Software Testing

Carlo van Kessel
S-DB-IP3 and S-DB-GP3
S3-DB03
Hans van Heumen, Marc van Grootel
03-01-2023
V2

Version control

Version	Changes
1	Initial version.
2.	Added How do you test software?

Contents

DOT Research Framework

For this research report I use the DOT framework. This research framework is split in three main parts:

1. The "What" of my research (the domains)
2. The "Why" of my research (the trade-offs)
3. The "How" of my research (the strategies and methods)

For more information about the DOT framework:
https://ictresearchmethods.nl/The_DOT_Framework

“What” is software testing?

Software testing is the process of evaluating a software application or system to determine whether it meets the specified requirements and works as intended. It involves executing the software, analysing the results, and comparing them with the expected behaviour. The goal of testing is to identify any defects or issues with the software, so that they can be fixed before the software is released to users.

Types of tests

1. Unit testing
2. Integration testing
3. Functional testing
4. End-to-end testing
5. Regression testing
6. Acceptance testing
7. Performance testing
8. Stress testing
9. Usability testing

“Why” is software testing necessary?

Software testing is necessary to ensure that a program or application is working correctly and meets the specified requirements. It helps to identify any defects or issues in the software and ensures that the software is of high quality and fit for its intended purpose. Without testing, it would be difficult to ensure that the software is dependable and performs as intended. Testing helps to catch and fix defects early in the development process, which can save time and resources in the long run. It also helps to improve the overall quality of the software and the user experience.

Benefits of software testing

1. Identifying defects and issues early in the development process, which can save time and money by allowing them to be fixed before the software is released to users.
2. Ensuring that the software meets the specified requirements and works as intended.
3. Improving the quality of the software by identifying and fixing defects.
4. Providing confidence in the software, both for the development team and for end users.
5. Improving the reliability and stability of the software.
6. Identifying potential performance bottlenecks and improving the performance of the software.
7. Identifying potential security vulnerabilities and fixing them before the software is released.
8. Improving the usability of the software by identifying and fixing issues that may cause confusion or frustration for end users.
9. Providing a way to verify that changes to the software have not introduced new defects or issues.
10. Helping to ensure that the software is fit for its intended purpose and meets the needs of the end user.

“How” do you test software?

As I stated before in “Different types of tests” there are a lot of types. Here I will explain what these tests are and how to know if you have done them correctly.

Unit testing

Unit testing is a type of software testing that involves evaluating individual units or components of a software application to verify that they are working as intended. A unit is the smallest testable part of an application, and it typically represents a single function or method. Unit tests are designed to test the functionality of a unit in isolation, without relying on the rest of the application. They are typically written by the developer as part of the development process, and they are run automatically every time the code is changed. The goal of unit testing is to identify defects or issues with a unit as early as possible in the development process, so that they can be fixed before the software is released. To determine whether a unit test has been successful, the developer compares the output of the unit under test with the expected results. If the output matches the expected results, the test is considered to have passed. It is important to have enough unit tests to provide coverage for all aspects of the unit's functionality. A code coverage tool can be used to determine the percentage of the code that is covered by unit tests.

Integration testing

Integration testing is a type of software testing that verifies the interaction between different components or systems. It is a systematic approach to building, testing, and delivering software. The goal of integration testing is to ensure that different parts of the system work together as expected and that the system behaves correctly. Integration testing typically involves combining multiple units or components and testing them as a group to verify that they function correctly when working together. This may involve testing the integration of different subsystems, testing the integration of the system with external systems or resources, or testing the integration of different components within a subsystem. Integration testing is typically performed after unit testing and is a crucial step in the software development process because it helps to ensure that the system is working correctly and is ready for use.

Functional testing

Functional testing is a type of software testing that is used to verify that a software application is working as intended and meets the specified requirements. It involves evaluating the functional aspects of the application, such as its user interface, database access, security, and performance. The goal of functional testing is to ensure that the application is working correctly and is fit for its intended purpose. To perform functional testing, testers create test cases that exercise the various functions of the application and use test data and test scripts to automate the testing process. They then compare the output of the tests with the expected results to determine whether the tests have been successful. Functional testing is a key step in the software development process because it helps to ensure that the application is of high quality and is ready for use by the end user.

End-to-end testing

End-to-end testing is a type of software testing that verifies the flow of an application from start to finish. It involves testing the entire application, including all components and interfaces, to ensure that it is working correctly and meets the specified requirements. The goal of end-to-end testing is to identify defects or issues that may occur when different parts of the application are combined and work together. To perform end-to-end testing, testers create test cases that exercise the various functions of the application from end to end. They may also use test data and test scripts to automate the testing process. End-to-end testing is typically performed after unit and integration

testing and is a crucial step in the software development process because it helps to ensure that the application is working correctly and is ready for use by the end user.

Regression testing

Regression testing is a type of software testing that is used to verify that changes to a software application have not introduced new defects or issues. It involves re-executing a selection of previously executed test cases to ensure that the existing functionalities of the application are still working correctly. The goal of regression testing is to identify defects or issues that may have been introduced because of changes to the application, and to ensure that the application is still working correctly. This testing is performed to confirm that new code changes do not have unintended side effects on the existing functionality of the application. Regression testing may involve testing the application's user interface, integration with external systems, security, performance, and other aspects of its functionality. To perform regression testing, testers typically create test cases that exercise the various functions of the application, and they may use test data and test scripts to automate the testing process. To determine whether a regression test has been successful, the tester compares the output of the test with the expected results. If the output matches the expected results, the test is considered to have passed.

Acceptance testing

Acceptance testing is a type of software testing that determines whether an application is ready for use by the end user. It is performed to verify that the application meets the specified acceptance criteria and functions correctly. Acceptance testing is usually carried out by the end user or the client and is the final phase of testing before the software is deployed to the production environment. Test cases are created to exercise the various functions of the application and test data and test scripts may be used to automate the testing process. The output of the tests is compared with the expected results to determine whether the tests have been successful.

Performance testing

Performance testing is a type of software testing that is used to evaluate the performance of a software application under a particular workload. It is designed to test the speed, response time, stability, reliability, scalability, and resource usage of the application. The main goal of performance testing is to identify and resolve performance bottlenecks in the application, to ensure that it can handle the required workload. Performance testing is a subset of performance engineering and is also known as "Perf Testing." It is a key step in the software development process because it helps to ensure that the application can handle the expected usage and perform at an acceptable level.

Stress testing

Stress testing is a type of software testing that is used to evaluate the stability and reliability of a software application under extreme load conditions. The goal of stress testing is to measure the application's ability to withstand heavy workloads and to ensure that it does not crash or fail under crunch situations. It involves testing the application beyond normal operating points and evaluating how it performs under extreme conditions. Stress testing is also known as endurance testing. During stress testing, the application under test (AUT) is subjected to a brief period of intense stress to determine its withstanding capacity. One common use of stress testing is to identify the limits at which the system or software breaks. It is also used to evaluate the effectiveness of the system's error management capabilities under extreme conditions. Stress testing can be used to simulate real-world usage scenarios and to identify potential performance issues before the application is deployed.

Usability testing

Usability testing, also known as user experience (UX) testing, is a method of evaluating how easy and user-friendly a software application is. It involves having a small group of target end-users use the application to identify any defects that may affect usability. The goal of usability testing is to assess the ease of use, flexibility, and effectiveness of the application from the user's perspective. It focuses on the user's experience of using the application and how well it meets its intended objectives. Usability testing is typically performed during the design phase of the software development life cycle (SDLC) to identify any potential issues that may affect the user experience. This testing is important because it helps to ensure that the application is easy and intuitive to use, which is essential for user satisfaction and adoption.

Sources

https://en.wikipedia.org/wiki/Software_testing
https://www.guru99.com/software-testing-introduction-importance.html
https://www.geeksforgeeks.org/benefits-of-software-testing/
https://www.guru99.com/unit-testing-guide.html
https://www.guru99.com/integration-testing.html
https://www.guru99.com/functional-testing.html
https://www.guru99.com/end-to-end-testing.html
https://www.guru99.com/regression-testing.html
https://www.guru99.com/user-acceptance-testing.html
https://www.guru99.com/performance-testing.html
https://www.guru99.com/stress-testing-tutorial.html
https://www.guru99.com/usability-testing-tutorial.html

https://www.youtube.com/watch?v=QIDhzBg5eWY