FULL STACK

Fundamentals of Testing

Md Rashed Karim

+8801711605286

ceo@fullstackbd.com

After completion of this chapter the student will –

1. learn the basic principles related to testing, the reasons why testing is required, and what the test objectives are.
2. understand the test process, the major test activities, and testware.
3. understand the essential skills for testing.

Chapter 1: Fundamentals of software testing

Agenda

fb.me/fullstackademy | +8801711605286 | info@fullstackbd.com

What is Testing?

- FL-1.1.1 (K1) Identify typical test objectives
- FL-1.1.2 (K2) Differentiate testing from debugging

The importance of software

- Software systems are an integral part of life, from business applications (e.g., banking) to consumer products (e.g., cars).
- The functioning of machines and equipment depends largely on software
- We cannot imagine large systems in telecommunication, finance or traffic control running without software.
- Most people have had an experience with software that did not work as expected.
- Software that does not work correctly can lead to many problems, including loss of money, time, or business reputation, and even injury or death.

What is Quality?

- A product is said to be of good quality if it satisfies the customer requirements in terms of performance, grade, durability, appearance and intended use/purpose, etc.
- The totality of functionality and features of a product/service that contribute to its ability to satisfy stated or implied needs and expectations.

What is Testing?

- Software testing is a way to assess the quality of the software and to reduce the risk of software failure in operation.
- Testing and reviewing ensure the improvement of the quality of software development process itself.

What is Quality Control?

- QC is the process through which we measure the actual quality performance and compare it with the set standards.
- And if there is a deviation then we take corrective actions.
- The typically quality control program is based on the periodic inspection, followed by feedback of the results and change or adjustment whenever required.

What is Quality Assurance?

- QA contains all those planned and systematic actions necessary to supply adequate confidence that a product or service will satisfy/fulfill given requirements for Quality.
- QA includes QC, but it emphasizes quality in the design of products and processes.

Testing and Debugging | Test | Debugging | Retest

- Testing and debugging are separate activities.
- Testing can trigger failures that are caused by defects in the software (dynamic testing) or can directly find defects in the test object (static testing)
- When dynamic testing triggers a failure, debugging is concerned with finding causes of this failure (defects), analyzing these causes, and eliminating them. The typical debugging process in this case involves:
    - Reproduction of a failure
    - Diagnosis (finding the root cause)
    - Fixing the cause
- Subsequent confirmation testing checks whether the fixes resolved the problem. Preferably, confirmation testing is done by the same person who performed the initial test.
- Subsequent regression testing can also be performed, to check whether the fixes are causing failures in other parts of the test object

Objectives of Testing /1

1. To evaluate work products such as requirements, user stories, design, and code
2. To verify whether all specified requirements have been fulfilled
3. To validate whether the test object is complete and works as the users and other stakeholders expect
4. To build confidence in the level of quality of the test object
5. To prevent defects
6. To find failures and defects
7. To provide sufficient information to stakeholders to allow them to make informed decisions, especially regarding the level of quality of the test object
8. To reduce the level of risk of inadequate software quality (e.g., previously undetected failures occurring in operation)
9. To comply with contractual, legal, or regulatory requirements or standards, and/or to verify the test object's compliance with such requirements or standards

Objectives of Testing /2

The objectives of testing can vary, depending upon -

- the context of the component or system, the test level, and the SDLC model.

These differences may include, for example:

- During component testing, one objective may be to find as many failures as possible so that the underlying defects are identified and fixed early. Another objective may be to increase code coverage of the component tests.
- During acceptance testing, one objective may be to confirm that the system works as expected. Another objective of this testing may be to give information to stakeholders about the risk of releasing the system at a given time.

FULL STACK

1.2 Why is Testing Necessary?

Software Failure History

https://youtu.be/qWYUn-TlOoY

fb.me/fullstackademy | +8801711605286 | info@fullstackbd.com

Causes of software failures \1

- A person can make an error (mistake), which can lead to the introduction of a defect (fault or bug) in the software code or in some other related work product.
- For example, a requirements elicitation error can lead to a requirements defect, which then results in a programming error that leads to a defect in the code.
- If a defect in the code is executed, this may cause a failure, but not necessarily in all circumstances.
- For example, some defects require very specific inputs or preconditions to trigger a failure, which may occur rarely or never.

Causes of software failures \2

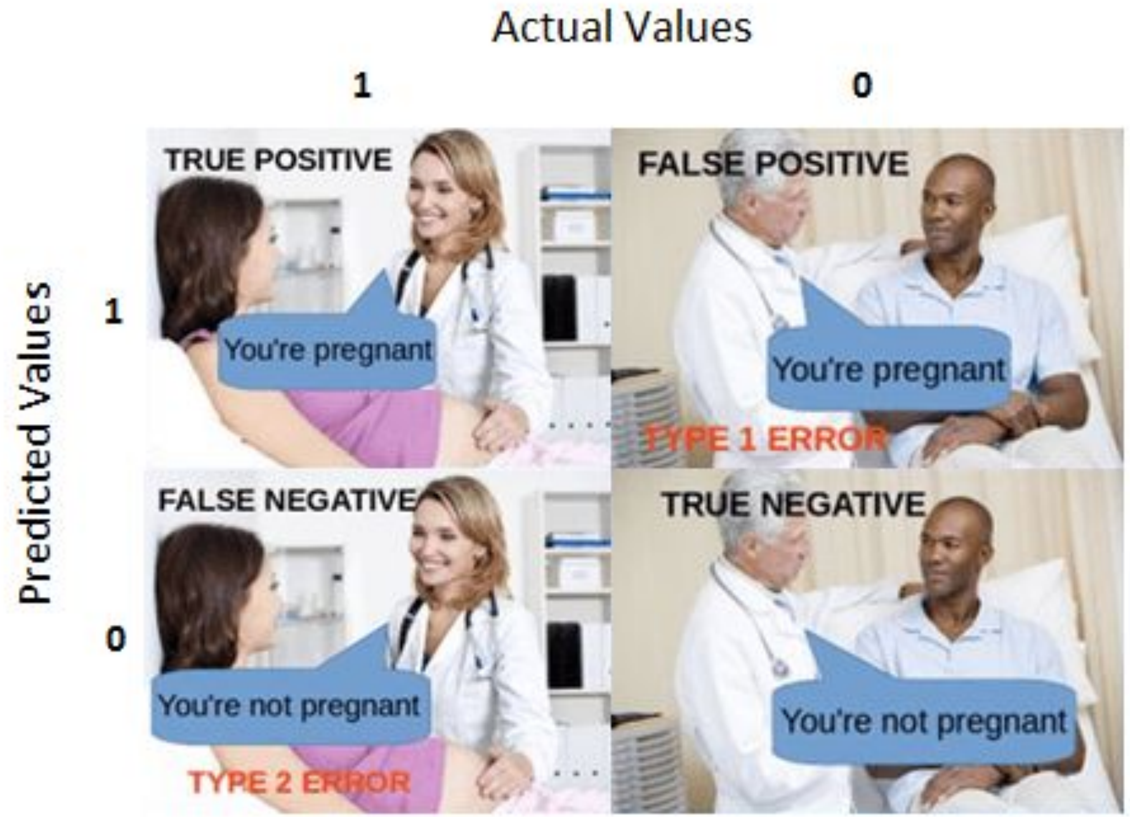
Errors may occur for many reasons, such as:

- Time pressure
- Human fallibility
- Inexperienced or insufficiently skilled project participants
- Miscommunication between project participants, including miscommunication about requirements and design
- Complexity of the code, design, architecture, the underlying problem to be solved, and/or the technologies used
- Misunderstandings about intra-system and inter-system interfaces, especially when such intra-system and inter-system interactions are large in number
- New, unfamiliar technologies

Causes of software failures \3

- In addition to failures caused due to defects in the code, failures can also be caused by environmental conditions.
  - For example, radiation, electromagnetic fields, and pollution can cause defects in firmware or influence the execution of software by changing hardware conditions.
- Not all unexpected test results are failures. False positives may occur due to errors in the way tests were executed, or due to defects in the test data, the test environment, or other test-ware, or for other reasons.
- The inverse situation can also occur, where similar errors or defects lead to false negatives. False negatives are tests that do not detect defects that they should have detected; false positives are reported as defects, but aren’t actually defects.

Causes of software failures \4



Root Causes, Defects, Failures

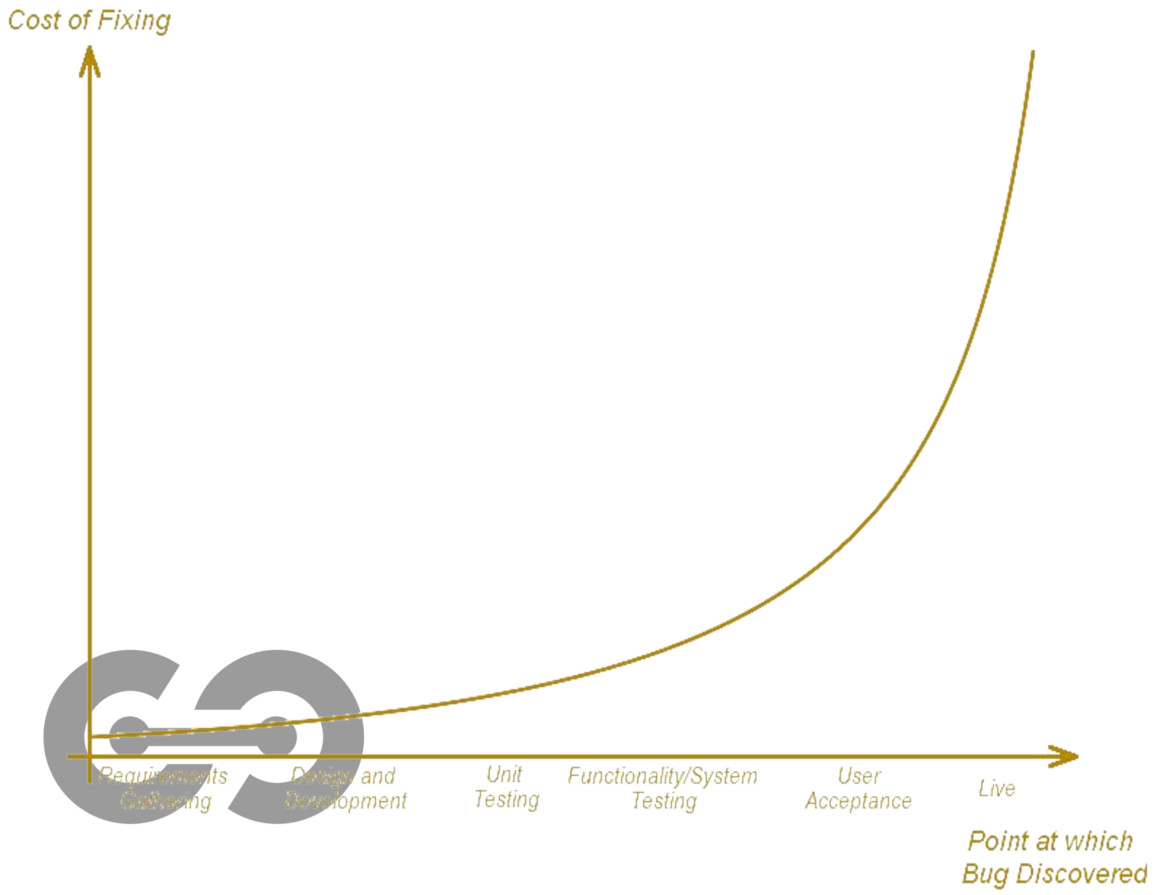
Root Causes: A human action that produces an incorrect result, e.g. a coding/design error

Defects: A flaw in a component or system to fail to perform its required function, e.g. an incorrect statement of data definition.

Failures: The physical or functional manifestation of a defect. A defect, if encountered during execution, may cause a failure. Deviation of the component or system from its expected delivery, service or result.

Costs of defects

- The costs of fixing defect is increase with the time they remain in the system.
- Detecting errors at an early stage allows for error correction at reduced cost.



Software Quality Attributes

- **Functional Q-attributes**
  - Accuracy
  - Compliance
  - Stability
  - Security
  - Interoperability

Software Quality Attributes

Non-Functional Q-attributes

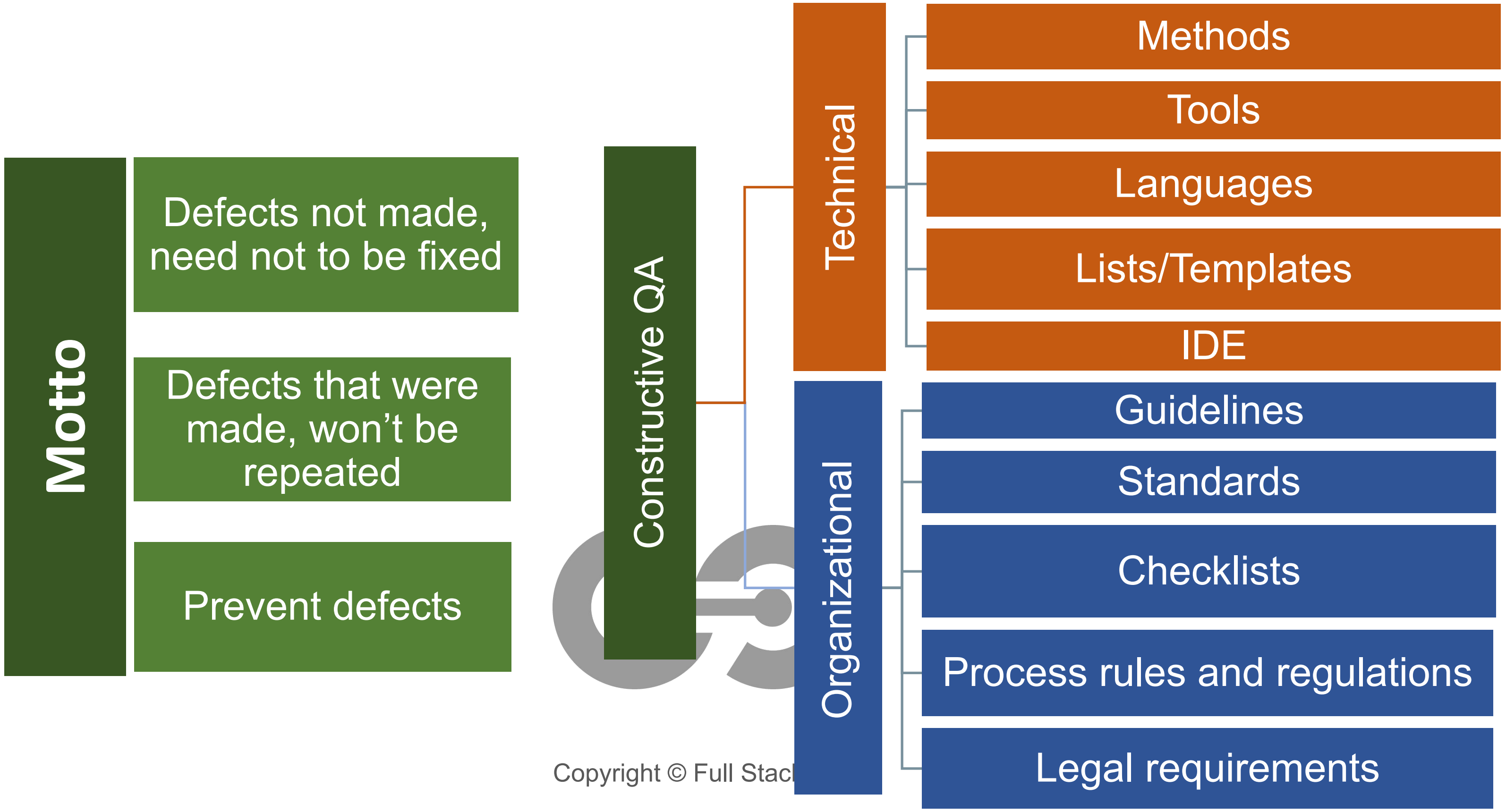
- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

Types of Quality Assurance (QA):

1. **Constructive Activities** to prevent defects, e.g. through appropriate methods of software engineering
2. **Analytical Activities** for finding defects, e.g. through testing leading to correcting defects and preventing failures, hence increasing the software quality.

Constructive quality assurance

Quality of process- Quality management



Analytical Quality Assurance

Quality of product- Verification and test procedure

Motto

Defects should be detected

As early as possible in the process

- Analytical QA
  - Static
    - Review/walkthroughs
    - Control flow analysis
    - Dataflow analysis
    - Computer metrics/analyzer
  - dynamic
    - Black box
      - Equivalence partitioning
      - Boundary value analysis
      - State transition testing
      - Decision Tables
      - Use case based testing
    - Experience-based techniques
    - White box
      - Statement coverage
      - Branch coverage
      - Condition coverage
      - Path coverage

Software quality- functional Q-attributes

- ***Functionality means:***

  Correctness: the functionality meets the required attributes / capabilities

  Completeness: the functionality meets all(functional) requirements.

- **Functionality includes (as per ISO/IEC 9126):**

* Stability

* Accuracy

* Interoperability

* Security

Non-Functional Quality Attributes/1

Reliability

* Maturity, fault tolerance, recovery after failure.
* Characteristic: under given conditions, a software/ a system will keep its capabilities/ functionality over a period of time.
* Reliability = quality/time

Usability

* Learn ability ,understanding ability, attractiveness.
* Characteristics: easy to learn, compliance with guidelines, intuitive handling.

Non-Functional Quality Attributes/2

Efficiency

- System behavior: functionality and time behavior.
- Characteristics: the system requires a minimal use of resources (e.g. CPU-time) for executing the given task.

Maintainability

- Verifiability, stability, analyzability, changeability, adaptability.
- Characteristics: amount of effort needed to introduce changes in system components.

Portability

- Reparability, compliance, install ability.
- Ability to transfer the software to a new environment (software, hardware, organization)
- Characteristics: easy to install and uninstall.

Quality Attributes

- Some quality attributes are influenced reciprocally, depending on the test object, attributes must be prioritized, e.g. efficiency vs. portability.
- Different kinds of tests will be performed in order to measure the different kinds of attributes .

How much testing is enough?

Exit criteria

Not finding (any more) defects is not an appropriate criterion to stop testing activities. Other metrics are needed to adequately reflect the quality level reached.

Risk based testing

Levels of risk determine the extent of testing carried out, i.e. liability for damages in case of failures occurring, economic and project related aspects.

Time based testing

The amount of time available might determine the extent of testing efforts.

Budget based testing

The amount of resources available (personal and budget) might determine the extent of testing efforts.

Test Basis / Test Base

- The source of information or the set documents of a component or system that is needed for test analysis on which the test cases are based.
- Test basis should be well defined and adequately structured so that one can easily identify test conditions from which test cases can be derived.
- If a document can be amended only by way of formal amendment procedure, then the test basis is called a frozen test basis.

Test Case

A set of input values, execution preconditions, expected results and execution postconditions, developed for a particular objective or test condition, such as to exercise a particular program path or to verify compliance with a specific requirement.

Test Case Template

Identity	
Conditions	
Values	
Relations	
Optionals	

Test Case Template:

1. Unique Identifier
2. Pre-Conditions
3. Expected Post-Conditions
4. Set of input values
5. Set of expected results
6. Dependence on other test cases
7. Reference to the requirement that will be tested
8. How to execute the test and check results (optional)
9. Priority (optional)

Summary:

1. **Software failures** may cause enormous damage
2. **Software quality** is the sum of attributes that refer to the capability of the software to meet given requirements
3. **Constructive** software quality, assurance deals with preventing defects
4. **Analytical** software quality, assurance deals with finding defects and correcting them
5. Functional and non-functional **quality attributes** define the total quality of the system
6. Each test has to have predefined exit criteria reaching the **exit criteria** will conclude testing activities.
7. Testers look for defects in the system and report them **(testing)** .
8. Developers look for errors and correct them (**debugging**)

FULL STACK

fb.me/fullstackademy | +8801711605286 | info@fullstackbd.com

1.3 Testing Principles

Testing Principles

1: Testing shows the presence, not the absence of defects

2: Exhaustive testing is impossible

3: Early testing saves time and money

4: Defects Cluster together

5: Tests wear out

6: Testing is context dependent

7: Absence-of-defects fallacy (misconception)

Principle 1: Testing shows the presence, not the absence of defects

- Testing can prove the presence of defects.
- Deviations discovered while testing show a failure.
- The cause of the failure might not be obvious.
- Testing reduces the probability of defects remaining undiscovered.
- The absence of failure does not prove the correctness of the software.
- The test procedure itself might contain error
- The test conditions might be unsuitable for finding errors.

FULL STACK

Principle 2: Exhaustive testing is impossible

Exhaustive testing

A test approach in which the test suite comprises all combinations of input values and preconditions

Test case explosion

Defines the exponential increase of efforts and costs when testing exhaustively

Sample test

The test includes only a (systematically or randomly derived) subset of all possible input values under real life conditions, sample tests are generally used.

Testing all combination of inputs and preconditions is only economically feasible in trivial cases.

FULL STACK

Principle 3: Early testing saves time and money

- The earlier a defect is discovered, the less costly is its correction.
- Highest cost effectiveness when errors are corrected before implementation
- Concepts and specifications may already be tested.
- Defects discovered at the conception phase are corrected with the least effort and costs.
- Preparing a test is time consuming as well.
- **Testing involves more than just test execution.**
- Test activities can be prepared before software development is completed.
- Testing activities (including reviews) should run in parallel to software specification and design.

FULL STACK

Principle 4: Defects Cluster together

- Find a defect and you will find more defects nearby!
- Defects often appear clustered like mushrooms or cockroaches
- It is worth screening the same module where one defect was found
- Testers must be flexible to revise the test cases
- Once a defect is found it is a good idea to reconsider the direction of further testing.
- The location of a defect might be screened at higher detail level, e.g. starting additional tests or modifying existing tests.

Principle 5: Tests wear out

- **Repeating tests under the same conditions is ineffective.** If the same tests are repeated over and over again, no new bugs can be found.
- To detect new defects, existing tests are need to be redesigned, and even new tests may need to be designed.
- **Tests must be revised regularly for different code modules.**
- It is necessary to repeat a test after changes have been made in the code (bug fixing, new functionality).
- In some cases, such as automated regression testing, the pesticide paradox has a beneficial outcome, which is the relatively low number of regression defects.

Principle 6: Testing is context dependent

- **Testing is done differently in different contexts**
- **Different test objects are tested differently.**
- The engine controller of a car requires tests **different** than those of an eCommerce application
- **Production environment is different than Test environment (Test Bed).** The test environment should be **very similar** to the production environment.
- There will always be **deviations** between test environment and the production environment.

Principle 7: Absence-of-defects fallacy (misconception)

- Some organizations expect that testers can run all possible tests and find all possible defects (but principles 2 and 1). Further, it is a fallacy to expect that just finding and fixing a large number of defects will ensure the success of a system.
- Successful testing find the most serious defects
- Not finding any more defect does not prove the quality of the software.
- The functionality of the software may not meet the expectations of the users.
- You can not test quality into the product, it must be built in from the very beginning!

Summary:

- ✔ **Tests** can help finding defects in the software, however; they can not give proof of the **absence of defects**
- ✔ For non-trivial system, **exhaustive testing** is impossible, sample testing necessary.
- ✔ **Early testing** helps reduce costs because defects discovered early on are fixed with less effort
- ✔ Defects show up **clustered**. Finding a defect at one place will mean you probably find another defect nearby.
- ✔ **Repeating** identical tests produces no new information
- ✔ Each **particular environmental** determines the way in which tests will run
- ✔ Error-free software does not imply **suitability for use.**

1.4 Test Process

Test Process

There is no one universal software test process, but there are common sets of test activities without which testing will be less likely to achieve its established objectives. These sets of test activities are called a test process.

Test Activities and Tasks

A test process consists of the following main groups of activities:

1. Test planning
2. Test monitoring and control
3. Test analysis
4. Test design
5. Test implementation
6. Test execution
7. Test completion

Test planning

- Test planning involves activities -
  - ✔ define the objectives of testing and exit criteria
  - ✔ Determining the scope and risk
  - ✔ the approach for meeting test objectives within constraints imposed by the context
  - ✔ specifying suitable test techniques and tasks
  - ✔ formulating a test schedule for meeting a deadline with the resources and budget
- Test plans may be revisited based on feedback from monitoring and control activities.

Test plan:

A document describing the scope, approach, resources and schedule of intended test activities. It includes, but is not limited to, the test items, the features to be tested, resources and contingency planning.

Test Strategy:

1. A high level description of the test levels to be performed and the testing within those levels for an organization or program (one or more projects)
2. According to the overall approach, the test efforts are divided among the test objects and the different test objectives: the choice of test methods, how and when the test activities should be done and when to stop testing (exit criteria)

Exit criteria:

The set of generic specific conditions, agreed upon with the stakeholders, for permitting a process to be officially completed. The purpose of exit criteria is to prevent a task from being considered completed when there are still parts of the task outstanding which have not been finished. **Exit criteria** are used to report against and to plan **when to stop testing**. This should **be done for each test level.**

Test monitoring and control

- Comparison of actual progress against the test plan using any test monitoring metrics defined in the test plan.
- It involves taking actions necessary to meet the objectives of the test plan.
- Evaluation of exit criteria, which are referred to as the definition of done.
- The evaluation of exit criteria for test execution as part of a given test level may include:
  - Checking test results and logs against specified coverage criteria
  - Assessing the level of component or system quality based on test results and logs
  - Determining if more tests are needed
- Test progress against the plan is communicated to stakeholders in test progress reports, including deviations from the plan and information to support any decision to stop testing.

Test analysis /1

During test analysis, the test basis is analyzed to identify testable features and define associated test conditions. In other words, test analysis determines "what to test" in terms of measurable coverage criteria.

Test analysis /2

Test analysis includes the following major activities:

- Analyzing the test basis appropriate to the test level being considered, for example:
  - ✔ Requirement specifications, user stories, or similar work products that specify desired functional and non-functional component or system behavior
  - ✔ Design such as system or software architecture, design specifications, modelling, interface specifications
  - ✔ The component or system itself, including code, database metadata and queries, and interfaces
  - ✔ Risk analysis reports, which may consider functional, non-functional, and structural aspects of the component or system

Test analysis /3

- Evaluating the test basis and test items to identify defects of various types, such as:
  - ✔ Ambiguities
  - ✔ Omissions
  - ✔ Inconsistencies
  - ✔ Inaccuracies
  - ✔ Contradictions
  - ✔ Superfluous statements
- Identifying features to be tested
- Defining and prioritizing test conditions for each feature based on analysis of the test basis, and considering functional, non-functional, and structural characteristics, other business and technical factors, and levels of risks
- Capturing bi-directional traceability between each element of the test basis and the associated test conditions

Test analysis /4

- Test analysis to reduce the likelihood of omitting important test conditions and to define more precise and accurate test conditions which are to be used as test objectives in test charters.
- Test charters are typical work products in some types of experience-based testing. When these test objectives are traceable to the test basis, coverage achieved during such experience-based testing can be measured.
- The identification of defects during test analysis is an important potential benefit, especially where no other review process is being used and/or the test process is closely connected with the review process.
- Test analysis activities not only verify whether the requirements are consistent, properly expressed, and complete, but also validate whether the requirements are properly captured.
- For example, techniques such as behavior driven development (BDD) and acceptance test driven development (ATDD), which involve generating test conditions and test cases from user stories and acceptance criteria prior to coding, also verify, validate, and detect defects in the user stories and acceptance criteria.

Test design

Test analysis answers the question "what to test?" while test design answers the question "how to test?"

Test design includes the following major activities:

- Designing and prioritizing test cases and sets of test cases
- Identify specific test conditions and required test data evaluate the availability of test data and/or the feasibility of generating test data
- Designing the test environment and identifying any required infrastructure and tools
- Connecting the test environment to adjacent systems
- Capturing bi-directional traceability between the test basis, test conditions, test cases, and test procedures

Positive tests give proof of the functionality, negative tests check the handling of error situations

Test design

Test data:

Data that exists in the system before a test is executed and affects or is affected by the component or system under test.

Input data:

A variable that is read by a component (whether stored within the system or outside)

Test coverage:

The degree of which a specified item has been exercised by a test suite (expressed as a percentage). Used mostly on white box tests to determine code coverage.

Test oracle:

A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), other software, a user manual, or an individual's specialized knowledge, but should not be the code.

Test Implementation /1

Necessary testware for test execution is created, including sequencing the test cases into test procedures. So, test implementation ensures everything in place to run the tests.

Test implementation includes the following major activities:

- Developing and prioritizing test procedures, creating automated test scripts
- Creating test suites from the test procedures
- Arranging the test suites within a test execution schedule for efficient test execution
- Building the test environment (test harnesses, service virtualization, simulators, and other infrastructure items) and verifying that everything has been set up correctly
- Preparing test data and ensuring it is properly loaded
- Verifying and updating bi-directional traceability between the test basis, test conditions, test cases, test procedures, and test suites

Test Implementation /2

- Test design and test implementation tasks are often combined.
- In exploratory testing and other types of experience-based testing, test design and implementation may occur, and may be documented, as part of test execution.
- Exploratory testing may be based on test charters (produced as part of test analysis), and exploratory tests are executed immediately as they are designed and implemented

Test Execution

During test execution, test suites are run in accordance with the test execution schedule.

Test execution includes the following major activities:

- **Recording** the IDs and versions of the test item(s) or test object, test tools, and testware
- **Executing tests** either manually or by using test execution tools
- **Comparing** actual results with expected results
- **Analyzing anomalies** to establish their likely causes (e.g., failures may occur due to defects in the code, but false positives also may occur)
- **Reporting defects** based on the failures observed
- **Logging** the outcome of test execution (e.g., pass, fail, blocked)
- **Repeating test activities** either as a result of action taken for an anomaly, or as part of the planned testing (e.g., execution of a corrected test, confirmation testing, and/or regression testing)
- **Verifying and updating bi-directional traceability** between the test basis, test conditions, test cases, test procedures, and test results.

Test suite/test sequence: A set of several test cases for a component or system, where post condition of one test is used as the precondition foe the next one

Test procedure (test scenario): A document specifying a sequence of action for the execution of a test. Also known as test script or manual test script.

Test execution: The process of running a test, producing actual results.

Test log (test protocol, test report): A chronological record of relevant details about the execution of tests: when the test was done, what result was produced.

Regression tests: Testing of a previously tasted program following modification of ensure that defected have not been introduced or uncovered in unchanged areas of the software , as a result of the changes made. It is performed when the software or its environment is changed.

Confirmation testing retest: Repeating a test after a defect has been fixed in order to confirm that the original defect has been successfully removed

Test completion /1

- Collect data from completed test activities to consolidate experience, testware, and any other relevant information.
- Test completion activities occur at project milestones such as when a software system is released, a test project is completed (or cancelled), an Agile project iteration is finished (e.g., as part of a retrospective meeting), a test level is completed, or a maintenance release has been completed.

Test completion /2

Test completion includes the following major activities:

- Checking whether all defect reports are closed, entering change requests or product backlog items for any defects that remain unresolved at the end of test execution
- Creating a test summary report to be communicated to stakeholders
- Finalizing and archiving the test environment, the test data, the test infrastructure, and other testware for later reuse
- Handing over the testware to the maintenance teams, other project teams, and/or other stakeholders who could benefit from its use
- Analyzing lessons learned from the completed test activities to determine changes needed for future iterations, releases, and projects
- Using the information gathered to improve test process maturity

FULL STACK

Test Process in Context /1

Contextual factors that influence the test process for an organization, include, but not limited to:

- SDLC model and project methodologies being used
- Test levels and test types being considered
- Product and project risks
- Business domain
- Operational constraints, including but not limited to:
  - Budgets and resources
  - Timescales
  - Complexity
  - Contractual and regulatory requirements
- Organizational policies and practices
- Required internal and external standards

Test Process in Context /2

The following sections describe general aspects of organizational test processes in terms of the following:

- Test activities and tasks
- Test work products
- Traceability between the test basis and test work products

Testware /1

- Testware is created as output work products from the test activities.
- There is a significant variation in how different organizations produce, shape, name, organize and manage their work products.
- Proper configuration management ensures consistency and integrity of work products.

Testware /2

The following list of work products is not exhaustive:

- Test planning work products
  - one or more test plans
- Test monitoring and control work products
  - test progress reports (produced on an ongoing and/or a regular basis)
  - test summary reports (produced at various completion milestones)
  - project management concerns, such as task completion, resource allocation and usage, and effort.
- Test analysis work products
  - defined and prioritized test conditions
  - creation of test charters
- Test design work products
  - test cases and sets of test cases
  - design and/or identification of the necessary test data
  - design of the test environment, and the identification of infrastructure and tools

Testware /2

- Test implementation work products
  - Test procedures and the sequencing of those test procedures
  - Test suites
  - A test execution schedule
- Test execution work products
  - Documentation of the status of individual test cases or test procedures (e.g., ready to run, pass, fail, blocked, deliberately skipped, etc.)
  - Defect reports
  - Documentation about which test item(s), test object(s), test tools, and testware were involved in the testing
- Test completion work products
  - test summary reports
  - action items for improvement of subsequent projects or iterations
  - change requests or product backlog items, and finalized testware.

Traceability between the Test Basis and Test Work Products

For effective test monitoring and control, it is required to maintain traceability between each element of the test basis and the various test work products associated with that element. In addition to the evaluation of test coverage, good traceability supports:

- Analyzing the impact of changes
- Making testing auditable
- Meeting IT governance criteria
- Improving the understandability of test progress reports and test summary reports to include the status of elements of the test basis (e.g., requirements that passed their tests, requirements that failed their tests, and requirements that have pending tests)
- Relating the technical aspects of testing to stakeholders in terms that they can understand
- Providing information to assess product quality, process capability, and project progress against business goals

Roles in Testing

Two principal roles in testing are covered:

- Test management role
  - overall responsibility for the test process, test team and leadership of the test activities.
  - is mainly focused on the activities of test planning, test monitoring & control and test completion.
- Testing role.
  - overall responsibility for the engineering (technical) aspect of testing.
  - is mainly focused on the activities of test analysis, test design, test implementation and test execution.

1.5 Essential Skills and Good Practices in Testing

1.5 Essential Skills and Good Practices in Testing

- Skill is the ability to do something well that comes from one's knowledge, practice and aptitude.
- Good testers should possess some essential skills to do their job well.
- Good testers should be effective team players and should be able to perform testing on different levels of test independence.

1.5 Essential Skills and Good Practices in Testing

Roles and Responsibilities

Developer role	Tester role
Implements requirements	Plans testing activities
Developers structures	Design test case
Design and programs the software	Is concerned only with finding defects
Creating a product is his success	Finding an error made by a developer is his success

Perception: **Wrong!**

Testing is a constructive activity as well,

It aims eliminating defects from a product!

Personal attributes of a good tester /1

- Thoroughness, carefulness, curiosity, attention to details, being methodical
- To comprehend the practical scenarios of the customer
- To be able to analysis the structure of the test (Domain and Technical knowledge)
- To discover details, where failure might show
- Skepticism and has a critical eye
- Test object contain defects- you just have to find them
- Do not believe everything you are told by the developers
- One must not get frightened by the fact that serious defects may often be found which will have impact on the course of the project.
- Analytical thinking, critical thinking, creativity

FULL STACK

Personal attributes of a good tester /2

- Good communication skills
- Active listening,
- being a team player (to interact effectively with all stakeholders, to convey information to others, to be understood, and to report and discuss defects)

Personal attributes of a good tester /3

Good communication skills

- ✔ To bring **bad news** to the developers
- ✔ To overbear frustration state of minds
- ✔ Both technical as well as issue of the practical use of the system must be understood and communicated
- ✔ Positive communication can help to avoid or to ease difficult situations.
- ✔ To quickly establish a working relationship with the developers
- ✔ **Personal factors** influencing error occurrence
- ✔ Experience helps identifying where **errors** might **accumulate**

Personal attributes of a good tester /4

- Testers are often the bearers of bad news. It is a common human trait to blame the bearer of bad news.
- This makes communication skills crucial for testers.
- Communicating test results may be perceived as criticism of the product and of its author.
- Confirmation bias can make it difficult to accept information that disagrees with currently held beliefs.
- Some people may perceive testing as a destructive activity, even though it contributes greatly to project success and product quality.
- To try to improve this view, information about defects and failures should be communicated in a constructive way.

1.5.2. Whole Team Approach

- the ability to work effectively in a team context and to contribute positively to the team goals.
- any team member can perform any task, and everyone is responsible for quality.
- The team members share the same workspace
- improves team dynamics, enhances communication and collaboration within the team, and creates synergy by allowing the various skill sets within the team to be leveraged for the benefit of the project.
- Testers work closely with other team members to ensure that the desired quality levels are achieved.
- This includes collaborating with business representatives to help them create suitable acceptance tests and working with developers to agree on the test strategy and decide on test automation approaches.
- Testers can thus transfer testing knowledge to other team members and influence the development of the product.
- Depending on the context, the whole team approach may not always be appropriate. For instance, in some situations, such as safety-critical, a high level of test independence may be needed.

1.5.3. Independence of Testing

Differences: to design - to develop - to test

- ❑ Testing requires a different mindset from designing developing new computer systems
  - **Common goal**: to provide good software
  - **Design mission**: help the customer to supply the right requirements
  - **Developer's mission**: convert the **requirements** into functions
  - **Tester's mission**: examine the **correct** implementation of the customer's requirements
- ❑ In principle, one person can be given all three roles to work at.
  - Differences in goal and role models must be taken into account
  - This is difficult but possible
  - Other solution (independent testers) are often easier and produce better results

Independent testing /1

Developer Test

- No Extra costs for the orientation of other person on the test object
- The developer will never examine his creation (biased: **emotional attachment**)
- Human being tend to **overlook their own faults.**
- Error made because of **misinterpretation** of the requirements will remain undetected.

Solution:

Setting up test teams where developers test each other's products helps to avoid or at least lessen this shortcoming.

Independent testing /2

Teams of developers

- Developers speak the same language
- Costs for orientation in the test object are kept moderate especially when the teams exchange test objects.
- Danger of generation of conflicts among developing teams
  - One developer who looks for and finds a defect will not be the other developer’s best friend

- Mingling development and test activities
  - Frequent switching of ways of thinking
  - Makes difficult to control project budget

FULL STACK

Independent testing /3

Test teams

- Creating test teams converting different project areas enhances the quality of testing.
- It is important that test teams of different areas in the project work independently

Independent testing /4

Outsourcing tests

- The separation of testing activities and development activities offers best independence between test object and tester.
- Outsourced test activities are performed by persons having relatively little knowledge about the test object and the project background
  - **Learning curve** bring high costs, therefore unbiased party experts should be involved at the early stages of the project
- External expert have a high level of **testing know how**:
  - An appropriate test design is ensured
  - Methods and tools find optimal

Designing test cases automatically

Computer aided generation of test cases, e.g. based on the formal specification documents, is also independent

Difficulties /1

- **Unable to understand each other**
  - Developers should have basic knowledge of testing
  - Tester should have basic knowledge of software development
- Especially in **stress situations**, discovering errors that someone has made often leads to conflicts.
  - The way of document defects and the way of the defects is described will decide how the situation will develop.
  - Persons should not be criticized, the defects must be stated **faulty**
  - Defect description should help the developers find the error
  - **Common objectives** must always be the main issue.

Difficulties /2

- Communication between tester and developers missing or insufficient. This can make impossible to work together.
  - Tester seen as "only messenger of **bad news**"
  - Improvement: try to see yourself in the other person's role. Did my message come through? Did the answer reach me?
- A solid test requires the appropriate **distance to the test object**
  - An independent and non-biased position is acquired through distance from the development
  - However, too large a distance between the test object and the development team will lead to more effort and time for testing.

Summary

- People make **mistakes**, every implementation has defects.
- Human nature makes it difficult to stand in front of one’s own defects(**error blindness**)
- Developer and tester means two different worlds meet each other.
  - **Developing is constructive**- something is created that was not there before
  - **Testing** seems destructive at first glance-defect will found
  - Together , **development and test** are constructive in their objective to ensure software with the least defects possible.
- **Independent testing enhances quality of testing**:

  instead of developer, use tester teams and teams with external personnel for testing.

fb.me/fullstackacademy | +8801711605286 | info@fullstackbd.com

Thank You