**黑盒测试用例设计方法**

黑盒测试用例设计方法包括等价类划分法、边界值分析法、错误推测法、因果图法、判定表驱动法、正交试验设计法、功能图法、场景图法等。

# （一）等价类划分法

定义：等价类划分法是把所有可能输入的数据，即程序的输入域划分策划国内若干部分（子集），然后从每一个子集中选取少数具有代表性的数据作为测试用例。方法是一种重要的、常用的黑盒测试用例设计方法。

等价类是指某个输入域的子集合。在该子集合中，各个输入数据对于揭露程序中的错误都是等效的，并合理地假定：测试某等价类的代表值就等于对这一类其他值的测试，因此，可以把全部输入数据合理划分为若干等价类，在每一个等价类中取一个数据作为测试的输入条件就可以用少量代表性的测试数据取得较好的测试结果。等价类划分有两种不同的情况：有效等价类和无效等价类。

有效等价类，是指对于程序的规格说明来说是合理的、有意义的输入数据构成的集合。利用有效等价类可检验程序是否实现了规格说明所规定的功能和性能。

无效等价类 指对程序的规格说明是不合理的或无意义的输入数据所构成的集合。对于具体的问题，无效等价类至少应有一个，也可能多个。

划分标准：

1) 完备测试、避免冗余

2) 划分等价类重要的是：集合的划分、划分为互不相交的一组子集，而子集的并是整个集合

3) 并是整个集合：备性

4) 子集互不相交：保证一种形式的无冗余性

5) 同一类中标识（选择）一个测试用例，同一等价类中，往往处理相同，相同处理映射到“相同的执行路径”。

划分方法：

1) 在输入条件规定了取值范围或值的个数的情况下，则可以确立一个有效等价类和两个无效等价类。如：输入值是学生成绩，范围是0~100；



2）在输入条件规定了输入值的集合或者规定了“必须如何”的条件的情况下，可确立一个有效等价类和一个无效等价类：

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。布尔量是一个二值枚举类型, 一个布尔量具有两种状态: true 和 false 。

4）在规定了输入数据的一组值（假定n个），并且程序要对每一个输入值分别处理的情况下，可确立n个有效等价类和一个无效等价类。

例：输入条件说明学历可为：专科、本科、硕士、博士四种之一，则分别取这四种的四个值作为四个有效等价类，另外把四种学历之外的任何学历作为无效等价类。

5）在规定了输入数据必须遵守的规则情况下，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）；

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，则应在将该等价类进一步的划分为更小的等价类。

转化为测试用例：

在确立了等价类后，可建立等价类表，列出所有划分出的等价类输入条件：有效等价类、无效等价类，然后从划分出的等价类中按以下三个原则设计测试用例：

1）为每一个等价类规定一个唯一的编号；

2）设计一个新的测试用例，使其尽可能多地覆盖尚未被覆盖地有效等价类，重复这一步，直到所有的有效等价类都被覆盖为止；

3）设计一个新的测试用例，使其仅覆盖一个尚未被覆盖的无效等价类，重复这一步，直到所有的无效等价类都被覆盖为止。

# （二）边界值分析法

定义：边界值分析法就是对输入或输出的边界值进行测试的一种黑盒测试方法。通常边界值分析法是作为对等价类划分法的补充，这种情况下，其测试用例来自等价类的边界。

与等价类区别：

1）边界值分析不是从某等价类中随便挑一个作为代表，而是使这个等价类的每个边界都要作为测试条件。

2）边界值分析不仅考虑输入条件，还要考虑输出空间产生的测试情况。

分析方法：

大量的错误是发生在输入或输出范围的边界上，而不是发生在输入输出范围的内部。因此针对各种边界情况设计测试用例，可以查出更多的错误。使用边界值分析方法设计测试用例，首先应确定边界情况。通常输入和输出等价类的边界，就是应着重测试的边界情况。应当选取正好等于，刚刚大于或刚刚小于边界的值作为测试数据，而不是选取等价类中的典型值或任意值作为测试数据。

常见边界值：

1）对16Bit的整数而言，32767和32768是边界

2）屏幕上光标在最左上、最右下位置

3）报表的第一行和最后一行

4）数组元素的第一个和最后一个

5）循环的第0次、第1次和倒数第2次、最后一次

边界值分析：

1）边界值分析使用与等价类划分法相同的划分，只是边界值分析假定错误更多地存在于划分的边界上，因此在等价类的边界上以及两侧的情况设计测试用例。

例：测试计算平方根的函数

输入：实数

输出：实数

规格说明：当输入一个0或比0大的数的时候，返回其正平方根；当输入一个小于0的数时，显示错误信息“平方根非法，输入值小于0”并返回0；库函数printLine可以用来输出错误信息。

2）等价类划分：

i. 可以考虑做出如下划分：

A、输入(i)<0 和(ii)>=0

B、输出(a)>=0和(b)Error

ii. 测试用例有两个

A、输入4，输出2.对应(ii)和(a)。

B、输入10，输出0和错误提示。对应与(i)和(b)

3）边界值分析

划分(ii)的边界为0和最大正实数；划分(i)的边界为最小负实数和0.由此得到一下测试用例：

A、输入｛最小负实数｝

B、输入｛绝对值很小的负数｝

C、输入0

D、输入｛绝对值很小的正数｝

E、输入｛最大正实数｝

4）通常情况下，软件测试所包含的边界检验有几种类型：数字、字符、位置、重量、大小、速度、方位、尺寸、空间等。

5）相应地，以上类型的边界值应该在：最大/最小、首位/末位、上/下、最快/最慢、最高/最低、最短/最长、空/满等情况下。

6）利用边界值作为测试数据



1. 内部边界值分析

在多数情况下，边界值条件是基于应用程序的功能设计而需要考虑的因素，可以从软件的规格说明或常识中得到，也是最终用户可以很容易发现问题的。然而，在测试用例设计过程中，某些边界值条件是不需要呈现给用户的，或者说用户是很难注意到的，但同时确实属于检验范畴内的边界条件，称为内部边界值条件或子边界值条件。

内部边界值条件主要有下面几种：

1. 数值的边界值检验：计算机是基于二进制进行工作的，因此，软件的任何数值运算都有一定的范围限制。

2）字符的边界值检验：在计算机软件中，字符也是很重要的表示元素，其中ASCII和Unicode是常见的编码方式。

3）其它边界值检验：在不同的行业应用领域，依据硬件和软件的标准不同而具有各自特定的边界值。

转化为测试用例：

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的值,以及刚刚超越这个范围边界的值作为测试输入数据。

Ø 例如，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……"。作为测试用例，我们应取10及50，还应取10.01,49.99,9.99及50.01等。

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测试数据。

Ø 例如，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) 将规则1）和2）应用于输出条件，即设计测试用例使输出值达到边界值及其左右的值。

Ø 例如，某程序的规格说明要求计算出"每月保险金扣除额为0至1165.25元"，其测试用例可取0.00及1165.24、还可取一0.01及1165．26等。

Ø 再如一程序属于情报检索系统，要求每次"最少显示1条、最多显示4条情报摘要"，这时我们应考虑的测试用例包括1和4，还应包括0和5等。

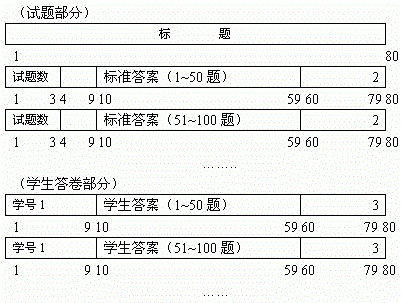
4) 如果程序的规格说明给出的输入域或输出域是有序集合,则应选取集合的第一个元素和最后一个元素作为测试用例。

5) 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。

6) 分析规格说明,找出其它可能的边界条件。

实例1，批阅试卷

现有一个学生标准化考试批阅试卷,产生成绩报告的程序。其规格说明如下:程序的输入文件由一些有80个字符的记录组成,如右图所示，所有记录分为3组：



1) 标题：这一组只有一个记录，其内容为输出成绩报告的名字。

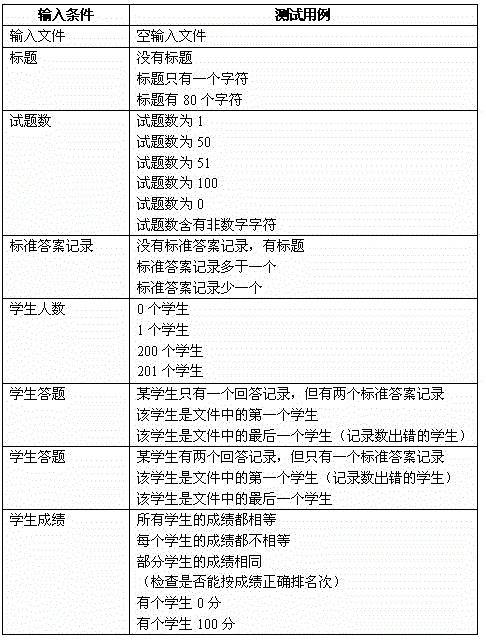
2)   
试卷各题标准答案记录：每个记录均在第80个字符处标以数字"2"。该组的第一个记录的第1至第3个字符为题目编号（取值为1一999）。第10至第59个字符给出第1至第50题的答案（每个合法字符表示一个答案）。该组的第2，第3……个记录相应为第51至第100，第101至第150，…题的答案。

3)   
每个学生的答卷描述：该组中每个记录的第80个字符均为数字"3"。每个学生的答卷在若干个记录中给出。如甲的首记录第1至第9字符给出学生姓名及学号，第10至第59字符列出的是甲所做的第1至第50题的答案。若试题数超过50，则第2，第3……纪录分别给出他的第51至第100，第101至第150……题的解答。然后是学生乙的答卷记录。

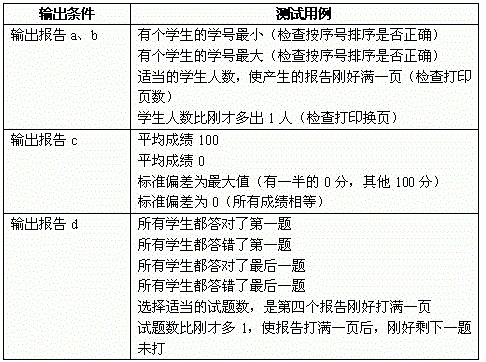
4) 学生人数不超过200，试题数不超过999。

5) 程序的输出有4个报告：  
a)按学号排列的成绩单，列出每个学生的成绩、名次。  
b)按学生成绩排序的成绩单。  
c)平均分数及标准偏差的报告。  
d)试题分析报告。按试题号排序，列出各题学生答对的百分比。

解答：分别考虑输入条件和输出条件，以及边界条件。给出下表所示的输入条件及相应的测试用例。



输出条件及相应的测试用例表。



# （三）错误推测法

定义：基于经验和直觉推测程序中所有可能存在的各种错误，从而有针对性的设计测试用例的方法。

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。

1. 例如,输入数据和输出数据为0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，前面例子中成绩报告的程序，采用错误推测法还可补充设计一些测试用例：

1) 程序是否把空格作为回答

2) 在回答记录中混有标准答案记录

3) 除了标题记录外，还有一些的记录最后一个字符即不是2也不是3

4) 有两个学生的学号相同

5) 试题数是负数

3. 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。

4. 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

# （四）因果图法

定义：因果图法是一种利用图解法分析输入的各种组合情况，从而设计测试用例的方法，它适合于检查程序输入条件的各种组合情况。

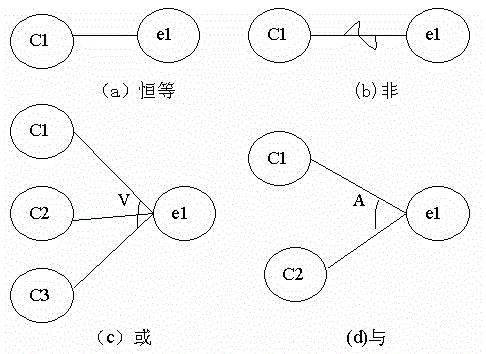
应用：

等价类划分法和边界值分析方法都是着重考虑输入条件，但没有考虑输入条件的各种组合、输入条件之间的相互制约关系。这样虽然各种输入条件可能出错的情况已经测试到了，但多个输入条件组合起来可能出错的情况却被忽视了。

如果在测试时必须考虑输入条件的各种组合，则可能的组合数目将是天文数字，因此必须考虑采用一种适合于描述多种条件的组合、相应产生多个动作的形式来进行测试用例的设计，这就需要利用因果图（逻辑模型）。

1. 因果图介绍

1) 4种符号分别表示了规格说明中向4种因果关系。



2) 因果图中使用了简单的逻辑符号，以直线联接左右结点。左结点表示输入状态（或称原因），右结点表示输出状态（或称结果）。

3) C1表示原因，通常置于图的左部；e1表示结果，通常在图的右部。C1和e1均可取值0或1，0表示某状态不出现，1表示某状态出现。

2. 因果图涉及的概念

1) 关系

Ø 恒等：若c1是1，则e1也是1；否则e1为0。

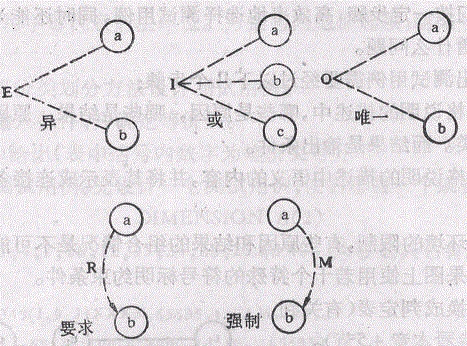
Ø 非：若c1是1，则e1是0；否则e1是1。

Ø 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。

Ø 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。

2) 约束

输入状态相互之间还可能存在某些依赖关系，称为约束。例如,某些输入条件本身不可能同时出现。输出状态之间也往往存在约束。在因果图中,用特定的符号标明这些约束。



Ø 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。

Ø 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。

3. 采用因果图法设计测试用例的步骤：

1) 分析软件规格说明描述中,那些是原因(即输入条件或输入条件的等价类),那些是结果(即输出条件),并给每个原因和结果赋予一个标识符。

2) 分析软件规格说明描述中的语义，找出原因与结果之间,原因与原因之间对应的关系，根据这些关系,画出因果图。

3) 由于语法或环境限制,有些原因与原因之间,原因与结果之间的组合情况不可能出现，为表明这些特殊情况,在因果图上用一些记号表明约束或限制条件。

4) 把因果图转换为判定表。

5) 把判定表的每一列拿出来作为依据,设计测试用例。

实例1，字符

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。

解答：

1) 根据题意，原因和结果如下：

原因：

1——第一列字符是A；

2——第一列字符是B；

3——第二列字符是一数字。

结果：

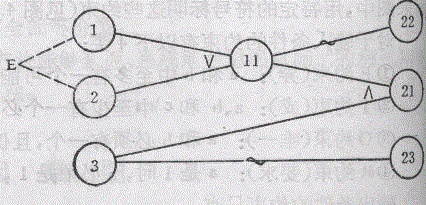
21——修改文件；

22 ——给出信息L；

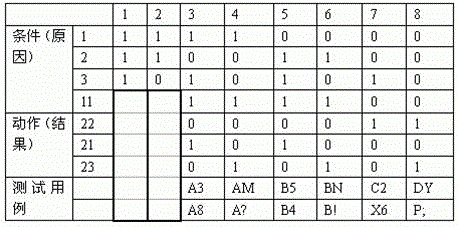
23——给出信息M。

2) 其对应的因果图如下：

11为中间节点；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。



3) 根据因果图建立判定表。



表中8种情况的左面两列情况中，原因①和原因②同时为1，这是不可能出现的，故应排除这两种情况。表的最下一栏给出了6种情况的测试用例，这是我们所需要的数据。

实例2，自动售货机

有一个处理单价为5角钱的饮料的自动售货机软件测试用例的设计。其规格说明如下：若投入5角钱或1元钱的硬币，押下〖橙汁〗或〖啤酒〗的按钮，则相应的饮料就送出来。若售货机没有零钱找，则一个显示〖零钱找完〗的红灯亮，这时在投入1元硬币并押下按钮后，饮料不送出来而且1元硬币也退出来；若有零钱找，则显示〖零钱找完〗的红灯灭，在送出饮料的同时退还5角硬币。

1) 分析这一段说明，列出原因和结果

原因：

1——售货机有零钱找

2——投入1元硬币

3——投入5角硬币

4——押下橙汁按钮

5——.押下啤酒按钮

结果：

21——售货机〖零钱找完〗灯亮

22——退还1元硬币

23——退还5角硬币

24——送出橙汁饮料

25——送出啤酒饮料

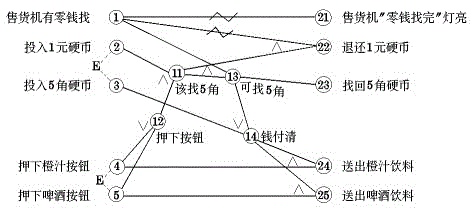
2) 画出因果图，如图所示。所有原因结点列在左边，所有结果结点列在右边。建立中间结点，表示处理的中间状态。中间结点：

11—— 投入1元硬币且押下饮料按钮

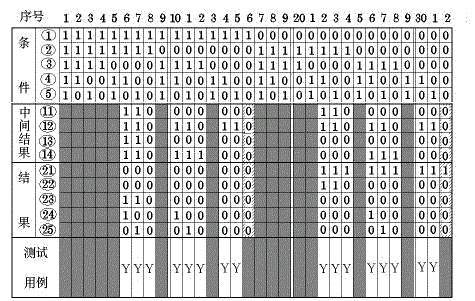
12——押下〖橙汁〗或〖啤酒〗的按钮

13——应当找5角零钱并且售货机有零钱找

14——钱已付清



3) 转换成判定表：



4) 在判定表中，阴影部分表示因违反约束条件的不可能出现的情况，删去。第16列与第32列因什么动作也没做，也删去。最后可根据剩下的16列作为确定测试用例的依据。

# （五）判定表（决策表）驱动法

定义：判定表是分析和表达多逻辑条件下执行不同操作的情况的工具。

优点：能够将复杂的问题按照各种可能的情况全部列举出来，简明并避免遗漏。因此，利用判定表能够设计出完整的测试用例集合。在一些数据处理问题当中，某些操作的实施依赖于多个逻辑条件的组合，即：针对不同逻辑条件的组合值，分别执行不同的操作。判定表适合于处理这类问题。

判定表由四部分组成，如下图：

1) 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。

2) 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。

3) 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。

4) 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。

规则及规则合并：

1） 规则：任何一个条件组合的特定取值及其相应要执行的操作称为规则。在判定表中贯穿条件项和动作项的一列就是一条规则。显然判定表中列出多少组条件取值，也就有多少条规则，既条件项和动作项有多少列。

2） 化简：就是规则合并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着极为相似的关系。

**判定表建立步骤：**

1） 确定规则的个数。假如有n个条件，每个条件有两个取值（0，1），故2n种规则。

2） 列出所有的条件桩和动作桩

3） 填入条件项

4） 填入动作项，等到初始判定表

5） 简化，合并相似规则（相同动作）

5、初始判定表化简。合并相似规则后得到

**实例2，NextData函数的精简决策表**

M1＝{月份， 每月有30天}

M2＝{月份， 每月有31天}

M3＝{月份， 2月} 有29＝512条规则

D1＝{日期，1～28} 12月末31日和其它31

D2＝{日期，29} 日月份的31日处理不同

D3＝{日期，30} 平年2月28日处理不同

D4＝{日期，31} 于2月27日

Y1 ＝{年：年是闰年}

Y2 ＝{年：年不是闰年}

**改进为：**

M1＝{月份： 每月有30天}

M2＝{月份： 每月有31天， 12月除外}

M4＝{月份：12月}

M3＝{月份： 2月}

D1＝{日期：1<=日期<=27}

D2＝{日期：28}

D3＝{日期：29}

D4＝{日期：30}

D5＝{日期：31}

Y1 ＝{年：年是闰年}

Y2 ＝{年：年不是闰年}

输入变量间存在大量逻辑关系的NextData决策表



3. 用决策表测试法测试以下程序：该程序有三个输入变量month、day、year（month、day和year均为整数值，并且满足：1≤month≤12和1≤day≤31），分别作为输入日期的月份、日、年份，通过程序可以输出该输入日期在日历上隔一天的日期。

例如，输入为2004年11月29日，则该程序的输出为2000年12月1日。

1) 分析各种输入情况，列出为输入变量month、day、year划分的有效等价类。

2) 分析程序规格说明，结合以上等价类划分的情况给出问题规定的可能采取的操作（即列出所有的动作桩）。

3) 根据（1）和（2），画出简化后的决策表。

案例分析如下：

Ø month变量的有效等价类：

M1: {month=4,6,9,11} M2: {month=1,3,5,7,8,10}

M3: {month=12 }M4: {month=2}

Ø day变量的有效等价类：

D1:{1≤day≤26} D2: {day=27} D3:   
{day=28} D4: {day=29} D5:   
{day=30} D6: {day=31}

Ø year变量的有效等价类：

Y1: {year是闰年} Y2: {year不是闰年}

4) 考虑各种有效的输入情况，程序中可能采取的操作有以下六种：

a1: day+2 a2: day=2 a3: day=1

a4: month+1 a5: month=1 a6: year+1

4. 判定表在功能测试中的应用

1) 一些软件的功能需求可用判定表表达得非常清楚，在检验程序的功能时判定表也就成为一个不错的工具。如果一个软件的规格说明指出：

Ø 当条件1和条件2满足，并且条件3和条件4不满足，或者当条件1、3和条件4满足时，要执行操作1。

Ø 在任一个条件都不满足时，要执行操作2。

Ø 在条件1不满足，而条件4被满足时，要执行操作3。 根据规格说明得到如下判定表：



这里，判定表只给出了16种规则中的8种。事实上，除这8条以外的一些规则是指当不能满足指定的条件，执行3种操作时，要执行1个默许的操作。在没必要时，判定表通常可略去这些规则。但如果用判定表来设计测试用例，就必须列出这些默许规则（如下表）。



2) 判定表的优点和缺点

Ø 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。

Ø 缺点：不能表达重复执行的动作，例如循环结构。

3) B. Beizer 指出了适合使用判定表设计测试用例的条件：

Ø 规格说明以判定表形式给出,或很容易转换成判定表。

Ø 条件的排列顺序不会也不影响执行哪些操作。

Ø 规则的排列顺序不会也不影响执行哪些操作。

Ø 每当某一规则的条件已经满足,并确定要执行的操作后,不必检验别的规则。

Ø 如果某一规则得到满足要执行多个操作,这些操作的执行顺序无关紧要。

B. Beizer提出这5个必要条件的目的是为了使操作的执行完全依赖于条件的组合。其实对于某些不满足这几条的判定表，同样可以借以设计测试用例，只不过尚需增加其它的测试用例罢了。

# （六）正交试验法

定义：从大量的（实验）数据（测试例）中挑选适量的,有代表性的点（例）,从而合理地安排实验（测试）的一种科学实验设计方法.类似的方法有:聚类分析方法,因子方法方法等.

利用正交实验设计测试用例的步骤：

1. 提取功能说明,构造因子--状态表

把影响实验指标的条件称为因子.而影响实验因子的条件叫因子的状态.利用正交实验设计方法来设计测试用例时,首先要根据被测试软件的规格说明书找出影响其功能实现的操作对象和外部因素,把他们当作因子,而把各个因子的取值当作状态.对软件需求规格说明中的功能要求进行划分,把整体的概要性的功能要求进行层层分解与展开,分解成具体的有相对独立性的基本的功能要求.这样就可以把被测试软件中所有的因子都确定下来,并为确定个因子的权值提供参考的依据.确定因子与状态是设计测试用例的关键.因此要求尽可能全面的正确的确定取值,以确保测试用例的设计作到完整与有效。

2. 加权筛选,生成因素分析表

对因子与状态的选择可按其重要程度分别加权.可根据各个因子及状态的作用大小,出现频率的大小以及测试的需要,确定权值的大小。

3. 利用正交表构造测试数据集

正交表的推导依据Galois理论（这里省略,需要时可查数理统计方面的教材）。

利用正交实验设计方法设计测试用例,比使用等价类划分,边界值分析,因果图等方法有以下优点:节省测试工作工时；可控制生成的测试用例数量；测试用例具有一定的覆盖率。

# （七）功能图法

定义：功能图由状态迁移图和布尔函数组成.状态迁移图用状态和迁移来描述.一个状态指出数据输入的位置（或时间）,而迁移则指明状态的改变.同时要依靠判定表或因果图表示的逻辑功能.例,一个简化的自动出纳机ATM的功能图。

应用：

1. 功能图介绍

一个程序的功能说明通常由动态说明和静态说明组成.动态说明描述了输入数据的次序或转移的次序.

静态说明描述了输入条件与输出条件之间的对应关系.对于较复杂的程序,由于存在大量的组合情况,因此,仅用静态说明组成的规格说明对于测试来说往往是不够的.必须用动态说明来补充功能说明.功能图方法是用功能图FD形式化地表示程序的功能说明,并机械地生成功能图的测试用例.

功能图模型由状态迁移图和逻辑功能模型构成.状态迁移图用于表示输入数据序列以及相应的输出数据.在状态迁移图中,由输入数据和当前状态决定输出数据和后续状态.逻辑功能模型用于表示在状态中输入条件和输出条件之间的对应关系.逻辑功能模型只适合于描述静态说明,输出数据仅由输入数据决定.测试用例则是由测试中经过的一系列状态和在每个状态中必须依靠输入/输出数据满足的一对条件组成.功能图方法其实是是一种黑盒白盒混合用例设计方法。

（功能图方法中,要用到逻辑覆盖和路径测试的概念和方法,其属白盒测试方法中 的内容.逻辑覆盖是以程序内部的逻辑结构为基础的测试用例设计方法.该方法要求测试人员对程序的逻辑结构有清楚的了解.由于覆盖测试的目标不同,逻辑覆盖可分为:语句覆盖,判定覆盖,判定-条件覆盖,条件组合覆盖及路径覆盖.下面我们指的逻辑覆盖和路径是功能或系统水平上的,以区别与白盒测试中的程序内部的.）

2. 测试用例生成方法

从功能图生成测试用例,得到的测试用例数是可接受的. 问题的关键的是如何从状态迁移图中选取测试用例. 若用节点代替状态,用弧线代替迁移,则状态迁移图就可转化成一个程序的控制流程图形式.问题就转化为程序的路径测试问题（如白盒测试）问题了.

3. 测试用例生成规则

为了把状态迁移（测试路径）的测试用例与逻辑模型（局部测试用例）的测试用例组合起来,从功能图生成实用的测试用例,须定义下面的规则.在一个结构化的状态迁移（SST）中,定义三种形式的循环:顺序,选择和重复.但分辨一个状态迁移中的所有循环是有困难的.（其表示图形省略）。

4. 从功能图生成测试用例的过程

1) 生成局部测试用例:在每个状态中,从因果图生成局部测试用例.局部测试用例由原因值（输入数据）组合与对应的结果值（输出数据或状态）构成。

2) 测试路径生成:利用上面的规则（三种）生成从初始状态到最后状态的测试路径。

3) 测试用例合成:合成测试路径与功能图中每个状态中的局部测试用例.结果是初始状态到最后状态的一个状态序列,以及每个状态中输入数据与对应输出数据的组合。

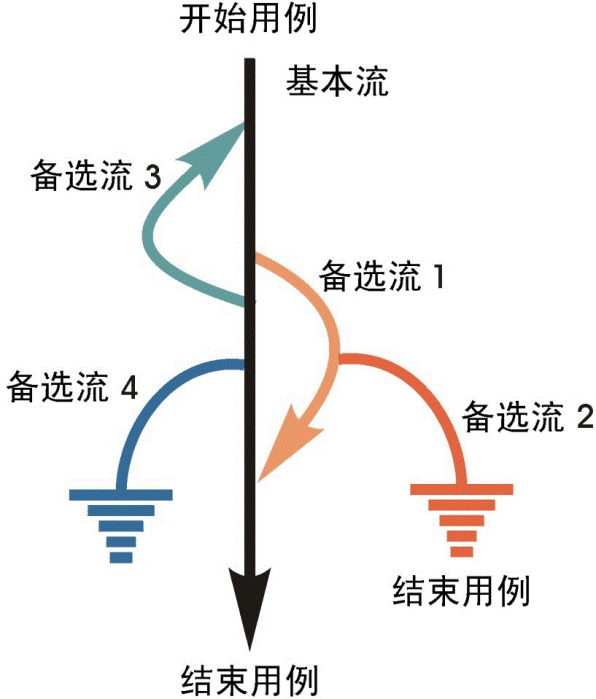
5. 测试用例的合成算法:采用条件构造树.

# （八）场景图法

定义：现在的软件几乎都是用事件触发来控制流程的，事件触发时的情景便形成了场景，而同一事件不同的触发顺序和处理结果就形成事件流。这种在软件设计方面的思想也可以引入到软件测试中，可以比较生动地描绘出事件触发时的情景，有利于测试设计者设计测试用例，同时使测试用例更容易理解和执行。

应用：

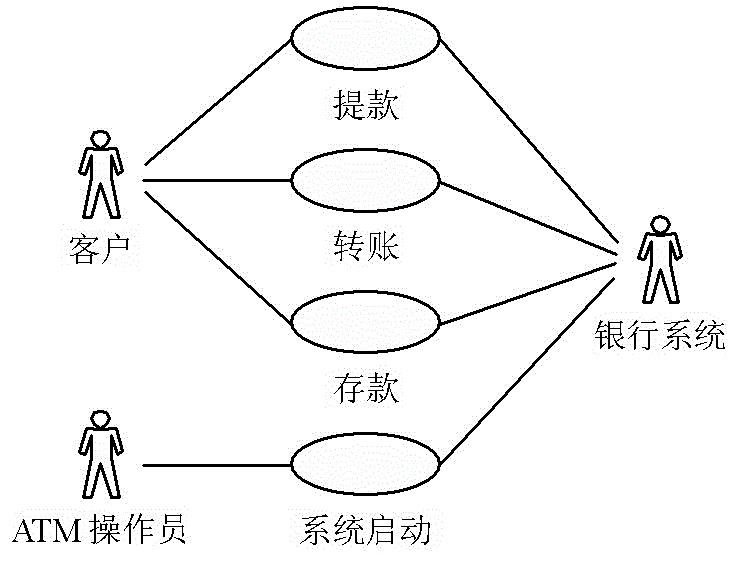
基本流和备选流：如下图所示，图中经过用例的每条路径都用基本流和备选流来表示，直黑线表示基本流，是经过用例的最简单的路径。备选流用不同的色彩表示，一个备选流可能从基本流开始，在某个特定条件下执行，然后重新加入基本流中（如备选流1和3）；也可能起源于另一个备选流（如备选流2），或者终止用例而不再重新加入到某个流（如备选流2和4）。



9.3. 实例

1. 例子描述

下图所示是ATM例子的流程示意图。



2. 场景设计：下表所示是生成的场景。

  
注：为方便起见，备选流3和6（场景3和7）内的循环以及循环组合未纳入上表。

3. 用例设计  
对于这7个场景中的每一个场景都需要确定测试用例。可以采用矩阵或决策表来确定和管理测试用例。下面显示了一种通用格式，其中各行代表各个测试用例，而各列则代表测试用例的信息。本示例中，对于每个测试用例，存在一个测试用例ID、条件（或说明）、测试用例中涉及的所有数据元素（作为输入或已经存在于[数据库](https://link.zhihu.com/?target=http://lib.csdn.net/base/mysql" \t "/Users/liuyun/Documents\\x/_blank)中）以及预期结果。  
表3-9 测试用例表



4. 数据设计  
一旦确定了所有的测试用例，则应对这些用例进行复审和验证以确保其准确且适度，并取消多余或等效的测试用例。  
测试用例一经认可，就可以确定实际数据值（在测试用例实施矩阵中）并且设定测试数据。

测试用例设计综合策略  
1. Myers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。  
2. 测试用例的设计步骤  
1) 构造根据设计规格得出的基本功能测试用例；  
2) 边界值测试用例；  
3) 状态转换测试用例；  
4) 错误猜测测试用例；  
5) 异常测试用例；  
6) 性能测试用例；  
7) 压力测试用例。  
3. 优化测试用例的方法  
1) 利用设计测试用例的8种方法不断的对测试用例进行分解与合并；  
2) 采用遗传算法理论进化测试用例；  
3) 在测试时利用发散思维构造测试用例；