

BS ISO/IEC/IEEE 29119-1:2013



BSI Standards Publication

Software and systems engineering — Software testing

Part 1: Concepts and definitions

bsi.

...making excellence a habit.
AVA Documents™
Document Sharing Hub

National foreword

This British Standard is the UK implementation of ISO/IEC/IEEE 29119-1:2013. It supersedes BS 7925-1:1998 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/15, Software and systems engineering.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2013. Published by BSI Standards Limited 2013

ISBN 978 0 580 73279 9

ICS 35.080

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 September 2013.

Amendments issued since publication

Date	Text affected
------	---------------

INTERNATIONAL STANDARD

**ISO/IEC/
IEEE
29119-1**

First edition
2013-09-01

Software and systems engineering — Software testing —

Part 1: Concepts and definitions

Ingénierie du logiciel et des systèmes — Essais du logiciel —

Partie 1: Concepts et définitions



Reference number
ISO/IEC/IEEE 29119-1:2013(E)

Documents
© ISO/IEC 2013
© IEEE 2013
Document Sharing Hub



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2013
© IEEE 2013

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from ISO, IEC or IEEE at the respective address below.

ISO copyright office
Case postale 56
CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

IEC Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland
E-mail inmail@iec.ch
Web www.iec.ch

Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York
NY 10016-5997, USA
E-mail stds.ipr@ieee.org
Web www.ieee.org

Published in Switzerland

Contents

Page

Foreword	v
Introduction.....	vi
1 Scope	1
2 Conformance	1
3 Normative references.....	1
4 Terms and definitions	1
5 Software Testing Concepts	12
5.1 Introduction to Software Testing	12
5.1.1 The Role of Testing in Verification and Validation	14
5.1.2 Exhaustive Testing.....	14
5.1.3 Testing as a Heuristic	14
5.2 Software Testing in an Organizational and Project Context.....	14
5.2.1 The Test Process	17
5.3 Generic Testing Processes in the Software Life cycle.....	19
5.3.1 Development Project Sub-processes and their Results	20
5.3.2 On-going Maintenance and its Results	21
5.3.3 Support Processes for the Software Development Life Cycle	22
5.4 Risk-based Testing.....	24
5.4.1 Using Risk-Based Testing in the Organizational Test Process.....	25
5.4.2 Using Risk-Based Testing in the Test Management processes	25
5.4.3 Using Risk-Based Testing in the Dynamic Testing processes	25
5.5 Test Sub-process	26
5.5.1 Test Objectives	26
5.5.2 Test Item	27
5.5.3 Testing of Quality Characteristics	27
5.5.4 Test Basis.....	28
5.5.5 Retesting and Regression Testing	29
5.5.6 Test Design Techniques	29
5.6 Test Practices	30
5.6.1 Introduction.....	30
5.6.2 Requirements-Based Testing.....	31
5.6.3 Model-Based Testing	31
5.6.4 Mathematical-Based Testing.....	32
5.6.5 Experience-Based Testing	32
5.6.6 Scripted and Unscripted Testing	33
5.7 Automation in Testing.....	34
5.8 Defect Management.....	34
Annex A (informative) The Role of Testing in Verification and Validation.....	35
Annex B (informative) Metrics and Measures	36
B.1 Metrics and Measures	36
Annex C (informative) Testing in Different Life Cycle Models	37
C.1 Overview.....	37
C.2 Agile Development and Testing	37
C.2.1 Agile Development Principles.....	37
C.2.2 Test Management in Agile Development	38
C.2.3 Test Sub-processes in Agile Development.....	39
C.3 Sequential Development and Testing	40
C.3.1 Sequential Development Principles	40

C.3.2	Test Management in Sequential Development	40
C.3.3	Test Sub-processes in Sequential Development	41
C.4	Evolutionary Development and Testing	41
C.4.1	Evolutionary Development Principles	41
C.4.2	Test Management in Evolutionary Development.....	42
C.4.3	Test Sub-processes in Evolutionary Development.....	42
Annex D	(informative) Detailed Test Sub-process Examples	44
D.1	Overview	44
D.2	Acceptance Test Sub-process	45
D.3	Detailed Design Test Sub-process.....	45
D.4	Integration Test Sub-process	46
D.5	Performance Test Sub-process.....	48
D.6	Regression Test Sub-process	49
D.7	Retest Test Sub-process.....	51
D.8	Story Set Test Sub-process	51
D.9	Story Test Sub-process	51
D.10	System Test Sub-process.....	52
D.11	Component Test Sub-process.....	53
Annex E	(informative) Roles and Responsibilities in Testing	54
E.1	Testing Roles	54
E.2	Communication in Testing.....	54
E.3	Independence in Testing.....	54
Bibliography		56

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of ISO/IEC JTC 1 is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is called to the possibility that implementation of this standard may require the use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. ISO/IEEE is not responsible for identifying essential patents or patent claims for which a license may be required, for conducting inquiries into the legal validity or scope of patents or patent claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance or a Patent Statement and Licensing Declaration Form, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from ISO or the IEEE Standards Association.

ISO/IEC/IEEE 29119-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*, in cooperation with the Software & Systems Engineering Standards Committee of the IEEE Computer Society, under the Partner Standards Development Organization cooperation agreement between ISO and IEEE.

ISO/IEC/IEEE 29119 consists of the following standards, under the general title *Software and systems engineering — Software testing*:

- *Part 1: Concepts and definitions*
- *Part 2: Test processes*
- *Part 3: Test documentation*
- *Part 4: Test techniques*

Introduction

The purpose of the ISO/IEC/IEEE 29119 series of software testing standards is to define an internationally-agreed set of standards for software testing that can be used by any organization when performing any form of software testing.

It is recognized that there are many different types of software, software organizations, and methodologies. Software domains include information technology (IT), personal computers (PC), embedded, mobile, and scientific and many other classifications. Software organizations range from small to large, co-located to world-wide, and commercial to public service-oriented. Software methodologies include object-oriented, traditional, data driven and agile. These and other factors influence software testing. This series of international standards can support testing in many different contexts.

This part of ISO/IEC/IEEE 29119 facilitates the use of the other ISO/IEC/IEEE 29119 Software Testing standards by introducing the vocabulary on which this series of international standards are built and provides examples of their application in practice. Part 1 is informative providing definitions, a description of the concepts of software testing and ways to apply the software testing process defined in this part of ISO/IEC/IEEE 29119 and guidance for the other parts.

Initially, general software testing concepts are discussed. The role of software testing in an organizational and project context is described. Software testing in a generic software life cycle is explained, introducing the way software test processes and sub-processes may be established for specific test items or with specific test objectives. It describes how software testing fits into different life cycle models. The use of different practices in test planning is demonstrated; as well as how automation can be used to support testing. The involvement of testing in defect management is also discussed. Annex A describes the role of testing within the larger scope of verification and validation. Annex B provides a brief introduction to metrics used to monitor and control testing. Annex C contains a set of examples showing how to apply the standard in different life cycle models. Annex D provides examples on detailed test sub-processes. Annex E provides additional information on the roles and responsibilities typically encountered in test groups and tester independence. Finally, the Bibliography is at the end of the document.

Note that Title Case is used throughout this part of ISO/IEC/IEEE 29119 to denote processes and documents that are specified in ISO/IEC/IEEE 29119-2 and ISO/IEC/IEEE 29119-3 (e.g. Test Planning Process, Test Plan), whereas lowercase letters are used for documents that form parts of other documents (e.g. the project test strategy is an element of the Project Test Plan).

The test process model that the ISO/IEC/IEEE 29119 series of software testing standards are based on is defined in detail in ISO/IEC/IEEE 29119-2 Test Processes. ISO/IEC/IEEE 29119-2 covers the software testing processes at the organizational level, test management level and for dynamic test levels. Testing is the primary approach to risk treatment in software development. This standard defines a risk-based approach to testing. Risk-based testing is a recommended approach to strategizing and managing testing that allows testing to be prioritized and focused.

Templates and examples of test documentation that are produced during the testing process are defined in ISO/IEC/IEEE 29119-3 Test Documentation. Software testing techniques that can be used during testing are defined in ISO/IEC/IEEE 29119-4 Test Techniques.

Together, this series of international standards aims to provide stakeholders with the ability to manage and perform software testing in any organization.

Software and systems engineering — Software testing —

Part 1: Concepts and definitions

1 Scope

This part of ISO/IEC/IEEE 29119 specifies definitions and concepts in software testing. It provides definitions of testing terms and discussion of concepts key to the understanding of the ISO/IEC/IEEE 29119 series of software testing international standards.

2 Conformance

ISO/IEC/IEEE 29119-1 is informative and no conformance with it is required.

The ISO/IEC/IEEE 29119 software testing series of standards contain three standards where conformance may be claimed:

- test processes;
- test documentation;
- test techniques.

Conformance is addressed in ISO/IEC/IEEE 29119-2, ISO/IEC/IEEE 29119-3 and ISO/IEC/IEEE 29119-4.

3 Normative references

This document does not require the use of any normative references. Standards useful for the implementation and interpretation of this part of ISO/IEC/IEEE 29119 are listed in the Bibliography.

4 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC/IEEE 24765 and the following apply.

NOTE The following terms and definitions are provided to assist with the understanding and readability of parts 1, 2, 3 and 4 of the ISO/IEC/IEEE 29119 Software Testing standards so some terms defined in ISO/IEC/IEEE 29119-1 will not be used in ISO/IEC/IEEE 29119-1 and will only be used in another standard in the ISO/IEC/IEEE 29119 series. Only terms critical to the understanding of these standards are included; this clause is not intended to provide a complete list of testing terms. The systems and software engineering Vocabulary ISO/IEC/IEEE 24765 should be referenced for terms not defined in this clause. This source is available at the following web site: <http://www.computer.org/sevocab>.

4.1

accessibility testing

type of usability testing used to measure the degree to which a test item can be operated by users with the widest possible range of characteristics and capabilities

4.2

actual results

set of behaviours or conditions of a test item, or set of conditions of associated data or the test environment, observed as a result of test execution

EXAMPLE Outputs to hardware, changes to data, reports, and communication messages sent.

4.3

backup and recovery testing

type of reliability testing that measures the degree to which system state can be restored from backup within specified parameters of time, cost, completeness, and accuracy in the event of failure

4.4

black-box testing

see specification-based testing (4.39)

4.5

capacity testing

type of performance efficiency testing conducted to evaluate the level at which increasing load (of users, transactions, data storage, etc.) compromises a test item's ability to sustain required performance

4.6

compatibility testing

type of testing that measures the degree to which a test item can function satisfactorily alongside other independent products in a shared environment (co-existence), and where necessary, exchanges information with other systems or components (interoperability)

4.7

coverage item

see test coverage item (4.54)

4.8

decision

types of statements in which a choice between two or more possible outcomes controls which set of actions will result

Note 1 to entry: Typical decisions are simple selections (e.g. if-then-else), to decide when to exit loops (e.g. while-loop), and in case (switch) statements (e.g. case-1-2-3-...-N).

4.9

dynamic testing

testing that requires the execution of the test item

4.10

endurance testing

type of performance efficiency testing conducted to evaluate whether a test item can sustain a required load continuously for a specified period of time

4.11

equivalence partition

subset of the range of values of a variable, or set of variables, within a test item or at its interfaces such that all the values in the partition can reasonably be expected to be treated similarly by the test item (i.e. they may be considered "equivalent") by the test item

4.12

equivalence partition coverage

proportion of identified equivalence partitions of a test item that are covered by a test set

Note 1 to entry: In many cases, the identification of equivalence partitions is subjective (especially in the sub-partitioning of "invalid" partitions), so a definitive count of the number of equivalence partitions in a test item could be impossible.

4.13

equivalence partitioning

test design technique in which test cases are designed to exercise equivalence partitions by using one or more representative members of each partition

4.14

error guessing

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

Note 1 to entry: The relevant knowledge could be gained from personal experience, or might be encapsulated in, for example, a defects database or a "bug taxonomy".

4.15

expected results

observable predicted behaviour of the test item under specified conditions based on its specification or another source

4.16

exploratory testing

experience-based testing in which the tester spontaneously designs and executes tests based on the tester's existing relevant knowledge, prior exploration of the test item (including the results of previous tests), and heuristic "rules of thumb" regarding common software behaviours and types of failure

Note 1 to entry: Exploratory testing hunts for hidden properties (including hidden behaviours) that, while quite possibly benign by themselves, could interfere with other properties of the software under test, and so constitute a risk that the software will fail.

4.17

feature set

collection of items which contain the test conditions of the test item to be tested which can be collected from risks, requirements, functions, models, etc.

Note 1 to entry: This could be the set of all features for the item (its full feature set), or a subset identified for a specific purpose (the functional feature set etc.).

4.18

Incident Report

documentation of the occurrence, nature, and status of an incident

4.19

installability testing

type of portability testing conducted to evaluate whether a test item or set of test items can be installed as required in all specified environments

4.20

load testing

type of performance efficiency testing conducted to evaluate the behaviour of a test item under anticipated conditions of varying load, usually between anticipated conditions of low, typical, and peak usage

4.21

maintainability testing

test type conducted to evaluate the degree of effectiveness and efficiency with which a test item may be modified

4.22

Organizational Test Policy

an executive-level document that describes the purpose, goals, and overall scope of the testing within an organization, and which expresses why testing is performed and what it is expected to achieve

Note 1 to entry: It is generally preferable to keep the Organizational Test Policy as short as possible in a given context.

4.23

Organizational Test Process

test process for developing and managing organizational test specifications

4.24

organizational test specification

document that provides information about testing for an organization, i.e. information that is not project-specific

EXAMPLE The most common examples of organizational test specifications are Organizational Test Policy and Organizational Test Strategy.

4.25

Organizational Test Strategy

document that expresses the generic requirements for the testing to be performed on all the projects run within the organization, providing detail on how the testing is to be performed

Note 1 to entry: The Organizational Test Strategy is aligned with the Organizational Test Policy.

Note 2 to entry: An organisation could have more than one Organizational Test Strategy to cover markedly different project contexts.

4.26

pass/fail criteria

decision rules used to determine whether a test item, or feature of a test item, has passed or failed after testing

4.27

performance testing

type of testing conducted to evaluate the degree to which a test item accomplishes its designated functions within given constraints of time and other resources

4.28

portability testing

type of testing conducted to evaluate the ease with which a test item can be transferred from one hardware or software environment to another, including the level of modification needed for it to be executed in various types of environment

4.29

procedure testing

type of functional suitability testing conducted to evaluate whether procedural instructions for interacting with a test item or using its outputs meet user requirements and support the purpose of their use

4.30

product risk

risk that a product may be defective in some specific aspect of its function, quality, or structure

4.31

project risk

risk related to the management of a project

EXAMPLE Lack of staffing, strict deadlines, changing requirements.

4.32

regression testing

testing following modifications to a test item or to its operational environment, to identify whether regression failures occur

Note 1 to entry: The sufficiency of a set of regression test cases depends on the item under test and on the modifications to that item or its operational environment.

4.33

reliability testing

type of testing conducted to evaluate the ability of a test item to perform its required functions, including evaluating the frequency with which failures occur, when it is used under stated conditions for a specified period of time

4.34

retesting

re-execution of test cases that previously returned a "fail" result, to evaluate the effectiveness of intervening corrective actions

Note 1 to entry: Also known as confirmation testing.

4.35

risk-based testing

testing in which the management, selection, prioritisation, and use of testing activities and resources are consciously based on corresponding types and levels of analyzed risk

4.36

scenario testing

class of test design technique in which tests are designed to execute individual scenarios

Note 1 to entry: A scenario can be a user story, use-case, operational concept, or sequence of events the software may encounter etc.

4.37

scripted testing

dynamic testing in which the tester's actions are prescribed by written instructions in a test case

Note 1 to entry: This term normally applies to manually executed testing, rather than the execution of an automated script.

4.38

security testing

type of testing conducted to evaluate the degree to which a test item, and associated data and information, are protected so that unauthorized persons or systems cannot use, read, or modify them, and authorized persons or systems are not denied access to them

4.39

specification-based testing

testing in which the principal test basis is the external inputs and outputs of the test item, commonly based on a specification, rather than its implementation in source code or executable software

Note 1 to entry: Synonyms for specification-based testing include black-box testing and closed box testing.

4.40

statement coverage

percentage of the set of all executable statements of a test item that are covered by a test set

4.41

statement testing

test design technique in which test cases are constructed to force execution of individual statements in a test item

4.42

static testing

testing in which a test item is examined against a set of quality or other criteria without code being executed

EXAMPLE Reviews, static analysis.

4.43

stress testing

type of performance efficiency testing conducted to evaluate a test item's behaviour under conditions of loading above anticipated or specified capacity requirements, or of resource availability below minimum specified requirements

4.44

structural testing

see structure-based testing (4.45)

4.45

structure-based testing

dynamic testing in which the tests are derived from an examination of the structure of the test item

Note 1 to entry: Structure-based testing is not restricted to use at component level and can be used at all levels, e.g. menu item coverage as part of a system test.

Note 2 to entry: Techniques include branch testing, decision testing, and statement testing.

Note 3 to entry: Synonyms for structure-based testing are structural testing, glass-box testing, and white box testing.

4.46

suspension criteria

criteria used to (temporarily) stop all or a portion of the testing activities

4.47

test basis

body of knowledge used as the basis for the design of tests and test cases

Note 1 to entry: The test basis may take the form of documentation, such as a requirements specification, design specification, or module specification, but may also be an undocumented understanding of the required behaviour.

4.48

test case

set of test case preconditions, inputs (including actions, where applicable), and expected results, developed to drive the execution of a test item to meet test objectives, including correct implementation, error identification, checking quality, and other valued information

Note 1 to entry: A test case is the lowest level of test input (i.e. test cases are not made up of test cases) for the test sub-process for which it is intended.

Note 2 to entry: Test case preconditions include test environment, existing data (e.g. databases), software under test, hardware, etc.

Note 3 to entry: Inputs are the data information used to drive test execution.

Note 4 to entry: Expected results include success criteria, failures to check for, etc.

4.49

Test Case Specification

documentation of a set of one or more test cases

4.50

Test Completion Process

Test Management Process for ensuring that useful test assets are made available for later use, test environments are left in a satisfactory condition, and the results of testing are recorded and communicated to relevant stakeholders

4.51

Test Completion Report

report that provides a summary of the testing that was performed

Note 1 to entry: Also known as a Test Summary Report.

4.52

test condition

testable aspect of a component or system, such as a function, transaction, feature, quality attribute, or structural element identified as a basis for testing

Note 1 to entry: Test conditions can be used to derive coverage items, or can themselves constitute coverage items.

4.53

test coverage

degree, expressed as a percentage, to which specified test coverage items have been exercised by a test case or test cases

4.54

test coverage item

attribute or combination of attributes that is derived from one or more test conditions by using a test design technique that enables the measurement of the thoroughness of the test execution

4.55

test data

data created or selected to satisfy the input requirements for executing one or more test cases, which may be defined in the Test Plan, test case or test procedure

Note 1 to entry: Test data could be stored within the product under test (e.g. in arrays, flat files, or a database), or could be available from or supplied by external sources, such as other systems, other system components, hardware devices, or human operators.

4.56

Test Data Readiness Report

document describing the status of each test data requirement

4.57

Test Design and Implementation Process

test process for deriving and specifying test cases and test procedures

4.58

Test Design Specification

document specifying the features to be tested and their corresponding test conditions

4.59

test design technique

activities, concepts, processes, and patterns used to construct a test model that is used to identify test conditions for a test item, derive corresponding test coverage items, and subsequently derive or select test cases

4.60

test environment

facilities, hardware, software, firmware, procedures, and documentation intended for or used to perform testing of software

Note 1 to entry: A test environment could contain multiple environments to accommodate specific test sub-processes (e.g. a unit test environment, a performance test environment etc.).

4.61

test environment readiness report

document that describes the fulfilment of each test environment requirement

4.62

Test Environment Requirements

description of the necessary properties of the test environment

Note 1 to entry: All or parts of the test environment requirements could reference where the information can be found, e.g. in the appropriate Organizational Test Strategy, Test Plan, and/or Test Specification.

4.63

Test Environment Set-up Process

dynamic test process for establishing and maintaining a required test environment

4.64

test execution

process of running a test on the test item, producing actual result(s)

4.65

Test Execution Log

document that records details of the execution of one or more test procedures

4.66

Test Execution Process

dynamic test process for executing test procedures created in the Test Design and Implementation Process in the prepared test environment, and recording the results

4.67

Test Incident Reporting Process

dynamic test process for reporting to the relevant stakeholders issues requiring further action that were identified during the test execution process

4.68

test item

work product that is an object of testing

EXAMPLE A system, a software item, a requirements document, a design specification, a user guide.

4.69

test level

specific instantiation of a test sub-process

EXAMPLE The following are common test levels that can be instantiated as test sub-processes: component test level/sub-process, integration test level/sub-process, system test level/sub-process, acceptance test level/sub-process.

Note 1 to entry: Test levels are synonymous with test phases.

4.70

test management

planning, scheduling, estimating, monitoring, reporting, control and completion of test activities

4.71

Test Management Process

test process containing the sub-processes that are required for the management of a test project

Note 1 to entry: See Test Planning Process, Test Monitoring and Control Process, Test Completion Process.

4.72

Test Monitoring and Control Process

Test Management Process for ensuring that testing is performed in line with a Test Plan and with organizational test specifications

4.73

test object

see test item (4.68)

4.74

test phase

specific instantiation of test sub-process

Note 1 to entry: Test phases are synonymous with test levels, therefore examples of test phases are the same as for test levels (e.g. system test phase/sub-process).

4.75

Test Plan

detailed description of test objectives to be achieved and the means and schedule for achieving them, organised to coordinate testing activities for some test item or set of test items

Note 1 to entry: A project can have more than one Test Plan, for example there could be a Project Test Plan (also known as a master test plan) that encompasses all testing activities on the project; further detail of particular test activities could be defined in one or more test sub-process plans (i.e. a system test plan or a performance test plan).

Note 2 to entry: Typically a Test Plan is a written document, though other plan formats could be possible as defined locally within an organization or project.

Note 3 to entry: Test Plans could also be written for non-project activities, for example a maintenance test plan.

4.76

Test Planning Process

Test Management Process used to complete test planning and develop Test Plans

4.77

test practice

conceptual framework that can be applied to the Organizational Test Process, the Test Management Process, and/or the Dynamic Test Process to facilitate testing

Note 1 to entry: Test Practices are sometimes referred to as test approaches.

4.78

test procedure

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

Note 1 to entry: Test procedures include detailed instructions for how to run a set of one or more test cases selected to be run consecutively, including set up of common preconditions, and providing input and evaluating the actual result for each included test case.

4.79

Test Procedure Specification

document specifying one or more test procedures, which are collections of test cases to be executed for a particular objective

Note 1 to entry: The test cases in a test set are listed in their required order in the test procedure.

Note 2 to entry: Also known as a manual test script. A test procedure specification for an automated test run is usually called a test script.

4.80

test process

provides information on the quality of a software product, often comprised of a number of activities, grouped into one or more test sub-processes

EXAMPLE The Test Process for a particular project may well consist of multiple sub-processes, e.g. a system test sub-process, a Test Planning sub-process (a part of the larger Test Management Process) or a static testing sub-process.

4.81

test requirement

see test condition (4.52)

4.82

test result

indication of whether or not a specific test case has passed or failed, i.e. if the actual result observed as test item output corresponds to the expected result or if deviations were observed

4.83

test script

test procedure specification for manual or automated testing

4.84

test set

set of one or more test cases with a common constraint on their execution

EXAMPLE A specific test environment, specialized domain knowledge or specific purpose.

4.85

test specification

complete documentation of the test design, test cases and test procedures for a specific test item

Note 1 to entry: A test specification could be detailed in one document, in a set of documents, or in other ways, for example in a mixture of documents and database entries.

4.86

test status report

report that provides information about the status of the testing that is being performed in a specified reporting period

4.87

test strategy

part of the Test Plan that describes the approach to testing for a specific test project or test sub-process or sub-processes

Note 1 to entry: The test strategy is a distinct entity from the Organizational Test Strategy.

Note 2 to entry: The test strategy usually describes some or all of the following: the test practices used; the test sub-processes to be implemented; the retesting and regression testing to be employed; the test design techniques and corresponding test completion criteria to be used; test data; test environment and testing tool requirements; and expectations for test deliverables.

4.88

test sub-process

test management and dynamic (and static) test processes used to perform a specific test level (e.g. system testing, acceptance testing) or test type (e.g. usability testing, performance testing) normally within the context of an overall test process for a test project

Note 1 to entry: A test sub-process could comprise one or more test types. Depending on the life cycle model used, test sub-processes are also typically called test phases, test levels, test stages or test tasks.

4.89

test technique

see test design technique (4.59)

4.90

test traceability matrix

document, spreadsheet, or other automated tool used to identify related items in documentation and software, such as requirements with associated tests

Note 1 to entry: Also known as: verification cross reference matrix, requirements test matrix, requirements verification table, and others.

Note 2 to entry: Different test traceability matrices could have different information, formats, and levels of detail.

4.91

test type

group of testing activities that are focused on specific quality characteristics

Note 1 to entry: A test type could be performed in a single test sub-process or could be performed across a number of test sub-processes (e.g. performance testing completed at a component test sub-process and also completed at a system test sub-process).

EXAMPLE Security testing, functional testing, usability testing, and performance testing.

4.92

testing

set of activities conducted to facilitate discovery and/or evaluation of properties of one or more test items

Note 1 to entry: Testing activities could include planning, preparation, execution, reporting, and management activities, insofar as they are directed towards testing.

4.93

testware

artefacts produced during the test process required to plan, design, and execute tests

Note 1 to entry: Testware can include such things as documentation, scripts, inputs, expected results, files, databases, environment, and any additional software or utilities used in the course of testing.

4.94

unscripted testing

dynamic testing in which the tester's actions are not prescribed by written instructions in a test case

4.95

volume testing

type of performance efficiency testing conducted to evaluate the capability of the test item to process specified volumes of data (usually at or near maximum specified capacity) in terms of throughput capacity, storage capacity, or both

4.96

white box testing

see structure-based testing (4.45)

5 Software Testing Concepts

5.1 Introduction to Software Testing

Software testing is necessary because:

- Information on the quality characteristics of the test item(s) is required by decision makers;
- The test item(s) being tested does not always do what it is expected to do;
- The test item(s) being tested needs to be verified;
- The test item(s) being tested needs to be validated; and/or
- Evaluation of the test item(s) needs to be conducted throughout the software and system development life cycle.

It is generally accepted that it is not possible to create perfect software. It is therefore necessary to test software before it is released to the users to reduce the risk of mistakes in software production having a negative impact when the software is used. It is equally necessary to ensure that testing is performed well.

Mistakes or introduced defects occur and may be largely unavoidable. A mistake or error, made by a human being, causes a defect to appear in the product that the person is working on; (e.g. a requirements specification or a software component). The defect has no impact on the operation of the software if it is not encountered when the software is used. But if a defect is encountered under the right conditions when the product is put to use, it may cause the product to fail to meet the user's legitimate need.. Serious consequences may follow from a user experiencing a software failure; for example, a defect may compromise business reputation, public safety, business economic viability, business or user security, and/or the environment.

Dynamic testing is necessary, but not sufficient to provide reasonable assurance that software will perform as intended. Additional static testing activities, such as peer reviews and static analysis, should be performed in combination with effective dynamic testing activities.

The primary goals of testing are to: provide information about the quality of the test item and any residual risk in relation to how much the test item has been tested; to find defects in the test item prior to its release for use; and to mitigate the risks to the stakeholders of poor product quality.

This information can be used for several purposes; including:

- to improve the test item by having the defects removed;
- to improve management decisions by providing information about quality and risk as the basis for the decisions; and
- to improve the processes in the organization by highlighting processes that allow defects to appear and/or to remain undiscovered where they could have been discovered.

Product quality has many aspects, including conformance to specifications, absence of defects, and fitness in meeting the requirements of the product's users. ISO/IEC 25010, *System and software quality models* defines eight quality characteristics that can be measured or assessed by testing (see 5.5.3).

Software testing should focus on providing information about a software product and finding as many defects as possible, as early as possible in the development process, under given constraints of cost and schedule. Testing considerations include:

- Testing is a process. A process is a set of interrelated or interacting activities that transforms inputs into outputs. The objective of this standard is to present and describe generic testing processes (see ISO/IEC/IEEE 29119-2 Test processes for more details).
- The organizational test process sets and maintains the test policies and test strategies that apply across the organization's projects and functions.
- Testing should be planned, monitored and controlled. ISO/IEC/IEEE 29119-2 Test processes includes a test management process, and can be applied to testing in all development life cycles and the management of exploratory testing.
- Testing processes and sub-processes can be applied to any phase or level of testing (e.g. system testing) or type of testing (e.g. performance testing).
- Testing entails examining a test item.
- Testing can be carried out on a product without executing the product on a computer. This is called static testing in this standard and in many areas of the industry, although other standards (e.g. IEEE 1028) may more specifically call this reviews, walkthroughs or inspections. For static testing this standard acknowledges and identifies the role of the tester in these activities even though they may be performed