泽众软件测试网
电子期刊

SPASVO

第二十期
2013年8月

上海泽众软件电子期刊

2013 年 8 月 第二十期

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
2013 年 8 月 第二十期 电子杂志期刊

谈软件测试人员定位---三年软件测试总结

因为一直从事 web 产品的测试，我的观点并不一定适合所有的类型项目。

工作已将近三年了，虽然这三个年头里我都在积极的学习着与测试相关的技术；但是能沉淀的东西很少。相信测试同学都有类似的感觉。

不要为了测试而测试

前几天做了一个测试的 PPT ，就是讲项目中要用到的测试技术，总结了半天其实实际的产品中没什么技术，熟悉需求，转化成用例，待项目上线后验证功能就 OK 了；对一个自身质量要求不高的项目，我们有时候为了体现自己价值，非要在一些不痛不养的问题上揪着不放。

举个不恰当例子，某钢琴高手开了一个补习班教钢琴，家长送来一孩子目的只是让孩子学学钢琴；钢琴高手为了体验自己的价值（牛 B），硬是按照贝多芬的标准去培养，孩子弹不会《XX 交响曲》不让孩子走。先不说孩子有没有贝多芬的钢琴天资，也许孩子压根就不想成为贝多芬。

当然了，如果你办的是“中国音乐家钢琴协会”，你有责任要求会员达到国际超一流水平，为国家和个人赢得荣誉。

有时候不要为了测试去测试，或为了体现自己的价值去做一些对整个项目贡献不大的事儿。当然，我在这里不是让测试人员放弃自己的原则。要知道不管是产品、开发、测试都是围绕着产品的发展贡献。

为贡献产品的发展测试远比为了测试了测试所带来的价值大得多；所以站在产品的发展上去看待测试工作更能体现自己的价值。

记得去年的总结再讨论自己对流程的理解。随着工作年龄的加长对这些问题也有进一步的看法；所以，再拿来炒一炒，希望能炒出新的味道。

没有最好的开发测试流程，只有最适合项目的开发测试的流程；

去年的一篇说软件测试流程，严格规范的测试流程一定比没流程好，敏捷的流程一定比传统的瀑布流程先进。这个观点没有大的错误，但是我们忽略了所做有产品这个“对象”；忽略了产品的特点与阶段。

例如两三个开发合伙开发一个项目（或产品），这时你让他们建立一套规范的流程，按流程实施，显然是不现实，我想摆在他们面前最主要的问题是，如何快速的把客户需要的功能开发出来换成 money ，维持生计以及公司运作。

例如一个各种功能已经成熟的项目，有着庞大的用户群，以维护为主的更新，它的版本功能的上线必须要建立严格的发布流程，经过充分的测试才能上线；用户群越大，暴露的问题越多，问题带来的影响也会越大。

同样是一个 web 产品，笔者目前所做的项目流程完全不是这样；我们的发布流程很简单，测试流程也很简单，不去写的规范又复杂的测试用例，放弃了使用缺陷管理工具来反馈问题；

沟通变得尤为重要；我不否认这样做会给产品带来了一定的风险；对于严重的问题，我们可以通过快速的版本回滚，对于轻微的问题，我们很快会在下个版本迭代中修复。是不是有点敏捷的味道在里面。

为什么会这样？因为这个产品属于前期开发阶段，很多功能还没上线。整个团队都在贡献着产品的发展；需要快速的将需求转化成功能给用户使用。

所以，没有最好的开发测试流程，只有最适合项目与阶段的开发测试的流程；

产品质量与用户容忍度

之前看过不少人讨论到底需不需要测试人员；我想说测试人员 N 年后不管是被重视了还是被淘汰了“测试的行为”永远不会消失；因为没有质量的产品基本上等于没有价值（也就是说没存在的意义），至于对产品质量的要求是由用户容忍度决定的。

Facebook 没有测试人员！但是测试行为一直都在。开发找需求，开发、自测、发布，获得用户反馈，决定功能下线还是上新的功能---相当于一条龙的服务。因为用户的容忍度允许他这么做。

微软不能这么干，修复一个 windows 的 bug 成本很高，而且用户是花钱买的，也许用户是用来创造价值的（办室、存储、管理），也许一个文件丢失，系统崩溃会给用户带来巨大损失；所以，微软需要很多的测试员。

拿修复成本与用户容忍度做标准，web 产品优于客户端产品；在 web 产品中也要分行业；用户对银行系统、火车票、购物网站的容忍度显然要低一些，反过 来说也就是对产品的质量要求更高，因为与钱挂钩。就算同一个产品，会员与免费用户的容忍度也是不一样的；因为会员用户有权得到更好质量与服务。

所以，关注分析用户的容忍度的测试才不会把自己变得格格不入。

提升自己的贡献

前面的东西貌似都在“弱化”测试存在的价值；俺本来就不被重视，所以俺就需要更加认真和努力找问题来提升自己存在的价值，你现在说，有些产品不需要太指着的去测试；那你说俺还能干啥？

当我们把测试看成是为开发和产品服务时，也许情况会完全不一样。我们可以提供哪些服务？

·用测试发现产品的不可以测试性

前面已经提到队团不管是否有测试人员，但测试行为一定会存在；如果一个产品都不可测试，如何去发现并修复 bug ，如何去维护与扩展？尤其对于 web 产品来讲，不可维护与扩展的产品无疑是致命的。（可以通过项目重构再解决）

·建立产品质量的评估方法

为项目团队提供每个版本的 bug 趋势分析数据，让项目中的每个人都了解项目当前的状态

通过分析 bug 数据来建立或完善各种 Checklist，帮助项目团队更好的完成需求评审、设计评审以及代码评审，减少 bug 出现的机会。同时，可以 定期将多个项目的 Checklist 进行合并，使单个项目的经验可以通过 Test Team 快速的流动起来，及时的作用于其他项目

主动为 Architect Team 提供每个项目的性能测试数据，帮助他们获取更多的实际项目信息，减少踏入“陷阱”的几率

·建立可持续运行的测试框架

建立自动化测试测试框架；

构建持续集成，使版本的迭代与更新得到快速的反馈。

·建立关注开发质量的开发文化

没有测试人员自测节省人力的了，尤其在单元测试层面。产品的质量应该由开发与测试共同承担。（现实中的责任到人，让团队很难形成这种文化）

·贡献产品发展

旧病成医，测试的产品多了自然会对产品有自己的理解，产品的定位，用户习惯与体验； 可以从测试的角度贡献产品的发展。（这个由产品的特点，公司文化决定）

困惑的软件测试员

一个家庭主妇在微软当软件测试员，《软件开发的科学与艺术》一书中讲述了这个真实的故事。那位妇女四十多岁了，高中毕业，非常初级的计算机水平还是跟 着自己的女儿学到的。让一个大学都没有上过的家庭妇女做测试人员是多么不可思议的事情！不过，她思维独特，怪点子很多，能很快的发现一些问题。微软最终决 定聘用她，而她也成为了一名优秀的测试员。

在 IT 圈里做技术的人群里，测试员可能是一个特殊的群体。他们之中计算机专业 “科班出身”的并不多，毕业于其它专业的大有人在，什么建筑、中文、营销专业的都有。另一方面，相对那些做研究开发、项目管理、软件实施甚至是售前售后技 术支持的技术人员，测试工作给人的感觉总是技术性低，因此测试员的薪酬水平与其它同等资历的研发人员相比总有一些差距，甚至在重视测试的外资软件公司也是 如此。测试不过是每天重复一些操作来发现 Bug（错误）。事实的确如此，哪怕是非常热爱这项职业的人也承认它的枯燥。

H 做专业的测试工程师已经有一年多的时间了，目前仍然在做较为底层的测试。有时候也会写写测试需要的代码，但还没有开始设计整个项目测试案例。目前 H 正在为微软的某一软件做测试，工作的流程非常严谨而明晰，这自然也意味着枯燥的重复。枯燥并没有淹没 H 的工作激情，发现一个 Bug 带来很大的成就感，特别 是想到每天将会有几百万人通过使用没有这个 Bug 的软件准确无误的达到他们的目的。

前途在 H 心目中是非常明朗的，颇有一些“随需”择业的味道。曾经有媒体报道过近来软件测试工程师在职场需求中的风光景况，尽管 IT 行业的总体需求仍然 疲软。在北京和上海等地，测试员的需求量占到了招聘总量的近 1/3。另一方面，H 认为从测试员成长为软件项目管理者是更有优势的。例如微软的开发方式本来就是“测试驱动”的，在测试过程中发现了墙角还有没涂到油漆 的小块，开发则根据这个思想再补上那一块。测试的经历恰好让人更能从用户的角度来考虑问题，更能深入了解程序开发过程中可能出现的问题，这都是成为一个优 秀的项目管理者的必要条件。尽管可能一整天都为了一个小控件“循规蹈矩”地反复测试并撰写测试文档，这样的重复被 H 当作了重要的积累。H 喜欢新东方学校的 徐新书《骑驴找马》中的一句话：“重复做汉堡，就是麦当劳；重复煮咖啡，就是星巴克；重复教托福，就是俞敏洪；重复做好事，就是活雷锋。”

不过，乐观的情况并不具有普遍性。在另一家软件公司做测试的 L 对工作感到厌倦。这是一家国内著名软件公司的子公司，整个公司测试员就只有两个人。公司 不久前才刚刚把测试作为一个单独的部门划分开，尽管建立了测试管理流程，但是没有代码测试，也没有测试工具。测试案例并没有完全规范化，很多测试都是随机 的手工操作。这样的测试部门更像是一个辅助性的、服务性的部门，测试员的收入也比开发人员低一个档次。在这样的工作环境下，L 觉得是为了生活而忍受枯燥， 最痛苦的是这种得不到锻炼和进步的状况。

L 所在公司对测试的态度在国内具有一定的代表性，将测试部门独立都只是最近的事情。更一些公司仍然停留在开发人员自行测试的阶段。可是如果开发者自己 能找到 Bug，谁还会在开发时犯下这样的错误呢？在软件业发达的国家，软件测试工程师地位丝毫不亚于程序开发员，一些公司对测试员的要求甚至是曾经做过程 序开发的。对测试的重视更体现在人员的配置上，以微软 Windows 2000产品团队中最主要的三类人员为例，项目经理约250人，开发人员约1700人，测试人员则是3200人左右。

国内软件业的测试员大都与 L 一样困惑。选择离开并不能解决问题。整个测试环节成熟起来，才将意味着测试员地位的改善。

烫手的山芋：中途接手项目的测试

随着我国软件开发和测试能力的提高，以及人力成本相对低廉，越来越多的跨国企业在中国设立了研发中心，从而将部分项目从国外转移到中国的研发中心。这时候，摆在测试团队面前的，将是如何在已有项目的基础上进行有效的测试，即如何对中途接手项目开展有效的测试。

对于测试团队而言，测试中途接手的项目和产品，存在很多的风险和挑战。本文首先从测试的角度阐述中途接手项目测试可能存在的风险和挑战；然后根据笔者的经验和知识，提出测试方面的建议，以帮助测试人员更好的开展此类条件下的测试活动。

1）风险和挑战

（1）项目测试经验欠缺

对于中途接手的项目，测试团队首先面临的一个挑战是缺少当前项目的测试经验，至少是不熟悉项目，例如：系统的整体设计架构、实现的主要功能、客户的关 注重点、系统的测试平台、系统使用的测试工具、系统配置和管理的命令、系统中主要的缺陷分布等。项目中这些信息和知识，都需要测试团队花费很多的时间进行 学习和理解。测试团队中尽管有的成员可能对系统功能有些经验和知识储备，但是对于该项目而言，测试经验方面肯定是欠缺的，或者是缺乏经验的。

（2）开发测试文档不全

中途接手的项目，测试团队面临的第二个问题是：原有的需求文档和测试文档可能不全，甚至没有，例如：由于软件开发的成熟度低下，或者由于项目移交的时 候没有将相关的文档成功交接。而后续的测试工作需要详细了解以前项目版本的需求和相关的测试文档，从而可以深入学习和理解以前项目的基本功能。

对开发团队而言，没有原始的需求文档也是一个非常大的挑战。首先，开发人员只能通过研究软件的代码来理解系统的功能和实现。而对系统各种可能的数据输 入都进行研究是不现实的，并且也很容易遗漏一些复杂的功能实现。其次，开发人员在面对特定输入情况下系统产生令人迷惑的输出，开发人员可能会想当然的认为 系统就是这样实现的。令情况更糟的是，开发人员在确定系统正确的输出和工作方式过程中，并没有进行相应的文档化，而是在这基础上直接进行代码设计，导致这 种猜测循环经常持续，从而使需求的收集和文档化更加困难。

而对于测试团队来说，同样面临严峻的挑战。虽然从表面上看，在已有系统上进行测试，测试人员可以通过比较以前软件的输出和新的软件输出结果进行对比。 但是，基于不同版本输出结果进行判断并不总是安全的，例如：原来系统的输出就是错误的。基于这样的判断，测试人员就会遗漏系统中的一些缺陷。或者，原来系 统实现的功能是不正确的，而在新的系统中，由于增加或升级了软件修改了原来代码的缺陷，其输出是正确的。基于两者之间的比较，测试人员会认为新的系统由于 代码的修改引入了缺陷，而提交了不应该提交的缺陷报告。假如开发人员修改了本来是正确的代码，新的软件中又引入了新的缺陷。原本应该在需求阶段确定的预期 结果的判断，落到了测试人员头上。

（3）不稳定的测试对象

在很多时候，项目在移交之前可能已经交付给客户使用了，后续的开发和测试会在用户版本的基础之上进行一些维护工作，例如：新功能的添加、缺陷的修改、 系统架构的更改、新技术代替老技术等。在这种情形下，整个系统看起来更象是移动的不稳定的实体，这样导致的结果是新的系统一直处于多种开发状态的混合体， 它们会经历不同的生命周期阶段，是个不稳定的测试对象。

（4）测试工作量的估算

由于中途接手的项目经常处于不稳定的状态，导致测试工作量、测试进度、测试人员和资源的估算更加困难。测试工作量的估算只能是基于对系统功能的粗略理 解，而所谓的系统功能甚至可能是错误的，或者系统在维护过程中功能经常发生变更。对于测试估算而言，即使有详细的需求文档的情况下，也是一件非常困难的事 情。因此，对中途接手的项目，由于其不稳定的特点，进行测试工作量的估算，将是难上加难的事情。

（5）回归测试用例选择

对于中途接手的项目，后续的开发和测试是基于前面的软件版本而展开的。针对开发活动而言，后续的开发主要是指软件版本新功能的增加、版本的升级、平台 的升级，以及前面版本中遗留的缺陷的修改等。而对于测试活动而言，后续的活动主要是验证新增功能是否符合系统的要求、确认是否满足客户的要求、确定缺陷是 否已经修复以及新增加的功能和缺陷修复没有在原来系统中引入新的缺陷。因此，对于中途接手的项目，测试团队的很多测试工作将关注在由于软件变更而进行的回 归测试。

在中途接手的项目中，回归测试在整个测试活动中会占有很大的比重。因此如何选择每次测试的回归测试用例，对于测试团队而言，也是一个很大的挑战。由于 前面提到的测试项目经验欠缺和开发测试文档的不全，导致测试团队很难进行测试风险的估算，从而很难确定回归测试的重点和优先级，影响回归测试用例的选择。

（6）项目知识的转移

中途接手的项目，还有一个很大的挑战是项目相关知识的转移，例如：系统、开发和测试相关知识的转移。项目是从一个研发中心转移到另外一个研发中心，由 于语言、文化和习惯的差异，在知识转移过程中，很难实现无缝的转移。由于测试团队中本身对项目的测试经验的不足，导致相关知识交流方面会更加困难。

2）经验和对策

从测试的角度，上面谈了中途接手项目中存在的7个主要风险和挑战。虽然存在比较多的困难和不确定因素，测试人员还是可以利用已有的测试经验和知识，采取一些合适的手段和方法来应对这些问题。

下面根据笔者在中途接手测试项目方面的经验，对上面提到的这些问题提供一些参考的信息和建议。这些建议并不是肯定适合的，在进行具体项目的时候，还需 要考虑不同企业组织和不同项目的背景。同时，下面的经验并不是对应解决上面的每个风险和挑战，而是从整体上对如何进行中途接手项目的测试提供了一些实践。

（1）合适的测试经理或专家

对于测试工作而言，测试经理应该是整个测试团队的灵魂，对于中途接手项目的测试中体现的尤为明显。中途接手项目中，测试团队的测试经验相对欠缺，因此 测试工作的计划、估算、执行以及控制等尤为重要，因此需要更加慎重的选择合适的测试经理来领导这样的项目。合适的测试经理，除了需要

具备的一些能力和知识 外，例如：熟悉测试过程、具备测试管理能力等，针对中途接手的项目测试，测试经理具备下面几个方面的能力也非常重要：

测试经理应该对项目产品相关的功能、协议等有很深厚的经验和知识，能够从全局上把握软件产品的风险、测试的重点和优先级。在项目测试初期，最好能够在测试团队中能够起到知识方面的引路人；

测试经理应该有良好的沟通能力，包括对内沟通和对外沟通。由于项目是从国外研发中心转移过来，因此需要测试经理有很熟练的英语沟通能力。

对于有的组织和项目而言，测试经理并不是技术方面的专家。那么，在面对中途接手的项目测试中，测试经理需要选择产品相关的测试技术专家（测试领域的专 家，例如：数据通信领域的专家）对整个测试过程中的技术进行把关。协助测试经理进行测试活动的计划、估算、协调和控制，同时帮助测试经理进行测试团队的构 建和发展，使得测试团队能够胜任中途接手项目的测试。

（2）合适的软件测试过程

选择了合适的测试经理或者测试专家，基本上可以保证测试团队对测试工作的适应性。测试质量中除了人的因素外，另外一个很重要的因素是过程的因素。测试质量的提高和保证，需要一个完善的测试过程来控制和保证。

当然，软件测试过程的选择，需要考虑企业组织的成熟度和测试项目的特性。一般来说，测试过程包含下面几个阶段：测试计划和控制、测试分析和设计、测试实现和执行、测试出口评估和报告以及测试结束活动。

测试过程定义中，需要明确每个测试过程阶段的输入和输出、每个测试阶段的主要测试活动以及每个测试阶段中的质量检查点。明确定义软件测试过程以及相关 的测试活动和输出，可以从制度上保证测试工作的顺利进行，并且可以有针对性的进行测试活动的控制。软件开发的产品质量是通过整个开发过程的质量控制来保证 的。同样，测试质量也需要测试过程质量来保证。

（3）完善以前版本相关文档

收集和整理项目以前版本的相关文档，包括开发文档和测试文档。从测试的角度来说，有两方面的内容：一是学习以前的相关软件需求和设计文档（可能是已有 的，或者通过开发团队收集和整理的文档），来了解软件的主要功能和实现方式；另一个是整理和收集测试相关的文档，比如每个版本的测试计划、概要测试用例、 详细测试用例以及以前版本的缺陷信息等。

假如对每个功能进行全面的分析，需要花费巨大的人力和时间成本，这可能是不太现实的。我们的经验是和开发团队合作，召集各个领域和应用方面的技术专 家，来对以前系统进行需求方面的文档化，至少对每个功能特征进行部分的描述，同时对一些复杂的功能接口、用户接口等进行详细的需求分析，形成比较详尽的系 统需求文档。

针对测试文档，根据前面得到的一些基本文档提供的信息，分析软件的主要功能和重要功能，提供一些测试用户场景以及它们的输出预期。也可以通过探测性测 试，得到软件的一些表现行为和结果输出。将这些用户场景和测试输出等进行文档化，作为后续项目测试的基础，也可以作为回归测试的输入和重点。

（4）软件新增功能的文档化

对新增加和升级的功能特征进行文档化，从确定开发和测试的系统版本开始，我们需要对每个增加的功能、升级修改的功能进行详尽的需求文档化，作为后续开 发测试活动的参考和基线。这样，可以在后续的开发设计、测试设计等方面拥有共同的输入和参考点。这对于系统的研发非常重要，这个环节没有做好，项目的开发 将一直处于混乱状态，例如：系统需求不明确、开发条目不清晰、测试输出预期没有标准等，无法保证项目产品的质量。

（5）基线化一个测试版本

确定一个固定的系统版本作为进行后续开发和测试的基础。所有项目的利益相关者需要在这一点上达到共识：为什么产品的开发是基于已经存在的软件系统之上？然后选择已有系统的一个版本作为新开发的基础，并且它是唯一的进行后续开发测试的版本。

固定后续开发和测试的版本，可以更加直观的来跟踪缺陷。通过对已有系统代码的升级或修改，可以判断在新的软件中某个现象是否是缺陷。在碰到输出结果存 在异议的时候，通过各个领域的专家来验证到底是新系统引入的新缺陷，还是旧系统中本来存在的问题。通过该策略，大家既可以对系统的输入和输出达成一致，也 可以作为对原有系统需求收集和文档化的手段之一。

（6）基于风险的回归测试

对于中途接手的项目测试，回归测试将是测试过程中的一个重要关注点。因此，如何选择回归测试用例，来提高测试的效率和有效性，是测试团队面临的一个重要问题。

通过前面提到的各种建议，可以对测试团队、测试过程和文档化方面进行改进，从而有利于回归测试用例的选择。回归测试用例的选择，基于测试风险而展开将是一个有效的手段。下面的实践经验可以作为一个参考：

分析软件中什么功能是客户最经常使用的？

什么功能对客户而言是最重要的？

什么功能在以前版本中发现的缺陷是最多的？

新增加的功能或者升级，对原来的什么功能和模块的影响是最大的？

在进行回归测试过程中，理解下面的建议将有助于回归测试的开展：

回归测试不是可有可无的，回归测试对于确认变更的正确性和验证没有引入新的缺陷方面有重要的意义和作用，同时可以提供对软件产品的信心。

回归测试不应该是流于形式的，应该制订严格的回归测试过程，包括软件变更分析、软件变更影响分析、定义回归测试策略、定义回归测试套件、执行回归测试套件，以及报告回归测试结果等。

回归测试的重点是保证软件基本功能的正确性，因此在回归测试过程中应该更多的关注在功能性，而不是非功能特性。

回归测试并不一定要覆盖所有的软件特征和功能，需要在测试风险、时间、成本和质量之间进行平衡。

回归测试用例需要不断进行维护和更新，而不是一成不变的。

（7）面对面的知识共享

项目从一个研发中心转移到另一个研发中心，在项目知识的交接过程中，由于语言、文化和背景等方面的差异，可能会存在比较多的问题。我们的经验是，在成 本允许的情况下，尽量采用面对面的知识共享方式，即可以派测试方面的专家到另一个研发中心去接手这个项目，主要的关注点包括：

学习整个项目的演变过程，以前版本包含的基本功能、用户关注的主要功能、每个版本中存在的主要问题等。

尽量多的收集需求文档、开发文档和测试文档，包括原来测试团队在前面项目中测试的经验教训等。

熟悉软件环境的搭建和配置，包括测试仪表的使用、测试环境的基本配置等。

中途接手的项目测试，应该说存在多种多样的问题和挑战，并不是一件容易的事情。但是，通过上面的一些经验、建议、措施和步骤，可以帮助测试团队更好的组织、计划、估算、控制测试活动，从而达到测试成本、质量、时间、风险等方面的平衡。

回顾我的 IT 生涯,从 800 到 8000 的奋斗史

首先我想说，同很多程序员一样，我是不善言辞的，只能用平实的言语来表达自己的感受。经常会来这里看大家发的帖子，却很少发言。今天在这里，写下自己 的真实经历，一是总结我这四年的 IT 生涯，二是给刚进入或即将进入 IT 行业的师弟师妹们一点个人建议，避免以后少走弯路。

真的，我不善言辞，如果有表达不当的地方，还请各位包涵，但求真实，毕竟我不是写小说的作家。

先简单的介绍自己的情况吧。85年出生于广东潮阳，2004岁高中毕业，2006年去深圳一家软件外包公司工作，到现在整整四年了，今年五一的时候 BOSS 给我涨工资了，现在的基本工资是8K 多一点（和福利加起来有9K），一年下来10万有余。说实话，对于这个貌似很诱人的数字，我并不是在这里炫耀 自己的工资水平，每个人获得的成功都是经过努力的，我只想告诉大家，不仅仅是 IT 这一行，不管做哪一行，只要你努力了，成功就离你不远。

2004年，19岁的我高中毕业了，对未来充满希望，却又感觉前途不知在何方。因为高考的成绩不理想，所以与心目中的大学失之交臂，在家里待了几天之 后，就出来打暑假工，在潮阳一家电子厂做装配工，一天工作12个小时候，要命的是上班还得站着，早上8点上班，晚上10点才能去厂宿舍休息。一天下来，骨 头都快散架了，坚持了30天，拿了一个月的工资就回家了。可回家之后，还是得面对高考落榜的现实。

有天我在家里看电视，一个同学打电话给我，问我要不要一起去汕头玩一下。我想了想，在家也挺无聊的，于是就说好。然后我们在车上看到了一所学校，就是广东硅谷软件学院，最终的结果是：我选择了在这个学校学习两年软件编程。这之间如何抉择的过程就省略不说了。

我永远记得报名那天，爸爸双手颤抖地交学费。我跟爸爸说，“老爸请放心，我一定会把握住最后一次机会，一定不会让你跟妈妈失望，我相信不一定要读大学就有好出路，我一定会比那些读大学的同学混得好！”

新的开始给了我信心，也给了我极大的动力。第一，我热爱 IT，想在这个 IT 行业有自己的一席之地；第二，为了自己的前途，为了父母，我一定要学出名堂。从报名的那一天起，便是我的 IT 奋斗史的开始。

在这个学校学习了两年，几乎每天都是七点钟起床，然后7点半到学校的公共机房上机练习，上午上课，然后下午就还在公共机房上机练习，一直到晚上11点 多，有的时候甚至是12点才回宿舍。真的，很怀念在这个学校学习软件编程的那两年的日子，虽然有点累，晚上睡眠时间不到7个小时，但是依然觉得很充实，不 仅仅是学到了技术，更重要的是，锻炼了一种坚持的精神，学会吃苦，这对我以后的工作提供了很大的帮助。

可以说，这两年过得很充实，不仅仅是学习方面，还有生活方面，觉得都很满足。与老师，还有同学的相处，不仅仅是技术上交流，还有人生，感情等等，我们 是师生，更是朋友。学校会经常组织一些户外活动，或者在校内举办一些活动，活跃气氛的同时，也大大的提高了同学们的综合素质，那些本来性格内向的人经过这 些各种活动，变得开朗乐观起来。

说到这个学校，我不得不说一个人，那个给我爱情的另一半，也就是我现在的女朋友。我们是在学校的公共机房认识的，她比我早来，有一天我问她关于 Java 多线程机制的一个问题，刚好她坐在我旁

边，然后我们就这样认识了。当然在学习的两年时间里，都是普通的同学关系，我们会天天在一起学习，互相讨论 问题。一直到毕业前夕，她才确定我们的恋爱关系。我们的感情很好，很少红过脸，而且我们都是来自农村的孩子，家庭背景都差不多，比较简单，可能就是因为这 样，我们才会有共同语言，然后相处的比较好吧。

2006年7月，跟女朋友商量之后，决定还是先自己去深圳找工作，自己实在是找不到就让学校推荐工作，她就先在汕头呆着，我安顿好了之后就让她上去。

（我清楚的记得，第一次到深圳的日子：2006年7月3日，当时中午一点多的样子，我就拿了一台电脑，一个装衣服的密码箱，风尘仆仆地下车了。好在学 校在深圳有就业办事处，所以到了那里以后就不要担心没地方住，可以在找到工作之前暂时住在那里。而且在就业基地的一段日子，很感谢就业辅导老师的帮助，尤 其是关于求职方面的技巧等等。

一边网上疯狂的投简历，一边去各个人才市场，找工作的日子里每天都是早出晚归，似乎除了在那里睡上一觉，其他的时间都是在外面跑。一个星期之后，有家 公司通知我去面试，当时我很兴奋，因为找了一个星期终于有结果了。面试的前一天晚上，我很早就回去了，就业老师跟我聊了很久，跟我讲面试时的一些东西，给 了我第二天的面试很多的帮助，让我能够很顺利地通过。

是的，我怎么也没想到，就是这家公司，我一去就让我在里面呆了四年。那天的面试很紧张，除了我之外，还有15个，第一轮的技术题笔试就刷掉了一半，就 剩下8个人了。初试时面试我的有两个人，一个是开发部的同事，还有一个是人力资源部负责招聘的同事。整个过程基本上都是那个同事跟我聊技术性的东西，跟他 讲在学校学的技术，面试很顺利，然后就进入复试了。

先简单的介绍一下公司吧，公司成立于2000年，老板很年轻，是典型的海归，2000年回国后就创办了这个公司，公司主要是做海外的软件外包。行业领域主要是金融，制造业和零售，客户基本多是在美国，加拿大和英国，日本也有。

复试的人是开发部的经理，看起来很严肃（其实是很不错的人，嘿嘿，因为认识他到现在四年了，他给了我很多帮助，不光是技术上的，还有其他，我从心底里感谢他）。

他觉得他的同事已经跟我主要是面试了技术上的，他就没必要再聊技术，就杂七杂八的聊了其他的话题。问我是哪个学校毕业的，我说，我今年刚毕业于广东硅 谷软件学院。之后就说到薪水，我当时觉得，如果有2K 就干了，可是没想到，他说，800。当时突然感觉倍受打击，800？在深圳，一个工厂妹随随便便一个 月也不只800啊，何况我是懂技术的。经理说，800是试用期的工资，如果转正的话你再跟我谈薪资要求，而且公司有免费提供住宿和工作餐。

我想了想，怎么说也是一份 IT 行业的 IT 岗位的工作，现在有工作不错了，比那些还在找工作到处碰壁的大学生要强。而且试用期是暂时的，又不是长期的。 相信自己的能力，我的价值绝对不是800。就这样了，我开始了我人生中的第一份正式的工作，一直到现在。现在的我想想，如果当初放弃了这次机会，现在的我 会是在哪里呢？

刚进公司的时候，有点傻傻的。我相信刚入职场的应届生都会这样，对一切充满着希望，把什么事情都想象的理想化。不过说真的，不管是学习还是工作，勤奋 是必须的。上班时间是早上九点，但是我基本上都是早上八点就到公司，学习。自知需要学习的东西很多很多，所以总觉得多付出一点点，结果就会不一样。晚上也 从来就是最后一次离开公司，不管是加班工作，还是学习，都是在工作。回到住的地方就是睡觉。

经理这个领导真的很不错，很多次看我一个人还在加班，老是提醒我多注意休息，保护眼睛，不要

长时间的对电脑。当时，很感动。现在的我已经是一个小小的项目经理，也会关心一下下面的同事，觉得很温馨，每一次的关心都会让我想起刚进公司的时候经理对我的关照。

两个月以后，我提前转正了。那个日期很巧，是9月10日，教师节。转正那天，可能是心情比较好吧，一来是转正了，二来就是教师节，于是下班后去了学校 在深圳的就业基地，看看就业老师，请老师出去了小撮了一顿。看到我这么快就转正，老师也非常开心，听说回去后还跟那些还未找到工作的师弟师妹们讲我的事 情。嘿嘿。。。。。

转正后的工资，全部加起来将近三千，这个数字对于刚踏入工作岗位的我来说，确实很诱人。所以也给了我很大的动力，和奋斗的信心。那天和老师在一起吃饭的晚上，我还说：我不会让学校的老师失望的，一定以后要在公司混出样子，到时候让师弟师妹们也方便过来工作。

女朋友是七月底到深圳的，来的时候一直是在学校在深圳的就业基地住，她找工作比较晚，8月底才找到工作，在一家互联网公司做网站测试，试用期工资比我 多很多，不过转正之后也高不了多少。在我转正之前，我没钱租房子，让她来我的公司宿舍住也不方便。后来转正后就跟爸妈说了这事，从家里拿了一点钱，就出去 了租了一居室。还好我们都在科技园上班，不用每天坐公车，住的地方离公司很近，每天都是步行过去。

虽然转正了，但是不等于就稳定了。还得继续努力。我依然坚持每天下班后学习（在外面租了房子之后，有时间就回家），更好笑的是，别人情侣在家里基本上都是谈情说爱的，我们却很多时候是在讨论工作，技术方面的。不知道的，还以为我们是同事。

2007年5月，公司接的一个北京医院的很大的医疗项目，派我们的经理和我去北京做实施与项目培训，由于北京医院那边的操作不当，导致机器全部瘫痪， 数据库里面的数据也提不出来。然后把全部责任归咎于我公司这边，还威胁我们不立即解决就停止协议。老板很重视这个项目，五一黄金周本应该是在深圳过的，可 是因为这事儿，我们不得不在北京呆上了十多天。5月中旬才回深圳。

医院的负责人坚持说是我们的软件有问题，与他们的操作失误无关。后来经过我们的检查，双方都有责任，软件存在 BUG 不错，医院的操作人员也存在失误。 于是经理找到医院的项目负责人，说尽了好话，说一定尽快将问题处理，另一方面施压于我。我就在那里，经过十多天的苦战，终于将问题解决了。为公司解除了险 些解约的危险。

回去后，公司为我和经理开了个小小的庆功会，我都感觉有点受宠若惊。我看得出来，经理真的很照顾我，在培养我。那天他跟我说，我理解在北京那段日子的 痛苦，但是你要明白，痛苦是最让人成长的东西，经过医疗项目的那件事，我想让你明白，一个项目真的可以考验一个人的能力，各方面的素质，领导也是从基层做 起来的，没做过开发的人是当不好管理层的。

令我惊喜的是，2007年6月，公司给我涨薪水了，5K。这个时候，从入职到现在，还不到一年的时间。同事都说，不错，有前途。我也想，也许是经理有 意培养我吧，我有个同学，技术还不错，刚进公司的时候就2K，一年之后涨了500，也就2500。一年之后，我的工资比他多一倍。我不敢说这没有运气的成 分。

日子就这样一天一天的过去，直到2008年的端午节前夕，我的职业生涯转入另外一个新阶段：负责一个不大不小的项目，对项目从需求分析到项目实施全面负责，成为项目经理。

其实这个项目的负责人是内部竞岗的，经理选拔几个人，其中包括我，进行竞岗。

说真的，我们几个人在技术上是不相上下的，但是后来被我拿到了项目负责人的位置。究其原因，

经理说我的团队组织能力以及沟通表达能力比其他几个人要好，在一个项目中，项目经理不需要技术牛，关键是与团队成员的沟通能力，组织团队沟通协作的能力，这才是最重要的。

嗯，很感谢在广东硅谷软件学院学习的两年，老师不仅仅是让我学习技术，还有职业素质的锻炼。真的，在职场上打拼，光有技术是不行的，要想往上爬，各方面的素质都需要。如果光技术牛，永远只能做基层的程序员，而光鲜的领导层与你无缘。李开复，唐骏就是很好的例子。

能力得到了老板和领导的认同，这个零售的项目在8月份完成之后，项目很成功，美国那边的客户很满意，公司再一次给我加薪：7K。那些在读书的时候想不到不敢想的数字，如今却成为现实。

2008年的金融风暴席卷全球，很多大公司都纷纷裁员降薪，我们公司也未能幸免，不过幸运的是，我留了下来。那天，经理发邮件点名叫了一批人去会议室 开会，完了之后发现他们每一个人的脸上都很难看。过后才知道，这些开会的人都是要离职的。想想都后怕，给人家打工真的很不容易，公司效益好，老板就高兴， 分你一些残羹剩饭，要是一不高兴，立刻把你开了你也没话说。所以说，当你觉得你没有很大的把握去创业的时候，不如好好做自己的工作，把工作做出色，一步一 步地发展。

重整后的公司团队，基本上都是属于实干型的，那些技术不好的或者领导觉得没什么潜力的都被刷下来了。留下来的人，包括我。都觉得很庆幸，要知道，那个 时候金融风暴闹的每个人都是人心惶惶，生怕自己被炒鱿鱼。于是，我们这些人也更卖命地工作了。由于海外的项目量大大的减少，公司也似乎随着金融风暴进入寒 冬。很佩服老板的魄力，很年轻的一个老板，竟然也带领着我们度过了这个恐怖的寒冬。2009年第二季度，公司的业绩开始复苏，可以说，恢复的比较快，也开 始步入正轨。

2009年9月，女朋友在原来的公司辞职了，原因是我觉得可能我现在待的公司更适合她吧。其实，真正的原因是，他们公司搞的那个 SNS 项目失利，风投 停止追加投资，导致资金链被破坏，公司与网站无法继续运转，不得不做出放弃这个项目的决定。我跟她说，与其被公司炒鱿鱼，不如提前辞职。

嘿嘿，现在我们在同一家公司，我在开发部，她在测试部。同上班，同下班。可是，上班的时候很少说话。真的。

从去年的11月到今年的4月份的半年时间，公司做了两个比较大的项目：一个是无线数据通信平台，另外一个是办公协同。在这两个项目中，我付出了很多的心血，也从中锻炼了很多，包括对项目管理，团队管理等等。都是得到很大的提高。

6年了，从开始学习软件编程的两年到从业四年，一路走来，很充实，有成功的喜悦，也有付出的汗水，感觉学到了很多东西，各方面的都有提高。感谢培养我 的广东硅谷软件学院，感谢老师不仅教了我软件编程的技术，还教会我如何做人，如何在职场发展；感谢我们经理对我的栽培，真的，如果没有他的待见，可能我没 有发展到这个水平。四年来，他都是有意给我锻炼的机会。我们不仅仅是上司与下属的关系，更好友情深厚的忘年交。当然了，我还感谢我的女朋友娜，在我学习和 工作后的支持。娜，谢谢你，谢谢你对我的支持，谢谢你给我的爱，相信我们，我们的未来一定会更加美好！

下面就说几点我的一些心得与建议吧：

1.打好基本功，不要妄想学了一门语言就能做出好软件，那些计算机基础课程还是很重要的，计算机组成原理，算法与数据结构，编译原理等等。不懂得这些，很难写出高质量的程序。

2.勤奋，不管做什么事情都得勤奋。程序是写出来的，不是说出来的。多看书，多看代码，多与别人交流。

3.不要 SB 地与人争论什么语言的好与不好，平台的先进与否。真正的牛人是不在乎什么语言或者平台的。每种语言或者每个平台都有自己的优点与不足，这些都不是选择的决定性因素。在你没发明一种语言之前，你没权利评论别人的语言怎么样怎么样。

4.学习讲究灵活，遇到一个棘手的问题不要死想，多与人沟通。求伯君一个人开发 WPS 的时代已经过去。

5.个人觉得，如果你的课本都没学好，就不要乱买什么参考书了。可以这样说，课本上的东西都是很基本的，如果连基本知识都没有掌握，还想理解其他的？参考书的意义在于课本不能满足你的需求。

6.有空多看看技术类的社区，如 CSDN，51CTO 等等，了解一下行业资讯和技术资讯，不要闭门造车，不要傻逼傻逼的学习一门早已绝对不是主流的语 言自己还不知道，还以为仍然是企业应用的主流技术。你要知道，至少是现在，主流的技术是 JAVA，.DONET 和 PHP 三种。

7.要忠于自己的选择，不要因为程序员的薪水而选择这个职业；那样你将写不出什么好软件，而且不久之后，你会觉得很累，迟早会离开这个行业。而真正喜欢这个行业的程序员是将编程当作自己的快乐之源，而不是收入之源。

8.永远要保持不断学习的精神，IT 技术的更新速度是惊人的，所以我建议，有空的话就多学习学习其他的技术。可能你现在的工作岗位用不到，但是相信我，一定会有用的时候，而且你会因为现在的坚持学习而感到成就感。

9.不要频繁的跳槽，进入一家公司，如果合适的话，就做下去。积累经验不是说在大公司才可以，问题是，如果都去了大公司，那中小公司谁去？不是每个人 都是技术牛人，不是每个人都可以去大公司。有些人总以为，自己跳槽要么就是薪水不高，要么就是没什么发展，可是很少的人会想到自己的技术又怎么样。

10. 更多建议，欢迎与我交流。

最有效的 5 条改进措施

1、分类管理项目

通过区分企业中不同类型的项目制定不同的管理策略、裁剪策略，保证了质量体系的实用性、灵活性，即减少了开发投入又保证了项目的质量，平衡了敏捷方法与规范方法。

有的企业区分了新产品研发、订单项目开发、系统维护等3类项目，又区分了大中小规模的不同，针对不同类不同规模的项目定义了管理的流程、文档模板。

2、用例+界面原型描述需求 use case+ prototype

用例站在用户的角度描述了用户与系统的交互序列，清楚的陈述了系统的功能需求，客户可以很容易的阅读用例描述，便于沟通。测试人员很容易基于用例编写 出系统测试用例。在为四川的两个软件企业进行咨询时导入了 USE CASE 描述需求的方法，开发人员与测试人员均认为提升了需求描述的清晰性、完备性。

界面原型克服了客户、最终用户、开发人员的认识障碍，通过直观的界面能够激发客户与最终用户的详细的操作需求，减少了需求描述的二义性，可以快速的捕 获需求，减少了需求的变更。南京某软件公司在为百盛集团开发一个门店选址系统时，快速构造了1400个界面供客户确认需求，客户确认需求后，发现实际的需 求和最初在合同中约定的需求超出了约50%的工作量，从而双方协商变更合同，追加投资。

3、任务结构分解 WBS

WBS 中识别项目中的所有的任务，并对这些任务进行了归类，可以基于 WBS 进行工作量的估计、责任的分配、项目进度计划的安排，规避了部分估算与计划的风险，帮助项目经理有条理的组织项目的开发活动，避免了混乱。

4、代码审查 code review

代码审查可以快速发现代码中的缺陷，代码评审的效率远高于测试的效率。

客户 A 的度量数据证明在其公司内：

对于算法类模块：代码评审的效率是测试的效率的15.7倍。

对于非算法类模块：代码评审的效率是测试的效率的4.68倍。

客户 B 的度量数据证明在其公司内：

代码审查的缺陷检出率平均为10.6个 bug/KLOC。

代码审查的效率是系统测试的4.25倍。

5、每日构建 daily build

通过每日构建及时发现模块与模块之间的接口问题，结合单元测试可以尽早发现程序中的错误，尽早修复错误，保证了每天都可以提交一个可以使用的软件版本，提高了一次集成的成功率，保证了项目的平稳的开发速度。

解析测试工程师职业瓶颈

经过这几年的发展，国内 IT 公司的测试水平有了很大的提高，但是与此同时，很多测试工程师也迎来了个人的发展瓶颈：很多人从测试工程师做到了测试经理的职位，不知道下一步如何发展；或者每天机械地从事着功能测试工作。

本文首先从分析测试工程师的发展现状和职业化过程遇到的问题入手；接着分析什么样的测试人员才是合格的；最后介绍测试人员的职业历程以及如何突破自己的职业发展瓶颈。

1 测试工程师帖子解析

下面是在一些测试网站上收集的帖子，主要是一些测试工程师介绍自己的成长历程或者对测试行业的看法。从这些帖子中，我们可以看出测试工程师职业发展遇到的一些问题。

帖子一：踏实地发展自己

我在北京工作有4年了。职业发展依次经历了测试员－测试工程师－测试分析师－测试经理。这就是我在北京的4年测试生涯。个人对测试工作有如下的观点：

1）软件测试不像一些人看起来那么简单，需要相当深厚的技术背景。但只要掌握要领，也不像我们一些人所认为的那么困难；

2）测试工程师和开发人员相比，可以有机会接触更多的、不同行业的项目，是一个大的优势。

3）测试工程师要想成功，更多的是靠平时的积累。不管是项目的积累，还是平时学习，两者都至关重要。

4）测试工程师要充分利用网络资源，与同行们充分交流，在互相帮助和学习的氛围中，可以加快自己成长速度。

点评：这是一位比较踏实的测试工程师，一步一个胶印地走着自己的测试之路，我们可以认为他是相对成功的典范。现实中我们很多测试工程师不是抱怨工资 低，就是抱怨自己公司的测试环境不好。如果要想在测试领域走向成功，重要的秘诀就是踏踏实实地学习，认认真真地做好本职工作。

帖子二：执着的测试工程师

我做测试工作快6年了。刚开始的时候，我是公司的第一个测试员，虽然公司也在做 ISO9000，但是什么规范都得自己摸索。可是，我仍然坚持下来了，而且大有收获，虽然在公司里不受重视。

但是随着测试工作的不断深入，自己对公司的主流业务（我们作的是行业软件）从外行变成了内行。而且还发现了一些产品的设计方面的欠缺，在老板和开发主 管面前树立了自己的一些威信。至少在一些项目进行需求分析的时候，会来征求我的意见。而且，目前做到了不经测试的产品不给客户。当然，在我和开发经理发生 分歧的时候，大部分时间老板还是支持开发经理，但我认为是正确的地方还是会据理力争的。

一句话，测试人员是孤独的，寂寞的。但只要坚持，总能有收获的，尤其是在发现了隐藏很深的一些 BUG 的时候，那种成就感和自豪感真的是一种很好的感觉。

实际上，做任何一行工作，都会遇到不公平。但为什么要去跟别人比呢，只要自己有提高，就是好的。

点评：“敢做冷板凳的人”才是勇敢的人，这位发帖子的测试工程师不但有勇气坐了冷板凳，而且能够坚持下来，直到取得了不错的成绩。“实际上，做任何一 行工作，都会遇到不公平。但为什么要去跟别人比呢，只要自己有提高，就是好的。”几句朴实无华的话说出了如何做好测试工作的真谛。

测试人员一定要给自己正确的定位，既然选择了目前地位有些低的测试工作，就应该踏实的做好，这是走向成功的必由之路。

帖子三：好学而有信心的新手

我在一家外企作了一段时间的兼职测试，之前我从未接触过测试。开始的时候只作一些 Manual test，后来就开始做 Automated test，修改原有的 test cases，或者重写一些 test cases。然后 test 小组的 leader 建议我写测试文档，他说写文档有利于一个 tester 技术水平的提高。因为你必须要熟悉软件项目的整体框架， 洞悉软件深层的结构才能写出高质量的测试用例。

于是，我在网上查了一些关于测试方面的资料，发现测试真的很重要。对一个软件项目而言，老外对软件测试尤其重视。我兼职的这个外企是有一半的员工是测 试的，大概有7、8个人。个人认为，国内的软件企业对测试的重视程度还不够，但是毋庸置疑，测试是软件企业产品线上和开发同等重要的。可以预言，未来的国 内软件行业，软件测试人员可能会占据软件团队半数左右。同时，对测试人员的能力要求也是比较高的。

综上，我觉得 Software test 很有前途。当然，这些都是一个测试新手的看法，可能比较片面，全当给大家打打气了。

点评：可以看出这是一位很有远见的测试工程师。现实中很多测试工程师是由于不能从事其它工作才从事测试的，因而工作中也是不断地抱怨待遇、团队环境等 不能满足自己的要求。在此建议测试工程师，如果选择了这个行业，就应该认真地对待工作，抱怨永远解决不了问题。只有像这位测试工程师一样认真分析自己的行 业，才可以有更好的职业化发展，否则还不如换一个自己喜欢的工作去试一试。

帖子四：郁闷而犹豫的测试工程师

我做软件测试也有两年多的时间了，但是在这两年中似乎没有太多的提高。因为都是黑盒测试，所以一般就是使用产品，最多也就是一些工具测试。可是这都是想到哪就测到哪，也没有真正按照测试文档执行。公司测的东西组合情况也很多，根本没办法。而且公司测试流程也不规范。

刚开始没签约的时候，还是专职测试，签了以后简直就变成打杂的了。什么都要做，连一些设计文档都推了过来，有时候还要去现场了解客户需求，真是非常佩 服老板把一个人当三个人用的能力。总的感觉在小公司里，根本就没有测试员这个概念，测试员一般什么都得做。当你提出一堆建议的时候，老板总是会说，现在公 司规模还不具备条件，以后会慢慢的改善的，可我感觉过两年公司可能已经倒闭了。

真是有点郁闷，改行做开发，又不想放弃这个被很看好的职业。可是光被看好又怎么样呢？说不定十几年前，扫大街的就像现在的做测试的。都是做质量保证，扫大街的要保证城市环境的质量，现在呢？

他们又怎么样？

点评：其实测试和开发一样，都不是那么想当然的理想，国内开发环境也和测试一样混乱。而开发人员发展到高级程序员就会成为很多人的发展瓶颈，薪资和职 务都很难再提升。而测试与开发相比的一个巨大优势在于它是一个新兴的领域，拥有更多的机会，测试人员工作三五年，再有一些管理经验，很容易做到测试主管， 虽然薪资待遇相对低些，但是给个人的长期发展打好了基础。看准了就应该去做，实实在在的学到知识才是最重要的。

2 国内软件测试测试工程师职业发展现状

我们可以从两个方面来解析测试测试工程师的职业瓶颈问题：

从企业方面：多数企业较难招聘到满意的测试工程师，尤其在软件测试外包企业，人才问题成了这类企业的发展瓶颈，这些恰恰反映了整个测试行业的发展遇到了瓶颈；

从个人方面：很多测试人员薪资和职位到了一定阶段就很难得到提升，例如很多测试工程师做到测试经理后，几年内得不到提升。

职业发展尤其体现在待遇方面，我们可以看出：

（1） 企业规模越大，越重视测试，而测试人员的待遇也越高；

（2） 掌握测试工具的测试人员待遇往往高于那些只能进行手工测试的工程师；

（3） 测试技术越熟练，待遇越高，而具备一定领导能力的测试工程师待遇会更高些；

但是我们就整个 IT 行业来看，尤其是与开发人员相比，测试工程师的待遇显得很低。就作者掌握的资料来看，同一级别的开发工程师要比测试工程师高1～2K（人民币），甚至更多。

与开发人员相比，测试工程师的职业目标则很少，主要下面几类：

测试组长（也可称之为测试负责人、测试经理）：这类测试人员通常是测试项目负责人，既要具备较高的测试技术能力，还要具备一定的管理能力。主要职责是 制定测试与编写测试计划、监控和管理整个测试过程。测试组长职位之所以受青睐，是因为测试组长可以向上发展为测试部门经理、质量经理，也可以横向发展为项 目经理，因此通常待遇相对高些。

测试分析师：主要职责是对系统的测试结果进行综合的分析，例如缺陷分析、性能分析等。测试分析师不但测试技术能力较强，还要具备数据库、操作系统等多方面的技术知识。这类职务的发展空间也不错，可以发展成系统设计师等。

自动化测试工程师、测试开发工程师：主要职责是编写测试程序、执行自动化测试任务。这类职位的测试人员至少要达到初级程序员的能力，因为经常和程序打 交道。发展空间也不错，甚至可以发展为程序员。（在国外，这类工作多由具有开发背景的测试人员来负责。而国内的 IT 公司不重视测试，程序员不愿意去做待遇 较低的测试工作，因此测试人员在具有一定的开发能力后，将会考虑转行去做开发。）

但是，国内的测试人员多数在测试圈子“打转转”，很难进入到开发领域。多数测试人员发展成测试经理/高级测试工程师后，职业化几乎到了尽头。于是，出现了一些大企业的测试人员自己去创业的情

形，但这种情形更是少之又少。

实际上，这一切的根本原因是由测试工程师的能力水平决定的。国内测试工程师普遍根基不牢，自然难获得较大的发展。下面将从测试工程师的基本素质谈起：只有那些基础知识扎实、潜质较好的测试工程师才是合格的工程师。

3 测试工程师基本素质

很多年轻或者刚刚从事测试工作的工程师，经常会问：“测试工程师需要什么技能或者具有什么素质才是合格的？”与开发人员相比，测试人员不但需要一技之长，还需要掌握诸如操作系统、数据库、网络等多方面的知识。

根据作者多年的经验，一个有竞争力的测试人员要具有下面三个方面的素质：

（1）计算机专业技能

计算机领域的专业技能是测试工程师应该必备的一项素质，是做好测试工作的前提条件。尽管没有任何 IT 背景的人也可以从事测试工作，但是一名要想获得更大发展空间和持久竞争力的测试工程师，计算机专业技能则是必不可少的。计算机专业技能主要包含三个方面：

测试专业技能

现在软件测试已经成为一个很有潜力的专业。要想成为一名优秀的测试工程师，首先应该具有扎实的专业基础，这也是本书的编写目的之一。因此，测试工程师应该努力学习测试专业知识，告别简单的“点击”之类的测试工作，让测试工作以自己的专业知识为依托。

测试专业知识很多，本书内容主要以测试人员应该掌握的基础专业技能为主。测试专业技能涉及的范围很广：既包括黑盒测试、白盒测试、测试用例设计等基础 测试技术，也包括单元测试、功能测试、集成测试、系统测试、性能测试等测试方法，还包括基础的测试流程管理、缺陷管理、自动化测试技术等知识。

软件编程技能

“测试人员是否需要会编程？”可以说是测试人员最常提出的问题之一。实际上，由于在我国开发人员待遇普遍高于测试人员，因此能写代码的几乎都去做开发 了，而很多人则是因为做不了开发或者不能从事其它工作才“被迫”从事测试工作。最终的结果则是很多测试人员只能从事相对简单的功能测试，能力强一点的则可 以借助测试工具进行简单的自动化测试（主要录制、修改、回放测试脚本）。

软件编程技能实际应该是测试人员的必备技能之一，在微软，很多测试人员都拥有多年的开发经验。因此，测试人员要想得到较好的职业发展，必须能够编写程序。只有能给编写程序，才可以胜任诸如单元测试、集成测试、性能测试等难度较大的测试工作。

此外，对软件测试人员的编程技能要求也有别于开发人员：测试人员编写的程序应着眼于运行正确，同时兼顾高效率，尤其体现在与性能测试相关的测试代码编 写上。因此测试人员要具备一定的算法设计能力。依据作者的经验，测试工程师至少应该掌握 Java、C#、C++之类的一门语言以及相应的开发工具。

网络、操作系统、数据库、中间件等知识：

与开发人员相比，测试人员掌握的知识具有“博而不精”的特点，“艺多不压身”是个非常形象的比喻。由于测试中经常需要配置、调试各种测试环境，而且在性能测试中还要对各种系统平台进行分析与调优，因此测试人员需要掌握更多网络、操作系统、数据库等知识。

在网络方面，测试人员应该掌握基本的网络协议以及网络工作原理，尤其要掌握一些网络环境的配置，这些都是测试工作中经常遇到的知识。

操作系统和中间件方面，应该掌握基本的使用以及安装、配置等。例如很多应用系统都是基于 Unix、linux 来运行的，这就要求测试人员掌握基本的操作命令以及相关的工具软件。而 WebLogic、Websphere 等中间件的安装、配置很多时候也需要掌握一些。

数据库知识则是更应该掌握技能，现在的应用系统几乎离不开数据库。因此不但要掌握基本的安装、配置，还要掌握 SQL。测试人员至少应该掌握 Mysql、MS Sqlserver、Oracle 等常见数据库的使用。

作为一名测试人员，尽管不能精通所有的知识，但要想做好测试工作，应该尽可能地去学习更多的与测试工作相关的知识。

（2）行业知识

行业主要指测试人员所在企业涉及的行业领域，例如很多 IT 企业从事石油、电信、银行、电子政务、电子商务等行业领域的产品开发。行业知识即业务知识， 是测试人员做好测试工作的又一个前提条件，只有深入地了解了产品的业务流程，才可以判断出开发人员实现的产品功能是否正确。

很多时候，软件运行起来没有异常，但是功能不一定正确。只有掌握了相关的行业知识，才可以判断出用户的业务需求是否得到了实现。

行业知识与工作经验有一定关系，通过时间即可以完成积累。

（3）个人素养

作为一名优秀的测试工程师，首先要对测试工作有兴趣：测试工作很多时候都是显得有些枯燥的，因此热爱测试工作，才更容易做好测试工作。因此，除了具有前面的专业技能和行业知识外，测试人员应该具有一些基本的个人素养，即下面的“五心”。

专心：主要指测试人员在执行测试任务的时候要专心，不可一心二用。经验表明，高度集中精神不但能够提高效率，还能发现更多的软件缺陷，业绩最棒的往往是团队中做事精力最集中的那些成员。

细心：主要指执行测试工作时候要细心，认真执行测试，不可以忽略一些细节。某些缺陷如果不细心很难发现，例如一些界面的样式、文字等。

耐心：很多测试工作有时候显得非常枯燥，需要很大的耐心才可以做好。如果比较浮躁，就不会做到“专心”和“细心”，这将让很多软件缺陷从你眼前逃过。

责任心：责任心是做好工作必备的素质之一，测试工程师更应该将其发扬光大。如果测试中没有尽到责任，甚至敷衍了事，这将会把测试工作交给用户来完成，很可能引起非常严重的后果。

自信心：自信心是现在多数测试工程师都缺少的一项素质，尤其在面对需要编写测试代码等工作的时候，往往认为自己做不到。要想获得更好的职业发展，测试工程师们应该努力学习，建立能“解决一

切测试问题”的信心。

“五心”只是做好测试工作的基本要求，测试人员应该具有的素质还很多。例如测试人员不但要具有团队合作精神，而且应该学会宽容待人，学会去理解“开发人员”，同时要尊重开发人员的劳动成果——开发出来的产品。

案例：测试人员首先要学会尊重自己

软件测试人员首先应该尊重自己的劳动成果——软件缺陷报告。我见过很多测试人员都不能清晰地描述一个软件缺陷，尤其分不清缺陷跟踪系统中 Summary 和 Description 的区别，例如图2-2中的软件缺陷描述——Summary 和 Description 中就输入了完全一样的内容。

严格的讲，Summary 通常用于概要性地描述软件缺陷内容或者发生问题时的现象，主要用于项目经理进行缺陷分配，因此要用最简短、精悍的语言来描述 是什么缺陷，使项目经理很快明白是什么问题、应该分配给哪个开发人员；而 Description 则用来描述缺陷的详细信息，通常描述缺陷的重现步骤，主要供开发人员修改缺陷时候查看。图2-3就是一个非常规范的软件缺陷描述。

软件缺陷报告是测试人员最直接的劳动成果，因此应该认真地描述自己所提交的每一个软件缺陷，这也是尊重自己劳动成果的一种表现。缺陷描述不清晰，不但 将会增加沟通成本，更重要的是不会得到开发人员的认可与尊重。测试人员在为开发人员的成果——产品找问题的同时，也要保证自己的成果没有问题。

因此，作为测试人员首先要学会清晰、准确地报告一个缺陷，这将是与开发人员互相赢得对方尊重的开端，也是尊重自己的表现。试想，如果自己都不爱惜自己的劳动成果，那别人如何会尊重你的成果呢？

如何应对非功能性需求变更？

辛辛苦苦的熬了几个月，软件开发终于快要告一段落了。系统功能已经基本完成了，在准备按部就班的完成最后的测试时，客户突然提出要改变某些非功能性需 求。这对于软件开发团队来说，不亚于晴天惊雷，这也是让所有软件开发人员感到最恐怖的事情之一。因为在多数情况下，对非功能性需求的变更都会演变成一个对 系统无休止的修改过程，最终会把客户和开发团队都拖进泥潭而难以自拔。

需求变更本应是客户的权力，如果确是需要变更，当然要满足客户需要。但问题是不能让变更权力滥用，把一些无关痛痒的非功能性需求变更宠惯养成堂而皇之 的变更。对于非功能性需求客户总会有新的想法，项目好像总没有办法终结。以前当出现这种情况时，我总觉得很沮丧，觉得自己非常不幸，怎样会碰上这样的客 户。可在读了《设计模式精解（Design Patterns Explained）》一书的一段话后，我恍然大悟，这不是我的错，世界原来就是这样子的啊，永远不变的就是变化。

令人烦恼的非功能性需求变更

在软件开发中，大家都会遇到过这样的问题：客户的一个新想法，就推翻了之前与客户经过再三讨论而确认定下来的需求。如果是功能性需求变更还会让人容易 接受一些，毕竟功能性需求不实现的话，是会大大影响到软件产品的质量。但现在我所负责的这个开发项目中遇到的都是一些非功能性的变更，而且许多是看起来无 关痛痒的、鸡毛蒜皮的变更。

（1）什么是非功能性需求？

在 IEEE 中，软件需求的定义是：用户解决问题或达到目标所需的条件或功能。一般包含业务需求、用户需求、功能需求、行业隐含需求和一些非功能性需 求。业务需求反映了客户对系统、产品高层次的目标要求；功能需求定义了开发人员必须实现的软件功能。所谓非功能性需求，是指为满足用户业务需求而必须具有 除功能需求以外的特性。包括系统性能、可靠性、可维护性、易用性和对技术和对业务适应性等。其中最常见的是软件界面、操作方便等一系列要求。

（2）非功能性需求变更的特点

让我们从客户角度和开发人员角度去看看非功能性需求的特点。首先，有些非功能性小需求从客户角度看起来工作量不大，但是实际上开发人员要耗费比较长的 时间去完成这些小功能。其次，许多非功能性需求，如界面美观、操作方便等都是客户头脑一热、或领导一拍脑袋就部署下去的需求，往往是原来在需求分析阶段所 没有注意的内容。

其实，非功能性需求是常常被轻视，甚至被忽视的。原因是非功能性需求描述很困难，它很难像功能性需求那样，可以通过结构化和量化的词语来描述清楚。在 描述这类需求时候，我们经常采用软件性能要好、操作要方便、软件界面要美观大方等较模糊的描述词语。例如，易用性就同时涉及到美工和 UI 界面、人机工程、 交互式设计、心理学、用户行为模式等内容。这类描述词语都是脱离了软件的执行环境，是对人和相关的场景的描述，因此很难体现到软件架构设计和具体的实现 中。

为什么非功能性需求变更会频繁发生？

为什么非功能性需求不能固定下来呢？或定下来后就不许变了呢？通常有许多人会问这样的问题。实际上，当他变成了客户时，他可能就不会问这个问题了。

（1）非功能性需求容易产生理解分歧

在软件需求分析阶段，客户和开发人员对非功能性需求的理解呈现"大体上共识多，细节上差异多"的特点。一般跟分析员的知识、背景，还有客户表述的标准 程度、双方的交流情况有关。即使通过反复沟通，但是以实践经验来看非功能性需求的描述还是永远不够清晰、不够明确的，主要是因为在这个阶段所谓的产品还只 存在于大家的大脑构思中。

作为一个客户，大多数情况下是不懂技术的，但他所需要的软件在他的心里还是有一个印象的。他会想象出软件的样子和功能，然后通过口头或者笔头的方式告 诉需求分析人员。简单的说，就是在这个阶段用户往往不能确切地定义自己需要什么。用户常常以为自己清楚，但实际上他们提出的需求只是依据当前的工作所需， 或者是他们想象出来的东西。结果是当客户向需求分析人员提出需求的时候，往往是通过自己的想法用自然语言来表达的，这样的表达结果对于真实的需求来说只是 一种描述，甚至只是某个角度的描述，但远远不能保证这样的描述可以得到百分之百的正确理解。

当客户提出要求之后，在双方认为理解大概没有分歧的时候，开发人员就开始工作了。但随着开发工作的不断进展，系统开始展现雏形，客户对系统的了解也逐 步深入。这个时候，客户就会对系统的界面、操作、功能、性能等有一些了解，就有可能提出需求变更要求，而且这些要求很多是基于主观的、人为的因素。总之， 他们了解得越多，新的要求也就会越多。

（2）没有明确的需求变更管理流程

在软件开发中的常识是，一旦发生需求变更不要一味的抱怨，也不要一味地去迎合客户的新需求，而是要管理和控制需求变更。但令人不解的是我们常常看到变 更的提出、讨论和执行常常只停留在口头上。这样做有两个弊端：首先是时间一长，无论是当事人还是开发团队都说不清楚变更是因何发生以及结果怎么样了。显 然，这对于提高项目质量、改进开发过程是很不利的。其次是由于缺乏形式上的约束和对变更代价的定量分析，变更会被非常随意地提出、或被草率地执行，也会大 大影响项目的进展和开发质量。

因此，没有明确的需求变更管理流程，就会使需求变更变得泛滥。因为并不是所有的变更都要修改，也不是所有变更都要立刻修改，需求变更管理的目的是为了 决定什么类型的变更需要修改和什么时候修改。比如界面风格问题，就可以先不修改，或者规划一下修改的时间待到以后进行优化。

（3）没有让客户知道需求变更的代价

对变更的影响没有评估是需求变更泛滥的根本原因。变更都是有代价的，应该要评估变更的代价和要让客户了解需求变更的后果。如果客户不知道需求变更付出的代价，对开发人员的辛苦就会难以体会。

相比于需求开发人员而言，客户可能对需求变更认识不足，认为他们出钱，软件开发团队就要为它服务。因此，客户对需求变更往往会肆无忌弹，将需求变更视 为儿戏，随个人喜好随意变更需求。所以，在和客户接触时应该挑明态度，特别是要让他们清楚需求随意变更所带来的代价和风险。如果客户认为代价太大，那么开 发人员就没有必要及时修改，按原来的进度走，但仍要记录变更，待下一版本在修改。

如何有效控制非功能性需求的变更?

做任何变更之前，我们都要考虑后果。由于非功能性需求变更在开发中所处的重要地位，一旦需求发生变化，影响面是很广的。因此，有效控制非功能性需求频繁变更是一件不容小视的事情。

（1）建立明确的非功能性需求基线

对于软件开发来说，变更无可避免，也无从逃避，只能积极应对。因此，在开发过程中，建立明确的非功能性需求基线是一件重要的事情。如果非功能性需求没 做好，基线范围就含糊不清，就容易被客户抓住空子，往往要付出许多无谓的变更。如果非功能性需求基线做得好，文档清晰且又有客户签字，那么后期客户提出的 非功能性需求变更就会大大减少。因此，在建立需求基线的时候千万不能手软，这并非要刻意针对客户，而是不能让客户养成经常变更的习惯，否则后患无穷。

（2）建立需求变更管理流程

需求变更对软件开发成败有重要影响，既不能一概拒绝客户的变更要求，也不能一味地迁就客户，所以必须要做好需求变更的控制。有句通俗的话说得非常好： 需求变更管理的目的不是控制变更的发生，而是对变更进行管理，确保变更有序进行。需求变更管理流程包括变更申请环节、审批人员、审批事项、审批流程等。

目的有两个：一是将客户下达变更的流程尽可能地规范化，减少张嘴就来的非必要、非紧急、非合理、非高层领导意图的无效变更。二是留下书面依据，为今后 可能的成本核算准备好变更账。因此，凡未履行审批程序的变更，一律是无效变更不予受理。在实践中，人们往往不愿意为小的需求变更去执行正规的需求管理过 程，认为降低了开发效率，浪费了时间。但正是由于这种观念才会使到需求变更逐渐变为不可控，最终导致项目的失败。

（3）确认客户是否接受变更的代价

需求变更作为一个计划外的风险对项目肯定存在冲击，只是大小的差别。而且客户的需求是永远不会满足的，可能一天一个样，为了达到控制频繁的需求变更。 需要将变更后产生的成本进行评估与量化，形成分析报告提交双方领导。否则，一味的妥协只会让项目进一步恶化。因此，要让客户认识到变更都是有代价的，不要 让客户养成随意变更的毛病。一般来说，如果客户认为该非功能性变更是必须的，而不是其上级领导拍脑袋提出的就会接受这些代价。通过与客户的沟通和协商，开 发团队即使没有回报也不会招致客户的埋怨。

（4）加强合同约束力

虽然软件开发合同很难在签订之初就能够精确定义每项需求，单靠合同是帮不上忙的，但也不能忽视合同的约束力。因为有时销售人员为使客户能够快速的签订 合同，往往草率决定和片面同意客户提出的需求。当客户提出新的需求时，销售人员往往一看"应该"只是一个小小的修改，没有太大的影响，可能会直接答应能变 更。所以，在与客户签订开发合同时，可以增加一些相关条款，如限定客户提出需求变更的时间，规定何种情况的变更可以接受、拒绝接受或部分接受，还可以规定 发生需求变更时必须执行变更控制流程等。

（5）加强感情沟通，注意沟通技巧

大多时候单靠合同的约束力是解决不了纷争的。客户着急了就是一句潜台词：做不做，不想做就滚蛋，想做的公司多着呢。例如，有时明明是不合理的要求，可是客户也会狡辩凭什么不给他们做，这可是合同范围内的工作。所以，单看合同是没用的。

那可怎么办呢？通常的做法是通过感情联络，争取客户的同情。我们常常对客户说的一句老生常谈的话，就是提需求也要讲究合情合理，这句话在变更管理中有 着独特的意义。用我们的行话说是：做好需求变更管理控制只是做好了一半的工作，还有一半的工作就是去讲道理，去用心、用感情劝客户回

头。

月有阴晴圆缺，潮涨潮落。变化并不一定总是给我们带来麻烦，有时也会带来惊喜。在软件开发中，对待客户提出的非功能性需求变更，我们需要用平常心去看待，不是一味拒绝，也不要一味答应。

质量管理——软件质量我做主

很多软件公司都设有 QA（Quality Assurance）部门，特别是那些通过了 CMMI 认证的企业。绝大多数企业设立这个部门都是因为 CMMI 体系里面有个 PPQA 这个过程域，需要这么一个角色来“保证”软件的质量。

扪心自问，有多少企业的这个部门又真正起到了保证质量的作用呢？ 关于其中原因，可以看我先前写的【我眼中的 CMMI 】系列。这里谈谈如何化解上个系列中提到的问题。

项目管理的核心其实就是目标管理，从范围、成本、进度到质量无一例外。但是，纵观软件项目经理，绝大部分都只盯着成本和进度的目标。觉得对客户按时把 东西交出去了，对公司在交付前成本控制在预算以内了就算项目做好了。诸不知，绝大部分国内软件项目，交付才是噩梦的开始，交付后的成本会急剧飙升，真正上 线日期会一拖再拖。导致这种状态的一个重要原因就是质量问题，没有质量目标的项目就如同顺水漂流一样，漂到哪里算哪里，运气好的就没事，不好的就翻船。

质量是个软性的概念，不像成本和进度那么直观，所以制定起来会比较难，这也正是诸多软件企业所不愿意投入大量精力去改进的客观原因。我这里先列举一些典型的场景，并附上制定质量目标的方法，供大家参考。（如有未列出的场景，大家可以留言，我会后续附上方法）

场景一：企业已运作稳定，有较多项目已完成。

分支场景1A：如企业在项目实施过程中已收集了规模和缺陷的相关数据，那么就是最容易制定目标的一个分支场景了。在此情形下，我们可以有两种方法来制定目标：

方法1：简单的做法就是选定所需设定的质量目标（选哪些质量目标后续会有说明），将收集到的数据分类，然后求的均值（所有数据的和除以数据个数）和极 值（所有数据中最大和最小的值）。均值可以粗略的表示也完成项目的平均水平，极值可以粗略表示已完成项目的水平上下限。通过这两个值，再结合该项目的实际 情况，即可初步制定出各质量目标。

---- 此方法的优点是计算简单，对数据和数学知识的要求都不高，便于企业开始实施制定质量目标这个活动。 不足是从数学的角度上说数据没有检查正态性，准确度不高。

---- 从我实践的经验来看，纯数学理论的方法不一定适用软件企业，在绝大多数中小软件企业刚开始做这件事的时候，还是很推荐此方法的。

方法2：复杂的做法就是将项目分类，各类项目找到至少6个符合正态性分布的数据，使用统计学的方法计算出均值、极值、方差、置信区间等数据，再进行假设检验，将通过假设检验的数据做为基线来做为制定目标的依据。

---- 此方法的优点是有统计学严谨的方法为基础，从数学角度上保证了数据的精确性。 不足是很少有企业能收集到符合统计学要求的大量有质量的数据，如将垃圾数据混合计算，出来的结果也必将是垃圾，就如丛斌博士经常给我提到 的：Garbage in garbage out。

---- 从我个人的经验来看，此方法不太适合中小型软件企业，就算是大型软件企业，如果度量工作不够严谨也不会得出准确的结果。而且还对数学知识有较高的要求，计算的复杂度和时间消耗也很大。实施起来还不如上一个方法。

分支场景1B：如果企业在实施项目过程中没有积累数据，同时也没有办法通过复盘等方法来找回数据，那么可以使用专家法（也称为 DELPHI 法）来制定，具体操作如下：

操作1：如果企业的人数不多，则将项目经理集中在一起，对选定的质量目标项根据各自经验估算出一个最高值、一个最低值、一个最有可能值，并且附上估算 各值的约束条件。将每个人的约束条件列举出来，通过讨论统一约束条件后，通过公式： （最高值+最低值+4*最有可能值）/6，即可计算出某一个人的目标估算值。再求所有估算值的均值和极值，即可得出目标值。

---- 此方法是较常用的方法，毕竟不是每个企业一开始就会关注数据积累这个活动，但是没数据不代表不能制定目标，起步总是艰难的，起步后再逐步收集数据也是可行的。

---- 以上的专家法与很多教科书上的专家法有一点区别，那就是要统一约束条件后再估算，而且要不受互相影响的独立估算。这样的估算值会更准确一些，这也是我在日常的项目管理活动中自己总结出

来的。

操作2：如果企业的人数较多，则可以将项目经理、组长、测试经理分成3组，每组按照以上方法计算估算值，然后再求总的均值和极值。

---- 此方法从不同角色的人员来估算会有更全面的考虑，数据会更准确。但需要较多人员投入，组织和协调工作量较大。

场景二：企业运作不稳定，没有太多的已完结项目。

分支场景2-A：公司刚起步，没有成功的项目经验。在此情况下，可以参考业界的相关数据，一般可以去中国质量协会、中国生产力促进中心协会、美国软件协会的官方网站上查询到。或者，找当地可以做 CMMI 咨询认证的企业索取，一般的费用在1000元以内。

---- 此方法的优点是资金和人力投入较小，可快速开展质量管理活动。不足是官方公布的数据一般都比较滞后，咨询公司的数据一般以国外为主。

---- 以我个人的经历觉得，虽然这类数据不是非常的准确，但是对于刚起步的软件企业也想开展质量目标管理的活动来说，还是很有参考价值的。数据的精度不是问题， 主要是需要一个可供参考的依据来启动质量目标管理活动，后期再制定相关的流程来收集所需数据，根据前一个大场景的方法来逐步完善质量管理活动。

分支场景2-B：公司已运作一段时间，但是没有较多的成功项目。在此情况下，如什么数据也没有或找不到，则参考上一个方法进行；如有之前的相关数据，则还是可以使用之前的数据做为参考。但是增加相关的排错活动。

之前的不成功项目的数据不能直接使用，需要找到不成功的关键问题点。现在用的比较多的方法就是鱼骨图、3W1H、5W1H 这些方法。从我个人的体会来 看，3W1H 比较适合此场景。通过对不成功的结果来提出 What、Why、Who、How 来找到到引发问题的根本原因，在这个过程中，需要借鉴日本企业的 砂锅法一起使用。那就是，通过3W1H 找到问题的第一层原因后不能放弃，一直使用此方法再次寻找，直至无法回答为止，最后的答案才是真正的原因。也就是打 破沙锅问到底的方法。

通过这种方法，找到不成功的真因后，再结合前一篇提到的我修改过的专家法针对不成功的数据大家来进行目标值估算，这样结合起来使用，制定出来的目标可靠度是比较高的。

---- 此种方法的优点是无需投入资金，通过自身的数据和问题分析来制定目标。不足是需要结合几种方法找到真因来调整不成功的数据，对人员素质有较高要求。

---- 我个人认为这种方法是在其它方法都不可行的情况下没有办法的办法。毕竟几个方法结合使用，还要调整不成功数据对人员的要求还是较高的。当然，如果通过此方 法来培养核心员工的整体工作素质也是一个非常棒的事情，当这些人都接受了这样的思维并运用到日常工作中，其作用远远不仅限与质量管理。

目前我自己的经历中就这些场景，先与大家分享，大家有其它场景可以直接给我发评论或者留言，我会及时与大家探讨找到更佳实践的。

以上根据场景描述了如何设定质量目标的方法，但是设定哪些质量目标比较合适呢？以什么度量单位来设定呢？

一谈到质量目标，很多人第一反应就是“交付后 N 个月内的代码缺陷率

但是，这里有几个问题需要更深入的思考一下：

问题1：假设一个项目的 n=0.05，另一个项目的 n=0.1 。那么，从数据上看，更小的那个项目应该品质更高，但是很有可能觉得品质好的项目会接受到更多的客户投诉。

毕竟，缺陷除了用数量来衡量，还需要用严重程度来衡量的，对于客户来说，他们是不管你的 n 等于多少的，他关注的是验收时的使用体验。当 n=0.05的 项目出现了1-2个重大缺陷严重影响了客户的验收使用，那么客户会对此项目留下非常差的印象；而 n=0.1的项目如果都是一些很小的缺陷基本不影响客户正 常验收使用，那么客户也是可以接受的。

所以，如果我们要制定交付后的品质目标，那么这么描述应该会比较明确一些“交付后 N 个月的免责维护期内，在没有1级和2级缺陷的前提下，代码的缺陷率 小于等于 n BUG/KLOC”。（假设我们对缺陷已经分级，1级为严重缺陷，2级为重大缺陷，3级为一般缺陷，4级为轻微缺陷）

这样的描述，明确的界定了缺陷数据采集的时间段、缺陷数据采集的类型以及最终的目标。同类项

目会有更多的可比性。

问题2：现在从事软件行业的人都知道缺陷的单位是“BUG/KLOC”，中文表达就是“每千行代码的缺陷数”。缺陷的数量有可比性，但是这千行代码确不是那么容易可比的。

场景 A：同一种语言，复用和不复用，代码的规模是有很大差异的。难道复用了1000行代码，自己编写了1000行代码，整个的规模要算2000行吗？ 这和自己编写2000行代码的项目规模是一样的吗？

场景 B：同一种语言，重构和不重构，也会存在很大的区别。一个项目有2000行代码，里面有1000行是复制粘贴的代码，其规模是2000行还是1000行呢？

场景 C：不同的语言，在避免场景 A 和 B 的情况下，其规模是不具有可比性的。为了完成同一个功能，高级语言和低级语言的行数有量级的差异，不同语言实现起来也有较大的差异。这些项目的缺陷率该如何横向对比呢？

场景 D：同一个项目包含多语言实现，那么总体的规模行数该如何描述呢？（现有的 Web 项目中此场景很常见）

看起来很简单的 KLOC 在场景 A-D 中都对我们产生了极大的挑战。我们希望目标数据能衡量项目交付后的品质，但是规模单位没有明确定义时，其数据基本上是没有太大意义的。

要解决这个问题，可以使用两种方法来完成：

方法一：在有良好度量制度和文化的公司，可以针对以上场景，挑选足够数量的项目和人员，通过代码行和工时的统计分析，得到不同语言间的规模比例关系，形成基线，供其它项目使用。但是这种公司目前看起来极少极少，所以大部分公司还是采用方法二比较合适。

方法二：方法一的研究在国外已经进行了几十年，有专门的大学和专门的团队持续跟踪研究，并且也发布其研究结果。他们将不同语言换算成一个标准代码行， 称为 SLOC，来统一衡量不同语言的项目规模。他们的数据也许没有方法一中那么精确，但至少可以为我们提供一个统一的方法来实现度量单位的统一，对于中小 型软件企业还是很少实用的。此方法可以解决场景 C 和场景 D，下面我列出一些常用的语言的换算表供大家参考：语言

比例

ASP

1.25

C

0.48

C++

1.10

Dos Batch File

0.63

Java

1.52

JavaScript

1.86

JSP

1.36

SQL

3.64

例：现计算出 JSP 代码1000行，JAVA 代码500行，换算成 SLOC=1000*1.36+500*1.52=2120。

场景 A 需要配置管理工具协助完成，场景 B 需要重复代码检查工具协助完成。（工具相关内容，我会单独编写一个系列，敬请大家关注。）

综上所述，我们对原来 KLOC 的定义需要修改为“SKLOC”，同时要规定全部为自行编写/修改的代码且不存在复制粘贴的代码。这样即可统一规模的度 量，使不同语言的项目具有横向可比性。当然，自己有能力的企业，还是可以尝试方法一来计算出自己的代码比例关系，这样会更准确一些。

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。

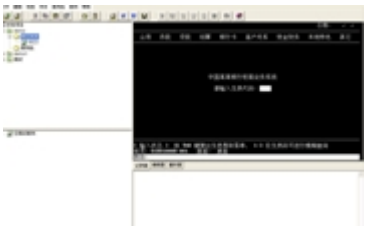
在线体验		产品租用	
企业版	免费版	在线申请	详情

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