# ISTQB GLOSSARY

## Version 3.0 2020-04-28

**Based on ”Standard Glossary of Terms Used in Software Testing, version 3.2”**

## Scope

This glossary is a copy of the ISTQB Standard Glossary of Terms Used in Software Testing issued by the ISTQB Glossary Working Group.

The ISTQB Standard Glossary of Terms Used in Software Testing contains the definitions of testing terms used in the different ISTQB syllabi. This includes all terms stated as keywords in the ISTQB syllabi, as well as other terms of major importance.

The ISTQB Glossary focuses on terms that have a specific meaning in testing. Some related non-testing terms are also included if they play a major role in testing, such as terms used in software quality assurance and software lifecycle models. However, most terms of other software engineering disciplines are not covered in this document, even if they are used in various ISTQB syllabi.

## Purpose of the ISTQB Glossary

The ISTQB Glossary has two main objectives:

- Support the ISTQB syllabi by defining the terms used in the various syllabi consistently;
- Support communication within the international testing community and with its stakeholders by providing a standard testing vocabulary.

In compiling this Glossary, the ISTQB Glossary Working Group has sought the views and comments of a broad spectrum of opinion in industry, commerce and government bodies and organizations, with the aim of producing an international testing standard that would gain wide acceptance. Total agreement will rarely, if ever, be achieved in compiling a document of this nature. Contributions to this glossary have been received from testing communities from all over the world.

Being written in English, the current version of the Glossary is designed to also support other languages. ISTQB Member Boards are encouraged to incorporate their translations.

## Glossary Structure

The glossary has been arranged in a single section of terms and their definitions, ordered alphabetically. For each term, the following additional attributes are shown where applicable:

- Ref: without the addition of “after”, e.g., ISO 25010, this means that the exact definition of the reference is used. In case of minor changes used to adapt the definition to the context of the ISTQB Glossary, the addition “after” is used, e.g., Ref: After ISO 25010. The complete list of references used in the ISTQB Glossary is listed below.
- Synonym: Some terms are preferred to other synonymous ones, in which case, the preferred term appears as an entry, with the synonyms indicated.
- See also: These entries contain cross-references to related terms. Such cross-references are indicated for relationships such as broader term to a narrower term and overlapping meaning between two terms.

## Acknowledgements

This Glossary has been produced by the Glossary Working Group of the International Software Testing Qualifications Board (ISTQB).

At the time the Glossary version 3.2 was completed the Glossary Working Group had the following members (alphabetic order):

Tobias Ahlgren (Sweden), Vineta Arnicane (Latvia), Armin Beer (Austria), Armin Born (Switzerland), Mette Bruhn-Pedersen (Denmark), Gergory Collina (USA), Matthias Daigl (Germany), Ernst Dűring (Norway), George Fialkovitz (Brazil), Matthias Hamburg (Chair, Germany), Tamás Horváth (Hungary), Leanne Howard (Australia), Ian Howles (Great Britain), Marek Majernik (Slovakia), Gustavo Marquez Sosa (Spain), Judy McKay (USA), Gary Mogyorodi (Vice-Chair, Canada), Ana Paiva (Portugal), Juha Pomppu (Finnland), Meile Posthuma (Netherlands). Adam Roman (Poland), Lucjan Stapp (Poland), Karolina Zmitrowitz (Poland).

It is our concern to recognize the pioneering merits of Erik van Veenendaal who has designed the first version of this Glossary and who conducted the Glossary Working Group during many years, from its beginnings until 2014.

Our special thanks go to Nicholas Humphries for the development of the interactive application.

Many more people, who are not mentioned here by name, have contributed to different versions of this Glossary. The editors would like to thank them all for their contributions.

# Definitions

**abnormal end:** The unintended termination of the execution of a component or system prior to completion.
Reference: After ISO 24765

**abuse case:** A use case in which some actors with malicious intent are causing harm to the system or to other actors.
See also: use case

**acceptance criteria:** The criteria that a component or system must satisfy in order to be accepted by a user, customer, or other authorized entity.
Reference: ISO 24765

**acceptance test-driven development:** (ATDD) A collaborative approach to development in which the team and customers are using the customers own domain language to understand their requirements, which forms the basis for testing a component or system.

**acceptance testing:** A test level that focuses on determining whether to accept the system.

**accessibility:** The degree to which a component or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.
Reference: After ISO 25010

**accessibility testing:** Testing to determine the ease by which users with disabilities can use a component or system.
Reference: Gerrard

**account harvesting:** The process of obtaining user account information based on trial and error with the intention of using that information in a security attack.

**accountability:** The degree to which the actions of an entity can be traced uniquely to that entity.
Reference: After ISO 25010

**accuracy testing:** Testing to determine the accuracy of a software product.

**acting (IDEAL):** The phase within the IDEAL model where the improvements are developed, put into practice, and deployed across the organization. The acting phase consists of the activities: create solution, pilot/test solution, refine solution and implement solution.
See also: IDEAL

**actor:** User or any other person or system that interacts with the test object in a specific way.

**actual result:** The behavior produced/observed when a component or system is tested.

**ad hoc review:** A review technique performed informally without a structured process.
Reference: After ISO 20246

**ad hoc testing:** Testing carried out informally. No formal test preparation takes place, no recognized test design technique is used, there are no expectations for results and arbitrariness guides the test execution activity.

**adaptability:** The degree to which a component or system can be adapted for different or evolving hardware, software or other operational or usage environments.
Reference: After ISO 25010

**Agile Manifesto:** A statement on the values that underpin Agile software development. The values are: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, responding to change over following a plan.

**Agile software development:** A group of software development methodologies based on iterative incremental development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.

**Agile testing:** Testing practice for a project using Agile software development methodologies, incorporating techniques and methods, such as extreme programming (XP), treating development as the customer of testing and emphasizing the test-first design paradigm.
See also: test-driven development

**alpha testing:** A type of acceptance testing performed in the developer's test environment by roles outside the development organization.

**analytical test strategy:** A test strategy whereby the test team analyzes the test basis to identify the test conditions to cover.

**analytical testing:** Testing based on a systematic analysis of e.g., product risks or requirements.

**analyzability:** The degree to which an assessment can be made for a component or system of either the impact of one or more intended changes, the diagnosis of deficiencies or causes of failures, or the identification of parts to be modified.
Reference: After ISO 25010

**anomaly:** Any condition that deviates from expectation based on requirements specifications, design documents, user documents, standards, etc., or from someone's perception or experience. Anomalies may be found during, but not limited to, reviewing, testing, analysis, compilation, or use of software products or applicable documentation.
Reference: IEEE 1044

**anti-malware:** Software that is used to detect and inhibit malware.
See also: malware

**anti-pattern:** Repeated action, process, structure or reusable solution that initially appears to be beneficial and is commonly used but is ineffective and/or counterproductive in practice.

**API testing:** Testing performed by submitting commands to the software under test using programming interfaces of the application directly.

**Application Programming Interface:** (API) A type of interface in which the components or systems involved exchange information in a defined formal structure.

**appropriateness recognizability:** The degree to which users can recognize whether a component or system is appropriate for their needs.
Reference: After ISO 25010

**assessment report:** A document summarizing the assessment results, e.g., conclusions, recommendations and findings.
See also: process assessment

**assessor:** A person who conducts an assessment. Any member of an assessment team.

**atomic condition:** A condition that does not contain logical operators.

**attack vector:** A path or means by which an attacker can gain access to a system for malicious purposes.

**attack-based testing:** An experience-based testing technique that uses software attacks to induce failures, particularly security related failures.

**attacker:** A person or process that attempts to access data, functions or other restricted areas of the system without authorization, potentially with malicious intent.
See also: hacker

**audio testing:** Testing to determine if the game music and sound effects will engage the user in the game and enhance the game play.

**audit:** An independent examination of a work product, process, or set of processes that is performed by a third party to assess compliance with specifications, standards, contractual agreements, or other criteria.
Reference: After IEEE 1028

**authentication:** A procedure determining whether a person or a process is, in fact, who or what it is declared to be.
See also: authorization

**authenticity:** The degree to which the identity of a subject or resource can be proved to be the one claimed.
Reference: ISO 25010

**authorization:** Permission given to a user or process to access resources.
See also: authentication

**automated testware:** Testware used in automated testing, such as tool scripts.

**automation code defect density:** Defect density of a component of the test automation code.
See also: defect density

**automotive safety integrity level:** (ASIL) One of four levels that specify the item's or element's necessary requirements of ISO 26262 and safety measures to avoid an unreasonable residual risk.
Reference: ISO 26262

**automotive SPICE:** (ASPICE) A process reference model and an associated process assessment model in the automotive industry that conforms with the requirements of ISO/IEC 33002:2015.
Reference: Automotive SPICE

**availability:** The degree to which a component or system is operational and accessible when required for use.
Reference: After ISO 25010

**back-to-back-testing:** Testing to compare two or more variants of a test item or a simulation model of the same test item by executing the same test cases on all variants and comparing the results.
Reference: Spillner

**balanced scorecard:** A strategic tool for measuring whether the operational activities of a company are aligned with its objectives in terms of business vision and strategy.
See also: corporate dashboard, scorecard

**behavior:** The response of a component or system to a set of input values and preconditions.

**behavior-driven development:** (BDD) A collaborative approach to development in which the team is focusing on delivering expected behavior of a component or system for the customer, which forms the basis for testing.

**benchmark test:** (1) A standard against which measurements or comparisons can be made. (2) A test that is used to compare components or systems to each other or to a standard as in (1).
Reference: After IEEE 610

**best practice:** A superior method or innovative practice that contributes to the improved performance of an organization under given context, usually recognized as "best" by other peer organizations.

**beta testing:** A type of acceptance testing performed at an external site to the developer's test environment by roles outside the development organization.

**big-bang testing:** An integration test approach in which software elements, hardware elements, or both are combined all at once into a component or an overall system, rather than in stages.
Reference: After ISO 24765

**black-box test technique:** A test technique based on an analysis of the specification of a component or system.

**black-box testing:** Testing, either functional or non-functional, without reference to the internal structure of the component or system.

**blocked test case:** A test case that cannot be executed because the preconditions for its execution are not fulfilled.

**botnet:** A network of compromised computers, called bots or robots, which is controlled by a third party and used to transmit malware or spam, or to launch attacks.

**bottom-up testing:** An incremental approach to integration testing where the lowest level components are tested first, and then used to facilitate the testing of higher level components. This process is repeated until the component at the top of the hierarchy is tested.

**boundary value:** A minimum or maximum value of an ordered equivalence partition.

**boundary value analysis:** A black-box test technique in which test cases are designed based on boundary values.

**boundary value coverage:** The coverage of boundary values.

**branch:** A transfer of control from a decision point.

**branch coverage:** The coverage of branches.

**branch testing:** A white-box test technique in which test cases are designed to exercise branches.

**bug hunting:** An approach to testing in which gamification and awards for defects found are used as a motivator.

**build verification test:** (BVT) A set of automated tests which validates the integrity of each new build and verifies its key/core functionality, stability and testability.

**burndown chart:** A publicly displayed chart that depicts the outstanding effort versus time in an iteration. It shows the status and trend of completing the tasks of the iteration. The X-axis typically represents days in the sprint, while the Y-axis is the remaining effort (usually either in ideal engineering hours or story points).

**business process-based testing:** An approach to testing in which test cases are designed based on descriptions and/or knowledge of business processes.

**call graph:** An abstract representation of calling relationships between subroutines in a program.

**Capability Maturity Model Integration:** (CMMI) A framework that describes the key elements of an effective product development and maintenance process. The Capability Maturity Model Integration covers best-practices for planning, engineering and managing product development and maintenance.
Reference: CMMI

**capacity:** The degree to which the maximum limits of a component or system parameter meet requirements.
Reference: After ISO 25010

**capture/playback:** A test automation approach in which inputs to the test object are recorded during manual testing to generate automated test scripts that can be executed later.

**capture/playback tool:** A type of test execution tool where inputs are recorded during manual testing in order to generate automated test scripts that can be executed later (i.e. replayed). These tools are often used to support automated regression testing.

**CAST:** Acronym for Computer Aided Software Testing.
See also: test automation

**causal analysis:** The analysis of defects to determine their root cause.
Reference: CMMI

**cause-effect diagram:** A graphical representation used to organize and display the interrelationships of various possible root causes of a problem. Possible causes of a real or potential defect or failure are organized in categories and subcategories in a horizontal tree-structure, with the (potential) defect or failure as the root node.
Reference: After Juran

**cause-effect graph:** A graphical representation of inputs and/or stimuli (causes) with their associated outputs (effects), which can be used to design test cases.

**cause-effect graphing:** A black-box test technique in which test cases are designed from cause-effect graphs.
Reference: After ISO 29119

**certification:** The process of confirming that a component, system or person complies with its specified requirements.

**change management:** (1) A structured approach to transitioning individuals and organizations from a current state to a desired future state. (2) Controlled way to effect a change, or a proposed change, to a product or service.

**change-related testing:** A type of testing initiated by modification to a component or system.

**checklist-based review:** A review technique guided by a list of questions or required attributes.
Reference: ISO 20246

**checklist-based testing:** An experience-based test technique whereby the experienced tester uses a high-level list of items to be noted, checked, or remembered, or a set of rules or criteria against which a product has to be verified.

**classification tree:** A tree diagram representing test data domains of a test object.

**classification tree technique:** A black-box test technique in which test cases are designed using a classification tree.
Reference: Grochtmann

**CLI testing:** Testing performed by submitting commands to the software under test using a dedicated command-line interface.

**closed-loop-system:** A system in which the controlling action or input is dependent on the output or changes in output.
Reference: Bakshi

**code coverage:** The coverage of code.

**code injection:** A type of security attack performed by inserting malicious code at an interface into an application to exploit poor handling of untrusted data.
See also: malware scanning, SQL injection

**codependent behavior:** Excessive emotional or psychological dependence on another person, specifically in trying to change that person's current (undesirable) behavior while supporting them in continuing that behavior. For example, in software testing, complaining about late delivery to test and yet enjoying the necessary "heroism", working additional hours to make up time when delivery is running late, therefore reinforcing the lateness.

**coding standard:** A standard that describes the characteristics of a design or a design description of data or program components.
Reference: ISO 24765

**co-existence:** The degree to which a component or system can perform its required functions while sharing an environment and resources with other components or systems without a negative impact on any of them.
Reference: After ISO 25010

**collapsed decision table:** A decision table in which combinations of inputs that are impossible or lead to the same outputs are merged into one column (rule), by setting the conditions that do not influence the outputs to don't care.

**combinatorial testing:** A black-box test technique in which test cases are designed to exercise specific combinations of values of several parameters

**command-line interface:** (CLI) A type of interface in which the information is passed in form of command lines.

**commercial off-the-shelf:** (COTS) A type of product developed in an identical format for a large number of customers in the general market.

**compatibility:** The degree to which a component or system can exchange information with other components or systems, and/or perform its required functions while sharing the same hardware or software environment.
Reference: After ISO 25010

**compiler:** A computer program that translates programs expressed in a high-order language into their machine language equivalents.
Reference: ISO 24765

**complexity:** The degree to which a component or system has a design and/or internal structure that is difficult to understand, maintain and verify.
See also: cyclomatic complexity

**compliance:** Adherence of the component or system to standards, conventions or regulations in laws and similar prescriptions.

**compliance testing:** Testing to determine the compliance of the component or system.

**component:** A minimal part of a system that can be tested in isolation.

**component integration testing:** Testing in which the test items are interfaces and interactions between integrated components.

**component specification:** A description of a component's function in terms of its output values for specified input values under specified conditions, and required non-functional behavior (e.g., resource-utilization).

**component testing:** A test level that focuses on individual hardware or software components.

**compound condition:** Two or more single conditions joined by means of a logical operator.

**Computer Aided Software Engineering:** (CASE) The computing-based processes, techniques and tools used to design and implement a component or system.

**computer forensics:** The practice of determining how a security attack has succeeded and assessing the damage caused.

**concurrency:** The simultaneous execution of multiple independent threads by a component or system.

**condition:** A logical expression that can be evaluated as True or False.

**condition coverage:** The coverage of condition outcomes that have been exercised by a test suite.

**condition outcome:** The evaluation of a condition to True or False.

**condition testing:** A white-box test technique in which test cases are designed to exercise outcomes of atomic conditions.

**confidence interval:** In managing project risks, the period of time within which a contingency action must be implemented in order to be effective in reducing the impact of the risk.

**confidentiality:** The degree to which a component or system ensures that data are accessible only to those authorized to have access.
Reference: After ISO 25010

**configuration:** The composition of a component or system as defined by the number, nature, and interconnections of its constituent parts.

**configuration item:** An aggregation of work products that is designated for configuration management and treated as a single entity in the configuration management process.
Reference: ISO 24765

**configuration management:** A discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify that it complies with specified requirements.

**configuration management tool:** A tool that provides support for the identification and control of configuration items, their status over changes and versions, and the release of baselines consisting of configuration items.

**confirmation testing:** A type of change-related testing performed after fixing a defect to confirm that a failure caused by that defect does not reoccur.

**connectivity:** The degree to which a component or system can connect to other components or systems.
Reference: After ISO 2382

**consultative test strategy:** A test strategy whereby the test team relies on the input of one or more key stakeholders to determine the details of the strategy.

**consultative testing:** Testing driven by the advice and guidance of appropriate experts from outside the test team (e.g., technology experts and/or business domain experts).

**content-based model:** A process model providing a detailed description of good engineering practices, e.g., test practices.

**context of use:** Users, tasks, equipment (hardware, software and materials), and the physical and social environments in which a software product is used.
Reference: ISO 9241-11

**continuous integration:** A software development procedure merging, integrating and testing all changes as soon as they are committed within an automated process.

**continuous representation:** A capability maturity model structure wherein capability levels provide a recommended order for approaching process improvement within specified process areas.
Reference: CMMI

**continuous testing:** An approach that involves a process of testing early, testing often, test everywhere, and automate to obtain feedback on the business risks associated with a software release candidate as rapidly as possible.

**contractual acceptance testing:** A type of acceptance testing performed to verify whether a system satisfies its contractual requirements.

**control chart:** A statistical process control tool used to monitor a process and determine whether it is statistically controlled. It graphically depicts the average value and the upper and lower control limits (the highest and lowest values) of a process.

**control flow:** The sequence in which operations are performed by a business process, component or system.

**control flow analysis:** A type of static analysis based on a representation of unique paths for executing a component or system.

**control flow graph:** An abstract representation of all possible control flows through a component or system.

**control flow testing:** A white-box test technique in which test cases are designed based on control flows.

**convergence metric:** A metric that shows progress toward a defined criterion, e.g., convergence of the total number of tests executed to the total number of tests planned for execution.

**conversion testing:** Testing of software used to convert data from existing systems for use in replacement systems.

**corporate dashboard:** A dashboard-style representation of the status of corporate performance data.
See also: balanced scorecard, dashboard

**cost of quality:** The total costs incurred on quality activities and issues and often split into prevention costs, appraisal costs, internal failure costs and external failure costs.

**coverage:** The degree to which specified coverage items have been determined or have been exercised by a test suite expressed as a percentage.
Reference: After ISO 29119

**coverage analysis:** Measurement of achieved coverage to a specified coverage item during test execution referring to predetermined criteria to determine whether additional testing is required and if so, which test cases are needed.

**coverage item:** An attribute or combination of attributes that is derived from one or more test conditions by using a test technique that enables the measurement of the thoroughness of the test execution.
Reference: After ISO 29119

**coverage tool:** A tool that provides objective measures of what structural elements, e.g., statements, branches have been exercised by a test suite.

**critical success factor:** An element necessary for an organization or project to achieve its mission. Critical success factors are the critical factors or activities required for ensuring the success.

**Critical Testing Processes:** (CTP) A content-based model for test process improvement built around twelve critical processes. These include highly visible processes, by which peers and management judge competence and mission-critical processes in which performance affects the company's profits and reputation.
See also: content-based model

**cross-browser compatibility:** The degree to which a website or web application can function across different browsers and degrade gracefully when browser features are absent or lacking.

**cross-site scripting:** (XSS) A vulnerability that allows attackers to inject malicious code into an otherwise benign website.
Reference: NIST.IR.7298

**crowd testing:** An approach to testing in which testing is distributed to a large group of testers.

**custom software:** Software developed specifically for a set of users or customers. The opposite is commercial off-the-shelf software.

**custom tool:** A software tool developed specifically for a set of users or customers.

**cyclomatic complexity:** The maximum number of linear, independent paths through a program.
Reference: After McCabe

**daily build:** A software development activity in which a system is compiled and linked daily so that it is consistently available at any time including all the latest changes.

**dashboard:** A representation of dynamic measurements of operational performance for some organization or activity, using metrics represented via metaphors such as visual dials, counters, and other devices resembling those on the dashboard of an automobile, so that the effects of events or activities can be easily understood and related to operational goals.
See also: corporate dashboard, scorecard

**data definition:** An executable statement where a variable is assigned a value.

**data flow:** An abstract representation of the sequence and possible changes of the state of data objects, where the state of an object is any of creation, usage, or destruction.
Reference: Beizer

**data flow analysis:** A type of static analysis based on the lifecycle of variables.

**data flow coverage:** The percentage of definition-use pairs that have been exercised by a test suite.

**data flow testing:** A white-box test technique in which test cases are designed to execute definition-use pairs of variables.

**data obfuscation:** Data transformation that makes it difficult for a human to recognize the original data.

**data privacy:** The protection of personally identifiable information or otherwise sensitive information from undesired disclosure.

**database integrity testing:** Testing the methods and processes used to access and manage the data(base), to ensure access methods, processes and data rules function as expected and that during access to the database, data is not corrupted or unexpectedly deleted, updated or created.

**data-driven testing:** A scripting technique that uses data files to contain the test data and expected results needed to execute the test scripts.

**dd-path:** A path between two decisions of an algorithm, or two decision nodes of a corresponding graph, that includes no other decisions.

**debugging:** The process of finding, analyzing and removing the causes of failures in a component or system.

**debugging tool:** A tool used by programmers to reproduce failures, investigate the state of programs and find the corresponding defect. Debuggers enable programmers to execute programs step by step, to halt a program at any program statement and to set and examine program variables.

**decision:** A type of statement in which a choice between two or more possible outcomes controls which set of actions will result.
Reference: ISO 29119

**decision condition coverage:** The percentage of all condition outcomes and decision outcomes that have been exercised by a test suite. 100% decision condition coverage implies both 100% condition coverage and 100% decision coverage.

**decision condition testing:** A white-box test technique in which test cases are designed to execute condition outcomes and decision outcomes.

**decision coverage:** The coverage of decision outcomes.

**decision outcome:** The result of a decision that determines the next statement to be executed.

**decision table:** A table used to show sets of conditions and the actions resulting from them.
Reference: ISO 24765

**decision table testing:** A black-box test technique in which test cases are designed to exercise the combinations of conditions and the resulting actions shown in a decision table.

**decision testing:** A white-box test technique in which test cases are designed to execute decision outcomes.

**defect:** An imperfection or deficiency in a work product where it does not meet its requirements or specifications.
Reference: After ISO 24765

**defect density:** The number of defects per unit size of a work product.
Reference: After ISO 24765
Synonym: defect density

**defect detection percentage:** (DDP) The number of defects found by a test level, divided by the number found by that test level and any other means afterwards.
See also: escaped defect

**defect management:** The process of recognizing, recording, classifying, investigating, resolving and disposing of defects.

**defect management committee:** A cross-functional team of stakeholders who manage reported defects from initial detection to ultimate resolution (defect removal, defect deferral, or report cancellation). In some cases, the same team as the configuration control board.

**defect management tool:** A tool that facilitates the recording and status tracking of defects.

**defect masking:** An occurrence in which one defect prevents the detection of another.
Reference: After IEEE 610

**defect report:** Documentation of the occurrence, nature, and status of a defect.
See also: incident report

**defect taxonomy:** A list of categories designed to identify and classify defects.
Synonym: bug taxonomy

**defect-based test technique:** A test technique in which test cases are developed from what is known about a specific defect type.
Synonym: defect-based technique

**definition-use pair:** The association of a definition of a variable with the subsequent use of that variable.

**demilitarized zone:** (DMZ) A physical or logical subnetwork that contains and exposes an organization's external-facing services to an untrusted network, commonly the Internet.
See also: network zone

**Deming cycle:** An iterative four-step problem-solving process (plan-do-check-act) typically used in process improvement.
Reference: After Deming

**denial of service:** (DoS) A security attack that is intended to overload the system with requests such that legitimate requests cannot be serviced.

**design-based testing:** (e.g., tests of interfaces between components or systems&#41) An approach to testing in which test cases are designed based on the architecture and/or detailed design of a component or system &#40
Reference: .

**desk checking:** Testing of software or a specification by manual simulation of its execution.

**device-based testing:** A type of testing in which test suites are executed on physical or virtual devices.

**diagnosing (IDEAL):** The phase within the IDEAL model where it is determined where one is, relative to where one wants to be. The diagnosing phase consists of the activities to characterize current and desired states and develop recommendations.
See also: IDEAL

**discount usability testing:** A test strategy for usability testing that puts emphasis on keeping costs down without compromising too much on the quality of the usability evaluation.

**domain analysis:** A black-box test technique that is used to identify efficient and effective test cases when multiple variables can or should be tested together. It builds on and generalizes equivalence partitioning and boundary values analysis.

**driver:** A temporary component or tool that replaces another component and controls or calls a test item in isolation.

**dynamic analysis:** The process of evaluating a component or system based on its behavior during execution.
Reference: After ISO 24765

**dynamic analysis tool:** A tool that provides run-time information on the state of the software code. These tools are most commonly used to identify unassigned pointers, check pointer arithmetic and to monitor the allocation, use and de-allocation of memory and to flag memory leaks.

**dynamic comparison:** Comparison of actual and expected results, performed while the software is being executed, for example by a test execution tool.

**dynamic testing:** Testing that involves the execution of the test item.
Reference: After ISO 29119

**effectiveness:** Extent to which correct and complete goals are achieved.
Reference: ISO 9241
See also: efficiency

**efficiency:** The degree to which resources are expended in relation to the accuracy and completeness with which a user achieves goals.
Reference: After ISO 25010

**efficiency testing:** Testing to determine the efficiency of a software product.

**elementary comparison testing:** A black-box test design technique in which test cases are designed to execute combinations of inputs using the concept of modified condition / decision coverage.
Reference: TMap

**embedded iterative model:** A development lifecycle sub-model that applies an iterative approach to detailed design, coding and testing within an overall sequential model. In this case, the high-level design documents are prepared and approved for the entire project but the actual detailed design, code development and testing are conducted in iterations.

**emotional intelligence:** The ability, capacity, and skill to identify, assess, and manage the emotions of one's self, of others, and of groups.

**emulator:** A device, computer program, or system that accepts the same inputs and produces the same outputs as a given system.
Reference: ISO 24765

**encryption:** The process of encoding information so that only authorized parties can retrieve the original information, usually by means of a specific decryption key or process.

**endurance testing:** Testing to determine the stability of a system under a significant load over a significant period of time within the system's operational context.

**entry criteria:** The set of conditions for officially starting a defined task.
Reference: Gilb and Graham
See also: exit criteria

**entry point:** An executable statement or process step which defines a point at which a given process is intended to begin.

**environment model:** An abstraction of the real environment of a component or system including other components, processes, and environment conditions, in a real-time simulation.
Reference: Wallentowitz

**epic:** A large user story that cannot be delivered as defined within a single iteration or is large enough that it can be split into smaller user stories.
Reference: Agile Alliance

**equivalence partition:** A subset of the value domain of a variable within a component or system in which all values are expected to be treated the same based on the specification.
Reference: After ISO 29119

**equivalence partition coverage:** The coverage of equivalence partitions that have been exercised by a test suite.

**equivalence partitioning:** A black-box test technique in which test cases are designed to exercise equivalence partitions by using one representative member of each partition.
Reference: After ISO 29119

**equivalent manual test effort:** (EMTE) Effort required for running tests manually.

**error:** A human action that produces an incorrect result.
Reference: ISO 24765
Synonym: human mistake

**error guessing:** A test technique in which tests are derived on the basis of the tester's knowledge of past failures, or general knowledge of failure modes.
Reference: ISO 29119

**error tolerance:** The degree to which a component or system can continue normal operation despite the presence of erroneous inputs.
Reference: After ISO 24765

**escaped defect:** A defect that was not detected in a previous test level which is supposed to find such type of defects.

**establishing (IDEAL):** The phase within the IDEAL model where the specifics of how an organization will reach its destination are planned. The establishing phase consists of the activities set priorities, develop approach and plan actions.
See also: IDEAL

**ethical hacker:** A security tester using hacker techniques.

**European Foundation for Quality Management excellence model:** (EFQM) A non-prescriptive framework for an organization's quality management system, defined and owned by the European Foundation for Quality Management, based on five 'Enabling' criteria (covering what an organization does), and four 'Results' criteria (covering what an organization achieves).

**executable statement:** A statement which, when compiled, is translated into object code, and which will be executed procedurally when the program is running and may perform an action on data.

**exercised:** A program element is said to be exercised by a test case when the input value causes the execution of that element, such as a statement, decision, or other structural element.

**exhaustive testing:** A test approach in which the test suite comprises all combinations of input values and preconditions.

**exit criteria:** The set of conditions for officially completing a defined task.
Reference: After Gilb and Graham

**exit point:** An executable statement or process step which defines a point at which a given process is intended to cease.

**expected result:** The predicted observable behavior of a component or system executing under specified conditions, based on its specification or another source.
Reference: After ISO 29119

**experience-based test technique:** A test technique only based on the tester's experience, knowledge and intuition.

**experience-based testing:** Testing based on the tester's experience, knowledge and intuition.

**expert usability review:** An informal usability review in which the reviewers are experts. Experts can be usability experts or subject matter experts, or both.
See also: informal review

**exploratory testing:** An approach to testing whereby the testers dynamically design and execute tests based on their knowledge, exploration of the test item and the results of previous tests.
Reference: After ISO 29119

**Extreme Programming:** (XP) A software engineering methodology used within Agile software development whereby core practices are programming in pairs, doing extensive code review, unit testing of all code, and simplicity and clarity in code.
See also: Agile software development

**facilitator:** The leader and main person responsible for an inspection or review process.
Reference: After IEEE 1028

**fail:** A test is deemed to fail if its actual result does not match its expected result.

**failover testing:** Testing by simulating failure modes or actually causing failures in a controlled environment. Following a failure, the failover mechanism is tested to ensure that data is not lost or corrupted and that any agreed service levels are maintained (e.g., function availability or response times).
See also: recoverability testing

**failure:** An event in which a component or system does not perform a required function within specified limits.
Reference: After ISO 24765

**failure mode:** The physical or functional manifestation of a failure.
Reference: ISO 24765

**Failure Mode and Effect Analysis:** (FMEA) A systematic approach to risk identification and analysis of identifying possible modes of failure and attempting to prevent their occurrence.
See also: Failure Mode, Effect and Criticality Analysis
Synonym: Software Failure Mode and Effect Analysis

**Failure Mode, Effects, and Criticality Analysis:** (FMECA) An extension of FMEA, as in addition to the basic FMEA, it includes a criticality analysis, which is used to chart the probability of failure modes against the severity of their consequences. The result highlights failure modes with relatively high probability and severity of consequences, allowing remedial effort to be directed where it will produce the greatest value.

**failure rate:** The ratio of the number of failures of a given category to a given unit of measure.
Reference: ISO 24765

**false-negative result:** A test result which fails to identify the presence of a defect that is actually present in the test object.

**false-positive result:** A test result in which a defect is reported although no such defect actually exists in the test object.

**fault attack:** Directed and focused attempt to evaluate a specific quality characteristic of a test object by attempting to force specific failures to occur.
See also: negative testing, security attack

**fault injection:** The process of intentionally adding defects to a system for the purpose of finding out whether the system can detect, and possibly recover from, a defect. Fault injection is intended to mimic failures that might occur in the field.

**fault seeding:** The process of intentionally adding known faults to those already in a component or system to monitor the rate of detection and removal, and to estimate the number of faults remaining.
Reference: After ISO 24765
Synonym: fault injection

**fault seeding tool:** A tool for seeding (i.e., intentionally inserting) faults in a component or system.

**fault tolerance:** The degree to which a component or system operates as intended despite the presence of hardware or software faults.
Reference: After ISO 25010

**Fault Tree Analysis:** (FTA) A technique used to analyze the causes of faults (defects). The technique visually models how logical relationships between failures, human errors, and external events can combine to cause specific faults to disclose.
Synonym: SFTA (Software Fault Tree Analysis)

**feasible path:** A path for which a set of input values and preconditions exists which causes it to be executed.

**feature:** A distinguishing characteristic of a component or system.
Reference: After ISO 24765

**feature-driven development:** An iterative and incremental software development process driven from a client-valued functionality (feature) perspective. Feature-driven development is mostly used in Agile software development.
See also: Agile software development

**field testing:** A type of testing conducted to evaluate the system behavior under productive connectivity conditions in the field.

**finding:** A result of an evaluation that identifies some important issue, problem, or opportunity.

**finite state machine:** A computational model consisting of a finite number of states and transitions between those states, possibly with accompanying actions.
Reference: ISO 24765

**firewall:** A component or set of components that controls incoming and outgoing network traffic based on predetermined security rules.

**formal review:** A type of review that follows a defined process with a formally documented output.
Reference: ISO 20246

**formative evaluation:** A type of evaluation designed and used to improve the quality of a component or system, especially when it is still being designed.
See also: summative evaluation

**freedom from risk:** The degree to which a component or system mitigates the potential risk to economic status, living things, health, or the environment.
Reference: After ISO 25010

**frozen test basis:** A test basis document that can only be amended by a formal change control process.

**Function Point Analysis:** (FPA) Method aiming to measure the size of the functionality of an information system. The measurement is independent of the technology. This measurement may be used as a basis for the measurement of productivity, the estimation of the needed resources, and project control.

**functional appropriateness:** The degree to which the functions facilitate the accomplishment of specified tasks and objectives.
Reference: ISO 25010

**functional completeness:** The degree to which the set of functions covers all the specified tasks and user objectives.
Reference: ISO 25010

**functional correctness:** The degree to which a component or system provides the correct results with the needed degree of precision.
Reference: After ISO 25010

**functional integration:** An integration approach that combines the components or systems for the purpose of getting a basic functionality working early.
See also: integration testing

**functional requirement:** A requirement that specifies a function that a component or system must be able to perform.
Reference: ISO 24765

**functional safety:** The absence of unreasonable risk due to hazards caused by malfunctioning behavior of Electric/Electronic(E/E) - Systems.
Reference: ISO 26262

**functional suitability:** The degree to which a component or system provides functions that meet stated and implied needs when used under specified conditions.
Reference: After ISO 25010

**functional test design technique:** Procedure to derive and/or select test cases based on an analysis of the specification of the functionality of a component or system without reference to its internal structure.

**functional testing:** Testing performed to evaluate if a component or system satisfies functional requirements.
Reference: After ISO 24765

**functionality testing:** The process of testing to determine the functionality of a software product.

**fuzz testing:** A software testing technique used to discover security vulnerabilities by inputting massive amounts of random data, called fuzz, to the component or system.

**generic test automation architecture:** Representation of the layers, components, and interfaces of a test automation architecture, allowing for a structured and modular approach to implement test automation.

**Goal Question Metric:** (GQM) An approach to software measurement using a three-level model conceptual level (goal), operational level (question) and quantitative level (metric).

**graphical user interface:** (GUI) A type of interface that allows users to interact with a component or system through graphical icons and visual indicators.

**GUI testing:** Testing performed by interacting with the software under test via the graphical user interface.

**hacker:** A person or organization who is actively involved in security attacks, usually with malicious intent.
See also: attacker

**hardware in the loop:** (HiL) Dynamic testing performed using real hardware with integrated software in a simulated environment.
Reference: Automotive SPICE

**hardware-software integration testing:** Testing performed to expose defects in the interfaces and interaction between hardware and software components.
See also: integration testing

**hashing:** Transformation of a variable length string of characters into a usually shorter fixed-length value or key. Hashed values, or hashes, are commonly used in table or database lookups. Cryptographic hash functions are used to secure data.

**hazard analysis:** A technique used to characterize the elements of risk. The result of a hazard analysis will drive the methods used for development and testing of a system.
See also: risk analysis

**heuristic:** A generally recognized rule of thumb that helps to achieve a goal.

**heuristic evaluation:** An evaluation of a work product that uses a heuristic.

**high-level test case:** A test case with abstract preconditions, input data, expected results, postconditions, and actions (where applicable).

**horizontal traceability:** The tracing of requirements for a test level through the layers of test documentation (e.g., test plan, test design specification, test case specification and test procedure specification or test script).

**human-centered design:** An approach to design that aims to make software products more usable by focusing on the use of the software products and applying human factors, ergonomics, and usability knowledge and techniques.
Reference: ISO 9241-210

**hyperlink:** A pointer within a web page that leads to other web pages.

**hyperlink test tool:** A tool used to check that no broken hyperlinks are present on a web site.

**IDEAL:** An organizational improvement model that serves as a roadmap for initiating, planning, and implementing improvement actions. The IDEAL model is named for the five phases it describes: initiating, diagnosing, establishing, acting, and learning.

**impact analysis:** The identification of all work products affected by a change, including an estimate of the resources needed to accomplish the change.
Reference: After ISO 24765

**incident:** An event occurring that requires investigation.

**incident management:** The process of recognizing, recording, classifying, investigating, resolving and disposing of incidents.

**incident management tool:** A tool that facilitates the recording and status tracking of incidents.
See also: defect management tool

**incident report:** Documentation of the occurrence, nature, and status of an incident.
Reference: ISO 29119

**incremental development model:** A type of software development lifecycle model in which the component or system is developed through a series of increments.
Reference: After PMBOK

**incremental testing:** Testing where components or systems are integrated and tested one or some at a time, until all the components or systems are integrated and tested.

**independence of testing:** Separation of responsibilities, which encourages the accomplishment of objective testing.
Reference: After DO-178b

**independent test lab:** An organization responsible to test and certify that the software, hardware, firmware, platform, and operating system follow all the jurisdictional rules for each location where the product will be used.

**indicator:** A measure that provides an estimate or evaluation of specified attributes derived from a model with respect to defined information needs.
Reference: ISO 25040

**infeasible path:** A path that cannot be executed by any set of input values and preconditions.

**informal group review:** An informal review performed by three or more persons.
Reference: ISO 20246

**informal review:** A type of review that does not follow a defined process and has no formally documented output.

**information assurance:** Measures that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and non-repudiation. These measures include providing for restoration of information systems by incorporating protection, detection, and reaction capabilities.
Reference: NIST.IR.7298

**information security:** The protection of information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide confidentiality, integrity, and availability.
Reference: NIST.IR.7298

**initiating (IDEAL):** The phase within the IDEAL model where the groundwork is laid for a successful improvement effort. The initiating phase consists of the activities: set context, build sponsorship and charter infrastructure.
See also: IDEAL

**input:** Data received by a component or system from an external source.
Reference: ISO 24765

**input value:** An instance of an input.
See also: input

**insider threat:** A security threat originating from within the organization, often by an authorized system user.

**insourced testing:** Testing performed by people who are co-located with the project team but are not fellow employees.

**inspection:** A type of formal review to identify issues in a work product, which provides measurement to improve the review process and the software development process.
Reference: After ISO 20246

**installability:** The degree to which a component or system can be successfully installed and/or uninstalled in a specified environment.
Reference: After ISO 25010

**installability testing:** Testing the installability of a software product.

**installation guide:** Supplied instructions on any suitable media, which guides the installer through the installation process. This may be a manual guide, step-by-step procedure, installation wizard, or any other similar process description.

**installation wizard:** Supplied software on any suitable media which leads the installer through the installation procedure.

**instrumentation:** The insertion of additional code into the program in order to collect information about program behavior during execution, e.g., for measuring code coverage.

**instrumenter:** A software tool used to carry out instrumentation.

**intake test:** A special instance of a smoke test to decide if the component or system is ready for detailed and further testing. An intake test is typically carried out at the start of the test execution phase.

**integration:** The process of combining components or systems into larger assemblies.

**integration testing:** A test level that focuses on interactions between components or systems.

**integrity:** The degree to which a component or system allows only authorized access and modification to a component, a system or data.
Reference: After ISO 25010

**interface testing:** A type of integration testing performed to verify integration between components or systems.

**interoperability:** The degree to which two or more components or systems can exchange information and use the information that has been exchanged.
Reference: After ISO 25010

**interoperability testing:** Testing to determine the interoperability of a software product.

**intrusion detection system:** (IDS) A system which monitors activities on the 7 layers of the OSI model from network to application level, to detect violations of the security policy.
See also: malware scanning

**invalid testing:** Testing using input values that should be rejected by the component or system.
See also: error tolerance, negative testing

**isolation testing:** Testing of individual components in isolation from surrounding components, with surrounding components being simulated by stubs and drivers, if needed.

**iterative development model:** A type of software development lifecycle model in which the component or system is developed through a series of repeated cycles.

**keyword-driven testing:** A scripting technique in which test scripts contain high-level keywords and supporting files that contain low-level scripts that implement those keywords.

**lead assessor:** The person who leads an assessment. In some cases, for instance CMMI and TMMi when formal assessments are conducted, the lead assessor must be accredited and formally trained.

**learnability:** The degree to which a component or system can be used by specified users to achieve specified goals of learning with satisfaction and freedom from risk in a specified context of use.
Reference: After ISO 25010

**learning (IDEAL):** The phase within the IDEAL model where one learns from experiences and improves one's ability to adopt new processes and technologies in the future. The learning phase consists of the activities: analyze and validate, and propose future actions.
See also: IDEAL

**level of intrusion:** The level to which a test object is modified by adjusting it for testability.

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

**lifecycle model:** A description of the processes, workflows, and activities used in the development, delivery, maintenance, and retirement of a system.
Reference: CMMI

**linear scripting:** A simple scripting technique without any control structure in the test scripts.

**load generation:** The process of simulating a defined set of activities at a specified load to be submitted to a component or system.
See also: load testing

**load management:** The control and execution of load generation, and performance monitoring and reporting of the component or system.

**load profile:** Documentation defining a designated number of virtual users who process a defined set of transactions in a specified time period that a component or system being tested may experience in production.

**load testing:** A type of performance testing conducted to evaluate the behavior of a component or system under varying loads, usually between anticipated conditions of low, typical, and peak usage.
Reference: After ISO 29119
See also: performance testing, stress testing

**low-level test case:** A test case with concrete values for preconditions, input data, expected results and postconditions and detailed description of actions (where applicable).
See also: high-level test case

**maintainability:** The degree to which a component or system can be modified by the intended maintainers.
Reference: After ISO 25010

**maintainability testing:** Testing to determine the maintainability of a software product.

**maintenance:** The process of modifying a component or system after delivery to correct defects, improve quality characteristics, or adapt to a changed environment.
Reference: After ISO 24765

**maintenance testing:** Testing the changes to an operational system or the impact of a changed environment to an operational system.

**malware:** Software that is intended to harm a system or its components.

**malware scanning:** Static analysis aiming to detect and remove malicious code received at an interface.
See also: intrusion detection system

**management review:** A systematic evaluation of software acquisition, supply, development, operation, or maintenance process, performed by or on behalf of management that monitors progress, determines the status of plans and schedules, confirms requirements and their system allocation, or evaluates the effectiveness of management approaches to achieve fitness for purpose.
Reference: After IEEE 610, IEEE 1028

**man-in-the-middle attack:** The interception, mimicking and/or altering and subsequent relaying of communications (e.g., credit card transactions) by a third party such that a user remains unaware of that third party's presence.

**manufacturing-based quality:** A view of quality, whereby quality is measured by the degree to which a product or service conforms to its intended design and requirements. Quality arises from the process(es) used.
Reference: After Garvin
See also: product-based quality, transcendent-based quality, user-based quality, value-based quality

**master test plan:** A test plan that is used to coordinate multiple test levels or test types.
See also: test plan

**math testing:** Testing to determine the correctness of the pay table implementation, the random number generator results, and the return to player computations.

**maturity:** (1) The capability of an organization with respect to the effectiveness and efficiency of its processes and work practices. (2) The degree to which a component or system meets needs for reliability under normal operation.
Reference: ISO 25010

**maturity level:** Degree of process improvement across a predefined set of process areas in which all goals in the set are attained.
Reference: TMMi

**maturity model:** A structured collection of elements that describe certain aspects of maturity in an organization, and aid in the definition and understanding of an organization's processes.

**MBT model:** Any model used in model-based testing.

**mean time between failures:** (MTBF) The average time between failures of a component or system.

**mean time to repair:** (MTTR) The average time a component or system will take to recover from a failure.

**measure:** The number or category assigned to an attribute of an entity by making a measurement.
Reference: After ISO 25040

**measurement:** The process of assigning a number or category to an entity to describe an attribute of that entity.
Reference: After ISO 24765

**measurement scale:** A scale that constrains the type of data analysis that can be performed on it.
Reference: ISO 14598

**memory leak:** A memory access failure due to a defect in a program's dynamic store allocation logic that causes it to fail to release memory after it has finished using it.

**method table:** A table containing different test approaches, testing techniques and test types that are required depending on the Automotive Safety Integrity Level (ASIL) and on the context of the test object.
Reference: ISO 26262

**methodical test strategy:** A test strategy whereby the test team uses a pre-determined set of test conditions such as a quality standard, a checklist, or a collection of generalized, logical test conditions which may relate to a particular domain, application or type of testing.

**methodical testing:** Testing based on a standard set of tests, e.g., a checklist, a quality standard, or a set of generalized test cases.

**metric:** A measurement scale and the method used for measurement.

**milestone:** A point in time in a project at which defined (intermediate) deliverables and results should be ready.

**mind map:** A diagram arranged around a general theme that represents ideas, tasks, words or other items.

**model coverage:** The coverage of model elements.

**model in the loop:** (MiL) Dynamic testing performed using a simulation model of the system in a simulated environment.
Reference: Automotive SPICE

**model-based test strategy:** A test strategy whereby the test team derives testware from models.

**model-based testing:** (MBT) Testing based on or involving models.

**modeling tool:** A tool that supports the creation, amendment, and verification of models of the component or system.
Reference: Graham

**moderator:** (1) The person responsible for running review meetings. (2) The person who conducts a usability test session.

**modifiability:** The degree to which a component or system can be changed without introducing defects or degrading existing product quality.
Reference: After ISO 25010

**modified condition / decision coverage:** (MC/DC) The coverage of all single condition outcomes that independently affect a decision outcome that have been exercised by a test suite.

**modified condition / decision testing:** A white-box test technique in which test cases are designed to exercise single condition outcomes that independently affect a decision outcome.

**modularity:** The degree to which a system is composed of discrete components such that a change to one component has minimal impact on other components.
Reference: After ISO 25010

**monitoring tool:** A software tool or hardware device that runs concurrently with the component or system under test and supervises, records and/or analyzes the behavior of the component or system.
Reference: ISO 24765

**monkey testing:** Testing by means of a random selection from a large range of inputs and by randomly pushing buttons, ignorant of how the product is being used.

**multiplayer testing:** Testing to determine if many players can simultaneously interact with the casino game world, with computer-controlled opponents, game servers, and with each other, as expected according to the game design.

**multiple condition coverage:** The coverage of combinations of all single condition outcomes within one statement that have been exercised by a test suite.

**multiple condition testing:** A white-box test technique in which test cases are designed to exercise outcome combinations of atomic conditions.

**mutation analysis:** A method to determine test suite thoroughness by measuring the extent to which a test suite can discriminate the program from slight variants (mutants) of the program.

**mutation testing:** Testing in which two or more variants of a component or system are executed with the same inputs, the outputs compared, and analyzed in cases of discrepancies.

**Myers-Briggs Type Indicator:** (MBTI) An indicator of psychological preference representing the different personalities and communication styles of people.

**negative testing:** Tests aimed at showing that a component or system does not work. Negative testing is related to the tester's attitude rather than a specific test approach or test design technique, e.g., testing with invalid input values or exceptions.
Reference: After Beizer.

**neighborhood integration testing:** A type of integration testing in which all of the nodes that connect to a given node are the basis for the integration testing.

**network zone:** A sub-network with a defined level of trust. For example, the Internet or a public zone would be considered to be untrusted.

**non-conformity:** Non-fulfillment of a specified requirement.
Reference: ISO 9000

**non-functional requirement:** A requirement that describes how the component or system will do what it is intended to do.
Reference: After ISO 24765

**non-functional test design technique:** Procedure to derive and/or select test cases for non-functional testing based on an analysis of the specification of a component or system without reference to its internal structure.

**non-functional testing:** Testing performed to evaluate that a component or system complies with non-functional requirements.

**non-repudiation:** The degree to which actions or events can be proven to have taken place, so that the actions or events cannot be repudiated later.
Reference: After ISO 25010

**N-switch coverage:** The coverage of valid sequences of N+1 transitions that have been exercised by a test suite.
Reference: Chow

**N-switch testing:** A form of state transition testing in which test cases are designed to execute all valid sequences of N+1 transitions.
Reference: Chow

**n-wise testing:** A black-box test design technique in which test cases are designed to execute all possible discrete combinations of any set of n input parameters.

**offline MBT:** Model-based test approach whereby test cases are generated into a repository for future execution.

**online MBT:** Model-based test approach whereby test cases are generated and executed simultaneously.

**open source tool:** A software tool that is available to all potential users in source code form, usually via the internet. Its users are permitted, usually under license, to study, change, improve and, at times, to distribute the software.

**openâ€“loop-system:** A system in which controlling action or input is independent of the output or changes in output.
Reference: Bakshi

**operability:** The degree to which a component or system has attributes that make it easy to operate and control.
Reference: After ISO 25010

**operational acceptance testing:** A type of acceptance testing performed to determine if operations and/or systems administration staff can accept a system.
See also: operational testing

**operational environment:** Hardware and software products installed at users' or customers' sites where the component or system under test will be used. The software may include operating systems, database management systems, and other applications.

**operational profile:** An actual or predicted pattern of use of the component or system.

**operational profile testing:** Statistical testing using a model of system operations (short duration tasks) and their probability of typical use.
Reference: Musa

**operational profiling:** The process of developing and implementing an operational profile.
See also: operational profile

**operational testing:** Testing performed to evaluate a component or system in its operational environment.
Reference: ISO 24765

**orthogonal array:** A 2-dimensional array constructed with special mathematical properties, such that choosing any two columns in the array provides every pair combination of each number in the array.

**orthogonal array testing:** A systematic way of testing all-pair combinations of variables using orthogonal arrays. It significantly reduces the number of all combinations of variables to test all pair combinations.

**output:** Data transmitted by a component or system to an external destination.
Reference: After ISO 24765

**output value:** An instance of an output.

**outsourced testing:** Testing performed by people who are not co-located with the project team and are not fellow employees.

**pacing time:** The defined time delay between iterations of the test scenario execution.

**pair programming:** An agile software development practice in which two programmers work together on one workstation.
Reference: extremeprogramming.org

**pair testing:** Two persons, e.g., two testers, a developer and a tester, or an end-user and a tester, working together to find defects. Typically, they share one computer and trade control of it while testing.

**pairwise integration testing:** A type of integration testing that targets pairs of components that work together as shown in a call graph.

**pairwise testing:** A black-box test technique in which test cases are designed to exercise pairs of parameter-value pairs.
Reference: After ISO 29119-4

**par sheet testing:** Testing to determine that the game returns the correct mathematical results to the screen, to the players' accounts, and to the casino account.

**Pareto analysis:** A statistical technique in decision making that is used for selection of a limited number of factors that produce significant overall effect. In terms of quality improvement, a large majority of problems (80%) are produced by a few key causes (20%).

**pass:** A test is deemed to pass if its actual result matches its expected result.

**pass/fail criteria:** Decision rules used to determine whether a test item has passed or failed.
Reference: After ISO 29119

**password cracking:** A security attack recovering secret passwords stored in a computer system or transmitted over a network.
Reference: after NIST.IR.7298

**path:** A sequence of events, e.g., executable statements, of a component or system from an entry point to an exit point.

**path coverage:** The coverage of paths.

**path sensitizing:** Choosing a set of input values to force the execution of a given path.

**path testing:** A white-box test technique in which test cases are designed to execute paths.

**peak load:** The maximum operating capacity of a component or system.

**peer review:** A type of review of work products performed by others qualified to do the same work.
Reference: After ISO 20246

**penetration testing:** A testing technique aiming to exploit security vulnerabilities (known or unknown) to gain unauthorized access.

**performance efficiency:** The degree to which a component or system uses time, resources and capacity when accomplishing its designated functions.
Reference: After ISO 25010

**performance indicator:** A metric that supports the judgment of process performance.
Reference: After ISO 33001

**performance profiling:** The process of analyzing and tuning the performance efficiency of a component or system.

**performance testing:** Testing to determine the performance efficiency of a component or system.

**performance testing tool:** A test tool that generates load for a designated test item and that measures and records its performance during test execution.

**perspective-based reading:** A review technique in which a work product is evaluated from the perspective of different stakeholders with the purpose to derive other work products.

**pharming:** A security attack intended to redirect a web site's traffic to a fraudulent web site without the user's knowledge or consent.

**phase containment:** The percentage of defects that are removed in the same phase of the software lifecycle in which they were introduced.

**phase test plan:** A test plan that typically addresses one test phase.

**phishing:** An attempt to acquire personal or sensitive information by masquerading as a trustworthy entity in an electronic communication.

**planning poker:** A consensus-based estimation technique, mostly used to estimate effort or relative size of user stories in Agile software development. It is a variation of the Wideband Delphi method using a deck of cards with values representing the units in which the team estimates.
Reference: Mountain Goat Software

**player perspective testing:** Testing done by testers from a player's perspective to validate player satisfaction.

**pointer:** A data item that specifies the location of another data item.
Reference: ISO 24765

**portability:** The degree to which a component or system can be transferred from one hardware, software or other operational or usage environment to another.
Reference: After ISO 25010

**portability testing:** Testing to determine the portability of a component or system.

**postcondition:** The expected state of a test item and its environment at the end of test case execution.

**post-execution comparison:** Comparison of actual and expected results, performed after the software has finished running.

**post-release testing:** A type of testing to ensure that the release is performed correctly and the application can be deployed.

**precondition:** The required state of a test item and its environment prior to test case execution.

**predicate:** A logical expression which evaluates to true or false to direct the execution path.
Reference: ISO 29119-4

**priority:** The level of (business) importance assigned to an item, e.g., defect.

**PRISMA:** A systematic approach to risk-based testing that employs product risk identification and analysis to create a product risk matrix based on likelihood and impact. Term is derived from Product RISk MAnagement.

**probe effect:** An unintended change in behavior of a component or system caused by measuring it.

**problem:** An unknown underlying cause of one or more incidents.
Reference: ISO 24765

**process:** A set of interrelated activities, which transform inputs into outputs.
Reference: ISO 12207

**process assessment:** A disciplined evaluation of an organization's software processes against a reference model.
Reference: after ISO 15504

**process cycle test:** A black-box test design technique in which test cases are designed to execute business procedures and processes.
Reference: TMap

**process improvement:** A program of activities designed to improve the performance and maturity of the organization's processes, and the result of such a program.
Reference: CMMI

**process model:** A framework in which processes of the same nature are classified into an overall model.

**process reference model:** A process model providing a generic body of best practices for comparison with actual processes, together with the recommended steps for improving actual processes to conform more closely with the reference model.

**process-compliant test strategy:** A test strategy whereby the test team follows a set of predefined processes, whereby the processes address such items as documentation, the proper identification and use of the test basis and test oracle(s), and the organization of the test team.

**process-compliant testing:** Testing that follows a set of defined processes, e.g., defined by an external party such as a standards committee.

**process-driven scripting:** A scripting technique where scripts are structured into scenarios which represent use cases of the software under test. The scripts can be parameterized with test data.

**product risk:** A risk impacting the quality of a product.
See also: risk

**product-based quality:** A view of quality, wherein quality is based on a well-defined set of quality characteristics. These characteristics must be measured in an objective and quantitative way. Differences in the quality of products of the same type can be traced back to the way the specific quality characteristics have been implemented.

**project:** A project is a unique set of coordinated and controlled activities with start and finish dates undertaken to achieve an objective conforming to specific requirements, including the constraints of time, cost and resources.
Reference: ISO 9000

**project retrospective:** A structured way to capture lessons learned and to create specific action plans for improving on the next project or next project phase.

**project risk:** A risk that impacts project success.
See also: risk

**protocol:** A set of conventions that govern the interaction of processes, devices, and other components within a system.
Reference: ISO 24765

**proximity-based testing:** A type of testing to confirm that sensors can detect nearby objects without physical contact.

**pseudo-random:** A series which appears to be random but is in fact generated according to some prearranged sequence.

**qualification:** The process of demonstrating the ability to fulfill specified requirements. Note the term "qualified" is used to designate the corresponding status.
Reference: ISO 9000

**quality:** The degree to which a component or system satisfies the stated and implied needs of its various stakeholders.
Reference: After ISO 25010

**quality assurance:** (QA) Activities focused on providing confidence that quality requirements will be fulfilled.
Reference: After ISO 24765
See also: quality management

**quality characteristic:** A category of quality attributes that bears on work product quality.
Reference: ISO 24765

**quality control:** (QC) A set of activities designed to evaluate the quality of a component or system.
Reference: after ISO 24765
See also: testing

**quality function deployment:** (QFD) A facilitated workshop technique that helps determine critical characteristics for new product development.
Reference: ISO 24765

**quality gate:** A special milestone in a project. Quality gates are located between those phases of a project strongly depending on the outcome of a previous phase. A quality gate includes a formal check of the documents of the previous phase.

**quality management:** Coordinated activities to direct and control an organization with regard to quality that include establishing a quality policy and quality objectives, quality planning, quality control, quality assurance, and quality improvement.
Reference: After ISO 24765

**quality risk:** A product risk related to a quality characteristic.
See also: quality characteristic, product risk

**RACI matrix:** A matrix describing the participation by various roles in completing tasks or deliverables for a project or process. It is especially useful in clarifying roles and responsibilities. RACI is an acronym derived from the four key responsibilities most typically used: Responsible, Accountable, Consulted, and Informed.

**ramp-down:** A technique for decreasing the load on a system in a measurable and controlled way.

**ramp-up:** A technique for increasing the load on a system in a measurable and controlled way.

**random testing:** A black-box test technique in which test cases are designed by generating random independent inputs to match an operational profile.

**Rational Unified Process:** (RUP) A proprietary adaptable iterative software development process framework consisting of four project lifecycle phases: inception, elaboration, construction and transition.

**reactive test strategy:** A test strategy whereby the test team waits to design and implement tests until the software is received, reacting to the actual system under test.

**reactive testing:** Testing that dynamically responds to the system under test and test results being obtained. Typically reactive testing has a reduced planning cycle and the design and implementation test phases are not carried out until the test object is received.

**reconnaissance:** The exploration of a target area aiming to gain information that can be useful for an attack.

**recoverability:** The degree to which a component or system can recover the data directly affected by an interruption or a failure and re-establish the desired state of the component or system.
Reference: After ISO 25010

**recoverability testing:** Testing to determine the recoverability of a component or system.

**regression:** A degradation in the quality of a component or system due to a change.

**regression testing:** A type of change-related testing to detect whether defects have been introduced or uncovered in unchanged areas of the software.

**regression-averse test strategy:** A test strategy whereby the test team applies various techniques to manage the risk of regression such as functional and/or non-functional regression test automation at one or more levels.

**regression-averse testing:** Testing using various techniques to manage the risk of regression, e.g., by designing re-usable testware and by extensive automation of testing at one or more test levels.

**regulatory acceptance testing:** A type of acceptance testing performed to verify whether a system conforms to relevant laws, policies and regulations.

**release note:** A document identifying test items, their configuration, current status and other delivery information delivered by development to testing, and possibly other stakeholders, at the start of a test execution phase.
Reference: After IEEE 829

**reliability:** The degree to which a component or system performs specified functions under specified conditions for a specified period of time.
Reference: After ISO 25010

**reliability growth model:** A model that shows the growth in reliability over time of a component or system as a result of the defect removal.

**reliability testing:** Testing to determine the reliability of a component or system.

**remote test lab:** A facility that provides remote access to a test environment.

**replaceability:** The degree to which a component or system can replace another specified component or system for the same purpose in the same environment.
Reference: After ISO 25010

**requirement:** A provision that contains criteria to be fulfilled.
Reference: ISO 24765

**requirements management tool:** A tool that supports the recording of requirements, requirements attributes and annotation, and facilitates traceability through layers of requirements and requirements change management.

**requirements phase:** The period of time in the software lifecycle during which the requirements for a software product are defined and documented.
Reference: ISO 24765

**requirements-based testing:** An approach to testing in which test cases are designed based on requirements.

**resource utilization:** The degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.
Reference: After ISO 25010

**resource utilization testing:** Testing to determine the resource utilization of a component or system.

**resumption criteria:** The criteria used to restart all or a portion of the testing activities that were suspended previously.

**resumption requirements:** The defined set of testing activities that must be repeated when testing is re-started after a suspension.

**retrospective meeting:** A meeting at the end of a project during which the project team members evaluate the project and learn lessons that can be applied to the next project.

**reusability:** The degree to which a work product can be used in more than one system, or in building other work products.
Reference: After ISO 25010

**review:** A type of static testing in which a work product or process is evaluated by one or more individuals to detect defects or to provide improvements.

**review plan:** A document describing the approach, resources and schedule of intended review activities. It identifies, amongst others: documents and code to be reviewed, review types to be used, participants, as well as entry and exit criteria to be applied in case of formal reviews, and the rationale for their choice. It is a record of the review planning process.

**review tool:** A tool that provides support to the review process. Typical features include review planning and tracking support, communication support, collaborative reviews and a repository for collecting and reporting of metrics.

**reviewer:** A participant in a review, who identifies issues in the work product.
Reference: After ISO 20246

**risk:** A factor that could result in future negative consequences.

**risk analysis:** The overall process of risk identification and risk assessment.

**risk assessment:** The process to examine identified risks and determine the risk level.

**risk identification:** The process of finding, recognizing and describing risks.
Reference: ISO 31000

**risk impact:** The damage that will be caused if the risk becomes an actual outcome or event.

**risk level:** The qualitative or quantitative measure of a risk defined by impact and likelihood.

**risk likelihood:** The estimated probability that a risk will become an actual outcome or event.

**risk management:** The process for handling risks.
Reference: After ISO 24765

**risk mitigation:** The process through which decisions are reached and protective measures are implemented for reducing or maintaining risks to specified levels.

**risk type:** A set of risks grouped by one or more common factors.

**risk-based testing:** Testing in which the management, selection, prioritization, and use of testing activities and resources are based on corresponding risk types and risk levels.
Reference: After ISO 29119

**robustness:** The degree to which a component or system can function correctly in the presence of invalid inputs or stressful environmental conditions.
Reference: ISO 24765

**robustness testing:** Testing to determine the robustness of the software product.

**role-based reviewing:** A review technique in which a work product is evaluated from the perspective of different stakeholders.

**root cause:** A source of a defect such that if it is removed, the occurrence of the defect type is decreased or removed.
Reference: CMMI

**root cause analysis:** An analysis technique aimed at identifying the root causes of defects. By directing corrective measures at root causes, it is hoped that the likelihood of defect recurrence will be minimized.

**S.M.A.R.T. goal methodology:** (SMART) A methodology whereby objectives are defined very specifically rather than generically. SMART is an acronym derived from the attributes of the objective to be defined: Specific, Measurable, Attainable, Relevant and Timely.

**safety:** The capability that a system will not, under defined conditions, lead to a state in which human life, health, property, or the environment is endangered.
Reference: After ISO 24765

**safety critical system:** A system whose failure or malfunction may result in death or serious injury to people, or loss or severe damage to equipment, or environmental harm.

**safety testing:** Testing to determine the safety of a software product.

**salting:** A cryptographic technique that adds random data (salt) to the user data prior to hashing.
See also: hashing

**scalability:** The capability of the software product to be upgraded to accommodate increased loads.
Reference: After Gerrard

**scalability testing:** Testing to determine the scalability of the software product.

**scenario-based review:** A review technique in which a work product is evaluated to determine its ability to address specific scenarios.

**scorecard:** A representation of summarized performance measurements representing progress towards the implementation of long-term goals. A scorecard provides static measurements of performance over or at the end of a defined interval.

**scribe:** A person who records information during the review meetings.
Reference: After IEEE 1028

**script kiddie:** A person who executes security attacks that have been created by other hackers rather than creating one's own attacks.
See also: hacker

**scripted testing:** Testing (manual or automated) that follows a test script.

**scrum:** An iterative incremental framework for managing projects commonly used with Agile software development.
See also: Agile software development

**security:** The degree to which a component or system protects information and data so that persons or other components or systems have the degree of access appropriate to their types and levels of authorization.
Reference: After ISO 25010

**security attack:** An attempt to gain unauthorized access to a component or system, resources, information, or an attempt to compromise system integrity.
Reference: after NIST.IR.7298

**security audit:** An audit evaluating an organization's security processes and infrastructure.

**security policy:** A high-level document describing the principles, approach and major objectives of the organization regarding security.

**security procedure:** A set of steps required to implement the security policy and the steps to be taken in response to a security incident.

**security risk:** A quality risk related to security.

**security testing:** Testing to determine the security of the software product.

**security testing tool:** A tool that provides support for testing security characteristics and vulnerabilities.

**security tool:** A tool that supports operational security.

**security vulnerability:** A weakness in the system that could allow for a successful security attack.

**sequential development model:** A type of software development lifecycle model in which a complete system is developed in a linear way of several discrete and successive phases with no overlap between them.

**service virtualization:** A technique to enable virtual delivery of services which are deployed, accessed and managed remotely.

**session-based test management:** (SBTM) A method for measuring and managing session-based testing.

**session-based testing:** An approach in which test activities are planned as test sessions.

**severity:** The degree of impact that a defect has on the development or operation of a component or system.

**short-circuiting:** A programming language/interpreter technique for evaluating compound conditions in which a condition on one side of a logical operator may not be evaluated if the condition on the other side is sufficient to determine the final outcome.

**simulation:** The representation of selected behavioral characteristics of one physical or abstract system by another system.
Reference: ISO 2382

**simulator:** A device, computer program or system used during testing, which behaves or operates like a given system when provided with a set of controlled inputs.
Reference: ISO 24765

**smoke test:** A test suite that covers the main functionality of a component or system to determine whether it works properly before planned testing begins.

**social engineering:** An attempt to trick someone into revealing information (e.g., a password) that can be used to attack systems or networks.
Reference: NIST.IR.7298

**software:** Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.
Reference: ISO 24765

**software development lifecycle:** (SDLC) The activities performed at each stage in software development, and how they relate to one another logically and chronologically.

**software in the loop:** (SiL) Dynamic testing performed using real software in a simulated environment or with experimental hardware.
Reference: Automotive SPICE

**software integrity level:** The degree to which software complies or must comply with a set of stakeholder-selected software and/or software-based system characteristics (e.g., software complexity, risk assessment, safety level, security level, desired performance, reliability or cost) which are defined to reflect the importance of the software to its stakeholders.

**software lifecycle:** The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software lifecycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and sometimes, retirement phase. Note these phases may overlap or be performed iteratively.

**software process improvement:** (SPI) A program of activities designed to improve the performance and maturity of the organization's software processes and the results of such a program.
Reference: After CMMI

**software qualification test:** Testing performed on completed, integrated software to provide evidence for compliance with software requirements.
Reference: Automotive SPICE

**software quality:** The totality of functionality and features of a software product that bear on its ability to satisfy stated or implied needs.
Reference: After ISO 9126

**Software Usability Measurement Inventory:** (SUMI) A questionnaire-based usability testing tool that measures and benchmarks user experience.
Reference: Kirakowski93

**specification:** Documentation that provides a detailed description of a component or system for the purpose of developing and testing it.
Reference: After ISO 24765

**specification by example:** (SBE) A development technique in which the specification is defined by examples.
See also: acceptance test-driven development

**specified input:** An input for which the specification predicts a result.

**spike testing:** Testing to determine the ability of a system to recover from sudden bursts of peak loads and return to a steady state.

**SQL injection:** A security attack inserting malicious SQL statements into an entry field for execution.

**stability:** The degree to which a component or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.
Reference: ISO 25010

**staged representation:** ( each level builds a foundation for subsequent levels. ) A model structure wherein attaining the goals of a set of process areas establishes a maturity level
See also: CMMI

**standard:** Formal, possibly mandatory, set of requirements developed and used to prescribe consistent approaches to the way of working or to provide guidelines (e.g., ISO/IEC standards, IEEE standards, and organizational standards).
Reference: After CMMI

**standard-compliant test strategy:** A test strategy whereby the test team follows a standard. Standards followed may be valid e.g., for a country (legislation standards), a business domain (domain standards), or internally (organizational standards).

**standard-compliant testing:** Testing that complies to a set of requirements defined by a standard, e.g., an industry testing standard or a standard for testing safety-critical systems.

**state table:** A grid showing each possible event and the resulting valid and invalid transitions for each state.

**state transition:** A transition between two states of a component or system.

**state transition diagram:** A diagram that depicts the states that a component or system can assume, and shows the events or circumstances that cause and/or result from a change from one state to another.
Reference: After ISO 24765

**state transition testing:** A black-box test technique in which test cases are designed to exercise elements of a state transition model.

**statement:** An entity in a programming language, which is typically the smallest indivisible unit of execution.

**statement coverage:** The coverage of executable statements.

**statement testing:** A white-box test technique in which test cases are designed to execute statements.

**static analysis:** The process of evaluating a component or system without executing it, based on its form, structure, content, or documentation.
Reference: After ISO 24765

**static analyzer:** A tool that carries out static analysis.

**static code analysis:** The analysis of source code carried out without execution of that software.

**static testing:** Testing a work product without the work product code being executed.

**statistical testing:** A test design technique in which a model of the statistical distribution of the input is used to construct representative test cases.

**status accounting:** An element of configuration management consisting of the recording and reporting of information needed to manage a configuration effectively. This information includes a listing of the approved configuration identification, the status of proposed changes to the configuration, and the implementation status of the approved changes.
Reference: IEEE 610

**stress testing:** A type of performance testing conducted to evaluate a system or component at or beyond the limits of its anticipated or specified workloads, or with reduced availability of resources such as access to memory or servers.
Reference: ISO 24765

**stress testing tool:** A tool that supports stress testing.

**structural coverage:** Coverage measures based on the internal structure of a component or system.

**structured scripting:** A scripting technique that builds and utilizes a library of reusable (parts of) scripts.

**stub:** A skeletal or special-purpose implementation of a software component, used to develop or test a component that calls or is otherwise dependent on it. It replaces a called component.
Reference: After IEEE 610

**subpath:** A sequence of executable statements within a component.

**suitability testing:** Testing to determine the suitability of a software product.

**summative evaluation:** A type of evaluation designed and used to gather conclusions about the quality of a component or system, especially when a substantial part of it has completed design.
See also: formative evaluation, testing

**suspension criteria:** The criteria used to (temporarily) stop all or a portion of the testing activities on the test items.
Reference: After IEEE 829

**syntax testing:** A black-box test design technique in which test cases are designed based upon the definition of the input domain and/or output domain.

**system:** A collection of interacting elements organized to accomplish a specific function or set of functions.
Reference: After ISO 24765

**system hardening:** The step-by-step process of reducing the security vulnerabilities of a system by applying a security policy and different layers of protection.

**system integration testing:** A test level that focuses on interactions between systems.

**system of systems:** Multiple heterogeneous, distributed systems that are embedded in networks at multiple levels and in multiple interconnected domains, addressing large-scale inter-disciplinary common problems and purposes, usually without a common management structure.

**system qualification test:** Testing performed on the completed, integrated system of software components, hardware components, and mechanics to provide evidence for compliance with system requirements and that the complete system is ready for delivery.
Reference: Automotive SPICE

**system testing:** A test level that focuses on verifying that a system as a whole meets specified requirements.

**system throughput:** The amount of data passing through a component or system in a given time period.
Reference: After ISO 24765

**system under test:** (SUT) A type of test object that is a system.

**System Usability Scale:** (SUS) A simple, ten-item attitude scale giving a global view of subjective assessments of usability.

**Systematic Test and Evaluation Process:** (STEP) A structured testing methodology also used as a content-based model for improving the testing process. It does not require that improvements occur in a specific order.

**technical review:** A type of formal review by a team of technically-qualified personnel that examines the quality of a work product and identifies discrepancies from specifications and standards.

**test:** A set of one or more test cases.

**test adaptation layer:** The layer in a test automation architecture which provides the necessary code to adapt test scripts on an abstract level to the various components, configuration or interfaces of the SUT.

**test analysis:** The activity that identifies test conditions by analyzing the test basis.

**test approach:** The implementation of the test strategy for a specific project.

**test architect:** (1) A person who provides guidance and strategic direction for a test organization and for its relationship with other disciplines. (2) A person who defines the way testing is structured for a given system, including topics such as test tools and test data management.

**test automation:** The use of software to perform or support test activities.

**test automation architecture:** An instantiation of the generic test automation architecture to define the architecture of a test automation solution, i.e., its layers, components, services and interfaces.

**test automation engineer:** A person who is responsible for the design, implementation and maintenance of a test automation architecture as well as the technical evolution of the resulting test automation solution.

**test automation framework:** A tool that provides an environment for test automation. It usually includes a test harness and test libraries.

**test automation manager:** A person who is responsible for the planning and supervision of the development and evolution of a test automation solution.

**test automation solution:** A realization/implementation of a test automation architecture, i.e., a combination of components implementing a specific test automation assignment. The components may include commercial off-the-shelf test tools, test automation frameworks, as well as test hardware.

**test automation strategy:** A high-level plan to achieve long-term objectives of test automation under given boundary conditions.

**test basis:** The body of knowledge used as the basis for test analysis and design.
Reference: After TMap

**test case:** A set of preconditions, inputs, actions (where applicable), expected results and postconditions, developed based on test conditions.
Reference: After ISO 29119

**test case explosion:** The disproportionate growth of the number of test cases with growing size of the test basis, when using a certain test design technique. Test case explosion may also happen when applying the test design technique systematically for the first time.

**test case result:** The final verdict on the execution of a test and its outcomes, such as pass, fail, or error. The result of error is used for situations where it is not clear whether the problem is in the test object.

**test case specification:** Documentation of a set of one or more test cases.
Reference: ISO 29119
See also: test specification

**test charter:** Documentation of the goal or objective for a test session.

**test closure:** During the test closure phase of a test process data is collected from completed activities to consolidate experience, testware, facts and numbers. The test closure phase consists of finalizing and archiving the testware and evaluating the test process, including preparation of a test evaluation report.

**test comparator:** A test tool to perform automated test comparison of actual results with expected results.

**test comparison:** The process of identifying differences between the actual results produced by the component or system under test and the expected results for a test. Test comparison can be performed during test execution (dynamic comparison) or after test execution.

**test completion:** The activity that makes testware available for later use, leaves test environments in a satisfactory condition and communicates the results of testing to relevant stakeholders.

**test condition:** A testable aspect of a component or system identified as a basis for testing.
Reference: After ISO 29119

**test control:** The activity that develops and applies corrective actions to get a test project on track when it deviates from what was planned.

**test cycle:** Execution of the test process against a single identifiable release of the test object.

**test data:** Data needed for test execution.

**test data management:** The process of analyzing test data requirements, designing test data structures, creating and maintaining test data.

**test data preparation:** The activity to select data from existing databases or create, generate, manipulate and edit data for testing.

**test data preparation tool:** A type of test tool that enables data to be selected from existing databases or created, generated, manipulated and edited for use in testing.

**test definition layer:** The layer in a generic test automation architecture which supports test implementation by supporting the definition of test suites and/or test cases, e.g., by offering templates or guidelines.

**test design:** The activity that derives and specifies test cases from test conditions.
Reference: After ISO 29119

**test design specification:** Documentation specifying the features to be tested and their corresponding test conditions.
Reference: ISO 29119
See also: test specification

**test design tool:** A tool that supports the test design activity by generating test inputs (and outputs) from a specification.

**test director:** A senior manager who manages test managers.
See also: test manager

**test environment:** An environment containing hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test.
Reference: ISO 24765

**test estimation:** An approximation related to various aspects of testing.

**test evaluation report:** A document produced at the end of the test process summarizing all testing activities and results. It also contains an evaluation of the test process and lessons learned.

**test execution:** The activity that runs a test on a component or system producing actual results.

**test execution automation:** The use of software, e.g., capture/playback tools, to control the execution of tests, the comparison of actual results to expected results, the setting up of test preconditions, and other test control and reporting functions.

**test execution layer:** The layer in a generic test automation architecture which supports the execution of test suites and/or test cases.

**test execution phase:** The period of time in a software development lifecycle during which the components of a software product are executed, and the software product is evaluated to determine whether or not requirements have been satisfied.
Reference: IEEE 610

**test execution schedule:** A schedule for the execution of test suites within a test cycle.

**test execution technique:** The method used to perform the actual test execution, either manual or automated.

**test execution tool:** A test tool that executes tests against a designated test item and evaluates the outcomes against expected results and postconditions.

**test generation layer:** The layer in a generic test automation architecture which supports manual or automated design of test suites and/or test cases.

**test harness:** A test environment comprised of stubs and drivers needed to execute a test suite.

**test hook:** A customized software interface that enables automated testing of a test object.

**test implementation:** The activity that prepares the testware needed for test execution based on test analysis and design.

**test improvement plan:** A plan for achieving organizational test process improvement objectives based on a thorough understanding of the current strengths and weaknesses of the organization's test processes and test process assets.
Reference: After CMMI

**test infrastructure:** The organizational artifacts needed to perform testing, consisting of test environments, test tools, office environment and procedures.

**test input:** The data received from an external source by the test object during test execution. The external source can be hardware, software or human.

**test item:** A part of a test object used in the test process.
See also: test object

**test leader:** On large projects, the person who reports to the test manager and is responsible for project management of a particular test level or a particular set of testing activities.
See also: test manager

**test level:** A specific instantiation of a test process.
Reference: After ISO 29119

**test log:** A chronological record of relevant details about the execution of tests.
Reference: ISO 24765

**test logging:** The activity of creating a test log.

**test management:** The planning, scheduling, estimating, monitoring, reporting, control and completion of test activities.
Reference: ISO 29119

**test management tool:** A tool that supports test management.

**test manager:** The person responsible for project management of testing activities, resources, and evaluation of a test object.

**Test Maturity Model integration:** (TMMi) A five-level staged framework for test process improvement, related to the Capability Maturity Model Integration (CMMI), that describes the key elements of an effective test process.

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

**test model:** A model describing testware that is used for testing a component or a system under test.

**test monitoring:** The activity that checks the status of testing activities, identifies any variances from planned or expected, and reports status to stakeholders.

**test object:** The work product to be tested.

**test objective:** The reason or purpose of testing.

**test oracle:** A source to determine an expected result to compare with the actual result of the system under test.
Reference: After Adrion

**test performance indicator:** A high-level metric of effectiveness and/or efficiency used to guide and control progressive test development, e.g., Defect Detection Percentage (DDP).

**test phase:** A distinct set of test activities collected into a manageable phase of a project, e.g., the execution activities of a test level.
Reference: After Gerrard

**test plan:** Documentation describing the test objectives to be achieved and the means and the schedule for achieving them, organized to coordinate testing activities.
Reference: After ISO 29119

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

**Test Point Analysis:** (TPA) A formula based test estimation method based on function point analysis.
Reference: TMap

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

**test procedure:** A sequence of test cases in execution order, and any associated actions that may be required to set up the initial preconditions and any wrap up activities post execution.
Reference: ISO 29119

**test procedure specification:** Documentation specifying one or more test procedures.
Reference: After ISO 29119

**test process:** The set of interrelated activities comprising of test planning, test monitoring and control, test analysis, test design, test implementation, test execution, and test completion.

**test process group:** (TPG) A collection of (test) specialists who facilitate the definition, maintenance, and improvement of the test processes used by an organization.
Reference: After CMMI

**test process improvement:** A program of activities designed to improve the performance and maturity of the organization's test processes and the results of such a program.
Reference: After CMMI

**test process improvement manifesto:** A statement that echoes the Agile manifesto, and defines values for improving the testing process. The values are: flexibility over detailed processes, best practices over templates, deployment orientation over process orientation, peer reviews over quality assurance (departments), business driven over model-driven.
Reference: Veenendaal08

**test process improver:** A person implementing improvements in the test process based on a test improvement plan.

**test progress report:** A type of test report produced at regular intervals about the progress of test activities against a baseline, risks, and alternatives requiring a decision.

**test pyramid:** A graphical model representing the relationship of the amount of testing per level, with more at the bottom than at the top.

**test report:** Documentation summarizing test activities and results.

**test reporting:** Collecting and analyzing data from testing activities and subsequently consolidating the data in a report to inform stakeholders.

**test reproducibility:** An attribute of a test indicating whether the same results are produced each time the test is executed.

**test result:** The consequence/outcome of the execution of a test. It includes outputs to screens, changes to data, reports, and communication messages sent out.

**test run:** The execution of a test suite on a specific version of the test object.

**test schedule:** A list of activities, tasks or events of the test process, identifying their intended start and finish dates and/or times, and interdependencies.

**test script:** A sequence of instructions for the execution of a test suite.

**test selection criteria:** The criteria used to guide the generation of test cases or to select test cases in order to limit the size of a test.

**test session:** An uninterrupted period of time spent in executing tests. In exploratory testing, each test session is focused on a charter, but testers can also explore new opportunities or issues during a session. The tester creates and executes on the fly and records their progress.

**test specification:** The complete documentation of the test design, test cases and test procedures for a specific test item.
Reference: ISO 29119

**test strategy:** Documentation aligned with the test policy that describes the generic requirements for testing and details how to perform testing within an organization.
Reference: After ISO 29119
Synonym: understandability

**test suite:** A set of test scripts or test procedures to be executed in a specific test run.

**test summary report:** A type of test report produced at completion milestones that provides an evaluation of the corresponding test items against exit criteria.

**test technique:** A procedure used to define test conditions, design test cases, and specify test data.

**test tool:** Software or hardware that supports one or more test activities.

**test type:** A group of test activities based on specific test objectives aimed at specific characteristics of a component or system.
Reference: After TMap

**testability:** The degree to which test conditions can be established for a component or system, and tests can be performed to determine whether those test conditions have been met.
Reference: After ISO 25010

**testability review:** A review to evaluate the testability of the test basis.
Reference: After TMap

**testable requirement:** A requirements that is stated in terms that permit establishment of test designs (and subsequently test cases) and execution of tests to determine whether the requirement has been met.
Reference: After IEEE 610

**test-driven development:** (TDD) A software development technique in which the test cases are developed, and often automated, and then the software is developed incrementally to pass those test cases.

**tester:** A person who performs testing.

**test-first development:** The practice of designing tests based on the specification of a test item before developing the corresponding test item.

**testing:** The process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation and evaluation of a component or system and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.

**testware:** Work products produced during the test process for use in planning, designing, executing, evaluating and reporting on testing.
Reference: After ISO 29119

**think aloud usability testing:** A usability testing technique where test participants share their thoughts with the moderator and observers by thinking aloud while they solve usability test tasks. Think aloud is useful to understand the test participant.

**think time:** The amount of time required by a user to determine and execute the next action in a sequence of actions.

**thread testing:** An approach to component integration testing where the progressive integration of components follows the implementation of subsets of the requirements, as opposed to the integration of components by levels of a hierarchy.

**three-point estimation:** A test estimation method using estimated values for the "best case", "worst case", and "most likely case" of the matter being estimated, to define the degree of certainty associated with the resultant estimate.

**time behavior:** The degree to which a component or system can perform its required functions within required response times, processing times and throughput rates.
Reference: After ISO 25010

**top-down testing:** An incremental approach to integration testing where the component at the top of the component hierarchy is tested first, with lower level components being simulated by stubs. Tested components are then used to test lower level components. The process is repeated until the lowest level components have been tested.

**Total Quality Management:** (TQM) An organization-wide management approach centered on quality, based on the participation of all members of the organization and aiming at long-term success through customer satisfaction, and benefits to all members of the organization and to society. Total Quality Management consists of planning, organizing, directing, control, and assurance.
Reference: After ISO 8402

**tour:** A set of exploratory tests organized around a special focus.

**TPI Next:** A continuous business-driven framework for test process improvement that describes the key elements of an effective and efficient test process.

**traceability:** The degree to which a relationship can be established between two or more work products.
Reference: After ISO 19506

**traceability matrix:** A two-dimensional table, which correlates two entities (e.g., requirements and test cases). The table allows tracing back and forth the links of one entity to the other, thus enabling the determination of coverage achieved and the assessment of impact of proposed changes.

**transactional analysis:** ( a transaction is defined as a stimulus plus a response. Transactions take place between people and between the ego states (personality segments) within one person's mind.) The analysis of transactions between people and within people's minds

**transcendent-based quality:** A view of quality, wherein quality cannot be precisely defined, but we know it when we see it, or are aware of its absence when it is missing. Quality depends on the perception and affective feelings of an individual or group of individuals toward a product.
Reference: After Garvin
See also: manufacturing-based quality, product-based quality, user-based quality, value-based quality

**unit test framework:** A tool that provides an environment for unit or component testing in which a component can be tested in isolation or with suitable stubs and drivers. It also provides other support for the developer, such as debugging capabilities.
Reference: Graham

**unreachable code:** Code that is impossible to execute.

**usability:** The degree to which a component or system can be used by specified users to achieve specified goals in a specified context of use.
Reference: After ISO 25010

**usability evaluation:** A process through which information about the usability of a system is gathered in order to improve the system (known as formative evaluation) or to assess the merit or worth of a system (known as summative evaluation).
See also: formative evaluation, summative evaluation

**usability lab:** A test facility in which unintrusive observation of participant reactions and responses to software takes place.

**usability requirement:** A requirement on the usability of a component or system.

**usability test participant:** A representative user who solves typical tasks in a usability test.

**usability test script:** A document specifying a sequence of actions for the execution of a usability test. It is used by the moderator to keep track of briefing and pre-session interview questions, usability test tasks, and post-session interview questions.
See also: test procedure specification

**usability test session:** A test session in usability testing in which a usability test participant is executing tests, moderated by a moderator and observed by a number of observers.

**usability test task:** A usability test execution activity specified by the moderator that needs to be accomplished by a usability test participant within a given period of time.

**usability testing:** Testing to evaluate the degree to which the system can be used by specified users with effectiveness, efficiency and satisfaction in a specified context of use.
Reference: After ISO 25010

**use case:** The specification of the behavior of a system with regards to its interaction with its users and any other systems.
Reference: After UML

**use case testing:** A black-box test technique in which test cases are designed to exercise use case behaviors.

**user acceptance testing:** A type of acceptance testing performed to determine if intended users accept the system.
See also: acceptance testing

**user error protection:** The degree to which a component or system protects users against making errors.
Reference: After ISO 25010

**user experience:** A person's perceptions and responses resulting from the use or anticipated use of a software product.
Reference: ISO 9241-210

**user interface:** All components of a system that provide information and controls for the user to accomplish specific tasks with the system.

**user interface aesthetics:** The degree to which a user interface enables pleasing and satisfying interaction for the user.
Reference: ISO 25010

**user interface guideline:** A low-level, specific rule or recommendation for user interface design that leaves little room for interpretation so designers implement it similarly. It is often used to ensure consistency in the appearance and behavior of the user interface of the systems produced by an organization.

**user story:** A user or business requirement consisting of one sentence expressed in the everyday or business language which is capturing the functionality a user needs, the reason behind it, any non-functional criteria, and also including acceptance criteria.

**user story testing:** A black-box test design technique in which test cases are designed based on user stories to verify their correct implementation.

**user survey:** A usability evaluation whereby a representative sample of users are asked to report subjective evaluation into a questionnaire based on their experience in using a component or system.

**user test:** A test whereby real-life users are involved to evaluate the usability of a component or system.

**user-agent based testing:** A type of testing in which a test client is used to switch the user agent string and identify itself as a different client while executing test suites.

**user-based quality:** A view of quality, wherein quality is the capacity to satisfy needs, wants and desires of the user(s). A product or service that does not fulfill user needs is unlikely to find any users. This is a context dependent, contingent approach to quality since different business characteristics require different qualities of a product.
Reference: after Garvin
See also: manufacturing-based quality, product-based quality, transcendent-based quality, value-based quality

**validation:** Confirmation by examination and through provision of objective evidence that the requirements for a specific intended use or application have been fulfilled.
Reference: ISO 9000

**value-based quality:** A view of quality wherein quality is defined by price. A quality product or service is one that provides desired performance at an acceptable cost. Quality is determined by means of a decision process with stakeholders on trade-offs between time, effort and cost aspects.
Reference: After Garvin
See also: manufacturing-based quality, product-based quality, transcendent-based quality, user-based quality

**variable:** An element of storage in a computer that is accessible by a software program by referring to it by a name.

**verification:** Confirmation by examination and through provision of objective evidence that specified requirements have been fulfilled.
Reference: ISO 9000

**vertical traceability:** The tracing of requirements through the layers of development documentation to components.

**virtual user:** A simulation of activities performed according to a user operational profile.

**V-model:** A sequential development lifecycle model describing a one-for-one relationship between major phases of software development from business requirements specification to delivery, and corresponding test levels from acceptance testing to component testing.

**volume testing:** Testing where the system is subjected to large volumes of data.

**vulnerability scanner:** A static analyzer that is used to detect particular security vulnerabilities in the code.

**walkthrough:** A type of review in which an author leads members of the review through a work product and the members ask questions and make comments about possible issues.
Reference: After ISO 20246

**Web Content Accessibility Guidelines:** (WCAG) A part of a series of web accessibility guidelines published by the Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C), the main international standards organization for the internet. They consist of a set of guidelines for making content accessible, primarily for people with disabilities.

**Website Analysis and Measurement Inventory:** (WAMMI) A commercial website analysis service providing a questionnaire for measuring user experience and assessing delivery of business goals online.

**white-box test technique:** A test technique only based on the internal structure of a component or system.
Reference: After ISO 29119

**white-box testing:** Testing based on an analysis of the internal structure of the component or system.

**Wideband Delphi:** An expert-based test estimation technique that aims at making an accurate estimation using the collective wisdom of the team members.

**wild pointer:** A pointer that references a location that is out of scope for that pointer or that does not exist.

**work breakdown structure:** (WBS) Deliverable oriented hierarchical decomposition of the work to be carried out by the project team to accomplish the project objectives.
Reference: After PMBOK

**XiL test environment:** (XiL) A generalized term for dynamic testing in different virtual test environments.

# Appendix A

**Standards**


[BS 7925/2] BS 2925:2001, Software Component Testing Standard, BCS SIGIST Working Draft 3.4

[DO-178b] DO-178B:1992. Software Considerations in Airborne Systems and Equipment Certification, Requirements and Technical Concepts for Aviation (RTCA SC167)

[IEEE 610] IEEE 610.12:1990. Standard Glossary of Software Engineering Terminology.

[IEEE 730] IEEE 730:2002. Software Quality Assurance Plans

[IEEE 829] IEEE 829:1998. Standard for Software Test Documentation

[IEEE 1008] IEEE 1008:1993. Standard for Software Unit Testing

[IEEE 1028] IEEE 1028:1997. Standard for Software Reviews and Audits

[IEEE 1044] IEEE 1044:1993. Standard Classification for Software Anomalies

[IEEE 1219] IEEE 1219:1998. Software Maintenance

[ISO 2382] ISO/IEC 2382-1:1993. Data processing - Vocabulary - Part 1: Fundamental terms

[ISO 8402] ISO 8402: 1994. Quality Management and Quality Assurance Vocabulary

[ISO 9000] ISO 9000:2005. Quality Management Systems – Fundamentals and Vocabulary

[ISO 9126] ISO/IEC 9126-1:2001. Software Engineering – Software Product Quality – Part 1: Quality characteristics and sub-characteristics

[ISO 9241] ISO 9241:2010. Ergonomics of human-system interaction – Part 210: Human-centered design for interactive systems

[ISO 12207] ISO/IEC 12207:1995. Information Technology – Software Lifecycle Processes

[ISO 14598] ISO/IEC 14598-1:1999. Information Technology – Software Product Evaluation - Part 1: General Overview

[ISO 14764] ISO/IEC 14764:2006. Software Engineering - Software Life Cycle Processes - Maintenance

[ISO 15504] ISO/IEC 15504-9: 1998. Information Technology – Software Process Assessment – Part 9: Vocabulary

[ISO 19506] ISO/IEC 19506:2012. Information technology -- Object Management Group Architecture-Driven Modernization (ADM) -- Knowledge Discovery Meta-Model (KDM)

[ISO 20246] ISO/IEC 20246:2017. Software and systems engineering -- Work product reviews

[ISO 24765] ISO/IEC/IEEE 24765:2017. Systems and software engineering -- Vocabulary

[ISO 25010] ISO/IEC 25010:2011. Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models

[ISO 25040] ISO/IEC 25040:2011. Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- Evaluation process

[ISO 29119] ISO/IEC/IEEE 29119-1:2013. Software and systems engineering -- Software testing -- Part 1: Concepts and definitions

[ISO 31000] ISO 31000:2018. Risk management

[NIST.IR.7298] U.S. Department of Commerce, National Institute of Standards and Technology – Glossary of Key Information Security Terms, Revision 2, May 2013

**Books and papers**


[Adrion] W. Adrion, M. Branstad and J. Cherniabsky (1982), Validation, Verification and Testing of Computer Software, in: Computing Surveys, Vol. 14, No 2, June 1982

[Akao] Akao, Yoji (1994), Development History of Quality Function Deployment - The Customer Driven Approach to Quality Planning and Deployment, Minato, Tokyo 107 Japan: Asian Productivity Organization, pp. 339, ISBN 92-833-1121-3

[Bach] J. Bach (2004), Exploratory Testing, in: E. van Veenendaal, The Testing Practitioner – 2nd edition, UTN Publishing, ISBN 90-72194-65-9

[Beizer] B. Beizer (1990), Software Testing Techniques, van Nostrand Reinhold, ISBN 0-442-20672-0

[Chow] T. Chow (1978), Testing Software Design Modelled by Finite-Sate Machines, in: IEEE Transactions on Software Engineering, Vol. 4, No 3, May 1978

[CMMI] M.B. Chrissis, M. Konrad and S. Shrum (2004), CMMi, Guidelines for Process Integration and Product Improvement, Addison Wesley, ISBN 0-321-15496-7

[Deming] D. W. Edwards (1986), Out of the Crisis, MIT Center for Advanced Engineering Study, ISBN 0-911379-01-0

[Egler63] J. F. Egler. 1963. A procedure for converting logic table conditions into an efficient sequence of test instructions. Commun. ACM 6, 9 (September 1963), 510-514. DOI=10.1145/367593.367595

[Fenton] N. Fenton (1991), Software Metrics: a Rigorous Approach, Chapman & Hall, ISBN 0-53249-425-1

[Fewster and Graham] M. Fewster and D. Graham (1999), Software Test Automation, Effective use of test execution tools, Addison-Wesley, ISBN 0-201-33140-3

[Freedman and Weinberg] D. Freedman and G. Weinberg (1990), Walkthroughs, Inspections, and Technical Reviews, Dorset House Publishing, ISBN 0-932633-19-6

[Garvin] D.A. Garvin (1984), What does product quality really mean?, in: Sloan Management Review, Vol. 26, nr. 1 1984

[Gerrard] P. Gerrard and N. Thompson (2002), Risk-Based E-Business Testing, Artech House Publishers, ISBN 1-58053-314-0

[Gilb and Graham] T. Gilb and D. Graham (1993), Software Inspection, Addison-Wesley, ISBN 0-201-63181-4

[Graham] D. Graham, E. van Veenendaal, I. Evans and R. Black (2007), Foundations of Software Testing, Thomson Learning, ISBN 978-1-84480-355-2

[Grochtmann] M. Grochtmann (1994), Test Case Design Using Classification Trees, in: Conference Proceedings STAR 1994

[Hetzel] W. Hetzel (1988), The complete guide to software testing – 2nd edition, QED Information Sciences, ISBN 0-89435-242-3

[Juran] J.M. Juran (1979), Quality Control Handbook, McGraw-Hill

[Kirakowski93] J. Kirakowski, M Corbett (1993), SUMI: the Software Usability Measurement Inventory, British Journal of Educational Technology, Volume 24, Issue 3, pages 210–212, September 1993

[McCabe] T. McCabe (1976), A complexity measure, in: IEEE Transactions on Software Engineering, Vol. 2, pp. 308-320

[Musa] J. Musa (1998), Software Reliability Engineering Testing, McGraw-Hill Education, ISBN 0-07913-271-5

[PMBOK]

[TMap] M. Pol, R. Teunissen, E. van Veenendaal (2002), Software Testing, A guide to the TMap Approach, Addison Wesley, ISBN 0-201-745712

[TMMi] E. van Veenendaal and J. Cannegieter (2011), The Little TMMi, UTN Publishing, ISBN 97-89490986-03-2

[Veenendaal08] E. van Veenendaal (2008), Test Improvement Manifesto, in: Testing Experience, Issue 04/08, December 2008


**Internet**


[extremeprogramming.org] http://www.extremeprogramming.org/; retrieved on the 04th of June, 2018.

**Trademarks**

In the ISTQB Glossary the following trademarks are used:
- CMMi and IDEAL are registered trademarks of Carnegie Mellon University
- EFQM is a registered trademark of the EFQM Foundation
- Rational Unified Process (RUP) is a registered trademark of Rational Software Corporation
- STEP is a registered trademark of Software Quality Engineering
- TMap, TPA and TPI Next are registered trademarks of Sogeti Nederland BV
- TMMi is a registered trademark of the TMMi Foundation

# Appendix B

Suggestions and comments for improvements to this glossary are warmly welcomed.

When submitting suggestions and comments do not forget to enter the following information:
– your name and contact options,
– version number of the glossary the comment applies
– detailed information about which part of the glossary the comment applies to and
– information on proposed change and a brief explanation for the amendment to be introduced.

Suggestions and comments are sent to *info@sstb.se*.