Test Planning and Estimation

1. Purpose and Content of a Test Plan

Test plan outlines test activities for development and maintenance projects. It is influenced by the test policy and test strategy, the development lifecycles and methods being used, the scope of testing, objectives, risks, constraints, criticality, testability and recources.

Test planning is continuous activity and is performed throughout the product's lifecycle. Feedback from test activities should be used to recognize changging risks so that planning can be adjusted.

Planning is documented in a master test plan and in separate test plans for test levels (system testing, acceptance testing, etc.) or for separate test types (usability testing, performance testing, etc.)

Test planning activities may include the following:

- Determining the scope, objectives, and risks of testing
- Defining the overall approach of testing
- Integrating and coordinating the test activities into the software lifecycle activities
- Deciding about what to test, the recources required to perform the various test activities, and how test activities will be carried out
- Scheduling of other steps of test process: test analysis, design, implementation, execution, and evaluation activities (on particular dates in sequential development or in the context of each iterations in iterative development)
- Selecting metrics for test monitoring and control
- Budgeting for the test activities
- Determining the level of detail and structure for test documentation (for instance, by providing templates or example documents)

The content of test plan vary, and can extend beyond the topics identified above.

2.1. Test Strategy

Test strategy provides a generalized description of the test process, usually at the product or organization level.

Common types of test strategies:

*** Analytical:** based on an analysis of requirement, risks and other factors. Risk-based testing is an example of an analytical approach.

*** Model-Based:** tests are designed based on some model or some required aspect of the product (function, business process, internal structure, or non-functional characteristics).

*** Methodical:** relies on making systematic use of some predefined set of tests or test conditions, such as a taxonomy of common or likely types of failures, a list of important quality characteristics, or company standards for products.

*** Process-compliant** (standard compliant)**:** involves analyzing, designing, and implementing tests based on external rules and standards.

*** Directed** (consultative)**:** driven primarily by the advice, guidance, or instructions of stakeholders, business domain experts, or technology experts who may be outside the test team or outside the organization itself.

*** Regression-averse:** Motivated by a desire to avoid regression of existing capabilities. Includes reuse of existing testware, extensive automation of regression tests, and standard test suites.

*** Reactive:** testing is reactive to the component or system being tested, and the events occuring during test execution. Tests are designed and implemented, and may immediately be executed in response to knowledge gained from prior test results. Exploratory testing is a common technique employed in reactive strategies.

Test strategy is often created by combining several of these types of test strategies. For instance, risk-based testing (analytical strategy) can be combined with exploratory testing (reactive strategy). They compliment each other and may achieve more effective testing when used together.

2.2. Test Approach

Test approach tailors the test strategy for a particular project or release. It is the starting point for selecting the test techniques, test levels, and test types, and for defining the entry criteria and exit criteria (definition of ready and definition of done in Agile development).

The tailoring of the strategy is based on decisions made in relation to the complexity and goals of the project, the type of product being developed, and product risk analysis. The selected approach depends on the context and may consider factors such as:

- risks
- safety
- resources and skills
- technology
- the nature of the system
- test objectives
- regulations

3.1. Entry Criteria (definition of ready)

Criteria which define when a given test activity should start helps in effective control over the quality of the software, and of testing.

Entry criteria define the preconditions for undertaking a given test activity. It entry criteria are not met, it is likely that the activity will prove more difficult, more time-consuming, more costly, and more risky. Entry criteria should be defined for each test level and test type, and will differ based on the test objectives.

Typical entry criteria include availability of the following:

- testable requirements, user stories, and/or models
- test items that have met the exit criteria for any prior test levels
- test environment
- necessary test tools
- test data and other necessary resources

3.2. Exit Criteria (definition of done)

Criteria which define when a given test activity is complete helps in effective control over the quality of the software, and of testing.

Exit criteria define what conditions must be achieved in order to declare a test level or a set of tests completed. Exit criteria should be defined for each test level and test type, and will differ based on the test objectives.

Typical exit criteria include:

- planned tests have been executed
- a defined level of coverage of requirements, user stories, acceptance criteria, risks, code has been achieved
- the number of unresolved defects is within an agreed limit
- the number of estimated remaining defects is sufficiently low
- the evaluated levels of reliability, performance efficiency, usability, security, and other relevant quality characteristics are sufficient

Even without exit criteria being satisfied, it is common for test activities to be curtailed due to budget being expended, the scheduled time being completed, and/or pressure to bring the product to market. It can be acceptable if the project stakeholders and business owners have reviewed and accepted the risk to go live without further testing.

4. Test Execution Schedule

Test cases and test procedures are produced → assembled into test suites → test suites are arranged in a test execution schedule

Test execution schedule defines the order in which test suites are to be run. It should take into account the following factors:

- prioritization
- dependencies
- confirmation tests
- regression tests
- the most efficient sequence for executing the tests.

Ideally, test cases would be ordered to run based on their **priority levels**, usually by executing the test cases with the highest priority first.

However, this practice may not work if the test cases have **dependencies** or the feature being tested have dependencies. If a test case with a higher priority is dependent on a test case with a lower priority, the lower priority test case must be executed first. If there are dependencies across test cases, they must be ordered appropriately regardless of their relative priorities.

Confirmation and regression tests must be prioritized based on the importance of rapid feedback on changes, but here again dependencies may apply.

5. Factors Influencing the Test Effort

Test effort estimation involves predicting the amount of test-related work needed for meeting the objectives of the testing for a particular project, release, or iteration.

Factors may include:

*** Product characteristics:**

- the risks associated with the product
- the quality of the test basis
- the size of the product
- the complexity of the product domain
- the requirements for quality characteristics (security, reliability, etc.)
- the required level of detail for test documentation
- requirements for legal and regulatory compliance

*** Development process characteristics:**

- the stability and maturity of the organization
- the development model in use
- the test approach
- the tools used
- the test process
- time pressure

*** People characteristics:**

- the skills and experience of the people involved, especially with domain knowledge
- team cohesion and leadership

*** Test results:**

- the number and severity of defects found
- the amount of rework required

6. Test Estimation Techniques

There are number of estimation techniques used to determine the effort required for adequate testing.

The most commonly used techniques are:

*** Metric-based technique:**

Estimating the test effort based on metrics of former similar projects, or based on typical values.

Example in Agile development: burndown charts (effort remaining is being captured and reported, and used to feed into the team's velocity to determine the amount of work the team can do in the next iteration).

Example in seqential development: defect removal models (volumes of defects and time to remove them are captured and reported, which then provides a basis for estimating future projects of a similar nature).

*** Expert-based technique:**

Estimating the test effort based on the experience of the owners of the testing tasks or by experts.

Example in Agile development: planning pocker (team members are estimating the effort to deliver a feature based on their experience).

Example in sequential development: Wideband Delphi estimation technique (a group of experts provides estimates based on their experience and consensus).