QUALITY MANAGEMENT AND IMPROVEMENT

Activity C: *Testing and metrics related to Quality Management Improvement*

Jordi Rafael Lazo Florensa

21 May 2021

Degree in Computer Engineering

Universitat de Lleida
Escola Politècnica Superior

Contents

List of Figures

1 Introduction

Today's world of technology is completely dominated by machines, and their behavior is controlled by the software powering it. Software testing provides the solution to all our worries about machines behaving the exact way we want them to. In this document, in addition to defining and explaining what testing is, the importance of quality in testing and its control will also be discussed. Finally will provide an in-depth knowledge about how testing works and a small example.

2 Quality Assurance (QA)

Quality assurance (QA) is a way of preventing mistakes and defects in manufactured products and avoiding problems when delivering products or services to customers; which ISO 9000 defines as "part of quality management focused on providing confidence that quality requirements will be fulfilled". Through audits and other forms of assessment, quality assurance efforts detect and correct problems or variances that fall outside established standards or requirements.

In other words, quality assurance ensures a high level of quality during the development of products or services.

Some examples of Quality Assurance metrics in software testing:

- Identify the main issues that need to be tested.
- Choose a specialist who knows what to do with the metrics.
- Test the most important aspects of the software.
- Calculate the metric's efficiency.
- Note if you need any changes.
- Improve the aspects you want to change.

3 Quality Control (QC)

Quality control (QC) is a procedure or set of procedures intended to ensure that a manufactured product or performed service adheres to a defined set of quality criteria or meets the requirements of the client or customer. ISO 9000 defines quality control as "A part of quality management focused on fulfilling quality requirements".

QC is similar to, but not identical with, quality assurance (QA). While QA refers to the confirmation that specified requirements have been met by a product or service, QC refers to the actual inspection of these elements.

4 Quality Testing (QT)

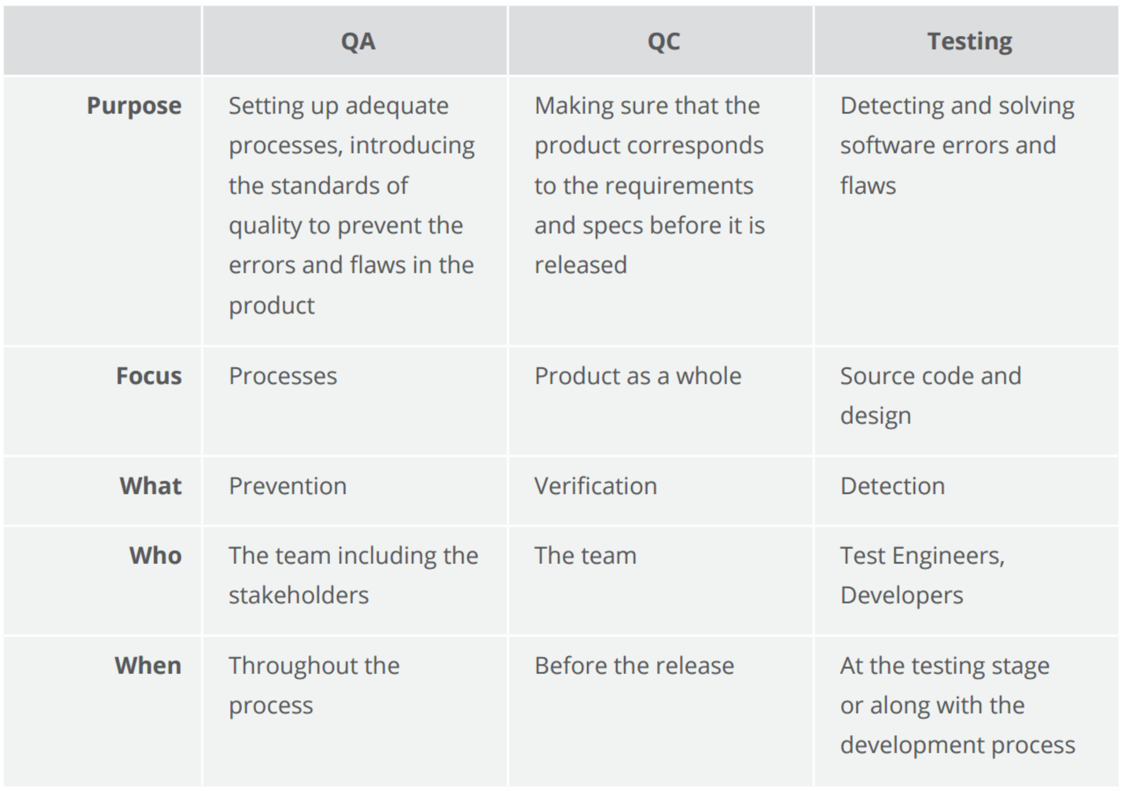
Testing can be described as the process of evaluating a system or its component(s) with the purpose to find whether it meets the specified requirements or not. Simply, testing is executing a system with an intention to identify any possible errors, gaps, or missing requirements in contradiction with the current requirements.

4.1 Why Quality Testing is important

- Quality testing is essential to indicate the defects and errors that may occur during the development phases.
- It's important since it verifies the customer's reliability and their content with the application.
- It is very important to guarantee the Quality of the product. Quality product supplied to the customers helps to gain their confidence.
- Testing is necessary while providing the facilities to the customers like the delivery of high quality product or software application which requires minimal maintenance cost and hence shows more precise, consistent and reliable results.
- Testing is required for an effective performance of software application or product.
- It's important to ensure that the application should not result into any failures because it can lead to extra expenses in the future or in the later stages of the development.
- It's required to stay in the business.

4.2 Quality Testing Objectives

- Detecting software defects that may be created by the programmer while being developed.
- Gaining confidence in software application and providing information about the level of quality.
- Preventing defects.
- Verifying that the final result meets the business and user requirements.
- Ensuring that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.
- Gaining customers confidence by providing them a quality product.

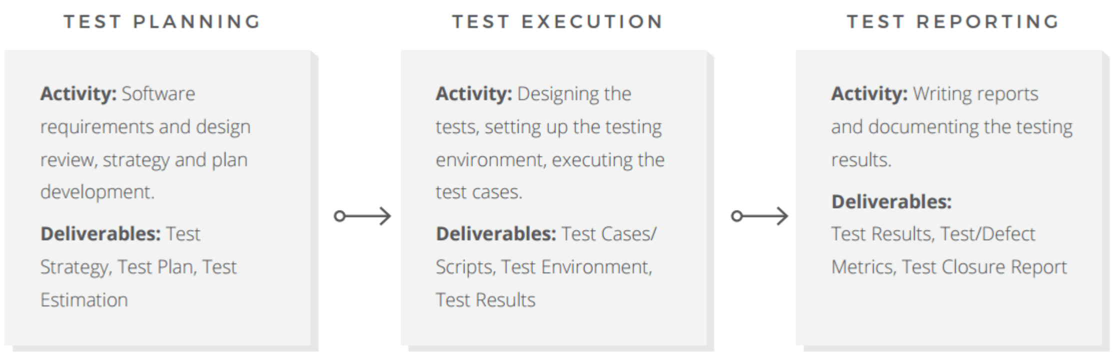


	QA	QC	Testing
Purpose	Setting up adequate processes, introducing the standards of quality to prevent the errors and flaws in the product	Making sure that the product corresponds to the requirements and specs before it is released	Detecting and solving software errors and flaws
Focus	Processes	Product as a whole	Source code and design
What	Prevention	Verification	Detection
Who	The team including the stakeholders	The team	Test Engineers, Developers
When	Throughout the process	Before the release	At the testing stage or along with the development process

Figure 1: *The concepts of quality assurance, quality control, and testing compared.*

5 Software Testing

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.[3]



TEST PLANNING

Activity: Software requirements and design review, strategy and plan development.

Deliverables: Test Strategy, Test Plan, Test Estimation

TEST EXECUTION

Activity: Designing the tests, setting up the testing environment, executing the test cases.

Deliverables: Test Cases/ Scripts, Test Environment, Test Results

TEST REPORTING

Activity: Writing reports and documenting the testing results.

Deliverables: Test Results, Test/Defect Metrics, Test Closure Report

Figure 2: *The stages of software testing.*

5.1 Principles of Software Testing

Testing shows presence of mistakes. Testing is aimed at detecting the defects within a piece of software. But no matter how thoroughly the product is tested, we can never be 100 percent sure that there are no defects. We can only use testing to reduce the number of unfound issues.

Exhaustive testing is impossible. There is no way to test all combinations of data inputs, scenarios, and preconditions within an application. For example, if a single app screen contains 10 input fields with 3 possible value options each, this means to cover all possible combinations, test engineers would need to create 59,049 (310) test scenarios. And what if the app contains 50+ of such screens? In order not to spend weeks creating millions of such less possible scenarios, it is better to focus on potentially more significant ones.

Early testing. As mentioned above, the cost of an error grows exponentially throughout the stages of the SDLC. Therefore it is important to start testing the software as soon as possible so that the detected issues are resolved and do not snowball.

Defect clustering. This principle is often referred to as an application of the Pareto principle to software testing. This means that approximately 80 percent of all errors are usually found in only 20 percent of the system modules. Therefore, if a defect is found in a particular module of a software program, the chances are there might be other defects. That is why it makes sense to test that area of the product thoroughly.

Pesticide paradox. Running the same set of tests again and again won't help you find more issues. As soon as the detected errors are fixed, these test scenarios become useless. Therefore, it is important to review and update the tests regularly in order to adapt and potentially find more errors.

Testing is context dependent. Depending on their purpose or industry, different applications should be tested differently. While safety could be of primary importance for a fintech product, it is less important for a corporate website. The latter, in its turn, puts an emphasis on usability and speed.

Absence-of-errors fallacy. The complete absence of errors in your product does not necessarily mean its success. No matter how much time you have spent polishing your code or improving the functionality if your product is not useful or does not meet the user expectations it won't be adopted by the target audience.

5.2 The Types of Software Testing

Testing is a process of executing a software program to find errors in the application being developed. Testing is critical for deploying error-free software programs. Each type of testing has its advantages and benefits. Software testing is broadly categorized into two types; Functional and Non-Functional testing.

5.2.1 Functional Testing

Functional Testing is used to verify the functions of a software application according to the requirements specification. Functional testing mainly involves black box testing and does not depend on the source code of the application.

Functional Testing involves checking User Interface, Database, APIs, Client/Server applications as well as security and functionality of the software under test. Functional testing can be done either manually or by making use of automation.

The various types of Functional Testing include the following:

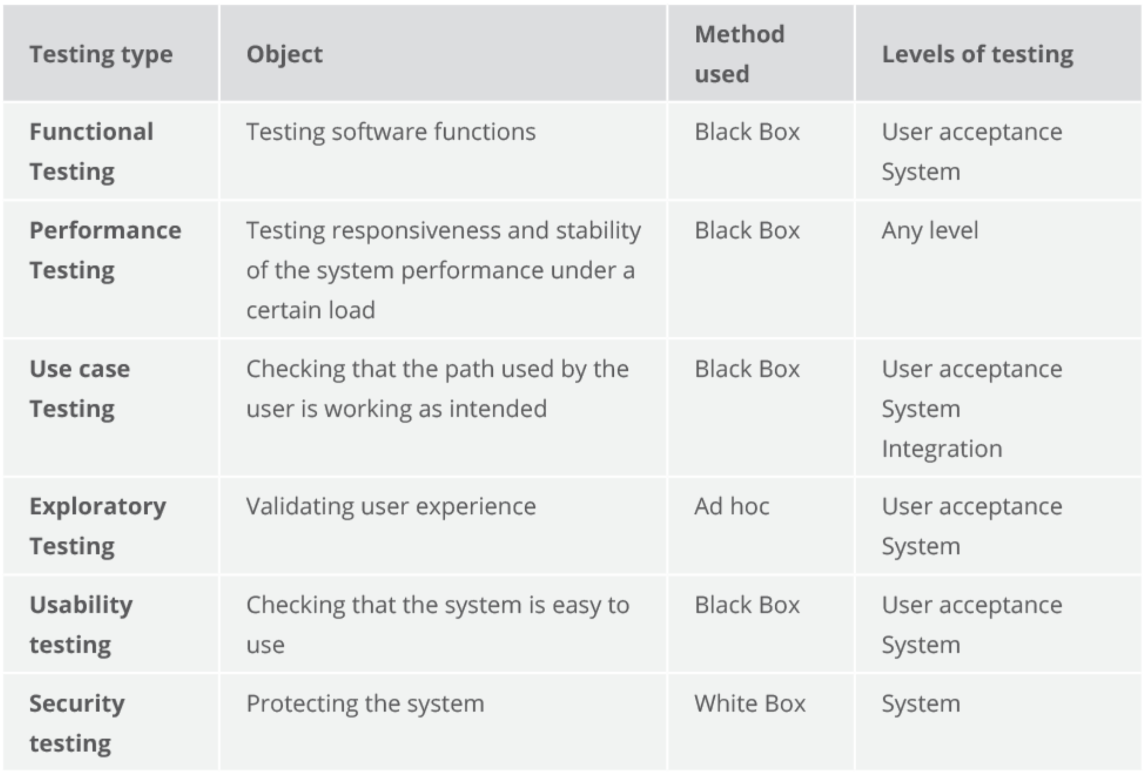
- Unit Testing
- Integration Testing
- System Testing
- Sanity Testing
- Smoke Testing
- Interface Testing
- Regression Testing
- Beta/Acceptance Testing

5.2.2 Non-Functional Testing

Non-Functional Testing is done to check the non-functional aspects such as performance, usability, reliability, and so on of the application under test.

The various types of Non-Functional Testing include the following:

- Performance Testing
- Load Testing
- Stress Testing
- Volume Testing
- Security Testing
- Compatibility Testing
- Install Testing
- Recovery Testing
- Reliability Testing
- Usability Testing
- Compliance Testing
- Localization Testing



Testing type	Object	Method used	Levels of testing
Functional Testing	Testing software functions	Black Box	User acceptance System
Performance Testing	Testing responsiveness and stability of the system performance under a certain load	Black Box	Any level
Use case Testing	Checking that the path used by the user is working as intended	Black Box	User acceptance System Integration
Exploratory Testing	Validating user experience	Ad hoc	User acceptance System
Usability testing	Checking that the system is easy to use	Black Box	User acceptance System
Security testing	Protecting the system	White Box	System

Figure 3: *The most popular testing types.*

5.3 The most common methods of Software Testing

Black-box Testing: Black-box testing is applied to verify the functionality of the software by just focusing on the various inputs and outputs of the application rather than going deep into its internal structure, design, or implementation. Black-box testing is performed from the user's perspective.

White-Box Testing: The White-Box software testing strategy tests an application with access to the actual source code as well as focusing on the internal structure, design, and implementation. This testing method is known by different names such as Open Box testing, Clear Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing, and Structural Testing. White-box testing offers the advantage of rapid problem and bug spotting.

Acceptance Testing: Acceptance Testing is a QA (Quality Assurance) process that determines to what extent a software attains the end user's approval. Also known as UAT (User Acceptance Testing) or system testing, it can be testing the usability or the functionality of the system or even both. Depending on the enterprise, acceptance testing can take the form of either end-user testing, beta testing, application testing, or field testing. The advantage of acceptance testing is that usability issues can be discovered and fixed at an early stage.

5.4 Software Testing Metrics

Software Testing Metrics are the quantitative measures used to estimate the progress, quality, productivity and health of the software testing process. The goal of software testing metrics is to improve the efficiency and effectiveness in the software testing process and to help make better decisions for further testing process by providing reliable data about the testing process.

A Metric defines in quantitative terms the degree to which a system, system component, or process possesses a given attribute.

Software testing metrics or software test measurement is the quantitative indication of extent, capacity, dimension, amount or size of some attribute of a process or product.

These are some examples of the most common Software Testing metrics:

- # of test cases – How many tests there are to execute?
- # of test cases passed – This metric shows how many of the total test cases were passed and provides a good sense of “quality” or lack thereof.
- # of test cases failed – This metric provides insight into weaknesses and bugs and can determine shortcomings in the development process. It gives information on the total number of bugs found during the test.
- # of test cases skipped – Any skipped test cases are those that remain outstanding and still need to be run.
- #of test cases not started – These test cases have not been performed yet, so they cannot be logged as passed, failed, or skipped.
- % complete – What is the percentage of completed tests? This metric is great for tracking the progress of the overall testing project, which can help drive other decisions.
- % test cases passed – A useful measurement that gives insight into the overall status of “quality.”
- # of test runs – A metric that shows how many end-to-end tests exist and gives a good sense of the overall workload.

6 Quality tool: pylint

Pylint is a tool that checks for errors in Python code, tries to enforce a coding standard and looks for code smells. It can also look for certain type errors, it can recommend suggestions about how particular blocks can be refactored and can offer you details about the code’s complexity.

6.1 How to install pylint

To install it, run the following command:

```
pip install pylint
```

The command to use pylint in a Python file is:

```
pylint filename.py
```

Returns results consisting of semantic errors, syntax errors, coding style errors, errors in the code, excessive and redundant code, and so on. It also assigns a score that indicates whether Python code is ideal to use and maintains a history of scores obtained when running a Python file, as well as after each edit.

In the next section, you can check out sample program demonstrating its usage.

6.2 Pylint example

Here is a simple program (pylint_sample.py) having some styling issues.

```
pycon = {2016: "Portland", 2018: "Cleveland"}
europython = {2017: "Rimini", 2018: "Edinburgh", 2019: "Basel"}

print({**pycon, **europython})
```

To run pylint we have:

```
pylint pylint_sample.py
```

And the following result is obtained:

```
************* Module pylint_sample
pylint_sample.py:1:0: C0114: Missing module docstring (missing-module-
    docstring)

-----------------------------------
Your code has been rated at 6.67/10
```

If we make a change to the code, the modified version looks like this:

```
'''
code to update dicts
'''

pycon = {2016: "Portland", 2018: "Cleveland"}
europython = {2017: "Rimini", 2018: "Edinburgh", 2019: "Basel"}

print({**pycon, **europython})
```

Then we run the code again and we have:

```
-------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)
```

7 Conclusions

Software testing is an important part of the software development process. It is not a single activity that takes place after code implementation, but is part of each stage of the lifecycle. A successful test strategy will begin with consideration during requirements specification. Testing details will be fleshed through high and low level system designs, and testing will be carried out by developers and separate test groups after code implementation. As with the other activities in the software lifecycle, testing has its own unique challenges. As software systems become more and more complex, the importance of effective, well planned testing efforts will only increase. Moreover, applying their extensive knowledge of the product, testers can bring value to the customer through additional services, like tips, guidelines, and product use manuals. This results in reduced cost of ownership and improved business efficiency.

References

[1] Wikipedia. *Quality Assurance.*
`https://en.wikipedia.org/wiki/Quality`$_a$*ssurance*

[2] Strong LTD. *Software Testing.*
`https://strongqa.com/qa-portal/knowledge-base/key-concepts/software-testing`

[3] Guru99. *What is Software Testing? Definition, Basics Types.*
`https://www.guru99.com/software-testing-introduction-importance.html1`

[4] Altexsoft. *Quality Assurance, Quality Control and Testing — the Basics of Software Quality Management.*
`https://www.altexsoft.com/whitepapers/quality-assurance-quality-control-and-testing-the-basics-of-software-quality-management/`

[5] Pylint. *Pylint User Manual.*
`http://pylint.pycqa.org/en/latest/`

[6] Diceus. *Top 10 software quality metrics that matter.*
`https://diceus.com/top-7-software-quality-metrics-matter/`

[7] Wikipedia. *Software testing.*
`https://en.wikipedia.org/wiki/Software_testing`