An overview on software testing and auditing

[1]Catia LOPES, [3]Maryam ABBASI, [2]Pedro MARTINS, [1]Filipe CARDOSO, [4]Filipe SA

[1] Department of Informatics, Polytechnic of Viseu, Viseu, Portugal
[2] CISeD - Research Centre in Digital Services, Polytechnic of Viseu, Viseu, Portugal
[3] University of Coimbra, Coimbra, Portugal
[4] Polytechnic of Coimbra, Coimbra, Portugal

csousalopes@hotmail.com, maryam@dei.uc.pt, pedromom@estgv.ipv.pt, fcardoso@estgv.ipv.pt, filipe.sa@isec.pt

Abstract. More and more people are dependent on technology. They increasingly use electronic services for day-to-day routines, and user loyalty to the software is essential, being defined by the excellence of the SW. The fewer flaws it has, the greater the likelihood of being able to retain the user's loyalty. For this situation to be possible, tests are crucial in the development stage since they have the main purpose of identifying errors. To be possible to have a good quality of software, it is important to realize the importance of carrying out tests, as well as to understand what types of tests exist and realize which ones fit in each situation. In addition, the article addresses the life cycle and levels associated with software testing. In terms of test automation, there are some tools for developing this type of test, referencing the Katalon studiom robot framework, Protractoe and Watir; each is framed in different practical situations.

Keywords: Software Tests, Manual Tests, Automated Tests, Software Test Levels, Test Life Cycle.

1 Introduction

Software testing is becoming increasingly important in day-to-day life. We constantly rely on the accurate execution of applications in our equipment such as cell phones, engine ignition, or everyday activity like online transactions, emails, bank transfers, or even on those concealed from us like back-office software for transaction processing Homès (2013). This software application simplifies our daily lives. However, when there are actions that must be taken and these, for some reason, are not complete, the impact can be trivial or catastrophic, depending on its consequences Homès (2013), Whittaker (2000). Basically, it is necessary to test the soft wares because we all make mistakes, and their consequences can be quite devastating Uddin & Anand (2019), Van Veenendaal & Graham (2008). There are recognized reasons to justify why it is necessary to test, such as to find defects, to reduce the risk to ensure that the software meets the requirements,

to give confidence, to achieve compliance and to measure quality Spillner et al. (2014).

Defects, errors and failures are somewhat confusing concepts, as they all seem to have the same meaning, which is not true. When it comes to failures, it must be associated with the inability of a system / component to perform functions necessary for certain requirements. The defect is associated with an incorrect process or definition of data, within a software or document. The wrong actions caused by the human being are called errors Olsen et al. (2018)Lazić & Medan (2003).

As stated, people are using more and more software applications to their routines, which leads to a great importance that the system is functioning properly, thereby achieving the user accomplish what you want. For the software to contain the desired quality, tests, both manual and automated, must be carried out in a structured manner.

With this work, we aim to perform a comprehensive study on models. This article aims to understand the needs and purpose of the tests and their levels and life cycle. As there are different tests, manual and automated, this paper will clarify their definition and differences. Four tools will be addressed concerning test automation, thus identifying its characteristics, advantages, and disadvantages.

This article is organized as follows, section II addresses the reason for the need to test, and the reasons for the existence of defects in software. Section III addresses the topic of software testing, focusing on the testing life cycle and levels of software testing. Section IV addresses the types of tests, manual and automated. Finally, in section V, the conclusions of this article are discussed.

2 Why it is important to test?

In several actions of our daily life, we are dependent on software and its execution in a correct way, being in our electronic equipment, as in actions that we do, such as making a bank transfer or shopping online Homès (2013). When there are actions that, for some reason, are not complete, the impact can be trivial or catastrophic, depending on its consequences Homès (2013)Whittaker (2000). It is necessary to test because we all make mistakes, and their consequences can be quite devastating Van Veenendaal & Graham (2008)Uddin & Anand (2019). There are recognized reasons to justify why it is necessary to test Spillner et al. (2014):

- To find defects;
- To reduce the risk;
- To ensure that the software meets the requirements;
- To give confidence;
- To achieve compliance;
- To measure quality.

3 Software tests

Software tests in an overview are known as activities to find errors Lewis (2017)Rierson (2017) or processes that analyze a software item intending to find differences between existing conditions and the requirements guaranteeing software quality Agarwal et al. (2009). This qualification is associated with a measurement of attributes given the applicable expectations and standards.

The quality qualification theme is considered an ambiguous term depending on the interest and quality attributes of the entities involved. The term quality is part of our daily lives and has different meanings in popular and professional uses Kan (2003). In a popular view, quality can be felt and judged but cannot be measured, and this is because each person has his perception and interpretation of it Kan (2003). In a professional view, the concept of quality has to be defined so that it is possible to have a comprehensive definition for every professional aspect. Since this need existed, Juran and Gryna (1970) defined quality as the suitability of use de Mello Cordeiro (2004). It is necessary to consider the expectations and requirements of customers since they are the same who are going to use the product. It must be prepared for their needs, to be suitable for use, all customers must use the product even though they have different ways of doing it. In 1979, this definition was reinforced, adding that quality is to comply with requirements Crosby & Free (1979), which will imply clarity in its specification so that they are well interpreted.

The tests include essential standard activities, where software auditors must create the test requirements. These requirements serve as a basis for the creation of real values and scripts. These scripts are converted to executable tests being analyzed the output of these executions. With this analysis, we will have the determining test state, thus revealing a success or failure. With the analysis of test criteria coverage, they can decide the best input parameters in the test execution, thereby increasing the probability of finding problems, making the software more reliable Ammann & Offutt (2016).

Several people estimate that a successful test does not find errors. In a test view, this idea is not correct. A successful test finds errors Rierson (2017). In this process, both components and requirements are assessed manually or using automation tools, confirming that the behavior aligns with the intended requirements Hooda & Chhillar (2015).

3.1 Testing Life Cycle

The tests are considered the most critical phase of the software development life cycle Lee et al. (2020). It defines the various activities performed continuously and in sequence to evaluate the same Hooda & Chhillar (2015)Tang (2010)Kapur et al. (2021). The best programmers can all work in the same company. However, even if the best ones can make mistakes, it is possible to find these errors and correct themKapur et al. (2021). In addition to this, it is also a way of saving time and effort Mladenova (2020).

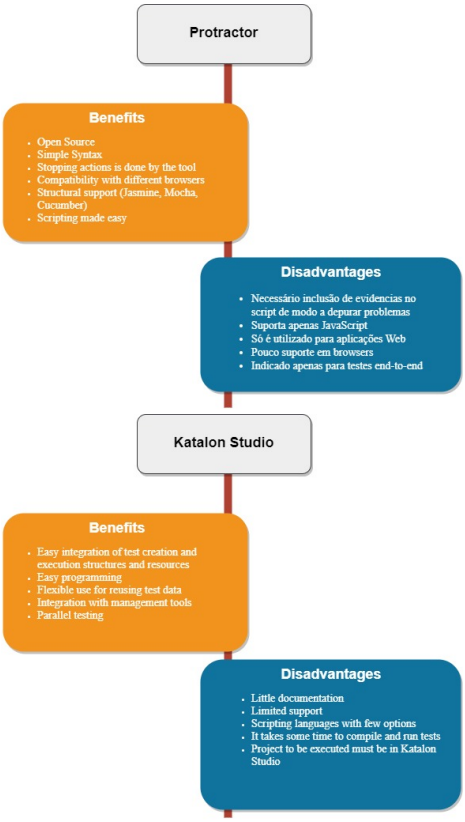

Fig. 1. Life cycle phases (part 1)

Each stage contains different objectives and results. Each stage is crucial in testing a specific product Tang (2010)Kapur et al. (2021), and there is coordinated work between the testing team and the development team Kapur et al. (2021). The life cycle comprises six phases, Figure 1 and Figure 2, which are:

– **Requirements analysis:** During this phase, functional requirements are be analyzed, where it is defined what the software should do, and non-functional requirements, which determine the security and performance of the system Hooda & Chhillar (2015)Lee et al. (2020)Kapur et al. (2021). In addition to the requirements, functional and technical specification documents are created Hooda & Chhillar (2015)Tang (2010). Thus, the team can acquire detailed knowledge of the tests to be carried out, being aware of their priorities Crosby & Free (1979). If some requirements are impossible to test due to system limitations and testing environment, the team must communicate them to the business team to plan the mitigation strategy Hooda & Chhillar (2015)Tang (2010).

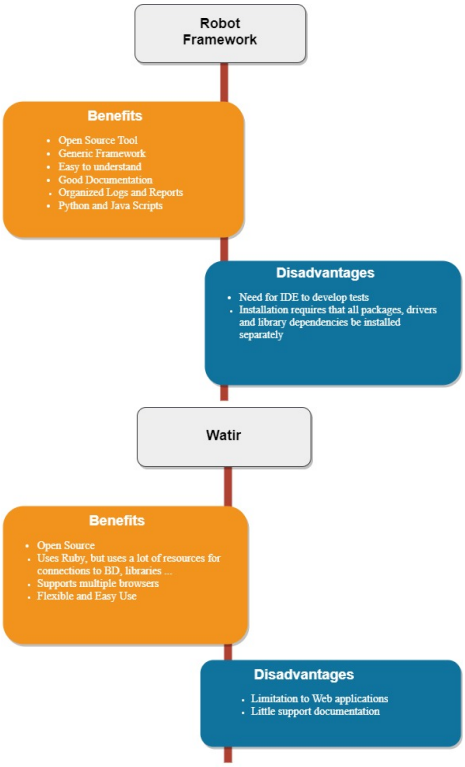

Fig. 2. Life cycle phases (part 2)

- **Planning phase:** This phase is one of the keys to the test's success Mladenova (2020). At this stage, there is a description of the context and objectives. After the product is understood, the team will analyze the risks and define test schedules and environments to create a strategy. After that, whoever is managing assigns roles and responsibilities to the rest of the people taking into account their skills Kapur et al. (2021). The questions of testing strategy, resources, responsibilities, and priorities are considered to guarantee the intended goals Tang (2010)Mladenova (2020).The test plan is created in a structured way, describing the topics previously described Van Veenendaal & Graham (2008)Hooda & Chhillar (2015)Lee et al. (2020)Mladenova (2020).
- **Design:** In this phase, the general objectives of the test are transformed into tangible test conditions Van Veenendaal & Graham (2008)Tang (2010). Inputs are identified to initiate the design of the test case, the creation of scenarios and data. Basically, in this phase, it is documented, in a detailed way, how to perform the test Mladenova (2020). If test automation is being developed, scripts are created at this stage. Before entering the Implementation phase, test cases, data, and automation scripts must be created Lee et al. (2020)Kapur et al. (2021).

- **Implementation and execution:** In the implementation phase, a high-level design is gathered, and the tests are started to be built and implemented, with or without the use of automation tools Van Veenendaal & Graham (2008)Lee et al. (2020). The test environments are set up with the requirements for execution. After system implementation and configuration, tests are performed Lee et al. (2020)Kapur et al. (2021). In this phase, the tests are carried out manually or automatically. After execution, the results are recorded, thus comparing the test output with the real scenario, validating that the execution output was expected. The errors found in the executions are sent to the development team to be corrected. After corrections, the development team will perform the tests to validate if the defect has been corrected (a confirmation test), ensuring that no other defects have been added with the corrections made Van Veenendaal & Graham (2008).
- **Closing the test cycle:** In this phase, the data of the test activities are collected to solidify experiences, trends, facts, and figures Van Veenendaal & Graham (2008)Lee et al. (2020). Tests can end for various reasons: a test performed successfully; information needed for the test is not well specified; the project manager canceled the project; the objectives were all achieved Van Veenendaal & Graham (2008). This phase is quite constructive, as it provides the test team with great detail of the entire process that has been carried out Kapur et al. (2021). With this detail, the team has the necessary material to gather, discuss and analyze what went well or wrong. Thus, they have the knowledge to adopt new strategies in the future, obtaining tremendous success in the following projects Lee et al. (2020).

3.2 Levels of Software Testing

Tests can be designed and built during all stages of software development. The most time-consuming parts of the test are the test's design and construction, so testing activities can and should be carried out throughout the development.

Each test level (acceptance tests, system tests, integration tests, and unit tests) accompanies each software development activity Ammann & Offutt (2016). The tests are composed of four levels: Acceptance tests; System tests; Integration tests, Unitary tests.

Customers carry out acceptance tests to validate whether the software complies with the indicated requirements Lewis (2017). At this stage, the customer will decide whether to accept or reject the software's delivery Agarwal et al. (2009). These tests can be carried out for long periods to discover clustered errors that can damage the system over time. System tests are tested in an operational environment before acceptance tests are performed Lewis (2017). These tests focus on finding bugs that may result from interactions between subsystems and system components while also checking existing non-functional characteristics Agarwal et al. (2009), including usability, performance, compatibility, and stress tests.

The integration tests are used to discover errors associated with the interface. All modules must be tested in an integrated manner. The test is created gradually

by slowly adding the modules that have already been tested unitarily Agarwal et al. (2009). This type of test aims to test whether it is possible to integrate the different modules without obtaining an error when passing parameters in each module's call Lewis (2017), creating an inclusion plan to detail the steps and sequences of the introduction/execution of the modules.

Concerning unit tests, they serve to evaluate software, focusing on implemented implementations based on individual components. This level focuses on verifying limits, performing internal interface analysis. In the smallest software design unit, each component is tested independently. This level aims to validate the code in the high and low-level design, ensuring that all branches are executed, validating error messages, codes, and output, and confirming that the code developed is consistent with what is specified in the files (LLD and HLD) Lewis (2017)Agarwal et al. (2009)Ammann & Offutt (2016).

4 Types of software tests

It is crucial to define manual and automated tests since software defects are detected by the human factor, more precisely by manual tests. These tests will be complemented by test automation Stefinko et al. (2016). It is quite difficult to find bugs in the system efficiently. Simultaneously, automation focuses on removing/reducing repetitive and straightforward tasks, finding flaws in these processes, and manual tests find most of the new bugs that can appear in the software Itkonen et al. (2009).

4.1 Manual tests

The Manual tests template is defined as tests performed manually Stefinko et al. (2016) by the people who develop the software or are directed to perform them. This type of test is performed without using tools or scripts, Bhatt (2017) being suitable for smaller projects Stefinko et al. (2016) or projects in which the specifications are always changing.

In this type of test, the human has to measure response times and compare results. It is not a good option, with a greater probability of human error, resulting in less reliability Bhatt (2017). With this type of test, it is possible to find real problems, something that can happen to the user, something that the automated tests cannot verify, as it is not prepared to validate a specific behavior Stefinko et al. (2016). Manual tests are prepared for change since it is quicker to change the paradigm of manual execution. The change in test automation requires changes to execution scripts, taking longer than manual testing Stefinko et al. (2016).

4.2 Automated Tests

In automated tests, tools and execution scripts are used Stefinko et al. (2016). These tests aim to repeat predefined actions comparing the requirements with

the actual test result Bhatt (2017). If these points do not match, it will be necessary to analyze the code and change it - this type of test covers more excellent coverage and speed in its execution Bhatt (2017).

This type of test has several advantages. The tests are carried out quickly and effectively, being a profitable process in the long run. The test data is filled in automatically, which leads to less exhaustion to perform the test. Anyone can validate any test's output since it is visible to everyone, not verified in manual tests. Compared to manual testing, automated testing is more beneficial for large projects Mailewa et al. (2015).

In the development of this type of test, there are six activities that we will have to take into account Garousi & Elberzhager (2017). 1. Test case design; 2. Script test; 3. Execution of the test; 4. Evaluation test; 5. Test results report; 6. Test management and other test engineering activities. There are several test automation tools available on the market, in which we will address their framework:

- **Katalon studio** is a platform associated with automated tests that contain several resources so that it is possible to automate tests for web applications, mobile applications, and API Umar & Chen (2019).
- **Robot Framework** is an open-source tool based on keyword-driven (allows services, data, and scripts to be independent of each other) Pajunen et al. (2011)Na & Huaichang (2015) developed to conduct tests at the level of acceptance Bisht (2013)Pajunen et al. (2011).
- **Protractor** is an end-to-end testing tool (testing flows/processes from start to finish), focusing on test automation in applications in Angular and Angular JS Bandwal et al. (2019).
- The **Watir tool** is open source, is used for the automation of web pages using Ruby scripting languages Islam (2016)Gogna (2014).

Each of them is adaptable to certain real situations. For example, if the robot framework fits into project A, it does not mean that projects B, C, and D have to use the same tool. These tools have several advantages and disadvantages, which can be analyzed before deciding which one to use.

5 Conclusions

Tests are created to find faults in the system. In a software test view, a successful test finds faults. The software tests include standard activities, of which test requirements must be created so that it is possible to compare the actual values with the execution scripts' values. We can ask, "why do failures occur in the system?", "Whether development projects are defined and structured?". Well, the answer is quite simple, it all happens because of the human being. A human being makes an error, this error causes a defect in the system, and in turn, with the execution of the system, the fault caused by the defect is visible. For the reasons mentioned above, testing is the most critical phase in software development. In addition to the testing phase, it is also possible to design and

carry out tests in all phases of development, considering that each phase of tests accompanies each phase of development. With this, it is possible to carry out tests in each phase correcting the actions that may contain errors.

Concerning software testing, we can target them in two ways, manual testing, and automated testing. Manual tests are performed manually by a human being. They are most effective on small projects or constantly changing, as they encounter problems that can be caused by human behavior. Tasks and execution scripts are used for automated tests, being more effective in large projects, routine projects, or well-defined processes.

With the definitions of test types, we can say that manual and automated tests are complementary. It is essential to consider these two types of tests in a software project. There are several testing tools on the market and an added value when it comes to testing automation.

Each one fits in a specific situation. Testers and developers must analyze the tools concerning the project where the automation of tests is necessary to validate which best fits.

5.1 Future work

In future work, we will develop practical test cases in each of the tools described, “katalon studio”, “robot framework”, “protractor”, and “watir”, to validate which one best suits a real situation. Besides, an application’s performance is essential, giving insight into how much load the software can handle to prevent real situations. The software does not respond to the expected behavior due to the number of existing users and requests. We will study two tools related to performance, JMETER and LOCUST, to validate the software performance and understand which one best fits the practical situation.

Acknowledgment

”This work is funded by National Funds through the FCT - Foundation for Science and Technology, IP, within the scope of the project Ref UIDB/05583/2020. Furthermore, we would like to thank the Research Centre in Digital Services (CISeD), the Polytechnic of Viseu for their support.”

Bibliography

Agarwal, B., Tayal, S. & Gupta, M. (2009), *Software engineering and testing*, Jones & Bartlett Learning.

Ammann, P. & Offutt, J. (2016), *Introduction to software testing*, Cambridge University Press.

Bandwal, S., Choudhary, A., Kulkarni, S., Bhase, K., Shahani, S. & Pawar, A. (2019), Automation framework for testing dynamic configurable tool, *in* '2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)', IEEE, pp. 1–5.

Bhatt, D. (2017), 'A survey of effective and efficient software testing technique and analysis', *Iconic Research and Engineering Journals (IREJOURNALS)* **326**.

Bisht, S. (2013), *Robot framework test automation*, Packt Publishing Ltd.

Crosby, P. B. & Free, Q. I. (1979), 'The art of making quality certain', *New York: New American Library* **17**, 174–83.

de Mello Cordeiro, J. V. B. (2004), 'Reflexões sobre a gestão da qualidade total: fim de mais um modismo ou incorporação do conceito por meio de novas ferramentas de gestão?', *Revista da FAE* **7**(1).

Garousi, V. & Elberzhager, F. (2017), 'Test automation: not just for test execution', *IEEE Software* **34**(2), 90–96.

Gogna, N. (2014), 'Study of browser based automated test tools watir and selenium', *International Journal of Information and Education Technology* **4**(4), 336.

Homès, B. (2013), *Fundamentals of software testing*, John Wiley & Sons.

Hooda, I. & Chhillar, R. S. (2015), 'Software test process, testing types and techniques', *International Journal of Computer Applications* **111**(13).

Islam, N. (2016), 'A comparative study of automated software testing tools', *St. Cloud State University* .

Itkonen, J., Mantyla, M. V. & Lassenius, C. (2009), How do testers do it? an exploratory study on manual testing practices, *in* '2009 3rd International Symposium on Empirical Software Engineering and Measurement', IEEE, pp. 494–497.

Kan, S. H. (2003), *Metrics and models in software quality engineering*, Addison-Wesley Professional.

Kapur, P., Panwar, S. & Kumar, V. (2021), Should software testing continue after release of a software: A new perspective, *in* 'Handbook of Advanced Performability Engineering', Springer, pp. 709–737.

Lazić, L. & Medan, M. (2003), Software quality engineering versus software testing process, *in* 'The Telecommunication Forum (TELFOR) journal: Beograd'.

Lee, S. H., Lee, S. J., Koo, S. R., Varuttamaseni, A., Yue, M., Li, M., Cho, J. & Kang, H. G. (2020), 'Optimization of software development life cycle quality for npp safety software based on a risk-cost model', *Annals of Nuclear Energy* **135**, 106961.

Lewis, W. E. (2017), *Software testing and continuous quality improvement*, CRC press.

Mailewa, A., Herath, J. & Herath, S. (2015), A survey of effective and efficient software testing, *in* 'The Midwest Instruction and Computing Symposium.(MICS), Grand Forks, ND'.

Mladenova, T. (2020), Software quality metrics–research, analysis and recommendation, *in* '2020 International Conference Automatics and Informatics (ICAI)', IEEE, pp. 1–5.

Na, Q. & Huaichang, D. (2015), Extension and application based on robot testing framework, *in* '2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)', IEEE, pp. 428–431.

Olsen, K., Parveen, T., Black, R., Friedenberg, D., Zakaria, E., Hamburg, M., McKay, J., Walsh, M., Posthuma, M., Smith, M. et al. (2018), 'Certified tester foundation level syllabus', *Proc. Int. Softw. Testing Qualifications Board (ISTQB)* pp. 1–96.

Pajunen, T., Takala, T. & Katara, M. (2011), Model-based testing with a general purpose keyword-driven test automation framework, *in* '2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops', IEEE, pp. 242–251.

Rierson, L. (2017), *Developing safety-critical software: a practical guide for aviation software and DO-178C compliance*, CRC Press.

Spillner, A., Linz, T. & Schaefer, H. (2014), *Software testing foundations: a study guide for the certified tester exam*, Rocky Nook, Inc.

Stefinko, Y., Piskozub, A. & Banakh, R. (2016), Manual and automated penetration testing. benefits and drawbacks. modern tendency, *in* '2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET)', IEEE, pp. 488–491.

Tang, J. (2010), Towards automation in software test life cycle based on multiagent, *in* '2010 International Conference on Computational Intelligence and Software Engineering', IEEE, pp. 1–4.

Uddin, A. & Anand, A. (2019), 'Importance of software testing in the process of software development', *IJSRD-International J. Sci. Res. Dev.* **6**, 2321–0613.

Umar, M. A. & Chen, Z. (2019), 'A study of automated software testing: Automation tools and frameworks', *International Journal of Computer Science Engineering (IJCSE)* **6**, 217–225.

Van Veenendaal, E. & Graham, D. (2008), '"foundations of software testing: Istqb certification', *Cengage Learning EMEA* p. 30.

Whittaker, J. A. (2000), 'What is software testing? and why is it so hard?', *IEEE software* **17**(1), 70–79.