Report on testing techniques in software development

Assessment 1

Systems Development – Testing software

Robert Lothian

HND Computer Science – First Year

Contents

Introduction

(Page 2)

What is the purpose of testing?

(page 2)

Test documentation in the software development lifecycle

(page 3)

What are the test levels in software development?

(Page 4)

WHAT IS WHITE AND BLack box testing?

(page 7)

Bibliography

(page 8)

**Introduction**

In this report we will be looking at various facets of testing. Beginning with why we test, why we document these tests, a look into the four test levels, investigating what they involve, when they are performed and their purpose. Whilst doing this we will take a closer look into a selection of testing types for these test levels and discuss what their purpose is.

**What is the purpose of testing?**

The many reasons why we test software, however these all serve the larger goal of ensuring that software is performing well and to the specifications of the client. We do this by finding defects that may be made by the programmer when they are developing the software (bugs) and testing to prevent further defects. We also test to ensure that the software meets our clients brief and in doing so we gain the customers confidence by delivering a quality product. This is done by having good test coverage in order to test the application thoroughly to the client’s specification. It is also important that we evaluate the work being done to ensure that the product is meeting the requirements set. This builds confidence in the level of quality of the product, not only for the client but also for the team. By sharing our test data with stakeholders, we allow them to make better inform decisions on the direction of the project, meaning that the end result is closer to their needs.

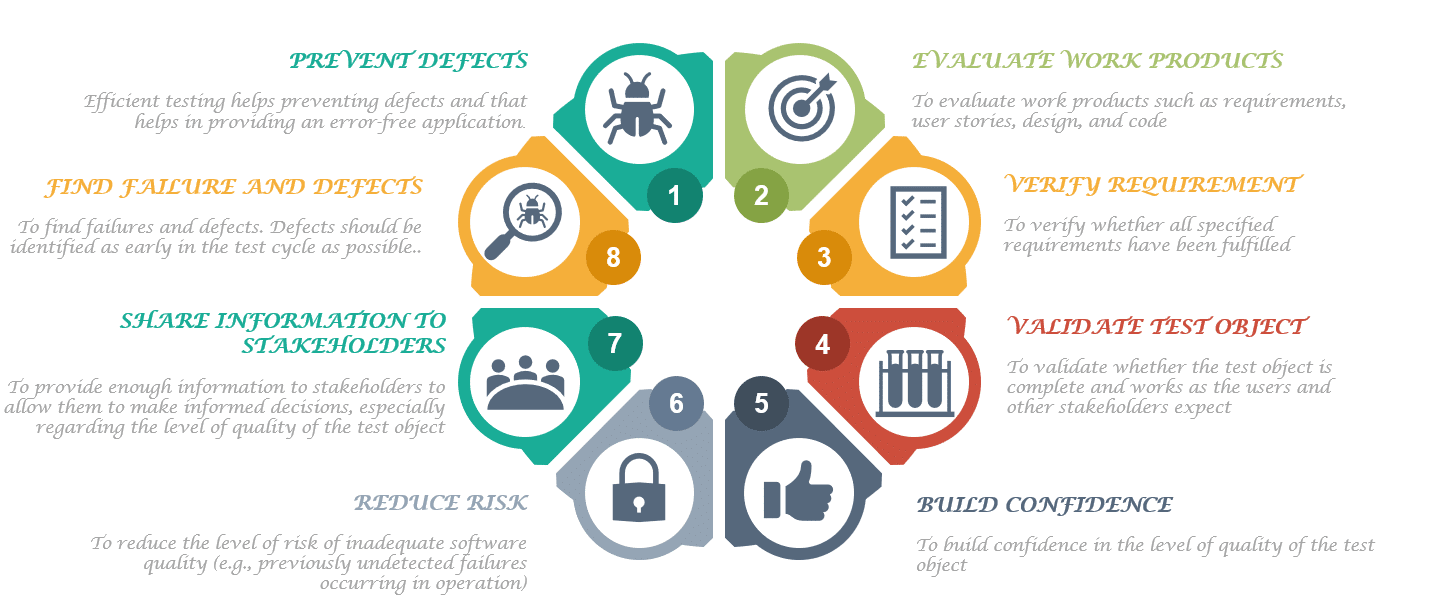


Figure Why do we test?

**Test documentation in the Software Development Lifecycle**

Testing is a vital part of the Software Development Lifecycle; the process helps teams in validating functional and non-functional features and requirements. In most Software Development Lifecycle models (agile models particularly) testing is a constant, this is to ensure that new features do not negatively influence previously deployed features, by doing this we can catch these issues before they can impact the end product. Implementing testing early means that we can identify and resolve issues as they come to light rather than having an extensive test period at the end of development where fixes made could break previously working features.

The importance of documenting these tests cannot be understated. It is vital to keep accurate documentation to ensure that the project testing is systematic in its approach. Having tests repeated due to a lack of recorded data is not only a drain on resources but also on time. It also allows us to demonstrate to the client that we have been thorough, this improves transparency to the client, which will help build and maintain a good relationship with them, increases their satisfaction and ultimately improves the chances of working with the client again.

Without good documentation it can be hard to keep track of a projects progress and it reflects poorly on the quality of the product. This can foster an unhealthy working atmosphere and lead to the loss of clients.

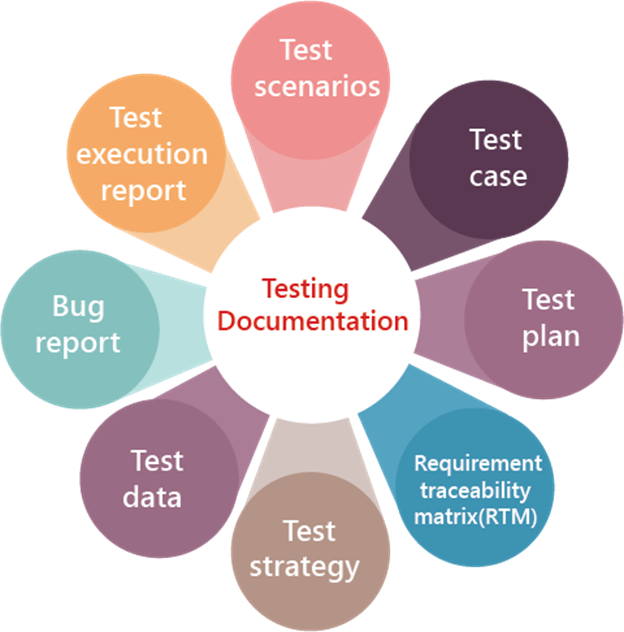


Figure Types of testing documentation

**What are the test levels in software development?**

**UNIT TESTING:** Unit testing is where individual components (or units) of software are tested. The goal of this is to ensure that each component is working as expected. It is performed during the during the coding phase of development. These components can be a function, method, object, module or procedure. During the SDLC it is the first form of testing carried out with Integration testing normally the next testing carried out.

The benefits of thorough Unit testing are that it eliminates bugs early in the project, it can be used in project documentation and it can help with code re-use, meaning that both the code and testing can be migrated into future projects.

Unit testing can be done either manually or automatically. Usually, it is automated to limit the man hours dedicated to it however, when performed manually a step-by-step instructional document may be employed. Examples of automated testing tools commonly used for unit testing are Junit (a free Java method testing tool), NUnit (an open source tool for testing .net languages), JMockit (an open source tool which allows mocking API with recording and verification syntax), EMMA (an open source Java testing tool covering, method, line and basic block) and PHPUnit (a program that takes small potions of PHP code and tests each separately).

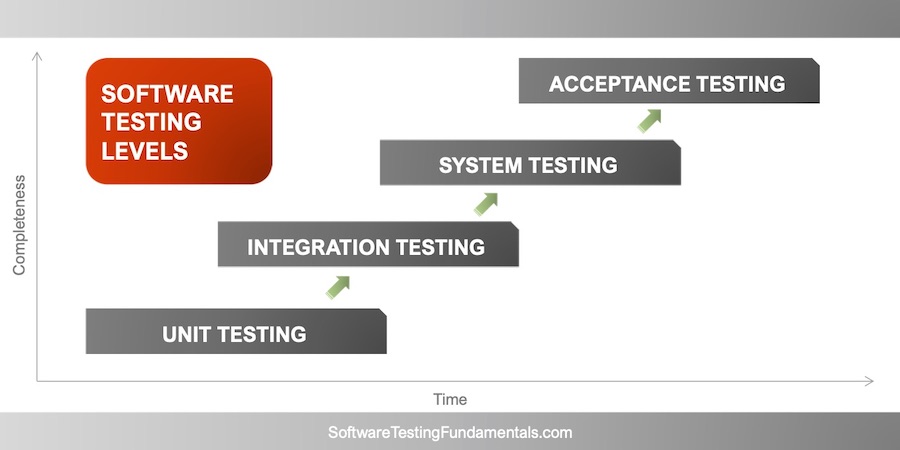


Figure 3 Testing level chart based on time and completeness of project

**INTEGRATION TESTING:** Integration testing is where components, often coded by different programmers, are brought together and tested as a group. The purpose of this is to ensure that these components are working correctly together and to root out any defects in how they interact with each other. There are a variety of approaches we can take in the execution of Integration testing.

* *Big Bang Approach* – in this approach all the components are integrated together then tested as a whole. If one of the components is incomplete, then the test will not work. This approach is well suited to testing smaller systems with less components. It is hard to find where faults are localized and as all components need to be complete it allows for less time for testing to be carried out.
* *Incremental Approach* – this approach is done by integrating two or more modules that are related and testing the functionality between them. The other related modules are added incrementally, and the process continues until all related modules are integrated and tested successfully. This approach is carried out by two different methods.

*Bottom-up* – lower-level modules are tested first these are then used to test higher level modules; this continues until all modules at the top are tested. This makes finding where faults occur easier and there is no need to wait until all modules are completed (unlike the Big-Bang approach), however, the critical modules that are tested last may be susceptible to defects and early prototyping is not possible.

*Top-down* – this method takes place from the top to bottom following the control of the software system. Higher level modules are tested first then the lower-level modules are integrated and tested. Again, this allows for finding where faults occur easier, it also means that prototyping can occur at an early stage and because critical modules are being tested first it allows for major design flaws to be found and fixed first. This does however mean that lower-level modules are tested inadequately and requires many Stubs (a dummy module that simulates data communication).

**SYSTEM TESTING:** System testing is performed on a complete software product. Its purpose is to test how the software interacts with other software and hardware systems; this is also called End to End testing scenario. It is very thorough and checks every input of the application to ensure the desire outcome is achieved. It also tests the users experience in using the application. As this is such an extensive testing level with more than 50 different types, we will focus just on the types a large developer would commonly use.

* *Usability testing* - this focuses on the user experience. We test to see how easy the application is to use; how flexible it is in handling controls and how well it meets the requirements set out for it.
* *Load testing* – this is to see how well the program performs under day to day working loads.
* *Regression testing* – this involves testing to ensure that none of the changes made or additional modules added during development have caused bugs.
* *Recovery Testing* – this ensures that the software is reliable and that it can recover from crashes.
* *Migration testing* – this ensures that the software can be moved from older systems to current systems without causing any issues.
* *Functional testing* – this is also known as functional completeness testing. Its purpose is to see in any other functions could be added to improve the software.
* *Hardware/Software Testing* – this focuses on the interaction between the software and hardware during testing, ensuring that those interactions are working correctly.

Due to the large number of types of System testing available, it is important to choose the right tests for the right project. Important factors to take into consideration are time, budget and resources available to the team.

**ACCEPTANCE TESTING:** Acceptance testing is used to determine whether software has met the requirement specifications. There are various forms of Acceptance testing.

* *User acceptance testing* – a type of testing performed by the end user/client to accept the software before it is moved into production
* *Business acceptance testing* – this testing ensures that the software meets the business and operational needs in the real world. It validates the software against business needs.
* *Alpha testing* – is used to identify all issues or bugs before a program is released to end users. It does this by identifying the tasks a typical user might do and test them.
* *Beta testing* – performed by “real users” of the software in a “real environment”. It’s the final test before shipping the product to customers.

During acceptance testing the following attributes are tested for; data integrity, functionality, data conversion, usability, performance, timeliness, security, installability and ungradeability, scalability and documentation.

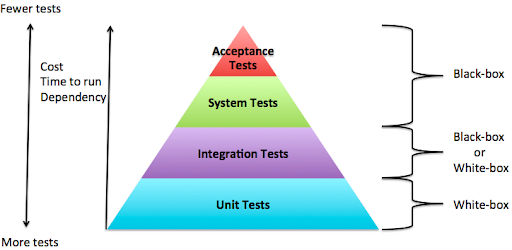


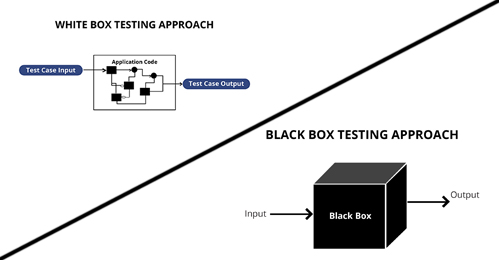
Figure demonstrating the Cost, Time to run dependencies, number of tests and whether it is black or white box testing based.

**What is White and Black box testing?**

In Black Box testing the internal workings of the code are not known to the tester. It focusses on the behaviour of the software testing from an end-user perspective. It is ideal for higher level testing such as Systems and Acceptance testing. It is hard to automate so it is usually performed manually. Access to the code is not needed so this type of testing can be outsourced and does not require detailed knowledge of coding or programming languages. It is well suited to large code segments.

White Box testing checks the internal function of the software. Therefore, testers are required to have a knowledge of the code and is sometimes known as glass box, clear box, transparent box or code-based testing. It is suited best to lower-level testing like Unit testing and Integration testing. It is easy to automate and as it is a time-consuming process that requires an expert tester, with vast experience and detailed knowledge of the code it predominantly uses automated testing. One drawback of this is that automated test cases can become redundant if the code base is changing rapidly.

It therefore can be stated that white box tests are focussed on structural elements while black box focusses on functional elements.



**Bibliography**

Level, I., Manager, I., Tutorial, A., Tests, I., Us, C., Policy, P., Use, T., Us, A., Samala, J., Mishra, A. and Guide, I., 2021. *What are software testing objectives and purpose?*. [online] Tryqa.com. Available at: <http://tryqa.com/what-is-the-software-testing-objectives-and-purpose/> [Accessed 30 April 2021].

2021. [online] Available at: <https://performancelabus.com/software-testing-importance-sdlc/#:~:text=Testing%20is%20one%20of%20the,product%20fulfills%20the%20client's%20needs.> [Accessed 30 April 2021].

Mansabdar, K., 2021. *Why is Documentation Important in Software Testing?*. [online] Blog.e-zest.com. Available at: <https://blog.e-zest.com/why-is-documentation-important-in-software-testing/> [Accessed 30 April 2021].

Guru99.com. 2021. *Unit Testing Tutorial: What is, Types, Tools & Test EXAMPLE*. [online] Available at: <https://www.guru99.com/unit-testing-guide.html#:~:text=UNIT%20TESTING%20is%20a%20type,an%20application%20by%20the%20developers.> [Accessed 30 April 2021].

Guru99.com. 2021. *Integration Testing: What is, Types, Top Down & Bottom Up Example*. [online] Available at: <https://www.guru99.com/integration-testing.html> [Accessed 4 May 2021].

Guru99.com. 2021. *What is System Testing? Types & Definition with Example*. [online] Available at: <https://www.guru99.com/system-testing.html> [Accessed 4 May 2021].

Tutorialspoint.com. 2021. *Acceptance Testing - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/software\_testing\_dictionary/acceptance\_testing.htm> [Accessed 4 May 2021].

Guru99.com. 2021. *What is User Acceptance Testing (UAT)? with Examples*. [online] Available at: <https://www.guru99.com/user-acceptance-testing.html> [Accessed 4 May 2021].

Trigent.com. 2021. [online] Available at: <https://www.trigent.com/assets/pdf/white-paper/Trigent\_WhitePaper\_opd\_Business-Acceptance-Testing-Is-Your-Product-User-Ready.pdf> [Accessed 4 May 2021].

Guru99.com. 2021. *Alpha Testing Vs Beta Testing: What's the Difference?*. [online] Available at: <https://www.guru99.com/alpha-beta-testing-demystified.html> [Accessed 4 May 2021].

Guru99.com. 2021. *Black Box Testing Vs. White Box Testing: Key Differences*. [online] Available at: <https://www.guru99.com/back-box-vs-white-box-testing.html> [Accessed 4 May 2021].