软件测试专题期刊

SOLUTIONS TESTING 泽众软件

2012年10月 第十期

软件测试管理知识

谷歌的端到端测试方法：隔离式服务器

软件测试工程师指南

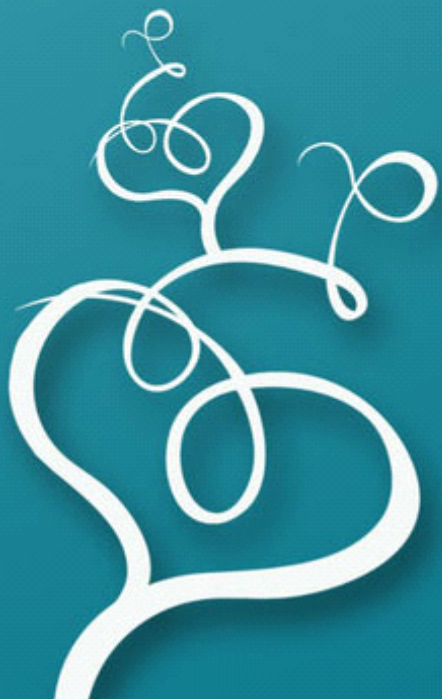
测试人员和开发人员相处的技巧

如何提高软件测试能力

测试发展的下一个阶段——Impact and Visibility

如何有效确定哪些是测试的重点

试析软件测试的错觉及发展方向



泽众软件——专业的软件测试门户

www.spasvo.com

上海泽众软件专题期刊

2012 年 10 月 第十期

主办单位：上海泽众软件科技有限公司

联系电话：021-61079698 传真：021-61079698 转 8017

意见反馈：fangmh@spasvo.com 投稿：wangmf@spasvo.com

公司地址：上海市普陀区曹杨路 450 号绿地和创大厦 18 楼 1801 室

邮政编码：200063

公司主页：www.spasvo.com 论坛：bbs.spasvo.com

目录

泽众软件——专业软件测试门户
2012 年 10 月 第十期 电子杂志期刊

软件测试管理知识

1、 测试团队结构是怎样的?

大多数测试团队，或者说传统测试团队，一般按照测试类型构建团队体系，如图所示：

优点：职能划分明确。

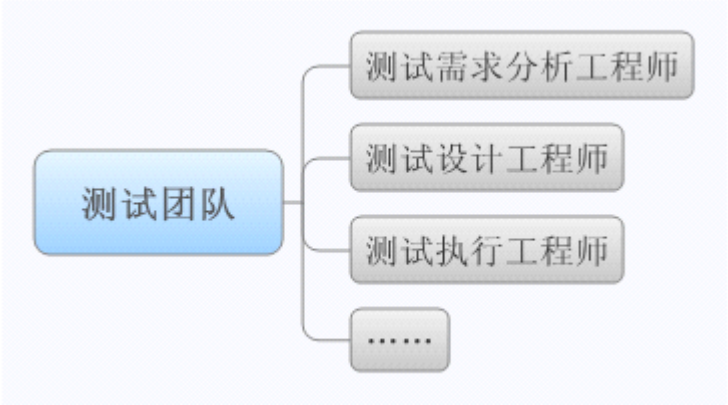
缺点：技能发展单一，协调成本较高。

有部分团队按照测试粒度构建体系，如图所示：

优点：测试提前。

缺点：测试成本偏高。

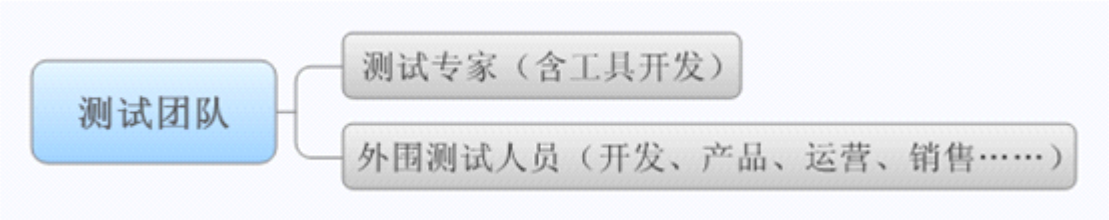
还有的按照测试阶段或者说测试能力构建体系，也就是通常说的流水线，如图所示：



优点：测试速度快。

缺点：测试职业发展容易形成瓶颈。

有极少数团队按照测试专业程度构建体系，这也是目前甚嚣尘上大肆鼓吹的结构，如图所示：



优点：测试成本低。

缺点：容易脱离实际业务。

上面几种结构本身并无高下之分，可结合团队实际情况进行选择。笔者所处团队的结构目前是第一种、第三种和第四种的混合体，如图所示：

优点：资源利用率最大化。

缺点：并行工作较多。

2、 开发需要什么样的测试?

测试时间短

测试质量高

善于交流

专业

深入了解产品

……

随随便便可以列一堆要求，但其实核心就一条，能做开发做不了的事。闻道有先后术业有专攻，测试自有其专业领域，测试人员的核心价值应该体现在哪?开发与测试的关系既泾渭分明又水乳交融，身为团队主导者应能准确辨别在当前整个研发体系中测试团队处在什么位置应起到何种作用。由此制定团队目标确定团队发展方向，而不是拍脑袋乱想，或者把测试团队孤立出来单独订目标。技术储备很重要，但技术储备的方向要靠主导者来确定，比开发更懂测试比测试更懂开发，这句玩笑话说出来真的很心酸，因为四不像。

一般测试团队会经历这么个过程：

草创，先不管别的能把基本的测试需求满足就好。

上升，普通的功能测试趋于成熟，开始引入性能、自动化测试等等，多采用第三方测试工具。

突破，有相当的测试积累，有较为丰富的测试资源，开始建立独有的测试体系，包括各种方法论与测试产品。

平稳，该做的好像都做的差不多了，也想不出什么革命性的创意，保持现状吧，开始著书立传。

下滑，人浮于事，毫无激情。

笔者见过的测试团队大多处在“突破”阶段，在此阶段要注意技术研究与实用性的关系。

说了半天，其实这个问题应该变成，企业需要什么样的测试团队。

3、 老板需要什么样的测试?

和上面问题有什么区别?有，上面是群体对测试的要求，这里是个体对测试的要求。

首先，这里的老板指的是整个研发体系的负责人，什么产品、开发、测试都在他那。

其次，有一说一，大多数老板对测试领域并没有太多深入的了解，对测试的认知更多来自其它团队的反馈及产品的最终质量。所以老板最关注的测试问题是什么呢?

测试成本：测试到底能为我带来多大的收益?这是每个老板都会问的问题。ROI 是每个人心里的一本账。笔者一直阐述资源利用率最大化、能量守恒的原因，就是建议用最少的资源办最多的事，要一个人承担多个任务而不是多个人做一件事。讲到这很多人会说你当我们不知道啊问题是怎么做到。如何降

低测试比例下文会谈到，但笔者在这首先想问，作为主管的你真的想缩小团队规模吗?是否因为其它因素反而想扩大规模?

技术含量：一家企业到底是以商业为主还是技术为主，这点不用费脑子多想，至少我们都是做技术的。测试技术到底包含哪些内容?上文提到过这里不重复。如果仍不清楚可以查阅笔者一系列文章。在这就说一点，很老套也很朴实，能发现更多更深入的别人发现不了的缺陷，就有技术含量，不管你在过

程中使用了何种技术何种方法。你说你用了多少高精尖的技术结果愣是一个缺陷没找到，有用吗?

产品质量：这条不用多说，缺陷预防才是王道啊。

团队协作：如果你认为开发、测试是两个团队，那么就一定是两个。职能划分明确没问题，但自扫门前雪就很有问题。这故障是开发弄的与测试无关，一旦有这种想法何谈协作?

无可否认，在整个研发体系中，测试不是核心，至少在当今各种各样的研发流程里它都不是。明确这一点，也就能明确老板到底需要什么样的测试团队了。

4、 如何提升测试开发比?

测试开发比应该是 1：3?1：4?1：10?甚至干脆就没有测试。这是个哲学问题，争论无止境。不过笔者还是要说，单纯的谈论测试开发比是毫无意义的，它涉及的因素太多太多，绝对不是越高越好。

列几个提升比例的切实有效的思路：

能量守恒：测试工作量不会消失而只会转移，转移给机器，转移给非测试人员。目前大多数团队的作法是转移给机器，这就是自动化测试长盛不衰的原因。题外话，虽然笔者并不认为自动化测试是银弹，但承认它至少是颗子弹。然后有少部分团队的作法是转移给人，即把测试工作发散出去，发散给非专业测试人员。说白了就是很容易开展测试活动，是个人就能来进行测试。想达到这点必须满足一个前提条件，待测产品具有极高的可测性。这也是笔者一直鼓吹的全民测试，测试工厂化、傻瓜化等等等等。具体如何提升产品可测性，请参看笔者另一篇文章《测试手段多样化》。

测试传承：百年老店，首重传承。这玩意的好处不想多说，当你团队人员流动率高的时候你就体会到了。传承要有体系，有脉络，绝对不能做成零散的海量信息。具体作法可参考国家图书馆、档案馆。笔者一直在做产品测试脉络图，也就是针对某个产品的各种功能点，分别要采用何种方法、手段进行测试，并把各个点给串起来，形成类似人体经脉那样的结构。

技术创新：智能化测试遥遥无期，笔者研究多年也没啥成效，汗颜。测试行业发展到现在，有多长时间没看见划时代革命性的创新了?就说放眼望去的大大小小测试工具，有哪一个是与众不同的?科学技术是第一生产力，同时，科学技术需要转换为生产力，尽量避免开发不实用的技术，更不要以开发不实用的艰深技术来展现自身能力。笔者建议大家，把各种测试手段融入到业务产品当中，也既是产品本身就具备测试功能，现在很多测试工具所实现的功能都可以放进去。最后，期待业内的测试专家，能开发出真正革命性、划时代的测试产品，很期待。

5、 怎样才是好的测试主管?

男要帅女要美，真的，不开玩笑。没看古时候入朝为官的士人讲究一表人才吗，长得挫状元都能给你降为探花。所以第一要素是形象气质俱佳，最好再有一副好口才。

第二，不要加班，没看错，至少不要明目张胆的加，要加偷偷摸摸的加。主管长期在公司加班很容易在团队内形成加班文化，甚至造成组员刻意把部分工作留在下班后完成。一旦形成此氛围对工作效率会造成极大的损害，大家会想反正要加班的，不急。忙或闲，每个人心里都清楚，最重要的是拿结果，能把活搞定管你在哪加班。

第三，是技术型还是管理型或者混合型都没差，但不能什么都不是，要有一方面在整个团队里无人可比。当然，指的是工作中用得上的能力，如果吃饭比别人吃的多那你顶多是个饭桶。题外话，能力与职位不符很痛苦的，不具备对应的能力还是别坐这个位置的好。做人做事讲天赋，不是什么能力都可以培养出来。

第四，准确评估工作合理进行工作分配。此点非常难做到，随便找本项目管理的书里面至少有一半在讲这个。在资源不足工作量大且有多数工作并行的情况下要做到合理分配那是难上加难，如果在此形式下还能让团队正常运转并且不加班，来来来，让大家叫你一声大师。

最后，好主管并不一定是好同事，做好主管就行了，无需要求自己面面俱到。

小记：从团队结构说到外部对测试的要求，再说到测试成本及团队管理者本身，其实笔者全文想说明的是，在他人眼中什么样的测试团队才是好团队，或者说能得到大多数人认可的团队是怎样的。

谷歌的端到端测试方法：隔离式服务器

Web 应用的测试经常需要进行从用户角度的系统性的功能测试，涉及到从用户请求到服务器响应的整个过程，也叫做端到端测试。在最近的 Google Test Blog 上，我看到一篇文章，觉得是进行这种测试一个不错的思路，翻译如下，供大家参考。

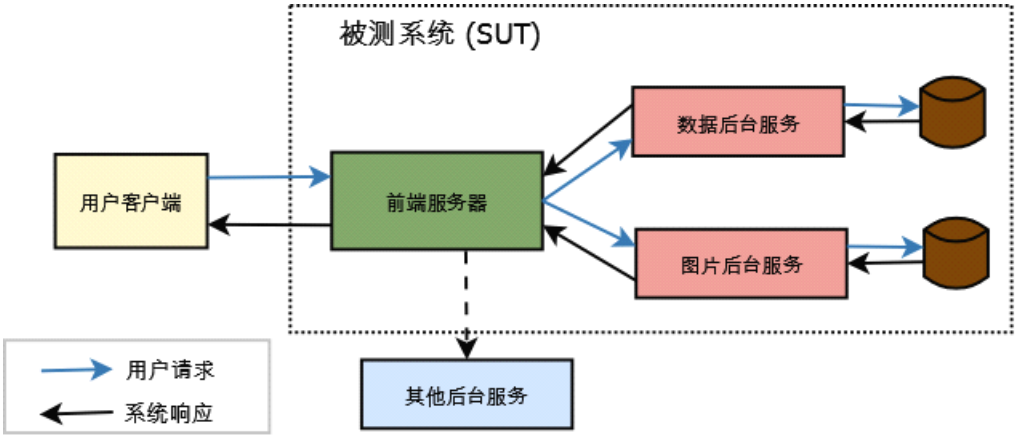
隔离式服务器

By Chaitali Narla and Diego Salas (原文地址)

让我们考虑一个复杂的 Web 应用程序。 实现它的可能是成堆的服务器，而且每台服务器都运行着不同的任务并相互通讯。 每个用户操作都会访问这个服务器集群，并经历一次从用户到数据存储再到用户的往返。 包括 GMail 和 Google+ 在内的很多 Google 的 Web 应用都是这样工作的。 那么我们如何为它们编写端到端的测试 (end-to-end test) 呢?

“端到端”的测试

在 Google 测试世界里，端到端测试是指作用于从用户请求到响应的整个流程和全部服务器集群的测试。下面是一个简化的由端到端测试所涵盖的被测系统 (SUT) 的示意图。注意图中被测系统的的前端服务器连接到了一个第三方后台服务器，但是这个第三方服务器并不被图中特定的用户请求所使用。



为这样的系统编写一个快速而可靠的端到端测试所面临的一个挑战是要避免网络访问。涉及到网络访问的测试要比那些只访问本地资源的测试执行得慢，而且访问外部服务器可能由于外部服务器不可用或不稳定而带来怪异的结果。

隔离式服务器

我们在 Google 设计端到端测试的一个技巧就是使用隔离式服务器 (Hemetic Servers)。

什么是隔离式服务器? 简短的定义就是“单机服务器”。 如果你能在一台单独的没有网络连接的机器上启动整个服务，并使其正常工作，它就是一台隔离式服务器! “隔离”的概念从更广义上来讲适用于所有隔离的系统而不一定是单台机器，但这里提到的是一个特例。

为什么隔离式服务器很有用呢? 因为如果整个被测系统由隔离式服务器构成，它就能在一台单独的机器上启动来用于测试;根本不需要网络连接! 这台单机可以是物理机，也可以是虚拟机。

设计隔离式服务器

构建隔离式服务器的过程需要从新服务器设计阶段的前期就要开始。需要注意下面一些事情：

服务器中所有与其他服务器的连接都需要在运行时注入。这可以通过一种适合的依赖注入形式完成，例如命令行标志或者 Guice 。

所有必需的静态文件都要打入服务器的二进制包。

如果服务器需要访问数据库，则需要保证数据库能被数据文件或者内存数据库模拟。

满足上述要求保证了我们的服务器具有高度的可配置性，从而有条件成为隔离式服务器。但它目前还不能被用于测试，我们还需要完成下面一些事情：

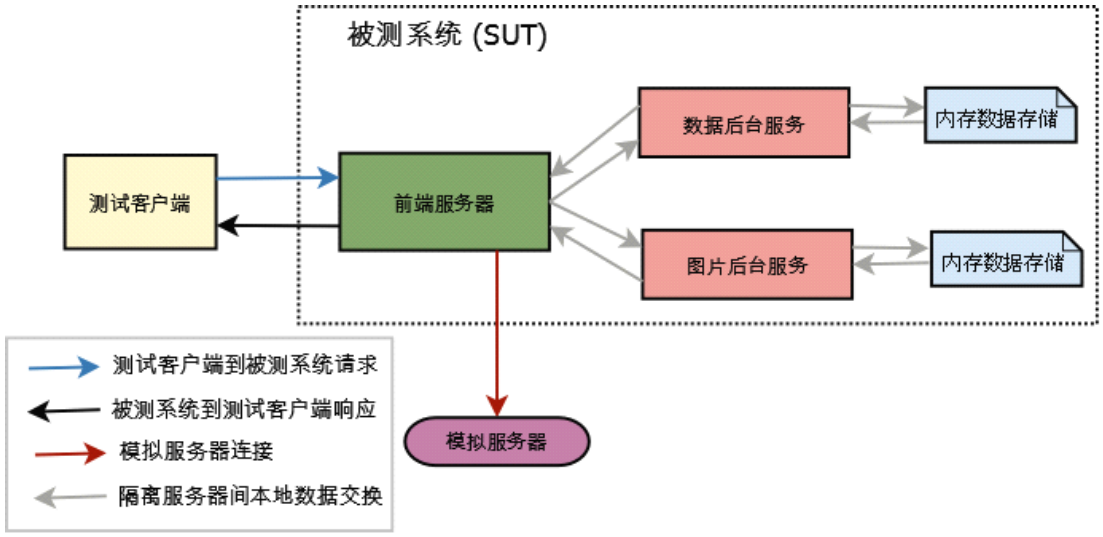
确保我们的测试不会访问到的连接点都具有相应的模拟对象，这样可以验证这些连接确实没被访问。

提供可以容易地向数据存储填充测试数据的功能单元。

提供日志记录模块，可以帮助跟踪请求/相应回路在被测系统中流转的过程。

在测试中使用隔离式服务器

我们看一下前面提到的那个被测系统。假设里面的所有服务器都是隔离式服务器，那我们的对同样一条用户请求的端到端测试就会是下面这个样子：



这个端到端测试执行了以下步骤：

在单独的一台机器上启动图中所示的整个被测系统

通过测试客户端向服务器发送请求

验证服务器响应

需要注意的是图中到模拟服务器的连接在这个测试中并不需要。如果被测试的请求需要连接这个后端服务，我们就需要在这个连接点也提供一个隔离式服务器。

由于没有用到网络连接，这个端到端测试就更加可靠了。因为它所需要的所有东西都在内存或者本

地硬盘存储，这个测试也会更加快速。我们在持续构建中执行这样的测试，让它们可以在每次提交影响到被测系统中任意一台服务器的情况下被运行。如果测试失败，就可以通过日志模块帮助在被测系统中跟踪失败出现的位置。

我们在大量的端到端测试中用到了隔离式服务器。一些基本的例子包括：

对使用了 Guice 框架的服务器进行启动测试，用来验证在启动过程中没有 Guice 的错误。

后台服务器的 API 测试

微型环境的基准性能测试

前台服务器的 UI 和 API 测试

结语

隔离式服务器还是存在局限性的。每次执行端到端测试的时候都启动整个被测系统，这回延长测试执行的时间。如果你的测试所用需资源是有限的，如内存和 CPU，隔离式服务器的使用可能会由于交互的复杂性而使得资源不足。使用内存数据存储也会让测试集的大小比生产环境数据存储小得多。

隔离式服务器是一个很好的测试工具。但像其他工具一样，需要在使用前好好思考它的适用性。

软件测试工程师指南

当你生活于网络时代，只要原地不动就很容易落伍了。

没有经验，不知如何跻身于测试工程师的行列？以下几个基本方向能使你从新手成为软件测试的行家里手。

软件工业是自动化工业的一部分。而且是最活跃发展最迅速的一个方面。到底有多迅速？任何人的想像力都不够！正如我们不会把我们的事务托付给不可靠的经纪，任何有分量的公司都不会采用没有质量保障的软件。软件测试人员，我是说有水平有经验的软件测试人员永远是供不应求的。软件测试经理不得不花很多的时间去面试有潜力的应聘者。一些应聘者在软件方面或者软件测试方面毫无实际经验，明知道软件测试工作是一个高回报的和最合适的软件工业入门，就是无法抓住一个又一个机会。这些人真正需要的是一个指南能告诉他们如何成为一个软件测试工程师。

首先，进入软件测试需要哪些技能？

1、软件工程技能你必须了解软件软件工程（设计、开发和简单测试），应用，系统，自动测试编程，及操作系统，数据库，网络系统和协议的设计和使用。

2、交流技巧如果想确定软件缺陷，你应当能够指出什么时候的缺陷算是缺陷。

3、组织技能如果你在别人都头脑发昏的时候保持清醒，你就可能是一个好的软件测试工程师。在网络时代软件测试是一项有压力的复杂性工作，但如果你能从这些纷繁中找到一种途径，它就是一项回报丰厚的事业。

4、实践技能当一个工作需要经验，而你又需要一个工作去丰富你的经验时该怎么办？这并不完全是一个两难的问题，你可能采用几种方式去获得实际经验。

5、态度除了技术水平，你需要理解和采取适当的态度去做软件测试。

6、必备特性

软件测试工程师除了技术，还要求具有否定性的创造力；探测技巧；总体理解产品的能力；用客户的眼光进行评估；怀疑的而不是敌意的态度；能经受得住坏消息而保持目标；拥抱新技术的热望等特征。

下面来详细的说一下这六项技能。

1、软件工程技能（Software Engineering Skills）

软件工程技能可以分成三大块：理解软件工程的规则，了解计算机编程和操作系统知识。

理解软件工程“规则”。有一种过时的眼光认为软件工程只是由一些在工作期限之前疯狂编程、靠着非凡的协调能力和超人般的咖啡消耗整夜不睡，不停地设计和测试程序的“专家”们组成的。这种现象确实存在，但你只有了解了软件开发的真正过程，才会是一个专业人员。

从哪开始呢？先到图书馆去走一走。你需要建立软件测试知识的软件工程基础。我的建议是阅 读 Roger Pressman 的软件工程：A Practitioner's Approach， fifth edition （职业入门，第五版，McGraw Hill，2000 年版）和 Glenford Myers 的 The Art of Software Testing（软件测试艺术，John Wiley & Sons， 1979 年版）。Pressman 的书是一个对软件工程原理的全面介绍。有很多关于软件技巧、项目管理、要求分析和软件设计等软件工程方面的好书，但 Pressman 对这些方面在一本书里作了介绍。Glenford Myers 不到二百页，1979 年发行，却是软件测试方面的圣经。Myers 定义及诠释的测试方法论已成为软件测试的基本模块。

Myers 还考查了软件测试中的经济（缺陷的代价）和心理学方面（测试的目标就是发现失误及不成功之处），以及主导软件开发和测试的基本原则。

对参考书进行基本研究是一个好的开端，但这只是单方对话。如果你能和上千个直接具有软件工程和测试经验的人以及想进入这一领域的人对话是不是再好不过了呢？感谢那些网络电子部落，你已经可以做到了。Comp.software-eng 覆盖了设计、编程、项目管理等软件工程的各个方面。Comp.software.testing 涵盖了软件测试的自动化、培训、技巧等方面。

等等，别只停留在这里！你是不是应当经常访问这些网址呢？Bug-Net 是有关软件缺陷的在线杂志。阅读有关缺陷的文章是学习如何工作及失败的极好方式。你也应当查阅软件测试及质量工程杂志。STQE 是确定网络软件测试资源很好的始发站。

计算机编程。不能想像有的人喜欢测试产品却从不阅读、检查和理解组成产品的软件一样。

不要误解我的意思。你不必花所有的时间去读源代码，但任何你做过的有关自己程序的设计、编写和纠错都能大大地有助于测试别人编写的程序。

你怎样学习编程？通过编程。可以严肃地说，开始学习写计算机程序是最简单的事。记住我说的是“开始学习”。软件编程环境，例如 Microsoft Windows Foundation Classes （MFC） or Sun's Java Foundation Classes （JFC， also called “Swing”）不断变得越来越复杂，越来越难跟得上。

但我在努力超越自己。你应当怎样学习编程呢？

首先，买 Microsoft Visual Basic。不要让名字骗了你。你能用这套组件建立相当复杂的程序。而且它只要一百元左右。下一步呢？等等，是 visual 编程警告的时候了！

现在你为你的 PC 买一个程序语言的时候，你其实是买了一个集成开发系统或称为 IDE。这些 IDE 通过对编程的简化把开发过程流水线化。这些 IDE 其实会帮你写很多编码。这非常有利于尽早开发出一个产品，却不利于你学习编程。如果你用 Windows 产生程序，你别无选择，因为环境介入太多使你无法从头编程。如果你从 Unix 系统产生程序，你能自己写所有的编码。

一旦你习惯了与参量、控制结构、对象、输入输出及更重要的 Visual Basic 纠错打交道的时候，你就可以开始学习 C 语言了。学习 C 能使你熟悉十六进制系统，通过指针分配和参考内存，存取个体位码及建立程序模块。

我总是认为在学 Java 之前最好先学会 C，因为 C 强迫你自己去完成许多任务而 Java 会自动处理（例如，释放未用的空间）。用 C 工作比 Java 难，但你能学到编程更多的基本方面。你其实能用 Visual C++ IDE 从头写 C 程序，但最好还是在 Unix 系统中学 C。

操作系统知识。你已经把它交给了在 Redmond， Washington 的那些人了。在短短的几年内，Windows NT 已经成为世界上大部分计算机的标准操作系统。如果你要用 NT 工作，你需要了解它的寄存地址。（它是一种用于存储你的系统结构的各个方面的数据库。）我发现 Peter Norton 写的 Inside Windows NT 4.0 （SAMS， 1998）是一本很好的介绍书。但是，如果你的应用或系统要求高的保密度、产出、可靠性及灵活性，Unix 依然是最好的选择。

如果你想成为一个成功的软件工程师，你必须能在 Unix 的世界里工作，如果你想从头学习编程，也要在 Unix 下进行。

你的选择是什么？你可以到当地的学校或大学学习课程，或者在家建立一个 Unix 系统。别昏过去了，你所需要的只是一台 PC 和一份能让你从网络免费下载的 Linux 拷贝。（你大约花二十九元能买一份在一个 CD-ROM 中带了所有文件的拷贝。）Linux 不是 Unix 的“玩具”版，它是真实的。它已经发行了七百万份拷贝，一些主要的 PC 生产商甚至先替你装载了它。

好了，你已经到了 Unix 或 Linux 系统了。你应当学些什么？文件和目录结构，标准输入输出和错误流，背景（background，也称为“daemon”）处理，从 C 调用系统功能，好，我可以接下去了。一个好的开端是读 Arnold Robbins 的 Unix in a Nutshell （O'Reilly & Associates， 1999）或者是 Ellen Siever 的 Linux in a Nutshell （O'Reilly & Associates，1999）。

2、交流技能（Communications Skills）

能写出计算机程序却写不出一个完整句子的软件工程师现在还有。但不幸的是，要成为一个成功的软件测试工程师，你需要清楚的交流。

你怎么去学习写？通过写。如果文字水平太粗糙，上一门创造性写作的课。每天写工程流水记录或发 email。关键是学习（或重新学习）怎样用清晰可懂的语言表达你的思想。一个好的写作参谋是 William Strunk Jr.和 E.B. White 写的 The Elements of Style（Allyn & Bacon， 2000），它一点也不象初中教科书。

测试工程师必须把产品测试的技术写成文件。测试计划提供指导并把测试设计转化为设置、实现测试和评估结果的步骤指导。具有一般软件和产品特性不同层次经验的工程师都能使用这样一个详细的测试计划。如此测试设计者或测试方案作者之外的工程师也能能进行测试。

测试计划也帮着佐证测试策略的正确性。项目中的每个人都应当参与审查（即市场、开发、支持、技术写作及测试人）。计划的审查是必不可少的，因为尽管测试工程师尽最大努力来达成一个对产品的全面定义，这一测试设计者所基于的定义不一定是完整或准确的。此外，就象开发者很难测试他们自己的编码一样，测试工程师也很难明确评估他们自己的测试计划。每一个计划审查者都可能根据其经验及

专长建议修改，有时候审查者还能提供测试工程师在组织产品定义时不具备的信息。例如，一个市场人

员可能了解到了新的客户要求，一个软件支持专家可能从有关的产品领域了解到了一个新的缺陷报告。

测试计划强调测试计划和执行的原则。在测试计划中描述进行测试所需的测试设计和步骤是另一层关于测试设计和计划的原则。在测试设计和计划中的错误与欠缺在设计转化成测试计划中特定的结构和测试步骤后就经常是再已无法弥补。

测试计划可作为其它项目，例如为不同的产品准备测试时的参考资料。当被测试软件找到缺陷解决并证实后，测试计划所述的测试可以用于证实缺陷的解决方案。同时，一个主要的测试设计信息来源，特别对于旧产品的新版本而言，是相关产品或前版本的测试计划。在建立新版本时，旧版本的软件测试计划都应当被重新审查。

与功能与设计说明不同，测试计划将从测试的角度来描述产品的功能操作。从这方面说，测试计划构成了公司公共档案的一部分。随着时间的流逝人们会离开公司，带走他们的知识。以前产品的测试计划就能帮助你定义新产品的测试。

软件测试工程师还要写测试结果报告。测试结果必须写成文档，这样就能确定被测软件的状态，提供关于必须要解决的缺陷的记录。产品测试中发现的所有缺陷的记录是测试部门最显眼、保存时间最长的文档。测试计划和测试报告在项目的最后常被遗忘，但现存缺陷的清单（或数据库）代表项目未完成的议程。这一议程没完成是因为一些缺陷必须在对原来产品的一个 patch 或 maintenance release 的时候纠正，或者它们在这个产品作为后续产品的基础之前被修复。

在与软件产品打交道的过程中，测试工程师比其他部门的人参与项目的更多方面。测试部门应当记录项目过程中重大事件（例如设计决定）的信息。这个信息应能帮助测试部门和其他部门避免在后续项目中犯同样的错误。错误是不可避免，在一个项目中可能出问题。从这些经验中学习就可能避免问题，避免今后的同样错误。从错误中学习的第一步就是记住它们，记忆的第一步就是把它们写下来。

3、组织技能（Organizational Skills）

每当执行一个软件项目的测试计划，几乎不可能不遇到至少会阻碍一些测试而必须解决的缺陷。一个测试工程师应当能灵活地停止测试产品的一部分而开始测试其他部分。有时被测软件需要做根本变动引起大量的测试结果失效，测试也许得重做不止一次。在问题被查找和改变在进行的过程中，测试工程师必须有条理，保持对执行测试的软件的前后关系的明确感受（例如目前被测试的程序特定版本的不同部分）。

网络时代要求的动态开发和测试模式使组织性的工作方式对测试工程师越来越重要。在整个开发过程中被测试软件可能会不断地改进。测试工程师在计划和实施测试的时候必须考虑这些变化因素，必须控制测试环境来保证测试结果的有效性。

记住计划是一个动词。作为一个软件工程师，你永远不会有你想要的所有时间和资源。你总是必须通过理解技术和产品，开发组织方式，从你和其他人的错误中学习，以及在设计必须改变和出问题的时候的迅速调整，使你的测试效果和效率最大化。如何能做到这点呢？基本代数：量化任务、目标和结果来减少方程中的变量数。把产品的功能定义成要求。在测试计划和测试中量化测试及其预期的和实际的结果，把信息提供给项目组。你东点一下西点一下是不能完成整个测试的。未来软件开发的组织模式要求有灵活的设计和不断进化的开发周期。对产品测试必须随着产品的进化而进化。

4、实践经验（Hands-On Experience）

这是个典型的两难问题。你需要软件测试经验来找工作，你没工作你就没经验。你该怎么办？

Be careful！

这需要勇气和你的 PC 的小心备份。

作为自愿者参与 beta 测试。怎样发现需要 beta 测试员的公司呢？首先，给你在软件公司工作的亲友打电话。偶尔有人会需要 beta 的测试人员。如果这不行，到你最喜欢的网络搜索引擎上去找“beta test”。你会发现很多小（和不那么小的）公司亟需 beta 测试员。为什么？这得感谢互联网，竞争的加剧使公司必须做出产品模型贴到他们的网址上作为“beta”版推出。这些公司希望人们不仅测试他们的产品，而且对这些免费品感兴趣进而购买他们的产品。

你也能参与开放资源的项目，例如 Mozilla，开放资源的网络浏览器是网络浏览器的基础。Mozilla 缺陷跟踪系统（bugzilla.mozilla.org）允许网上任何感兴趣的人直接

在 http：//65.54.244.250/cgi-bin/linkrd...2emozilla%2eorg 的开放资源项目中直接报告和跟踪缺陷

一句忠告：如果你要把很多 beta 软件下载到你家里的 PC 里，投资你的备份设备和防病毒组件。

5、态度（Attitude）

“我希望你幸福的梦想，被你打破了!”

我打赌这句话能勾起一些人童年记忆的创伤。我不是心理学家，但我还敢说这种说法是因为我们渴望看到成功。在软件测试中，你不仅要证实软件在做它该做的，还要证实它不会做它不该做的。为了做到这一点，你得找出软件的失败之处。

进行软件测试需要很多人的眼光要进行一百八十度的转变，因为测试的目标是要让被测软件失败，由此产生出等同于其他东西工作正确时的成功。在软件测试中，一个成功的测试揭示一个缺陷。进行软件测试也是因为互联网的来临要求人们用一种大不同以往的眼光来看待动态的开发和测试模型。

6、必备特性（Necessary Traits）

软件测试工程师除了技术，还要求具有否定性的创造力；探测技巧；总体理解产品的能力；用客户的眼光进行评估；怀疑的而不是敌意的态度；能经受得住坏消息而保持目标；拥抱新技术的热望等特征。

否定性的创造力。一个软件工程师不能怕引起一个产品的瘫痪或烧毁。在软件测试中，边界意味着被超越而不是被遵从。如果一个程序对某个值的极限为 10（例如，可以在一时间被打开的最大文件数），测试工程师的第一想法应当是“如果我把那个值取 11，或 0，或 10.1，甚至不设这个值会如何？”

在我的早期的工作生涯中，有一次我测试一个开发和 QA 工程师遗漏下来的 PC 数据库。有问题的数据库是 2.01 版。这本身就说明产品有问题。2.0 版没解决 1.0 版的所有缺陷吗？或者 2.0 版又加入了新的缺陷？很遗憾因为时间紧我没有调查这些，只是证实了最后的缺陷修复后就告捷了。

这是很大的错误。我应当重测开发人员所谓“没有变化”的所有产品功能。2.0 版本中的缺陷确实复修了，但在修复的过程中，有人破坏了请求。事实就是如此，在数据库里不能搜索数据了，第一个收到这项产品的 beta 客户发现了这个缺陷。

我宣布以前的测试无效，要求对产品进行全面测试。找到几个缺陷之后，我发现这个数据库读取写保护文件或写保护了的磁盘的时候就会引起瘫痪。开发人员很吃惊我会试着写保护一个数据库。他们的反应就是：“没人会这么干的!”产品的市场经理很快用他们的方式承认了错误。

探测技巧。在一个理想的世界中，软件测试应当在一个经常更新的写得很清楚的功能与设计说明文件（一般被称为“specifications”）中被完整而精确地描述。不幸的是，这一完善被开发程序每一方面文件的任务，包括记录在开发中对程序不可避免的改变，要花很多的时间和精力以至于人们无法完成编程。而且花费也太大。

正式与非正式的信息源：正式系统、要求文件、功能说明书、设计说明书、非正式系统、用户文件、与其他开发人员的交流、与软件支持人员的交流、有关产品的文件、有关产品的缺陷、从工作于相关或早期版本产品获得的“局部知识”。

因为我们不是在理想世界里编程，测试工程师应当能够自己找出工作的方式。典型的是，总会有一些设计和功能说明书让测试工程师用于开始他的研究。这些文件能看成为描述被测试软件的“正式”系统。测试工程师应当能用更广大的“非正式”系统的信息来扩展“正式”系统的信息。同时，在项目周期的任何一个点，任何文件都可能是正确或不正确的，所以测试工程师必须根据对软件工作模式的观察，与开发人员和其他项目人员的交谈，或对有关或看上去不那么相关文件的审核，来确定文件的精确性。

总体理解产品。在一个程序项目是，软件开发工程师主要把他们的精力和注意力集于自己的项目部分。结果当这些项目部分组合在一起进行测试的时候，就会碰到兼容性的问题。到产品寄给一个客户之前，唯一能见到整个产品的就是测试工程师。因此测试工程师必须能够对整个产品的操作与使用保持一种“系统”的眼光。

测试工程师对产品的任何一部分的操作可能不是最好的专家，但他必须是产品整体操作的专家。例如，如果被测的产品是一个类似于 Microsoft Office 的由文字处理、扩展页和其他有关程序组成的办公室自动组件，测试工程师必须了解每个程序的操作，各个程序之间的相互作用和客户其他的软件硬件和软件环境。

用客户的眼光进行评。测试工程师必须是客户的拥护者。被测程序有可能运行可靠满足所有的设计要求，但在客户的软件环境中未必能够用。产品被送到客户之前的测试之一就是要证实产品达到了客户

的要求与期望。在这项测试中，测试工程师必须模拟用户的软件环境，把自己放到他们的位置上。

关于软件功能“正确”而不能满足客户需要的一个悲剧性的例子就是美国航空公司 965 航班 1995 年在哥伦比亚卡利市的一次失事。在飞行着陆时，空中信号控制系统指示机组人员朝一个叫“Rozo”的航空信号灯飞。这个信号灯在航空图中标为 R。机组人员把 R 输入到飞行管理计算机中，看到了明显是由近到远列出的六个航空信号灯。机组人员选了第一个信号灯，以为这就是 Rozo。但那不是。自动驾

驶仪把飞机向左转了九十度，撞到了山上。

什么地方出错了呢？当航空表里把 Rozo 列为 R 的时候，飞行管理计算机要求机组人员输入信号灯的全名调出它的方位。同时，计算机只显示了信号灯的编码字母和方位。计算机功能“正确”，但不满足用户的需求。

要求变化。项目刚开始时的要求与最终项目完成时的要求一致的情况是极少见的。有时技术变化了，产品必须改变以适应于技术。有时竞争对手的产品具有你的产品所没有的功能。很多情况下，客户的或潜在客户的要求需要变化。这些因素合在一起的一个例子就是目前 Microsoft Internet Explorer 和 Netscape 的竞争。

随着计算机首次用户的迅速增加，今天的测试工程师比以往更需要把自己置于客户的位置上。这些新的非技术用户不愿意接受缺陷，对缺陷的解释或理性思考，或通过“升级”修正缺陷。他们只希望他们所买产品的软件和硬件都是能工作的。

怀疑的而不是敌意的态度。测试工程师不能按表面值接受事物，必须执着地对一切提出疑问直到被证实。工程师必须用一种与项目的其他的人合作精神来平衡这种怀疑性与执着性。测试部门与其有关部门的关系可能会变得紧张，特别是在大量缺陷被发现后，或者在每个找出的缺陷会潜在地延迟产品的发货时间而延迟了项目时。测试工程师应当记住要攻击程序的整体性，而不是程序员。

经受得住坏消息而保持目标的能力。一个测试工程师必须忠实地汇报产品中的缺陷。这一信息应当被项目组欢迎，因为每一个测试工程师遇到的问题（除非加入新的问题）都意味着减少客户会面临的问题。但不幸的是很多人不想听到有问，特别是在程序项目的后期。

测试工程师应当能处理因为工作做得太好而引起责备的情况。这对有些人来说是很难做到的，会严重地影响斗志与自尊。

看起来常常是测试工程师阻挠了向客户交货。客观的项目经理才能感觉到测试工程师是在对项目提供有价值的服务。我清楚地记得一个项目经理举起他的手求我他要的是：“解决方案，不是问题！”（他不明白解决方案的实现有时要求一个问题的解决。）有时项目经理在项目计划不方便的时候对于因为发现缺陷而打折是有压力的。在这些情况下，测试工程师应当能基于他对产品的经验和知识进行辩护，但他不应表现为象是他个人受到了威胁。

如何避免这些情形呢？就测试的内容、时间及如何更新测试结果和缺陷信息，设定其他项目组成员的期望。我曾经为一个希望延迟产品发送日期的 QA 经理工作过。他的目的不是为了产品成功，而是政治权力的操纵。他确信自己能被提升，把一些为他工作的工程师指定为“manager”，开始自称为“director”，还要大楼管理人员把他的办公隔间加宽一英尺。（这没有实现，但至少他的座位有了更多伸脚的余地。）

拥抱新技术的热望。对多数人来说，年龄越大越难学习。在商业世界里，人员越往公司的食物链高处走，越远离他们所建立的技术基础。这一部分是因为他们需要把精力集中于其他的经营和指导其下属的任务中。有时也是因为他们不幸地认为自己已不需要进行实践的技术工作了。互联网增加了技术变化的速度。不继续学习或跟着发展就无法做出商务与技术的决断。

从前的一个经理给我树立了如何对待新技术的榜样。我跟他工作的时候他年近六十，但他象新手一样地热心于学习新技术。他大量地获取信息，不断补充在网络服务器、防火墙、和 Perl 或 Expect 等新语言的知识。他还重视做 QA 或测试组织的工作。他的最初背景是软件开发和开发管理，但他并不认为做 QA 经理是在降低他的声望。他明白一个独立的测试或 QA 组所进行的完整测试能使开发经理的工作变得多简化。

正象我所说的，当你生活于网络时代，只要原地不动就很容易落伍了。相对于其他软件工程人员，软件测试工程师的知识面应该非常宽广，但最重要的品质应该是能够在第一时间内接受新技术。

由于公司之间的竞争日益集中在质量方面，所以公司对软件测试人员的需求量也越来越大，这一点，在北美尤为明显，这决定了软件测试行业的前景可喜，同时也为愿意不断进取、学习新技术的华人移民提供了广阔的就业空间，软件测试工程师的就业机会一直都是非常多的，最关键，要善于抓住机遇并肯付出努力，踏踏实实的学起来、做起来。

测试人员和开发人员相处的技巧

当测试人员证明了应用程序充满了 bug 时，她正在做一份令人满意的工作还是糟糕的工作？从一些开发人员的角度看，那是一份糟糕的工作。看上去很可笑，项目经理责备测试人员拖延了产品的发货期，开发人员（通常是开玩笑地）抱怨说“测试人员对程序太粗暴了”。很显然，没有比 bug 数量更能代表成功的测试了。以下是一些关于测试人员如何和开发人员建立成功的关系的技巧。

当我以作为一个软件测试人员开始我的职业生涯时，我就意识到在开发人员和测试人员之间正在进行的对抗。我根本没有花时间和精力就确信这种情况是非常普遍的。我收到了来自开发人员的各种不友好的回应，我认为所有的测试人员都在他们的职业生涯中经历过这些事。

从冷漠的耸肩到明显的敌意（有时会用同情的微笑掩饰），一个测试人员不得不忍受来自开发人员的许多态度。很难保持一个积极的态度。但是保持我们的优先的正直，并且向前推动高质量的项目是由我们自己决定的。

我从 Cem Kaner 的《Testing Computer Software》里挑出一句很好的话：“最好的测试人员不是那些发现最多 bug 的人，或使最多开发人员尴尬的人。最好的测试人员应该是能够使最多的 bug 得以修复的人。”原文：“The best tester is not the one who finds the most bugs or who embarrasses the most developers. The best tester is the one who gets the most bugs fixed.” －经典。

那么我们可以做什么呢？

热忱并且耐心（Be Cordial and Patient ）

作为一个测试人员，你或许发现使开发人员信服你发现的缺陷是非常困难的。通常，如果一个测试人员找到了一个 bug，程序员将准备 10 个理由。有时让开发人员接受他们的代码是有缺陷的（并且是其他的人发现的）这个事实是很困难的。

开发人员需要来自测试小组的支持，测试小组可以保证发现的新 bug 是值得关注的，健康的并且对于使产品更好是非常重要的。一个人性的方法是经常帮助测试人员更多的了解编程人员。相信我，不用多久，相同的一个人将站在你身边了并且笑着指出引起 bug 的错误。热忱将帮助开发人员对你的错误报告说“Yes”。这是重要的第一步。

处事老练（Be Diplomatic ）

试着巧妙地表述你的发现，并且不带任何责备地解释 bug。“我确信这是一个很小的 bug，你不用花多少时间就可以处理掉。到目前为止这还是一个不错的程序。”开发人员将会跳起来并且拥抱你的 bug。

用一种心理方法。有时表扬一下开发人员的工作。为什么大多数开发人员不喜欢我们的错误报告的原因非常简单：就是他们认为我们在诋毁他们的辛勤工作。有些测试人员只在出现问题的时候才和开发人员沟通。对于大多数开发人员而言，软件是他们自己的孩子，而你只是一个妨碍他们的外人。我告诉我的开发人员因为他们我才存在于公司，而且由于我的存在，他们的工作才得以继续。测试人员和开发人员之间的关系是一种共生及互惠的关系。

不要害怕尴尬（Don’t Embarrass ）

没有人喜欢被指出错误。这是人类的天性。试着解释修复那个特别的 bug 的需要胜于只是用庞大的 bug 报告向开发人员开火。一连串的缺陷不只会激怒开发人员，而且会使你的辛苦工作对他们来说是无用的。

正象一个人不可能独自测试完一个程序一样，开发人员也不能设计程序没有任何错误，而且在其他

事情发生之前，他们需要先了解清楚。有错误是预料之中的事，他们也是过程中的一个正常的部分。

你赢得了一些，你也失去了一些（You Win Some， You Lose Some ）

我知道有些测试人员尽可能将自己的错误报告强硬。他们甚至不听开发人员关于为什么不能修复一个错误和不能实现一个功能的解释。尝试一些可以让自己放松的方法。做到开发人员身边和他一起分析错误的优先级和严重程度。如果开发人员在其不愿变更的背后有一个合理有效的解释，试着理解他。只是确信了解了要在什么地方划定界限以保护你产品最终的质量。

谨慎一些（Be Cautious ）

外交手段和适应能力不能替代谨慎的需要。开发人员经常会找借口说因为他们没有意识到（或者你没有告诉他们）那个错误有多严重所以他们拒绝修复它。用足能够清楚展示风险和问题严重性的方法设计你的错误报告和测试文档。甚至更好的办法是召开一个会议并且向他们解释那些问题。

一个聪明的测试人员是在倾听和执行之间保持平衡的人。如果开发人员不能使你信服错误不应该被修复，那么你的责任就是使他信服要修复错误。

如何提高软件测试能力

软件测试对每一个项目都是十分重要的，下面是一位软件测试员的学习测试的相关经验

- 测试准备工作

在测试工作伊始，软件测试工程师应该搞清楚软件测试工作的目的是什么。如果你把这个问题提给项目经理，他往往会这样回答：“发现我们产品里面的所有 BUG，这就是你的工作目的”。作为一名软件测试新手，如何才能发现所有的 BUG ?如何开始测试工作?即便面对的是一个很小的软件项目，测试需要考虑的问题也是方方面面的，包括硬件环境、操作系统、产品的软件配置环境、产品相关的业务流程、用户的并发容量等等。该从何处下手呢?

- 向有经验的测试人员学习

如果你进入的是一家运作规范的软件公司，有独立的软件测试部门、规范的软件测试流程、软件测试技术有一定的积累，那么，恭喜你!你可以请求测试经理委派有经验的测试人员作为你工作上的业务导师，由他列出软件测试技术相关书籍目录、软件测试流程相关文档目录、产品业务相关的文档目录，在业务导师的指导下逐步熟悉软件测试的相关工作。其实，在很多运作规范的软件公司，已经把上述的师父带徒弟的方式固化到流程中。

如果你进入的是一个软件测试一片空白的软件企业，那么，也恭喜你!你可以在这里开创一片自己的软件测试事业，当然，前提是老板确实认识到软件测试的重要性，实实在在需要提高产品的质量。这时候，可以到国内的软件测试论坛和相关网站上寻找软件测试资源，这种情况下，自学能力和对技术的悟性就至关重要了。

- 阅读软件测试的相关书籍

现在，中文版的软件测试书籍越来越多，有的是国人自己写的，有的是翻译国外经典之作。可以到 http://www.chinapub.com/ 或者 http://www.dangdang.com/ 等网络购书的站点查找软件测试相关的书籍。目前，从国外引入的软件测试书籍有很多经典之作，但是，翻译成中文后，翻译质量对阅读效果有很大的影响。

- 走读缺陷跟踪库中的问题报告单

如果您所在的公司已经有软件缺陷跟踪库了，无论采用的是商用工具，如 ClearQuest 、TestDirecter 等工具，还是采用的 Bugzilla 、 Mantis 等开源工具，这都无关紧要，缺陷跟踪库中的缺陷报告单才是有价值的。缺陷跟踪库中的问题报告单是软件测试工程师工作绩效的集中体现，同时也是软件产品问题的集中体现。一般来说，缺陷报告单中最关键的几个部分包括：第一部分是发现缺陷的环境，包括软件环境、硬件环境等;第二部分是缺陷的基本描述;第三部分是开发人员对缺陷的解决方法。通过对上述缺陷报告单的三个部分作仔细分析，不知不觉你已经吸收了其他软件测试人员的工作经验，并掌握了软件产品常见的基本问题。这是迅速提高软件测试经验的好方法。

- 走读相关产品的历史测试用例

如果你所在的公司有测试用例管理系统，那么，走读相关产品的软件测试用例是迅速提高测试用例设计水平的一条捷径。走读测试用例也是有技巧的。测试用例写作一般会包括测试用例项和根据测试用

例项细化的测试用例，下面举例说明。 “ 测试用户登录的功能 ” 是一个测试项，该测试项的目

的是测试用户登录功能是否正确，是否能够完成正常的登录功能，是否能够对非法用户名和密码做异常处理等等。因此，根据该用例项，可以设计出若干个测试用例，大多数情况下，测试用例项和测试用例是一对多的关系。

通过走读测试用例项目，你可以掌握应该从哪些功能点着手未来的测试工作;通过走读软件测试用例，你可以了解如何根据被测试的功能点开展软件测试用例的设计工作，包括如何确定测试用例的输入、测试用例的操作步骤和测试用例的输出结果等。

总之，走读其他软件测试人员设计的优秀软件测试用例，是提高自身用例设计水平的好方法。

测试发展的下一个阶段——Impact and Visibility

进入测试行业不觉已经快六年了，而自己离 senior 还是有一定的距离。最近一年经历的很多，尤其是两次 reorg，换老板，换 feature 这些变动，没想到对自己的帮助相当的大。俗话说，人挪活，树挪死，还真是这么个道理。从进入公司起的三年，一直都是拿自己当作一个新人从而工作中束手束脚，没有表现出自己应有的实力与技术。而加入新成立的 team 以后，大家都是同一个起点，站在同一条起跑线上，因此可以自由的发挥，自己也像换了一个人似的。

进入测试行业以来的这几年，主要是 focus 在了测试和与测试相关的技术研究上，并没有考虑其他的东西太多。而我自己现在也算是全面掌握了各种测试技术与方法了，至少是有了相关的工作经验，比如手工测试，自动化测试，黑盒测试，白盒测试，可靠性测试，压力测试，性能测试，安全测试，代码覆盖率，等等。我一直觉得这些东西很重要，因为他们是基础，他们是每一个测试人员想往上发展都需要熟悉与了解的，而不应该人为地去把他们分割，对立起来。比如黑盒与白盒的对立，手工与自动化的对立，等等。但是，有了这些基础之后如何向上发展则又成了一个新问题。或者说，当你认为自己已经掌握了足够的测试经验与技术以后，如何获得自己 team，自己公司，甚至测试行业的认可？我发现我这些年忽略了这些问题，从而在一些方面吃了不小的亏，而现在是时候要重视他们了。

很多人认为我一定是一个 senior，其实并不是，我只是一个中级。我还只是在 senior 这个目标前进的道路上，而到达 senior 的那天我将会给自己树立一个新的目标或计划，那一天也将是我多年努力的一个终点，而一个新的起点也应该会开始的。话说回来，我认为我在通向 senior 的路上，技术已经不是一个主要问题了，而更多的是一些软实力，比如 drive 事情的能力，协作的能力，等等。当然，每个人都有每个人的优点和特点，每一个 senior 都会有一些鲜明的特点，而我自己也应该给自己树立起一些标签了。当然了，我现在主要处在探索阶段，并没有成熟的东西来分享，之所以写这篇文章出来，主要是我发现可能我的一些话，一些思路，总有可能某些网友会有相同的感悟或共鸣。我也想让中国的测试同行能够持续的看到我的各个发展阶段的情景。其实，我的各个发展阶段 focus 的东西不一样，而每一个阶段的总结，从我现在来看还是比较正确的。比如我现在看我两年前的文章的观点，还是 80%认可的，尤其是技术方面。

之所以罗嗦这么多，就是因为硬技术（测试，开发技术）已经基本告一段落，而软技术是很难描述的，很难说清，甚至我不知道我应不应该说，因为每一个人在软的方面都会有不同，甚至很大的不同。所以，我觉得技术阶段以后，每个人的路也许是会多样化的。那么我觉得我现在所作的就是充分利用自己对测试的深刻感悟以及丰富的经验，从而产生不同于其他人的 idea，为 team，项目组，甚至公司作出一些特殊的贡献，从而受到同事，team， 项目组，公司，甚至测试行业的认可。基本上来说，一个人不能只是自己觉得自己水平好，要想向上发展，受到别人的认可是必不可少的一步。所以，如果有些网友已经处于类似的情况了，就应该或被动，或主动地寻找机会得到别人的认可和重视。这里举几个例子：

1、接手一个 legacy feature， 以前的自动化需要三台 machine，运行 10 天完成一次测试。而经过我的 redesign 和重新实现以后，只需要 2 台 machine， 运行 5 个小时则可以达到相同的 coverage. 因此，director （管理 200 多人） 颁发了公司的杰出贡献奖。

2、发觉项目组（2000 人）有一个流程对员工和管理层都很不方便，自行开发一个 tool 去 automate 这个流程。起初在自己的小 team 里运行和使用，后来推广到大的 team （50 人），被 manager 所注意，往上推广到 director， 从而应用在 200 多人中，又被 director 推荐，因此多个 team 私下向我要这个 tool，目前可能至少应该有 500 人用这个 tool。最终受到另外 director 的重视，要求一个 principle 测试跟我合作把它做成一个 generic 的 tool。（一般来说，一个普通测试人员跟 principle 测试合作的机会还是很少的）

3、由于公司流程的改变，使得测试人员很难开展 code coverage 的测试。因此，实现了一套专门为

测试工程师使用的 code coverage system， 正在推广当中。

4、自己报的 bug 的数量，质量一直保持最高，从而受到其他 lead，manager 的注意，要求我给大家介绍经验。

我们项目的测试大头是 VP，下面是 director->manager->lead。我举这些例子想说明的就是，自己一定要有技术，有了技术以后也需要让大家知道，尤其是领导层知道。实力越强，责任也就越大，做的工作也应该更重要，visibility 也应该更高。更多，更高级的的管理层也应该知道你的名字，你的水平，你对 team 的影响力也应该越大。请测试人员千万不要只是守住自家的一亩三分地，你的视野需要开拓。

如何有效确定哪些是测试的重点

在实际软件产品的测试过程中测试团队经常面临的一个挑战是有限的测试时间，即测试人员必须在某个时间段之内完成所有的测试任务;按照传统的测试思路按部就班的执行每个测试用例将会面临各种问题。首先，可能无法完成预先计划的测试任务;其次，测试的效率比较低下，如无法快速地发现测试对象中存在的缺陷。此时，基于潜在的风险列表选择测试重点将是一个有效的测试策略。

基于潜在的风险列表选择测试重点的核心思想是针对测试对象的每个功能模块，根据罗列的潜在风险列表评估风险，然后将潜在的风险列表评估的数值相加得到该功能模块的风险级别;根据功能模块的不同风险级别确定其测试重点并选择合适的测试策略。具体步骤如下。

1. 测试人员检查风险列表中的每个风险，评估该风险对测试对象功能模块产生影响的可能性。

2. 将功能模块相关的风险的可能性相加得到每个功能模块的风险级别。

3. 按照风险级别排序测试优先级。

基于潜在风险列表选择测试重点

某通信产品在一轮测试中主要覆盖 4 个功能模块，为简单起见，分别以功能模块 1~功能模块 4 表示。其中潜在的风险列表如下。

(1)功能模块将会被用户频繁使用。

(2)功能模块提供的功能非常复杂。

(3)功能模块经常被修改或者升级。

(4)功能要求具有很高的实用性。

(5)功能要求保持一定的性能级别。

(6)功能的实现采用了新的开发工具和语言。

(7)功能模块具有众多接口。

(8)功能模块由经验欠缺的开发人员所开发。

(9)功能模块开发缺乏相关用户的充分介入。

(10)功能模块的开发由庞大的开发团队实施。

(11)完全是新的功能。

(12)功能模块的开发在极端时间压力下完成;

(13)对利益相关者而言是非常重要的功能;

(14)功能模块的历史版本中发现了很多的缺陷;

将测试对象中功能模块产生影响的可能性分为如下 4 个级别。

(1)不适用的(0)：该风险不适用于该模块或者功能。

(2)低的(1)：该风险偶尔会发生，但不经常。

(3)中等的(3)：该风险可能会发生，并且可能会影响该模块或者功能。

(4)高的(5)：该风险很可能发生，并且绝对会对模块或者功能产生影响。

表 1 所示为根据前面定义的风险列表和可能性分析该通信产品的 4 个功能模块之后得到的评估结果。

表 1 评估结果

潜在风险列表	功能模块 1	功能模块 2	功能模块 3	功能模块 4
（1）功能模块将会被用户频繁使用	5	0	3	5
（2）功能模块提供的功能非常复杂	5	5	1	3
（3）功能模块经常被修改或者升级	1	1	1	1
（4）功能要求具有很高的实用性	0	0	0	0
（5）功能要求保持一定的性能级别	3	0	1	3
（6）功能的实现采用了新的开发工具和语言	0	0	0	0
（7）功能模块具有众多的接口	0	0	1	3
（8）功能模块是由经验欠缺的开发人员开发	0	0	0	0

（9）功能模块的开发缺乏相关用户的充分介入	5	5	5	5
（10）功能的开发是由庞大的开发团队实施的	5	5	5	5
（11）完全是新的功能	5	0	0	5
（12）功能模块的开发在极端时间压力下完成	3	1	1	5
（13）对利益相关者而言是非常重要的功能	3	1	3	5
（14）功能模块的历史版本中发现了很多的缺陷	0	0	1	5
总的风险级别	35	17	22	44

读者可以根据实际需要将该表做成工具的形式，以提高风险评估的效率。

测试团队通过评估每个功能模块潜在风险列表中的风险，分别得到总的风险级别。测试人员可以根据功能模块总的风险级别采取相应的应对手段，并选择其测试重点。在本案例中，测试团队将根据总的风险级别，为该软件产品提供的 4 个功能模块进行基于风险的广度优先策略，并执行不同的强度测试。

另外，需要注意的是基于潜在风险列表选择测试重点，预先定义风险列表中的各个风险是顺利开展该活动的基础。潜在风险列表需要测试团队不断地积累和更新，以定义合适测试背景的风险列表。

试析软件测试的错觉及发展方向

摘要：市场对软件质量重要性的认识逐渐增强，但是由于还存在部分公司软件项目过程不规范，导致重视编码和轻视测试的现象。本文重点介绍代表性的认识错觉，并作了剖析和相应的解释。

关键词：软件测试

随着软件规模的不断扩大，软件设计的复杂程度不断提高，软件开发中出现错误或缺陷的机会越来越多，而市场对软件质量重要性的认识逐渐增强，所以，软件测试在软件项目实施过程中的重要性日益突出，但是由于还存在部分公司软件项目过程不规范，导致重视编码和轻视测试的现象，对于软件测试的重要性，测试方法和流程等还存在很多错误的认识。这进一步影响了软件测试活动的发展和真正提高软件测试质量。一下列举了几种有代表性的认识错觉，并作了相应的解释：

一、软件测试普遍被认为是在开发后开始执行

软件测试是一个系列过程活动，包括软件测试需求分析，测试计划设计，测试用例设计执行测试。因此，软件测试贯穿于软件项目的整个生命过程。在软件项目的每一个阶段都要进行不同目的和内容的测试活动，以保证各个阶段的正确性。软件测试的对象不仅仅是软件代码，还包括软件需求文档和设计文档。软件开发与软件测试应该是交进行的，例如，单元编码需要单元测试，模块组合阶段需要集成测试。如果等到软件编码结束后才进行测试，那么，测试的时间将会很短，测试的覆盖将很不全面，测试的效果也将大打折扣。更严重的是如果此时发现了软件需求阶段或概要设计阶段的错误，如果要修复该类错误，将会耗费大量的时间和人力。

二、软件测试人员承担发布后所有的质量问题

这种认识很打击测试人员的积极性。软件中的错误可能来自软件项目中的各个过程，软件测试只能确认软件存在错误，不能保证软件没有错误，因为从根本上讲，软件测试不可能发现全部的错误。从软件开发的角度看，软件的高质量不是软件测试人员测出来的，是靠软件生命周期的各个过程中设计出来的。出现软件错误不能简单地归结为某一个人的责任，有些错误的产生可能不是技术原因，可能来自混乱的项目管理。应该分析软件项目的各个过程，从过程改进方面寻找产生错误的原因和改进的措施。

三、公司和软件测试人员都不重视测试

软件工程学的发展和软件项目管理经验的提高，软件测试已经形成了一个独立的技术学科，演变成一个具有巨大市场需求的行业。软件测试技术不断更新和完善，新工具，新流程，新测试设计方法都在不断更新，需要掌握和学习很多测试知识。所以，具有编程经验的程序员不一定是一名优秀的测试工程师。软件测试包括测试技术和管理两个方面，完全掌握这两个方面的内容，需要很多测试实践经验和不断学习精神。这是不重视软件测试的表现，也是软件项目过程管理混乱的表现，必然会降低软件测试的质量。一个软件项目的顺利实现需要有合理的项目进度计划，其中包括合理的测试计划对项目实施过程中任何问题，都要有风险分析和相应的对策，不要因为开发进度的延期而简单的缩短测试时间，人力和资源。

四、程序员认为测试与我无关

开发和测试是相辅相成的过程，需要软件测试人员，程序员和系统分析师等保持密切的联系，需要更多的交流和协调，以便提高测试效率。另外，对于单元测试主要应该由程序员完成，必要时测试人员可以帮助设计测试用例。

五、软件测试不可能成为软件测试高手

项目的成功往往靠个别全能程序员决定，他们负责总体设计和程序详细设计，认为软件开发就是编写代码。因此，在这种环境下，软件测试很不受重视，软件测试人员的地位和待遇自然就很低了，甚至软件测试变得可有可无。随着市场对软件质量的不断提高，软件测试将变得越来越重要，相应的软件测试人员的地位和待遇将会逐渐提高。在微软等软件过程比较规范的大公司，软件测试人员的数量和待遇与程序员没有多大差别，优秀测试人员的待遇甚至程序员还要高。

由于软件系统越来越精密，越来越复杂，软件及系统的质量测试正在成为 IT 行业中一个新亮点，软件测试行业未来发展前景良好。对软件测试工作的发展方向有以下几点分析：

1、BUG 预防和早期检测，因为现在把重点放在产品交付的质量上来，预防实践和静态分析仪这样的检测工具将成为主流。

2、仿真工具变的很普通，使得仿造计算机环境变得容易起来。在开发过程的早期就可以进行意外和错误流程的测试。代码稳定后，在用真实环境验证仿真是否准确无误。庞大的测试用例管理系统将成为昔日的东西，大量的测试用例生成了却没有使用。

3、有用的方法，比如需求覆盖，模型覆盖，代码覆盖将驱动项目开发。机器将代替测试人员做大部分他们以往创建测试所做的繁琐工作，测试小组需要比以往更少的测试人员，留下来的测试人员将是经过更多高度培训过的。

4、测试人员的角色更换，测试中界限模糊，在测试领域工作使得专职测试的人员和专职创建测试工具的人员界限模糊，一个既是通过程序破坏事物的测试员又是创建程序用于破坏事物的程序员的专业出现了，关于如何称呼这个新的专业，新闻圈内的人们在进行着无休止的争论。测试于开发界限模糊，测试人员与开发人员一前一后，共同创造可测试的，高质量的代码，测试人员帮助开发人员消除需求中的问题，使得开发人员的工作更易完成，同时，开发人员写出更清晰，可测性更高的代码，使得测试人员的工作更易完成。

5、顾客反馈与测试合为一体，交付的产品质量更高。测试人员进行根本原因的分析，我们会问比如我们怎么会遗漏了这个 BUG 呢？或者我们将来如何防止这类 BUG？这些问题，我们的工作就是使顾客满意。复杂的相互关联的计算机世界使得了测试安全这一类新的问题让测试人员不断努力工作，但这没有关系因为这些挑战使测试人员精力充沛。

6、测试人员获得尊重测试人员将不再是在最后时刻才被叫来对产品狂轰烂炸，他们将在整个软件开发过程中提供一个可见的，重要的，增值服务。人们意识到，测试是有益的，有趣的甚至富有乐趣。软件测试人员开始扬眉吐气，而且，由于破坏事物至少可以带来创建事物一样的乐趣，人们开始在开发和测试角色之间转换，所有的人将学到更多如何得到良好代码的知识。

泽众软件工具使用技术支持

电话：021-61079698

Email：sales@spasvo.com

QQ：1404189128

MSN：spasvo_support@hotmail.com

产品租用		
下载	在线申请	详细

AutoRunner 是一款自动化测试工具。AutoRunner 可以用来执行重复的手工测试。主要用于：功能测试、回归测试的自动化。它采用数据驱动和参数化的理念，通过录制用户对被测系统的操作，生成自动化脚本，然后让计算机执行自动化脚本，达到提高测试效率，降低人工测试成本。

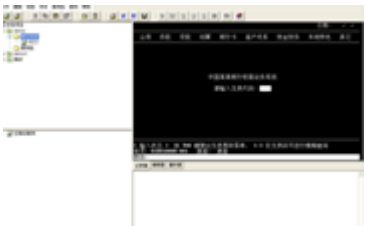
在线体验		产品租用	
企业版	免费版	在线申请	详情

TestCenter 是一款功能强大的测试管理工具，它实现了：测试需求管理、测试用例管理、测试业务组件管理、测试计划管理、测试执行、测试结果日志察看、测试结果分析、缺陷管理，并且支持测试需求和测试用例之间的关联关系，可以通过测试需求索引测试用例。

其他测试工具

Precise Project Management

Terminal AutoRunner



PerformanceRunner

有关培训、产品购买及试用授权方法等事宜

电话：021-61079698

Email： sales@spasvo.com

QQ：1404189128

MSN： jennyding0829@hotmail.com