不懂测试行话”是不少软件测试新人经常面临的一个普遍问题。

“行有行规”，不懂行话危害极大。各位可能对《林海雪原》中杨子荣与座山雕见面时讲的“江湖黑话”印象比较深刻吧。座山雕说“天王盖地虎”，杨子荣对“宝塔镇河妖”，如果杨子荣不懂行话，肯定性命难保，难以完成铲除土匪的重任。

软件测试也有很多“行话”。这里的“行话”就是指各种测试术语。对于软件测试的新手而言，特别对于进行软件国际化/本地化测试，由于测试文档都是英语的，而且不少经常使用这些术语的缩写形式，所以经常令初学者感到困惑。

不懂这些测试专业术语，当然不会“掉脑袋”，但是将会影响测试的正确理解，产生测试错误，影响测试质量和效率，也影响与测试团队的交流。

现代软件测试属于比较新兴的学科，测试的类型众多，测试技术和理论还在不断发展，更多的新名词、新术语将会不断出现。即使对于多年测试经验的“老江湖”，也需要不断跟踪和学习。

如何跨越软件测试术语这只“拦路虎”，是摆在测试人员面前的一大难题。

笔者根据以往的测试经验和对测试知识的学习，结合软件国际化/本地化软件测试的实际需要，对最常见的软件测试术语进行了归纳和整理，按照字母顺序排序，进行了英文和中文的对照，并进行了简短介绍。希望这有助于测试新手快速理解术语，早日跨进软件测试大门。

Aeptance testing（验收测试），系统开发生命周期方法论的一个阶段，这时相关的用户和 / 或独立测试人员根据测试计划和结果对系统进行测试和接收。它让系统用户决定是否接收系统。它是一项确定产品是否能够满足合同或用户所规定需求的测试。这是管理性和防御性控制。

Ad hoc testing（随机测试），没有书面测试用例、记录期望结果、检查列表、脚本或指令的测试。主要是根据测试者的经验对软件进行功能和性能抽查。随机测试是根据测试说明书执行用例测试的重要补充手段，是保证测试覆盖完整性的有效方式和过程。

Alpha testing（α测试),是由一个用户在开发环境下进行的测试，也可以是公司内部的用户在模拟实际操作环境下进行的受控测试，Alpha测试不能由程序员或测试员完成。

Automated Testing（自动化测试），使用自动化测试工具来进行测试，这类测试一般不需要人干预，通常在GUI、性能等测试中用得较多。

Beta testing(β测试),测试是软件的多个用户在一个或多个用户的实际使用环境下进行的测试。开发者通常不在测试现场，Beta测试不能由程序员或测试员完成。

Black box testing（黑盒测试），指测试人员不关心程序具体如何实现的一种测试方法。根据软件的规格对软件进行各种输入和观察软件的各种输出结果来发现软件的缺陷的测试，这类测试不考虑软件内部的运作原理，因此软件对用户来说就像一个黑盒子。

Bug?(错误)，有时称作defect（缺陷）或error（错误），软件程序中存在的编程错误，可能会带来不必要的副作用，软件的功能和特性与设计规格说明书或用户需求不一致的方面。软件缺陷表现特征为：软件未达到产品说明书标明的功能；软件出现产品说明书指明不会出现的错误；软件功能超出产品说明书指明的范围；虽然产品说明书未指出但是软件应达到的目标；软件测试人员或用户认为软件难以理解，不易使用，运行速度缓慢等问题。

Bug report（错误报告），也称为“Bug record（错误记录）”，记录发现的软件错误信息的文档，通常包括错误描述、复现步骤、抓取的错误图像和注释等。

Bug tracking system（错误跟踪系统，BTS），也称为“Defect?tracking system，DTS”，管理软件测试缺陷的专用数据库系统，可以高效率地完成软件缺陷的报告、验证、修改、查询、统计、存储等任务。尤其适用于大型多语言软件的测试管理。

Build（工作版本），软件开发过程中用于内部测试的功能和性能等不完善的软件版本。工作版本既可以是系统的可操作版本，也可以是展示要在最终产品中提供的部分功能的部分系统。

Compatibility Testing（兼容性测试），也称“Configuration testing（配置测试）”，测试软件是否和系统的其它与之交互的元素之间兼容，如：浏览器、操作系统、硬件等。验证测试对象在不同的软件和硬件配置中的运行情况。

Capture/Replay Tool（捕获/回放工具），一种测试工具，能够捕获在测试过程中传递给软件的输入，并且能够在以后的时间中，重复这个执行的过程。这类工具一般在GUI测试中用的较多。

Crash（崩溃），计算机系统或组件突然并完全的丧失功能，例如软件或系统突然退出或没有任何反应（死机）。

Debug（调试），开发人员确定引起错误的根本原因和确定可能的修复措施的过程。一般发生在子系统或单元模块编码完成时，或者根据测试错误报告指出错误以后，开发人员需要执行调试过程来解决已存在的错误。

Deployment（部署），也称为shipment(发布)，对内部IT系统而言，指它的第一个版本通过彻底的测试、形成产品、交付给付款客户的阶段。

Dynamic testing（动态测试），通过执行软件的手段来测试软件。

Exception（异常/例外），一个引起正常程序执行挂起的事件。

Functional testing（功能测试），也称为behavioral testing（行为测试），根据产品特征、操作描述和用户方案，测试一个产品的特性和可操作行为以确定它们满足设计需求。本地化软件的功能测试，用于验证应用程序或网站对目标用户能正确工作。使用适当的平台、浏览器和测试脚本，以保证目标用户的体验将足够好，就像应用程序是专门为该市场开发的一样。

Garbage characters（乱码字符），程序界面中显示的无意义的字符，例如，程序对双字

节字符集的字符不支持时，这些字符不能正确显示。

GB 18030 testing（GB 18030测试），软件支持GB 18030字符集标准能力的测试，包括GB 18030字符的输入、输出、显示、存储的支持程度。

Installing testing（安装测试），确保该软件在正常情况和异常情况的不同条件下，例如，进行首次安装、升级、完整的或自定义的安装都能进行安装。异常情况包括磁盘空间不足、缺少目录创建权限等。核实软件在安装后可立即正常运行。安装测试包括测试安装代码以及安装手册。安装手册提供如何进行安装，安装代码提供安装一些程序能够运行的基础数据。

Integration testing（集成测试），被测试系统的所有组件都集成在一起，找出被测试系统组件之间关系和接口中的错误。该测试一般在单元测试之后进行。

International testing（国际化测试），国际化测试的目的是测试软件的国际化支持能力，发现软件的国际化的潜在问题，保证软件在世界不同区域中都能正常运行。国际化测试使用每种可能的国际输入类型，针对任何区域性或区域设置检查产品的功能是否正常，软件国际化测试的重点在于执行国际字符串的输入/输出功能。国际化测试数据必须包含东亚语言、德语、复杂脚本字符和英语（可选）的混合字符。

Localizability testing(本地化能力测试)，本地化能力是指不需要重新设计或修改代码，将程序的用户界面翻译成任何目标语言的能力。为了降低本地化能力测试的成本，提高测试效率，本地化能力侧是通常在软件的伪本地化版本上进行。本地化能力测试中发现的典型错误包括：字符的硬编码（即软件中需要本地化的字符写在了代码内部），对需要本地化的字符长度设置了国定值，在软件运行时以控件位置定位，图标和位图中包含了需要本地化的文本，软件的用户界面与文档术语不一致等。

Load testing（负载测试），通过测试系统在资源超负荷情况下的表现，以发现设计上的错误或验证系统的负载能力。在这种测试中，将使测试对象承担不同的工作量，以评测和评估测试对象在不同工作量条件下的性能行为，以及持续正常运行的能力。负载测试的目标是确定并确保系统在超出最大预期工作量的情况下仍能正常运行。此外，负载测试还要评估性能特征，例如，响应时间、事务处理速率和其他与时间相关的方面。

Localization testing（本地化测试），本地化测试的对象是软件的本地化版本。本地化测试的目的是测试特定目标区域设置的软件本地化质量。本地化测试的环境是在本地化的操作系统上安装本地化的软件。从测试方法上可以分为基本功能测试，安装/卸载测试，当地区域的软硬件兼容性测试。测试的内容主要包括软件本地化后的界面布局和软件翻译的语言质量，包含软件、文档和联机帮助等部分。

Performance testing（性能测试），评价一个产品或组件与性能需求是否符合的测试。包括负载测试、强度测试、数据库容量测试、基准测试等类型。

Pilot testing（引导测试），软件开发中，验证系统在真实硬件和客户基础上处理典型操作的能力。在软件外包测试中，引导测试通常是客户检查软件测试公司测试能力的一种形式，只有通过了客户特定的引导测试，软件测试公司才能接受客户真实软件项目的软件测试。

Portability testing（可移植性测试），测试瞄准于证明软件可以被移植到指定的硬件

或软件平台上。Priority（优先权），从商业角度出发是指错误的重要性，尤其是从客户和用户的角度出发，是指错误对于系统的可行性和可接受性的影响。与“Severity（严重性）”相对照。

Quality assurance（质量保证QA），采取的所有活动以保证一个开发组织交付的产品满足性能需求和已确立的标准和过程。

Regression testing（回归测试），在发生修改之后重新测试先前的测试以保证修改的正确性。理论上，对软件的任何新版本，都需要进行回归测试，验证以前发现和修复的错误是否在新软件版本上再现。

Review（评审），在产品开发过程中，把产品提交给项目成员、用户、管理者或其它相关人员评价或批准的过程。

Sanity testing（健全测试），软件主要功能成分的简单测试以保证它是否能进行基本的测试。参考“Smoke testing（冒烟测试）”。

Screen shot（抓屏、截图），软件测试中，将软件界面中的错误（窗口、菜单、对话框等）的全部或一部分，使用专用工具存储成图像文件，以便于后续处理。

Severity（严重性），错误对被测系统的影响程度，在终端用户条件下发生的可能性，软件错误妨碍系统使用的程度。与“Priority（优先权）”相对照。

Smoke testing（冒烟测试），冒烟测试的对象是每一个新编译的需要正式测试的软件版本，目的是确认软件基本功能正常，可以进行后续的正式测试工作。冒烟测试的执行者是版本编译人员。参考“Sanity testing（健全测试）”。

Software life cycle（软件生命周期），开始于一个软件产品的构思，结束于该产品不再被使用的这段期间。

Static testing（静态测试），不通过执行来测试一个系统。如代码检查，文档检查和评审等。

Structured query language（结构化查询语句，SQL），在一个关系数据库中查询和处理数据的一种语言。

TBD(To be determined，待定)，在测试文档中标是一项进行中的尚未最终确定的工作。

Test（测试），执行软件以验证其满足指定的需求并检测错误的过程。检测已有条件之间的不同，并评价软件项的特性软件项的分析过程。软件工程过程的一个活动，它将软件在预定的条件下运行以判断软件是否符合预期结果。

Test case（测试用例），为特定目标而开发的一组测试输入、执行条件和预期结果，其目标可以是测试某个程序路径或核实是否满足某个特定的需求。

Testing coverage（测试覆盖），指测试系统覆盖被测试系统的程度，一项给定测试或一组测试对某个给定系统或构件的所有指定测试用例进行处理所达到的程度。

Testing environment（测试环境），进行测试的环境，包括测试平台、测试基础设施、测试实验室和其他设施。

Testing item（测试项），作为测试对象的工作版本。

Testing plan（测试计划），描述了要进行的测试活动的范围、方法、资源和进度的文档。它确定测试项、被测特性、测试任务、谁执行任务，并且任何风险都要冲突计划。

Testing procedure（测试过程），指设置、执行给定测试用例并对测试结果进行评估的一系列详细步骤。

Testing script（测试脚本），一般指的是一个特定测试的一系列指令，这些指令可以被自动化测试工具执行。

Testing suite（测试包），一组测试用里的执行框架；一种组织测试用例的方法。在测试包里，测试用例可以组合起来创造出独特的测试条件。

Unit testing（单元测试），指一段代码的基本测试，其实际大小是未定的，通常是一个函数或子程序，一般由开发者执行。

User interface（用户界面，UI），广义是指使用户可以和计算机进行交互的硬件和/或软件。狭义是指软件中的可见外观及其底层与用户交互的部分（菜单、对话框、窗口和其它控件）。

User interface testing（用户界面测试），指测试用户界面的风格是否满足客户要求，文字是否正确，页面是否美观，文字，图片组合是否完美，操作是否友好等等。UI 测试的目标是确保用户界面会通过测试对象的功能来为用户提供相应的访问或浏览功能。确保用户界面符合公司或行业的标准。包括用户友好性、人性化、易操作性测试。

White box testing（白盒测试），根据软件内部的工作原理分析来进行测试,基于代码的测试，测试人员通过阅读程序代码或者通过使用开发工具中的单步调试来判断软件的质量，一般白盒测试由项目经理在程序员开发中来实现。