavior of an application across many different platforms at the same time. Although it isn't a substitute for the human capacity to design and execute a well thought out, high quality process, it promises to find more defects with lesser efforts.

3. WHY AUTOMATION TESTING?

Test Automation is suddenly in high demand. It's a must for every new project. The mystique of automation is so strong that it seems that there is no longer a need to write test cases, run manual testing, or do impact analysis. The following reasons explain why Automated Testing is a salient need:

Behavioral Analysis

Automated tests don't check that if a code is doing the right thing; they document the code in such a way that anybody can check if it continues to behave in the same way after some changes have been made in the code. This has some very interesting implications. The primary one is that the tests specify exactly what your business does. If you don't validate the consistency of behavior that means behavior is not important to your business.

Time/Cost Savings

Writing automated tests can take more time than manual test cases. However, considering these tests are run multiple times, the marginal work (i.e. cost/time) to execute automated tests is lesser by several orders in magnitude. Thus, automated tests are cheap to run and facilitate system changes over time.

Documentation

There is no truer way to know how a system works than its tests. Any other documentation is usually out of date the moment it's written, but tests (at least those that pass) reveal how things actually work.

4. WHEN TO START TEST AUTOMATION?

A test automation program should be considered if the key program factors indicate that the overall development program is not meeting expectations and there are no cost-effective alternatives to test automation. The key indicators from a program objective would be:

- Quality objectives are not being met
- Defect escape velocity into production is deemed unacceptable
- Target deployment velocity is not being met because testing is perceived as a bottleneck
- Testing is not being completed within the assigned timeframe
- When there is a need to run Regression/Sanity/Smoke test suite multiple times
- Testing single functionality with multiple data sets

- When there are a number of test cases under one test-suite
- Increase in test coverage

5. PRE-REQUISITES

In order to achieve successful automation, it is very important to have certain things set beforehand. Automation is not a sure shot formula to a quality product unless it is carried out in the right manner and at the right stage. Below are the few pre-requisites that one should keep in mind before starting Automated Testing:

- **Product or Application should be stable enough:** The first thing that needs to be ensured is that the product/application is fairly stable in terms of functionality. Even if it is slated to incorporate new features, the new features should not disturb the existing functionality.
- **Scope of the automation should be defined:** Before setting out to automate the testing of your application/product, it is essential to define the intended coverage of the automation tool.
- **Identify which test cases need to be automated:** Automation suite should be looked upon as a baseline test suite to be used in conjunction with manual testing, rather than as a replacement for it. It should aim at reducing the manual testing effort gradually, but not doing away with manual testing altogether. It needs to make sure that the following kind of test cases are eliminated from the scope of automation:
    - Test cases that are long and complicated, and require manual inspection/intervention in between.
    - Test cases that take tremendous amount of time in automation and are difficult to ensure re-usability even if they are automated.
- **Right tools must be decided:** There are lots of automation tools available in the market. A careful effort has to go into deciding which tool would be most suitable for automation to test your product/application.

- **Right mode of automation must be defined:** If you are going for GUI test automation, then points worth considering while making a sane decision are:
  - Record and Playback approach for creation of test scripts and test suites is easy to develop but difficult to maintain.
  - Error recovery cannot be incorporated by just recording a test script.
  - In data-driven tests, achieving reusability of test scripts will be very limited. Creation of test data and integration of the same with test scripts is time consuming. When a function is coded for the same purpose with data input file, maximum re-usability and ease is ensured.

6. HOW TO GET STARTED WITH TEST AUTOMATION?

Automated testing will shorten your development cycles, avoid cumbersome repetitive tasks and help to improve software quality, but how do you get started? These, step by step, guidelines give a successful foundation to start improving your software quality.

Selecting the Right Tool

Selecting an automated testing tool is the first step and the most important aspect of test automation. There are a lot of automated testing tools in the market, and it is essential to choose the tool that best suits your overall requirements.

Factors to be considered while selecting an automation tool are:

- Failure cost.
- Manual as well as automation test execution and analysis time.
- Automation tools acquisition, assessment and analysis time.
- Tools training costs & test machine cost.

- Test machine environment activity period
- Test case development and debugging time
- Automated test cycle analysis and maintenance time
- Anytime execution i.e. CI Integration support and deciding right mode of automation

- **Recording or Scripting:** After deciding the tool comes the need to decide on the mode of automation i.e. recording or scripting. We can choose the Record & Play Back method or Descriptive Programming after analyzing the features provided by any automation tool and the limitations of the product/application. Test automation tools assist in recording or scripting test scripts, automatically executing test scripts and recording test evidences. There are various methods or frameworks (described in the next section) to automate an application testing which can be employed as per the need of the application. The following basic methodologies are used for automating an application:

- **Record & Playback**
  - Tester manually records each step, and then plays back the recorded script
  - During recording, the automation tool reads the user input and generates the scripts

- **Scripting**
  - Test case logic is defined in test scripts
  - All user actions are converted to reusable actions
  - Any new scripts can make use of these actions

Selecting Right Automation Framework

A test automation framework can be defined as a set of processes, standards and interactions between the components in which scripts are designed and executed.The classification of test automation frameworks can be broadly divided into:

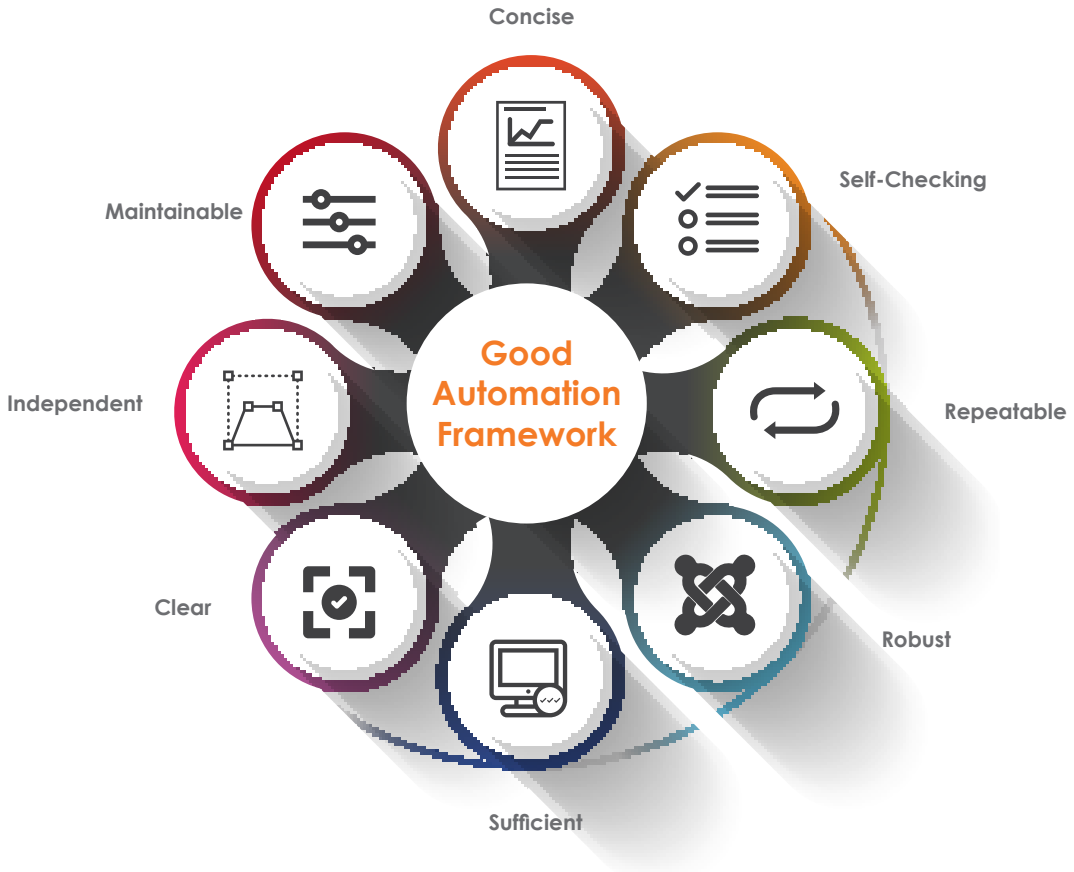
- **Data-Driven Automation Framework:** Test input and output values are read from data files (ODBC sources, CVS files, Excel files, DAO objects, ADO objects). Test flows and navigations are coded into test scripts.
- **Keyword-Driven Automation Framework**: Allows the development of data tables and keywords, independent of the test automation tool in use. In a keyword-driven test, the functionality of the application-under-test is documented in a table as well as in step-by-step instructions for each test.
- **Hybrid Test Automation Framework**: Combination of the above techniques, pulling from their strengths and trying to mitigate their weaknesses. Allows data driven scripts to take advantage of the powerful libraries and utilities in a keyword based approach.

After settling on the right tool and right automation framework based on the needs of the product, a software tester also makes sure that the automation framework is handling scripts and data separately, creating libraries, following coding standards, offering high extensibility, providing less maintenance efforts and ensuring script/framework version control. Besides this, a tester should also keep in mind that any framework should have the following basic features:

- **Concise:** As simple as possible but no simpler.
- **Self-Checking:** Reports its own results; needs no human interpretation.

- **Repeatable**: Test can be run many times in a row without human intervention.
- **Robust**: Test produces same results now and forever. Tests are not affected by changes in the external environment.
- **Sufficient**: Tests verify all the requirements of the software
- **Clear**: Every statement is easy to understand
- **Independent**: Each test can be run by itself or in a suite with an arbitrary set of other tests in any order.
- **Maintainable**: easy to understand, modify and extend.

Graph 1: Basic features of Automation Testing



Writing Test Cases

When you have the automation framework ready to be in use, test case designing is started. Writing automated test scripts requires expert knowledge of scripting languages. In addition, a tester should validate test cases manually first, then document the identified test cases, and then automate.

- **Identify Test Cases to Automate**

It is impossible to automate every bit of testing. The first step towards successful automation is to determine the test cases that need to be automated. We can get the most benefit out of the automated testing efforts by automating:

- Repetitive tests that run for multiple builds.
- Tests that are highly subject to human error.
- Tests that require multiple data sets.
- Frequently-used functionality that introduces high risk conditions.
- Tests that are difficult to perform manually.

- **Create UI-Resistant Automated Tests**

Automated tests created with scripts or keyword tests are dependent on the application under test. The user interface of the application may change between builds, especially in the early stages. These changes may affect the test results, or your automated tests may no longer work with future versions of the application.

If you provide unique names for your controls, it makes your automated tests resistant to these UI changes and ensures that your automated tests work without any need to make changes to the test itself. This best practice also prevents the automated

tests work without any need to make changes to the test itself. This best practice also prevents the automated testing tool from relying on location coordinates to find the control, which is less stable and breaks easily.

- **Identify Common Steps and Convert them into Functions**

At the outset, the steps common amongst different test cases should be identified and converted into functions. Such functions can be placed in a common file, from where they can be called by different test cases by passing suitable parameters as per the need.

Associating Test Data

Good test data is extremely useful for data-driven testing. The data that should be entered into input fields during an automated test is usually stored in an external file. This data might be read from a database or any other data source like text or XML files, Excel sheets, and database tables. A good automated testing tool actually understands the content of the data files and iterates over the content to test all the possible combinations.

Executing & Reporting

Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significant reduction in integration problems and allows a team to develop cohesive software more rapidly.

After the automation suite is run, the concluding step is to check and analyze the results produced. Different tools offer different ways of report generation, which let the testers analyze the progress and state of the product in terms of its quality.

The report shown after running a test suite provides a general overview of how many of the test cases have been executed successfully, have failed or have been blocked. Each executed test case and all its child modules can be analyzed in depth by expanding a particular test case.

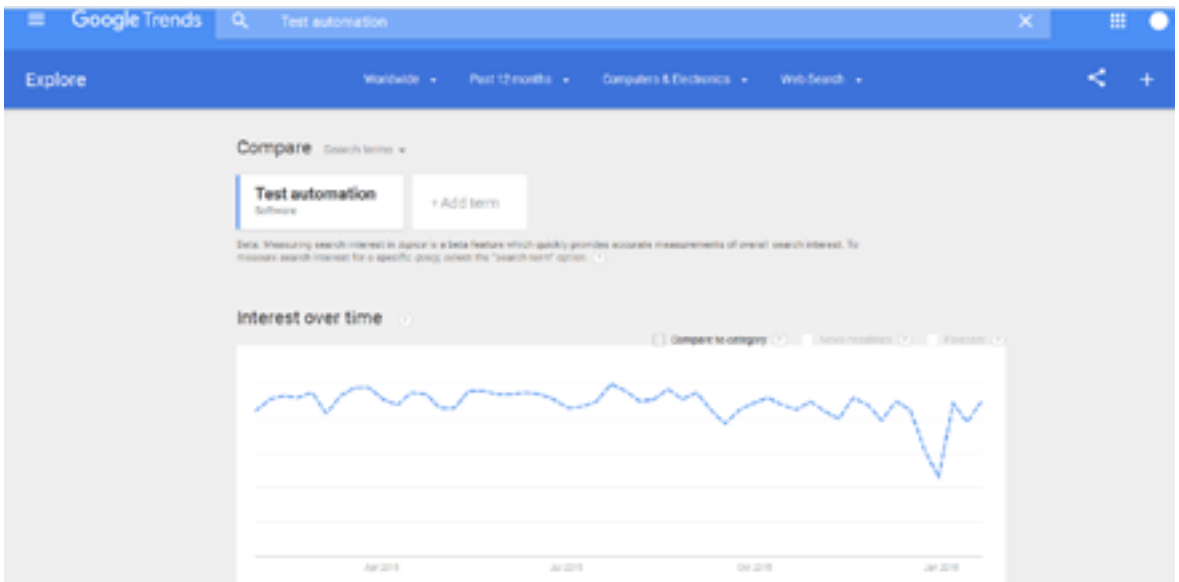
In addition to that, a test suite report also informs you about the following:

- System information like execution time, machine name, operating system, screen dimensions, language, duration, total errors, total warnings
- Global parameter values

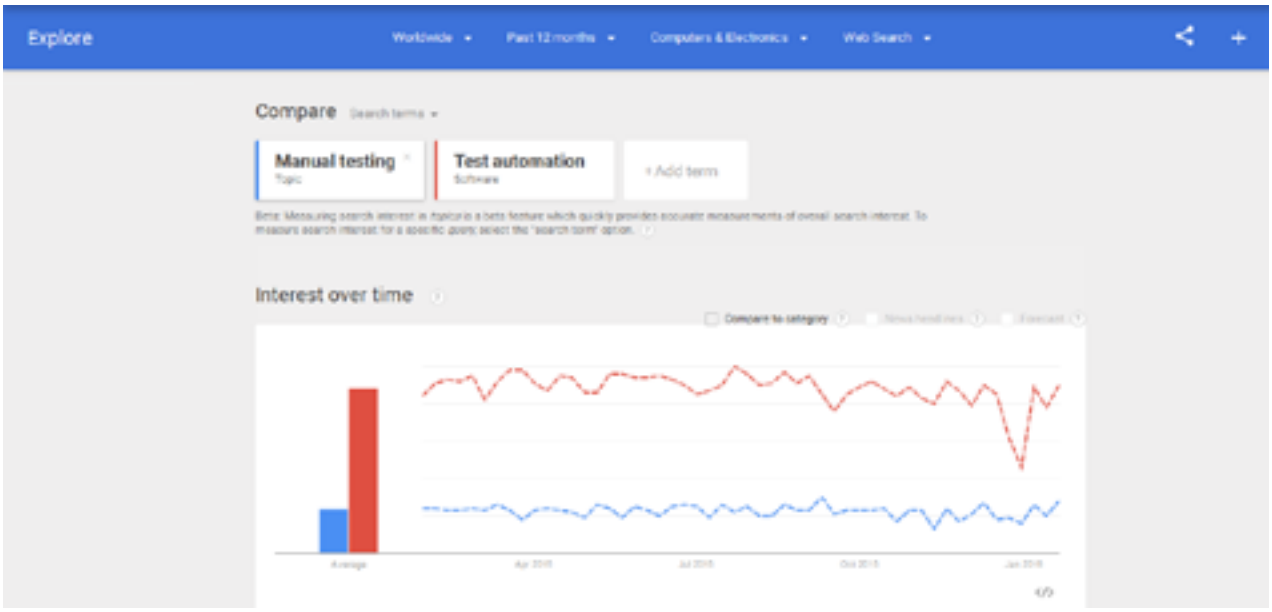
7. LATEST TRENDS

Below are few automation trends which have been noted in the industry in the recent times.

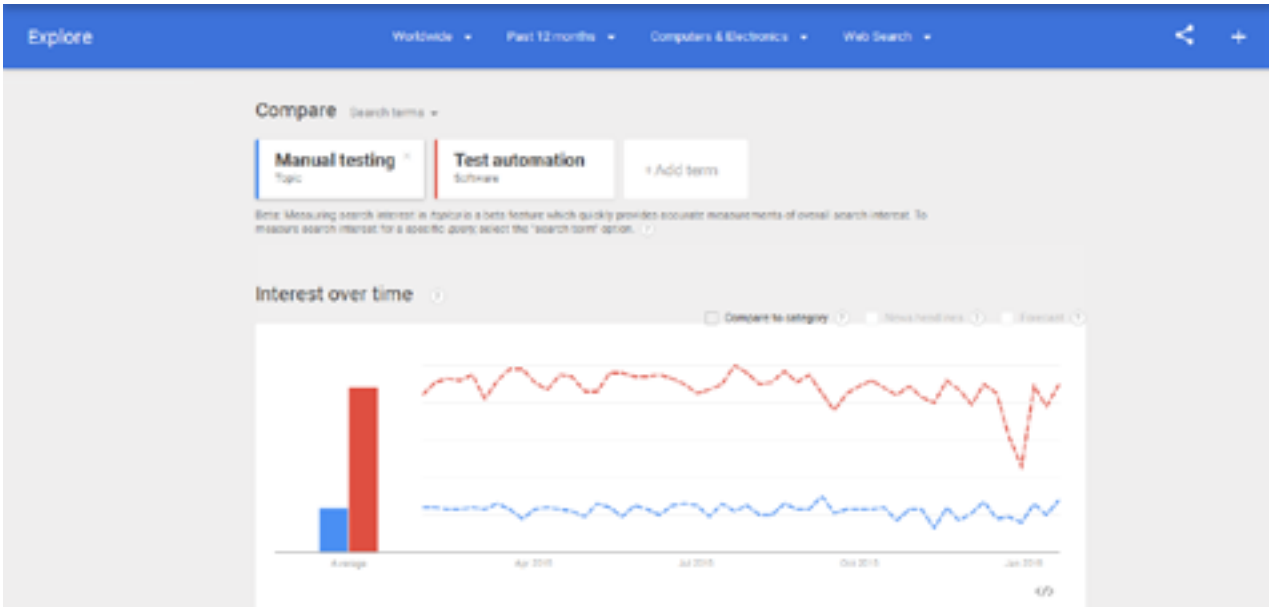
Graph 1: Automation TestingTrends



Graph 2: Manual vs. Automation Testing Trends



Graph 3: Manual vs. Automation Testing Trends



8. THE OTHER SIDE OF AUTOMATION TESTING

When we talk about the other or secondary side of Automation Testing, we found some interesting facts that should also be considered:

- Humans can notice bugs that automation ignores.
- Tools are not limited to looking at what appears on the screen; they can look at the data structures that lie behind it.
- Being a tester it's annoying to discover a bug in manual testing and then find you can't reproduce it. Probably you did something that you don't remember doing. Automated tests rarely have that problem (though sometimes they're dependent on parts of the environment that change without you noticing it).
- An automated test suite can explore the whole product every day. A manual testing effort will take longer to revisit everything. So the bugs automation does find will tend to be found sooner after the incorrect change was made.
- After a programmer makes a change, a tester should check it. This includes re running a stock set of old tests, possibly with variations. It certainly includes devising new tests specifically for the change. Sometimes due to miscommunication testers may not be informed a few changes. With luck, some automated tests will break, causing the testers to notice the change, thus be able to test it and report bugs while they are still cheaper to fix.

9. CONCLUSION

In a world that is becoming more and more flexible and agile, automated test cases execution should be considered for applications used in regulated environments. While the initial effort seems rather high, an automated test suite can provide numerous advantages. Although neither all benefits nor all negative aspects are quantifiable, it is worthwhile to calculate when and how to start and strategize test automation.

Automated software testing is the best way to increase the effectiveness, efficiency and coverage of your software testing efforts. Once automated tests are created, they can easily be repeated and they can be extended to perform tasks impossible with manual testing. Because of this, savvy managers have found that automated software testing is an essential component of successful development projects.

10. REFERENCES

Bach, J. "Test Automation Snake Oil", 1999. Published in Windows Tech Journal, 10/96, and proceedings of the 14th International Conference and Exposition on Testing Computer Software

Hoffman, D. (1999) "Test Automation Architectures: Planning for Test Automation," Quality Week 1999

Kaner, C. (1997) "Improving the Maintainability of Automated Test Suites,"Quality Week 1997

Kaner, C. (1998) "AvoidingShelfware: A Manager'sView of Automated GUI Testing"

11. ABOUT THE AUTHORS

Sarthak Srivastava
Senior Engineer, QA

Sarthak has rich experience working in startup environments and he is passionate about exploring new technologies. He has worked on BDD and agile frameworks with an excellent knowledge of APIs, Databases, Desktop automation, front-end and back-end automation testing.

Keshav Kashyap
Engineer, QA

Keshav is an expert automation testing engineer with immense experience in web and mobile based tools and technologies. As an ISTQB and Oracle SQL certified tester, he is involved in test development of web and mobile applications throughout their various stages. He is enthusiastic about building stable software systems that deliver values.

12. ABOUT TO THE NEW

TO THE NEW is a digital technology company that builds disruptive products and transforms businesses. We leverage the power of experience design, cutting-edge engineering, cloud and analytics led marketing to enable digital transformation.

Our passionate team of 750+ people includes passionate technologists, digital analytics experts, video specialists and creative mavericks who have transformed businesses of more than 300 companies spread across 30 countries worldwide. We take pride in our culture which is driven by passion for making an impact through technology.

13. EXPLORE MORE RESOURCES

A Pragmatic Approach to Performance Testing

DOWNLOAD WHITEPAPER

Software Testing in Agile Environment

DOWNLOAD WHITEPAPER

TO THE NEW

info@tothenew.com
www.tothenew.com

LETS CONNECT