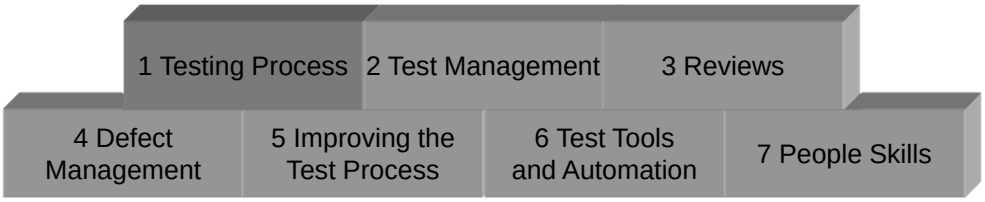
ISTQB Advanced Level
Test Manager

Testing Process

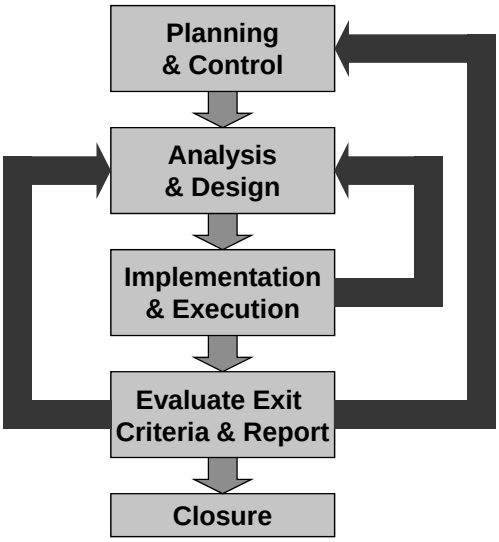


1.1

Contents

1.2

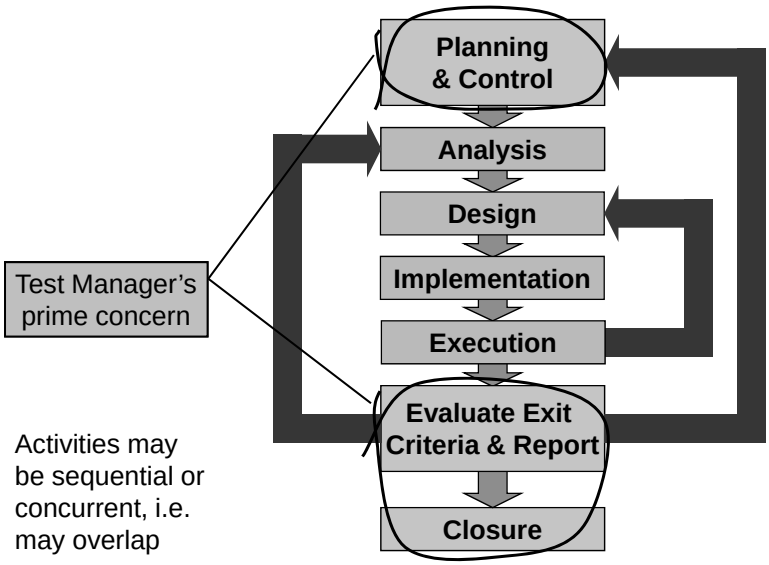
The fundamental test process (Foundation Level)



Activities may be sequential or concurrent, i.e. may overlap

1.3

The fundamental test process (Advanced Level)



Activities may be sequential or concurrent, i.e. may overlap

1.4

Contents

1.5

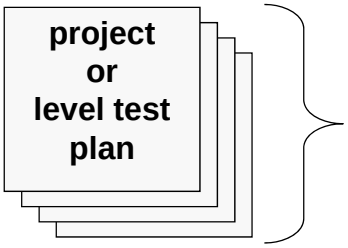
Test planning purpose

- identify activities and resources to
  - verify the testing mission
  - meet test objectives
- in practical terms:
  - implement the test strategy
    - i.e. make an operational plan
  - communicate the planned testing to the stakeholders
- test mission
  - high level/abstract statements describing overall purpose of testing
    - e.g. ‘find defects’, ‘keep customers happy’
- test objectives
  - measureable activities
    - e.g. ‘find 98% of highest severity defects’

1.6

Test planning implements the test strategy

- e.g. risk based test strategy could influence the test plan by
  - identifying product risks and how testing can help
  - highlighting additional static testing (reviews) of poor quality documentation needed as a test basis
  - relative priorities of the test activities

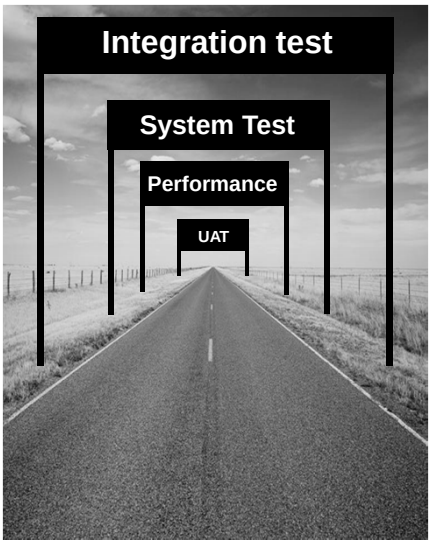


- the test plan prescribes the scope, approach, resources and schedule of testing activities
- it identifies scope (what should and should not be tested)
- it establishes how the strategy applies to the software under test

1.7

Test planning – an overview

Should begin as early as possible
- tailored to the organisation



Planning identifies methods for
- gathering and tracking metrics

Metrics used to
- guide the project
- determine adherence to plan
- assess whether objectives are met

1.8

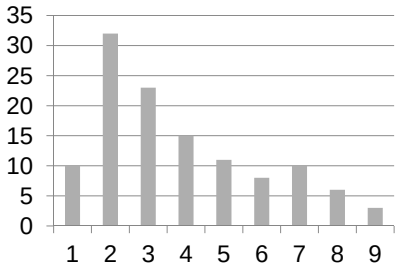
Test planning – additional considerations

- need to understand relationships between test basis, test conditions and tests
  - often many-to-many relationships
- specification of what features are within scope and explicit identification of what is out of scope
  - helps avoid ‘misunderstandings’
- identification of test environment requirements
  - work with project architects
  - verify resource availability
  - understand costs / timescales
- identify all external dependencies
  - people / organisations providing resources / services
  - agree any Service Level Agreements (SLA)

1.9

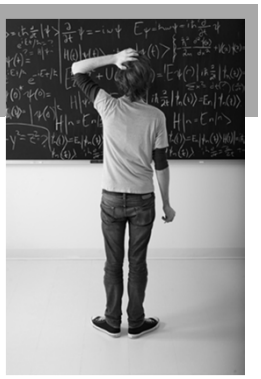
Test monitoring

- purpose
  - provide timely and accurate information
  - to support informed decisions
- applicable throughout the project
  - starts with initial test schedule
  - includes all testing activities
- metrics to monitor may include
  - risk information
  - defect information
  - tests designed, built, executed, passed / failed
  - test coverage
  - effort

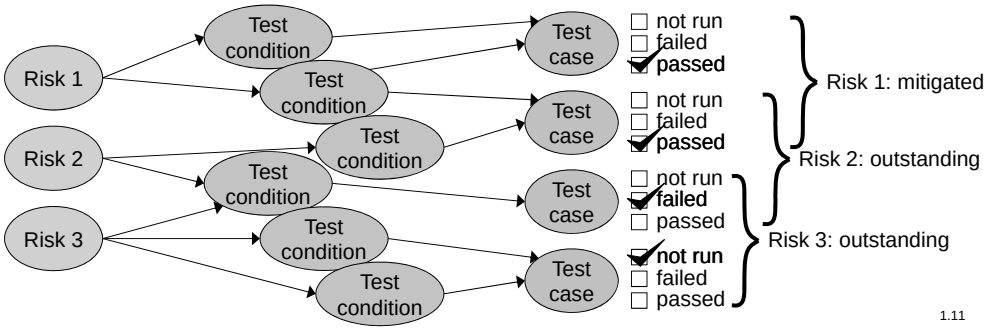


1.10

Relating test progress to the test basis



- for effective monitoring
  - relate status of test work products and activities to the test basis and test objectives
    - in a manner that is understandable and relevant to the stakeholders
    - for example:



1.11

Reporting to stakeholders

- test reports must keep the target audience in mind
  - sometimes the detailed measures and targets that stakeholders want monitored will not relate directly to system functionality or specification
    - particularly where there is little or no formal documentation
  - e.g. stakeholder may require coverage against operational business cycle
- involve business stakeholders early to help define suitable measures and targets
  - provide better control
  - drive and influence testing
    - e.g. structure test design and execution to support accurate monitoring against stakeholder measures

1.12

Test control

- purpose
  - to guide testing to fulfil mission and objectives
- an on-going activity
  - comparing actual progress with the plan
- identify deviations
  - behind / ahead of schedule
  - fewer / more defects
  - early / late availability
- determine appropriate corrective actions
  - earlier the better
    - less severe controlling actions
  - e.g. reallocate resources, re-prioritise

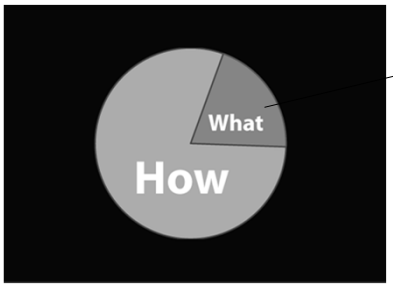
1.13

Contents

1.14

Test analysis versus test design

Test analysis defines WHAT is to be tested

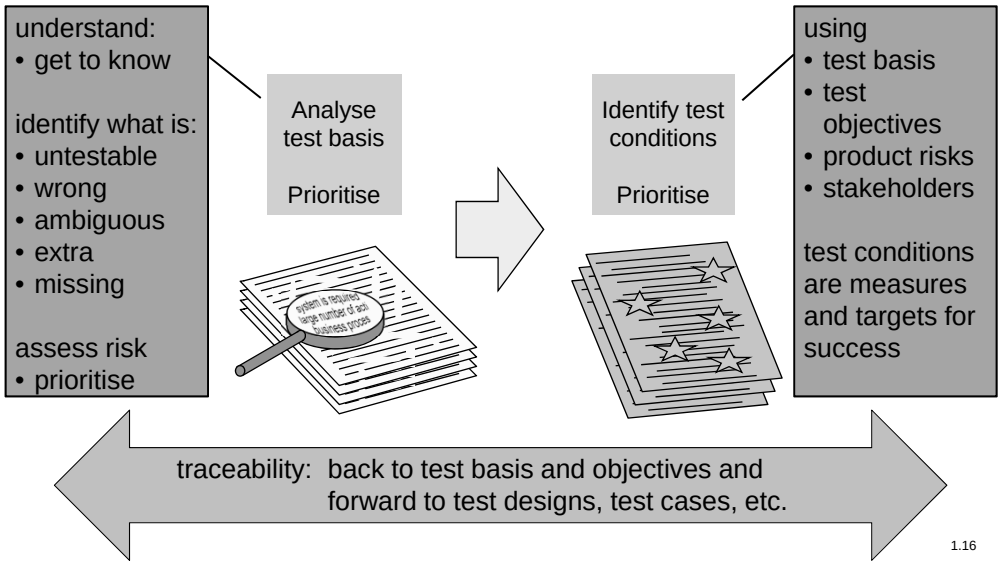


If the 'what' is wrong, much of the 'how' effort may be wasted!

Test design defines HOW it is to be tested

1.15

Test analysis



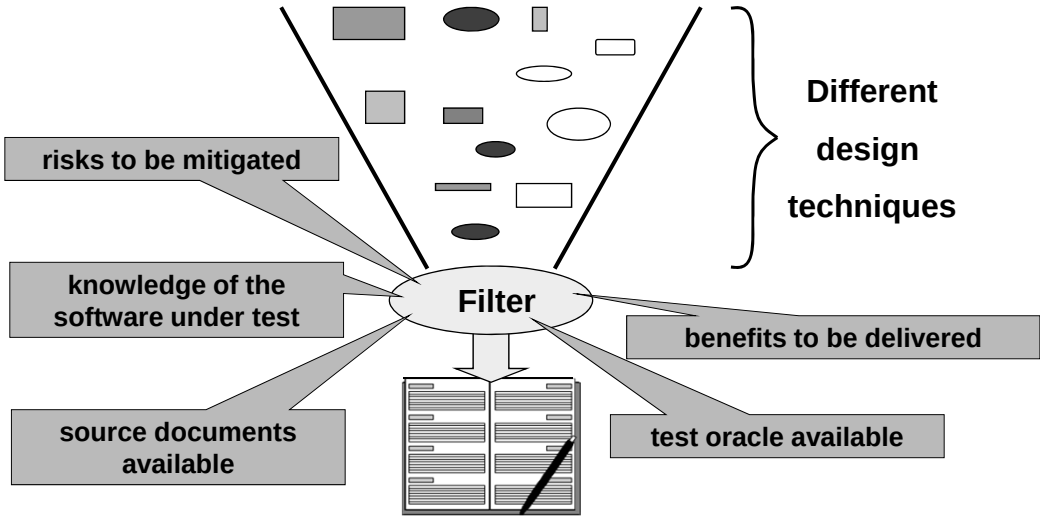
1.16

Test basis

- information available describing the software
  - typically one or more specifications
    - e.g. requirement spec., use cases, system spec., functional spec., business process spec.
  - but could also be
    - contracts, letters, emails, user feedback / requests
- sometimes, no documentation exists at all
  - must rely on own knowledge / understanding
    - and that of others – ask questions!
- level of testing may decide test basis
  - user acceptance testing ➡ requirements spec.
  - component testing ➡ low-level design spec.
- should detail quality characteristics as well as behaviour

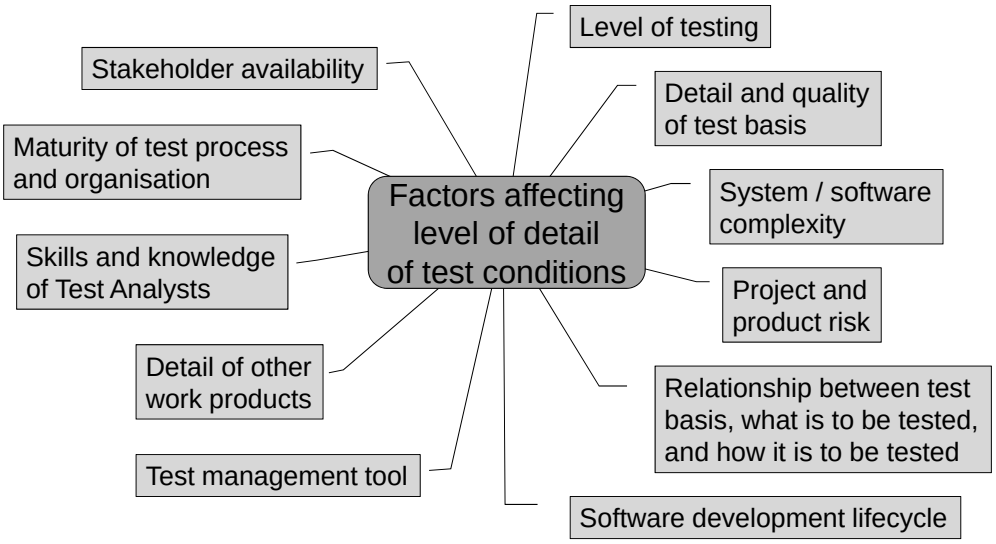
1.17

Choosing test design techniques



1.18

Factors affecting the level of detail



1.19

Detailed test conditions: pros and cons

Advantages

- better and more detailed monitoring and control
- defect prevention
- better relates test work products to stakeholders
- helps influence and direct activities
- enables more efficient coverage
- clearer horizontal traceability

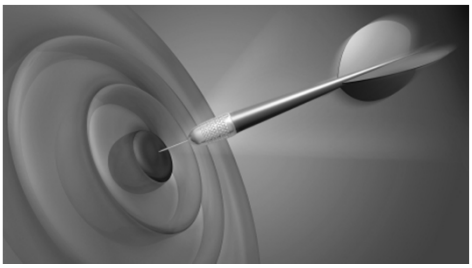
Disadvantages

- time consuming
- maintenance becomes more difficult
- level of formality needs more control

1.20

Detailed test conditions: when most appropriate

- are specially effective when
  - lightweight test design documentation methods (e.g. checklists) are in use
    - reduces need for further test documentation (i.e. test cases)
  - formal requirements or development work products are absent
    - build the knowledge into test conditions
  - project is large-scale, complex or high risk
    - being more thorough



1.21

Less detailed test conditions

- less detailed test conditions are appropriate when the test basis can be related easily and directly to test design work products
  - component level testing
  - less complex projects with simple hierarchical relationships
  - acceptance testing

1.22

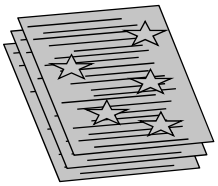
Contents

1.23

Test Design

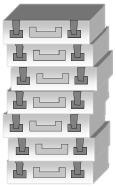
starts when test conditions are identified and sufficient information available

From
test conditions



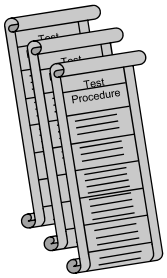
Design & specify test cases

to
test cases



Develop test procedures

(and possibly)
test procedures

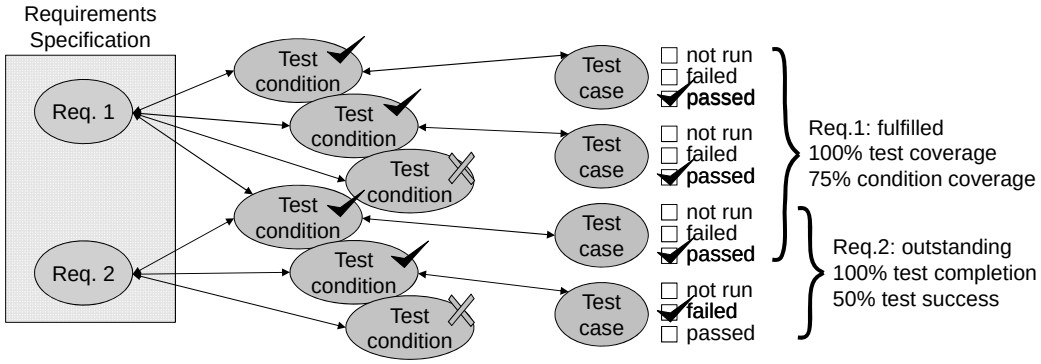


can be high-level (logical) or low-level (concrete)

1.24

Relating test cases to test basis

degree of traceability depends on the approach to test monitoring and control



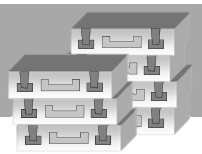
1.25

Separate versus integrated activity

- for higher levels of testing
  - e.g. system, acceptance testing
  - test design more likely a separate activity following test analysis
- for lower levels of testing
  - e.g. component, component integration testing
  - it is more likely that test analysis and design will be an integrated activity

1.26

Concrete and logical test cases



concrete (low-level) test cases

- content
  - all specific information and procedures required
    - to execute and verify tests
- useful when
  - requirements well defined
  - testing staff less experienced
  - external verification required
- good points
  - reproducibility
- but
  - maintenance effort significant
  - limits tester ingenuity & coverage

logical (high-level) test cases

- content
  - guidelines for what's to be tested
    - testers vary data/procedures
- useful when
  - requirements not well defined
  - testing staff experienced
    - with testing and the product
  - formal documents not required
- good points
  - may provide better coverage
  - can be defined earlier
- but
  - loss of reproducibility

1.27

Contents

1.28

Test implementation

- tests are organised and prioritized by Test Analysts
- implementation of concrete test cases, test procedures and test data
  - IEEE 829 compliance:
    - inputs and expected results in test case specifications
    - test steps in procedures

- creation of stored test data
  - files and databases
- final checks
  - everything ready for test execution?
    - environments, people, reviewed test cases, code, etc.
  - explicit entry and exit criteria
- develop description of test environment and data

1.29

Level of detail required

- influenced by the detail of other test work products
  - e.g., test conditions and test cases
- tests may provide detailed steps necessary to execute a test
  - e.g. where tests are to be archived for long-term re-use in regression testing
    - to ensure reliable, consistent execution regardless of the tester executing the test.
- if regulatory rules apply, tests should provide evidence of compliance to applicable standards

1.30

Test execution schedule

- tasks for the Test Manager during test implementation:
  - check for constraints
    - including risks and priorities
  - check for tests that require
    - running in a particular order
    - running on specific equipment
  - check and note dependencies on
    - test environment
    - test data

1.31

Early test implementation: pros and cons

Advantages

- test documentation ready for execution
  - provides worked examples of expected behaviours
- verification of concrete tests by domain experts may be easier
  - further early defect detection
- preparation all done
  - e.g. test data

Disadvantages

- dramatic code changes in Agile between iterations
  - may make earlier work on implementation obsolete
- maintenance of test documentation
  - particularly with iterative / incremental lifecycles and poorly managed sequential
- extra re-work if changes occur

1.32

Contents

1.33

Prerequisites for test execution

- test execution starts when:
  - test object is delivered, and
  - entry criteria are satisfied
- tests should have been designed / defined
- tools should be in place
  - test management
  - defect tracking
  - test execution (if applicable)
- processes in place
  - test results tracking
  - metrics tracking
- data to be tracked should be understood by all team members
- standards for test logging and defect reporting available

1.34

Scripted and unscripted testing

- tests should be executed as documented
  - that's why they're documented!
- consider allowing time to:
  - cover additional interesting test scenarios and behaviours
- for failures detected during unscripted testing
  - describe the variations from the written test case that are necessary to reproduce the failure.
- when following a test strategy that is at least in part reactive
  - reserve time for test sessions using experience-based and defect-based techniques.
- automated tests will follow their defined instructions without deviation

1.35

Test Manager responsibilities during test execution

- monitor progress
  - compare with the test plan
  - achieved by
    - using traceability between
      - test objectives
      - test basis
      - test conditions
      - test cases / procedures
      - test results
- initiate control actions
  - to achieve the mission and objectives

1.36

Contents

1.37

Evaluating exit criteria

- assess exit criteria for each level as defined in the test plan
- generate more tests if required or change the exit criteria
- example information to collect throughout testing
  - number of test planned versus executed
  - number of tests passed versus failed
  - total defects raised, by severity and outstanding
  - number of changes (change requests) raised
  - planned expenditure versus actual expenditure
  - planned elapsed time versus actual elapsed time
  - risks outstanding
  - percentage of test time lost due to blocking events
  - total test time planned against effective test time carried out

1.38

Reporting

- produce test progress reports for stakeholders
  - document progress
    - raise any exceptions
    - must be meaningful to recipient
    - use of dashboards encouraged

- Test Manager should ensure that members of the test team are providing the information required in an accurate and timely manner so as to facilitate effective evaluation and reporting

1.39

Contents

1.40

Test closure

- test closure activities fall into four main groups
  - checking completion
    - ensuring that all test work is indeed concluded (consolidation of test activities at the end of testing phase)

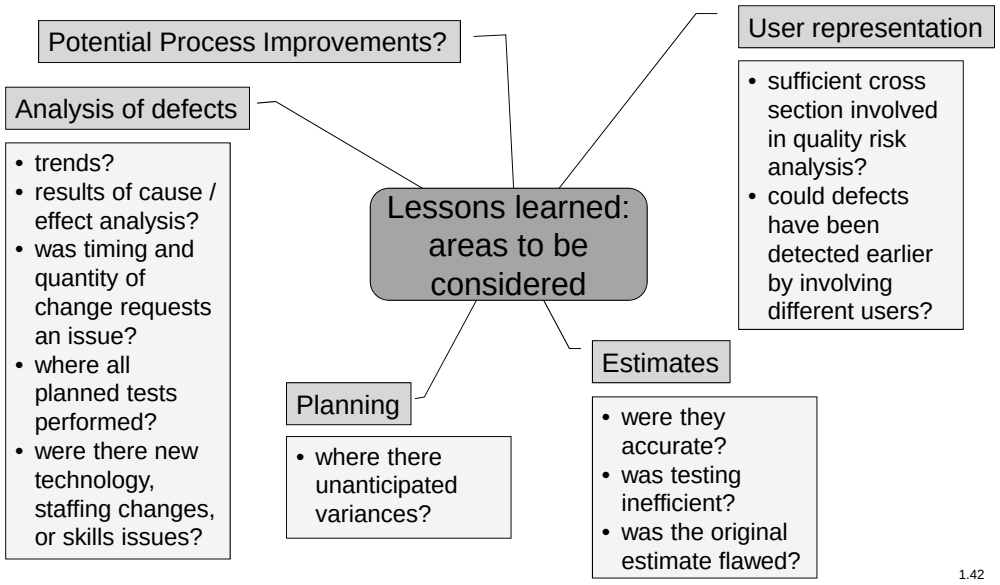
  - delivery
    - delivering valuable work products to those needing them
      - e.g. deferred defects to those using and supporting the system
      - tests and test environments to those maintaining the system

  - retrospectives
    - performing or participating in retrospective meetings (lessons learned – reinforce the good, mitigate the bad)

  - archiving
    - results, logs, reports, and other documents and work products in the configuration management system

1.41

Lessons learned



1.42

Session 1

Testing Process

Terms
exit criteria, test case, test closure, test condition, test control, test design, test execution, test implementation, test level, test log, test planning, test procedure, test script, test summary report.

From the ISTQB Glossary

exit criteria: The set of generic and specific conditions, agreed upon with the stakeholders for permitting a process to be officially completed. The purpose of exit criteria is to prevent a task from being considered completed when there are still outstanding parts of the task which have not been finished. Exit criteria are used to report against and to plan when to stop testing. [After Gilb and Graham]

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. [After IEEE 610]

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. See also test process.

test condition: 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.

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. See also test management.

test design: (1) See test design specification. (2) The process of transforming general testing objectives into tangible test conditions and test cases.

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

test implementation: The process of developing and prioritizing test procedures, creating test data and, optionally, preparing test harnesses and writing automated test scripts.

test level: A group of test activities that are organized and managed together. A test level is linked to the responsibilities in a project. Examples of test levels are component test, integration test, system test and acceptance test. [After TMap]

test log: A chronological record of relevant details about the execution of tests. [IEEE 829]

test planning: The activity of establishing or updating a test plan.

test procedure specification: A document specifying a sequence of actions for the execution of a test. Also known as test script or manual test script. [After IEEE 829] See also test specification.

test procedure: See test procedure specification.

test script: Commonly used to refer to a test procedure specification, especially an automated one.

test summary report: A document summarizing testing activities and results. It also contains an evaluation of the corresponding test items against exit criteria. [After IEEE 829]

1.1 Introduction

Test and development processes

Testing is an important part of software development and, like development, testing is not a single process or activity but a set of processes and activities. These development and testing processes are mixed together throughout the software lifecycle. Without the development processes, testing processes mean nothing, as the testing processes 'test' what the development processes produce. Without the development processes there would be nothing for testing to test! Testing processes support the development processes. Without the testing processes the development processes will not succeed

The testing processes are interconnected with the development processes and with project support processes. Some examples of development and supporting processes are given here.

Example development processes

- Requirements engineering
- Design
- Coding
- Software maintenance

Example project supporting processes

- Quality assurance
- Project management
- Configuration management
- Change management
- Technical writing (production of technical documentation)
- Technical support (environment and tool support)
- Requirements management

Testing is a part of the quality assurance process and therefore testing is also a supporting process. Testware artefacts, the materials created by testing such as test plans, test specifications etc. can and should themselves be subject to quality assurance and testing. This makes testing recursive - meaning that testing interfaces with itself.

Quality assurance and project management are supporting processes that typically are carried out throughout the software lifecycle (we'll be discussing software lifecycles next). Similarly, configuration and change management are also important tasks that support software testing throughout the lifecycle. Without configuration management, concurrent versions of the software and testware artefacts may be lost or mismanaged. Without proper change management, the impact of changes on the system may not be evaluated properly.

The software lifecycle

The software lifecycle is the period of time that begins when a software product is conceived and ends when the software is no longer available for use. Typically there are several phases within the software lifecycle but, at a high level, we can see the development phase in which

the product is designed, built and tested, and the operation and maintenance phase in which the product is used and maintained.

Software development lifecycle

The way in which development and testing processes are structured form the development phase of the overall software lifecycle, and this is called the software development lifecycle. There are many different ways of structuring the development and testing processes but there are some common structures and these are described by software development lifecycle models that each fall into one of the following three categories.

- Sequential e.g. Waterfall model, V-model and W-model
- Iterative e.g. Rapid Application Development (RAD) and Spiral model
- Incremental e.g. Evolutionary and Agile methods

The fundamental test process

The ISTQB® Foundation Level syllabus describes a fundamental test process that includes the following activities:

- Planning and control
- Analysis and design
- Implementation and execution
- Evaluating exit criteria and reporting
- Test closure activities

The Foundation Level syllabus states that although logically sequential, the activities in the process may overlap or take place concurrently. Tailoring these main activities within the context of the system and the project is usually required.

For the Advanced Level syllabi some of these activities are considered separately in order to provide additional refinement and optimization of the processes, to better fit with the software development lifecycle, and to facilitate effective test monitoring and control. The activities are now considered as follows:

- Planning, monitoring and control
- Analysis
- Design
- Implementation
- Execution
- Evaluating exit criteria and reporting
- Test closure activities

These activities can be implemented sequentially or some can be implemented in parallel, e.g., design could be performed in parallel with implementation (e.g., exploratory testing).

The activities of test planning, monitoring and control, evaluating exit criteria and reporting and test closure are all of particular concern to the Test Manager. Although Test Analysts and Technical Test Analysts focus on the activities of analysis, design, implementation and execution, they should also be involved in the other fundamental test process activities, supporting the Test Manager with their skills and knowledge and providing accurate and timely information.

It is important for Test Managers to consider the different software development lifecycles as well as the type of system being tested, as these factors can influence the approach to testing.

1.2 Test Planning, Monitoring and Control

Learning Objective

TM-1.2.1	K4	Analyse the test needs for a system in order to plan test activities and work products that will achieve the test objectives.

From the ISTQB Glossary

level test plan: A test plan that typically addresses one test level. *See also test plan.*

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. [After IEEE 610]

test condition: 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.

test design: (1) See test design specification. (2) The process of transforming general testing objectives into tangible test conditions and test cases.

test design specification: A document specifying the test conditions (coverage items) for a test item, the detailed test approach and identifying the associated high level test cases. [After IEEE 829] *See also test specification.*

test level: A group of test activities that are organized and managed together. A test level is linked to the responsibilities in a project. Examples of test levels are component test, integration test, system test and acceptance test. [After TMap]

test management: The planning, estimating, monitoring and control of test activities, typically carried out by a test manager.

test mission: The purpose of testing for an organization, often documented as part of the test policy. *See also test policy.*

test objective: A reason or purpose for designing and executing a test.

test plan: A document describing the scope, approach, resources and schedule of intended test activities. It identifies amongst others test items, the features to be tested, the testing tasks, who will do each task, degree of tester independence, the test environment, the test design techniques and entry and exit criteria to be used, and the rationale for their choice, and any risks requiring contingency planning. It is a record of the test planning process.

test planning: The activity of establishing or updating a test plan.

test policy: A high level document describing the principles, approach and major objectives of the organization regarding testing.

test process: The fundamental test process comprises test planning and control, test analysis and design, test implementation and execution, evaluating exit criteria and reporting, and test closure activities.

test specification: A document that consists of a test design specification, test case specification and/or test procedure specification.

test strategy: A high-level description of the test levels to be performed and the testing within those levels for an organization or programme (one or more projects).

This section focuses on the processes of planning, monitoring and controlling testing. As was discussed at the Foundation Level, these activities are test management roles.

1.2.1 Test Planning

Purpose

The purpose of test planning is to identify the activities and resources (both human and material) required to:

- verify the mission of the testing (as given in the test policy/test strategy)
- meet the test objectives (goals/purpose)

Note that the glossary definition of the ‘test mission’ states that this *is often documented* within the test policy, while the syllabus states that it *is documented* in the test strategy. Since the names and content of these documents vary across different organisations, we need to appreciate that variations will exist. Where test policy and test strategy documents do not exist, the test mission may be stated in the master test plan.

Mission statements are a high level or abstract statement concerning the overall purpose of testing (such as ‘find defects, ‘keep customers happy’, ‘manage risks’). The test objectives

are more specific targets such as ‘find 98% of the defects with the highest impact on customers or the business’, ‘achieve customer satisfaction scores greater than 75%’, and ‘run tests to mitigate the highest level of product quality risks’.

Test objectives are measurable activities that need to happen in order to meet the mission statement. Therefore test objectives are more likely to be quantified in some way (making it possible to measure progress toward them and to confirm their achievement). There can be different test objectives at different levels from organisation-wide objectives, project-specific objectives and test level-specific objectives.

The syllabus states that the test objectives are ‘defined in the test strategy’ but also implies that test objectives for a particular level of testing may be defined in the associated level test plan (or master test plan). The glossary definition of ‘test policy’ includes ‘major objectives’ in its content.

Test planning is the means by which we transform the test strategy into an operational plan. For example, a risk-based test strategy will focus test effort on the highest risk levels and this may be implemented by planning a large proportion of the test effort, and possibly use of more thorough testing techniques, on the highest risk level areas of the software. The resources required will include staff, time, facilities and tools. We will need to put in place an organisational structure if this doesn’t already exist. If it does, we may need to adapt it to better meet the demands of the new test project.

The work product produced by test planning is, of course, the test plan. The test plan document is the primarily means of communicating the planned testing work to the stakeholders.

When

Generally, test planning should begin as soon as possible at the start of the project. Where the project is large, the initial test planning should result in a master test plan that outlines the testing activities across the entire project. Subsequently more detailed test planning may be performed for each level of testing, resulting in a number of level test plans.

The benefits of early test planning include:

- early visibility of potential problems
- possibility of providing input to the development plan
- time to do planning well
- time to involve stakeholders

Test planning is not a single activity that is only done once; it is a continuous activity that goes on throughout the project. This applies to both the master test plan and the level test plans. After the initial planning has been done and the other testing activities are progressing, information about what is happening may justify changes to the master and/or level test plans to keep them in step with actual progress. Such changes are inevitable because we do not know (and cannot know) in detail what may happen, how long some activities will take, what problems will be found, etc. Consequently the test plans are based on many assumptions, several of which are likely to be proven inaccurate if not completely wrong!

Approach

In addition, the test planning stage is where the approach to testing is clearly defined by the Test Manager, including

- which test levels will be employed
- the goals and objectives of each level, and
- what test techniques will be used at each level of testing.

For example, in risk-based testing of certain avionics systems, a risk assessment would be used to determine what level of code coverage is required and thereby which testing techniques should be used.

Level test plans

The detailed level test planning specific to each test level should begin at the start of the test process for that level and carry on throughout the project until completion of the closure activities for that level. Detailed planning for a test level can start as soon as the documentation on which the testing is to be based has reached a reasonable degree of completeness (i.e. it need not be in its final version, but what does exist should be fairly stable). This documentation may be draft but should have sufficient information in it to enable the planning to progress. The level test planning should be based on the test strategy and be consistent with the master test plan and the overall project plan.

The structure of the test plans should be tailored to the organisation. Many organisations have templates that help to ensure consistency and completeness. The content of test plans is discussed in more detail in Section 2.4 “Test Documentation and Other Work Products”.

Gathering and tracking metrics

As stated earlier, a primary purpose of test planning is to identify the activities and resources required to meet the test mission and objectives. However, test planning also includes identifying the methods for gathering and tracking the metrics that will be used to guide the project, determine adherence to the plan and assess achievement of the objectives. By determining useful metrics during the planning stages, tools can be selected, training can be scheduled and documentation guidelines can be established. This will help to ensure that metric collection and analysis is as efficient as it needs to be and that everyone involved will know how to contribute to effective and consistent use of metrics.

Determining the testing tasks

The strategy (or strategies) selected for the testing project help to determine the tasks that should occur during planning and later stages. For example, when using the risk-based testing strategy (see Chapter 2), risk analysis is used to guide the test planning process regarding the mitigating activities required to reduce the identified product risks and to help with contingency planning. If a likely risk of serious potential defects related to security is identified, a significant amount of effort should be spent developing and executing security tests. Likewise, if it is identified that serious defects are usually found in the design specification, the test planning process could schedule additional static testing (reviews) of the design specification.

Priorities

Risk information may also be used to determine the priorities of the various testing activities. For example

- where system performance poses a high risk, performance testing may be conducted as soon as integrated code is available.
- if a reactive strategy is to be employed, planning for the creation of test charters and tools for dynamic testing approaches such as exploratory testing may be needed; again, test planning should target these at the areas of highest risk.

Test basis to test cases

Complex relationships may exist between the test basis (e.g., specific requirements or risks), test conditions and the tests that cover them. Many-to-many relationships often exist between these work products. These need to be understood to enable effective implementation of test planning, monitoring and control. Tool decisions may also depend on the understanding of the relationships between the work products.

Work product relationships

Relationships may also exist between work products produced by the development team and the testing team. For example, the traceability matrix may need to track the relationships between the detailed design specification elements from the system designers, the business requirements from the business analysts, and the test work products defined by the testing

team. Depending on the levels of formality and documentation appropriate to the project, each feature that is within scope may be associated with a corresponding test design specification.

If low-level test cases are to be designed and used, there may be a requirement defined in the planning stages that the detailed design documents from the development team must be approved before test case creation can start.

When following an Agile lifecycle, informal transfer-of-information sessions may be used to convey information between team members. If this reveals something that should be tested but isn't documented, that thing should be added to the traceability matrix.

Scope

The test plan may list the specific features of the software that are within its scope (based on risk analysis, if appropriate), as well as explicitly identifying features that are not within its scope. Doing this can help avoid misunderstandings between testing and the other stakeholders. It is all too easy for someone to assume that a particular feature will be tested and may know a good reason why it should be. It can happen that the people involved with test planning are not made aware of this reason and so decide not to test the feature (or not to test it as thoroughly as they should). Test planning provides the opportunity for all stakeholders to check that the planned testing matches their expectations.

Test environment

There may be a requirement at this stage for the Test Manager to work with the project architects to define the initial test environment. This must include all necessary hardware, software (including operating systems and other elements such as browsers, database systems), test tools including monitoring tools, office space and equipment.

Early specification of the test environment is important to verify availability of the resources required, to ensure that the people who will configure the environment are committed to do so, and to understand costs, delivery timescales, and the work required to make the test environment ready for test execution.

External dependencies

All external dependencies and associated Service Level Agreements (SLAs) should be identified and, if required, initial contact should be made with those people or organisations that provide resources, products or services. Examples of dependencies are resource requests to outside groups, dependencies on other projects (if working within a program), external vendors or development partners, the deployment team, and database administrators.

1.2.2 Test Monitoring and Control

From the ISTQB Glossary

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. See also test management.

test implementation: The process of developing and prioritizing test procedures, creating test data and, optionally, preparing test harnesses and writing automated test scripts.

Test monitoring

Test monitoring is concerned with all the testing activities that start to occur as soon as the initial test planning has produced a schedule for them. Monitoring the testing is as crucial as the testing itself. It is important to know both “where we are” and “where we are going”. As Watts Humphreys said “If you don’t know where you are – a map won’t help!” Monitoring is all about the supply of timely and accurate information on which we can make informed decisions. Having planned our testing, we need to compare what is actually happening and take any necessary controlling actions.

Schedule and monitoring framework

In order for a Test Manager to provide efficient test control, a testing schedule and a monitoring framework need to be established. Together these will enable tracking of test work products and resources against the plan. The framework should include the detailed measures and targets that are needed to relate the status of test work products and activities to the plan and to the test objectives. The latter relationship (status of work products and activities to the test objectives) is particularly important because one hundred per cent progress in terms of test schedule means that all the planned tests are completed but does not necessarily mean that the test object has been sufficiently tested, i.e. that the test objectives have been met.

To provide an indication of test thoroughness, test coverage measures can be used. Different measures are applicable at different test levels. For example, in component testing, code coverage metrics are typically used. At system and acceptance testing levels, requirement or feature coverage may be used.

For small and less complex projects, it may be relatively easy to relate test work products and activities to the plan and test objectives, but for larger / more complex projects more detailed objectives will need to be defined in order to achieve this.

Reporting progress

There are many ways in which we can report test progress using words, tables and/or graphs. When using graphs and tables we may need to provide an interpretation of the results, since people can jump to the wrong conclusions. A good graph is usually the best way to communicate high-level information quickly (ideal for giving high level managers a feel for the "big picture").

Most of the commercially available Test Management tools supply graphical monitoring as part of the tool set. This not only helps with test monitoring but also encourages a consistent approach throughout the organisation. These tools can also be used to give immediate feedback when requested, e.g. reports at the touch of a button.

Another benefit of using graphical representation is the fact that we can predict what is likely to happen by extrapolating the shape of the graph / curve. This helps when we need to make decisions about the future of the testing and project.

Mapping to test basis

Metrics that we may gather include those related to risk, defects, tests, test coverage, and effort. The choice of metrics is important because the wrong ones may mislead stakeholders or convey no meaning to them at all. It is necessary to relate the status of test work products and activities to the test basis in a manner that is understandable and relevant to the project and business stakeholders.

Defining targets and measuring progress based on test conditions and groups of test conditions can be used as a means to achieve this by relating other testing work products to the test basis via the test conditions. Properly configured traceability, including the ability to report on traceability status, makes the complex relationships that exist between development work products, the test basis, and the test work products more transparent and comprehensible.

Sometimes, the detailed measures and targets that stakeholders require to be monitored do not relate directly to system functionality or a specification, especially if there is little or no formal documentation. For example, a business stakeholder may be more interested in establishing coverage against an operational business cycle even though the specification is defined in terms of system functionality. For example, a project may be responsible for delivering just one part of a system of systems in which any single stage of a business cycle is supported by multiple systems, a business stakeholder may require coverage across multiple systems to be reported so it maps to the business cycle with which the business stakeholder is familiar.

Involvement of business stakeholders at an early stage in a project can help define these measures and targets, which not only can be used to help provide better control during the

project, but can also help to drive and influence the testing activities throughout the project. For example, stakeholder measures and targets may result in the structuring of test design and test implementation work products and/or test execution schedules to facilitate the accurate monitoring of testing progress against these measures. These targets also help to provide traceability for a specific test level and have the potential to help provide information traceability across different test levels.

Test control

Test control is an on-going activity. It involves comparing actual progress against the plan, to identify which activities are progressing in line with the plan and which ones are not. Any deviations from the plan will need to be addressed. Typically, the sooner this is done the less severe the corrective action needs to be. Deviations may be that progress is behind or ahead of the planned schedule, that significantly more or less than the expected quantity of defects have been found, or the early / late availability of some necessary resources.

Deviations from the plan are inevitable because, as we have said before, at the time of initial planning we do not know (and cannot know) in detail what may happen, how long some activities will take, what problems will be found, etc. Consequently the test plans are based on many assumptions, several of which are likely to be proven inaccurate if not completely wrong.

For example, we may discover that there are more defects in one part of the software than we had planned for. As each defect will take some time to report, track and re-test and this effort must be included in the schedule, we therefore have to estimate the number of defects that will be found. If this estimate proves to be very different from the number actually found we may need to allocate more resources (or fewer resources, if fewer defects than estimated are found) to testing that part of the software.

The purpose of test control is to guide the testing to fulfil the mission, strategies, and objectives. Control activities include reallocation of resources, re-prioritisation of activities and tests, and even changing the scope of the testing. All of these will require changes to the test plan. Exactly what controlling actions are taken will depend on what the test plan said and what has actually happened.

We must look at the effects of taking controlling action – the before and after pictures. For example, if we decided to allocate more time to the analysis and design of good quality test cases, in doing this we may find that the time spent on test design goes up but the amount of time in test execution and re-work goes down.

There are many different actions that we can take to control testing, but there is one area that cannot be affected and that is the number of defects in the software – they are already there by the time test execution starts. If testing finds more defects than expected, it is no use blaming testing (but this is what sometimes happens).

Further details of test control activities are covered in Session 2.

1.3 Test Analysis

Learning Objectives

TM-1.3.1	K3	Use traceability to check completeness and consistency of defined test conditions with respect to the test objectives, test strategy, and test plan.
TM-1.3.2	K2	Explain the factors that might affect the level of detail at which test conditions may be specified and the advantages and disadvantages for specifying test conditions at a detailed level.

From the ISTQB Glossary

exit criteria: The set of generic and specific conditions, agreed upon with the stakeholders for permitting a process to be officially completed. The purpose of exit criteria is to prevent a task from being considered completed when there are still outstanding parts of the task which have not been finished. Exit criteria are used to report against and to plan when to stop testing. [After Gilb and Graham]

Rather than consider test analysis and design together as described in the Foundation Level syllabus, the Advanced syllabi consider them as separate activities, albeit recognizing that they can be implemented as parallel, integrated, or iterative activities to facilitate the production of test design work products.

Test analysis is the name given to a set of activities that develop a set of test conditions from available information. The test conditions can then be used to test the software described by that information. This information is known as the test basis (it is the information on which the tests are based). Test analysis applies to every level of testing, from component testing through to acceptance testing.

During test planning, the scope of the testing project is defined. During test analysis, the Test Analyst uses this scope definition to:

- analyse the test basis; and
- identify the test conditions.

Analysing the test basis

The first goal for the test analyst is to become very familiar with the test basis: understanding all the information contained within it, and gaining a sense of what has not been specified (but probably should have been!). This is something that cannot be achieved by reading through the document once. Time should be taken to read the test basis at least two or three times, and possibly more. The aim of this is to understand the document as well as the author does. Being this thorough will almost certainly mean that the test analyst will find defects in the test basis such as insufficient detail, contradictory information; and ambiguous statements. It is good to find these defects where they exist – testing has begun!

Having gained a good understanding of the document it is then time to consider its testability. For each requirement or function description we should ask “How can I test this?” and “What are the risks associated with this?”. Sometimes it will be easy to see how a requirement can be tested, but not always. Often requirements (and function descriptions) are vague or lack details that are needed before any tests can be designed. For example, the use of the word “etcetera” (or the more common abbreviated form: “etc.”) signifies missing detail. Other words that do not belong in a specification include: “maybe”, “might”, “could”, “perhaps” or “should”. These and many other words suggest requirements that are not testable.

The problem of missing detail and vague statements will also affect the development of the system, but developers may, consciously or not, make assumptions about missing details or the real meaning of vague statements. Their assumptions will not always be correct and consequently the software they are producing will contain defects that the testing needs to find. By highlighting such defects in the test basis it should be possible to obtain answers that will give us a more correct understanding of the test basis and the ability to design good tests.

Assessing the risks associated with each requirement (or function description) is frequently an important task. Ideally this would already have been done and the results of the analysis (assignment of ranked risk levels to each requirement) incorporated into the test basis document. However, it is more often the case that such risk analysis has not been performed. Thus, it falls to the test analyst to undertake at least some basic risk analysis, the results of which can be used to focus the test effort. The requirements that have the highest risk levels should be given the most attention.

Another outcome from analysis of the test basis is the selection of test techniques. While the use of certain techniques may already be mandated or recommended by the test strategy or test plan (in the section on test approach), we should not miss the opportunity to identify additional techniques that may be well suited to this type of development. Our choice of techniques should also be influenced by the results of any risk analysis.

Before completing the analysis of the test basis we should consider priorities. Which parts of the test basis are the most important? Which parts must we work on first and which parts can be left until last? What proportions of our effort should be spent on each part? This may already be clear if risk analysis has been carried out but, even if it has, we may still need to prioritise amongst those items that have equal risk levels.

Identifying test conditions

Test analysis is the activity that defines “what” is to be tested, in the form of test conditions. Test conditions can be identified by analysis of the test basis, test objectives, and product risks. In some situations, where documentation may be old or non-existent, the test conditions may be identified by talking to relevant stakeholders (e.g., in workshops or during sprint planning).

Test conditions can be viewed as the detailed measures and targets for success (e.g., as part of the exit criteria) and should be traceable back to the test basis and defined strategic objectives, including test objectives and other project or stakeholder criteria for success. Test conditions should also be traceable forward to test designs and other test work products as those work products are created.

Test analysis for a given level of testing can be performed as soon as the basis for testing is established for that level. Formal test techniques (e.g. equivalence partitioning, boundary value analysis, decision tables, etc.) and other general analytical techniques (e.g. analytical risk-based strategies and analytical requirements-based strategies) can be used to identify test conditions. The level of detail at which test conditions are specified can vary. Test conditions may state specific values (detailed test conditions) or may merely identify the variables (high level test conditions) depending on several factors such as the level of testing, the information available at the time of carrying out the analysis and the chosen level of detail (i.e., the degree of granularity of the documentation).

Factors affecting level of detail

There are a number of factors to consider when deciding on the level of detail at which to specify test conditions, including:

- Level of testing

  At the component testing level, test conditions are more likely to be focused at a detailed level whereas, at higher levels of testing, they are more likely to be specified at a more abstract (higher) level.

- Level of detail and quality of the test basis

  The more detailed the test basis the more detailed the test conditions can be, and vice versa.

- System / software complexity

  The greater the complexity of the system / software, the greater the need for more detailed test conditions. Greater complexity typically requires more thorough testing and this can be achieved by identifying and testing more, and more detailed, test conditions.

- Project and product risk

  High risk items require more thorough testing so we would expect to see more detailed test conditions.

- The relationship between the test basis, what is to be tested and how it is to be tested

  The approach to testing (the ‘how’) has a particular impact on the level of test conditions. Scripted testing is typically served well by many detailed test conditions, whereas unscripted approaches to testing that use the testers’ intuition are usually well served by fewer high level test conditions.

- Software development lifecycle in use

  Incremental and iterative lifecycles are more likely to identify test conditions at a high level whereas sequential lifecycles (particularly for large projects) are more likely to use detailed test conditions.

- Test management tool being used

Different test management tools support different test processes, some encouraging more detailed test conditions than others.

- Level at which test design and other test work products are to be specified and documented

  If the test work products are specified at a detailed level it is likely that test conditions should be consistent with this.

- Skills and knowledge of the test analysts

  Specifying detailed test conditions is one way for the test analysts to pass on their skills and knowledge to others who may have to execute the tests. If the test analysts with the skills and knowledge will be the people executing the tests then it may not be necessary for them to specify test conditions at such a detailed level.

- The level of maturity of the test process and the organization itself

  A higher level of maturity may require a greater level of detail in the test conditions, when for example safety-critical software is subjected to a rigorous testing process; or, it may allow a lesser level of detail when availability of skilled and experienced staff allows good enough testing to be accomplished without so much formality.

- Availability of other project stakeholders for consultation

  If it will be difficult to consult project stakeholders during testing, it may be of greater benefit to identify test conditions at a more detailed level. If however, project stakeholders can be consulted easily throughout the project then test conditions need not be specified in so much detail.

Detailed (low level) test conditions

Specifying test conditions in a detailed fashion will tend to result in a larger number of test conditions. For example, you might have a single general test condition, “Test checkout,” for an e-commerce application. However, this might be split into multiple detailed test conditions, with one or several test conditions for each supported payment method, with perhaps additional conditions for each possible destination country, and so on.

Advantages of specifying test conditions at a detailed level include:

- Facilitates more flexibility in relating other test work products (e.g., test cases) to the test basis and test objectives, thus providing **better and more detailed monitoring and control** for a Test Manager.
- Contributes to **defect prevention**, as discussed at Foundation Level, by occurring early in a project for higher levels of testing, as soon as the test basis is established and potentially before system architecture and detailed design are available.
- **Better relates testing work products to stakeholders** (i.e. in terms that they can understand). Often, test cases and other testing work products mean nothing to business stakeholders and simple metrics such as the number of test cases executed mean nothing to the coverage requirements of stakeholders.
- **Helps to influence and direct** not just other testing activities, but also other development activities if done early enough.
- Enables test design, implementation and execution, together with the resulting work products, to be optimized by **more efficient coverage** of detailed measures and targets.
- Provides the basis for **clearer horizontal traceability** within a test level (i.e., to the related artefacts at that level).

Disadvantages of specifying test conditions at a detailed level include:

- Potentially **time-consuming**.

  More effort will be required to identify and document a larger number of test conditions. Also, as the test conditions are more detailed, they will be more specific. Determining the precise values to be used in detailed test conditions will take more effort to specify than it takes to document high level test conditions.

- **Maintenance** can become difficult in a changing environment.

  Clearly the more test conditions that are identified and documented, the more updates may be needed when the software changes. It will not necessarily be more difficult, but will require more effort to keep the test conditions up to date.

- **Level of formality** needs to be defined and implemented across the team.

  Achieving consistency of detail is important. Ensuring all the team members understand the depth of detail that is required may take some time, effort and practice.

Specification of detailed test conditions can be particularly effective in the following situations:

- Lightweight test design documentation methods, such as checklists, are being used to accommodate an incremental or iterative development lifecycle, cost and / or time constraints or other factors.

  Although detailed test conditions may seem to be a contradiction to an otherwise 'lightweight' approach, the idea here is that the detailed test conditions serve as the primary test documentation for what is effectively unscripted test execution (i.e. no test case documentation is generated).

- Few or no formal requirements or other development work products are available as the test basis.

  In these situations the detailed test conditions become the main test documentation driving the testing.

- The project is large-scale, complex or high risk and requires a level of monitoring and control that cannot be delivered by simply relating test cases to development work products.

  Greater formality and attention to detail (being more thorough) is better served with detailed test conditions.

High level (less detailed) test conditions

Test conditions may be specified with less detail when the test basis can be related easily and directly to test design work products. This is more likely to be the case for the following:

- Component level testing.
- Less complex projects where simple hierarchical relationships exist between what is to be tested and how it is to be tested.
- Acceptance testing, where use cases can be used to help define tests.

1.4 Test Design

Learning Objective

TM-1.4.1	K3	Use traceability to check completeness and consistency of designed test cases with respect to the defined test conditions.

Test design is the activity that defines "how" something is to be tested. It involves the identification of test cases by the stepwise elaboration of the test conditions that were identified during analysis (or of the test basis, if test analysis was not performed earlier), possibly using test techniques identified in the test strategy and/or the test plan.

Relating test cases to test basis and test objectives

Depending on the approaches being used for test monitoring, test control and traceability, test cases may be directly related (or indirectly related via the test conditions) to the test basis and defined objectives. These objectives include strategic objectives, test objectives and other project or stakeholder criteria for success.

Separate versus integrated activity

Test design for a given test level can be performed once test conditions are identified and enough information is available to enable the production of either low or high-level test cases, according to the employed approach to test design. For higher levels of testing, it is more likely that test design is a separate activity following earlier test analysis. For lower levels of testing, it is likely that test analysis and design will be conducted as an integrated activity.

Early implementation activities

It is also likely that some tasks that normally occur during test implementation will be included in the test design process when using an iterative approach to building the tests required for execution; e.g., the creation of test data. In fact, this approach can optimize the coverage obtained by the test conditions, either creating low-level or high-level test cases in the process.

Concrete and logical test cases

One of the jobs of the Test Analyst is to determine the best types of test case for a given situation. Concrete test cases provide all the specific information and instructions needed for the tester to execute the test (including any data requirements) and verify the results. Concrete test cases are useful when:

- the requirements are well-defined,
- the testing staff are less experienced, and
- when external verification of the tests, such as audits, is required.

Concrete test cases provide excellent reproducibility (i.e., another tester will get the same results), but may also require a significant amount of maintenance effort. Concrete test cases tend to limit tester ingenuity and, therefore, test coverage during execution.

Logical test cases provide guidelines for what should be tested, but allow the Test Analyst to vary the actual data or even the procedure that is followed when executing the test. Logical test cases may provide better coverage than concrete test cases because they will vary slightly each time they are executed. This, however, also leads to a loss in reproducibility. Logical test cases are best used when:

- the requirements are not well-defined,
- the Test Analyst who will be executing the test is experienced with both testing and the product, and
- when formal documentation is not required (e.g., no audits will be conducted).

Logical test cases may be defined early in the requirements process, when the requirements are not yet well-defined. These test cases may be used to develop concrete test cases when the requirements become more defined and stable. In this case, the test case creation is done sequentially, flowing from logical to concrete with only the concrete test cases used for execution.

1.5 Test Implementation

Learning Objective

TM-1.5.1	K3	Use risks, prioritization, test environment and data dependencies, and constraints to develop a test execution schedule which is complete and consistent with respect to the test objectives, test strategy, and test plan.

From the ISTQB Glossary

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

test procedure: See test procedure specification.

test procedure specification: A document specifying a sequence of actions for the execution of a test. Also known as test script or manual test script. [After IEEE 829] See also test specification.

test script: Commonly used to refer to a test procedure specification, especially an automated one.

Scope

Test implementation is the activity in which tests are made ready to be executed. An important part of this is that they should be organized and prioritized by the Test Analysts. In formally-documented contexts, test implementation is the activity in which test designs are implemented as concrete test cases, test procedures, and test data. Some organizations following the IEEE 829 [IEEE829] standard define inputs and their associated expected results in test case specifications, and test steps in test procedure specifications. More commonly, each test's inputs, expected results, and test steps are documented together. Test implementation also includes the creation of stored test data (e.g., in flat files or database tables).

Note that some implementation tasks may be undertaken early and integrated with the test design activity as described under "Early implementation activities" in section 1.4 "Test Design".

The main challenge when designing and organizing test procedures is deciding in what order to test things. More specifically, in developing test procedures we are putting sets of test cases into groups that define their execution order. Each test procedure will comprise a sequence of test cases in which the post conditions of one test case form the pre-conditions of the next. Each test procedure is a logical unit of test execution work that will typically be executed from start to finish in one go (if the software doesn't fail).

To create the necessary preconditions for a given test case, we may need to run one or more other test cases that have already been incorporated into an earlier test procedure. This duplication should be avoided if possible (though it will not always be possible to do so).

Test implementation also involves final checks to ensure that the test team is ready for test execution to take place. Checks could include ensuring delivery of the required test environment, test data and code (possibly running some test environment and/or code acceptance tests as a "smoke test") and that all test cases have been written, reviewed and are ready to be run. It may also include checking against explicit and implicit entry criteria for the test level in question (see Section 1.7). Test implementation can also involve developing a detailed description of the test environment and test data.

Level of detail

The level of detail and associated complexity of work done during test implementation will be influenced by the detail of other test work products (e.g., test cases and test conditions). In some cases, particularly where tests are to be archived for long-term re-use in regression testing, tests may provide detailed descriptions of the steps necessary to execute a test, so as to ensure reliable, consistent execution regardless of who runs the test. If regulatory rules apply, tests should provide evidence of compliance to applicable standards (see section 2.9).

Test execution schedule

In addition to the test procedures, we may also be required to prepare a test execution schedule that defines the order in which the test procedures (both manual and automated tests) are to be executed. This should also include when they are to be performed and by whom. It will need to consider the logical dependencies between the test procedures (some may have to be executed before others, when one set of test procedures tests software that depends on the data created by another set). The test execution schedule should also allow for regression testing to be performed on each delivery of software. This may mean that some test procedures will be repeated many times.

Test Managers should carefully check for constraints, including risks and priorities that might require tests to be run in a particular order or on particular equipment. Dependencies on the test environment or test data must be known and checked.

Early test implementation

There may be some disadvantages to early test implementation. With an Agile lifecycle, for example, the code may change dramatically from one iteration to the next, rendering much of the earlier implementation work obsolete. Even without a lifecycle as change-prone as Agile, any iterative or incremental lifecycle may result in significant changes between iterations, making scripted tests unreliable or subject to high maintenance. The same is true for poorly-managed sequential lifecycles, where the requirements can change frequently even late into the project. Before embarking on an extensive test implementation effort, it is wise to understand the software development lifecycle and the predictability of the software features that will be available for testing.

There may be some advantages in early test implementation. For example, concrete tests provide worked examples of how the software should behave, if written in accordance with the test basis. Business domain experts are likely to find verification of concrete tests easier than verification of abstract business rules, and may thereby identify further weaknesses in software specifications. Such verified tests may provide illuminating illustrations of required behaviour for software designers and developers.

1.6 Test Execution

Learning Objective

TM-1.6.1 K3 Use traceability to monitor test progress for completeness and consistency with the test objectives, test strategy, and test plan.

From the ISTQB Glossary

test log: A chronological record of relevant details about the execution of tests. [IEEE 829]

Entry criteria

Clearly test execution cannot begin until the test object has been delivered. However, test execution should not be started until all the entry criteria to test execution are satisfied (or waived). Although the entry criteria will vary from project to project, common criteria include:

- tests have been designed or at least defined

  Depending on the approach to test execution, for example, the extent to which the testing will be scripted / unscripted.

- necessary tools are in place

  Arguably the most necessary tools are test management, defect tracking and (if applicable) test execution automation.

- test results tracking, including metrics tracking, should be working

  Also the tracked data should be understood by all team members.

- Standards for test logging and defect reporting should be available and published.

  Again, all team members should be familiar with and understand these.

By ensuring that these items are in place prior to test execution, the execution can proceed efficiently.

Scripted and unscripted testing

Tests should be executed according to the designed / defined test cases. Having put the effort in to identify and (probably) design tests, it is important to ensure that they are executed correctly. However, the Test Manager should consider allowing some additional time in the schedule so that the testers can cover additional interesting test scenarios and behaviours that are observed during testing. This integration of scripted and unscripted testing techniques

helps to guard against defect escapes due to gaps in scripted coverage, and to circumvent the pesticide paradox.

When following a test strategy that is at least in part reactive, time should be reserved for test sessions using experience-based and defect-based techniques. Of course, any failure detected during such unscripted testing must describe the variations from the written test case that are necessary to reproduce the failure.

Properly automated tests, of course, will follow their defined instructions without deviation.

Test Manager responsibilities

The main role of a Test Manager during test execution is to monitor progress according to the test plan and, if required, to initiate and carry out control actions to guide testing toward a successful conclusion in terms of mission, objectives, and strategy. To do so, the Test Manager can use traceability from the test results back to the test cases / procedures, the test conditions, the test basis, and ultimately the test objectives, and also from the test objectives forward to the test results. This process is described in detail in Section 2.6.

1.7 Evaluating Exit Criteria and Reporting

Learning Objective

TM-1.7.1	K2	Explain the importance of accurate and timely information collection during the test process to support accurate reporting and evaluation against exit criteria.

Documentation of, and reporting for, test progress monitoring and control are discussed in detail in Section 2.6.

Effective evaluation and reporting

This covers two of the Test Manager's key responsibilities: regular reporting of the status and progress of testing, and final reporting of the results obtained. It is important to ensure that effective processes are in place to provide the information necessary for this. Whilst Test Analysts and Technical Test Analysts are responsible for gathering accurate and timely information, it is the Test Manager’s responsibility to ensure that appropriate processes and mechanisms are in place to allow the testers to fulfil these responsibilities.

Definition of the information requirements and methods for collection are part of test planning, monitoring and control. During test analysis, test design, test implementation and test execution, the Test Manager should ensure that members of the test team responsible for these activities are providing the information required in an accurate and timely manner so as to facilitate effective evaluation and reporting.

Detail and frequency

The frequency and level of detail required for reporting are dependent on the project and the organization. This should be negotiated during the test planning phase and should include consultation with relevant project stakeholders.

1.8 Test Closure Activities

Learning Objectives

TM-1.8.1	K2	Summarize the four groups of test closure activities.
TM-1.8.2	K3	Implement a project retrospective to evaluate processes and discover areas to improve.

From the ISTQB Glossary

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. See also test process.

Once test execution is determined to be complete, the key outputs should be captured and either passed to the relevant person or archived. Collectively, these are test closure activities. Test closure activities fall into four main groups:

1. **Test completion check** - ensuring that all test work is indeed concluded. For example, all planned tests should be either run or deliberately skipped, and all known defects should be either fixed and confirmation tested, deferred for a future release, or accepted as permanent restrictions.
2. **Test artefacts handover** - delivering valuable work products to those who need them. For example, known defects deferred or accepted should be communicated to those who will use and support the system. Tests and test environments should be given to those responsible for maintenance testing. Regression test sets (either automated or manual) should be documented and delivered to the maintenance team.
3. **Lessons learned** - performing or participating in retrospective meetings where important lessons (both from within the test project and across the whole software development lifecycle) can be documented. In these meetings, plans are established to ensure that good practices can be repeated and poor practices are either not repeated or, where issues cannot be resolved, they are accommodated within project plans. Areas to be considered include the following:

    - Was the user representation in the quality risk analysis sessions a broad enough cross-section? For example, due to late discovery of unanticipated defect clusters, the team may decide that a broader cross-section of user representatives should participate in quality risk analysis sessions on future projects.

    - Were the estimates accurate? For example, estimates may have been significantly misjudged and therefore future estimation activities will need to account for this together with the underlying reasons, e.g., was testing inefficient or was the estimate actually lower than it should have been.

    - What are the trends and the results of cause and effect analysis of the defects? For example, assess if late change requests affected the quality of the analysis and development, look for trends that indicate bad practices, e.g., skipping a test level which would have found defects earlier and in a more cost effective manner, for perceived savings of time. Check if defect trends could be related to areas such as new technologies, staffing changes, or the lack of skills.

    - Are there potential process improvement opportunities? If yes, what are they and how might they be implemented?

    - Were there any unanticipated variances from the plan that should be accommodated in future planning? Let's anticipate them next time!

4. **Archiving** results, logs, reports, and other documents and work products in the configuration management system. For example, the test plan and project plan should both be stored in a planning archive, with clear links to the system and version they were used on.

It is common for one or more of these tasks to be omitted, usually due to premature reassignment or dismissal of project team members, resource or schedule pressures on subsequent projects, or team burnout. They are, however, important and should be explicitly included as part of the test plan; then, it will be easier for the Test Manager to resist any pressure to overlook them. On projects carried out under contract, such as custom

development, the contract should specify the tasks required so that there is a clear obligation to perform them.

Metrics to monitor test closure activities may include:

- Percentage of test cases run during test execution (coverage),
- Percentage of test cases checked into a re-usable test case repository,
- Ratio of test cases automated : to be automated,
- Percentage of test cases identified as regression tests,
- Percentage breakdown of outstanding defect reports closed (e.g. deferred, no further action, change request, etc.),
- Percentage of work products identified and archived.