4 测试用例设计方法

常用黑盒测试用例设计方法包括：等价类划分法 、边值分析法、因果图法、判定表法、状态迁移图法、流程分析法、正交实验法、异常分析法、错误猜测法等。

4.1 等价类划分法

对程序的穷举输入测试是无法实现的。因此，当测试某个程序时，我们就被限制在从所有可能的输入中努力找出某个小的子集。当然，找的这个子集必须是正确的，并且是可能发现最多错误的子集。

等价类划分设计方法是把所有可能的输入数据，即程序的输入域划分成若干部分(子集)，然后从每一个子集中选取少量具有代表性的数据作为测试用例。

- 等价类：某个输入域的集合，在这个集合中每个输入条件都是等效的，如果其中一个的输入不能导致问题发生，那么集合中其它输入条件进行测试也不可能发现错误
- 有效等价类：有效等价类是程序规格说明有意义，合理的输入数据
- 无效等价类：无效等价类是程序规格说明无意义，不合理的输入数据。

1. 等价类划分原则

1) 如果输入条件规定了取值范围或值的个数，则可以确定一个有效等价类和两个无效等价类；

例 1：某电商网站注册时，对于用户名长度的限制，必须大于等于 3 位，小于等于 14 位。

有效等价类	3<=用户名长度<=14
无效等价类	用户名长度<3 位，用户名长度>14 位

例 2：商品的价格设置限定必须在 0.00-999999.00 的范围内：

有效等价类	0.00-999999.00
无效等价类	小于 0.00，大于 999999.00

2) 输入条件规定了输入值的集合，或是规定了必须如何的条件，则可以确定一个有效等价类和一个无效等价类；

例 1：登录验证码必须为 XY12；

有效等价类	XY12
无效等价类	任意非 XY12 的集合

例 2：订单状态取值为“未支付、已支付、已发货、退货、已完成”；

有效等价类	“未支付、已支付、已发货、退货、已完成”中的任意值
无效等价类	不在“未支付、已支付、已发货、退货、已完成”中的任意值

3) 在输入条件是一个布尔量的情况下，可确定一个有效等价类和一个无效等价类。（布尔值，只有 true 和 false 两个值。也可以说真和假、有和无，有些地方也表示成 1 和 0）

例 1：某电商系统后台商店设置中，是否启用库存管理的选项，是和否。

有效等价类	选择“是”，也就是布尔值为 true

无效等价类	选择“否”，也就是布尔值为 false

例 2：电商后台商店设置中，对于商品显示的设置，是否显示货号的选项有：显示和不显示。

有效等价类	选择“显示”，也就是布尔值为 true
无效等价类	选择“不显示”，也就是布尔值为 false

4) 如果我们确知，已经划分的等价类中各个元素在程序中的处理方式不同的，则应该将此等价类进一步划分；

例 1：学校学生成绩评判等级的系统，1-100 的分数录入是有效的，但是系统程序会根据录入分数的不同评判出不同等级，例如 0-59.9 是不及格、60-89.9 是合格、90-100 是优秀；那会对等价类进一步划分。

例 2：某特价商品只能 VIP 会员购买；

有效等价类	VIP 会员
无效等价类	普通会员、非会员

然而 VIP 会员又分为黄金会员、白金会员、钻石会员，且价格根据会员等级又有不同的优惠，那么又需要对有效等价类再次进行划分：黄金会员、白金会员、钻石会员。

5) 在规定了输入数据必须遵守的规则的情况下，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）。

例 1：注册时用户名必须是字母加数字的组合；

有效等价类	字母加数字
无效等价类	1）仅字母，2）仅数字，3）特殊字符，4）空,5）已注册的用户名

例 2：某电商推出的优惠券，在指定的时间范围内购买限定的分类商品且总金额满 300 才能使用，一个账号只能领取一次该优惠券；

有效等价类	购买限定商品的总金额大于等于 300，且在规定时间内拥有领取未使用的优惠券
无效等价类	1）购买非限定的商品，2）金额小于 300，3）日期小于开始日 4）日期大于结束日期，5）未领取优惠券，6）优惠券已使用

2. 用例设计步骤

1. 为每个输入划分等价类，得到等价类表，为每个等价类规定一个唯一编号；

输入条件	有效等价类	无效等价类

2. 设计一个测试用例，使其尽可能多的覆盖所有尚未覆盖的有效等价类。重复这一步骤，使得有效等价类均被测试用例所覆盖；
3. 设计一个测试用例，使其只覆盖一个无效等价类。重复这一步骤使得所有无效等价类均被覆盖。

3. 实例分析

下面是电商项目的注册功能中，对于办公电话的具体说明。

办公电话格式要求：区号（3 或 4 位）-电话号码（7 或 8 位）形式。

- **条件分析**
  - 办公电话应该是数字
  - 第一部分是区号，要求 3 或者 4 位数字；
  - 第二部分是电话号码，要求是 7 或者 8 位数字。
- **等价类划分分析思路**

步骤一、根据电话号码是数字，根据等价类划分规则 2，确定一个有效等价类，和一个无效等价类。

条件	有效等价类	无效等价类
内容	数字	非数字

步骤二、根据要求，区号是 3 位或者 4 位，根据等价类划分规则 1，确定一个有效等价类和两个无效等价类。

条件	有效等价类	无效等价类
区号	区号位数是 3 位或 4 位	电话号码位数<3 位
		电话号码位数>4 位

步骤三、根据要求，电话号码是 7 位或者 8 位，根据等价类划分规则 1，确定一个有效等价类和两个无效等价类。

输入条件	有效等价类	无效等价类
电话号码	电话号码数是 7 位或 8 位	电话号码位数<7 位
		电话号码位数>8 位

最终设计的出等价类表如下：

输入条件	有效等价类	无效等价类
内容	(1) 输入数字；	(2) 输入非数字字符；
区号	(3) 3 位数字； (4) 4 位数字；	(5) 少于 3 位数字； (6) 多于 4 位数字。
电话号码	(7) 7 位数字； (8) 8 位数字；	(9) 少于 7 位数字； (10) 多于 8 位数字。

- **设计用例覆盖等价类**

序号	覆盖等价类	区号	电话号	预期结果
1	(1)（3）(7)	028	85412351	办公号码有效
2	(1)（3）(8)	010	2311218	办公号码有效
3	(1)（4）(7)	0816	2311218	办公号码有效
4	(1)（4）(8)	0816	82311218	办公号码有效
5	(2)（3）(7)	02a	8453212	办公号码无效

6	(2)（3）(8)	02!	84453212	办公号码无效
7	(2)（4）(7)	082b	8453212	办公号码无效
8	(2)（4）(8)	082b	88453212	办公号码无效
9	(1)（5）(7)	02	8453212	办公号码无效
10	(1)（5）(8)	01	84453212	办公号码无效
11	(1)（6）(7)	02123	8453212	办公号码无效
12	(1)（6）(8)	01214	84453212	办公号码无效
13	(1)（3）(9)	021	845321	办公号码无效
14	(1)（3）(10)	021	844532129	办公号码无效
15	(1)（4）(9)	0812	845321	办公号码无效
16	(1)（4）(10)	0821	844532129	办公号码无效

- **生成测试用例**

用例编号	功能模块	测试标题	前置条件	输入数据	操作步骤	期望结果	测试优先级
JX_ST_user_01	用户管理	注册_使用3位数字的区号、8位数字的电话号码	N/A	办公电话：028-25412351	1、在办公电话输入框输入测试数据	提示：“办公电话符合要求”	高
JX_ST_user_02	用户管理	注册_使用4位数字的区号、7位数字的电话号码	N/A	办公电话：0931-2635412	1、在办公电话输入框输入测试数据	提示：“办公电话符合要求”	高
JX_ST_user_03	用户管理	注册_使用非数字的区号、6位数字的电话号码	N/A	办公电话：09a-265413	1、在办公电话输入框输入测试数据	提示：“办公电话不合法”	中
JX_ST_user_04	用户管理	注册_使用4位非数字的区号、9位非数字的电话号码	N/A	办公电话：09v1-2@3c54!4	1、在办公电话输入框输入测试数据	提示：“办公电话不合法”	中

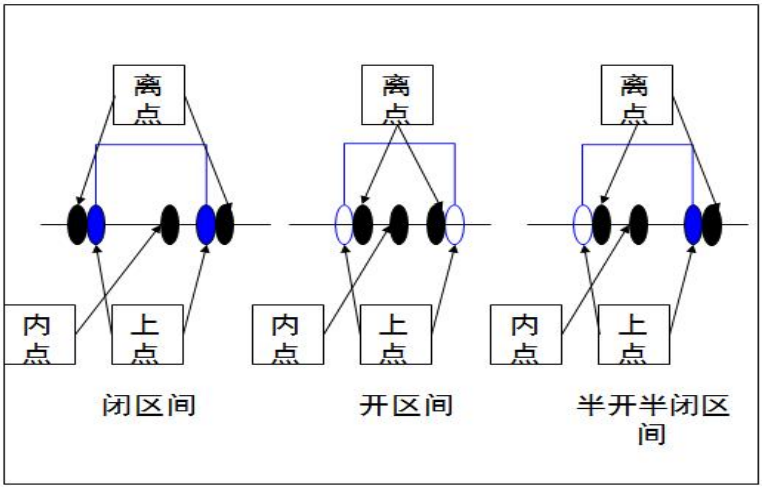
4.2 边值分析法

边值分析方法的理论基础，是假定大多数的错误是发生在各种输入条件的边界上，如果在边界附近的取值不会导致程序出错，那么其它的取值导致程序错误的可能性也很小。

输入条件明确了一个值的取值范围，或是规定了值的个数；

输入条件明确了一个有序集合。

1. **边值点定义**
   - 上点：边界上的点，如果域的边界是封闭的，上点就在域范围内；如果域的边界是开放的，上点就在域范围外；
   - 离点：就是离上点最近的一个点，如果域的边界是封闭的，离点就在域范围外，如果域的边界是开放的，离点就在域范围内；
   - 内点：顾名思义，就是在域范围内的任意一个点。



区间术语解释：

- 闭区间：表示范围的边界值包含在范围内，用方括号表示；比如[1, 10]，可以看成 1 ≤ x ≤ 10;
- 开区间：表示范围的边界值不包含在范围内，用圆括号表示；比如(1, 10)，可以看成 1 < x < 10;
- 半开半闭：表示范围的边界值一边包含在内，一边不包含在范围内，包括左开右闭、左闭右开；比如(1, 10]，可以看成 1 < x ≤ 10； [1, 10)，可以看成 1 ≤ x < 10。

边界值分析举例：

正整数值域[66,88]在不同开闭区间情况下上点、离点、内点取值情况：

取值范围	上点	离点	内点
[66,88]	66，88，都是在域内	65，89 都在域外	域内任意点，如 70
(66,88)	66，88，都是在域外	67，87 都在域内	域内任意点，如 70
(66,88]	66，88，其中 66 是域外，88 是域内	67，89，恰好与上点相反，67 域内，89 域外	域内任意点，如 70
[66,88)	66，88，其中 66 是域内，88 是域外	65，87，恰好与上点相反，65 域外，87 域内	域内任意点，如 70

小技巧：

上点毫不犹豫取范围边界上的点，根据上点是否在范围内确定离点的值。上点在范围外则离点就在范围内距上点最近的点；上点在范围内，则离点就在范围外距上点最近的点，对于离点的取值可以采用**开内闭外**的方法进行选取，即**开区间在范围内，闭区间在范围外**。

2. 边界值分析的原则

1) 如果输入（输出）条件规定了取值范围，或是规定了值的个数，则应该以该范围的边界内及边界附近的值作为测试用例；

例 1：某月份输入框，月份范围为 1~12 月，那么可获得的边界值为 0,1,12 和 13。

例 2：如果程序的规格说明中规定："重量在 10 公斤至 50 公斤范围内的邮件，其邮费计算公式为……"。我们应取 10 及 50，还应取 9.99 及 50.01 等。

以上两个例子，到底是采用加减 1 还是加减 0.01，主要取决于输入框能否输入小数点；如果输入框能输入小数点，则边界加减 0.01 更为有效。

2) 如果输入（输出）条件规定了值的个数的取值范围，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测试数据；

例 1：电商系统需求中，对于商品名称输入框，限制只能输入 50 个字符，那么应根据 0,1,50 和 51 个字符设计用例；

例 2：一个输入文件应包括 1~255 个记录，则测试用例可取 1 和 255，还应取 0 及 256 等。

3) 如果程序规格说明中提到的输入或输出是一个有序的集合，应该注意选取有序集合的第一个和最后一个元素作为测试用例；（数字、字母，月份、星期、时间）

例：某日期下拉菜单，提供了周一至周日的选项，那么就应该至少选择周一、周日两条数据作为用例。

4) 如果程序中使用了一个内部数据结构，则应当选择这个内部数据结构的边界上的值作为测试用例。

例 1：某添加功能从数据库查询了一组数据并根据某表主键排序供用户选择，那么要根据查询出来的数据，取第一条和最后一条作为用例；

例 2：比如常见编程语言中 int 型（整数类型）的数据，最大的取值范围为 -2147483648-2147483647。那么如果某个输入框取值设定为 int 型，那么接收的最大数字就是 2147483648。

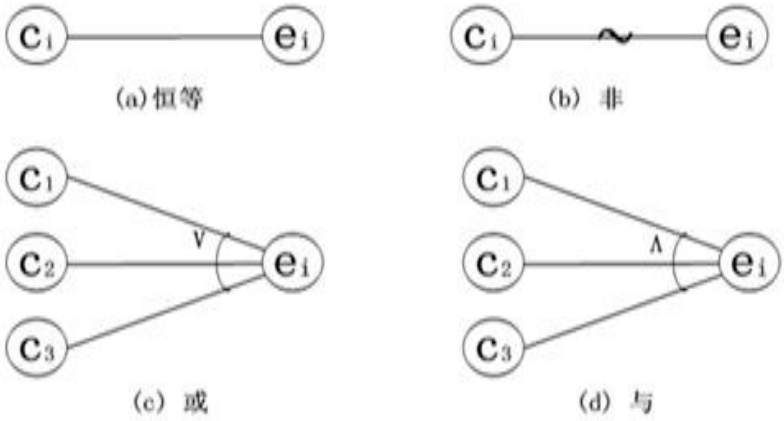
3. 用例设计的步骤

- 分析输入参数的类型：从测试规格中分析得到输入参数类型；
- 等价类划分（可选）：对于输入等价类划分方法进行等价类的划分；
- 确定边界：运用域测试分析方法确定域范围的边界（上点、离点与内点）；
- 相关性分析（可选）：如果存在多个输入域，则需要运用因果图、判定表方法对这些输入域边界值的组合情况进行进一步分析；
- 形成测试项：选择这些上点、离点与内点或者这些点的组合形成测试项。

4.3 因果图法

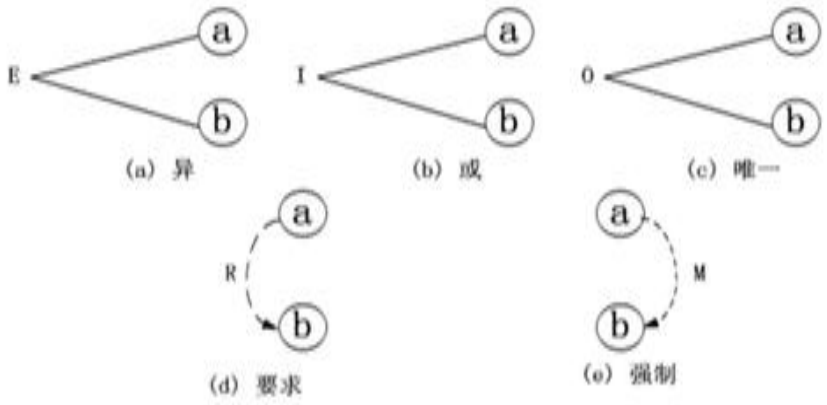
因果图（Cause-Effect Graphing）提供了一个把规格转化为判定表的系统化方法，从该图中可以产生测试数据。其中，原因是表示输入条件，结果是对输入执行的一系列计算后得到的输出。因果图方法最终生成的就是判定表。它适合于检查软件输入条件的各种组合情况。

1. 因果图基本符号



以上是关系符号:

- 恒等：若 c1 是 1，则 e1 也是 1；否则 e1 为 0。
- 非：若 c1 是 1，则 e1 是 0；否则 e1 是 1。
- 或：若 c1 或 c2 或 c3 是 1，则 e1 是 1；否则 e1 为 0。“或”可有任意个输入。
- 与：若 c1 和 c2 都是 1，则 e1 为 1；否则 e1 为 0。“与”也可有任意个输入。



以上是条件约束符号：

输入条件的约束有以下 4 类：

- E 约束（异）：a 和 b 中至多有一个可能为 1，即 a 和 b 不能同时为 1。
- I 约束（或）：a、b 和 c 中至少有一个必须是 1，即 a、b 和 c 不能同时为 0。
- O 约束（唯一）：a 和 b 必须有一个，且仅有 1 个为 1。
- R 约束（要求）：a 是 1 时，b 必须是 1，即不可能 a 是 1 时 b 是 0。

输出条件约束类型

- 输出条件的约束只有 M 约束（强制）：若结果 a 是 1，则结果 b 强制为 0。

2. 用例设计的步骤

- 把大的系统规格划分解成可以测试的规格片段；

- 分析分解后待测的系统规格，找出哪些是原因，哪些是结果
- 画出因果图
- 把因果图转换成判定表
- 简化判定表
- 用判定表中的每一项生成测试用例

3. 因果图法优缺点

优点：

- 等价类划分法尽管各个输入条件可能出错的情况都考虑到了，但是多个输入条件组合起来出错的情况却被忽略了；
- 因果图法能够帮助我们按照一定步骤，高效的选择测试用例，设计多个输入条件组合用例；
- 因果图分析还能为我们指出，程序规格说明描述中存在什么问题。

缺点：

- 输入条件与输出结果的因果关系，有时难以从软件需求规格说明书得到；
- 即使得到了这些因果关系，也会因为因果关系复杂导致因果图非常庞大，测试用例数目极其庞大。

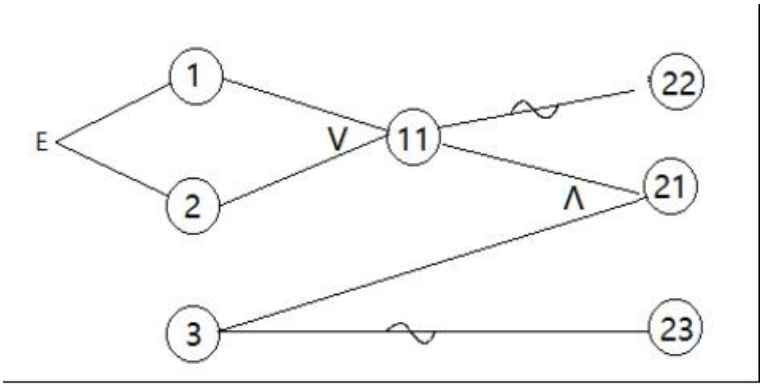
4. 实例分析

某软件规格说明书包含这样的要求：第一列字符必须是 A 或 B，第二列字符必须是一个数字，在此情况下进行文件的修改，但如果第一列字符不正确，则给出信息 L;如果第二列字符不是数字，则给出信息 M。

分析原因和结果：

原因	结果
1 - 第一列字符是 A	21 - 修改文件
2 - 第一列字符是 B	22 - 给出信息 L
3 - 第二列字符是一数字	23 - 给出信息 M

画出因果图：



考虑到原因 1 和原因 2 不可能同时为 1，因此在因果图上施加 E 约束；并且原因 1 或者原因 2 只要有一个成立，那么表示第一列已输入，为更好的表示这种关系，加入了 11 这个中间节点。

5. 测试用例

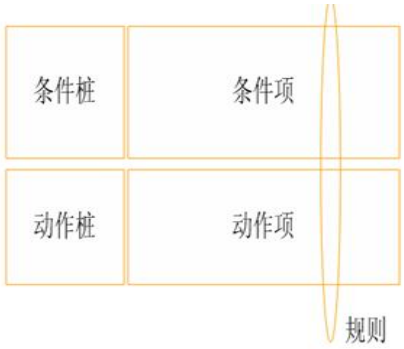
用例编号	功能模块	测试标题	前置条件	输入数据	操作步骤	期望结果	测试优先级
JX_ST_file_001	文件管理	第一行是A，第二行是数字	N/A	第一列字符：A 第二列字符：8	1、第一列输入字符A 2、第二列输入字符8	显示修改文件	高
JX_ST_file_002	文件管理	第一行是B，第二行是数字	N/A	第一列字符：B 第二列字符：2	1、第一列输入字符B 2、第二列输入字符2	显示修改文件	高
JX_ST_file_003	文件管理	第一行非A或B，第二行是数字	N/A	第一列字符：! 第二列字符：5	1、第一列输入字符! 2、第二列输入字符5	显示信息L	中
JX_ST_file_004	文件管理	第一行非A或B，第二行非数字	N/A	第一列字符：$ 第二列字符：d	1、第一列输入字符$ 2、第二列输入字符d	显示信息M	中

修改图

4.4 判定表法

判定表是分析和表达多种输入条件下系统执行不同动作的工具，它可以把复杂的逻辑关系和多种条件组合的情况表达得既具体又明确。

- 条件桩（Condition Stub）
- 动作桩（Action Stub）
- 条件项（Condition Entry）
- 动作项（Action Entry）



条件桩（Condition Stub）：列出了问题的所有条件。通常认为列出的条件的次序无关紧要。

动作桩（Action Stub）：列出了问题规定可能采取的操作。这些操作的排列顺序没有约束。

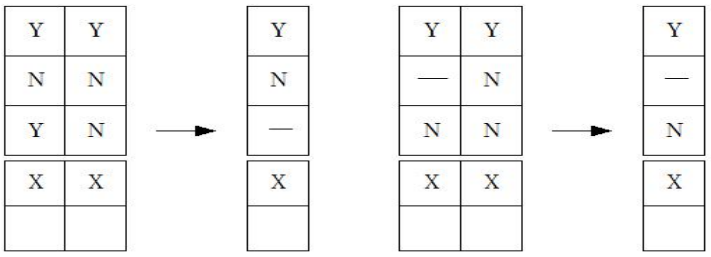
条件项（Condition Entry）：列出针对它左列条件的取值。在所有可能情况下的真假值。

动作项（Action Entry）：列出在条件项的各种取值情况下应该采取的动作。

条件项与动作项共同构成规则，一列包含条件项与动作项的列就是一条规则。

1. 判定表法的合并

化简工作是以合并相似规则为目标的。如果表中有两条或多条规则具有相同的动作，并且其条件项之间存在极为相似的关系，我们便可以将其合并



2. 用例设计的步骤

- 确定规则的个数。如这里有 3 个条件，每个条件有两个取值，故应有 2*2*2=8 种规则
- 列出所有的条件桩和动作桩

- 填入条件项（参照二进制 00000-11111 填入所有的条件项，针对每一列条件项依据每一列需求规格分析得到其对应的动作项，形成一条规则）
- 填入动作桩和动作项
- 化简，合并相似规则
- 将每条规则转化为用例

3. 实例分析

实例：某订购单检查逻辑如下：如果金额超过 500 元，且未过期，则发出批准单和提货单；如果金额超过 500 元，但过期了，则不发批准单；如果金额低于 500 元，则不论是否过期都发出批准单和提货单，在过期的情况下还需要发出通知单。

设计测试用例步骤如下：

a) 确定规则的个数，这里有金额是否超过 500 和订购单是否过期两个条件，每个条件两个取值，则规则个数 2*2=4 种规则；

b) 列入所有的条件桩和动作桩；

条件桩	条件项
金额是否超过500	1: 超过500 0: 小于等于500
订购单是否过期	0: 未过期 1: 已过期
动作桩	**动作项**
发出批准单	1: 表示发出批准单， 0: 表示不发
发出提货单	1: 表示发出提货单， 0: 表示不发
发出通知单	1: 表示发出通知单， 0: 表示不发

c) 填入条件项和动作项，条件项加动作项就形成一条规则（一列就是一条规则）；

		1	2	3	4
条件	金额是否超过500	1	1	0	0
	订购单是否过期	0	1	0	1
动作	发出批准单	1	0	1	1
	发出提货单	1	0	1	1
	发出通知单	0	0	0	1

d) 合并化简，根据合并原则（1、以相同动作项出发；2、相同的条件项直接合并；3、相反的条件忽略）进行合并；由于合并可能造成遗漏，因此要结合实际业务明确是否能够合并，如果不能，则不予合并。

		1	2	3	4
条件	金额是否超过500	1	1	0	0
	订购单是否过期	0	1	0	1
动作	发出批准单	1	0	1	1
	发出提货单	1	0	1	1
	发出通知单	0	0	0	1

合并

		1	2	3
条件	金额是否超过500	-	1	0
	订购单是否过期	0	1	1
动作	发出批准单	1	0	1
	发出提货单	1	0	1
	发出通知单	0	0	1

最终得到三条测试用例，其中第一条的金额可输入任意有效金额即可。

测试用例：

用例编号	功能模块	测试标题	前置条件	输入数据	操作步骤	期望结果	测试优先级
JX_ST_order_01	订单管理	订购单_金额超过500，且已过期的订单，不发批准单	存在金额超过500元，且已过期的订单	N/A	1、检查金额金额超过500元，且已过期的订单的状态	没有发出批准单	高
JX_ST_order_02	订单管理	订购单_金额超过501，且未过期的订单，发批准单和提货单	存在金额超过500元，且未过期的订单	N/A	1、检查金额金额超过500元，且未过期的订单的状态	发出批准单和提货单	高

实例 2：电商项目购物车，是比较重要的一个模块，每个用户都有一个属于自己的购物车，可以根据喜好任意删除购物车中的商品或添加新的商品，当用户结账时先从购物车中读出来再进行合计，结账完成后　要清空购物车，用户退出后，购物车将自动清空。分析：

a) 确定规则个数，这里有用户是否退出和是否完成支付两个条件。每个条件两个取值，则规则个数 2*2=4 种规则

b) 确定条件桩，动作桩。

条件桩	条件项
是否完成支付	1：完成
	0：未完成
用户退出登录	1：退出
	0：不退出
动作桩	**动作项**
清空购物车	1：清空
	0：不清空

c) 填入条件项和动作项，一条规则就是一条测试用例。

		1	2	3	4
条件	是否完成支付	1	1	0	0
	用户退出登录	1	0	1	0
动作	清空购物车	1	1	1	0

d） 测试用例

用例编号	功能模块	测试标题	前置条件	输入数据	操作步骤	期望结果	测试优先级
JX_ST_cart_01	购物车管理	购物车_用户完成支付，未退出系统	进入购物车，且购物车有添加的商品	N/A	1、在购物车界面，点击【结算中心】 2、支付页面，点击【提交订单】 3、返回购物车，查看购物车中商品。	1、跳转至结算界面 2、订单提交成功，并生成订单号 3、购物车清空	高
JX_ST_cart_02	购物车管理	购物车_用户未支付，退出系统	进入购物车，且购物车有添加的商品	N/A	1、用户点击【退出】。 2、再次登录，点击【购物车】，查看购物车中商品	1、退出系统 2、购物车清空	高

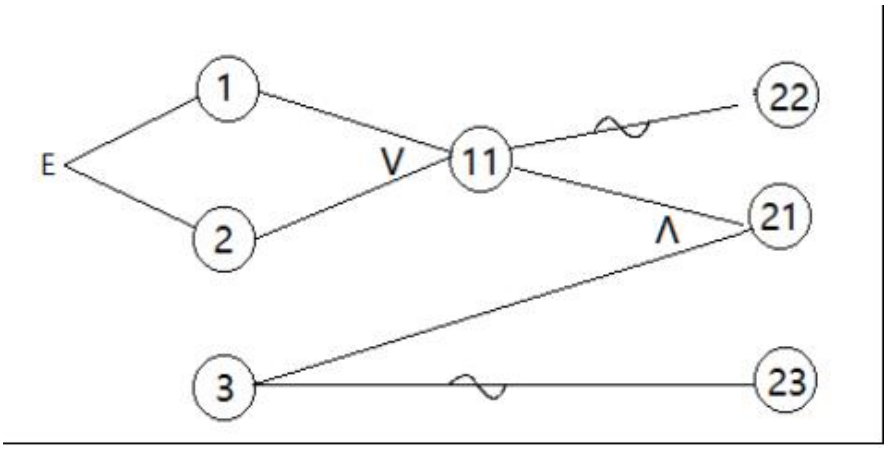
4. 判定表的优缺点

优点：它能把复杂的问题按各种可能的情况一一列举出来，简明而易于理解，也可避免遗漏

缺点：合并存在漏测的风险。一个显然易见的原因是，虽然某个输入条件在输出接口上是无关的，但是在软件设计上，内部针对这个条件走了不同的程序分支（因分析内部业务流程而定）；输入和输出的逻辑关系和明确用判定表，不是很明了的用因果图然后使用判定表。

5. 因果图判定表结合实例

根据因果图法中实例得到的因果图：



得出判定表和用例如下：

		1	2	3	4	5	6	7	8
条件	1	1	1	1	1	0	0	0	0
	2	1	1	0	0	1	1	0	0
	3	1	0	1	0	1	0	1	0
	11			1	1	1	1	0	0
动作	22			0	0	0	0	1	1
	21			1	0	1	0	0	0
	23			0	1	0	1	0	1
测试用例				A3	AM	B5	BN	C2	DY
				A8	A?	B4	B!	X6	P$

思考：

尝试对上面的判定表进行合并化简。

4.5 状态迁移图法

许多需求用状态机的方式来描述，状态机的测试主要关注在测试状态转移的正确性上面。对于一个有限状态机，通过测试验证其在给定的条件内是否能够产生需要的状态变化，有没有不可达的状态和非法的状态，可能不可能产生非法的状态转移等。

通过构造能导致状态迁移的事件来测试状态之间的转换，用这种方法可以设计逆向的测试用例，如状态和事件的非法组合。

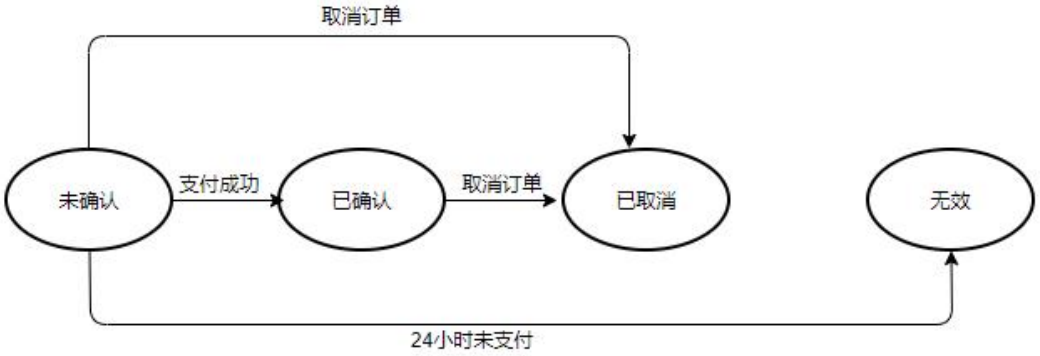
1. **用例设计步骤**
   1. 画出状态迁移图
   2. 列出状态—事件表
   3. 从状态转换树推导出测试路径
   4. 根据测试路径编写合法测试用例
   5. 编写非法测试用例
2. **实例分析**

某电商的订单支付需求描述如下：

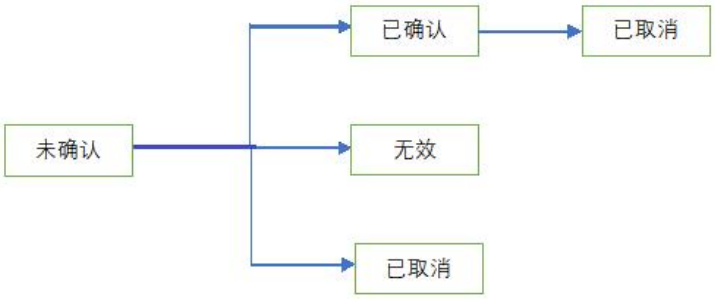
提交订单成功，则订单状态为未确认；当支付成功后，订单状态更新为已确认；商家会安排发货。如果订单提交成功后，24 小时未支付成功，则此订单状态为无效。如果用户发现订单的商品并不适合，可选择取消，执行成功后，则该订单状态为已取消。

分析：

根据需求描述，提取出订单的状态。画出状态迁移图



状态转换树



状态转换树生成要点：

- 初始状态或开始状态是树的根；
- 对于状态图内从开始状态出发到达一个任意可达状态的每个可能的转换，转换树都包含了从根出发到达一个代表此状态的下一个后续状态的节点的分支；
- 对转换树的每个叶节点重复上一个步骤，直到满足下列两个结束条件之一：
  1. 与叶节点相关的状态已经出现过一次从状态树的根到叶节点的连接上，这个结束条件对应于状态图中的一遍循环；
  2. 与叶节点相关的节点是一个结束状态，并且也没有更多的状态转换需要考虑。
- 推出测试路径；
- 根据测试路径编写测试用例，每一条路径就是一条测试用例。

状态转换测试完成准则的定义：

- 每个状态至少到达一次；
- 每个状态转换至少执行一次；
- 所有不符合规格说明的状态转换都已检查。

4.6 流程分析法

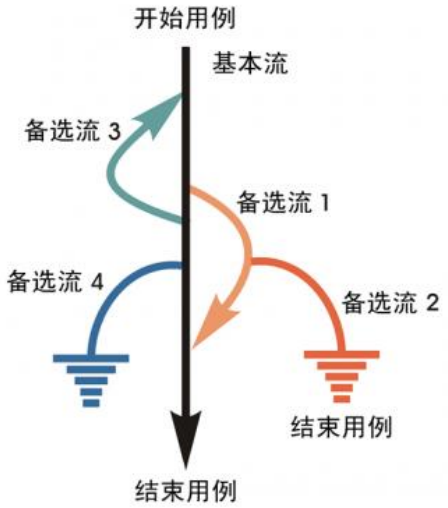
流程分析法是将软件系统的某个流程看成路径，用路径分析的方法来设计测试用例。根据流程的顺序依次进行组合，使得流程的各个分支都能走到。这是从白盒测试中路径覆盖分析法中推广到黑盒测试中来的测试分析方法。

流程分析法也叫场景分析法，通过分析软件应用的场景，从用户的角度出发，从场景的角度来设计测试用例，是一种面向用户的测试用例设计方法。关心用户做什么，而不是关心产品做什么。

优点：实用性强，有效，设计出来的用例有价值；

缺点：可能使用的场景不一定能对事件系列进行全面的分析，设计出来的用例不完整。

场景分析是通过描述流经用例路径来确定的过程，这个流经过程要从用例开始到结束遍历其中所有基本流 ：直黑线表示基本流，是最基本、最简单的路径；（软件功能按照正确的事件流实现的一条正确流程无任何错，程序从开始直到结束）



遵循上图中每个经过用例的可能路径，可以确定不同的用例场景。从基本流开始，再将基本流和备选流结合起来，可以确定以下用例场景：

场景1	基本流			
场景2	基本流	备选流1		
场景3	基本流	备选流1	备选流2	
场景4	基本流	备选流3		
场景5	基本流	备选流3	备选流1	
场景6	基本流	备选流3	备选流1	备选流2
场景7	基本流	备选流4		
场景8	基本流	备选流3	备选流4	

1. 用例设计步骤

a） 画出业务流程图

b） 确定测试场景（路径）

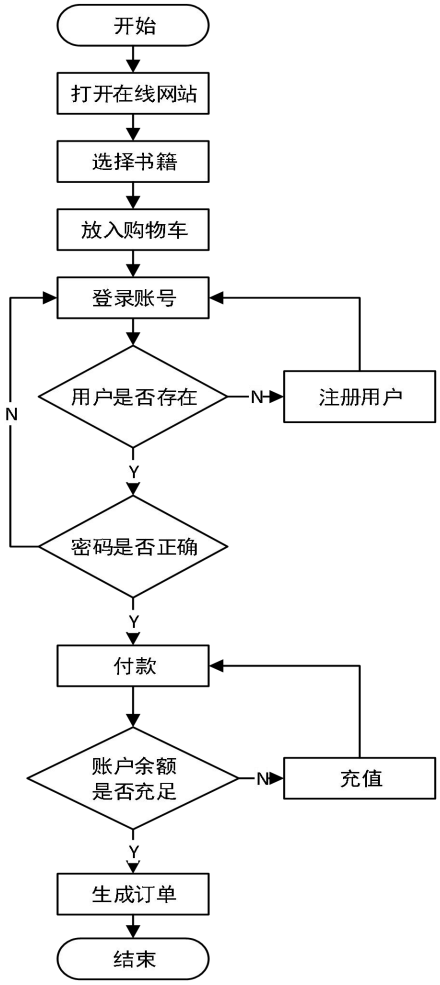
c） 构造测试用例

d） 选取测试数据

2. 实例分析

某电商网站业务：用户打开网站后，进行书籍的选择，当选好自己心仪的书籍后进行订购，这时把所需图书放进购物车，等进行结账的时候，用户需要登录自己注册的账号，登录成功后，进行付款交易，交易成功后，生成订购单，整个购物过程结束。

第一步、画出流程图，确定基本流和备选流；



基本流：打开在线网站→选择书籍→放入购物车→登录账号→付款→生成订单；

备选流 1 ：用户不存在→注册用户；

备选流 2 ：密码不正确；

备选流 3 ：账户余额不足→充值。

第二步、根据基本流和各项备选流确定场景；

场景 1 （成功购物）：基本流

场景 2 （账户不存在）：基本流 备选流 1

场景 3 （账户密码错误）：基本流 备选流 2

场景 4 （账户余额不足）：基本流 备选流 3

第三步、对每一个场景生成测试用例；

用例编号	场景步骤	步骤描述	输入	预期结果
1	场景1：成功购物	登录在线网站		成功购物
		选择书籍	总价：80	
		放入购物车		
		登录账号	账户：张三 密码：123456	
		付款	账户余额：200	
		生成订单		
2	场景2：账户不存在	登录在线网站		提示帐户不存在，返回基本流步骤4
		选择书籍	总价：80	
		放入购物车		
		登录账号	账户：李四 密码：123456	
		账户不存在，注册用户		
		登录账号		
		付款	账户余额：100	
		生成订单		
3	场景3：账户密码错误	登录在线网站		提示帐户密码错误，返回基本流步骤4
		选择书籍	总价：80	
		放入购物车		
		登录账号	账户：张三 密码：123	
		密码错误，重新输入登录	账户：张三 密码：123	
		付款	账户余额：120	
		生成订单		
4	场景4：账户余额不足	登录在线网站		提示帐户余额不足，请充值； 返回基本流步骤5
		选择书籍	总价：80	
		放入购物车		
		登录账号	账户：李四 密码：123456	
		付款	账户余额：20	
		余额不足，充值	冲值：60	
		付款		
		生成订单		

3. 测试用例

用例编号	功能模块	测试标题	前置条件	输入数据	操作步骤	期望结果	测试优先级
JX_ST_flow_01	购物流程	购物流程_登录账号存在	1、已存在注册的合法账户 2、账户余额：200元	总价：80元 用户名：张三 密码：Zh12534	1、访问在线网站 2、选择书籍，点击加入购物车 3、在购物车，点击【提交订单】 4、输入登录账号和密码，点击【登录】 5、支付订单，完成付款操作	1、成功访问在线网站 2、加购物车成功 3、提交订单时，提示先登录，并跳转至登录页 4、提示：登录成功，并跳转至支付页面 5、支付成功	高
JX_ST_flow_02	购物流程	购物流程_登录账号不存在	账号：李四不是该系统的注册账号	用户名：李四 密码：123456	1、访问在线网站 2、选择书籍，点击加入购物车 3、在购物车，点击【提交订单】 4、输入登录账号和密码，点击【登录】	1、成功访问在线网站 2、加购物车成功 3、提交订单时，提示先登录，并跳转至登录页 4、提示：账户不存在，并跳转至登录页面。	高

4.7 正交试验法

正交试验设计(Orthogonal experimental design)是研究多因素多水平的一种设计方法，它是根据正交性从全面试验中挑选出部分有代表性的点进行试验，这些有代表性的点具备了“均匀分散，齐整可比”的特点，正交试验设计是分式析因设计的主要方法。是一种高效率、快速、经济的实验设计方法。

日本著名的统计学家田口玄一将正交试验选择的水平组合列成表格，称为正交表。例如作一个三因素三水平的实验，按全面实验要求，须进行 3*3*3=27 种组合的实验，且尚未考虑每一组合的重复数。若按 $L_9(3^3)$ 正交表安排实验，只需作 9 次显然大大减少了工作量。因而正交实验设计在很多领域的研究中已经得到广泛应用。

正交表的构成：

- **行数（Runs）**：正交表中行数的个数，即实验的次数，也是我们通过正交实验法设计的测试用例个数；
- **因素数（Factors）**：正交表中列的个数，即我们要测试的功能点；
- **水平数（Levels）**：任何单个因素能够取得值的最大个数。

- **正交表公式：**

$$L_n(m^k)$$

n 为行数，也就是实验次数；m 为水平数，也就是单个因素取值的个数；k 为因素数，也就是要测试的功能点的数量。

正交表具有以下两项性质：

整齐可比性

在同一张正交表中，每个因素的每个水平出现的次数是完全相同的。由于在试验中每个因素的每个水平与其它因素的每个水平参与试验的机率是完全相同的，这就保证在各个水平中最大程度的排除了其它因素水平的干扰。

例如在两水平正交表中，任何一列都有数码“1”与“2”，且任何一列中它们出现的次数是相等的；如在三水平正交表中，任何一列都有“1”、“2”、“3”，且在任一列的出现数均相等。

均衡分散性

在同一张正交表中，任意两列（两个因素）的水平搭配（横向形成的数字对）是完全相同的。这样就保证了试验条件均衡地分散在因素水平的完全组合之中，因而具有很强的代表性，容易得到好的试验条件。

例如在两水平正交表中，任何两列(同一横行内)有序对子共有 4 种：(1，1)、(1，2)、(2，1)、(2，2)。每种对数出现次数相等。在三水平情况下，任何两列(同一横行内)有序对共有 9 种，1.1、1.2、1.3、2.1、2.2、2.3、3.1、3.2、3.3，且每对出现数也均相等。

以上两点充分的体现了正交表的两大优越性，即“均匀分散性，整齐可比”。通俗的说，每个因素的每个取值与另一个因素每个取值各碰一次，这就是正交性。

1. **用例设计步骤**

一、提取功能说明，构造因子－状态表；

$L_n(m^k)$	因素1	因素2	……	因素m
行数1				
行数2				
……				
行数n				

二、计算各因子和状态的权值，删去一部分权值较小的，即重要性较小的因子或状态，使最后生成的测试用例集缩减到允许范围；

三、利用正交表构造测试数据集：

1、如果各个因子的状态数是不统一的，几乎不可能出现均匀的情况 。必须首先用逻辑命令来组合各因子的状态，作出布尔图；

2、根据布尔图查找最接近的相应阶数的正交表；

3、依照因果图上根节点到叶子节点的顺序逐步替换正交表上的中间节点，得到最终的正交表。

四、选择合适的正交表；

五、利用正交表每行数据构造测试用例；

六、补充重要的但未在表中体现的组合。

2. 正交表举例

电商系统的登录界面如下：

确定因素：

为简化示例，只用户名、密码、验证码考虑填写或不填写。

选择合适的正交表：

1、表中的因素数>=3；

2、表中至少有 3 个因素数的水平数>=2；

3、行数取最少的一个。

刚好有合适的正交表$L_4(2^3)$。

映射正交表：

$L_4(2^3)$	因素1	因素2	因素3
行数1	1	1	1
行数2	1	2	2
行数3	2	1	2
行数4	2	2	1

1. 填
2. 不填

$L_4(2^3)$	用户名	密码	验证码
行数1	填	填	填
行数2	填	不填	不填
行数3	不填	填	不填
行数4	不填	不填	填

3. 测试用例

用例 1：填写用户名、填写密码、填写验证码；

用例 2：填写用户名、不填写密码、不填写验证码；

用例 3：不填写用户名、填写密码、不填写验证码；

用例 4：不填写用户名、不填写密码、填写验证码。

用例编号	功能模块	测试标题	前置条件	输入数据	操作步骤	期望结果	测试优先级
JX_ST_login_01	登录	登录_填写用户名、密码和验证码	有已成功注册的用户数据	用户名：王斌 密码：123456 验证码：V8DJ	1、在登录界面，输入用户名、密码和验证码 2、点击【立即登陆】	1、明文显示已输入的用户名和验证码，密码加密显示。 2、登录成功	高
JX_ST_login_02	登录	登录_填写用户名，不填密码和验证码	有已成功注册的用户数据	用户名：王斌 密码： 验证码：	1、在登录界面，只输入用户名 2、点击【立即登陆】	1、明文显示已输入的用户名 2、登录失败，给出响应的提示信息。	中

如果对因素的所有取值进行全覆盖的话，需要 2*2*2=8 个用例，通过正交表设计得出只需要 4 个用例。

注意：

通过上例中也得出了正交表的缺陷，正交表主要是为了满足多因素各个取值均进行一次碰撞，并没有区分先后和逻辑关系，因此会遗漏部分重要的测试用例。比如：填写用户名、填写密码、不填写验证码的情况，这是测试验证码是否有效的用例，通过正交表无法得出；另外，如果全部不填，直接登录也是一个常规的用例。

因此在应用正交表的时候，一定要注意如果各因素的不同取值之间存在先后顺序和逻辑关系的时候，需要添加额外的测试用例。

虽然正交表存在这样的缺点，但是当我们对于因素的取值进行权衡后，对于没有逻辑关系的取值应用正交表方法，可以非常有效的减少我们的测试用例。

比如一般系统中常见的多条件组合查询的功能，由于各条件之间极少存在逻辑关系和先后顺序 ，那么更适合应用正交表的方法。

4.8 异常分析法

系统异常分析法就是针对系统有可能存在的异常操作、软硬件缺陷引起的故障进行分析，依此设计测试用例。主要针对系统的容错能力、故障恢复能力进行测试。例如：

1) 比如说操作进行时，断电、断网、死机等原因导致的信息丢失的异常。
2) 订购过程中，用户或产品的状态变化引起的异常。例如商品下架或价格调整的处理；或是用户在下单后付款前，被取消等。
3) 操作中应该选择的选项没有选择时的场景，例如购买产品服务时，未选择同意服务协议的场景，此时付款按钮应该灰显，无法进行付款操作。
4) 通过构造 URL 产生的异常场景。例如用户存在某产品失效的订单，通过 URL 进入订单支付的页面的异常情况，此时应该提示此订单已经失效，支付不成功。
5) 打开两个页面做相同操作时的异常流。例如，用户满足订购该产品的条件，用户同时打开两个购买该产品的页面 A 和 B，当在 A 页面订购成功后，点击 B 页面的订购可能有三种可能：一是若订购的产品是周期型的，则进入续费的流程；二是，若订购的产品是永久型的，则会提示不可重复订购；三是，若订购的产品是计量型的，则可继续正常订购。
6) 用户账户余额不足，充值失败的异常场景。

4.9 错误猜测法

随着在产品测试的实践中对产品的了解和测试经验的丰富，使用错误猜测法设计的测试用例往往非常有效，可以作为测试设计的一种补充手段。并且积累的经验越丰富，方法使用效率越高。那么到底什么是错误猜测法呢，下面我们将通过定义和实际测试案例来加深对错误猜测法的认识。

我们先来看看错误猜测法的定义：**有经验的测试人员往往可以根据自己的工作经验和直觉推测出程序可能存在的错误，从而有针对性的进行测试。**它的要素共有三点，分别为：经验、知识、直觉。关于如何使用的问题，我们提炼出两点：

1. 错误猜测法只能作为测试设计的补充而不能单独用来设计测试用例，否则可能会造成测试的不充分；

2. 基本思想：列举出程序中所有可能有的错误和容易发生错误的特殊情况,根据他们选择测试用例。

我们知道经验是错误猜测法的一个重要要素，也就说带有主观性，那么这就决定了错误猜测法的优缺点，首先我们来看优点：

- 充分发挥人的直觉和经验
- 集思广益
- 方便使用
- 快速容易切入

对应的缺点有：

- 难以知道测试的覆盖率
- 可能丢失大量未知的区域
- 带有主观性且难以复制

错误猜测的一些例子：

1. 例如， 输入数据和输出数据为 0，1 的情况；输入表格为空格或输入表格只有一行。 这些都是容易发生错误的情况。可选择这些情况下的例子作为测试用例。
2. 例如，测试一个对线性表（比如数组）进行排序的程序，可推测列出以下几项需要特别测试的情况：
   1) 输入的线性表为空表
   2) 表中只含有一个元素
   3) 输入表中所有元素已排好序
   4) 输入表已按逆序排好
   5) 输入表中部分或全部元素相同
3. 例如，测试手机终端的通话功能，可以设计各种通话失败的情况来补充测试用例：
   1) 无 SIM 卡插入时进行呼出（非紧急呼叫）
   2) 插入已欠费 SIM 卡进行呼出
   3) 射频器件损坏或无信号区域插入有效 SIM 卡呼出
   4) 网络正常，插入有效 SIM 卡，呼出无效号码（如 1、888、333333、不输入任何号码等）
   5) 网络正常，插入有效 SIM 卡，使用“快速拨号”功能呼出设置无效号码的数字

要想灵活运用错误猜测法，平时工作中就需要多观察总结，将一些容易犯错的地方进行归类整理；并多观察 bug 系统上的历史问题，通过对历史问题进行归类分析；同时多了解业内一些网站、论坛上的经典 bug 分享。这些方式都能快速增加自己的经验积累，变成错误猜测法的素材。

4.10 测试用例综合设计策略

Myers（软件测试经典之作《软件测试的艺术》提出了使用各种测试方法的综合策略：

1) 在任何情况下都必须使用边界值分析方法，经验表明用这种方法设计出测试用例发现程序错误的能力最强。
2) 必要时用等价类划分方法补充一些测试用例。

3) 用错误推测法再追加一些测试用例。

4) 对照程序逻辑，检查已设计出的测试用例的逻辑覆盖程度，如果没有达到要求的覆盖标准，应当再补充足够的测试用例。

5) 如果程序的功能说明中含有输入条件的组合情况，则一开始就可选用因果图法和判定表。

优化测试用例的方法

1) 利用设计测试用例的 9 种方法不断的对测试用例进行分解与合并；

2) 优化测试用例；