白盒测试

白盒测试也称为结构测试、透明盒测试、逻辑驱动测试或基于代码的测试，主要用于检测软件编码过程中的错误。软件程序基本的语法错误在程序调试时，就能够很及时地发现，然后及时进行改正。但是软件程序在运算顺序、逻辑判断以及运行路径上的错误很难发现。也就是说白盒测试是在测试程序结构已知的情况下进行的，也称结构测试或逻辑驱动测试，通过测试来检测产品内部动作是否按照规格说明书的规定正常进行，按照程序内部的结构测试程序，检验程序中的每条通路是否都有能按预定要求正确工作，而不顾它的功能，能全面地测试程序代码结构。

思想与方法

白盒测试是程序结构分析，根据源代码可以首先绘制程序的流程图，然后根据流程图分析程序的结构，再对所有的逻辑路径进行穷举测试。在使用这一方法时，测试者从检查程序的逻辑着手，得出测试数据。白盒测试时，一般要：

① 对程序模块的所有独立的执行路径至少测试一次；
② 对所有的逻辑判定，取“真”与取“假”的两种情况都能至少测试一次；
③ 在循环的边界和运行界限内执行循环体；
④ 测试内部数据结构的有效性。

白盒测试基本方法有：

①语句覆盖

作为最基本的逻辑覆盖方法，语句覆盖的含义是：选择足够多的测试数据，使得被测程序中的每个语句至少执行一次。通过语句覆盖，可以直观地从源代码得到测试用例，无须细分每条判定表达式；

②判定覆盖

判定覆盖也称分支覆盖。其含义为：不仅每个语句必须至少执行一次，而且每个判定的每种可能的结果都应该至少执行一次，即每个判定的每个分支都至少执行一次判定覆盖相对于语句覆盖，其逻辑覆盖能力更强。

③条件覆盖

条件覆盖的含义是，不仅每个语句至少执行一次，而且使判定表达式中的每个条件都取到各种可能的结果。相对于判定覆盖，条件覆盖的覆盖能力更强，因为判定覆盖只关心整个判定表达式的值，而条件覆盖使判定表达式中每个条件都取到了不同的结果。

④判定 / 条件覆盖

一种既能满足判定覆盖标准又能满足条件覆盖标准的覆盖方法，其含义是：选取足够多的测试数据，使得判定表达式中的每个条件都取到各种可能的值，而且每个判定表达式也都取到各种可能的结果。

⑤条件组合覆盖

条件组合覆盖是更强的逻辑覆盖标准，其含义是：选取足够多的测试数据，使得每个判定表达式中条件的各种可能组合都至少出现一次。满足条件组合覆盖准则的测试数据必然满足判定覆盖、条件覆盖和判定／条件覆盖准则。

⑥路径覆盖

路径覆盖要求选取足够多的测试数据，覆盖序中所有可能的路径。其优点是：可以对程序进行彻底的测试，比前述五种的覆盖面都广。

优势与劣势

白盒测试较为容易自动化进行，能增大测试代码的覆盖率，检测隐藏的错误。程序员根据测试得到的结果和数据进行代码优化，可以消除程序可能存在的缺陷。

但是一个大型工程的结构往往是十分复杂的，工程的所有的独立路径会非常多，一般来说不可能把所有路径全部测试。另外由于穷举路径测试是对现有程序的测试，无法检测出代码本身是否符合要求，无法查出程序是否遗漏了某些路径，可能也检查不出同数据有关的错误。也就是说白盒测试只能检查出现有程序本身的结构错误，但对程序本身是否正确无法保证。

黑盒测试

黑盒测试也称为功能测试、数据驱动测试或基于规格说明的测试。测试者不了解程序的内部情况，不需具备应用程序的代码、内部结构和编程语言的专门知识，只知道程序的输入、输出和系统的功能。是从用户的角度针对软件界面、功能及外部结构进行的测试，不考虑程序内部逻辑结构。测试案例是依应用系统应该做的功能，照规范、规格或要求等设计。测试者选择有效输入和无效输入来验证是否正确的输出。

测试方法

(一)等价类划分法

等价类划分法是一种典型的、重要的黑盒测试方法，它将程序所有可能的输入数据划分为若干个等价类。然后从每个部分中选取具有代表性的数据当做测试用例。测试用例由有效等价类和无效等价类的代表数据组成，从而保证测试用例具有完整性和代表性。

使用该方法设计测试用例主要有两个步骤：

1.确定等价类

等价类是指被测软件的一个输入数据的集合，该集合中的任一元素对于揭露被测程序中的错误而言是等价的，即若该集合中的一个元素测试程序发现不了某类功能上明显的错误，那么其它元素测试该程序也发现不了这种错误。

确定等价类是将每一个输入条件划分为有效等价类和无效等价类。有效等价类指程序规格说明书中规定的、合理的、有意义的输入数据。通过测试有效等价类中的数据可以测试被测软件是否实现了规格说明书中预先规定的功能和性能。无效等价类是有效等价类的补集，指软件规格说明书中没有规定的、没有意义的、不合理的输入数据集合。

2.生成测试用例

①为每一个等价类设置一个唯一的编号；

②设计新的测试用例，尽可能多地覆盖那些尚未被覆盖的有效等价类，直到所有的有效等价类都被测试用例所覆盖(包含进去)；

③设计新的测试用例，覆盖一个仅一个尚未被覆盖的无效等价类，直到所有的无效等价类都被测试用例所覆盖。

(二)边界值分析法

边界值分析法是对程序输入或输出的边界值进行测试的一种黑盒测试方法。实际的测试工作证明，考虑了边界条件的测试用例比那些没有考虑边界条件的测试用例具有更高的测试回报率。这里所说的边界条件，是指输入和输入等价类中那些恰好处于边界、或超过边界、或在边界以下的状态。

利用边界值分析法设计测试用例的原则：

1.如果输入条件规定了值的范围，那么应针对范围的边界设计有效的等价类测试用例，针对刚刚越界的情况设计无效等价类输入测试用例。

2.如果输入条件规定了输入值的数量(包括个数的多少，时间的长短)，则应对该数量的最大值、最小值及比最大值小一、最小值大一的情况分别设计有效的输入测试用例。

3.如果程序中使用了一个内部数据结构，则应该内部数据机构的边界值设计测试用例。

4.如果程序的规格说明给出的输入域或输出域是有序集合，则应该取集合的第一个元素和最后一个元素设计测试用例。

(三)因果图法

因果图法也是较常用的一种黑盒测试方法，是一种简化了的逻辑图。因果图能直观地表明输入条件和输出动作之间的因果关系，能帮助测试人员把注意力集中到与程序功能有关的输入组合上。

因果图法是一种适合于描述对于多种输入条件组合的测试方法，根据输入条件的组合、约束关系和输出条件的因果关系，分析输入条件的各种组合情况，从而设计测试用例的方法，它适合于检查程序输入条件的各种组合情况。

利用因果图法设计测试用例的步骤：

1.将规格说明分解为可执行的片段。该步骤必不可少，因为因果图不善于处理较大的规格说明。

2.分析并确定可执行片段中哪些是原因，哪些是结果。原因是指输入条件或者输入条件的等价类，而结果指输出条件。

3.为每一个原因和结果赋予唯一的标号，并根据规格说明书中的描述，画出因果图。

4.通过仔细地跟踪图中的状态变化情况，将因果图转换成一个有限项的判定表。表中的每一列代表一个测试用例。

(四)错误推测法

错误推测法是基于以往的经验和直觉，参照以往的软件系统出现的错误，推测当前被测程序中可能存在的缺陷和错误，有针对性地设计测试用例。

用错误推测法设计测试用例的基本思想是：列举出程序中可能犯出现的错误或容易发生错误的特殊情况的清单，然后根据清单和已经设计好的测试用例来编写特定的测试用例。例如，程序中出现的输入数据为“0”或者字符为空就是一种错误易发情况；在出现输入或输出的数量不定的地方，数量为“没有”和“一个”也是错误易发情况。特别需要注意的是，在阅读规格说明时联系程序员可能做的假设来确定测试用例，测试人员要站在用户的角度来考虑输入信息，而不必去管这些信息对于被测程序是合理还是不合理的输入。

优势与劣势

黑盒测试较为简单，不需要了解程序内部的代码及实现。这种测试方法侧重于输入的设计，与程序的内部实现无关，当程序发生改变而设计规格不变时，原有设计的测试用例可重复使用。但是黑盒测试代码覆盖率较低，同样存在仅靠数据无法检测的错误。

白盒测试与黑盒测试

在定义上：

白盒测试需要从代码句法发现内部代码在算法，溢出，路径，条件等等中的缺点或者错误，进而加以修正。而黑盒测试着重测试软件功能，它并不涉及程序的内部结构和内容特性。黑盒测试并不能取代白盒测试，它与白盒是互补的测试方法，它很可能发现白盒测试不易发现的其他类型错误。

在测试目的上：

黑盒测试的目的是检测是否有不正确或遗漏的功能；数据或者参数上，输入能否正确接收；是否有数据结构错误或外部信息访问错误；性能上是否能够满足要求；是否有初始化或终止性错误。而白盒测试的目的是通过在不同点检查程序的状态，确定实际的状态是否与预期的状态一致，而不顾它的功能。

在检测方式上：

白盒测试是穷举路径测试，黑盒测试是穷举输入测试，这两种方法是基于完全不同的观点，反应了事物的两个极端，它们各有侧重和优势，但不能彼此替代。在现代的测试理念中，这两种测试方法不是截然分开的，而是交叉使用。

实际应用测试中一般都是白盒测试和黑盒测试一同交叉使用的。