Software testing automation tools and methods: The good, the bad & the ugly

Research · November 2020

DOI: 10.13140/RG.2.2.21514.08645

2 authors:

Irene Laplaza Osta
Lappeenranta – Lahti University of Technology LUT
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Simo Suurkuukka
Lappeenranta – Lahti University of Technology LUT
3 PUBLICATIONS **0** CITATIONS
SEE PROFILE

25.10.2020

Lappeenranta University of Technology

CT70A5000 – Impact and Benefits of Digitalization

Autumn 2020

Mini Project – Project Report

Software testing automation tools and methods: The good, the bad & the ugly

Authors:

Irene Laplaza

Simo Suurkuukka

Course Professors:

Ari Happonen

TABLE OF CONTENTS

1 INTRODUCTION

Software Testing Automation (STA) refers to the use of tools to control and configure the conditions prior to software tests and its execution. It compares the results obtained and those expected, i.e. use/create one software to test another (Quality-Stream, 2019). Testing Automation is considered an important practice in software development nowadays. There is evidence of its benefits for preventing defects and increasing the effectiveness of testing. Hence, many organizations in the software engineering field are starting to invest in it.

The main goal of this report is to create an overview of the kind of implications that software testing automation has when it comes to business units, operations, processes, strategic aspects, and end users, and formulate a knowledge of the subject we can later utilize in working life. According to its structure, the report briefly goes through the STA's history and presents several applications that are advisable to follow when automating software tests. This report also details the benefits, drawbacks as well as the impacts and negative effects from different viewpoints, and groups them between "Good", "Bad" and "Ugly" aspects. Opportunities for future research and trends that will impact the area have also been discussed.

1.1. Background

Based on the information provided by Weiss (2007) and Brown (2018), **Figure 1** has been created in order to provide an overview of Software Testing Automation evolution. Note that the periods are illustrative.

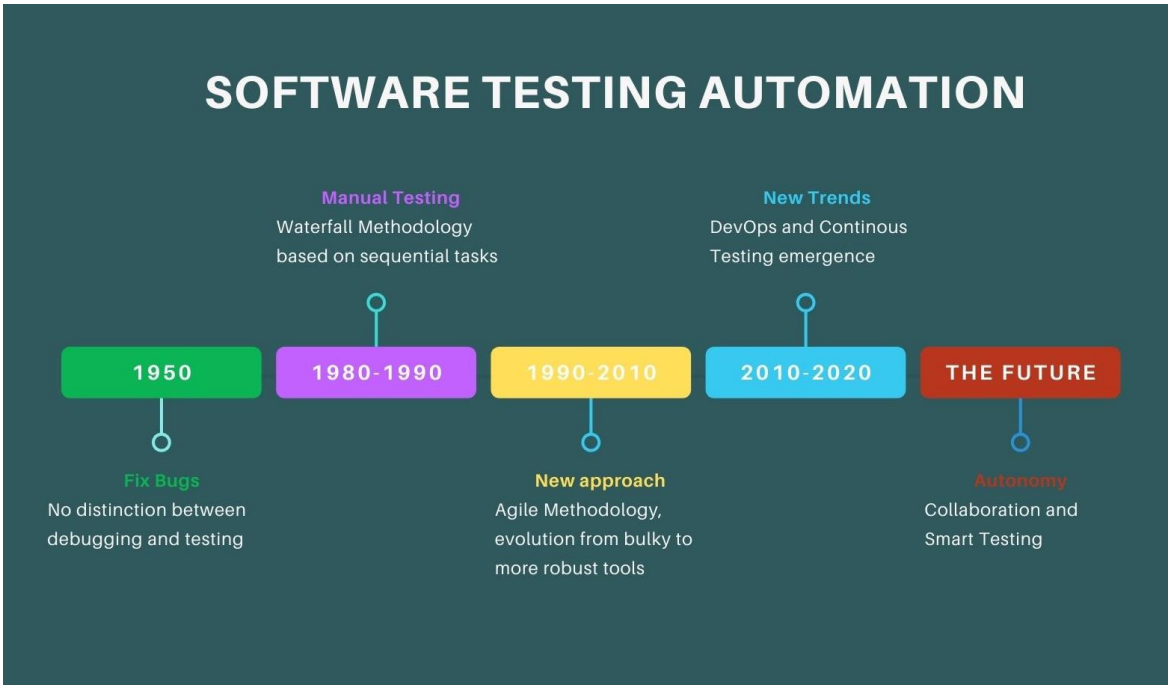

Figure 1. Software Automation Testing Timeline

The definition of testing has been evolving periodically. Around 1950s, programmers did not make clear distinctions between fixing existing bugs in their programs (faults and difficulties), and testing. Their tasks consisted of compiling and changing the code for it not to crash. Almost a decade later, they started to analyze testing functionalities exhaustively and concluded that the main goal of testing was to demonstrate whether programs worked properly. However, afterwards, they observed that testing was not build to prove correctness but to find errors, so they started to test if a program did "what it was not supposed to do" rather than "what it was supposed to do". (Weiss, 2007)

From approximately the mid-eighties, concept of testing moved towards manual activities done during a phase of the software lifecycle and it was based on Waterfall Methodologies and bulky tools. The introduction of more robust automation tools and the experimentation with different development approaches in the in the early 2000's, brought the Agile Methodologies, which are based on iterative tasks rather than sequential. According to Collins et al. (2012), testing automation is at the heart of agile testing. Testing is crucial for obtaining good quality software, but it requires agile methodologies during the implementation process. These methodologies help speed up performance and allow developers to spend time on what really needs their attention.

In the last decade, new technologies and key approaches such as DevOps (Development + Operations + Quality Assurance) and Continuous Testing began to emerge. Together with Agile, they aim to get tests done as quickly as possible, ensure collaboration, and customer satisfaction. Consequently, greater autonomy than what already exists, collaboration enhancement and smart testing is what is expected in the future.

1.2. Methodology

The methodology used to create this report has been based on team members' research in literature and extra information provided by previous working experiences in the field. Among the data sources used to develop the report, we can remark Google search engine, Scopus, Web of Science, different companies-related information, course literature and some Software Testing Automation webinars. In this project work, our biggest limitations have been the time constraints as students and the broad topic.

2 SOFTWARE TESTING AUTOMATION APPLICATIONS

It is important to highlight that not everything can be automated in a project and that automated tests are **not aimed to substitute** manual tests but complement each other. First and foremost, the manual system requires testing for being understandable, i.e. knowing what the steps of the functionalities are, to later recreate them in an automated script (programming code). Therefore, software testers must develop new technical skills and master existing ones so as to adapt to the upcoming working requirements.

Secondly, with the aim of performing it correctly it is important to follow a process (Guru99, 2019). The first step is to select the appropriate automation tools (see Section ***2.1. Tools***). Next, the scope of the application under which automation will take place needs to be defined. Then, planning, design and development phases are prepared in order to later execute the test scripts. Finally, continuous maintenance of the software functionalities is required as a means to start generating positive results.

According to Quality-Stream (2019), the practices and processes that are recommended to be automated are:

- Test cases that need to be run multiple times (iterative tasks); those that focus on the risk areas of the application; and those that need to be executed against different data sets or handle large volumes of data.
- Critical routes of the system (priority functionalities for the business).
- Functionalities that present a high degree of error and therefore are difficult to control during manual tests.
- Tests that need to be run against multiple devices, operating systems, and browsers.

On the other hand, there are some items that should not be automated such as unstable functionalities, i.e. those functionalities that are not yet fully developed or are under request for change; unpredictable results, e.g. Captcha, as we do not know the image is going to be shown in the user interface, the tool cannot be warned about the result it has to evaluate; and projects with insufficient time and resources to execute the manual tests in a limited period of time, because the planning, design, and implementation of the automations will require much more time in that short term than what the tool can actually save.

2.1. Tools

Testing Automation needs adequate tools to guide the way in which automation is performed since its correct application can deliver great benefits (Anderson, 2017). These tools can be open-sourced or licensed and are available to all people and organizations. However, since not all systems are

automated in the same way, it is crucial to choose those tools that best meet the specific project requirements for the automation test to succeed, i.e. choose the tools according to the technology the application under test is built on (Guru99, 2019).

Currently, the core functionalities of STA are based on Agile, DevOps, and Continuous Integration methodologies and should align with new technological trends (Brown, 2018; Anderson, 2017). The main goal is to get the most out of these trends, which are mainly centered in Artificial Intelligence and Machine Learning. Therefore, it is important to select STA tools according to:

- Support API and services testing

- Ease of use

- Project scripting language

- Analytical and AI/ML capabilities

- Support multiple frameworks, etc.

They have very different features that evolve as time goes by, but all must fulfill the main objective which is to deliver quality at speed. According to Anderson (2017), a decent tool should support basic optimization, automation of test case and data generation, smarter solutions, and analytics. The top ones are Selenium, TestComplete, Katalon Studio and QTP, but there are many more.

3 BENEFITS: THE GOOD

In order to achieve STA benefits, it is important to understand where and how test automation sits in relation to the software development process and how testing personnel are positioned within the organization. For the project quality, and therefore the added value, to be optimal, testing automation developers should be included in the project teams from the beginning of the development cycle and collaborate with software developers as much as possible. (Paju, 2020)

Tests created must be periodically reviewed to ensure their correct operation and avoid problems such as false negatives. It is also necessary to ensure that test automation runs concurrently with software development and follows its adjustments. Since if this is done when the software is already developed, the project will be much more difficult, time consuming and expensive.

The benefits that software testing automation has can be analyzed from different viewpoints. The primarily aim is to increase efficiency, improve product quality and test coverage in the development process (Collins & de Lucena, 2012). Nevertheless, these benefits unleash other competitive advantages for companies. According to Sultania (2015), the main benefits are:

- Optimizes time (less time to market, release speed-up, product-delivery acceleration)
- Cost reduction
- Reusability
- More frequent testing is enabled which leads to better quality control
- Improved test reliability
- Allow low skilled staff to run already created tests

3.1. Business units, operations, and processes

Product quality is an important aspect for any software project and is ultimately measured with software testing. The implementation of software testing automation should be carefully considered at a strategic business level so that a favorable result is possible. When STA is decided to be implemented, the processes related should be well planned to make sure all the value available can be extracted from the investment. The organization's knowledge about different software testing tools and products should be at a high level to find the best fitting solution in order to remain competitive or even gain new competitive advantages in the market. These is also needed to understand the hidden needs of software testing and address those needs appropriately. (Paju,2020)

Software testing has always been a time-consuming process. However, with automation testing **time** is **optimized**. (Kumar & Mishra, 2016) Once tests are created, the execution is carried out in a more autonomous way and does not require testers to monitor it constantly. Tests' releasing times are

therefore speeded-up and time to market is slightly decreased. However, although time used for testing is cut down, it also increases workload in the creation and maintenance of the automated testing, amounting to roughly similar amount of workload as manual testing. Nevertheless, these workloads can be reduced as testing automation knowledge, experience and tools' libraries are accumulated. (Kumar & Mishra, 2016)

Several studies claim that, by implementing software testing automation, organizations can save great amounts of resources and money, i.e. they can be used for reducing costs. In the long run is true but is just a consequence of the STA integration in their core activities and development processes. As in any system, it is important that its parts are aligned and complement each other for the optimization of results. Consequently, the main goal should be driven into this integration perspective. By reducing the amount of testing of successive versions of a product, **software costs are also reduced** since the amount of costs generated from doing testing manually and repeating a process multiple times decrease.

Although testing is crucial for obtaining good quality software, it also needs agile methodologies during the process. One of the biggest benefits it generates is **consistency in test execution**. It allows executing the same test cases by repeating the steps (iterative), in a lesser amount of time thus eliminating human error. This implies a high degree of **reusability** since scripts can be reprocessed with automated tests, functions, etc. and established testing frameworks can be applied to multiple projects.

3.2. Value for customers

The primary goal of STA is to help customers achieve **greater coverage**. For example, a client requests a 5% modification of the system's functionalities. When those modifications are updated, testers have to execute the testing of the remaining 95% to verify that none of the existing functionalities were modified by mistake and so they continue working normally. (Quality-Stream, 2019) This effort grows exponentially as the system grows, so it can become manually unsustainable at a functional and economic level. Thus, with automation, test cases can be accumulated throughout the project life cycle and new and existing functionalities can be easily analyzed at the same time.

More frequent testing leads to **better quality controls** in the product developed. By creating and maintaining automated testing for software, the repeated efforts take less time and make it possible for **less skilled or experienced personnel to run tests** successfully (Sultania, 2015). However, the security aspects of STA need to be considered and the security level must be selected according to customer and product specific needs. Only in that case, test **reliability** would be improved.

With the emergence of digital technologies, new product and services demands appear. The demand for **delivering quality software faster** requires, as mentioned before, organizations to search for solutions in Agile, continuous integration (CI) and DevOps methodologies (Brown, 2018; Anderson, 2017). And automation is the outcome of the successful adoption of those methodologies.

Overall, when maintenance is considered, not only the quality benefits gained from test automation are considerable, but also the speed advantage it brings to the development team to test and release products. Likewise, if the implementation of software test automation is managed and led properly, development cycles will be shortened. Customers should then be able to get better quality products quicker and expect faster updates and patches.

4 DISADVANTAGES: THE BAD

Software testing automation is not a 'penicillin', a miracle cure for all your software's ailments. As mentioned, STA can be an effective tool to increase testing efficiency, coverage and product quality when implemented and used correctly and in the appropriate tasks. But it is not without problems. (Paju, 2020) This section will outline the pitfalls an organization could face when implementing software testing automation and how to negate them to achieve the best possible result.

There are hordes of commercial testing automation tools and vendors in the market today. The problem with them is that commercial STA vendors sell the idea that just by capturing and recording the tests for a software, a person with relatively low amount of knowledge can create a complete and comprehensive testing automation for a software. This is simply not true. Recorded tests are slow, brittle, fragile and hard to maintain. (Paju, 2020)

Recorded tests usually heavily rely on user interface (UI) to conduct its actions, since it is easier to understand for a less knowledgeable employee. UI tests should only be used when testing for specifically UI functionalities. As a rule of thumb, 80% of software testing automation should be conducted in other levels of the software than UI (See **Figure 2**). This is because UI tests do not account for e.g. system and application architecture. (Paju, 2020)

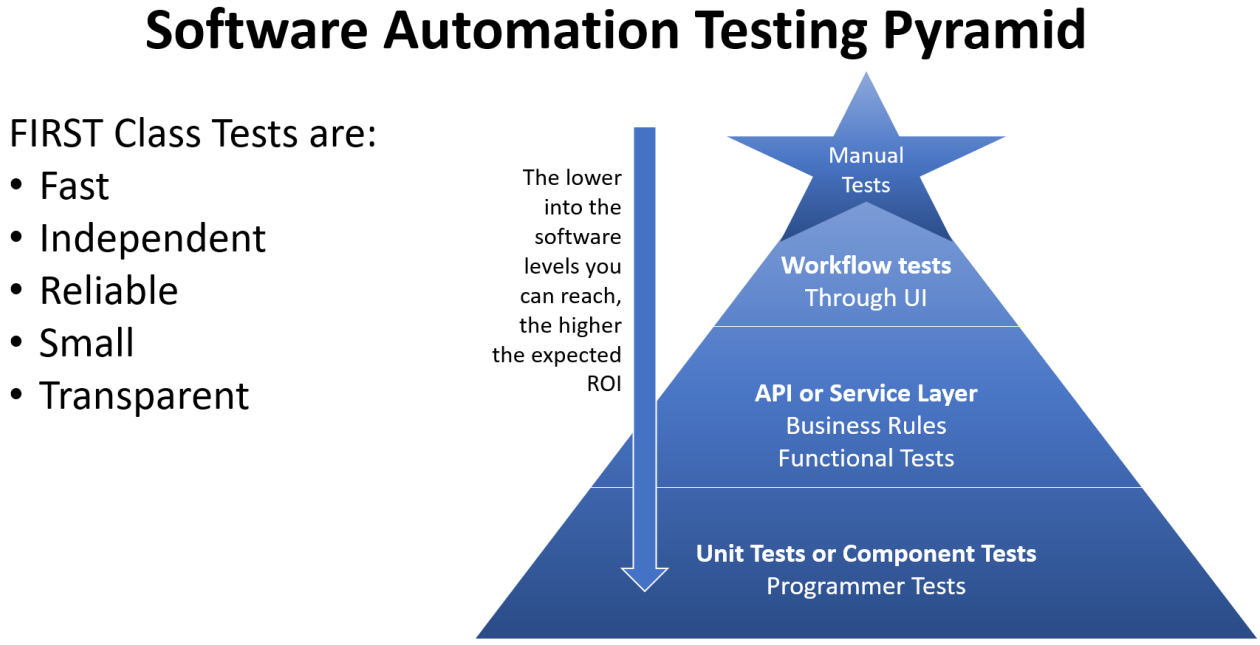

Figure 2. Software Automation Testing Pyramid (Paju, 2020)

Software testing automation generally does not find new defects. The best way to find new problems in the software is still manually and only after the discovery of a problem they can be addressed with automated testing. Finding bugs requires experience, skill, and intelligence. As it stands, testing

programs are not and likely will not substitute manual testing in the immediate future. This means that testing teams will not become obsolete and the amount of work done by testers is not necessarily less but shifts into different areas. (Paju, 2020)

Commercial software testing automation tools have a problem of scalability in that the number of licenses must be scaled up in accordance with increased testing. This leads to a contradiction of the basic idea of testing automation. Increased testing will increase costs in a linear fashion and is a clear demotivating factor for more testing from a business perspective. To increase quality and speed of development, increased testing should not increase costs and testing should be always available to as many people as possible without restrictions or limits. How to avoid this problem? one solution is using open source tools such as robot framework. Using open source tools means more development knowledge about testing automation is needed, which means larger initial investments. On the other hand, the scalability is almost limitless, an open source tool has no license limits and testing can be increased almost limitlessly. Another solution is to create a testing automation system from scratch. This is highly dependent on very high levels of competences and takes a lot of time and money. (Paju, 2020)

The old-fashioned idea of separate departments doing different tasks in the organization independently and without collaboration with other departments is an easy mindset to get stuck into if an organization's leadership doesn't look into enough detail when implementing e.g. software testing automation. This will most likely lead to dysfunctional testing automation processes that hinder, rather than create, product value. When an organization structure is set up so that testing automation is delegated to a special group, it hinders collaboration. It also creates a situation where duplicate tests and coverage could be created when teams do not communicate properly and are working on the same project. To avoid this problem the testing Agile methods should be applied (see **Figure 3** for more in-depth view of agile testing). Automation knowledge should be included into project teams from the start of the project and testing automation viewed as a part of production code. The communication between testing automation developers and application developers should be facilitated to create the best results. To make sure the well collaborating team works together, the definition of quality as it is relevant to the project should be established and the desired system quality defined. (Collins, Vicente and De Lucena, 2012; Sultania, 2015; Paju, 2020)

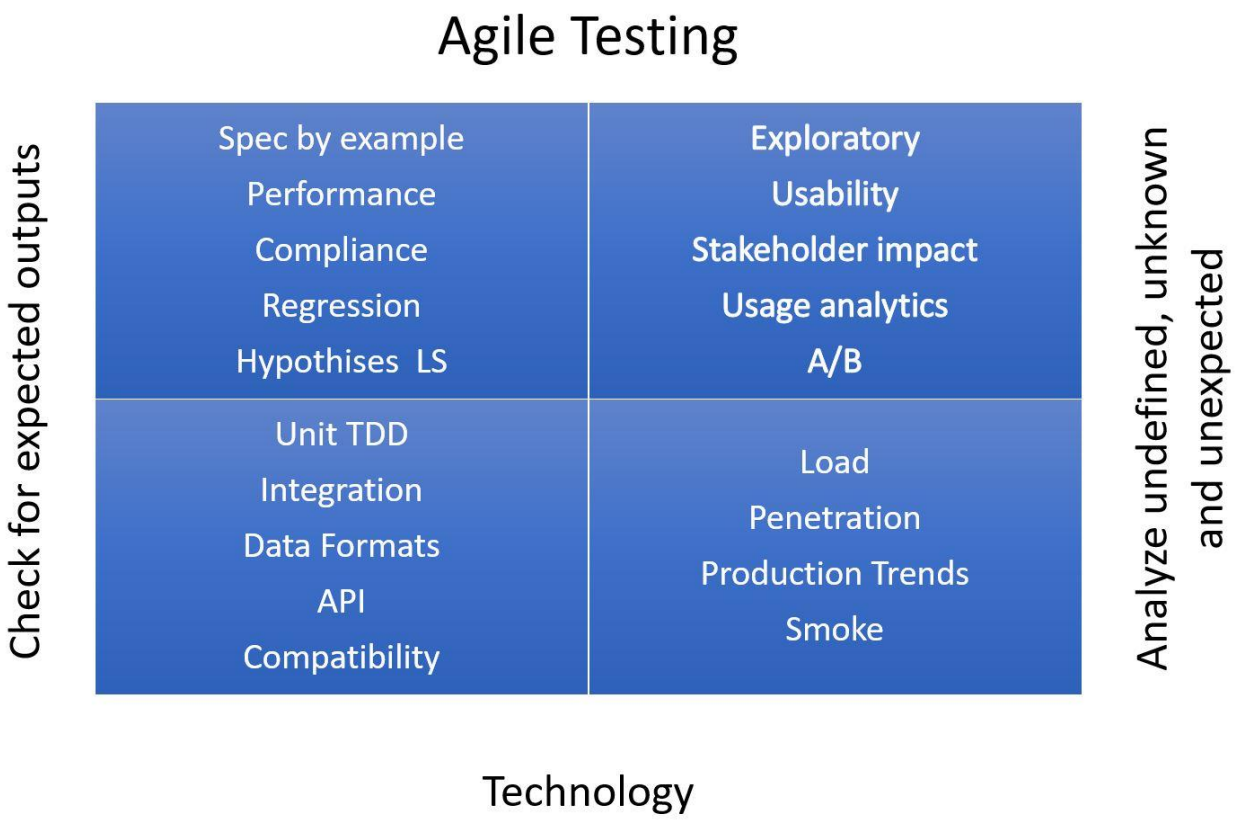

Figure 3. Agile Testing (Paju, 2020)

The danger of implementing testing automation in a product or project but not maintaining it is very real. The problem with this is that the investment made in the testing automation development is largely realized in the developmental phases while the actual speed and quality gains might only be realized late in the product life cycle. When testing automation is not maintained the value, it provides will erode over time as the product code changes. The solution to this problem is to have a plan to actively update the testing automation continuously, even in cases where the system is not bring actively developed. (Paju, 2020)

The assumption that implementing software testing automation to degrease costs is a very dangerous one also. This is an easy assumption to make, especially when similar tools and technologies are used in the field of Robotic Process Automation (RPA). Unlike in RPA, in testing automation you cannot think about saving money through less working hours used. The amount of work done when implementing and maintaining testing automation is similar to the amount of work done when manually testing products. The benefits of testing automation can be initially found elsewhere than the bank statement; Increased quality in product, quicker time to market, faster product development cycles, faster update schedules due to faster and wider testing available with less experienced employees. (Paju, 2020)

5 NEGATIVE IMPACTS AND EFFECTS: THE UGLY

Software testing automation has many implications and aspects in the modern software development process. The impacts and benefits that STA has on business, process, value delivered, and efficiency are discussed in the previous sections. In this section we will discuss the negative human level impacts of STA.

5.1. Human level impacts

Employees' motivation could be affected by automation of tasks; "Successful automation, where a person's work and life are enhanced may be a motivating factor for that person. However, there are circumstances where automation of all or part of a job could be demotivating. These include (1) the fear of redundancy (2) the effect of an unbalanced task mix (3) the effect of the automation being flawed." (Evans et al., 2020). It is also possible that "the term 'test automation' threatens to dissociate people from their work." (Bach and Bolton, 2016). Testers could face a threat to their self-perception posed by automation. If the task mix within the job is altered by testing automation to strip the repetitive tasks and leave more stressful tasks intact, an over stimulating task mix remains. This could make the job boring and stressful. If the implementation of STA is not successful, this could increase the stress of testers by leaving them to deal with the flawed automation and to do their high stress tasks. This could negatively affect motivation. The clash between what is expected of STA and what is delivered by STA can prove to cause frustration in employees when the expectations are set too high compared to what can actually be achieved. (Evans et al., 2020).

6 FUTURE VISIONS

Software Automation Testing involves continuous research and improvement of emerging technologies, which requires innovative approaches from organizations to overcome the challenges they pose. The process of adapting to new updates must be fast and efficient since a minimal delay can make the processes obsolete. The tools that are used to develop the scripts also need to be updated simultaneously, otherwise they can be a big hurdle.

New technologies are continuously being developed to increase the percentage of total tasks in software testing that can be automated by STA and make testing more efficient. An example of this applying genetic algorithms to identify the most critical path cluster in a program. This increases testing efficiency because exhaustive testing is many times not possible due to the sheer volume of testing that needs to be conducted even in medium size software. Finding the most critical parts and to go ahead and test them first can greatly increase the testing efficiency and create more value (Xu et al., 2006).

AI already is a big factor in STA, as it increasingly is in other parts of the software industry. It has many aspects which it can benefit STA (see **Figure 4**). According to Hourani et al. (2019), "The results show that AI can achieve better results in Software Testing and AI-driven testing will lead the new era of the quality assurance (QA) work in future. AI Software Testing will reduce time to market and will increase the efficiency of the organization to produce more sophisticated software and will create smarter automated testing." AI software testing is expected to replace the QA role and testing engineers. This is expected to transform the QA team and software testers' roles into monitoring and tuning the AI which does the actual testing. (Hourani et al., 2019)

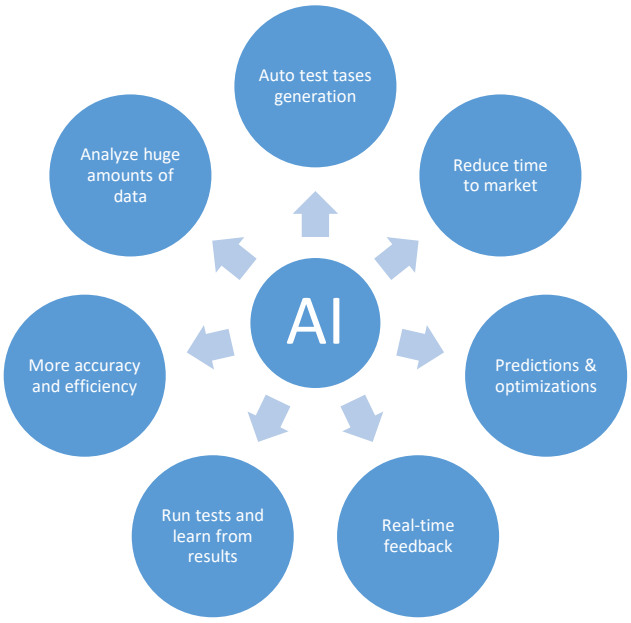

Figure 4. The AI Software Testing Key Advantages (Source: Hourani et al., 2019)

7 CONCLUSIONS

As the outcome of this report, we can conclude that software testing automation is a useful and increasingly necessary tool in software development. If STA is understood and implemented correctly, it entails plenty of benefits, especially in the long run. Therefore, its application should be focused on a long-lasting strategic solution rather than an answer for specific crises.

Similarly, it is crucial to understand that they do not replace manual testing, but rather complement each other. When implementing STA in an organization, there are multiple aspects that need to be considered when managing change; there has to be a good understanding of limitations, processes and available tools. As a consequence, developers are able reach the market faster and efficiently, which is strategically essential.

REFERENCES

Anderson, B. (2017). Best Automation Testing Tools for 2020 (Top 15 reviews). [Online post]. Available via: https://medium.com/@briananderson2209/best-automation-testing-tools-for-2018-top-10-reviews-8a4a19f664d2 [Accessed 20 Oct. 2020]

Bach, B. J. and Bolton, M. (2016) A Context - Driven Approach to Automation in Testing A Context - Driven Approach to Automation in Testing. Satisfice, Inc., 2016.

Brown, A.J., (2018). Testing trends in 2019 and further. *QATestLab Blog*. [Online Article]. Available via: https://blog.qatestlab.com/2018/11/06/testing-trends-2019/ [Accessed 13 Oct. 2020]

Collins, E. & de Lucena Jr., V.F. (2012). Software Test Automation practices in agile development environment: An industry experience report. *2012 7th International Workshop on Automation of Software Test, AST 2012 - Proceedings*, art. no. 6228991, pp. 57-63. doi: 10.1109/iwast.2012.6228991

Collins, E., Dias-Neto, A. & de Lucena Jr., V.F. (2012). Strategies for Agile Software Testing Automation: An Agile Industrial Experience. *2012 IEEE 36th Annual Computer Software and Applications Conference Workshops*. pp. 440-445 doi:10.1109/compsacw.2012.84

Evans, I., Porter, C., Micallef, M. & Harty, J. (2020) Stuck in limbo with magical solutions: The testers' lived experiences of tools and automation. *VISIGRAPP 2020 - Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and* Applications, 2(Visigrapp), pp. 195–202. doi: 10.5220/0009091801950202.

Guru99 (2019). Automation Testing Tutorial: What is Automated Testing? [Website]. Available via: https://www.guru99.com/automation-testing.html [Accessed 08 Oct. 2020]

Hourani, H., Hammad, A. and Lafi, M. (2019) 'The impact of artificial intelligence on software testing', *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology, JEEIT 2019 - Proceedings*, pp. 565–570. doi: 10.1109/JEEIT.2019.8717439.

Kumar, D., & Mishra, K. K. (2016). The Impacts of Test Automation on Software's Cost, Quality and Time to Market. Procedia Computer Science, 79, 8–15. doi:10.1016/j.procs.2016.03.003

Paju, I. (2020) *Testiautomaation 7 kuolemansyntiä, Webinaari: Testiautomaation 7 kuolemansyntiä*. Available at: https://www.tieturi.fi/webinaari/webinaari-testiautomaation-7-kuolemansyntia/ [Accessed 23 Oct. 2020].

Quality-Stream (2019). Introducción a la Automatización de Pruebas de Software. [YouTube Video]. Available via: https://www.youtube.com/watch?v=R_hh3jAqn8M&ab_channel=Quality-Stream [Accessed 08 Oct. 2020]

Sultania, A. K. (2015). Developing Software Product and Test Automation Software Using Agile methodology. *Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT).* pp. 1-4, doi:10.1109/c3it.2015.7060120

Weiss, N. (2007). Automated Testing. *SlidePlayer*. Available via: https://slideplayer.com/slide/5712292/ [Accessed 13 Oct. 2020]

Xu, B. *et al.* (2006) 'Application of genetic algorithms in software testing', *Advances in Machine Learning Applications in Software Engineering*, 3(4), pp. 287–317. doi: 10.4018/978-1-59140-941-1.ch012.