BS ISO/IEC/IEEE 29119-4:2015

BSI Standards Publication

Software and systems engineering — Software testing

Part 4: Test techniques

bsi.

...making excellence a habit.™

Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 29119-4:2015

National foreword

This British Standard is the UK implementation of ISO/IEC/IEEE 29119-4:2015. It supersedes BS 7925-2:1998 which is withdrawn.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 2015.
Published by BSI Standards Limited 2015

ISBN 978 0 580 76329 8

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 31 December 2015.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 29119-4:2015

BS ISO/IEC/IEEE 29119-4:2015

INTERNATIONAL STANDARD

ISO/IEC/ IEEE 29119-4

First edition
2015-12-01

Software and systems engineering — Software testing —

Part 4: Test techniques

Ingénierie du logiciel et des systèmes — Essais du logiciel —

Partie 4: Techniques d'essai

Reference number
ISO/IEC/IEEE 29119-4:2015(E)

© ISO/IEC 2015
© IEEE 2015

Full document details: BS ISO/IEC/IEEE 29119-4:2015

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2015
© IEEE 2015
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
Ch. de Blandonnet 8 · CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
copyright@iso.org
www.iso.org

IEC Central Office
3, rue de Varembé
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

Documents
Document Sharing Hub

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Full document details: BS ISO/IEC/IEEE 29119-4:2015

Contents

Page

Foreword ..... **v**

Introduction ..... **vi**

1 Scope **1**

2 Conformance **1**
2.1 Intended Usage 1
2.2 Full Conformance 1
2.3 Tailored Conformance 1

3 Normative References **1**

4 Terms and Definitions **2**

5 Test Design Techniques **4**
5.1 Overview 4
5.2 Specification-Based Test Design Techniques 7
5.2.1 Equivalence Partitioning 7
5.2.2 Classification Tree Method 8
5.2.3 Boundary Value Analysis 9
5.2.4 Syntax Testing 11
5.2.5 Combinatorial Test Design Techniques 12
5.2.6 Decision Table Testing 15
5.2.7 Cause-Effect Graphing 15
5.2.8 State Transition Testing 16
5.2.9 Scenario Testing 17
5.2.10 Random Testing 18
5.3 Structure-Based Test Design Techniques 18
5.3.1 Statement Testing 18
5.3.2 Branch Testing 19
5.3.3 Decision Testing 20
5.3.4 Branch Condition Testing 20
5.3.5 Branch Condition Combination Testing 21
5.3.6 Modified Condition Decision Coverage (MCDC) Testing 21
5.3.7 Data Flow Testing 22
5.4 Experience-Based Test Design Techniques 25
5.4.1 Error Guessing 25

6 Test Coverage Measurement **25**
6.1 Overview 25
6.2 Test Measurement for Specification-Based Test Design Techniques 26
6.2.1 Equivalence Partition Coverage 26
6.2.2 Classification Tree Method Coverage 26
6.2.3 Boundary Value Analysis Coverage 26
6.2.4 Syntax Testing Coverage 26
6.2.5 Combinatorial Test Design Technique Coverage 27
6.2.6 Decision Table Testing Coverage 27
6.2.7 Cause-Effect Graphing Coverage 28
6.2.8 State Transition Testing Coverage 28
6.2.9 Scenario Testing Coverage 28
6.2.10 Random Testing Coverage 28
6.3 Test Measurement for Structure-Based Test Design Techniques 29
6.3.1 Statement Testing Coverage 29
6.3.2 Branch Testing Coverage 29
6.3.3 Decision Testing Coverage 29
6.3.4 Branch Condition Testing Coverage 29
6.3.5 Branch Condition Combination Testing Coverage 29
6.3.6 Modified Condition Decision (MCDC) Testing Coverage 30

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

AVA Documents Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 29119-4:2015

6.3.7 Data Flow Testing Coverage 30
6.4 Test Measurement for Experience-Based Testing Design Techniques 31
6.4.1 Error Guessing Coverage 31

Annex A (informative) **Testing Quality Characteristics** **32**

Annex B (informative) **Guidelines and Examples for the Application of Specification-Based Test Design Techniques** **43**

Annex C (informative) **Guidelines and Examples for the Application of Structure-Based Test Design Techniques** **103**

Annex D (informative) **Guidelines and Examples for the Application of Experience-Based Test Design Techniques** **126**

Annex E (informative) **Guidelines and Examples for the Application of Interchangeable Test Design Techniques** **129**

Annex F (informative) **Test Design Technique Coverage Effectiveness** **133**

Annex G (informative) **ISO/IEC/IEEE 29119-4 and BS 7925-2 Test Design Technique Alignment** **135**

Bibliography **137**

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 7, *Software and Systems Engineering*.

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*

The following parts are under preparation:

— *Part 5: Keyword-driven testing*

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

Introduction

The purpose of this part of ISO/IEC/IEEE 29119 is to provide an International Standard that defines software test design techniques (also known as test case design techniques or test methods) that can be used within the test design and implementation process that is defined in ISO/IEC/IEEE 29119-2. This part of ISO/IEC/IEEE 29119 does not prescribe a process for test design and implementation; instead, it describes a set of techniques that can be used within ISO/IEC/IEEE 29119-2. The intent is to describe a series of techniques that have wide acceptance in the software testing industry.

The test design techniques presented in this part of ISO/IEC/IEEE 29119 can be used to derive test cases that, when executed, generate evidence that test item requirements have been met and/or that defects are present in a test item (i.e. that requirements have not been met). Risk-based testing could be used to determine the set of techniques that are applicable in specific situations (risk-based testing is covered in ISO/IEC/IEEE 29119-1 and ISO/IEC/IEEE 29119-2).

NOTE A "test item" is a work product that is being tested (see ISO/IEC/IEEE 29119-1).

EXAMPLE 1 "Test items" include systems, software items, objects, classes, requirements documents, design specifications, and user guides.

Each technique follows the test design and implementation process that is defined in ISO/IEC/IEEE 29119-2 and shown in Figure 1.

Of the activities in this process, ISO/IEC/IEEE 29119-4 provides guidance on how to implement the following activities in detail for each technique that is described:

— Derive Test Conditions (TD2);

— Derive Test Coverage Items (TD3);

— Derive Test Cases (TD4).

A test condition is a testable aspect of a test item, such as a function, transaction, feature, quality attribute, or structural element identified as a basis for testing. This determination can be achieved by agreeing with stakeholders which attributes are to be tested or by applying one or more test design techniques.

EXAMPLE 2 If a test completion criterion for state transition testing was identified that required coverage of all states then the test conditions could be the states the test item can be in. Other examples of test conditions are equivalence classes and boundaries between them.

Test coverage items are attributes of each test condition that can be covered during testing. A single test condition may be the basis for one or more test coverage items.

EXAMPLE 3 If a specific boundary is identified as a test condition, then the corresponding test coverage items could be the boundary itself and immediately either side of the boundary.

A test case is a set of preconditions, inputs (including actions, where applicable), and expected results, developed to determine whether or not the covered part of the test item has been implemented correctly.

Specific (normative) guidance on how to implement the other activities in the test design & implementation process of ISO/IEC/IEEE 29119-2, including activities TD1 (Identify Feature Sets), TD5 (Assemble Test Sets), and TD6 (Derive Test Procedures), is not included in Clauses 5 or 6 of this part of ISO/IEC/IEEE 29119 because the process is the same for all techniques.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

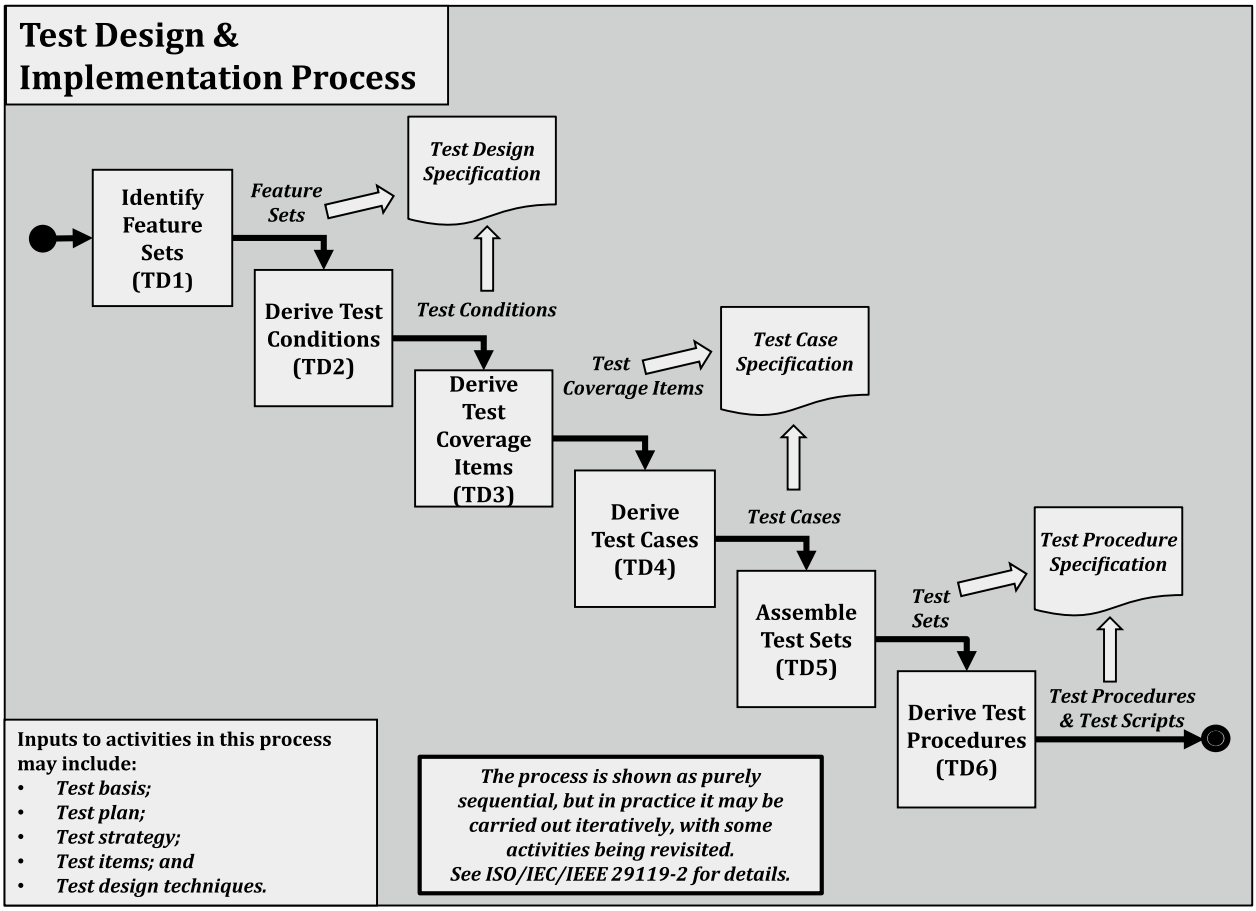


Figure 1 — ISO/IEC/IEEE 29119-2 Test Design and Implementation Process

ISO/IEC/TR 19759 (SWEBOK) defines two types of requirements: functional requirements and quality requirements. ISO/IEC 25010 defines eight quality characteristics (including functionality) that can be used to identify types of testing that may be applicable for testing a specific test item. Annex A provides example mappings of test design techniques that apply to testing quality characteristics defined in ISO/IEC 25010.

Experience-based testing practices like exploratory testing and other test practices such as model-based testing are not defined in this part of ISO/IEC/IEEE 29119 because this part of ISO/IEC/IEEE 29119 only describes techniques for designing test cases. Test practices such as exploratory testing are described in ISO/IEC/IEEE 29119-1.

Templates and examples of test documentation that are produced during the testing process are defined in ISO/IEC/IEEE 29119-3 Test Documentation. The test techniques in this part of ISO/IEC/IEEE 29119 do not describe how test cases should be documented (e.g. they do not include information or guidance on assigning unique identifiers, test case descriptions, priorities, traceability, or pre-conditions). Information on how to document test cases can be found in ISO/IEC/IEEE 29119-3.

This part of ISO/IEC/IEEE 29119 aims to provide stakeholders with the ability to design test cases for the testing of software in any organization.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

Documents
Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 29119-4:2015

INTERNATIONAL STANDARD ISO/IEC/IEEE 29119-4:2015(E)

Software and systems engineering — Software testing —

Part 4: Test techniques

1 Scope

This part of ISO/IEC/IEEE 29119 defines test design techniques that can be used during the test design and implementation process that is defined in ISO/IEC/IEEE 29119-2.

This part of ISO/IEC/IEEE 29119 is intended for, but not limited to, testers, test managers, and developers, particularly those responsible for managing and implementing software testing.

2 Conformance

2.1 Intended Usage

The normative requirements in this part of ISO/IEC/IEEE 29119 are contained in Clauses 5 and 6. It is recognised that particular projects or organizations may not need to use all of the techniques defined by this standard. Therefore, implementation of this standard typically involves selecting a set of techniques suitable for the project or organization. There are two ways that an organization or individual can claim conformance to the provisions of this standard – full conformance or tailored conformance. The organization or individual shall assert whether full or tailored conformance to this standard is claimed.

2.2 Full Conformance

Full conformance is achieved by demonstrating that all of the requirements (i.e. 'shall' statements) of the chosen (non-empty) set of techniques in Clause 5 and/or the corresponding test coverage measurement approaches in Clause 6 have been satisfied.

EXAMPLE An organization could choose to conform only to one technique, such as boundary value analysis. In this scenario, the organization would only be required to provide evidence that they have met the requirements of that one technique in order to claim conformance to this part of ISO/IEC/IEEE 29119.

2.3 Tailored Conformance

Tailored conformance is achieved by demonstrating that the chosen subset of requirements from the chosen (non-empty) set of techniques and/or corresponding test coverage measurement approaches have been satisfied. Where tailoring occurs, justification shall be provided (either directly or by reference) whenever the normative requirements of a technique defined in Clause 5 or measure defined in Clause 6 are not followed completely. All tailoring decisions shall be recorded with their rationale, including the consideration of any applicable risks. Tailoring shall be agreed by the relevant stakeholders.

3 Normative References

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

ISO/IEC/IEEE 29119-1, *Software and systems engineering — Software testing — Part 1: Concepts and definitions*

ISO/IEC/IEEE 29119-2, *Software and systems engineering — Software testing — Part 2: Test processes*

ISO/IEC/IEEE 29119-3, *Software and systems engineering — Software testing — Part 3: Test documentation*

NOTE Other International 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 Use of the terminology in this part of ISO/IEC/IEEE 29119 is for ease of reference and is not mandatory for conformance with the standard. The following terms and definitions are provided to assist with the understanding and readability of this part of ISO/IEC/IEEE 29119. Only terms critical to the understanding of this part of ISO/IEC/IEEE 29119 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 can be referenced for terms not defined in this clause.

4.1
Backus-Naur Form
formal meta-language used for defining the syntax of a language in a textual format

4.2
base choice
see base value

4.3
base value
input parameter value used in 'base choice testing' that is normally selected based on being a representative or typical value for the parameter. Also called base choice

4.4
c-use
see computation data use

4.5
computation data use
use of the value of a variable in any type of statement

4.6
condition
Boolean expression containing no Boolean operators

EXAMPLE "A < B" is a condition but "A and B" is not.

4.7
control flow
sequence in which operations are performed during the execution of a test item

4.8
control flow sub-path
sequence of executable statements within a test item

4.9
data definition
statement where a variable is assigned a value. Also called variable definition

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

4.10
data definition c-use pair
data definition and subsequent computation data use, where the data use uses the value defined in the data definition

4.11
data definition p-use pair
data definition and subsequent predicate data use, where the data use uses the value defined in the data definition

4.12
data definition-use pair
data definition and subsequent data use, where the data use uses the value defined in the data definition

4.13
data use
executable statement where the value of a variable is accessed

4.14
decision outcome
result of a decision (which therefore determines the control flow alternative taken)

4.15
decision rule
combination of conditions (also known as causes) and actions (also known as effects) that produce a specific outcome in decision table testing and cause-effect graphing

4.16
definition-use pair
data definition and subsequent predicate or computational data use, where the data use uses the value defined in the data definition

4.17
definition-use path
control flow sub-path from a variable definition to a predicate-use (p-use) or computational-use (c-use) of that variable

4.18
entry point
point in a test item at which execution of the test item can begin

Note 1 to entry: An entry point is an executable statement within a test item that may be selected by an external process as the starting point for one or more paths through the test item. It is most commonly the first executable statement within the test item.

4.19
executable statement
statement which, when compiled, is translated into object code, which will be executed procedurally when the test item is running and may perform an action on program data

4.20
exit point
last executable statement within a test item

Note 1 to entry: An exit point is a terminal point of a path through a test item, being an executable statement within the test item which either terminates the test item, or returns control to an external process. This is most commonly the last executable statement within the test item.

4.21
p-use
see predicate data use

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

4.22
P-V pair
combination of a test item parameter with a value assigned to that parameter, used as a test condition and coverage item in combinatorial test design techniques

4.23
path
sequence of executable statements of a test item

4.24
predicate
logical expression which evaluates to TRUE or FALSE, normally to direct the execution path in code

4.25
predicate data use
data use associated with the decision outcome of the predicate portion of a decision statement

4.26
sub-path
path that is part of a larger path

4.27
test model
representation of a test item that is used during the test case design process

4.28
variable definition
see data definition

5 Test Design Techniques

5.1 Overview

ISO/IEC/IEEE 29119-4 defines test design techniques for specification-based testing (5.2), structure-based testing (5.3) and experience-based testing (5.4). In specification-based testing, the test basis (e.g. requirements, specifications, models or user needs) is used as the main source of information to design test cases. In structure-based testing, the structure of the test item (e.g. source code or the structure of a model) is used as the primary source of information to design test cases. In experience-based testing, the knowledge and experience of the tester is used as the primary source of information during test case design. For specification-based testing, structure-based testing and experience-based testing, the test basis is used to generate the expected results. These classes of test design techniques are complementary and their combined application typically results in more effective testing.

Although the techniques presented in ISO/IEC/IEEE 29119-4 are classified as structure-based, specification-based or experience-based, in practice some of them can be used interchangeably (e.g. branch testing could be used to design test cases for testing logical paths through the graphical user interface of an Internet-based system). This is demonstrated in Annex E. In addition, although each technique is defined independently of all others, in practice they can be used in combination with other techniques.

EXAMPLE The test coverage items derived by applying equivalence partitioning could be used to populate input parameters of test cases derived using scenario testing.

ISO/IEC/IEEE 29119-4 uses the terms specification-based testing and structure-based testing; these categories of techniques are also known as “black-box testing” and “white-box testing” (or “clear-box testing”) respectively. The terms “black-box” and “white-box” refer to the visibility of the internal structure of the test item. In black-box testing the internal structure of the test item is not visible (hence the black box), whereas for white-box testing the internal structure of the test item is visible. When a technique is applied while utilising a combination of knowledge from the test item’s specification and structure, this is often called “grey-box testing”.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

ISO/IEC/IEEE 29119-4 defines how the generic test design and implementation process steps TD2 (derive test conditions), TD3 (derive test coverage items), and TD4 (derive test cases) from ISO/IEC/IEEE 29119-2 (see Introduction) shall be used by each technique. It does not provide context-specific definitions of the techniques that describe how each technique should be used in all situations. Users of ISO/IEC/IEEE 29119-4 may refer to Annex B, Annex C, Annex D and Annex E for detailed examples that demonstrate how to apply the techniques.

The techniques that are defined in ISO/IEC/IEEE 29119-4 are shown in Figure 2. This set of techniques is not exhaustive. There are techniques that are used by testing practitioners or researchers that are not included in ISO/IEC/IEEE 29119-4.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

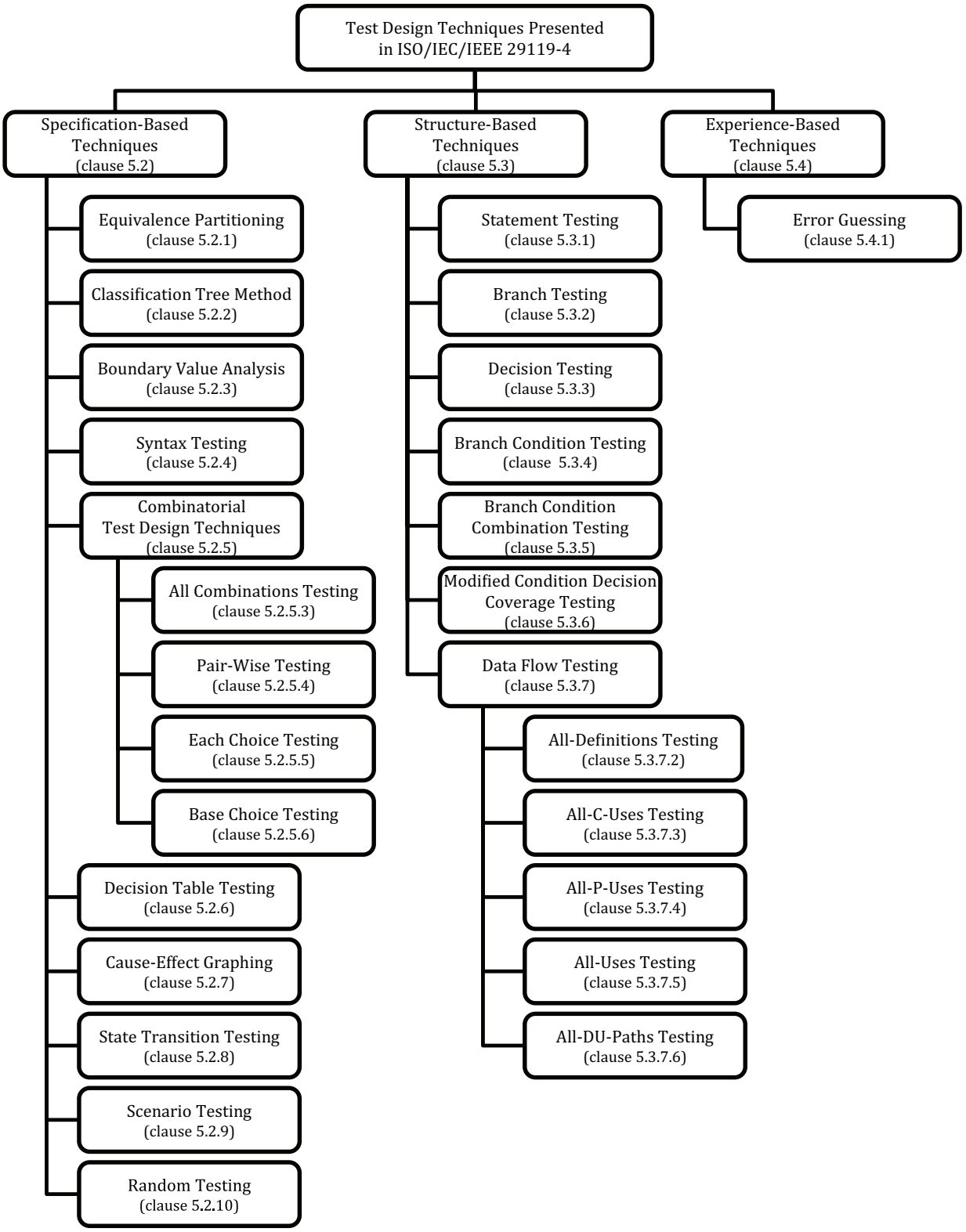


Figure 2 — The set of test design techniques presented in ISO/IEC/IEEE 29119-4

Of the six activities in the test design and implementation process (see Figure 1), test design techniques provide unique and specific guidance on the derivation of test conditions (TD2), test coverage items (TD3) and test cases (TD4). Therefore, each technique is defined in terms of these three activities.

There are varying levels of granularity within steps TD2 (derive test conditions), TD3 (derive test coverage items) and TD4 (derive test cases). Within each technique, the term "model" is used to

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

describe the concept of preparing a logical representation of the test item for the purposes of deriving test conditions in step TD2 (e.g. a control flow model is required for deriving test conditions for all structural techniques). Some situations may require the entire model to be a test condition, whereas in other situations, one part of the model may be a test condition.

EXAMPLE 1 In state transition testing, if there is a requirement to cover all states then the entire state model could be the test condition. Alternatively, if there is a requirement to cover specific transitions between states, then each transition could be a test condition.

In addition, since some techniques share underlying concepts, their definitions contain similar text.

EXAMPLE 2 Both equivalence partitioning and boundary value analysis are based on equivalence classes.

In the test case design step (TD4) of each technique, test cases that are created may be "valid" (i.e. they contain input values that the test item should accept as correct) or "invalid" (i.e. they contain at least one input value that the test item should reject as incorrect, ideally with an appropriate error message). In some techniques, such as equivalence partitioning and boundary value analysis, invalid test cases are usually derived using the "one-to-one" approach as it avoids fault masking by ensuring that each test case only includes one invalid input value, while valid test cases are typically derived using the "minimized" approach, as this reduces the number of test cases required to cover valid test coverage items (see 5.2.1.3 and 5.2.3.3).

NOTE Invalid cases are also known as "negative test cases".

Although the techniques defined in ISO/IEC/IEEE 29119-4 are each described in a separate clause (as if they were mutually exclusive), in practice they could be applied in a blended way.

EXAMPLE 3 Boundary value analysis could be used to select test input values, after which pair-wise testing could be used to design test cases from the test input values. Equivalence partitioning could be used to select the classifications and classes for the classification tree method and then each choice testing could be used to construct test cases from the classes.

The techniques presented in ISO/IEC/IEEE 29119-4 could also be used in conjunction with the test types that are presented in Annex A. For example, equivalence partitioning could be used to identify user groups (test conditions) and representative users (test coverage items) from those groups in test cases that are to be tested during usability testing.

The normative definitions of the techniques are provided in Clause 5. The corresponding normative coverage measures for each technique are presented in Clause 6. This is supported by informative examples of each technique in Annexes B, C, D and E. Although the examples of each technique demonstrate manual application of the technique, in practice, automation can be used to support some types of test design and execution (e.g. statement coverage analyzers can be used to support structure-based testing). Annex A provides examples of how the test design techniques defined in this standard can be applied to testing the quality characteristics that are defined in ISO/IEC 25010.

5.2 Specification-Based Test Design Techniques

5.2.1 Equivalence Partitioning

5.2.1.1 Derive Test Conditions (TD2)

Equivalence partitioning (BS 7925-2:1998; Myers 1979) uses a model of the test item that partitions the inputs and outputs of the test item into equivalence partitions (also called "partitions" or "equivalence classes"), where each equivalence partition shall be defined as a test condition. These equivalence partitions shall be derived from the test basis, where each partition is chosen such that all values within the equivalence partition can reasonably be expected to be treated similarly (i.e. they may be

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

considered "equivalent") by the test item. Equivalence partitions may be derived for both valid and invalid inputs and outputs.

EXAMPLE For a test item expecting lowercase alphabetical characters as (valid) inputs, invalid input equivalence partitions that could be derived include equivalence partitions containing integers, reals, uppercase alphabetical characters, symbols and control characters, depending on the level of rigour required during testing.

NOTE 1 For output equivalence partitions, corresponding input partitions are derived based on the processing described in the test item's specification. Test inputs are then selected from the input partitions.

NOTE 2 Invalid output equivalence partitions typically correspond to any outputs that have not been explicitly specified. As these are not specified their identification often results in equivalence partitions based on the subjectivity of the individual tester. This subjective form of test design may also occur when applying experience-based techniques like error guessing.

NOTE 3 Domain analysis (Beizer 1995) is often classified as a combination of equivalence partitioning and boundary value analysis.

5.2.1.2 Derive Test Coverage Items (TD3)

Each equivalence partition shall be identified as a test coverage item (i.e. for equivalence partitioning the test conditions and test coverage items are the same equivalence partitions).

5.2.1.3 Derive Test Cases (TD4)

Test cases shall be derived to exercise the test coverage items (i.e. the equivalence partitions). The following steps shall be used during test case derivation:

a) Decide on an approach for selecting combinations of test coverage items to be exercised by test cases, where two common approaches are (BS 7925-2:1998; Myers 1979):

   1) one-to-one, in which each test case is derived to cover a specific equivalence partition;

   2) minimized, in which equivalence partitions are covered by test cases such that the minimum number of test cases derived covers all equivalence partitions at least once.

   NOTE Other approaches to selecting combinations of test coverage items to be exercised by test cases are described in 5.2.5 (Combinatorial Test Design Techniques).

b) Select test coverage item(s) for inclusion in the current test case based on the approach chosen in step a);

c) Identify input values to exercise the test coverage items to be covered by the test case and arbitrary valid values for any other input variables required by the test case;

d) Determine the expected result of the test case by applying the input(s) to the test basis;

e) Repeat steps b) to d) until the required level of test coverage is achieved.

5.2.2 Classification Tree Method

5.2.2.1 Derive Test Conditions (TD2)

The classification tree method (Grochtmann and Grimm 1993) uses a model of the test item that partitions the inputs of the test item and represents them graphically in the form of a tree called a classification tree. The test item's inputs are partitioned into "classifications", where each classification consists of a disjoint (non-overlapping) set of "classes" and often sub-classes, and the set of classifications is complete (all classifications of all inputs relevant to the test item domain being modelled have been identified and included). Each classification shall be a test condition. "Classes" that result from decomposing the classifications may be partitioned further into "sub-classes" depending on the level of rigour required in the testing. Classifications and classes may be derived for both valid

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

and invalid input data, depending on the level of test coverage required. The hierarchical relationships between classifications, classes and sub-classes are modelled as a tree, in which the input domain of the test item is placed as the root node, the classifications as branch nodes, and the classes or sub-classes as leaf nodes.

NOTE The process of partitioning in the classification tree method is similar to equivalence partitioning. The key difference is that in the classification tree method, the partitions (which are classifications and classes) must be completely disjoint, whereas in equivalence partitioning, they could overlap depending on how the technique was applied. In addition, the classification tree method also includes the design of a classification tree, which provides a visual representation of the test conditions.

5.2.2.2 Derive Test Coverage Items (TD3)

Test coverage items shall be derived by combining classes using a chosen combination approach.

EXAMPLE Two example approaches for combining classes into test coverage items are:

— minimized, in which classes are included in test coverage items such that the minimum number of test coverage items are derived to cover all classes at least once;

— maximized, in which classes are included in test coverage items such that each possible combination of classes is covered by at least one test coverage item.

NOTE 1 Other approaches to selecting combinations of test coverage items are described in 5.2.5 (Combinatorial Test Design Techniques).

NOTE 2 The test coverage items are often illustrated in a combination table (see Figure B.5 in B.2.2.5).

NOTE 3 The original publication of the classification tree method (Grochtmann and Grimm 1993) used the terms “minimal” and “maximal” instead of “minimized” and “maximized”.

5.2.2.3 Derive Test Cases (TD4)

Test cases shall be derived to exercise the test coverage items. The following steps shall be followed during test case derivation:

a) Based on the combinations of classes created in step TD3, select one combination for inclusion in the current test case that has not already been covered by a test case;

b) Identify input values for any classes that do not already have an assigned value;

c) Determine the expected result of the test case by applying the input(s) to the test basis;

d) Repeat steps a) to c) until the required level of test coverage is achieved.

5.2.3 Boundary Value Analysis

5.2.3.1 Derive Test Conditions (TD2)

Boundary value analysis (BS 7925-2:1998; Myers 1979) uses a model of the test item that partitions the inputs and outputs of the test item into a number of ordered sets and subsets (partitions and sub-partitions) with identifiable boundaries, where each boundary is a test condition. The boundaries shall be derived from the test basis.

EXAMPLE For a partition defined as integers from 1 to 10 inclusive, there are two boundaries, where the lower boundary is 1 and the upper boundary is 10, and these are the test conditions.

NOTE For output boundaries, corresponding input partitions are derived based on the processing described in the test item’s specification. Test inputs are then selected from the input partitions.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

5.2.3.2 Derive Test Coverage Items (TD3)

One of the following two options for the derivation of test coverage items shall be applied:

— two-value boundary testing; or

— three-value boundary testing.

For *two-value boundary testing*, two test coverage items shall be derived for each boundary (test condition) corresponding to values on the boundary and an incremental distance *outside* the boundary of the equivalence partition. This incremental distance shall be defined as the smallest significant value for the data type under consideration.

For *three-value boundary testing*, three test coverage items shall be derived for each boundary (test condition) corresponding to values on the boundary and an incremental distance *each* side of the boundary of the equivalence partition. This incremental distance shall be defined as the smallest significant value for the data type under consideration.

NOTE 1 Some partitions could have only a single boundary identified in the test basis. For example, the numerical partition "age ≥ 70 years" has a lower boundary but not an obvious upper boundary. In some cases a boundary value imposed by the actual implementation can be used as a boundary value, such as the largest value that is accepted by the input field (such decisions should be documented; for example, in the Test Specification documentation).

NOTE 2 Two-value boundary value testing is typically adequate in most situations; however, three-value boundary testing could be required for certain circumstances (e.g. for rigorous testing to check that no errors were made in determining the boundaries of variables in the test item by both testers and developers).

NOTE 3 In both two- and three-value boundary value testing, contiguous partitions (partitions that share a boundary) will result in duplicate test coverage items, in which case it is typical practice to only exercise these duplicated values once. For an example of duplicate boundaries, see B.2.3.4.3.

5.2.3.3 Derive Test Cases (TD4)

Test cases shall be derived to exercise the test coverage items. The following steps shall be used during test case derivation:

a) Decide on an approach for selecting combinations of test coverage items to be exercised by test cases, where two common approaches are (BS 7925-2:1998; Myers 1979):

   1) one-to-one, in which each test case is derived to exercise a specific boundary value;

   2) minimized, in which boundary values are included in test cases such that the minimum number of test cases are derived to cover all boundary values at least once.

   NOTE 1 In minimized boundary value analysis, each test case can cover multiple test coverage items.

   NOTE 2 Other approaches to selecting combinations of test coverage items to be exercised by test cases are described in 5.2.5 (Combinatorial Test Design Techniques).

b) Select test coverage item(s) for inclusion in the current test case based on the approach chosen in step a);

c) Identify arbitrary valid values for any other input variables required by the test case that were not already selected in step b);

d) Determine the expected result of the test case by applying the input(s) to the test basis;

e) Repeat steps b) to d) until the required level of test coverage is achieved.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

5.2.4 Syntax Testing

5.2.4.1 Derive Test Conditions (TD2)

Syntax testing (Beizer 1995; Burnstein 2003) uses a formal model of the inputs to a test item as the basis for test design. This syntax model is represented as a number of rules, where each rule defines the format of an input parameter in terms of "sequences of", "iterations of", or "selections between" elements in the syntax. The syntax may be represented in a textual or diagrammatic format. The test condition in syntax testing shall be the whole or partial model of the inputs to the test item.

EXAMPLE 1 Backus-Naur Form is a formal meta-language that can be used to define the syntax for a test item in a textual format.

EXAMPLE 2 An abstract syntax tree can be used to represent formal syntax diagrammatically.

5.2.4.2 Derive Test Coverage Items (TD3)

In syntax testing the test coverage items are derived based on two goals: positive testing, in which test coverage items are derived to cover the valid syntax in various ways, and negative testing, in which test coverage items are derived to deliberately violate the rules of the syntax. Test coverage items for positive testing shall be "options" of the defined syntax, whereas test coverage items for negative testing shall be "mutations" of the defined syntax.

The following guidelines may be used to derive "options" (although alternative guidelines may be used where appropriate):

— whenever selection is mandated by the syntax, an "option" is derived for each alternative provided for that selection;

EXAMPLE 1 For the input parameter "colour = Blue | Red | Green" (where | represents the "OR" Boolean operator), three options "Blue", "Red" and "Green" will be derived as test coverage items.

— whenever an iteration is mandated by the syntax, at least two "options" are derived for the iteration; one with the minimum number of repetitions and the other with more than the minimum number of repetitions;

EXAMPLE 2 For the input parameter "letter = [A – Z | a – z]$^{+}$" (where "$^{+}$" represents "one or more"), two options, "one letter" and "more than one letter", will be derived as test coverage items.

— whenever iteration is mandated with a maximum number of repetitions, at least two "options" are derived for the iteration; one with the maximum number of repetitions and the other with more than the maximum number of repetitions.

EXAMPLE 3 For the input parameter "letter = [A – Z | a – z]^{100}" (where "^{100}" represents the fact that a letter can be chosen up to 100 times), two options "100 letters" and "more than 100 letters" will be derived as test coverage items.

The following guideline may be used to derive "mutations" (although alternative guidelines may be used where appropriate):

— for any input, the defined syntax may be mutated to derive invalid inputs ("mutations").

EXAMPLE 4 For the input parameter "colour = Blue | Red | Green", one "mutation" could be to introduce an invalid value for the parameter as the test coverage item, such as selecting the value "Yellow", which does not appear in the input parameter list. Other example mutations are provided in B.2.4.5.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

5.2.4.3 Derive Test Cases (TD4)

Test cases for syntax testing shall be derived to cover the chosen options and mutations. The following steps shall be used during test case derivation:

a) Decide on an approach for selecting combinations of test coverage items to be exercised by test cases, where two common approaches are:

   1) one-to-one, in which each test case is derived to exercise a specific option, iteration and/or mutation; or

   2) minimized, in which options, iterations and/or mutations are included in test cases such that the minimum number of test cases are derived to cover all options, iterations and/or mutations at least once.

   NOTE Other approaches to selecting combinations of test coverage items to be exercised by test cases are described in 5.2.5 (Combinatorial Test Design Techniques).

b) Select test coverage item(s) for inclusion in the current test case;

c) Identify input values to exercise the test coverage item(s) to be covered by the test case and arbitrary valid values for any other input variables required by the test case;

d) Determine the expected result of the test case by applying the input(s) to the test basis;

e) Repeat steps b) to d) until all derived options, iterations and/or mutations are exercised.

5.2.5 Combinatorial Test Design Techniques

5.2.5.1 Overview

Combinatorial test design techniques are used to systematically derive a meaningful and manageable subset of test cases that cover the test conditions and test coverage items that are derivable during testing. The combinations of interest are defined in terms of test item parameters and the values these parameters can take. Where numerous parameters (each with numerous discrete values) must interact, this technique enables a significant reduction in the number of test cases required without compromising functional coverage.

5.2.5.2 Derive Test Conditions (TD2)

The test item parameters represent particular aspects of the test item that are relevant to the testing, and often correspond to the input parameters to the test item, but other aspects may also be used.

EXAMPLE 1 In configuration testing, the parameters could be various environment factors, such as operating system and browser.

Each test item parameter can take on various values. For use in this technique the set of values needs to be finite and manageable. Some test item parameters may be naturally constrained to only take on a small set of possible values while other test item parameters may be far less constrained.

EXAMPLE 2 A test item parameter that is naturally constrained to a small number of values is a "day_of_week" parameter that can only take on seven specific values "[Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday]".

EXAMPLE 3 A test item parameter that is far less constrained is a parameter consisting of any Real number, which exists along a potentially infinite number line.

For unconstrained test item parameters, it may first be necessary to apply other test design techniques, such as equivalence partitioning or boundary value analysis, to reduce a large set of possible values for a parameter to a manageable subset.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

The test conditions for combinatorial testing are the same for all the combinatorial test design techniques; each test condition shall be a chosen test item parameter (P) taking on a specific value (V), resulting in a P-V pair.

5.2.5.3 All Combinations Testing

5.2.5.3.1 Derive Test Coverage Items (TD3)

In 'all combinations' testing (Grindal, Offutt and Andler 2005) the test coverage items shall be members of the set of all unique combinations of P-V pairs, such that each parameter is included at least once in the set.

5.2.5.3.2 Derive Test Cases (TD4)

Test cases shall be derived in which each test case exercises one unique combination of P-V pairs. The following steps shall be used during test case derivation:

a) Select test coverage item(s) for inclusion in the current test case that have not already been covered by a test case;

b) Determine the expected result of the test case by applying the input(s) to the test basis;

c) Repeat steps a) and b) until the required level of test coverage is achieved.

NOTE The minimum number of test cases required to achieve 100% all combinations testing coverage corresponds to the product of the number of P-V pairs for each test item parameter.

5.2.5.4 Pair-wise Testing

5.2.5.4.1 Derive Test Coverage Items (TD3)

In pair-wise testing (Grindal, Offutt and Andler 2005), the test coverage items shall be unique pairs of P-V pairs, where each P-V pair within the pair is for a different test item parameter. Instead of all possible combinations of the parameters (as was required for all combinations testing), this technique covers all possible pairs of the selected values within the total set, thereby efficiently exercising the test item with fewer test cases. Pair-wise testing is also known as "all pairs" testing.

5.2.5.4.2 Derive Test Cases (TD4)

Having first identified the P-V pairs, test cases shall be derived to exercise pairs of P-V pairs, where each test case exercises one or more unique pairs. The following steps shall be used during test case derivation:

a) Select test coverage item(s) for inclusion in the current test case, where each pair of P-V pairs covers a different pair of parameter values that have not yet been included in a test case;

b) Identify arbitrary valid values for any other parameters present in the test case;

c) Determine the expected result of the test case by applying the input(s) to the test basis;

d) Repeat steps a) to c) until all unique pairs of P-V pairs have been exercised.

The minimum number of test cases required to achieve 100% pair-wise testing is not easily calculated. A near-optimal set may be considered acceptable and may be calculated using one of the following three options:

— manually determine a near-optimal set using an algorithm;

— use an automated tool (implementing an algorithm) to determine a near-optimal set; or

— use orthogonal arrays (Mandl 1985) to determine a near-optimal set.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

5.2.5.5 Each Choice Testing

5.2.5.5.1 Derive Test Coverage Items (TD3)

In each choice (or 1-wise) testing (Grindal, Offutt and Andler 2005), the test coverage items shall be members of the set of P-V pairs such that each parameter value is included at least once in the set.

5.2.5.5.2 Derive Test Cases (TD4)

Test cases shall be derived to exercise P-V pairs, where each test case exercises one or more P-V pairs that have not been previously included in a test case. The following steps shall be used during test case derivation:

a) Select test coverage item(s) for inclusion in the current test case, where at least one selected test coverage item has not been included in a prior test case;

b) Identify arbitrary valid values for any other parameters present in the test case;

c) Determine the expected result of the test case by applying the input(s) to the test basis;

d) Repeat steps a) to c) until the required level of test coverage is achieved.

The minimum number of test cases required to achieve 100% each choice testing corresponds to the maximum number of values any one of the test item parameters can take.

5.2.5.6 Base Choice Testing

5.2.5.6.1 Derive Test Coverage Items (TD3)

In base choice testing (Grindal, Offutt and Andler 2005), the test coverage items shall be sets of P-V pairs for each of the input parameters, where all parameters except one are set to their "base" value and the final parameter is set to one of its other valid values.

NOTE There are a number of approaches for choosing the base values for each parameter. For example, they can be chosen from the operational profile, from the typical path in scenario testing, from the test coverage items that are derived during equivalence partitioning or from the default (most frequently used) values for the parameter.

5.2.5.6.2 Derive Test Cases (TD4)

Having first identified the P-V pairs, base choices for each of the parameters shall be chosen. Test cases shall be derived by setting all but one parameter to its base choice and then setting the final parameter to a valid value until the required level of test coverage of PV-pairs is achieved. The following steps shall be used during test case derivation:

a) Derive the base-choice test case by setting each parameter to its "base" value;

b) Create a new test case by setting one parameter to a valid (non-base choice) value while keeping the remaining set of parameters to their base choice values;

c) Determine the expected result of the new test case by applying the inputs to the test basis;

d) Repeat steps b) and c) until the required level of test coverage is achieved.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

5.2.6 Decision Table Testing

5.2.6.1 Derive Test Conditions (TD2)

Decision table testing (BS 7925-2:1998, Myers 1979) uses a model of the logical relationships (decision rules) between conditions (causes) and actions (effects) for the test item in the form of a decision table, where:

— each Boolean condition defines a pair of input equivalence partitions for the test item, one corresponding to the "true" case, and one to the "false" case;

— each action is an expected outcome or a combination of outcomes for the test item expressed as a Boolean;

— a set of decision rules defines the required relationships between conditions and actions.

The test conditions shall be the conditions and actions.

NOTE If conditions consist of multiple values rather than simple Booleans, this results in an "extended entry" decision table, the testing of which can be handled by equivalence partitioning.

5.2.6.2 Derive Test Coverage Items (TD3)

In decision table testing, each decision rule, which defines the relationship between a unique combination of the test item's conditions and actions, is a test coverage item.

5.2.6.3 Derive Test Cases (TD4)

Test cases shall be derived to exercise the decision rules (test coverage items), where each test case defines the relationship between the inputs and outputs, and each decision rule corresponds to a unique combination of Boolean conditions. The following steps shall be used during test case derivation:

a) Select test coverage item(s) from the decision table for implementation as a test case;

b) Identify input values to satisfy the input condition(s) of the decision rule(s) to be covered by the test case and arbitrary valid values for any other input variables required to execute the test case;

c) Determine the expected result of the test case by applying the input(s) to the decision table;

d) Repeat steps a) to c) until the required level of test coverage is achieved.

NOTE If the decision table contains dependent input conditions then this could result in infeasible combinations (e.g. "age less than 18" and "age greater than 65" both set to true). In this situation such infeasible decision rules should be identified and documented and are not used to derive test cases.

5.2.7 Cause-Effect Graphing

5.2.7.1 Derive Test Conditions (TD2)

Cause-effect graphing (BS 7925-2:1998, Myers 1979, Nursimulu and Probert 1995) uses a model of the logical relationships (decision rules) between causes (e.g. inputs) and effects (e.g. outputs) for the test item in the form of a cause-effect graph, where:

— each Boolean cause defines a pair of input equivalence partitions for the test item, one corresponding to the "true" case, and one to the "false" case; and

— each effect defines an expected output condition or combination of output conditions for the test item expressed as a Boolean.

The test conditions shall be the causes and effects.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

The cause-effect graph models logical relationships between causes and effects as a Boolean logic network weighted by Boolean operators, and can optionally model syntactic and semantic constraints over relationships between causes and relationships between effects (see Figures B.11 and B.12 in Annex B.2.7.4).

5.2.7.2 Derive Test Coverage Items (TD3)

In cause-effect graphing, each decision rule, which defines the relationship between a unique combination of the test item's causes and effects, is a test coverage item.

5.2.7.3 Derive Test Cases (TD4)

Test cases shall be derived to exercise the test coverage items. A corresponding decision table may be produced from the cause-effect graph and used to derive the test cases. The following steps shall be used during test case derivation:

a) Select test coverage item(s) to be implemented in the current test case;

b) Identify input values to exercise the test coverage item(s) to be covered by the test case and arbitrary valid values for any other input variables required to execute the test case;

c) Determine the expected result of the test case by applying the input(s) to the cause-effect graph and/or decision table;

d) Repeat steps a) to c) until the required level of test coverage is achieved.

5.2.8 State Transition Testing

5.2.8.1 Derive Test Conditions (TD2)

State transition testing (BS 7925-2:1998, Copeland 2004) uses a model of the states the test item may occupy, the transitions between states, the events which cause transitions and the actions that may result from the transitions. The states of the model shall be discrete, identifiable and finite in number. An individual transition may be constrained by an event guard, which defines a set of conditions that must be true when the event occurs, in order for the transition to occur. In state transition testing, the test conditions may be all states of the state model, all transitions of the state model or the entire state model, depending on the coverage requirements of testing. The model may be represented as a state transition diagram or a state table (although other representations may also be used).

5.2.8.2 Derive Test Coverage Items (TD3)

In state transition testing, test coverage items will change depending on the chosen test completion criterion and test design approach. Possible test completion criteria include but are not limited to the following:

— states, in which test coverage items shall be derived to enable all states in the state model to be "visited";

— single transitions (0-switch coverage), in which test coverage items shall be derived to cover valid single transitions in the state model;

— all transitions, in which test coverage items shall be derived to cover *both* valid transitions in the state model and "invalid" transitions (transitions from states initiated by events in the state model for which no valid transition is specified);

— multiple transitions (N-switch coverage), in which test coverage items shall be derived to cover valid sequences of N+1 transitions in the state model.

NOTE "1-switch" coverage is a popular variant of "N-switch" coverage that requires pairs of transitions to be exercised.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

5.2.8.3 Derive Test Cases (TD4)

Test cases for state transition testing shall be derived to exercise the test coverage items. The following steps shall be used during test case derivation:

a) Select test coverage item(s) for inclusion in the current test case;

b) Identify input values to exercise the test coverage item(s) to be covered by the test case;

c) Determine the expected result of the test case by applying the input(s) to the test basis (the expected result may be defined in terms of outputs and the states visited as described in the state model);

d) Repeat steps a) to c) until the required level of test coverage is achieved.

5.2.9 Scenario Testing

5.2.9.1 Derive Test Conditions (TD2)

Scenario testing (Desikan and Ramesh 2007) uses a model of the sequences of interactions between the test item and other systems (in this context users are often considered to be other systems) for the purpose of testing usage flows involving the test item. Test conditions shall either be one sequence of interactions (i.e. one scenario) or all sequences of interactions (i.e. all scenarios).

In scenario testing, this step shall include identification of:

— the "main" scenario which is the typical sequence of actions that are expected of the test item or an arbitrary choice when no typical sequence of actions is known; and

— "alternative" scenarios that represent alternative (non-main) scenarios that may be taken through the test item.

NOTE 1 Alternative scenarios can include abnormal use, extreme or stress conditions and exceptions.

NOTE 2 Scenario testing is typically used for conducting "end-to-end testing" during functional testing, such as during system testing or acceptance testing.

One common form of scenario testing called use case testing (Bath 2008; Hass 2008) utilises a use case model of the test item that describes how the test item interacts with one or more actors for the purpose of testing sequences of interactions (i.e. scenarios) involving the test item.

NOTE 3 In use case testing, the use case model must be of a type that describes how various actions are performed by the test item as a result of various triggers from the actors. An actor can be a user or another system.

NOTE 4 Transaction flow testing (Beizer 1995) is often classified as a type of scenario testing.

5.2.9.2 Derive Test Coverage Items (TD3)

The test coverage items shall be the main and alternative scenarios (i.e. the test coverage items are the same as the test conditions). Therefore, no further action is required at this step for this technique.

5.2.9.3 Derive Test Cases (TD4)

Test cases for scenario testing shall be derived by covering each scenario (test coverage item) with at least one test case. The following steps shall be used during test case derivation:

a) Select test coverage item(s) to exercise in the current test case;

b) Identify input values to exercise the test coverage item(s) covered by the test case;

c) Determine the expected result of the test case by applying the input(s) to the test basis;

d) Repeat steps a) to c) until the required level of test coverage is achieved.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

5.2.10 Random Testing

5.2.10.1 Derive Test Conditions (TD2)

Random testing (BS 7925-2:1998; Craig and Jaskiel 2002; Kaner 1988) uses a model of the input domain of the test item that defines the set of all possible input values. An input distribution for the generation of random input values shall be chosen. The entire input domain shall be the test condition for random testing.

EXAMPLE Types of input distributions include the normal distribution, uniform distribution and operational profile.

5.2.10.2 Derive Test Coverage Items (TD3)

There are no recognised test coverage items for random testing.

5.2.10.3 Derive Test Cases (TD4)

Test cases for random testing shall be chosen by randomly selecting input values from the input domain of the test item (or pseudo-randomly if using a tool) according to the chosen input distribution. The following steps shall be used during test case derivation:

a) Select an input distribution for the selection of test inputs;

b) Generate random values for the test inputs based on the input distribution chosen in step a);

c) Determine the expected result of the test case by applying the input(s) to the test basis;

d) Repeat steps b) and c) until the required testing has been completed.

NOTE 1 The required testing can be defined in terms of a number of tests executed, an amount of time spent testing or some other measure of completion.

NOTE 2 Step b) is normally automated.

5.3 Structure-Based Test Design Techniques

5.3.1 Statement Testing

5.3.1.1 Derive Test Conditions (TD2)

A model of the source code of the test item which identifies statements as either executable or non-executable shall be derived (BS 7925-2:1998, Myers 1979). Each executable statement shall be a test condition.

NOTE The identification of non-executable statements could be carried out during step TD4, using an automated tool.

5.3.1.2 Derive Test Coverage Items (TD3)

Each executable statement shall be a test coverage item (i.e. the test coverage items are the same as the test conditions). Therefore, no further action is required at this step for this technique.

5.3.1.3 Derive Test Cases (TD4)

The following steps shall be followed during test case derivation:

a) Identify control flow sub-path(s) that reach one or more test coverage items that have not yet been executed during testing;

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

b) Determine the test inputs that will cause the control flow sub-path(s) to be exercised;

c) Determine the expected result from exercising the control flow sub-path(s) by applying the corresponding test inputs to the test basis;

d) Repeat steps a) to c) until the required level of test coverage is achieved.

5.3.2 Branch Testing

5.3.2.1 Derive Test Conditions (TD2)

A control flow model that identifies branches in the control flow of the test item shall be derived (BS 7925-2:1998; 4.7.1). Each branch in the control flow model shall be a test condition.

A branch is:

— a conditional transfer of control from any node in the control flow model to any other node; or

— an explicit unconditional transfer of control from any node to any other node in the control flow model; or

— when a test item has more than one entry point, a transfer of control to an entry point of the test item.

NOTE 1 Complete branch testing covering 100% of all branches requires all arcs (links or edges) in the control flow graph to be tested, including sequential statements between an entry and exit point that contains no decisions.

NOTE 2 Branch testing can require testing of both conditional and unconditional branches, including entry and exit points to a test item, depending on the level of test coverage required.

NOTE 3 Function and method calls are not identified as separate test conditions in branch testing.

5.3.2.2 Derive Test Coverage Items (TD3)

Each branch in the control flow model shall be a test coverage item (i.e. the test coverage items are the same as the test conditions). Therefore, no further action is required at this step for this technique.

5.3.2.3 Derive Test Cases (TD4)

The following steps shall be followed during test case derivation:

a) Identify control flow sub-path(s) that reach one or more test coverage items that have not yet been executed during testing;

b) Determine the test inputs that will cause the control flow sub-path(s) to be exercised;

c) Determine the expected result from exercising the control flow sub-path(s) by applying the test inputs to the test basis;

d) Repeat steps a) to c) until the required level of test coverage is achieved.

NOTE 1 If there are no branches in the test item then a single test condition, test coverage item and test case is still required.

NOTE 2 If there are multiple entry points to the test item, sufficient test cases will be required to cover each entry point.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

5.3.3 Decision Testing

5.3.3.1 Derive Test Conditions (TD2)

A control flow model of the test item that identifies decisions shall be derived. Decisions are points in the test item where two or more possible outcomes (and hence sub-paths) may be taken by the control flow (BS 7925-2:1998, Myers 1979). Typical decisions are used for simple selections (e.g. if-then-else in source code), to decide when to exit loops (e.g. while-loop in source code), and in case (switch) statements (e.g. case-1-2-3-...-N in source code). In decision testing, each decision in the control flow model shall be a test condition.

5.3.3.2 Derive Test Coverage Items (TD3)

The decision outcomes from each decision shall be identified as test coverage items.

5.3.3.3 Derive Test Cases (TD4)

The following steps shall be followed during test case derivation:

a) Identify control flow sub-path(s) that reach one or more test coverage items that have not yet been executed during testing;

b) Determine the test inputs that will cause the control flow sub-path(s) to be exercised;

c) Determine the expected result from exercising the control flow sub-path(s) by applying the test inputs to the test basis;

d) Repeat steps a) to c) until the required level of test coverage is achieved.

NOTE If there are no decisions in the test item then a single test condition, test coverage item and test case is still required.

5.3.4 Branch Condition Testing

5.3.4.1 Derive Test Conditions (TD2)

A control flow model of the test item that identifies decisions and conditions within decisions shall be derived. Decisions are points in the test item where two or more possible outcomes (and hence sub-paths) may be taken by the control flow (BS 7925-2:1998, Myers 1979). Typical decisions are used for simple selections (e.g. if-then-else in source code), for deciding when to exit loops (e.g. while-loop in source code), and for case statements (e.g. case-1-2-3-...-N in source code, also referred to as switch statements). In branch condition testing (BS 7925-2:1998), each decision shall be a test condition.

EXAMPLE In program source code the decision statement “if A OR B AND C then” is a test condition that contains three conditions related by logical operators.

5.3.4.2 Derive Test Coverage Items (TD3)

In branch condition testing, all Boolean values (true/false) of the condition(s) within decisions shall be identified as test coverage items. The decision outcomes from each decision shall also be identified as test coverage items.

5.3.4.3 Derive Test Cases (TD4)

The following steps shall be followed during test case derivation:

a) Identify control flow sub-path(s) that cover one or more test coverage items that have not yet been executed during testing;

b) Determine the test inputs that will cause the control flow sub-path(s) to be exercised;

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

c) Identify a subset of test inputs from step b) to cover the Boolean values of conditions within the decision and the decision outcomes;

d) Determine the expected result from exercising the control flow sub-path(s) by applying the test inputs to the test basis;

e) Repeat steps a) to d) until the required level of test coverage is achieved.

NOTE If there are no decisions in the test item then a single test condition, test coverage item and test case is still required.

5.3.5 Branch Condition Combination Testing

5.3.5.1 Derive Test Conditions (TD2)

A control flow model of the test item which identifies decisions and conditions shall be derived. In branch condition combination testing (BS 7925-2:1998), each decision shall be a test condition.

EXAMPLE In program source code the decision statement "if A OR B AND C then" is a test condition that contains three conditions related by logical operators.

5.3.5.2 Derive Test Coverage Items (TD3)

Each unique feasible combination of Boolean values of conditions within each decision shall be identified as a test coverage item (BS 7925-2:1998, Myers 1979). This includes simple decisions, where combinations consist of two individual Boolean outcomes of a single condition within a decision.

5.3.5.3 Derive Test Cases (TD4)

The following steps shall be followed during test case derivation:

a) Identify control flow sub-path(s) that reach one or more test coverage items that have not yet been executed during testing;

b) Determine the set of test inputs that will cause the identified control flow sub-path(s) to be exercised;

c) Identify a subset of test inputs from step b) to cover the selected combination of Boolean values of conditions within the decision;

d) Determine the expected result by applying the selected test inputs to the test basis;

e) Repeat steps a) to d) until the required level of test coverage is achieved.

NOTE If there are no decisions in the test item then a single test condition, test coverage item and test case is still required.

5.3.6 Modified Condition Decision Coverage (MCDC) Testing

5.3.6.1 Derive Test Conditions (TD2)

A control flow model of the test item that identifies decisions and conditions shall be derived. In modified condition decision coverage (MCDC) testing (BS 7925-2:1998), each decision shall be a test condition.

EXAMPLE In program source code the decision statement "if A OR B AND C then" is a test condition that contains three conditions related by logical operators.

5.3.6.2 Derive Test Coverage Items (TD3)

Each unique feasible combination of individual Boolean values of conditions within a decision that allows a single Boolean condition to independently affect the outcome of the decision shall be identified

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

as a test coverage item. Whether a condition independently affects the outcome of a decision is shown by varying only that condition while holding all other possible conditions to fixed states.

NOTE This includes simple decisions, where combinations consist of two individual Boolean outcomes of a single condition within a decision.

5.3.6.3 Derive Test Cases (TD4)

The following steps shall be followed during test case derivation:

a) Identify control flow sub-path(s) that reach one or more test coverage items that have not yet been executed during testing;

b) Determine the set of test inputs that will cause the identified control flow sub-path(s) to be exercised;

c) Identify a subset of test inputs from step b) to cover the selected combinations of individual Boolean values of conditions within the decision that allow a single Boolean condition to independently affect the outcome;

d) Determine the expected result by applying the selected test inputs to the test basis;

e) Repeat steps a) to d) until the required level of test coverage is achieved.

NOTE If there are no decisions in the test item then a single test condition, test coverage item and test case is still required.

5.3.7 Data Flow Testing

5.3.7.1 Derive Test Conditions (TD2)

In data flow testing (Burnstein 2003), a model of the test item shall be derived that identifies control flow sub-paths through the test item, within which each definition of a given variable is linked to subsequent use(s) of the same variable and within which there has been no intervening redefinition of the variable's value.

"Definitions" are where a variable is possibly given a new value (sometimes a definition will result in a variable retaining the same value as it had before). A "use" is an occurrence of a variable in which the variable is not given a new value; "uses" can be further distinguished as either "p-uses" (predicate-use) or "c-uses" (computation-use). A p-use denotes the use of a variable in determining the outcome of a condition (predicate) within a decision, such as a while-loop, if- then- else, etc. A c-use occurs when a variable is used as an input to the computation of the definition of any variable or of an output.

In data flow testing, each definition-use pair for a variable in the test item is a test condition.

There are a number of forms of data flow testing, which are all based on the same test conditions. The five forms defined in this part of ISO/IEC/IEEE 29119 are: all-definitions testing, all-c-uses testing, all-p-uses testing, all-uses testing, and all-du-paths testing.

5.3.7.2 All-Definitions Testing

5.3.7.2.1 Derive Test Coverage Items (TD3)

The control flow sub-paths from each variable definition to some use (either p-use or c-use) of that definition shall be identified as test coverage items. Each sub-path is known as a "definition-use" path. "All-definitions" testing requires that at least one definition-free sub-path (with relation to a specific variable) from the definition to one of its c-uses or p-uses will have been covered for all variable definitions.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

5.3.7.2.2 Derive Test Cases (TD4)

The following steps shall be followed during test case derivation:

a) Identify definitions that have not yet been executed during testing;

b) Determine the test inputs that will cause the control flow sub-path(s) from the identified definitions to be exercised;

c) Determine the expected result from exercising the control flow sub-path(s) by applying the test inputs to the test basis;

d) Repeat steps a) to c) until the required level of test coverage is achieved.

NOTE In practice, these steps could require automation.

5.3.7.3 All-C-Uses Testing

5.3.7.3.1 Derive Test Coverage Items (TD3)

The control flow sub-paths from each variable definition to each c-use of that definition shall be identified as test coverage items. "All-c-uses" testing requires that at least one of the definition-free sub-paths (with relation to a specific variable) from the definition to each one of its c-uses will have been covered for all relevant variable definitions.

5.3.7.3.2 Derive Test Cases (TD4)

The following steps shall be followed during test case derivation:

a) Identify control flow sub-path(s) from variable definitions to a subsequent c-use of that definition (with no intervening definitions) that have not yet been executed during testing;

b) Determine the test inputs that will cause the control flow sub-path(s) to be exercised;

c) Determine the expected result from exercising the control flow sub-path(s) by applying the test inputs to the test basis;

d) Repeat steps a) to c) until the required level of test coverage is achieved.

5.3.7.4 All-P-Uses Testing

5.3.7.4.1 Derive Test Coverage Items (TD3)

The control flow sub-paths from each variable definition to each p-use of that definition shall be identified as test coverage items. "All-p-uses" testing requires that at least one of the definition-free sub-paths (with relation to a specific variable) from the definition to each one of its p-uses will have been covered for all relevant variable definitions.

5.3.7.4.2 Derive Test Cases (TD4)

The following steps shall be followed during test case derivation:

a) Identify control flow sub-path(s) from variable definitions to subsequent p-use of that definition (with no intervening definitions) that have not yet been executed during testing;

b) Determine the test inputs that will cause the control flow sub-path(s) to be exercised;

c) Determine the expected result from exercising the control flow sub-path(s) by applying the test inputs to the test basis;

d) Repeat steps a) to c) until the required level of test coverage is achieved.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

5.3.7.5 All-Uses Testing

5.3.7.5.1 Derive Test Coverage Items (TD3)

The control flow sub-paths from each variable definition to every use (both p-use and c-use) of that definition shall be identified as test coverage items. "All-uses" testing requires that at least one sub-path from each variable definition to each one of its uses that can be reached (without an intervening definition of the variable) is covered.

5.3.7.5.2 Derive Test Cases (TD4)

The following steps shall be followed during test case derivation:

a) Identify control flow sub-path(s) from variable definitions to a subsequent p-use or c-use of that definition that have not yet been executed during testing;

b) Determine the test inputs that will cause the control flow sub-path(s) to be exercised;

c) Determine the expected result from exercising the control flow sub-path(s) by applying the test inputs to the test basis;

d) Repeat steps a) to c) until the required level of test coverage is achieved.

5.3.7.6 All-DU-Paths Testing

5.3.7.6.1 Derive Test Coverage Items (TD3)

The control flow sub-paths from each variable definition to every use (both p-use and c-use) of that definition shall be identified as test coverage items. "All-du-paths" testing requires that all sub-paths from each variable definition to each of its uses that can be reached (without an intervening definition for the variable) is covered.

NOTE All-du-paths testing requires *all* loop-free sub-paths from a variable definition to its use be tested to attempt to achieve 100% test item coverage. This differs from All-uses testing, which requires only *one* path from each variable definition to its use to be tested to attempt to achieve 100% test item coverage.

5.3.7.6.2 Derive Test Cases (TD4)

The following steps shall be followed during test case derivation:

a) Identify control flow sub-path(s) from variable definitions to every subsequent p-use and c-use of that definition that have not yet been executed during testing;

b) Determine the test inputs that will cause the control flow sub-path(s) to be exercised;

c) Determine the expected result from exercising the control flow sub-path(s) by applying the test inputs to the test basis;

d) Repeat steps a) to c) until the required level of test coverage is achieved.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

5.4 Experience-Based Test Design Techniques

5.4.1 Error Guessing

5.4.1.1 Derive Test Conditions (TD2)

Error guessing (BS 7925-2:1998, Myers 1979) involves the design of a checklist of defect types that may exist in the test item, allowing the tester to identify inputs to the test item that may cause failures, if those defects exist in the test item. Each defect type shall be a test condition.

NOTE The checklist of defect types could be derived by various means, such as taxonomies of known errors, information contained in incident management systems, from a tester's knowledge, experience and/or understanding of the test item(s) and/or similar test items or from the knowledge of other stakeholders (e.g. system users or programmers).

5.4.1.2 Derive Test Coverage Items (TD3)

There are no recognised test coverage items for error guessing.

5.4.1.3 Derive Test Cases (TD4)

Test cases for error guessing are typically derived by selecting a defect type from a checklist of defect types that are to be covered and deriving test cases that could detect that defect type in the test item, if it existed. The following steps shall be used during test case derivation:

a) Select a defect type(s) for coverage by the current test case;

b) Identify input values that could be expected to cause a failure corresponding to the selected defect type(s);

c) Determine the expected result of the test case by applying the input(s) to the test basis;

d) Repeat steps a) to c) until the required testing has been completed.

6 Test Coverage Measurement

6.1 Overview

The coverage measures defined in this part of ISO/IEC/IEEE 29119 are based on differing degrees of coverage that are achievable by test design techniques. Coverage levels can range from 0% to 100%. In each coverage calculation, a number of test coverage items may be infeasible. A test coverage item shall be defined to be infeasible if it can be shown to not be executable or impossible to be covered by a test case. The coverage calculation shall be defined as either counting or discounting infeasible items; this choice will typically be recorded in the Test Plan (defined in ISO/IEC/IEEE 29119-3). If a test coverage item is discounted, justification for its infeasibility will typically be recorded in a test report. In each coverage calculation, if there are no test coverage items of a given type in a test item, 100% coverage for that type of coverage will be defined as being inapplicable for that test item.

When calculating coverage for any technique, the following formula shall be used:

$$Coverage = \left(\frac{N}{T} \times 100 \right) \%$$

where:

— Coverage is the coverage achieved by a specific test design technique;

— N is the number of test coverage items covered by executed test cases;

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

— T is the total number of test coverage items identified by the test design technique.

Specific values for Coverage, N and T for each technique are defined in the clauses below. The set of coverage measures presented in the following clauses are designed to be used with the test design techniques presented in this standard. They are not designed to be an exhaustive list; there may be other coverage measures that are used by organizations that are not mentioned in this clause.

EXAMPLE Other measures that could be required for assessing the completeness of testing include measuring the overall percentage of requirements that have been covered during testing.

6.2 Test Measurement for Specification-Based Test Design Techniques

6.2.1 Equivalence Partition Coverage

Coverage for equivalence partitioning shall be calculated using the following definitions:

— Coverage is the equivalence partition coverage;

— N is the number of partitions covered by executed test cases;

— T is the total number of partitions identified.

6.2.2 Classification Tree Method Coverage

Coverage for the classification tree method shall be calculated using the following definitions.

For minimized combination coverage, the following definitions shall be used:

— Coverage is the minimized combination coverage classification tree method coverage;

— N is the number of classes covered by executed test cases;

— T is the total number of classes.

For maximized combination coverage the following definitions shall be used:

— Coverage is the maximized combination classification tree method coverage;

— N is the number of combinations of classes covered by executed test cases;

— T is the total number of combinations of classes.

6.2.3 Boundary Value Analysis Coverage

Coverage for boundary value analysis shall be calculated using the following definitions:

— Coverage is the boundary value coverage;

— N is the number of distinct boundary values covered by executed test cases;

— T is the total number of boundary values.

The decision to apply two-value or three-value boundary testing should be recorded.

6.2.4 Syntax Testing Coverage

There is currently no industry agreed approach to calculating coverage for syntax testing.

NOTE Calculating coverage as a percentage is not possible for syntax testing due to the potentially extremely large number of options and infinite number of mutations possible.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

6.2.5 Combinatorial Test Design Technique Coverage

6.2.5.1 All Combinations Testing Coverage

Coverage for all combinations testing shall be calculated using the following definitions:

— Coverage is all combinations coverage;

— N is the number of unique combinations of P-V pairs covered by executed test cases;

— T is the total number of unique P-V pair combinations (the product of the number of P-V pairs for each test item parameter).

NOTE For a definition of P-V pair, refer to 4.2.3.

6.2.5.2 Pair-wise Testing Coverage

Coverage for pair-wise testing shall be calculated using the following definitions:

— Coverage is pair-wise coverage;

— N is the number of unique pairs of P-V pairs covered by executed test cases;

— T is the total number of unique pairs of P-V pairs.

6.2.5.3 Each Choice Testing Coverage

Coverage for each choice testing shall be calculated using the following definitions:

— Coverage is each choice coverage;

— N is the number of P-V pairs covered by executed test cases;

— T is the total number of unique P-V pairs.

6.2.5.4 Base Choice Testing Coverage

Coverage for base choice testing shall be calculated using the following definitions:

— Coverage is base choice coverage;

— N is the number of base choice combinations covered by executed test cases (all but one test item parameter set to the base value, and the remaining test item parameter set to valid values), plus one (for when all test item parameters are set to the base value) if exercised;

— T is the total number of base choice combinations (all but one test item parameter set to the base value, and the remaining test item parameter set to valid values), plus one (for when all test item parameters are set to the base value).

6.2.6 Decision Table Testing Coverage

Coverage for decision table testing shall be calculated using the following definitions:

— Coverage is decision table coverage;

— N is the number of feasible decision rules covered by executed test cases;

— T is the total number of feasible decision rules.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

6.2.7 Cause-Effect Graphing Coverage

Coverage for cause-effect graphing shall be calculated using the following definitions:

— Coverage is the cause/effect coverage;

— N is the number of decision rules covered by executed test cases;

— T is the total number of feasible decision rules.

6.2.8 State Transition Testing Coverage

Coverage for all states testing shall be calculated using the following definitions:

— Coverage is all states coverage;

— N is the number of states covered by executed test cases;

— T is the total number of states.

Coverage for single transitions (0-switch coverage) shall be calculated using the following definitions:

— Coverage is 0-switch coverage;

— N is the number of single valid transitions executed by test cases;

— T is the total number of single valid transitions.

Coverage for all transitions shall be calculated using the following definitions:

— Coverage is all transitions transition coverage;

— N is the number of valid and invalid transitions attempted to be covered by executed test cases;

— T is the total number of valid and invalid transitions between identified states initiated by valid events.

Coverage for N+1 transitions (N switch testing) shall be calculated using the following definitions:

— Coverage is N-switch coverage;

— N is the number of N+1 valid transitions covered by executed test cases;

— T is the total number of sequences of N+1 valid transitions.

6.2.9 Scenario Testing Coverage

Coverage for scenario testing (including use case testing) shall be calculated using the following definitions:

— Coverage is scenario coverage;

— N is the number of scenarios covered by executed test cases;

— T is the total number of scenarios.

6.2.10 Random Testing Coverage

There is currently no industry agreed approach to calculating coverage for random testing.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

6.3 Test Measurement for Structure-Based Test Design Techniques

6.3.1 Statement Testing Coverage

Coverage for statement testing shall be calculated using the following definitions:

— Coverage is statement coverage;

— N is the number of executable statements covered by executed test cases;

— T is the total number of executable statements.

6.3.2 Branch Testing Coverage

Coverage for branch testing shall be calculated using the following definitions:

— Coverage is branch coverage;

— N is the number of branches covered by executed test cases;

— T is the total number of branches.

NOTE For situations where there are no branches in the test item, a single test is required to achieve 100% branch coverage.

6.3.3 Decision Testing Coverage

Coverage for decision testing shall be calculated using the following definitions:

— Coverage is decision coverage;

— N is the number of decision outcomes covered by executed test cases;

— T is the total number of decision outcomes.

NOTE For situations where there are no decisions in the test item, a single test is required to achieve 100% decision coverage.

6.3.4 Branch Condition Testing Coverage

Coverage for branch condition testing shall be calculated using the following definitions:

— Coverage is branch condition coverage;

— N is the number of Boolean values of conditions within decisions plus the number of decision outcomes covered by executed test cases;

— T is the total number of Boolean values of conditions within decisions plus the total number of decision outcomes.

NOTE For situations where there are no decisions in the test item, a single test is required to achieve 100% branch condition coverage.

6.3.5 Branch Condition Combination Testing Coverage

Coverage for branch condition combination testing shall be calculated using the following definitions:

— Coverage is branch condition combination coverage;

— N is the number of combinations of Boolean values of conditions within each decision covered by executed test cases;

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

AVA Documents Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

— T is the total number of unique combinations of Boolean values of conditions within decisions.

NOTE For situations where there are no decisions in the test item, a single test is required to achieve 100% branch condition combination coverage.

6.3.6 Modified Condition Decision (MCDC) Testing Coverage

Coverage for modified condition decision coverage testing shall be calculated using the following definitions:

— Coverage is modified condition decision coverage;

— N is the number of unique feasible combinations of individual Boolean values of conditions within decisions that allow a single Boolean condition to independently affect the decision outcome to be covered by executed test cases;

— T is the total number of unique combinations of individual Boolean values of conditions within decisions that allow a single Boolean condition to independently affect the outcome.

NOTE For situations where there are no decisions in the test item, a single test is required to achieve 100% modified condition decision coverage.

6.3.7 Data Flow Testing Coverage

6.3.7.1 All-Definitions Testing Coverage

Coverage for all-definitions testing shall be calculated using the following definitions:

— Coverage is all-definitions coverage;

— N is the number of definitions associated with data-definition-use pairs covered by executed test cases;

— T is the total number of definitions-use pairs from distinct variable definitions.

6.3.7.2 All-C-Uses Testing Coverage

Coverage for all-c-uses testing shall be calculated using the following definitions:

— Coverage is all-c-uses coverage;

— N is the number of definition-c-use pairs covered by executed test cases;

— T is the total number of definition-c-use pairs.

6.3.7.3 All-P-Uses Testing Coverage

Coverage for all-p-uses testing shall be calculated using the following definitions:

— Coverage is all-p-uses coverage;

— N is the number of definition-p-use pairs covered by executed test cases;

— T is the total number of definition-p-use pairs.

6.3.7.4 All-Uses Testing Coverage

Coverage for all-uses testing shall be calculated using the following definitions:

— Coverage is all-uses coverage;

— N is the number of unique definition-use pairs covered by executed test cases;

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

— T is the total number of definition-use pairs from each definition to both p-use *and* c-use of that definition.

6.3.7.5 All-DU-Paths Testing Coverage

Coverage for all-du-paths testing shall be calculated using the following definitions:

— Coverage is all-du-paths coverage;

— N is the number of sub-paths from each variable definition to each of its uses that can be reached (without an intervening definition of the variable) covered by executed test cases;

— T is the total number of simple sub-paths from each definition to both p-use *and* c-use of that definition.

6.4 Test Measurement for Experience-Based Testing Design Techniques

6.4.1 Error Guessing Coverage

There is currently no industry agreed approach to calculating coverage for error guessing.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

Annex A
(informative)

Testing Quality Characteristics

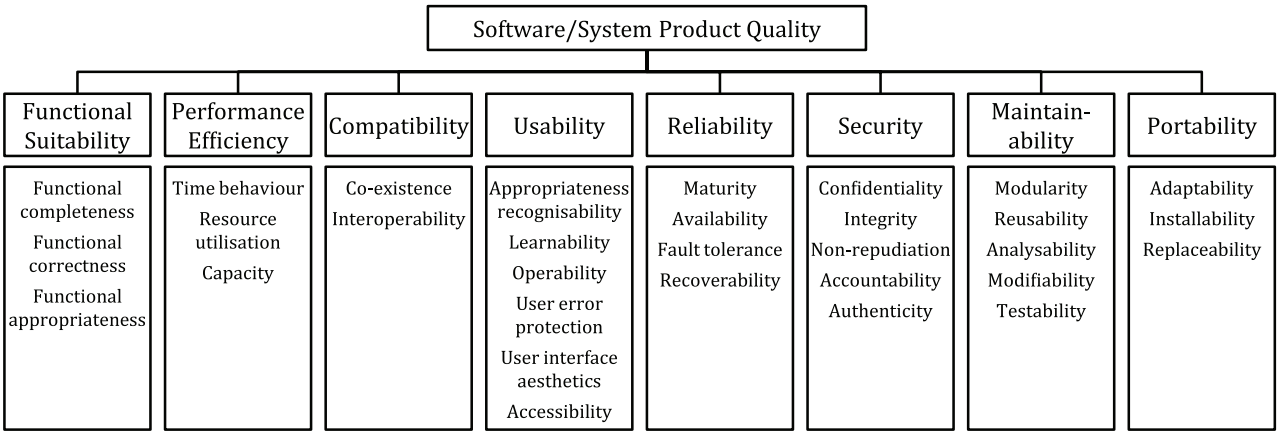
A.1 Quality Characteristics

A.1.1 Overview

Software testing is carried out to collect evidence as to whether required quality criteria have been satisfied by a test item. This informative annex contains examples of how quality characteristics defined in ISO/IEC 25010 (ISO/IEC 25010:2011) can be mapped to the test design techniques defined in this part of ISO/IEC/IEEE 29119. The test design techniques defined in this standard could be used for testing a variety of these characteristics. There may be other quality characteristics that could be considered (e.g. privacy).

ISO/IEC 25010 defines the product quality model shown below in Figure A.1, which categorizes system/software product quality properties into eight characteristics: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability and portability. Each characteristic is composed of a set of related sub-characteristics. In some situations, there may be regulatory requirements (e.g. government policies or laws) that mandate that certain quality characteristics are met by a system. Various test design techniques and types of testing can be used to test for each characteristic (see Clauses A.3 and A.4).

Clause A.2 provides an explanation of types of testing that can be used to test the quality characteristics presented in Figure A.1. A mapping of each quality characteristic to applicable types of testing is presented in Clause A.3. The relationship between the quality characteristics and the specification-based and structure-based test design techniques covered in this part of ISO/IEC/IEEE 29119 is explained in Clause A.4.



NOTE ISO/IEC 25030 (ISO/IEC 25030:2007) can be used to identify and document software quality requirements that are applicable to a test item. These can then be used to identify quality characteristics in ISO/IEC 25010 (ISO/IEC 25010:2011) and corresponding types of testing that apply to testing each quality requirement.

Figure A.1 — ISO/IEC 25010 product quality model

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

A.2 Quality-Related Types of Testing

A.2.1 Accessibility Testing

The purpose of accessibility testing is to determine whether the test item can be operated by users with the widest range of characteristics and capabilities (ISO/IEC 25010:2011), including specific accessibility requirements (e.g. due to age, visual impairment or hearing impairment). Accessibility testing uses a model of the test item that specifies its accessibility requirements, including any accessibility design standards to which the test item must conform. Accessibility requirements are concerned with the ability of a user with specific accessibility needs to achieve accessibility objectives. For example, this could include a requirement for the test item to support visual and/or hearing impaired users.

NOTE The World Wide Web Consortium (W3C) defines standards for accessibility, including accessibility of web applications and devices. Visit <http://www.w3.org/standards/> for more information.

A.2.2 Backup/Recovery Testing

The purpose of backup/recovery testing is to determine if, in the event of failure, a test item can be restored from backup to its pre-failure state. Backup/recovery testing uses a model of the test item that specifies its backup and recovery requirements, which specify the need to back up the operational state of a test item at a point in time, including data, configuration and/or environment and restore the state of the test item from that backup. Backup/recovery testing then focusses on testing the correctness of the test item's backup and the correctness of the restored state of the test item against its pre-failure state. Backup/recovery testing can also be used to verify whether the backup and recovery procedures for the test item achieve specified recovery objectives. This type of testing may be carried out as part of a disaster recovery test (see A.2.5).

A.2.3 Compatibility Testing

The purpose of compatibility testing is to determine whether the test item can function alongside other independent or dependent (but not necessarily communicating) products in a shared environment (i.e. co-existence). Compatibility testing may also be applied to multiple copies of the same test item or to multiple test items sharing a common environment.

Compatibility requirements for test items typically include one or more of the following sub-requirements:

— Order of installation. Explicit order(s) of installation (otherwise it should be assumed that all possible orders of installation are valid) results in a configuration where each test item will subsequently perform its required functions correctly.

— Order of instantiation. Explicit order(s) of instantiation (otherwise it should be assumed that all possible orders of instantiation are valid) result in a run-time configuration where each test item will subsequently perform its required functions correctly.

— Concurrent use. The ability of two or more test items to perform their required functions while running (but not necessarily communicating) in the same environment.

— Environment constraints. Features of the environment, such as memory, processor, architecture, platform or configuration, that may affect the ability of the test item to perform its required functions correctly.

A.2.4 Conversion Testing

The purpose of conversion testing is to determine whether data or software can continue to provide required capabilities after modifications are made to their format, such as converting a program from one programming language to another or converting a flat data file or database from one format to another. One common subtype of conversion testing is data migration testing. Conversion testing uses a model of the test item that specifies its conversion requirements, including those that must remain

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

invariant through the conversion process, those that are new, modified, or obsoleted by the conversion and any required conversion design standards to which the test item must conform.

A.2.5 Disaster Recovery Testing

The purpose of disaster recovery testing is to determine if, in the event of failure, operation of the test item can be transferred to a different operating site and whether it can be transferred back again once the failure has been resolved. Disaster recovery testing uses a model of the test item (typically a disaster recovery plan) that specifies its disaster recovery requirements, including any required disaster recovery design standards to which the test item must conform. The test item in disaster recovery testing may be an entire operational system, with associated facilities, personnel, and procedures. Disaster recovery testing may cover factors such as procedures to be carried out by operational staff, relocation of data, software, personnel, offices, or other facilities, or recovering data previously backed up to a remote location.

A.2.6 Functional Testing

The purpose of functional testing is to determine whether the functional requirements of the test item have been met. For example, this could include identifying whether a function had been implemented according to its specified requirements. It could be carried out using the specification-based and structure-based test design techniques that are specified in Clause 5.

A.2.7 Installability Testing

The purpose of installability testing is to determine if a test item(s) can be installed, uninstalled/removed and/or upgraded as required in all specified environments. Installability testing uses a model of the installability requirements of the test item, which are typically specified in terms of the installation, uninstallation or upgrade processes (as described in the installation manual or guidelines), the people who will carry out the installation, uninstallation or upgrade, the target platform(s) and the test item(s) to be installed, uninstalled or upgraded.

A.2.8 Interoperability Testing

The purpose of interoperability testing is to determine if a test item can interact correctly with other test items or systems either in the same environment or different environments, including whether the test item can make effective use of information received from other systems. Interoperability testing uses a model of the test item that specifies its interoperability requirements, including interoperability design standards to which the test item must conform. This could include assessing whether a test item running in one environment can interact accurately with another test item or system in another separate environment.

A.2.9 Localization Testing

The purpose of localization testing is to determine whether the test item can be understood within the geographical region it is required to be used in. Localization testing can include (but is not limited to) analysis of whether the user interface and supporting documentation of the test item can be understood by users within each country or region of use.

A.2.10 Maintainability Testing

The purpose of maintainability testing is to determine if a test item can be maintained by using an acceptable amount of effort. Maintainability testing uses a model of the maintainability requirements of the test item, which are typically specified in terms of the effort required to effect a change under the following categories:

— corrective maintenance (i.e. correcting problems);

— perfective maintenance (i.e. enhancements);

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

— adaptive maintenance (i.e. adapting to changes in environment); and

— preventive maintenance (i.e. actions to reduce future maintenance costs).

Maintainability can be indirectly measured by applying static analysis.

A.2.11 Performance-Related Testing

The purpose of this family of techniques is to determine whether a test item performs as required when it is placed under various types and sizes of "load". This includes performance, load, stress, endurance, volume, capacity and memory management testing, which each use a model of the test item that specifies its performance requirements, including any required performance design standards to which the test item must conform. For example, this may include assessing the performance of the test item in terms of transactions per second, throughput response times, round trip time and resource utilization levels. The "typical" load of the test item under "normal" conditions may be defined in the operational profile of the test item.

There are numerous techniques for assessing the performance of the test item:

— Performance testing is aimed at assessing the performance of the test item when it is placed under a "typical" load.

— Load testing is aimed at assessing the behaviour of the test item (e.g. performance and reliability) when it is placed under conditions of varying loads, usually between anticipated conditions of low, typical and peak usage.

— Stress testing is aimed at assessing the performance of the test item when it is pushed beyond its anticipated peak load or when available resources (e.g. memory, processor, disk) are reduced below specified minimum requirements, to evaluate how it behaves under extreme conditions.

— Endurance testing (also called soak testing) is aimed at assessing whether the test item can sustain the required load for a continuous period of time.

— Volume testing is aimed at assessing the performance of the test item when it is processing specified levels of data. For example, this may include assessing test item performance when its database is nearing maximum capacity.

— Capacity testing (also called scalability testing) is aimed at assessing how the test item will perform under conditions that may need to be supported in the future. For example, this may include assessing what level of additional resources (e.g. memory, disk capacity, network bandwidth) will be required to support anticipated future loads.

— Memory management testing is aimed at assessing how the test item will perform in terms of the amount (normally maximum) of memory used (e.g. hard disk memory, RAM and ROM), the type of memory (e.g. dynamic or allocated/static) and/or defined levels of memory leakage experienced during testing. Memory requirements will typically be specified in terms of specific operating conditions (e.g. a peak memory requirement over a particular period of operation under defined transaction loads may be specified).

A.2.12 Portability Testing

The purpose of portability testing is to determine the degree of ease or difficulty to which a test item can be effectively and efficiently transferred from one hardware, software or other operational or usage environment to another. Portability testing uses a model of the test item that specifies its portability requirements, including any required portability design standards to which the test item must conform. Portability requirements are concerned with the ability to transfer the test item from one environment to another, or to alter the configuration of the existing environment to other required configurations. For example, this could include assessing whether the test item can be operated from a variety of different browsers.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015

A.2.13 Procedure Testing

The purpose of procedure testing is to determine whether procedural instructions meet user requirements and support the purpose of their use. Procedure testing uses a model of the procedural requirements of the test item as a complete and delivered unit. Procedure requirements define what is expected of any procedural documentation and are written in the form of procedural instructions. These procedural instructions will normally come in the form of one of the following documents:

— a user guide;

— an instruction manual;

— a user reference manual.

This information will normally define how the user is meant to:

— set up the test item for normal usage;

— operate the test item in normal conditions;

— become a competent user of the system (tutorial files);

— trouble-shoot the test item when faults arise;

— reconfigure the test item.

A.2.14 Reliability Testing

The purpose of reliability testing is to evaluate the ability of the 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. Reliability testing uses a model of the test item that specifies its required level of reliability (e.g. mean time to failure, mean time between failures). The model should include a definition of failure and either the operational profile of the test item or an approach to derive the operational profile.

A.2.15 Security Testing

The purpose of security testing is to evaluate the degree to which a test item and its associated data are protected so that unauthorised persons or systems cannot use, read or modify them and authorised persons or systems are granted required access to them. Security testing uses a model of the test item that specifies its security requirements, including any required security design standards to which the test item must conform. Security requirements are concerned with the ability to protect the data and functionality of a test item from unauthorised users and malicious use. For example, this could include assessing whether the test item prevents unauthorised users from accessing data, or whether certain functions of a test item that are only required to be accessible by certain user types are protected from other user types.

There are a number of techniques for assessing the security of a test item:

— Penetration testing involves attempted access to a test item (including its functionality and/or private data) by a tester that is mimicking the actions of an unauthorised user.

— Privacy testing involves attempted access to private data and verification of the audit trail (i.e. trace) that is left behind when users access private data.

— Security auditing is a type of static testing in which a tester inspects, reviews or walks through the requirements and code of a test item to determine whether any security vulnerabilities are present.

— Vulnerability scanning involves the use of automated testing tools to scan a test item for signs of specific known vulnerabilities.

© ISO/IEC 2015 – All rights reserved
© IEEE 2015 – All rights reserved
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 29119-4:2015