白盒子测试方法

软件测试方法：分为两类

（1）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

（2）动态测试：通过输入一组预先按照一定的测试准则构造的实例数据动态运行程序，而达到发现程序错误的过程，特点如下：

1 必须生成测试数据来运行被测试程序，取得程序运行的真实情况、动态情况，进而进行分析

1 测试质量依赖于测试数据

1 生成测试数据，分析测试结果的工作量大，使开展测试工作费时、费力、费人

1 动态测试中涉及多方面工作，人员多，设备多，数据多，要求有较好的管理和工作规程

一．概述

1． 定义

也称结构测试或逻辑驱动测试，按照程序内部的结构对程序进行测试，通过测试来检查产品内部动作是否按照设计规格说明书的规定正常进行，检查程序中的每条通路是否能按照预定要求正确工作

2． 测试内容

把测试对象看成是一个打开的盒子，测试人员依据程序内部逻辑结构相关信息，设计或选择测试用例，对程序的所有逻辑路径进行测试，通过不同点检查程序的状态，确定实际的状态与预期的状态一致

3． 测试基本技术

（1）词法分析与语法分析

（2）静态错误分析

（3）程序插桩技术

4． 测试方法

（1）代码检查法

（2）静态结构分析法

（3）静态质量度量法

（4）逻辑覆盖法

（5）基本路径测试法

（6）域测试

（7）符号测试

（8）Z 路径覆盖

（8） 程序变异

5．黑盒测试与白盒测试

黑盒测试 白盒测试

不涉及程序结构 考查程序逻辑结构

用软件规格说明书生成测试用例 用程序结构信息生成测试用例

可适用于从单元测试到系统联调 适用于单元测试和集成测试

某些代码段得不到测试 对所有逻辑路径进行测试

二．白盒测试基本技术

1．词法和语法分析

（1）获取信息

1 可以获取软件组成的重要基本因数，如变量标识符、过程标识符、常量等

l 组合获取的基本因数，可以得到软件的基本信息，如：

v 标号交叉引用表：列出各模块中出现的全部标号及标号的属性，模块以外的全局、计算标号

v 变量交叉引用表：列出变量定义及引用信息，变量的属性，变量类型（全局、局部）

v 子程序、宏和函数表：列出各个子程序、宏及函数的属性，输入、输出参数信息

v 等价表：列出在等价语句和等值语句中出现的全部变量和标号

v 常数表：列出全部数字常数和字符常数

（2）作用

l 直接从表中查出说明/使用错误，如标号交叉引用表、变量交叉引用表

l 为用户提供辅助信息，如子程序、宏和函数表、等价表、常数表

l 用来做错误预测和程序复杂度计算，如操作符和操作数的统计表

2．静态错误分析

用于确定在源程序中是否有某类错误或‘危险’结构，包括以下几种：

（1） 类型和单位分析

对源程序的类型进行检查，为了强化检查效果，扩充一些新的数据类型，进行静态预处理程序，分析程序中的类型错误

（2） 引用分析

l 对程序中变量的引用进行检查，发现引用异常错误（如变量在定义前被引用，变量定义后未被引用）。

l 采用深度优选的方法遍历程序流图的每一条路径

l 建立引用异常的探测工具，包括变量定义表和变量引用表

（3） 表达式分析

对表达式进行分析，以发现和纠正在表达式出现的错误，如：

l 在表达式中不正确的使用了括号造成错误

l 数组下标越界错误

l 除数为零

l 浮点数计算的误差（最复杂）

（4） 接口分析

接口一致性是程序的静态错误分析和设计分析共同研究的题目，接口分析主要对下内容时进行一致性的分析：

l 各模块之间接口一致性

l 模块与外部数据库的接口一致性

l 形参与实参在类型，数量，顺序，维数，使用上的一致性

l 全局变量和公共数据区在使用上的一致性

3．程序插桩技术

（1） 概述

在动态测试中，是一种基本的测试手段，有广泛的应用

主要借助向程序中插入操作，来实现测试目的的方法（即向源程序中添加一些语句（也称探测器），实现对程序语句的执行、变量的变化等情况进行检查）

（2） 设计时考虑的问题

l 明确要探测哪些信息

l 在程序的什么部位设置探测点

l 需要设计多少个探测点

（3） 探测点设置位置（以 Fortran 为例）

l 程序块的第一个可执行语句之前
l entry 语句的前后
l 有标号的可执行语句处
l 循环语句之后
l 条件语句之后
l logical if 语句之后
l call 语句之后
l go to 语句之后

（4） 断言语句

在程序中的特定部位插入某些用以判断变量特性的语句，使得程序执行中这些语句得以证实，从而使程序的运行特性得到证实，我们把这些插入的语句称为断言语句。

三．白盒测试方法－静态测试

1． 代码检查法

（1） 目的

通过桌面检查，代码审查和走查方式，对以下内容进行检查

l 检查代码和设计的一致性
l 代码对标准的遵循、可读性
l 代码逻辑表达的正确性
l 代码结构的合理性
l 程序编写与编写标准的符合性
l 程序中不安全、不明确和模糊的部分
l 编程风格问题等

（2） 代码检查方式

方式名称 执行人员 检查内容 检查过程

桌面检查 程序员 对源程序代码进行分析、检验，并补充相关的文档，发现程序中的错误

代码审查 程序员和测试员组成的审查小组 通过阅读、讨论和争议，以程序进行静态分析的过程 第一步：小组成员提前阅读设计规格书、程序文本等相关文档第二步：召开程序审查会，开发人员读程序，审查小组讨论、发现、解决问题

走查 程序员和测试员组成的审查小组 通过逻辑运行程序，发现问题 第一步：小组成员提前阅读设计规格书、程序文本等相关文档第二步：利用测试用例，使程序逻辑运行，记录程序的踪迹，发现、讨论、解决问题

（3） 代码检查项目（采用分析技术）

l 检查变量的交叉引用表：检查未说明的变量和违反了类型规定的变量，变量的引用和使用情况
l 检查标号的交叉引用表：验证所有标号的正确性
l 检查子程序、宏、函数：验证每次调用与所调用位置是否正确，调用的子程序、宏、函数是否存在，参数是否一致
l 等价性检查：检查全部等价变量的类型的一致性
l 常量检查：确认常量的取值和数制、数据类型
l 标准检查：检查程序中是否违反标准的问题
l 风格检查：检查程序的设计风格
l 比较控制流：比较设计控制流图和实际程序生成的控制流图的差异
l 选择、激活路径：在设计控制流图中选择某条路径，到实际的程序中激活这条路径，如果不能激活，则程序可能有错

l 对照程序的规格说明，详细阅读源代码，比较实际的代码，从差异中发现程序的问题和错误
l 补充文档
根据以上检查项目，可以编制代码规则，规范和检查表等作为测试用例
（4） 编码规范
程序编写过程中必须遵守的规则，规定代码的语法格式、语法规则，如排版、注释、标识符命名、可读性、变量、函数、过程、可测性、程序效率、质量保证、代码编辑、编译、审查、代码测试、维护、宏等各方面的编码要求
（5） 代码检查规则
对程序逻辑结构检查时，所规定的规则，形成
（6） 缺陷检查表
主要包括一些容易出错的地方和在以往工作中遇到的典型错误，形成表格形式
重要性 审查项 结论
文件结构 重要 头文件和定义文件的名称是否合理
2． 静态结构分析法
在静态结构分析中，测试者通过使用测试工具分析程序源代码的系统结构、数据结构、数据接口、内部控制逻辑等内部结构，生成函数调用关系图、模块控制流图、内部文件调用关系图等各种图形图表，清晰地标识整个软件的组成结构，便于理解，通过分析这些图表，检查软件有没有存在缺陷或错误；包括控制流分析、数据据流分析、接口分析、表达式分析
（1） 函数调用关系图：通过应用程序各函数之间的调用关系展示了系统的结构。列出所有函数，用连线表示调用关系，作用：
l 可以检查函数的调用关系是否正确
l 是否存在孤立的函数而没有被调用
l 明确函数被调用的频繁度，对调用频繁的函数可以重点检查
（2） 模块控制流图：由许多结点和连接结点的边组成的图形，其中每个结点代表一条或多条语句，边表示控制流向，可以直观地反映出一个函数的内部结构。
*例子1－GIS 软件：存在无法执行的死代码；有多个出口，可能没有在所有出口进行内存释放与回收，有内存泄露的可能
*例子2－MIS 软件：有多个出口，存在内存泄露的可能；有10逻辑判断结点，易出现逻辑错误，降低可靠性，可能会破坏对 CPU 操作进行优化的处理，影响其运行性能
3． 静态质量度量法
（1） 软件质量：根据 ISO/IEC9126 国际标准，包括以下六个方面：
l 功能性（functionality）
l 可靠性（reliability）
l 可用性（usability）
l 有效性（efficiency）
l 可维护性（maintainability）
l 轻便性（portability）
（2） 质量度量模型（从上到下）
l 质量因素（Factors）：与分类标准的计算方式相似，依据各分类标准取值组合权重方法来计算，依据结果将软件质量分为四个等级，与分类标准等级内容相同
l 分类标准（criteria）：对某一软件质量分为不同的分类标准，每个分类标准由一系列度量规则组成，每个规则分配一个权重，每个分类标准的取值由规则的取值与权重值计算得出，依据结果将软件质量分为四个等级：

v 优秀（excellent）：符合本模型框加中的所有规则（可以接受）
v 良好（good）：未大量偏离模型框架中的规则（可以接受）
v 一般（fair）：违背了模型框架中的大量规则（可以接受）
v 较差（poor）：无法保障正常的软件可维护性（不可以接受）
l 度量规则（Metrics）：使用代码行数、注释频度等参数度量软件各种行为属性
四． 白盒测试方法－动态测试（即设计测试用例的方法）
1． 白盒测试的动态测试原则－根据程序的控制结构设计测试用例
（1） 保证每个模块的所有独立路径至少被使用一次
（2） 对所有的逻辑值均测试 true 和 false
（3） 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
2． 逻辑覆盖法
（1） 概述
逻辑覆盖是通过对程序逻辑结构的遍历实现程序的覆盖
（2） 分类－依据覆盖源程序语句的详尽程度
l 语句覆盖 SC（Statement Coverage）
l 判定覆盖 DC（Decision coverage）
l 条件覆盖 CC（Condition Coverage）
l 条件判定组合覆盖 CDC（Condition/ Decision Coverage）
l 多条件覆盖 MCC （Multiple Condition Coverage）
l 修改条件判定覆盖 MCDC（Multiple Condition Decision Coverage）
（3） 语句覆盖
l 选择足够多的测试数据，使被测程序中每条语句至少执行一次
l 缺点：对程序执行逻辑的覆盖很低
（4） 判定覆盖
l 设计足够多的测试用例，使得程序中的每一个判定至少获得一次‘真’值和‘假’值，或者使得程序中的每一个取‘真’分支或取‘假’分支至少经历一次，因此又称分支覆盖
l 可以满足语句覆盖
l 缺点：主要对整个表达式最终取值进行度量，忽略了表达式内部取值
（5） 条件覆盖
l 设计足够多的测试用例，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
l 不能够满足判定覆盖
（6） 条件判定组合覆盖
l 设计足够多的测试用例，使得判定中的每个条件的所有可能（真/假）至少出现一次，并且每个判定本身的判定结果也至少出现一次
l 缺点：没有考虑单个判定对整体结果的影响，无法发现逻辑错误
（7） 多条件覆盖
l 也称条件组合覆盖，设计足够多的测试用例，使得每个判定中条件的各种可能组合都至少出现一次（以数轴形式划分区域，提取交集，建立最少的测试用例）
l 满足条件覆盖一定满足判定覆盖、条件覆盖、条件判定组合覆盖
l 缺点：判定语句较多时，条件组合值比较多
（8） 修正条件判定覆盖
l 每一个程序模块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

l 程序的判定被分解为通过逻辑操作符（and，or）连接的 bool 条件，每个条件对于判定的结果值是独立的

练习1：采用多条件覆盖方法，对下程序进行白盒测试用例设计

```
if ((a >1 )&&( b= = 0))
{
    x=x/a;
}
if (( a = = 2)‖ (x > 1 ))
{
    x=x+1;
}
```

3． 基本路径覆盖

（1） 概述

l 在程序控制流图的基础上，通过分析程序控制流图的环路复杂性，导出基本可执行路径的集合，然后据此设计测试用例

l 设计出的测试用例要保证在测试中程序的每一条可执行语句至少执行一次

（2） 程序控制流图

l 控制流图是描述程序控制流的一种方式

l 图形符号：圆圈代表一个结点 表示一个或多个无分支的语句或源程序语句

l 边和点圈定的部分叫做区域。当对区域计数时，图形外的一个部分也应记为一个区域

l 判断语句中的条件为复合条件时，即条件表达式由一个或多个逻辑运算符连接的逻辑表达式（a and b），则需要改变复合条件的判断为一系列只有单个条件的嵌套的判断

图形符号图所示

（3） 程序环路复杂性

l 程序的环路复杂性即 McCabe 复杂性度量，简单的定义为控制流图的区域数

l 从程序的环路复杂性可导出程序基本路径集合中的独立路径条数，这是确保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

l 独立路径：包括一组以前没有处理的语句或条件的一条路径

l 通常环路复杂性可用以下三种方法求得：

v 将环路复杂性定义为控制流图中的区域数。

v 设 E 为控制流图的边数，N 为图的结点数，则定义环路复杂性为 V(G)＝E－N＋2。

v 若设 P 为控制流图中的判定结点数，则有 V(G)＝P＋1。

（4） 基本路径测试步骤

l 以详细设计或源代码为基础，导出程序的控制流图

l 计算得到控制流图 G 的环路复杂性 v(g)

l 确定线性无关的路径的基本集

l 生成测试用例，确保基本路径集中每条路径的执行

五． 其他白盒测试方法

1． 域测试

（1） 概述

是一种基于程序结构的测试方法，基于对程序输入空间（域）的分析，选择适的测试点进行测试

（2） Howden 错误分类－相对于程序路径分类

l 域错误：程序的控制流存在错误，对于某一特定的输入可能执行的是一条错误路径，这种

错误称为路径错误，也叫做域错误
l 计算型错误：对于特定输入执行的路径正确，但赋值语句的错误导致输出结果错误，称为计算型错误
l 丢失路径错误：由于程序中的某处少了一个判定谓词而引起的
（3） 测试理想结果：检验输入空间的每一个输入元素是否都产生正确的结果
（4） 缺点
l 为进行域测试对程序提出的限制过多
l 当程序存在很多路径时，所需的测试点很多
2． 符号测试
（1） 概述
l 基本思想是允许程序的输入不仅仅是具体的数值数据，而且包括符号值，符号值可以是基本的符号变量值，也可以是符号变量值的表达式。
l 符号测试执行的是代数运算，可以作为普通测试的一个扩充
l 符号测试可以看作是程序测试和程序验证的一个折衷办法
（2） 测试理想情况：程序中仅有有限的几条执行路径，如果都完成了符号测试，就可把握的确认程序的正确性了
（3） 缺点
l 分支问题
l 二义性问题
l 大程序问题
3． Z 路径覆盖
（1） 概述
对循环机制进行简化，减少路径的数量，使得覆盖所有路径成为可能，简化循环意义下的路径覆盖称为 Z 路径覆盖
（2） 循环简化：限制循环次数，只考虑循环一次或零次情况
4． 程序变异
（1） 概述
是一种错误驱动测试
错误驱动测试：指该方法是针对某类特定程序错误的，即专门测试某类错误是否存在
错误驱动测试分类：程序强变异和程序弱变异
（2） 优点：便于集中目标对软件危害最大的可能错误，提高测试效率，降低成本
六． 白盒测试综合策略
1． 白盒测试中测试方法的选择策略
（1） 在测试中，首先尽量使用测试工作进行静态结构分析
（2） 采用先静态后动态的组合方式，先进行静态结构分析，代码检查和静态质量度量，然后现进行覆盖测试
（3） 利用静态结构分析的结果，通过代码检查和动态测试的方法对结果进一步确认，使测试工作更为有效
（4） 覆盖率测试是白盒测试的重点，使用基本路径测试达到语句覆盖标准；对于重点模块，应使用多种覆盖标准衡量代码的覆盖率
（5） 不同测试阶段，侧重点不同
l 单元测试：以代码检查、逻辑覆盖
l 集成测试：增加静构结构分析、静态质量度量
l 系统测试：根据黑盒测试结果，采用白盒测试

2． 最少测试用例数计算

l 将构成循环操作的重复型结构用选择结构代替，因此在 N-S 图中只存在顺序和分支操作

l N-S 图按分支结构分层，整个程序的最少测试用例数为每个分层中最少测试用例数的乘积

3． 测试覆盖标准

l Foster 的 ESTCA 覆盖标准

l Woodward 等人的层次 LCSAJ 覆盖标