构建之法阅读笔记三

2016-06-07 18:32

上周老师讲了软件测试，今天我在看构建之法时看到了这一方面，仔细看了看，把自己的感受说一下。

　　由于软件是由多人合作完成的，每个人分工合作，彼此依赖，因此如何让自己负责的模块功能更加明确就显得非常重要。这时，我们就要用到单元测试了，单元测试能很好的帮助程序员记录每个模块的信息，不仅能给自己提供方便，而且在别人使用你的模块时也能清晰的表达出来。那么，到底要怎么写单元测试呢？这里的话，其实我也从没进行过单元测试，只能结合构建之法上的内容简要的说一下，但是很快我们也要进行单元测试了。

　　怎么才算一个好的单元测试呢？能准确快速的保证最基本的模块是正确的，这才能称为一个合格的单元测试。那么单元测试要不要自己写呢？答案是肯定的，只有最熟悉代码的人来写，才能更好和更快的完成单元测试，因为你很了解你自己代码的功能，局限和最容易出错的地方。这里可能有人会问，不是有专门管测试这一方面的吗？当然在一些极限测试编程的方法中，是可以的，但是还是由自己做单元测试才是最好的。

　　还有一个问题，单元测试要测试到什么情况才结束呢，要等到程序完美无缺吗？我认为应该不是这样的，无论你做多少测试，做过多少修改，程序都不可能是完美的，只要能测试到你的程序不会影响到之后的工作就可以了，（这里的话是我自己的理解，请谨慎考虑，可能是错误的）。把单元测试的责任和代码作者绑定在一起后，有个很好的作用，这样可以让代码作者更真切地体会到复杂代码的副作用，因为验证复杂代码的正确性要困难得多。嘿嘿，这样的话，你还会故意写复杂的代码吗？我很赞同作者的这一观点，自己做的事要自己负责，这样因为测试是自己要做的，可能在写代码的时候最会更加认真仔细。

　　“100%的代码覆盖率并不等同于100%的正确性”，这是构建之法中作者说的一句话，这句话意思很清楚，但是原因我确实有点看不清楚，先说说意思吧，大概的意思就是你把你全部的代码都测试到，也不能保证测试完后的代码100%是正确的，那么原因呢，作者给了四个原因，1.代码覆盖率对于“应该写但没有写的代码”无能为力；也就是说你想写但没写，自热也没覆盖到，在这一块可能就会出现错误。2.代码中的效能问题，虽然代码执行了，并且也正确返回了，但是代码效率非常低；这句话不是特别明白什么意思，可能是代码是正确的，但是在时间和空间上花费的太多，也说明这代码是有问题的。3.多线程环境中的同步问题，这个问题和代码执行的时序，共享资源的锁定有关；这句话我就实在不明白了，也不做解释了。4.其他与外部条件相关的问题；这个很容易理解，也就是外部因素所致。

博客地址：http://www.cnblogs.com/java-meng/p/5568011.html