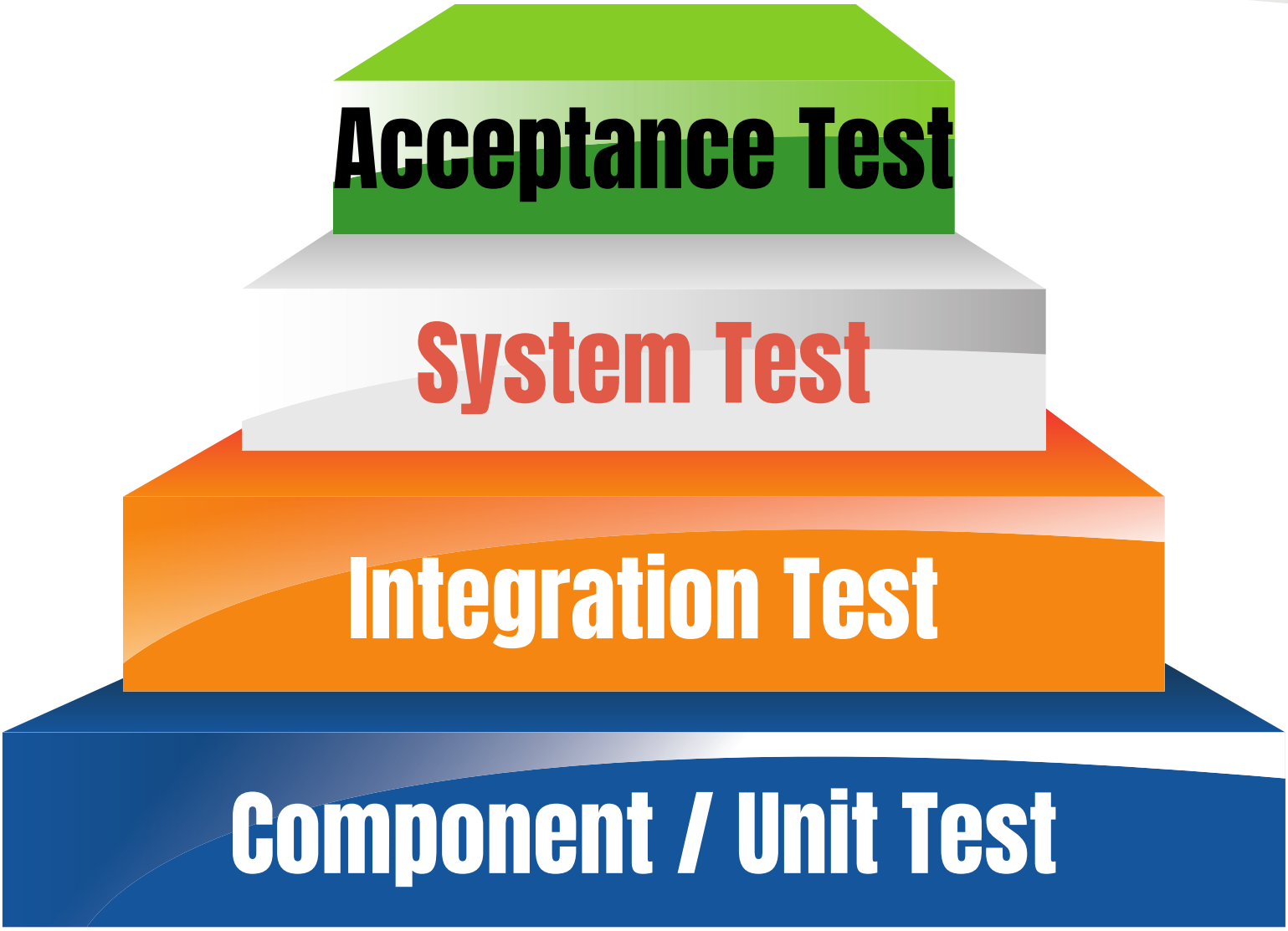
1

Back to the basics

In testing there are **4 Test Levels.** They can be encountered when testing both a **system** and a **system of systems** (multiple, dispersed, independent systems in context as part of a larger, more complex system).



Component / Unit Test

focuses on components that are separately testable.

Integration Test

focuses on interactions between components or systems.

System Test

focuses on the behavior and capabilities of a whole system or product, often considering the end-to-end tasks the system can perform and the non-functional behaviors.

Acceptance Test

focuses on the behavior and capabilities of a whole system or product: confidence, completion, fit for use & purpose.

The best way to navigate across the Test Levels is to follow a predefined course, to constantly monitor it as you go along and apply course corrections whenever you see fit.

Start with a

Plan

that you will

Monitor

by using

Metrics

in order to

Control

Test Planning

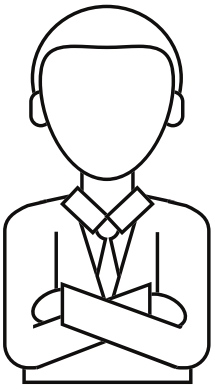
Test planning applies for each test level and also includes the methods for monitoring for each.
Is the activity of establishing or updating a test plan which starts at the initiation of the test process and in line with the Test Strategy.

Test Plan

is a document describing the scope, approach, resources and schedule of intended test activities.

As a record of the test planning process, it also covers:

- test items
- features to be tested
- testing tasks
- who will do each task
- degree of tester independence
- test environment
- test design techniques
- Entry and Exit Criteria (with their rationale)
- risk assessment based on requirements
- contingency planning based on risk assessment
- integration of reactive test techniques in execution



During test planning, the **Test Manager** **defines** the approach for **each level**:

- What is tested
- Goals & Objectives
- Test techniques & tools

In order to have an effective planning, we need to consider the complex relationships between test phases, but also between development and test. Some examples would be:

- the requirement traceability matrix
- informal transfer of information.

In other words, the **requirement traceability** matrix is a document that maps and traces user requirement with test cases. The main purpose of Requirement Traceability Matrix is to see that all test cases are covered so that no functionality should miss while doing Software testing.

Another factor for effective planning would be the proper listing of the testing scope with each feature associated with a design specification, environment, etc.

Contact with all stakeholders has to be initiated at this stage, but also all external dependencies identified and service level agreements put in place.

In order to properly measure the progress, evaluate the Entry and Exit criteria and to exercise control, we need to put in place metrics starting with Test Planning.

Test Plan content example as per IEEE829 standard

- Test plan identifier
- Introduction
- Test items
- Features to be tested
- Features not to be tested
- Approach
- Item pass/fail criteria
- Suspension criteria and resumption requirements
- Test deliverables
- Testing tasks
- Environmental needs
- Responsibilities
- Staffing and training needs
- Schedule
- Risks and contingencies
- Approvals

Metrics

Are a measurement scale and the method used for measurement.

It is important that the proper set of metrics is established as they are mainly used to measure the progress of testing. This will also enable testers to report results in a consistent way and with coherent tracking (Example: % of test coverage, % of test execution, etc.).

Although they should be as automated as possible to allow immediate understandings of where we are, metrics should be defined based on specific objectives that can also be presented to stakeholders at various meetings, for various concerns.

Project metrics

measure progress toward established project exit criteria.

Product metrics

measure some attribute of the product, such as the extent to which it has been tested or the defect density.

Process metrics

measure the capability of the testing or development process, such as the percentage of defects detected by testing.

People metrics

measure the capability of individuals or groups, such as the implementation of test cases within a given schedule.

Monitor & Control

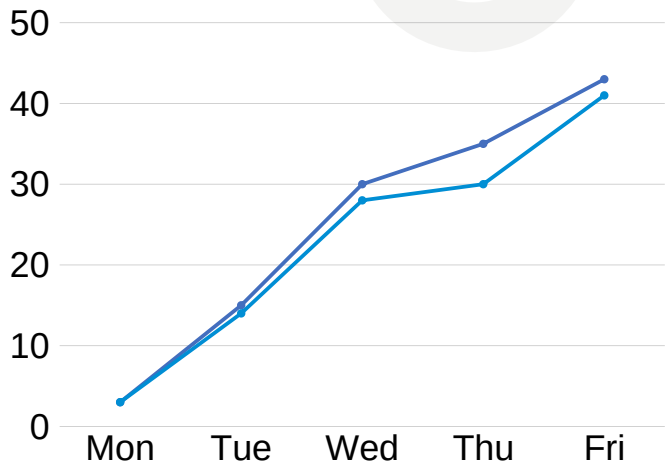
A testing schedule and monitoring framework need to be established to track progress versus plan. Due to this, all ongoing activities should have targets which are tracked via ongoing measurements.
When I am stating all ongoing activities, I am also referring at test **analysis**, test **design**, test **implementation** and not only at test **execution**.

It is important to be able to relate the information and status of the test work products in an understandable and relevant manner based on the audience of those reports. Not everyone is looking for the same level of details.

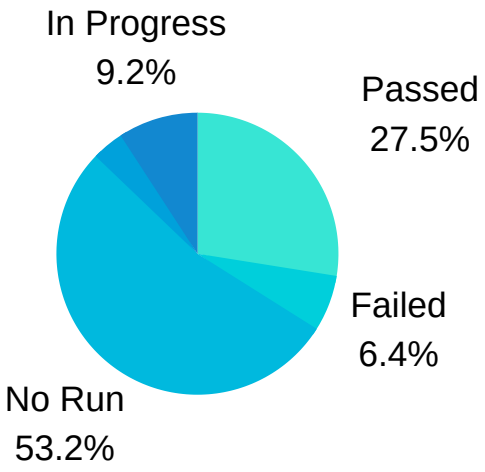
The aim of **test control** is to compare actual progress versus the plan and **implement corrective actions**.

Common examples

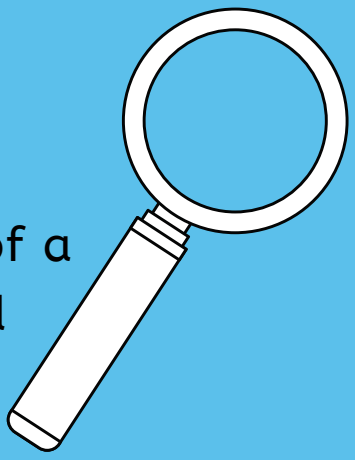
Test Condition execution vs plan



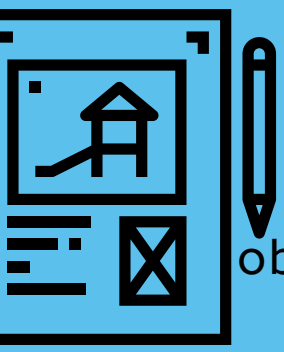
Test Case status



Test Analysis

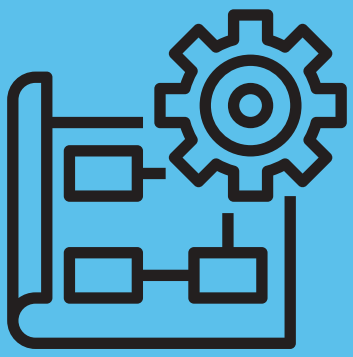


Is process of analyzing the test basis (all documents from which the requirements of a component or system can be inferred) and defining test objectives.



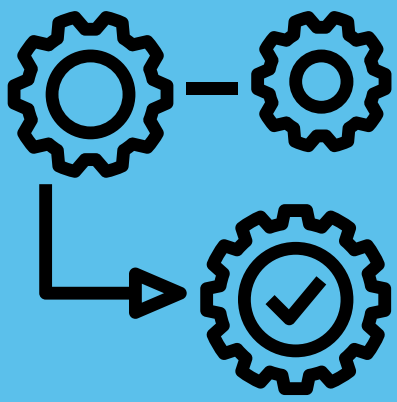
Test Design

Is the process of transforming general test objectives into tangible test conditions and test cases.



Test Implementation

Is the process of developing and prioritizing test procedures, creating test data and, optionally, preparing test harnesses and writing automated test scripts.



Test Execution

Is the process of running a test on the component or system under test, producing actual result.

Test Analysis

Is process of analyzing the test basis (all documents from which the requirements of a component or system can be inferred) and defining test objectives.

Covers WHAT is to be tested in the form of test conditions and can start as soon as the basis for testing is established for each test level.
It can be performed in parallel, integrated or iteratively with Test Design.

Evaluates and reviews the test objectives and product risks, while it defines detailed measures and targets for success.
Deciding on the level of detail should consider:

- The level of testing; level of detail and quality of the test basis
- System/software complexity and development lifecycle used
- Project and product risk
- Relationship between test basis, what is to be tested and how is to be tested
- Test management tool used
- The level of maturity of the test process and the skills and knowledge of the test analysts
- The level at which Test Design and other test work products are specified
- Availability of stakeholders for consultation

Test Condition

Is an item or event of a component or system that could be verified by one or more test cases (ex: function, transaction, feature, etc.).

A test condition may or may not specify values or variables. It all depends on the context at that test level. Some might be generic like "Test Payment" and others may be specific like "Test Payment with VISA for 3 items and a cost over 100$".

Don't forget, if you go specific, then expect a higher number of test conditions. Check what you need at that stage and adapt. It may not be the same for Component Test as for System Test.

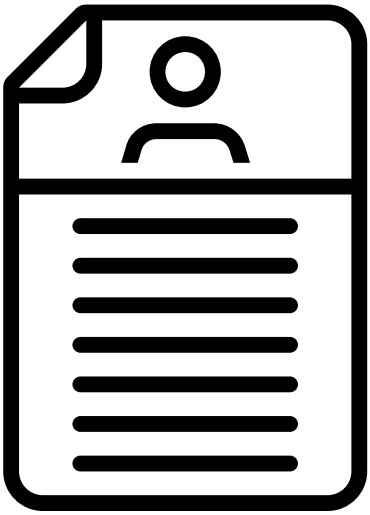
advantages of detailed test conditions

- More flexibility in relating other test work products
- Better and more detailed monitoring and control
- Contributes to defect prevention by occurring early
- Relates testing work products to stakeholders in terms that they can understand
- Influences and directs other testing activities, but also other development activities
- Enables test design, implementation and execution to be optimized by more efficient coverage
- Basis for clearer horizontal traceability within a test level

disadvantages of detailed test conditions

- Potentially time-consuming
- Maintainability can become difficult
- Level of formality needs to be defined and implemented across the team

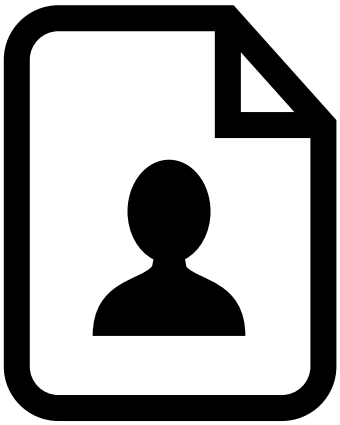
GO detailed when



- Lightweight test design documentation methods
- Little or no formal requirements or other development work products
- The project is large-scale, complex or high risk

GO generic when

- Component (Unit) level testing
- Less complex projects where simple hierarchical relationships exist
- Acceptance testing where use cases can be utilized to help define tests



Test Design

Is an item or event of a component or system that could be verified by one or more test cases (ex: function, transaction, feature, etc.).

Covers HOW something is to be tested by identifying test cases with step wise elaboration for the test conditions (from Test Analysis) or from the test basis using techniques identified in the test strategy or plan.

This phase can start for a given Test Level once Test Conditions are identified and enough information is available to enable the production of **Test Cases**.

In other words, a test case is 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.

Although it can be merged together with Test Analysis, for higher levels of testing it will remain a separate activity.

It is likely that some tasks that normally occur during test implementation will be integrated into the test design process. Especially when using an iterative approach.

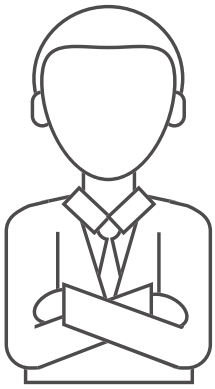
The coverage of test conditions by either creating low-level and high-level test cases can be optimized by the creation of test data starting in Test Design.

Test Implementation

Is the process of developing and prioritizing test procedures, creating test data and, optionally, preparing test harnesses and writing automated test scripts.

This is when tests are **organized and prioritized** and when test designs are implemented as test cases, test procedures and test data.

It is of great importance to pick the right tests and run them in the right order. The importance of this even grows exponentially in risk-based strategies when we prioritize based on the likelihood of risk and problems.



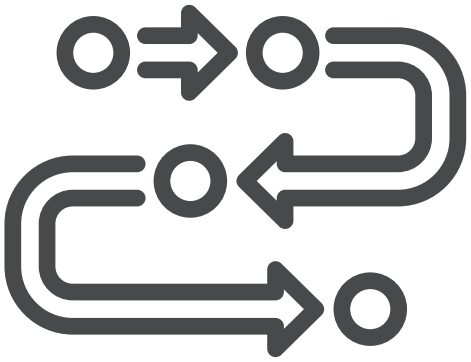
At this stage, the **Test Manager** **should** ensure:

- delivery of test environment
- delivery of test data
- constraints, risks and priorities are checked
- test team is ready for execution
- entry criteria is checked (explicit/implicit)

Some organizations may follow the IEEE829 standard to define inputs and their associated expected results during testing. Other only have rigorous rules when they need to provide evidence of compliance for regulatory projects or for adherence to standards.

In the most common cases, the test inputs are usually documented together with expected results, test steps and stored test data.

Just like test conditions and test cases, even during test implementation we will face the decision to go into an extensive (detailed) stage or to have a light (generic) approach. This decision should be taken by your understanding of the development lifecycle and by the predictability of software features under test.

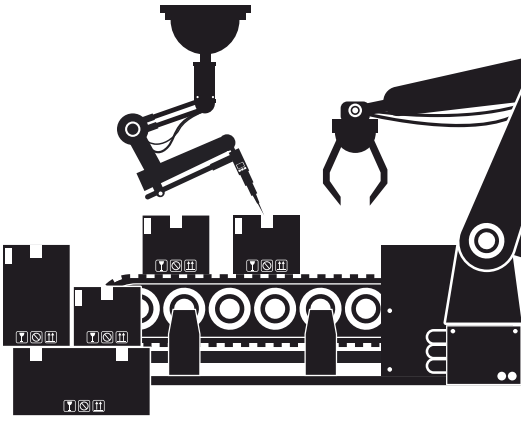


For example, in agile or iterative lifecycles where code changes dramatically from iteration to iteration, the implementation work changes significantly between each stage.

Please do not count off the extensive implementation preparation due to the above:

- Concrete test cases provide working examples of how the software behaves
- When tests are archived for long term and re-use in regression these details may become valuable
- Domain experts are likely to verify versus a concrete test rather than an abstract business rule
- Further weakness in software specification is identified

Some defects can be found only in production-like test environments. These are often expensive to procure and difficult to configure and manage. Similar challenges are also faced for the use of production data or production like data which can even lead to data privacy or other headaches.



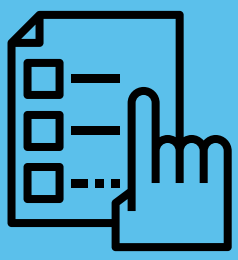
Test implementation is not all about manual testing, this is the stage where automation scripting takes place, the stage where automation versus manual prioritization and execution order is established.

And I am not talking only about automation, even tool acquisition is done here, especially for test data generation required to prepare for load, volume or performance testing.

Quick reminder before moving forward

Test Case

A set of preconditions, inputs, actions (where applicable), expected results and post conditions, developed based on test conditions.

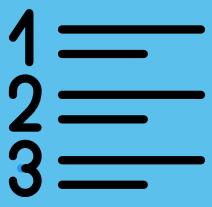


Test Script

A sequence of instructions for the execution of a test.

Test Suite

groups of test scripts, as well as a test execution schedule.



Test Charter

An instruction of test goals and possible test ideas on how to test. Documentation of test activities in session-based exploratory testing.

Test Execution

Is the process of running a test on the component or system under test, producing actual result.

Should finish before execution starts

- Tests are designed or at least defined
- Tools are in place for test management and defect management and test automation (if applicable)
- Standards for test logging and defect reporting are published

Execution begins once

- The test object is delivered
- The Entry criteria for test execution is met

During execution, a **Test Managers role** is to:

- Monitor progress according to the plan
- Initiate and carry out control actions to guide testing
- Ensure that test logs provide an adequate record of relevant details for tests and events

During execution it is important to keep a traceability between test conditions, the test basis and the test objective and to have the appropriate level of test logging.

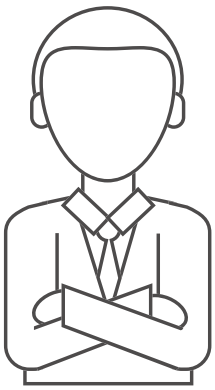
Time should be reserved for experienced-based and defect-based test sessions driven by testers findings.

Entry Criteria

- Set of generic and specific conditions for permitting a process to go forward with a defined task
- Purpose is to prevent a task from starting which would entail more effort compared to the effort needed to remove the failed entry criteria

Exit Criteria

- Set of generic and specific conditions, agreed with stakeholders for permitting a process to complete
- Prevent a task from being considered completed when there are still outstanding tasks not finished
- Used to report progress against a plan and to know when to stop testing

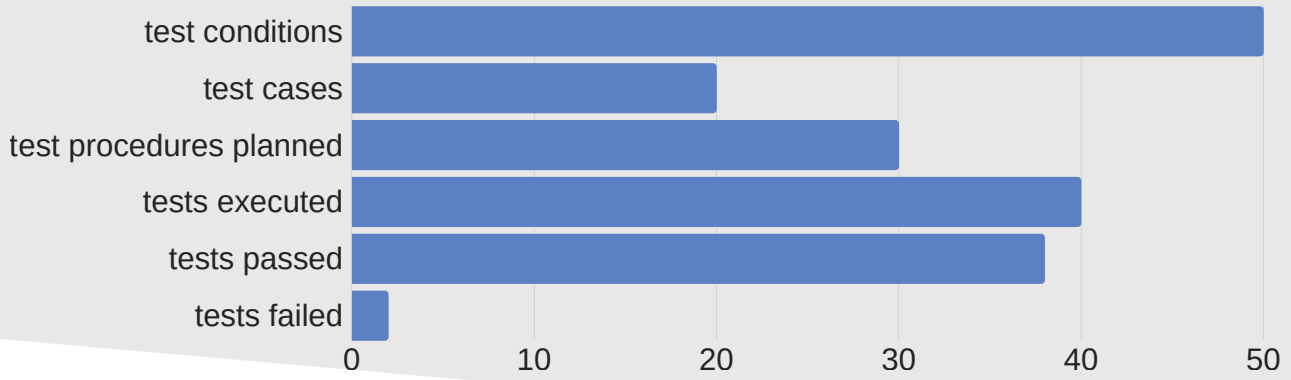


A **Test Managers should**:

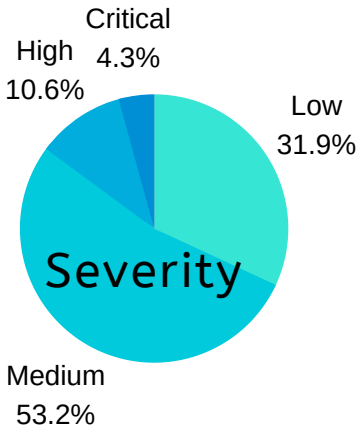
- ensure that effective processes are in place to provide necessary information for evaluating entry & exit criteria
- make sure that the definition of the information requirements and methods for collection are part of test planning
- ensure that members of the test team are responsible for providing the information required in an accurate and timely manner

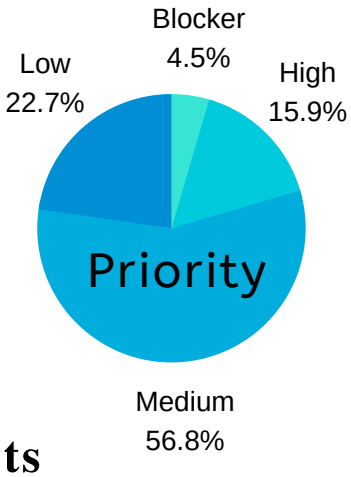
The evaluation of exit criteria and reporting of results is a test management activity.

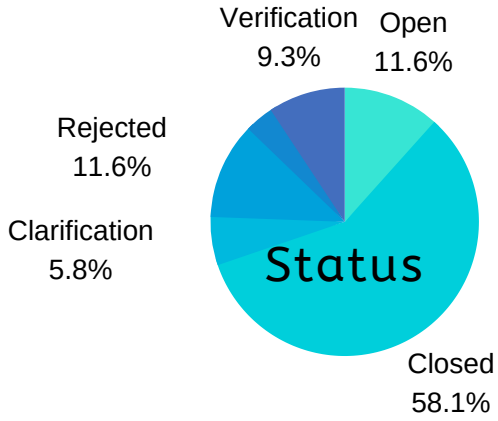
Number of test conditions, cases, pass, failed, etc.



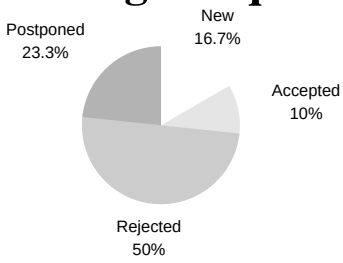
Total defects, classified by severity, priority, status



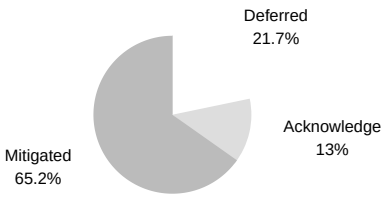




Change requests



Quality risks



Planned versus actual

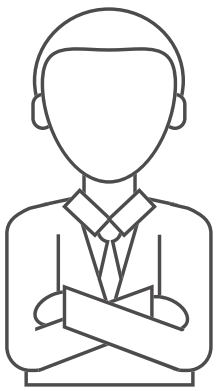
Test Closure

consists of finalizing and archiving the testware and evaluating the test process, including preparation of a test evaluation report.

Once test execution is deemed to be complete, the key outputs should be captured:

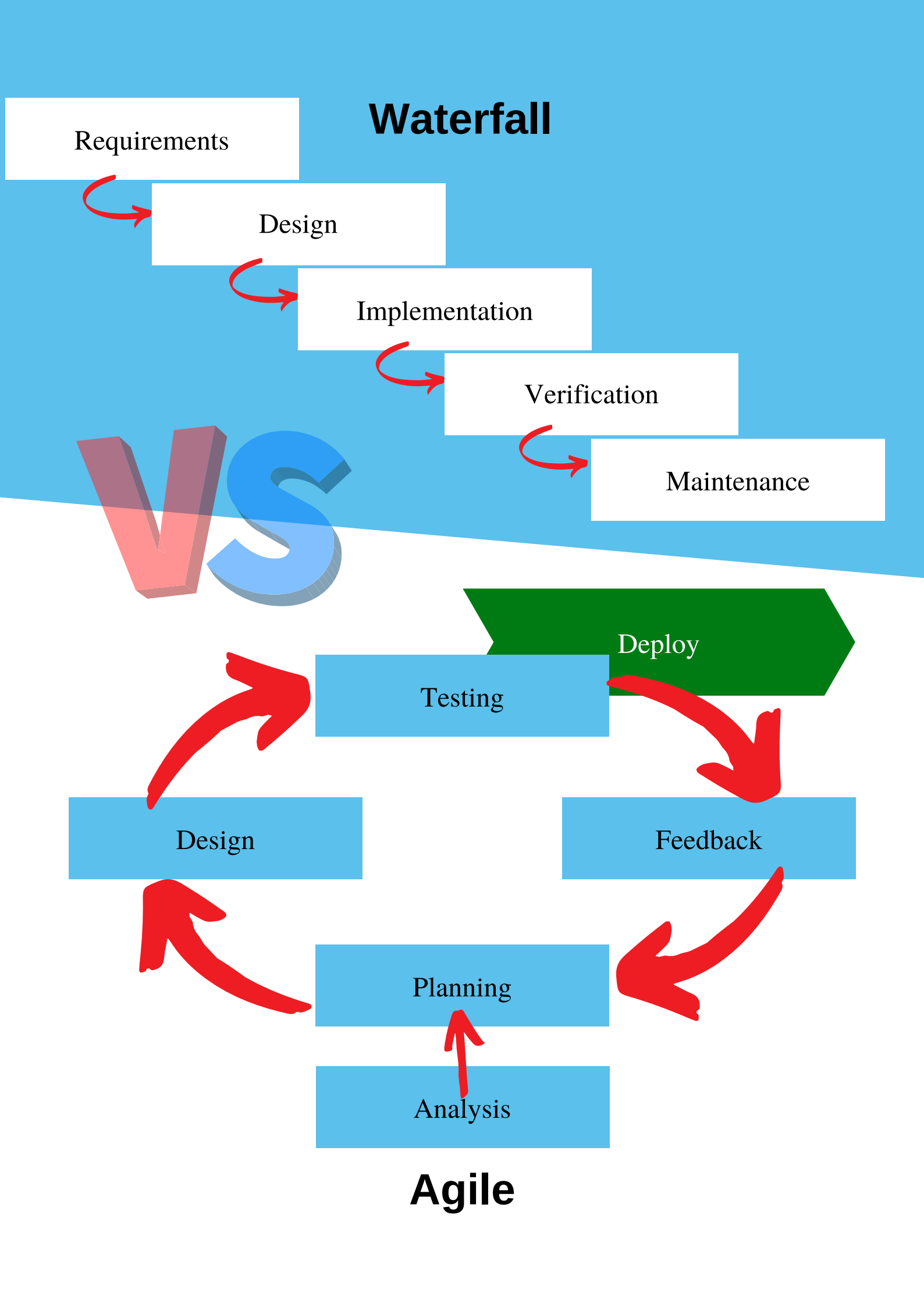
- **Test completion check** - ensuring that all test work is indeed concluded
- **Test artifacts handover** - delivering valuable work products to those who need them
- **Lessons learned** - performing or participating in retrospective meetings where important lessons
- **Archiving** results, logs, reports, and other documents

These tasks are important (often missed) and should be explicitly included as part of the test plan.

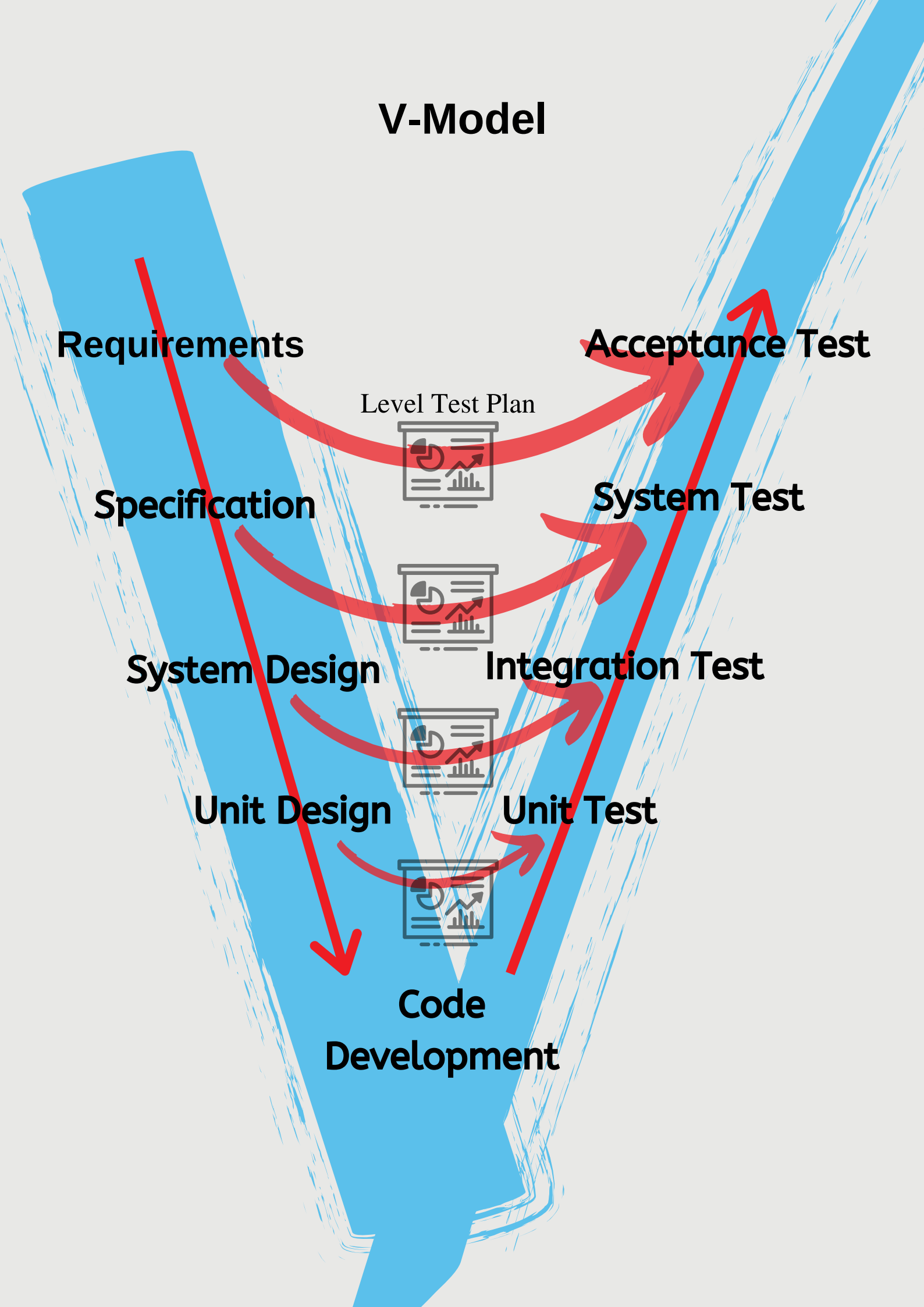


A **Test Managers should**:

- Look for opportunities to reuse test work products
- Keep in mind that retrospectives should apply to testing as well as to the entire project and indeed the wider organization. Problems tend to be systemic, not isolated


Waterfall
Requirements
Design
Implementation
Verification
Maintenance
VS
Deploy
Testing
Design
Feedback
Planning
Analysis
Agile

V-Model


Requirements
Acceptance Test
Level Test Plan
Specification
System Test
System Design
Integration Test
Unit Design
Unit Test
Code
Development

Glossary

Each of the terms specified below are defined as per the ISTQB® Glossary which is displayed online at:

https://glossary.istqb.org/en/search/

software lifecycle

The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software lifecycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and sometimes, retirement phase. Note these phases may overlap or be performed iteratively.

system of systems

Multiple heterogeneous, distributed systems that are embedded in networks at multiple levels and in multiple interconnected domains, addressing large-scale inter-disciplinary common problems and purposes, usually without a common management structure.

test basis

The body of knowledge used as the basis for test analysis and design.

test planning

The activity of establishing or updating a test plan.

test plan

Documentation describing the test objectives to be achieved and the means and the schedule for achieving them, organized to coordinate testing activities.

measurement

The process of assigning a number or category to an entity to describe an attribute of that entity.

metric

A measurement scale and the method used for measurement.

test analysis

The activity that identifies test conditions by analyzing the test basis.

test design

The activity of deriving and specifying test cases from test conditions.

test condition

An aspect of the test basis that is relevant in order to achieve specific test objectives.

test case

A set of preconditions, inputs, actions (where applicable), expected results and postconditions, developed based on test conditions.

test script

A sequence of instructions for the execution of a test.

test procedure

A sequence of test cases in execution order, and any associated actions that may be required to set up the initial preconditions and any wrap up activities post execution.

test log

A chronological record of relevant details about the execution of tests.

test implementation

The activity that prepares the testware needed for test execution based on test analysis and design.

test execution

The process of running a test on the component or system under test, producing actual result(s).

test control

A test management task that deals with developing and applying a set of corrective actions to get a test project on track when monitoring shows a deviation from what was planned.

exit criteria

The set of conditions for officially completing a defined task.

test closure

During the test closure phase of a test process data is collected from completed activities to consolidate experience, testware, facts and numbers. The test closure phase consists of finalizing and archiving the testware and evaluating the test process, including preparation of a test evaluation report.

test summary report

A test report that provides an evaluation of the corresponding test items against exit criteria.

Test Techniques

boundary value analysis

A black-box test technique in which test cases are designed based on boundary values.

branch testing

A white-box test technique in which test cases are designed to exercise branches.

cause-effect graphing

A black-box test design technique in which test cases are designed from cause-effect graphs.

classification tree method

A black-box test design technique in which test cases, described by means of a classification tree, are designed to execute combinations of representatives of input and/or output domains.

condition testing

A white-box test design technique in which test cases are designed to execute condition outcomes.

condition determination testing

A white-box test technique in which test cases are designed to exercise single condition outcomes that independently affect a decision outcome.

control flow analysis

A form of static analysis based on a control flow graph.

data flow analysis

A form of static analysis based on the definition and usage of variables.

decision table testing

A form of static analysis based on a control flow graph.

decision testing

A white-box test technique in which test cases are designed to execute decision outcomes.

defect-based test design technique

A procedure to derive and/or select test cases targeted at one or more defect types, with tests being developed from what is known about the specific defect type.

defect taxonomy

A system of (hierarchical) categories designed to be a useful aid for reproducibly classifying defects.

dynamic analysis

The process of evaluating behavior, e.g., memory performance, CPU usage, of a system or component during execution.

error guessing

A test technique in which tests are derived on the basis of the tester's knowledge of past failures, or general knowledge of failure modes.

equivalence partitioning

A black-box test technique in which test cases are designed to exercise equivalence partitions by using one representative member of each partition.

exploratory testing

An approach to testing whereby the testers dynamically design and execute tests based on their knowledge, exploration of the test item and the results of previous tests.

experience-based test technique

A procedure to derive and/or select test cases based on the tester's experience, knowledge and intuition.

multiple condition testing

A white-box test design technique in which test cases are designed to execute combinations of single condition outcomes (within one statement).

pairwise testing

A black-box test design technique in which test cases are designed to execute all possible discrete combinations of each pair of input parameters.

path testing

A white-box test design technique in which test cases are designed to execute paths.

requirements-based testing

An approach to testing in which test cases are designed based on test objectives and test conditions derived from requirements, e.g., tests that exercise specific functions or probe non-functional attributes such as reliability or usability.

specification-based technique
A procedure to derive and/or select test cases based on an analysis of the specification, either functional or non-functional, of a component or system without reference to its internal structure.
static analysis
The process of evaluating a component or system without executing it, based on its form, structure, content, or documentation.
statement testing
A white-box test technique in which test cases are designed to execute statements.
state transition testing
A black-box test technique using a state transition diagram or state table to derive test cases to evaluate whether the test item successfully executes valid transitions and blocks invalid transitions.
structure-based technique
A procedure to derive and/or select test cases based on an analysis of the internal structure of a component or system.
test charter
Documentation of test activities in session-based exploratory testing.
use case testing
A black-box test technique in which test cases are designed to execute scenarios of use cases.
wild pointer
A pointer that references a location that is out of scope for that pointer or that does not exist.

Types of Testing

accessibility testing
Testing to determine the ease by which users with disabilities can use a component or system.
accuracy testing
Testing to determine the accuracy of a software product.
black-box testing
Testing, either functional or non-functional, without reference to the internal structure of the component or system.

heuristic evaluation

A usability review technique that targets usability problems in the user interface or user interface design. With this technique, the reviewers examine the interface and judge its compliance with recognized usability principles (the "heuristics").

interoperability testing

Testing to determine the interoperability of a software product.

maintainability testing

Testing to determine the maintainability of a software product.

operational acceptance test

Operational testing in the acceptance test phase, typically performed in a (simulated) operational environment by operations and/or systems administration staff focusing on operational aspects, e.g., recoverability, resource-behavior, installability and technical compliance.

portability testing

Testing to determine the portability of a software product.

recoverability testing

Testing to determine the recoverability of a software product.

reliability testing

Testing to determine the reliability of a software product.

security testing

Testing to determine the security of the software product.

suitability testing

Testing to determine the suitability of a software product.

usability testing

Testing to evaluate the degree to which the system can be used by specified users with effectiveness, efficiency and satisfaction in a specified context of use.

white-box testing

Testing based on an analysis of the internal structure of the component or system.

Exercises

ISTQB® Foundation 2011: 3
ISTQB® Foundation 2018 Exam A: 7, 8
ISTQB® Foundation 2018 Exam B: 6, 7
ISTQB® Foundation 2018 Exam C: 6

ASTQB® Foundation Exam 1: 9, 10, 11, 12, 13
ASTQB® Foundation Exam 2: 9, 10, 11, 12, 13

ISTQB® Advanced Test Manager: 1, 2, 3, 4, 5, 6, 7, 8, 9

ASTQB® Advanced Test Manager: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14