DRAFT INTERNATIONAL STANDARD
**ISO/IEC/IEEE/DIS 29119-4**

ISO/IEC JTC **1**/SC **7** Secretariat: **BIS**

Voting begins on: **2020-07-10** Voting terminates on: **2020-10-02**

# Software and systems engineering — Software testing —

Part 4:
**Test techniques**

*Ingénierie du logiciel et des systèmes — Essais du logiciel —*

*Partie 4: Techniques d'essai*

ICS: 35.080

Reference number
ISO/IEC/IEEE/DIS 29119-4:2020(E)

## Contents

# 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*

— *Part 5: Keyword-driven testing*

This is the second version of this standard. The main difference between the first version and the second version is that the test techniques in this standard are defined using a new form of the Test Design and Implementation Process from ISO/IEC/IEEE 29119-2. In the first version, this process was based on the use of test conditions. Feedback on use of the standard highlighted a problem with users' understanding of test conditions and their use for deriving test cases. This second version has replaced the use of test conditions with test models. Annex H provides more detail on this change and the following Introduction describes the new process.

## Introduction

The purpose of this document 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 document does not describe a process for test design and implementation; instead, it describes a set of techniques that can be used within the test design and implementation process defined in ISO/IEC/IEEE 29119-2. The intent is to describe a series of techniques that have wide acceptance in the software testing industry. This standard is originally based on the British standard, BS 7925-2.

The test design techniques presented in this document can be used to derive test cases that, when executed, generate evidence that test item requirements have been met 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, this standard provides guidance on how to implement the following activities in detail for each technique that is described:

— Create Test Model (TD1);

— Identify Test Coverage Items (TD2);

— Derive Test Cases (TD3).

A test model represents testable aspects of a test item, such as a function, transaction, feature, quality attribute, or structural element identified as a basis for testing. The test model reflects the required test completion criterion in the test strategy.

EXAMPLE 2 If a test completion criterion for state transition testing was identified that required coverage of all states then the test model would show the states the test item can be in.

Test coverage items are attributes of the test model that can be covered during testing. A single test model will typically be the basis for several test coverage items.

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 Create Test Procedures activity (TD4) in the test design & implementation process of ISO/IEC/IEEE 29119-2 is not included in Clauses 5 or 6 of this document because the process is the same for all techniques.

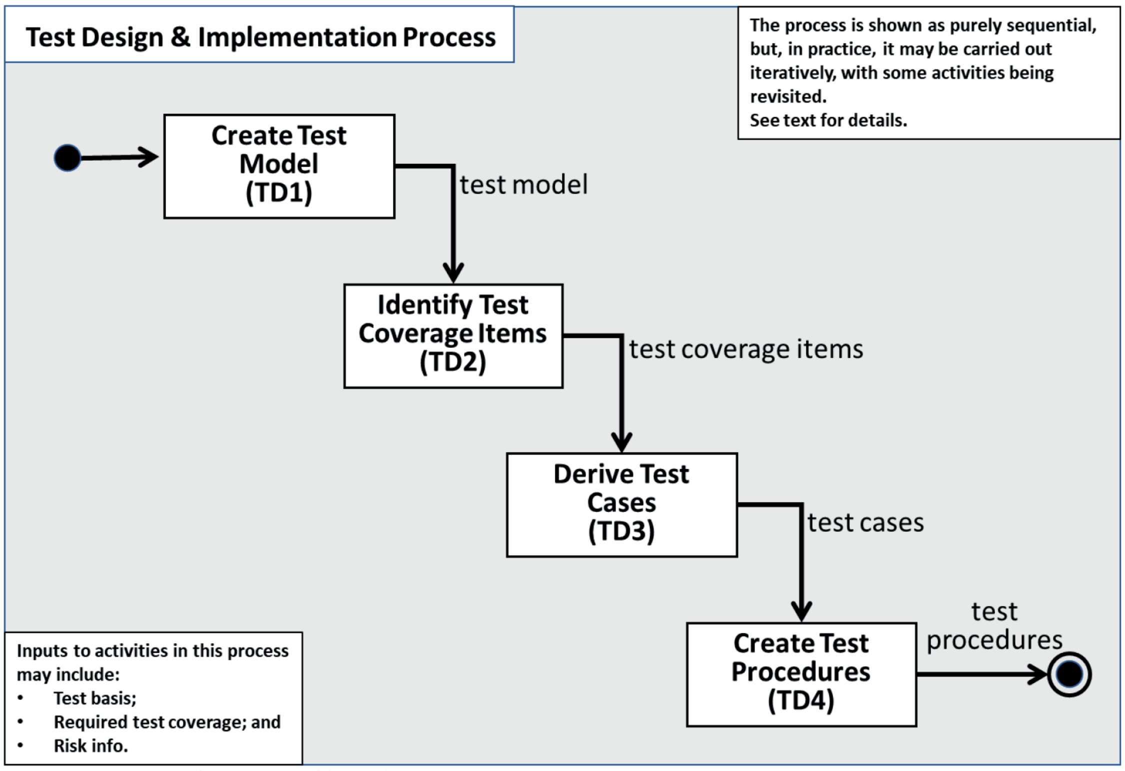
Test Design & Implementation Process
The process is shown as purely sequential, but, in practice, it may be carried out iteratively, with some activities being revisited.
See text for details.
Create Test Model (TD1)
test model
Identify Test Coverage Items (TD2)
test coverage items
Derive Test Cases (TD3)
test cases
Create Test Procedures (TD4)
test procedures
Inputs to activities in this process may include:
• Test basis;
• Required test coverage; and
• Risk info.

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

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 document because this document only describes techniques for designing test cases. Test practices such as exploratory testing, which could use the test techniques defined in this document, 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 document do not describe how to document test cases (e.g. they do not include information or guidance on assigning unique identifiers, test case descriptions, priorities, traceability or pre-conditions to test cases). Information on how to document test cases can be found in ISO/IEC/IEEE 29119-3.

This document aims to provide stakeholders with the ability to design test cases for software testing in any organization.

# Software and systems engineering — Software testing —

# Part 4:
# Test techniques

## 1 Scope

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

Each technique follows the test design and implementation process that is defined in ISO/IEC/IEEE 29119-2 and shown in Figure 1. This document is intended for, but not limited to, testers, test managers, and developers, particularly those responsible for managing and implementing software testing.

## 2 Normative References

The following document, in whole or in part, is normatively referenced in this document and is indispensable for its application. The latest edition of the referenced document (including any amendments) applies.

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

NOTE Other International Standards useful for the implementation and interpretation of this document are listed in the bibliography.

## 3 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 document is for ease of reference and is not mandatory for conformance with this document. The following terms and definitions are provided to assist with the understanding and readability of this document. Only terms critical to the understanding of this document 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. This source is available at the following web site: http://www.computer.org/sevocab. All terms defined in this clause are also intentionally included in ISO/IEC/IEEE 29119-1, as that international standard includes all terms that are used in parts 1 to 4 of ISO/IEC/IEEE 29119.

**3.1**
**Backus-Naur Form**
formal meta-language used for defining the syntax of a language in a textual format

**3.2**
**base choice**
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.

**3.3**
**c-use**
computation data use
use of the value of a variable in any type of statement

**3.4**
**condition**
Boolean expression containing no Boolean operators

EXAMPLE “A < B” is a condition but “A and B” is not.

**3.5**
**control flow**
sequence in which operations are performed during the execution of a test item

**3.6**
**control flow sub-path**
sequence of executable statements within a test item

**3.7**
**data definition**
variable definition
statement where a variable is assigned a value. Also called variable definition

**3.8**
**data definition c-use pair**
data definition and subsequent computation data use, where the data use uses the value defined in the data definition

**3.9**
**data definition p-use pair**
data definition and subsequent predicate data use, where the data use uses the value defined in the data definition

**3.10**
**data use**
executable statement where the value of a variable is accessed

**3.11**
**decision outcome**
result of a decision that determines the branch to be executed

**3.12**
**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

**3.13**
**definition-use pair**
data definition-use pair
data definition and subsequent predicate or computational data use, where the data use uses the value defined in the data definition

**3.14**
**entry point**
point in a test item at which execution of the test item can begin

EXAMPLE An entry point is an executable statement within a test item that could 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.

**3.15**
**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

**3.16**
**exit point**
last executable statement within a test item

EXAMPLE 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.

**3.17**
**p-use**
predicate data use
data use associated with the decision outcome of the predicate portion of a decision statement

**3.18**
**P-V pair**
parameter-value pair
combination of a test item parameter with a value assigned to that parameter, used as a test coverage item in combinatorial test design techniques

**3.19**
**path**
sequence of executable statements of a test item

**3.20**
**predicate**
logical expression which evaluates to TRUE or FALSE to direct the execution path in code

**3.21**
**sub-path**
path that is part of a larger path

**3.22**
**test model**
representation of an aspect of the test item, which is in the focus of testing

EXAMPLE 1 Requirements statements, equivalence partitions, state transition diagram, use case description, decision table, input syntax description, source code, control flow graph, parameters & values, classification tree, natural language.

EXAMPLE 2 The test model and the required test coverage are used to identify test coverage items.

EXAMPLE 3 A separate test model may be required for each different type of required test coverage included in the test completion criteria.

EXAMPLE 4 A test model can include one or more test conditions.

EXAMPLE 5 Test models are commonly used to support test design (e.g. they are used to support test design in ISO/IEC/IEEE 29119-4, and they are used in model-based testing). Other types of models exist to support other aspects of testing, such as test environment models, test maturity models and test architecture models.

## 4 Conformance

### 4.1 Intended Usage

The requirements in this document are contained in Clauses 5 and 6. It is recognised that particular projects or organizations will not need to use all of the techniques defined by this standard. Implementation of ISO/IEC/IEEE 29119-2 involves using a risk-based approach to selecting a subset of techniques suitable for a given project or organization. There are two ways that an organization or individual can claim conformance to the provisions of this standard – full conformance and tailored conformance.

The organization shall assert whether it is claiming full or tailored conformance to this document.

### 4.2 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 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 document.

### 4.3 Tailored Conformance

Tailored conformance is achieved by demonstrating that the chosen subset of requirements from the chosen (non-empty) set of techniques and 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.

## 5 Test Design Techniques

### 5.1 Overview

This document defines test design techniques for specification-based testing (clause 5.2), structure-based testing (clause 5.3) and experience-based testing (clause 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 specification-based, structure-based or experience-based, in practice some of them can be used interchangeably (e.g. branch testing could be applied to the specification of the graphical user interface of a web-based system). This is demonstrated in Annex E. In addition, although each technique is defined independently of all others, in practice some can be used in combination with other techniques.

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

This document 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".

This document defines how the generic test design and implementation process steps TD1(create test model), TD2 (identify test coverage items), and TD3 (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 this document 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 this document are shown in Figure 2. This set of techniques is not exhaustive. There are useful techniques that are used by testing practitioners or researchers that are not included in this document.

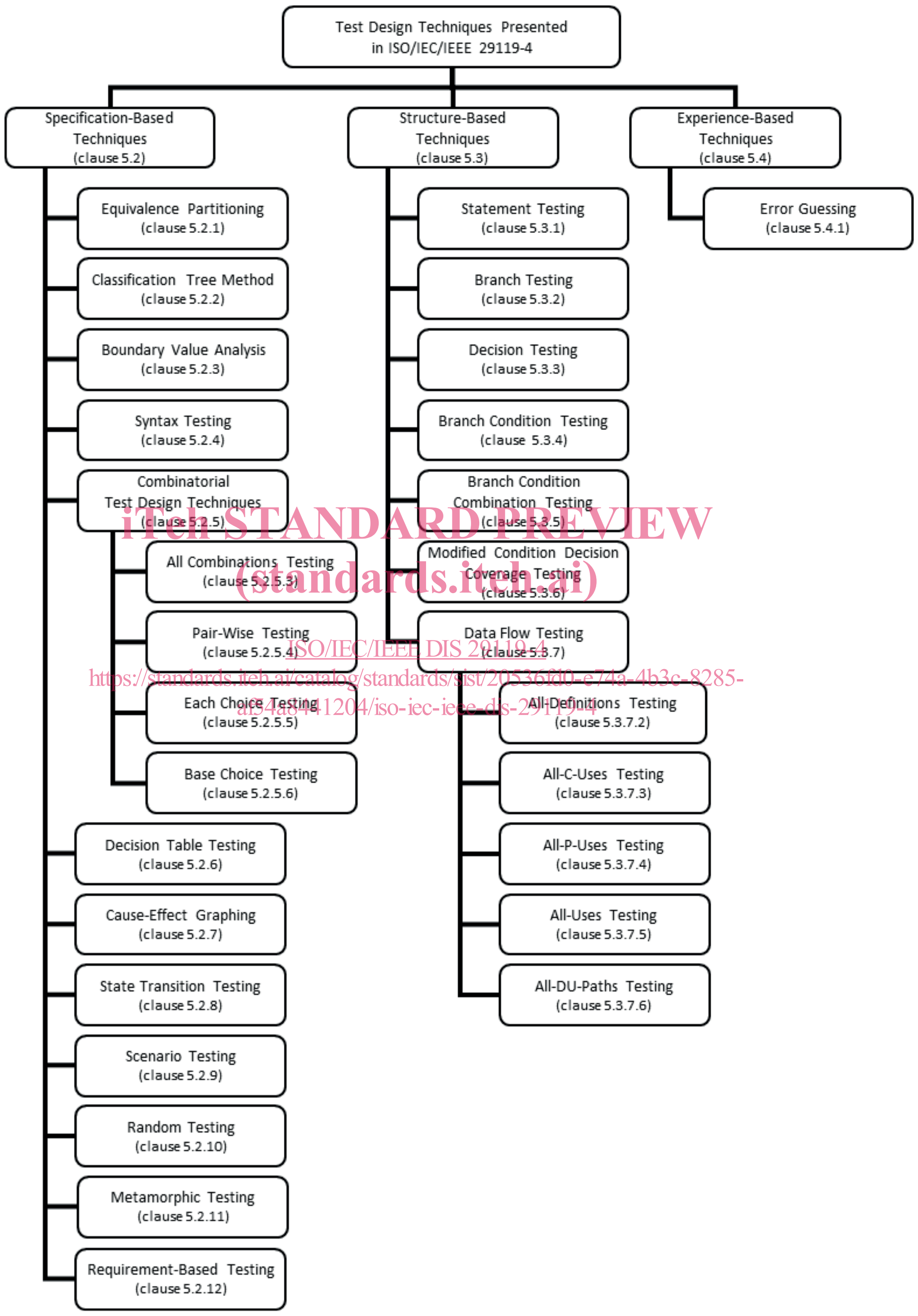
Test Design Techniques Presented in ISO/IEC/IEEE 29119-4
Specification-Based Techniques (clause 5.2)
Structure-Based Techniques (clause 5.3)
Experience-Based Techniques (clause 5.4)
Equivalence Partitioning (clause 5.2.1)
Classification Tree Method (clause 5.2.2)
Boundary Value Analysis (clause 5.2.3)
Syntax Testing (clause 5.2.4)
Combinatorial Test Design Techniques (clause 5.2.5)
All Combinations Testing (clause 5.2.5.3)
Pair-Wise Testing (clause 5.2.5.4)
Each Choice Testing (clause 5.2.5.5)
Base Choice Testing (clause 5.2.5.6)
Decision Table Testing (clause 5.2.6)
Cause-Effect Graphing (clause 5.2.7)
State Transition Testing (clause 5.2.8)
Scenario Testing (clause 5.2.9)
Random Testing (clause 5.2.10)
Metamorphic Testing (clause 5.2.11)
Requirement-Based Testing (clause 5.2.12)
Statement Testing (clause 5.3.1)
Branch Testing (clause 5.3.2)
Decision Testing (clause 5.3.3)
Branch Condition Testing (clause 5.3.4)
Branch Condition Combination Testing (clause 5.3.5)
Modified Condition Decision Coverage Testing (clause 5.3.6)
Data Flow Testing (clause 5.3.7)
All-Definitions Testing (clause 5.3.7.2)
All-C-Uses Testing (clause 5.3.7.3)
All-P-Uses Testing (clause 5.3.7.4)
All-Uses Testing (clause 5.3.7.5)
All-DU-Paths Testing (clause 5.3.7.6)
Error Guessing (clause 5.4.1)

**Figure 2 — The set of test design techniques presented in this document**

Of the four activities in the test design and implementation process (see Figure 1), test design techniques provide unique and specific guidance on the derivation of test models (TD1), identification of test coverage items (TD2) and the derivation of test cases (TD3). Therefore, each technique is defined in terms of these three activities.

There are varying levels of granularity within steps TD1 (create test model), TD2 (identify test coverage items) and TD3 (derive test cases). Within each technique, the term "model" is used to describe the concept of preparing a logical representation of the test item for the purposes of deriving test coverage items in step TD2 (e.g. a control flow model is normally used as a test model for identifying test coverage items for all structural techniques).

EXAMPLE 1 In state transition testing, if there is a requirement to cover all states then the state model for the test item would form the test model. If there is a requirement to cover all transitions between states, then each transition would be a test coverage item.

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 activity (TD3) 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 “Valid” test cases are also known as “positive” test cases. Invalid cases are also known as “negative test cases”.

Although the techniques defined in this document 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 complete test cases from the test input values. Similarly, 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 this document 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 and then representative users (test coverage items) from those groups to create test cases for 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. These are 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.

EXAMPLE 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 non-functional quality characteristics defined in ISO/IEC 25010.

## 5.2 Specification-Based Test Design Techniques

### 5.2.1 Equivalence Partitioning

Equivalence partitioning (BS 7925-2:1998; Myers 1979) is based on exercising equivalence partitions. These equivalence partitions are derived based on the expectation that values within a partition should be treated similarly by the test item.

#### 5.2.1.1 Create Test Model (TD1)

A test model shall be created that identifies the equivalence partitions for the test item. A suitable test model shows the various functions expected to be performed by the test item from which equivalence partitions can be derived. These functions can be considered as a set of processes, which can be decomposed into input processes, internal processes and output processes.

The functions described in the test item's specification can typically be used to identify its internal processes. Consideration of the inputs and outputs of the test item can be used to help determine its probable input and output processes.

Equivalence partitions should be identified for both valid and invalid functions (leading to both valid and invalid outputs) and valid and invalid inputs to the test item. In this context, invalid functions are those that are not included in the specification and which cannot reasonably be expected to be included in a correct implementation of the test item. Similarly, invalid inputs are those that should either be ignored by the test item or cause the test item to output an error message.

Equivalence partitions for a test item can overlap, and in some situations, one partition can be a subset of another partition. By adding detail to the test model, further equivalence partitions can often be derived, often as subsets of existing partitions. As more detail and partitions are added, the number of test cases required to fully exercise the test model is likely to increase.

EXAMPLE For a test item expecting lowercase alphabetical characters as (valid) inputs, invalid input partitions could be derived containing integers, reals, uppercase alphabetical characters, symbols and control characters, depending on the level of rigour required during testing. If each of these partitions were included separately in the model, each would be expected to be exercised to achieve full coverage, whereas if a single partition was created to cover all invalid inputs, then only a single partition would need to be exercised to achieve full partition coverage.

NOTE 1 Invalid equivalence partitions that correspond to inputs, functions or outputs that have not been specified (as is often the case) are typically based on the experience and imagination of the tester, which means different testers will often derive different invalid partitions. This subjective form of test design may also occur when applying experience-based techniques like error guessing.

NOTE 2 Domain analysis (Beizer 1995) is often classified as a combination of equivalence partitioning and boundary value analysis.

#### 5.2.1.2 Identify Test Coverage Items (TD2)

Each equivalence partition shall be identified as a test coverage item.

#### 5.2.1.3 Derive Test Cases (TD3)

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 the minimum number of test cases is derived to cover each equivalence partition at least once.

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

b) Select test coverage items 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;