Manual Testing

Rashed Karim

Manual Testing
Agenda

Session 1

1.1 What is Testing?
1.2 Why is Testing Necessary?
1.3 Test Levels

What is Testing?

I-Fundamentals of Testing
01- Understanding software testing

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.

I-Fundamentals of Testing
01- Understanding software testing

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.

I-Fundamentals of Testing
01- Understanding software testing

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.

I-Fundamentals of Testing
01- Understanding software testing

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.

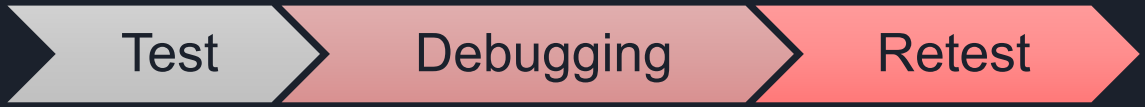
I-Fundamentals of Testing
01- Understanding software testing

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.

I-Fundamentals of Testing
01- Understanding software testing

Testing and Debugging



- Executing tests can show failures that are caused by defects in the software.
- Debugging is the development activity that finds, analyzes, and fixes such defects.
- Subsequent confirmation testing checks whether the fixes resolved the defects.
- In some cases, testers are responsible for the initial test and the final confirmation test, while developers do the debugging and associated component testing.
- However, in Agile development and in some other lifecycles, testers may be involved in debugging and component testing.

I-Fundamentals of Testing
01- Understanding software testing

Objectives of Testing /1

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

I-Fundamentals of Testing
01- Understanding software testing

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.

1.2 Why is Testing Necessary?

Software Failure History

https://youtu.be/qWYUn-TlOoY

I-Fundamentals of Testing
01- Understanding software testing

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.

I-Fundamentals of Testing
01- Understanding software testing

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

I-Fundamentals of Testing

01- Understanding software testing

Root Causes, Defects, Failures and Effects

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.

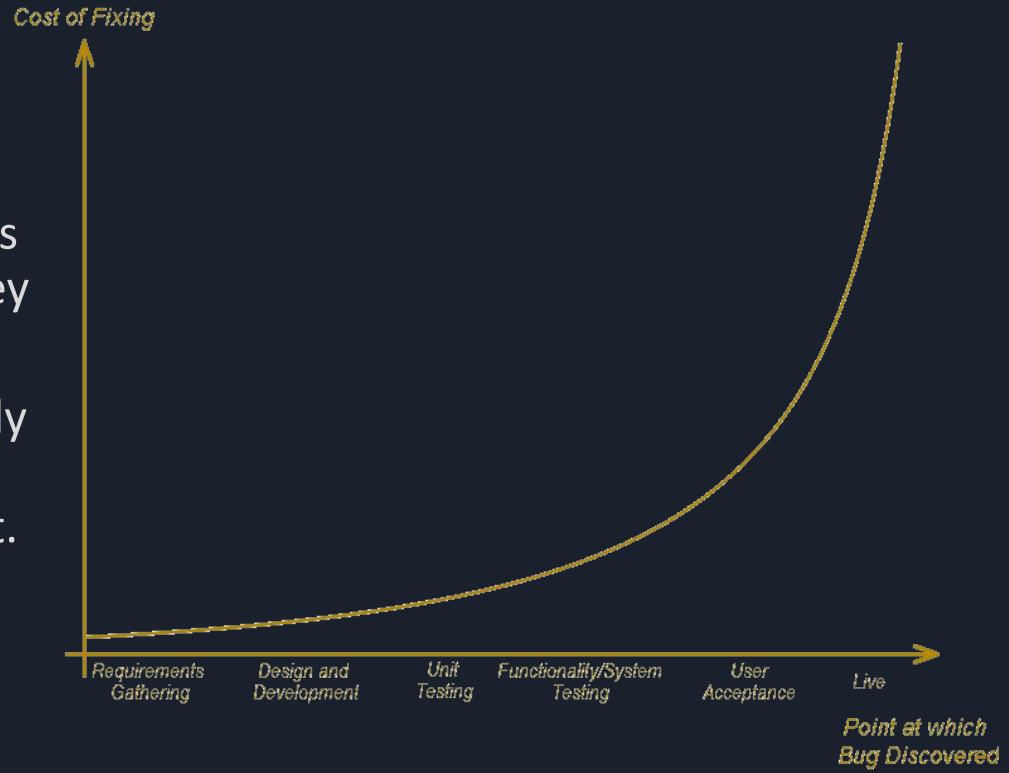
Failures: The physical or functional manifestation of a defect. A defect, if encountered during execution, may cause a failure. Deviation of the component or system from its expected delivery, service or result.

Effects: customer complaints are effects

I-Fundamentals of Testing
01- Understanding software testing

Costs of defects

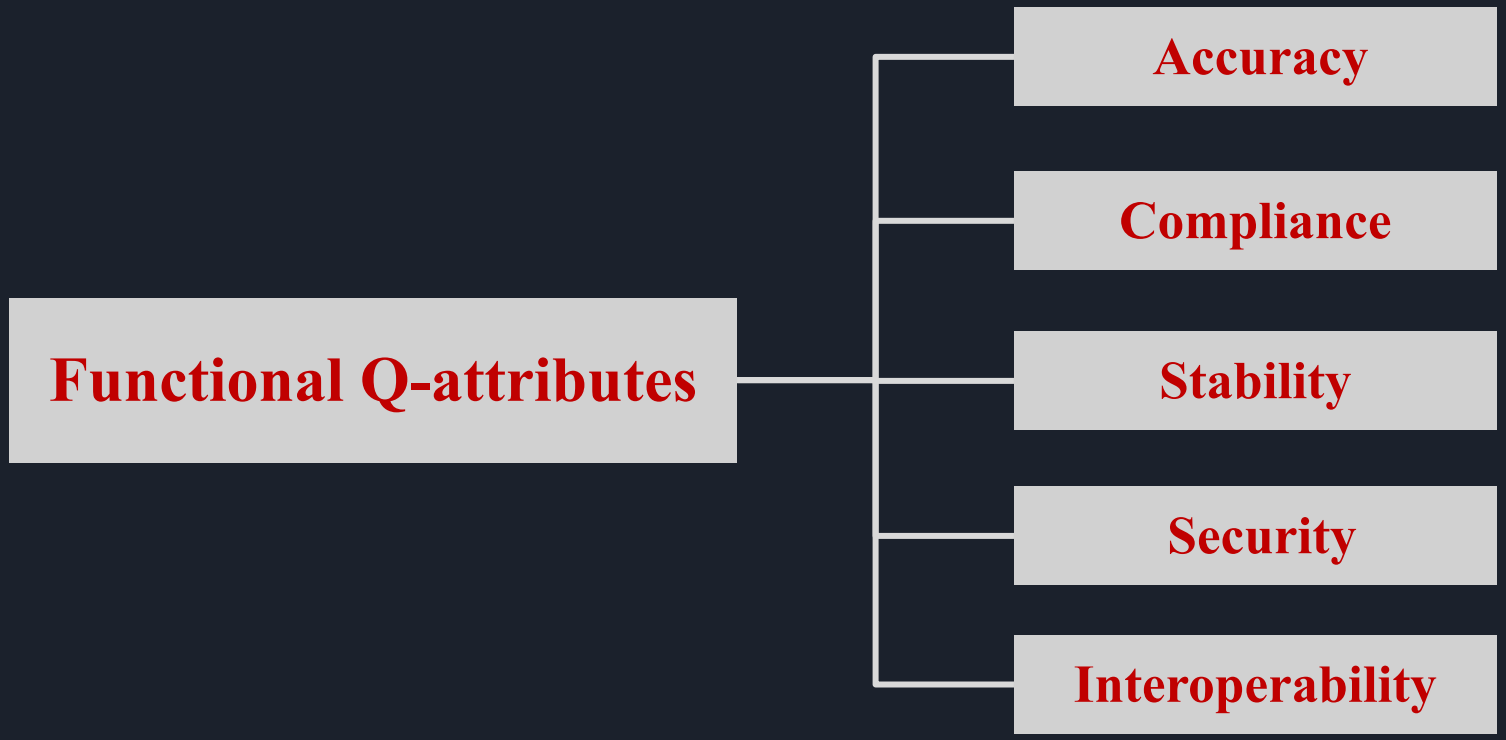
- The costs of fixing defect is increase with the time they remain in the system.
- Detecting errors at an early stage allows for error correction at reduced cost.



I-Fundamentals of Testing

01- Understanding software testing

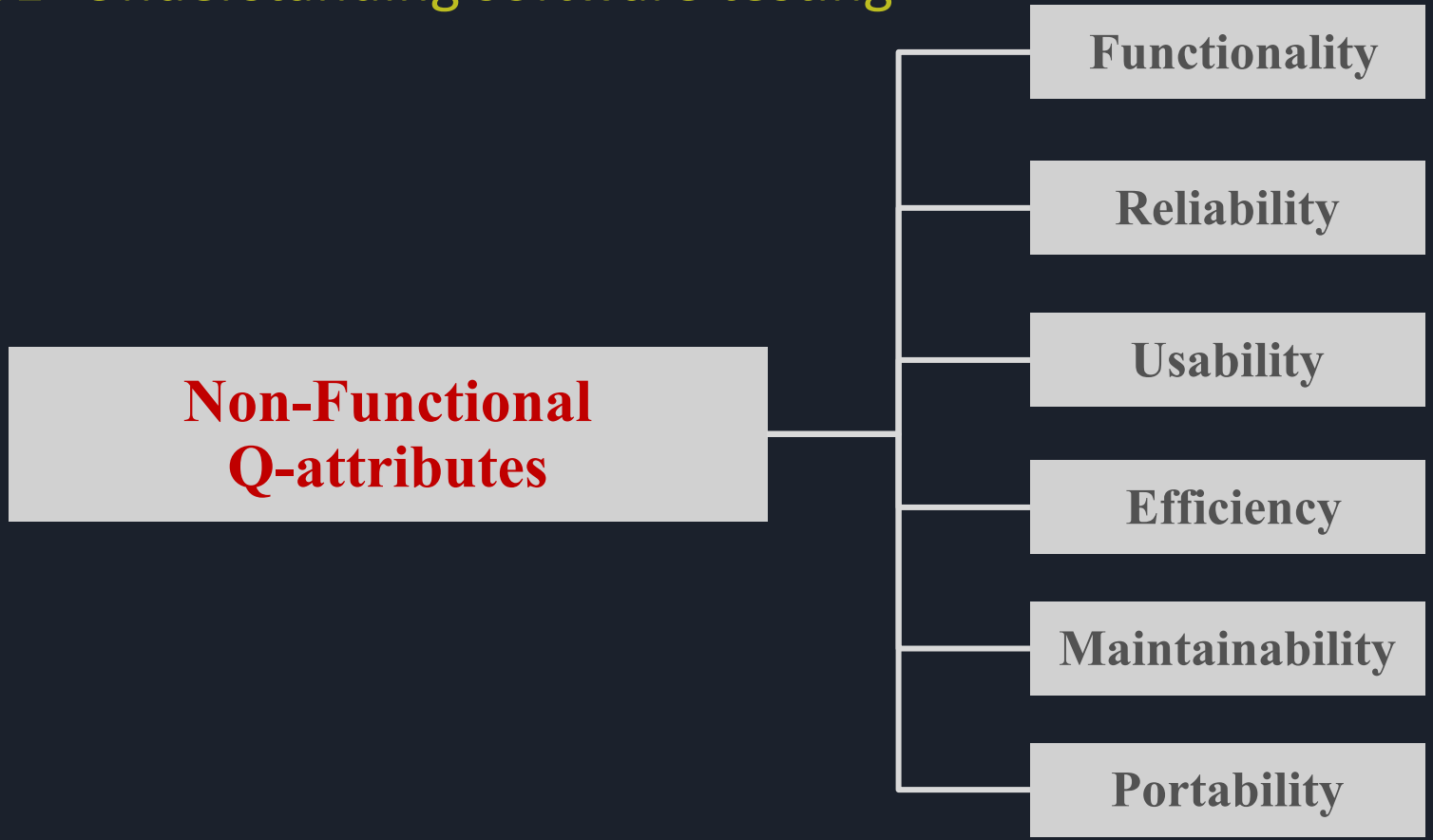
Software Quality Attributes



I-Fundamentals of Testing

01- Understanding software testing

Software Quality Attributes



I-Fundamentals of Testing

01- Understanding software testing

Types of Quality Assurance (QA):

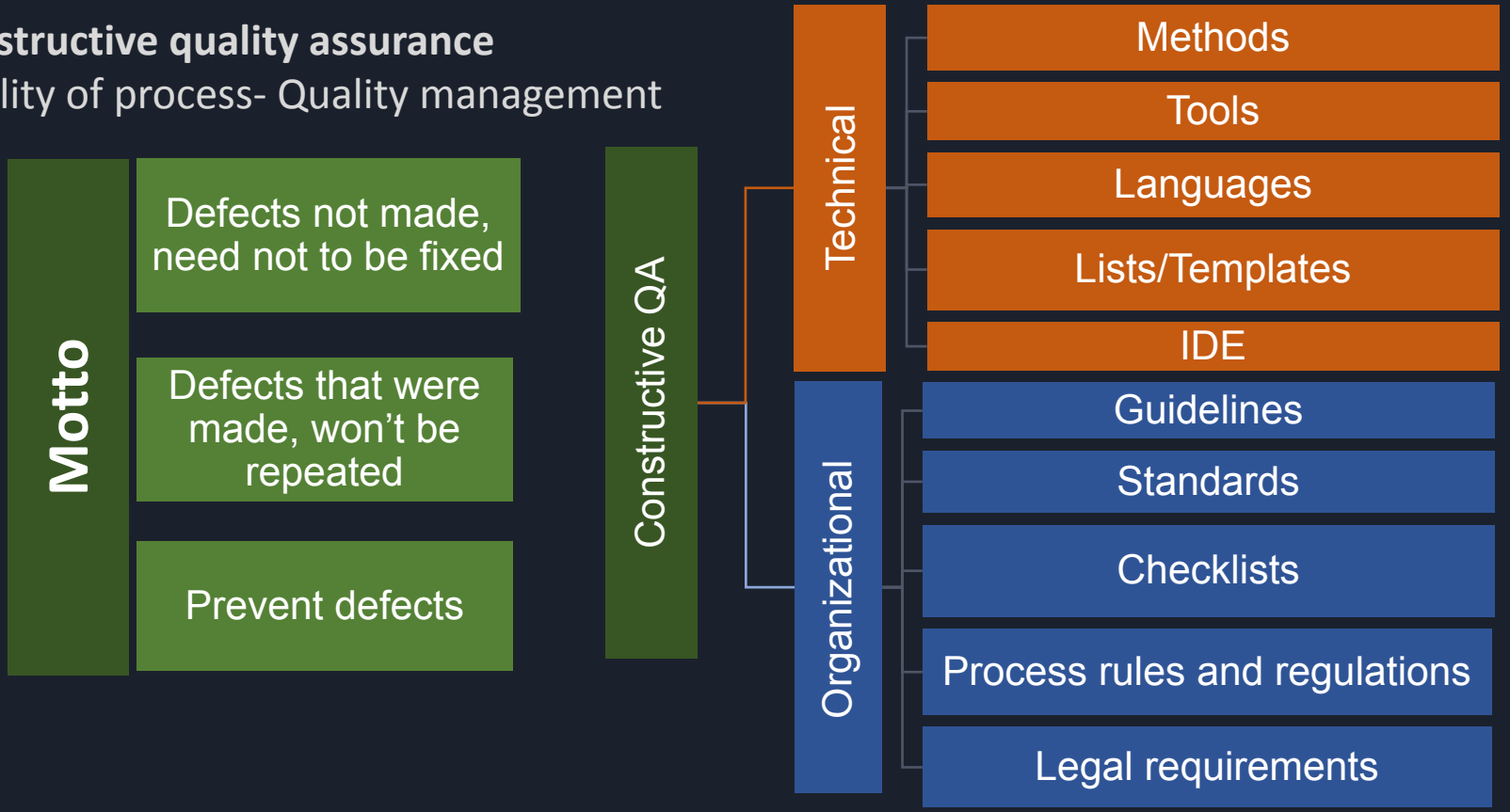
1. **Constructive Activities** to prevent defects, e.g. through appropriate methods of software engineering
2. **Analytical Activities** for finding defects, e.g. through testing leading to correcting defects and preventing failures, hence increasing the software quality.

I-Fundamentals of Testing

01- Understanding software testing

Constructive quality assurance

Quality of process- Quality management



I-Fundamentals of Testing

01- Understanding software testing

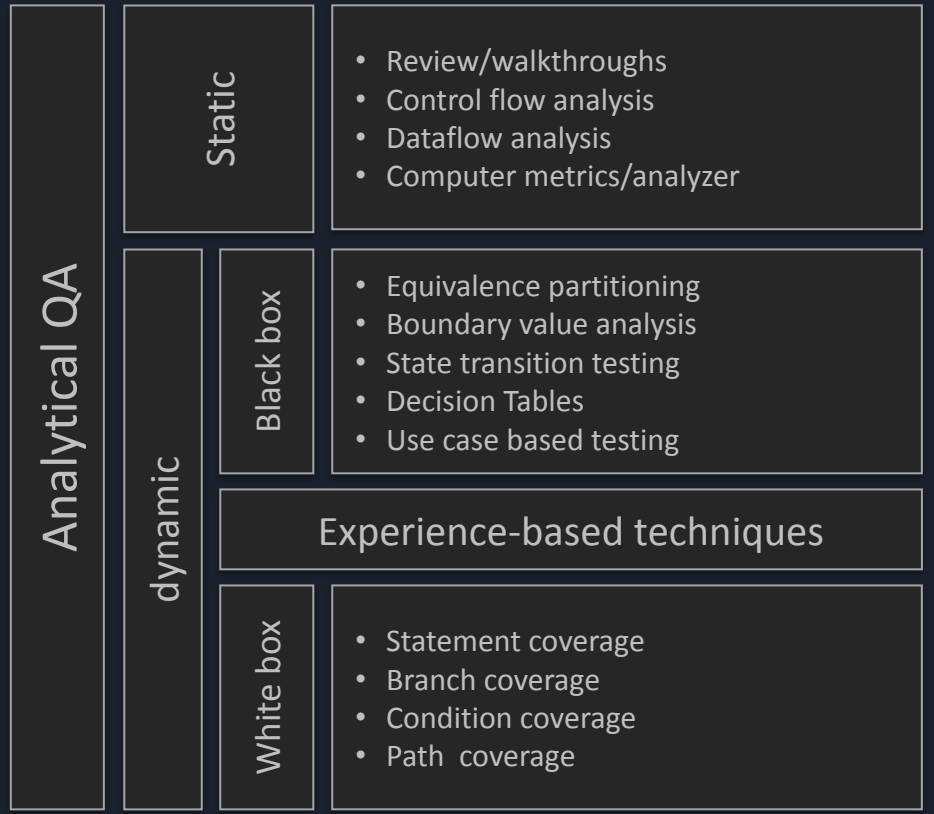
Analytical Quality Assurance

Quality of product- Verification and test procedure

Motto

Defects should be detected

As early as possible in the process



I-Fundamentals of Testing

01- Understanding software testing

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.

I-Fundamentals of Testing

01- Understanding software testing

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.

I-Fundamentals of Testing

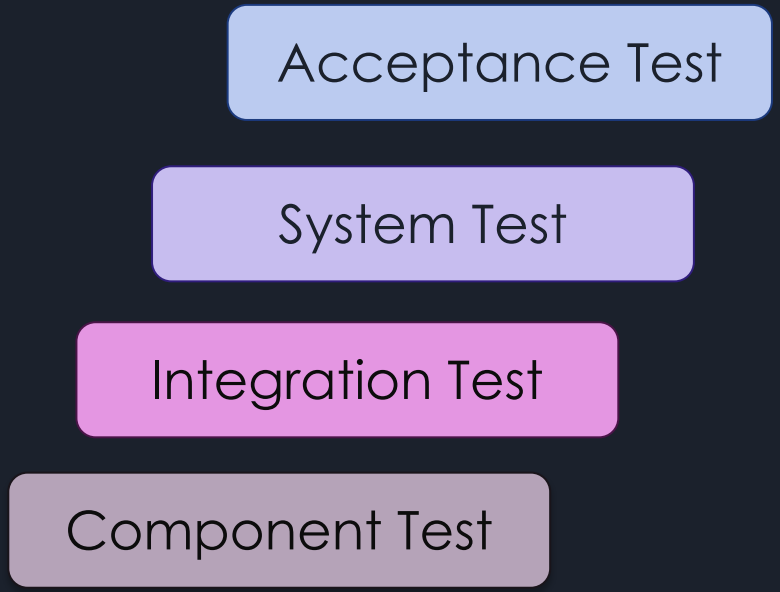
01- Understanding software testing

Test Case Template:

- Identity
- Conditions
- Values
- Relations
- Optionals

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)

Testing levels of the V-Model



Component Test

Component Testing

Definition

Test of each software component after its realization

Because of the naming of components in different programming languages, the component test may be referred to as:

- **module test** (e.g. in C)
- **class test** (e.g. in java or C++)
- **unit test** (e.g. in Pascal)

- The components are referred to as modules, classes or units.
- Because of the possible involvement of developers in the test execution, they are called **developer's test**

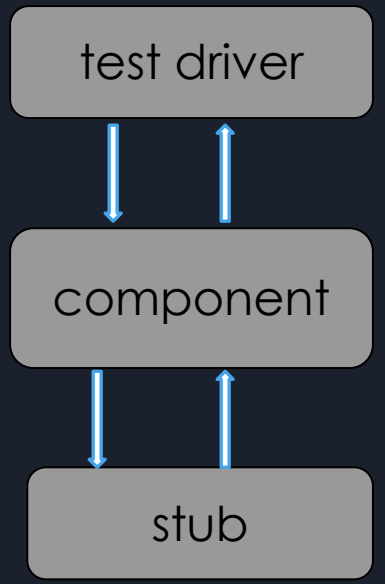
Component testing: Scope

- Only **single components** are tested
  - Components may consists of several smaller units
  - Test objects often cannot be tested stand alone
- **Every** component is tested **on its own**
  - Finding failures caused by internal defects
  - Cross effects between components are not within the scope of this test
- **Test cases** may be **derived** from
  - Component specifications
  - Software design
  - Data model

Component testing: Test harness

Test execution of components often requires drivers and stubs

- Drivers handle the interface to the component
  - **drivers** simulate inputs, record outputs and provide a test harness
- **Stubs** replace or simulate components not yet available or not part of the test object
- To **program drivers and/or stubs** you
  - must have **programming skills**
  - must to have the source code available
  - may need special tools



Component testing: Methods

- The program code is available to the tester
- in case “tester = **developer**”: testing take place with a strong **development focus**
- knowledge about **functionality**, **component structure** and **variable** may be applied to **design test case**
- often functional testing will apply additionally, the use of **debuggers** and other development tools
- (e.g. unit test frameworks) will allow to **directly access** program **variables**
- Source code knowledge allows to use white box methods for component test

Integration test

Integration testing (also: interface testing)

- Integration is the activity of combining individual software components into a larger subsystems
- Examine the **interaction** of software elements (components) after integration
- **Further** integration of **subsystems** is also part of the system integration process.
- Each component has already been tested for its **internal functionality** (component test).
- Integration test examine the **external functions**.
- May be performed by **developers**, **testers** both

Integration testing: Scope

- Integration tests examine the **interaction** of software components (subsystems) with each other
  - interfaces with the other **components**
  - interfaces among **GUIs / MMIs**
- Integration tests examine the interfaces with the **system environment**.
  - In most cases, the interaction testes is that of the component and **simulated environment** behavior.
  - Under **real conditions**, additional environmental factors may **influence** the components behavior
- **Test case** may be derived from**, interface specifications**, architectural design or data models.

System test

System test

- Testing the integrated software system to prove compliance with the specified requirements
- Software quality is looked at from the user's point of view
- System tests refer to:
  - **functional** and **non-functional** requirements (functionality, reliability)
- **Test cases** may be derived from
  - - Functional specifications
  - - Use cases
  - - Business processes
  - - Risk assessments

System test: functional requirements

- Three approaches for testing functional requirements:
  - **Business process based test**
    - each business process as basis for driving test cases
- the ranking order of the business process can be applied for prioritizing test cases
  - **Use case based test :**
    - Test cases are derived from sequences of expected or reasonable use
    - sequence used more frequently receive a higher priority
  - **Requirements based test**
    - Test cases are derived from the requirement specification
    - The number of the cases will vary accounting to the type/depth of specification Requirements based test

System test: Non-functional requirements

- Non-functional requirements is **difficult to achieve:**
  - Their definition is often very vague (e.g. easy to operate, well structured user interface, etc.)
  - They are not stated explicitly. They are an implicit part of system description, still they are expected to be fulfilled
  - **Quantifying** them **is difficult,** often non-objective metrics must be used, e.g. looks pretty, quite safe, easy to learn.
- Testing / inspiriting documentation
  - Is documentation of programs in live with the actual system, is it concise, complete and easy to understand?
- Testing maintainability
  - Have all programmers complied with the respective Coding-Standards?
  - Is the system designed in a structured, modular fashion?

Acceptance test

Objectives of acceptance testing

Acceptance testing, like system testing, typically focuses on the behavior and capabilities of a whole system or product. Objectives of acceptance testing include:

- Establishing confidence in the quality of the system as a whole
- Validating that the system is complete and will work as expected
- Verifying that functional and non-functional behaviors of the system are as specified

so that it can be accepted by the customer.

❖ Acceptance testing may produce information to assess the system's readiness for deployment and use by the customer.

Common forms of acceptance testing include the following:

- Contractual and regulatory acceptance testing
- Operational acceptance testing
- Alpha and beta testing (for COTS)

Contractual and regulatory acceptance testing

- ***The main objective*** is building confidence that contractual or regulatory compliance has been achieved.
- ***Contractual acceptance testing***
  - performed against a contract's acceptance criteria for producing custom-developed software.
  - Acceptance criteria should be defined when the parties agree to the contract.
  - often performed by users / independent testers.
  - All the **contractual** requirements are fulfilled?
- ***Regulatory acceptance testing***
  - performed against any regulations that must be adhered to, such as government, legal, or safety regulations.
  - often performed by users / independent testers, sometimes with the results being witnessed or audited by regulatory agencies.
- With formal acceptance **legal milestone** are reached: begin of **warranty**, **payment milestones**, **maintenance agreements**, etc.
- Often **customer** select **test case** for acceptance testing
  - Possible misinterpretations of the requirements come to light and can be discussed, "The customer knows best"
- Testing is done using the **customer environment**
  - Customer environment may cause new failures

Acceptance testing: operational acceptance testing

- ***The main objective*** of operational acceptance testing is building confidence that the operators or system administrators can keep the system working properly for the users in the operational environment, even under exceptional or difficult conditions.
- The tests focus on operational aspects, and may include:
  - Testing of backup and restore
  - Installing, uninstalling and upgrading
  - Disaster recovery
  - User management
  - Maintenance tasks
  - Data load and migration tasks
  - Checks for security vulnerabilities
  - Performance testing
  - Compatibility with other systems (hardware, OS, database, etc.)
- Operational acceptance testing is often done by the customer's system administrator

Acceptance testing: alpha and beta (or filed) testing

- Alpha and beta testing are typically used by developers of commercial off-the-shelf (COTS) software who want to get feedback from potential or existing users, customers, and/or operators before the software product is put on the market.
- ***Alpha testing*** is performed at the developing organization's site, not by the development team, but by potential or existing customers, and/or operators or an independent test team.
- ***Beta testing*** is performed by potential or existing customers, and/or operators at their own locations.
- Beta testing may come after alpha testing
- ***Objective of alpha and beta testing*** -
    - building confidence among potential or existing customers, and/or operators that they can use the system under normal, everyday conditions, and in the operational environment(s) to achieve their objectives with minimum difficulty, cost, and risk.
    - may be the detection of defects related to the conditions and environment(s) in which the system will be used, especially when those conditions and environment(s) are difficult to replicate by the development team.

Acceptance testing: Test basis /1

Examples of work products that can be used as a test basis for any form of acceptance testing include:

- Business processes
- User or business requirements
- Regulations, legal contracts and standards
- Use cases and/or user stories
- System requirements
- System or user documentation
- Installation procedures
- Risk analysis reports

Acceptance testing: Test basis /2

In addition, as a test basis for deriving test cases for operational acceptance testing, one or more of the following work products can be used:

- Backup and restore procedures
- Disaster recovery procedures
- Non-functional requirements
- Operations documentation
- Deployment and installation instructions
- Performance targets
- Database packages
- Security standards or regulations

Acceptance testing: Typical test objects

Typical test objects for any form of acceptance testing include:

- System under test
- System configuration and configuration data
- Business processes for a fully integrated system
- Recovery systems and hot sites (for business continuity and disaster recovery testing)
- Operational and maintenance processes
- Forms
- Reports
- Existing and converted production data

Acceptance testing: Typical defects and failures

Examples of typical defects for any form of acceptance testing include:

- System workflows do not meet business or user requirements
- Business rules are not implemented correctly
- System does not satisfy contractual or regulatory requirements
- Non-functional failures such as security vulnerabilities, inadequate performance efficiency under high loads, or improper operation on a supported platform

Thank You