一、测试流程  
需求分析----测试需求----测试计划----测试方案----测试用例----执行测试----测试报告  
1. 测试用例  
测试用例:指导测试,是测试的依据。包括输入(数据、文件)、操作步骤、预期结果  
1.1 测试用例的定义  
通过设计输入数据,执行步骤,按此步骤产生的预期结果,它是指导测试进行的依据。  
1.2 测试用例的目的  
高效率地发现软件缺陷而精心设计的少量测试数据。  
1.3 测试用例的特性  
(1)有效性:测能够被使用、且被不同人员使用测试结果一致  
(2)可复用性:良好的测试用例具有重复使用  
(3)易组织性:分门别类地提供给测试人员参考和使用  
(4)可评估性:从测试管理的角度,测试用例的通过率和软件缺陷的数目是软件产品质量好坏的  
测试标准  
(5)可管理性:可以校验测试人员进度、工作量以及跟踪/管理测试人员工作效率的因素  
1.4 测试用例的编制要素  
用例编号、测试模块、用例标题(表达用例的用途)、用例级别、测试环境、测试输入  
、执行操作、预期结果  
1.5 测试用例的设计原则  
(1)保证测试用例的明确性  
(2)保证测试用例的代表性  
(3)保证测试用例的简洁性  
二、测试用例设计方法  
1.等价类  
1.1定义  
设计输入具有代表性的数据子集  
1.2分类  
分为有效等价类和无效等价类  
设计数据时要注意:无效等价类只能在一个用例中出现一次  
有效等价类在一个用例中可以出现多次  
测试用例分析最好用xmind  
按等价类给出数据最好用excel表格  
1.3输入框的关注点  
长度、类型、组成规则、是否为空、是否可重复、是否区分大小写、是否去前中后空格  
1.4完成一个页面的测试步骤  
划分功能点(动词、不可再分割、包括输入输出处理)  
独立功能展开需求分析  
(1)可见  
(2)不可见(网络、系统、权限、数据库)  
分析可见参数特点---划分等价类  
1.5.适用对象  
输入框  
页面上的多个输入框不存在逻辑关系  
1.6 缺点  
没有关注到边界的问题，没有关注到输入框的逻辑关系  
2.边界值  
为了规避程序员经常在边界出现错误的问题  
eg:  
[1,100] 上点 1,100 离点 0,101 内点 2,99  
(1,100) 上点 2,99 离点 1,100 内点 3,98  
3.判定表  
等价类不能解决多输入且输入之间不同组合产生不同结果的情形  
，此时可以考虑判定表  
判定表通常由4部分组成  
条件桩:列出问题的所有条件  
条件项:针对条件桩给出的条件列出所有可能取值  
动作桩:列出问题规定的可能采取的操作  
动作项:指出在条件项的各组取值下应采取的动作  
规则:将任何一个条件组合的特定取值以及相应要执行的动作称为一条规则  
如果需求为文字描述,将文字转换为流程图  
4.正交实验  
所有因子的值两两正交(两个因子的值在同一用例中出现)  
多个选项,都是正值,没有逻辑关系  
利用正交工具

5.场景法

基本流:从系统的某个初始状态开始,经一系列状态后到达终止状态的过程中最主要的一个业务流程。

备选流:以基本流为基础,在经过的每个判定节点处满足不同的触发条件而导致的其他事件流。

基本流和备注流的区别

|  |  |  |
| --- | --- | --- |
|  | 基本流 | 备注流 |
| 测试重要性 | 重要 | 次要 |
| 数目 | 1条 | 1条或多条 |
| 初始节点位置 | 系统初始状态 | 基本流或其他备选流 |
| 终止节点位置 | 系统终止状态 | 基本流或系统其他终止状态 |
| 是否是完整的业务流程 | 是 | 否，仅为业务流程的执行片段 |
| 能否构成场景 | 能 | 否,需要和基本流共同构成场景 |

一般步骤

构造基本流和备选流

根据基本流和备选流构造场景

根据场景设计测试用例

对每个测试用例补充必要的测试数据

6.状态迁移图

所有参数都是有效的，参数之间存在约束条件(功能之间的约束、状态之间的约束)

做法：

1. 先看懂状态矩阵
2. 先选中某一状态(能到达最多的状态)，将其能迁移到的状态依次列于下方
3. 从最左方状态开始，将其能迁移到的状态依次列于下方。如有重复，抹掉
4. 最左列不要有重复(最后一行不计
5. 如果还有没有覆盖到的状态，补齐(一般在第一行)

测试用例

深度优先迁移图

1. 自上而下全部状态(所有的状态迁移都是纵向的)
2. 从最左边开始，其所能达到的状态都是一个测试用例
3. 先从左边纵向依次向下达到所能达到的状态，然后再分别迁移到此状态下的横向状态，最后依次去掉前面的一个状态，直到达到本状态为止

7.因果图

参数之间存在逻辑关系，不同逻辑组合会输出不同结果

参数之间存在约束关系，输出结果不确定

因果符号

恒等：条件满足时，一定会输出结果

非：条件满足时，一定不会输出结果

原因符号

或：多个条件满足其中任意一个时，一定会输出结果

与：多个条件同时满足时，一定会输出结果

异：原因a,b只能有一个为真

结果符号

或: 原因a,b至少有一个为真

画因果图的步骤

1. 找出原因

2. 找出结果

3.找到中间状态

4.按照因果图的约束条件画因果图

优点

1.充分考虑了输入条件之间的组合，对组合情况覆盖充分

2.最终每个用例覆盖多种输入情况，有利于提高测试效率

3.设计过程中，对输入条件的约束关系做了考虑。避免了无效用例，用例的有效性高。

4.能够同时得出每个测试项目的预期输出

缺点

1. 当被测试特性输入较多时，判定表的规模会非常大

2. 输入之间的约束条件不能有效区分输入是否需要进行组合测试，会造成不需要组合测试的输入做了组合，从而产生冗余

8.其他方法

输出域:覆盖所有的输出，不管输入

输入域:覆盖所有的输入，不管输出

异常分析法:断电、断网的异常情况(重点关注)

错误猜测法:有经验的人对系统做出判断,找到可能出现错误的地方

探索性测试:根据自己的测试经验,尝试性测试系统功能

9.易混淆的测试用例方法之间的比较

因果图和判定表的比较

|  |  |  |
| --- | --- | --- |
|  | 因果图 | 判定表 |
| 有无中间状态 | 有 | 无 |
| 条件项是否存在制约关系 | 是 | 否 |

判定表和正交实验的比较

|  |  |  |
| --- | --- | --- |
|  | 判定表 | 正交实验 |
| 是否允许出现负值 | 是 | 否 |

场景法和状态迁移的比较

|  |  |  |
| --- | --- | --- |
|  | 场景法 | 状态迁移 |
| 流程是否有序 | 是 | 否 |