Course Coordinator: Ms.Simra Najm
Designation: Assistant Professor.
E-mail : simranajm@neduet.edu.pk.

Introduction and Basic Concept of Software Testing & its Importance

LECTURE # 1

Software Quality Factors:

Flexibility and Extensibility:

Flexibility is the ability of software to add/modify/remove functionality without damaging current system. Extensibility is the ability of software to add functionality without damaging system, so it may be thought as a subset of flexibility. Those functionality changes may occur according to changing requirements or an obligation if development process is one of the iterative methods. Change is inevitable in software development and so, this is one of the most important properties of quality software

Maintainability and Readability

Maintainability is a little similar with flexibility but it focuses on modifications about error corrections and minor function modifications, not major functional extensibilities. It can be supported with useful interface definitions, documentations and also self-documenting code and/or code documentation. The more correct and useful documentation exists, the more maintainability can be performed.

Performance and Efficiency

Performance is mostly about response time of the software. This response time should be in acceptable intervals (e.g. max. a few seconds), and should not increase if transaction count increases. And also, resources are expensive. Efficiency must be supported with resource utilization. As an exaggerated example, ability of performing a simple function only by using a 32 processor machine or 1 TB disk space is not acceptable. Optimal source/performance ratio must be aimed.

Scalability

A scalable system responds user actions in an acceptable amount of time, even if load increases. Of course more hardware may be added for handling increasing user transaction, but the architecture should not change while doing this. This is called vertical scalability. Ability of running on multiple, increasing count of machines is multiple processing. If the software can perform that type of processing, this is called horizontal scalability. A preferred scalable system should suit both of these methods.

Availability, Robustness, Fault Tolerance and Reliability: *A robust software should not lose its availability even in most failure states. Even if some components are broken down, it may continue running. Besides, even if whole application crashes, it may recover itself using backup hardware and data with fault tolerance approaches. There should always be B and even C, D plans. Reliability also stands for the integrity and consistency of the software even under high load conditions. So it is relevant with availability and scalability. An unreliable system is also unscalable.*

Usability and Accessibility

*User interfaces are the only visible parts of software according to the viewpoint of user. So, simplicity, taking less time to complete a job, fast learnability etc. are very important in this case. The most well known principle for this property is (**Keep It Simple**). Simple is always the best. A usable software should also support different accessibility types of control for people with disabilities.*

Platform Compatibility and Portability

A quality software should run on as much various platforms as it can. So, more people can make use of it. In different contexts we may mention different platforms, this may be OS platforms, browser types etc. And portability is about adapting software that can run on different platforms, for being more platform compatible. In this sense, portability is also related with flexibility

Testability and Manageability

Quality software requires quality testing. Source code should be tested with the most coverage and with the most efficient testing methods. This can be performed by using encapsulation, interfaces, patterns, low coupling etc. techniques correctly. Besides testability, a qualified software should be manageable after deployment. It may be monitored for e.g. performance or data usage status, or may enable developer to configure system easily. Creating a successful logging system is another very important issue about manageability.

Security

Security is a very important issue on software development, especially for web or mobile based ones which may have millions of users with the ability of remote accessing to system. You should construct a security policy and apply it correctly by leaving no entry points. This may include authorization and authentication techniques, network attack protections, data encryption and so on. all possible types of security leaks should be considered, otherwise one day only one attack may crash your whole applicaion and whole company.

Functionality and Correctness

Functionality (or correctness) is the conformity of the software with actual requirements and specifications. In fact this is the precondition attribute of an application, and maybe not a quality factor but we wanted to point that as the last quality factor, for taking attention: Quality factors are not meaningful when we are talking about unfunctional software. First, perform desired functionality and produce correct software, then apply quality factors on it. If you can perform both parallel, it is the best.

SOFTWARE TESTING

Is *the root for developing a quality software. Testing is done to ensure the Quality of software including all quality factors.*

Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools. Some prefer saying Software testing as a white box and Black Box Testing

.

***Software testing** is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect free in order to produce the quality product*

***According to ANSI/IEEE 1059** standard – A process of analyzing a software item to detect the differences between existing and required conditions (i.e., defects) and to evaluate the features of the software item.*

Testing Strategy:

To perform testing in a planned and systematic manner, software testing strategy is developed.

A testing strategy is used to identify the levels of testing which are to be applied along with the methods, techniques, and tools to be used during testing.

This strategy also decides test cases, test specifications, test case decisions, and puts them together for execution.(Testing Life Cycle/test plan)

The choice of software testing strategy is highly dependent on the nature of the developed software

A healthy software testing or QA strategy requires tests at all technology stack levels to ensure that every part, as well as the entire system, works correctly.

1. ***Leave time for fixing**. Setting aside time for testing is pointless if there is no time set aside for fixing. Once problems are discovered, developers required time to fix them and the company needs time to retest the fixes as well. With a time and plan for both, then testing is not very useful.*
2. ***Discourage passing the buck**. The same way that testers could fall short in their reports, developers could also fall short in their effort to comprehend the reports. One way of minimizing back and forth conversations between developers and testers is having a culture that will encourage them to hop on the phone or have desk-side chat to get to the bottom of things. Testing and fixing are all about collaboration. Although it is important that developers should not waste time on a wild goose chase, it is equally important that bugs are not just shuffled back and forth.*

3. ***Manual testing has to be exploratory****. A lot of teams prefer to script manual testing so testers follow a set of steps and work their way through a set of tasks that are predefined for software testing. This misses the point of manual testing. If something could be written down or scripted in exact terms, it could be automated and belongs in the automated test suite. Real-world use of the software will not be scripted, thus testers must be free to probe and break things without a script.*

4. ***Encourage clarity****. Reporting bugs and asking for more information could create unnecessary overhead costs. A good bug report could save time through avoiding miscommunication or a need for more communication. In the same way, a bad bug report could lead to a fast dismissal by a developer. These could create problems. Anyone reporting bugs should make it a point to create bug reports that are informative. However, it is also integral for a developer to out of the way to effectively communicate as well.*

5. ***Test often.*** *The same as all other forms of testing, manual testing will work best when it occurs often throughout the development project, in general, weekly or bi-weekly. This helps in preventing huge backlogs of problems from building up and crushing morale. Frequent testing is considered the best approach.*
Testing and fixing software could be tricky, subtle and political even. Nevertheless, as long as one is able to anticipate and recognize common issues, things could be kept running smoothly

- ✓ *The output produced by the software testing strategy includes a detailed document, which indicates the entire test plan including all test cases used during the testing phase. A testing strategy also specifies a list of testing issues that need to be resolved.*
- ✓ *An efficient software testing strategy includes two types of tests, namely, lowlevel tests and high-level tests. Low-level tests ensure correct implementation of small part of the source code and high-level tests ensure that major software functions are validated according to user requirements. A testing strategy sets certain milestones for the software such as final date for completion of testing and the date of delivering the software. These milestones are important when there is limited time to meet the deadline.*

In spite of these advantages, there are certain issues that need to be addressed for successful implementation of software testing strategy. These issues are discussed here.

1. *In addition to detecting errors, a good testing strategy should also assess portability and usability of the software.*
2. *It should use quantifiable manner to specify software requirements such as outputs expected from software, test effectiveness, and mean time to failure which should be clearly stated in the test plan.*
3. *It should improve testing method continuously to make it more effective.*

4. *Test plans that support rapid cycle testing should be developed. The feedback from rapid cycle testing can be used to control the corresponding strategies.*
5. *It should develop robust software, which is able to test itself using debugging techniques.*
6. *It should conduct formal technical reviews to evaluate the test cases and test strategy. The formal technical reviews can detect errors and inconsistencies present in the testing process.*

Test Strategy is also known as test approach defines how testing would be carried out. TEST APPROACHES has two techniques:

•*(a)* ***Proactive*** *- An approach in which the test design process is initiated as early as possible in order to find and fix the defects before the build is created.*

•*(b)* ***Reactive*** *- An approach in which the testing is not started until after design and coding are completed.*

Types of Software Testing Strategies

There are different types of software testing strategies, which are selected by the testers depending upon the nature and size of the software. The commonly used software testing strategies are listed below

1. ***Analytic testing strategy:*** *This uses formal and informal techniques to access and prioritize risks that arise during software testing. It takes a complete overview of requirements, design, and implementation of objects to determine the motive of testing. In addition, it gathers complete information about the software, targets to be achieved, and the data required for testing the software****.(Pro active)***

2. ***Model-based testing strategy:*** *This strategy tests the functionality of the software according to the real world scenario (like software functioning in an organization). It recognizes the domain of data and selects suitable test cases according to the probability of errors in that domain****.(Reactive)***
3. ***Methodical testing strategy:*** *It tests the functions and status of software according to the checklist, which is based on user requirements. This strategy is also used to*

test the functionality, reliability, usability, and performance of the software. ***(Reactive)***

4. ***Process-oriented testing strategy:*** *It tests the software according to already existing standards such as the IEEE standards. In addition, it checks the functionality of the software by using automated testing tools.* ***(Pro active) (Reactive)***
5. ***Dynamic testing strategy:*** *This tests the software after having a collective decision of the testing team. Along with testing, this strategy provides information about the software such as test cases used for testing the errors present in it.* ***(Reactive)***
6. ***Philosophical testing strategy:*** *It tests the software assuming that any component of the software can stop functioning anytime. It takes help from software developers, users and systems analysts to test the software.* ***(Reactive)***
7. **Standard-compliant approach** specified by industry-specific standards.

Factors to be considered in choosing a Software Strategy:

- *Risks of product or risk of failure or the environment and the company*
- *Expertise and experience of the people in the proposed tools and techniques.*
- *Regulatory and legal aspects, such as external and internal regulations of the development process*
- *The nature of the product and the domain*

1. ***RISKS****. Risk management is paramount during testing, thus consider the risks and the risk level. For an app that is well-established that's slowly evolving, regression is a critical risk. That is why regression-averse strategies make a lot of sense. For a new app, a risk analysis could reveal various risks if choosing a risk-based analytical strategy.*
2. ***OBJECTIVES****. Testing should satisfy the requirements and needs of stakeholders to succeed. If the objective is to look for as many defects as possible with less up-front time and effort invested, a dynamic strategy makes sense.*
3. ***SKILLS.*** *Take into consideration which skills the testers possess and lack, since strategies should not only be chosen but executed as well. A standard compliant strategy is a smart option when lacking skills and time in the team to create an approach.*
5. ***PRODUCT.*** *Some products such as contract development software and weapons systems tend to have requirements that are well-specified. This could lead to synergy with an analytical strategy that is requirements-based.*
6. ***BUSINESS.*** *Business considerations and strategy are often important. If using a legacy system as a model for a new one, one could use a model-based strategy.*

7. ***REGULATIONS.*** *At some instances, one may* ***not only*** *have to satisfy stakeholders, but regulators as well. In this case, one may require a methodical strategy which satisfies these regulators.*

You must choose testing strategies with an eye towards the factors mentioned earlier, the schedule, budget, and feature constraints of the project and the realities of the organization and its politics.

CONCLUSION:

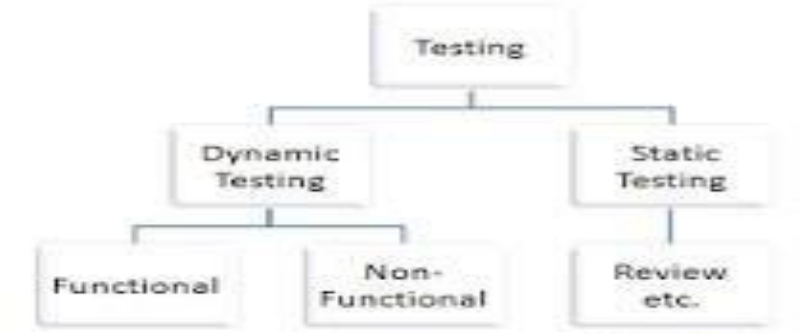
A Test Strategy document is a high level document and normally developed by Project Manager. This document defines "Software Testing Approach" to achieve testing objectives.

The Test Strategy is normally derived from the Business Requirement Specification document.

The Test Strategy document is a static document meaning that it is not updated too often. It sets the standards for testing processes and activities and other documents such as the Test Plan draws its contents from those standards set in the Test Strategy Document.

Some companies include the "Test Approach" or "Strategy" inside the Test Plan, which is fine and it is usually the case for small projects. However, for larger projects, there is one Test Strategy document and different number of Test Plans for each phase or level of testing.

TESTING METHODS



Static Testing?

- ✓ *Under **Static Testing** code is not executed. Rather it manually checks the code, requirement documents, and design documents to find errors. Hence, the name "static".*
- ✓ ***Static Testing Techniques*** provide a powerful way to improve the quality and productivity of software development *by finding errors in early stages of the development cycle.*
- ✓ *Static Testing* may be conducted manually or through the use of various software testing tools. It starts early in the ***Software Development Life Cycle*** and so it is done during the ***Verification Process***.
- ✓ *Static testing* is not magic and it should not be considered a replacement for ***Dynamic Testing***, but all software organizations should consider using reviews in all major aspects of their work including *requirements, design, implementation, testing, and maintenance*.
- ✓ *It examines work documents and provides review comments Work document can be of following:*

- *Requirement specifications*
- *Design document*
- *Source Code*
- *Test Plans*
- *Test Cases*
- *Test Scripts*
- *Help or User document*

- ***Types of defects that are easier to find during static testing are:***

- ✓ *Deviations from standards,*
- ✓ *Missing requirements,*
- ✓ *Design defects,*
- ✓ *Non-maintainable code and Inconsistent interface specifications.*

Techniques of Static Testing

- ***Review*** *– Typically used to find and eliminate errors or ambiguities in documents such as requirements, design, test cases, etc.*
- ***Informal Review***
- ***WalkThrough***
- ***Technical Review***
- ***Inspection***

Static Analysis *–*

The code written by developers are analysed (usually by tools) for structural defects that may lead to defects.

- ***Static Analysis***
- ***Data Flow***
- ***Control Flow***

Advantages of Static Testing:

- *Since static testing can start early in the life cycle, early feedback on quality issues can be established.*
- *By detecting defects at an early stage, rework costs are most often relatively low.*
- *Since rework effort is substantially reduced, development productivity figures are likely to increase.*
- *The evaluation by a team has the additional advantage that there is an exchange of information between the participants.*
- *Static tests contribute to an increased awareness of quality issues.*

Dynamic Testing?

- ✓ *Under **Dynamic Testing** code is executed. Dynamic execution is applied as a technique to detect defects and to determine quality attributes of the code*
- ✓ *It checks for functional behavior of software system , memory/cpu usage and overall performance of the system. Hence the name "Dynamic"*
- ✓ *Main objective of this testing is to confirm that the software product works in conformance with the business requirements. This testing is also called as Execution technique or **Validation testing.***
- ✓ *Dynamic testing executes the software and validates the output with the expected outcome. Dynamic testing is performed at all levels of testing and it can be either black or white box testing.*

.

Dynamic Testing *and* ***Static Testing*** *are complementary methods, as they tend to find different types of defects effectively and efficiently. But as it does not start early in the* ***Software Development Life Cycle*** *hence it definitely increases the cost of fixing defects.*
*It is done during **Validation Process** evaluating the finished product.*

Dynamic Techniques are subdivided into three more categories:

1. ***Specification Based Testing :*** *This includes both* ***Functional Testing*** *and* ***Non Functional Testing.***
2. ***Structure Based Testing***
3. ***Experience Based Testing***

Advantages of Dynamic Testing

1. ***Considered High Level Exercise****: It can always find errors that static testing cannot find and that is the reason why it is considered as high level exercise.*

2. ***Improves Quality****: Executing the software leads to the chances of finding more bugs in the application, so it ensures error free software to some extent.*

Disadvantages of Dynamic Testing

1. ***Time Consuming****: It is a time consuming task because its aim is to execute the application or software and as a result more test cases are needed to execute.*
2. ***Increases cost of Product****: It is not done early in the software life cycle and hence bugs fixed in later stages result in more cost.*
3. ***Execution Costly****: It requires more man power to complete the task*

Difference between Verification and Validation

The distinction between the two terms is largely to do with the role of specifications.

Validation *is the process of checking whether the specification captures the customer's needs. "Did I build what I said I would?"*

Verification *is the process of checking that the software meets the specification. "Did I build what I need?"*

Verification	***Validation***
*1. **Verification** is a static practice of verifying documents, design, code and program.*	*1. **Validation** is a dynamic mechanism of validating and testing the actual product.*
2. It does not involve executing the code.	*2. It always involves executing the code.*
3. It is human based checking of documents and files.	*3. It is computer based execution of program.*
*4. **Verification** uses methods like inspections, reviews, walkthroughs, etc.*	*4. **Validation** uses methods like black box (functional) testing, gray box testing, and white box (structural) testing etc.*
*5. **Verification** is to check whether the software conforms to specifications.*	*5. **Validation** is to check whether software meets the customer expectations and requirements.*
6. It can catch errors that validation cannot catch. It is low level exercise.	*6. It can catch errors that verification cannot catch. It is High Level Exercise.*
7. Target is requirements specification, application and software architecture, high level, complete design, and database design etc.	*7. Target is actual product-a unit, a module, a bent of integrated modules, and effective final product.*
*8. **Verification** is done by QA team to ensure that the software is as per the specifications in the SRS document.*	*8. **Validation** is done by QC team and carried out with the involvement of testing team.*
9. It generally comes first-done before validation.	*9. It generally follows after **verification**.*

*Conclusion on difference of **Verification and Validation in software testing**:*

1. *Both Verification and Validation are essential and balancing to each other.*
2. *Different error filters are provided by each of them.*
3. *Both are used to finds a defect in different way, Verification is used to identify the errors in requirement specifications & validation is used to find the defects in the implemented Software application.*

Quality Assurance /Quality control:

Definition	*QA is a set of activities for ensuring quality in the processes by which products are developed.*	*QC is a set of activities for ensuring quality in products. The activities focus on identifying defects in the actual products produced.*
Focus on	*QA aims to prevent defects with a focus on the process used to make the product. It is a proactive quality process.*	*QC aims to identify (and correct) defects in the finished product. Quality control, therefore, is a reactive process.*
Goal	*The goal of QA is to improve development and test processes so that defects do not arise when the product is being developed.*	*The goal of QC is to identify defects after a product is developed and before it's released.*
How	*Establish a good quality management system and the assessment of its adequacy. Periodic conformance audits of the operations of the system.*	*Finding & eliminating sources of quality problems through tools & equipment so that customer's requirements are continually met.*
What	*Prevention of quality problems through planned and systematic activities including documentation.*	*The activities or techniques used to achieve and maintain the product quality, process and service.*
Responsibility	*Everyone on the team involved in developing the product is responsible for quality assurance.*	*Quality control is usually the responsibility of a specific team that tests the product for defects.*
Example	*Verification is an example of QA*	*Validation/Software Testing is an example of QC*
Statistical Techniques	*Statistical Tools & Techniques can be applied in both QA & QC. When they are applied to processes (process inputs & operational parameters), they are called Statistical Process Control (SPC); & it becomes the part of QA.*	*When statistical tools & techniques are applied to finished products (process outputs), they are called as Statistical Quality Control (SQC) & comes under QC.*
As a tool	*QA is a managerial tool*	*QC is a corrective tool*
Orientation	*QA is process oriented*	*QC is product oriented*

Types of Software Testing

Typically Testing is classified into three categories.

1.Functional Testing

Functional tests are processes designed to confirm that all of the components of a piece of code or software operate correctly. Functional testing focuses on testing the interface of the application to ensure that all user requirements for a properly working application are met.

2.Non-Functional Testing or Performance Testing

Non-functional testing is a type of testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

- *An excellent example of non-functional test would be to check how many people can simultaneously login into a software.*
- *Non-functional testing is equally important as functional testing and affects client satisfaction.*

3.Maintenance (Regression and Maintenance)

Testing Category	***Types of Testing***
Functional Testing	☐ *Unit Testing* ☐ *Integration Testing* ☐ *Smoke* ☐ *UAT (User Acceptance Testing)* ☐ *Localization* ☐ *Globalization* ☐ *Interoperability* ☐ *So on*
Non-Functional Testing • *Non-functional testing should increase usability, efficiency, maintainability, and portability of the product.* • *Helps to reduce production risk and cost associated with non-functional aspects of the product.* • *Optimize the way product is installed, setup, executes, managed and monitored.* • *Collect and produce measurements, and metrics for internal research and development.* • *Improve and enhance knowledge of the product behavior and technologies in us*	☐ *Performance* ☐ *Endurance* ☐ *Load* ☐ *Volume* ☐ *Scalability* ☐ *Usability* ☐ *So on*
Maintenance	☐ *Regression* ☐ *Maintenance*

This is not the complete list as there are more than 150 types of testing types and still adding. Also, note that not all testing types are applicable to all projects but depend on nature & scope of the project.