泽众软件测试网
电子期刊

SPASVO

第二十六期
2014年2月

任用当地的教授：软件测试行业和学术界需要合作

程序员最艰巨的十大任务

做测试最高的境界是什么？

软件测试迷航：高端人才不只是找BUG高手

Google是如何做测试的

缺陷测试中容易忽略掉的几个问题

我认为的最佳职业生涯建议

从圈复杂度谈谈代码质量

上海泽众软件电子期刊

2014 年 2 月 第二十六期

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
2014 年 2 月 第二十六期 电子杂志期刊

任用当地的教授：软件测试行业和学术界需要合作

哲学博士 Patricia A. McQuaid，是美国加州州立理工大学的一名信息系统教授。

在她的职业生涯中一直于商学院和工程学院授课并在银行及制造业工作。她的研究方向包括：软件测试、软件工程管理、软件质量及软件过程改进。

她是美国软件测试资格审查委员会（简称 ASTQB）的共同创始人和主席。她是美洲，2000年在日本举办的第二界及2005年在德国举办的第三界世界软件质量大会的负责人。 她是2008年9月在美国华盛顿举办的第四届世界软件质量大会的副负责人。

她拥有计算机科学和工程的博士学位，商业硕士学位，会计本科学士学位。她是一名国际信息系统审计师（简称 CISA）。她是国际认证软件初级测试工程师。

Patricia 是电气电子工程师学会 IEEE 的一名成员，是美国质量学会（ASQ）的高级会员，是软件测试质量专业期刊的副编辑，她还加入了美国质量学会的软件分支委员会。

她是软件质量工程师（ASQ 质量出版社）书籍第一卷和第二卷的基本概念的投稿作者，也是 ASQ 质量工程手册（ASQ 质量出版社）的作者之一。

我常常在会议上听测试行业内部的人说：“现在啊，刚毕业的大学生都不具备我们所需要的技能……”这话没错，可我倒是有一个提议可以稍微缓解一下这个实际上很简单的问题。简单来说就是，为什么不帮这些大学生一把，或者接受当地的学术界专家教授呢？

一个大学和企业都不得不面对的难题就是预算限制（尤其在现今的经济状况之下）。就像许多企业已经大大减少了他们的可支配支出，不少大学也是，或许大学减少的还要多些。众所周知，专业会议的价格是相当昂贵的，你的专业领域的专业培训课程也是。现在，在计算机科学和其他信息技术相关的领域里，我们需要了解的知识主体是处于不断变化中的，而且很快就会过时。

现在所用的技术——硬件和软件，在我们许多专家还在研究生院时都没发明出来。所以，我们现在课上所教的基本都是我们自学的。如果那时够幸运的话，我们或许能够参加培训班去学习我们现在所教授的内容。但更可能的是我们不得不自学。

所以，努力工作的员工怎样才能得到他们所需要的培训，将来成为软件测试/软件工程的专家呢？——通过正式或非正式地让企业和大学建立更多的合作关系。

非正式联盟

企业花费大量经费来培养他们的员工，花钱让他们获得正规的大学教育，派他们参加相关会议和专业发展研讨会。许多这类研讨会大多是在线的，要么请了咨询顾问来上培训课要么由公司自己内部的员工举办。通常顾问讲师是按学员个数来要价，但也有时按一定范围的人数要价，比如：20到25人一个固

定价。但愿这是我最后一次这么说了。下次你们邀请当地教授免费来帮你们上在线培训课，怎么样？

怎样才能更好的确保你们所招顾问讲师所教的东西就是你们希望你们将来的员工能学会的？这得花费多少经费？另一套培训资料和一顿午餐吗？或者如果他们确实是按每个人来收费，那这样，你还要额外花这么多钱去培养一个专家还值不值得？或许吧.....

正式联盟

建立一个更加正式的联盟是另一个的选择，它或许对参与其中的每个人都更具价值。这种正式联盟有多种形式，包括：提供资金让学校教授去参与测试会议或课程；让公司员工在大学里做讲座嘉宾；提供机会让教职员工和学生来公司参观；让学校的教职员工来公司工作一段时间。给学校的教授们提供可以在课上用得着的材料和例子也是极具价值的。

我的一个难题是：找出对学生既有意义又有挑战的现实生活中的实例与练习。你们可以把任何有关你们公司行业机密的东西留着，只提供可以融入课堂的知识。

许多公司已经制定了正式的教授计划，如：设立一个持续几个月或者几年的客座教授的职位。我很幸运，在几年前的一个夏天加入了这样一个联盟，作为一个大企业的学院合作计划的一部分，我在他们的一个重要的软件测试实验室工作了五个礼拜。那次经历是相当珍贵的。在学校和企业都受益的同时，我计划着在学校开一门新的软件测试的课程。

作为开新课的一部分准备，我想学习当前的实例并熟悉一些现在大家使用的自动化测试工具；我想了解成为一名全职测试员是怎样的：测试员会面对哪些挑战？他们用哪些自动化工具？这些工具的优劣点是什么？成为一位测试队伍的项目经理又是怎样的？我想知道企业是如何在软件测试的业务流程中整合工具的；我想加强我们学校和测试企业的合作关系。

我也可以看书来了解这些，但是我想知道更多，想提高我对这方面的理解，想把这些知识带到课堂。公司企业的目标是从长远角度积极影响招聘员工。更明确地来说，他们在向我们展示他们为其公司内部需求和客户需求所设计的测试软件的同时，也想要加强和学校的联系。因为他们以及许多其他公司都为了该如何招到有专业测试背景和把软件测试作为一个职业选择的学生烦恼着。

我被任用了

作为他们院校合作计划的一部分，我有幸能够采访和近距离观察测试员的日常工作，可以和软件测试项目经理探讨问题。

其中一个不错的经历就是我花了几个小时向他们的可用性专家学习了可用性测试。一个意外的收获是：他在他们可用性实验室为我录制了在一个网页上运行可用性测试的过程。我利用录制的内容给学生阐释说明了一个正规的可用性测试的流程。我也参加了几次培训课学习了两家供应商的测试软件。

这次联盟的一个结果是我能够获得我为了开软件测试课程所需要的准备工作。除了我接受的技术培训，我还对现下业务测试行业的难题与挑战有了更好的了解。下个学期，我就开办了软件测试课程。班级的学生有来自商业学院信息系统专业的，还有来自工程学院计算机科学专业的。他们团队合作，我会确保一队中有两名学生是信息系统专业（商业）的，另外两名学生是计算机科学专业（工程）的。这样来自两所学院的学生就不得不合作了。

通过加强和我们学校的合作并且通过让学生更好地了解了测试相关的职业，而公司的招聘成功率随之上升，公司就从中获益。

这样，学生会把软件测试作为一个职业选择。潜在员工变得更适合企业的需求：学生获得暑期实习的机会，毕业后立即成为该公司的正式员工。

我现在有新的合作伙伴了

当我在软件公司时，一位经理联系了自动化软件测试工具的一家供应商的 CEO，恳请他们为我校捐赠测试工具。这样子，其中的每一方都能受益：学校可以收到最好的测试工具来用到课上教学；公司可以雇用对自动化测试工具有经验的新人；自动化测试工具的供应商可以把他们的软件送到将来的软件测试决策者们的手上。同时，软件公司开始以我们学校为成功案例和其他高校建立正式项目。

与这家公司合作了差不多一年的时候，他们宣布了新的计划：提供软件和培训材料俩帮助学术机构发展他们的技术课程。

我们学校是第一个参与这个合作计划并收到捐赠的机构。在课上，我使用他们的网页应用测试软件，网页下载测试软件和管理需求、制定测试计、划追踪缺陷的产品。

其他公司也给我们学校捐赠了软件（和硬件），供我们上课使用。它的意义是许多公司都开始十分慷慨地捐赠他们的产品给学校供上课使用了，但也许只有我们提出时他们才会这么做。

挑战

这是一项巨大的任务，但只有我一个人在做着。所以，如果软件公司愿意给一所学校捐赠软件，他们必须有提供额外培训和支持的准备。不然的话，这项计划就会破产。

一所学校里多个不同的学科的人都加入的话，这项计划成功的几率就大些，但这样的机会也不是经常有的。但确实有来自其他学科（如：商业信息系统和计算机科学）的教职员工参与也是很吸引人的一点。

为了成功开设一门软件测试课并把软件与课堂整合得花上不少时间。至少以我自己的例子来说，我除了平常的教学任务就一心一意专门做这个。不必说，我很忙。所以，有什么能大大地提高成功的几率呢？——让企业对与当地教授合作产生兴趣并在大学的 release time 提供资金。这是什么意思呢？就是说，这个教授一学期教两门课，这笔资金就提供给大学来支付教授教一门课的工资，这样的话，他们就可以少教一门课。

我最需要的就是时间。在一些大学，如果正处于资金短缺时期，那么教学工作量（每学期每个教授要教的课程的数量）就会增加。那样，我们教课比平常要多，就没时间花在软件测试课的事情上了。

但是这样做会耗费企业的经费吗？当然会。但是，为了让你能够招到合格的大学毕业生，做什么事是值得的呢？当雇佣到优秀的员工时，想想公司花在培训上的所有资金吧，那就是答案。

另一个选择就是企业给教授发工资，事实上一付就要长达一年。但或许你更愿意一直不断吸引更换新的教授来上课而不提供资金，这样的话你或许不会像你提供稳定的资金那么成功。

话虽如此，一些教授倒愿意作为客座教授，或许离开校园，在企业公司里工作个一、两年。

一个永久的选择是：建一个由企业提供资金的讲座职位。这个职位或许是比较耗费资金的，但可能是更成功的。在这种情况下，一家公司要长达一年投入大笔资金和精力在这个职位上。

如果你们学校设有硕士、博士学位，那么学校的研究生们就能和公司测试员工一起做项目。反之，这些项目也可以成为他们的硕士、博士论文的实例。

总结：

显然，通过更近一步的合作，测试行业和学术界都可以获益。想要了解你们当地学校的教职员工的更好的方法是什么？让测试行业和学术界结成联盟的更好的方法是什么？帮助你们招到符合要求的学生的更好的方法是什么？——一大批当地学校的教授求任用！

程序员最艰巨的十大任务

程序员最艰巨的任务跟编写代码没有多少关系。编码是逻辑思路的一种实践，这跟程序员日常工作中的其它任务比起来相对简单。如果你认为自己还是一个水平一般的程序员，在你真正的能进入到高手行列前，请确保你已经克服了下列晋级的障碍。

1. 解释你在干什么

解释软件开发过程是一个很困难的事情。那些非程序员职业的人也许知道很多关于编程的事情，但很显然，他们不会编程。对于他们来说，我们的生活就是在一间黑暗的屋子里趴在键盘前消耗着咖啡。

你会在你的朋友、家人和同事中遇到这样的人，他会认为编码不是一个正确的职业。

2. 形象的说出软件解决方案

根据一些简短的需求——通常是一知半解的，你需要设计出数据结构，软件架构，代码算法，通信协议，以及其它所有针对商业问题的解决方案各种组成部分。然后你需要用一种外行人听的懂的术语将它们表达出来，并需要在规定的时间里提交给客户。

很少有程序员能做好这些。

3. 评估工期

这是程序员痛苦的根源。在开发任务没有完成之前，你是绝对没有可能确定完成这个任务需要的时间。也许程序跟以前写的很相似，但环境变了，问题变了，限制条件变了。

经验会提供一定的判断力，但大部分的程序员都习惯于低估问题难度。这其中的原因是他们只考虑编码方面的因素，而忽略了这个任务清单上的其它事务。

4. 维护他人的代码

针对一个问题可能会有一万种解决方案，一万种写法。接手别人写的代码，意味着你要花无数的时间在成千上万的代码行里探索，理解当初作者的思路。而且，如果是一个不相信注释和文档的程序员留下的半个项目，麻烦就更大了。

5. 软件边界的模糊蔓延和让人吐血的奇怪功能需求

虽然敏捷开发方法给软件范围的膨胀提供了一定的预备空间，但这并没有起到任何的作用——尤其是当你遇到一些由一时兴起的怪念头产生的功能需求。你知道这样做必定会失败。你的团队知道这样做必定会失败。但客户觉得很好，而当失败不可避免的出现时，全是你的错，因为是你没有理解他们的真实意图。

6. 在缺少优化和过度优化之间找到平衡点

复杂的软件永远不会做到完美;总会有一些更好的方案。你完全可以没完没了的优化下去，这就是为什么软件项目从来都没有提前完工的。

而另一面，“这样就行了——我以后会优化它的”这种心态也是常见的。代码今天好用，但你知道明天可能会出现麻烦或不能用。当然了，你是不需要去修改它的，它将会留给下一个倒霉蛋程序员。

7. 测试你的代码

单元测试你也写了，软件也提交了测试组，但 bug 依旧存在…

软件是复杂的，可能包含成千上万行代码。系统中可能存在百万的各种交互和逻辑路径;你不可能完全测试它们。

类似的，软件会在不同的条件下跟不同的平台上的不同的软件交互。你不可能所有的都测到。

写出好的单元测试是一种枯燥且辛苦的工作。理想情况下，测试应该在着手开发前就已经写好——但你如何向客户解释为什么四个星期过去了仍然没有可用的软件?

单元测试并不能覆盖每个问题点。在理想的世界里，应该有一个独立的团队来写测试并积极的去发现问题。不幸的是，对大多数项目来说，这样成本太高，时间不够，于是用开发团队来写测试程序。而开发团队潜意识的会避免很多极端的边界情况。

程序员喜欢用符合逻辑的方式处理所有问题。但用户很少是这样的。他们会发现你永远意想不到的问题。

8. 写软件文档

给代码写文档是一项费力耗时的工作。很少有程序员擅长这个、喜欢这个的，并且很少有程序员会花时间去读它们。

9. 处理 IT 问题

你每天都在研究技术。你也许是一个 HTML 或 PHP 程序员，但你很可能会遇到一些例如硬盘损坏、驱动冲突或软件崩溃的问题。解决这些事情不是你的主要责任，但是，除非你解决了这些问题，否者你将无法继续你的开发工作。

不幸的是，对于 IT 圈外的人来说，程序员应该是软硬件都精通的人。当他们遇到了问题，他们自己不花时间就解决，直接会找你。不论是遇到什么问题：你是用计算机的，你一定知道如何将预算表导入 Sage，如何配置 Oracle，或为何在他们的黑莓手机上发不出邮件。

当然了，这些打搅绝对不能成为你完不成工作的理由，也没有报酬，不是吗?

10. 处理人的问题

上面的这些难题都可以总结为“人的问题”。很少有外行人会去建议一个飞行员如何开飞机或建议一个电器工程师如何布线。但很多人却会兴致勃勃的勇敢的建议如何开发软件。

我相信对于这些人没有什么好办法。你需要接受这样的事实：这世界上有一半的智力是低于平均水平的!

做测试最高的境界是什么？

今天参加了单位组织的测试架构师培训。外部的咨询老师问了我一个问题：做测试最高境界是什么？我当时给出的仓促回答是：“帮助你所在的组织改善树立正确的质量观念。

帮助所在建立起有效预防和发现 bug 的流程体系与技术栈。”限于时间，这个问题没有展开深入探讨，却让我总是忍不住去想它。做测试最高的境界是什么呢？仔细思索后，发现我这个答案还挺不错的。

最高境界代表着极致，极致就是在已经达到非常好的时候还在不断的追寻：我还能做得更好么？如果持续这样做，在某个时候，一定会产生质的飞跃，让你跳出原有的框框得到柳暗花明的答案。这样的例子比比皆是：在人们找到钨丝代替爱迪生的竹炭纤维作为白炽灯的灯丝以后，后续的工程师做了上千倍爱迪生的努力，也无法让白炽灯的光电转换比有效提高了，直到节能灯泡的出现，换了思路，能效提高了一个数量级。

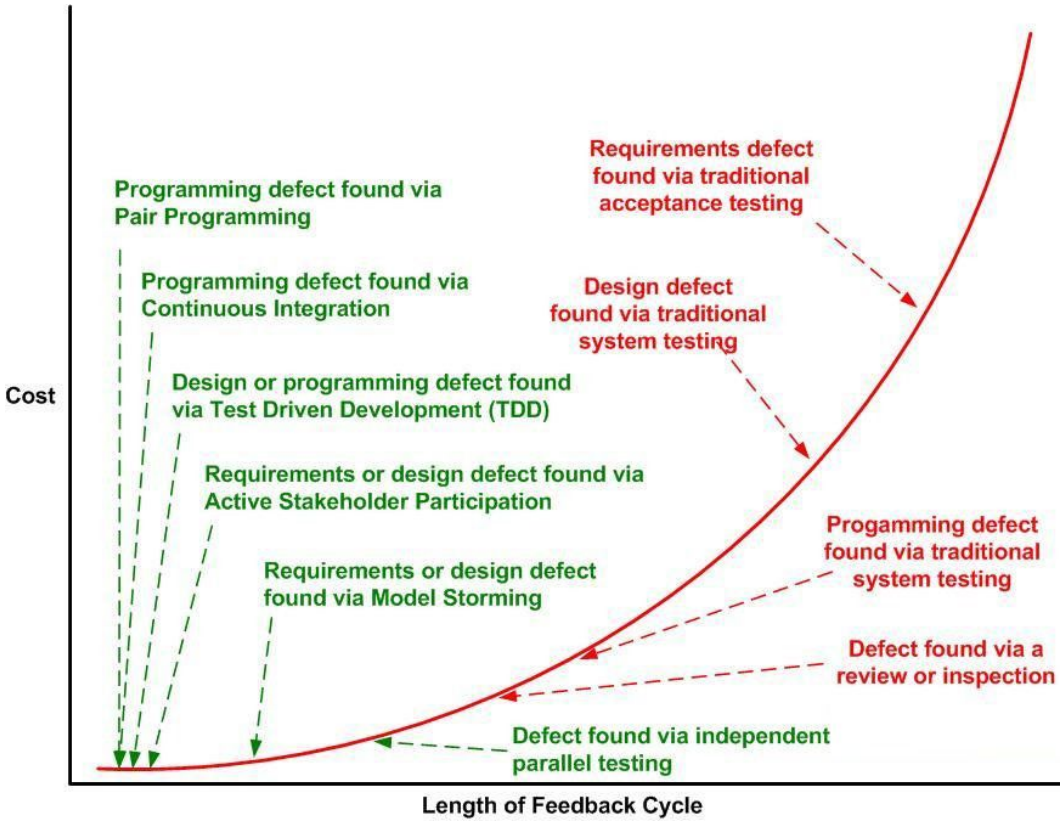
人们对测试最初的定义就是发现缺陷。大家都会熟悉下面图中的这条曲线：缺陷发现的越晚，修复的代价就越大。照这个思路来看，发现缺陷越早越好？什么叫早呢？如果是一般的软件生命周期，那能在需求阶段就发现所有缺陷就 perfect 了。这显然是天方夜谭，代码还没写呢，你怎么能发现编码过程中产生的缺陷呢? 缺陷产生于软件生命周期的各个阶段，试想如果在每个阶段的初期就能发现这个阶段的所有缺陷也是不错的，但这也是不可能的，再拿编码过程产生的缺陷来说，当一个方法还没有被编写出来，你如何发现它的缺陷呢？解决这个问题有 2 个思路：

1.阶段细分，每个阶段的工作再细分（敏捷的思路：工作细分，如 scrum 模型，它让每个 sprint 都能实现可以被测试的特性）

2. 换个思路，如何能够做到比检验做到更好呢？那就是预防。基于这个理论就有了各种 checklist，各种入口评审（相对于本阶段是检验，相对于下阶段是预防），测试驱动开发严格意义上也算一种预防的手段。

做到这两点的本质就是：帮助所在建立起有效预防和发现 bug 的流程体系与技术栈。

而这不仅仅是靠测试人员本身的力量能够做到的了，要靠团队所有人的共同努力才能做到，做测试的同学应该作为推动者和实践者，把质量意识灌输到每个团队成员的意识甚至是潜意识中去（注意：灌输意识不仅仅是测试同学的责任!）



最后简要回顾一个小故事：扁鹊是古代名医，但当人们问到他的时候，他却认为给人们做保健的大哥是神医。但在别人的眼里他的大哥只是个普通的郎中。

最后一句话其实不会发生在软件测试人员身上。你能做到这一点，或者推动做到。就会被大家普遍认可：）IT 届的意识还是很先进的。

软件测试迷航：高端人才不只是找 BUG 高手

一边是人才缺口几十万，测试工程师招聘、培训红红火火；一边是看不清职业发展、“成功没我份，失败全我错”的测试岗位从业者，软件测试工程师该何去何从？

一边是软件开发、测试谨小慎微，一边是系统宕机、瘫痪事故频发。供应商、用户、第三方应如何做好测试管理，提升软件质量？

中国软件测试市场究竟有多大？互相矛盾的现象让人难以看明白。

借助对1066家企业 IT 应用质量的一手调查，对10位业内人士的独家采访，我们来分析如何细化测试岗位和行业的专业分工、提升复合的测试技能、化解测试与开发亦敌亦友难题、消除测试考核评估的掣肘……

让我们一层层揭开软件测试的迷局。

2010 年世界杯足球赛期间，Twitter 的多次大规模宕机事件让用户无法忍受；2007年，奥运票务系统因无法承受瞬间每小时800万次的流量而宕 机；2006年，英国伦敦希思罗机场航站楼因应用缺陷致行李处理系统故障，积压行李达万件；近期，国内某银行核心业务系统发生故障，导致该银行包括柜台、 网银、ATM 机在内的所有渠道的业务停止4.5小时……

重开发、轻测试，让软件系统故障频发。

为什么经过测试的软件系统还是会出现问题？测试与业务有着怎样错综复杂的关系？软件测试的瓶颈究竟是什么？国内软件测试将呈现怎样的发展趋势？

借助中国测试平台网对1066家企业的一手调查，以及中国计算机报记者对10位业内人士的独家采访，让我们层层揭开软件测试的迷局。

测试软件不能承受之轻

捷克作家米兰·昆德拉告诉我们，生命中有太多事情看似轻如鸿毛，却让人难以承受。在 IT 应用中，软件测试就是如此。

2007 年10月30日，奥运票务系统因无法承受瞬间达到每小时800万次的流量而宕机，这也许是美国票务系统提供商史上最没面子的时刻。这家公司是2004年雅 典奥运会票务独家供应商，其系统技术已经经过市场的考验。据说在2008年奥运会的票务系统中，他们已经提高了峰值流量的预设值，可是没想到还是估计不 足，才出了大问题。

2009年11月22日，eBay 网站长时间宕机，造成卖家蒙受相当于当天销售额80%的损失。已经不止一次的宕机事件让 eBayCEO 脸面无光，不得不对 eBay 的系统负载能力重视起来。

另据业内人士透露，拥有600多家分店、18家配送中心，每天向全球180万客户提供种类繁多的设备维护、修理和运作产品的工业品分销商固安捷(Grainger)曾在 SAP 系统实施

过程中，由于系统功能性故障损失了2300万美元，使其无法完成当季收入指标。

“测试时候不把好关，后期上线后应用就会出现大的问题。”清华大学教授、中国软件行业协会系统与软件过程改进分会常务副会长郑人杰在接受中国计算机报记者采访时表示，近20年来他一直在关注软件质量的问题。

通过剪报等方式，郑人杰收集了这方面的诸多新闻素材，其中包括：上世纪90年代海湾战争时期美国部队由于导弹系统故障炸了自己军营；2003年美国出 现史上最严重的大面积停电，影响1/4国土面积的居民；2008年英国航站楼系统故障，导致15000件行李积压；2006年 ATM 机故障，造成轰动一时 的许霆案；近期不断出现的 ATM“双倍吐钱”等问题还不断出现。

相信这些系统上线前是经过了测试的。那么，为什么经过测试的系统还是会出现问题呢？

“业务需求不清楚，以及现有软件测试能力有限，导致了上述问题的发生。”中国惠普企业业务集团软件及解决方案部技术总监于志伟在接受记者采访时表示，软件测试已经不只是一个 IT 问题，它和业务息息相关。

软件质量问题不只造成了上述有特别恶劣影响事件出现，中国软件评测中心金融电信测试部总经理罗文兵向《中国计算机报》记者介绍说，众多项目的上马时间一推再推，以及软件产品面临层出不穷的更新，背后都是软件质量的问题，而软件测试问题是矛头所指。

现在，软件测试已经不是传统意义上软件产品交付前单一的“找错”过程，而是软件正常交付、应用提升的一大利器。它贯穿于软件生产过程的始终，是一个科 学的质量控制过程。从一个软件项目的需求调研、设计、编码、验收，直到运行维护，整个过程都需要有软件测试工程师的介入和把关。

罗文兵介绍说，根据执行体的不同，目前他们把测试分为三类：第一方测试，指的是软件开发商、系统集成商内部的测试；第二方测试，指的是用户单位的测 试，即用户委托他人开发了一套系统或者购买了软件产品后，需要对系统或产品进行验收测试；第三方测试，指的是独立的机构或者单位进行的测试，像中国软件评 测中心这样的第三方检测实验室，以及惠普等 IT 巨头，都有独立测试部门为客户提供测试服务。测试外包本质上等同于第一方测试，因为整个测试的要求和管理都 是由第一方负责的。

从被测内容角度看，软件测试涵盖了单元测试、集成测试、系统测试等不同内容。罗文兵介绍，从软件生命周期来看，单元测试、集成测试更偏重于代码级测 试，一般主要是由第一方测试来实现，少量依靠第三方。用户测试和第三方测试做的更多的是系统级测试，主要是从业务执行角度，来看软件能不能完成业务要求。 系统测试层次更高，全面的系统测试包括系统的功能测试、性能测试、安全测试等。

“软件测试的重要性不言而喻。微软做 Windows 产品开发时，测试人员与开发人员的比例是1：1，甚至达到了2：1。他们边开发边测试，测试是贯穿整个开发过程的。”某 IT 领域资深人士对记者说。

罗文兵强调，现在对于电子商务、金融、电信等行业企业而言，系统和业务是一体的，因为其信息化依赖程度很高，信息系统的质量直接决定着经营能力，它们的产品创新都是

依赖后台的信息系统来实现出来。如果它们的软件测试做得不好，对业务的影响是显而易见的。

那么，测试工作怎么才能更贴近业务？细化行业分工和提升测试能力是两条关键的途径。

三方测试一个都不能少

“第一方、第二方、第三方，三方测试都是必要的，一个都不能少。”罗文兵说，“软件测试链条中的各个角色，必须各司其职：软件开发商和系统集成商必须 自己做好严格的测试，为用户提供高质量、可信的软件产品；用户要根据自己的需求，做好自开发和所购买产品的验收测试；第三方测试机构则更是要一丝不苟地为 第一方的产品质量把关，让用户方放心。”

阜外心血管病医院信息中心主任赵韡对此有相同的看法。他认为，软件测试应该做到谁的东西谁负责：软件供应商应该做好测试，保证自己的产品质量；阜外心 血管病医院也要根据自身的需求，做好自行开发系统以及所购买产品和系统的测试。赵韡介绍说，阜外心血管病医院有80多个系统，有自己开发的，也有直接购买 的标准产品，各系统之间需要进行很好的集成。赵韡指出，金融、电信等行业因为拥有巨大的并发用户数和数据量，实时性要求很高，而对测试环境要求非常高，但 对于软件测试而言，医院属于小行业，因此测试环境的要求也就不那么高。他介绍说，阜外心血管病医院信息中心的测试部门有4人，他们应用的基本是虚拟机，甚 至用一些旧机器，就可以搭出测试环境，完成产品、系统的测试。虽然，对测试环境要求不高，但医院的软件测试却有自己的特点，那就是要求测试人员必须对医院 业务非常熟悉。鉴于目前第三方测试机构的行业积累还比较低，赵韡认为，目前医院还是自己来测试比较靠谱。

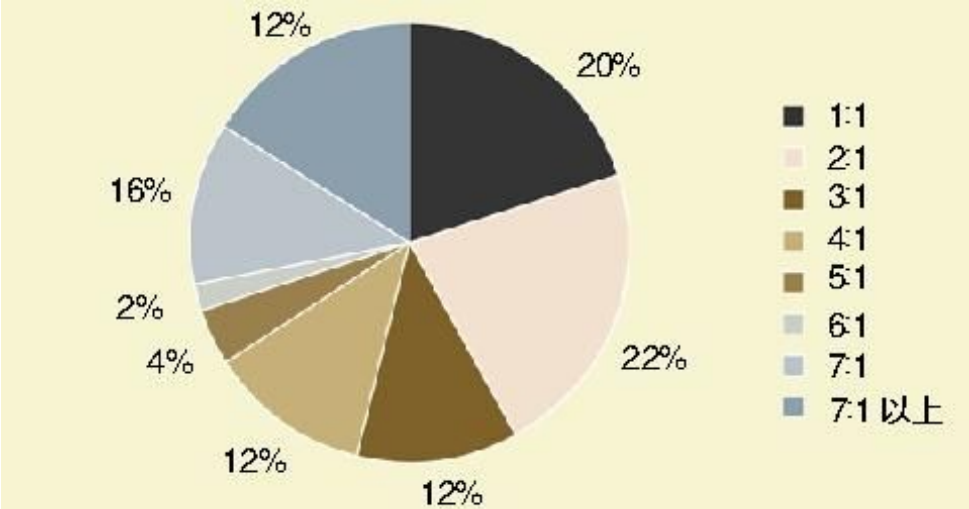
某金融机构的测试工程师朱倩在接受记者采访时表示，她所在公司的主要业务是从纽交所等国外金融机构购买金融信息，提供给国内的一些诸如像大智慧这样的 金融市场行情软件提供商。他们每天要保障海量金融数据的准确性，必须做好数据库系统的测试，因为行情软件提供商也要用他们的软件来进行数据分析和发布。 “我们必须不断地测试，以保障我们的数据库数据的准确性。”朱倩说。

对于任何类型的测试执行主体而言，无论是产品级还是系统级的测试，都是需要衡量成本收益的。测试团队的建立、测试环境的搭建、测试工具的选择、测试过程的管理、外包与否，企业都要根据自己需求和实际情况来衡量后再做决定。

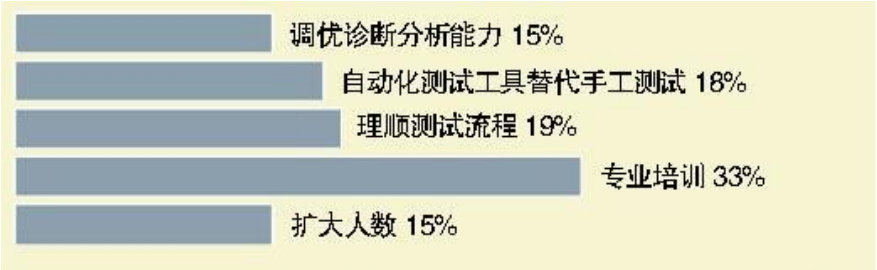
北京中原地产信息技术部经理王雨在接受记者采访时表示，他们现在没有独立的测试部门，测试基本是由开发人员与业务人员组队完成，或者是通过内部用户小范围试用来完成测试，“如有需要，我们会考虑将软件测试外包给第三方，因为这样对我们来说投入产出最合适。”

于志伟介绍，软件测试与开发同样重要，必须从测试需求、测试工具、测试环境等方面提升软件测试的专业性，更好地保证软件质量。另外，于志伟强调，测试 具有非常强的行业特征，比如同样是客户关系管理系统，电信行业和金融行业测试的重点是不一样的。因此，于伟认为，第三方测试将走向更独立、更专业、更细化 的行业分工是必然的趋势。罗文兵也强调，第三方测试机构必须专注于行业。“与软件开发一个道理，做金融开发的集成商是相对固定的，不会随意跳转行业，因为只有通过更多的积累才 能对行业需求了解

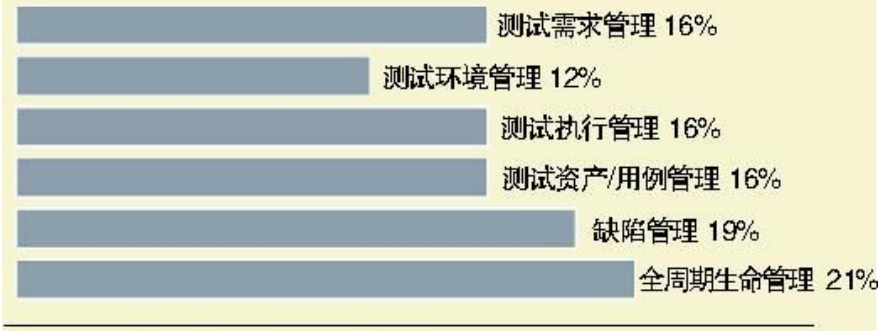
得更透彻。中国软件评测中心之所以成立金融电信测试部，就是顺应这种市场需求。这样，我们天天做金融或者电信业务的测试，会更加熟悉被测 软件的业务特点。如果对业务的了解不到位，就会出现漏测等问题，会最终影响业务的正常开展。”罗文兵说。



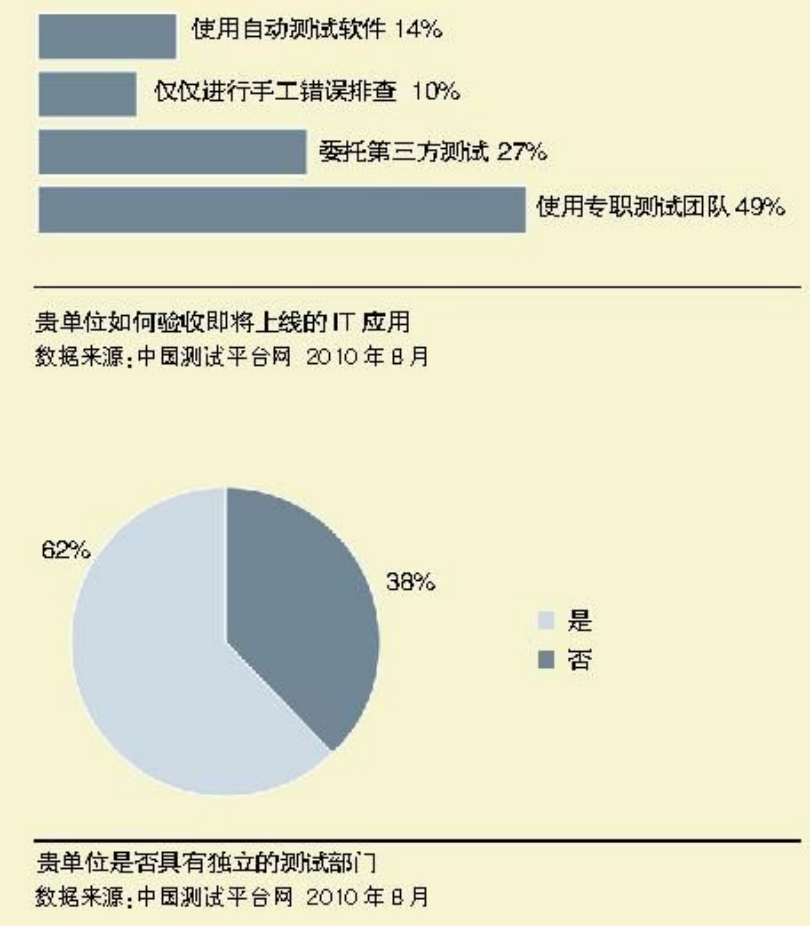
贵单位开发人员与测试人员比例
数据来源:中国测试平台网 2010 年 8 月

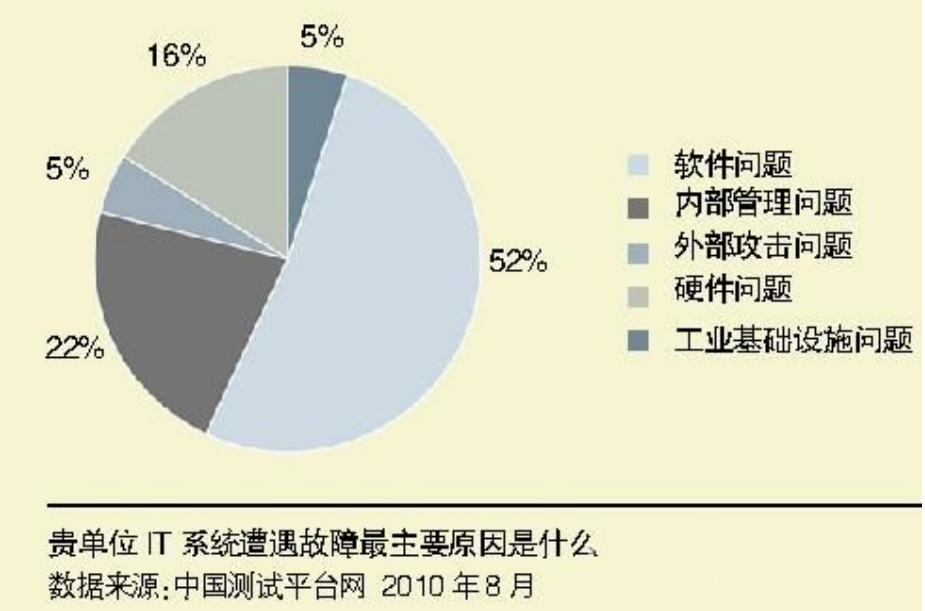


贵单位的测试团队在哪些方面最需要得到加强
数据来源:中国测试平台网 2010 年 8 月



贵单位测试管理在哪些方面最需要得到加强
数据来源:中国测试平台网 2010 年 8 月





中国测试平台网通过电话和网络对 1066 家企业 IT 应用质量的调查显示：企业其研发人员和测试人员的比例远高于国外的 1:1，对测试工程师进行测试技能培训是目前企业测试工作最需加强的，而对测试管理的各方面内容都需要加强。

高端人才不只是找 BUG 高手

葛优在《天下无贼》里那句话说得好，“21 世纪最缺的是什么？人才！”

君不见“我国软件测试人才缺口已达 20 万、30 万、40 万”之类的新闻频频见于报端，各类测试工

程师培训机构如雨后春笋般冒出，测试工程师已经成为各大招聘网站中最热门、活跃的职业之一。

测试工程师的进入门槛看似很低，实际上要做一名合格的、高层次的测试工程师并非易事。而目前国内的测试领域最缺的就是高端测试人才。

那么，到底什么是高端的测试人才呢？

于志伟用围棋中的段位形象地向记者做了说明。“10 个测试工程师中有 5 个可能是业余级别的最高段位。他们不是专业级别的。”于志伟进一步描述说，“类似业余级别的一般测试人员更关注‘测’，对测试工具运用得淋漓尽致。如果发现了 BUG，该测试人员会兴高采烈地跟开发人员说有一个 BUG，但是开发人员也不知道应该怎么修改处理。为什么呢？因为该测试人员有本事把 BUG 找出来，但对 BUG 的描述不到位，也就是不能告诉开发人员这个 BUG 具体是什么，究竟是什么原因造成的等细节。他们不能提供一个合适的路径让开发人员直接找到 BUG。开发人员还得来回测才能确定 BUG。可见，这样的测试人员只是一个找 BUG 的高手，但是他不能解决问题。”

“要做好软件测试并不容易。软件测试是一个终生的职业，越老越吃香。”郑人杰认为，行业业务经验的积累和专业测试能力的提升，是成就一个出色的软件测试工程师的两个同等重要的条件。

于志伟认为，目前，软件测试领域最受关注的都是工具的应用等具体测试问题，而高端测试人才更应该关注如何做好从测试需求、测试计划、测试流程、测试执行，到将测试结果很好地展现在报告上，进而追踪问题，并最终解决问题的全过程。

同时，高端测试人才还要有更先进的理念，即不只是关注测试，而是要更加关注软件质量。“也许有时候及时发现了 BUG 也不应改动，因为它对客户体验没有影响，而改动后会带来更大的影响，并增加成本。但怎么判定这个 BUG 该不该动呢？现在市场上缺的就是能够做出正确判断的高端测试人才。他们具有先进的理念，掌握科学的方法论。”于志伟分析说。

怎样才能培养出大量的高端测试人才呢？

通过第三方的测试平台，把先进测试的理念、更专业的测试方法和工具传达给喜欢测试的 fans(热衷者)，让他们吸收更多的最佳实践，不断提升自身的测试技能，借此来实现对高端测试人才的培育和储备，这是中国惠普培育软件测试市场的一条尝试途径，也是它们与中国信息主管网合作建立中国测试平台网的初衷所在。

中国测试平台网

测试 fans 的资源池

中国测试平台网希望能够搭建一个针对测试 fans 的资源池，让他们能在这里发现、吸收软件测试的先进理念、知识、专家指导、工具技巧、实践经验、职业机会等各种资源。

“当前，信息化正在改变 IT 角色，IT 已成为企业发展的重要支撑元素。企业在 IT 建设中，必须缩短开发周期，快速响应业务需求。然而，IT 项目愈是‘短、平、快’，愈是容易在一定程度上降低系统的可靠性。IT 链条的牢固程度取决于其最薄弱环节——软件测试的疏漏。”作为中国测试平台网项目负责人，中国信息主管网副总编杨春晖介绍，“当软件质量成为软件产业新的核心竞争力时，作为软件质量‘把门人’——软件测试人才自然成为企业争抢的稀缺资源。软件测试人才已成为 2010 年 IT 业就业的主力，其职业具有极大发展潜力。”

正是在这样的背景下，中国信息主管网联姻软件质量管理领域的龙头企业——中国惠普，共同建立了软件测试领域的专业技术网站——中国测试平台网。“中国测试平台网坚持以报道软件测试专业技术为核心，关注软件测试领域的前沿技术和管理思想，定期举办各种在线活动以推动软件测试相关领域的交流，力求为中国广大软件厂商、系统集成商、IT 渠道，以及 IT 专家、测评专业人士、热心网友等提供一个软件测试、专家问询、交流沟通的在线互动平台。”杨春晖说。

“中国测试平台网也会为软件测试爱好者提供很多职业发展的机会。惠普现在就很需要高端的测试人才。”于志伟向记者透露。

除了用中国测试平台网这个平台网聚、培育软件测试人才，惠普也在逐渐向其大量的高端客户渗透软件测试的一些先进理念，以得到他们的认同并付诸实践，从而带动上下游的公司形成良好的市场环境，促进产业内软件测试工作的进步。

当记者提到现在由于企业没有成套考核体系来评估软件测试人员，所以没有足够动力推进软件测试工作更上一层楼时，于志伟笑了笑说：“这确实是目前大家迷茫的地方，大家都不知道软件测试的评估、考核该怎么做，但这也恰恰是体现惠普优势的地方。”

于志伟介绍说，惠普拥有先进的测试生命周期管理、应用生命周期管理的方法和最佳实践，并且在这个领域完成了很多成功项目，能够明确测试的定位、测试与开发的关系，知道哪些指标应该分给开发，明白不同的测试项目中这些指标应该怎么调整，从而帮助企业一步步走向更高水平的软件测试，或者更准确地说是保障软件质量。“这需要一个过程。要通过先进的理念和经验的积累，才能慢慢摸索出测试的考核、评估办法。”

中国惠普企业业务集团软件及解决方案部华东、华南售前经理，测试专家王慧慧向记者介绍了惠普的应用质量管理(AQM)的质量成熟度模型。王慧慧说，该模型不仅参考了已有 IS09000、ITIL、CMM 等行业标准，而且总结了惠普客户的实践经验，将企业软件质量保障情况从最基础的第一步到最高级的最后一步 CoE(CenterofExcellence)，分为 5 个阶段，模型中涉及测试人员的素质，外界对测试人员的认可度，测试人员自身的定位，测试的组织架构、流程、绩效考核等制度问题，还涵盖了测试的先进方法。

“对用户而言，这个成熟度模型是其衡量自己软件测试水平的一个标杆。通过比对，并且在惠普的专家、产品和服务的帮助下，企业可以在现有状态的基础上一步一步提升，最终走向 CoE。”王慧慧说，“这个模型不是静止的，而是在不断充实、改进中。每个用户在改进的投入、时间、步骤等方面都会不同。对于某一用户，惠普专家会为其做出评估，量身定制出一条适合用户自身情况的改进路径。”

记者手记

软件测试七大戒律

软件要控制人的思维，但思维是不可控的。这决定了没有 BUG 的软件是不存在的。“测试是要被终止的”，这是测试圈内一条原则性的定律，意思是说软件测试要适度，不能不问代价一测到底，过分追求没有 BUG 的完美软件。通过大量的采访，受到业内多位专家的启发，记者认为当前我国软件测试领域有七个趋势：

一、不只是为了测试，软件测试工作应有更高的定位，那就是提升软件质量。测试人员的工作目标不仅仅是找 BUG，而是与开发人员、业务人员协作，以得到质量更高的软件。

二、对任何类型的执行主体而言，软件测试都是一项需要衡量投入产出、成本收益的工作，因此测试团队的建立、测试环境的搭建、测试工具的选择、测试过程的管理、外包与否，企业都要根据自己的需求衡量决定。

三、开发与测试是对立统一的整体，二者的工作内容和考核都不可能严格分开。企业可以从管理学和心理学的角度加强对测试管理的研究和实践。

四、测试工作的专业分工将更加细化，测试工程师岗位分工和测试机构行业分工趋势将显现。

五、当前我国测试人员的能力有待提升。我们除了需要熟练的找 BUG 高手，还需要能站到更高层面，从开发、测试的整体层面掌握测试需求、设计、流程和结果展现的人才。这要求测试工程师掌握更新的测试理念、更高的测试技能，具备更深的业务积累。

六、已经渗透至整个软件开发生命周期的测试管理工作是一个有机的整体，缺陷管理、测试需求管理、测试环境管理、测试用例管理、测试执行管理是组成木桶的木板，任何一块都不能短。

七、如何对测试工作进行考核评估是当前业界的一个难题，有待进一步研究、实践。

Google 是如何做测试的

在所有我被问及的问题中，最多的就是关于谷歌是如何测试的。尽管在博客中【google testing blog】中有过零碎的解释说明，但还是需要更多的系统阐述。虽然谷歌的技术路线在执行的过程中不断地进化，但公司的测试策略却从来没有变化过。谷歌现在是一家拥有搜索、应用、广告、移动、操作系统等产品的公司，我们在这些涉及到的产品领域里发挥着非常有意义的作用。当我们涉及到一些新的领域或者在旧领域里快速成长的时候，必须要求我们的测试也在同步的扩张和改进。在这个系列文章中提及的测试技术，多数是我们当前正在使用的，还有一些是希望以后在不久的将来可以用到。

首先先介绍一下组织结构，这一部分也可能会让你感到惊奇。其实在谷歌没有真正的测试部门，测试依托在各个产品领域部门里，我们称之为“工程生产力”【原文， Engineering Productivity】。工程生产力部门拥有数量不等的水平或者垂直的工程学科，测试是其中的大头。简单地说，工程生产力部门由以下几部分构成：

1. 一个工具产品团队【a product team】，负责内部和外部开源的促进生产力的工具开发与维护，这些工具会被公司范围内的各种工程师使用。这些工具包括代码分析工具、IDE、测试用例管理系统、自动化测试工具、Build 系统、源码管理系统、代码审核调度系统、缺陷管理系统等等。 这些工具的都是为了提高工程师效率的，并且这些工具在策略上的目标多数是为了防止问题的发生，而不是发现问题本身。

2. 一个服务团队【a services team】，给产品部门【注：这里的产品部门团队是和工程生产力部门平级的，例如 Search、Gamil、Chrome 等产品部门】提供一些专业的建议，包括一系列工具、文档、测试、发布管理、培训等方面，这些专家建议涵盖可靠性、安全、国际化等，甚至包括产品团队面对的功能问题。所有的其他产品领域也都会得到这样的建议指导。

3. 嵌入式的工程师【Embedded engineers】，在需要的时候被产品部门高效地“借”去使用，这些工程师有些会和产品部门的团队坐在一起工作数年，另外一些当他们被需要的时候会被借调到其他的产品团队。谷歌鼓励所有他们的工程师更换产品团队，用以保持团队忙绿且不断有新面孔，并如实地不带有任何偏见与政治。测试人员也是这样，但是可以有节奏地选择更换产品团队的频率。我的下属里，有测试人员已经在 Chrome 团队工作了好几年，也有一些待了一年半后就换到了其他团队。对于测试经理来说，必须在团队的产品经验和新鲜度上做出很好的平衡。

所以这意味着测试同学向工程生产力部门的经理汇报，但是他们会把自己看成产品部门团队的一员，像搜索、邮箱、和 Chrome 部门。从组织架构上看，测试都是两个团队的一部分。测试和产品团队坐在一起，参与计划，一起吃饭，共享奖金，享受像全职的产品团队成员一样的待遇。这种单独的组织汇报关系的好处是可以给测试人员之间提供良好的共享信息的讨论机会，好的测试思路可以很容易的在工程生产力部门内部蔓延，无论公司内的哪条产品线，都可以很快地使用这些最好的测试技术。

测试人员的这种项目分离和汇报组织结构也有它的缺点，目前来看，最大的问题是测试人员被看做外部资源。产品部门团队不能对测试人员有太多的依赖，他们自己必须要合理地控制产品质量。是的，没错，在谷歌，是产品部门团队对产品质量负责，而不是测试人员。每个产品部门的开发人员都需要做测试工作，测试人员的任务是为产品部门团队搭建自动化测试基础设施和流程，测试人员让开发可以自给自足地、独立地做完成测试工作。

在这种模式下，我比较喜欢的是，开发和测试将有相同的地位。在质量方面，开发和测试成为了真正的伙伴，最大的质量重担交给本应属于的开发人员，开发的职责就是正确地实现产品功能。另外这样

可以保持多对一的开发测试比率，开发人员在数量上远超测试人员，并且测试工作做的越多，开发测试比率就会越大。产品部门团队也会对这样的高开发测试比率而感到骄傲。

好，现在好像大家都是好朋友了，对吧？ 相信你已经看到这种模式的一个问题，开发人员不能很好地驱动缺陷【Bug】的运转，开发不会测试。难道我要否认这一点么？不管怎样，我都不会否认，特别是去年在GTAC talk 【 GTAC 2010: Turning Quality on its Head ， link http://www.youtube.com/watch?v=cqwXUTjcabs】上做了一个开发和测试对抗的游戏后。（友情提示：测试赢了游戏）【这里感觉翻译的不好，原文是： No amount of corporate kool-aid could get me to deny it，especially coming off my GTAC talk last year where I pretty much made a game of developer vs. tester (spoiler alert: the tester wins).】

在谷歌，解决这个问题的办法是将角色再细分，我们通过设立不同的测试角色来解决这两种不同的测试问题。在下一篇文章里，我将详细阐述这些测试角色和谷歌是怎样将测试问题分成两部分来分别解决的。

第二篇：

为了实现”谁的屁股谁自己擦”这句名言所说的那样，在传统的软件开发人员的之上，有必要增加了几个角色，特别是需要工程技术方面的特殊角色，这种角色可以让开发更高效低做测试。在谷歌，这样角色的职责是让其他人工作的更有效率，这样的工程师通常会把自己当做测试人员，但他们真正的使命是提高生产力/生产率。他们的存在是为了让开发人员效率提升，特别是在质量方面的提升，因为产品质量是生产率中最重要的一部分。这里是这些角色的总结：

【注，“you build it， you break it”， you build it ，you break it ， you fix it， 原意指在 Build Lab 的人永远不会去修复 build break 的问题，只有开发人员自己才能修复。这里的意思是开发人员自己要对自己写的代码负责，比专职的测试人员更适合做测试工作。这里意译为”谁拉的 shi，谁的屁股谁自己擦”】

软件开发工程师【SWE，Software Engineer】， 就是传统的开发人员。软件工程师实现一些功能代码并把最终产品提供给用户使用，他们创建设计文档、设计数据结构和总体的架构搭建，他们大多数时间都在写代码和评审代码。同时，他们也会写很多的测试代码，包括测试驱动设计，单元测试，并参与后面的文章会讲到的小、中、大型测试的创建工作。软件工程师需要对他们自己写的代码、修复缺陷的代码、改进的代码，只要是他们接触过的代码的质量负责。

软件测试开发工程师【SET or Software Engineer in Test】，和软件开发工程师一样是开发工程师，主要负责软件的可测试性。他们参与设计评审，近距离地关注代码质量和风险，对代码做重构为了系统有更好的可测试性，同时他们负责写单元测试框架和自动化测试的框架。在代码级别上他们和软件开发工程师是合作伙伴，但如果和增加新功能或提升性能相比较，他们更关心产品的质量和测试覆盖率的提升。

软件测试工程师【Test Engineer】，和软件测试开发工程师【SET】恰恰相反，他得主要工作是做测试而不是开发。许多谷歌的软件测试工程师会花很多的时间在写测试代码上，包括自动化脚本、使用场景的代码、甚至模拟最终用户的操作方面的代码。他们对软件开发工程师和软件测试开发工程师的测试工作做一些组织安排，解释测试结果、驱动测试的执行，特别是在项目即将发布的后期将起到非常重要的作用。软件测试工程师既是产品专家也是质量顾问更是风险分析师。

从质量的角度来看，软件开发工程师对功能开发和质量负有全责。同时，他们还负责容错设计、故障恢复、TDD、单元测试、和在软件测试开发工程师的帮助下写测试代码，这些测试代码会验证开发的功能。

软件测试开发工程师是提供测试支持的开发人员。他们提供一种能够将新添加的代码通过模拟其依赖的方式做功能验证的技术框架，并应用在代码提交之前的提交队列管理之中【注，这样可以在代码 check in 的时候保证新代码的功能完备】。可以这样说，软件测试开发工程师就是为了让软件工程师可以测试他们的功能代码，所有真正的测试都是软件开发工程师完成的，软件测试开发工程师是保证这些功能有很好的可测试性，这样可以让软件开发工程师很积极地参与到测试用例代码的编写中去。

现在所有的一切很清楚了，软件测试开发工程师就是服务人员，他们的主要职责就让开发人员很方便简单的做测试并保证模块级别的产品质量。读者可能已经意识到一个问题，在这样的研发流程下，使用软件的最终用户会怎样？

在谷歌，软件测试工程师的职责就是最终用户级别的测试。如果软件开发工程师和软件测试开发工程师很好地做了模块级别的功能测试，下一个工作就是看许多功能集成和数据的组合是否能够满足最终用户的使用需求。软件测试工程师在这里就是扮演一个双重确认开发工程师测试工作的角色，任何明显的缺陷都会说明之前一轮的开发自测不够或比较草率，如果没有出现这种情况之后，软件测试工程师会将注意力转移到普通用户的使用场景测试上，保证性能、安全、国际化等方面没有问题。软件测试工程师们需要做大量的测试工作，并在测试工程师、测试合同工、吃狗粮的尝鲜者、beta 测试用户、早期最终用户之间做很多沟通交流的工作，他们会和基础设计、功能复杂度、避免错误的方法等方面遇到问题的人做确认。一旦软件测试工程师开始介入，总是会没完没了。

好了，现在大家对这些角色应该有比较好的理解了，接下来我会在他们之间是怎么分工合作做更详尽的剖析。下次再聊。。。谢谢您的关注。

第三篇：

经过前两篇的介绍之后，评论里留下许多问题。并没有逐一回复，当然不是想把这些评论置之不理，而是希望在这里和后面的文章中做详细介绍和解释这些问题。从这一篇开始，我将开始讲谷歌是如何测试软件的了。

在谷歌，质量不等于测试，是的，我确定在其他所有的公司也都是这样。“质量不是被测出来的”，这句陈词滥调是再正确不过的了。不管汽车制造还是软件开发，如果在最初的设计建造的时候就有问题，那它永远都会有问题。试问任何一家曾经被迫大量召回汽车的公司，逃避质量问题的代价是多么的昂贵。

但是，“质量不是被测出来的”这句话本身并不像它听起来的那么简单和准确。虽然质量并不是被测出来的，但同样也有证据表明，未经过测试，也不可能开发出有质量的产品。你连测试都没有做过，又是怎么知道产品功能是否正确，并有高质量呢？

对于这种难题，最简单的办法是不要区分开发和测试，不要把他们当成对立的活动。测试和开发【注，两种行为，不是人】最好能手牵手的并行，写一点代码就立刻进行测试，写的越多，测的就要越多。最好是，在编码的同时，甚至在编码之前，就考虑清楚这些代码将如何被测试。测试不是一个单独的工作，测试就是开发的一部分。所以，质量并不等同于测试，当把开发和测试混在一起，搅拌直到分不清他们彼此的时候，就得到了质量。

这就是谷歌的想法，把开发和测试工作混在一起，不分彼此。写点代码，就必须测试，多写一些就多测一些。关键的问题就是谁来做测试工作？ 由于谷歌的专职测试人员非常的少，唯一的答案就只能是开发人员。还有比实际写代码的开发人员更适合来测试这些代码的人吗？还有比程序的作者更懂得怎样去发现程序 bug 的吗？是谁更想知道程序在第一次运行时是否有没有问题呢？谷歌之所以用这么少的专职测试人员的原因就是开发对质量负全责。实际上，如果一个团队在过于依赖测试的时候，通常情况下这个团队在开发上一定也会有问题。如果在这个团队里，测试人员比较多，这也是一个强烈的信号，

这表明开发和测试融入到一起的程度还不够，失衡了，缺乏测试，单纯地增加测试人员并不能解决任何问题。

这意味着，对于质量来说，预防问题比发现问题本身更重要。质量是开发人员的问题，而不是测试人员的问题。通过把测试工作融入到开发过程中，我们能降低那些富产 Bug 的人的出错机会，不仅可以避免了大量最终用户的使用问题，而且还可以极大地降低测试人员报无效 Bug 的数量。在谷歌软件测试工程师的工作目标就是检查这种预防措施是否有效，软件测试工程师不停地寻找一些证据来证明作为 bug 的作者和预防者的“软件开发工程师-软件测试开发工程师”组合是否存在问题，一旦发现任何不正常，就会拉响警笛。

这种开发和测试一体的场景随处可见，不管是在代码审核的时候问“你的测试呢？”，还是在厕所蹲坑里张贴着的最佳测试实践–臭名昭著的马桶测试指南【译者注，参见 google test blog，有关于”Testing On The Toilet“的更多介绍】。测试是开发过程中必不可少的一环，质量是开发和测试合体的产物。软件开发工程师，软件测试开发工程师，软件测试工程师，所有的人都是测试人员。

如果你所在的公司也想要做这种开发和测试的统一，请也给大家分享一下其中经验和教训。如果没有，这将是一个帮助你公司的机会：让开发和质量划等号。你大概知道谚语里说的，鸡和猪为了一顿有培根和鸡蛋的早餐都乐于奉献自己，但是猪却牺牲了。好吧，这就是事实，尝试跑到开发工程师那里，对他们”哼哼“（猪叫声）两声，看他们是否也用”哼哼“来回应，如果他们”咯咯哒“（鸡叫声）来回应，那就说明有问题了。【译者注，崩溃了，这里比较难懂。James 这里引用了一个猪和鸡的英语谚语，（参见，http://en.wikipedia.org/wiki/The_Chicken_and_the_Pig ），谚语的意思大概是，猪和鸡都参与制作培根鸡蛋早餐，猪变成了猪肉（培根），鸡只下了一个蛋，说明对于早餐，猪和鸡的奉献程度是不同的。并在这里把测试工程师比喻成鸡，开发工程师比喻成猪，早餐就是质量，猪的奉献大。测试人员跑到开发人员那里，如果发现他们没有做猪的事情，早餐将做不成，那说明质量也将会有问题。】

第四篇：

爬，走，跑。

在比其他公司少很多测试人员的情况下，谷歌做的还不错的一个关键原因是，很少尝试在一次发布中包含很多的功能。实际上，谷歌经常反其道而行之，在一个产品的核心模块被开发后，如果有一定数量的受益人群就立刻发布，然后不断的得到用户反馈再迭代开发新功能。这也是我们在 Gmail 上的做法，Gmail 被贴上 Beta 版本的标签在线上运营了四年。通过这个 Beta 标签也可以来警示用户，Gmail 还并非完美的产品，有出错的可能。只有当邮件数据达到 99.99%的时间都是可用的时候，我们目标就算达到了，这个 Beta 标签才会被去除。很明显，质量是一个不断改进的过程。

这里的这个改进过程，并不像西部牛仔那样，一下子什么都能搞出来。实际上，一个产品为了达到我们称之为 Beta 的版本，也要经历一系列的过程，并在过程中证明其价值。Chrome 是我加入谷歌前两年一直所工作的团队，它同样经历了多个版本。在每个版本里，基于对产品质量的信心和不断寻求的客户反馈才让我们进入到下一个版本。这些版本历程大致如下：

金丝雀版本（Canary Channel），不太可靠的版本，并不适用于发布。就像一只金丝雀在煤堆里一样，如果不幸身亡，那说明还有工作要去做。只有超强容忍能力的用户才有可能使用金丝雀版本来试验运行，你不能依赖这样的应用能把实际工作完成。

开发版本（Dev Channel），是开发工程师们日常工作中使用的版本。所有的同一产品组的工程师都需要去安装这个版本，并在真正的工作中使用他们。

测试版本（Test Channel），是给内部的狗食【译者注，dog food，一般指自己团队的产品，给自己或者公司内部的人尝试使用的中间产品】，如果能够有持续不错的性能表现的话，也可能会是 beta 版本的候选。

Beta 版本或发布版本(The Beta Channel or Release Channel)，是给外部用户使用的第一个版本。只有在之前的各种版本历程中通过了测试和真实用户的枪林弹雨般的验证后，才会成为 beta 版。

上述的这种从爬到走、走到跑的模式，让我们在运行一些测试同时又可以对我们的应用系统做一些试验调整，并从真实用户和每个版本的每日自动化测试那里得到及时的反馈。

对于这样的过程，还有一些分析角度的益处。例如，发现了一个 bug，测试人员可以根据这个 bug 创建一个测试用例，并针对所有的每一个版本都运行这个测试用例，从而可以验证这个 bug fix 是否在所有的版本中都真正得到了修复。

【译者注：关于 Chrome 与 Chrome OS 各版本的称呼，可以参见 http://www.chromi.org/chromedownload ，其中也涉及到本文中各个版本的称呼，但并不是与 James 文中完全一致，实际上像金丝雀版本，一些喜欢尝鲜的外部用户也在使用】

<谷歌如何测试> 翻译第五篇

正文，

Wednesday， March 23， 2011 8:27 PM

By James Whittaker

对于测试范围的形式，谷歌并没有使用通用的代码测试、集成测试、系统测试这些常用术语来做区分，而是使用小规模测试、中等规模测试、大规模测试这样的称呼【译者注：代码测试(code testing)， 通常指单元测试和 API 级别的测试，一般使用 XUnit、Gtest 框架，但谷歌并没有使用代码级别测试这种说法】。小规模测试就是针对小量代码的测试，中等规模测试、大规模测试以此类推。所有的三种工程师角色【译者注，软件开发工程师、软件测试开发工程师、软件测试工程师，参见本系列第二篇】，都会去执行上面的三类测试，可能是自动化的测试，也可能是手动测试。

小规模测试，通常（但也并非所有）是自动化的，一般是针对一个单独的函数或者模块。这种测试一般由软件开发工程师【SWE】或者软件测试开发工程师【SET】来实现，通常在运行的时候会依赖模拟环境，当软件测试工程师【TEs】需要去诊断定位一个特定错误时，会去筛选一些小规模测试集合并运行来验证特定问题。对于小规模测试，主要集中在常见功能问题验证上，例如数据损坏、错误边界、发生错误时如何结束等。小规模测试尝试去解决的问题是，代码是否按照其假定的方式运行。

中等规模测试，可以是自动化的或者手动的，涉及到2个及以上功能模块，特别是要覆盖这些功能模块之间交互的地方。有不少软件测试开发工程师【SET】把这种测试描述成“测试一个函数，以及它最近的邻居们”【”testing a function and its nearest neighbors.”】。软件测试开发工程师在独立的功能模块开发完毕后会驱动进行这种测试，软件开发工程师是写这些测试代码、并调试和维护这些测试的主要力量。如果一个测试用例运行失败或者运行错误，相应的开发会自动地跳出来查看处理。在开发周期的后期，软件测试工程师会运行这些中等规模测试，可能是手动的方式（如果很难或者需要投入比较大成本去自动化的时候）或者自动化的方式去运行。中等规模测试尝试去解决的问题是，一些相近的交互功能模块组合在一起是否和预期一致。

大规模测试，涵盖三个及以上（通常更多）功能模块，描述最终用户的使用场景及其可能扩展。所有的功能模块集成为一个整体的时候需要去关心许多问题，但在谷歌，对于大规模测试，更倾向于着重结果，例如，这个软件是用户期望的那样么？所有的工程师都会参与到大规模测试中，无论是使用自动化还是探索性测试方法。大规模测试尝试去解决的问题是，这个产品运行地是否是最终用户期望的那样。

小规模测试、中等规模测试、大规模测试这些术语本身其实并不重要，你可以给它们取任何你想的名称。对于谷歌的测试人员来说，有了这样一个统一的称谓后，就可以使用这些称谓来讨论正在进行什么样的测试以及其测试范围。有一些雄性勃勃的测试人员也会谈到第四种测试，被称为超级大规模测试，公司的其他测试人员可以认为这样的测试是一个非常大的系统级别的测试，涵盖到几乎所有的功能而且会持续很长的时间，其他的解释都会比较多余了。

哪些需要被测试及测试范围的确定，这是一个动态变化的过程，在不同的产品之间会有比较大的差异。谷歌更倾向于频繁发布，从产品的外面用户那里得到反馈之后再迭代开发。如果谷歌开发了一些产品，或者在已有产品上增加了新功能，会尽可能早地对外发布并让外部用户能使用并从中受益。在这个过程中需要较早地把用户和外部开发者牵扯进来，并要有一个很好的处理规则来验证是否满足发布条件。

最后，自动化测试和手动测试，对于所有的三种类型测试【小规模、中等规模、大规模测试】来说当然更喜欢前者。如果能够被自动化，而且不需要任何人智力和直觉判断，那就应该把它变成自动化的。只有在特别需要人为判断的时候，例如用户的界面是否漂亮、或暴漏一些涉及用户隐私的内容时，在这些情况下应该保留手动测试。

话虽如此，对于谷歌来说非常重要的是仍然使用了大量的手动测试，不管是使用文本记录的方式还是使用探索性测试，虽然有些已经进入了自动化测试的视线。业界使用的录制技术将手动测试转变成自动化测试，可以在每个版本后自动地重复运行，这样保证了最少的回归工作，并把手动测试的重点放在新问题上。而且，谷歌已经将提交 BUG 的过程和一些手动测试的日常工作也自动化了，例如，如果一个自动化测试运行失败，系统会自动检测到最后一次代码变更的信息，一般来说这是引起测试失败的原因，系统会给这次代码提交的作者发送一封通知邮件同时自动创建一个 BUG 来记录这个问题。在测试上，“人类智慧的最后一英寸”体现在测试设计上，谷歌的下一代测试工具也正在这个方向上努力尝试，将其自动化。

这些工具在以后的文章中会被提及强调。不过，下一篇文章还是会将重点放在软件测试开发工程师【SET】的工作上。希望能得到你的持续关注。

<谷歌如何测试> 翻译第六篇

Monday， May 02， 2011 12:05 PM

By James Whittaker

软件测试开发工程师【SET】的生命

软件测试开发工程师【Software Engineers in Test】是软件工程师，专注在测试实现。首先，软件测试开发工程师是开发角色，在招聘和内部晋升资料中被我们奉为100%的编码角色。当在招聘面试软件测试开发工程师的时候，对于编码的要求几乎和对软件开发工程师的要求是一模一样的，而且更强调如何去测试自己写的代码。换句话说，软件开发工程师和软件测试开发工程师都需要回答编码问题，而且软件测试开发工程师会被问到一系列测试相关的问题。

正如你可能想到的，这是一个很难满足的角色。软件测试开发工程师的数量如此之少的最有可能的原因是，事实具备软件测试开发工程师所需技能的人非常难找，而不是我们刻意使用了什么神奇的生产率公式【译注，开发测试比率公式】， 这也是我们适应当前工程实践的一个必然结果。我们还在优化我们的工程实践–这是一个非常重要的任务，并且为所有参与的人构建一些流程。

通常来说，软件测试开发工程师不会在早期设计阶段就介入。不是故意这样做，而是和多数谷歌的产品是如何诞生的有关。一个常见的新产品诞生的场景是这样，已有的谷歌产品的员工会投入20%时间来开始新的产品。Gmail 和 Chrome OS 这2个产品，从一个简单的想法开始，并非正式地由谷歌授权去做的，慢慢地随着时间的推移，越来越多的开发和测试加入进来并把产品发布。在这种情况下，早期的开发要关注的重心并不是质量，更关注提供一些理念，在解决了扩展性和性能的问题之后，再更多地关注质量。如果你创建了一个 web service，但是不具有可扩展性时，测试这时候还并不是你最大的问题。

一旦这个产品已经明确未来可以发布的时候，研发团队就开始寻求测试的介入了。

你可以想象这样一个过程，某个人有了一个好主意，他开始思考并写了一些试验性质的代码，从其他人那里获取一些建议然后对这些代码做了改进，并劝说更多的人加入，写了越来越多的代码，然后意识到他们做的事情很重要，通过更多的代码把这个想法变成可以呈现给他人并得到反馈的模型… 这个项目在谷歌的项目库中就创建了，这个项目慢慢地变成了一个真实的项目，测试也只有在项目变成真实的项目之后才会介入进来。

所有真实的项目都有专职的测试人员么？ 默认情况下是没有的。小型项目和只有受限用户使用的项目一般是开发人员自己做测试。其他的一些对个人或者企业用户有潜在风险的项目，测试会介入。

当开发团队寻求测试团队参与并帮助他们时，他们有责任使测试人员相信他们的项目是令人兴奋并充满潜力的。开发总监会给测试总监解释他们的项目、进度、发布计划，一起讨论测试工作如何划分，并就开发需要满足的单元测试水平及开发参与测试工作程度上达成一致，发布流程中开发与测试的责任也需要明确。软件测试开发工程师在项目初期可能不会参与进来，一旦项目变成真实的项目后，测试人员将在软件开发过程中发挥巨大的影响力。

当我说“测试”时，并不是仅仅意味着单纯的检查验证代码路径。测试人员不是从开始就参与进来的，但“测试”却至始至终都有。实际上，一个开发尝试去 check in 代码的时，测试人员的影响力在这个时刻可能就已经显现出来了。【译，这里指软件测试开发工程师会对开发人员的测试程度做一些要求，开发人员在 check in code 的时候需要想一下自己是否满足这些要求，这就是测试人员的影响力】。欢迎继续收听并尝试理解我所说的这些东西。

缺陷测试中容易忽略掉的几个问题

在系统测试和确认测试中，有些缺陷由于某些原因往往被忽略了，这就给软件留下了隐患或者危机。本文通过描述这些容易忽略的缺陷，提供一个完善测试用例和测试执行的参考。

通常软件测试会暴露软件中的缺陷，经过修正后可以保证软件系统的功能满足需求并正确运行。但是，在系统测试和确认测试中，测试人员容易遗漏一些隐藏的缺陷。众所周知，软件测试不可能发现所有的缺陷，而软件开发周期各个阶段仍然存在注入缺陷的可能，但是，有一些缺陷是测试中容易忽略的，也就是说，通过测试方法和用例可以充分暴露这些缺陷，遗憾的是，它们往往被忽略或者某种原因忘记测试了，这就给软件留下了隐患或者危机。这些容易被忽略的缺陷包括：

1、安装缺陷

通常项目组完成代码后，发布时候安装打包是最后一个环节，而软件测试人员通常在测试的时候，没有仔细的测试这一部分，而把用例集中在其他功能上。安装时候的缺陷通常通过拷贝而不是运行安装程序方式给测试人员安装软件，结果正式安装时候出现问题，引起例如控件没有注册，注册表没有导入等。删除时候没有注意安装文件夹是否存在用户文件，造成数据丢失；使用绝对路径；安装顺序没有说明书。

2、配置文件

有些文件在 ini 等配置文件中写出了管理员口令密码等信息，而且是明文的！这是一个安全隐患。另外，有些安装文件的 XML 文件，为了方便在数据库和中间层连接文件中写入了 Admin 口令和密码。作为一个合格的软件测试人员，必须检查这些可以用记事本打开的文件。因为，一个稍有常识而且喜欢探索的用户，可能从中获取信息而成为不自觉的黑客。所以，配置文件可能成为软件安全方面的一个缺陷。

3、网页安全缺陷

现在网站开发已经注意到：登陆网站进入其内部网页后，直接拷贝网址，然后粘贴到另一 IE 窗口输入，可以绕过登陆直接访问。也许商业网站很关注这个问题，但是很多行业软件却很容易忽略。

网页安全缺陷还可能存在于 IE 弹出的子窗口。有些设计不严格的软件，在主页面关闭的时候子页面还可以运行，这是一个明显的漏洞，而且还大大增加了错误发生的几率。

4、判断顺序/逻辑缺陷

对界面进行多个输入判断的时候，非常容易出现这种问题。例如判断年月顺序，判断长度，判断非空等。假如操作员仅仅满足单个条件，保存不能成功；而按界面从上之下顺序一一满足条件之后，保存是没有问题的。但是，改变一下输入的次序，校验失效。例如，一一满足条件之后，不保存，倒过来将上面的输入改成非法输入，然后保存，结果居然也能成功，这是因为原先的判断由于发生过，或者根据语句顺序只检查最后一个判断，所以没有报错。这种错误尤其在 Java scrīpt 脚本的页面中要注意。能够保存不能保证数据正确，有可能引起系统崩溃或者后续数据错误。所以，在测试的时候，不要按照正常的顺序输入，而是要打乱步骤，看看代码是否强健，是否在判断逻辑上没有错误。良好的代码应该经得起折腾，至少保存时会再此全部进行判断，而不只是简简单单走到判断的最后一行。

我认为的最佳职业生涯建议

原文作者尼古拉斯·泽卡斯(Nicholas C. Zakas)是一位前端大牛工程师，目前在 Box 公司任职，之前是在雅虎将近工作 5 年。在雅虎期间，他是雅虎首页的前端技术主管，并且是 YUI 库的贡献者。Nicholas 编写的技术书有：《Maintainable JavaScript |编写可维护的 JavaScript》、《Professional JavaScript for Web Developers |JavaScript 高级程序设计》、《High Performance JavaScript | 高性能 JavaScript》、《Professional Ajax》。

最近我与一同事有一次有意思的讨论。我们回忆了各自所走过的职业历程以及不同个性如何长期消极影响我们的职业。事实情况是，我曾经是那种从大学里走出来的令人讨厌的人(有些人可能会说我现在仍然是那种人，但这是另一回事儿)。当时我很傲慢并且很刻薄，是一个十足的愤青。我自以为我很了解自己的性格并且为这种性格感到骄傲。

我曾经经常指出更有经验的工程师的错误之处。尽管我所提出的大部分错误是正确的，但是由于我的个性问题使得解决这些错误并没有这么高效。比如在一次对话中，其中的一名高级工程师突然恶狠狠的说道，“假如你不闭嘴，我就用屎**把你赶出去。”我只是笑笑因为知道他不敢。一年之后我就意识到，他是真心想做这件事的。

从那时开始我成长了很多，开始学习如何说话，如何尊重人。这种挖苦在职业环境下得到了控制;当我与好朋友在一起的时候，我把它们放到一边。这些自我控制能力伴随着其它无价的教训并非来自自身内部，而是由一路上的人生导师引导的。如果没有他们，我的人际关系将会使我的职业生涯变的很糟糕。

因为工作中接触到很多优秀的人，所以我是幸福的。我的经理们一直以来将自己塑造成性格很好的人。我为他们感到自豪。更甚，受到他们影响，我不仅成为一名好的编程人员—-也成为一名优秀团队成员和优秀个人。他们对我的人生影响很大，以致于我经常将他们的建议讲给我所指导的同事们。

我发现这些建议具有普遍适用性，所以决定将它们分享给大家。当然，有些内容是经过改述的(本人记性不大好，不能把每个词都记住)，但相信我现在已经抓住了主要思想。

不要成为只会做快餐的厨师

我的第一份工作持续了8个月，之后这家公司就关闭了。当跟经理讨论下一步我该做什么的时候，他建议我：

“Nicholas，你的价值不只有你的代码。无论接下来的路是什么，确保你自己不是一个仅会做快餐的厨师。不要去接受那些有明确目标并且步骤已经很详细的工作(译者注：以我理解应该是像软件外包那种工作)。你应该去那些赏识你的洞察力以及构建产品能力的公司”

我牢记这句话很多年。做代码实现者不够好—-我们应该参与到整个开发过程中。一名好的工程师不仅是按部就班的实现功能，还应该给予反馈，与产品的拥有者一起工作，这样才能构造出更好的产品。很幸运，我的工作选择都很明智并且我从来不会在一家不尊重、不重视我的洞察力的公司待很长时间。

自我推销

有一天，在 Yahoo 的经理将我拉到一边给了我些建议。他监督我的工作，后来发现我有点内向：

“你工作很棒。我喜欢你代码的风格以及它的连贯性。然而，其他人并没有看到。为了使你现在的工作得到好评，你应该让别人看到你的代码。你需要做一些自我推销来引起注意。”

刚开始我并没有理解他的话，但后来我明白了其中道理。即使你工作很棒，但如果没有人看到你所做的内容，这并没有帮到你多少。你经理能支持你，但不能为你做证明。你组织里的人需要知道你的价值所在，最好的方式就是告诉他们你做了什么。

我将这个建议告诉过许多同事了。自我推销并不是说，“看我，我很牛逼。”它意味着让别人知道你的工作有了巨大进展或者让他们知道你学到了一些新内容。它意味向别人展示你所骄傲的成果。它意味着庆祝自己以及别人的成就。它意味着向你所在的组织证明你的价值。坐在角落默默敲代码的工程师总是有一些神秘感—-不要那样。一封简短的邮件，“好，我完成了新邮件的布局。你看看有什么建议吗。”，往往会起到很大的作用。

“人”比技术重要

在职业生涯的早期阶段，我是头衔驱动型。我总是想着如何做才能被提拔。在雅虎主页上与新经理的第一次一对一会议中，我问需要做什么才能得到提拔。他的话仍然在脑海中盘旋：

“从某种意义上讲，你应该结束对自己技术的评判，开始关注与人交流的方式。”

之后，我没有收到过对软件工程这个职业比这更具洞察力的见解了。他完全正确。在那时，没有人怀疑我的技术能力。我以写高质量，几乎零 bugs 的代码而出名。我所缺少的是领导能力。

从那时起，我看到无数工程师处于他们职业生涯的瓶颈期。他们聪明，写着一手好代码，然而缺乏有效的与同事高效交流的能力。这将他们困在原地。一旦有人困在他们软件工程生涯的瓶颈期，我都会给他们这个建议。

“问题”不是问题

我在 Yahoo 失意过一段时间。可能“失意”这个词并不正确，更像是愤怒。我经常愤怒地与人争论。结果事情变的很糟糕，我自己也不想这样。有一天，我心情非常差，就问我导师如何在面对这么多问题时保持冷静的。他回答：

“很容易。这些问题都不是问题。有这么多垃圾代码混到站点中，致使其崩溃，那又如何?工作并不是你生活的全部。这些不是真正的问题，他们是工作上的问题。工作之外所发生的事情才是值得关注的。我回到家里，我妻子在等我。那才是幸福的。”

那时，我从马萨诸塞州搬到加州，人生地不熟，很难交到朋友。这样工作就是我的全部，它是我保持正常的寄托所在，所以一旦工作出现问题也就意味着我的生活也出现问题。通过这次谈话我明白生活中需要某项我能够回去然后忘掉工作中遇到的麻烦的事物。

他是对的，当我调整心态并且将这些工作中遇到的令人恼火的事情重新归为“工作”的时候，我能够思考的更加清楚。我还能够让自己冷静下来与人进行更愉悦的交流。

权威，由你做主

当被提升为雅虎的首席工程师时，我与主管一起讨论这个职位所需要承担的责任。我明白这个职位更应该是个领导者，但是我并不知道如何使自己更具权威性。我请他帮忙。这是他所说的：

“我不能告诉你应该如何具有权威性，每个人的风格不同，你应该自己发掘出来。你应该做的是找到适合自己的风格。我不能告诉你你的风格是什么，但是你应该找到适合这个职位的。”

那一年，我花了很多时间来观察那些有权威的人以及他们与人交流的方式。我把他们走路的方式，讲话的方式以及处理问题的方式记录了下来。我试过许多不同的方式，最后终于找到了能为我用的风格。我的风格只适合我，任何处于权威性位置的人都会经历同样痛苦的学习过程。我的优势是领导一开始就跟我讲明了情况。

从“怎样?”转到“什么?”

在与经理的一次交谈中，我问道这个新职位的期望是什么。他回答说：

“到现在为止，你的职业在回答“怎样?”这个问题。即我们告诉你应该做什么然后你想出怎样做。而从这一刻开始，你应该回答的问题是“什么?”。我希望你能够过来告诉我应该做什么。”

我看到许多工程师都在这个部分犯错误。如果没有这个建议我同样会陷入困境。从“怎样?”转到“什么?”是很困难的，并且需要许多时间来发展。你需要对自己所向往的以及所关注的事情有一个比较成熟的认识。毕竟，假如你能够花费时间在任何你想的事情上，你也应该独自对自己所创作的作品负责。

在盒子中，我们称其为“开环运行”，意味着在最少的监督下你完成工作并且仍然对组织和公司有一个整体的积极影响。就在这一阶段许多工程师失败了，我将这个建议给那些努力想要到下一阶段的工程师。

表现出你在负责

以往开会的时候，我只是坐在那儿并不知道该讲些什么。在与主管的一次面对面交谈中，我提到我只是在开会，并不知道我为什么会在那儿并且也没做什么贡献。他说道：

“以后永远都不要这样。假如你在会议中，那是因为你参加了。假如不确定自己为什么会在那儿，停下来问一问。如果你不需要在那儿，那就离开。你在一个领导的位置，那就表现的像领导。不要静静的走进一个房间。只要表现出你在负责，那么人们就会相信。”

从这个建议里，我的导师使我想起从高中学到的一个教训：没有人知道你什么时候在表演。假如你很紧张但是表现出并不紧张的样子，那么别人就不会知道你很紧张。领导能力也是一样的。一句古语“久演必成真”出现在脑海中。从那时，我从来没有在会议中静静的坐着。我确保自己只去参加那些需要我参加的会议。

让他们赢

我经历过一段时期，在这段时期团队中有许多争论。我为自己使用权威来结束这些争论而感到很满意。我有一个“我的规则是最终的结果”的心态，我的经理注意到这件事情并且给我建议说：

“我看到你们团队有许多争论，而你经常逼进他们，赢了很多。我知道大部分时间你是对的，但每隔一会儿应该让他们赢。选择那些对你要紧的事情，对这些进行推进，其它的事情让他们赢。没有必要赢取每一次争论。”

这是一则我一开始就坚持的建议。几乎所有时候我都是正确的，那为什么应该让其他人赢呢?然而，随着我的成长我开始相信他的本能，我决定试一试。结果是：争论减少了。他们不想要必须赢过我一次

了，并且反过来，我能够更好的识别不需要太关心的事。我坚持那些重要的问题，将那些不重要的事情让别人来解决。所有对话的强烈程度都大大的降低了。

结论

回头看看那个刚刚毕业、非常无礼的小男孩，我的职业生涯可能非常不一样。我曾经被认为是一个不满现状，聪明但是很难伺候的人。假如不是因为一路上所遇到的导师以及在职业初期所遇到的一些令人羞辱的失败，我的交际能力(缺乏)会令我疲惫不堪。这些天，我经常找到那些比我更具经验的人并且向他们索取建议。我可能不会再犯一些大的错误，但是我也不会等着一个错误发生然后去找个我信任的人问经验性见解。

在 Yahoo 的接近五年时间是我职业生涯中变化最大的。我工作面对的都是大规模的有趣问题，但是我更庆幸自己能够同一系列非常优秀的经理和导师在一起工作。将我变成现在所自豪的人(无论是工作还是生活上的)的原因是那些对话。

假如我能够给你们一条最重要的建议的话，那就是：找到从某一方面(无论是技术上还是组织能力上等方面)比你明智的人，然后“黏”上他。比如如果你们能够定期的一起吃午饭或者喝咖啡，那么就开始挖掘他们脑袋里的大量知识。通过这样做，你的职业生涯甚至你的生活都会变的非常不同。

在软件行业里，几乎所有的开发人员都在谈代码质量，而每个人对代码质量都有一套自己的看法。甚至术语代码味道(code smell) 也已进入大众词汇表，成为描述代码需要改进的一种方式。

代码味道是由我们开发人员根据自己的一些工作经验积累来判断的，有人觉得代码注释可以体现代码结构和质量，还有些人又认为代码注释是用来解释过于复杂代码的一种说明机制。显然，Javadocs™ 很有用，但是多少内嵌注释才足以维护代码?如果代码已经编写得足够好，它还需要自己解释吗?从这些我们可以看出代码味道是一种主观评估的机制，在很多情况下面，尽管一些看其来糟透了的代码可能是他人曾经编写最好的代码。在我们工作中，是否有很多这样的声音，”是的，初看起来有点乱，但是它的扩展性不错”。

因此，我们需要客观评估代码质量的方法，某种可以决定性地告诉我们正在查看的代码是否存在风险的东西。不管您是否相信，这种东西确实存在!用来客观评估代码质量的机制已经出现了一段时间了，只是大多数开发人员忽略了它们。这些机制被称为代码度量 (code metric)。

目前一些公司如华为、普元等都在代码质量方面有比较严格的要求，采用 CMMI5的规范来评估代码质量。他们根据单元测试覆盖率作为代码质量的一种保证手段。单元测试覆盖的种类有下面几种：语句覆盖、分支覆盖、条件覆盖、路径覆盖。在单元测试中前三种覆盖率都非常容易达到，但会存在一定的缺陷。在这篇文章中，我就不详细解说前三种覆盖率的计算方法了，重点谈一下路径覆盖率的问题。

圈复杂度，它可以精确地测量路径复杂度。通过利用某一方法路由不同的路径，这一基于整数的度量可适当地描述方法复杂度。实际上，过去几年的各种研究已经确定：圈复杂度大于 10 的方法存在很大的出错风险。因为圈复杂度通过某一方法来表示路径，这是用来确定某一方法到达 100% 的覆盖率将需要多少测试用例的一个好方法。公式圈复杂度 V(G)=P+1 ，P 是代码中判定结点的数量，下面我们看一个简单的类。

```
package com.alisoft.kplan.atest;

public class PathTest {

public String testA(boolean p1){

String a = null;

if(p1){

a = “”+ p1+ “”;

}

return a.trim();

}
```

```
}
```

在我们平时开发过程中，通常这样写一个测试用例，语句覆盖率达到100%

```
package com.alisoft.kplan.atest;

import junit.framework.Assert;

import org.junit.Test;

public

class PathTestTest {

@Test

public void testTestA() {

PathTest pt = new PathTest();

Assert.assertEquals(pt.testA(true)， “true”);

}

}
```

这个测试用例虽然语句覆盖率达到100%但是我们会发现，其中有一个潜在的空指针错误没有被发现。问题来了，那么我们在编写测试用例的时候，怎么来写一个优秀的测试用例呢，答案很简单，就是根据圈复杂度来计算你的类方法复杂度，圈复杂度值越大，就说明你的方法越复杂，存在的缺陷会越多。通过计算公式 V(G)=P+1 testA 方法的圈复杂度为2，那么我们只要编写两个测试用例就可以就可以完成testA()方法的基本路径覆盖。我们在看一下这个测试用例

```
public class PathTestTest {

@Test

public void testTestA() {

PathTest pt = new PathTest();

Assert.assertEquals(pt.testA(true)， “true”);

}

@Test

public void testTestAfalse() {

PathTest pt = new PathTest();
```

```
Assert.assertEquals(pt.testA(true)， “false”);

}

}
```

通过这个测试用例，我们就可以很容易的发现方法中的那个空指针错误。从这个例子来看，这个方法非常简单，因为它的圈复杂度只有2 ，像我们有些系统中某些方法的圈复杂度值高达150左右，那么你还能这么容易的发现你的程序缺陷吗?按理论值来算的话，你需要编写150个测试用例才能完成每个基本分支的测试。为什么要 TDD 模式开发?为什么要求大家都写单元测试?为什么评估软件质量要用覆盖率来评估?归根结底一句话：降低代码复杂度才能保证软件质量。

推荐大家使用圈复杂度计算的工具 JavaNCSS，可以生成 html 报告。PMD 等工具都可以评估代码复杂度。

在持续集成环境中，随时间变化评估方法的复杂度是很有必要的。如果某一方法的圈复杂度值在不断增长，那么您有两个响应选择：

1、确保相关测试用例的路径覆盖率是否覆盖到方法中所有的路径。

2、重构方法，降低长期维护风险。

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

SPASVO