我的测试观点与经验

初中级篇

克莱沃曼（Jan16，2009）

Contents

前言

本书收录了我一年多所写的关于软件测试的文章，内容包含初中级测试工程师工作所涉及到的内容。测试总的来说还是比较简单，我认为作为初中级 tester 来说，这些文章基本涵盖了应该掌握的知识点，和一定的职业指导与规划。今后我写文章可能不会再涉及这些方面，而会从更高的角度去讨论测试技术与发展，因此把这些文章编辑成册，以方便大家的阅读。

测试与开发

关于开发与测试

最近总是有一些网友问我一个问题"现在有机会转开发，应不应该转?"。我想我也需要单独发表一篇文章来表达一下自己的看法了。我会从一个新入行的测试人员一直往上发展大概会是一个怎样的情况入手，来表达我的观点"开发技术是测试人员发展的基础，也是测试人员发展的归宿"。当然了，我现在表达的是测试人员一直往上发展，如果你已经对自己的发展感到很满意了，则不在这里的讨论范畴。（相信会有不少这样的人来提出自己的疑义）

从你刚入行开始，你面前基本上是有两条路可以发展。一是技术，二是管理。先说管理，很多人做了很短的时间就走向了管理，team lead，甚至 manager。他们的工作已经很少用到技术了，更多的是学习和运用管理的知识。这样的人不在少数，我当时做测试 3 个月一转正就是 team lead 了，老板也非常希望我专注在管理方面。可是下一步怎么办？一般出现这种情况的都是在中小公司，而不是大而正规的外企．下一步你的发展会由于技术不够扎实而出现了瓶颈．第一，你不可能回头去搞技术了，因为你管理的地位，工资都高于技术人员，而且还有年龄的问题，这些因素造成了你只能继续管理。第二，如果你想跳槽到大的外企搞管理，可能性几乎为零，因为你没有大外企的经验又没有出众的技术。第三，如果你想在本公司或类似公司往上爬，比如做 director,很可惜，这些公司往往不配备 test director 的职位，并且其他 director 的职位也很难由一个做测试的人来承担。（可能会有特例，但是应该很少）。因此，刚入行不久，没有扎实的技术沉淀，就走向了管理，很快就会发展到头了。

再说技术，我们知道测试的最高 title 就是 SDET 和 SET 了。实际上都是一个东西，微软叫 SDET,Google 叫 SET。说白了，这种职位本质上就是开发，只不过不是产品的开发，而是测试的开发。需要很强的开发背景。因此，一个普通测试人员想要得到这种工作机会，开发的经验是必不可少的。另外，微软， Google 这种最 top 的公司，基本上已经没有 STE, SQAE 这种职位了。而我们大多数人做的测试工作恰好正是这种职位。顶尖大公司的这种职位都外包了。因此，你想通过普通的 STE, SQAE 这种职位进入顶尖公司，可能性基本为零。因此，如果从技术上来讲发展，无论从 title 上还是公司上往往上走，开发的功夫都是必须的。

因为我说的是一直往上发展，因此到了 SDET 后怎么发展？基本上三条路，第一管理，第二开发，第三继续测试。先说管理，你从技术上一路走过来再走向管理跟前面讲到的那种管理是天壤之别了。首先，你已经是在顶尖公司了，其次，你已经具备了高深的技术水准了。放眼望去，整个测试行业还能有几个适合你的位子呢？还能有多少人能跟你相提并论呢？这条路可以 lead->manager->director->vp->senior vp 往上发展。那么发展到了最后还怎么发展呢？这个时候，我估计应该年纪很大了吧？钱很多了吧？享受生活吧。

再说转开发，因为 SDET 具备了比较强的开发水平（比一般公司的开发人员的水平要高），因此可以比较容易的转向开发。转向开发之后就属于开发的发展之路了，就不多说了。

最后说继续测试，一直以来都有个疑惑，为什么我从来没见过 senior SDET？本来 SDET 可以向 senior, principle 这条路走下去。我以前也想过自己发展成 test architect。可是，为什么没见过？依我现在的感觉来说，测试的技术很快就学到头了，开发的技术由于一直是在做测试程序，而不是真正的产品，因此提高的程度也受到了很大的限制。因此，在技术上来讲，一直工作下去就会维持在一种不是很高技术水平的状态。这种状态达不到 senior 的要求。这也是为什么周围很多 SDET 不知道工作了多少年，还是 SDET。目前来看，想发展的话，维持在 SDET 不算现实。必须走向管理，或者转向开发了。这也是为什么 director 在回答我的问题"我在测试方面应该怎么发展？"。回答竟然是"短期来看要学好 C，长期来看还是 C，转向开发"。根本就没有提到测试。

以上是我自己的分析与理解。

大概几个发展路径

1.QA->management

2.QA->SDET->management

3.QA->SDET->Dev

当然还有其他的可能，比如

4.Dev->QA.....

5.QA->Dev.....

6.QA->Dev->SDET.....

7.Dev->SDET.....

8.SDET……

我想大家可以分析一下自己以前的发展之路，看看以后应该如何发展？当然了，如果自己已经满意了就算了。我的发展之路是这样的

Dev->QA->Management->SDET->(Lead/Dev/Senior if possible)

总而言之，在整个的发展路径中，如果你缺少 Dev,都会限制你更好的往上发展。还有就是发展的高低，一要看 title,二要看公司。很多小公司的 manager 可能根本都进不了大公司。很多大公司的普通员工一旦跳槽到小公司很容易就能升职。比如一个朋友在国内做了 6 年的外包经理，面试大公司总是失败，因为技术不行。还有朋友在顶尖公司只是 SDET,跳到盛大就是 director.

这里我想破除很多测试人员的一个幻想，我强调的一点是“不懂开发的测试没有太大的前途”。

再度解析开发与测试

不久前参加了一个座谈会。这个座谈会邀请了公司里在测试领域发展非常出色的华人，基本都是 test manager，来给大家解析一些测试发展的问题。由于可能公司没有华人做到 test director 的职位，因此他们可能已经达到了华人在测试发展的最高点了。他们应该是很牛的人，为什么呢？举个例子，我知道两个普通的测试人员回国就是两个大公司的测试 director。而这两个人跟他们的差距还是很大的，可见他们如果在国内应该是多么牛的地位。当然了，他们在世界的测试领域来说也应该是佼佼者了。座谈会的焦点没有任何意外地集中在了开发与测试的比较上边来了，这个话题也是多人曾经探讨过，我个人也发表过一些自己的看法的。这次想总结一下从他们的观点中透露出来的信息以及个人的一些自己理解 。

首先说明的是，测试发展的两条路还是管理和技术，由于真正能在管理上发展的像他们那样成功的人实在是太少了，尤其是在公司的内部，因此焦点还是放在了技术发展上。其次，与绝大多数公司不同的是，这里的工资是按照级别而不是按照工种来区分的。简单地说就是，同一个级别的开发与测试的工资是相当的，因此在技术发展的焦点上就放在了级别的提升上。下边是个人的一些总结：

1. 他们承认测试发展的 ceiling 是比开发要低的，比如 VP 测试出身的只有一个，还说了个什么我没听明白，测试的一个没有。当然这个对个人来说没有什么问题，我想基本没人能够发展到 reach 到这个 ceiling。
2. 测试的 senior 比例是比开发要低。
3. 测试 senior 比例低的主要原因是很多人没有达到 senior 就转成开发了。
4. 测试通过自己的提高是有机会达到 senior 的
5. 测试从 junior 发展到中级跟开发是没什么区别的
6. 测试从中级发展到高级的时间是一年半到 never
7. 他们见到很多人在中级呆了很长时间，比如 10 年，还是停留在这个级别，因为自己不努力了。

从以上的简单总结我们可以看出，首先测试的发展跟开发相比是处于劣势的，其次，通过个人的努力测试是可能跟开发平起平坐的。这里边就牵扯到一个问题：怎样通过努力去达到 senior？这里边又牵扯到了另

外一个问题，为什么很多人测试没有达到 senior 就转成开发了？他们如果不转成开发是否就能达到 senior 测试呢？因为他们说很多人 10 年都到不了 senior。

他们的解释是可以做安全测试呀，可以考虑客户的需求呀，等等，等等，去往 senior 发展。我想说说自己的理解：

1. 我见过的 senior 确实很少，因此可能理解不够全面，可是我感觉他们应该是具有多年的开发经验的。也就是说，他们今天能够到达 senior 这个级别是和他们多年的开发功力密不可分的。
2. 由于公司是按照级别来给工资，里边的现实是，如果你想达到 senior 的级别，你的水平要和 senior 的开发相当才行。一个做测试的人，怎么才能达到与 senior 开发水平相当呢？除了以前有过多年的开发经验，否则是一件非常困难的事情。困难的地方就在于，测试相对开发来说还是一个比较简单的工作，在每天做这种相对简单的工作中是不可能像开发人员得到同样的水平进步的。
3. 确实测试里边也有很多高端的测试工作或技术，比如安全测试。可是测试总体来说还是比较简单的，你的老板不太可能会给你时间去学习和分配高端测试的工作给你，否则那些简单的测试谁来做呢？因此，在工作任务的压迫下，使你转向做高端测试的机会变得比较渺小。
4. 当然你自己可以去努力，花大量的业余时间去提高。可是这里边还有两个问题，一个是你是否能长期坚持，因为这不是工作驱动的，人很容易偷懒或者放弃。第二是谁来指导你？如果你身边很少 senior 的测试，你被指导的机会就会远远小于开发人员。
5. 这也就算解释了为什么测试人员要转到开发，很大程度上是不得已而为之，如果想继续往上发展的话。即使一个人非常的热爱测试工作，他如果想往上继续发展，走向开发，或者走向开发再转回来都变得是一条更现实的路。

因此，我还是鼓励大家，如果有机会做开发，或者转开发就不要犹豫，如果没机会，也要尽量地去学习一些开发知识。这些对测试的长期发展是很有好处的，我本人也是得益于以前有两年的开发经验。

关于测试与开发的工资

今天查了一下开发与测试的工资水平，一些数据应该能说明一些问题。我们以硅谷的 Mountain View 为代表（Google 总部所在地）。

开发：

Junior Software Engineer: $62K

Software Engineer: $98K

Senior Software Engineer: $110K

Principal Software Engineer: $110K

Lead Software Engineer: $113K

测试：

Junior QA Analyst: $53K

Junior QA Engineer: $61K

QA Test Engineer: $76K

Software QA Engineer: $86K

QA Team Lead: $88K

Senior Software QA Engineer: $93K

SQAE Manager: $100K

Junior SDET: $95K

Software Development Enginner in Test: $102K

Senior SDET: $108K

Lead SDET: $108K

可见，SDET 是测试工资最高的职位了，比一般的 Software Engineer 还高一些。不过到了 Senior 就比开发低一些了，而且根本没有 Principle 的职位。即使 SDET Lead 工资也没有高出去。能够跟开发的工资水平相提并论的测试职位就是 SDET 了。

黑盒与白盒

与同行关于白盒测试，黑盒测试的讨论

作者: cleverman **时间:** 2007-7-8 10:49 **标题:** 到底什么是白盒测试？

白盒测试，黑盒测试是刚学测试时候的基本概念。我们大部分人可能做的都是黑盒测试，也就是看不到代码，对产品进行功能测试。

可是什么才算是白盒测试呢？

Code coverage 算不算白盒测试？

自己和别人一样做黑盒测试，可是自己没事的时候也看看代码，在黑盒测试的时候发现 bug，自己可以去在代码里定位，找到 root cause。那算不算是白盒测试呢？

如果不算的话，因为测试的过程中也分析了代码，明显不算是黑盒测试。如果算得话，自己跟同事做同样的工作，只是自己没事看看代码就可以说别人做黑盒，自己做白盒？

要不算灰盒测试？

我们公司并没有白盒，黑盒测试的概念。可是这又是测试最基本的概念。我有点 confused.大家讨论一下好吗？

作者: wyzwise **时间:** 2007-7-8 11:43

White box 就是在全部理解系统的基础上去 go through source code 和 architecture documents。

Code coverage（当然是拉） 是 White box 的一个重要的优势和 black box 比较。。

首先做 white box 的时候要有计划，不要无计划地去看 source code, 应该去根据你所在的 project 的 design document 去做特定地区的 code 分析，还要看看 algorithm..这个部分主要通过 peer review 去发现，当你感觉可能有问题的时候，应该去和 senior developer 讨论，并把问题，写成书面的报告报告给经理。。。有经理决定它的 priority. 在决定是否是需要 fix 这个。。。

现在你是有点混淆了 white box 和 grey box 的概念， white box 是在正常情况下，你的 project 一切的 design documents 都很全的情况下去进行的。意味这你有 full knowledge of the system。

而 grey box 是在 limited knowledge of the internals of a system..只有在这个 project 的 design documents 不全的情况下，进行的。。。

不知道有没有解决你的问题：）

[*本帖最后由 wyzwise 于 2007-7-8 12:00 编辑*]

作者: shanxi **时间:** 2007-7-8 11:52
白盒、黑盒
自动化、手工
回归等等
具体做事的时候没有区分的那么细致，很少是只做其中一种，基本是交叉着本着解决测试问题的效率来采用。

作者: cleverman **时间:** 2007-7-8 12:02
QUOTE:

原帖由 *wyzwise* 于 2007-7-8 11:43 发表
White box 就是在全部理解系统的基础上去 go through source code 和 architecture documents。
Code coverage（当然是拉） 是 White box 的一个重要的优势和 black box 比较。。
首先做 white box 的时候要有计划 ...

那我说的第二种情况,算不算白盒?
还有就是能不能举个灰盒的例子.

作者: cleverman **时间:** 2007-7-8 12:05
QUOTE:

原帖由 *shanxi* 于 2007-7-8 11:52 发表
白盒、黑盒
自动化、手工
回归等等
具体做事的时候没有区分的那么细致，很少是只做其中一种，基本是交叉着本着解决测试问题的效率来采用。

我也有这种认为,感觉测试很多理论上的东西,在工作中很不相关.
我现在倒是感觉,真正用的上的,还是开发的理论,而不是测试的理论.

作者: wyzwise **时间:** 2007-7-8 12:09
QUOTE:

原帖由 *cleverman* 于 2007-7-8 12:05 发表
我也有这种认为,感觉测试很多理论上的东西,在工作中很不相关.
我现在倒是感觉,真正用的上的,还是开发的理论,而不是测试的理论.

这个一部分是对得，我以前得公司，计划，管理不是很好，所以每次经常什么东西都混在一起，不是很明确。。。
其实如果管理好，这些测试的理论是很有用得。。。我现在得 team 就是按照成型得 MVC 得 testing phase 一步一步走，每一步每个在 team 里面得人都很清楚，都知道什么时候在什么事情。。。 测试还是要根据不同阶段做不同得事情得。。。

作者: seifer1754 **时间:** 2007-7-8 13:40
既然白盒，黑盒 的概念你都知道了，那么举个例子来说明一下好了。

```
void test(x)
{
  if (x<0) return;
  if( (x+1) < 0 )
  return 9;
}
```

看看上面这个函数，很明显 第二个 if 判断是永远不可能为 真，也就是说，采用黑盒的方式，用输入 参数 x 来测试，是永远不可能出现 9 这个输出的。
那么我们用不用考虑第二个 if 的成立情况呢？
我这个例子是简单了一些，也没有什么意义。不过在实际的开发中，还是会出现类似无法被黑盒测试覆盖的路径的，也许这些路径在正常情况下永远不可能被走到，可是我们在做软件设计的时候，必须要考虑到会出现的异常，而专门设计了一个对这种异常情况做出处理的分支。
例如，计算机的寄存器被恶意软件所篡改，如果没有一个对异常处理的分支，很可能会造成系统的崩溃。
那么，既然这种分支需要测试，我们就必须想办法来覆盖这个分支，这就是白盒测试需要考虑的事情了。
我们可以在这个分支判断上设置断点，然后通过修改寄存器的方法来改变装载参数 x 的 寄存器的值，从而使程序能够走 x+1<0 的分支，使输出结果为 9 。
这就是白盒测试需要考虑的事情，是对程序的逻辑与路径进行分析测试。
我举的例子可能不是特别恰当，希望你能够明白。

作者: cleverman **时间:** 2007-7-8 14:21
QUOTE:

原帖由 *seifer1754* 于 2007-7-8 13:40 发表
既然白盒，黑盒 的概念你都知道了，那么举个例子来说明一下好了。

```
void test(x)
```

{
  if (x

我明白你的例子，不过你认为我说的第二种情况算白盒测试吗？也就是说跟黑盒测试没有任何区别。可是，遇到 bug 自己可以去代码里找到 root cause.

作者: wyzwise **时间:** 2007-7-8 14:32
"可是自己没事的时候也看看代码，在黑盒测试的时候发现 bug，自己可以去在代码里定位，找到 root cause。那算不算是白盒测试呢？"
这个不是 testing...是 bug fixing:)

作者: cleverman **时间:** 2007-7-8 14:32 **标题:** 还有
“例如，计算机的寄存器被恶意软件所篡改，如果没有一个对异常处理的分支，很可能会造成系统的崩溃。那么，既然这种分支需要测试，我们就必须想办法来覆盖这个分支，这就是白盒测试需要考虑的事情了。”
请问你们公司是必须要测试这种 case 吗？我问题的意思是，概念可能没什么难理解的，可是到了实际中去，情况就复杂了。以我的经验，你说的这种情况是不需要进行测试的。因为，重要的是怎样防止恶意软件侵入你的计算机，真正侵入进来之后，就麻烦大了。如果要考虑这种情况的话，那可以设计无数的 case 了，不算很现实。请问，你用“必须”这个词是理论上来说呢，还是你们公司就这么要求的呢？
还有就是以我所知，无论是系统软件还是杀毒软件都是注重恶意软件的侵入，而不是侵入之后如何保证自己的程序不受影响。当然了，现在的 UAC 是这样考虑的，不过他们也不是在硬件的 level 来进行测试的。

作者: cleverman **时间:** 2007-7-8 14:35
QUOTE:

原帖由 *wyzwise* 于 2007-7-8 14:32 发表
"可是自己没事的时候也看看代码，在黑盒测试的时候发现 bug，自己可以去在代码里定位，找到 root cause。那算不算是白盒测试呢？"
这个不是 testing...是 bug fixing:)

不对。真正好的测试人员 report bug 的时候，是要写明白代码的 root cause 的。这只是发现 bug，怎么能算 fix 呢？
真正的 fix 是要开发人员来做的，测试人员没有这个权力。
你找到代码里的问题，不代表就解决了这个问题。明白吗？

作者: seifer1754 **时间:** 2007-7-8 14:43
通过黑盒测试的方法发现 bug,然后去代码中定位，找出原因。

这个不属于白盒测试，白盒测试和黑盒测试 在测试的理念上是不同的，也就是说，设计的用例都是不同的，采用黑盒测试方法设计的用例，不见得能够 100%的覆盖代码的所有路径和逻辑。例如我上文举的例子。 虽然你能够根据黑盒测试方法发现的 bug 而定位错误，这也只能说明你的编程功底比较强，对代码的理解比较强。可是这并不能说明你设计的用例能够覆盖所有的代码逻辑和路径。也就是说，这并不能说明你做的是白盒测试。
在嵌入式白盒测试领域，经常需要追踪寄存器和内存，这些都是典型的白盒测试，你需要在程序中设置很多断点来追踪程序流程。而黑盒测试，只是考虑函数的输入和输出，对程序的内部逻辑是忽略的，很可能，同样的输入和输出，代码采用了不符合需求的方式而实现了。
所以，即便你能够定位代码的错误，可是你做的是黑盒测试，手中获得的可能只是一份功能需求文档，而不是一份详细的代码流程图，你定位的错误，只是表示程序的功能有问题，不能证明程序的逻辑设计有问题。
这就是白盒测试与黑盒测试的根本区别。

作者: cleverman **时间:** 2007-7-8 15:08
那么什么是黑盒测试呢？不是说黑盒测试是不接触代码的吗？我以上的例子是有 review code 的工作的，应该和黑盒测试定义相矛盾呀。
我明白你说的嵌入式白盒测试是属于白盒测试，可是白盒测试肯定不限于只是这种类型吧。按照你的理解，如果我有详细的代码流程图，做黑盒测试，而我定位的错误也发现了是逻辑设计有问题，那怎么算呢？你也不能说我用黑盒测试，review 代码就一定不能发现逻辑设计有问题吧？
为什么要问这个问题呢？我所在的公司，当然不是测试嵌入式的，是测试系统软件的。我们有所有的资源，可是我们的 test case 是按照功能来设计的，也就是说不会在代码级设计测试用例。如果只是设计，执行，那么肯定是算黑盒测试。可是我们也 review code，要求发现 bug 尽量找到 root cause 写到 report 里。因此，我们有详细的代码流程图，也可以发现逻辑设计有问题。我们甚至要在 functional spec 阶段就要 involve 进去。当然了，在找 root cause 的时候，我们也要用 debugging tools,向你所说，要追踪内存，寄存器，堆栈等等。那么这算不算白盒测试呢？

作者: cleverman **时间:** 2007-7-8 15:11
你以上的解释有点已经先定性了我就是黑盒测试，在这个基础上来解释的。我想现在唯一的区别就是从功能设计 test case 和从代码设计 test case 了。
但是，我想这不是黑盒，白盒的根本区别。你说的根本区别也不是这个。

作者: seifer1754 **时间:** 2007-7-8 15:43
如果要详细的划分黑盒与白盒，不是从是否能看到代码来划分。
我想应该是从设计 case 的角度去划分。
例如，作功能测试，测试一个登陆窗口，这是典型的黑盒测试了，那么你肯定是根据需求文档，输入有代表性的字符来验证这个窗口是否正常。如果输入某一个特殊字符，窗口的功能实现出现问题，你也许可以定位到某一个条件判断语句有异常。然后根据测试的步骤，异常的现象，自己推测的原因来提交缺陷报告。

可是你所设计的用例，都是围绕着这个窗口的功能而设计的，你肯定不会去考虑这个实现这个窗口的代码有多少个循环，多少个判断。
但是用白盒测试来测试这个窗口的代码，就需要根据具体的代码流程图来设计，要判断程序的逻辑，实现的方法 是否与程序设计需求一致。甚至需求中会规定内存分配需要多少空间，你也要对内存空间进行用例设计与测试。 所有的设计都是根据代码需求文档来设计的。
你介绍了你工作的内容，能够接触到详细的代码流程，甚至也需要追踪代码逻辑。那么我认为，如果你设计的 case， 是为了追踪代码的逻辑而设计的，这就是一个白盒测试用例，你做的就是一个白盒测试工程师所应该做的事情了。
虽然公司可能把你划分为黑盒测试，可是不代表你所有的 case 都是采用黑盒的角度去设计，也不代表你不能根据代码流程发现逻辑错误。
我们只是讨论黑盒与白盒测试的区别，不代表一个黑盒测试工程师就无法接触到白盒测试。
一个好的测试工程师确实需要能够协助开发人员定位错误，可是最需要的是能够采用多角度去设计 case，然后发现 bug。
至于我楼上举的例子，因为我是做航空嵌入式系统的，所以对代码的异常处理非常严格，要求所有的路径都要覆盖一次以上，才有了“必须”这一说法，有些职业化了，到让别人产生了误解。

作者: wyzwise **时间:** 2007-7-8 17:28
QUOTE:

原帖由 *cleverman* 于 2007-7-8 14:35 发表
不对。真正好的测试人员 report bug 的时候，是要写明白代码的 root cause 的。这只是发现 bug，怎么能算 fix 呢？
真正的 fix 是要开发人员来做的，测试人员没有这个权力。
你找到代码里的问题，不代表就解决了这 ...


也许你们公司的特殊要求吧。。。我对我的 team 的要求是，tester 不需要去找到原因，找到哪里出了问题。。如果每一个 tester 都花时间去看 code，找 root cause，这个涉及到成本问题，毕竟人工是一个人每小时上百人民币。。。呵，cost and timeframe 是一个 trade off...
最省事省力的做法是，找到问题，report to manager,assign to the function owner...fix bug...new build version...regression.....这个是常规做法。。。
还有一个就是 white box 和 black box 。。简单说是，white box is to test the design(defect),black box is to test the system without going through the code inside...

作者: scorix **时间:** 2007-7-8 18:25
楼主应该自问一下：“在发现错误之前的测试过程中，我是否只关心软件的内部构造？”

答案如果是肯定的，那么就是白盒测试。

至于定位错误，我想大部分的情况都要翻看代码才能确定哪里出了问题，但这不代表就是白盒测试。

作者: DERYCK　**时间:** 2007-7-8 21:14

看了以上几位的看法，我觉得还是在测试过程中你是根据外部功能设计的用例进行测试，还根据内部代码的结构设计的用例进行测试。

只要分清楚这两种区别就明白什么是黑盒测试和白盒测试了。

我个人认为定位错误不是测试。

作者: cleverman　**时间:** 2007-7-9 01:18

QUOTE:

原帖由 *scorix* 于 2007-7-8 18:25 发表

楼主应该自问一下："在发现错误之前的测试过程中，我是否只关心软件的内部构造？"

答案如果是肯定的，那么就是白盒测试。

至于定位错误，我想大部分的情况都要翻看代码才能确定哪里出了问题，但这不代表就是 ...

不对呀。我们在 functional spec 的时候就 involve 进去了，很关心软件的内部构造呀。我们对 Windows Internal Knowledge 的实现都要精通，更不要说自己的软件了。好像这不能说明是白盒测试呀。

作者: cleverman　**时间:** 2007-7-9 01:23

QUOTE:

原帖由 *wyzwise* 于 2007-7-8 17:28 发表

也许你们公司的特殊要求吧。。。我对我的 team 的要求是，tester 不需要去找到原因，找到哪里出了问题。。如果每一个 tester 都花时间去看 code，找 root cause，这个涉及到成本问题，毕竟人工是一个人每 ...

不是我们公司有特殊要求，最 top 的公司很多都这样要求的。对于一个 strong 的测试人员来说，他还需要跟开发人员讨论 solution options，在 fix 之前就发现可能引起的其他问题。别说测试人员了，就是 pm 很多都很精通 coding。他们都会 code review 去报 bug。这里的人工比你们贵多了。还有就是，

white box is to test the design. What design? internal design or funcationality design, or code design? Backbox is to test the syste without going through the code inside. If so, my second example is a not a black box at all. right?

[*本帖最后由 cleverman 于 2007-7-9 01:24 编辑*]

作者: cleverman **时间:** 2007-7-9 01:27
QUOTE:

原帖由 *DERYCK* 于 2007-7-8 21:14 发表
看了以上几位的看法，我觉得还是在测试过程中你是根据外部功能设计的用例进行测试，还根据内部代码的结构设计的用例进行测试。
只要分清楚这两种区别就明白什么是黑盒测试和白盒测试了。
我个人认为定位错误不 ...

不只是定位错误，是要找到错误的 root case。 不然的话，你不能定 bug 的优先级别和严重程度。
比如我不久前发现了一个 bug，会导致 windows 重启。你怎么定优先级呢和严重程度呢？windows 重启也够常见了吧？
因此一定要找到 root cause，最后我们发现是一个安全漏洞，可能被黑客利用进行攻击。因此就给了最高的优先级和严重程度。开发人员当天就 fix 了。
这就是我们为什么要求测试人员有这个能力。大公司里，老板不可能管任何事情，很多都需要测试人员来 drive。但是，你必需要有个好的判断，不能出错。
[*本帖最后由 cleverman 于 2007-7-9 06:45 编辑*]

作者: cleverman **时间:** 2007-7-9 06:43 **标题:** 回复 #15 seifer1754 的帖子
随便 Google 一下 definition of black box and white box.
Blackbox:Also known as functional testing. A software testing technique whereby the internal workings of the item being tested are not known by the tester. For example, in a black box test on a software design the tester only knows the inputs and what the expected outcomes should be and not how the program arrives at those outputs. The tester does not ever examine the programming code and does not need any further knowledge of the program other than its specifications.
里边明确说明了，在黑盒测试中，测试人员不知道 internal working, 而且测试人员只知道输入和输出，不知道程序怎样得到这个输出。测试人员从来不检查代码，而且不需要任何规格说明之外的知识。我想我举的第二个例子，和黑盒测试的定义是相矛盾的。从我的理解上来看，我举的例子是不应该属于黑盒测试的范畴。
White Box：Also known as glass box, structural, clear box and open box testing. A software testing technique whereby explicit knowledge of the internal workings of the item being tested are used to select the test data. Unlike black box testing, white box testing uses specific knowledge of programming code to examine outputs. The test is accurate only if the tester knows what the program is supposed to do. He or she can then see if the program diverges from its intended goal. White box testing does not account for errors caused by omission, and all visible code must also be readable.
里边说明了，internal working 是需要的，要有专门的 coding 的知识。从定义上来看，黑盒白盒的主要区

别应该是在 knowledge of internal working 和 knowledge of programming code 上面。也就是说是否你的测试中需要内部工作原理和编程知识。很明显，我举的第二个例子是比较符合白盒的定义，而且跟黑盒定义很矛盾。因此，literally, it should belongs to white box testing. Right?
你的观点，“从 case 设计的角度来划分黑盒，白盒”，有没有理论依据呢？从定义上来说，黑盒白盒指的是一种测试技术。测试技术应该是包括很多东西的说法，设计 case 只是其中一种吧。你可以认为从代码来设计 case 就是白盒测试，可是是否能说不是从代码设计 case 就是黑盒测试呢？毕竟设计 case 只是测试过程的一小部分。比如，我从功能设计 case，可是在执行 case，发现 bug，定位 bug，finding root cause 的过程中，用到了大量的 coding 技术和内部工作原理。也就是白盒测试定义中提到的技术，难道还是算黑盒测试吗？

作者: cleverman **时间:** 2007-7-9 07:08
我想单纯的来区分黑盒/白盒本身肯定不是件很容易的事情，不然就不会有灰盒测试的概念了。
Definitions of Gray box testing:
Gray box testing - Tests involving inputs and outputs, but test design is educated by information about the code or the program operation of a kind that would normally be out of scope of view of the tester.[Cem Kaner]
我第二个例子比较满足这个定义。
Gray box testing - Test designed based on the knowledge of algorithm, internal states, architectures, or other high -level descriptions of the program behavior. [Doug Hoffman]
这个定义好象比较满足你所定义的白盒。从设计测试用例的角度。
Gray box testing - Examines the activity of back-end components during test case execution. Two types of problems that can be encountered during gray-box testing are:
A component encounters a failure of some kind, causing the operation to be aborted. The user interface will typically indicate that an error has occurred.
The test executes in full, but the content of the results is incorrect. Somewhere in the system, a component processed data incorrectly, causing the error in the results.
这个定义非常满足我的第二个例子。是从 case 的执行角度来说的。
Even though you probably don't have full knowledge of the internals of the product you test, a test strategy based partly on internals is a powerful idea. We call this gray box testing. The concept is simple: If you know something about how the product works on the inside, you can test it better from the outside. This is not to be confused with white box testing, which attempts to cover the internals of the product in detail. In gray box mode, you are testing from the outside of the product, just as you do with black box, but your testing choices are informed by your knowledge of how the underlying components operate and interact. ;
这个定义也非常满足我说的第二个例子。
因此，从概念上来看，我说的例子绝对不属于黑盒测试，非常接近灰盒测试的概念。另外，灰盒和白盒也没有明确的分界线，灰盒本身也没有一个统一的定义。不同的定义也有一定的差别。这里我的理解是，

不懂代码，不懂内部工作就是黑盒。
从代码级详细的进行测试就是白盒。
中间的部分就是灰盒。
不过中间的分界线应该是模糊的，很多例子可能很难清晰的去划分。不过我举的第二个例子，应该算是灰盒。

作者: seifer1754 **时间:** 2007-7-9 09:39
Done.....

作者: shanxi **时间:** 2007-7-9 10:35
按分工来说 Tester 只需要找出 bug
但实际工作中，如果在时间允许的情况下，运用 Windbg 等工具对 bug 进行比较准确的定位难道不是一个资深或者高级的 Tester 应该具备的技能吗？
另外，当软件代码行数异常庞大后，整个软件的逻辑交叉也会异常繁杂，这时候已经不适合看某部分代码来 fix bug 的可能性(因为 Cross function 和组件开发部门保密性加强导致沟通很困难)，这时候需要 Windgb 等工具去辅助定位。Tester 也会通过问题的定位及时发现类似的问题。

浅谈测试的分类

经常听到有人说"我现在做手工测试,技术含量低,想转到自动化测试",也有人说"我现在做黑盒测试,技术含量低,想转到白盒测试".手工测试或者黑盒测试就一定技术含量低吗?其实并不见得.自动化测试或者白盒测试就一定技术含量高吗?其实也不见得.这里我想简单谈谈我的看法.

测试的分类目前我感觉并不成熟.比如,你分为黑盒,白盒就不是非常科学,也不能适用于所有的情况,因此出来一个模糊的灰盒的概念.灰盒介于黑盒,白盒之前,但是具体一个人从事灰盒工作到底有多少是白盒内容,有多少是黑盒内容,里边的差别可就大了.把测试简单地分为手工和自动化也并不是非常科学,与黑盒白盒类似,有很多测试也是很难归于手工或者自动化,实际上应该是介于手工和自动化之间的测试也是很多的.因此,与灰盒的概念类似,还应该存在一种测试,我暂且叫做半自动化测试.以上我想说明的是,测试目前很难简单地被分类,而你所从事的测试工作也不应该简单地就归于哪类.下面就谈谈我觉得怎样分类可能更科学,更能体现实际的情况.为了说明方便,我就暂且不谈灰盒和半自动化测试.

请看下表，我把测试分为了 4 大类。其实我们所谓的技术含量低的测试主要指的是“黑盒手工测试”。为什么说他技术含量低呢？因为黑盒测试不需要懂代码，手工测试不需要会编程。总的来说就是不需要编程能力。当然做黑盒手工测试的人未必就水平低，黑盒手工测试也未必就代表技术含量低。请注意，我先前所说的是不需要编程能力，不代表编程能力甚至软件开发的高级能力不能在黑盒手工测试中派上用场。也就是说，如果一个人软件开发能力很强，他即使只用黑盒手工测试也照样可以做出高技术含量的工作，或者说找到高难度的ｂｕｇ。最显著的例子就是黑客了，那些具有高水准的黑客高手很多情况下都是在没有源代码的情况下通过工具的使用来发现那些安全漏洞。区别在哪里？区别就在于他们的技术比我们一般的黑盒手工测试人员的技术不知道要高多少倍。因此，我的意思是，在测试的工作中采用什么测试方法并不能决定这个工作技术含量的高低，高水平的人无论用什么方法都能做出高质量的工作出来。通常我们都会选用最适当的测试方法来进行工作，而我所强调的是不要把注意力过多地花费在测试方法上，而更应该注重提高自己的个人能力，尤其是编程，软件开发的能力。

	手工测试	自动化测试
黑盒测试	黑盒手工测试	黑盒自动化测试
白盒测试	白盒手工测试	白盒自动化测试

另外我想说的是，做手工测试的人也未必一定要转到自动化，你也可以向白盒手工的方向发展，而做黑盒测试的人也未必一定要转到白盒，你可以向黑盒自动化的方向发展。当然了，一个黑盒手工测试人员无论是从横向（黑盒自动化），还是纵向（白盒手工），又或是垂直（高级黑盒手工测试）发展，都是离不开软件开发能力的，这也是为什么我一直强调编程能力的原因。没有良好的编程水平，就只能限制在了低级黑盒手工测试这个范畴了（管理发展路线除外）。

当你具备了比较强的开发能力之后，你再回头看这些测试方法的时候，你应该会头脑清晰很多。正像很多人说的“白盒代替不了黑盒”，或者“自动化代替不了手工”那样。是的，每种测试方法都有自身的优势，都是不可替代的，一个测试人员没有必要去强求只在一种测试方向去发展。一个高级测试人员应该是所有的方

法都会，而且能够选择最恰当的方法去进行测试。而一些黑盒手工测试人员也不应该由于黑盒不可替代，手工不可替代，就不去学习与了解自动化或者白盒测试技术，更不应该去贬低自动化测试 。

总而言之，测试的方法是多样的，测试的发展也是多姿多彩的，敞开你的胸怀去了解与学习更多的测试技术吧。了解的越多越好，理解的越深越好，这样才能使你在测试的工作中如鱼得水，胸有成竹。测试没有最好的测试方法，只有最恰当的测试方法，多了解一种测试方法，你就多增添一份工作能力 。

黑盒测试就比白盒测试技术含量低？

常常碰到有人做黑盒测试，觉得技术含量低，想向白盒测试的方向转。这里我想谈一谈是否白盒测试就比黑盒测试技术含量高呢？我的看法是未必。前几天参加了一个黑客会议，发现几乎 100%的黑客都是通过黑盒测试的方法发现系统漏洞的。道理其实很简单，软件厂商不可能给黑客看源代码，因此黑客只能进行黑盒测试。那么问题就来了，既然黑客基本都是做黑盒测试的，那么我们怎么能说黑盒测试的含量低呢？

其实不论黑盒测试，还是白盒测试只是一种测试方法而已。而测试水平的高低往往不是用使用什么测试方法而区分的，更主要的是要看测试人员自身的水平如何。所谓水平，不仅仅是测试水平，而是对整个计算机技术的一个综合水平，包括了方方面面。诚然，白盒测试的门槛要比黑盒测试要高，可是如果想在测试行业发展，想向更高的水平迈进，未必一定要转向白盒测试。并且，白盒测试也是分水平的，如果你在黑盒测试不能达到一个高水平，你又怎么能说到了白盒测试就会达到一个高水平呢？何况，白盒测试也未必就比黑盒测试更有意思。

测试的意义在于在现有资源的情况下，能够找出更多，更深入的 bug。如果我们有察看代码的权利或可能，白盒测试是一个非常有效而重要的方法，如果我们没有查看代码的权利和可能，我们只能进行黑盒测试，可是我们还是能有很多东西可做的，最简单的就是自动化测试。更深入的我们可以采取黑客常用的测试方法进行测试，比如 fuzzing。据统计，70%的安全漏洞都是通过 fuzzing test 发现的。

最后说一下个人的建议。还是那句话，想做好测试，想做测试牛人，是需要计算机技术综合能力的，不仅仅是什么黑盒测试，白盒测试，自动化测试这些测试方法的掌握，更重要的是开发的功力，系统内核，等等更深入技术的精通。那么对于测试人员来说，不要以为转去搞白盒自己就能一步登天了，抓好计算机的基本功，学习更多，更深入的计算机知识才是关键问题。

希望做黑盒测试的同仁不要弃坐，继续努力，黑盒测试做好也是很有前途的.

黑盒测试比白盒测试更难，技术要求更高

几个月前我还在谈论黑盒测试不一定比白盒测试技术含量低，现在我却可以比较肯定地说，黑盒测试比白盒测试更难，技术要求更高。道理其实非常简单，黑盒，白盒测试的本质区别在于源代码的访问权利，白盒测试具有这种权利，因此也就具有更多的资源和信息进行测试，当然事情就会变得容易很多，而黑盒测试由于不能看到源代码，就使得对于白盒测试人员发现的 bug，你要花更多的时间，并且具有更高的技术才有可能发现。

我做黑盒测试已经 4 年多了，是一个地地道道的黑盒测试人员，可是我具有源代码访问的权利，也就是说，虽然我是做黑盒测试的，但是我所拥有的信息并不比白盒测试人员少。随着我黑盒测试经验和技术的提高，我突然发现我已经完全依赖与源代码提供的信息了，如果没有源代码，我的黑盒测试的工作将会变得复杂很多，困难很多，甚者无法实现。这也让我有了一个强烈的感觉，就是黑盒测试比白盒测试更难。

在 Symantec 出版的一本书《The Art of Software Security Test》里边就有这个说法。这本书我觉得一般般，但是里边体现着这个道理，就是，“对于白盒测试，一个公司可以组成一个测试队伍来进行，而对于黑盒测试，可能就很少有公司有这个能力了，只能去外边聘请专业的公司来作，这个成本是很高的，但是是值得的”。

经常听到有人抱怨“我在公司是做黑盒测试的，没什么技术含量，我的目标就是转到白盒测试”，我一直觉得这个说法是可以质疑的，也希望看了我的这篇文章以后，不要再出现这种声音，更不要再拿它当成自己不去提高测试技术的一个冠冕堂皇的借口了。

为什么我们大多数人，包括以前我自己都会认为黑盒测试比白盒测试的技术含量低呢？那是因为，我们绝大多数人都是在做低端黑盒测试的。很早以前我就曾想过，黑客都是通过黑盒测试的方法来寻找安全漏洞的，我们怎么能说黑盒测试技术含量低呢？随着自己的水平向黑客的方向接近，自己也越来越有更深，更丰富的理解和体会了。

如果我们把刚进入黑盒测试领域的新人的技术打分为 0，而黑客的技术打分为 5 的话，那么根据技术水平我有这样一个列表：

0.测试新手

1.黑盒手工测试

2.黑盒自动化测试

3.具有白盒测试能力

4.安全测试

5.黑客

大家注意，很多人把自己的测试技术的提高依赖于公司，依赖于 team，依赖于 project，这是不对的。我本人在公司的工作内容不过就是黑盒自动化测试，可是这并不影响我可以向更高的方向发展，现在 internet 这么发达，什么资料不能找到呢？各种各样的计算机书籍，网上各种各样的计算机技术交流探讨的论坛，博客等等。很多人觉得跳槽，换个工作自己就能更好的发展测试技术，这也是有误区的。说句实话，个人发展本质上还是个人的问题，并不是公司的问题，或者你的 lead，你的 manager 的问题，一个公司既然要你了，就说明你自己的能力和水平跟公司对你的要求还是比较接近的，公司对你已经有一个期望值了，也就是说你能胜任这份工作了，而再往上的发展并不属于公司对你的期望了，绝大多数的情况还是要靠个人的。因此，我个人认为，无论在任何的工作环境，工作内容的情况，你都是有技术提高余地的，但是这事情要由你自己来 drive，而不要太多地依赖外部环境。我从小到大的学习，主要是靠自学，我很少能集中精力地去听完老师的一堂课。包括现在，我很多 training 都是没听完就走人了，或者有些签个到就溜。我的这个性格造就了我很独立的学习能力，自己为自己规划学习，不知道对大家是否有借鉴作用。

话说回来，因为大家对 0，1，2 级别应该都是比较熟悉的，我想谈谈 3，4，5 级别。

3. 作为一个黑盒测试人员，没有人会要求你不具备白盒测试能力，如果你有源代码访问的权利，那很好，你完全可以利用这个优势，把通过查看源代码得到的信息应用到你的黑盒测试中去。如果你没有源代码访问的能力，这也并不能阻碍你在这个领域进行探索和实践。如果你的项目是 Java，.NET 这种，你可以反编译，如果你的项目是 C, C++这种，你可以反汇编。总而言之，所谓具有白盒测试能力的意思是，发现一个 bug 能够定位到代码里，是什么代码，为什么产生这个 bug？可以进行代码级的测试用例的设计。一般来说，这个级别的要求是具备良好的代码读写的能力。

4. 安全测试与白盒测试的根本区别在于安全意识，黑客的思维。有一本书《Writing Secure Code》里面提到“你可以培训一个人具有测试安全 feature 的能力，你很难培训一个人具有黑客的思维方式，如果你发现了这样一个人，你就 Hire 他”。在这个级别的人要具有良好的安全意识，知道各种各样的攻击方式，当发现一个 bug 的时候要就有安全方面的判断，比如“是否一个安全漏洞”，“是否能够被黑客利用”，“严重程度如何”，等等。同样，自己的测试内容里要包含大量的安全测试用例。

5. 黑客级别要求就更高了。对于安全测试来说，只是分析到“是否能够被黑客利用”，而黑客就要分析“如何利用”以及写出攻击代码进行攻击。至少对我而言，他们要具备非常熟练的汇编编程能力。

以前我认为，要想进行安全测试，或者说做高端测试，多年的开发经验必不可少，实践证明也未必。同理，要想进行高端测试，你也未必要先转向白盒测试。从我个人的经历来说，只要你自己有心，只要你自己用心，你总能发展和提高的，外部环境固然重要，但是起决定因素的还是自己。安全测试完全不属于我的工作内容与职责，可是在一个月的时间里，我已经连续发现 4 个安全漏洞了。如果你在工作中也能够发现你项目的安全漏洞，公司还能如何不重视你？

测试发展

关于测试人员的发展

那我说一下我的看法吧。因为大家都是搞测试的，这里我也只谈测试。
首先，我们可以有两条路发展，技术和管理。管理就是做 team lead, manager, director 这么走。因为我没有走这条路，所以，我这里也只谈技术。而且，即使走管理，也是应该

具备很强的技术能力才行，所以技术是我们的发展之本。我个人不喜欢技术不精通的领导，也不喜欢被这种人管理。

技术的发展是分阶段的，基本上你要是能发展到最后的阶段，工作，钱，房子，车子，老婆都不用发愁了。当然要一步一步走，不可能一步升天，而且一路走过来也不是很容易，

应该说大部分人可能都达不到。不过只要你肯努力，坚持不懈，就一定能达到。

第一阶段：就是基本功的问题。这个阶段从大学入学就开始了，我接触不少人工作几年都没有达到要求。这个要求是一定要达到的，不然以后没法往高发展。大学的一些课程一定

要学好，主要是数据结构，算法，数据库，操作系统，计算机网络。争取精通两门。数据结构，算法对软件开发非常的重要，很多大公司面试就考这些。你不过关，根本通过不了

面试，一两道算法题一下就把你难住了。另外，我可以告诉你，顶尖公司的面试 80%都是考算法，你有没有经验不要紧，做没做过项目不要紧。关键是考察你的基本功，基本功打好

了，其他工作就都容易很多了，基本功打不好，什么都白说。操作系统，争取要精通 windows 或者 linux 内核，看你走哪条路了，我是搞 windows 的，不过他们之间很多地方也是相通

的。计算机网络，争取精通 TCP/IP 协议。数据库我不怎么懂，我的理解是要精通 oracle, sqlserver, 还有 sql 编程。
另外就是编程技术了. C,C++,面向对象一定要搞懂，搞熟。大公司面试的算法就是要你用 C/C++实现的。这些搞熟了，学习其他语言就是几个小时的事情。（我指的是上手，不是精

通）。这些东西搞不透，不管你其他语言用多少年，回来学他们还是难。
再有就是英语水平了，听说读写，各个方面都要达到要求。技术到了一定程度，英语对你的发展就起到了非常决定性的作用了。你英语好，就可以去外企，就可以外派出国，甚至

在国外发展。
以上这些都是在大学应该掌握好的。当然了，能在大学掌握好这些的毕竟是少数。这些少数人就是去了微

软，google 的那些，一毕业就拿到月薪上万工资的。大部分人都是达不到

要求的，这没关系，毕业后一定要找时间把这些基本功补上。不然的话，在下个阶段的发展就很受限制了。

第二阶段：计算机知识的扩展，行业知识的精通。这个阶段从你大学毕业走向第一个工作岗位开始。工作之后，发现计算机的世界比大学的知识要博大精深很多。一开始工作，就

要拼命吸收以前没有接触过的，新的知识。这个就不多说了，大家都会有很多感受的，会觉得很多东西都不会，不会就学。以后你跳槽去面试，人家就会看你工作几年，这几年干

什么了。工作 1，2 年之后，很重要的一件事情就是要选择一个行业了。也许是你现在正在从事的行业，也许是一个新的行业。总之，你自己要为自己规划，选择一个适合自己，而

且又热门，以后有发展的行业。无论是现在的行业，还是跳槽到一个新的行业，都需要你开始积累在这个行业的经验了，要精通这个行业。有这个基础之后，就要去这个行业里 top

的公司了，国企，外企都可以，一定要有名气，大公司。比如，通信的华为，搜索的百度，等等。如果你精通了这个行业，去这些公司不是很难。
另外有一点很重要，如果你本科不是一所名校毕业的话，争取能上一个名校的研究生，全职，兼职都可以。这样可以为下一阶段做好充分的准备，否则的话会有比较大的困难。总

之了，是自己的短处都要想办法去弥补，不然发展总会受限制。

第三阶段：国际著名大公司。有了前两个阶段的积累，加上自己的英文水平，就要找机会进入国际的大公司了。相信这个时候就会有很多猎头来联系你了。选择你这个行业的世界

前 3，最好是第一或者第二。进去之后要学习两个方面，一是英文，中国人可以学一辈子英文的。另外一个就是大公司的管理。可以这样说，国际大公司的管理有很多类似的地方，

因此他们的招聘非常愿意招其他国际大公司的职员。这就是为什么，你一旦踏上一家公司，一辈子都不用愁工作了，可以在这些大公司跳来跳去，工资节节高。到了这个阶段，你

基本上可以有个比较不错的生活了，房子，车子都不会是太大的问题。

第四阶段：向国际化发展。如果你还不满足，觉得自己还有能力更进一步，那我就建议你向国际化发展了。中国的工资毕竟有限，到了第三阶段也不过就是 20 万左右，你可能还不

满足。那么你就可以联系国外的公司了，有了你的英文，你的经验，你的背景，到时候就是水到渠成了。我相信国际的猎头也会盯上你的。

最后说一下，如果你现在已经具备了我所说的各个阶段的能力，那么你的简历是任何公司都很难拒绝的了。因为目前的情况，具有这些素质的测试人员在世界都紧缺。很多公司都

招不到人，即使连 google,MS 也不列外。他们都在到处寻找这种人。

最后说一下测试。我一直没有讨论测试的问题，因为我一直没有把测试当作一个难得东西来看待。我认为测试是表面上的，我前边提到的东西要比它重要的多。欢迎大家一起来讨

论。我也是进入测试才 2 年多的时候，其中大多数的时间也像大家一样的迷惘，很多时候也很悲观。不过通过自己的努力，最后终于得到了一个满意的结果。我发现自己对测试这个

行业的理解和很多人都不同，希望我的理解能给大家一点帮助。

测试职业发展的三“步”曲

进入测试行业也有两年半了，从一点不懂，迷惘的状态，到了现在也有些自己的理解了。常常听到有人问各种各样的问题，不同的人也对测试表达出不同的理解，甚至很多自相矛盾。比如，测试到底有没有前途，测试的待遇如何，测试与开发哪个好，哪个更重要。我们更看到有些人具有测试十几年的经验出口说“测试不需要懂编程”，可是我们又看到很多手工测试人员由于不懂编程而被公司所淘汰。我们常听到有些手工测试人员讲“自动化不是万能的，很多产品不适合自动化”，可是我们又看到很多大公司又只招自动化工程师。因此出现了测试，开发互相看不起，手工测试，自动化测试互相看不起的奇怪现象。正是因为有这么多的问题，这么多的疑惑，也没有人能给出一个很好的解答，或者大家的解答都各不相同，造成了如此的情景。到底隐藏在这些表面现象的深处，根本的东西又是什么呢？我认为，是大家没有真正理解什么是测试。测试不同于开发，开发经过几十年的发展已经到了很成熟的阶段了，因此大家对于开发就不会有很多的争论与问题。测试发展才多久呢？有人说很长了，可是我们想想大学什么时候才开设有测试知识的课程呢？可能现在也不是所有计算机专业都能在大学就接触到测试的知识吧？不然现在也就没有这么多测试培训班了。测试是一个新兴的行业，它正在快速的发展着。因此，我们对于测试的理解，千万不能只是停留在某一个层次。对于测试人员来讲，不能妄自菲薄，也不能坐井观天，我们需要共同的努力去推动测试流程，测试技术的发展，充分发挥自己的主观能动性，发现或发明更好的测试方法，能在测试的发展上留下自己的足迹，自己的贡献。这里说一下测试相对于开发来说比较吸引我的方面。开发人员的工作实际上是比较死板的，一定要根据设计文档来实现，偏离了设计文档就是 bug 了，甚至编程的风格也要按照公司的规范来，因此，工作中并没有太多创造性的东西。而测试人员相对来说就会自由很多，因为测试没有什么固定的模式来做，我们的目标就是保证产品的质量，而手段就是找 bug。具体怎样找，基本上你可以完全来自我控制，手工找，自动化，看代码。因此，测试工作给我发挥创造力，想象力的机会，所以，我喜欢他。

说了很多闲话了，现在步入正题吧。我说一下目前我的状态对测试的理解，谈一谈测试职业发展的三步曲。说一下每一步的工作内容。

第一步：手工测试/黑盒测试：这个大家都是太熟悉不过了，主要是设计测试用例，执行测试用例，发现 bug，报告 bug，验证 bug fix。每一步都有 junior, senior, architect 的区别。junior 刚入门，就是熟悉学习这些东西，这些东西都搞熟了，加上对产品的较深理解就是 senior 了。 senior 要对一些较大的模块能够做计划，能够带领 junior 的一起工作。architect 要能够对整个产品有深刻的理解，可以规划整个产品的测试，包括需要多少硬件，需要什么软件，需要多少人力，需要多少时间，等等。

第二步：自动化测试。手工测试人员和自动化测试人员最大的区别在于懂编程。不过如果你只是会用 script 编写一些程序的话，还不能称之为自动化测试人员，至少还要有软件设计的能力。junior 刚入门除了要学习手工测试的那些知识以外，还要能够使用某种高级语言，某种测试工具自动化自己所负责的测试用例。senior 除了手工测试的那些要求以外，还要能够规划一个较大模块的自动化，能够解决各式各样 junior 在

自动化过程中发现的问题。architect 除了手工测试的要求以外，还要能够对整个产品进行自动化的设计，比如采用什么语言，采用什么工具，各个模块自动化的整合，自动化的 schedule，自动化的 report 等等。

手工测试人员的 title,往往叫做 SQAA(Software QA Analist), junior SQAA, senior SQAA, principle/staff SQAA。

自动化测试人员的 title，往往叫做 SDET(Software Design Engineer in Test), junior SDET, senior SDET, principle/staff SDET。

还有更常见的 title，SQAE (Software QA Engineer), 是处于这两者之间的，既要手工测试，也要懂得自动化测试。基本上大多数的测试人员都是发展在这条 path 上。因此，你可以看看自己，如果是 SQAA，就要往 SQAE 的方向发展，如果已经是 SQAE 了就要往 SDET 方向发展。不同的 path，虽然有不同的级别，但是工资也是有区别的。比如 senior SQAA=junior SQAE, senior SQAE=junior SDET。而且，不同的 path 可能最终能够发展到的级别也有区别，比如 SQAA 可能就不会设有 principle SQAA 的级别。也就是说，如果想达到 architect 的级别，只是会手工测试是远远不够的。

达到 Senior SDET 应该就是比较高级的测试人员了。编程序，自动化这些都是小菜一碟，就是跟开发人员比起来也能做一个准 senior 的 developer 了。可是这还没有发展到头，以我现在的观点来看，还有第三步。

第三步：安全测试。我们知道各式各样产品最终发布出去最头疼的并不是用户找到多少 bug，而是安全问题。很多知名大公司发布产品后，还要投入大量的人力去进行安全漏洞的修补。安全漏洞严格来说也是质量问题，那么这些安全漏洞有没有可能在产品发布之前被测试人员所发现呢？答案是肯定的。因此作为我们测试人员的话，把手工测试，自动化测试精通之后，就要努力向安全测试的方向发展了。具备有安全测试能力的工程师基本上都可以称之为测试专家了。这需要有非常强的编码能力，非常深的系统内核知识，甚至黑客的背景。更重要的是，要随时能够从安全的角度来分析产品的质量。我们要了解程序员实现的具体方法与步骤，结合 review 他们的代码，大量的试验来发现安全漏洞。这里举个例子，前不久学习一个文件加密的实现过程，发现它会把每个要加密的文件做一个备份，加密之后再删除这个备份。备份的作用是一旦加密失败，数据可以被恢复。那么我当时就考虑，这个备份删除之后，是否内容还留在硬盘呢？后来经过试验发现，确实内容还存留在硬盘上。这就是个安全漏洞，虽然你的文件加密的，可是黑客还是可以通过找到以前备份文件的内容来得到敏感的信息。

以上是自己对测试现状的理解，自己可以怎样发展？自己也正在有意识的向第三步发展。我觉得测试人员一定不要停留在自己的目前技术水平，技术没有尽头，上面的发展空间还非常的广阔，也许还有四步曲，五步曲.....

领导奇怪我为什么不跳槽？

公司不是大公司，我是第一个海归，我后边又来了两个海归，不过都是在短时间内就跳槽了。从此之后，公司基本决定不再招收海归了。
很长一段时间之后，在跟 HR 闲谈的时候，谈到了这个话题。她告诉我，领导很奇怪我为什么没有跳槽。我当时并没有怎么回答这个问题，回头想想，可能是这个原因。
之前工作过 3 个公司，没有一个超过一年的，各个都是不同的行业。因此自己的感受特别深，跳槽很不利于行业经验的积累，往往要从头做起。而我当时可以说已经得到了除了行业经验之外的所有东西。因此我的目标就很明确，就是要积累行业经验，在自己选择的行业里精通下去。当然当时的目标是开发不是测试，后来目标就调整为行业经验+测试经验。而且，我相信只要在自己的领域做到精通，以后待遇都不会有任何问题。因此一直工作了 20 个月，自己感到成熟的时候才开始考虑。当然最后事实证明也跟我当时的想法基本一样，这个行业世界排第二的公司在面试了我 15 分钟以后就给了我 offer,并且 salary 远远超出了我的预期。并且最后 contract 上的 salary 又比当时许诺我的高了一些。关于待遇问题，我没有 negotiate,他们问我要多少，我的回答是，“你们根据我的水平来给就可以了，我个人没有什么特别要求”。关于面试，他们主要让我谈了对这个领域的看法，测试方面并没有问多少，这点也验证了我所说的，测试只是个基础，行业知识的精通才是关键。

希望我的经历对大家的跳槽决定能有所帮助。

测试人员如何能够受到重视？

常常听到有人抱怨说公司不重视测试，不重视测试人员。没错，这个在小公司是非常常见的现象。其实，即使工作在大公司，虽然测试和测试人员的地位得到了很大程度的提高，可是相对开发和 PM 的地位还是有一定的差距，也就是说测试在公司中的地位还是比较低的，这个其实是不争的事实。道理也很简单，测试人员的技术水平，测试工作的难易度和测试在产品开发过程中的相对地位都决定了这个现实。由于测试的性质造成了相对的地位低下，那么如果想使得自己作为一个测试人员在公司或者项目组的地位得到提高，一个行之有效的办法就是超越测试的工作范围。简单来说，一个测试人员在工作中的重要性的大小不是仅仅由测试的工作范围来决定的，更重要的是你能够在多大程度上去 cover 开发和 PM 的工作。我知道在很多很多公司，包括很多大公司，都不需要你这样去做，我也不期望很多测试人员会这样去做。可是我最近理解到，去 take 开发和 PM 的 responsibility 对于个人的发展是多么的重要。由于各个公司的测试情况千差万别，个人的测试发展之路也是各式各样，这里主要是谈个人的理解，很可能只适合少数测试人员。这里先讲一些事例：

1。本人以前在一个世界前几软件公司中担任 team lead, title 是 senior SQAE，来到现在的公司是按照最低级别录用的，也就是 entry level。为什么差别这么大？主要是各个公司对测试人员的要求差别太大了，这里如果只是满足测试的工作内容的话，都很难升入中级，我可见到不少水平不错，工作好几年的员工连个中级都不是呢。可是这样的员工跳到国内的公司就有做 director 的。

2。以前也给大家介绍过我问 director 测试应该如何发展，他的回答是"短期要学好 C，长期还是学好 C"。可见他的回答完全跟测试没关系，应该是完全是开发的范畴。

3。在一次会议上有人问 director 的老板，测试 senior 的实在太少了，如何才能发展成 senior。他的回答主要是强调测试人员要更加的贴近客户，从客户的需求去考虑问题，不能局限于技术。可见他强调的又是 PM 的范畴。

4。自己的老板对自己提出的要求也是完全超出了测试的范畴，基本点如下：

- Debugging: find root cause of a bug （开发）
- Code review （开发）
- Answer customers' questions (PM)
- Researching competitors' products and showing options in functional spec (PM)
- Don't only focus on my components, responsible for whole feature

5。一个印度 PM 同事就很牛，工作就是超出 PM 的范畴，开发的东西他知道底下具体是如何实现的，出了什么问题他都能估计出可能是什么问题。测试的设计他也能提出很多好的建议，很多测试的 case 也得请教他。他就升的特别快，几乎一年至少一级的往上升，最近发现都 senior 了。

以上的例子只是想说明作为一个真正出类拔萃的测试人员各方面的功力都不能少。很多人还在争测试，开发的地位高低，水平高低。其实对于高级的测试人员和高级的开发人员来说，他们的技术视野都应该是比较一致的。因此，如果作为一个测试人员真的想提高自己的地位，就不要把、开发和测试对立起来，要把他们融合在一起才对。最后想说的是，能够把这些都做好的人不会太多，那个印度 PM 也算是很少见的了，他是真的很努力，负责。我本人以前也习惯性的局限在自己的任务范围之内，看到是其他人的工作就不管了，幸亏得到老板的指点，最近无论是什么问题，只要是跟自己产品相关的都积极主动地去关心，思考和处理，感觉进步很大。也希望能尽快的达到老板的要求，就是“只要是这个产品的问题，别人第一时间就会想到去问你”。我想这个时候，自己的重要性也无需争辩了。

以上是一些个人心得，不知道是否对大家有帮助。

微软的 Principle SDET 到底是什么样的牛人？

如果你要问起微软的测试人员关于 Senior SDET 的话，可能很多人都会说没见过，或者很少见。如果你要问起微软的测试人员关于 Principle SDET 的话，可能立即就会有人指出微软不存在这样的人。可见，在微软 Senior SDET 都是凤毛麟角，就更不要说 Principle SDET 了。那么微软到底是否存在 Principle SDET 呢？如果存在，他们又会是什么样的人呢？这是很多人心里的一个很大的疑问，尤其是对于想在技术这条路上（区别于管理的路）提升自己的测试工程师。本人有幸接触到几个这样的牛人，这里我想分别从他们的教育背景，工作经历以及他们对于测试人员的建议来归纳总结一下。里边有些人的背景可能离我们很远，我基本上是由远及近的顺序来介绍。

TV

教育：本科三学位：计算机，数学和物理 （是不是很牛呢？）研究生：计算机硕士

Before MS: 第一家公司做操作系统开发，第二家公司做应用科学家，研究领域包括图像处理，编译器以及模拟器，第三家公司做卫星分析工具的开发

MS: 2001 年加入微软做开发 lead， 创新一种高速远程文件系统，并且获得操作系统和数据库方面的专利无数，后转行做 Principal SDET。

建议：

1.Deep product and scenario knowledge (personal knowledge, customer interactions)

2.Strong analysis skills (RCA, Debugger, cause-effect)

3.3P’s – Persistence, Passion, Positive Attitude

4.Leadership skills (vs management skills)

5.Fast learner and willing to share that knowledge with others

总结：

这位无论从教育背景还是开发背景都是牛的不得了。本科的计算机，数学和物理的三学位就可见他的个人能力是多么的突出了，加入微软之前的开发经验也不是闹着玩的，因此进入微软就是 team lead。在微软的成就也不是一般人所能比拟的。这种人想做什么不行呢？

JM

教育：本科

Before MS: 一年开发

MS: 1996 年加入微软做测试，然后 lead，中间曾经做过测试经理，然后又回到测试 lead, 一共在五个不同的 team 做 lead/manager 之后，做 Test Architect，最后做到 Principle SDET

建议：

1.Dependable

2.Productive

3.Persistent

¤"Relentless individual contributor" – a prev. mgr.

4.Pragmatic / Analytical

5.Maleable / Adaptive

总结：

这位测试和管理的经验都极其丰富，在微软这么多 team 都混过，这么多职位都混过自然很不简单。都已经做过测试经理了，虽然跟上边那位好像还有差距，不过做 Principle SDET 也是顺利成章了吧？

BK

教育：本科双学位：计算机和机械

Before MS: 在三家公司做过，包括医疗，网络设备等等。具体工作内容不详。

MS: 加入微软 9 年，一直工作在一个 team

建议：

1.Ensures tests owned are highly reliable and stable

2.Possesses in-depth component/feature knowledge

3.Possesses strong debugging skills (can narrow down issues)

4.Works towards continuous self-improvement

5.Provides critical feedback during Spec / Design reviews

总结：

这位在微软一个 team 一做就是 9 年，真是兢兢业业呀。从我对他的接触来看他，他在技术以外的东西也非常厉害。有时间我好好整理一下他的语录研究一下。

LV

教育：计算机本科

Before MS: 罗马尼亚计算机研究所工作１３年，具体工作内容不详

MS: １９９７年加入微软做测试，其间换过三个ｔｅａｍ

建议：

1.Possesses in-depth product knowledge AND is big picture expert as well

2.Most of the time works in pro-active mode (anticipates rather than reacts)

3.Able to generate support from others in order to achieve the desired outcome (impact and influence)

4.Understands the business

5.Demonstrates strong leadership skills

别的不说，他加入微软之前就已经在研究所工作１３年了，加入微软也已经１１年了，２４年的工作经验做 Principle SDET 是不是会水到渠成？请问中国有人踏踏实实搞技术这么多年的人物吗？

最后说一下，好好看看他们的建议吧。有几点都是好几个人不约而同提到的，对我们会有很大帮助的。

牛人为什么要做测试?

不久前写了一篇文章<<微软的 Principle SDET 到底是什么样的牛人？>>. 有个网友问了一个问题,我觉得非常的好,这里想简单解答一下. 这个问题是

“够牛。不过。为什么非要做测试呢。Principle SDET 相当于什么职位（与管理职位对比）当了 Principle SDET 还要受一个小 teamleader 领导，受得了吗？”

首先，Principle 是一个级别，很难跟管理职位相对比，级别主要决定了工资水平。从管理的职位上分，有 Principle Team Lead, Principle Manager, Principle Director 等等。级别是不会降的，比如如果一个 Principle SDET 想做 Team Lead，那就会是 Principle Team Lead，如果想做 Manager，那就是 Principle Manager。因此对应的是级别，而不是职位。当然各个职位的侧重点不同，比如 Principle SDET 当然侧重于技术了，未必能一下子转成 Principle Manager。但是，可以先转 Principle Lead, 再转 Manager，从而在管理的发展上能够循序渐进。从我个人的理解上，Principle SDET 是不可能转成 Senior Lead, 或 Senior Manager 的，因为这样就降级了。微软的发展一般在早期就会确立路线，技术或管理，因此一般某人会在一条路线上坚持下去的。在高级别的技术和管理来回转的情况应该很少，但不排除有全才在两方面都很出色，当然就可以转来转去了。

由于 Principle SDET 的级别已经很高了，他们不可能被小 team leader 领导，至少也得是 Principle Team Lead 领导。我查了一下这四个人，其中两个是被 Principle Test Manager 领导，一个是被 Partner Test Manager 领导，一个是被 Test Director 领导。因此不存在受不了的问题。

对于为什么要做测试的问题，牛人 TV 曾经做过一些解释，我个人很赞同他的观点，当然只有牛人才能从这么高的角度去看待测试。

¨Why move from development to test?

- Hardest problems for Microsoft right now are in test
- Test is wide open and needs leaders

- Breaking code is as much fun as building it
- You get to write more code in test (10:1)
- Continuous ship cycle
- Really know the product from the customer perspective
- Get to solve more complex issues earlier in career

我根本不算牛人，我也觉得测试技术含量还是比开发低（从我的层次上看），但是经过了 3 年多的测试经历，我相对更喜欢测试一些，其中主要的原因如下：

1. 测试的工作非常的灵活：开发相对来说压力太大，PM，TEST 都盯着他，程序不能按进度完成就必须要加班，程序的编写一定要按照各式各样的规范，更多的时间是 fix bug 而不是 write code。而测试人员的工作就没有太多固定的模式，完全可以按照自己的想法去进行，比如自己安排自己的时间，进度，自己决定什么 case 手工，什么 case 自动化。自己可以选择自己喜欢的测试工具，编程语言等等。
2. 测试的生命周期更长：开发一般专注于某种技术，或者某类技术。一旦市场上淘汰这种技术，他们就存在很大的转型的痛苦。并且，想跨行业跳槽也相对来说很困难。而测试一般不需要对某种技术做非常深刻的研究，因此有大量的时间去接触其他的技术，加上测试工作对技术的深度要求不算太高，转行相对要容易很多。因此，测试人员搞技术的生命周期更长，职业发展也更灵活 。
3. 测试行业还很不成熟，里边有大量潜在的机会。
4. 测试人才相对来说比开发水平要低，因此更容易上位。

测试的缺点：

1. 毕竟工作技术含量有限，在公司的重视程度不如开发
2. 职位的晋升和工资的水平相对开发还是有差距
3. 测试行业的普遍环境还不是很好，仅仅几个大公司才能给你充分发展的机 会

我个人的想法是要测试，开发两手抓，两手都要硬。在大公司搞测试挺好，万一因为什么原因离开大公司，去小公司就要做开发了，因为我不相信小公司能给我提供发挥我测试技术的平台 。

步入安全测试（兼谈个人测试技术发展轨迹）

进入测试领域已经满四年了，最近感觉自己可能有点里程碑似的改变了，因此回顾一下四年的测试技术发展轨迹供大家分享。

有个网友说的很好，从测试的难度上来讲应该是自动化测试<性能测试<安全测试。我从来都是忽略性能测试的，原因也很简单，因为我基本没有从事过跟性能测试相关的工作，谈不上什么理解与感受。但是，我还是能感觉到性能测试的难度确实要大于自动化测试。一两年前还在热烈地跟大家讨论自动化测试的重要性，如何自动化测试等等，现在已经觉得没什么意思了。手工测试，自动化测试，黑盒测试，白盒测试，等等不过都是一种种测试方法而已，我们的目标很简单，就是要发现 bug。而在发现 bug 的过程中，你是不可避免的要用到任何可能的测试方法和测试工具，包括自己编写测试工具。单纯地讨论他们孰优孰劣，哪个更高级并没有根本的意义。下面回顾一下自己的四年测试发展路径。

- 第一个半年忙于手工黑盒测试 （很枯燥）
- 第二个半年开始考虑自动化测试 （很迷惘和无奈）
- 一年之后有一些自动化的 ideas， 学习 C#, .NET, TestComplete 等等，把自己设计的自动化平台实现 （自己对自动化有一些自己的理解和实践）
- 一年半之后在以前的基础之上开始寻找更高端的测试职位，实际就是寻找专门的自动化测试的职位（选择面很小，适合自己的也没几家公司，尤其是在国内,被迫出国发展）
- 将近两年的时候如愿以偿，入职自动化测试的职位 （开始把自己对自动化的理解和新的公司和同事交流，发现还是 match 的）
- 入职之后的一年内，也就是进入测试的第三年，在自动化测试方面有了大量的工作经验 （觉得自动化测试还是比较简单的工作）
- 进入测试的第四年开始按照老板的要求进行 debugging 和 code review 的工作 （在自己的 feature 上做到了测试支柱）
- 四年之后也就是现在忽然发现自己有了比较大的进步，把以前零零散散的东西都能够联系起来了，感觉自己真正的走入了 right path。最近一两个月的提升好像顶以前半年的，debugging 技术已经比较熟练，code review 也觉得比较轻松，并且在 debuging 和 code review 的过程中任何不明白的东西都会尽力去搞明白，这样在 C 语言，Win32，Windows Kernal，汇编，编译，还有 Security 方面都有了一个突然提升，在脑海里这些东西变得立体起来。也是最近才明白为什么以前 director 给我的建议是“短期学好 C 语言，长期也是学好 C 语言了”。

以前也曾经 hack 过一个网站，并且在与那个网站的 developer 的较量中明显占据了上风。以前也曾经发现过 security 的 bug，但是基本属于瞎猫碰倒死耗子的情况。周六晚上突然有点 idea,由于太晚就没有回公司。周日早上 5 点多睡醒脸也没洗就去了公司，一直工作到 12 点多，file 上了 bug。其实大概工作了一两

个小时就攻击成功，可以远程把一个 server crash 掉。但是为了弄明白里边的一些情况以及这个漏洞的范围和黑客攻击的难易程度，又花了大量的时间。

终于有了一些安全测试的感觉了，而回想起来也完全是由于前几年测试工作中知识，技术和经验的积累所成就的。以前一直觉得没有多年开发经验的不太可能做到安全测试，现在自己证明也未必，虽然确实不容易。最近两年由于英文的障碍使得自己的发展慢了不少,虽然也是 average 的速度，有些遗憾。更大的遗憾在于我 30 岁之前浪费了太多的光阴了，不说大学之前，就是大学之后到 30 岁的这段工作时间也基本上是浪费，没有真正地去深入过任何领域。30 岁之后才寻找到了自己正确的道路，是有些悲哀，但总比没有找到强。我知道很多测试同行都像我以前那样迷惑，迷惘。希望大家在黄金年龄的朋友不要让青春太快的流走，抓紧时间去寻找自己的道路吧。

自动化测试

关于我的自动化测试系统

首先，这里介绍的是 architecture,我基本上想设计一个 open, flexible, extensible 的系统。并且我也希望能学习新的知识和技术，因此用到了对我来说比较新的东西。

系统共分 4 层，所以是一个 layered framework。从上往下，依次是 application layer, service layer, module layer and environment layer.

1. Application layer: 就是一个网站，作用是 system configuration, test case creation, test case distribution, test case scheduling，test execution monitoring, and test results reporting. 也就是说在这个网站上进行测试系统的配置，测试用例的生成与分发，schedule, 监视测试用例的执行情况，还有就是观看最后生成的测试报告。

   用到的技术有：C#, ASP.NET, ActiveX, Web service, SOAP, XML。

2. Service layer: 就是一个 Web service, 作用是从 Application layer 接受到所有的测试任务（写在 xml 里），负责解释 XML，并且调度测试机，给测试机发命令，回收测试结果，组合结果返回给 Application layer.

   用到的技术有：C#, ASP.NET, Web service, SOAP, XML, TCP/IP, Client/Server.

   另外一个在 Service layer 的是一个 Windows service, 它的作用是接受 Web service 的命令，执行并返回结果。

   用到的技术有：C#, TCP/IP, Client/Server.

3. Module layer: 就是一个动态链接库 dll, 负责给 Service layer 提供 basic interface, 把一些公用的功能封装起来。

   用到的技术有：C#, Dll, TCP/IP, XML。

   这层是整个系统的技术核心，其他层都是从这个层扩展出去的。

4. Environment layer: 就是测试环境层。这一层就完全是根据所测试产品的需要来设计了。我的设计是一台 real machine 上边运行 VMware, 这台 machine 的作用不是进行测试，是负责根据需要打开不同的虚拟

机。真正的测试是在虚拟机上进行的。虚拟机上的测试有 command line 和 GUI 两种。 Commandline 就直接运行，GUI 的测试是用 Test Complete + Jscript 来实现的。

用到的工具与技术有：Jscript, Java, Test Complete, VMware workstation.

有一些关键的设计有一定的原因。Application Layer 就是一个网站，它具有能力去集成所有需要测试的项目。每个项目需要具有一个 Web service,因此多个项目就会有多个 Web service。每个项目的的 XML 文件的输入格式都是不一样的，这个可以在 Application layer 在自定义。Application layer 只是负责编辑 XML, Service layer 需要进行解释，也只有它才能进行解释。这是因为，每个项目的测试工程师才真正的明白他们想测什么，想怎么测。由于 Service layer 把不同的项目分隔开，因此，不同的项目也可以设计不同的 Environment layer。测试的 logic 是掌握在 Service layer 的手里。

因此整个一个测试的流程是这样的：在网站上自定义 XML 的结构，因此生成相应的 Test case 界面。进行系统和 test case 的设计，生成相应的 XML 文件。通过 Web service, 传送 XML 到 Service layer, Service layer 解释 XML, 一个 case，一个 cased 的去执行。每一个 case 需要不同的虚拟机，Service layer 告诉那台 real machine 去打开相应的虚拟机，之后告诉虚拟机需要执行什么样的任务。虚拟机启动 command line 命令，或者调用 Jscript 脚本通过 Test complete 来执行 GUI 的操作。之后返回结果给 Service layer。所有 case 执行完毕，Service layer 生成最后的报告传递给 Application layer。Application layer 负责显示报告给用户。还有就是 Service layer 间隔一定时间把测试情况返回给 Application layer, 这样用户可以监视测试的状态。

我对 UI 自动化测试的理解

引用一位很好的同事也是很好的朋友的一句话"UI 的自动化，听起来很神秘，学起来很简单，真正用起来却很困难"。通过自己的经历，我很赞同这句话。最开始确实觉得很神秘，可以用程序来控制鼠标，键盘去操作软件，以前从来没接触过。后来学了一下几个流行的测试工具，感觉没什么东西，就是 record and play。可是，真正用到项目里的时候确实是困难重重。这里想谈一下自己的感受，这方面不是专家，不过应该给测试的新手能有所帮助。

UI 自动化最关键的一点是要选择一个适合自己项目的工具。每个测试工具都有它的优点，有它的缺点，每个被测试的项目也有它自己本身的特点。比如，项目是用什么语言编写的， C, C++, Java, or C#? 还有就是项目是什么类型的，Desktop or Web Application? 很难说一种工具就可以搞定所有或者大部分的项目，也很难说一个项目就能单纯的靠一种工具来搞定。也不太可能你专门开发一个工具来 100%或者 90%以上适合自己的项目，除非你的公司是微软，Google 才有这个实力。因此对测试人员来说就有两个要求，一是要掌握尽可能多的工具，要了解它们的优缺点。这样才能在不同的项目中，一个项目不同的 components 去合理的应用它们。第二就是要有一定的开发功底，在测试工具不能胜任的时候，自己开发工具来作为补充。当然更可能的情况是每个公司只是拥有一种工具的 license, 你没有选择的权利，这样你的开发能力就更加的重要了。（这里所说的开发能力不是自动化脚本的编写，主要是指 C, C++, 至少是 C#, Java 的开发能力）。

下面说说如何去使用测试工具。最初接触就是 record and replay， 感觉非常的简单。也碰到有些人竟然认为自动化测试就是 record and replay。我必须说他们很无知，有这种思维的人可能以后都很难成大器，因为他们理解问题的能力太浅显了。希望论坛没有朋友会这么认为。（这里我说话不太好听，是因为为这种人生了太多的气了，希望大家谅解）。其实，我们 record 的 script 基本上每一句话都需要进行修改和优化。

UI 自动化最重要的一件事情就是得到要操作的对象，比如一个 textbox or button。必须先能够访问他们，得到他们才能够操作他们。这其实也是 recording script 的唯一的用途，告诉我们如何能够得到这个对象。这里会有两个问题，一是测试工具不能够得到这个对象。另外就是测试工具的脚本得到了这个对象，可是在 replay 的时候，对象却不存在。可以说 UI 自动化最核心的 chanllenge 就是是否能够得到对象了。得到了对象其他的相对来说都会容易很多。那么如果出现这两个问题怎么办呢？首先要分析是谁的问题？如果是测试工具本身的对象识别能力的问题，那只能找其他的方法绕过去了。比如自己用高级语言编些程序，或者用其他的方法来跳过操作这个对象来执行同样的操作。这个地方最能考查一个人水平的高低了，有些人束手无策，有些人就能够想到有效的办法。而且这个地方的 chanllenge 往往比一般的开发人员的工作要有难度。如果是程序本身的问题，就可以报 bug 了，让开发人员来修正。其实，程序的 accessibility 的 bug 是很多的，而且大部分公司或者开发人员都不重视，也许你报了也没用，没人理。可是他们如果不 fix 就会 block 你的 Automation。**对 accessibility 的重视程度也可以看出一个公司的自动化测试水平的高低**。

以上讲了 UI Automation 的核心问题：怎样得到要操作的对象。有问题的话，解决办法是开发人员 fix, 等待测试工具的升级，自己想 workaround。得到了对象以后的 chanllenge 更多的是测试流程的控制和对异常的处理。自动生成的 script，操作之间的 timeout 是定死的，这也是我们需要修改 script 的最重要的原因。我们要操作的对象，往往在 timeout 之后还没有出现，或者还没有 enable。这个时候自动生成的 script 在 replay 的时候就会出错。那么我们就需要修改代码来等待足够的时候，一般的测试工具都提供了这样的功能。注意不要只是 Sleep，这样的话无论这个对象出现的多快，你都是一定要等这段时间的。通常都是，约定一个时间，对象出现或者可用后就立即返回，无须等待剩余的时间。如果最后还没有出现就抱错吧。还有一个比较重要的问题就是其他意外窗口的干扰，比如一个窗口突然出现盖住了你要操作的对象怎么办？这时候你需要把你要操作的对象重新激活到桌面的最前面，这样鼠标才能正确的点击到目标的身上。还有一个比较好的办法就是不用鼠标去点击，调这个对象的 click 函数。也就是说，平时是鼠标点击然后激活 click 函数，我们可以直接用程序激活 click 函数。这样，即使目标被其他窗口盖住，也可以正确的执行你期望的操作。

以上是我想到的 UI 自动化的一些基本问题。以我的经验来看，你在自动化的过程中还是可能出现很多你意想不到的问题。这些问题可能的原因也是千奇百怪，可能是软件的 bug，可能是测试工具的 bug，甚至可能是操作系统的 bug。对这些意想不到问题的解决能力非常重要，一方面通过测试经验的积累，另一方面就一定要靠你的想象力和创造力了。从自动化测试遇到的这些难题来讲，比一般的开发工作要困难得多。这里也说一下，水平高的测试人员不比开发人员差，比他们还要强。当然了，开发也有很多的难题，总的来说还是会比测试遇到的难题更多更难。因此，**开发，测试没有水平高低的区别。真正的水平就是对难题的解决能力如何。**

一般来说，desktop 软件要比 html 网页容易一些，windows 程序要比 java 容易一些。也就是说一般的测试工具对 windows deskop 的支持都比较好，对 jave 和 html 的支持会比较差。这是因为微软对 programatic accessiblity 有着严格的定义与要求。因此，如果你测试 windows 程序，自动化起来就省力多了，测试 java 程序就麻烦多了。我当时选择的 Test Complete, 因为公司只有 Silktest 的 license,虽然对 java 支持，可是支持的很不好，也许是版本太低了。而我发现新版的 Test Complete 竟然增加了对 Java 的支持，而且效果还不差，就选择了它。Winrunner 我试过，好像得需要插件，我没有。Robot 我忘记怎么回事了，好像也不是很好用，装不上还是怎么了。不太清楚他们对 Java 的支持如何。个人对 Test Complete 的功能，价钱，灵活性还是比较满意的，可以用来做一些东西，大家如果没经验可以从这个工具入手，建议用 jscript 语言。

再谈 UI 自动化测试

最近还是发现有一些文章，个人对于自动化测试报有很大的怀疑态度，本人也对相关的文章给与了驳斥。我个人和公司对自动化测试都是报有很积极的态度的。这里我想再次的写一篇文章来阐述到底 UI 自动化测试可以做什么，作为一个优秀的 UI 自动化测试工程师应该具备有什么方面的技能，以及本人对 UI 自动化的一些经验和体会。

首先还是要强调一点，API 和 command line 程序都是非常适合用自动化来进行测试的。我想这个观点，即使那些反对自动化测试的人也不应该否认吧？至少我觉得他们应该有这个意识。因此，对于他们反对自动化就集中在了 UI 自动化方面，我这里也完全站在 UI 自动化测试的角度来写这篇文章，后边就不再强调了。

再次套用朋友的一句话，＂自动化测试听起来很神秘，学起来很简单，用起来很麻烦＂。我想有过自动化测试经验的人，可能大多都有这个体会吧？前边的过程我就不提了，以后主要探讨为什么用起来会麻烦和怎样简单化自动化测试。总而言之，想搞好自动化测试，还是需要测试人员比较高的技术水平，尤其是编程能力和解决问题，分析问题的能力。

首先，我要谈谈自动化测试工具和编程语言的关系。作为一个优秀的自动化测试人员，他的最基本的能力就是编程水平了。所谓编程就是至少要精通一门高级语言，比如Ｊａｖａ，Ｃ＃等等，脚本语言不计算在内。请记住，不是熟悉，是精通。高级编程语言给我们提供很强的能力来实现一些东西。你所精通的语言能力越强，你在自动化测试可以做的事情就越多，越好。简单来讲，论程序的能力来排序是这样的，测试工具－〉脚本语言－〉高级语言（Ｊａｖａ，Ｃ＃）－〉Ｃ／Ｃ＋＋－〉Ｃ＋＋／ＣＬＩ。根据这个语言能力的排序，结合你自己的语言能力，你可以想想你到底具备多少自动化测试能力。我所强调的是，如果你想优秀，你至少要精通一门高级语言，这是必不可少的。如果很多人还只是用测试工具，脚本语言工作而抱怨自动化，我要强烈的建议他们好好去学习一下编程能力先了。他们的问题在于，由于编程能力的不足，使得自动化测试的很多问题没有能力和办法去解决。再谈一下自动化测试工具，无论哪种测试工具，无论他们设计的多么强大，从编程语言来讲，他们最多能够达到脚本语言的能力。也就是说，如果你完全用测试工具来进行自动化的开发，很多问题你还是无法解决的。因此，我推荐的自动化开发方法是高级语言结合测试工具。我的自动化测试逻辑是，用测试工具只是完成ＵＩ操作，其他部分完全用高级语言来实现。我们不能否认高级语言所具有的能力，他们创造出了世界上这么多丰富多彩，这么多优秀的软件，难道开发测试程序会有问题吗？因此，我们的焦点就落在了测试工具的ＵＩ操作部分。

第二，关于测试工具。开发语言重要，选择一个合适的测试工具也同样的重要。一个灵活，强大的测试工具可以使你的自动化开发起到事半功倍的作用。结合不同的项目，不同的语言，你可能会有不同的选择。不过，这里我想解释的是，具有了高级语言的开发能力之后，我们期望测试工具来为我们做什么。我前边也说过了，我们所要求自动化测试工具所做的就是ＵＩ的操作。这里边比较重要的是三个方面，一是找到ＵＩ对象，二是操作ＵＩ对象，三是同步。如果一个工具能够让你找到所有的ＵＩ对象，并且能成功操作这些对象，就完全满足我们的自动化开发需要了。如果，工具能够提供同步的功能，就使你能够如虎添翼，不然的话要自己去实现，会麻烦不少。到了这里，你已经具有了所有ＵＩ的操作能力（测试工具提供），并且具有了高级语言的实现能力（高级语言提供），你才有了基本的能力去做一个优秀的自动化开发。没有这些能力的人，我严重怀疑能否做出好的自动化测试。

第三，怎样自动化。我的自动化的原则是，尽量少的进行ＵＩ的操作，除非是你本身要测试的ＵＩ。道理很简单，ＵＩ操作由于可能受各种问题的干扰，很容易失败。通过非ＵＩ的方法去实现是更加可靠和快速的。这也是我为什么要强调对于高级语言的精通，具有高级语言的开发能力，你就能过把大量的任务从ＵＩ操作转向了程序操作，使得你的自动化程序的可靠性大大的增强。这里还需要强调的一点能力就是系统应用的能力，比如Ｗｉｎｄｏｗｓ使用的能力。Ｗｉｎｄｏｗｓ的很多的操作是有相关的命令来实现的，不一定非得通过大家熟悉的ＵＩ。记住这个原则：除非是你要测试的ＵＩ，否则尽可能的通过高级语言来实现。我想大家对于高级语言来实现的工作应该还是有信心吧？因此，下边我要谈的内容就完全的与你要测试的界面相关了。

第四，怎样进行ＵＩ测试。首先要尽量的减少ＵＩ操作，除非是你必须要测试的操作。比如简洁快速的启动你要测试的界面，用快捷键代替鼠标操作等等。总而言之，理想状态下我们进行的每一次ＵＩ操作，都是我们需要测试的，其他操作尽量避免，不能避免用最可靠的方式去实现。那么我们现在的焦点就变成了，怎样来处理我们真正要测试的ＵＩ了。ＵＩ测试的开发基本上就三个问题：发现对象，操作对象和同步。简单解释一下同步，同步就是有一个机制告诉你何时可以执行一个ＵＩ操作。很多人是用ｓｌｅｅｐ的方式，等待一定的时间去执行下一个操作，这是我非常反对的。我的原则是，尽量少用ｓｌｅｅｐ，就算要用每次最多不要超过一秒。滥用ｓｌｅｅｐ会严重影响测试程序的性能（具体的ＵＩ自动化过程，大家可以参考我的其他文章）。

第五，ＵＩ测试错误／异常的解决和Ｄｅｂｕｇ。通过以上的解释，我们只是在自己需要测试的ＵＩ操作才进行ＵＩ操作，否则通过高级语言或者系统命令来实现。是不是我们的ＵＩ自动化就完美了呢？绝对不是，这只是一个基础，还远远没有达到完美。我们在自动化开发和应用的过程中，大部分的时间其实是花费在了异常／错误处理和Ｄｅｂｕｇ上面。这跟真正的程序开发非常的类似，你如果去看代码的话，大量

的是在进行返回值得检验和异常的处理。如果我们的程序在运行过程中出了问题怎么办，或者如果没有出现我们期望的结果怎么办？一般来说有三种问题，第一是产品的问题，我们可以报ｂｕｇ了，第二是你测试程序的ｂｕｇ，你需要ｆｉｘ。第三是其他的问题，比如测试工具，甚至高级语言本身的问题，你需要ｗｏｒｋａｒｏｕｎｄ。总而言之，优秀的测试程序最终的目的是，一旦程序的运行发现了问题，就是产品的问题，就是可以报ｂｕｇ的。能够达到这种境界才能算自动化测试的完美，才能算是一个真正优秀的测试人员。（当然了，正如软件产品不可能没有ｂｕｇ，你的测试程序也不可能完全没有ｂｕｇ。但是，由于软件产品是有大量的用户来使用，而你的测试程序只是很小范围内来使用，使得你消除影响测试过程的ｂｕｇ成为完全可能）

综上所述，一个优秀的自动化测试工程师必须要具备高级语言的开发能力，自动化工具的灵活应用能力，系统命令和使用的熟练能力等这些基本功，还更要具备优秀的Ｄｅｂｕｇ，Ｆｉｘｂｕｇ的能力，和保持程序稳定性能力。换句话讲，一个优秀的自动化测试工程师必定也是一个优秀的软件开发工程师。

最后谈一下我为什么要转向Ｃ＋＋／ＣＬＩ？从上边的排序大家可以看到，Ｃ＋＋／ＣＬＩ是目前Ｗｉｎｄｏｗｓ平台最强大的编程语言。在我的自动化开发的过程中，我需要高级语言和系统命令都不能完成的功能。如果没有Ｃ＋＋／ＣＬＩ我就必须要通过ＵＩ来实现，从而降低我程序的可靠性。而有了Ｃ＋＋／ＣＬＩ的功能，我就可以绕过ＵＩ操作了。总之，能够绕过ＵＩ操作的能力也体现出一个自动化测试人员的能力。从这个角度讲，测试人员有很多东西要学的。最后说一下，我自动化工作的要求是１００％可靠，我还不能完全满足，因为使用我程序的人是那些手工测试的人，他们的使用环境的变化有可能引起一些问题的产生，基本上还不是我程序的问题，而是测试工具，或者其他模块的问题，我需要想办法去ｗｏｒｋａｒｏｕｎｄ。不过，随着一定时间问题的积累和解决，如果环境不变，应该可以达到１００％可靠。（可是环境的变化是不会停止的，因此实际上很难达到永久的可靠，不过一段时间的可靠还是应该可以达到的，或者说我们的测试开发必须有这样一个目标，就如同软件开发的目标一样）

一个ＵＩ自动化的小例子

随便用一个小例子来解释一下ＵＩ自动化的开发吧.

我先现在有一个Ｂｕｔｔｏｎ是ｄｉｓａｂｌｅ的状态，一旦Ｂｕｔｔｏｎ　ｅｎａｂｌｅ，我们就Ｃｌｉｃｋ弹出一个窗口.

我们使用的测试工具就有同步的功能.

1．自动化工具生成的程序（发现和操作控件，不能真正运行）

```
button=FindButton();

ClickButton(button);
```

2．傻瓜的自动化程序（通过加入ｓｌｅｅｐ变成可以运行的程序）

```
button=FindButton();

Sleep(10);

ClickButton(button);

Sleep(10);

window=FindWindow();
```

3．简单的自动化程序（加入同步，使得更可靠和有效率）

```
button=FindButton();

WaitButtonEnable(button);

ClickButton(button);

window=WaitWindowOpen();
```

4．完整的自动化程序（保证１００％可靠,没有测试程序ｂｕｇ,简单写了一下，没有包含 exception 的控制，时间急，可能也会有错误，不过就是这个意思）

```
Button button=null;
for(int i=0;button==null&&i<3;i++) //如果 FindButton 不稳定，调用三次 in case
{
   button=FindButton();
   if(button==null)
   {
      Log.Error("Tryout{0}:Can not find button",i); //测试工具不稳定
   }
   else
   {
      break;
   }
}

if(button==null)
{
   Log.Error("Cannot find button. Quit"); //测试工具找不到 button，或者产品问题
   Log.Screen();//截图,只是为了示例,以后不再单独写
   return;
}

if(！WaitButtonEnable(button))
{
```

```
    if(button.Enabled==true) //测试工具问题，没有得到 enable 的消息
    {
        Log.Error("enabled, but tool didn't detect");
    }
    else//测试工具问题，不能成功检测 button 的状态，或者产品问题没有 enable
    {
        Log.Error("don't enable");                    return;


    }
}

Window window=null;

for(i=0;window==null&i<3;i++)//ClickButton 不稳定，或者没有得到 open event，或者产品问题
{
    ClickButton(button);
    window=WaitWindowOpen();
    if(window==null)//没有 click 或者没有得到消息，或者产品问题
    {
        int count=0;
findwindow://FindWindow 不稳定，重试 3 次
        window=FindWindow();
        if(window!=null) //没有得到消息,但是窗口弹出
        {
            Log.Error("didn't get event");
            break;
        }
        else //没有 click，或者产品问题, 或者 FindWindow 不稳定
        {
            Log.Error("Tryout{0}:didn't get window",i);
            count++;
            if(count>3)
            {
            }
            else //FindWindow 不稳定，workaround
            {
```

```
                Log.Error("goto{0}",count);
                goto findwindow;
            }
        }
    }
    else //成功
    {
        break；
    }
}

if(window==null)
{
    Log.Error("didn't get window, maybe tool or product problem.");
    return;
}
```

用一个小例子来说明手工测试，自动化测试，系统命令，编程语言，API 的关系

很多人理解的自动化就是把手工测试 case 用脚本和工具转变成自动化测试。也就是说把手工测试的每一个步骤用脚本来模拟，从而执行 test case。那么自动化的所有问题就归结于，如何用工具和脚本来转化手工操作步骤了。还有很多非常 senior 的，但是不会 coding 的手工测试工程师强调 case 的 design 能力是如何如何重要，自动化相对来说不是那么重要。我这里可以肯定的说，没有好的编程功底，你也不可能涉及出非常好的 test case, 自动化的开发也不应该是仅仅把手工操作用脚本来模拟，而是应该大幅度的改变 test case，使得能够用最好的方式来进行自动化。那些手工测试人员所谓的设计 case 的重要性，和他们设计 case 的高水平，实际上只是在他们的知识范围之内产生的观点。下边我用一个小例子来说明，编程能力在自动化过程中起的作用到底有多大。基本上来讲，有多强的开发水平，就有多强的自动化设计，实现水平。自动化开发和产品的开发实际上都是一样的，都是有需求，你来实现。当然，不同水平的人，实现起来的效果是千差万别的。这也就是为什么开发有高手，有低手，自动化测试的开发也同样有低手，有高手。自动化测试水平没有上限，你要学会发挥自己的无穷潜力。

不多说了，现在说一下我们要自动化什么问题。我们有两个计算机帐号， A 和 B。我们需要用 B 帐号进行系统的设置，也就是测试的准备工作，然后用 A 帐号来进行测试。下边来说一下不同水平的人是如何进行自动化的。

1. 手工测试人员

- Log on B
- Configure
- Log out
- Log on A
- Test

2. 初级自动化人员（直接把手工 case 转成自动化）

- Set autologon B
- Set autorun
- Record test status: 0
- Logout
- Check status

```
if(status==0)
{
```

```
    Configure
    Set autologon A
    Record test status:1
    Logout
}

if(status==1)
{
    Test
}
```

这个级别的人，需要懂得脚本编程，需要懂得系统设置， autologon and autorun。

3. 有一定经验的自动化人员（改变手工测试 case 以利于自动化的更简单，可靠的实现）

- 不需要 log out and log on
- 利用 Windows 命令 Runas
- 用高级语言调用 Runas
- 利用重定向来输入 Password

这个级别的人，需要懂得高级语言，重定向， Windows 系统命令 Runas

4. 中级自动化人员（具有更丰富的开发经验，可以用程序代替 UI 和系统命令）

- 不需要 Runas 命令
- 利用.NET 的 Process 对象
- 用 B 的身份生成一个 Process 来进行配置工作

这个级别的人，要比较熟悉高级语言，比较熟悉高级语言的类库，懂得操作系统的内核基本概 念

5. 高级自动化人员（精通高级语言，精通操作系统内核）

- 不需要多生成一个进程
- 用本线程 impersonate 用户 B
- 利用.NET WindowsIdentity 对象
- 必须要调用 Windows API，LogonUser

这个级别的人，要精通 C/C++和 Java，C#等高级语言，精通 Windows 内核的知识和 Windows API

从以上的例子可以看到，针对同一个 test case，不同的测试人员，从手工到高级自动化，由于自己知识面的原因，会设计出非常不同的 case 出来。越高级的自动化越灵活，稳定，可靠，也更需要掌握更多的开发和内核的知识。因此，我们看到很多人在强烈的否定自动化，你先看看他到底在哪个层次中。越下边层次的自动化人员，由于技术的原因，碰到的问题会越多，能解决的问题却越少，因此对自动化的抱怨也就越大了。这些都是可以理解的，不过以此来否定自动化，我觉得还是不太应该，毕竟自己技术还不过关 。

从微软 Windows 产品线的测试看自动化测试的重要性

最近父母来美国访问，大部分时间都陪他们到处游玩，因此也没有写任何新的文章出来，今天还是老声常谈。从写测试文章以来就陷于自动化测试话题的漩涡中，虽然我觉得这个话题对我来说已经没有任何讨论的意义，可是还是怕很多反对自动化的文章对大家产生误导作用。以前的文章我主要是从技术的角度来讨论，而且从这个角度我认为我已经基本表达了所有我对自动化测试的理解，那么今天我就从业内实际的产品测试的范例来谈自动化测试的重要性。

众所周知，微软的 Windows Vista 投入了 6000 工程师，花费了 5 年的时间来进行开发，不夸张的说，这是人类历史上规模最大的一次软件产品的开发。因此，对于 Windows 的测试流程是非常值得拿来进行研究的。今天我也想从我自己了解的一些情况来简单的谈一谈。

首先，微软没有专门的手工测试人员，基本全部是自动化测试人员。微软的一个测试人员的测试用例大概是其他公司 3，4 个人的数量。我们知道测试量不能仅仅用测试用例的数量来衡量，还有一个测试周期的问题。而 Windows 的测试周期是一天，甚至有的时候不到一天。其他公司的测试周期一般至少要一个星期。那么微软一个测试人员一天的时间要完成其他公司 3，4 个人一个星期的测试任务，听起来是不现实的。那么他们是怎样完成的呢？下面我从纵向和横向来分析。

纵向来看，由于 Windows 开发团队的规模宏大，使得他们的组织一定要分层，分级的。最高层的 build 就是在市面上发布的 build，而每一个部门都会有自己单独的 build。平时的开发和 bug fix 都是在自己的 build 中来进行的，经过测试人员的 sign off 才能传送到上一级的 build。而一般来说，都是分 3 或 4 级的。也就是说，每一级的 build 你都需要进行测试，而每一级一般每天都会出一个新的 build。那么最多每天你需要测试 4 个 build，大家想想，如果不采用自动化，只是纯手工的方式，如何能够完成呢？而实际上，Windows 聘用了大量的外包人员，他们专门安装每天出的新 build，并且运行测试人员的自动化测试程序，然后把测试报告发布出去。而测试人员得到报告以后，要分析任何的测试错误，进行相应的处理。一般来说，错误分三种，测试环境的问题，测试程序的问题，产品的问题。测试环境的问题要外包人员来负责，测试程序的问题要报测试的 bug，自己解决。产品的问题，当然就要报产品的 bug 由开发人员来解决了。这种模式就使得测试人员能够把测试的开发和运行分离开来，使得测试人员能够专注于测试的开发工作，也就是说测试人员大部分的时候是在 coding 中。这也是为什么微软的测试人员 title 是 SDET(software development engineer in test), 也就是说微软的测试人员本质上就是开发人员。

横向来看，Windows 的测试根据时间的长短和重要性分成很多种。以上所说的是测试每天的 build，在自己部门的 build 往上一级传送之前的测试又是一种，还有一种是要测试一个 build 使得质量可以使得微软的员工进行内部的使用，还有就是每个 milestone 的测试，以及 release 之前的 beta, rc 阶段的各种测试。因此，测试用例的优先级在这里就显得十分的重要。由于各种原因，往往不能 100%的进行测试用例的自动化，因此手工测试还是必不可少的。如果我们把测试用例按照优先级分为 P0-P3。那么一般来说，P0 和 P1 的 test case 是一定要实现自动化的。那么对于 P2 和 P3 的手工测试，就只在非常重要的测试阶段来进行。在每天的测试，和不太重要的测试阶段，也没有要求保证全部的测试用例要通过。这样的安排，就使得

SDET 能够尽量少的去执行手工测试，从而能够专注自动化测试的开发和对产品进行更加深入的测试，以及安全测试等等。一般来讲，需要进行手工测试的情况的发生周期要大于一个月，而因为 SDET 平时尽量的能自动化更多的测试（包括 P2 和 P3），手工测试往往需要 2 天的时间就能完成。

从以上纵向和横向的分析来看，Windows 的测试任务是十分浩大的，纯手工测试也是无法想像的。可是，由于微软大量的采用自动化测试技术和人才，使得不可能变为可能，实在是做了一个自动化测试软件工程的典范，值得任何公司和个人去思考。

面试

我的 Oracle 面试经历

一直都在注意 Oracle 的招聘信息，毕竟是世界第二大软件公司呀。可是，奇怪的是观察了很长的时间都没有发现任何招聘广告。并且，从我对它的初步了解来看，与微软，Google 中国研发中心比起来，它显得那样的无声无息。终于有一天的早上，打开 zhaopin.com，第一次发现了 Oracle 的招聘信息。高兴的是，里边包括若干个测试的职位。马上把简历投了过去。下午接到了电话，一听果然就是 Oracle 的，要跟我安排面试。Oracle 的招聘比较奇怪，没有 recruiter,都是 hiring manager 跟你联系。

面试当天，早了几分钟来到公司，被前台领到一间会议室里面，给了一套试题来做。同屋已经有好几个人在做题了，心理顿感不爽。本来以为是要单独给我面试的，没想到竟然会跟大家一起做题。更没想到的是，当我打开试题的时候，我好像基本都不会做。实话实说，我并没有什么 Oracle 的技术背景，可是我觉得做测试也不需要懂这么多试题上的知识吧？硬着头皮做了两道，实在做不下去了，起身就走。到了前台，告诉他们我感觉这个面试不对劲。他们才反应过来给搞错了。忙着给我的 hiring manager 打电话，让我去了她那里。

Hiring manager 是个中年女士，看起来人就很 nice。面试也是在一间会议室进行的，做了一屋子的人。看来对一个测试人员的面试也搞得挺隆重。Hiring manager 并没有提问，第一个提问的是个技术大拿。他面试的场面我还从来没经历过。简单的说就是，他看着你简历问你问题，问得肯定是你做过的东西，但是，问得深度是特别的深，一般你还回答不上来。这种知识面，深度的人，到任何地方都是大拿。我是挺佩服的。第二个面试的可能是一个香港人，他主要是考察我的英文水平，用英文跟我聊了很长时间。第三个人主要是考察我的 Linux 的技术，很抱歉好久没有用 Linux 了，问我的很多简单的命令，本来我以前是会的，可是忘记了。不过，这个时候那个 manager 讲话了，说我的背景学 Linux 应该是很快的事情。然后我主要跟他们聊了自动化测试的东西，他们竟然还没有什么自动化的测试，基本还是靠手工测试。他们也希望我能够在自动化测试方面能够给公司带来些新鲜的东西。

第二天，manager 给我打电话，说美国的老板想跟我聊聊。与此同时，我也接受到美国的一封 email,要跟我电话面试，搞得我有点迷惑。最后才知道，是两个不同的老板要跟我面试。定了周末两天的上午，因为对方是在加州，我们合适的时间只有每天的上午。第一天面试是一个中国人，不过我们全是用英文交谈的。看得出他对我很满意，多次给我说这个职位不是在他的 team，如果对方的不要我，我可以来他的 team。他是做日本方面的项目，另一个 team 是做 open source 的项目。一直谈的都很愉快，直到谈到待遇的问题。说实话，我当时的目标是每月 2 万，不过由于微软，Google 的情况还不明朗，不想再错过这次机会，因此就要了每月 1.5 万。没想到他却明确回答不能满足，并跟我说他们有很多福利等等。最后问我要多少钱，我就回答算上福利全年 20 万。实际上当时有点傻，因为月薪 1.5 万和加上福利年薪 20 万应该是差不多的。对方就没有再说什么，我们结束了谈话。第二天是个老外，主要谈一些技术的问题吧，我也没感觉是好是坏，因为昨天的人已经说过不行就去他的 team 了。

不过，后来他们就再也没跟我联系过。我估计是我的要价太高了，而且我最近知道我一个朋友有多年的 Oracle 经验，刚刚进入他们那里，也不过是 20 万年薪。这样解释了我的一个疑惑。我当时就奇怪，堂堂一个 Oracle 怎么把研发中心放到上地。看来他们本身就不想投入太多。后来又听人说，这个中心的老板不地道，总之有一些 rumor,不知道是真是假。

我想当时如果我真想进的话，开口小点应该是没什么问题的。这里给大家介绍一个面试的经验，对方的技术你不熟悉可能不是一个大问题。面试的时候，一定要有自信，让对方知道你有能力，有激情，有头脑。并且，一定要大胆跟他们讨论一些你擅长的技术问题。让他们更细致的了解你的能力，留下与其他人不一样的印象，就更容易打动他们。还有一个经验就是，谈待遇的时候，如果你真的想进这个公司，你就不要说具体数目了，就说我很珍惜这个工作进会，我一直都期望能进入你们的公司，你们可以按照标准给我工资就可以了，我没什么特殊的要求。我后来的面试在待遇上都是如此回答的，并且都得到了 offer。

下次有时间，讲讲我去微软，北电面试的经验教训。

我的 google 面试经历

以前投过 google，都会收到没有合适位置的礼貌性回信。后来自己的经验背景增强以后，又重新投过。不久收到回信，又是没有合适的位置。有些失望。不过 10 几分钟以后，又收到一封来信要跟我商量电话面试的时间。Google 的招聘过程相比其他大公司来说显得有些混乱，这也是经历过 Google 面试人的共同感受，在我电话面试的时候也专门的跟面试官提起过，他也承认他们的面试流程存在问题，正在改善。这可能也是 Google 在全力扩张时期的一个必然要经历的阶段吧。

Google 的测试人员分为两种 SQAE 和 SET，SQAE90%都是手工测试，因此面试只需要懂得基本的编程技术即可。SET90%都是自动化测试，因此对编程，对算法的要求都相当的高，可能是业界对测试人员要求最高的了。

电话面试大概是去年 2 月底吧，也可能是 3 月初。电话是从 Mountain View 总部打过来的，因为时差的原因，面试只能在早上进行。电话面试全英文的，问了很多技术的问题，也问了我自己的工作经验。我说我的产品专门让你们的 Google desktop 不起作用，他也很感兴趣。没想到最后竟然用中文跟我对话，很 surprised,以为是个老外，原来是个台湾人。最后跟我用中文说了一句，“我觉得你很好，真的”。心里很高兴。也问了一下是考虑我总部的职位还是北京的职位，他说是北京的，我说 OK 了。

不久安排 Onsite 面试，问需不需要定机票，可以订世界任何城市到北京的机票。看来北京的招聘是下了大成本的，不过我不需要，有点遗憾了。不幸的是，面试的前一天早上开始发高烧，都不能坚持上班了，面试的时候依然在高烧状态。HR 是个小女孩，只懂得一点中文，英文速度非常快，我听着也比较困难，幸好话不多。面试一共四关，全部是中文面试。都是总部派来的台湾或者大陆会讲国语的工程师。第一关是一个 SQAE,应该不怎么懂编程。面试了一些可能是她工作中常用到的技术，不难，不过我没怎么用过，我搞得东西她也不懂，感觉考察不出自己的水平。好像也问了一道算法题，我没有答出来。第二关是一个 SET, 因为职位相近，因此这关感觉最好。谈得很投机，问了两道算法题，竟然有一道就是刚才没做出来的。这次集中精力，做出来了，不过回答的有些糊涂，也确实脑子高烧的糊涂。她也问我怎么这么糊涂，我说我在发高烧，这道题刚才那个人就给我出了，我刚才没答出来。她责怪我怎么不早告诉她，还说要告诉刚才那个人给我把 review 写好点。后边两关是两个 developer, 难题全部出在了算法。这是我第一次面试算法题，以前从来没经历过算法面试。总之答得不好，脑子烧得糊涂，又没有准备。第三关的人感觉态度挺冷淡的，也可能看我面试效果不好，我也没跟他说我在发烧。第四关的人感觉挺 nice 的，我不会，他也告诉我答案了。确实不容易，都不是直接能出来的，需要数学的推理，根据推理后的结果，才能生成程序。这个东西必须事先准备，否则基本是不可能做出来。

感觉面试的效果很不好，离开的时候，送了一件 T-shirt, 一支笔，一本李开复的书。我跟 HR 说，如果 SET 不行的话，能不能考虑我做 SQAE。她说会把我的意思告诉 committee 的，一个月之后出结果。

之前面试了不少公司了，都是因为待遇的问题没有谈妥。也挺崇拜 Google, 也很珍惜这次机会，很不理解为什么会突然发烧，这么影响自己的面试表现。

等了一个多月了，根本没有跟我联系。发信去问，几天后那个组织面试的 HR 给我打电话过来，问我为什么想跳槽，还问我给我 Local 的待遇能不能接受，我说能。后来就再也没有音讯了。

国内求职不顺，只能考虑国外了。总之 6 月在国外工作的时候，Google 的总部有人跟我联系，说对我感兴趣，并且是总部的职位，SET。这个时候已经拿到另外大公司的 offer，只是还没有最终确定，还没有 100%。因此又参加了他们的电话面试。先是 HR 的人，问得到都是技术上的问题。只是很死板，好像他们根本不懂，只是拿着试题和答案发问。记得我回答的也不好。没两天安排技术面试，到现在我也不明白是那个 HR 的面试我通过了，还是没通过，这次面试只是另外一个部门呢？总之都不是同一个人组织的。

面试的可能是一个老印，英文不是很容易听。用 Google 的 Online 那个字处理软件。我听不懂就让他写在上边。还是那些东西，不过这次有经验了，算法题基本都做对了。总之，我感觉是挺好的。没想到组织中国面试的那个 HR 又发邮件跟我联系了，问我还做不做北京的 SQAE 的职位。我说不做了，可以考虑总部的 SET 的职位。她说问问领导，后来跟我说可以，不过还要重新面试，而且是 SQAE。我告诉她我正在跟 Google 的其他部门进行 SET 的面试，另外也告诉她了，我已经得到其他公司的 SET 职位。因此我们就不再继续下去了。

后来另外一个部门的 HR 给我写信说我的电话面试不错，希望能再安排一次面试。没有回，准备去新公司上班了。已经上班以后，又收到她的邮件说希望再跟我安排一次面试，这次就明确说明了我已经开始新工作了，不再考虑跳槽了。以下是她给我的最后一封邮件，我和 Google 的缘分也结束了。

Hi XXXX，

Congratulations on your offer with XXXX and thanks for following up with me. I greatly appreciate it! If things should change with your situation, please let me know. The door is open to continue at anytime with your application at Google.

Cheers to you!!

XXXXX

记一次加拿大面试

早就知道加拿大的滑铁卢城市有美国一个著名安全公司的研发中心，也曾经梦想过有朝一日能在那里工作。因此，一直关注他们是否在招聘测试人员。

机会终于来了，一天突然在招聘网站上看到他们招聘两个测试工程师，马上投了简历上去。

第二天，接到了 recruiter 的 email，问我是否能去加拿大参加面试（我当时在北京工作）。因为恰巧要拜访加拿大的一朋友，朋友家在滑铁卢旁边的一个城市，就答应了。看来他们挺着急，给我约到了我到达加拿大的第二天上午 10：00。

终于等到这一天了，头天大概晚上 12 点才到达加拿大，赶紧休息，为了有精神第二天面试。第二天早上驱车赶赴这家公司。

公司在著名的滑铁卢大学的北面，是独立的一个大房子，只有一层，大概几百名员工。环境来说，在加拿大还是很不错的了。没想到 recruiter 竟然是个男士，让我在会议室等待。

不一会儿，来了两人，一男一女，年纪都不算小了。后来才知，男的是 team lead，女的是 test manager。

面试大概是半个小时，其中前 15 分钟是他们问我问题，后 15 分钟是我问他们问题。没有问到具体的技术细节问题。他们主要是了解我做过什么项目，问了我对安全领域的看法，还有就是他们对这个职位的技术需求，并且告诉我不是 team lead 的置位，我是否愿意。我主要是问了他们自动化测试的情况，能不能看开发代码，有没有白盒测试等等。面试之后并没有什么结果，但是自我感觉还好。

然后驱车回去，刚到家门，朋友就说有人给我打电话了。一看号码，正是这家公司，赶紧打了回去。recruter 上来就跟我说，准备给我 offer，想讨论一下工资的问题，问我期望多少。说实话，我对工资并没有特别高的期望，我更重视有海外的工作机会，因此回答你们看我的水平来给就可以了。他们回答是 X 万加币，我回答 it's reasonable,实际上已经超出了加拿大平均 IT 的工资不少了，我也很满足了。

后来的 offer 竟然又给我在这个基础上加了 2000 加币，感觉心里很舒畅。不像国内的企业，看你满足了，可能还要给你减点。大公司还是很大气，你满足了还多给你点，公司没多付出多少，可是员工的感激之情就不一样了。

记一次美国面试

去年想跳槽的时候，在北京面试了 Microsoft，Google，Oracle 等等之后，并没有接到什么太明朗的 Offer，因此开始把目光转向了海外。

一天早上突然接到了一间美国很大规模软件公司的一封 Email，问我美国总部有一个职位，愿不愿意来这里工作。当时有些吃惊，因为这个机会是自己还没有幻想过的，回答了“YES”。

很快，收到了他们的一套技术问题，有些简单，一些复杂，有些问得自己一头雾水。问了一些 developer，他们也不太清楚。无奈，按照自己的理解回答，并 reply。没想到的是，第二天他们竟然邀请我去美国面试，并且开始准备我的行程。

等了两天又没有音讯了，接到邮件问我，他们不但考虑我做 engineer,还考虑我做 team lead,问我愿不愿意做 team lead。回答“YES”。

又没有音讯了，再次接到邮件让我重新 submit resume,说他们联系两个人名字相同，要验证一下到底是哪个？狂晕，自己认为肯定是那人了。

最后的结果还真的是我，然后准备行程，正好五一将近，定得五一的时间。

大公司果然是不一样，机票，旅馆，租车，都提前搞定。自己只需要把邮件打印出来带在身上就够了。每天的伙食费是 75 美金，可以报销。

为了准备面试，提前两天到达美国。在飞机将要到达目的地的时候，看到了一座异常雄伟的雪山，据说是座活火山，酷似富士山。下了飞机，在机场找到了租车的地方，稍得一会儿，一个黑人给我把车开来，并且开始跟我聊天，问我是哪里人，来做什么。听到我说是中国人，他很兴奋的说你的 head 前不久来过这里。我有些误会，以为说是这个公司的 head，还想问他怎么知道我是面试哪个 head。然后才明白原来指的是我们的胡主席。看来主席的来访都惊动了美国的底层民众。不过他感兴趣的是，主席来这里买了很多东西，花了很多钱，促进了美国的经济。

从机场出来，一路都是高速。事先已经查好地图，因此没有迷路，直接开到 Hotel。在 Hotel 的所有花销都不需要自己付账，签个字，公司会来买单，因此给小费的时候也异常的大方。公司离 Hotel 很近，安顿之后就开车去公司踩点了一番。然后回去上网，吃晚饭，去健身室等等。

第二天又去公司参观，感觉很大，很漂亮。心里想呀，这里真是 IT 人员的天堂呀，能在这种环境工作还有何求？下午去附近一个很著名的美国大学校园玩，路上远远的又看到那座活火山，心情很舒畅。

第三天的面试很辛苦，从早上九点一直到晚上六点半。除了 recruiter,面试了 6 个人。面试的内容是要保密的，六个人有两个 manager,三个 team lead，一个 senior engineer。问题有些简单，有些就很复杂，不过运气还好，绝大部分都回答出来了。听说面试的时间越长，见得人越多就越有希望。因此，见了 6 个人，回去之后心情还是比较轻松的。

睡了一个好觉，第四天一早飞到机场，还车，又飞了回来。

这一年的五一假期过得很有意义。

简历（初中级时期）

Technical Background and Skills:
@Software Development
-3 years industrial software development experience
-Analysis and design of Object-Oriented
-Mastering C, C++ and C#
-Familiar with MFC, .NET framework
-Network programming: TCP/IP, Socket, C/S, B/S, P2P, Distributed
-Development experience in Java, VB, Perl, PHP, Javascript, PowerBuilder, SQL etc.
-Solid Knowledge of ASP.net, Web service, SOAP, XML
-Familiar with Linux

@Software Testing
-2 years software testing and management experience
-Ability to design automation testing architecture
-Experience with Winrunner, Rational Robot, Silktest, Testcomplete etc.

Professional Experience:
Sep 2004-present ***, Beijing/Fremont
Position: QA Team lead

Project: ***
.In charge of ***Team(4 people)
.Making test plan, assigning tasks to team members, reporting test status to director.
.Engine testing, worm testing, application compatibility, template testing.

Project: ***
.Working at Beijing, China/Fremont, US
.In charge of the functional testing for the whole product(4 people)
.Recruting testers, building up QA team, communicating and coordinating with US colleagues.
.Making testing plan, writing test cases, assigning test tasks, process tracing, reporting.
.Design and implementation of automation testing framework.
.Product localization(Chinese).

Project: ***
Language and tools: C#, ASP.NET, Jscript, Test Complete, VMware Workstation

.Architect of Automation framework
.Organizing automation team. (6 people)
.Designed the four layers of the system: Application layer, Service layer, Module layer and Environment layer.
.Implemented Service layer and Module layer independently and insgtructed the implementation of Application layer.
.Found an alternative way wo work around VMware Virtual Center, which made 90% cost reduction.

May 1999 - Mar. 2001 ***
Position: Software Developer
Project: ***
Language: C
Platform: PSOS
My role: Developing and maintaining Radius protocol module.

Feb. 1998 - May. 1999 ***
Position: Software Developer
Project: ***
Language: VC5.0
Platform: NT4.0
My role: Designing and developing***module.

Education:
2001 - 2004 ***
Major: Computer Science
Degree: MSC.
Research: Grid Computing, Neural Network, Distributed, OO Modeling, Networking, Parallel
Thesis: ***
Language: C, C++, Java, Perl
Platform: Linux and Windows
Positions:
RA (Grid Computing, Neural Network, SuperComputer)
TA (Operating System, Software Engineering)
Awards:***

1994 - 1998 ***
Major: Computer and Its Application
Degree: BSC.

Interests:
Video games, Soccer