Test Automation: Enhancing Software Quality and Efficiency

Alexander Vaptsarov
Computer Science and Artificial Intelligence
Technische Hochschule Ingolsatdt
alv3651@thi.de

ABSTRACT

Software testing has become an pillar of importance in modern software development due to the demand for more complex and higher-quality products. With that said, the best teams of software testers face the pressure of time and effort needed to test their product entirely. For this reason, test automation offers significant improvements in efficiency and quality, when it comes to the continuously growing complexity of Software Development. This paper tries to unveil the fundamental aspects of test automation, including pros and cons of the necessity, specific techniques and practical impact. Through detailed examples and practical approaches, it illustrates how test automation can effectively improve the quality, complexity and time required for enhancing the software development processes.

I INTRODUCTION

Definition of Test Automation: Test automation refers to the implementation of specific software tools or scripts to control the execution of tests and compare actual outcomes with expected results. This process helps verifying that the software performs as expected without the need of manual intervention.

Historical Context

Software testing can be traced all the way back to the 50s of the last century. Back when programmers and developers were using many different methods to examine their code for faults, errors and security vulnerabilities, but up to this time it was still all done manually which is, as it might be expected a very time consuming and tremendously overwhelming task. Nevertheless the first fully committed and highly devoted software testing team has been formed in the late 50s at IBM. The head of this team was the famous computer scientist and author Gerald M. Weinberg, who was a respected and influential person in computer science and software development field. He was responsible for testing the operating system for the IBM 704 computer, one of the original commercialized mainframe computers at that time. [1]
Computer scientist at the time were aware of the potential for the concept of test automation and even in the early days of computing traces of this idea can be found, but not in the same manner as we are familiar with today. The first real implementation of an automated test took place many years after that. One of the original examples, preceding the current times in the field of test automation in the current age is the Automated Test Engineer (ATE) system developed by IBM in the 70s. The establishment of this system which was designed to automate the testing of mainframe software applications, amounted to the significant breakthrough and set a prior exemplar for future test automation systems we are familiar with today. [2]
Nowadays as standardized approach for test automation and testing in general has been mainly developed by the ISTQB Board. This stream focuses on many different parts of software testing and members of that board has been representing the qualifications and guidelines on Software Quality and Testing!

Importance in Modern Software Development

In recent years integration of technology has become a considerably bigger part of each and every aspect of our daily life. We can notice software in various fields, like for instance financial applications but also medical software which has considerably more priority within our domain of Software Development and also you can imagine other applications, like web-technologies and even games. In today's fast-paced development environments, the need for high-speed and reliable testing is of much greater importance. Test automation allows for continuous integration (CI) and continuous deployment (CD) pipelines that integrate into every stage of the development cycle by enabling frequent and consistent testing but with the added benefit of cost reduction, time required and complexity of the tests. On the other hand test automation is keeping the efficiency and usability and ensuring high-quality software releases.

II PROBLEM STATEMENT

By focusing on more intricate test scenarios and exploratory testing cycles testers can improve the overall software quality. But some of the testing techniques are prone to errors and also slow down the testing process especially when it comes to large and complex applications.

Challenges in Manual Testing

Manual testing is inherently a time-consuming and error prone technique. When software systems grow in complexity and size, manual testing becomes almost useless due to scalability issues and simply the volume of repetitive tasks required.

III BENEFITS OF AUTOMATION TESTING

Automation addresses a large number of challenges that manual testing poses, especially in large and complex software environments. Automation testing refers to the use of specialized software tools to execute pre-scripted tests on a software application before it is released into production. This method of testing offers significant advantages over traditional manual testing, including increased efficiency, improved accuracy, enhanced coverage, and the reusability of test scripts. These benefits not only streamline the testing process but also enhance the quality and reliability of the final product.

Increasing Efficiency

Automation reduces the time and effort in the testing process.
When we are dealing with the development of software, balancing time and cost efficiency is especially important. If you are able to increase efficiency, it should be your number one priority and this is the main motive for automating labor-intensive tasks. Automation allows for the execution of those mundane test cases without a human stepping in, which in turn makes both running exhaustive tests significantly decreased in time and the effort required to execute a lot lower. The technique of automating tests is even more beneficial when the software we deal with has large and complex functionalities, or when it requires multiple integrated components, where manual testing can become exceedingly time-consuming and repetitive.

Improving Accuracy

Automation reduces the human error.
People are more susceptible to errors that machines, we are all humans after all. When we deal with such mundane tasks like manual testing, this is especially true. Testers can without no doubt leave unnoticed or misinterpret many aspects and features of the software and testing execution cycle in general. This leads to a lot of inconsistencies in test process and also many other problems. Automation testing diminishes these risks by performing each test in exactly the same way every time it is run, ensuring high levels of precision and consistency in the operation.
Automated tests are written and ran based on the exact requirements and specifications defined by either the developers or the testers. The ambiguity that can usually only arise in manual testing can in this way be eliminated, leading to more reliable and much more predictable outcomes. Furthermore, automated testing tools can in many cases offer built-in means of error detection that can automatically identify defects with ease, which in turn enhances the capability of the testing software.

Enhancing Coverage

Automation increases the scope and depth of tests.
The extensiveness and scope of exactly how well the software performs on tests, allows developers to focus on and conduct even more in-depth and detailed examinations of the application on hand and the automation of those tasks can significantly increase the test coverage. Automated tests can without a problem check the stability and limits of an application and stress the system under heavy load, thus measuring the performance, and simulating thousands of users at once interacting and connecting with a network application for instance. These kind of task are practically impossible with the original approach of manual testing.
Moreover, the idea of automation becomes even more crucial when dealing with complex test cases that verify the integration of multiple interconnected components within the software application on hand. Precisely this coverage helps in identifying defects in the software at an earlier stage rather that after all the components are already implemented, because if there is a problem at that stage, then the solution to it will be much more extensive and complex and will be a lot more time-exhaustive and harder to fix. This reduces the cost and effort associated with fixing bugs post-release and the main reason for this is the complex integration of the components. Another very important addition to the robustness of the testing procedure is the idea to be able run tests on multiple hardware or cross-platforms configuration, making sure that the application performance in predictable across different user environments.

Reusability of Test Scripts

Importance of reusable scripts in maintaining consistent testing standards
Another reason for using automation testing is it's significant advantage of the ability to reuse test scripts. Once a test script has been created and established, different versions and components of the application can be simultaneously tested, especially when some aspects of the application changes if we have to manual test we are back at the repetitive action of testing the same thing manually over and over again. This not only saves time, but also ensures that there is consistency in testing cycle. When we look at it like this, reusing scripts can become a real valuable asset for testing future projects or parallel versions and components. This makes it really easy to maintain a software, as scripts can be swiftly adjusted for use of regression testing always when we change the source code. This contributes to maintaining a high level of quality throughout the software development cycle.

IV DISADVANTAGES OF AUTOMATION TESTING

With all these benefits that automation testing offers over traditional manual testing, just like everything in life, there are also certain things that testers must consider before implementing any type of scrips. Challenges and limitations primarily relate to the fact that test automation requires carefully consideration for a correct and complete setup and those initial setup cost can be relatively high compared to manual testing. There are also a lot of other things to be done for the maintenance of test scripts, and of course there is a large amount technical skills that go into the effective use of automation tools. Automation testing requires these drawbacks to be thoroughly understood for a balanced testing lifestyle.

Expensive Setup

Higher Investment for automation software
Setup costs for automation testing can be quite real, especially for small to medium-sized enterprise businesses (SMEs). [3] The purchase of tools, setting up the necessary infrastructure, and even as simple as training and paying for workforce to handle these tools are only some of what the expenses this cost includes. Usually the most widely used software requires significant investment and for that reason automation tools vary broadly in terms of pricing. [4]

In addition to the costs of software and infrastructure, resulting costs also play a consequential role. This includes the time required to set up and configure the automation environment, which can slow up the development cycle. As employees need to understand not only how to use the tools but also how to write and maintain complex test scripts, so the cost of training is another factor.
This whole process can be a considerable financial expense, and the return can not only take time to be realized, but also sometimes lose more than what was actually investment and promised, particularly when the automated tests are not used frequently. For that reason testers need to be very careful and knowledgeable when not to use automation. Because sometimes it is just not desirable to take time to create automated scripts that are only going to be executed once. There is time and place for everything and sometimes it does not make sense to invest in something that doesn't require that level of complexity. [5]

Maintenance of Test Scripts

Challenges in keeping test scripts updated with changes in the software.
We already talked about cases where test automation is undesirable. For example when dealing with simple application or when the software at hand consist only of single component the use of automation script bay not be the best idea. We already talked about how software development is a ever changing field. The continuous need for updates can consume a lot of resources, especially time and effort that takes developers to satisfy the requirements. Maintaining the test cases can be a intimidating task when software grows with time. Each change in the application's source code can require an update in the test scripts to ensure each feature remains effective, relevant and error-free.
In agile methodology the significance of keeping a maintained tests becomes an even bigger challenge. This is when continuous integration and continuous deployment (CI/CD) [6] principles are to be complied with. Here, the software is a subject to a lot of changes, leaving no choice but to almost constantly update the test scripts. If this is not managed properly, the test scripts will become unusable, which can lead to problems developers will not know about.

Technical Skill Requirements

Skilled technicians to create and maintain automation scripts
Effective implementation and utilization of automation testing require a high level of technical expertise. Testers need to be proficient in programming to write and maintain complex test scripts. They must also have a deep understanding of the software's architecture to create effective test cases. This skill set is more advanced than what is typically required for manual testing.
The demand for such skills can lead to challenges in staffing. Skilled automation testers are in high demand and can command high salaries, increasing operational costs. Moreover, the learning curve for mastering automation tools and scripting can be steep for existing employees, requiring significant training and practice. This can lead to a gap in productivity as personnel adjust to the new tools and techniques.
In addition, the reliance on technical skills can lead to issues with scalability and flexibility. Small teams may find it difficult to manage large suites of automated tests without adequate manpower. Similarly, businesses may find themselves dependent on a few key individuals who possess the necessary technical knowledge, creating a risk if those individuals leave the organization. *The need for skilled personnel to develop and maintain automation systems*

V TOOLS & FRAMEWORKS

Choosing effective automation tools
Achieving the goals of test automation can only be realized with the appropriate tools and frameworks. A differentiation between both terms is needed to avoid any misunderstanding, Tools are software products that can create something on their own while frameworks are in a supporting role where they can be used to create a work product or help in reducing the time and effort needed to automate a task. [7] According to the International Software Testing Qualification Board (ISTQB), [8] tools that can be used are divided into the following:
Management tools – increase the test process efficiency by facilitating management of the SDLC, requirements, tests, defects, configuration
Example: **DOORS, Testcollab** Static testing tools – which can be used to check the logic of the code without executing the code itself
Example: **SonarQube** Non-functional testing tools – which allow the tester to test non-functional aspects of the code like performance and security.
Example: Meter End to End tools: these are tools that test the application from start to finish, testing all the features and interactions on their way
Examples: **Selenium, Cypress** DevOps tools – are becoming increasingly important in building a continuous integration and continuous testing pipeline
Example: **GitLab** Frameworks on the other hand are developed usually for a specific use case but here we will mention Examples of most popular ones that can be used in multi tasks:
Webdriver IO Which is framework for testing web apps

and enable Robot Framework Which is a generic framework used for Robot process automation (RPA)tasks automation, acceptance testing and acceptance test driven development
Pytest It is a framework incorporating python language and can be used to develop test scripts in python fr unit tests, integration tests, end to end tests and functional tests.

Advantages of using framework

Code reusability and reducing the time needed to develop test automation scripts Minimal manual interference as most test automation frameworks are already established with rules and guidelines lower maintenance costs as frameworks does not usually require maintenance efforts Types of testing frameworks: Linear automation framework In these frameworks, tester does not need code writing skills as he can manually record the steps such as navigation of the browser, and user inputs, and then play back the whole test.

Modular-based testing framework

In this framework the system under test is divided into separate and isolated modules and each module gets it is own script to test it and then all the test scripts will be combined together to test the whole system. Library architecture testing frameworks Like the Modular based testing frameworks but similar tasks within the scripts are identified and later grouped by function, so the application is ultimately broken down by common objectives.

Data driven framework

Some times tester will need to repeat the same tests with different data and this framework solve this problem where it isolate the data from the script itself. In this approach we create a data source like a data base or excel sheet and then link it with the test script being used, that way if the data is needed to being changed then we only change the data source.

Keyword-driven framework

Here each function of the application under test is represented in a table with a corresponding instruction of each test that needs to be run. In a similar fashion to the data-driven framework. Here keywords are added in an external data table making them independent from the automated testing tool keywords in this context represent the actions being done like login or adding a user name.

Hybrid testing framework

a hybrid framework is formed by combining two or more of the previously mentioned frameworks to combine multiple features in one framework.

Selecting the test tool

As there is a wide variety of test tools that can suit multiple purposes choosing the right tool can be a hard task however, we will mention a few aspects to consider in order to make the selection process easier.

Ease of use

Does this tool require advanced technical knowledge? Or for example is it a code free tool? This aspect have to be taken into consideration in parallel with the testing team knowledge and experience.

Adoption time:

How much time is available for the team to understand and get familiar with the tool? Is it an ongoing support? Or obsolete tool? Using a tool that have an on-going support can be very helpful in case the team encountered a problem during the project. Open source or closed source tool? Here the factors that determine our choice are usually cost, security requirements and contractual agreements.

Report generation and logging

As documentation is a vital aspect of testing a tool that provide a detailed report of the test run would be of great help. Depending on the software under test, logging the inputs and outputs might also be an important aspect needed.

VI CASE STUDY

The growth of software application both in scale and complexity makes it essential for tests to be automated rather than being implementing manually. The case study from Daniela Crișan who is a Senior Tester at ISDC, [9] that is going to be explored in here illustrates really how essential test automation is for modern software development field.

Key Components for a successful Test Automation

Daniela states: *"Successful test automation needs both ingenuity and perseverance."* (Daniela Crișan, Senior Tester, @ISDC) [9]

For the most part the factors that really turn test automation into the success it really is today are the management support and robust technical architecture. Firstly, management includes aiming for realistic goals and allocating sufficient resources to ensure that the planned investments (ROI) or goals are met.

Secondly, it is of crucial importance that the technical architecture of the test software (testware) are also met. Here abstraction is an important pillar to be able offer the necessary efficiency. This reduces both the maintenance cost of the testware and the cost associated with the automation processes. It also details the phased approach to implementing test automation. The pilot phase was the first one, giving clear definition of the strategy and approach. Successive phases released more and more scope of automated tests-from environment checks, all the way to full regression suite of tests. Continuous integration tools such as Team City, but also workflow based on Scrum clearly indicate structured and iterative approach to automation.
Conclusion

As has been well documented in this case study, the road to successful test automation is a hairy one, calling upon both creativity and persistence. Indeed, it wasn't all technical solutions but strategic management and calculated planning that yielded a successful implementation. Many benefits have been realized, including faster release velocity, increased efficiency for testing resources, and improved software quality in general. This case shall be a nice guide for organizations looking to either start or refine their test automation strategies.

VII FUTURE TRENDS IN TEST AUTOMATION

AI and Machine Learning

One of the primary problems of test coverage and efficiency can be solved by integrating AI into test automation. AI powered tools can identify patterns and predict potential problems by analyzing tons of data and past user interactions. Without the need for developers to manually create test cases it allows testers to extend their focus unexplored errors. Machine Learning algorithms can predict based on train data and predict how to optimizations test cases, making this process more efficient.

VIII CONCLUSION

Test automation has proved to be one of the best tool we have in the modern day software development, mainly it is enhancing the efficiency and quality of the overall software cycle. Tester can forget about having to manually create routine and scripts for repetitive testing tasks. Then companies can optimize the testing, how? By reducing human errors, and focus on more important aspect of the required software. The quality standards and complexity are always growing and with them it is crucial for testing systems to become autonomous. These autonomous testing systems can free testing teams and let them focus their skills and time on complex test cases and explode invent and create even more automated tests that manual ones might overlook.

The integration of AI and machine learning, promises even greater advances but this is still a new area that requires more time to make them more adaptive and intelligent.The future of test automation is promising, because of those potential to changes. We expect testers to become even more dynamic, efficient, and effective in practice.

Software development will become an even more modern process by in summary making it possible to contribute to the quality of the required product, which in this fast-paced world plays a critical role in the tech industry.

References

[1] Meerts, J. The History of Software Testing, Testing references - the history of software testing.

[2] Balakin, R. (2024) The history of Test Automation, testRigor AI-Based Automated Testing Tool.

[3] In-Depth Industry Outlook: Automation testing market size, forecast. (2024, May 1). Verified Market Research. https://www.verifiedmarketresearch.com/product/automation-testing-market/

[4] When NOT to write automated tests? (n.d.). https://bssw.io/blogposts/when-not-to-write-automated-tests

[5] R, B. (2024, May 16). Power of Automation Testing: A Comprehensive guide. QA Touch. https://www.qatouch.com/blog/automation-testing-guide/

[6] What is CI/CD? (n.d.). https://www.redhat.com/en/topics/devops/what-is-ci-cd

[7] Manaher, S. (2024, February 28). Tool vs Framework: When to Opt for One Term Over Another. Grammar Beast. https://grammarbeast.com/tool-vs-framework/

[8] CTFL. (2024, August 7). https://www.isartal-akademie.de/index.php?id=421gadsource=1gclid=Cj0KCQjwsaqzBhDdARIsAK2gqnfjc sGWsV1hX8rQw-urAW8uA4aAsJjEALwwcB

[9] Jeremydmiller. (2020, May 6). A Small Case Study in Test Automation (and other things). The Shade Tree Developer. https://jeremydmiller.com/2020/05/06/a-small-case-study-in-test-automation-and-other-things/