# 功能测试

## 第一部分 黑盒测试方法

## 一、相关概念

黑盒测试又称为功能测试，是把测试对象当作看不见内部结构的黑盒。在完全不考虑程序内部结构的和处理过程的情况下，测试者仅依据程序功能的需求规范考虑确定测试用例和推测结果的正确性。

使用黑盒测试可以导出执行程序所有功能需求的输入条件集，实现功能覆盖。它要求通过设定一定的测试用例，使得每个需求的点都要被测到。

## 二、主要方法

## （一）、等价类划分法

**1、等价类划分的方法**

（1）按区间划分

（2）按数值划分

（3）按数值集合划分

（4）按限制条件或规划划分

（5）按处理方式划分

**划分等价类重要的是：**

**（1）每个子集内部所有的数据都是等价的**

**（2）子集之间互不相交**

**（3）所有子集的并集是整个输入域或输出域**

**2、划分等价类的经验原则**

（1）输入条件的**取值范围**，可以划分出一个有效等价类和两个无效等价类 。

（2）如果输入条件规定了输入值的集合，或者是规定了“必须如何”的条件，这时可确立一个有效等价类和一个无效等价类。

（3） 如果输入条件是一个布尔量，则可以确定一个有效等价类和一个无效等价类。

（4） 如果规定了输入数据的一组值（假设N个），而且程序要对每个输入值分别进行处理。

（5）如果规定了输入数据必须遵守的规则，则可确立一个有效等价类(符合规则)和若干个无效等价类(从不同角度违反规则)。

（6）在确知已划分的等价类中，各元素在程序处理中的方式不同的情况下，则应再将该等价类进一步地划分为更小的等价类。

**补充：什么是有效等价类？什么是无效等价类？**

**【有效等价类】**是相对于规格说明**合理的**、**正确的**、**有意义的输入数据**构成的集合。

【无效等价类】是相对于规格说明不合理的、错误的、无意义的输入数据构成的集合。

**举个例子：**

每个学生可选修1-3门课程 ，那么：

（1）可以划分一个有效等价类：选修1-3门课程。

（2）可以划分两个无效等价类：未选修课；选修课超过3门。

**3、划分等价类步骤**

**（1）确立测试用例**

在确立了等价类之后，建立等价类表，列出所有划分出的等价类

**（2）**从已列出的等价类表中按以下原则**选择测试用例**

a.为每个等价类规定一个唯一的编号

b.设计一个新的测试用例，使其尽可能多地覆盖尚未覆盖的有效等价类。

c.重复这一步，最后使得所有有效等价类均被测试用例所覆盖

d.设计一个新的测试用例，使其只覆盖一个无效等价类。重复这一步使所有无效等价类均被覆盖

**（3）创建等价类表。**

**（4）构建测试用例**

a.为每一个等价规定一个唯一编号。

b.使用测试案例尽可能多的覆盖有效等价类。

c.使用单独的一个测试案例覆盖单独的一个无效等价类。

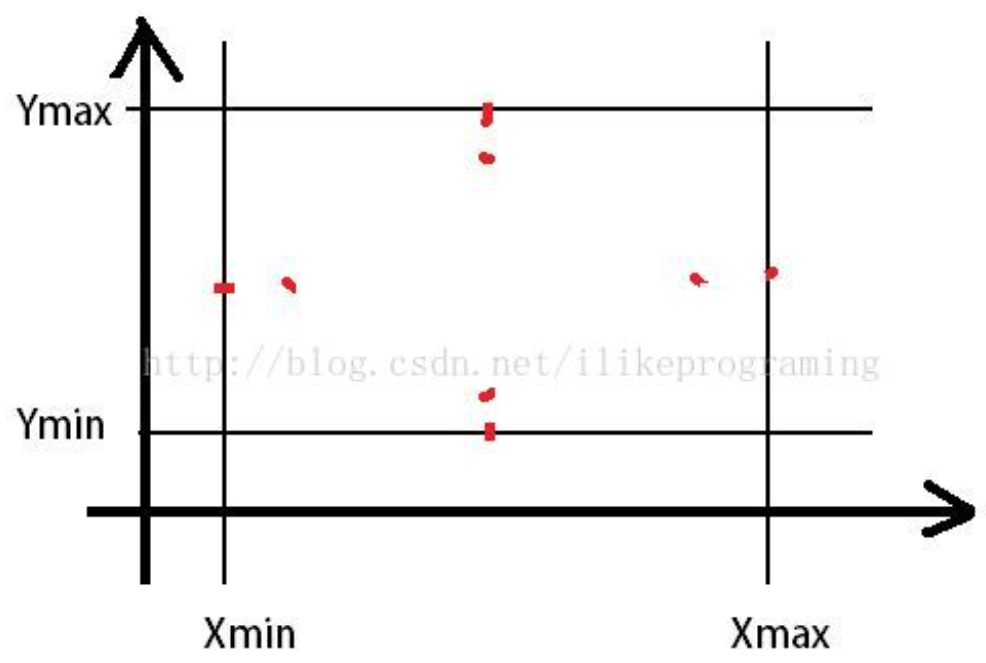
d.最后，直到所有的有效等价类和无效等价类均被覆盖。

## （二）、边界值分析法

（1）例如只有**一个输入条件**时，可以这么选取测试用例。（以坐标轴举例。以红点表示测试用例）



（2）例如当有**两个输入条件**的时候，可以这么选取测试用例。（以红点表示测试用例）



补充：

倘若要测试软件**健壮性**的时候，应选取**略小于最小值**的无效测试数据或者**略大于最大值**的无效测试数据进行测试。

**边界值分析原则：**

（1）如果输入条件规定了**值的范围**，则应取刚达到这个范围的**边界值**，以及刚刚超越这个范围的边界值作为测试的输入数据。

（2）如果输入条件规定了**值的个数**，则用**最大个数**，**最小个数**，比最小个数少一，比最大个数多一的数作为册数数据。

（3）根据规格说明的每个输出条件，使用原则（1）

（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**，则应选取集合的**第一个元素**和**最后一个元素**作为测试用例。

（5）分析规格说明，找出其他可能的边界条件。

## （三）、因果图

**1、基于因果图的测试方法要考虑如下问题：**

（1）规格说明书有哪些**原因**？

（2）规格说明书有哪些**结果**？

（3）规格说明书中各种**原因**之间的**关系**怎么样？

（4）规格说明书中各种**结果**之间的**关系**怎么样？

（5）规格说明书中**原因和结果**之间的**约束条件**怎么样？

（6）如何从规格说明书中的原因和结果**设计测试用例**？

**2、因果图**

|  |  |
| --- | --- |
| **因果图** | **描述** |
|  | 恒等: 若c1为1，则e1也为1。若c1为0，则e1也为0； |
|  | 非: 若c1是1，则e1是0.若c1为0，则e1是1； |
|  | 或: 若c1与c2中有一个是1或者两个都为1,则e1是1。若c1和c2都为0，则e1是0； |
|  | 与: 当且仅当c1和c2都是1,则e1为1,否则e1为0. |
|  | E约束(异；异或): a,b最多有一个可能为1,不能同时为1. |
|  | 约束(或；包含): a,b,c中至少有一个必须为1,不能同时为0. |
|  | O约束(惟一): a和b必须有一个且仅有一个为1 |
|  | R约束(要求):a是1时,b必须是1,即a为1时,b不能为0 |
|  | M约束:对输出条件的约束,若结果a为1,则结果b必须为0. |

## （四）、决策表

**1、决策表的建立步骤**

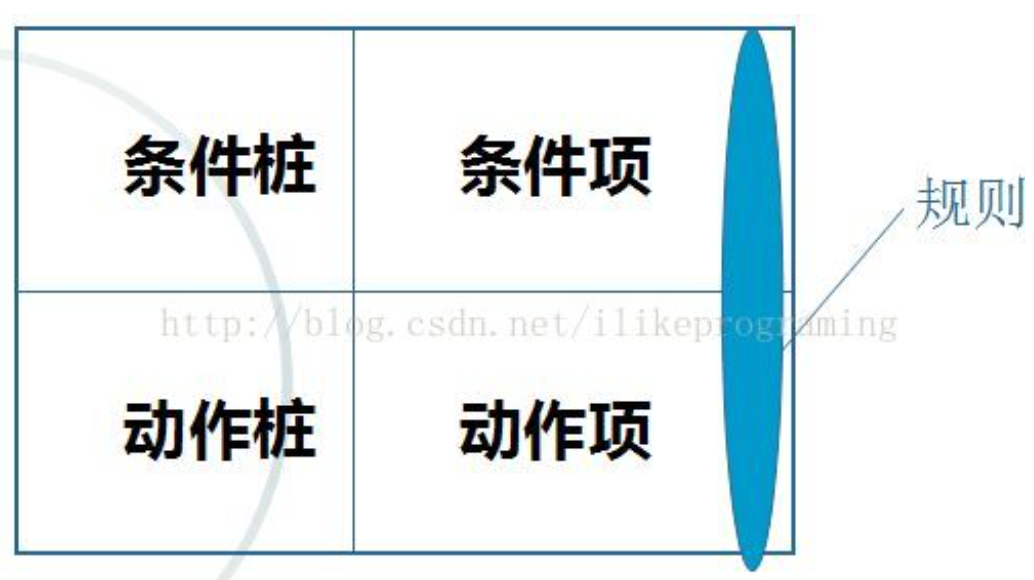
1、列出所有的条件桩和动作桩；

2、填入条件项；

3、填入动作项，制定初始判定表；

4、简化；合并相似规则或者相同动作。

**2、决策表模型**



条件桩：列出了问题的所有条件。

动作桩：列出了问题规定可能采取的操作。

条件项：列出针对它所列条件的取值，在所有可能情况下的真假值。

动作项：列出在条件项的各种取值情况下应该采取的动作。

规则：由动作项和条件项组成。

## （五）、场景法

**1、场景法的基本设计步骤**

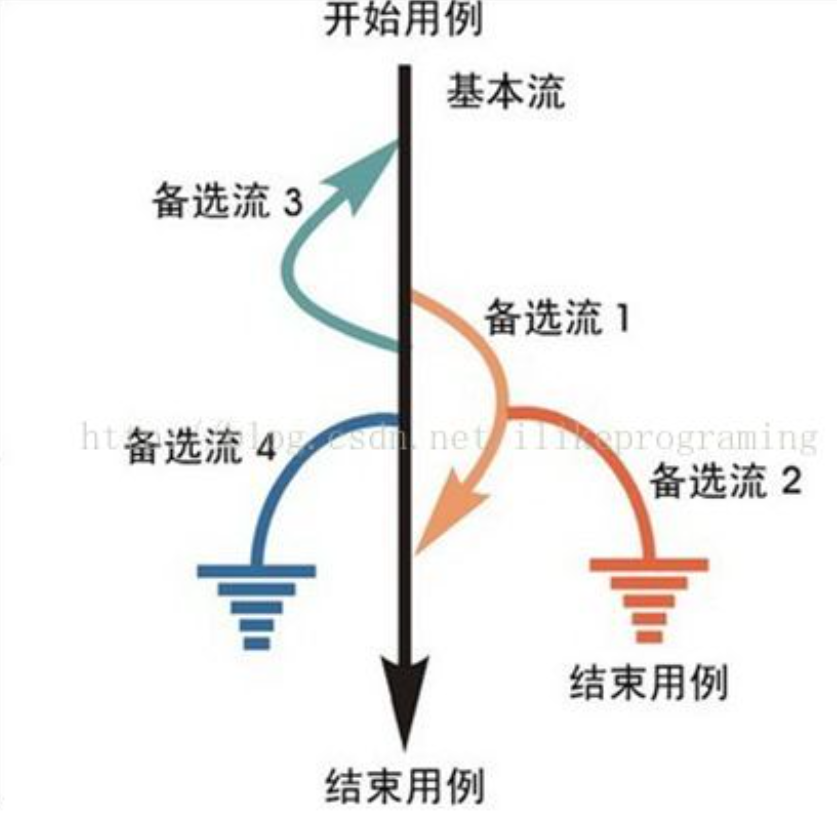
（1）根据说明，描述出程序的**基本流**及各项**备选流**

（2）根据基本流和各项备选流生成不同的**场景**

（3）对每一个场景生成相应的**测试用例**

（4） 对生成的所有测试用例重新**复审**，去掉多余的测试用例，测试用例确定后，对每一个测试用例确定测试数据值

**2、示意图**



**“场景”=“基本流”+“备选流**

## 第二部分 白盒测试方法

## 一、相关概念

白盒测试是指基于一个应用代码的内部逻辑知识，即覆盖全部代码、分支、路径、条件，使用程序设计的控制结构导出测试用例。

**白盒测试方法必须遵循以下4条原则：**

（1）保证一个模块中的**所有独立路径至少被测试一次**；

（2）所有逻辑值均需要**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；

（4）在上、下边界及**可操作**范围内运行**所有循环**。

## 二、白盒测试在不同测试阶段的侧重点：

（1）单元测试：代码检查、逻辑覆盖

（2）集成测试：增加静态结构分析、静态质量度量

（3）系统测试：根据黑盒测试结果，采用白盒测试。

## 三、白盒测试的方法

白盒测试的优点是帮助软件测试人员增大代码的覆盖率、提高代码的质量、发现代码中隐藏的问题。白盒测试主要用于检查程序的内部结构、逻辑、循环和路径。常用的测试用例设计方法如下：

|  |  |
| --- | --- |
| **方法** | **描述** |
| 代码检查法 | 主要通过桌面检查、走查和代码审查的方式进行 |
| 静态结构分析法 | 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的方法。 |
| 静态质量度量法 | 测试者通过软件质量、质量度量和度量规则进行分析 |
| 逻辑覆盖法 | 通过对程序逻辑结构的遍历实现对程序的覆盖 |
| 基本路径测试法 | 在控制流图的基础上，通过分析控制结构的环路复杂度，导出执行路径的基本集，再从该基本集设计测试用例。 |
| 域测试方法 | 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，选择测试点进行测试。 |
| 符号测试法 | 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。 |
| Z路径覆盖法 | 它是对循环机制进行了简化，减少了路径的数量，使得覆盖所有路径成为可能 |
| 程序变异测试法 | 它是一种错误驱动测试，专门测试某类错误是否存在 |

## 第三部分 软件缺陷管理

1. **软件缺陷管理的目的**

（1）保证信息的一致性；

（2）保证缺陷得到有效的跟踪和解决，缩短沟通时间，解决问题更高效；

（3）获取正确的Bug信息，利于缺陷分析、产品度量，使项目情况可视化加强。

二、**软件缺陷管理的相关属性**

|  |  |
| --- | --- |
| **属性** | **描述** |
| 缺陷发现人 | 在提交缺陷的时候，测试人员一般是测试的发现人，便于统计分析测试人员的能力，方便公司进行绩效考核。 |
| 缺陷发现时间 | 缺陷发现时间是一个统计的计数点，或者数据点，便于企业负责人选择合适的产品发布时间。 |
| 缺陷的状态 | （1）New：缺陷的初始状态（发现问题，提交问题，提交问题后，这个缺陷就处于New的状态）  （2）Open：开发人员开始修改缺陷（测试人员提交问题，开发人员接受并开始修改问题）  （3）Fixed：开发人员修改缺陷完毕  （4）Closed：回归测试通过（测试人员进行回归测试，回归测试通过，该问题改为Close状态）  （5）Reopen：回归测试失败（测试人员进行回归测试，回归测试不通过，该问题改为Reopen状态）  （6）Postpone：推迟修改  （7）Rejected：开发人员认为不是程序问题，拒绝缺陷  （8）Duplicate：与已经提交的Defect重复  （9）Abandon：被Reject和Duplicate的Defect，测试人员确认后的确不是问题，将Defect置为此状态  **比较理想的缺陷流程：从new状态——>open——>fixed——>closed状态。** |
| 缺陷的严重程度 | 是站在用户的角度，指Bug出现后对用户和系统的影响程度，可以简单地将软件缺陷的严重性划分为4个等级：致命、严重、一般、提示。  （1）致命缺陷：例如软件的意外退出甚至操作系统崩溃，造成数据丢失。  （2）严重缺陷：系统无法满足基本的商业要求且没有便捷可用的工作区。性能、功能或使用方面严重不达标，例如由于单功能失效引起多个功能失效。  （3）一般缺陷：系统能够满足商业要求。有快捷方便的工作区可供使用。性能、功能或使用方面并不是严重不达标，例如软件单个功能失效。  （4）提示：微小修改，希望提出建议，最好能够修正，但不是必需的。在发布准确性或实用性方面不会产生重大影响 |
| 缺陷的优先级 | 是站在开发/项目的角度，综合权衡修改Bug的时间、成本、技术和风险，决定Bug修改的先后顺序。  （1）优先级0（Priority 0）  这类软件缺陷必须在24小时之内被解决  （2）优先级1（Priority 1）  这类软件缺陷必须修复然后才能发布产品或者才能达到用户体验所包含的最主要目标，需要在1—2天内修改  （3）优先级2（Priority 2）  软件缺陷应该在2—4天内被修复  （4）优先级3（Priority 3）  最好在一周内修改  （5）优先级4（Priority 4）  发布周期内修改或者不修改 |
| 缺陷的类型 | （1）从**质量特性**角度考虑有功能、性能、安全性、易用性、可靠性缺陷；  （2）从**功能性**角度考虑有：错误、遗漏、多余的、可优化的缺陷；  （3）从缺陷**产生的原因**考虑有：需求规格说明书SRS、设计问题、编码问题、需求变更、设计变更、配置问题、测试问题 |
| 缺陷的版本 | （1）发现缺陷的版本（必须说明）  （2）修改bug的版本  （3）回归测试的版本（一般是最新版本） |
| 缺陷的修改日期 | 是主要对开发人员进行考核的参数。 |

## 第四部分 测试用例

## 一、相关概念

（1）测试用例是测试执行的**指导**；

（2）是测试执行的**实体**，是测试方法、测试质量、测试覆盖率的重要依据和表现形式；

（3）是团队内部交流以及**交叉测试的依据**，便于测试工作的跟踪管理，包括测试执行的进度跟踪，测试质量的跟踪，以及测试人员的工作量的跟踪和考核；

（4）在测试执行工作开展前完成测试用例的编写，可以避免测试工作开展的盲目性；

（5）测试用例是说服用户相信产品质量的最佳依据，同时也可以提供给客户作为项目验收的依据。

## 二、参考模板

<http://blog.csdn.net/dml1220/article/details/44150807>

**参考资料：**

<http://blog.csdn.net/xiaohui_loveless/article/details/48956107>

<http://blog.csdn.net/ilikeprograming/article/details/9255267>

<http://blog.csdn.net/huangjing8203/article/details/72403748>

<http://blog.csdn.net/u013457642/article/details/41680853>

<http://blog.csdn.net/kerry0071/article/details/39181095>

<http://blog.csdn.net/zxxssdsd/article/details/49314203>

<http://blog.csdn.net/qq_33642117/article/details/54406625>

<http://blog.csdn.net/dml1220/article/details/44150807>