# Definitions

## Anomaly

* Any **condition that** **deviates from expectation** based on requirements specifications, design documents, user documents, standards, etc. or from someone’s perception or experience.

## Error (mistake)

* A human action that produces an **incorrect result.**

## Bug/Defect/Fault/Problem

* A flaw in a component or system that can cause the component or system to **fail to perform its required function.**
* A **defect**, if encountered during execution, may **cause a failure** of the component or system.

## Failure

* **Actual deviation** of the component or system from its expected delivery, service or result

## Defect/fault masking

* An occurrence in which **one defect prevents the detection of another**.

## Software Quality

* The degree to which a system, component, or process **meets specified requirements.**
* The degree to which a system, component, or process **meets customer or user needs or expectations.**

## Role of testing in software development, maintenance and operations

* Reduce the risk of problems
* Reduce long-term defect-related costs
* Contribute to the quality of the software
* Help meeting standards:
  + Contractual or legal requirements
  + Industry-specific standards

## Testing and Quality

* Testing can give confidence in the quality of the software **if it finds few or no defects.**
* If defects are found, the quality increases when those **defects are fixed.**
* Lessons learnt from previous mistakes improve **future performance.**

## Testing

* The process of **exercising** software
  + To verify that it satisfies specified requirements and to detect errors
* The process of **analyzing** a software item
  + To detect the differences **between existing and required conditions** (that is, bugs),
  + To **evaluate the features** of the software item
* The activity that **initially finds failures** in a software item
* Main Test Activities
* Planning and control (how is the software going to be tested, which and how much tests there will be; control – are the tests designed and executed, daily meetings)
* Choosing test conditions
* Designing and executing test cases
* Checking results
* Evaluating exit criteria
* Reporting on the testing process and system under test
* Finalizing or completing closure activities
* Main Objectives
* Finding defects (if no defects are found maybe the software is not high quality)
* Gaining confidence about the level of quality (more test and results => more confidence for the software)
* Providing information for decision-making (is the software ready – are there enough tests, executed and ready for release, QA makes the decision)
* Preventing defects (more early testing – much better and cheaper)

## Debugging

* The development activity that finds, analyses and removes the **cause of the failure.**

## Re-testing

* **Ensures** that the fix does indeed **resolve the failure.**

## Seven Testing Principles

#### Testing shows presence of defects

* Testing can show that defects are present
* Cannot prove that there are no defects
* Appropriate testing reduces the probability for defects

#### Exhaustive testing is impossible

* All combinations of inputs and preconditions are usually **almost infinite number**
* Testing everything is **not feasible**
  + Except for trivial cases
* **Risk analysis and priorities** should be used to focus testing efforts

#### Early testing

* Testing activities shall be started as early as possible
  + And shall be focused on defined objectives
* The later a bug is found – the more it costs!

#### Defect clustering

* Testing effort shall be **focused proportionally**
  + To the expected and later observed defect density of modules
* **A small number of modules** usually contains most of the defects discovered
  + Responsible for most of the operational failures

#### Pesticide paradox

* Same tests repeated over and over again tend to lose their effectiveness
  + Previously undetected defects remain undiscovered
* New and modified test cases should be developed

#### Testing is context dependent

* Testing is done differently in different contexts
  + Example: safety-critical software is tested differently from an e-commerce site.

#### Absence-of-errors fallacy

* Finding and fixing defects itself does not help in these cases:
  + The system built is **unusable**
  + Does not fulfill the **users’ needs and expectations**

## Test Planning and Control

* **Starts at the beginning** of the software development project
* Must be **regularly checked**, updated, and adjusted

## Test Control

* **Monitoring** of the test activities
* **Comparing** with the plan
* **Reporting** status of deviations from the plan
* Taking actions for **correction**
* **Updating** the test plan according to the feedback

## Test Prioritization

* Software projects are often run under severe **time pressure**
* Prioritization guarantees that the critical **software parts are tested first**

## Test Plan

* The results from the planning activities should be documented in a **test plan**
* The **test plan** is a formal document that describes how tests will be performed
  + List of test **activities** to be performed to ensure meeting the requirements
  + **Features** to be tested, testing approach, schedule, acceptance criteria

## Test Analysis and Design

* Identify **test conditions**
* Defining what should be tested
* An item or event of a component or system that could be verified by one or more test cases
* E.g., a function, transaction, feature, quality attribute, or structural element
* Designing **test cases**
* Defining what should be tested starts with reviewing the **test basis**
* **Product specification** may not be testable
  + Unclear expected outcomes or behaviors
  + Rework of the requirements has to be done

## Test Basis

* All documents from which the requirements of a component or system can be inferred. The documentation on which the test cases are based. 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 post conditions, developed for a particular objective or test condition, such as to exercise a particular program path or to verify compliance with a specific requirement
* According to the **level of concreteness:**
  + **Logical** test cases
    - They have to be defined first
    - Do not include concrete input/output values
  + **Concrete** test cases
    - The actual inputs that are chosen
    - Priority of the next phase of the test process
* **Initial situation (precondition)** must be described
  + Needed environmental conditions
* Which **results** and **behavior** are expected
* Outputs
* Changes to global (persistent) data and states
* Any other consequences of the test case
* Test cases can be designed for:
* **Expected inputs**
* Specified behavior, output, and reaction
* Specified handling of exception and error cases
* **Unexpected inputs**
* Invalid and unexpected inputs or conditions
* Have no specified exception handling

### The Test Oracle

* + A mechanism for **predicting the expected results**
    - Can be the product specification
    - Can be another similar product
    - The result can be inverted and compared to the initial input
  + The **code itself** should not be used as a test oracle

### Test Case Execution

* + Follows the **priority** of the test cases set in the test plan
  + Grouping test cases into **test suites**

### Test Protocols

* + Tests without a **protocol** are of no value
  + The test execution must be exactly and completely **logged**
    - **What** tests were made
    - **Who** made the tests
    - **Which parts**
    - **When**
    - **How intensively**
    - With what **results**
    - **Software version**

### Reproducibility

* + The tests must be **easily repeated**
  + Test environment
  + Input data
  + Test logs
  + Etc.

### Failure Found?

* + Is it really a failure?
    - Erroneous or inexact test **specification**
    - Problematic test **infrastructure** or test case
    - Incorrect test **execution**
  + If it is a failure:
    - The failure must be **documented**
    - Rough analysis of **possible causes**
    - **Additional test cases** might be required

## Test Implementation and Execution

* Test conditions and logical test cases are transformed into **concrete test cases**
* The **environment** is set up to support the test execution activity
* Tests are **executed and logged**

## Exit Criteria

* The set of generic and specific conditions for **permitting a process to be officially completed**
  + Agreed upon with the stakeholders
* Used to report against and to plan when to stop testing
* Test exit criteria might turn to be **unrealistic**
  + Then exit criteria should be corrected

## Test Summary Report

* Summary reports might have different **size**
  + **Simple message** to the project manager
    - Used in lower level tests
    - E.g., component tests
  + **Formal reports** for the stakeholders
    - Used in higher-level tests
    - E.g., integration tests, system tests

## Test Closure Activities – Save the Experience

* The experience gathered should be analyzed and made available for further projects
  + Achieved results
  + Unexpected events
    - What were their causes?
  + Open change requests
    - Why were they not implemented?
  + User acceptance after deploying

## Metrics and Measurement

* What can be subjected to a metric and tracked through measurement?
  + Test **coverage**
  + **Defects**
    - Including total found, total fixed, current backlog, average closure periods, and configuration, subsystem, priority, or severity distribution
  + **Workload** and **resource** usage
  + Planned and actual **costs**
* Metrics and measurements should be applied throughout the software development lifecycle
  + Should be aligned with project goals and objectives
  + This enables test analysts to **track and report** **test and quality results** to management in a consistent and coherent way

## Lack of Metrics

* A lack of metrics and measurements leads to purely **subjective assessments** of quality and testing
* This results in **disputes** over the meaning of test results toward the end of the lifecycle
* Also results in a lack of clearly perceived and communicated **value**, **effectiveness**, and **efficiency** for testing

1. **Unit**

* The smallest compilable component

1. **Component**

* A unit is a component
* The integration of **one** or more components is a component
  + “**One**” stands for components that call themselves recursively

1. **Stubs**

* In Component testing **called components** are replaced with **stubs**, simulators, or trusted components

1. **Drivers**

* **Calling components** are replaced with **drivers** or trusted super-components