# 软件测试专业术语对照表

顾问　　居德华

主编　　刘琴 杜庆峰

评审专家　沈备军 周震漪　崔启亮

参与志愿者 马均飞 刘小茵 李军 李华北 何根海 郑文强　单晓炯 赵国峰 黄晶

（按姓氏笔画排列）

# 目 录

# 前言

此术语表为国际软件测试认证委员会（ISTQB）发布的标准术语表。此表历经数次修改、完善，集纳了计算机行业界、商业界及政府相关机构的见解及意见，在国际化的层面上达到了罕有的统一性及一致性。参与编制此表的国际团体包括澳大利亚、比利时、芬兰、德国、印度、以色列、荷兰、挪威、葡萄牙、瑞典、英国和美国。

多数软件测试工程师使用1998年发布的BS 7925-1标准。英国信息系统考试委员会(ISEB)也以此标准作为基础级别和从业级别认证的首要参考标准。BS 7925-1标准最初是围绕着单元测试撰写的，自发布之后许多旨在改进和扩展此标准，以覆盖更广义范围的软件测试领域的新概念和提议不断涌现。最新版的BS 7925-1标准中的软件测试词汇吸纳、融合了上述概念和提议。此国际软件测试认证委员会（ISTQB）发布的标准术语表即是以最新版的BS 7925-1标准为基础制定的国际化软件测试标准术语。

# 1 简介

行业界、商业界、政府及学术机构曾经花费大量精力和时间以解释和区分一些常见的软件测试专业术语以期在各社会部门或机构之间达成交流，例如：语句覆盖(statement coverage) 和条件覆盖(decision overage)； 测试套件(test suite)、 测试规格说明书(test specification)和测试计划(test plan)等。上述机构与专职机构定义的同名术语在含义上又往往有很大偏差。

# 2 范畴

本文档旨在提供概念、条款、和定义为软件测试及相关从业人员进行有效交流的平台。

# 3 结构

术语表中的词汇按字母顺序排列。术语如有同义词汇，本术语表解释最通用的词汇，其同义词汇会的仅被列出，不予重复解释。例如*结构测试(structural testing)* 和*白盒测试(white box testing)*。此类同义词在术语表中用“参见”列出，以便读者检索。“参见”往往连接着广义和狭义词或含义重叠的词汇。

# 4 标准参考

至截稿日期，此标准有效版本为1.2。如所有其他标准一样，本术语表仍需根据以下相关标准的最新版本不断修正。此标准由IEC 和 ISO 成员根据目前有效的国际相关标准进行更新。

- BS 7925-2:1998. Software Component Testing.
- DO-178B:1992. Software Considerations in Airborne Systems and Equipment Certification, Requirements and Technical Concepts for Aviation (RTCA SC167).
- IEEE 610.12:1990. Standard Glossary of Software Engineering Terminology.
- IEEE 829:1998. Standard for Software Test Documentation.
- IEEE 1008:1993. Standard for Software Unit Testing.
- IEEE 1012:1986. Standard for Verification and Validation Plans
- IEEE 1028:1997. Standard for Software Reviews and Audits.
- IEEE 1044:1993. Standard Classification for Software Anomalies.
- IEEE 1219:1998. Software Maintenance.
- ISO/IEC 2382-1:1993. Data processing - Vocabulary - Part 1: Fundamental terms.
- ISO 9000:2000. Quality Management Systems – Fundamentals and Vocabulary.
- ISO/IEC 9126-1:2001. Software Engineering – Software Product Quality – Part 1: Quality characteristis and sub-characteristics.
- ISO/IEC 12207:1995. Information Technology – Software Life Cycle Processes.
- ISO/IEC 14598-1:1996. Information Technology – Software Product Evaluation - Part 1: General Overview.

| A | | |
|---|---|---|
| **abstract test case** | **抽象测试用例** | 参见 high level test case. |
| **acceptance** | **验收** | 参见 acceptance testing. |
| **acceptance criteria** | **验收准则** | 为了满足组件或系统使用者、客户或其他授权实体的需要，组件或系统必须达到的准则。[IEEE 610] |
| **acceptance testing** | **验收测试** | 一般由用户/客户进行的确认是否可以接受一个系统的验证性测试。是根据用户需求，业务流程进行的正式测试以确保系统符合所有验收准则。[与 IEEE 610 一致] |
| **accessibility testing** | **可达性测试** | 可达性测试就是测试残疾人或不方便的人们使用软件或者组件的容易程度[Gerrard]。即被测试的软件是否能够被残疾或者部分有障碍人士正常使用，这其中也包含了正常人在某些时候发生暂时性障碍的情况下正常使用，如怀抱婴儿等。 |
| **accuracy** | **准确性** | 软件产品的提供的结果的正确性、一致性和精确程度的能力。[ISO9126] 参见 functionality testing |
| **actual outcome** | **实际结果** | 参见 actual result |
| **actual result** | **实际结果** | 组件或系统测试之后产生或观察到的行为 |
| **ad hoc review** | **临时评审** | 非正式评审（和正式的评审相比） |
| **ad hoc testing** | **随机测试** | 非正式的测试执行。即没有正式的测试准备、规格设计和技术应用，也没有期望结果和必须遵循的测试执行指南。 |
| **adaptability** | **适应性** | 软件产品毋需进行额外修改，而适应不同特定环境的能力。[ISO9126] 参见 protability |
| **agile testing** | **敏捷测试** | 对使用敏捷方法，如极限编程(Extreme programming)开发的项目进行的软件测试，强调测试优先行的设计模式，见 test driven development |
| **algorithm test [TMap]** | **算法测试** | 参见 branch testing |
| **alpha testing** | **Alpha 测试** | 由潜在用户或者独立的测试团队在开发环境下或者模拟实际操作环境下进行的测试,通常在开发组织之外进行。通常是对现货软件(COTS)进行内部验收测试的一种方式。 |
| **analyzability** | **可分析性** | 软件产品缺陷或运行失败原因可被诊断的能力，或对修改部分的可识别能力。[ISO 9126] 参见 maintainability. |
| **analyzer** | **分析器** | 参见 static analyzer |

| | | |
|---|---|---|
| **anomaly** | **异常** | 任何和基于需求文档、设计文档、用户文档、标准或者个人的期望和预期之间偏差的情况，都可以称为异常。异常可以在但不限于下面的过程中识别：评审(review)、测试分析(test analysis)、编译(compilation)、软件产品或应用文档的使用等。参见 defect, deviation, error, fault, failure, incident, problem |
| **arc testing** | **弧测试** | 参见 branch testing |
| **attractiveness** | **吸引力** | 软件产品吸引用户的能力.[ISO9126]参见 usability |
| **audit** | **审计** | 对软件产品或过程进行的独立评审，来确认产品是否满足标准、指南、规格说明书以及基于客观准则的步骤等，包括下面的文档：(1)产品的内容与形式(2)产品开发应该遵循的流程(3)度量符合标准或指南的准则。[IEEE1028] |
| **audit trail** | **审计跟踪** | 以过程输出作为起点，追溯到原始输入（例如：数据）的路径。有利于缺陷分析和过程审计的开展。[与 TMap 一致] |
| **automated testware** | **自动测试件** | 用于自动化测试中的测试件，如，工具脚本 |
| **availability** | **可用性** | 用户使用系统或组件的可操作和易用的程度，通常以百分比的形式出现。[IEEE 610] |
| | | |
| **B** | | |
| **back-to-back testing** | **比对测试** | 用相同的输入，执行组件或系统的两个或多个变量，在产生偏差的时候，对输出结果进行比较和分析。 |
| **baseline** | **基线** | 通过正式评审或批准的规格或软件产品。以它作为继续开发的基准。并且在变更的时候，必须通过正式的变更流程来进行。[与 IEEE 610 一致] |
| **basic block** | **基本块** | 一个或多个连续可执行的语句块，不包含任何分支语句。 |
| **basis test set** | **基本测试集** | 根据组件的内部结构或规格说明书设计的一组测试用例集。通过执行这组测试用例可以保证达到 100%的指定覆盖准则(coverage criterion)的要求。 |
| **bebugging** | **错误散播** | 参见 error seeding |
| **behavior** | **行为** | 组件或系统对输入值和预置条件的反应。 |
| **benchmark test** | **基准测试** | (1)为使系统或组件能够进行度量和比较而制定的一种测试标准；(2)用于组件或系统之间进行的比较或和（1）中提到的标准进行比较的测试。[与 IEEE 610 一致] |
| **bespoke software** | **定制软件** | 为特定的用户定制开发的软件。与之对比的是现货软件(off-the-shelf software)。 |
| **best practice** | **最佳实践** | 在界定范围内，帮助提高组织能力的有效方法或创新实践，通常被同行业组织视最佳的方法或实 |

| | | |
|---|---|---|
| | | 践。 |
| **beta testing** | Beta 测试 | 用户在开发组织外，没有开发人员参与的情况下进行的测试，检验软件是否满足客户及业务需求。这种测试是软件产品获得市场反馈进行验收测试的一种形式。 |
| **big-bang testing** | 大爆炸测试 | 非增量集成测试的一种方法，测试的时候将软件单元、硬件单元或者两者同时,而不是阶段性的，集成到组件或者整个系统中去进行测试。[与 IEEE 610 一致]参见 integration testing。 |
| **black-box technique** | 黑盒技术 | 参见 black box test design technique |
| **black-box testing** | 黑盒测试 | 不考虑组件或系统内部结构的功能或非功能测试。 |
| **black-box test design technique** | 黑盒测试设计技术 | 基于系统功能或非功能规格说明书来设计或者选择测试用例的技术，不涉及软件内部结构。 |
| **bottom-up testing** | 自底向上测试 | 渐增式集成测试的一种，其策略是先测试底层的组件，以此为基础逐步进行更高层次的组件测试，直到系统集成所有的组件。参见 integration testing。 |
| **boundary value** | 边界值 | 通过分析输入或输出变量的边界或等价划分(equivalence partition)的边界来设计测试用例，例如，取变量的最大、最小值、中间值、比最大值大的值、比最小值小的值等。 |
| **boundary value analysis** | 边界值分析 | 一种黑盒设计技术(black box test design technique)，基于边界值进行测试用例的设计。 |
| **boundary value coverage** | 边界值覆盖 | 执行一个测试套件(test suite)所能覆盖的边界值(boundary value)的百分比。 |
| **boundary value testing** | 边界值测试 | 参见 boundary value analysis。 |
| **branch** | 分支 | 在组件中，控制从任何语句到其它任何非直接后续语句的一个条件转换，或者是一个无条件转换。例如: case, jump, go to, if-then-else 语句. |
| **branch condition** | 分支条件 | 参见条件(condition) |
| **branch condition combination coverage** | 分支条件组合覆盖 | 参见 multiple condition coverage. |
| **branch condition combination testing** | 分支条件组合测试 | 参见 multiple condition testing. |
| **branch condition coverage** | 分支条件覆盖 | 参见 condition coverage. |
| **branch coverage** | 分支覆盖 | 执行一个测试套件(test suite)所能覆盖的分支(branch)的百分比。100%的分支覆盖(branch coverage)是指 100%判定条件覆盖(decision covergate) 和 100%的语句覆盖(statement covergage)。 |
| **bug** | 缺陷 | 参见 defect。 |
| **bug report** | 缺陷报告 | 参见 defect report。 |
| **business process-based testing** | 基于业务过程测试 | 一种基于业务描述和/或业务流程的测试用例设计方法。 |
| | | |

| C | | |
|---|---|---|
| **Capability Maturity Model (CMM)** | **能力成熟度模型** | 描述有效的软件开发过程关键元素的一个五个等级的框架，能力成熟度模型包含了在软件开发和维护中计划、工程和管理方面的最佳实践(best practice)，缩写为 CMM。[CMM] |
| **Capability Maturity Model Integration (CMMI)** | **能力成熟度模型集成** | 描述有效的软件产品开发和维护过程的关键元素框架，能力成熟度模型集成包含了软件开发计划、工程和管理等方面的最佳实践，是 CMM 的指定的继承版本。 |
| **capture/playback tool** | **捕获/回放工具** | 一种执行测试工具，能够捕获在手工测试过程中的输入，并且生成可执行的自动化脚本用于后续阶段的测试（回放过程）。这类工具通常使用在自动化回归测试(regression test)中。 |
| **capture/replay tool** | **捕获/回放工具** | 参见 capture/playback tool |
| **CASE** | **计算机辅助软件工程** | Computer Aided Software Engineering 的首字母缩写。 |
| **CAST** | **计算机辅助软件测试** | Computer Aided Software Testing 的首字母缩写，参见 test automation。在测试过程中使用计算机软件工具进行辅助的测试。 |
| **cause-effect graph** | **因果图** | 用来表示输入（原因）与结果之间关系的图表，因果图可以用来设计测试用例。 |
| **cause-effect graphing** | **因果图技术** | 通过因果图(case-effect graph)设计测试用例的一种黑盒测试设计技术。 |
| **cause-effect analysis** | **因果分析** | 参见因果图技术(case-effect graphing)。 |
| **cause-effect decision table** | **因果决策表** | 参见决策表 (decision table)。 |
| **certification** | **认证** | 确认一个组件、系统或个人具备某些特定要求的过程，比如通过了某个考试。 |
| **changeability** | **可变性** | 软件产品适应修改的能力，[ISO 9126] 参见 maintainability |
| **change control** | **变更控制** | 参见 configuration control |
| **change control board** | **变更控制委员会 CCB** | 参见 configuration control board |
| **checker** | **检验员** | 参见评审员(Reviewer) |
| **chow's coverage metrics** | **N 切换覆盖度量** | 参见 N 切换覆盖(N-switch coverage)[Chow] |
| **classification tree method** | **分类树方法** | 运用分类树法而进行的一种黑盒测试设计技术，通过输入和/或输出域的组合来设计测试用例 [Grochtmann] |
| **code** | **代码** | 计算机指令和数据定义在程序语言中的表达形式或是汇编程序、编译器或其他翻译器的一种输出形式。 |
| **code analyzer** | **代码分析器** | 参见静态分析器(static code analyzer) |

| | | |
|---|---|---|
| **code coverage** | **代码覆盖** | 一种分析方法，用于确定软件的哪些部分被测试套件(test suite)覆盖到了，哪些部分没有。例如：语句覆盖(statement covergage)，判定覆盖(decision coverage)和条件覆盖(condition covergate)。 |
| **code-based testing** | **基于代码的测试** | 参见 white box testing |
| **co-existence** | **共存性** | 软件产品与通用环境下与之共享资源的其它独立软件之间共存的能力。[ISO 9126] 参见可移植性(portability)。 |
| **commercial off-the-shelf software** | **商业现货软件** | 参见现货软件(off-the shelf software) |
| **comparator** | **比较器** | 参见 test comparator。 |
| **compiler** | **编译器** | 将高级命令语言编写的程序翻译成能运行的机器语言的工具[IEEE 610]. |
| **complete testing** | **完全测试** | 参见穷尽测试(exhaustive testing) |
| **completion criteria** | **完成准则** | 参见退出准则(exit criteria) |
| **complexity** | **复杂性** | 系统或组件的设计和/或内部结构难于理解、维护或验证的程度。参见 cyclomatic complexity. |
| **compliance** | **一致性** | 软件产品与法律和类似规定的标准、惯例或规则的一致性方面的能力。[ ISO9126] |
| **compliance testing** | **一致性测试** | 确定组件或系统是否满足标准的测试过程。 |
| **component** | **组件** | 一个可被独立测试的最小软件单元。 |
| **component integration testing** | **组件集成测试** | 为发现集成组件接口之间和集成组件交互产生的缺陷而执行的测试。 |
| **component specification** | **组件规格说明** | 根据组件的功能定义为特定输入而应该产生的输出规格进行的功能性和非功能性行为的描述。例如：资源使用(resource utilization). |
| **compound condition** | **复合条件** | 通过逻辑操作符(AND, OR 或者 XOR)将两个或多个简单条件连结起来：如，“A>0 AND B<1000” |
| **concrete test case** | **具体测试用例** | 参见低阶测试用例(low level test case). |
| **concurrency testing** | **并发测试** | 测试组件或系统的两个或多个活动在同样的间隔时间内如何交叉或同步并发。[与 IEEE 610 一致] |
| **condition** | **条件** | 一个可被判定为真、假(true,false)的逻辑表达式。例如： A>B. |
| **condition combination coverage** | **条件组合覆盖** | 参见多条件覆盖(multiple condition coverage). |
| **condition combination testing** | **条件组合测试** | 参见多条件测试(multiple condition testing). |
| **condition coverage** | **条件覆盖** | 执行测试套件(test suite)能够覆盖到的条件百分比。100%的条件覆盖要求测试到每一个条件语句真、假(true,false)的条件。 |
| **condition determination coverage** | **条件决定覆盖** | 执行测试套件(test suite)覆盖到的能够独立影响判定结果的单个条件的百分比。100%的条件决定覆盖意味着 100%的判定条件覆盖。 |
| **condition determination testing** | **条件决定测试** | 一种白盒测试技术，是对能够独立影响决策结果的单独条件的测试。 |
| **condition testing** | **条件测试** | 一种白盒测试技术，设计测试用例以执行条件的结果。 |

| | | |
|---|---|---|
| **condition outcome** | **条件结果** | 条件判定的结果，为真或假。 |
| **confidence test** | **置信测试** | 参见冒烟测试(smoke testing) |
| **configuration** | **配置** | 根据定义的数值、特性及其相关性综合设置一个组件或者系统。 |
| **configuration auditing** | **配置审核** | 对配置库及配置项的内容进行检查的过程，比如检查标准的一致性。［IEEE 610］ |
| **configuration control** | **配置控制** | 配置管理的一个方面，包括在正式配置完成之后对配置项进行评价、协调、批准或撤消、以及变更修改的控制。［IEEE 610］ |
| **configuration control board (CCB)** | **配置控制委员会** | 负责评估、批准或拒绝配置项修改的组织，此组织应确保被批准的配置修改的执行。 [IEEE 610] |
| **configuration identification** | **配置标识** | 配置管理的要素之一，包括选择配置项，并在技术文档中记录其功能和物理特性。[IEEE 610] |
| **configuration item** | **配置项** | 配置管理中的硬件、软件或软、硬件结合体的集合，在配置管理过程中通常被当做一个实体。[IEEE 610] |
| **configuration management** | **配置管理** | 一套技术和管理方面的监督原则，用于确定和记录一个配置项的功能和物理属性、控制对这些属性的变更、记录和报告变更处理和实现的状态、以及验证与指定需求的一致性。[IEEE 610] |
| **configuration management tool** | **配置管理工具** | 支持对配置项进行识别、控制、变更管理、版本控制和发布配置项基线(baseline)的工具.[IEEE 610] |
| **configuration testing** | **配置测试** | 参见可移植性测试(portability testing) |
| **confirmation testing** | **确认测试** | 参见再测试(re-testing) |
| **conformance testing** | **一致性测试** | 参见符合性测试(compliance testing)。 |
| **consistency** | **一致性** | 在系统或组件的各组成部分之间和文档之间无矛盾，一致，符合标准的程度。[IEEE 610] |
| **control flow** | **控制流** | 执行组件或系统中的一系列顺序发生的事件或路径。 |
| **control flow graph** | **控制流图** | 通过图形来表示组件或系统中的一系列顺序发生的事件或路径。 |
| **control flow path** | **控制流路径** | 参见路径(path) |
| **conversion testing** | **转换(移植）测试** | 用于测试已有系统的数据是否能够转换到替代系统上的一种测试。 |
| **COTS** | **现货软件** | Commercial Off-The-Shelf software 的首字母缩写。参见 Off-The-Shelf software |
| **coverage** | **覆盖** | 用于确定执行测试套件所能覆盖项目的程度，通常用百分比来表示。 |
| **coverage analysis** | **覆盖分析** | 对测试执行结果进行特定的覆盖项分析，判断其是否满足预先定义的标准，是否需要设计额外的测试用例。 |
| **coverage item** | **覆盖项** | 作为测试覆盖的基础的一个实体或属性：如等价划分(equivalent partitions)或代码语句(code statement)等。 |
| **coverage tool** | **覆盖工具** | 对执行测试套件(test suite)能够覆盖的结构元素如语句(statement)、分支(branch)等进行客观测量的工具。 |
| **custom software** | **定制软件** | 参见 bespoke software。 |

| | | |
|---|---|---|
| **cyclomatic complexity** | **圈复杂度** | 程序中独立路径的数量。一种代码复杂度的衡量标准,用来衡量一个模块判定结构的复杂程度,数量上表现为独立现行路径条数,即合理的预防错误所需测试的最少路径条数,圈复杂度大说明程序代码可能质量低且难于测试和维护,根据经验,程序的可能错误和高的圈复杂度有着很大关系。圈复杂度=L-N + 2P,其中 L 表示为结构图（程序图）的边数；N 为结构图（程序图）的节点数目；P 为无链接部分的数目。[与 McCabe 一致] |
| **cyclomatic number** | **圈数** | 参见 cyclomatic complexity。 |
| | | |
| **D** | | |
| **daily build** | **每日构建** | 每天对整个系统进行编译和链接的开发活动，从而保证在任何时候包含所有变更的完整系统是可用的。 |
| **data definition** | **数据定义** | 给变量赋了值的可执行语句。 |
| **data driven testing** | **数据驱动测试** | 将测试输入和期望输出保存在表格中的一种脚本技术。通过这种技术，运行单个控制脚本就可以执行表格中所有的测试。像录制/回放这样的测试执行工具经常会应用数据驱动测试方法。[Fewster and Graham],参见 keyword driven testing. |
| **data flow** | **数据流** | 数据对象的顺序的和可能的状态变换的抽象表示，对象的状态可以是：创建、使用和销毁。[Beizer] |
| **data flow analysis** | **数据流分析** | 一种基于变量定义和使用的静态分析(static analysis)模式。 |
| **data flow coverage** | **数据流覆盖** | 执行测试套件(test suite)能够覆盖已经定义数据流的百分比。 |
| **data flow testing** | **数据流测试** | 一种白盒测试设计技术：设计的测试用例用来测试变量的定义和使用路径。 |
| **data integrity testing** | **数据完整性测试** | 参见 database integrity testing。 |
| **database integrity testing** | **数据库完整性测试** | 对数据库的存取和管理进行测试的方法和过程，确保数据库如预期一样进行存取、处理等数据功能，同时也确保数据在存取过程中没有出现不可预料的删除、更新和创建。 |
| **dead code** | **死代码** | 参见 unreachable code。 |
| **debugger** | **调试器** | 参见 debugging tool。 |
| **debugging** | **调试** | 发现、分析和去除软件失败根源的过程。 |
| **debugging tool** | **调试工具** | 程序员用来复现软件失败、研究程序状态并查找相应缺陷的工具。调试器可以让程序员单步执行程序、在任何程序语句中终止程序和设置、检查程序变量。 |
| **decision** | **判定** | 有两个或多个可替换路径控制流的一个程序控制点。也是连接两个或多个分支的节点。 |

| | | |
|---|---|---|
| **decision condition coverage** | **判定条件覆盖** | 执行测试用例套件(test suite)能够覆盖的条件结果(condition outcomes)和判定结果(decision outcomes)的百分比，100%的判定条件覆盖意味着100%的判定覆盖和 100%的条件覆盖。 |
| **decision condition testing** | **判定条件测试** | 一种白盒测试(white box)设计技术，设计的测试用例用来测试条件结果(condition outcoems)和判定结果(decision outcomes)。 |
| **decision coverage** | **判定覆盖** | 执行测试套件能够覆盖的判定结果(decsion outcomes)的百分比。100％的判定覆盖(decision converage)意味着 100 的分支覆盖(branch coverage)和 100％的语句覆盖(statement coverage)。 |
| **decision table** | **决策表** | 一个可用来设计测试用例的表格，一般有条件桩、行动桩和条件规则条目和行动规则条目组成。 |
| **decision table testing** | **决策表测试** | 一种黑盒测试设计技术，设计的测试用例用来测试判定表中各种条件的组合。[Veenendaal] |
| **decision testing** | **决策测试** | 白盒测试设计技术的一种，设计测试用例来执行判定结果。 |
| **decision outcome** | **判定结果** | 判定的结果（可以来决定执行哪条分支）。 |
| **defect** | **缺陷** | 可能会导致软件组件或系统无法执行其定义的功能的瑕疵，例如：错误的语句或变量定义。如果在组件或系统运行中遇到缺陷，可能会导致运行的失败。 |
| **defect density** | **缺陷密度** | 将软件组件或系统的缺陷数和软件或者组件规模相比的一种度量（标准的度量术语包括，如每千行代码、每个类或功能点存在的缺陷数）。 |
| **Defect Detection Percentage (DDP)** | **缺陷发现百分比** | 在一个测试阶段发现的缺陷数除以在测试阶段和之后其他阶段发现的缺陷总数所得的百分比数。 |
| **defect management** | **缺陷管理** | 发现、研究、处置、去除缺陷的过程。包括记录缺陷、分类缺陷和识别缺陷可能造成的影响。[与 IEEE 1044 一致] |
| **defect management tool** | **缺陷管理工具** | 一个方便记录和跟踪缺陷的工具，通常包括以缺陷修复操作流程为引导的任务分配、缺陷修复、重新测试等行为的跟踪和控制，并且提供文档形式的报告。参见 incident management tool. |
| **defect masking** | **缺陷屏蔽** | 一个缺陷阻碍另一个缺陷被发现的情况[与 IEEE 610 一致] |
| **defect report** | **缺陷报告** | 对造成软件组件或系统不能实现预期功能的缺陷进行描述的报告文件。 |
| **defect tracking tool** | **缺陷跟踪工具** | 参见 defect management tool |
| **definition-use pair** | **定义-使用对** | 变量在程序中定义和使用的相关性，变量使用包括变量计算（比如：乘）或者变量引导程序执行一条路径（预定义）。 |
| **deliverable** | **交付物** | 过程中生成的交付给客户的（工作）产品。 |
| **design-based testing** | **基于设计的测试** | 根据组件或系统的构架或详细设计设计测试用例的一种测试方法（例如：组件或系统之间接口的测试）。 |
| **desk checking** | **桌面检查** | 通过手工模拟执行来对软件或规格说明而进行的测试。参见 static analysis. |
| **development testing** | **开发测试** | 通常在开发环境下，开发人员在组件或系统实现过程中进行的正式或非正式的测试。[与 IEEE 610 一致] |

| | | |
|---|---|---|
| **deviation** | **偏离** | 参见 incident。 |
| **deviation report** | **偏离报告** | 参见 incident report。 |
| **dirty testing** | **负面测试** | 参见 negative testing。 |
| **documentation testing** | **文档测试** | 关于文档质量的测试，例如:对用户手册或安装手册的测试。 |
| **domain** | **域** | 一个可供有效输入和/或输出值选择的集合。 |
| **driver** | **驱动器** | 代替某个软件组件来模拟控制和/或调用其他组件或系统的软件或测试工具。[与 TMap 一致] |
| **dynamic analysis** | **动态分析** | 组件或系统的执行过程中对其行为评估的过程，例如对内存性能、CPU 使用率等的估算。[与 IEEE 610 一致] |
| **dynamic analysis tool** | **动态分析工具** | 为程序代码提供实时信息的工具。通常用于识别未定义的指针，检测指针算法和内存地址分配、使用及释放的情况以及对内存泄露进行标记。 |
| **dynamic comparison** | **动态比较** | 在软件运行过程中（例如用测试工具执行），对实际结果和期望结果的比较。 |
| **dynamic testing** | **动态测试** | 通过运行软件的组件或系统来测试软件。 |
| | | |
| **E** | | |
| **efficiency** | **效率** | 一定条件下根据资源的使用情况，软件产品能够提供适当性能的能力。[ISO 9126] |
| **efficiency testing** | **效率测试** | 确定测试软件产品效率的测试过程。 |
| **elementary comparison testing** | **基本比较测试** | 一种黑盒测试设计技术：根据判定条件覆盖的理念，设计测试用例来测试软件各种输入的组合。[TMap] |
| **emulator** | **仿真器** | 一个接受同样输入并产生同样输出的设备、计算机程序或系统。[IEEE 610]参见 simulator |
| **entry criteria** | **入口准则** | 进入下个任务（如测试阶段）必须满足的条件。准入条件的目的是防止执行不能满足准入条件的活动而浪费资源[Gilb and Graham] 。 |
| **entry point** | **入口点** | 一个组件的第一个可执行语句。 |
| **equivalence class** | **等价类** | 参见 equivalence partition。 |
| **equivalence partition** | **等价类划分** | 根据规格说明，输入域或输出域的一个子域内的任何值都能使组件或系统产生相同的响应结果。 |
| **equivalence partition coverage** | **等价划分覆盖** | 执行测试套件能够覆盖到的等价类的百分比。 |
| **equivalence partitioning** | **等价类划分技术** | 黑盒测试用例设计技术：该技术从组件的等价类中选取典型的点进行测试。原则上每个等价类中至少要选取一个典型的点来设计测试用例。 |
| **error** | **错误** | 人为的产生不正确结果的行为。[与 IEEE 610 一致] |

| | | |
|---|---|---|
| **error guessing** | **错误推测** | 根据测试人员以往的经验，猜测在组件或系统中可能出现的缺陷以及错误，并以此为依据来进行特殊的用例设计以暴露这些缺陷。 |
| **error seeding** | **错误散播** | 在组件或系统中有意插入一些已知缺陷（defect）的过程，目的是为了得到缺陷的探测率和除去率，以及评估系统中遗留缺陷的数量。[IEEE 610] |
| **error tolerance** | **容错** | 组件或系统存在缺陷的情况下保持连续正常工作状态的能力。[与 IEEE 610 一致] |
| **evaluation** | **评估** | 参见 testing。 |
| **exception handling** | **异常处理** | 组件或系统对错误输入的行为反应。错误输入包括人为的输入、其他组件或系统的输入以及内部失败引起的输入等。 |
| **executable statement** | **可执行语句** | 语句编译后可以转换为目标代码，同时在程序运行的时候可以按步骤执行并且可以对数据进行相应的操作。 |
| **exercised** | **被执行** | 测试用例运行后被执行的语句、判定和程序的结构元素 |
| **exhaustive testing** | **穷尽测试** | 测试套件包含了软件输入值和前提条件所有可能组合的测试方法。 |
| **exit criteria** | **出口准则** | 和利益相关者达成一致的系列通用和专门的条件，来正式的定义一个过程的结束点。出口准则的目的可以防止将没有完成的任务错误地看成任务已经完成。测试中使用的出口准则可以来报告和计划什么时候可以停止测试。[与 Gilb 和 Graham 一致] |
| **exit point** | **出口点** | 组件中最后一个可执行语句。 |
| **expected outcome** | **预期结果** | 参见 expected result。 |
| **expected result** | **预期结果** | 在特定条件下根据规格说明或其他资源说明，组件或系统预测的行为。 |
| **experienced-based test design technique** | **基于经验的测试设计技术** | 根据测试人员的经验、知识和直觉来进行用例设计和/或选择的一种技术。 |
| **exploratory testing** | **探索性测试** | 非正式的测试设计技术：测试人员能动的设计一些测试用例，通过执行这些测试用例和在测试中得到的信息来设计新的更好的测试用例。[和 Bach 一致] |
| | | |
| **F** | | |
| **fail** | **失败** | 假如测试的实际结果与预期结果不一样，我们就认为这个测试的状态为失败。 |
| **failure** | **失效** | 组件/系统与预期的交付、服务或结果存在的偏差。[与 Fenton 一致] |
| **failure mode** | **失效模式** | 失效在物理上或功能上的表现。例如，系统在失效模式下，可能表现为运行缓慢、输出错误或者执行的彻底中断。[IEEE 610] |

| | | |
|---|---|---|
| **Failure Mode and Effect Analysis (FMEA)** | **失效模式和影响分析 （FMEA）** | 一个系统的进行风险识别和标识可能的失效模式的系统方法，用来预防失效的发生。 |
| **failure rate** | **失效率** | 指定类型中单位度量内发生失效的数目。例如，单位时间失效数、单位处理失效数、单位计算机运行失效数。[IEEE 610] |
| **fault** | **故障** | 参见 defect。 |
| **fault density** | **故障密度** | 参见 defect density。 |
| **Fault Detection Percentage (FDP)** | **故障发现率（FDP）** | 参见 Defect Detection Percentage (DDP)。 |
| **fault masking** | **故障屏蔽** | 参见 defect masking。 |
| **fault tolerance** | **故障容限** | 软件产品存在故障或其指定接口遭到破坏时，继续维持特定性能级别的能力。[ISO 9126] 参见 reliability。 |
| **fault tree analysis** | **故障树分析** | 分析产生故障原因的一种方法。 |
| **feasible path** | **可达路径** | 可通过一组输入值和入口条件而执行到的一条路径。 |
| **feature** | **特性** | 需求文档指定的或包含的一个组件或者系统的属性（例如：reliability, usability 或者 design constraints）。 [与 IEEE 1008 一致] |
| **field testing** | **现场测试** | 参见 beta testing |
| **finite state machine** | **有限状态机** | 包含有限数目状态和状态之间转换的一种计算模型，同时可能伴随一些可能的（触发）行为。[IEEE 610] |
| **finite state testing** | **有限状态测试** | 参见 state transition testing。 |
| **formal review** | **正式评审** | 对评审过程及需求文档化的一种特定的评审。例如，检视(inspection)。 |
| **frozen test basis** | **冻结测试基准** | 测试基准文档，只能通过正式的变更控制过程进行修正。参见 baseline |
| **Functional Point Analysis (FPA)** | **功能点分析** | 对信息系统功能进行规模度量的一种方法。该度量独立于具体的技术实现，可以作为生产率度量、资源需求估算和项目控制的基础。 |
| **functional integration** | **功能集成** | 合并组件/系统以尽早实现基本功能的一种集成方法。参见 integration testing。 |
| **functional requirement** | **功能需求** | 指定组件/系统必须实现某项功能的需求。[IEEE 610] |
| **functional test design technique** | **功能测试设计技术** | 通过对组件或系统的功能规格说明分析来进行测试用例的设计和/或选择的过程，该过程不涉及软件的内部结构。参见 black box test design techinque。 |
| **functional testing** | **功能测试** | 通过对组件/系统功能规格说明的分析而进行的测试。参见 black box testing。 |
| **functionality** | **功能性** | 软件产品在规定条件下使用时，所提供的功能达到宣称的和隐含需求的能力。[ISO 9126] |
| **functionality testing** | **功能性测试** | 判断软件产品功能性的测试过程。 |
| | | |

| | | |
|---|---|---|
| **G** | | |
| **glass box testing** | **玻璃盒测试** | 参见 white box testing |
| | | |
| **H** | | |
| **heuristic evaluation** | **启发式评估** | 一种静态可用性测试技术，判断用户接口和公认的可用性原则的符合度。 |
| **high level test case** | **概要测试用例** | 没有具体的（实现级别）输入数据和预期结果的测试用例。实际值没有定义或是可变的，而用逻辑概念来代替。参见 low level test case。 |
| **horizontal traceability** | **水平可追踪性** | 一个测试级别的需求和相应级别的测试文档（例如测试计划、测试设计规格、测试用例规格和测试过程规格或测试脚本）之间的可追踪性。 |
| | | |
| **I** | | |
| **impact analysis** | **影响分析** | 对需求变更所造成的开发文档、测试文档和组件的修改的评估。 |
| **incident** | **事件** | 任何有必要调查的事情。[与 IEEEE 1008 一致] |
| **incident logging** | **事件日志** | 记录所发生的（例如，在测试过程中）事件的详细情况。 |
| **incident management** | **事件管理** | 识别、调查、采取行动和处理事件的过程。该过程包含对事件进行记录、分类并辨识其带来的影响。 [IEEE 1044] |
| **incident management tool** | **事件管理工具** | 辅助记录事件并对事件进行状态跟踪的工具。这种工具常常具有面向工作流的特性，以跟踪和控制事件的资源分配、更正和再测试，并提供报表。参见 defect management tool |
| **incident report** | **事件报告** | 报告任何需要调查的事件(如在测试过程中需要调查的事件)的文档。[IEEE 829] |

| | | |
|---|---|---|
| **incremental development model** | **增量开发模型** | 一种开发生命周期：项目被划分为一系列增量，每一增量都交付整个项目需求中的一部分功能。需求按优先级进行划分，并按优先级在适当的增量中交付。在这种生命周期模型的一些版本中（但不是全部），每个子项目均遵循一个“微型的 V 模型”，具有自有的设计、编码和测试阶段。 |
| **incremental testing** | **增量测试** | 每次集成并测试一个或若干组件/系统，直到所有组件/系统都已经被集成或测试的一种测试。 |
| **independence** | **独立** | 职责分离，有助于客观地进行测试。 [DO-178b] |
| **infeasible path** | **不可达路径** | 通过任何输入都无法执行到的路径。 |
| **informal review** | **非正式评审** | 一种不基于正式（文档化）过程的评审。 |
| **input** | **输入** | 被组件读取的变量（无论存储于组件之内还是之外）。 |
| **input domain** | **输入域** | 有效输入的集合。参见 domain |
| **input value** | **输入值** | 输入的一个实例。参见 input |
| **inspection** | **审查** | 一种同级评审，通过检查文档以检测缺陷，例如不符合开发标准，不符合更上层的文档等。这是最正式的评审技术，因此总是基于文档化的过程。[IEEE 610, IEEE 1028] 参见 peer review |
| **inspection leader** | **审查负责人** | 参见 moderator |
| **inspector** | **检视人/审查员** | 参见 reviewer |
| **installability** | **可安装性** | 软件产品在指定环境下进行安装的性能。 [ISO 9126] 参见 portability |
| **installability testing** | **可安装性测试** | 测试软件产品可安装性的过程。 参见 portability testing |
| **installation guide** | **安装指南** | 帮助安装人员完成安装过程的使用说明，可存放在任何合适的介质上。可能是操作指南、详细步骤、安装向导或任何其他类似的过程描述。 |
| **installation wizard** | **安装向导** | 帮助安装人员完成安装过程的软件，可存放在任何合适的介质上。它通常会运行安装过程、反馈安装结果，并提示安装选项。 |
| **instrumentation** | **探测** | 在程序中插入附加代码，以便在程序执行时收集其执行信息。例如，用于度量代码覆盖。 |
| **instrumenter** | **探测工具** | 用于执行探测的软件工具。 |
| **intake test** | **预测试** | 冒烟测试的一种特例，用于决定组件/系统是否能够进行更深入的测试。通常在测试执行的初始阶段实施。 |
| **integration** | **集成** | 把组件/系统合并为更大部件的过程。 |
| **integration testing** | **集成测试** | 一种旨在暴露接口以及集成组件/系统间交互时存在的缺陷的测试。参见 component integration testing, system integration testing |
| **integration testing in the large** | **系统集成测试** | 参见 system integration testing |
| **integration testing in the small** | **组件集成测试** | 参见 component integration testing |
| **interface testing** | **接口测试** | 一种集成测试类型，注重于测试组件/系统之间的接口。 |

| | | |
|---|---|---|
| **interoperability** | **互操作性** | 软件产品与一个或多个指定组件/系统进行交互的能力。［ISO 9126］参见 functionality |
| **interoperability testing** | **互操作性测试** | 判定软件产品可交互性的测试过程。参见 functionality testing |
| **invalid testing** | **无效性测试** | 使用应该被组件/系统拒绝的输入值进行的测试。参见 error tolerance |
| **isolation testing** | **隔离测试** | 将组件与其周边组件隔离后进行的测试。如果有必要，使用桩(stubs)或驱动器(drivers)来模拟周边程序。 |
| **item transmittal report** | **版本发布报告** | 参见 release note |
| **iterative development model** | **迭代开发模型** | 一种开发生命周期：项目被划分为大量迭代过程。一次迭代是一个完整的开发循环，并（对内或对外）发布一个可执行的产品，这是正在开发的最终产品的一个子集，通过不断迭代最终成型的产品。 |
| | | |
| **K** | | |
| **key performance indicator** | **关键性能指标** | 参见 performance indicator |
| **keyword driven testing** | **关键字驱动测试** | 一种脚本编写技术，所使用的数据文件不单包含测试数据和预期结果，还包含与被测程序相关的关键词。用于测试的控制脚本通过调用特别的辅助脚本来解释这些关键词。 |
| | | |
| **L** | | |
| **LCSAJ** | **LCSAJ** | （Linear Code Sequence And Jump）线性代码序列和跳转。包含以下三项（通常通过源代码清单的行号来识别）：可执行语句的线性序列的开始、结束以及在线性序列结尾控制流所转移到的目标行。 |
| **LCSAJ coverage** | **LCSAJ 覆盖** | 测试套件所检测的组件的 LCSAJ 百分比。LCSAJ 达到 100％意味着决策覆盖(decision coverage)为 100％。 |
| **LCSAJ testing** | **LCSAJ 测试** | 一种白盒测试设计技术，其测试用例用于执行 LCSAJ。 |
| **learnability** | **易学性** | 软件产品具有的易于用户学习的能力。[ISO 9126］参见 usability |
| **level test plan** | **级别测试计划** | 通常用于一个测试级别(test level)的测试计划。参见 test plan |

| | | |
|---|---|---|
| **link testing** | **组件集成测试** | 参见 component integration testing |
| **load testing** | **负载测试** | 一种通过增加负载来测量组件或系统的测试方法。例如：通过增加并发用户数和（或）事务数量来测量组件或系统能够承受的负载。参见 stress testing |
| **logic-coverage testing** | **逻辑覆盖测试** | 参见 white box testing [Myers] |
| **logic-driven testing** | **逻辑驱动测试** | 参见 white box testing |
| **logical test case** | **逻辑测试用例/抽象测试用例** | 参见 high level test case |
| **low level test case** | **详细测试用例** | 具有具体的（实现级别 implementation level）输入数据和预期结果的测试用例。抽象测试用例中所使用的逻辑运算符被替换为对应于逻辑运算符作用的实际值。参见 high level test case |
| | | |
| **M** | | |
| **maintenance** | **维护** | 软件产品交付后对其进行的修改，以修正缺陷，改善性能或其他属性，或者使其适应新的环境。[IEEE 1219] |
| **maintenance testing** | **维护测试** | 针对运行系统的更改，或者新的环境对运行系统的影响而进行的测试。 |
| **maintainability** | **可维护性** | 软件产品是否易于更改，以便修正缺陷、满足新的需求、使以后的维护更简单或者适应新的环境。[ISO 9126] |
| **maintainability testing** | **可维护性测试** | 判定软件产品的可维护性的测试过程。 |
| **management review** | **管理评审** | 由管理层或其代表执行的对软件采购、供应、开发、运作或维护过程的系统化评估，包括监控过程、判断计划和进度表的状态、确定需求及其系统资源分配，或评估管理方式的效用，以达到正常运作的目的。[IEEE 610, IEEE 1028] |
| **master test plan** | **主测试计划** | 通常针对多个测试级别的测试计划。参见 test plan |
| **maturity** | **成熟度** | （1）组织在其过程和工作实践上的有效性和高效性的能力。 参见 Capability Maturity Model, Test Maturity Model。（2）软件产品在存在缺陷的情况下避免失效的能力。[ISO 9126] 参见 reliability |
| **measure** | **测量** | 测度时赋予实体某个属性的数值或类别。[ISO 14598] |
| **measurement** | **测度** | 给实体赋予一个数值或类别以描述其某个属性的过程。[ISO 14598] |
| **measurement scale** | **度量标准** | 约束数据分析类型的标准。 |
| **memory leak** | **内存泄漏** | 程序的动态存储分配逻辑存在的缺陷，导致内存使用完毕后不能收回而不可用，最终导致程序因为内存缺乏而运行失败(fail)。 |

| | | |
|---|---|---|
| **metric** | **度量** | 测量所使用的方法或者度量标准(measurement scale)。[ISO 14598] |
| **migration testing** | **移植测试** | 参见 conversion testing |
| **milestone** | **里程碑** | 项目过程中预定义的（中间的）交付物和结果就绪的时间点 |
| **mistake** | **错误** | 参见 error |
| **moderator** | **主持人** | 负责检视或其他评审过程的负责人或主要人员 |
| **modified condition decision coverage** | **改进的条件判定覆盖** | 参见 condition determination coverage |
| **modified condition decision testing** | **改进的条件判定测试** | 参见 condition determination coverage testing |
| **modified multiple condition coverage** | **改进的复合条件覆盖** | 参见 condition determination coverage |
| **modified multiple condition testing** | **改进的复合条件测试** | 参见 condition determination coverage testing |
| **module** | **模块** | 参见 component |
| **module testing** | **模块测试** | 参见 component testing |
| **monitor** | **监测器/监视器** | 与被测组件/系统同时运行的软件工具或硬件设备，对组件/系统的行为进行监视、记录和分析。[IEEE 610] |
| **monitoring tool** | **监测工具/监视工具** | 参见 monitor |
| **multiple condition** | **复合条件/多重条件** | 参见 compound condition |
| **multiple condition coverage** | **复合条件覆盖** | 测试套覆盖的一条语句内的所有单条件结果组合的百分比。100％复合条件覆盖意味着 100％条件判定覆盖(condition determination coverage)。 |
| **multiple condition testing** | **复合条件测试** | 一种白盒测试设计技术，测试用例用来覆盖一条语句中的单条件所有可能的结果组合。 |
| **mutation analysis** | **变异分析** | 一种确定测试套件完整性的方法，即判定测试套件能够区分程序与其微变体之间区别的程度。 |
| **mutation testing** | **变异测试** | 参见 back-to-back testing |
| | | |
| **N** | | |
| **N-switch coverage** | **N 切换覆盖** | N+1 个转换的序列在一个测试套件中被覆盖的百分比。[Chow] |
| **N-switch testing** | **N 切换测试** | 一种状态转换测试的形式，其测试用例执行 N+1 个转换的所有有效序列。[Chow] 参见 state transition testing |

| | | |
|---|---|---|
| **negative testing** | **逆向测试** | 一种旨在表现组件/系统不能正常工作的测试。逆向测试取决于测试人员的想法，态度，而与特定的测试途径或测试设计技术无关，例如使用无效输入值测试或在异常情况下进行测试。[Beizer] |
| **non-conformity** | **不一致** | 没有实现指定的需求。［ISO 9000］ |
| **non-functional requirement** | **非功能需求** | 与功能性无关，但与可靠性(reliability)、高效性(efficiency)、可用性(usability)、可维护性(maintainability)和可移植性(portability)等属性相关的需求。 |
| **non-functional testing** | **非功能测试** | 对组件/系统中与功能性无关的属性（例如可靠性、高效性、可用性、可维护性和可移植性）进行的测试。 |
| **non-functional test design techniques** | **非功能测试设计技术** | 推导或选择非功能测试所需测试用例的过程，此过程依据对组件/系统的规格说明进行分析，而不考虑其内部结构。参见 black box test design technique |
| | | |
| **O** | | |
| **off-the-shore software** | **现货软件** | 面向大众市场（即大量用户）开发的软件产品，并且以相同的形式交付给许多客户。 |
| **operability** | **可操作性** | 软件产品被用户操作或控制的能力。［ISO 9126］参见 usability |
| **operational environment** | **运行环境** | 用户或客户现场所安装的硬件和软件产品，被测组件/系统将在此环境下使用。软件可能包括操作系统、数据库管理系统和其他应用程序。 |
| **operational profile testing** | **运行概况测试** | 对系统运作模型（执行短周期任务）及其典型应用概率的统计测试。[Musa] |
| **operational testing** | **运行测试** | 在组件/系统的运作环境下对其进行评估的一种测试。［IEEE 610］ |
| **oracle** | **基准** | 参见 test oracle |
| **outcome** | **结果** | 参见 result |
| **output** | **输出** | 组件填写的一个变量（无论存储在组件内部还是外部）。 |
| **output domain** | **输出域** | 可从中选取有效输出值的集合。参见 domain |
| **output value** | **输出值** | 输出的一个实例/实值。参见 output |
| | | |

| P | | |
|---|---|---|
| **pair programming** | **结对编程** | 一种软件开发方式，组件的代码(开发和/或测试)由两名程序员在同一台计算机上共同编写。这意味着实时地执行代码评审。 |
| **pair testing** | **结对测试** | 两个人员，比如两个测试人员、一个开发人员和一个测试人员或一个最终用户和一个测试人员，一起寻找缺陷。一般地，他们使用同一台计算机并在测试期间交替操控。 |
| **partition testing** | **划分测试** | 参见 equivalence partitioning [Beizer] |
| **pass** | **通过** | 如果一个测试的实际结果与预期结果相符，则认为此测试通过。 |
| **pass/fail criteria** | **通过/失败准则** | 用于判定测试项（功能）或特性通过或失败的决策规则。[IEEE 829] |
| **path** | **路径** | 组件/系统从入口(entry point)到出口(exit point)的一系列事件（例如，可执行语句）。 |
| **path coverage** | **路径覆盖** | 测试套件执行的路径所占的百分比。100%的路径覆盖意味着100%的线性代码序列和跳转(LCSAJ)覆盖。 |
| **path sensitizing** | **路径感知** | 选择一组输入值，以强制执行某指定路径。 |
| **path testing** | **路径测试** | 一种白盒测试设计技术，设计的测试用例用于执行路径。 |
| **peer review** | **同行评审** | 由研发产品的同事对软件产品进行的评审，目的在于识别缺陷并改进产品。例如，审查(inspetion)、技术评审(technical review)和走查(walkthrough)。 |
| **performance** | **性能** | 组件/系统在给定的处理周期和吞吐率(throughput rate)等约束下，完成指定功能的程度。[IEEE 610] 参见 efficiency |
| **performance indicator** | **性能指标** | 一种有效性(effectiveness)和/或高效性(efficiency)的高级(抽象)度量单位，用于指导和控制开发进展。例如，软件交付时间的偏差(lead-time slip for software devlopment)。[CMMI] |
| **performance testing** | **性能测试** | 判定软件产品性能的测试过程。参见 efficiency testing |
| **performance testing tool** | **性能测试工具** | 一种支持性能测试的工具，通常有两个功能：负载生成(load gerneartion)和测试事务(test transation)测量。负载生成可以模拟多用户或者大量输入数据。执行时，对选定的事务的响应时间进行测量并被记录。性能测试工具通常会生成基于测试日志的报告以及负载－响应时间图表。 |
| **phase test plan** | **阶段测试计划** | 通常用于一个测试阶段的测试计划。参见 test plan |
| **portability** | **可移植性** | 软件产品在不同硬件或软件环境之间迁移的简易性。[ISO 9126] |
| **portability testing** | **可移植性测试** | 判定软件产品可移植性的测试过程。 |

| | | |
|---|---|---|
| **postcondition** | **后置条件** | 执行测试或测试步骤后必须满足的环境和状态条件。 |
| **post-execution comparison** | **执行后比较** | 实际值与预期值的比较，在软件运行结束后执行。 |
| **precondition** | **前置条件** | 对组件/系统执行特定测试或测试步骤之前所必须满足的环境和状态条件。 |
| **predicted outcome** | **预期结果** | 参见 expected result |
| **pretest** | **预测试** | 参见 intake test |
| **priority** | **优先级** | 赋予某项（业务）重要性的级别，如，缺陷。 |
| **probe effect** | **探测影响** | 在测试时由于测试工具（例如，性能测试工具或监测器）对组件/系统产生的影响。比如，使用性能测试工具可能会使系统的性能有小幅度降低。 |
| **problem** | **问题** | 参见 defect |
| **problem management** | **问题管理** | 参见 defect management |
| **problem report** | **问题报告** | 参见 defect report |
| **process** | **过程** | 一组将输入转变为输出的相关活动。［ISO 12207］ |
| **process cycle test** | **过程周期测试** | 一种黑盒测试设计技术，设计的测试用例用于执行业务流程或过程。［TMap］ |
| **product risk** | **产品风险** | 与测试对象有直接关系的风险。参见 risk |
| **project** | **项目** | 一个项目是一组以符合特定需求为目的的，相互协同的，具有开始和结束时间的受控活动。这些特定需求包括限定的周期、成本和资源。［ISO 9000］ |
| **project risk** | **项目风险** | 与（测试）项目的管理与控制相关的风险。参见 risk |
| **program instrumenter** | **程序插装器** | 参见 instrumenter |
| **program testing** | **程序测试** | 参见 component testing |
| **project test plan** | **项目测试计划** | 参见 master test plan |
| **pseudo-random** | **伪随机** | 一个表面上随机的序列，但事实上是根据预定的序列生成的。 |
| | | |
| **Q** | | |
| **quality** | **质量** | 组件、系统或过程满足指定需求或用户/客户需要及期望的程度。［IEEE 610］ |
| **quality assurance** | **质量保证** | 质量管理的组成部分，提供达到质量要求的可信程度。［ISO 9000］ |
| **quality attribute** | **质量属性** | 影响某项质量的特性或特征。［IEEE 610］ |
| **quality characteristic** | **质量特征** | 参见质量属性（quality attribute）。 |
| **quality management** | **质量管理** | 在质量方面指导和控制一个组织的协同活动。通常包括建立质量策略和质量目标、质量计划、质量控制、质量保证和质量改进。［ISO 9000］ |

| | | |
|---|---|---|
| **R** | | |
| **random testing** | **随机测试** | 一种黑盒测试设计技术，选择测试用例以匹配某种运行概貌情况（可能使用伪随机生成算法）. 这种技术可用于测试非功能性的属性，比如可靠性和性能。 |
| **recorder** | **记录员** | 参见 scribe |
| **record/playback tool** | **录制/回放工具** | 参见 capture/playback tool |
| **recoverability** | **可恢复性** | 软件产品失效(faliure)后，重建其特定性能级别以及恢复数据的能力。［ISO 9126］参见 reliability |
| **recoverability testing** | **可恢复性测试** | 判定软件产品可恢复性的测试过程。参见 reliability testing |
| **recovery testing** | **恢复测试** | 参见 recoverability testing |
| **regression testing** | **回归测试** | 测试先前测试过并修改过的程序，确保更改没有给软件其他未改变的部分带来新的缺陷(defect)。软件修改后或使用环境变更后要执行回归测试。 |
| **regulation testing** | **规范性测试** | 参见 compliance testing |
| **release note** | **发布说明** | 标识测试项、测试项配置、目前状态及其他交付信息的文档，这些交付信息是由开发、测试和可能的其他风险承担者在测试执行阶段开始的时候提交的。[ISO 9126] |
| **reliability** | **可靠性** | 软件产品在一定条件下(规定的时间或操作次数等)，执行其必需的功能的能力。［ISO 9126］ |
| **reliability testing** | **可靠性测试** | 判定软件产品可靠性的测试过程。 |
| **replaceability** | **可替换性** | 在相同环境下，软件产品取代另一指定软件产品以达到相同目的的能力。［ISO 9126］参见 portability |
| **requirement** | **需求** | 系统必须满足的，为用户解决问题或达到目的，条件或者能力。通过系统或者系统的组件的运行以满足合同、标准、规格或其它指定的正式文档定义的要求。[IEEE 610] |
| **requirements-based testing** | **基于需求的测试** | 根据需求推导测试目标和测试条件以设计测试用例的方法。例如,执行特定功能的测试或探测诸如可靠性和可用性等非功能性属性的测试。 |
| **requirements management tool** | **需求管理工具** | 一种支持需求记录、需求属性（例如，优先级）和注解的工具，能够通过多层次需求和需求变更管理达到可追踪性。一些需求管理工具还支持静态分析，如一致性检查以及预定义的需求规则之间的冲突。 |
| **requirements phase** | **需求阶段** | 在软件生命周期中定义和文档化软件产品需求的阶段。［IEEE 610］ |

| | | |
|---|---|---|
| **resource utilization** | **资源使用** | 软件产品在规定的条件下执行其功能时，使用适当数量和类型资源的能力。例如，程序使用的主存储器和二级存储器容量，需要的临时或溢出文件的大小。［ISO 9126］参见 efficiency |
| **resource utilization testing** | **资源使用测试** | 判定软件产品资源使用的测试过程。参见 efficiency testing |
| **result** | **结果** | 测试执行的成果，包括屏幕输出、数据更改、报告和发出的通讯消息。参见 actual result, expected result |
| **resumption criteria** | **继续准则** | 在重新启动被中断(或者延迟)的测试时，必须重复执行的测试活动。［After IEEE 829］ |
| **re-testing** | **再测试** | 重新执行上次失败的测试用例，以验证纠错的正确性。 |
| **review** | **评审** | 对产品或产品状态进行的评估，以确定与计划的结果所存在的误差，并提供改进建议。例如，管理评审(management review)、非正式评审(informal review)、技术评审(technical review)、审查(inspection)和走查(walkthrough)。［After IEEE 1028］ |
| **reviewer** | **评审人** | 参与评审的人员，辨识并描述被评审产品或项目中的异常。在评审过程中，可以选择评审人员从不同角度评审或担当不同角色。 |
| **review tool** | **评审工具** | 对评审过程提供支持的工具。典型的功能包括计划评审、跟踪管理、通讯支持、协同评审以及对具体度量（单位）收集与报告的存储库。 |
| **risk** | **风险** | 将会导致负面结果的因素。通常表达成可能的(负面）影响。 |
| **risk analysis** | **风险分析** | 评估识别出的风险以估计其影响和发生的可能性的过程。 |
| **risk-based testing** | **基于风险的测试** | 倾向于探索和提供有关产品风险信息的测试。[After Gerrard] |
| **risk control** | **风险控制** | 为降低风险到或控制风险在指定级别而达成的决议和实施防范（度量）措施的过程。 |
| **risk identification** | **风险识别** | 使用技术手段(例如,头脑风暴(brainstorming)、检验表(checklists)和失败历史记录（faliure history））标识风险的过程。 |
| **risk management** | **风险管理** | 对风险进行标识、分析、优先级划分和控制所应用的系统化过程和实践。 |
| **risk mitigation** | **风险缓解** | 参见 risk control |
| **robustness** | **健壮性** | 在出现无效输入或压力环境条件下,组件/系统能够正常工作的程度。［IEEE 610］ 参见 error-tolerance, fault-tolerance |
| **robustness testing** | **健壮性测试** | 判定软件产品健壮性的测试。 |
| **root cause** | **根本原因** | 导致不一致的根本因素，并具有通过过程改进彻底清除的可能。 |

| S | | |
|---|---|---|
| safety | 安全性 | 软件产品在特定的使用环境中，达到对人、业务、软件、财产或环境可接受的危害风险级别的能力[ISO 9126]。 |
| safety testing | 安全性测试 | 判定软件产品安全性的测试。 |
| sanity test | 健全测试 | 参见冒烟测试(smoke test)。 |
| scalability | 可扩展性 | 软件产品可被升级以容纳更多负载的能力[Gerrard] |
| scalability testing | 可扩展性测试 | 判定软件产品可扩展性的测试。 |
| scenario testing | 场景测试 | 参见用例测试(use case testing)。 |
| scribe | 记录员 | 在评审会议中将每个提及的缺陷和任何过程改进建议记录到日志表单上的人员，记录员要确保日志表单易于阅读和理解。 |
| scripting language | 脚本语言 | 一种用于编写可执行测试脚本（这些脚本被测试执行工具使用，如录制/回放工具）的编程语言。 |
| security | 安全性 | 软件产品防止对程序和数据未授权访问（无论是有意的还是无意的)的能力的属性。[ISO 9126]。参见功能性(functionality)。 |
| security testing | 安全性测试 | 判定软件产品安全性的测试，参见功能性测试(functionality testing)。 |
| security testing tool | 安全性测试工具 | 测试安全特性和脆弱性的工具。 |
| security tool | 安全性工具 | 提高运行安全性的工具。 |
| serviceability testing | 服务能力测试 | 参见维护能力测试(maintainability test) |
| severity | 严重性 | 缺陷对组件/系统的开发或运行造成的影响程度。[IEEE 610] |
| simulation | 模拟 | 一个实际或抽象系统的特定行为特征由另一个系统来代表。[ISO 2382/1] |
| simulator | 模拟器 | 测试时所使用的设备、计算机程序或者系统，当提供一套控制的输入集时它们的行为或运行与给定的系统相似。[IEEE 610 DO178b]。参见模拟器(emulator) |
| site acceptance testing | 现场验收测试 | 用户/客户在他们现场进行的验收测试，以判定组件/系统是否符合他们的需求和业务流程，通常包括软件和硬件。 |
| smoke test | 冒烟测试 | 所有定义的/计划的测试用例的一个子集，它覆盖组件/系统的主要功能，以确保程序的绝大部分关键功能正常工作，但忽略细节部分。每日构建和冒烟测试是业界的最佳实践。参见预测试(intake test)。 |
| software | 软件 | 计算机程序、过程和可能与计算机系统运行相关的文档和数据。 |
| software feature | 软件特性 | 参见特性(feature)。 |

| | | |
|---|---|---|
| software quality | 软件质量 | 软件产品的功能和特性总和，能够达到规定的或隐含的需求。［ISO 9126］ |
| software quality characteristic | 软件质量特性 | 参见质量属性(quality attribute) |
| software test incident | 软件测试事件 | 参见事件(incident) |
| software test incident report | 软件测试事件报告 | 参见事件报告(incident report) |
| Software Usability Measurement Inventory(SUMI) | 软件可用性度量调查表 | 一种基于调查表的可用性测试技术，以评估组件/系统的可用性，如用户满意度。[Veenendaal] |
| source statement | 源语句 | 参见语句(statement) |
| specification | 规格说明 | 说明组件/系统的需求、设计、行为或其他特征的文档，常常还包括判断是否满足这些条款的方法。理想情况下，文档是以全面、精确、可验证的方式进行说明的。 |
| specification-based testing | 基于规格说明的测试 | 参见黑盒测试(black box testing) |
| specification-based test design technique | 基于规格说明的测试设计技术 | 参见黑盒测试设计技术(black box test design technique) |
| specified input | 特定的输入 | 在规格说明中预测结果的输入。 |
| stability | 稳定性 | 软件产品避免因更改后导致非预期结果的能力。(ISO9126) 参见可维护性(maintainability) |
| standard software | 标准软件 | 参见现货软件(off-the-shelf software) |
| standards testing | 标准测试 | 参见一致性测试(compliance testing) |
| state diagram | 状态图 | 一种图表， 描绘组件/系统所能呈现的状态，并显示导致或产生从一个状态转变到另一个状态的事件或环境。 |
| state table | 状态表 | 一种表格，显示每个状态的有效和无效的转换及可能的伴随事件。 |
| state transition | 状态转换 | 组件/系统的两个状态之间的转换。 |
| state transition testing | 状态转换测试 | 一种黑盒测试设计技术，所设计的测试用例用来执行有效和无效的状态转换。参见 N-切换测试(N-switch testing) |
| statement | 语句 | 编程语言的一个实体，一般是最小的、不可分割的执行单元。 |
| statement coverage | 语句覆盖 | 由测试套件运行的可执行语句的百分比。 |
| statement testing | 语句测试 | 一种白盒测试设计技术，所设计的测试用例用来执行语句。 |
| static analysis | 静态分析 | 分析软件工件（如：需求或代码），而不执行这些工作产品。 |
| static analysis tool | 静态分析工具 | 参见静态分析器(static analyzer) |
| static analyzer | 静态分析器 | 执行静态分析的工具。 |
| static code analysis | 静态代码分析 | 分析软件的源代码而不执行软件。 |

| | | |
|---|---|---|
| static code analyzer | 静态代码分析器 | 执行静态代码分析的工具。工具对源代码的一些特性进行检查，例如，对编码规范的遵循、质量度量或数据流异常等。 |
| static testing | 静态测试 | 对组件/系统进行规格或实现级别的测试，而不是执行这个软件。比如，代码评审或静态代码分析。 |
| statistical testing | 统计测试 | 用输入的统计分布模型来构造有代表性的测试用例的一种测试设计技术。参见运行概貌测试(operational profile testing)。 |
| status accounting | 状态记录 | 配置管理的一个要素,包括纪录和报告有效地管理配置所需的信息。这些信息包括被认可的配置标识的列表、提议的配置变更的状态和被认可的变更的实施状态。[IEEE 610] |
| storage | 存储 | 参见资源利用(resource utilization) |
| storage testing | 存储测试 | 参见资源利用测试(resource utilization testing) |
| stress testing | 压力测试 | 在规定的或超过规定的需求条件下测试组件/系统，以对其进行评估。[IEEE 610] 参见(load testing) |
| structure-based techniques | 基于结构的技术 | 参见白盒测试设计技术(white box test design technique) |
| structural coverage | 结构覆盖 | 基于组件/系统内部结构的覆盖度量 |
| structural test design technique | 结构测试设计技术 | 参见白盒测试设计技术(white box test design technique) |
| structural testing | 结构测试 | 参见白盒测试(white box testing) |
| structured walkthrough | 结构走查 | 参见走查(walkthrough) |
| stub | 桩 | 一个软件组件框架的实现或特殊目的实现，用于开发和测试另一个调用或依赖于该组件的组件。它代替了被调用的组件。 [IEEE 610] |
| subpath | 子路径 | 组件中的可执行语句序列。 |
| suitability | 适用性 | 软件产品为特定任务和用户目标提供一套合适功能的能力。 [ISO 9126]。 参见功能性(functionality) |
| suspension criteria | 暂停准则 | 用来（暂时性地）停止对测试条目进行的所有或部分测试活动的准则。[IEEE 829] |
| syntax testing | 语法测试 | 一种黑盒测试设计技术，测试用例的设计是以输入域和（或）输出域的定义的依据。 |
| system | 系统 | 组织在一起实现一个特定功能或一组功能的一套组件。[IEEE 610] |
| system integration testing | 系统集成测试 | 测试系统和包的集成；测试与外部组织（如：电子数据交换、国际互联网）的接口 |
| system testing | 系统测试 | 测试集成系统以验证它是否满足指定需求的过程。 [Hetzel] |
| | | |

| T | | |
|---|---|---|
| technical review | 技术评审 | 一种同行间的小组讨论活动，主要为了对所采用的技术实现方法达成共识。[Gilb 和 Graham, IEEE 1028] 参见同行评审(peer review)。 |
| test | 测试 | 一个或更多测试用例的集合 [IEEE 829]。 |
| test approach | 测试方法 | 针对特定项目的测试策略的实现，通常包括根据测试项目的目标和风险进行评估之后所做的决策、测试过程的起点、采用的测试设计技术、退出准则和所执行的测试类型。 |
| test automation | 测试自动化 | 应用软件来执行或支持测试活动，如测试管理、测试设计、测试执行和结果检验。 |
| test basis | 测试依据 | 能够从中推断出组件/系统需求的所有文档。测试用例是基于这些文档的。只能通过正式的修正过程来修正的文档称为固定测试依据。 [TMap] |
| test bed | 测试台 | 参见测试环境(test environment) |
| test case | 测试用例 | 为特定目标或测试条件(例如，执行特定的程序路径，或是验证与特定需求的一致性)而制定的一组输入值、执行入口条件、预期结果和执行出口条件。[IEEE 610] |
| test case design technique | 测试用例设计技术 | 参见测试设计技术(test design technique) |
| test case specification | 测试用例规格说明 | 为测试项指定一套测试用例（目标、输入、测试动作、期望结果、执行预置条件）的文档。[IEEE 829] |
| test case suite | 测试用例集 | 参见测试套(test suite) |
| test charter | 测试章程 | 对测试目标的陈述，还可能包括关于如何进行测试的测试思路。测试章程通常用在探索测试中。参见探索测试(exploratory testing) |
| test closure | 测试结束 | 从已完成的测试活动中收集数据，总结基于测试件及相关事实和数据的测试结束阶段，包括对测试件的最终处理和归档，以及测试过程评估(包含测试评估报告的准备)。参见测试过程(test process)。 |
| test comparator | 测试比较器 | 执行自动测试比较的测试工具。 |
| test comparison | 测试对比 | 区分被测组件/系统产生的实际结果和期望结果的差异的过程。测试对比可以在测试执行时进行（动态比较），或在测试执行之后进行。 |
| test completion criteria | 测试完成准则 | 参见退出准则(exit criteria)。 |
| test condition | 测试条件 | 组件/系统中能被一个或多个测试用例验证的条目或事件。例如，功能、事务、特性、质量属性或者结构化元素。 |
| test control | 测试控制 | 当监测到与预期情况背离时,制定和应用一组修正动作以使测试项目保持正常进行的测试管理工 |

| | | |
|---|---|---|
| | | 作。参见测试管理(test management) |
| test coverage | **测试覆盖** | 参见覆盖(coverage) |
| test cycle | **测试周期** | 针对一个可分辨的测试对象发布版本而执行的测试过程。 |
| test data | **测试数据** | 在测试执行之前存在的数据（如在数据库中），这些数据与被测组件/系统相互影响。 |
| test data preparation tool | **测试数据准备工具** | 一种测试工具，用于从已存在的数据库中挑选数据，或创建、生成、操作和编辑数据以备测试。 |
| test design | **测试设计** | 参见测试设计规格说明(test design specification) |
| test design specification | **测试设计规格说明** | 为一个测试条目指定测试条件（覆盖项）、具体测试方法并识别相关高层测试用例的文档 |
| test design technique | **测试设计技术** | 用来衍生和/或选择测试用例的步骤。 |
| test design tool | **测试设计工具** | 通过生成测试输入来支持测试设计的工具。测试输入可能来源于 CASE 工具库（如需求管理工具）中包含的规格，工具本身包含的特定测试条件。 |
| test driver | **测试驱动器** | 参见驱动器(driver)。 |
| test driven development | **测试驱动开发** | 在开发软件之后，运行测试用例之前，首先开发并自动化这些测试用例的一种软件开发方法 |
| test environment | **测试环境** | 执行测试需要的环境，包括硬件、仪器、模拟器、软件工具和其他支持要素。 |
| test evaluation report | **测试评估报告** | 在测试过程的结尾用来总结所有的测试活动和结果的文档。也包括测试过程的评估和吸取的教训。 |
| test execution | **测试执行** | 对被测组件/系统执行测试，产生实际结果的过程。 |
| test execution automation | **测试执行自动化** | 使用软件（例如捕捉/回放工具）来控制测试的执行、实际结果和期望结果的对比、测试预置条件的设置和其它的测试控制和报告功能。 |
| test execution phase | **测试执行阶段** | 软件开发生命周期的一个阶段，在这个阶段里执行软件产品的组件，并评估软件产品以确定是否满足需求。 |
| test execution schedule | **测试执行时间表** | 测试过程的执行计划。这些测试过程包含在测试执行时间表中，执行时间表列出了执行任务间的关联和执行的顺序。 |
| test execution technique | **测试执行技术** | 用来执行实际测试的方法，包括手工的和自动的。 |
| test execution tool | **测试执行工具** | 使用自动化测试脚本执行其他软件（如捕捉/回放）的一种测试工具。[Fewster 和 Graham] |
| test fail | **测试失败** | 参见失败(fail)。 |
| test generator | **测试产生器** | 参见测试数据准备工具(test data preparation tool)。 |
| test harness | **测试用具** | 包含执行测试需要的桩和驱动的测试环境。 |
| test incident | **测试事件** | 参见事件(incident)。 |
| test incident report | **测试事件报告** | 参见事件报告(incident report) |
| test infrastructure | **测试基础设施** | 执行测试所需的组成物件，包括测试环境、测试 |

| | | |
|---|---|---|
| | | 工具、办公环境和过程。 |
| test input | **测试输入** | 在测试执行过程中，测试对象从外部源接收到的数据。外部源可以是硬件、软件或人。 |
| test item | **测试项** | 需被测试的单个要素。通常是一个测试对象包含多个测试项。参见测试对象(test object)。 |
| test item transmittal report | **测试项移交报告** | 参见发布说明(release note)。 |
| test leader | **测试组长** | 参见测试经理(test manager)。 |
| test level | **测试级别** | 统一组织和管理的一组测试活动。测试级别与项目的职责相关联。例如，测试级别有组件测试、集成测试、系统测试和验收测试。[TMap] |
| test log | **测试日志** | 按时间顺序排列的有关测试执行所有相关细节的记录。 |
| test logging | **测试记录** | 把测试执行信息写进日志的过程。 |
| test manager | **测试经理** | 负责测试和评估测试对象的人。他（她）指导、控制、管理测试计划及调整对测试对象的评估。 |
| test management | **测试管理** | 计划、估计、监控和控制测试活动，通常由测试经理来执行。 |
| test management tool | **测试管理工具** | 对测试过程中的测试管理和控制部分提供支持的工具。它通常有如下功能：测试件的管理、测试计划的制定、结果纪录、过程跟踪、事件管理和测试报告。 |
| Test Maturity Model (TMM) | **测试成熟度模型** | 测试过程改进的五级阶段框架，它与能力成熟度模型(CMM)相关，后者描述了有效测试过程的关键要素。 |
| test monitoring | **测试监控** | 处理与定时检查测试项目状态等活动相关的测试管理工作。准备测试报告来比较实际结果和期望结果。参见测试管理(test management)。 |
| test object | **测试对象** | 需要测试的组件或系统。参见测试项(test item)。 |
| test objective | **测试目标** | 设计和执行测试的原因或目的。 |
| test oracle | **测试准则** | 在测试时确定预期结果与实际结果进行比较的源。一个准则可能是现有的系统（用作基准），一份用户手册，或者是个人的专业知识，但不可以是代码。[Adrion] |
| test outcome | **测试结果** | 参见结果(result)。 |
| test pass | **测试通过** | 参见通过(pass)。 |
| test performance indicator | **测试绩效指标** | 一种高级别的度量，表明需要满足的某种程度的目标值或准则。通常与过程改进的目标相关。例如，缺陷探测率。 |
| test phase: | **测试阶段** | 组成项目的一个可管理阶段的一组独特的测试活动。例如，某测试级别的执行活动。[Gerrard] |
| test plan | **测试计划** | 描述预期测试活动的范围、方法、资源和进度的文档。它标识了测试项、需测试的特性、测试任务、任务负责人、测试人员的独立程度、测试环境、测试设计技术、测试的进入和退出准则和选择的合理性、需要紧急预案的风险，是测试策划过程的一份记录。[IEEE 829] |
| test planning | **测试策划** | 制定或更新测试计划的活动。 |

| | | |
|---|---|---|
| test policy | 测试方针 | 描述有关组织测试的原则、方法和主要目标的高级文档。 |
| Test Point Analysis (TPA) | 测试点分析（TPA） | 基于功能点分析的一种公式化测试估计方法。[TMap] |
| test procedure | 测试规程 | 参见测试规程规范(test procedure specification)。 |
| test procedure specification | 测试规程规格说明 | 规定了执行测试的一系列行为的文档。也称为测试脚本或手工测试脚本。[IEEE 829] |
| test process | 测试过程 | 基本的测试过程包括计划、规约、执行、记录、检查完全性和测试结束活动。 |
| Test Process Improvement (TPI) | 测试过程改进（TPI） | 用于测试过程改进的一个连续框架，描述了有效测试过程的关键要素，特别针对于系统测试和验收测试。 |
| test record | 测试记录 | 参见测试日志(test log)。 |
| test recording | 书写测试记录 | 参见测试日志(test logging)。 |
| test repeatability | 测试重复性 | 一个测试的属性，表明每次执行一个测试时是否产生同样的结果。 |
| test report | 测试报告 | 参见测试总结报告(test summary report)。 |
| test requirement | 测试需求 | 参见测试条件(test condition)。 |
| test run | 测试运行 | 对测试对象的特定版本执行测试。 |
| test run log | 测试运行日志 | 参见测试日志(test log)。 |
| test result | 测试结果 | 参见结果(result)。 |
| test scenario | 测试场景 | 参见测试规程规约(test procedure specification)。 |
| test script | 测试脚本 | 通常指测试规程规约，尤其是自动化的。 |
| test set | 测试集 | 参见测试套件(test suite)。 |
| test situation | 测试状况 | 参见测试条件(test condition)。 |
| test specification | 测试规约说明 | 由测试设计规约、测试用例规约和/或测试规程规约组成的文档。 |
| test specification technique | 测试规约说明技术 | 参见测试设计技术(test design technique)。 |
| test stage | 测试阶段 | 参见测试级别(test level)。 |
| test strategy | 测试策略 | 一个高级文档，该文档定义了需要对程序（一个或多个项目）执行的测试级别和需要进行的测试。 |
| test suite | 测试套件 | 用于被测组件/系统的一组测试用例。在这些测试用例中，一个测试的出口条件通常用作下个测试的入口条件。 |
| test summary report | 测试总结报告 | 总结测试活动和结果的文档。也包括对测试项是否符合退出准则进行的评估。 |
| test target | 测试目标 | 参见退出准则(exit criteria)。 |
| test technique | 测试技术 | 参见测试设计技术(test design technique)。 |
| test tool | 测试工具 | 支持一个或多个测试活动（例如，计划和控制、规格制定、建立初始文件和数据、测试执行和测试分析）的软件产品。[TMap] 参见 CAST. |
| test type | 测试类型 | 旨在针对特定测试目标，测试组件/系统的一组测试活动。例如，功能测试、易用性测试、回归测试等。一个测试类型可能发生在一个或多个测试 |

| | | |
|---|---|---|
| | | 级别或测试阶段上。［TMap］ |
| testability | **可测试性** | 软件产品修改后被测试的能力。[ISO 9126] 参见可维护性(maintainability)。 |
| testability review | **可测试性评审** | 详细检查测试依据，以判定测试依据在测试过程中作为输入文档是否达到质量要求。 |
| testable requirements | **可测的需求** | 对需求的一种程度说明，表示是可依据需求进行测试设计（以及后续的测试用例）和执行测试，以及判断是否满足需求。[IEEE 610] |
| tester | **测试员** | 参与测试组件/系统的专业技术人员。 |
| testing | **测试** | 包括了所有生命周期活动的过程，有静态的也有动态的。涉及到计划、准备和对软件及其相关工作产品的评估，以发现缺陷来判定软件或软件的工作产品是否满足特定需求，证明它们是否符合目标。 |
| testware | **测试件** | 在测试过程中产生的测试计划、测试设计和执行测试所需要的人工制品。例如，文档、 脚本、输入、预期结果、安装和清理步骤、文件、数据库、环境和任何在测试中使用的软件和工具。[Fewster 和 Graham] |
| thread testing | **线程测试** | 组件集成测试的一个版本，其中，组件的渐进式集成遵循需求子集的实现，与按层次的组件集成相反。 |
| time behavior | **时间行为** | 参见性能(performance)。 |
| top-down testing | **自顶向下测试** | 集成测试的一种递增实现方式，首先测试最顶层的组件，其它组件使用桩来模拟，然后已被测试过的组件用于测试更低层的组件，直到最底层的组件被测试。参见集成测试(integration testing)。 |
| traceability | **可追溯性** | 识别文档和软件中相关联条目的能力。例如，需求与相关测试关联。参见水平可跟踪性(horizontal traceability)，垂直可跟踪性(vertical traceability)。 |
| | | |
| **U** | | |
| understandability | **可理解性** | 软件产品对于用户是否易于理解、软件是否适用、怎样应用于特定任务和应用的条件的能力。 |
| unit | **单元** | 参见组件(component)。 |
| unit testing | **单元测试** | 参见组件测试(component testing)。 |
| unreachable code | **不可达代码** | 不能够到达因而不可能被执行的代码。 |
| usability | **可用性** | 软件能被理解、学习、使用和在特定应用条件下吸引用户的能力。[ISO 9126] |

| | | |
|---|---|---|
| usability testing | 可用性测试 | 用来判定软件产品的可被理解、易学、易操作和在特定条件下吸引用户程度的测试。 |
| use case | 用例 | 用户和系统进行对话过程中的一系列交互，能够产生实际的结果。 |
| use case testing | 用例测试 | 一种黑盒测试设计技术，所设计的测试用例用于执行用户场景。 |
| user acceptance testing | 用户验收测试 | 参见验收测试(acceptance testing)。 |
| user scenario testing | 用户场景测试 | 参见用例测试(use case testing)。 |
| user test | 用户测试 | 由真实用户参与的评估组件/系统可用性的测试。 |
| | | |
| **V** | | |
| V-model | V-模型 | 描述从需求定义到维护的整个软件开发生命周期活动的框架。V-模型说明了测试活动如何集成于软件开发生命周期的每个阶段。 |
| validation | 确认 | 通过检查和提供客观证据来证实特定目的功能或应用已经实现。[ISO 9000] |
| variable | 变量 | 计算机中的存储元素，软件程序通过其名称来引用。 |
| verification | 验证 | 通过检查和提供客观证据来证实指定的需求是否已经满足。[ISO 9000] |
| vertical traceability | 垂直可跟踪性 | 贯穿开发文档到组件层次的需求跟踪。 |
| version control | 版本控制 | 参见配置控制(configuration control)。 |
| volume testing | 容量测试 | 使用大容量数据对系统进行的一种测试。参见资源利用测试(resource-utilization testing)。 |
| | | |
| **W** | | |
| walkthrough | 走查 | 由文档作者逐步陈述文档内容，以收集信息并对内容达成共识。[Freedman 和 Weinberg, IEEE 1028]。参见同行评审(peer review)。 |
| white-box test design technique | 白盒测试设计技术 | 通过分析组件/系统的内部结构来产生和/或选择测试用例的规程。 |
| white-box testing | 白盒测试 | 通过分析组件/系统的内部结构进行的测试。 |

| | | |
|---|---|---|
| Wide Band Delphi | 宽带德尔菲法 | 一种专家测试评估的方法，旨在集团队成员的智慧来做精确的评估。 |