**第一章: 白盒测试与代码质量**

**1. 白盒测试的相关概念:**

1) 定义:白盒测试又称为结构测试,透明盒子的测试,逻辑驱动测试,

基于代码的测试

2) 白盒测试是一种测试方法,是一种测试用例设计的方法.

3) 白盒测试主要针对被测程序的源代码,主要用于软件的验证,不考虑软件的功能实现,只验证内部的动作是否按照软件的设计说明书的规定来进行.

4) 测试用例由测试输入数据和输出结果组成.测试用例设计的好坏直接决定测试的效果和结果.

5) 黑盒测试用来检测程序每个功能是否能正常使用,它只检查程序的功能是否按照需求规格说明书的规定正常使用,程序是否能适当地接收输入的数据而产生正确的输出信息.主要针对软件的界面和软件的功能进行测试.

6) 为什么要做白盒测试:

1.黑盒测试无法对程序代码进行测试.

2.白盒测试可以发现黑盒测试中没有发现的问题,比如:

代码中存在的一些内存泄露,短时间黑盒发现不了;

有些分支,路径可能没有测试到,而这些分支和路径可能存在缺陷.

有些异常和错误情况在功能测试中很难模拟,需要通过白盒测试分析程序的源代码来找出错误.

7) 单元测试(测试过程) 单元:一个函数/过程/模块/一段代码/一个类

8) 白盒测试主要用在编码阶段,单元测试阶段,集成测试阶段,提高代码的质量.

9) 黑盒测试主要用在集成测试,系统测试的阶段.

10)灰盒主要用在集成测试阶段.

11)白盒测试必须遵守以下几条原则:

1.保证一个模块中的所有独立路径至少被执行一次.

2.所有逻辑值均需测试真和假两种情况.

3.检查程序的内部数据结构,保证其结构的有效性(变量,属性,方法)

4.在上,下边界以及可操作范围内运行所有循环.

12)根据测试程序是否运行,白盒测试分为:静态白盒测试和动态白盒测试.

1.静态黑盒测试(文档测试->) 2.动态黑盒测试(对软件进行测试,功能测试)

3.静态白盒测试:桌面检查,代码复查,代码走查

4.动态白盒测试:写测试代码进行测试(驱动程序/桩程序)

**2. 白盒测试的意义及相关内容:**

1.单元测试阶段:发现逻辑,边界条件,变量未初始化,内存越界等问题;

2.集成测试阶段:发现接口错误,任务配合等问题;

3.系统测试阶段:发现业务流程,界面操作(需求说明书)等问题.

1) 白盒测试需要开发部门提供的材料:

软件源代码,用户文档,规格需求说明书,系统软硬件环境,概要和详细设计.

2) 常用的测试用例设计技术:

逻辑覆盖测试->循环覆盖测试->基本路径测试->数据流分析->程序插桩

3) 白盒测试结果:统计信息->单元测试报告/质量报告(使用工具)

4) 正确高效的使用白盒测试工具.(自动化)

5) 白盒测试要求测试员了解编程技术/语言和方法.

**4. 代码质量**

1) 衡量软件质量指标:缺陷密度 顾客满意度 可靠性

**3. 动态白盒测试常用测试方法:逻辑覆盖法和循环测试法.**

1) 常用逻辑覆盖标准:语句覆盖,判定覆盖,条件覆盖,判定-条件覆盖,

条件组合覆盖,路径覆盖. –白盒测试6大法.

1.语句覆盖:选择足够多的测试用例.使得程序中的每个语句至少被执行一次.

2.判定覆盖:选择足够多的测试用例.使得程序中每个判定的真和假都执行一次.

满足判定覆盖就一定满足语句覆盖,满足语句覆盖不一定满足判定覆盖.

3.条件覆盖:选择足够多的测试用例.使得程序中每个判定的条件的真和假都执行一次. 满足条件判定,不一定满足语句或判定覆盖

if(条件1 ||条件2) -> 假 真 真 真 假 真

4.判定-条件覆盖:满足判定覆盖和条件覆盖

满足判定-条件覆盖,一定满足语句/判定/条件覆盖

5.条件组合覆盖: 选择足够多的测试用例.使得程序中每个条件的所有可能组合至少被执行一次.

满足条件组合覆盖一定满足判定-条件覆盖/语句/判定/条件覆盖

6.路径覆盖: 选择足够多的测试用例, 使得程序中所有路径至少被执行一次.

画出流程图,找出路径-> 赋值数据确保每条路径都走一遍

路径覆盖不一定满足条件组合/判定-条件/条件覆盖

路径覆盖一定满足判定覆盖和语句覆盖.

使用时选择顺序:条件组合>>路径覆盖>>判定-条件>>条件/判定>>语句覆盖

条件组合/条件覆盖和路径覆盖的关系:

**4. 循环覆盖测试(注重于循环构造的有效性)**

1)简单循环测试:

1.0次循环:从循环入口直接跳到循环出口

2.一次循环: 查找循环初始值方面的错误

3.两次循环:检查在多次循环才能暴露的错误

4.m次循环:检查在m次循环才能暴露的错误(m<n)

5.n-1,n次通过循环:检查在边界条件上和最大次数时可能出现的错误.

2)串接循环测试:1.如果这两个循环是相互独立的,就按照2个简单循环测试方法

2.如果两个循环存在依赖关系,就按照嵌套循环方法处理.

3)对于嵌套循环:1.从最内层循环开始,设置所有其他外层循环为最小值.

2.对内层循环做简单循环的全部测试,测试时保持外层循环的变量为最小值.

3.逐步外推,对其外面一层循环进行测试,测试时保持外层循环的变量为最小值

所有其它嵌套内层循环的循环变量取”典型”值

4.反复进行,直到所有各层循环测试完毕.

5.对全部各层循环同时取最小循环次数或者同时取最大循环次数.

4)对于不规则循环(非结构的循环):goto语句

首先重新设计循环结构,使他成为其他循环方式,再进行测试.

**5. 基本路径覆盖法:简化路径数量的测试方法**

1)4个步骤:1.分析程序,画出程序控制流图; 2.计算出圈复杂度(环复杂度)

3.导出独立路径; 4.设计测试用例

2)流图:圆+箭头 ->圆:流图的节点,代表一个条件(代表一个或多个语句);

箭头:流图的边,代表控制流; 边和节点圈定的范围,叫做区域

3)常见的流图的结构:顺序结构,if结构,case结构,while结构,until结构(do while)

**1**

**2**

**3**

**4**

**5**

**6**

**7**

**8**

4)环形复杂度:三种方法1.V(G)=控制流图中区域的数量

2.V(G) =E(边)-N(节点)+2 3.V(G)=P(判定节点)+1

5)独立路径数量等于圈复杂度: (原则,每次一次路径不同)1-2-3-4-6-7-8

1-2-5-6-7-8 / 1-2-3-5-6-8 /1-2-5-6-8

最小原则:1-2-5-6-8 / 1-2-5-6-7-8 / 1-2-3-5-6-8 / 1-2-3-4-6-7-8

6)图形矩阵:是在基本路径测试中起辅助作用的软件工具,利用它可以实现自动地确定一个基本路径集合.

根据程序流程图,画出图形矩阵:

1 2 3 4 5 6 7 8

1 1

2 1 1

3 1 1

4 1

5 1

6 1 1

7 1

8

int a[][] =new int [8] [8]

a[0][1]=1; a[1][2]=1; a[2][3]=1; a[3][5]=0; a[5][6]=1; a[6][7]=1

有两个1的行算1个判定节点;有多少个1表示多少个边;多少行表示多少节点

连接权值:最简单的情况下,连接权值是1,用来代表存在连接,是0表示不存在

连接,可以给连接权值赋予更多的属性信息,比如:执行这条边的概率,执行这条

边所花的时间等.