### 测试用例的设计与如何编写测试用例

**常见的开发模型：**

　　V模型、瀑布模型、敏捷开发模型、W模型

**软件生命周期：**

　　1、问题的定义及规划

　　2、[需求分析](http://www.51testing.com/html/26/javascript:;" \t "http://www.51testing.com/html/26/_self)

　　3、软件设计（明确怎么做！）

　　4、软件编码

　　5、[软件测试](http://www.51testing.com/html/26/javascript:;" \t "http://www.51testing.com/html/26/_self)

　　6、运行维护

**测试生命周期：**

[单元测试](http://www.51testing.com/html/26/javascript:;" \t "http://www.51testing.com/html/26/_self)：一般是开发完成时

　　集成测试：单元测试之后，单元之间接口是否正确，数据是否正常传递。比如说注册和充值两个功能是否能够连通。

[系统测试](http://www.51testing.com/html/26/javascript:;" \t "http://www.51testing.com/html/26/_self)：根据[测试用例](http://www.51testing.com/html/26/javascript:;" \t "http://www.51testing.com/html/26/_self)，进行完整的系统测试

　　验收测试：用户对软件进行验收

**软件测试阶段：**

　　单元、集成、系统、验收（正式验收、Alpha测试，Beta测试）

**软测方法：**

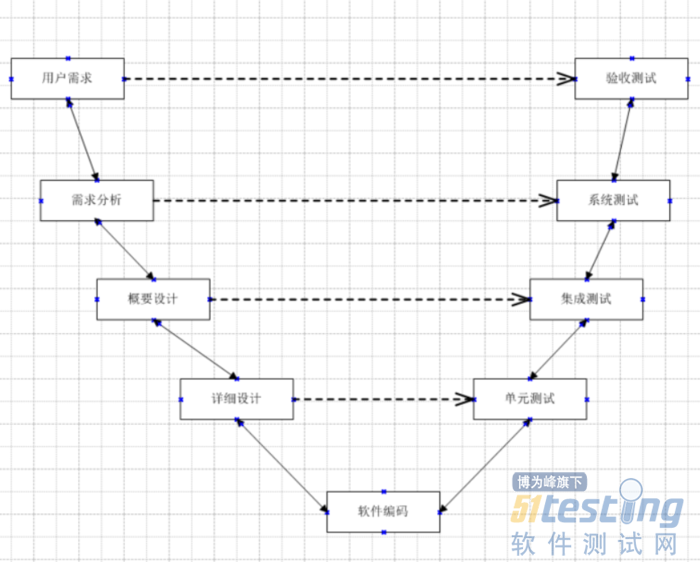
[白盒测试](http://www.51testing.com/html/26/javascript:;" \t "http://www.51testing.com/html/26/_self)、[黑盒测试](http://www.51testing.com/html/26/javascript:;" \t "http://www.51testing.com/html/26/_self)、灰盒测试

**软测类型：**

　　功能、界面、安全、兼容性、易用性、性能、压力、负载、恢复测试等

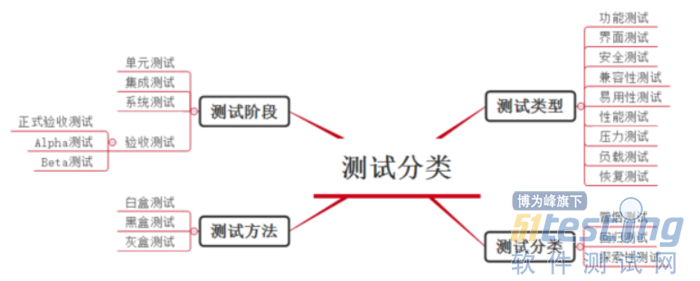
　　其他测试分类：冒烟测试、回归测试、探索性测试

**常用的开发的模型：V模型**

[](http://www.51testing.com/batch.download.php?aid=81374)

　　V模型

**软件测试的分类**

[](http://www.51testing.com/batch.download.php?aid=81375)

　　软测分类

**什么是黑盒测试？**

　　黑盒测试也称[功能测试](http://www.51testing.com/html/26/javascript:;" \t "http://www.51testing.com/html/26/_self)，它是通过测试来检测每个功能是否都能正常使用。不考虑内部结构，在程序接口进行测试。

**Alpha、Beta测试的区别？**

　　Alpha测试：前期的用户测试，公司内部在模拟实际操作环境下进行的一种验收测试。

　　Beta测试：后期的用户测试，此时已经通过内部测试，即将真实发布，是软件的在一个或者多个用户的实际使用环境下进行的测试

**冒烟测试和回归测试区别？**

　　冒烟测试：在新版本出来的时候，将软件的全部功能过一遍，功能可以正常进行不会影响测试进度，这个版本就可以真正测试了

　　回归测试：对以前版本中发现的bug在新的版本中验证是否存在且是否引发新的bug

**软测用例的设计方法**

**1、边界值：**

　　选取等于、刚刚大于、刚刚小于边界的值作为测试数据

　　基本思想是在最小值、略高于最小值、正常值、略低于最大值和最大值等处取值

**2、等价类划分：**

　　等价类划分就是把程序的输入域划分成若干部分，然后从每部分选取少量的具有代表性的数据作为测试用例。

　　无效等价类：不合理的、无意义的输入数据结婚，验证程序处理意外数据的能力

　　有效等价类：有意义的输入数据的集合，检验程序是否实现了规格说明总的功能和性能

　　等价类划分方法：按区间划分、数值划分、数值集合划分、限制条件和规则划分

**3、错误推算法：**

　　进行错误的操作，验证程序是否对出错的场景和情况有些应对能力，来选择测试用例数据

**4、因果法/判定表法：**

　　将判定表的每一列作为依据，设计测试用例。检查输入条件的各种组合情况

**5、场景法：**

　　通过描述的业务流程，设计用例来列出不同业务场景，作为测试用例的测试数据

　　基本流：主要是功能的正常操作流程

　　分支流：需要程序做非法判断处理的

**\*测试用例方法的选择\*（划重点）**

　　1、进行等价类划分，主要是输入条件的划分，这是提高测试效率最有效的方法

　　在任何情况下都必须使用边界值分析法，这种方法设计出测试用例发现程序错误的能力最强

　　2、用错误推测法追加测试用例

　　3、如果程序说明中含有输入组合情况，则一开始就用判定表法（判定表法很少用到）

　　4、如果还没有达到覆盖标准，应当再补充足够的测试用例（场景法）

**如何做软件测试需求分析？**

　　1、列出需求文档中的可测试性的原始需求

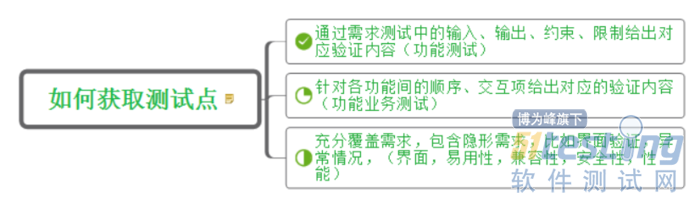
　　2、对每一条需求进行细化分解，形成可测试的测试点

　　3、针对测试点确定执行适合的测试类型

　　4、建立测试需求分析矩阵，对测试需求进行管理

　　软件测试需求的重点是“测什么”。

　　测试需求分析的目的：获取测试点，根据测试点编写用例

[](http://www.51testing.com/batch.download.php?aid=81376)

　　重要！！

**看到电梯，你可以写出它的测试点吗？**

　　按钮指示灯：按压上下按钮指示灯是否亮

　　电梯门开关：按压上下按钮电梯门在当前楼层是否能打开

　　按向上按钮：电梯是否关门且向上面楼层方向走

　　按向下按钮：电梯是否关门且向下面楼层方向走

　　当电梯门没有关上：按开电梯门按钮，门是否开

　　当电梯门没有关上：按关闭电梯门按钮，门是否关闭

　　电梯内：按各个楼层，对应的指示灯是否亮

　　电梯内报警装置：报警装置是否正常

　　电梯内通话设备：按通话按钮能否接通外界

　　电梯内灯光：电梯内灯光是否亮，是否有无损坏

　　电梯内通风：是否通风

　　按各个楼层按钮：是否到当前楼层停止并开门

　　当超过最高重量：电梯是否报警打开电梯门，直到小于最高承重

　　电梯当前楼层是否和电梯内显示屏楼层一直

　　显示屏内是否有当前楼层，当前向上或者向下箭头，且与当前操作一致

　　电梯门超过规定时间未关门是否会有报警提示

　　上下按钮是否控制一个电梯或者两个电梯的开关门，如果控制两个电梯，按向上或者向下按钮，另一个电梯是否受控制

　　电梯是否分单双层？

　　在单层电梯情况下，按双层电梯，对应双层电梯数字是否亮，是否会到这一层

　　在双层电梯情况下，按单层电梯，对应单层电梯数字是否亮，是否会到这一层

　　电梯限层：按超过限层的电梯层数，数字是否亮，是否会到这一层

　　双击某楼层：是否会取消这个楼层且楼层灯灭

　　假如我在9楼，有人先按12楼，有人后按1楼，此时电梯是否先上12楼，再下1楼？

　　电梯感应：有人或者物体在门中间卡着，门是否会关闭，是否会有警铃提示？

　　电梯到达指定楼层是否有声音提示？

　　电梯是否刷卡：刷卡的电梯，如果没有刷卡是否能选楼层

　　维修开关：电梯内是否有维修开关

**如何编写测试用例**

　　测试用例：指导性执行测试，帮助证明软件功能或发现软件缺陷的一种说明。每一个测试点的数据设计和步骤设计。

**测试用例的重要性：**

　　（1）、便于测试计划的实施

　　一般主要适用于集成测试、系统测试、回归测试。根据用例知道自己的进度

　　（2）、规划测试数据的准备

　　比如测注册，要提前准备好[手机](http://www.51testing.com/html/26/javascript:;" \t "http://www.51testing.com/html/26/_self)号、身份证号、不重复的用户名，邮箱等

　　（3）、编写测试脚本的根本

　　自动测试的中心任务是编写测试脚本。测试脚本就是以测试用例为基础。

　　（4）、评估测试结果的基准

　　通过测试用例的覆盖性和错误率，可以判断测试的结果，是否能发布

　　（5）、分析缺陷标准

　　收集缺陷，对比测试用例。分析是漏测还是缺陷复现。反应了测试的不完善，应立即补充相应的测试用例

　　\*测试标题如何写：测试点，对测试点进行细化分解。比如：输入正确用户名、密码，能否正常登陆。

**测试用例编写格式注意：**

　　（1）、测试标题一定要描述测试点（验证什么写什么），简洁明了，不存在重复

　　（2）、测试步骤要有指导性的意义，涉及测试数据输入最好包含具体的测试数据

　　（3）、预期结果是唯一的，不能出现“发送成功或失败”

**如何编写测试用例？**

　　用例包含：用例编号、功能模块、用例标题、前提条件、操作步骤、期望结果（含判断标准）、实际结果、备注

　　编写方式：按照功能+业务逻辑

　　（1）、首先保证单个功能是正常的

　　（2）、然后功能联合起来的业务逻辑是对的

　　比如：登录、充值、提现功能都是好的。业务逻辑，就是把所有的功能联合起来走一遍，看是否是好的

**用例覆盖：包含正面和反面的用例**

　　（1）、正面用例：根据功能模块划分，针对要测试的功能模块，所有正常输入数据的测试用例都写出来

　　（2）、反面用例：例如登录失败等，输入非法数据，违反唯一约束等等