|  |
| --- |
|  |
| An Introduction to software testing |
| By Chris Dworczyk |

|  |
| --- |
| Administrator  9/13/2017 |

Contents

[The Purpose of this Report 2](#_Toc497311391)

[What is high quality Software 2](#_Toc497311392)

[Why test software 2](#_Toc497311393)

[Understand the role of a tester 2](#_Toc497311394)

[How to test software 2](#_Toc497311395)

[What is a test plan? 2](#_Toc497311396)

[The Purpose of a test plan 3](#_Toc497311397)

[Key testing strategies and approaches 3](#_Toc497311398)

[Stratagies 3](#_Toc497311399)

[Testing Approaches 3](#_Toc497311400)

[Static 3](#_Toc497311401)

[Dynamic 4](#_Toc497311402)

[Modern Tests methods 4](#_Toc497311403)

[Unit testing 4](#_Toc497311404)

[INterface Testing 4](#_Toc497311405)

[System TEsting 4](#_Toc497311406)

[Automated Testing 4](#_Toc497311407)

[Usability Testing 5](#_Toc497311408)

[Acceptance testing 5](#_Toc497311409)

[Alpha test 5](#_Toc497311410)

[Beta Test 5](#_Toc497311411)

[Appreciating the importance of test documentation 5](#_Toc497311412)

[Works Cited 6](#_Toc497311413)

# The Purpose of this Report

This report will be discussing the importance of testing in software development as well as the different strategies and approaches that can be used in testing and why it is so important to keep everything documented.

# What is high quality Software

High quality software can mean many things such as it’s free of any bugs or crashes when using it for its intended use. This is also called reliable software as its intended use matches its specifications. However we can also take it a step further in making it robust, which means it’s programmed in such a way as to expect bad user input to try and avoid any user created crashes. High quality software is well coded but also well documented to provide an easy way of maintaining it.

To ensure that a program is of the highest quality an agile testing methodology is adapted during the Software development life cycle (SDLC). This means that at the end of each stage of development there are test strategies created and run on the program to ensure that throughout the development it’s constantly being tested and inspected for any flaws or problems that can be addressed and fixed early in development.

# Why test software

The obvious answer is to make sure that the software works correctly and doesn’t have any bugs or crashes when the end user is using it.

The software needs to be tested to ensure that the program matches the specifications given by the client and everything runs as expected. It is also key to make sure the program meets the expectations of the client and the requirements for the intended use of that program. Sometimes the program may be error free but it does not fully

If software is no tested correctly it can lead to small problems where the customers may encounter bugs or crashes and will want the software debugged. However testing may be a much more important step in software that deals with serious world issues that if the software gets wrong may result in catastrophic results. For example the software in planes needs to be thoroughly checked to make sure everything is working correctly as aboard there can be upwards of 400 people.

# Understand the role of a tester

The role of a tester is to design testing suites and testing scenarios, he should have sound understanding of software test design and test execution methodologies.

He should be able to communicate very well with his colleagues – as well as the client – to be able to extract necessary information to provide a reliable test. He will then have to pass on his findings to the development team for any further development.

# How to test software

## What is a test plan?

A test plan is a very important piece of document which specifies every aspect of how the software will be tested. It includes the areas which will be tested, what data and the expected results. (Test Plan fundementals)

## The Purpose of a test plan

A test plan will include many important testing scenarios and the test data to test in them. It will also have a comprehensive coverage of all the tests and how they will be performed.

It is important to test all the components and units of a program to ensure it matches the client’s expectations and the requirement specification.

## Key testing strategies and approaches

### Stratagies

#### Top down

Top down testing is done by making your way down from the biggest modules to the smaller one creating STUB modules in place of any modules that are not complete. A STUB module is created to simulate the real one if the currently tested module depends on it.

An advantage of top down testing is that if there are any major bugs in the big modules they can be easily addressed.

The disadvantage is that the STUB modules have to be created in the first place, they are sometimes notorious for being tricky to make in some cases. (Top down Testing vs Bottom up Testing)

#### Bottom Up

The bottom up approach is the opposite and instead you are testing the modules at the bottom of the hierarchy and working you’re way up creating modules called DRIVERS to simulate all the unfinished main modules.

This type of testing would help in catching any bugs or glitches in the code near the bottom modules of the program.

A testing of this kind may be appropriate for object-oriented system as the SQA sources describes it. (SQA, 2006). Since the individual objects can be tested with their individual simulated modules.

## Testing Approaches

### Static

The execution of code is not necessary with static testing as it is tested manually or by automated tools. You would usually have two steps to this testing with review being the first:

This is to eliminate any problems in the documentation of the program like the requirement specification, design, test, and any other that may be needed.

The code is the run thorough special software to look at the structure of the code itself and try find any ambiguities or sections of code that may lead to problems later in the development.

### Dynamic

#### White Box

This method requires knowledge of programming and is sometimes referred to as “glass-box testing”. This is because you can image the module/method you are testing is in a glass box where you can see all the code running and the variable changing values. This way if an error is thrown it can be looked further into and possibly recreated to try and find a solution for it.

#### Black Box

As Carey (Wodehouse) in the source says it’s when the tester has no access or even sometimes experience with code. An input is fed into the program and the result is the only thing that matters and is looked at. This way you’re only testing the functionality of the program that it doesn’t crash or throw any errors. It also checks for any interface problems that a user may encounter.

# Modern Tests methods

## Unit testing

Unity testing is part of the test driven development where each of the smallest parts of the program is looked at in a lot of detail independently away from all the other working parts of the program to ensure it’s working to a correct standard.

## INterface Testing

The interface testing is much different from “User Interface” where you would check all the controls (buttons/radio buttons/text boxes), labels and pictures have the correct data in them and work correctly. In interface testing you are checking that your piece of software is working correctly with another software system and that all the data flows correctly between them. The interface is the thing connecting those two software systems together. You’d also need to make sure that all errors are caught and dealt with correctly.

An interface example would be like a web server and application server interface.

## System TEsting

System testing takes places once all the other units are tested and integrated into the full program.

Once the full program is put together it would be then tested again against the program requirement specification to see of it works as intended.

## Automated Testing

Automated testing is what it sounds like, an automated way of testing the code. Often too complicated or expensive for small developer teams yet very powerful and useful for bigger companies who can afford the time and resources into it.

## Usability Testing

The usability test is usually done by the users who will be using the software system in the end. They would be sat down in front of the software and asked to perform tasks. The developer would then take notes and listen to try and find any problems in the interface or any other area the user encountered a problem.

## Acceptance testing

The acceptance testing is the also step where the whole program is looked at as a whole and checked if it matches requirements requested by the customer/business.

### Alpha test

Alpha testing is performed near the end of the development testing and before beta testing. During this phase the testing is usually performed internally by the developers to try and spot any bugs that everyday user may encounter. To do this they would use techniques such as White and Black box testing to try and simulate real world scenarios.

### Beta Test

On the other hand beta testing is performed usually done by a selected few of real world users in an environment the software will be usually running in. The developers get to see their program being approached from many different ways and sometimes find a new bug from a user who used the program in such a away it crashed or caused a bug. It’s sometimes hard to find bugs yourself when you’ve looked at nothing but the code you write so it’s very useful to receive user validation.

# Appreciating the importance of test documentation

Documenting the whole testing process is essential in the development life cycle. Testing can sometimes be as much as half the entire project. Saying that it is very important that it is done correctly as not only will this save time and money but also will make the maintenance much better.

To ensure testing is reliable and as efficient as possible documentation can aid in such tasks as the developers will have something to look at for easier understanding of the program.

It is important to document the testing results as they can be passed to the developers for any further development needed. It can also be passed onto the client to gauge their acceptance of the program.

# Works Cited

SQA. (2006). *Testing Strategies*. Retrieved 2017, from HN Computing: https://www.sqa.org.uk/e-learning/SDPL03CD/page\_19.htm

*Test Plan fundementals*. (n.d.). Retrieved from Software Testing Fundementals: http://softwaretestingfundamentals.com/test-plan/

*Top down Testing vs Bottom up Testing*. (n.d.). Retrieved from softwaretestingstuff: http://www.softwaretestingstuff.com/2007/10/top-down-testing-vs-bottom-up-testing.html

Wodehouse, C. (n.d.). *White-box Testing vs. Black-box Testing*. Retrieved from Upwork: https://www.upwork.com/hiring/development/white-box-testing-vs-black-box-testing/