**如何设计和执行测试用例**

测试需求收集完毕后，开始测试设计。

　　测试用例是什么?测试用例就是一个文档，描述输入、动作、或者时间和一个期望的结果，其目的是确定应用程序的某个特性是否正常的工作。设计测试用例需要考虑以下问题：

　　测试用例的基本格式：

　　软件测试用例的基本要素包括测试用例编号、测试标题、重要级别、测试输入、操作步骤、预期结果，下面逐一介绍。

　　用例编号： 测试用例的编号有一定的规则，比如系统测试用例的编号这样定义规则： PROJECT1-ST-001 ，命名规则是项目名称+测试阶段类型(系统测试阶段)+编号。定义测试用例编号，便于查找测试用例，便于测试用例的跟踪。

　　测试标题： 对测试用例的描述，测试用例标题应该清楚表达测试用例的用途。比如 “ 测试用户登录时输入错误密码时，软件的响应情况 ” 。

　　重要级别： 定义测试用例的优先级别，可以笼统的分为 “ 高 ” 和 “ 低 ” 两个级别。一般来说，如果软件需求的优先级为 “ 高 ” ，那么针对该需求的测试用例优先级也为 “ 高 ” ;反之亦然，

　　测试输入： 提供测试执行中的各种输入条件。根据需求中的输入条件，确定测试用例的输入。测试用例的输入对软件需求当中的输入有很大的依赖性，如果软件需求中没有很好的定义需求的输入，那么测试用例设计中会遇到很大的障碍。

　　操作步骤： 提供测试执行过程的步骤。对于复杂的测试用例，测试用例的输入需要分为几个步骤完成，这部分内容在操作步骤中详细列出。

　　预期结果： 提供测试执行的预期结果，预期结果应该根据软件需求中的输出得出。如果在实际测试过程中，得到的实际测试结果与预期结果不符，那么测试不通过;反之则测试通过。

　　软件测试用例的设计主要从上述 6 个域考虑，结合相应的软件需求文档，在掌握一定测试用例设计方法的基础上，可以设计出比较全面、合理的测试用例。具体的测试用例设计方法可以参见相关的测试书籍，白盒测试方法和黑盒测试方法在绝大多数的软件测试书籍中都有详细的介绍。

　　一般来说，每个软件公司的项目可以分为固定的几大类。可以按业务类型划分，比如 ERP 软件、产品数据管理软件、通信软件、地理信息系统软件等等;可以按软件结构来划分，比如 B/S 架构的软件、 C/S 架构的软件、嵌入式软件等等。参考同类别软件的测试用例，会有很大的借鉴意义。如果，公司中有同类别的软件系统，千万别忘记把相关的测试用例拿来参考。如果，系统非常接近，甚至经过对测试用例简单修改就可以应用到当前被测试的软件。 “ 拿来主义 ” 可以极大的开阔测试用例设计思路，也可以节省大量的测试用例设计时间。

　　加强测试用例的评审：

　　测试用例设计完毕后，最好能够增加评审过程。

　　同行评审是 CMM3 级的一个 KPA ，如果因为公司没有通过 CMM3 级，就不开展同行评审是不恰当的。测试用例应该由产品相关的软件测试人员和软件开发人员评审，提交评审意见，然后根据评审意见更新测试用例。如果认真操作这个环节，测试用例中的很多问题都会暴露出来，比如用例设计错误、用例设计遗漏、用例设计冗余、用例设计不充分等等;如果同行评审不充分，那么，在测试执行的过程中，上述本应在评审阶段发现的测试用例相关问题，会给测试执行带来大麻烦，甚至导致测试执行挂起。

　　定义测试用例的执行顺序 ：

　　在测试用例执行过程中，你会发现每个测试用例都对测试环境有特殊的要求，或者对测试环境有特殊的影响。因此，定义测试用例的执行顺序，对测试的执行效率影响非常大。比如某些异常测试用例会导致服务器频繁重新启动，服务器的每次重新启动都会消耗大量的时间，导致这部分测试用例执行也消耗很多的时间。那么在编排测试用例执行顺序的时候，应该考虑把这部分测试用例放在最后执行，如果在测试进度很紧张的情况下，如果优先执行这部分消耗时间的异常测试用例，那么在测试执行时间过了大半的时候，测试用例执行的进度依然是缓慢的，这会影响到测试人员的心情，进而导致匆忙地测试后面的测试用例，这样测试用例的漏测、误测就不可避免，严重影响了软件测试效果和进度。因而，合理地定义测试用例的执行顺序是很有必要的。

　　测试用例执行

　　测试用例设计完毕后，接下来的工作是测试执行，测试执行中应该注意以下几个问题：

　　搭建软件测试环境，执行测试用例

　　测试用例执行过程中，搭建测试环境是第一步。一般来说，软件产品提交测试后，开发人员应该提交一份产品安装指导书，在指导书中详细指明软件产品运行的软硬件环境，比如要求操作系统系统是 Windows 2000 pack4 版本，数据库是 Sql Server 2000 等等，此外，应该给出被测试软件产品的详细安装指导书，包括安装的操作步骤、相关配置文件的配置方法等等。对于复杂的软件产品，尤其是软件项目，如果没有安装指导书作为参考，在搭建测试环境过程中会遇到种种问题。

　　如果开发人员拒绝提供相关的安装指导书，搭建测试中遇到问题的时候，测试人员可以要求开发人员协助，这时候，一定要把开发人员解决问题的方法记录下来，避免同样的问题再次请教开发人员，这样会招致开发人员的反感，也降低了开发人员对测试人员的认可程度。

　　测试执行过程应注意的问题 ：

　　测试环境搭建之后，根据定义的测试用例执行顺序，逐个执行测试用例。在测试执行中需要注意以下几个问题：

　　全方位的观察测试用例执行结果： 测试执行过程中，当测试的实际输出结果与测试用例中的预期输出结果一致的时候，是否可以认为测试用例执行成功了?答案是否定的，即便实际测试结果与测试的预期结果一致，也要查看软件产品的操作日志、系统运行日志和系统资源使用情况，来判断测试用例是否执行成功了。全方位观察软件产品的输出可以发现很多隐蔽的问题。以前，我在测试嵌入式系统软件的时候，执行某测试用例后，测试用例的实际输出与预期输出完全一致，不过在查询 CPU 占用率地时候，发现 CPU 占用率高达 90 %，后来经过分析，软件运行的时候启动了若干个 1ms 的定时器，大量的消耗的 CPU 资源，后来通过把定时器调整到 10ms ， CPU 的占用率降为 7 %。如果观察点单一，这个严重消耗资源的问题就无从发现了。

　　加强测试过程记录： 测试执行过程中，一定要加强测试过程记录。如果测试执行步骤与测试用例中描述的有差异，一定要记录下来，作为日后更新测试用例的依据;如果软件产品提供了日志功能，比如有软件运行日志、用户操作日志，一定在每个测试用例执行后记录相关的日志文件，作为测试过程记录，一旦日后发现问题，开发人员可以通过这些测试记录方便的定位问题。而不用测试人员重新搭建测试环境，为开发人员重现问题。

　　及时确认发现的问题： 测试执行过程中，如果确认发现了软件的缺陷，那么可以毫不犹豫的提交问题报告单。如果发现了可疑问题，又无法定位是否为软件缺陷，那么一定要保留现场，然后知会相关开发人员到现场定位问题。如果开发人员在短时间内可以确认是否为软件缺陷，测试人员给予配合;如果开发人员定位问题需要花费很长的时间，测试人员千万不要因此耽误自己宝贵的测试执行时间，可以让开发人员记录重新问题的测试环境配置，然后，回到自己的开发环境上重现问题，继续定位问题。

　　与开发人员良好的沟通： 测试执行过程中，当你提交了问题报告单，可能被开发人员无情驳回，拒绝修改。这时候，只能对开发人员晓之以理，做到有理、有据，有说服力。首先，要定义软件缺陷的标准原则，这个原则应该是开发人员和测试人员都认可的，如果没有共同认可的原则，那么开发人员与测试人员对问题的争执就不可避免了。此外，测试人员打算说服开发人员之前，考虑是否能够先说服自己，在保证可以说服自己的前提下，再开始与开发人员交流。

　　及时更新测试用例

　　测试执行过程中，应该注意及时更新测试用例。往往在测试执行过程中，才发现遗漏了一些测试用例，这时候应该及时的补充;往往也会发现有些测试用例在具体的执行过程中根本无法操作，这时候应该删除这部分用例;也会发现若干个冗余的测试用例完全可以由某一个测试用例替代，那么删除冗余的测试用例。

　　总之，测试执行的过程中及时地更新测试用例是很好的习惯。不要打算在测试执行结束后，统一更新测试用例，如果这样，往往会遗漏很多本应该更新的测试用例。

　　提交一份优秀的问题报告单 ：

　　软件测试提交的问题报告单和测试日报一样，都是软件测试人员的工作输出，是测试人员绩效的集中体现。因此，提交一份优秀的问题报告单是很重要的。软件测试报告单最关键的域就是 “ 问题描述 ” ，这是开发人员重现问题，定位问题的依据。问题描述应该包括以下几部分内容：软件配置、硬件配置、测试用例输入、操作步骤、输出、当时输出设备的相关输出信息和相关的日志等。

　　软件配置： 包括操作系统类型版本和补丁版本、当前被测试软件的版本和补丁版本、相关支撑软件，比如数据库软件的版本和补丁版本等。

　　硬件配置： 计算机的配置情况，主要包括 CPU 、内存和硬盘的相关参数，其它硬件参数根据测试用例的实际情况添加。如果测试中使用网络，那么网络的组网情况，网络的容量、流量等情况。硬件配置情况与被测试产品类型密切相关，需要根据当时的情况，准确翔实的记录硬件配置情况。

　　测试用例输入 \ 操作步骤 \ 输出： 这部分内容可以根据测试用例的描述和测试用例的实际执行情况如实填写。

　　输出设备的相关输出信息： 输出设备包括计算机显示器、打印机、磁带等等输出设备，如果是显示器可以采用抓屏的方式获取当时的截图，其他的输出设备可以采用其它方法获取相关的输出，在问题报告单中提供描述。

　　日志信息： 规范的软件产品都会提供软件的运行日志和用户、管理员的操作日志，测试人员应该把测试用例执行后的软件产品运行日志和操作日志作为附件，提交到问题报告单中。

　　根据被测试软件产品的不同，需要在 “ 问题描述 ” 中增加相应的描述内容，这需要具体问题具体分析。

　　测试结果分析

　　软件测试执行结束后，测试活动还没有结束。测试结果分析是必不可少的重要环节， “ 编筐编篓，全在收口 ” ，测试结果的分析对下一轮测试工作的开展有很大的借鉴意义。前面的 “ 测试准备工作 ” 中，建议测试人员走读缺陷跟踪库，查阅其他测试人员发现的软件缺陷。测试结束后，也应该分析自己发现的软件缺陷，对发现的缺陷分类，你会发现自己提交的问题只有固定的几个类别;然后，再把一起完成测试执行工作的其他测试人员发现的问题也汇总起来，你会发现，你所提交问题的类别与他们有差异。这很正常，人的思维是有局限性，在测试的过程中，每个测试人员都有自己思考问题的盲区和测试执行的盲区，有效的自我分析和分析其他测试人员，你会发现自己的盲区，有针对性的分析盲区，必定会在下一轮测试用避免盲区。

　　同时我们还应该注意设计测试用例的时候，有许多经典的测试理论。比如边界法、等价法，这些经常用到我们日常的工作中。当然也有许多的理论，比如正交分解法是使用起来非常费劲。往往转化为实际的容易理解的测试语言就非常困难。

　　测试的时候，我们也会碰到难堪的场景，那就是测试遗漏。

　　我们来分析下，开发的过程。开发拿到需求后，就会开发相应的代码，然后简单的测试下功能。代码之间有可能是互相调用的，代码可能影响到的模块，有些开发是知道的，有些是不知道的。如果是有关数据库的操作，一个地方的改动有可能影响了多个模块。所以问题的复杂性就体现在这里了。