# 如何选择嵌入式白盒测试工具

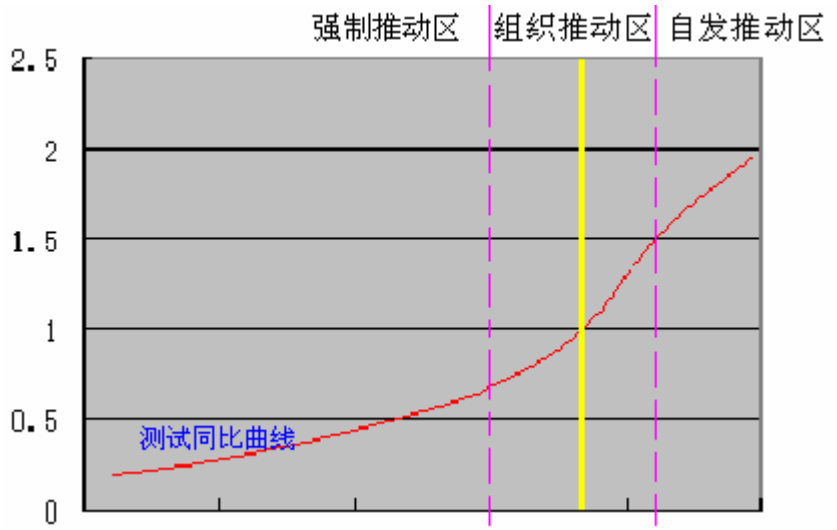
2006-9-25

恩格斯说“劳动从制造工具开始”，人和动物的本质区别是：人会制造与使用工具。IT 产品研发也从选择合适的工具开始，工具好坏对项目成败往往起着关键作用，尤其是嵌入式领域的白盒测试工具选型。尽管业界已有众多商用工具，但大部分仍处于可将白盒测试推动起来的边缘状态，选择工具稍有不慎，就导致白盒测试整体做不起来，最终严重影响推向市场的产品质量。

## 先澄清两个概念

在分析如何进行工具选型之前，我们先剖析嵌入式软件，当前状况下影响白盒测试开展的最主要障碍是什么？然后才推导嵌入式软件白盒测试工具选型应遵循的评估模型。

先澄清两个概念，其一，在嵌入式研发领域，影响白盒测试推行的最主要障碍是工具的使用效率，或者说，借助测试工具，你要花多长时间才将单元测试与集成测试做完整。在《企业如何推行白盒测试》一文中，我们介绍了白盒测试的分区推动理论，如下图：




测试同比曲线反映了测试工具的使用效率，测试效率越高，该指标取值就越高。如果测试效率偏低，测试同比小于 2/3（大致是每写 2 天代码要 3 天才能测完整）是强制推动区，这个区域对于绝大多数企业来说，白盒测试作为一项组织行为注定要失败。而测试效率够高，测试同比超过 3/2（大致是每写 3 天代码 2 天就测完整）是自发推动区，白盒测试即使没有相关流程推动，研发人员也能自觉、自发的实施起来。

所以，选择测试工具至少要求使用它的效率应保证测试同比大于 1，测试同比为 1 是个拐点，即每写一天代码只用一天就测完整，请注意，我这里讲的是“测完整”，不是简单的比划几下，而是用例总量、覆盖率等都达到一定的指标，另外强调“每写一天代码”，指的是代码

每次改动，都有白盒测试跟进，而不是一次性编码、一次性测试，如果是一次性测试，相信多数商用工具都能超越拐点，但保证整个产品周期都做到这一点，就很难了。目前适用做嵌入式白盒测试的商用工具中，大多数都没达到该要求，所以，多数情况下必需有良好的组织，有强有力的流程推动，白盒测试才做得起来。

另一个概念，嵌入式产品面对复杂的运行环境，形形色色的实时系统、编译器与设备驱动，都导致白盒测试困难重重，但白盒测试必须要到实际运行环境中去做吗？未必，也不应该这样推崇。《实施白盒测试的几个误区》一文已有详细分析，嵌入式软件应在仿真机环境实施白盒测试，“上真实环境做代码级测试”实际上是个伪命题，实践中很难行得通，或者说，行得通但代价太高，远没突破前面所提的效率拐点，所以，在各种条件受限的实时环境下做白盒测试，还不能将它上升到过程有保障的组织行为。

# 嵌入白盒工具的评估模型

评估一个测试工具的好坏，采用评估标准不同，所站的角度不同，评估结果大相径庭。所谓每个人的心中都有杆称，让测试人员选工具，他会站在测试的角度去选择，会更注重白盒测试能做得下去，之后才有兴趣深入去做，如果让质量人员去选，他会侧重于质量保障环节，比如非常看重覆盖率评估、测试报告提交等，但如果让企业老板选工具，恐怕他首先考虑的是这个工具的价格。所以，测试工具的选型过程，必然是各种因素综合考虑的权衡过程。进行公正的工具选型首要问题是：如何选择评估要素并赋予不同的权重，套用一句规范术语，我们先建模，确定评估模型，再按条目打分作决策。

建立评估模型应考虑如下几个因素：

1. 应用范围

   首先明确你期望引入某工具的应用范围，这个业务范围内都有哪几类利益相关人，然后确定评估项目，为各评估项分配权重。

   如果不明确工具适用的业务范围，或确定范围不恰当，肯定会影响评估的准确性，比如你希望某个白盒测试工具，既支持单元测试，又支持集成测试，这是一种想法，如果把它换成：想要一个能支持单元测试的工具就够了。这两种目的最终的评估结果肯定很不一样，还有，应关注适用范围的条件限定，比如，你想要一款既支持 C，又支持 C++的测试工具，或限定要支持某特定编译器（如 GCC）的。期望工具的适用范围不仅要明确，还要合理，比如：你期望一款既支持白盒测试，又支持功能测试，另外还支持性能测试的工具，最后的选型结论肯定会让你失望，没有这种万能工具。

   确定适用范围后就可以分析利益相关人了，比如你选择单元测试工具，重点是考虑编码人员的需求（注：单元测试的主体应由编码者自己承担，这是另一个话题，本文不展开），而你要的工具既支持单元测试，又支持集成测试，就不能不考虑测试人员的提议了。

2. 合理选择评估项目，分配不同的权重

   上面讲到先确定应用范围，由应用范围确定相关人后，选择评估要素就容易明确下来了，最简单的方法是：把相关人叫过来，让他们一条一条的说出他关心哪些问题，把这些问题排个序。当然，叫相关人员过来讨论并非必须，如果评估者对各个适用领域

都很熟悉，他站在各个利益相关人的角度细想一遍也行。

需要注意两点，一是不要漏掉相关人，比如想买一款性能测试工具，老板也是利益相关人，不站在他的角度考虑问题，申购单可能得不到批准，比如眼下公司的现金流吃紧，你要让老板购买一套奇贵无比的工具（尽管可能带来很大的效益），该项申购多半会流产。二是保证各个评估要素是恰当的、合理的，比如选择一款单元测试工具，许多人会要求这个工具应支持上单板做测试，根据前面我们分析的，这个评估项实际上似是而非。

3. 注意阶段时效性

这一点是分析工具的应用范围要考虑，但很容易被大家忽略。

前面我们介绍了白盒测试的分区推动特性，当前企业所处的区段不同，会有不同的要求。比如当前本企业的白盒测试尚处强制推动区，那首要考虑的问题是怎样把白盒测试做起来，如果当前已经在组织推动区了，如何保证深入测试会考虑得多些，而到自发推动区，质量评估环节应加大份量，像覆盖率评估等应赋予较高权重来评估。

本人曾见过有企业刚开始推行白盒测试，就因为某工具尚未支持 MCDC 覆盖评估，就立即将它拒之门外，选择了别一款很贵，看起来很牛 X 的工具。这种评估偏差很大，他错过一款测试效率奇高的工具了，像覆盖率，刚才始推白盒测试，有语句覆盖与分支覆盖就足够了，MCDC 覆盖对高复杂度、重要的代码才必要，白盒测试都没做起来，首先保证工具有足够的效率才是最重要的，有没有 MCDC 是锦上添花的事，等白盒测试顺畅起来日显必要。

这里提供一份来源于长期实践，适用于嵌入式白盒工具的评估模型，模型先把评估要素划为两大类：商务属性、价值属性。商务属性主要包括价格、品牌、服务等，价值属性主要评估一个测试工具在不分背景、不论出处的情况下的应用价值。

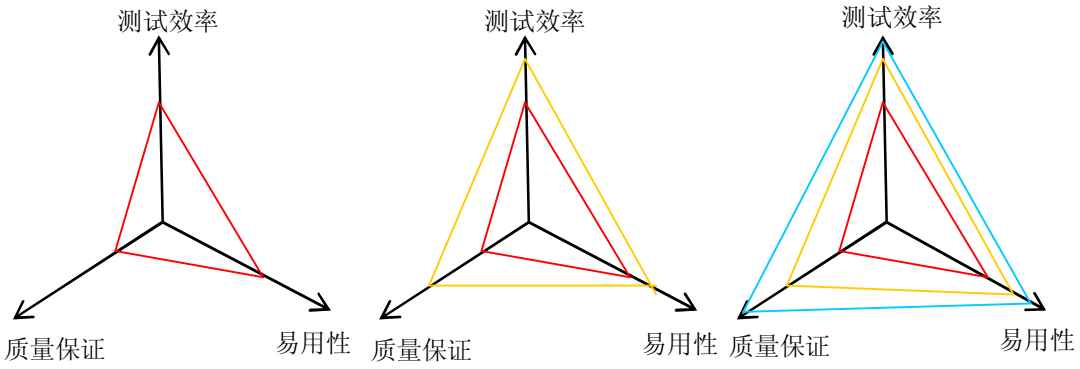
评估测试工具主要是对比功能，但同类测试工具功能差异可能很大，比如两个工具遵循的理念大相径庭，比如拿 XUnit 与 CodeTest 作对比，前者属第 3 代白盒方法，后者是第 2 代方法，后者除覆盖测试还支持性能测试，针对这种可比性存在偏差的情况该如何处理？这时，最忌讳的是失去自我主见，倾向某个工具就以它现成功能为基准列出评估项目。恰当的做法应是：先理清自己到底想要什么，关心哪些问题，然后看看两个工具支撑得怎么样，如果某关键功能两者都不支持，评估者可能还要另想招去解决。

由于白盒工具的核心功能比较确定，都离不开写脚本做测试，要打桩、构造测试数据、看覆盖率等，所以，鉴于本领域的特殊性，我们不必按功能逐项评估，只需评价这些功能服务于“测试效率提升”、“工具易用性”、“对质量保证的促进”的能力如何就可以了。如果被评价的工具在常规功能之外的，如 CodeTest 的性能测试，我们另列，按差异化功能逐条列出即可。将差异化功能分开考虑，可保证某些非核心的条目干扰我们决策过程，尤其在对比遵循不同方法论的工具时，象第 4 代白盒方法要求在线测试（在线测试驱动、在线脚本桩、在线测试改进），前 3 代白盒方法对此不作要求，如果按功能对比，可比性缺失，但按照设置这些功能的目的还是可以比较的，在线测试主要目的是提高测试效率。

又如，某工具提供“指定参数范围后用例自动生成”的功能，另一个工具没有，那是不是前者比后者一定好出不少呢？未必，这个功能只对工具易用性与工作效率产生影响，但如果指定参数范围要弹出一个接一个对话框，用户要不停点击鼠标，还用键盘输入上限值与下限值，那他的工作效率没有提升，反倒下降了，因为如此选参数远没有写一句 for 循环脚本来得快。

所以，我们以期望工具能达到的目标为基准，以“测试效率提升”、“工具易用性”、“对质量保证的促进”3 个大类作评估，分析起来更为准确。

把工具的价值属性与商务属性分开是有现实意义的，因为嵌入式领域的白盒测试普遍不够成熟，技术难度与风险偏高，考虑工具的实用价值不应受商务因素干扰。另外，本企业的白盒实践是否成熟，对工具选型影响也比较大，把价值属性独立出来利于在不同阶段给它分配不同权重。下图描述了白盒测试实践从强制推动区向自发推动区演进时，在各阶段应关注的不同重点：




强制推动区　　组织推动区　　自发推动区

处于强制推动区，企业应更关心选择高效率的测试工具，让白盒测试能做得下去，所以测试工具的使用效率与易用性最为关键，要达到一定的基准（否则如果用不下去就一票否决了），此阶段的白盒测试即便没有太明显的绩效也算成功了，因为测试做得下去，就有了可改善的基础。在组织推动区，须要解决一些基础的质量保证问题，至少，要保证白盒测试实践富于实效，测试能深入进去，比如测试先行等实践能有效的支撑起来。到了自发推动区，工具在测试效率、易用性与质量保证各环节都达到较高水准，对质量环节提升的要求更多些，此时质量要求集中于“平稳、无风险、可拷贝的运作”。

本评估模型的一个实例化表格如下：

| 大类 | 小类 | 评估项目 | 权重 |
|---|---|---|---|
| 差异化功能 | 功能差异 1 | | |
| | 功能差异 2 | | |
| | 功能差异 3 | | |
| 价值属性 | 测试效率 | 各项测试操作的使用效率与衔接能力（测试设计/调试/运行及改进等） | |
| | | 对其它研发过程的重用或促进的能力（代码审查、调试、问题定位、其它类型测试等） | |
| | | 构造测试数据的能力与效率 | |
| | | 测试代码快速生成能力 | |
| | | 编辑器的使用效率 | |
| | 易用性 | 操作是否直观（否在线操作，是否所见即所得，拖拉、拷贝等操作的易用程度） | |
| | | 操作是否便捷（快捷键、提示信息如何，在线帮助是否易用） | |
| | | 工具的学习成本 | |

| | | | |
|---|---|---|---|
| | | 配套工具集成性 | |
| | 质量保证能力 | 支持测试设计的深度 | |
| | | 测试能力的完备程度 | |
| | | 测试评估手段 | |
| | | 持续平稳运作的能力 | |
| | | 报表与统计 | |
| 商务属性 | 产品价格 | | |
| | 技术支持 | | |
| | 产品成熟度 | | |
| | 提供配套解决方案能力 | | |
| | 品牌 | | |

# 工具选型常见误区

上面已详细介绍嵌入式白盒工具选型的过程与方法，这里我们再补充选型中常见的认识误区：

1. 过于追求不必写脚本的测试

   业界有少数白盒工具，宣称不必编写脚本（一行都不写）就完成测试，因为所有用例都是自动生成的。此类工具学习成本较低，易用性无疑比较好，但不见得完全符合测试的价值，如果它生成的脚本是自闭的，不允许用户修改，那么这个优点会立即变为致命缺点。

   因为测试代码与被测代码是对等的，代码的性质，包括复杂性与易变性很接近，甚至，如果进行完备测试的话，测试脚本的规模应接近于被测代码的规模，所以，一行脚本都不写的测试有点难以理喻，如果不是功能太弱，就是操作太复杂，这是必然的，操作界面一个接一个，表面上易用性提高了，但用例的设计、调试、维护效率直线下降。

2. 找一款让白盒测试一步到位的工具

   相对产品研发的其它软件过程，白盒测试总体还不够成熟，还没有充分发展，尤其是嵌入式领域的白盒测试，运行环境复杂，又主要针对 C 代码做测试，推动白盒测试的门槛较高。所以，我们不必寄希望引入一个工具就解决所有问题，至少，白盒测试过程不纯粹是工具的问题，还与流程管理、人员素质等因素相关。

   企业应该规划白盒测试的发展线路图，从强制推动区到组织推动区，再到自发推动区循次渐进的去发展。许多事情不可一蹴而就，尤其在组织推动区向自发推动区迈进时，测试要向深层次发展，推行测试先行等实践需要有个过程。这对工具选型提出要求：最好选择跨越各阶段，能够持续促进的白盒工具。

3. 追求功能大而全

   商用白盒工具除了支持写脚本做测试，还通常具备其它辅助功能，比如：静态代码检查，代码可维护性评估，编程规范检查，支持性能测试、内存泄露检查等。毫无疑问，一个工具支持功能越多当然越好，但如果因为多支持一些功能而大幅抬升售价，评估者就不能不小心权衡了。

一般而言，多数附带上述辅助功能的商用工具，辅助功能的品质往往不能与专业工具相提并论，比如 C/C++语言的代码检查，业界最佳是 pclint，代码质量或编程规范检查，业界较佳的是 logiscope，内存泄露既有商用工具 purify、BoundCheck，也有不少免费工具，如 Visual Leak Detector。如果一个白盒工具的拥有良好的外部命令扩展能力，其它工具如 pclint、Visual Leak Detector 等能顺利的嵌入或配合着使用，就达到最佳组合，不仅整体功能强大，支付费用也低。

4. 不从自己的需求出发，以某工具现有功能为基准罗列评估条目

前面已提及，这里再强调一下，对白盒测试理解尚不深入的企业很容易误入歧途。

本文最后，我们推荐几款适用于嵌入式软件的常见白盒测试工具，下表供大家参考：

| 工具名称 | 厂商 | 所属测试方法 |
|---|---|---|
| VcTester | ezTester inc. | 第 4 代白盒测试方法 |
| CppUnit/CUnit | 开源工具 | 第 3 代白盒测试方法 |
| CodeTest | METROWERKS | 第 2 代白盒测试方法 |
| RTRT | IBM Rational | 第 2 代白盒测试方法 |
| Cantata++ | IPL | 第 2 代白盒测试方法 |
| Logiscope | Telelogic | 第 2 代白盒测试方法 |
| C++Test | Parasoft | 第 2 代白盒测试方法 |
| TrueCoverages | NuMega | 第 2 代白盒测试方法 |
| VectorCast | Vector Software | 第 2 代白盒测试方法 |
| PureCoverage | Rational | 第 2 代白盒测试方法 |

参考文献：

1. Wayne Chan，《第 4 代白盒测试方法介绍》
2. ezTester，《企业如何推行白盒测试》
3. ezTester，《实施白盒测试的几个误区》