Vol-2546
urn:nbn:de:0074-2546-4

CS&SE@SW 2019
2nd Student Workshop on Computer Science & Software Engineering

Proceedings of the 2nd Student Workshop on Computer Science & Software Engineering (CS&SE@SW 2019)

Kryvyi Rih, Ukraine, November 29, 2019.

Edited by

Arnold E. Kiv *
Serhiy O. Semerikov **
Vladimir N. Soloviev **
Andrii M. Striuk ***

* Ben-Gurion University of the Negev, Beer Sheba, Israel
** Kryvyi Rih State Pedagogical University, Kryvyi Rih, Ukraine
*** Kryvyi Rih National University, Kryvyi Rih, Ukraine

Table of Contents

Invited Talks

Computer Science

Software Engineering

2020-01-24: submitted by Serhiy O. Semerikov, metadata incl. bibliographic data published under Creative Commons CC0
2020-02-09: published on CEUR-WS.org |valid HTML5|

Automation in software testing, can we automate anything we want?

Bohdan Oliinyk[0000-0003-3670-2605] and Vasyl Oleksiuk[0000-0003-2206-8447]

Ternopil Volodymyr Hnatiuk National Pedagogical University,
2, Maxyma Kryvonosa Str., Ternopil, 46027, Ukraine
{olijnyk_bm, oleksyuk}@fizmat.tnpu.edu.ua

Abstract. The article considers the problem of test automation software. The authors analyze testing tasks that can be automated. They also cite cases where the use of automation is inappropriate. The key factors of using automation are time and cost savings. According to the authors, the advantages of automated tests are: the ability to check the latest changes in the application (regression testing), speed of execution, saving the time of testers, the ability to create self-tests by developers. The disadvantages of automatic tests are: insufficient reliability, need for support, fewer errors detected, a false sense of product quality. The following processes are identified, which can be automated: background processes, file logging, database entry, registration and payment systems, load tests, data entry operations, long-end-to-end scripts, checking complex mathematical calculations, checking correct search. The article provides statistics on the use of programming languages for developing automated tests. A comparative analysis of ready-made software products for automated testing is offered. Based on research analysis and experience, the authors believe that human intelligence is always required to validate the program. So, the authors justify the need to perform a manual and automated test.

Keywords: Software Testing, Quality Assurance, Manual Software Testing, Automation of Software Testing.

1 Introduction

Testing is a very important stage in software development life cycle (SDLC). This process ensures that most errors are found. However, one of the principles of testing is that all errors cannot be found – exhaustive testing is impossible [8]. Therefore, IT experts believe that the end product (software or hardware) works as it should, if it is as close as possible to certain functionality, reliability, performance.

As the research and practical experience shows, the software development lifecycle takes place during the following stages – analysis, design, development, implementation, testing, deployment, and support (Fig. 1) [5].

As the speed of software development is increasing today, there is a need for quality and timely testing. But some of the testing tasks are too time-consuming to perform their manually. In addition, more and more companies are moving to work according

to agile methodology. An agile process begins with initiating the project; in the activities that follow, the software will be developed and deployed into the user environment through multiple iterations. In most agile methodologies, maintenance does not appear as a separate phase, but is rather performed through further iterations of the main development phases [2]. That's why automation is becoming more important than ever [14].

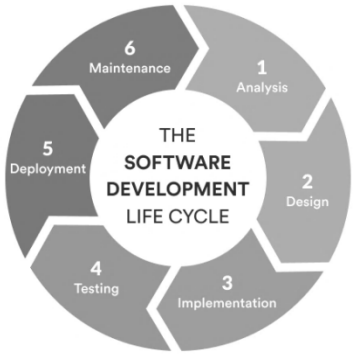

Fig. 1. The Software Development Life Cycle

Even 5 years ago, testing automation in Ukraine was economically unattractive.

In article [2] Nikolay Alimenkov writes: "… look who supports conferences for automated testers? And almost none. In large companies, this does not fit into the "strategic development plan". Small is simply not interesting. As a result, automated testing in Ukraine lives on thanks to the enthusiasm of the automators themselves" [2].

Today the situation has changed. Almost every company wants to have a automated tester, no matter if it's a big company or a small one. Today, automation is not only relevant in startups. As a result, writing self-tests is usually not time consuming. In general, startups may not even have test cases. The growing numbers of communities, forums, channels where automated testers discuss their problems, can also testify to the great development of automation.

2 Relevance of the study

Automated testing is now evolving rapidly. This is due to the fact that automated, debugged processes will require less money, which is one of the key factors in IT. As experience in the early days of the introduction of automation shows, it requires more material costs, while fewer tests are performed instead. However, over time, it is possible to significantly increase the number of test runs without significant investment. The popularity of automated testing is also explained by the fact that more and more "manual" testers want to develop in programming. Automation is a next stage of their development. "The job was pretty boring. We were just supposed to look at flagged differences and decide if they were a big deal or not. I wasn't happy with that. I wanted to know what caused the differences, so I started digging a little deeper", – John Sonmez talks about it in the article "Going from QA (or Another Technical Role) to Software

Developer" [12]. In the paper reports an empirical study on the relationship between code visibility and testability. The authors claim for manual testing, code visibility does not necessarily affect test code coverage and fault detection rate. However, for automated testing using testing tools, low code visibility often leads to low code coverage and poor fault detection rate [10].

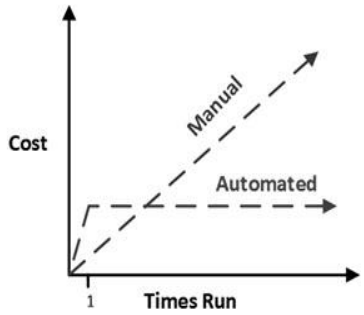

Fig. 2. The dependence of the cost of the test on the number of executions in manual and automated testing

Based on our own experience, we can say that the process depends largely on the project. One can often hear among testers that manual testing is not a tedious process, but an approach is important.

Systematic and automated approaches have shown capable of reducing the overwhelming cost of engineering automated tests. Industrial success cases have been openly reported and academic interest continues to grow as observed by the increasing number of researchers in the field. While there exist various trends on evolving the automation in software testing, the provision of sound empirical evidence is still needed on such area [6].

The authors of [1] describes a strategy to develop automated testing suites to assess the correctness of consent and revocation. This strategy is based on a formal language in order to provide rigorous and unambiguous consent and revocation specifications, and comprises of two novel procedures that facilitate the process of eliciting testing requirements for privacy properties and creating automated privacy-testing suites.

The purpose of the article. The dynamic pace of development of automated testing requires the answer to the question "can we all automate?" What is the role of manual testers in modern software development processes?

3 Automated testing, its advantages and disadvantages

3.1 Types of testing

Software testing can be performed for different purposes. According to the purpose there are such types of testing [11]:

- Functional Testing is a type of software testing whereby the system is tested against the functional requirements/specifications.
- Usability Testing is a type of software testing done from an end-user's perspective to determine if the system is easily usable.
- Security Testing is a type of software testing that intends to uncover vulnerabilities of the system and determine that its data and resources are protected from possible intruders.
- Performance Testing is a type of software testing that intends to determine how a system performs in terms of responsiveness and stability under a certain load.
- Regression testing is a type of software testing that intends to ensure that changes (enhancements or defect fixes) to the software have not adversely affected it.
- Compliance Testing is a type of testing to determine the compliance of a system with internal or external standards.

All these types of tests can be performed both manual and automated.

Researchers identify such phases of the software testing process [9]:

1. Preliminary Testing phase is conducted especially for testers to clarify the specification requirements of the customer. Preliminary testing is performed during the following steps: review requirements specification, prepare test plan, prepare software tool, prepare test environment, prepare test case, prepare test automation tool, determine acceptance test tool.
2. The testing phase is a separate phase which is conducted by a different test team after the implementation is completed.
3. User acceptance testing phase which provides for checks integration testing, test strategy document, integration testing signoff, repair and coordinate release.

Note that manual and automated testing can be used together at different stages of software quality verification.

Automated testing or testing automation is a method of testing software. The method involves the use of special software tools to control the performance of tests. Then the actual test results are compared with the predicted ones. Most operations are performed automatically, with little or no intervention by the test engineer. Automated testers write scripts (so-called automated test cases) that have a set of actions and checks. Properly written automated tests can have many benefits and can be very useful for the project and organization. However, there are some disadvantages of automated tests that you should also be aware of.

3.2 Features and benefits of automated testing

Regression testing. Automation testing is the most common for this type of testing. Regression testing is a type of testing aimed at checking changes made to an application or environment (debugging, code merging, migration to another operating system, database, web server), to confirm that existing functionality is still working. Regression can be both functional and non-functional tests. Typically, regression testing uses test cases written in the early stages of development and testing. That is, regression

automated tests are performed at a predetermined time interval. They are usually downloaded after every successful compilation (in small projects) or every night or every week. This guarantees that the changes to the new version of the program do not damage the already existing functionality. In the paper [7], authors discuss the advantages and drawbacks of using UML diagrams for regression testing and analyze that UML model helps in identifying changes for regression test selection effectively.

Speed of execution. Automated verification scripts may take some time. However, it takes less time to complete them than the person who would perform these checks manually. Therefore, self-tests help to provide quick feedback to the development team.

Time saving for testers. Test automation frees up testers' time. Therefore, they may be more focused on exploring new features. Automated checks can be started automatically with minimal supervision, or without any supervision, or manually. Usually when you do not want to use automated tests there is a cyclical situation of lack of time (Fig. 3).

Fig. 3. Causal link lack of time and unwillingness to test automatically

Ability to create automated tests by developers. Automated tests are usually written in the same language as the product being tested. As a result, the responsibility for writing, conducting and performing tests becomes a shared responsibility. Everyone in the development team, not just testers, can contribute to the quality of the software.

3.3 Disadvantages of automated testing

False sense of product quality. For this reason, it is worth paying special attention to successfully passed automated tests. This is especially important for user-level (UI) or system-level functionality testing. The automated test only checks what is programmed to be tested.

All automated tests in the test suite may pass, but some mistakes may be not identified. The reason for this is that this test was not designed to detect these specific failures.

Insufficient reliability. Automated checks may not be successful due to many factors. For example, automatic checks can be broken by changing the user interface, shutting down a service, or having a network problem. These problems do not directly affect the program being tested, but may affect the outcome of automated tests.

Need for support. It should be understood that automated tests require maintenance. Automated checks are short-lived. Failure to update them will cause crashes. It is also

possible that some checks are no longer relevant or that they do not correspond to new implementations of the software. These failures can affect the test results.

Writing an automated test case is not a one-time effort. To get the most out of automated tests, they need to be updated and up-to-date. This usually takes time, effort and resources.

Fewer errors detected than manual testing. Most errors are detected "accidentally" or during exploratory testing. This type of testing involves the simultaneous study of a software product, the design of tests and their execution. Its specificity is that at each exploratory testing session there is an opportunity to test the application in different ways.

On the other hand, automated checks always follow the set path, sometimes with the same test dataset. One can mention here one of the principles of testing – the "paradox of the pesticide". It is that by performing the same tests over and over again, we face the fact that they find fewer and fewer errors. This is due to the development of the system, as a result of many found defects are corrected and old test cases no longer work. This in turn reduces the likelihood of finding new defects in the product.

"Test automation is not always testing". Here, it is understood that testing is a research activity. Testing requires specific knowledge, a purposeful mind, and a willingness to learn the application. Unfortunately, many people are wrong about the importance of test automation. They get a test automation tool and want to get rid of all the "manual testers". But this is not possible because testing is not simply about performing a set of predefined testing steps and comparing actual results with expected results. The most recent tasks are automated checks.

Ability to group failures into clusters. In the paper [10] present an approach to automatically detect passing and failing executions using cluster-based anomaly detection on dynamic execution data. The key hypothesis underlying the approach is that failures will group into small clusters whereas passing executions will group into larger ones.

A person's intelligence is always required to validate the program. Here's how Tommy Wyher says it: "Every day, we see hundreds of new apps and products out in the market. A lot of testing takes place before they are released to the public. Automated testing can speed up the process and is often seen as a replacement for manual testing. However, manual testing still has a critical role in the QA process. By dedicating your QA resources to only one of these approaches, you'll miss many opportunities to improve quality. Using manual and automated testing together will lead to a higher quality, more stable product" [15].

3.4 Tests that can be automated

Let's look at software components and processes that can be automated.

1. Hard to reach places in the system: background processes, file logging, database entry.

2. Frequently used functionality where there is a high risk of errors: payment systems, registrations, etc. Automation of critical functionality checks ensures fast errors, since the test takes an average of several minutes.
3. Load tests that test the functionality of a system with a large number of requests.
4. Template operations, including data searches, input of forms with many fields, checking their preservation.
5. Validation messages: fill in the fields with incorrect data and validation check.
6. Long end-to-end scenarios. For example, an online store scenario that involves: user registration, product detail page (PDP), shopping cart, shopping cart, product purchase, and confirmation of purchase.
7. Verification of data requiring accurate mathematical calculations, eg accounting or analytical processing.
8. Checking the correctness of the data search.

Although newer automated testing tools are emerging, it is still difficult to test the functionality of the user interface. It is also not possible to automate the testing of a new feature without at least one manual test run. Therefore, to say that you can automate everything, for now, will be an exaggeration.

Consider which tools are most often used to write automated tests.

Automated tests are written in various programming languages, the most common being Java. Figure 4 shows the distribution of automated programming languages in the world [16]:

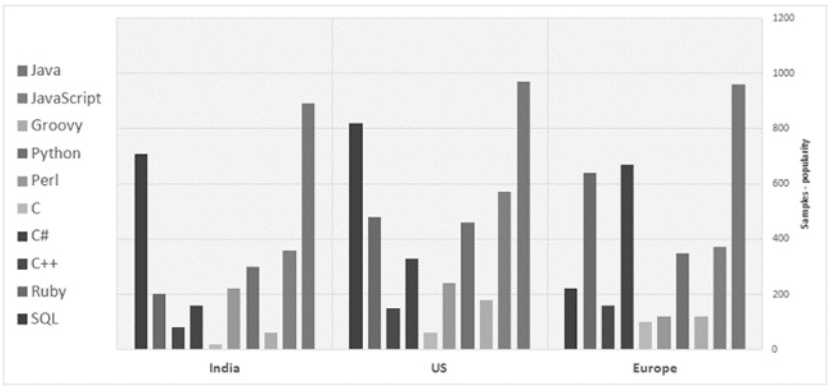

Fig. 4. Use programming languages to create automated tests

3.5 Some testing automation tools

In addition to programming languages, there are other automation tools available in software testing. Their comparative analysis is given in the Table 1 [3].

Table 1. Comparative of testing automation tools

Automation tools	**Selenium**	**Katalon Studio**	**Unified Functional Testing**	**Testcomplete**	**Watir**
Available since	2004	2015	1998	1999	2008
Application Under Test	Web applications	Web, Mobile applications	Web, Desktop and mobile applications	Web apps, Desktop, Mobile applications	Web applications
Pricing	Free	Free	~10000 USD	~100 USD	Free
Supported Platforms	Windows, Linux	Windows, Linux	Windows	Windows	Windows, Linux
Scripting Languages	Java, C#, Perl, Python, JavaScript, Ruby PHP	Java, Groovy	VBScript	VBScript, Python, JavaScript, Ruby PHP, C#, C++	Ruby
Programming Skills	Advanced skills needed to integrate various tools	No required. Recommended for advanced test script	No required. Recommended for advanced test script	No required. Recommended for advanced test script	Advanced skills needed to integrate various tools
Ease of installation and Use	Require advanced skills to install and use	Easy setup and use	Complex in installation. Need training to properly user the tool	Easy setup and use. Need training to properly user the tool	Require advanced skills to install and use

Each of these tools has its own peculiarities and scope. The most common tool for automation is Selenium. This is because it is free and scripts can be written in many programming languages, unlike other tools. Developed over 15 years ago, it has evolved over the next decade. It is now a web browser automation tool. In most cases, it is used to test Web applications, but this does not limit its scope.

JMeter is used for load testing. Initially, this application was intended to test the operation of the Apache Tomcat servlet container, which is essentially a web server. With the development of Jmeter, the user interface has been improved and additional features have been added that have made it an effective tool for performance testing and load testing of web applications.

It is impossible to say exactly what kind of tools will be popular in the future, as they are in rapid development. In our opinion, this will greatly simplify testing automation.

The authors of the article [4] recommend to automate the user interface according to the Mike Cohn scheme (Fig. 5). It shows the recommended proportion of tests to be implemented on each test level. The pyramid contains 3 levels:

1. Low-level tests are much faster by nature. Faster tests give you faster feedback. Faster feedback from tests execution allows to catch issues early on, saving huge amounts of costs.

2. Low-level tests are executed much earlier in the QA automation pipeline. Usually, unit tests are run before each commit test. If this is true, then testers prevent bugs and do not let them get into the project repository.
3. Low-level tests are much more stable than high-level ones.

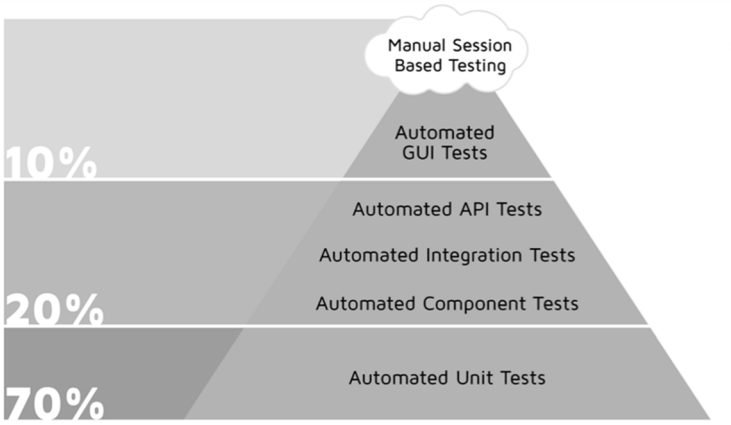

Fig. 5. Agile test pyramid initially designed

It's need to remember that high-level tests should be only the third defense shield, for catching all the remaining issues that were not caught on the first two levels.

4 Conclusions

In the process of the research, we found out the importance of testing in the software development process. We found out what automation is and what role in testing. We analyzed tools for writing automated scripts, provided statistics on the popularity of programming languages to test automation across regions.

Test automation is evolving very quickly, new tools and technologies are emerging almost every week, but they are not yet fully replaceable, and everything cannot be automated. Without human intervention, it is impossible to construct the testing process, since testing requires intellectual intervention. In our opinion in the near future, there will be a trend of increasing number of automated testers and a gradual decrease in manual ones. It will become an obvious educational goal to become an expert in automation. After training in the practical use the testing tools, manual testers will be able to perform automated testing. Therefore, every tester who wants to be more in demand in the IT should learn at least one programming language. So learning programming and testing are processes that depend on each other.

It cannot be said that automated testing will completely replace manual testing. As a result, the conclusion is one: in the world of software testing, there will always be room for both manual and automatic testing. The choice of testing methods is left for every worker in this area.

References

1. Agrafiotis, I., Creese, S., Goldsmith, M.: Developing a Strategy for Automated Privacy Testing Suites. In: Camenisch J., Crispo B., Fischer-Hübner S., Leenes R., Russello G. (eds.) Privacy and Identity Management for Life. Privacy and Identity 2011. IFIP Advances in Information and Communication Technology, vol. 375, pp. 32–44. Springer, Berlin, Heidelberg (2012). doi:10.1007/978-3-642-31668-5_3
2. Alimenkov, N.: Avtomatizatciia testirovaniia v Ukraine nikomu ne nuzhna (Nobody needs testing automation in Ukraine). XP Injection. https://xpinjection.com/articles/test-automation-not-needed-in-ukraine (2013). Accessed 20 Sep 2019
3. Anderson, B.: Best Automation Testing Tools for 2020 (Top 10 reviews). https://medium.com/@briananderson2209/best-automation-testing-tools-for-2018-top-10-reviews-8a4a19f664d2 (2017). Accessed 21 Sep 2019
4. Bushnev, Y.: Top 15 UI Test Automation Best Practices You Should Follow 2019. BlazeMeter. https://www.blazemeter.com/blog/top-15-ui-test-automation-best-practices-you-should-follow (2019). Accessed 27 Sep 2019
5. Casaglia, G., Pisani, F.: The integration and distribution phase in the software life cycle. In: Ehrig H., Floyd C., Nivat M., Thatcher J. (eds.) Formal Methods and Software Development. TAPSOFT 1985. Lecture Notes in Computer Science, vol. 186, pp. 371–384. Springer, Berlin, Heidelberg (1985). doi:10.1007/3-540-15199-0_24
6. Endo, A.T., Bertolino, A., Maldonado, J.C., Delamaro, M.E.: Guest editorial foreword for the special issue on automated software testing: trends and evidence. Journal of Software Engineering Research and Development **6**(2) (2018). doi:10.1186/s40411-018-0047-3
7. Fahad, M., Nadeem, A.: A Survey of UML Based Regression Testing. In: Shi Z., Mercier-Laurent E., Leake D. (eds.) Intelligent Information Processing IV. IIP 2008. IFIP – The International Federation for Information Processing, vol. 288, pp. 200–210. Springer, Boston (2008). doi:10.1007/978-0-387-87685-6_25
8. Ghahrai, A.: Seven Principles of Software Testing. https://www.testingexcellence.com/seven-principles-of-software-testing (2018). Accessed 21 Sep 2019
9. Lawanna, A.: The Theory of Software Testing. AU Journal of Technology **16**(1), 35–40. http://www.journal.au.edu/au_techno/2012/jul2012/journal161_article05.pdf (2012)
10. Mariani, L., Hao, D., Subramanyan, R., Zhu, H.: The central role of test automation in software quality assurance. Software Quality Journal **25**(3), 797–802 (2017). doi:10.1007/s11219-017-9383-5
11. Software Testing Types. Software Testing Fundamentals. STF. http://softwaretestingfundamentals.com/software-testing-types. Accessed 21 Sep 2019
12. Sonmez, J.: Going from QA (or Another Technical Role) to Software Developer. Simple Programmer. https://simpleprogrammer.com/going-from-qa-to-software-developer (2016). Accessed 20 Sep 2019
13. Tasharofi, S., Ramsin, R.: Process Patterns for Agile Methodologies. In: Ralyté J., Brinkkemper S., Henderson-Sellers B. (eds.) Situational Method Engineering: Fundamentals and Experiences. ME 2007. IFIP – The International Federation for Information Processing, vol. 244, pp. 222–237. Springer, Boston (2007). doi:10.1007/978-0-387-73947-2_18
14. What Is The Benefit of Test Automation and Why Should We Do It? SmartBear Software. https://smartbear.com/solutions/automated-testing (2019). Accessed 20 Sep 2019

15. Wyher, T.: 5 Reasons Why Manual Testing Is Still Very Important. DZone DevOps. https://dzone.com/articles/5-reasons-why-manual-testing-is-still-very-importa (2016). Accessed 21 Sep 2019
16. Yehezkel, S.: World's Most Desirable Test Automation Skills!, TestProject. https://blog.testproject.io/2015/12/03/worlds-most-desirable-automation-skills (2015). Accessed 21 Sep 2019