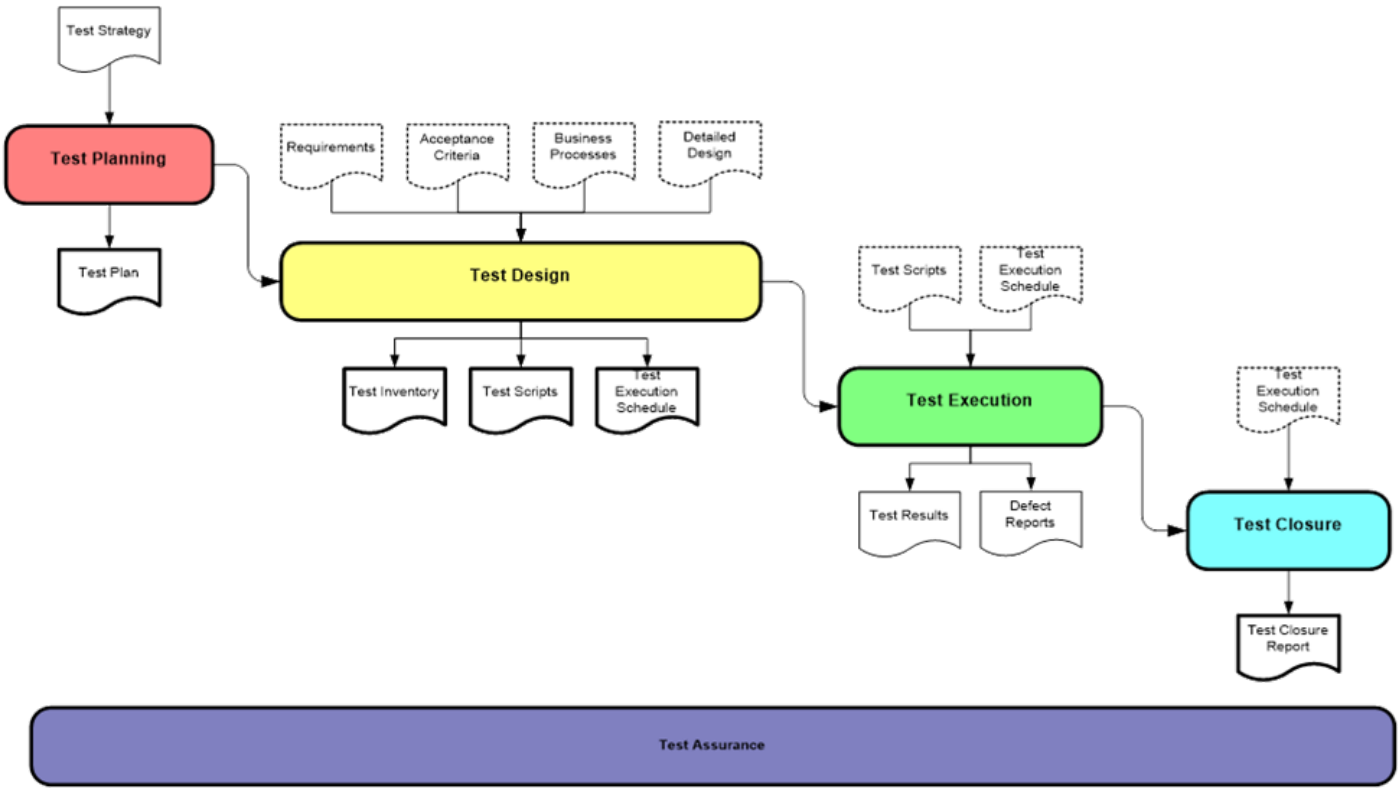
Fundamental Test Planning Process and Strategies

The four stages of the Fundamental Testing Process must be completed in the correct order. These are Test planning, Test Design, Test Execution and Test Closure. These stages should be carried out in this logical order for any Test Level i.e Unit, System, System Integration Testing and User Acceptance Testing (UAT)) to maximize the benefits to the testing process and overall quality assurance.

Whilst at first glance using the process may appear excessive for some projects, the stages can be as formal or informal as required. This underpins the Context-Driven-Testing philosophy which states, “The value of any practice depends on its context”. As long as the basic principles of the process are being followed then the testing process will produce the required result. The test process may vary in terms of duration and activities involved depending on the development methodology being used by the development teams. For example, Application Development are very much Agile in their development methodology while other areas such as research Systems are very much traditionally waterfall / V Model in their development methodology.

Below is an overview of each stage of the Fundamental Testing Process and also the test deliverables that should be produced as the outputs of the stage.

1. Test Planning
2. Test Design
3. Test Execution
4. Test Closure



To aid and assist in each of the test stages, the following sections provide some guiding principles on how to ensure each stage is performed effectively and to obtain the optimum benefits (both to testing and the wider project delivery).

Test Planning

The objective of test planning is to prescribe the scope, approach, resources, and schedule of testing activities for a level of testing. It also includes identifying the items being tested, the features to be tested, the testing tasks to be performed, and the personnel responsible for each task.

Make sure the resources needed to support the test design and execution phases are decided upon and assigned early enough to allow for sufficient test preparation.
To do this, a Test Initiation Meeting should be held at the very beginning of the Project.
For each project, a test plan should be prepared and evaluated with end users, user groups, and all other relevant stakeholders whenever practicable (and practical). At least the following steps should be respected:

1) First, analyze product structure and architecture.
2) Design the test strategy.
3) Define all the test objectives.
4) Define the testing area.
5) Define all the usable resources.
6) Schedule all activities in an appropriate manner.
7) Determine all the Test Deliverables.
8) Collapse your test plan, avoid overlapping and redundancy.

9) Update plan when needed and do not use an outdated and unused document

Testing risks and Mitigations

Any risks that will affect the testing process during a Project / Change should be listed along with the mitigation. By documenting a risk, its occurrence can be anticipated well ahead of time and therefore proactive action may be taken to prevent it from occurring, or to mitigate its damage. An example of a risk to testing could be that a particular testing tool required for that project may not be available. Any such risks to testing should be documented in the Test Plan for that test level / project.

Test schedules

The length of time it will take to complete the testing phase should be estimated. The test schedule for the particular test level should be formed from the estimations and included in the test plan. Scheduling for testing should factor in the following:

- First of all, testers are required to run each test case at least once (usually covering new features and regression testing).
- Additionally, if a flaw is discovered, the developers will need to resolve the issue. The failing test case should then be retested until it is working properly.
- Last but not least, the tester must perform further regression testing toward the end of the testing cycle to ensure that the software's other components have not changed.

Test priorities

As part of a risk -based approach to testing, when designing test cases it is important to assign priorities for each test or group of tests. When testing is required on projects, certain test cases will be treated as the most critical tests and if they fail, the product cannot be released. Some other test cases may be deemed less critical if they were to fail. As part of the Fundamental Test Process, the test design activity will involve creation of test cases that will form 3 test methods:

1. Verify the new functionality / changes being introduced (verification testing). This involves testing of written requirements
2. Attempt to find any unexpected behaviors in functionality / changes being introduced (defect discovery testing). Known as testing the unwritten requirements
3. Verify that existing functionality continues to behave and perform as normal following introduction of changes (regression testing)

Each test should have a priority weighting attached to it:

- High – Any tests deemed to be in this category mean they must be run initially as the functionality being tested is critical and if failed would have a severe impact
- Medium - Any tests deemed to be in this category mean they must be run only after the High category tests. The functionality being tested is important and if failed would have a high but not severe impact

- Low - Any tests deemed to be in this category mean they must be run only after the medium category tests. These are effectively ‘nice to have tests. The functionality being tested is not critical important and if failed would only impact a small volume of users whilst at the same

It is worth mentioning that the weightings above must, where possible, be agreed with key stakeholders, especially the stakeholder who has requested the Project / Change.
Whilst this may seem like overkill, this is introducing a risk-based view to test coverage and if the time allocated for testing is reduced the high priorities areas would be able to be covered first.

Test case design

Different projects may require different levels of test documentation. Ensure any test cases that are created are done so only if they are going to add value.

Not every project will require the creation of large volumes of test cases, but practically every project will need some kind of documented testing.

Before testing starts, make sure tests have been designed, evaluated, and authorized. Test cases ought to include:

1) The reference to the requirement being tested. This is to ensure traceability of requirements.
2) Description of the function or feature being tested.
3) Name of person who designed the test & date.
4) Test data & instructions to run the test
5) Expected results.
6) Level of priority as detailed above

Testers should approach all testing with a "test to break" mentality while planning and carrying out tests, with the goal of identifying any faults.

- It is crucial to keep in mind that tests should be designed to test both the specified and unwritten requirements (including usual user behavior and arbitrary scenarios that could push the system to its breaking point).
- Test Scenarios should be the first part of the test design process. Scenarios should be captured in a “Test Scenario Inventory”.
- Scenarios should be kept very short and used as a high-level view to describe the function or process to be tested.
- Scenarios should be written so that they can be easily reviewed by anyone wishing to understand what is being tested.

Test execution

Before starting any Test Execution phases where testers may be unfamiliar with the testing process, it is worthwhile to have a preliminary informal meeting to go through the key expectations such as:

1. How the testers will record the results. This includes agreeing on any automated tools for tracking testing and recording results.
2. Walkthrough of some example test cases to make them feel more comfortable. This will avoid a lot of downtime during the testing window with testers asking questions.
3. How the testers should record defects. Walk through the process of how issues will be reported and what information is required.
4. How long the testers have to complete the testing. This will help testers focus and plan execution of their test cases within the available time.
5. Who will be required to support the testing activities. For example developers may be required to support testing in the event issues are identified that block testing progress. Similarly this also ensures the required resources from the relevant business units or subject matter experts are available.
6. Location for the testing activities. Attempt to bring the testers together in one room or location where possible.
7. Test progress reporting. Detail what should be reported on, frequency and audience..

Test closure

Make sure to run all the tests that can be executed.

Make sure you have justifications for any tests that cannot be completed in writing.
Take the time to analyze the results independently and make sure you are comfortable with the message you will be conveying before any Go/No Go quality gates at the final stage of testing.

After testing is completed, look at any open defects.

1. If any defects are open, then reasons for defects being open will need to be given. The defects should be analysed and impact on the software and operations documented. Based on this a decision should be made on which defects must be closed before the project or change can go to Production
2. Stakeholders will understand these points as long as you have documented reasons.

Test levels

Utilizing test levels, or test phases as they are more frequently known, encourages quality risk mitigation as early as feasible and to the fullest extent possible. The table that follows lists the different test levels that, unless otherwise specified in the applicable Test Plan, should be carried out for all Projects and Changes.

Note: There should be a specific representative assigned to each Test Level who is in charge of carrying out or supervising the activity.
The notable exception to this rule is the System Integration Test Level, which is only necessary when a project or change introduces changes that have an impact on multiple systems or a system's interface with another system.

There are 4 levels of Testing, each of which carries a specific functional purpose, to be carried out in chronological order.

Level	Owner	Objective	Key areas of testing
Unit	Development Team	● Detect defective code in units ● Reduce risk of unit failure in Live Service	● Functionality ● Resource utilization
System	Development Team	● Detect defects in end-to-end scenarios concentrating on systematically validating whether each function performs as expected or not ● Assist in mitigating risk of unmet business requirements	● Functionality ● Performance ● Reliability ● Usability ● Resource utilization, ● Maintainabil ● Installability, portability interoperabilit
System Integration	Development	● Detect defects in unit interfaces ● Reduce risk of dataflow and workflow failures in Production	● Functionality ● Data quality ● Unit interoperabilit ● Compatibilit ● Performance
Acceptance (User, Business, Operational)	Senior Business Partner (who requested the deliverable of the Project / Change	● Demonstrate the product works as expected in a real-life environment using real life operational scenarios ● Detect defects in user workflows ● Ensure key business acceptance criteria are met	● Functionality context of nor usage ● Operational Processes ● Pre-agreed acceptance cri

Regression testing

The purpose of regression testing is to verify that one fix or change does not inadvertently impact other areas in that program or in any other interface.

When considering any form of testing it is strongly recommended that regression testing is always considered when planning the scope of the test level. It is the intention of these guidelines to put regression testing at the heart of the testing design and execution process.

The following statements describe the approach to regression testing that should be undertaken:

1. Regression testing of a brand-new system is not possible initially. However once a new system has a defect and associated fix applied, a set of focused regression tests should be performed not only on the failed area but also areas surrounding that function.
2. If the Project / Change is amending or modifying an existing system, then a high degree of regression testing should be performed.
3. Regression testing should not be the last element of any testing that is required. If the requirement for regression is high, then as well as initially focusing on the new functionality during test execution, a focused set of regression tests should be planned.
4. Where possible, a set of regression tests should be designed and maintained for each mission critical system. The regression tests should each be prioritized using the following weightings:

    - High – Any regression tests assigned to this category are required to be executed as early on in the testing window as possible. This classification is used where functionality being tested is critical and if it does not perform as required would have a severe impact.
    - Medium - Any regression tests assigned to this category are required to be executed only after the High category tests. The functionality being tested is important and if failed would have a high but not severe impact
    - Low - Any regression tests assigned to this category mean they must be run only after the Medium category tests. These are effectively ‘nice to have’ tests. The functionality being tested is not critical important and if failed would only impact a small volume of users

---

Revision #2
Created 7 October 2025 19:47:09 by RISA
Updated 7 October 2025 20:00:59 by RISA