第 4 代白盒测试方法通俗释义

2006-9-26

第 4 代白盒测试方法论（4GWM）在网上公开有一段时间了，近来总有一些网友询问第 4 代方法区别其它方法的主要差异是什么？还有不少人提到：第 3 代相对第 2 代的界限较清晰，但第 4 代相对第 3 代的差别还不甚明了。这里，我们集中解答这些问题，补充阐述 4GWM 的内涵与外延，本文可作为《第 4 代白盒测试方法介绍（理论篇）》的补充学习材料。

一、从第 1 代白盒方法到第 4 代白盒方法

第 1 代到第 4 代白盒方法的主要差别如下表：

	是否评估测试效果	是否自动测试	是否持续测试	是否调测一体
第 1 代白盒测试方法	否	否	否	否
第 2 代白盒测试方法	是	是	否	否
第 3 代白盒测试方法	是	是	是	否
第 4 代白盒测试方法	是	是	是	是

第 1 代白盒方法属于无组织、无约束的测试方法，典型情况是拿调试当测试，或者简单的在被测代码中加入 print、assert 等语句。第 2 代白盒方法主要克服第 1 代方法的两大缺陷：一是没有测试评估（比如覆盖率），二是未对测试操作进行重用，重用测试操作必然以某种形式化语言描述测试过程，该形式化描述可有两类，一类是使用被测代码自身开发使用的语言，另一类是使用一种抽象层次更高、更为易用的脚本语言，常见的第 2 代白盒测试工具有：Rational 的 RTRT、Parasoft 的 CppTest、IPL 的 Cantata++等。

第 3 代白盒方法主要代表是 xUnit 系列测试工具，如 JUnit、NUnit、DUnit 等，第 3 代白盒方法区别第 2 代方法最主要特点是：它支持持续集成的操作模式，这在理念上有了一次飞跃。

第 4 代白盒方法继承了第 3 代白盒方法所要求的持续测试，这种继承性是包含关系，就像第 3 代方法继承第 2 代方法的形式化测试描述一样。第 4 代方法在第 3 代方法的基础上，强调了操作的重用，将调试操作重用到测试中来，至于这一步重用是否具备重要意义，是否足以升级换代，下文我们再详细叙述。这里先明确一个概念：**第 4 代白盒测试方法相对第 3 代方法，主要是引入了调测一体的理念**。遵循第 4 代白盒方法的测试工具主要以 VcTester（http://www.ezTester.com）为代表。

二、评估白盒测试方法首先看提升多少测试效率，然后看提高多少测试质量

评估一个测试工具是好是坏，可能要查看多种因素，如软件易用性、是否能提高工作效率，是否促使测试更加深入等。但对于现状下的大多数企业，这些评估因素并非对等的，如果

一个企业从未做过白盒测试，突然想把白盒测试推行起来，那首先应该确认选用工具是否足够高效，而不应该看某工具少支持一种覆盖评估（比如不支持 MCDC 覆盖评估就改选其它工具）。

实际上，尽管白盒测试很重要，该实践对于多数企业仍是鸡肋，尤其是针对 C 语言开发的项目，食之无味，弃之可惜，其瓶颈不在于白盒测试做得好不好，而在于白盒测试能不能做得下去！根据我们在本领域的多年推行经验来看，白盒测试失败的项目中 95%以上是因为测试做不下去，测试效率过低导致投入成本居高不下，实施难以为继是失败主要原因。可以说，现状下保证白盒测试能否做起来是根本问题，能否做好则是其次问题，下一步等测试做起来后再着重改善，前者更为根本，具有一票否决的性质，后者是为了做得更好，对多数企业现状下属于锦上添花，等步入正轨了才突显重要，而且解决起来也相对容易。

我们可从另一侧面验证上述观点，大家知道 JUnit 与 NUnit 在 Java 与 C#中用得很好，有许多成功案例，但 CppUnit 在 C++中成功案例就少多了，而 CUnit 能成功的寥寥无几，极少数成功的也是付出巨大投入、有良好组织保证时才获得的。不难看出，基于相同理念的测试工具，在不同语言中实施效果差距很大，最主要原因还在于工具对于测试效率提升的差异性很大，Java 与 C#开发语言本身的抽象层次较高，用它编写用例、调试用例、维护用例等，效率都很高，而过程性编程语言，如 C 语言，在 CUnit 中用 C 语言描述测试操作，必然很难成功了。

三、从第 1 代到第 4 代白盒方法，每一步都大幅提升了“可重用”能力

从第 1 代到第 2 代，对测试进行形式化描述，使测试操作以脚本（或某种语言）方式记录，支持重复测试，一次测试操作可在后续测试中重复使用，当然大幅提升了测试效率。

从第 2 代到第 3 代，持续集成使软件稳定性得到重用，怎么讲？第 3 代白盒测试方法要求以“写一点、测一点”不断迭代的形式推进项目研发，每一次迭代被测代码都相对稳定，这时，对于新冒出的 Bug 马上能被识别，马上定位，通常只需分析最近修改的数十行代码，就能推断问题所在。这操作模式下查错与改错的效率非常高，本质上来说，持续集成模式对“软件稳定性”实现了重用，直接收益处体现在两方面，其一，查错、改错的效率提高了，其二，被测系统随时处于可运行状态，待实现的功能提前展现，有利降低研发风险。

从第 3 代到第 4 代，坚持调测一体的理念后，测试代码与被测代码真正同等的看成一种产品代码，两者代码一同添加、一并维护，不是把被测代码写完整了再设计测试脚本，维护两者也是对等的，只要相关联的代码一处修改了，另一处也要跟着改。不仅如此，两者的调试过程也可融为一体，调试测试脚本与调试被测代码有许多共性，可重用的地方很多，比如调试被测代码先要构造运行环境，调试测试脚本也构造类似运行环境，调试中经常修改变量、查看变量，修改变量为了后续单步跟踪按特定路径进行，这对应于测试，修改变量或打测试桩也让特定路径得到覆盖，调试中手工查看变量是否预期，对应于测试，则是用例脚本判断测试是否通过。如果调试操作在测试中重用了，或者，更准确一点按照第 4 代白盒方法所遵循的理念，调试与测试合为一体了，无疑会大幅提升软件研发效率。

嵌入式软件的白盒测试主要针对 C 语言，正是 xUnit 系列工具难以撼动的领域，但 VcTester 之所以能突破这个障碍，是因为调试操作也得到重用了，另外，CSE 脚本语言能较好的

仿真出 C 语言行为，测试用例编写与调试的效率提高上去了。VcTester 的市场反馈充分证明了这一点，大家普通反映使用 VcTester 开发用例的效率远远高出其它工具，促进效率提升最主要有两方面因素，一是在线测试，二是调试重用，调试重用最大的贡献还不是因为调试操作能转化为脚本，更重要的是，VcTester 检视器支持“一种驱动多个用例”的测试设计机制，大大节约了编写用例的工作投入。

四、第 4 代白盒方法中的其它要点

第 4 代白盒测试方法明确规定为 3 个关键域 9 项关键特征，如下：

A. 第一关键域：在线测试
   1、 在线测试驱动
   2、 在线脚本桩
   3、 在线测试用例设计、运行，及评估改进

B. 第二关键域：灰盒调测
   4、 基于调用接口
   5、 调试即测试
   6、 集编码、调试、测试于一体

C. 第三关键域：持续测试
   7、 测试设计先行
   8、 持续保障信心
   9、 重构测试设计

只要把握住前面所提的调测一体理念，理解第 4 代白盒方法就容易多了。在线测试是一种手段，直接拉通测试小循环，也促进研发大循环被拉通，这种手段是提升工作效率的便捷途径；灰盒调测实际就是“调测一体”理念的实践方法与表现特征；持续测试则将第 3 代方法所遵循的持续集成理念，以明确方式规定下来。此外，第 4 代方法中的持续测试，还尝试克服 XP 中测试驱动开发（即 TDD 实践）遇到的困难，毕竟，第 4 代方法有在线测试、调测合一等手段，大大缓解在编码前就写用例让人无所适从的困境。

参考文献：

1. ezTester，Wayne Chan，《第 4 代白盒测试方法介绍（理论篇）》
2. ezTester，Wayne Chan，《第 4 代白盒测试方法介绍（VcTester 实践篇）》