# 白盒测试

白盒测试:又称结构测试、透明盒测试、逻辑驱动测试或基于代码的测试。白盒测试是一种测试用例设计方法，盒子指的是被测试的软件， 白盒指的是盒子是可视的，你清楚盒子内部的东西以及里面是如何运作的。"白盒"法全面了解程序内部逻辑结构、对所有逻辑路径进行测试。"白盒"法是穷举路 径测试。在使用这一方案时，测试者必须检查程序的内部结构，从检查程序的逻辑着手，得出测试数据。贯穿程序的独立路径数是天文数字。

采用什么方法对软件进行测试呢？常用的软件测试方法有两大类：静态测试方法和动态测试方 法。其中软件的静态测试不要求在计算机上实际执行所测程序，主要以一些人工的模拟技术对软件进行分析和测试；而软件的动态测试是通过输入一组预先按照一定 的测试准则构造的实例数据来动态运行程序，而达到发现程序错误的过程。在动态分析技术中,最重要的技术是路径和分支测试。下面要介绍的六种覆盖测试方法属 于动态分析方法。

**测试方法：**

       白盒测试的测试方法有代码检查法、静态结构分析法、静态质量度量法、逻辑覆盖法、基本路径测试法、域测试、符号测试、路径覆盖和程序变异。

白盒测试法的覆盖标准有逻辑覆盖、循环覆盖和基本路径测试。其中逻辑覆盖包括语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖和路径覆盖。六种覆盖标准发现错误的能力呈由弱到强的变化：

1.语句覆盖每条语句至少执行一次。

2.判定覆盖每个判定的每个分支至少执行一次。

3.条件覆盖每个判定的每个条件应取到各种可能的值。

4.判定/条件覆盖同时满足判定覆盖条件覆盖。

5.条件组合覆盖每个判定中各条件的每一种组合至少出现一次。

6.路径覆盖使程序中每一条可能的路径至少执行一次。

**要求：**

1.保证一个模块中的所有独立路径至少被使用一次。

2.对所有逻辑值均需测试 true 和 false。

3.在上下边界及可操作范围内运行所有循环。

4.检查内部数据结构以确保其有效性。

**目的：**

       通过检查软件内部的逻辑结构，对软件中的逻辑路径进行覆盖测试;在程序不同地方设立检查点，检查程序的状态，以确定实际运行状态与预期状态是否一致。

**特点：**

      依据软件设计说明书进行测试、对程序内部细节的严密检验、针对特定条件设计测试用例、对软件的逻辑路径进行覆盖测试。

**实施步骤：**

1.测试计划阶段：根据需求说明书，制定测试进度。

2.测试设计阶段：依据程序设计说明书，按照一定规范化的方法进行软件结构划分和设计测试用例。

3.测试执行阶段：输入测试用例，得到测试结果。

4.测试总结阶段：对比测试的结果和代码的预期结果，分析错误原因，找到并解决错误。

# 优缺点

**优点**

1.迫使测试人员去仔细思考软件的实现

2.可以检测代码中的每条分支和路径

3.揭示隐藏在代码中的错误

4.对代码的测试比较彻底

5.最优化

**缺点**

1.昂贵

2.无法检测代码中遗漏的路径和数据敏感性错误

3.不验证规格的正确性

**局限**

但即使每条路径都测试了仍然可能有错误。可能出现的情况如下：

1.穷举路径测试决不能查出程序违反了设计规范，即程序本身是个错误的程序。

2.穷举路径测试不可能查出程序中因遗漏路径而出错。

3.穷举路径测试可能发现不了一些与数据相关的错误。

**工具挑选**

白盒测试主要用在具有高可靠性要求的软件领域，例如：军工软件、航天航空软件、工业控制软件等等。白盒测试工具在选购时应当主要是对开发语言的支持、代码覆盖的深度、[嵌入式软件](https://baike.baidu.com/item/%E5%B5%8C%E5%85%A5%E5%BC%8F%E8%BD%AF%E4%BB%B6)的测试和测试的可视化等。

[](https://baike.baidu.com/pic/%E7%99%BD%E7%9B%92%E6%B5%8B%E8%AF%95/934440/0/e78c6589ccc1718a0e24446e?fr=lemma&ct=single)

### 语言支持

白盒测试工具是对[源代码](https://baike.baidu.com/item/%E6%BA%90%E4%BB%A3%E7%A0%81)进行的测试，测试的主要内容包括[词法分析](https://baike.baidu.com/item/%E8%AF%8D%E6%B3%95%E5%88%86%E6%9E%90)与[语法分析](https://baike.baidu.com/item/%E8%AF%AD%E6%B3%95%E5%88%86%E6%9E%90)、[静态](https://baike.baidu.com/item/%E9%9D%99%E6%80%81)[错误分析](https://baike.baidu.com/item/%E9%94%99%E8%AF%AF%E5%88%86%E6%9E%90)、动态检测等。但是对于不同的开发语言，测试工具实现的方式和内容差别是较大的。测试工具主要支持的开发语言包括：标准C、C++、Visual C++、Java和Visual J++等。

### 覆盖深度

从覆盖源[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F" \t "_blank)语句的详尽程度分析，[逻辑覆盖](https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E8%A6%86%E7%9B%96)标准包括以下不同的覆盖标准：语句覆盖、判定覆盖、[条件覆盖](https://baike.baidu.com/item/%E6%9D%A1%E4%BB%B6%E8%A6%86%E7%9B%96)、条件判定组合覆盖、多条件覆盖和修正[判定条件覆盖](https://baike.baidu.com/item/%E5%88%A4%E5%AE%9A%E6%9D%A1%E4%BB%B6%E8%A6%86%E7%9B%96)。

1.语句覆盖 为了暴露[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)中的错误，程序中的每条语句至少应该执行一次。因此语句覆盖（Statement Coverage）的含义是：选择足够多的测试数据，使被测[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F" \t "_blank)中每条语句至少执行一次。语句覆盖是很弱的[逻辑覆盖](https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E8%A6%86%E7%9B%96)。

2.[判定覆盖](https://baike.baidu.com/item/%E5%88%A4%E5%AE%9A%E8%A6%86%E7%9B%96)比[语句覆盖](https://baike.baidu.com/item/%E8%AF%AD%E5%8F%A5%E8%A6%86%E7%9B%96)稍强的覆盖标准是判定覆盖（Decision Coverage）。[判定覆盖](https://baike.baidu.com/item/%E5%88%A4%E5%AE%9A%E8%A6%86%E7%9B%96" \t "_blank)的含义是：设计足够的[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)，使得[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)中的每个判定至少都获得一次“真值”或“假值”，或者说使得程序中的每一个取“真”分支和取“假”分支至少经历一次，因此判定覆盖又称为[分支覆盖](https://baike.baidu.com/item/%E5%88%86%E6%94%AF%E8%A6%86%E7%9B%96" \t "_blank)。

3.[条件覆盖](https://baike.baidu.com/item/%E6%9D%A1%E4%BB%B6%E8%A6%86%E7%9B%96)在[设计程序](https://baike.baidu.com/item/%E8%AE%BE%E8%AE%A1%E7%A8%8B%E5%BA%8F)中，一个判定语句是由多个条件组合而成的复合判定。为了更彻底地实现[逻辑覆盖](https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E8%A6%86%E7%9B%96)，可以采用[条件覆盖](https://baike.baidu.com/item/%E6%9D%A1%E4%BB%B6%E8%A6%86%E7%9B%96)（Condition Coverage）的标准。[条件覆盖](https://baike.baidu.com/item/%E6%9D%A1%E4%BB%B6%E8%A6%86%E7%9B%96" \t "_blank)的含义是：构造一组[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)，使得每一判定语句中每个逻辑条件的可能值至少满足一次。

4.多[条件覆盖](https://baike.baidu.com/item/%E6%9D%A1%E4%BB%B6%E8%A6%86%E7%9B%96" \t "_blank)。多条件覆盖也称[条件组合覆盖](https://baike.baidu.com/item/%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E8%A6%86%E7%9B%96)，它的含义是：设计足够的[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)，使得每个判定中条件的各种可能组合都至少出现一次。显然满足多[条件覆盖](https://baike.baidu.com/item/%E6%9D%A1%E4%BB%B6%E8%A6%86%E7%9B%96)的[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)是一定满足[判定覆盖](https://baike.baidu.com/item/%E5%88%A4%E5%AE%9A%E8%A6%86%E7%9B%96)、条件覆盖和条件判定组合覆盖的。

5.修正条件[判定覆盖](https://baike.baidu.com/item/%E5%88%A4%E5%AE%9A%E8%A6%86%E7%9B%96" \t "_blank)。修正条件判定覆盖是由欧美的航空/航天制造厂商和使用单位联合制定的“航空运输和装备系统[软件](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6" \t "_blank)认证标准”，在国外的国防、航空航天领域应用广泛。这个覆盖度量需要足够的[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B" \t "_blank)来确定各个条件能够影响到包含的判定的结果。它要求满足两个条件：首先，每一个[程序模块](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F%E6%A8%A1%E5%9D%97)的入口和出口点都要考虑至少要被调用一次，每个程序的判定到所有可能的结果值要至少转换一次；其次，程序的判定被分解为通过逻辑操作符（and、or）连接的[布尔](https://baike.baidu.com/item/%E5%B8%83%E5%B0%94" \t "_blank)条件，每个条件对于判定的结果值是独立的。

不同的测试工具对于[代码](https://baike.baidu.com/item/%E4%BB%A3%E7%A0%81" \t "_blank)的覆盖能力也是不同的，通常能够支持修正条件[判定覆盖](https://baike.baidu.com/item/%E5%88%A4%E5%AE%9A%E8%A6%86%E7%9B%96)的测试工具价格是极其昂贵的。

### 嵌入式测试

对于[嵌入式软件](https://baike.baidu.com/item/%E5%B5%8C%E5%85%A5%E5%BC%8F%E8%BD%AF%E4%BB%B6)的测试，我们还需要一方面进一步考虑测试工具对于[嵌入式操作系统](https://baike.baidu.com/item/%E5%B5%8C%E5%85%A5%E5%BC%8F%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F)的支持能力，例如DOS、Vxworks、Neculeus、Linux和Windows CE等；另一方面还需要考虑测试工具对于硬件平台的支持能力，包括是否支持所有64/32/16位CPU 和 MCU，是否可以支持 PCI/VME/CPCI 总线。

### 可视化

白盒测试是工作量巨大并且枯燥的工作，可视化的设计对于测试来说是十分重要的。在选购白盒测试工具时，应当考虑该款测试工具的可视化是否良好，例如：[测试过程](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%BF%87%E7%A8%8B)中是否可以显示覆盖率的[函数](https://baike.baidu.com/item/%E5%87%BD%E6%95%B0)分布图和上升趋势图，是否使用不同的颜色区分已执行和未执行的[代码段](https://baike.baidu.com/item/%E4%BB%A3%E7%A0%81%E6%AE%B5)显示分配内存情况实时图表等，这些对于测试效率和测试质量的提高是具有很大的作用的。

## 测试方法

白盒测试的测试方法中运用最为广泛的是基本路径测试法。

基本[路径测试](https://baike.baidu.com/item/%E8%B7%AF%E5%BE%84%E6%B5%8B%E8%AF%95)法是在程序控制流图的基础上，通过分析控制构造的环路复杂性，导出基本可执行[路径](https://baike.baidu.com/item/%E8%B7%AF%E5%BE%84)集合，从而设计测试[用例](https://baike.baidu.com/item/%E7%94%A8%E4%BE%8B)的方法。设计出的[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)要保证在测试中[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)的每个可执行语句至少执行一次。

在[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)控制流图的基础上，通过分析控制构造的环路复杂性，导出基本可执行路径集合，从而设计[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)。包括以下4个步骤和一个工具方法：

**1.**[**程序**](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)**的**[**控制流图**](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E6%B5%81%E5%9B%BE)：描述程序控制流的一种图示方法。

**2.**[**程序**](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)**圈复杂度**：McCabe复杂性度量。从[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)的环路复杂性可导出程序基本[路径](https://baike.baidu.com/item/%E8%B7%AF%E5%BE%84)集合中的独立路径条数，这是确定程序中每个可执行语句至少执行一次所必须的测试用例数目的[上界](https://baike.baidu.com/item/%E4%B8%8A%E7%95%8C)。

**3. 导出**[**测试用例**](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)：根据圈复杂度和[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)结构设计用例数据输入和预期结果。

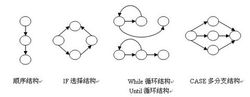
**4. 准备**[**测试用例**](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)：确保基本[路径](https://baike.baidu.com/item/%E8%B7%AF%E5%BE%84)集中的每一条路径的执行。

### 工具方法

1.图形[矩阵](https://baike.baidu.com/item/%E7%9F%A9%E9%98%B5)：是在基本[路径测试](https://baike.baidu.com/item/%E8%B7%AF%E5%BE%84%E6%B5%8B%E8%AF%95)中起辅助作用的[软件](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6)工具，利用它可以实现自动地确定一个基本路径集。

2.程序的[控制流图](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E6%B5%81%E5%9B%BE)：描述程序控制流的一种图示方法。

3.圆圈称为[控制流图](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E6%B5%81%E5%9B%BE)的一个结点，表示一个或多个无分支的语句或源[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F" \t "_blank)语句。

[](https://baike.baidu.com/pic/%E7%99%BD%E7%9B%92%E6%B5%8B%E8%AF%95/934440/0/5243fbf2b211931312e70a1866380cd790238df6?fr=lemma&ct=single)

流图只有二种图形符号：

1.图中的每一个圆称为流图的结点，代表一条或多条语句。

2.流图中的箭头称为边或连接，代表[控制流](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E6%B5%81)，任何过程设计都要被翻译成[控制流图](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E6%B5%81%E5%9B%BE)。

在将[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)流程图简化成[控制流图](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E6%B5%81%E5%9B%BE)时，应注意：

1.在选择或多分支结构中，分支的汇聚处应有一个汇聚结点。

2.边和结点圈定的区域叫做区域，当对区域计数时，图形外的区域也应记为一个区域。

### 步骤

基本[路径测试](https://baike.baidu.com/item/%E8%B7%AF%E5%BE%84%E6%B5%8B%E8%AF%95)法的步骤：

**第一步：画出**[**控制流图**](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E6%B5%81%E5%9B%BE)

流程图用来描述程序控制结构。可将[流程图](https://baike.baidu.com/item/%E6%B5%81%E7%A8%8B%E5%9B%BE)映射到一个相应的流图(假设流程图的菱形决定框中不包含复合条件)。在流图中，每一个圆，称为流图的结点，代表一个或多个语句。一个处理方框序列和一个菱形决测框可被映射为一个结点，流图中的箭头，称为边或连接，代表[控制流](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E6%B5%81)，类似于流程图中的箭头。一条边必须终止于一个结点，即使该结点并不代表任何语句(例如：if-else-then结构)。由边和结点限定的范围称为区域。计算区域时应包括图外部的范围。

**第二步：计算圈复杂度**

圈复杂度是一种为[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)逻辑复杂性提供定量测度的[软件度量](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%BA%A6%E9%87%8F)，将该度量用于计算程序的基本的独立路径数目，为确保所有语句至少执行一次的测试数量的[上界](https://baike.baidu.com/item/%E4%B8%8A%E7%95%8C)。独立路径必须包含一条在定义之前不曾用到的边。

有以下三种方法计算圈复杂度：

流图中区域的数量对应于环型的复杂性;

给定流图G的圈复杂度V(G)，定义为V(G)=E-N+2，E是流图中边的数量，N是流图中结点的数量;

给定流图G的圈复杂度V(G)，定义为V(G)=P+1，P是流图G中判定结点的数量。

**第三步：导出**[**测试用例**](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)**根据上面的计算方法，**可得出四个独立的路径。(一条独立路径是指，和其他的独立路径相比，至少引入一个新处理语句或一个新判断的[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)通路。V(G)值正好等于该[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)的独立路径的条数。)

路径1：4-14

路径2：4-6-7-14

路径3：4-6-8-10-13-4-14

路径4：4-6-8-11-13-4-14

根据上面的独立[路径](https://baike.baidu.com/item/%E8%B7%AF%E5%BE%84)，去设计输入数据，使[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)分别执行到上面四条路径。

**其他参考资料**：<https://blog.csdn.net/tihumenjiang/article/details/79324264>

<https://blog.csdn.net/zhang_xiaoyi/article/details/79007741>