

Test Phases

Each phase of the software testing life cycle allows developers to assess specific characteristics of the software and evaluate whether the software is suitable for use. The various evaluations during testing can identify errors and deficits in the application before it enters production and deployment. Finding and resolving such issues early on helps preserve your reputation and clients' confidence in your work.

Early and effective software testing can also be financially beneficial. By allowing developers to address flaws in software design, functionality and security as soon as testers discover them, software testing spares the need for costly changes to the software while it's in wide use. Resolving such problems during development also helps ensure that customers have high regard for the software, potentially leading to increased sales.

What are the 5 phases of testing software?

In the software testing life cycle, there are usually five phases of testing:

1. Static testing

During static testing, developers work to avoid potential problems that might arise later. Without executing the code, they perform manual or automated reviews of the supporting documents for the software, such as requirement specifications, searching for any potential ambiguities, errors or redundancies. The goal is to preempt defects before introducing them to the software system.

2. Unit testing

The next phase of software testing is unit testing. During this phase, the software undergoes assessments of its specific units, or its functions and procedures, to ensure that each works properly on its own. The developers may use white box testing to evaluate the software's code and internal structure, commonly before delivering the software for formal testing by testers. Unit testing can occur whenever a piece of code undergoes change, which allows for quick resolution of issues.

3. Integration testing

Integration testing involves testing all the units of a program as a group to find issues with how the separate software functions interact with one another. Through integration testing, the developers can determine the overall efficiency of the units as they run together. This phase is important because the program's overall functionality relies on the units operating simultaneously as a complete system, not as isolated procedures.

4. System testing

In the system testing phase, the software undergoes its first test as a complete, integrated application to determine how well it carries out its purpose. For this, the developers pass the software to independent testers who had no involvement in its development to ensure that the testing results stem from impartial evaluations. System testing is vital because it ensures that the software meets the requirements as determined by the client.

5. Acceptance testing

Acceptance testing is the last phase of software testing. Its purpose is to evaluate the software's readiness for release and practical use. Testers may perform acceptance testing alongside individuals who represent the software's target audience. Acceptance testing aims to show whether the software meets the needs of its intended users and that any changes the software experiences during development are appropriate for use. The representative individuals are crucial to this phase because they can offer insight into what customers may want from the software. Once the software passes acceptance testing, it moves on to production.

5 types of additional software tests


In addition to the five main phases of testing, you may also need additional tests to evaluate software. The necessity of these tests depends on the type of company you work for and the software you're creating. These tests include:

1. Performance testing

In performance testing, testers evaluate how well the software handles various scenarios and workloads. There are several subtypes of performance testing. A common performance test is load testing, which recreates real-life user conditions to determine how the software performs in common scenarios. Another test is stress testing, in which testers intentionally overload the software to discover how much it can handle before it fails.

2. Regression testing

Regression testing is a procedure that occurs throughout the testing life cycle. After developers implement a change to the software, testers perform regression testing to ensure that previously tested and functional operations remain intact. This is a good way to help guarantee consistency in functionality.

This document is available free of charge on  **studocu**

3. Usability testing

Usability testing focuses on ease of use. Testers approach the software from the perspective of end users, validating that its interface and design are simple to understand and that the application is easy to operate. Ideally, the user can learn the software on their own and enjoy a satisfying experience with it.

4. Compatibility testing

Compatibility testing evaluates the software's ability to function as intended in various computing environments. These include operating systems, mobile platforms and web browsers. Such environments have their own specifications for the software they run, so it's important to confirm that the software meets all of those differing specifications.

5. Security testing

Security testing is an assessment of the software in terms of threats, risks and vulnerabilities. Testers might examine the software for flaws that expose a user's personal data to hackers or make the software susceptible to malware. If testers find any such flaws, the developers can secure them with coding.

Test Strategy

A high-level document is used to validate the test types or levels to be executed for the product and specify the **Software Development Life Cycle's** testing approach is known as Test strategy document.

Once the test strategy has been written, we cannot modify it, and it is approved by the **Project Manager, development team.**

The test strategy also specifies the following details, which are necessary while we write the test document:

- o **What is the other procedure having to be used?**
- o **Which module is going to be tested?**
- o **Which entry and exit criteria apply?**
- o **Which type of testing needs to be implemented?**

In other words, we can say that it is a document, which expresses how we go about testing the product. And the approaches can be created with the help of following aspects:

- o **Automation or not**
- o **Resource point of view**

We can write the test strategy based on **development design documents**.

The development design document includes the following documents:

- o **System design documents:** Primarily, we will use these documents to write the test strategy.
- o **Design documents:** These documents are used to specify the software's functionality to be enabled in the upcoming release.
- o **Conceptual design documents:** These are the document which we used Infrequently.

The Objective of Test Strategy

- o The primary objective of writing the test strategy is to make sure that all purposes are covered entirely and understood by all stakeholders, we should systematically create a test strategy.
- o Furthermore, a test strategy objective is to support various quality assurance stockholders in respect of **planning of resources, language, test and integration levels, traceability, roles and responsibilities**, etc

Features of Test Strategy Document

- o In **SDLC (Software Development Life Cycle)**, the test strategy document plays an important role. It includes various significant aspects, such as who will implement the testing, what will be tested, how it will be succeeded, and what risks and incidents will be are related to it.
- o Some of the additional characteristics of the Test Strategy document are as follows:

○ The test strategy document is approved and reviewed by the following's peoples:

- **Test Team Lead**
- **Development Manager**
- **Quality Analyst Manager**
- **Product Manager**

This document is available free of charge on

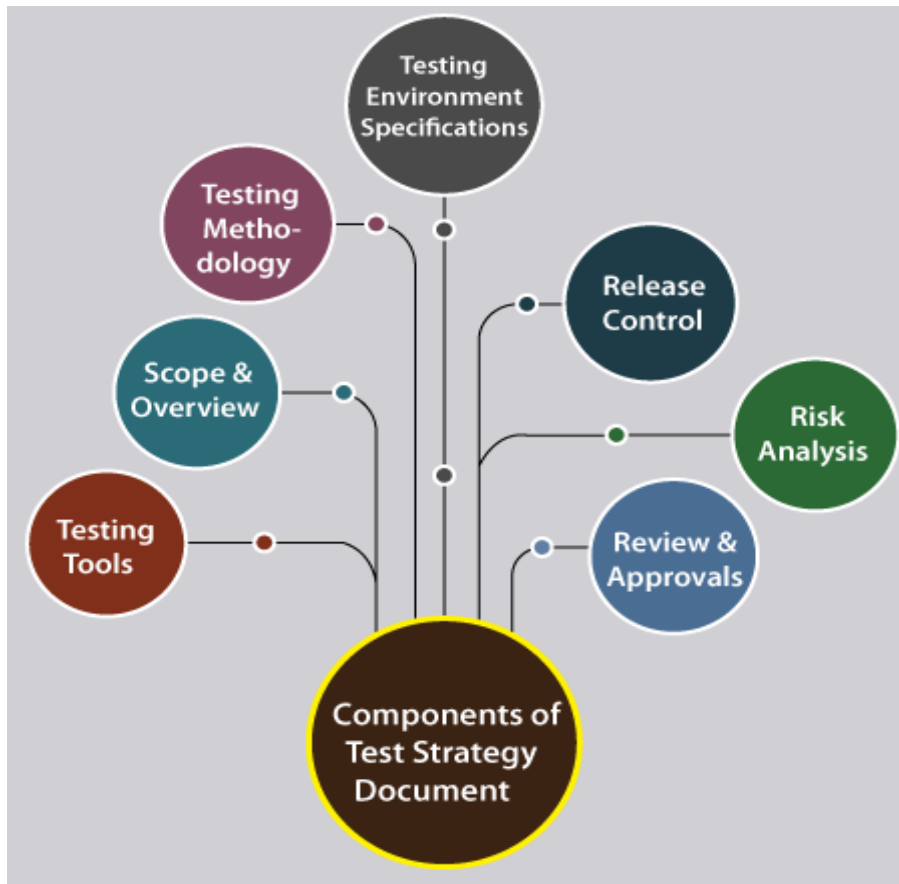


- For different testing activities, the test strategy document specifies the resources, scope, plan, methodology, etc.
- In order to direct how testing will be achieved, it is used by the project test team once it is ready or completed.
- Primarily, it is obtained from the **BRS (Business Requirements Specifications)** documents.
- The test strategy document is a high-level document, which generally remains constant, implying no frequent and pointless modification is made in the document.
- The respective team easily accomplishes the objectives of testing with the help of a test strategy document.
- The respective team easily accomplishes the objectives of testing with the help of test strategy document.

Components of Test Strategy Document

We understand that the test strategy document is made during the requirements phase and after the requirements have been listed.

Like other testing documents, the test strategy document also includes various components, such as:




- **Scope and Overview**
- **Testing Methodology**
- **Testing Environment Specifications**
- **Testing Tools**
- **Release Control**
- **Risk Analysis**
- **Review and Approvals**

Let's see them one by one for our better understanding:

1. Scope and Overview

- The first component of the test strategy document is **Scope and Overview**.
- The overview of any product contains the information on who should approve, review and use the document.

This document is available free of charge on  **studocu**

- The test strategy document also specified the testing activities and phases that are needed to be approved.

2. Testing Methodology

- The next module in the test strategy document is **Testing methodology**, which is mainly used to specify the **levels of testing, testing procedure, roles, and responsibilities** of all the team members.
- The testing approach also contains the change management process involving the modification request submission, pattern to be used, and activity to manage the request.
- Above all, if the test strategy document is not established appropriately, then it might lead to **errors or mistakes** in the future.

3. Testing Environment Specifications

- Another component of the test strategy document is **Testing Environment Specification**.
- As we already aware of the specification of the **test data** requirements is exceptionally significant. Hence, clear guidelines on how to prepare test data are involved in the testing environment specification of the test strategy document.
- This module specifies the information related to **the number of environments and the setup demanded**.
- The backup and restore strategies are also offered to ensure that there is no data loss because of the coding or programming issues.

4. Testing Tools

- **Testing tools** are another vital component of the test strategy document, as it stipulates the complete information about the **test management** and **automation tools** necessary for test execution activity.;
- For **security, performance, load testing**, the necessary methodologies, and tools are defined by the details of **the open-source or commercial tool** and the number of users that can be kept by it.

5. Release Control

- Another important module of the test strategy document is **Release Control**.

- It is used to ensure that the correct and effective **test execution** and release management strategies should be systematically developed.

6. Risk Analysis

- The next component of the test strategy document is **Risk Analysis**.
- In the test strategy document, all the possible risks are described linked to the project, which can become a problem in test execution.
- Furthermore, for inclining these risks, a clear strategy is also formed in order to make sure that they are undertaking properly.
- We also create a contingency plan if the development team faces these risks in real-time.

7. Review and Approvals

- The last component of the Testing strategy document is **Review and Approval**.
- When all the related testing activities are specified in the test strategy document, it is reviewed by the concerned people like:
 - **System Administration Team**
 - **Project Management Team**
 - **Development Team**
 - **Business Team**
- Together with the **correct date, approver name, comment, and** summary of the reviewed variations should be followed while starting the document.
- Likewise, it should be constantly reviewed and updated with the testing process improvements.

Types of Test Strategies

Here, we are discussing some of the significant types of test strategies document:

Types of Test Strategies



- **Methodical strategy**
- **Reactive strategy**
- **Analytical strategy**
- **Standards compliant or Process compliant strategy**
- **Model-based strategy**
- **Regression averse strategy**
- **Consultative strategy**

Let's understand them one by one in detail:

1. METHODICAL STRATEGY

- The first part of test strategy document is **Methodical strategy**.
- In this, the test teams follow a **set of test conditions, pre-defined quality standard**(like ISO25000), **checklists**.
- The Standard checklists is occurred for precise types of testing, such as **securitytesting**.

2. REACTIVE STRATEGY

- The next type of test strategy is known as **Reactive strategy**.
- In this, we can design the test and execute them only after the real software is delivered, Therefore, the **testing is based upon the identified defects** in the existing system.
- Suppose, we have used the **exploratory testing**, and the test approvals are established derived from the existing aspects and performances.
- These test approvals are restructured based on the outcome of the testing which is implemented by the test engineer.

3. ANALYTICAL STRATEGY

- Another type of test strategy is **Analytical strategy**, which is used to perform testing based on requirements, and requirements are analyzed to derive the test conditions. And then **tests are designed, implemented, and performed** to encounter those requirements. **For example, risk-based testing or requirements-based testing.**
- Even the outcomes are recorded in terms of **requirements**, such as **requirements tested and passed.**

4. STANDARDS COMPLIANT OR PROCESS COMPLIANT STRATEGY

- In this type of test strategy, the test engineer will follow the **procedures or guidelines created by a panel of industry specialists or committee standards** to find test conditions, describe test cases, and put the testing team in place.
- Suppose any project follows the **Scrum** Agile technique. In that case, the test engineer will generate its complete test strategy, beginning from classifying test criteria, essential test cases, performing tests, report status, etc., around each **user story**.
- Some **good examples** of the standards-compliant process are **Medical systems following US FDA (Food and Drugs Administration) standards.**

5. MODEL-BASED STRATEGY

- The next type of test strategy is a **model-based strategy**. The testing team selects the **current or expected situation** and produces a model for it with the following aspects: inputs, **outputs, processes, and possible behavior**.
- And the models are also established based on the current data speeds, software, hardware, infrastructure, etc.

6. REGRESSION AVERSE STRATEGY

- In the regression averse strategy, the test engineer mainly **emphasizes decreasing regression risks** for **functional or non-functional** product shares.
- **For example**, suppose we have one web application to test the regression **issues** for the particular application. The testing team can develop the test automation for both **typical and exceptional use cases** for this scenario.
- And to facilitate the tests can be run whenever the application is reformed, the testing team can use **GUI-based automation tools**.

7. Consultative strategy

- The consultative strategy is used to consult **key investors as input to choose the scope** of test conditions as in user-directed testing.
- In order of priority, the client will provide a list of **browsers and their versions, operating systems, a list of connection types, anti-malware software**, and also the contradictory list, which they want to test the application.
- As per the need of the items given in provided lists, the test engineer may use the various testing techniques, such as **equivalence partitioning**

We can combine the two or more strategies as per the needs of the product and organization's requirements. And it is not necessary to use any one of the above listed test strategies for any testing project.

TEST STRATEGY SELECTION

The selection of the **test strategy** may depend on the below aspects:

- The selection of test strategy depends on the **Organization type and size**.
- We can select the test strategy based on the **Project requirements**, such as **safety and security** related applications require rigorous strategy.
- We can select the test strategy based on the **Product development model**.

The final document of the test strategy contains important details about the following factors:

- Scope and Overview
- Re-usability of both software and testing work products.
- Details of different Test levels, relationships between the test levels, and procedure to integrate different test levels.
- Testing environment
- Testing techniques
- Level of automation for testing
- Different testing tools
- Risk Analysis
- For each test level Entry as well exit conditions
- Test results reports
- Degree of independence of each test
- Metrics and measurements to be evaluated during testing
- Confirmation and regression testing
- Managing defects detected

- Managing test tools and infrastructure configuration
- Roles and responsibilities of Test team members
- **Conclusion**

After understanding the **test strategy document**, at last, we can say that the test strategy document provides a vibrant vision of what the test team will do for the whole project.

The test strategy document could prepare only those who have good experience in the **product domain** because the test strategy document will drive the entire team.

And it cannot be modified or changed in the complete project life cycle as it is a static document.

Before any testing activities begin, the Test strategy document can distribute to the entire testing team.

If the test strategy document is written correctly, it will develop a high-quality system and expand the complete testing process.