第十章--软件测试

白盒测试

白盒测试把被测软件看作一个透明的白盒子， 测试人员可以完全了解软件的设计或代码， 按照软件内部逻辑进行测试。

白盒测试又称玻璃盒测试。

1)语句覆盖法

2)判定覆盖(分支)

3)条件覆盖

4)判定/条件覆盖

5)条件组合覆盖

6)路径覆盖

黑盒测试

- 黑盒测试把程序看成一个黑盒子，完全不考 虑程序内部结构和处理过程。

  *黑盒测试是在程序接口进行测试，它只是检 查程序功能是否按照需求规约正常使用。

- 黑盒测试又称功能测试、行为测试，在软件开发后期执行。

边界值划分

边界值划分法使被测程序在边界值及其附近运行， 从而更有效地暴露程序中潜藏的错误。不仅根据输入条件，它还根据输出情况设计测试用例

例： 重量在10公斤至50公斤范围内的邮件，其邮费计算公式为 ……
应取10及50，还应取10.01, 49.99, 9.99, 50.01等

等价类划分

- 试遍所有输入数据是不可能的

- 等价类划分的办法是把程序的输入域划分成若干部分，然后从每个部分中选取少数代表性数据当作测试用例。

- 输入的数据划分为合理等价类和不合理等价类

  步骤：
  (1)设计一个测试用例，使其尽可能多地覆盖有效等价类，重复这一步，最终使得所有有效等价类均被覆盖。

  (2)设计一个测试用例，使其只覆盖一个无效等价类，重复这一步，最终使得所有无效 等价类均被覆盖。

  例：每个学生可以选取修1至3门课程

  有效等价类:选修1至3门课程

  无效等价类:没选修课程以及超过3门课程

  错误推测法(error guessing)
  猜测被测程序在哪些地方容易出错，针对可能的薄弱环节来设计测试用例

  例子1:对一个排序程序，可测试:

  输入表为空

  输入表中只有一行

  输入表中所有的值具有相同的值

  输入表已经是排序的

  因果图法(Cause - Effect Graphics)
  When
  检查输入条件的各种组合情况

  在等价类划分方法和边界值方法中未考虑输入条件的各种组合，当输入条件较多时，输入条件组合的数目会相当大

How

分析需求规约，找出因(输入条件)和果(输出或程序状态的修改)

画出因果图

通过因果图功能说明转换成一张判定表，然后为判定表的每一例设计测试用例

例如，有一个处理单价为5角钱的饮料自动售货机 软件，其需求规约如下:
有一个处理单价为1元5角的盒装饮料的自动售货 机软件。

若投入1元5角硬币，按下“可乐”，“ 雪碧”或“红茶”按钮，相应的饮料就送出来。 若投入的是两元硬币，在送出饮料的同时退还5角 硬币。

测试层次

单元测试 (Unit testing)

单元测试(unit testing)，又称为模块测试，是针 对软件结构中独立的基本单元(如函数、子过程、类) 进行的测试。

模块接口

保证被测基本单元的信息能够正常地流入和流出，局部数据结构。

边界条件

在到达边界值的极限或受限处理的情形下仍能正确执行。

独立的路径

执行控制结构中的所有独立路径以确保基本单元中的所有语句至少执行一次。

错误处理路径

对所有的错误处理路径进行测试

集成测试 (Integration testing)

集成测试(integration testing)，又称组装测试， 它根据设计将软件模块组装起来，进行有序的、递增的 测试，并通过测试评价它们之间的交互。

集成测试重点关注:

在把各个软件单元连接起来的时候，穿越单元接口的数据是否会丢失;

一个软件单元的功能是否会对另一个软件单元的功能产生不利的影响;

各个子功能组合起来，能否达到预期要求的父功能;

全局数据结构是否有问题;

单个软件单元的误差累积起来，是否会放大，从而达到不能接受的程度。

系统测试 (System testing)

功能性测试(functionality testing)
又称为正确性测试或 一致性测试，其目的是用以确认软件在指定条件下使用时， 软件产品提供满足明确和隐含要求的功能的能力。

适用性测试
测试软件为指定的任务和用户目标提供一 组合适的功能的能力。

准确性测试
测试软件提供具有所需精度的正确或相符 的结果或效果的能力。

互操作测试
测试软件与一个或更多的规定系统进行交 互的能力。

安全性测试
测试软件保护信息和数据的能力，以使未 授权的人员或系统不能阅读或修改这些信息和数据，而 不拒绝授权人员或系统对它们的访问。

功能依从性测试

测试规约中所有的功能都应实现，而 且应是正确的。

软件可靠性测试(reliability testing)

用以测试在故 障发生时，软件产品维持规定的绩效级别的能力。

成熟性测试

测试软件为避免由软件中故障而导致失效 的能力。

容错性测试

测试在软件出现故障或者违反其指定接口 的情况下，软件维持规定的性能级别的能力。

易恢复性测试

测试在失效发生的情况下，软件重建规 定的性能级别并恢复受直接影响的数据的能力。

可靠性的依从性测试

测试软件遵循与可靠性相关的规 约、标准或法规的能力。

性能测试(performance testing)

用来测试软件在规定条件下，相对于所用资源的数量，可提供适当性能的能力。

时间特性测试

测试在规定条件下，软件执行其功能时，提 供适当的响应和处理时间以及吞吐率的能力。

资源利用性测试

测试在规定条件下，软件执行其功能时， 使用合适数量和类别的资源的能力。这些资源包括CPU、内存 、网络等。

性能依从性测试:测试软件遵循性能相关的规约、标准或法 规的能力。

压力测试(stress testing)

又称强度测试，是一种超常情况 下的性能测试。它需要在超常数量、频率或资源的方式下执行 系统，以获得系统对非正常情况下(如大数据量的输入、处理 和输出，大并发数等)的承受程度。

自动化的性能/压力测试

HP的Loadrunner，开源的Jmeter等。

可移植性测试

用以测试软件 从一种环境迁移到另外一种环境的能力。

适应性测试

测试软件毋需采用额外的活动或手段就可 适应不同指定环境的能力。

易安装性测试

测试软件在指定环境中被安装的能力。

共存性测试
测试软件在公共环境中同与其分享公共资源的其他独立软件共存的能力。

易替换性测试
测试软件在同样环境下，替代另一个相同用途的指定软件产品的能力。
可移植性的依从性测试
测试软件遵循可移植性相关的规约、标准或法规的能力。

可维护性测试(maintainability testing)
用以测试软 件可被修改的能力，包括纠正、改进或软件对环境、 需求变化的适应。

易分析性测试
测试软件诊断缺陷或失效原因或识别待修改 部分的能力。
易改变性测试
测试软件使指定的修改可以被实现的能力。
稳定性测试
测试软件避免由于软件修改而造成意外结果的
能力。
易测试性测试
测试软件使已修改软件能被确认的能力。
维护依从性测试
测试软件遵循可维护性相关的规约、标准或法规的能力。

α测试与β测试

α测试是邀请小规模的、有代表性的潜在用户，在开发环境中，由开发者"指导"下进行的测试(试用)，开发者负责记录使用中出现的问题和软件的缺陷，因此α测试是在一个受控的环境中进行的。

B测试是由用户在一个或多个用户环境下进行的测试， 是产品正式发布前的系统测试形式。一组有代表性的用 户和消费者在典型操作条件下尝试做常规使用，由用户 记录下测试中发现的问题或任何希望改进的建议，报告 给开发者。

B测试与α测试不同的是，β测试时开发者通常不在测 试现场，由用户去使用，软件在一个开发者不能控制的环境中的"活的"试用。

回归测试(regression testing)

为了保证软件返工时没有引进新的错误，要全部或 部分地重复以前做过的测试。 在集成测试、缺陷纠正后的重新测试、迭代开发的 后续迭代测试中常常用到回归测试。

回归测试可以手工执行，也可以使用自动化的回归 测试工具(又称功能测试工具)。回归测试工具使 得软件工程师能够捕获到执行过的测试，然后进行 回放和比较。

回归测试应重新执行所有执行过的测试，或者对受 影响的软件部分进行局部回归测试。仅仅对修改的 软件部分进行重新测试常常不够的。

三角形测试用例

三角形测试用例类别

输入条件	有效等价类	无效等价类
是否是三角形	(A>0) (1) (B>0) (2) (C>0) (3) (A+B>C) (4) (B+C>A) (5) (C+A>B) (6)	(A<=0) (7) (B<=0) (8) (C<=0) (9) (A+B<=C) (10) (B+C<=A) (11) (C+A<=B) (12)
是否是等腰三角形	(A=B) (13) (B=C) (14) (C=A) (15)	(A!=B)and(B!=C)and(C!=A) (16)
是否是等腰直角三角形	(A=B)and(A^2+B^2=C^2) (17) (B=C)and(B^2+C^2=A^2) (18) (C=A)and(C^2+A^2=B^2) (19)	(A!=B)and(B!=C)and(C!=A) (20)
是否是等边三角形	(A=B)and(B=C)and(C=A) (21)	(A!=B) (22) (B!=C) (23) (C!=A) (24)

参数化测试demo

```
package com.company;
import static org.junit.Assert.assertEquals;
import org.junit.Test;
import java.util.Arrays;
import java.util.Collection;


import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import org.junit.runners.Parameterized.Parameters;
import static org.junit.Assert.*;
```

```
import static org.junit.Assert.assertEquals;
import java.util.Arrays;
import java.util.Collection;
import org.junit.Assert;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import org.junit.runners.Parameterized.Parameters;

/**
 * 参数化测试的类必须有Parameterized测试运行器修饰
 *
 */
@RunWith(Parameterized.class)
public class TriangleTest {
    private int param1;
    private int param2;
    private int param3;
    private String exp_type="";
    /**
     * 准备数据。数据的准备需要在一个方法中进行，该方法需要满足一定的要求：

     1）该方法必须由Parameters注解修饰
     2）该方法必须为public static的
     3）该方法必须返回Collection类型
     4）该方法的名字不做要求
     5）该方法没有参数
     * @return
     */
    @Parameters
    @SuppressWarnings("unchecked")
    public static Collection date(){
        //将所有测试数据放在object数组对象中
        //object数组对象可使用任意数据类型
//实际测试时 自己加入更多的测试数据
        Object[][] object={

                {-3,4,5,"边取值超出范围"},
```

```
                {3,-4,5,"边取值超出范围"},
                {3,4,-5,"边取值超出范围"},
                {-3,-4,-5,"边取值超出范围"}
                {0,1,3,"边取值超出范围"},
                {1,0,3,"边取值超出范围"},
                {1,3,0,"边取值超出范围"},
                {2,3,5,"不构成三角形"},
                {2,5,3,"不构成三角形"},
                {3,5,2,"不构成三角形"},
                {3,3,3,"等边三角形"},
                {5,3,3,"等腰三角形"},
                {5,8,8,"等腰三角形"}
                {3,4,5,"直角三角形"},
                {5,12,13,"直角三角形"}
                {3,4,6,"一般三角形"},
                {4,3,6,"一般三角形"},
                {3,6,4,"一般三角形"},

        };

        return Arrays.asList(object);
    }
    /*此构造函数不可缺少，并且参数要和参数结合中的类型及数量一致*/
    public TriangleTest(int param1,int  param2,int param3,String type){
        this.param1=param1;
        this.param2=param2;
        this.param3=param3;
        this.exp_type=type;
    }

    @Test
    public void testtype(){
        Triangle triangle=new Triangle(param1,param2,param3);
        String result_type=triangle.GetTriangleType();
        assertEquals(exp_type,result_type);
    }
}
/*public class TriangleTest {
```

```
    @Before
    public void setUp() throws Exception {
    }

    @After
    public void tearDown() throws Exception {
    }

    @Test
    public void getTriangleType() {
    }

    @Test
    public void main() {
    }
}*/



/**
 * 三角形问题实现
 * 给定三条边，判断三角形的形状
 *
 *
 */
package com.company;
import java.util.Scanner;

public class Triangle {
    private int side1;
    private int side2;
    private int side3;



    public Triangle(int a,int b,int c)
    {
        this.side1 = a;
        this.side2 = b;
```

```
        this.side3 = c;
    }

    /**
     * 获取三角形类型
     * @return
     * @throws Exception
     */
    public String GetTriangleType()
    {
        if(side1<1||side1>200||side2<1||side2>200||side3<1||side3>200)
        {
            System.out.println("边取值超出范围");
            return "边取值超出范围";
        }
        if(side1+side2<=side3||side1+side3<=side2||side2+side3<=side1)
        {
            System.out.println("不构成三角形");
            return"不构成三角形";
        }
        if(side1==side2&&side2==side3)//等边三角形
        {
            System.out.println("等边三角形");
            return "等边三角形";
        }
        else if(side1==side2||side2==side3||side1==side3)//等腰三角形
        {
            System.out.println("等腰三角形");
            return "等腰三角形";
        }
        else if(IsRtTriangle(side1,side2,side3)) {
            System.out.println("直角三角形");
            return "直角三角形";
        }
        else//一般三角形
        {
            System.out.println("一般三角形");
            return "一般三角形";
        }
```

```
    }

    /**
     * 判断是否为直角三角形
     * @param a
     * @param b
     * @param c
     * @return
     */
    private boolean IsRtTriangle(int a,int b,int c)
    {
        int a_2 = a*a;
        int b_2 = b*b;
        int c_2 = c*c;
        if(a_2+b_2==c_2||a_2+c_2==b_2||b_2+c_2==a_2)
            return true;
        return false;
    }

    /*public static void main(String[] args) {
        int a;
        int b;
        int c;
        System.out.println("请输入三角形的三边: ");
        Scanner scanner=new Scanner(System.in);
        a=scanner.nextInt();
        b=scanner.nextInt();
        c=scanner.nextInt();
       Triangle  triangle =new Triangle(a,b,c);
       triangle.GetTriangleType();

    }*/
}
```