泽众软件测试网
电子期刊

SPASVO

第十九期
2013年7月

上海泽众软件电子期刊

2013 年 7 月 第十九期

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

谈谈高科技行业工程师发展的尴尬

经济危机好像一头出笼的野兽，不管有没有被他咬伤，都看到了老虎的利害，一贯流动性很大的 IT 工程师们也都小心翼翼起来，本来计划好的涨薪计划，跳槽计划 也都一一泡汤，同时，寄生于这些外企的猎头公司，仿佛似乎也如履薄冰，公司的萧落，死亡，似乎都有点突然。

通信业最先倒掉的是北电，接着是 moto 的手机部门，这些由管理以及公司战略引起的问题在经济危的波澜中一触即发，终于到了危如累卵的地步，UT 也裁掉了鸡肋一般的小灵通部门，intel，EMC，Cisco 等巨头也都实行了人才紧缩计划，连华为都捡起了多年没有实行的末位淘汰制， 最惨烈的当属前几年异常火爆的 IC 业，中小的 IC 芯片公司更是难以生存，哀鸿遍野。

纵观裁员的公司，被裁掉的人可以分为几类

1. 消费电子类技术人员。相关的产业链都有波及，包括消费电子类的 IC 工程师，通信公司里的手机开发都是比较危险的部门，消费电子类在人们都愿意消费的时候，是很容易产生利润的，可当人们把钱紧紧捂在口袋里，加上资金链断裂，这一连串的行业链就开始如多米诺骨牌一样顺势而倒，尤其是这类的小公司，创业容易，倒闭也容易。

2. 相对来说比较容易招人的部门，比如售后服务，技术支持，销售人员，这类职位是比较容易收缩或者扩张的部门，当然，业绩好的销售人员不用说，仍然是公司保留的重点保护对象。

3. 公司里比较鸡肋的部门，或者即将要淘汰的业务，或者是由某个巨头并购的一些小公司，在公司巨大的资金的风险下，这些部分都是最可能要被剥落的。

4. 身居不高不低的领导阶层，尤其对于那些内部管理繁冗累赘的公司，据说某个大公司今年将计划裁掉上千只做计划和协调的一些职位，比如专门做项目计划的 PL，PM 等等。

出现危机的时候，需求总是从高到低一层层的被抛弃，跟生存越接近的行业就越稳定，相反，风险就越大，你的能力越具备不可替代性，就越稳定。

所谓顺势而为，不是没有人不看好颓势中的机会，中国人多而便宜，勤快，有点耽于安逸，外企又带着一贯的光环，是个好去处，老外明白，只要多开点钱，当然这点钱比国外的工程师便宜好几倍，所以趁着这个局面，也是个借机转移人力的一个机会，比如诺基亚西门子，便裁掉了大量北欧高成本地区的人才，在杭州和上海纷纷扩张。除了这些巨头，还有些小公司企图在这经济的谷底吸金吸人，逆势而上，获得自己的一份立足之地。

历史总是符合历史发展规律的，就如同这个野兽终将受制，抛开大环境的暂时影响不说，对于大多数从事高科技产业的工程师来说，个人的职业命运又该如何发展？

一个 IT 工程师在刚毕业后的三年，还做同样的工作，工作的兴奋度就基本上没有了，每天按部就班，累也不算累， 做着雷同的工作，这时候，他开始思索起自己的职业路线，似乎只有跳槽才能给自己带来一些工作的快乐。其实跳槽除了能给你带来薪水的一点涨幅之外，更多时候就是一种幻觉，很少人能因为跳槽而让你从一个工程师变成一个 leader，从一个 leader 变成一个 manager.除非是从大公司到一个创业型小公司。而你一旦从大公司到小公司，你的正统的职业规划就被打破了，再想回大公司就难

了。而对于大多数的工程师来说，他们更多的只是维持现状罢了，其实很多人心里充满着迷茫，困倦，而随着时光的流逝，我们发现自己已经衰老了，即使有很多想法，也是力不从心了。

而在一个管理技术都规范成熟的大公司，机会是很少的，尤其是在外企，老外宁愿把更高级别的职位给台湾人香港人，都不愿意给一个大陆人，在思科这种公司，做到一个高级经理 已经算是很高的职位了。所以，对于大多数的 IT 工程师来说，所谓个人的职业规划在中国的今天，更多只是一个空洞的口号而已，大多数人为了生活而磨尽了棱角。所有的企业实体都是以利益为核心，他并不是为任何一个人的职业发展而存在，而对于个体在这个大环境下的职业命运，只有我们自己才能把握。

对于一个工程师来说，做到一个 team lead 再到部门经理，是条最传统的职业路线了，或者干几年再平级跳一次槽，其实对于在成熟的跨国公司发展的工程师来说，这也就够了，四，五十万的薪水已经很不错了。进取心强，有足够的技术，更重要的是有足够的情商，慢慢滴熬，时间到了，走到这一步也不是很难，这就够了。

其实更多剩下的是那些已经磨平了棱角了的工程师，对技术的追求也已经变的模糊，隔几年跳一次槽，加几次工资，其实职业大部分也就是生存的手段而已，对于一般的工程师，拿到二十多万的薪水就已经可以进行不错的白领生活了，过多的抱怨其实没有意义，如果不承受创业的风险，这样的收入我们应该可以满足了，当然也不得不满足。

职场上的人们很容易被他们的工作对象所感染，比如面对着机器，人的特征就会变得跟机器有些相似，规范，按部就班，不够灵动，如果你的工作对象是人，你会明显发现这个人在气质上则灵活很多。所以很多整天面对着机器的偏内向的工程师，则需要多弥补跟人群的接触，跟生活化的软性元素的接触。

很多时候，当一切趋于稳定，也离乏味不远了，但但其实回过头来想，我们的工作不就是为了生活吗？生活是什么？在健康的基础上描绘出更多的色彩，当眼前的一切如此现实，我们要更多的去关注自身的内心，定期去做做运动，去参加一些的兴趣小组，定一些可以实现的小小愿望，给自己一些安全却又值得期待的色彩。

生存总是第一位的，目前中国的现状是，毕业的大学生为了生存而找工作，很多毕业生并没有结合过自己的性格，特点去选择一个合适的职业。甚至当等待过了两三年之后，明白了更多的职业特点，才发现，中国那么大地方，那么多公司，转个行却是难上加难，对于某些人来说，工作几年后，如果轻易放弃自己目前的方向，再转行，薪水可能就会打折，仿佛重头再来，这是很难让人接受的，这是需要勇气的，即使在一个行业之内，比如在分工如此细密的高科技领域，更是如此，很多人都认为，差不多的行业都可以换来换去，我的学习能力很强，但对于公司来说，他们更希望找到在该领域有匹配经验的人，而且不要培训，直接就能上岗，直接就可以创造价值，公司根本不怕找不到合适的人，他为什么要给你机会，还要给你培训，对于跳槽如此频繁的今天，他们还会担心在培训完之后你就很快跳槽，得不偿失。

若目前职业和兴趣是南辕北辙的，这种尴尬就会加倍。不知道每天再埋头苦干的那些工程师们，有多少人还曾回忆过自己年轻时候的梦想？在今天，我们每个月固定盼着那些薪水的发放，关心着房价，规划着自己的车子，房子，这个世界如此的现实理性并且无奈，我们被莫名的规则所束缚，被送上了大多人必须遵守的轨道，而关于梦想的话题，却很少有人再提起了。

其实我一直认为，兴趣作为职业，这是可遇不可求的，而一旦你选择兴趣作为基础的职业之后，你也会发现，被规范的兴趣 也变得索然无味，所以现代文明里，我们必须妥协一部分的内心需求，而去找寻那个更为适中的平衡点。

对于刚毕业的人来说，我奉劝你一定要去做做职业测评，在中国，选择一个正确的起点比一个好的专业还有意义，因为一旦你到了某个不高不低的高度，再转行要考虑的因素是如此之多。在找工作前要

好好的考虑一下自己的性格特征，爱好，全方面的去了解自己----很多人都以为了解自己是件很容易的事情，其实并非易事。我建议大家可以去做做人格测试，心理测试，或者你甚至可以从星座，血型去了解自己的优势劣势，刚毕业之后低点薪水都没关系，关键是要选择合适自己的路，简单点说，更多人不甚了解各类公司的特点，也不甚明白公司里那些大大小小职位的分类及特点，也不知道哪类行业更有发展优势，或者即将被淘汰，比如即使一个说起来简单的软件部门，可能就会分配你去做界面，平台，中间件，驱动，内核等等。。。即使是细小的分类也可能对你将来发展有很大的影响。在学校我们学到说的东西如此之多，却如此苍白，只能说明我们的大学教育和就业之间缺乏最基本的交流。

若要找寻更多的原因，需要往一个人之前的教育回朔，甚至牵涉到专业的选择，家庭的教育，这是个太过于复杂的体系了，只是希望更多的人没有太多的“如果”。人生到某个点开始清醒明白的时候，我们已经没有如果了，但能做的还是要做，我们要的是满满的，丰富的，快乐的人生。

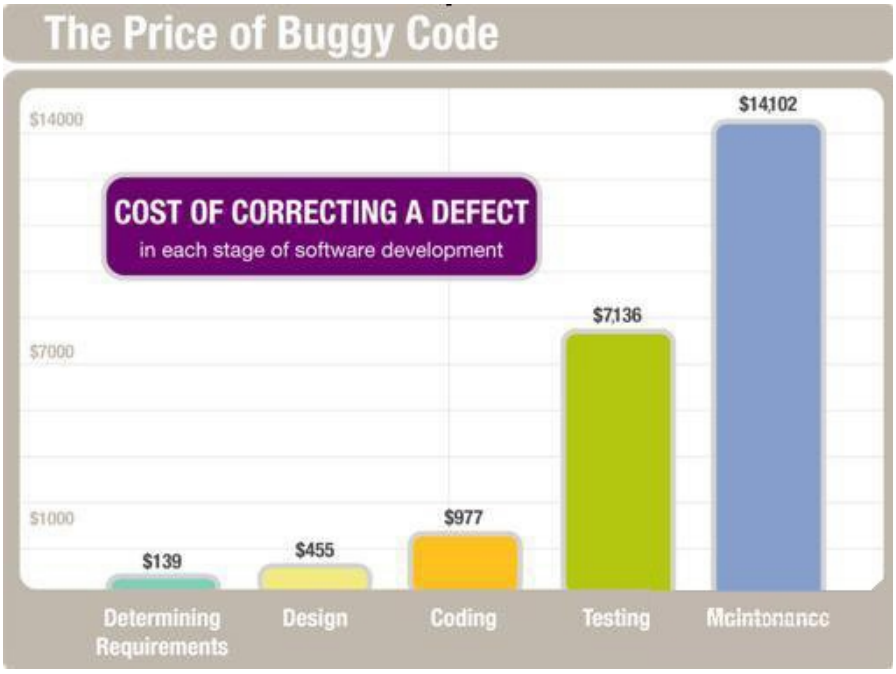
有效利用白盒工具提高代码质量

代码质量是在软件满足了设计功能的前提下，对软件代码执行的可靠性、稳定性和高性能的一种更高的要求。如何能够有效提高代码质量，又节约程序员查找和 修复 bug 的时间，成了一个难题。白盒测试工具的引入，恰好解决了这一难题。本文将通过多个专题，阐述程序员如何有效利用白盒测试工具，提高代码质量和开 发效率。

提高代码质量的巨大经济收益

从提高软件投资回报的角度出发，企业应在降低开发成本的同时，提高软件的可用性，这就意味着尽量减少应用程序中的 bug 和性能缺陷。而由于代码编写过 程的人为因素，代码中的 bug 是不可避免的。据统计，每千行软件代码中，就可能存在20到30个 bug。在无法避免 bug 产生的情况下，如何发现并及时消 除这些 bug，就成了提高软件投资回报的唯一可行办法。

首先，我们来看看修复一个 bug 所需要的成本。在软件交付周期的不同阶段，修复一个 bug 所需的成本差别非常之大。越是到了软件交付的后期，修复 bug 越困难，成本也就越高。从图1可以看出，在测试阶段修复 bug 的代价是开发阶段的几倍，而一旦产品上线，进入维护期后，所需的代价更是达到几十倍。



根据国际上的经验，在测试阶段修复一个 bug 的时间往往要比在代码编写阶段要多出15到75倍。

因此，越早发现 bug，越早解决，企业所付出的代价就越小。我们建议在软件进入测试阶段之前就解决掉大部分的 bug，即交付高质量代码的意义所在。

提高代码质量的几个关注点

代码质量是在程序满足了设计功能的前提下，对软件代码执行的可靠性、稳定性和高性能的一种更高的要求。为了保证交付给测试人员的代码能够满足上述质量方面的要求，在开发阶段应该对代码进行

如下方面的测试：

1.静态代码分析

为保正团队内各程序员之间编写习惯的一致性和规范性，消除容易导致错误的语法隐患，通常会制定一系列的编程规范。静态代码分析，即要求程序员对所有代码行进行规范性检查，消灭潜在隐患，提高代码的可靠性。

静态代码分析的好处在于能发现大量潜在的软件缺陷。比如下面一段 VB .NET 代码中，使用 Asc()函数返回 ASCII 码。

If (Asc(mChar) <> 13) And (Asc(mChar) <> 10) Then

从语法上分析，这段代码是完全正确的。但若 mChar 为空，Asc()就会抛出 ArgumentException 错误，导致程序异常或中断。因此，从代码可靠性的角度出发，在使用 Asc()之前，应对引用的字符串作判断，如下：

If Len(strchars) > 0 And strchars <> nothing Then Ascii = Asc(strchars)

这样，就可以避免潜在的异常错误。而类似的这样问题，IDE 的语法检查通常没有办法，只有通过手工或自动化的代码检查来发现和解决。

没有经过静态分析的程序可能跑起来不错，而实际上可能像一块奶酪一样，满身是漏洞，问题一出现就很严重而且直接面对问题的是可能是最终用户。

2、代码覆盖率分析

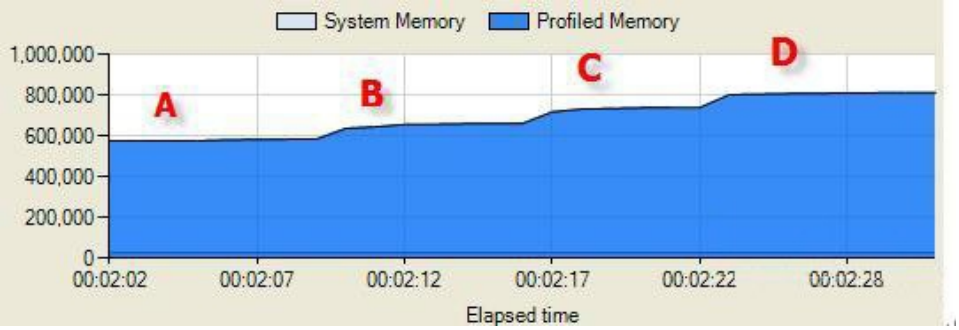
只有经过充分测试的代码，质量才是有保障的。而程序员在做单元测试时，往往很难遍历所有的分支情况，尤其是程序对错误输入的反应往往被忽略。因此，对 代码的测试覆盖率分析是保证交付高质量代码的关键。利用一些代码覆盖率检查工具，不仅可以了解代码的整体测试水平，也能够指出未经过测试的代码，给程序员 以方向上的指引。

3、运行时内存分析

内存往往是导致严重性能故障的根本原因，而一些不良的编程习惯经常会导致运行时的内存泄露。这类问题几乎无法通过单元测试或功能验证来发现，而又经常 被程序员忽视，直到软件投入生产后才逐渐暴露出来，这时再查找和修复内存问题就非常困难了。因此，应尽量在代码交付之前消灭内存问题。运行时内存分析，即 在代码调试阶段，对所有对象的内存占用状态的运行时分析，查找内存泄露的隐患。

运行时内存分析的第一步就是内存使用的监控，以便了解在运行期间，程序对内存的使用和释放情况，查找程序对内存的不当使用。

如下图所示，在 A 点位置，程序稳定运行，对内存的使用也基本稳定。在 B 点，我们打开程序的某个窗口时，程序使用了更多的内存来储存新的对象。当窗口关 闭时，这些对象被释放，内存也应随之释放。而从图上 B 点之后的内存使用情况来看，这些内存并没有被释放。因而，可能存在着内存泄露。在 C 点和 D 点再次打开 前面的窗口，内存还是不断增加而没有回收。可以断定，该窗口对应的代码存在内存泄露的问题。



虽然无论是 Java 还是.Net，都提供了自动的内存回收机制，但内存泄露仍是引起应用系统性能劣化的一个主要原因。除此之外，运行时的内存分析还应深入的对每个对象、方法的内存占用进行分析，以便为内存使用的优化提供方向。

4、性能分析和优化

代码的执行效率，直接影响着应用程序的可用性和可靠性。因此，软件的性能问题应该在开发阶段就充分加以考虑，提高代码的执行效率，而不是把性能问题全部留到压力测试阶段去解决。

性能分析和优化，即要求在代码提交之间，对每一个功能实现的响应时间以及每个方法的效率进行分析，并对运行效率较低的代码进行优化，从而提高代码的整体性能，保证交付高质量的代码。

程序员总是被要求优化某段代码，缩短执行时间，但这并不是一件容易的事情。困难就在于，代码执行是一个非常复杂的过程，包含了太多的分支和无数的方 法、代码行，往往让程序员无从着手。这就需要借助一些动态的代码分析工具，帮助程序员了解每个方法及代码行的执行效率，从而有针对性地对那些执行时间最长 的方法或代码行进行性能改进。动态代码分析工具应能够提供诸如运行期间调用了哪些文件、方法、代码行，每个文件、方法和代码行的执行时间、对总体运行时间 的影响程度等等。有了这些信息，程序员就能够找出影响执行时间的关键路径，有效改进代码性能。

5、 线程分析、错误检测

在 Java 编程中，线程是一个非常好的技术，它可以让程序更加灵活 更加强大。但是，线程技术的误用 同样也会带来非常严重的问题，而且线程问题是最难定位和修复的问题之一。在代码交付之前，程序员应充分分析和判断代码执行过程中是否存在线程死锁以及代码 在什么位置使用了锁和同步机制等等，避免将线程问题拖延到系统测试阶段。与性能分析类似，线程的死锁很难通过手工的方式去判断，必须日借助动态的代码分析 工具，了解线程间的调用次序、同步机制以及判断死锁。

告别“刀耕火种”的手工时代

随着软件应用环境的日趋复杂，对软件质量的要求越来越高。而随着敏捷式开发等新的开发方式的出现，开发的周期越来越短。显然，再依靠过去人工的方式逐 行 Review 代码、统计测试覆盖率、查找 bug 等，实在是力不从心。幸运的是，越来越强大的白盒测试工具的出现，弥补了这一点人力的不足。

自动化白盒工具的引入

目前市场上存在着多种多样的白盒测试工具，有的是只为某项测试而设计的开源软件，如仅限于代码覆盖率的检查，也有功能强大、覆盖面广的商业套装软件， 如 Compuware 的 DevPartner 系列、IBM 的 Rational 系列工具等。笔者在这里简单介绍几款主流工具，供有兴趣的读者参考。

1、Compuware DevPartner 工具。

虽然 Compuware 在国内的知名度还不是很高，但在北美市场，其白盒测试工具 DevPartner 却是声名远播、屡屡获奖，在最近的 Visual Studio 杂志的2008年工具评选中，其 DevPartner Studio 再次赢得了最受读者青睐奖 。

DevPartner 主要有面向.Net 和 Java 两个版本，提供代码覆盖率统计、静态代码检查、内存分析和性能分析等多种分析手段，使用相对简单，可以与 Visual Studio 或 Eclipse 等开发工具 IDE 集成，但暂时还没有中文版本。

2、IBM Rational 工具集

IBM 的 Rational 系列也是一套为广大读者熟知的白盒工具集，如用于覆盖率检查的 PureCoverage 和内存检查的 Purity。Rational 系列工具对使用人员的专业技能要求较高，要想把这个工具用好，要求每个成员至少有两年的使用经验。

3、C++ Test

C++ TesT 是 ParASoft 公司出品的一个针对 C/C++源代码进行自动化单元测试的工具，支持白盒测试、黑盒测试以及回归功测试。C++ Test 对于简单的静态代码分析和边界测试来说，是一款非常不错的工具，但由于自身性能问题，笔者认为不适合用于大型项目的开发。

如何挑选白盒测试工具工欲善其事，必先利其器。挑选一款合适的白盒测试工具，能够有效的提高代码质量和节约开发人员的时间。下面笔者就结合自己所知，尝试解读挑选白盒测试工具的几个考虑因素。

1.功能因素

首先，应考虑工具支持的语言和平台。目前主流测试工具支持的开发语言包括.Net 和 Java，针对不同的语言，实现方式一般有较大差异。其次，选择适 合的功能。并非功能越丰富越好，而是应该选择适用于自己应用场景和测试要求的工具。若对软件质量的要求较为严格，如军工、航天软件，则需要更多、更强的的 白盒测试手段支持。若一般的自用或简单商用软件，则可以考虑某些功能简单又有较强针对性的工具，甚至是开源工具。再次，要考虑工具的集成能力。那些提供与 开发环境向集成的白盒测试工具，能够避免程序员在开发和调试阶段在多个窗口之间频繁切换，节约宝贵的时间。

2、价格因素

任何一个软件的开发，都必须考虑成本的压力。对功能的要求越高，相应的工具软件肯定越昂贵，例如 Rational 就不是每个企业都负担得起的。

3、易用性

在选择白盒工具时，还要考虑到使用人员的技术素养以及工具的易用性。某些工具虽然功能强大，但易用性不高，需要较长时间摸索学习。一般，提供网上下载试用的工具会有比较好的易用性，如 Compuware 的 DevPartner。

如何有效利用白盒工具提高代码质量

为了让读者对白盒工具的使用有一个直观、深入的认识，本文将分为四个专题，选择白盒测试的典型应用，阐述如何利用白盒测试工具提高代码质量。

1.静态代码分析(JAVA)

2.运行时内存分析(JAVA)

3.性能优化及分析(JAVA)

4.运行时内存分析、死锁检测以及测试覆盖率分析(JAVA)

测试工程师职业发展规划

1－2年，测试技能：熟悉整个测试过程及产品业务领域，学习和掌握自动测试工具，学习测试自动化编程技术；开发和执行测试脚本，承担系统测试实施任务；掌握编程语言、操作系统、网络与数据库方面的技能。（转自51esting 软件测试网）

3－4年，测试过程：深入了解测试过程，掌握测试过程设计及改进，参与软件工作产品的同行评审；进一步了解产品业务领域，改进测试自动化编程技术；能指导初级测试工程师；加强编程语言、操作系统、网络及数据库方面的技能。

4－5年，测试组织工作：管理1～3名测试工程师，担任任务估算、管理及进度控制；进一步培养在软件项目管理及支持工具方面的技能。

5－6年，技术管理：管理4～8名测试工程师，提高任务估算、管理及进度控制能力，完成测试规划及支持工具的技能；用大量时间为其他测试工程师提供技术及过程方面的指导；开始与客户打交道并做演示推界。

6～12年，测试管理：管理8名以上测试工程师，负责一个或多个项目的测试工作，与客户打交道并做演示推界；保持使用项目管理及支持工具的技能。

让我们忘记"敏捷"这个词吧

经常听到很多敏捷实践者的问题：

什么才是真敏捷？

我们就是山寨敏捷，为什么不用 TDD？

你持续集成了吗？

我们站立会议超过了30分钟，有关系吗？

Scrum 就是一些管理实践，不用上 TDD，持续集成这些技术实践，敏捷有啥用阿？

要不要结对阿？

问这些愚蠢的问题，就像问是不是用筷子吃饭，才叫真正的吃饭。而真正的问题是我饿了，我手抓羊肉，吃面包，喝水，甚至去打麻将，让我忘记饿的感觉，都是解决办法。

我写过这样一篇文章，《敏捷迷雾背后的本质》 ，关于这个，我还是想罗嗦几句。

把事情作对的方式很多，敏捷实践是其中的一种方式，它一方面带来了很多最佳实践，另一方面是一种新的思维方式。

关注问题，关注组织中的浪费，寻求方法进行改进。将敏捷实践看成做出这些改进，可以参考的集合。

如果一味的强调敏捷，就好像寻求银弹一样，在你还不清楚自己存在什么问题的时候，试图去寻找一个解决所有目前和预想未来可能存在问题的工具。

这，是不靠谱的。

敏捷只是一个代名词而已。

我们可以推演一下敏捷中的实践，它背后隐藏着的实际问题是什么。

1）站立式会议

这个主要是沟通问题，沟通永远不够。在固定的时间，固定的地点，交流固定的主题，就是为了沟通信息，从而事情的推进能够自然，紧凑的方式进行。

2）编写测试

用什么方式保证你写的代码没问题。答案可能有很多，可以依赖功能测试，依赖 Code Review，依赖你的细心。。。。，最终的目的，是使你的代码正确表达它的意思。如何验证它的正确？是“我觉得”，是“应该。。。”，靠主观是不能解决 问题的。就像毒奶粉，靠用味觉品尝，靠用眼睛看，这些都不是能确

定奶粉是否合格的方式。至于测试先行，还是测试后行，我觉得并不重要了，可以当作哲学问题 来讨论，关键是最后你的测试，是否验证你的代码正确的表达了它的意图和逻辑。

3）团队计划会议

如果每个项目经理，都熟悉具体工作任务和优先级，没什么问题。如果不是这样，大部分项目经理独立做出来的计划，是拍脑袋的，包括任务划分和优先级都是 欠合理的。 这时，就可以依赖团队的力量，一起做计划，将计划变得更加合理。只有开发人员自己才知道，具体细粒度的开发任务如何安排更加合理。

4）看板

看板并不是敏捷实践独有的，我在以前的博客说过，越狱中，大家可以看到大量应用看板的实际例子。看板，解决的是公共信息沟通的问题。没有它，很多项目组成员公共的信息，需要通过很多点对点的沟通来弥补。沟通的效率会非常低。我们的群公告也可以起这个作用。

5）用户故事

为什么选择用户故事作为需求描述的一种方式呢？ 用户故事是从用户角度，给用户带来的价值角度来描述产品需求的。相比冗长的需求规格，这个是易于理解的。需求描述的最终目的，无非也是，在各个角色当中达 成一致的理解。 如果一个功能实现的周期过长，就增加了很多不确定性，所以用户故事要求是小的，很容易实现的。这也就意味着，我在较短的时间内，就可以得到明确的结果。从 价值流角度分析，用户故事粒度小，这些价值是持续交付的。 一个功能开发时间越长，在开发这段时间，价值是没有被交付的。

6）持续集成

如果说站立会议，保证 team 成员之间沟通无偏差。那么持续集成就保证，我们的系统，模块，始终能正确的沟通和表达。如果其中的问题发现的晚，就会导致解决问题的成本高。持续集成，就表明，我们想要的，一直都是 OK。消灭问题与萌芽之中。

7）坐在一起

为什么要坐在一起？还是为了沟通，及时沟通，如果信息不一致，可能有同事基于错误的信息，做了错误的工作，白白浪费时间。作为信息民工，我们的工作的 输入就是信息，如果输入信息 delay（比如等邮件通知，等确认结果），那工作自然要 delay，如果输入信息有误，自然会导致工作浪费。信息不仅仅是信 息，它起的是控制作用。

这里我就列举了一些，其它的大家自己去思考吧。

总的来说，就是

1. 利用团队的力量来思考，来解决问题。

2. 强调端到端价值的交付 .

3. 强调信息沟通的及时性，一致性和准确性。

忘记敏捷，忘记 agile，关注问题，关注细节，关注团队，勤思考，提高工作效率和软件质量的办法总是存在的。

把敏捷实践作为你的工具箱，它不是全部，它不了解你实际的问题，也不知道你将去向何方。

作为代名词，可以继续使用，但是它并不意味着什么，代表着任何你想给它的含义。

外包，从拧螺丝钉开始

我们先来看一组数据，如果我们把当前的 IT 业分为咨询、应用开发、Offshore 等三部分，那么目前它们的增长率是这样的：咨询业务约为3-5％， 应用开发约为8-10％，而 Offshore 目前则达到25％还多。为什么会有这么大的差距？其根本原因在于社会分工。从前我们的公司类型多为枣核型，即 研发产品为主，而现在越来越多的公司逐渐转型为葫芦型，保留自己的咨询与市场，而将研发这部分进行外包。已经有调查表明，现代公司的竞争力与其外包业务的 多少成正比关系。

社会已经经过了一次创新的浪潮，现在对我们更重要的是将这些已经创新的知识进行消化，促成 IT 服务业的创新。在这一方面，IBM 是一个榜样，目前它是 全球最大的外包公司。HP 现在也在努力转型，2005年的集团内部重组就是一个例子，后又传闻收购普华永道的咨询业务，但被 IBM 抢先一步。这些现象都表 明，IT 服务业有着巨大的市场与潜力，也可以回答软件外包有没有前途这个问题。

从模仿到超越

现在很多人常把软件外包简单理解为做测试，其实这是不全面的，外包包括很多个方面，但一些大的公司多数是从做测试开始起步的，比如印度的 InfoSys、TCS 等，文思也是如此。印度软件外包业的发展起源于2000年前后的千年虫与欧元的问题，当时一个形象的比喻是印度 IT 人员常常是一飞 机一飞机地被运往美国。后来，随着印度国内经济的发展，原来在美国工作的印度 IT 从业人员开始回流，最终促成印度软件外包业的规模发展。一个例子是说当时 Wipro 仅有300人不到，但现在在全球已经拥有59000多名雇员。虽然现在像 Wipro、InfoSys 这样的公司还会从事一些测试的项目，但它们 的咨询业务和解决方案产品已经可以和 IBM 等公司竞争，从一个学生发展到和老师平起平坐的位置。从印度软件外包的发展史来看，从事软件外包不失为提高 IT 产业水平的一条正确道路。在文思创新的发展道路上，也有这样的例子，以承担测试业务起家，但在2006年3月，在与 InfoSys 的竞争中成功拿到 Tibco 公司的产品研发项目，让很多印度企业都大吃一惊。

如果要做个比喻，完全可以将软件外包比作中国的制造业。在汽车行业，奇瑞是一个很好的例子，从模仿别人开始，到现在已经可以成批地将汽车销往美国。发 生在我身上的一个例子是有关电视的，从前家里以拥有一台 Sony 电视为荣，但现在我家里用的是 TCL。这两个行业我们都是从为别人拧螺丝钉开始，走出了自 己的道路。

挑战中的中国优势

相比于印度，虽然我们没有抓住6年前的那两次机会，但现在我们有着更优越的环境。有四个方面值得我们注意：

１．目前中国的经济环境发展很快。很多国外的金融行业逐渐进入中国这个市场，这时把它们相关的 IT 业务放在中国来做，是一箭双雕的事情。印度公司虽然有经验，但考虑到各种因素，中国公司优势非常明显；

２．地缘政治的影响。软件外包行业目前在全球蓬勃发展，但多数公司不会把鸡蛋放在一个篮子里，一旦某一个国家发生重大变故，将会带来灾难性的后果。中国拥有高素质而成本低的优秀人才，显然是个不错的选择；

３．大批海外人才回归。印度当年软件外包业的兴起要归功于大批在美从事 IT 技术人员的回归，而中国在改革开放20多年后，也迎来了这个浪潮。这些海归人员带来了先进的管理与技术知识，这也是

我们机会之一；

４．最后一点就是我们具有充裕的人才储备。经过几年的发展，印度目前的软件外包从业人员已经达到40～50万人，现在 InfoSys 等公司依然保持着 30％的前增长率，已经面临着人才资源枯竭的威胁。而中国在2010左右，每年将有近1000万的大学生毕业，这一点也是印度所无法比拟的。

有机遇也有挑战。在我们还在学习印度的时候，印度师傅已经进入了中国。相信几年后我们的外包产业也会和中国汽车一样，开遍世界！

程序员，建立你的商业意识

程序员代表着技术的拥有者，比起其它的人群，他们拥有更强大的力量，甚至可以说拥有的是那个传说中点石成金的金手指。然而，过渡的迷恋技术也让很多程序员无法获得那段秘诀，本书将讲述那些已经找到秘诀的程序员的故事，并帮你分析他们是如何找到秘诀的。

1.1 程序员为什么需要商业意识

几年前，当我刚刚认识 Fishman 的时候，听到他神奇的创业经历，觉得非常不可思议。甚至还专门写了一篇报道发到《电脑报》上，题目是《从程序员到 CEO》。不久，Fishman 将创建的又一个新公司天夏科技卖给了丁磊的网易。丁磊以原来的天夏游戏开发团队为基础，开发的《大话西游》系列的游戏给市 场带来的冲击，完全扭转了门户网站的竞争格局。

尽管当时的网易股票处于极其底迷的状态，Fishman 放弃了很多股票而选择了现金，但 Fishman 从中获取的收益不是一般人所能够想像到的。刘韧也曾经在不同的场合表示过对 Fishman 技术和商业结合能力的钦佩。记得有一次在 Donews 的聚会上，我提到了一个想法，Fishman 立刻从不同的角度进行了一番分析，并做出了他的判断，尽管对于这些我仍然持一定的保留意见，但他 的分析思维和方式、方向给我很大的提醒。这提起了我对一个技术出身的程序员是如何逐渐拥有这些商业意识的关注。

拼音加加的作者廖恒毅也是我很好的朋友。他曾经是《中文之星2.0版》的开发者，曾经担任过微软开发合作部经理，对微软技术有着一种执著的热爱。直到 他现在担任佳软公司的 CTO，都依然亲身编码，他曾经说过“生命不止，编码不休”的类似话语。然而，当我三年前第一次去采访他的时候，他曾经告诉我说已经 停止了拼音加加的开发，原因很简单，尽管拥有了大量的用户，尽管每月有上百元的注册费（这在当时的共享软件中已经算是不错的成绩了），但对于他来说，这样 的回报显然不能令人满意。于是将精力完全放在了搭建企业级软件架构方面。

当时拼音加加做了一个非常奇怪的功能，这便是将未注册用户的首页指向了新浪的新闻页面，只有用户注册之后才能去掉这个令人感觉不爽的功能，除了这个功 能之外，软件再也没有其它的限制了。尽管后来出了很多的破解版本，但初期的破解往往治标不治本，过一段时间，进行某一个操作之后，首页便有会被修改为新浪 的首页。廖恒毅曾亲口对我说过，其实他在防止这个功能被破解方面做了很多工作，他在系统目录中保留了一个备份，随机的过一段时间便会检测一下首页，如果发 现被篡改，便会再次修改。这里充分体现了一个程序员的高明和狡猾。

不过重点并不在这里，而是拼音加加所指向的新浪新闻页面，一直以来，我认为新浪是需要为此向拼音加加附费的，但廖恒毅后来才告诉我：“指向这个网页纯粹是因为他自己喜欢看新浪的新闻而已。”

后来，我认识了百度的一位朋友吴京川，他是负责推广搜索引擎推广的，他们有一种可以和软件产品合作的业务，我介绍给了廖恒毅，说不如将拼音加加的页面指向百度，利用一些关键词挣钱。廖恒毅后来对我说：“他在与吴京川聊了一会之后，便发现了另外的一块广阔天地。”

事情的发展是这样的，廖恒毅在将拼音加加的首页首先指向了自己的网页，并把首页放置了一个与百度首页完全一样的搜索框。当然，搜索框中的有一个隐含的 参数，这便是用来定义有多少搜索产生付费的功能。而后，他又照抄了百度的网站导航页面，后来被百度的使用者发现之后，提出了抗议，于是他又去抄了另外的一 个网址站。

现在，当你再访问加加在线的时候，你会发现这里已经成为了一个小小的门户，每月廖恒毅从中获得的回报在10万元左右，而拼音加加的升级变得更加快速和频繁起来，甚至他们在两个月之内用.NET和 C++语言重写了软件。

同样一个软件，之前每月几百元，现在的收入可以养活几个人的开发团队，这之间的差距仅仅在于一个意识。这个意识便是商业意识。当然，拥有这样意识的程 序员也越来越多了，超级魔法兔子的作者蔡旋便在最新推出的版本中修改用户的首页指向了自己的网址站 Haokan123。

我曾经在网上与 FlashGet 的作者候延堂有过深入的交流，作为一个在陕西地区的程序员，他取得的成绩有目共睹。几乎一直是下载领域的明星。甚至在 国外的下载站点，FlashGet 也一直是最强有力的竞争者。而他自己每年的收入也在百万级，而这一切靠的仅仅是上门的一个大 banner，一个小 banner 和软件发放过程中对3721等插件的捆绑。然而，当我与好朋友小林沟通的时候，他告诉我说：“候延堂根本没有将 FlashGet 的能量利用起来，其软件的 价值也不仅于此。”为了更好的利用这个软件的价值，他所在的265公司在 IDG 投资后，甚至曾考虑收购 FlashGet 软件。

讲了这么多，我的目的便是真的希望能够挖掘一下程序员的商业潜力有多大。而这又不得不将程序员与商业意识挂上钩。

1.2 缺乏商业意识的程序员

平时，与众多的程序员或者技术人员沟通，我都很羡慕他们拥有的技术，你可以经常感受到他们内心深处那种对技术的狂热和执著。然而，我们又非常遗憾的看 到，很多他们引以为傲的技术是国外的程序员开发的，而且技术在不断的发展和进步，有很多程序员在盲目的学习和追随中失去了自我，进入了技术的漩涡。我曾经 与一位程序员合租过房子，那是我第一次见到一个人可以拥有如此多的书籍，在床头上摆满了各种技术书籍，高处顶到了屋顶。不过，与我们经历过的大学生活一 样，会发现很多书籍都是新的，让人感觉既敬佩又可笑。

很多程序员可能会说：我们不想创业，我们热爱技术，我们希望能够在在公司中一直埋头技术，钻研技术，所以我们不需要有商业意识。当然，可能大部分人不 会说这句话，但他的内心也会存在这样的想法。面对这样的观点，我只能说表示同情。因为在中国整个大环境下，能够给这些程序员提供如此的环境的地方不能说是 蓬莱的海市蜃楼，也差不多是屈指可数了。除了在高校中的研究机构外，任何一个企业都会要求以市场为导向进行发展。

四通打字机的发明人王辑志曾经写过一篇文章《软件不能在独木桥上走》，讲了他自己的一个亲身经历。一位朋友托他到澳大利亚演示自己所写的一套软件，然 而到了目的地后，他发现无论如此也无法成功的演示，回来后向这位朋友报怨。这位朋友很不以为然的向他演示了应该“这样……这样……这样操作就不会出问 题。”这让王辑志感受破深，一个程序员规定好的步骤不能出丝毫差错的让普通的用户使用一遍，不啻于让用户在独木桥上行走。这也是其缺乏商业意识的表现。

现在，就来参加一个小小的测验吧，如果你具有一下症状中的两条以上，估计就需要挖掘一下商业意识了。

程序员缺乏商业意识表现为：

* 拍脑袋就编程，做出决定
* 对解决了一个技术难题而得意洋洋，最后发现用户对这个功能根本不关心
* 不经过充足的测试便发布软件
* 骂用户笨，讨厌软件出来之后用户的报怨，不喜欢与软件服务人员沟通

1.3 商业意识对软件成功的重要性

很长一段时间以来，我们杂志跟踪微软亚洲研究院的发展，也和几位院长和不同的员工进行的交流。在外界看来，微软亚洲研究院是一个冰封的世界，一群技术狂人在随心所欲的创造和发明，他们完全是研究者的气质，仿佛与微软在外界市场上厮杀没有多大的关系。

然而，当你真的走进微软亚洲研究院的内部，了解了其发展的前前后后，你便会改变你的认识。因为，在这里，院长的指导思想，他们的商业意识无时无刻的不 在指引着微软亚洲研究院的发展和前进。微软亚洲研究院每定一个技术方向，都需要进行详细的论证，包括市场上有没有需求，与微软现有的产品如何的结合，如何 保证微软在未来的几年可以推出更具有竞争力的产品等等。从第一任李开复院长起，他们的工业界经验让微软亚洲研究院就一直沿着商业的意识在不断前进。张亚勤 博士曾经讲过，尽管微软亚洲研究院要研究的可能是未来五－十年才会用到的技术，但一定要结合微软公司的发展策略和方向，要能够为微软的长期发展奠定基础。 比如微软要在未来的人机交互界面方面需要很多新的技术，于是微软亚洲研究院就在 TabletPC 技术上做出了很大的贡献，同时还在智能数码笔方面做了很大 的尝试，这些技术研究都是在整个公司的大的发展策略下进行的，也唯有如此，微软亚洲研究院才获得了比尔盖茨和其它同事的认同。对于研究院的院长来说，如何 在大的框架下面选择好方向就是他们的职责。而现在，随着微软亚洲工程院的出现，更是将这种商业意识贯彻进了产品的开发中。

我注意到，在与已经工作超过五年的程序员，尤其是曾经带领过一个产品或者项目的技术负责人沟通的时候，他们所表现出来对商业意识的醒悟往往让你有深刻 的领悟。尤其当做出的产品在市场上遇到挫折的时候，他们却发现原来并不是自己的技术不行，也不是产品开发周期的问题，而是没有充分的考察市场的需求，没有 用商业的意识来指导自己做事。

在我采访汉王科技的时候，曾经与三位不同战线的技术和产品负责人一起畅谈，有两位负责人在汉王工作了十年之久，他们用亲身的经历验证了汉王发展中遇到 的问题。最初，汉王的核心技术是手写识别，公司创始人总裁刘迎健也是一位非常出色的技术人员，他曾经几十年钻研手写识别的核心技术。但在汉王创立的初期， 他们曾经度过了一段非常艰苦的日子，后来曾经有一个非常好的机遇，这就是 PDA 的兴起，当时恒基伟业的老板力邀汉王加盟，但汉王认为 PDA 这种产品实在没有什么技术含量，根本不看好，最终只拿到自己的技术授权使用费了事，这与后来恒基伟业所取得丰硕成果相比差距巨大，不过现在看来，随 着 PDA 市场的快速滑弱，对当初的决定是否合适便成为了一个无法说清的话题。然而，汉王后来的思路转变却在市场上取得了不错的回报。比如汉王在拥有手写识 别技术之后，开始考虑如何利用这些核心技术做出市场上用户需要的产品。“从核心技术到产品”，这成为了汉王的跨越标志。

名片通和文本王便是非常有代表性的两个例子，其实 OCR 软件已经出现了很长的时间，拥有这项技术的公司也不只汉王一家，但汉王最早利用这些技术做出了 一个在市场上销售火爆的产品。名片通就是为了解决现在大家交流频繁，名片聚集过多，但不易查找的问题。据说，名片通的推销人员打电话给微软公司公关部门的 人员的时候，本来根本不愿意理会，但后来听到介绍和看了演示之后，当即购买了数台。一个成本不足百元的黑白扫描仪和一套软件销售的价格超过的千元。

对于文本王，更是如此，最初提出这个想法的是汉王的一位技术人员。他当时表示看到了很多单位中都有扫描仪，但用的人很少，经过研究发现，原来 OCR 软 件的使用是需要比较专业的，首先需要配合扫描的分辨率，而且扫描出来的图片还需要进行校正，尽管这在程序员看来简直就是小菜一碟。但这些操作对于扫描仪潜 在的消费人群办公室人员来说就是不小的使用障碍了。于是他提出了对 OCR 软件和扫描仪的改进意见，这样汉王投入了一个团队进行开发，将 OCR 软件与扫描仪 更好的结合，比如可以自动校正，自动识别，自动输出 Word 文档等公那过。上市之后取得的效果是令人震惊的，用户的反馈也非常的热烈。

这让我回想起 PDA 最初发明的故事，一直以创新为理念的苹果公司创造了第一台掌上电脑

Newton，但由于体积庞大，运算速度慢，手写识别效果差等问题在市场上铩羽而 归。但有一个人从中看到了问题的本质，先从手写软件的识别着手，由于速度慢，识别用户不同的笔迹对 CPU 运算的速度要求很高，于是他发明了一套输入法，定 义了一套“任何字符必须一笔写完”的规定，尽管这对用户来说还需要学习，但学习的成本并不大，而且一旦学会，识别的效果会非常好，再加上其对电脑的功能进行了不断的裁减，只集中到了几个必要的功能上，这样在体积方面也达到了用户需要的地步，这款产品上市后取得了意想不到的成功，甚至从3COM 公司分拆上 市，这便是 PALM。

因此，在一个公司中，作为普通的一个程序员，同样要具有自己的商业意识，这些意识并非是为了考虑公司的经营，并非是为了转向管理，而是为了提升自己开发的价值。

1.4 商业意识不如用 Business Sense

我既不是“海龟”，英语讲的也不好。不过，我仍然在这里希望能够用一个英文单词来表明我的观点。因为在中文里面，商业已经被人用的过烂了，再加上国内 的这些商业体系的不完善，总是给人感觉不如 Business 的味道更加纯正。而对于“意识”一词，我更加是认为一直以来被用作政治用词，我们在这里拿来颇 有些“挂羊头卖狗肉”的意味。而 Sense 一词却有一种不可言传的体会，他讲究的是一种感觉，一种感悟。因此，Business Sense 可能更加适合作为我们希望宣扬的重点吧。

微软曾经重金从美国邀请来一位著名的作家，他写了一本书，我们出版社也出了这本书的中文版，其中他讲到一个重要的总结：他认为日本的软件做的是工程 （Engineering），欧洲的软件是当作科学在做，而在美国，之所以他的软件产业发展的很好，这是因为他们一直是将软件当作商业 （Business）在做。这也成为我写这本书的重要论据。

看到了这样三种模式，中国将要采取那种模式呢？就像公有制、私有制共存一样，中国存在的也是一种混合的体制，既有希望自己开发产品做商业的美国模式公 司，也有热衷与外包服务的工程公司，而在各大专院校中，软件不也同样被当作科学在搞吗？如果不是这样，同样开始进行开发的青鸟 UML 软件，怎会落得这样的 下场。

1.5 打破技术误区，拥抱商业意识

一直以来，中国的软件产业都没有按照正规的商业市场化进行操作，这一方面与国情有密切的关系。但同时，我们发现，很多成功的软件企业无一不是钻了某些 政策方面的漏洞，或者是利用了某个时期的政策而发展起来的。这一方面造成中国的软件产业没有按照商业的规模发展，同时也给很多程序员带来了很多误解。

误解之一是只要埋头写一个好的软件，自己就能够成为百万富翁。从一开始塑造的一系列类似求伯军这样的软件英雄开始，尽管激励了一大批高手进入了软件领 域，但这时恰逢整个软件开发的技术进行更新换代，同时软件的开发规模和质量要求也更加高了起来，在这样的大背景下，一个程序员缔造出辉煌显得是那么的无 助。成功的归结于机遇，失败的归罪于大环境的恶劣，而恰恰没有考虑的是否一开始就按照市场的规则做事，是否拥有了商业的意识和经验。

误解之二就是中国的市场很大，不必考虑国际市场。直到现在，仍然有一大批人鼓吹中国的市场很大，先把国内的市场作好就可以了。我认为，这简直就像“攘 外必先安内”的语调。在中国，唯一几个成功的将软件产品在全球行销很好的就是几个台湾的公司，包括趋势和友立科技。我在与友立科技董事长采访的时候，他很 明确的表示，当初开始做产品行销就考虑到了全球，为什么？就是因为台湾的市场很小，所以不得不把眼光放到全球市场。当友立科技还很小的时候，曾经因为触动 了 Adobe 公司的市场，备受打压，活的非常的艰难，而一旦 Adobe 发现这种打压很难真的消灭到 Ulead 的时候，他们又祭起了领悟一个法宝：收购。尽 管这次收购让 Ulead 没有机会成为与 Adobe 那样规模的公司，但付给

Ulead 的这笔钱让友立科技更好的发展了其它的产品。如果没有在全球上的这种拼 杀，Ulead 又怎能在图像处理市场上占据这样的地位呢。公司小不可怕，怕的是缩在家里，不愿意也不敢出去。

而程序员也在这个过程中失去了与全球软件开发思想同步的机会，当我们还乐衷与几个人埋头开发小软件的时候，大规模软件开发工程的思想已经在美国传播开 来，甚至还诞生了 Rational 这样的工具厂商。因此，我想说，中国的市场大对中国的程序员来说是个好事，还是个怀事真的是需要好好思量的。我觉得后者 的可能性还是要大一些。

误解之三：利用最先进的开发工具就可以做出更好的产品。中国的程序员生活很艰苦，但有一点他们永远都不会最差，这便是他们手中的武器：开发工具。伴随 着盗版，在开发人员报怨自己开发的软件被破解的同时，自己也在使用者被破解的开发工具，一个新的版本出来，立刻便会尝试使用，开发人员的机器上安装的永远 都是最新的开发工具。于是，中国一批又一批的程序员都变成了脱离某些 RAID 工具不会编程的“开发高手”。这首先，就是他们缺乏商业意识，对知识产权的不 尊重，在这种情况下，你能够要求他们深入学习和掌握手中的工具吗？我在采访一些美国的程序员时，他们很多人还习惯直接用最简单的开发工具写代码，因为公司 没有那么多资金用来升级开发环境，但这也造就了他们扎实的编程基本功。而国内的很多程序员只能够被牢牢的捆绑到一个平台上，当 Linux 出现的时候，又有 多少人可以迅速的从 Windows 平台转变到过来呢，尽管在很多人看来，这两个平台的开发在底层上其实并不多大的区别。

其实，程序员拥有商业意识并非让他们脱离技术的轨道，而是对其人生的更好补充。有了这些商业意识，在公司可以更好的理解公司发展的策略，做产品可以更好的做出成功的产品，自己创业可以更好的走向成功。

工作中 2 个高难度问题的思考

首先在我们回顾下这2个问题：

问题1:如何最大限度的降低需求的变更对测试质量的影响?

问题2:有没有什么有效的方法来降低测试用例的不断修改率?

记得好像测试用例 PK 里面也提出这2个问题，而且这些问题对于测试人员在工作过程中也是比较难以解决的，同样需要不断的尝试与探索新的方法与流程来增 加问题解决的可能性。其实我觉得这2个问题是有一定的共性，第一个问题解决了，第二个问题也就解决了一大半了。那我就首先谈谈个人对第一个问题的理解：

注意看题目的几个关键词：一个是“最大限度的降低”，另一个是“需求的变更”。最后一个是“测试质量”。我们要解决好这个问题，必须要很好的理解这个几个关键词的内在含义。我想并不是所有人都能很好的理解这些术语。

那什么叫需求的变更呢?顾名思义就是项目需求发生了变化与修改(先不谈什么原因)，对应测试这边测试需求也就相应的变化。这里可不要忘了一个重要的时 间点，那就是一旦 PRD 评审通过后。其后任何一个时间点，不管是 UC 设计还是测试设计或是什么，只要需求发生了变化，这都属于需求的变更。

那什么叫测试质量呢?这个概念比较大，一般我们讲软件质量，这个就与我们测试人员的工作职责相关的。而对于测试质量，个人认为包括2大块，一个是测试 各个阶段的产出的高质量，一个是测试各个阶段的控制的高效性。解释一下第一个是各个阶段的产出的文档的高质量，这里面包括文档的规范性，完整性，正确性， 统一性。第二个是各个阶段的进度控制和项目管理的高效率。包括测试目标以及发现缺陷，甚至是缺陷预防的持续改进等。

想必大家都知道需求变更不是个好东西吧，那我们就要想办法减少需求变更。首先需求变更往往是不可避免的。通常是项目负责人员花费了大量的气力避免需求 变更，可最后需求变更总是会出现。在需求并更发生之前尽量减少需求变更，以将需求变更带来的风险降低到最低。因此，在需求人员(PD)同用户代表或用户部 门主管人员接触时，就应该向他们挑明态度，和他们协商好，特别是应该让他们清楚软件的定价应该与软件的功能相关，以及需求随意变更所带来的风险的承担者应 该由客户和项目开发者共同承担。简单说让客户明白减少需求变更的重要性后，需求分析人员应该采取合适的方法同客户交流，帮助他们明确他们的需求。

还有一个就是我们要规范需求文档，需求文档应该按照一定的格式和规范来写，而且应该具备完整性、一致性、基线控制、历史记录等特性。需求变更发生后，也应该生成相应的文档，并且这些文档的书写也应该采用规范的形式来写。

前面说到得是减少需求变更的方法，这个不是一个人能做到的，是整个项目组成员共同努力的。正如前面所说，需求变更不可避免的会发生，那么当需求变更发 生后我们测试人员应该如何应对呢?一般来讲，需求的变更通常意味着需求的增加，需求的减少相对很少，而且处理也比较容易。而且发现现在开发人员和 PM 对需 求变更起主导作用，同时需求的变更并不能实时反馈到测试人员。这就需要流程的监控了，之前也

说了一旦出现什么样的需求变更，就相应的走需求变更的流程(相信这里应该充分考虑测试人员的参与度)。充分的与开发人员沟通加上 SQA 的实时跟踪也许会减少需求的变更对测试质量的影响。

我们说到了对测试质量的影响，那么有哪些呢?一个测试设计与测试用例的文档的修改;一个是与开发沟通需求变更的成本;再一个是测试人员重复测试执行的 成本。另外最关键的是由于需求变更带来的测试覆盖率的风险，特别是在测试执行后期阶段，时间计划都已经安排的很合理，这时由于需求变更，其影响可想而知。

那总结下解决第一个问题的思路：

减少需求变更

规范需求文档

与开发及时沟通需求问题

需求变更流程控制执行到位

尽量避免需求变更在测试后期阶段(成本大，风险大)

及时采取办法应对需求变更(时间延期，覆盖率，评估需求变更)

至于第二个问题就要搞清楚为什么测试用例需要修改，个人理解大概有这些情况：一个是上面说的由于需求变更，之前根据之前的需求写的测试用例肯定要修 改;一个是由于理解需求有误导致测试用例本来就写错了，但测试用例评审没有发现问题;另一个是比较小的修改，就是一些测试用例的细节与测试执行时得到的结 果描述有偏差(最常见的就是报错信息内容);最后一个是自己在测试执行时，通过探索性测试发现了 bug，而且没有测试用例，这时就需要增加测试用例。等 等。

搞清楚了原因，那么我们就可以有针对性的解决测试用例的修改率问题。比如第一个就需要把控需求变更的问题;或延长测试设计的时间，增加需求理解的正确性;还有就是加强测试用例的评审力度和监控力度;多关注测试用例的细节，用例设计时要求开发给出明确的输出信息。

这里同样的是测试用例的修改也是不可避免的，这样做的目的就是要做好测试用例的基线，更好的管理和维护测试用例，更好的回归测试日常需求。更好的加强自己以后在测试设计时思维角度的多变性。

以上是自己对这2个问题的理解，其实这2个问题也是在测试领域比较难以量化和解决的，这些解决思路都需要不断的探索和思考，去不断的总结，加上自己 Team 内部管理的一些规范，共同探索出适合自己的解决办法。

抛开技术做技术才是出路

短短一生不过数十载，对于很多人而言，作 IT、作技术只是生命中的某一段，并非所有。而无论是换工作还是换行业，只是一种形式而已，最终我们追求的是成功、是荣誉、是收获。于是在年轻的这几年里，作为技术人员理应认真思考自己将来的出路并为之而脚踏实地的去积累。

“01年大学毕业，去了老师开的一家网络教学软件公司，作教育软件；04年，首次创业，作了个休闲游戏公司，经验以及资金问题，创业失败；05年，一 个偶然的机会，进了网易，作服务器端开发，有幸结识了一帮志同道合且极有职业精神的同事。”这是大宝的职业经历。择过业也创过业的他分享了自己对于技术人 员出路的一些感悟。

抛开技术做技术才更容易成功

“在中国的这块土地上，抛开技术作技术，才可能更容易成功”大宝说。听起来颇有些拗口和矛盾的一句话，大宝对此做了解释。

遍观当今成功的这些互联网公司和产品，比如淘宝(电子商务)，征途(网游)，百度(搜索引擎)，他们之所以能够取得成功，其共同之处，都在于：紧紧抓 住了中国本土市场的需求以及众多的“小白化”设计。尤其值得关注的是：“小白化”设计。说得通俗一点，“小白化设计”指的是：易上手，易操作，便利性。

淘宝与征途，另一个共同之处在于：马云根本不算是一个精通网络的“业内人”，虽然淘宝已经是 C2C 的老大，但马云本人还是只会上上网，收收邮件；史玉 柱从未被网游业内人认为是“专业的网游开发者甚至专业的网游玩家”，但征途游戏的便利性，国内游戏在它之前没有一个做得到（当然，更多的人是不敢像他那样 去想）。马云和史玉柱的共同之处都是：外行领导内行。

“抛开技术作技术，才可能更容易成功”意思是在于提醒我们：要以外行的思维来开发内行的软件，一定要注意发掘小白用户的需求，要时时刻刻把自己当作从 未使用过这个软件的用户，多想一想这样的用户第一次使用你的软件时，他会觉得如何操作才会更方便，以及，我作为一个从未用过你软件的人有什么理由选择你的 软件。我们所作的软件，绝不仅仅是只要有了功能即可，你还得有“把你的软件功能营销出去”的概念，如果只是有了功能，但功能很难用，用户连尝试的勇气都没 有，那你作了也是白作。

而且很多的时候，对于技术人而言的一个所谓的颇具创新性、创造力乃至震撼力的技术实施方案，对于普通用户而言可能觉得根本是无所谓的，在这样的情况 下，如果时间比较紧，应该果断放弃你觉得颇具创造力的方案转而将产品功能尽最快速度实现。特别是在互联网产品开发领域，产品的推出速度是产品致胜非常重要 的一环，有的时候，为了尽快推出产品，我们就不能在这些细枝末节的地方，对用户体验没有太大障碍的地方浪费太多精力，而要集中精力把主体功能完成向用户推 出来，然后在后续的时间里再精益求精地不断改善。

大宝说，“很多人不会像征途那样去做，是因为他们不敢像史玉柱那样去想。而问题关键的关键，就在于‘敢想’二字。如果你连敢想的勇气或者空间都没有， 那将是最大的失败，也是最无法挽回的失败。只有把自己首先定位成自己产品的小白用户，你才会‘敢想’，也才‘能想’出来。”

纯做技术是自娱自乐的生活态度

大宝认为“作纯技术的人，抱着的是一种自娱自乐的生活态度，他们生活在自己想象的‘纯洁空间’里，拒绝与人打交道。”虽然不能说这是一种不好的方式， 但是显然对于大多数人的发展而言，这样的个人发展途径他不认同，他更提倡的是一种积极、张扬而又务实的生活和工作态度。

这是因为往往在某一方面作得太深入的人，在另一方面就会少有时间和精力投入。比如，比较喜欢作纯技术的人，他们一头扎进技术里，却往往忽略了真实的市 场用户需求和感受，从而让自己作出来的东西，欣赏的人只有自己，对于以赢利为目的的公司而言，这不能不说是另一种失败。

大宝更提倡的是“作产品”的概念，而不只是“作技术”。“我们作的东西最终是要销售出去，要给公司赢利，要给团队带来收益。“作产品”的概念，更强调 把眼光放在全局，关注这个产品从策划，设计，研发，测试，交付，乃至到版本更新这样一系列的过程，关注用户最终使用感受和效果，关注用户是否愿意为此付 费。”

团队或公司给成员的回报，永远是看成员给团队或公司创造了多少价值。他认为有“作产品”概念的人，工作会更有针对性和主动性，甚至有前瞻性，只要坚持 这样的想法和工作方式，就必然会给公司创造更多的价值。所以，单纯的以技术与非技术来说谁能混到更好的职位，有点太笼统了，这个还要看各人贡献。

技术员——>高级技术员——>管理

不少技术人员将“技术员——>高级技术员——>管理”定位成自己成长的路径。对此大宝不置可否，“只是管理职位只有那么有限的几个，不可能人人都能坐，这怎么办？”他反问。

大宝认为作为一个技术人员，如果只看到只有这一条成长途径，那表明他经历的还太少、眼界还太窄。“原来，我也一直很困惑：到了三十岁，四十岁，我还会 继续写程序吗？如果不写程序的话，我到时又能干些什么？现在，通过近几年自己的经历和体会，我慢慢明白了，其实，一个技术人员他将来如何发展，或怎样发 展，取决于我们自己是不是愿意打开眼界去看看外面的世界，去思考思考技术之外的世界，去关注关注与人打交道时的乐趣。”

而之所以有之前的那种恐慌，是因为我们一直以来都以培养和锻炼自己的专业技术能力为唯一的能力追求(比如 c++，比如算法等)，从不曾考虑过锻炼一些 可以跨行业，跨年龄段，跨公司的能力，这些能力包括：正确作事的心态和正确作事的方法（坚定的信念和具体细节的操作），团队的概念以及团队协作的方法(学 会与人打交道)，全身心投入作成功一款产品而不只关注把代码写出来（有大局观）。代码写出来，并不表示产品就可以挣钱，就可以取得成功，对用户真实需求的 深度挖掘，诸多的用户体验细节，便利性以及效率，乃至与竞争对手的相对优势都是决定产品成败的相关因素。

当具备了把产品作成功的大局观之后，如果愿意慢慢尝试去锻炼自己上述的能力，就会再细细研究下去，看在目前的环境下如何对产品的成功实施更加积极的影 响。“而当你有了这些能力后，你会发现自己的发展空间被空前拓展了，那时，你就尽可能选择一个自己感兴趣的方向作下去，而这个方向，很有可能已不再是技 术。此时的你，已经有能力自己去拓展新的空间，所以，你的空间也不再仅仅局限在技术，局限在这一家公司身上。”

“作事业”的心态至关重要

“我自己的开发实践还在继续，还在继续写着代码，只是现在已经很少跟人就某个技术细节再进行多么深入多么深刻的讨论了(只跟我的同事讨论，网上讨论的 很少)，而是把精力更多的放在了关注产品整体质量，促进产品整体质量方面。”这个“质量”不仅仅包括技术，还包括策划和设计环节，包括服务环节，甚至包括 运营、维护环节。

所以大宝觉得“作事业”的心态，对于一个人的成长与成功非常重要。我们现在很多人，抱怨上天不给自己机会，待遇不好，环境不好，的确，很多的时候，我 们确实是面临着这样的困难境地。但是，即使在不太好的环境里，我们也有选择的权利，我们有选择积极面对还是临阵退缩的权利，有选择激情奋斗还是得过且过的 权利。积极的工作和生活态度，会让我们终生受用。当你以“作事业”的心态去面对自己的工作时，你会发现更宽广的空间和舞台，你自己的成长也会更快。

有很多人可能会说，“作事业”的心态说起来容易，我是把它当事业来作了，但我没取得应得的回报呀。我不反对确实存在这样的情况，这个世界永远充满着黑 暗面，如果你总是以这样的心态去拒绝付出，那我只能说，你从团队得到的回报就会比你付出的更少。有这样抱怨的朋友，我建议你可以对比一下：自己当前的能力 是不是与你的回报成正比的，不仅与公司内的比，也与公司外的比一比，与整个行业比一比。你要是实在觉得委屈，你大可以换一个公司，但是，有句话，我还是想 提醒：换工作，不能仅仅为了看得到的那点工资，换工作的成本同时还包括了你要重建你的同事关系，上下级关系甚至行业资源，而这些，是决定你是不是能把一件 事作成作大的重要因素，人的眼光，有的时候要放远一点，如果老板是值得跟随的老板，从未亏待过你，如果公司在可预期的未来不会马上倒闭进而导致你没有饭 吃，何不再坚持一下呢？

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。

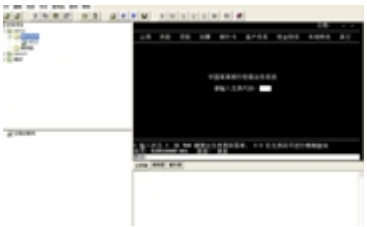
在线体验		产品租用	
企业版	免费版	在线申请	详情

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