
**Software and systems engineering —
Software testing —**

**Part 11:
Guidelines on the testing of AI-based
systems**

**iTeh STANDARD PREVIEW
(standards.iteh.ai)**

[ISO/IEC TR 29119-11:2020](https://standards.iteh.ai/catalog/standards/sist/ff3c797a-5cda-4aa3-a3c9-73d784d6157b/iso-iec-tr-29119-11-2020)

<https://standards.iteh.ai/catalog/standards/sist/ff3c797a-5cda-4aa3-a3c9-73d784d6157b/iso-iec-tr-29119-11-2020>



iTeh STANDARD PREVIEW (standards.iteh.ai)

[ISO/IEC TR 29119-11:2020](https://standards.iteh.ai/catalog/standards/sist/ff3c797a-5cda-4aa3-a3c9-73d784d6157b/iso-iec-tr-29119-11-2020)

<https://standards.iteh.ai/catalog/standards/sist/ff3c797a-5cda-4aa3-a3c9-73d784d6157b/iso-iec-tr-29119-11-2020>



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2020

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier; Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword	v
Introduction	vi
1 Scope	1
2 Normative references	1
3 Terms, definitions and abbreviated terms	1
3.1 Terms and definitions.....	1
3.2 Abbreviated terms.....	10
4 Introduction to AI and testing	11
4.1 Overview of AI and testing.....	11
4.2 Artificial intelligence (AI).....	11
4.2.1 Definition of ‘artificial intelligence’.....	11
4.2.2 AI use cases.....	12
4.2.3 AI usage and market.....	12
4.2.4 AI technologies.....	13
4.2.5 AI hardware.....	15
4.2.6 AI development frameworks.....	16
4.2.7 Narrow vs general AI.....	16
4.3 Testing of AI-based systems.....	16
4.3.1 The importance of testing for AI-based systems.....	16
4.3.2 Safety-related AI-based systems.....	17
4.3.3 Standardization and AI.....	17
5 AI system characteristics	19
5.1 AI-specific characteristics.....	19
5.1.1 General.....	19
5.1.2 Flexibility and adaptability.....	20
5.1.3 Autonomy.....	20
5.1.4 Evolution.....	21
5.1.5 Bias.....	21
5.1.6 Complexity.....	21
5.1.7 Transparency, interpretability and explainability.....	22
5.1.8 Non-determinism.....	22
5.2 Aligning AI-based systems with human values.....	23
5.3 Side-effects.....	23
5.4 Reward hacking.....	24
5.5 Specifying ethical requirements for AI-based systems.....	24
6 Introduction to the testing of AI-based systems	25
6.1 Challenges in testing AI-based systems.....	25
6.1.1 Introduction to challenges testing AI-based systems.....	25
6.1.2 System specifications.....	25
6.1.3 Test input data.....	25
6.1.4 Self-learning systems.....	26
6.1.5 Flexibility and adaptability.....	26
6.1.6 Autonomy.....	26
6.1.7 Evolution.....	26
6.1.8 Bias.....	26
6.1.9 Transparency, interpretability and explainability.....	27
6.1.10 Complexity.....	27
6.1.11 Probabilistic and non-deterministic systems.....	27
6.1.12 The test oracle problem for AI-based systems.....	27
6.2 Testing AI-based systems across the life cycle.....	27
6.2.1 General.....	27
6.2.2 Unit/component testing.....	28

6.2.3	Integration testing.....	28
6.2.4	System testing.....	28
6.2.5	System integration testing.....	29
6.2.6	Acceptance testing.....	29
6.2.7	Maintenance testing.....	29
7	Testing and QA of ML systems.....	29
7.1	Introduction to the testing and QA of ML systems.....	29
7.2	Review of ML workflow.....	29
7.3	Acceptance criteria.....	29
7.4	Framework, algorithm/model and hyperparameter selection.....	30
7.5	Training data quality.....	30
7.6	Test data quality.....	30
7.7	Model updates.....	30
7.8	Adversarial examples and testing.....	30
7.9	Benchmarks for machine learning.....	31
8	Black-box testing of AI-based systems.....	31
8.1	Combinatorial testing.....	31
8.2	Back-to-back testing.....	32
8.3	A/B testing.....	32
8.4	Metamorphic testing.....	33
8.5	Exploratory testing.....	34
9	White-box testing of neural networks.....	34
9.1	Structure of a neural network.....	34
9.2	Test coverage measures for neural networks.....	36
9.2.1	Introduction to test coverage levels.....	36
9.2.2	Neuron coverage.....	36
9.2.3	Threshold coverage.....	36
9.2.4	Sign change coverage.....	36
9.2.5	Value change coverage.....	36
9.2.6	Sign-sign coverage.....	36
9.2.7	Layer coverage.....	37
9.3	Test effectiveness of the white-box measures.....	37
9.4	White-box testing tools for neural networks.....	37
10	Test environments for AI-based systems.....	38
10.1	Test environments for AI-based systems.....	38
10.2	Test scenario derivation.....	39
10.3	Regulatory test scenarios and test environments.....	39
	Annex A Machine learning.....	40
	Bibliography.....	49

ITeH STANDARD PREVIEW
 (standards.iteh.ai)
 ISO/IEC TR 29119-11:2020
<http://standards.iteh.ai/catalog/standards/sist/fb3c797a-5cda-4aa3-a3c9-73d784d6157b/iso-iec-tr-29119-11-2020>

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.

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) or the IEC list of patent declarations received (see patents.iec.ch).

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

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*.

A list of all parts in the ISO/IEC/IEEE 29119 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

The testing of traditional systems is well-understood, but AI-based systems, which are becoming more prevalent and critical to our daily lives, introduce new challenges. This document has been created to introduce AI-based systems and provide guidelines on how they might be tested.

[Annex A](#) provides an introduction to machine learning.

This document is primarily provided for those testers who are new to AI-based systems, but it can also be useful for more experienced testers and other stakeholders working on the development and testing of AI-based systems.

As a Technical Report, this document contains data of a different kind from that normally published as an International Standard or Technical Specification, such as data on the “state of the art”.

iTeh STANDARD PREVIEW (standards.iteh.ai)

[ISO/IEC TR 29119-11:2020](#)

<https://standards.iteh.ai/catalog/standards/sist/ff3c797a-5cda-4aa3-a3c9-73d784d6157b/iso-iec-tr-29119-11-2020>

Software and systems engineering — Software testing —

Part 11: Guidelines on the testing of AI-based systems

1 Scope

This document provides an introduction to AI-based systems. These systems are typically complex (e.g. deep neural nets), are sometimes based on big data, can be poorly specified and can be non-deterministic, which creates new challenges and opportunities for testing them.

This document explains those characteristics which are specific to AI-based systems and explains the corresponding difficulties of specifying the acceptance criteria for such systems.

This document presents the challenges of testing AI-based systems, the main challenge being the test oracle problem, whereby testers find it difficult to determine expected results for testing and therefore whether tests have passed or failed. It covers testing of these systems across the life cycle and gives guidelines on how AI-based systems in general can be tested using black-box approaches and introduces white-box testing specifically for neural networks. It describes options for the test environments and test scenarios used for testing AI-based systems.

In this document an AI-based system is a system that includes at least one AI component.

2 Normative references

ISO/IEC TR 29119-11:2020
<https://standards.iteh.ai/catalog/standards/sist/ff3c797a-5cda-4aa3-a3c9-785714c15570/iso-iec-tr-29119-11-2020>

There are no normative references in this document.

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- ISO Online browsing platform: available at <https://www.iso.org/obp>
- IEC Electropedia: available at <http://www.electropedia.org/>

3.1.1

A/B testing

split-run testing

statistical testing approach that allows testers to determine which of two systems or components performs better

3.1.2

accuracy

<machine learning (3.1.43)> performance metric used to evaluate a classifier (3.1.21), which measures the proportion of classifications (3.1.20) predictions (3.1.56) that were correct

3.1.3

activation function

transfer function

<*neural network* (3.1.48)> formula associated with a node in a neural network that determines the output of the node (*activation value* (3.1.4)) from the inputs to the neuron

3.1.4

activation value

<*neural network* (3.1.48)> output of an *activation function* (3.1.3) of a node in a neural network

3.1.5

adaptability

ability of a system to react to changes in its environment in order to continue meeting both functional and non-functional requirements

3.1.6

adversarial attack

deliberate use of *adversarial examples* (3.1.7) to cause a *ML model* (3.1.46) to fail

Note 1 to entry: Typically targets ML models in the form of a *neural network* (3.1.48).

3.1.7

adversarial example

input to an *ML model* (3.1.46) created by applying small perturbations to a working example that results in the model outputting an incorrect result with high confidence

Note 1 to entry: Typically applies to ML models in the form of a *neural network* (3.1.48).

ITeH STANDARD PREVIEW
(standards.iteh.ai)

3.1.8

adversarial testing

testing approach based on the attempted creation and execution of *adversarial examples* (3.1.7) to identify defects in an *ML model* (3.1.46)

Note 1 to entry: Typically applied to ML models in the form of a *neural network* (3.1.48).

3.1.9

AI-based system

system including one or more components implementing *AI* (3.1.13)

3.1.10

AI effect

situation when a previously labelled *AI* (3.1.13) system is no longer considered to be AI as technology advances

3.1.11

AI quality metamodel

metamodel intended to ensure the quality of *AI-based systems* (3.1.9)

Note 1 to entry: This metamodel is defined in detail in DIN SPEC 92001.

3.1.12

algorithm

ML algorithm

<*machine learning* (3.1.43)> algorithm used to create an *ML model* (3.1.46) from the *training data* (3.1.80)

EXAMPLE ML algorithms include linear regression, logistic regression, *decision tree* (3.1.25), SVM, Naive Bayes, kNN, K-means and random forest.

3.1.13**artificial intelligence****AI**

capability of an engineered system to acquire, process and apply knowledge and skills

3.1.14**autonomous system**

system capable of working without human intervention for sustained periods

3.1.15**autonomy**

ability of a system to work for sustained periods without human intervention

3.1.16**back-to-back testing****differential testing**

approach to testing whereby an alternative version of the system is used as a *pseudo-oracle* (3.1.59) to generate expected results for comparison from the same test inputs

EXAMPLE The pseudo-oracle may be a system that already exists, a system developed by an independent team or a system implemented using a different programming language.

3.1.17**backward propagation**

<*neural network* (3.1.48)> method used in artificial neural networks to determine the weights to be used on the network connections based on the computed error at the output of the network

Note 1 to entry: It is used to train *deep neural networks* (3.1.27).

3.1.18**benchmark suite**

collection of benchmarks, where a benchmark is a set of tests used to compare the performance of alternatives

3.1.19**bias**

<*machine learning* (3.1.43)> measure of the distance between the predicted value provided by the *ML model* (3.1.46) and a desired fair *prediction* (3.1.56)

3.1.20**classification**

<*machine learning* (3.1.43)> machine learning function that predicts the output class for a given input

3.1.21**classifier**

<*machine learning* (3.1.43)> *ML model* (3.1.46) used for *classification* (3.1.20)

3.1.22**clustering**

grouping of a set of objects such that objects in the same group (i.e. a cluster) are more similar to each other than to those in other clusters

3.1.23**combinatorial testing**

black-box test design technique in which test cases are designed to execute specific combinations of values of several *parameters* (3.1.53)

EXAMPLE *Pairwise testing* (3.1.52), all combinations testing, each choice testing, base choice testing.

3.1.24

confusion matrix

table used to describe the performance of a classifier (3.1.21) on a set of *test data* (3.1.75) for which the true and false values are known

3.1.25

decision tree

<*machine learning* (3.1.43)> supervised-learning *model* (3.1.46) for which inference can be represented by traversing one or more tree-like structures

3.1.26

deep learning

approach to creating rich hierarchical representations through the training of *neural networks* (3.1.48) with one or more hidden layers

Note 1 to entry: Deep learning uses multi-layered networks of simple computing units (or “neurons”). In these neural networks each unit combines a set of input values to produce an output value, which in turn is passed on to other neurons downstream.

3.1.27

deep neural net

neural network (3.1.48) with more than two layers

3.1.28

deterministic system

system which, given a particular set of inputs and starting state, will always produce the same set of outputs and final state

3.1.29

distributional shift

dataset shift

<*machine learning* (3.1.43)> distance between the *training data* (3.1.80) distribution and the desired data distribution

Note 1 to entry: The effect of distributional shift often increases as the users' interaction with the system (and so the desired data distribution) changes over time.

3.1.30

drift

degradation

staleness

<*machine learning* (3.1.43)> changes to *ML model* (3.1.46) behaviour that occur over time

Note 1 to entry: These changes typically make *predictions* (3.1.56) less accurate and may require the model to be re-trained with new data.

3.1.31

explainability

<*AI* (3.1.13)> level of understanding how the *AI-based system* (3.1.9) came up with a given result

3.1.32

exploratory testing

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

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

3.1.33**F1-score**

<*machine learning* (3.1.43)> performance metric used to evaluate a *classifier* (3.1.21), which provides a balance (the harmonic average) between *recall* (3.1.61) and *precision* (3.1.55)

3.1.34**false negative**

incorrect reporting of a failure when in reality it is a pass

Note 1 to entry: This is also known as a Type II error.

EXAMPLE The referee awards an offside when it was a goal and so reports a failure to score a goal when a goal was scored.

3.1.35**false positive**

incorrect reporting of a pass when in reality it is a failure

Note 1 to entry: This is also known as a Type I error.

EXAMPLE The referee awards a goal that was offside and so should not have been awarded.

3.1.36**feature engineering****feature selection**

<*machine learning* (3.1.43)> activity in which those attributes in the raw data that best represent the underlying relationships that should appear in the *model* (3.1.46) are identified for use in the *training data* (3.1.80)

3.1.37**flexibility**

ability of a system to work in contexts outside its initial specification (i.e. change its behaviour according to its actual situation to satisfy its objectives)

3.1.38**fuzz testing**

software testing approach in which high volumes of random (or near random) data, called fuzz, are used to generate inputs to the test item

3.1.39**general AI**

strong AI

AI (3.1.13) that exhibits intelligent behaviour comparable to a human across the full range of cognitive abilities

3.1.40**graphical processing unit****GPU**

application-specific integrated circuit (ASIC) specialized for display functions such as rendering images

Note 1 to entry: GPUs are designed for parallel data processing of images with a single function, but this parallel processing is also useful for executing AI-based software, such as *neural networks* (3.1.48).

3.1.41**hyperparameter**

<*neural network* (3.1.48)> variable used to define the structure of a neural network and how it is trained

Note 1 to entry: Typically, hyperparameters are set by the developer of the *model* (3.1.46) and may also be referred to as a tuning *parameter* (3.1.53).

3.1.42

interpretability

<AI (3.1.13)> level of understanding how the underlying (AI) technology works

3.1.43

machine learning

ML

process using computational techniques to enable systems to learn from data or experience

3.1.44

metamorphic relation

description of how a change in the test inputs from the source test case to the follow-up test case affects a change (or not) in the expected outputs from the source test case to the follow-up test case

3.1.45

metamorphic testing

testing where the expected results are not based on the specification but are instead extrapolated from previous actual results

3.1.46

model

ML model

<machine learning (3.1.43)> output of a *ML algorithm* (3.1.12) trained with a training dataset that generates *predictions* (3.1.56) using patterns in the input data

3.1.47

narrow AI

weak AI

AI (3.1.13) focused on a single well-defined task to address a specific problem

iTeh STANDARD PREVIEW
(standards.iteh.ai)

3.1.48

neural network

artificial neural network

network of primitive processing elements connected by weighted links with adjustable weights, in which each element produces a value by applying a nonlinear function to its input values, and transmits it to other elements or presents it as an output value

ISO/IEC TR 29119-11:2020
<https://standards.iteh.ai/catalog/standards/sist/ff3c797a-5cda-4aa3-a3c9-73d784d6157b/iso-iec-tr-29119-11-2020>

Note 1 to entry: Whereas some neural networks are intended to simulate the functioning of neurons in the nervous system, most neural networks are used in *artificial intelligence* (3.1.13) as realizations of the *connectionist model* (3.1.46).

Note 2 to entry: Examples of nonlinear functions are a threshold function, a sigmoid function, and a polynomial function.

[SOURCE: ISO/IEC 2382:2015, 2120625, modified — The admitted term "neural net" has been removed; notes 3 to 5 to entry have been removed.]

3.1.49

neuron coverage

proportion of activated neurons divided by the total number of neurons in the *neural network* (3.1.48) (normally expressed as a percentage) for a set of tests

Note 1 to entry: A neuron is considered to be activated if its *activation value* (3.1.4) exceeds zero.

3.1.50

non-deterministic system

system which, given a particular set of inputs and starting state, will not always produce the same set of outputs and final state

3.1.51**overfitting**

<*machine learning* (3.1.43)> generation of a *ML model* (3.1.46) that corresponds too closely to the *training data* (3.1.80), resulting in a model that finds it difficult to generalize to new data

3.1.52**pairwise testing**

black-box test design technique in which test cases are designed to execute all possible discrete combinations of each pair of input *parameters* (3.1.53)

Note 1 to entry: Pairwise testing is the most popular form of *combinatorial testing* (3.1.23).

3.1.53**parameter**

<*machine learning* (3.1.43)> parts of the *model* (3.1.46) that are learnt from applying the *training data* (3.1.80) to the *algorithm* (3.1.12)

EXAMPLE Learnt weights in a neural net.

Note 1 to entry: Typically, parameters are not set by the developer of the model.

3.1.54**performance metrics**

<*machine learning* (3.1.43)> metrics used to evaluate *ML models* (3.1.46) that are used for *classification* (3.1.20)

EXAMPLE Typical metrics include *accuracy* (3.1.2), *precision* (3.1.55), *recall* (3.1.61) and *F1-score* (3.1.33).

3.1.55**precision**

<*machine learning* (3.1.43)> performance metric used to evaluate a *classifier* (3.1.21), which measures the proportion of predicted positives that were correct

3.1.56**prediction**

<*machine learning* (3.1.43)> machine learning function that results in a predicted target value for a given input

EXAMPLE Includes *classification* (3.1.20) and *regression* (3.1.62) functions.

3.1.57**pre-processing**

<*machine learning* (3.1.43)> part of the ML workflow that transforms raw data into a state ready for use by the *ML algorithm* (3.1.12) to create the *ML model* (3.1.46)

Note 1 to entry: Pre-processing can include analysis, normalization, filtering, reformatting, imputation, removal of outliers and duplicates, and ensuring the completeness of the dataset.

3.1.58**probabilistic system**

system whose behaviour is described in terms of probabilities, such that its outputs cannot be perfectly predicted

3.1.59**pseudo-oracle**

derived test oracle

independently derived variant of the test item used to generate results, which are compared with the results of the original test item based on the same test inputs

Note 1 to entry: Pseudo-oracles are a useful alternative when traditional *test oracles* (3.1.76) are not available.

3.1.60

reasoning technique

<AI (3.1.13)> form of AI that generates conclusions from available information using logical techniques, such as deduction and induction

3.1.61

recall

sensitivity

<machine learning (3.1.43)> performance metric used to evaluate a classifier (3.1.21), which measures the proportion of actual positives that were predicted correctly

3.1.62

regression

<machine learning (3.1.43)> machine learning function that results in a numerical or continuous output value for a given input

3.1.63

regulatory standard

standard promulgated by a regulatory agency

3.1.64

reinforcement learning

<machine learning (3.1.43)> task of building a ML model (3.1.46) using a process of trial and reward to achieve an objective

Note 1 to entry: A reinforcement learning task can include the training of a ML model in a way similar to supervised learning (3.1.74) plus training on unlabelled inputs gathered during the operation phase of the AI (3.1.13) system. Each time the model makes a prediction (3.1.56) a reward is calculated, and further trials are run to optimize the reward.

Note 2 to entry: In reinforcement learning, the objective, or definition of success, can be defined by the system designer.

<https://standards.iteh.ai/catalog/standards/sist/f3c797a-5cda-4aa3-a3c9-73d784d6157b/iso-iec-tr-29119-11-2020>

Note 3 to entry: In reinforcement learning, the reward can be a calculated number that represents how close the AI system is to achieving the objective for a given trial.

3.1.65

reward hacking

activity performed by an agent to maximise its reward function to the detriment of meeting the original objective

3.1.66

robot

programmed actuated mechanism with a degree of autonomy (3.1.15), moving within its environment, to perform intended tasks

Note 1 to entry: A robot includes the control system and interface of the control system.

Note 2 to entry: The classification (3.1.20) of robot into industrial robot or service robot is done according to its intended application.

3.1.67

safety

expectation that a system does not, under defined conditions, lead to a state in which human life, health, property, or the environment is endangered

[SOURCE: ISO/IEC/IEEE 12207:2017, 3.1.48]

3.1.68

search algorithm

<AI (3.1.13)> algorithm (3.1.12) that systematically visits a subset of all possible states (or structures) until the goal state (or structure) is reached

3.1.69**self-learning system**

adaptive system that changes its behaviour based on learning from the practice of trial and error

3.1.70**sign change coverage**

proportion of neurons activated with both positive and negative *activation values* (3.1.4) divided by the total number of neurons in the *neural network* (3.1.48) (normally expressed as a percentage) for a set of tests

Note 1 to entry: An activation value of zero is considered to be a negative activation value.

3.1.71**sign-sign coverage**

coverage level achieved if by changing the sign of each neuron it can be shown to individually cause one neuron in the next layer to change sign while all other neurons in the next layer stay the same (i.e. they do not change sign)

3.1.72**simulator**

<testing> device, computer program or system used during testing, which behaves or operates like a given system when provided with a set of controlled inputs.

3.1.73**software agent**

digital entity that perceives its environment and takes actions that maximize its chance of successfully achieving its goals

3.1.74**supervised learning**

<*machine learning* (3.1.43)> task of learning a function that maps an input to an output based on labelled example input-output pairs

3.1.75**test data**

<*machine learning* (3.1.43)> independent dataset used to provide an unbiased evaluation of the final, tuned *ML model* (3.1.46)

3.1.76**test oracle**

source of information for determining whether a test has passed or failed

Note 1 to entry: The test oracle is often a specification used to generate expected results for individual test cases, but other sources may be used, such as comparing actual results with those of another similar program or system or asking a human expert.

3.1.77**test oracle problem**

challenge of determining whether a test has passed or failed for a given set of test inputs and state

3.1.78**test scenario**

situation or setting for a test item used as the basis for generating test cases

3.1.79**threshold coverage**

<*neural network* (3.1.48)> proportion of neurons exceeding a threshold *activation value* (3.1.4) divided by the total number of neurons in the neural network (normally expressed as a percentage) for a set of tests

Note 1 to entry: A threshold activation value between 0 and 1 is chosen as the threshold value.