第3章 黑盒测试

1．概述

黑盒测试又称功能测试或数据驱动测试。在测试时，把被测程序视为一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下进行。采用黑盒测试的目的主要是在已知软件产品所应具有的功能基础上进行，黑盒测试试图发现以下类型的错误：

（1）检查程序功能能否按需求规格说明书的规定正常使用，测试各个功能是否有遗漏，测试性能等特性是否满足；

（2）检测人机交互是否错误，检测数据结构或外部数据库访问是否错误，程序是否能适当地接收输入数据而产生正确的输出结果，并保持外部信息（如数据库或文件）的完整性；

（3）检测程序初始化和终止方面的错误。

2．教学重点与难点

1）重点

（1）等价类划分法设计测试用例；

（2）决策表法设计测试用例；

（3）因果图法设计测试用例；

（4）场景法设计测试用例。

2）难点

（1）等价类划分法设计测试用例；

（2）因果图法设计测试用例；

（3）各种方法的灵活运用。

3.1 等价类划分法

等价类划分法是一种典型的、重要的黑盒测试方法，它将程序所有可能的输入数据（有效的和无效的）划分成若干个等价类。然后从每个部分中选取具有代表性的数据当做测试用例进行合理的分类，测试用例由有效等价类和无效等价类的代表组成，从而保证测试用例具有完整性和代表性。利用这一方法设计测试用例可以不考虑程序的内部结构，以需求规格说明书为依据，选择适当的典型子集，认真分析和推敲说明书的各项需求，特别是功能需

求，尽可能多地发现错误。等价类划分法是一种系统性的确定要输入的测试条件的方法。

由于等价类是在需求规格说明书的基础上进行划分的，并且等价类划分不仅可以用来确定测试用例中的数据的输入输出的精确取值范围，也可以用来准备中间值、状态和与时间相关的数据以及接口参数等，所以等价类可以用在系统测试、集成测试和组件测试中，在有明确的条件和限制的情况下，利用等价类划分技术可以设计出完备的测试用例。这种方法可以减少设计一些不必要的测试用例，因为这种测试用例一般使用相同的等价类数据，从而使测试对象得到同样的反应行为。对于等价类我们从以下几个方面讨论它的划分方法。等价类划分的方法分为两个主要的步骤：划分等价类型和设计测试用例。

1. 有效等价类划分

有效等价类指对于程序规格说明来说，是合理的、有意义的输入数据构成的集合。利用有效等价类可以检验程序是否实现了规格说明预先规定的功能和性能。有效等价类可以是一个，也可以是多个，根据系统的输入域划分为若干部分，然后从每个部分中选取少数有代表性数据当做数据测试的测试用例，等价类是输入域的集合。有效等价类数据集包括：终端用户输入的命令，与最终用户交互的系统提示，接受相关的用户文件的名称，提供初始化值和边界等，提供格式化输出数据的命令，在图形模式（比如鼠标点击时）提供的数据，失败时显示的回应消息。

2. 无效等价类划分

无效等价类和有效等价类相反，无效等价类是指对于软件规格说明而言，没有意义的、不合理的输入数据集合。利用无效等价类，可以找出程序异常说明情况，检查程序的功能和性能的实现是否有不符合规格说明要求的地方。无效等价类数据集包括：在一个不正确的地方提供适当的值，验证边界值，验证外部边界的值，用户输入的命令，最终用户与系统交互的提示，验证与边界和外部边界值的数值数据。

3. 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。

4. 等价类划分的原则

（1）在输入条件规定的取值范围或值的个数的情况下，可以确定一个有效等价类和两个无效等价类。例如，输入值是学生成绩，范围是 0～100；如图 3.1 所示。

（2）在规定了输入数据的一组值中（假定有 n 个值），并且程序要对每个输入值分别处理的情况下，可以确定 n 个有效等价类和一个无效等价类。例如 ，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都作为无效等价类。

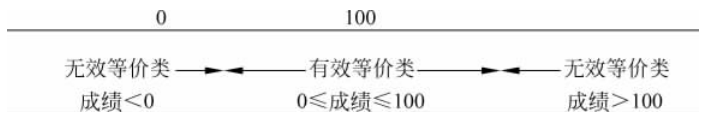

图 3.1 输入值是学生成绩的等价类划分

(3) 在规定输入数据必须遵守的规则的情况下,可以确定一个有效等价类和若干个无效等价类。

(4) 在输入条件规定了输入值的集合或规定了“必须如何”的条件下,可以确定一个有效等价类和一个无效等价类。

(5) 在确定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,则应将该等价类进一步地划分为更小的等价类。

5. 设计测试用例

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的有效等价类,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。

6. 函数 *F* 的功能扩展

函数 F 的功能扩展,即有两个变量 x_1 和 x_2 的函数 F。如果函数 F 实现为一个程序,则输入两个变量 x_1 和 x_2 会有一些(可能未规定)边界如图 3.2 所示。

$a \leqslant x_1 \leqslant d$ 区间为$[a,b)$,$[b,c)$,$[c,d]$

$e \leqslant x_2 \leqslant g$ 区间为$[e,f)$,$[f,g]$

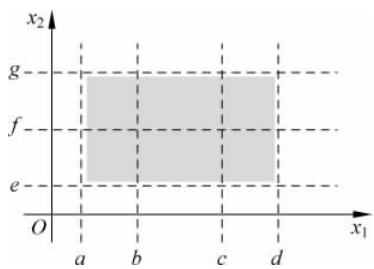

图 3.2 等价类区间划分

3.1.1 弱一般等价类测试

弱一般等价类测试通过使用一个测试用例中的每个等价类(区间)的一个变量实现,如图 3.3 所示。

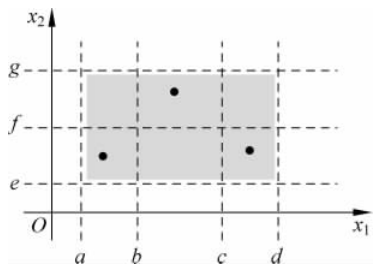

图 3.3 弱一般等价类测试用例图

这三个测试用例使用每个等价类中的一个值。我们以对称方式标识这些测试用例，于是得到外在的模式。事实上，永远都有等量的弱等价类测试用例，因为划分中的类对应最大子集数。

3.1.2 强一般等价类测试

强一般等价类测试基于多缺陷假设，因此需要等价类笛卡儿积的每个元素对应的测试用例，如图 3.4 所示。

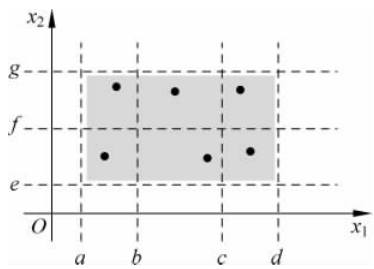

图 3.4 强一般等价类测试用例图

请注意，这些测试用例的模式与命题逻辑中的真值表构造具有相似性。笛卡儿积可保证两种意义上的“完备性”：一是覆盖所有的等价类，二是有可能输入组合中的一个。

3.1.3 弱健壮等价类测试

这种测试的名称显然与直觉矛盾，并且自相矛盾。怎么能既弱又健壮呢？说它健壮，是因为这种测试考虑了无效值；说它弱，是因为有单缺陷假设。

对于有效输入，使用每个有效类的一个值（就像我们在所谓弱一般等价类测试中所做的一样。请注意，这些测试用例中的所有输入都是有效的）。

对于无效输入，测试用例将拥有一个无效值，并保持其余的值都是有效的（因此，“单缺陷”会造成测试用例失败）。

按照这种策略产生的测试用例如图 3.5 所示。

弱健壮等价类测试有两个问题：第一个问题是，规格说明常常并没有定义无效测试用例所预期的输出是什么；第二个问题是，强类型语言没有必要考虑无效输入。

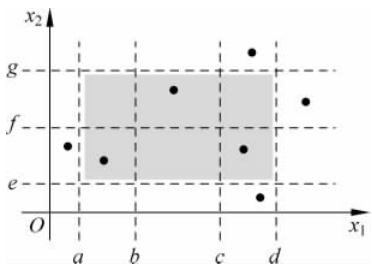

图 3.5 弱健壮等价类测试用例图

3.1.4 强健壮等价类测试

我们从所有等价类笛卡儿积的每个元素中获得测试用例，如图 3.6 所示。

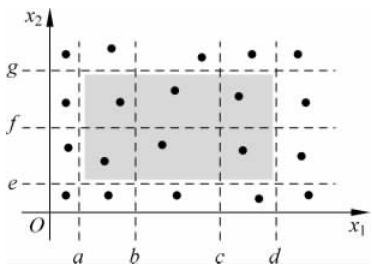

图 3.6 强健壮等价类测试用例

3.1.5 单元实践

1. 三角形问题的等价类测试用例

四种可能出现的输出：非三角形、不等边三角形、等腰三角形和等边三角形。

可以使用这些输出标识如下所示的输出(值域)等价类：$R1=\{\langle a,b,c\rangle$：有三条边 a、b 和 c 的等边三角形$\}$，$R2=\{\langle a,b,c\rangle$：有三条边 a、b 和 c 的等腰三角形$\}$，$R3=\{\langle a,b,c\rangle$：有三条边 a、b 和 c 的不等边三角形$\}$，$R4=\{\langle a,b,c\rangle$：三条边 a、b 和 c 不构成三角形$\}$。

四个弱一般等价类测试用例如表 3.1 所示。

表 3.1 四个弱一般等价类测试用例

测试用例	*a*	*b*	*c*	预期输出
WN1	5	5	5	等边三角形
WN2	2	2	3	等腰三角形
WN3	3	4	5	不等边三角形
WN4	4	1	2	非三角形

由于变量 a、b 和 c 没有有效区间，则强一般等价类测试用例与弱一般等价类测试用例相同。

考虑 a、b 和 c 的无效值产生的额外弱健壮等价类测试用例，如表 3.2 所示。

表 3.2 额外弱健壮等价类测试用例

测试用例	a	b	c	预期输出
WR1	−1	5	5	a 取值不在所允许的取值值域内
WR2	5	−1	5	b 取值不在所允许的取值值域内
WR3	5	5	−1	c 取值不在所允许的取值值域内
WR4	201	5	5	a 取值不在所允许的取值值域内
WR5	5	201	5	b 取值不在所允许的取值值域内
WR6	5	5	201	c 取值不在所允许的取值值域内

表 3.3 所示是额外强健壮性等价类测试用例三维立方的一个“角”。

表 3.3 额外强健壮性等价类测试用例

测试用例	a	b	c	预期输出
SR1	−1	5	5	a 取值不在所允许的取值值域内
SR2	5	−1	5	b 取值不在所允许的取值值域内
SR3	5	5	−1	c 取值不在所允许的取值值域内
SR4	−1	−1	5	a、b 取值不在所允许的取值值域内
SR5	5	−1	−1	b、c 取值不在所允许的取值值域内
SR6	−1	5	−1	a、c 取值不在所允许的取值值域内
SR7	−1	−1	−1	a、b、c 取值不在所允许的取值值域内

请注意，预期输出如何完备地描述无效输入值。

等价类测试显然对用来定义类的等价关系很敏感。如果在输入定义域上定义等价类，则可以得到更丰富的测试用例集合。三个整数 a、b、c 有些什么可能的取值？这些整数可以都相等，有一对整数相等（有三种相等方式），或都不相等。

$D1=\{\langle a,b,c\rangle: a=b=c\}$

$D2=\{\langle a,b,c\rangle: a=b, a\neq c\}$

$D3=\{\langle a,b,c\rangle: a=c, a\neq b\}$

$D4=\{\langle a,b,c\rangle: c=b, a\neq c\}$

$D5=\{\langle a,b,c\rangle: b\neq a\neq c\}$

作为一个单独的问题，我们可以通过三角形的性质来判断三条边是否构成一个三角形。

$D6=\{\langle a,b,c\rangle: a\geqslant b+c\}$

$D7=\{\langle a,b,c\rangle: b\geqslant a+c\}$

$D8=\{\langle a,b,c\rangle: c\geqslant a+b\}$

2. NextDate 函数的等价类测试用例

NextDate 是一个三变量函数，即月份、日期和年，这些变量的有效值区间定义如下：

$M1=\{$月份：$1\leqslant$月份$\leqslant 12\}$

$D1=\{$日期：$1\leqslant$日期$\leqslant 31\}$

$Y1=\{$年：$1812\leqslant$年$\leqslant 2012\}$

无效等价类：

$M2$＝{月份：月份＜1}

$M3$＝{月份：月份＞12}

$D2$＝{日期：日期＜1}

$D3$＝{日期：日期＞31}

$Y2$＝{年：年＜1812}

$Y3$＝{年：年＞2012}

由于有效类的数量等于独立变量的个数，因此只有弱一般等价类测试用例出现，并且与强一般等价类测试用例相同，如表 3.4 所示。

表 3.4 测试用例表

用例 ID	月份	日期	年	预期输出
WN1,SN1	6	15	1912	1912 年 6 月 16 日

表 3.5 所示是弱健壮测试用例的完整集合。

表 3.5 弱健壮测试用例

用例 ID	月份	日期	年	预 期 输 出
WR1	6	15	1912	1912 年 6 月 16 日
WR2	－1	15	1912	月份不在有效值域 1..12 中
WR3	13	15	1912	月份不在有效值域 1..12 中
WR4	6	－1	1912	日期不在有效值域 1..31 中
WR5	6	32	1912	日期不在有效值域 1..31 中
WR6	6	15	1811	年不在有效值域 1812..2012 中
WR7	6	15	2013	年不在有效值域 1812..2012 中

与三角形问题一样，表 3.6 所示是额外强健壮等价类测试用例三维立方的一个“角”。

表 3.6 额外强健壮等价类测试用例

用例 ID	月份	日期	年	预 期 输 出
SR1	－1	15	1912	月份不在有效值域 1..12 中
SR2	6	－1	1912	日期不在有效值域 1..31 中
SR3	6	15	1811	年不在有效值域 1812..2012 中
SR4	－1	－1	1912	月份不在有效值域 1..12 中 日期不在有效值域 1..31 中
SR5	6	－1	1811	日期不在有效值域 1..31 中 年不在有效值域 1812..2012 中
SR6	－1	15	1811	月份不在有效值域 1..12 中 年不在有效值域 1812..2012 中
SR7	－1	－1	1811	月份不在有效值域 1..12 中 日期不在有效值域 1..31 中 年不在有效值域 1812..2012 中

如果更仔细地选择等价关系，所得到的等价类将会更有用。前面曾经提到过，等价关系的要点是，类中的元素要被“同样处理”。理解传统方法不足的一种方法，是注意到“处理”在

有效/无效层次上进行。通过关注更具体的处理可降低粒度。

如果它不是某个月的最后一天，则 NextDate 函数会直接对日期加 1。到了月末，下一个日期是 1，月份加 1。到了年末，日期和月份都会复位到 1，年加 1。最后，闰年问题要确定有关的月份的最后一天。经过这些分析，可以假设等价类：

$M1$＝{月份：每月有 30 天}

$M2$＝{月份：每月有 31 天}

$M3$＝{月份：此月是 2 月}

$D1$＝{日期：1 ≤日期≤28}

$D2$＝{日期：日期＝29}

$D3$＝{日期：日期＝30}

$D4$＝{日期：日期＝31}

$Y1$＝{年：年＝2000}

$Y2$＝{年：年是闰年}

$Y3$＝{年：年是平年}

通过选择有 30 天的月份和有 31 天的月份的独立类，可以简化月份最后一天问题。通过把 2 月份分成独立的类，可以对闰年问题给予更多关注。

一个弱一般等价类测试用例如表 3.7 所示。

表 3.7 弱一般等价类测试用例

用例 ID	月份	日期	年	预 期 输 出
WN1	6	15	2000	2000.6.16
WN2	7	29	1996	1996.7.30
WN3	2	30	2002	输入日期不正确
WN4	6	31	2000	输入日期不正确

经过改进的强一般等价类测试用例如表 3.8 所示。

表 3.8 强一般等价类测试用例

用例 ID	月份	日期	年	预 期 输 出
Test1	6	15	2000	2000.6.16
Test2	6	15	1996	1996.6.15
Test3	6	14	2002	2002.6.15
Test4	6	29	2000	2000.6.30
Test5	6	29	1996	1996.6.30
Test6	6	29	2002	2002.6.30
Test7	6	30	2000	2000.6.31（不可能的日期）
Test8	6	30	1996	1996.6.31（不可能的日期）
Test9	6	30	2002	2002.6.31（不可能的日期）
Test10	6	31	2000	2000.7.1(无效输入日期)
Test11	6	31	1996	1996.7.1(无效输入日期)
Test12	6	31	2002	2002.7.1（无效输入日期）
Test13	7	15	2000	2000.7.15

续表

用例 ID	月份	日期	年	预 期 输 出
Test14	7	14	1996	1996.7.15
Test15	7	14	2002	2002.7.15
Test16	7	29	2000	2000.7.30
Test17	7	29	1996	1996.7.30
Test18	7	29	2002	2002.7.30
Test19	7	30	2000	2000.7.31
Test20	7	30	1996	1996.7.31
Test21	7	30	2002	2004.7.31
Test22	7	31	2000	2000.8.1
Test23	7	31	1996	1996.8.1
Test24	7	31	2002	2002.8.1
Test25	2	14	2000	2000.2.15
Test26	2	14	1996	1996.2.15
Test27	2	14	2002	2002.2.15
Test28	2	29	2000	2000.3.1(无效的输入日期)
Test29	2	29	1996	1996.3.1
Test30	2	29	2002	2002.3.1（不可能的日期）
Test31	2	30	2000	2000.3.1(无效输入日期)
Test32	2	30	1996	1996.3.1(无效输入日期)
Test33	2	30	2002	2002.3.1(无效输入日期)
Test34	6	31	2000	2000.7.1(无效输入日期)
Test35	6	31	1996	1996.7.1(无效输入日期)
Test36	6	31	2002	2002.7.1(无效输入日期)

从弱一般测试转向强一般测试会产生一些冗余。从弱到强的转换，不管是一般类还是健壮类，都要做独立性假设，都要以等价类的叉积表示。3 个月份类乘以 4 个日期类乘以 3 个年类，产生 36 个强一般等价类测试用例。

3. 佣金问题的等价类测试用例

佣金问题的输入定义域，由于枪机、枪托和枪管的限制而被“自然地”划分。这些等价类也正是通过传统等价类测试所标识的等价类。

输入变量有效类是：

$L1=\{$枪机：$1\leqslant$枪机$\leqslant 70\}$

$L2=\{$枪机$=-1\}$

$S1=\{$枪托：$1\leqslant$枪托$\leqslant 80\}$

$B1=\{$枪管：$1\leqslant$枪管$\leqslant 90\}$

输入变量对应的无效类是：

$L3=\{$枪机：枪机$=0$ 或枪机$<-1\}$

$L3=\{$枪机：枪机$>70\}$

$S2=\{$枪托：枪托$<1\}$

$S3$={枪托：枪托>80}

$B2$={枪管：枪管<1}

$B3$={枪管：枪管>90}

但是有一个问题，变量枪机还用做指示不再有电报的标记。当枪机等于－1 时，While 循环就会终止，总枪机、总枪托和总枪管的值就会被用来计算销售额，进而计算佣金。

除了变量的名称和端点值区间不同之外，与 NextDate 函数的第一个版本完全相同。因此，也只有一个弱一般等价类测试用例，这个测试用例同样也等于强一般等价类测试用例。同样也有 7 个弱健壮测试用例。最后，额外弱健壮等价类测试用例三维立方的一个“角”如表 3.9 所示。

表 3.9 弱健壮测试用例

用例 ID	枪机	枪托	枪管	预 期 输 出
SR1	－1	40	45	枪机值不在有效值域 1..70 中
SR2	35	－1	45	枪托值不在有效值域 1..80 中
SR3	35	40	－1	枪管值不在有效值域 1..90 中
SR4	－1	－1	45	枪机值不在有效值域 1..70 中 枪托值不在有效值域 1..80 中
SR5	－1	40	－1	枪机值不在有效值域 1..70 中 枪管值不在有效值域 1..90 中
SR6	35	－1	－1	枪托值不在有效值域 1..80 中 枪管值不在有效值域 1..90 中
SR7	－1	－1	－1	枪机值不在有效值域 1..70 中 枪托值不在有效值域 1..80 中 枪管值不在有效值域 1..90 中

销售额是所售出的枪机、枪托和枪管数量的函数：

销售额＝45×枪机＋30×枪托＋25×枪管

我们可以根据佣金值域定义三个变量的等价类：

$S1$={〈枪机，枪托，枪管〉：销售额≤1000}

$S2$={〈枪机，枪托，枪管〉：1000<销售额≤1800}

$S3$={〈枪机，枪托，枪管〉：销售额>1800}

$S1$ 元素是接近原点金字塔中的整数点，$S2$ 的元素是金字塔与其他输入空间之间的“三角片”，$S3$ 的元素是不在 $S1$ 和 $S2$ 中的立方体中的点。输入定义域的强等价类所发现的错误案例都在立方体之外，如表 3.10 所示。

表 3.10 输出值域等价类测试用例

测试用例	枪机	枪托	枪管	销售额/美元	佣金/美元
OR1	5	5	5	500	50
OR2	15	15	15	1500	175
OR3	25	25	25	2500	360

等价类划分总结如表 3.11 所示。

表 3.11 等价类划分总结

划 分 法	特 点
弱一般等价类测试	不考虑无效等价类，选取的测试用例只需覆盖到有效等价类
强一般等价类测试	不考虑无效等价类，选取测试用例时，要根据等价类笛卡儿积，各有效区间的组合都要覆盖到
弱健壮等价类测试	基于单缺陷假设，考虑无效等价类，选取的测试用例要覆盖每一个有效等价类和无效等价类，但是不能同时覆盖两个无效等价类
强健壮等价类测试	每个无效等价类和有效等价类的组合都要覆盖到，考虑所有的有效和无效情况

我们已经介绍了三个例子，最后讨论关于等价类测试的一些观察和等价类测试指导方针。

(1) 显然，等价类测试的弱形式(一般或健壮)不如对应的强形式的测试全面。

(2) 如果实现语言是强类型的(无效值会引起运行时错误)，则没有必要使用健壮形式的测试。

(3) 如果错误条件非常重要，则进行健壮形式的测试是合适的。

(4) 如果输入数据以离散值区间和集合定义，则等价类测试是合适的。当然也适用于变量值越界就会出现故障的系统。

(5) 通过结合边界值测试，等价类测试可得到加强(我们可以“重用”定义等价类的工作成果)。

(6) 如果程序函数很复杂，则函数的复杂性可以帮助标识有用的等价类，就像 NextDate 函数一样。

(7) 强等价类测试假设变量是独立的，相应的测试用例相乘会引起冗余问题。如果存在依赖关系，则常常会生成“错误”测试用例，就像 NextDate 函数一样。

(8) 在发现“合适”的等价关系之前，可能需要进行多次尝试，就像 NextDate 函数例子一样，在其他情况下，存在“明显”或“自然”等价关系。如果不能肯定，最好对任何合理的实现进行再次预测。

(9) 强和弱形式的等价类测试之间的差别，有助于区分累进测试和回归测试。

3.2 边界值测试

3.2.1 边界值分析

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。

以下讨论涉及有两个变量 x 和 y 的函数 F。如果函数 F 实现为一个程序，则输入两个变量 x 和 y 会有一些(可能未规定)边界：

$$a \leqslant x \leqslant b$$
$$c \leqslant y \leqslant d$$

但是，区间[a,b]和[c,d]是 x 和 y 的值域，带阴影矩形中的任何点都是函数 F 的有效输入。如图 3.7 所示。

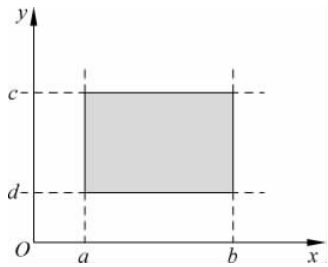

图 3.7 两变量函数的输入定义域

边界值分析关注的是输入空间的边界，来标识测试用例。边界值测试背后的基本原理是错误更可能出现在输入变量的极值附近。

边界值分析的基本思想是使用在最小值、略高于最小值、正常值、略低于最大值和最大值处取输入变量值。

边界值分析的下一个部分基于一种关键假设，在可靠性理论中叫做“单缺陷”假设。这种假设是说，失效极少是由两个(或多个)缺陷的同时发生引起的。因此，边界值分析测试用例的获得，通过使所有变量取正常值，只使一个变量取极值。两变量函数 F(如图 3.8 所示)的边界值分析测试用例是：

{<xnom,ymin>;<xnom,ymin+>;<xnom,ymax>;<xnom,ymax->;<xmin,ynom>;<xmin+,ynom>;<xmax,ynom>;<xmax-,ynom>;<xnom,ynom>;}

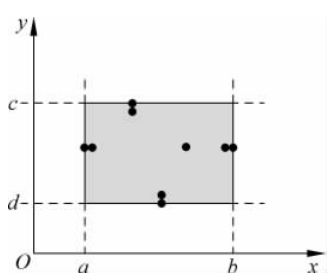

图 3.8 两变量函数边界值分析测试用例

基本边界值分析手段可以用两种方式归纳：通过变量数量和通过值域的种类。归纳变量数量很容易：如果有一个 n 变量函数，使除一个以外的所有变量取正常值，使剩余的那个变量取最小值、略高于最小值、正常值、略低于最大值和最大值，对每个变量都重复进行，这样，对于一个 n 变量函数，边界值分析会产生 $4n+1$ 个测试用例。

归纳值域取决于变量本身的性质(或更准确地说是类型)。例如，对于 NextDate 函数，我们有月份、日期和年对应的变量。采用类似于 FORTRAN 的语言，很有可能对这些变量编码，使得 1 月对应 1，2 月对应 2，等等。采用支持用户定义类型的语言(如 Pascal 或 Ada)，可以把变量“月份”定义为枚举类型{1 月，2 月，……，12 月}。不管采用什么语言，根据语境可以很清楚地确定最小值、略高于最小值、正常值、略低于最大值和最大值。如果变量具有离散、有界值，例如佣金问题中的变量，则最小值、略高于最小值、正常值、略低于最大

值和最大值也可以很容易地确定。如果没有显式地给出边界，例如三角形问题，通常必须创建一种“人工”边界。边长的最低值显然是 1，但是上限怎样确定呢？在默认情况下，最大可表示整数（在某些语言中叫做 MAXINT）是一种可能，也可以任意规定上限，例如 200 或 2000。

边界值分析对布尔变量没有什么意义，极值是 TRUE 和 FALSE，但是其余三个值不明确，布尔变量可以进行基于决策表的测试。逻辑变量也代表一种边界值分析。

如果被测程序是多个独立变量的函数，这些变量受物理量的限制，则很适合边界值分析。简单看一下 NextDate 的边界值分析测试用例，就会发现这些测试用例是不充分的。例如，没怎么强调 2 月和闰年。这里的真正问题是，月份、日期和年变量之间存在有意思的依赖关系。边界值分析假设变量是完全独立的。即便如此，边界值分析也能够捕获月末和年末缺陷。边界值分析测试用例通过引用物理量的边界独立变量极值导出，不考虑函数的性质，也不考虑变量的语义含义。

3.2.2 健壮性测试

健壮性测试是边界分析的一种简单扩展：除了变量的五个边界值分析取值，还要通过采用一个略超过最大值（max+）的取值，以及一个略小于最小值（min−）的取值，看看超过极值时系统会有什么表现。健壮性测试最有意义的部分不是输入，而是预期的输出，如图 3.9 所示。

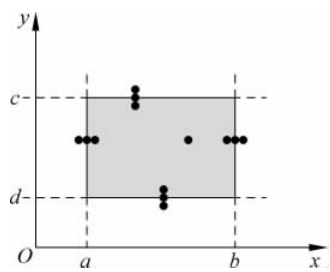

图 3.9 两变量函数的健壮性测试用例图

对于一个含有 n 个变量的程序，保留其中一个变量，让其余的变量取正常值，被保留的变量依次取 min、min+、min−、nom、max−、max、max+值，对每个变量都重复进行。这样，对于一个有 n 个变量的程序，边界值分析测试程序会产生 $6n+1$ 个测试用例。

3.2.3 最坏情况测试

在拒绝“单缺陷假设”理论的情况下，对所有变量的边界值集合进行 5 元素笛卡儿积计算，用于生成测试用例，对于 n 变量函数的最坏测试基于边界值分析会产生 $5n$ 个测试用例，基于健壮性分析则产生 $7n$ 个测试用例。相比而言，最坏情况测试代价较高，因此其最佳应用场合是物理变量具有大量交互作用，或者函数失效的代价极高的情况下。

最坏情况测试显然更彻底，因为边界值分析测试是最坏情况测试用例的真子集。最坏情况测试还意味着更多的工作量：n 变量函数的最坏情况测试，会产生 5^n 个测试用例，而边界值分析只产生 $4n+1$ 个测试用例，如图 3.10 所示。

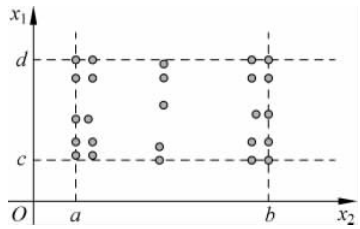

图 3.10 两变量函数的最坏情况测试用例图

最坏情况测试的归纳模式与边界值分析的归纳模式一样，两者也有相同的局限性，特别是独立性要求方面的局限性。最坏情况测试的最佳应用场合，可能是物理变量具有大量交互作用，或者函数失效的代价极高的情况。对于确实极端的测试，会采用健壮最坏情况测试。这种测试使用健壮性测试的 7 元素集合的笛卡儿积，如图 3.11 所示。

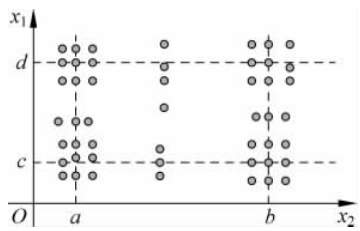

图 3.11 两变量函数的健壮最坏情况测试用例图

3.2.4 单元实践

1. 三角形问题的测试用例

关于三角形的边，除了说明是整数外，没有给出其他条件。我们任意取 200 作为高端边界，表 3.12 是边界值分析测试用例，表 3.13 是最坏情况测试用例。

表 3.12 边界值分析测试用例

用 例	*a*	*b*	*c*	预 期 输 出
1	100	100	1	等腰三角形
2	100	100	2	等腰三角形
3	100	100	100	等边三角形
4	100	100	199	等腰三角形
5	100	100	200	非三角形
6	100	1	100	等腰三角形
7	100	2	100	等腰三角形
8	100	100	100	等边三角形
9	100	199	100	等腰三角形
10	100	200	100	非三角形
11	1	100	100	等腰三角形
12	2	100	100	等腰三角形
13	100	100	100	等边三角形

续表

用　例	*a*	*b*	*c*	预期输出
14	199	100	100	等腰三角形
15	200	100	100	非三角形

表 3.13　最坏情况测试用例

用　例	*a*	*b*	*c*	预期输出
1	1	1	1	等边三角形
2	1	1	2	非三角形
3	1	1	100	非三角形
4	1	1	199	非三角形
5	1	1	200	非三角形
6	1	2	1	非三角形
7	1	2	2	等腰三角形
8	1	2	100	非三角形
9	1	2	199	非三角形
10	1	2	200	非三角形
11	1	100	1	非三角形
12	1	100	2	非三角形
13	1	100	100	等腰三角形
14	1	100	199	非三角形
15	1	100	200	非三角形
16	1	199	1	非三角形
17	1	199	2	非三角形
18	1	199	100	非三角形
19	1	199	199	等腰三角形
20	1	199	200	非三角形
21	1	200	1	非三角形
22	1	200	2	非三角形
23	1	200	100	非三角形
24	1	200	199	非三角形
25	1	200	200	等腰三角形
26	2	1	1	非三角形
27	2	1	2	等腰三角形
28	2	1	100	非三角形
29	2	1	199	非三角形
30	2	1	200	非三角形
31	2	2	1	等腰三角形
32	2	2	2	等边三角形
33	2	2	100	非三角形
34	2	2	199	非三角形
35	2	2	200	非三角形
36	2	100	1	非三角形
37	2	100	2	非三角形

续表

用 例	*a*	*b*	*c*	预 期 输 出
38	2	100	100	等腰三角形
39	2	100	199	非三角形
40	2	100	200	非三角形
41	2	199	1	非三角形
42	2	199	2	非三角形
43	2	199	100	非三角形
44	2	199	199	等腰三角形
45	2	199	200	不等边三角形
46	2	200	1	非三角形
47	2	200	2	非三角形
48	2	200	100	非三角形
49	2	200	199	不等边三角形
50	2	200	200	等腰三角形
51	100	1	1	非三角形
52	100	1	2	非三角形
53	100	1	100	等腰三角形
54	100	1	199	非三角形
55	100	1	200	非三角形
56	100	2	1	非三角形
57	100	2	2	非三角形
58	100	2	100	等腰三角形
59	100	2	199	非三角形
60	100	2	200	非三角形
61	100	100	1	等腰三角形
62	100	100	2	等腰三角形
63	100	100	100	等边三角形
64	100	100	199	等腰三角形
65	100	100	200	非三角形
66	100	199	1	非三角形
67	100	199	2	非三角形
68	100	199	100	等腰三角形
69	100	199	199	等腰三角形
70	100	199	200	不等边三角形
71	100	200	1	非三角形
72	100	200	2	非三角形
73	100	200	100	非三角形
74	100	200	199	不等边三角形
75	100	200	200	等腰三角形
76	199	1	1	非三角形
77	199	1	2	非三角形
78	199	1	100	非三角形
79	199	1	199	等腰三角形

续表

用　例	*a*	*b*	*c*	预 期 输 出
80	199	1	200	非三角形
81	199	2	1	非三角形
82	199	2	2	非三角形
83	199	2	100	非三角形
84	199	2	199	等腰三角形
85	199	2	200	不等边三角形
86	199	100	1	非三角形
87	199	100	2	非三角形
88	199	100	100	等腰三角形
89	199	100	199	等腰三角形
90	199	100	200	不等边三角形
91	199	199	1	等腰三角形
92	199	199	2	等腰三角形
93	199	199	100	等腰三角形
94	199	199	199	等边三角形
95	199	199	200	等腰三角形
96	199	200	1	非三角形
97	199	200	2	不等边三角形
98	199	200	100	不等边三角形
99	199	200	199	等腰三角形
100	199	200	200	等腰三角形
101	200	1	1	非三角形
102	200	1	2	非三角形
103	200	1	100	非三角形
104	200	1	199	非三角形
105	200	1	200	等腰三角形
106	200	2	1	非三角形
107	200	2	2	非三角形
108	200	2	100	非三角形
109	200	2	199	不等边三角形
110	200	2	200	等腰三角形
111	200	100	1	非三角形
112	200	100	2	非三角形
113	200	100	100	非三角形
114	200	100	199	不等边三角形
115	200	100	200	等腰三角形
116	200	199	1	非三角形
117	200	199	2	不等边三角形
118	200	199	100	不等边三角形
119	200	199	199	等腰三角形
120	200	199	200	等腰三角形
121	200	200	1	等腰三角形

续表

用 例	a	b	c	预 期 输 出
122	200	200	2	等腰三角形
123	200	200	100	等腰三角形
124	200	200	199	等腰三角形
125	200	200	200	等边三角形

2. NextDate 函数的测试用例

表 3.14 所示是 NextDate 函数的最坏情况测试用例。

表 3.14 最坏情况测试用例

用 例	月 份	日 期	年	预 期 输 出
1	1	1	1812	1812 年 1 月 2 日
2	1	1	1813	1813 年 1 月 2 日
3	1	1	1912	1912 年 1 月 2 日
4	1	1	2011	2011 年 1 月 2 日
5	1	1	2012	2012 年 1 月 2 日
6	1	2	1812	1812 年 1 月 3 日
7	1	2	1813	1813 年 1 月 3 日
8	1	2	1912	1912 年 1 月 3 日
9	1	2	2011	2011 年 1 月 3 日
10	1	2	2012	2012 年 1 月 3 日
11	1	15	1812	1812 年 1 月 16 日
12	1	15	1813	1813 年 1 月 16 日
13	1	15	1912	1912 年 1 月 16 日
14	1	15	2011	2011 年 1 月 16 日
15	1	15	2012	2012 年 1 月 16 日
16	1	30	1812	1812 年 1 月 31 日
17	1	30	1813	1813 年 1 月 31 日
18	1	30	1912	1912 年 1 月 31 日
19	1	30	2011	2011 年 1 月 31 日
20	1	30	2012	2012 年 1 月 31 日
21	1	31	1812	1812 年 2 月 1 日
22	1	31	1813	1813 年 2 月 1 日
23	1	31	1912	1912 年 2 月 1 日
24	1	31	2011	2011 年 2 月 1 日
25	1	31	2012	2012 年 2 月 1 日
26	2	1	1812	1812 年 2 月 2 日
27	2	1	1813	1813 年 2 月 2 日
28	2	1	1912	1912 年 2 月 2 日
29	2	1	2011	2011 年 2 月 2 日
30	2	1	2012	2012 年 2 月 2 日
31	2	2	1812	1812 年 2 月 3 日

续表

用　例	月　份	日　期	年	预 期 输 出
32	2	2	1813	1813 年 2 月 3 日
33	2	2	1912	1912 年 2 月 3 日
34	2	2	2011	2011 年 2 月 3 日
35	2	2	2012	2012 年 2 月 3 日
36	2	15	1812	1812 年 2 月 16 日
37	2	15	1813	1813 年 2 月 16 日
38	2	15	1912	1912 年 2 月 16 日
39	2	15	2011	2011 年 2 月 16 日
40	2	15	2012	2012 年 2 月 16 日
41	2	30	1812	错误
42	2	30	1813	错误
43	2	30	1912	错误
44	2	30	2011	错误
45	2	30	2012	错误
46	2	31	1812	错误
47	2	31	1813	错误
48	2	31	1912	错误
49	2	31	2011	错误
50	2	31	2012	错误
51	6	1	1812	1812 年 6 月 2 日
52	6	1	1813	1813 年 6 月 2 日
53	6	1	1912	1912 年 6 月 2 日
54	6	1	2011	2011 年 6 月 2 日
55	6	1	2012	2012 年 6 月 2 日
56	6	2	1812	1812 年 6 月 3 日
57	6	2	1813	1813 年 6 月 3 日
58	6	2	1912	1912 年 6 月 3 日
59	6	2	2011	2011 年 6 月 3 日
60	6	2	2012	2012 年 6 月 3 日
61	6	15	1812	1812 年 6 月 16 日
62	6	15	1813	1813 年 6 月 16 日
63	6	15	1912	1912 年 6 月 16 日
64	6	15	2011	2011 年 6 月 16 日
65	6	15	2012	2012 年 6 月 16 日
66	6	30	1812	1812 年 7 月 31 日
67	6	30	1813	1813 年 7 月 31 日
68	6	30	1912	1912 年 7 月 31 日
69	6	30	2011	2011 年 7 月 31 日
70	6	30	2012	2012 年 7 月 31 日
71	6	31	1812	错误
72	6	31	1813	错误
73	6	31	1912	错误

续表

用 例	月 份	日 期	年	预 期 输 出
74	6	31	2011	错误
75	6	31	2012	错误
76	11	1	1812	1812 年 11 月 2 日
77	11	1	1813	1813 年 11 月 2 日
78	11	1	1912	1912 年 11 月 2 日
79	11	1	2011	2011 年 11 月 2 日
80	11	1	2012	2012 年 11 月 2 日
81	11	2	1812	1812 年 11 月 3 日
82	11	2	1813	1813 年 11 月 3 日
83	11	2	1912	1912 年 11 月 3 日
84	11	2	2011	2011 年 11 月 3 日
85	11	2	2012	2012 年 11 月 3 日
86	11	15	1812	1812 年 11 月 16 日
87	11	15	1813	1813 年 11 月 16 日
88	11	15	1912	1912 年 11 月 16 日
89	11	15	2011	2011 年 11 月 16 日
90	11	15	2012	2012 年 11 月 16 日
91	11	30	1812	1812 年 12 月 1 日
92	11	30	1813	1813 年 12 月 1 日
93	11	30	1912	1912 年 12 月 1 日
94	11	30	2011	2011 年 12 月 1 日
95	11	30	2012	2012 年 12 月 1 日
96	11	31	1812	错误
97	11	31	1813	错误
98	11	31	1912	错误
99	11	31	2011	错误
100	11	31	2012	错误
101	12	1	1812	1812 年 12 月 2 日
102	12	1	1813	1813 年 12 月 2 日
103	12	1	1912	1912 年 12 月 2 日
104	12	1	2011	2011 年 12 月 2 日
105	12	1	2012	2012 年 12 月 2 日
106	12	2	1812	1812 年 12 月 3 日
107	12	2	1813	1813 年 12 月 3 日
108	12	2	1912	1912 年 12 月 3 日
109	12	2	2011	2011 年 12 月 3 日
110	12	2	2012	2012 年 12 月 3 日
111	12	15	1812	1812 年 12 月 16 日
112	12	15	1813	1813 年 12 月 16 日
113	12	15	1912	1912 年 12 月 16 日
114	12	15	2011	2011 年 12 月 16 日
115	12	15	2012	2012 年 12 月 16 日

续表

用　例	月　份	日　期	年	预 期 输 出
116	12	30	1812	1812 年 12 月 31 日
117	12	30	1813	1813 年 12 月 31 日
118	12	30	1912	1912 年 12 月 31 日
119	12	30	2011	2011 年 12 月 31 日
120	12	30	2012	2012 年 12 月 31 日
121	12	31	1812	1813 年 1 月 1 日
122	12	31	1813	1814 年 1 月 1 日
123	12	31	1912	1913 年 1 月 1 日
124	12	31	2011	2012 年 1 月 1 日
125	12	31	2012	2013 年 1 月 1 日

3. 佣金问题的测试用例

步枪销售商在亚利桑那州境内销售制造商制造的步枪机、枪托和枪管。枪机卖 45 美元，枪托卖 30 美元，枪管卖 25 美元。销售商每月至少要售出一支完整的步枪，生产限额考虑到大多数销售商在一个月内可销售 70 个枪机、80 个枪托和 90 个枪管。销售商在每访问一个镇子之后，给制造商发出电报，说明在那个镇子中售出的枪机、枪托和枪管数量。到了月末，销售商要发出一封很短的电报，以便制造商知道当月的销售情况。销售商的佣金为：销售额不到(含)1000 美元部分，为 10%；1000(不含)到 1800(含)美元的部分，为 15%；超过 1800 美元的部分为 20%。佣金程序生成月份销售报告，汇总售出的枪机、枪托和枪管总数，销售商的总销售额，以及佣金，如表 3.15 所示。

表 3.15　输出边界值分析测试用例

用例	枪机	枪托	枪管	销售额/美元	佣金/美元	注　释
1	1	1	1	100	10	输出最小值
2	1	1	2	125	12.5	输出略大于最小值
3	1	2	1	130	13	输出略大于最小值
4	2	1	1	145	14.5	输出略大于最小值
5	5	5	5	500	50	中点
6	10	10	9	975	97.5	略低于边界点
7	10	9	10	970	97	略低于边界点
8	9	10	10	955	95.5	略低于边界点
9	10	10	10	1000	100	边界点
10	10	10	10	1025	103.75	略高于边界点
11	10	11	10	1030	104.5	略高于边界点
12	11	10	10	1045	106.75	略高于边界点
13	14	14	14	1400	160	中点
14	18	18	17	1775	216.25	略低于边界点
15	18	17	18	1770	215.5	略低于边界点
16	17	18	18	1755	213.25	略低于边界点
17	18	18	18	1800	220	边界点

续表

用例	枪机	枪托	枪管	销售额/美元	佣金/美元	注　　释
18	18	18	19	1825	225	略高于边界点
19	18	19	18	1830	226	略高于边界点
20	19	18	18	1845	229	略高于边界点
21	48	48	48	4800	820	中点
22	70	80	89	7775	1415	输出略小于最大值
23	70	79	90	7770	1414	输出略小于最大值
24	69	80	90	7755	1411	输出略小于最大值
25	70	80	90	7800	1420	输出最大值

低于较低平面的值，对应低于 1000 美元门限的销售额。两个平面之间的值是 15%佣金区域。使用输出值域确定测试用例的部分原因是，通过输入值域生成的测试用例几乎都在 20%区域。我们要找出强调边界值 100 美元、1000 美元、1800 美元和 7800 美元对应的输入变量组合。这些测试用例是通过电子表格开发的，节省了大量计算工作。最大值和最小值的确定很容易，给出的数正好便于生成边界点。测试用例 9 是 1000 美元的边界点。如果调整输入变量，则可以得到略低和略高于该边界的值，如表 3.16 所示。

表 3.16　输出特殊值测试用例

用例	枪机	枪托	枪管	销售额/美元	佣金/美元	注释
1	10	11	9	1005	100.75	略高于边界点
2	18	17	19	1795	219.25	略低于边界点
3	18	19	17	1805	221	略高于边界点

3.2.5　随机测试

随机测试是根据测试说明书执行用例测试的重要补充手段，是保证测试覆盖完整性的有效方式和过程。随机测试的基本思想是：不是永远选取有界变量的最小值、略高于最小值、正常值、略低于最大值的最大值，而是使用随机数生成器选出测试用例值。随机测试可以避免出现测试偏见，但是也可能带来一个严重问题：多少随机测试用例才是充分的？这些测试用例都是用选择有界变量 a≤x≤b 值的一个 Visual Basic 应用程序生成的，x 满足下式：

```
x = Int(b - a + 1) * Rnd + a
```

其中函数 Int 返回浮点数的整数部分，函数 Rnd 生成区间[0,1]内的随机数。这个程序持续生成随机测试用例，直到每种输出至少出现一次。在每张表中，该程序进行七次“循环”，以“很难生成”的测试用例结束，如表 3.17～表 3.19 所示。

表 3.17 三角形程序的随机测试用例

测试用例	非三角形	不等边三角形	等腰三角形	等边三角形
1289	663	593	32	1
15436	7696	7372	367	1
17091	8556	8164	367	1
2603	1284	1252	66	1
6475	3197	3122	155	1
5978	2998	2850	129	1
9008	4447	4353	207	1
平均值	49.83%	47.87%	2.29%	0.01%

表 3.18 佣金程序的随机测试用例

测试用例	10%	15%	20%
91	1	6	84
27	1	1	25
72	1	1	70
176	1	6	169
48	1	1	46
152	1	6	145
125	1	4	120
平均值	1.01%	3.62%	95.37%

表 3.19 NextDate 程序的随机测试用例

测试用例	有 31 天的月份的 1～30 日	有 31 天的月份的 31 日	有 30 天的月份的 1～29 日	有 30 天的月份的 30 日
913	542	17	274	10
1101	621	9	358	8
4201	2448	64	1242	46
1097	600	21	350	9
5853	3342	100	1804	82
3959	2195	73	1252	42
1436	786	22	456	13
平均值	56.76%	1.65%	30.91%	1.13%
可能值	56.45%	1.88%	31.18%	1.88%
2 月的 1～27 日	**闰年的 2 月 28 日**	**非闰年的 2 月 28 日**	**闰年的 2 月 29 日**	**不可能的日期**
45	1	1	1	22
83	1	1	1	19
312	1	8	3	77
92	1	4	1	19
417	1	11	2	94
310	1	6	5	75
126	1	5	1	26
7.46%	0.04%	0.19%	0.08%	1.79%
4.26%	0.07%	0.20%	0.07%	1.01%

3.2.6 边界值测试的指导方针

除了特殊值测试，基于函数（程序）输入定义域的测试方法，是所有测试方法中最基本的。这类测试方法都有一种假设，即输入变量是真正独立的，如果不能保证这种假设，则这类方法会产生不能令人满意的测试用例。这些方法还有两方面的区别：正常值与健壮值，单缺陷与多缺陷假设。仔细地运用这些差别就能产生较好的测试。这些方法都可以用于程序的输出值域，就像我们在佣金问题中所做的那样。

另一种很有用的基于输出的测试用例形式，可用于生成错误消息的系统。测试人员应该设计测试用例，以检查在适当的时候，错误消息是否被生成，并且不会被错误地生成。定义域分析还可以用于内部变量，例如循环控制变量、索引和指针。健壮性测试是测试内部变量的一种好的选择。

3.3 决策表法

决策表又称判断表，是一种呈表格状的图形工具，适用于描述处理判断条件较多，各条件又相互组合、有多种决策方案的情况。决策表可精确而简洁地描述复杂逻辑的方式，将多个条件与这些条件满足后要执行的动作相对应。但不同于传统程序语言中的控制语句，决策表能将多个独立的条件和多个动作直接地联系、清晰地表示出来。

3.3.1 决策表

自 20 世纪 60 年代初以来，决策表一直被用来表示和分析复杂逻辑关系。表 3.20 给出了基本决策表术语。

表 3.20 决策表的各个部分

桩	规则 1	规则 2	规则 3、4	规则 5	规则 6	规则 7、8
C1	T	T	T	F	F	F
C2	T	T	F	T	T	F
C3	T	F	—	T	F	—
a1	X	X		X		
a2	X				X	
a3		X		X		
a4			X			X

决策表一般分为条件桩、条件项、动作桩、动作项 4 个部分。每个条件对应一个变量、关系或预测，“候选条件”就是它们所有可能的值；动作指要执行的过程或操作；动作入口指根据该入口所对应的候选条件集，是否或按怎样的顺序执行动作。

许多决策表在候选条件中使用“不关心”符号来化简决策表，尤其是当某一条件对应要执行的动作影响很小时。有时，所有的条件在开始时都被认为是重要的，但最后却发现没有一个条件对执行的动作有影响，都是无关的条件。

在这 4 个部分的基础上，决策表根据候选条件和动作入口的表现方法的变化而变化。有些决策表使用 true/false 作为候选条件值(类似于 if-then-else)，有些使用数字(类似于 switch-case)，有些甚至使用模糊值或概率值。对应动作入口，可以简单地表示为动作是否执行(检查动作执行)，或更高级些，罗列出要执行的动作(为执行的动作排序)。

为了使用决策表标识测试用例，我们把条件解释为输入，把行动解释为输出。有时条件最终引用输入的等价类，行动引用被测软件的主要功能处理部分。这时规则就解释为测试用例。

表 3.21 所示的决策表给出了有关表示方法的另一种考虑：条件的选择可以大大地扩展决策表的规模。将(C1：a、b、c 构成三角形?)扩展为三角形特性的三个不等式的详细表示，如表 3.22 所示。如果有一个不等式不成立，则三个整数就不能构成三角形。我们还可以进一步扩展，因为不等式不成立有两种方式：一条边等于另外两边的和，或严格大于另外两条边的和。

表 3.21 三角形问题决策表

条 件 桩	1	2	3	4	5	6	7	8	9
c1：a、b、c 构成三角形?	N	Y	Y	Y	Y	Y	Y	Y	Y
c2：a＝b?	—	Y	Y	Y	Y	N	N	N	N
c3：a＝c?	—	Y	Y	N	N	Y	Y	N	N
c4：b＝c?	—	Y	N	Y	N	Y	N	Y	N
a1：非三角形	√								
a2：不等边三角形									√
a3：等腰三角形					√		√	√	
a4：等边三角形		√							
a5：不可能			√	√		√			

表 3.22 经过修改的三角形问题决策表

条 件 桩	1	2	3	4	5	6	7	8	9	10	11
c1：a＜a＋c?	F	T	T	T	T	T	T	T	T	T	T
c2：b＜a＋c?	—	F	T	T	T	T	T	T	T	T	T
c3：c＜a＋b?	—	F	T	T	T	T	T	T	T	T	
c4：a＝b?	—	—	—	T	T	T	T	F	F	F	F
c5：a＝c?	—	—	—	T	T	F	F	T	T	F	F
c6：b＝c?	—	—	—	T	F	T	F	T	F	T	F
a1：非三角形	X	X	X								
a2：不等边三角形											X
a3：等腰三角形							X		X	X	
a4：等边三角形				X							
a5：不可能					X	X		X			

表 3.23 所示的决策表是 NextDate 问题，引用了可能的月份变量相互排斥的可能性。

表 3.23 带有相互排斥条件的决策表

条 件	规则 1	规则 2	规则 3
C1：月份在 M1 中？	T	—	—
C2：月份在 M2 中？	—	T	—
C3：月份在 M3 中？	—	—	T
a1			
a2			
a3			

不关心条目的使用，对完整决策树的识别方式有微妙的影响。对于有限条目决策表，如果有 n 个条件，则必须有 2^n 条规则。如果与表述不相关，则可以按以下方法统计规则数，如表 3.24 和表 3.25 所示。

表 3.24 表 3.22 所示规则条数统计的决策表

条 件 桩	1	2	3	4	5	6	7	8	9	10	11
c1：a<a+c？	F	T	T	T	T	T	T	T	T	T	T
c2：b<a+c？	—	F	T	T	T	T	T	T	T	T	T
c3：c<a+b？	—	F	T	T	T	T	T	T	T	T	
c4：a=b？	—	—	—	T	T	T	T	F	F	F	F
c5：a=c？	—	—	—	T	T	F	F	T	T	F	F
c6：b=c？	—	—	—	T	F	T	F	T	F	T	F
规则条数统计	32	16	8	1	1	1	1	1	1	1	1
a1：非三角形	X	X	X								
a2：不等边三角形											X
a3：等腰三角形							X		X	X	
a4：等边三角形				X							
a5：不可能					X	X		X			

表 3.25 带有相互排斥条件的决策表规则条数统计

条 件	规则 1	规则 2	规则 3
C1：月份在 M1 中？	T	—	—
C2：月份在 M2 中？	—	T	—
C3：月份在 M3 中？	—	—	T
规则条数统计	4	4	4
a1			

应该只有 8 条规则，为了找出问题所在，我们扩展所有的条件规则，用可能的 T 或 F 替代“—”，如表 3.26 所示。

表 3.26 表 3.25 的扩展版本

条 件	1.1	1.2	1.3	1.4	2.1	2.2	2.3	2.4	3.1	3.2	3.3	3.4
C1：月份在 M1 中？	T	T	T	T	T	T	F	F	T	T	F	F
C2：月份在 M2 中？	T	T	F	F	T	T	T	T	T	F	T	F
C3：月份在 M3 中？	T	F	T	F	T	F	T	F	T	T	T	T
规则条数统计	1	1	1	1	1	1	1	1	1	1	1	1
a1												

所有条目都是 T 的规则有三条：规则 1.1、规则 2.1、规则 3.1；条目是 T、T、F 的规则有两条：规则 1.2 和规则 2.3。如果去掉这种重复，最后得到 7 条规则，缺少的规则是所有条件都是假的规则。这种处理的结果如表 3.27 所示。

表 3.27 包含不可能出现的规则的相互排斥条件

条 件	1.1	1.2	1.3	1.4	2.3	2.4	3.4
C1：月份在 M1 中？	T	T	T	T	F	F	F
C2：月份在 M2 中？	T	T	F	F	T	T	F
C3：月份在 M3 中？	T	F	T	F	T	F	T
规则条数统计	1	1	1	1	1	1	1
a1：不可能	X	X	X		X		

3.3.2 实例

1. 三角形问题的测试用例

根据表 3.22 可得到 11 个功能性测试用例：3 个不可能测试用例，3 个测试用例违反三角形性质，1 个测试用例可得到等边三角形，1 个测试用例可得到不等边三角形，3 个测试用例可得到等腰三角形，如表 3.28 所示。

表 3.28 根据表 3.22 得到的测试用例

用例 ID	a	b	c	预 期 输 出
DT1	4	1	2	非三角形
DT2	1	4	2	非三角形
DT3	1	2	4	非三角形
DT4	5	5	5	等边三角形
DT5	?	?	?	不可能
DT6	?	?	?	不可能
DT7	2	2	3	等腰三角形
DT8	?	?	?	不可能
DT9	2	3	2	等腰三角形
DT10	3	2	2	等腰三角形
DT11	3	4	5	不等边三角形

2. NextDate 函数测试用例

决策表最突出的优点是：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。利用决策表能够设计出完整的测试用例集合，运用决策表设计测试用例可以将条件理解为输入，将动作理解为输出。决策表如表 3.29 所示。

M1：{month：month 有 30 天}

M2：{month：month 有 31 天，12 月除外}

M3：{month：month 是 12 月}

M4：{month：month 是 2 月}

D1：{day：1≤day≤27}

D2：{day：day＝28}

D3：{day：day＝29}

D4：{day：day＝30}

D5：{day：day＝31}

Y1：{year：year 是闰年}

Y2：{year：year 不是闰年}

表 3.29 有 256 条规则的决策表

条　　件	1	2
C1：月份在 M1 中？	T	
C2：月份在 M2 中？		
C3：月份在 M3 中？	T	T
C4：日期在 D1 中？		
C5：日期在 D2 中？		
C6：日期在 D3 中？		
C7：日期在 D4 中？		
C8：日期在 Y1 中？		
A1：不可能		
A2：NextDate		

这一次采用扩展条目决策表开发，如果规则条目之间存在“重叠”，则会存在冗余情况，使得多个规则都能满足。决策表如表 3.30 所示。

M1：{month：month 有 30 天}

M2：{month：month 有 31 天}

M3：{month：month 此月是 2 月}

D1：{day：1≤day≤28}

D2：{day：day＝29}

D3：{day：day＝30}

D4：{day：day＝31}

Y1：{year：year＝2000}

Y2：{year：year 是闰年}

Y3：{year：year 是平年}

表 3.30 有 36 条规则的决策表

条 件	1	2	3	4	5	6	7	8
c1：月份	M1	M1	M1	M1	M2	M2	M2	M2
c2：日期	D1	D2	D3	D4	D1	D2	D3	D4
c3：年	—	—	—	—	—	—	—	—
规则条数统计	3	3	3	3	3	3	3	3
行为								
a1：不可能				X				
a2：day 加 1	X	X			X	X	X	
a3：日期复位			X					X
a4：month 加 1			X					?
a5：月份复位								?
a6：year 加 1								?
条 件	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
c1：月份	M3	M3	M3	M3	M3	M3	M3	M3
c2：日期在	D1	D1	D1	D2	D2	D2	D3	D3
c3：年在	Y1	Y2	Y3	Y1	Y2	Y3	—	—
规则条数统计	1	1	1	1	1	1	3	3
行为								
a1：不可能						X	X	X
a2：日期加 1		X						
a3：日期复位	X		X	X	X			
a4：月份加 1	X		X	X	X			
a5：月份复位								
a6：年加 1								

规则 1、规则 2、规则 3 都涉及有 30 天的月份 day 类 D1、D2 和 D3，并且它们的动作项都是 day 加 1，因此可以将规则 1、规则 2、规则 3 合并。类似地，有 31 天的月份 day 类 D1、D2、D3 和 D4 也可合并，2 月的 D4 和 D5 也可合并。

通过引入等价类的第 3 个集合，可以澄清年末问题。

M1：{月份：每月有 30 天}

M2：{月份：每有 31 天，12 月除外}

M3：{月份：此月是 12 月}

M4：{月份：此月是 2 月}

D1：{日期：1≤日期≤27}

D2：{日期：日期＝28}

D3：{日期：日期＝29}

D4：{日期：日期＝30}

D5：{日期：日期＝31}

Y1：{年：年是闰年}

Y2：{年：年不是闰年}

(1) 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}

(2) day 变量的有效等价类：

D1：{1≤day≤27}　　D2：{day=28}　　D3：{day=29}

D4：{day=30}　　D5：{day=31}

(3) year 变量的有效等价类：

Y1：{year 是闰年}　　Y2：{year 不是闰年}

(4) 程序中可能采取的操作有以下 6 种：

a1：不可能　　a2：day+1　　a3：day=1

a4：month+1　　a5：month=1　　a6：year+1

表 3.31 所示的决策表是 NextDate 函数源代码的基础。这个例子从另一个方面说明测试如何能够很好地改进程序设计。所有决策表分析都应该在 NextDate 函数的详细设计期间完成，如表 3.32 所示。

表 3.31　NextDate 函数的决策表

	1	2	3	4	5	6	7	8	9	10	11
条件：	M1	M1	M1	M1	M1	M2	M2	M2	M2	M2	M3
c1：月份在	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5	D1
c2：日期在	—	—	—	—	—	—	—	—	—	—	—
c3：年在											
行为：	√	√	√	√	√	√	√	√	√	√	√
a1：不可能				√						√	
a2：日期加 1											
a3：日期复位											
a4：月份加 1											
a5：月份复位											
a6：年加 1											

表 3.32　NextDate 函数的精简决策表

	12	13	14	15	16	17	18	19	20	21	22
条件：	M3	M3	M3	M3	M4	M4	M4	M4	M4	M4	M4
c1：月份在	D2	D3	D4	D5	D1	D2	D2	D3	D3	D4	D5
c2：日期在	—	—	—	—	—	Y1	Y2	Y1	Y2	—	—
c3：年在											
行为：	√	√	√	√	√	√	√	√	√	√	√
a1：不可能				√			√	√			
a2：日期加 1				√							
a3：日期复位											
a4：月份加 1											
a5：月份复位											
a6：年加 1											

相应的测试用例如表 3.33 所示。

表 3.33 NextDate 函数的决策表测试用例

测试用例	month	day	year	预期输出
Test1～Test3	6	16	2001	17/6/2001
Test4	6	30	2004	1/7/2004
Test5	6	31	2001	不可能
Test6～Test9	8	16	2004	17/8/2004
Test10	8	31	2001	1/9/2001
Test11～Test14	12	16	2004	17/12/2004
Test15	12	31	2001	1/1/2002
Test16	2	16	2004	17/2/2004
Test17	2	28	2004	29/2/2004
Test18	2	28	2001	1/3/2001
Test19	2	29	2004	1/3/2001
Test20	2	29	2001	不可能
Test21～Test22	2	30	2004	不可能

3. 佣金问题的测试用例

决策表分析不太适合佣金问题。在佣金问题中只有很少的决策逻辑。

3.3.3 指导方针

决策表的测试对于某些应用程序(例如 NextDate 函数)很有效,但是(例如佣金问题)不值得用决策表。

(1) 决策表技术适用于具有以下特征的应用程序:

① if-else 逻辑突出;

② 输入变量之间存在逻辑关系;

③ 涉及输入变量子集的计算;

④ 输入与输出之间存在因果关系;

⑤ 很高的圈复杂度。

(2) 决策表不能很好地伸缩(有 n 个条件的有限条目决策表有 2^n 个规则)。有多种方法可以解决这个问题——使用扩展条目决策表、代数简化表,将大表“分解”为小表,查找条件条目的重复模式。

(3) 与其他技术一样,迭代会有所帮助。第一次标识的条件和行动可能不那么令人满意。把第一次得到的结果作为“铺路石”,逐渐改进,直到得到满意的决策表。

3.4 因果图法

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。因果图法正适用于输入条件组合复杂的情况。

因果图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，它适用于检查程序输入条件的各种组合情况。

因果图中出现的基本符号，如图 3.12 所示。

图 3.12 因果图中出现的基本符号

通常在因果图中用 *ci* 表示原因，用 *ei* 表示结果，各节点表示状态，可取值 0 或 1。0 表示某状态不出现，1 表示某状态出现。

在因果图中，原因与结果之间主要有以下几种关系，如图 3.13 所示。

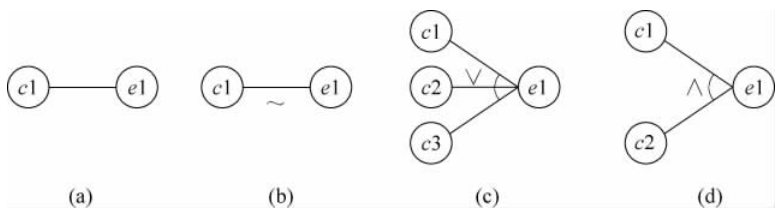

图 3.13 因果图中各种关系示意图

(a) 恒等：若 *c*1 是 1，则 *e*1 也为 1，否则 *e*1 为 0；

(b) 非：若 *c*1 是 1，则 *e*1 为 0，否则 *e*1 为 1；用符号“～”表示。

(c) 或：若 *c*1 或 *c*2 或 *c*3 是 1，则 *e*1 是 1，否则 *e*1 为 0，“或”可有任意个输入；用符号“∨”表示。

(d) 与：若 *c*1 和 *c*2 都是 1，则 *e*1 为 1，否则 *e*1 为 0，“与”也可有任意个输入。

在实际问题当中输入状态相互之间还可能存在某些依赖关系，称为“约束”，如图 3.14 所示。

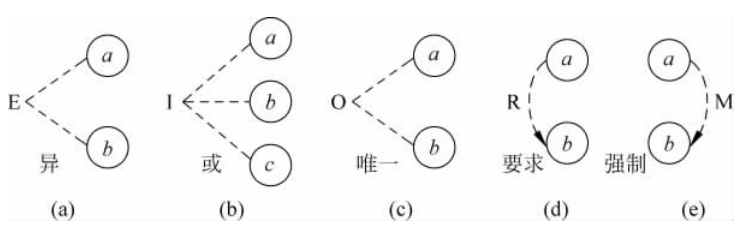

图 3.14 因果图中各种约束示意图

对于输入条件的约束有 4 种。

(a) E 约束(异)：*a* 和 *b* 中最多有一个可能为 1，即 *a* 和 *b* 不能同时为 1；

(b) I 约束(或)：*a*、*b*、*c* 中至少有一个必须是 1，即 *a*、*b*、*c* 不能同时为 0；

(c) O 约束(唯一)：*a* 和 *b* 必须有一个且仅有一个为 1；

(d) R 约束(要求)：*a* 是 1 时，*b* 必须是 1。

对于输出条件的约束只有 M 约束。

(e) M 约束(强制)：若结果 a 是 1，则结果 b 强制为 0。

用因果图法设计测试用例，首先从程序规格说明书的描述中，找出原因(输入条件)和结果(输出结果或者程序状态的改变)，然后通过因果图转换为判定表，最后为判定表中的每一列设计一个测试用例。具体步骤是：

(1) 分析待测得系统规格，找出原因与结果。

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

(2) 画出因果图。

分析软件规格说明描述中的语义。找出原因与结果之间、原因与原因之间的对应关系。根据这些关系，画出因果图。

(3) 标记约束或限制条件。

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下不可能出现。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) 用判定表中的每一项生成测试用例。

【因果图实例】

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，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。

(1) 对说明进行分析，得到原因和结果：

原因：

$c1$：第一列字符是 A；

$c2$：第一列字符是 B；

$c3$：第二列字符是一个数字。

结果：

$a1$：修改文件；

$a2$：给出信息 L；

$a3$：给出信息 M。

(2) 其对应的因果图为：11 为中间节点；考虑到原因 C1 和原因 C2 不可能同时为 1，故在因果图上施加 E 约束，如图 3.15 和图 3.16 所示。

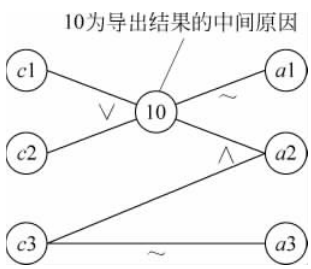

图 3.15 因果图

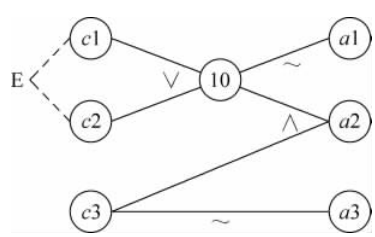

图 3.16 带有约束的因果图

(3) 根据因果图建立判定表,如表 3.34 所示。

表 3.34 判定表

		1	2	3	4	5	6	7	8
条件	$c1$	1	1	1	1	0	0	0	0
	$c2$	1	1	0	0	1	1	0	0
	$c3$	1	0	1	0	1	0	1	0
中间条件	11			1	1	1	1	0	0
动作	$a1$							Y	Y
	$a2$			Y		Y			
	$a3$				Y		Y		Y
	不可能	Y	Y						
测试用例				＃3	＃B	＊7	＊M	$c2$	cM

(4) 把判定表的每一列拿出来作为依据,设计测试用例。

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,如表 3.35 所示。

表 3.35 测试用例

编 号	测试用例	预期输出
Test1	＃3	修改文件
Test2	＃B	给出信息 M
Test3	＊7	修改文件
Test4	＊M	给出信息 M
Test5	$c2$	给出信息 N
Test6	cM	给出信息 M

因果图法的特点:

(1) 考虑到了输入情况的各种组合以及各个输入情况之间的相互制约关系。

(2) 能够帮助测试人员按照一定的步骤,高效率地开发测试用例。

(3) 因果图法是将自然语言规格说明转化成形式语言规格说明的一种严格的方法,可以指出规格说明存在的不完整性和二义性。

3.5 场景法

场景法是指通过运用场景来对系统的功能点或业务流程进行描述,从而提高测试效果的一种方法。以用例场景来测试需求是指模拟特定场景边界发生的事情,通过事件来触发某个动作的发生,观察事件的最终结果,从而发现需求中存在的问题。我们通常以正常的用例场景分析开始,然后再着手进行其他的场景分析。场景法一般包含基本流和备用流。从一个流程开始,通过描述经过的路径来确定过程,经过遍历所有的基本流和备用流来完成整个场景。场景主要包括 4 种主要的类型:正常的用例场景、备选的用例场景、异常的用例场景和假定推测的场景。

现在的软件几乎都是由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形成事件流。这种在软件设计方面的思想也可被引入到软件测试中，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，同时测试用例也更容易得到理解和执行。

如图 3.17 所示。经过用例的每条不同路径都反映了基本流和备选流，都用箭头来表示。基本流用直黑线来表示，是经过用例的最简单的路径。每个备选流自基本流开始，之后，备选流会在某个特定条件下执行。备选流可能会重新加入基本流中（备选流 1 和备选流 3），还可能起源于另一个备选流（备选流 2），或者终止用例而不再重新加入某个流（备选流 2 和备选流 4）。

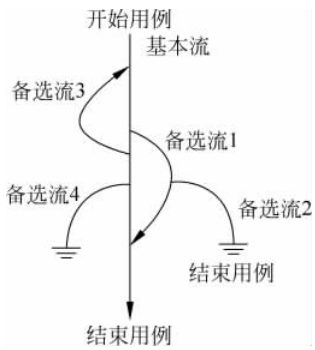

图 3.17　用例场景图

遵循图 3.17 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。

注：为方便起见，场景 5、场景 6 和场景 8 只描述了备选流 3 指示的循环执行一次的情况。

【ATM 机实例】

示例：

表 3.36 包含了图 3.18 中提款用例的基本流和某些备用流：

（1）用户必须能从 ATM 卡的任一有效账户上提取现金，提取的金额为 50.00 元的整数倍，每次现金支付时，必须得到银行的认可。

（2）用户必须能从 ATM 卡的任一有效账户上存款。

（3）用户必须能在 ATM 卡的任一有效账户之间进行货币转账。

（4）用户必须能查询 ATM 卡的任一有效账户上的存款余额。

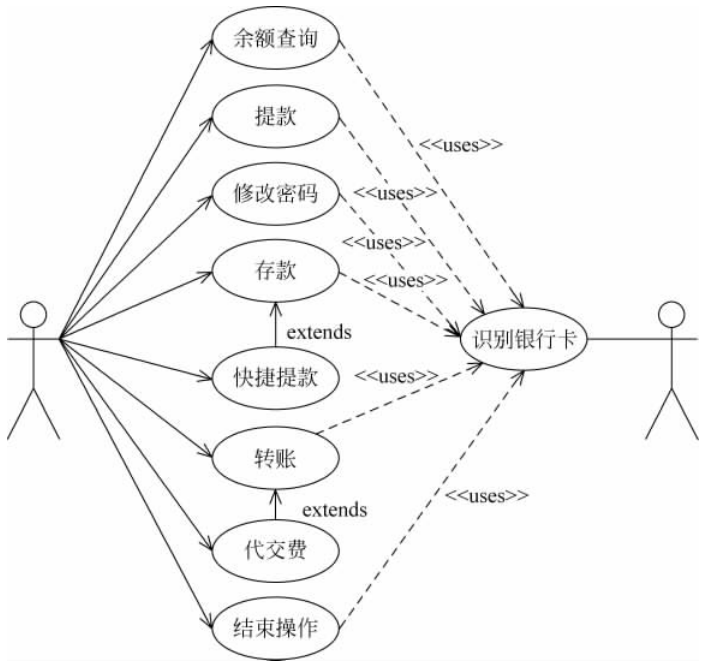

图 3.18 ATM 的用例图

(5) 如果银行确认用户的 PIN 无效,在事务进行之前,要求用户再输入 PIN,如果用户输入 3 次都不成功,ATM 将永久保留 ATM 卡,用户必须与银行联系方可取回 ATM 卡。

(6) ATM 机每次交互都通知银行以获得银行的验证。

(7) 对于每一个成功的事务处理,ATM 机给用户打印一个收据,提示日期、时间、ATM 机位置、交互类型、账户、数额、转出与转入账户余额。

(8) ATM 机有一个带有钥匙操作开关面板,安置在银行内部,让银行操作员启动或停止用户服务。

场景法设计测试用例的步骤如下:

(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) 对于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,确定测试用例后,对每一个测试用例确定测试数据值。

对于 ATM 系统的场景法设计测试用例的设计步骤是:

(1) 基本事件流。

① 用户向 ATM 提款机中插入银行卡,如果银行卡是合法的,ATM 提款机界面提示用户输入提款密码;

② 用户输入该银行卡的密码,ATM 提款机与主机传递密码,检验密码的正确性。如果输入密码正确,提示用户输入取钱金额,提示信息为“请输入您的提款额度”;

③ 用户输入取钱金额,系统校验金额正确,提示用户确认,提示信息为“您输入的金额是……,请确认,谢谢!”,用户按下确认键,确认需要提取的金额;

④ 系统同步银行主机，点钞票，输出给用户，并且减掉数据库中该用户账户中的存款金额；

⑤ 用户提款，银行卡自动退出，用户取走现金，拔出银行卡，ATM 提款机界面恢复到初始状态。

(2) 备选事件流。

备选流 1：如果插入无效的银行卡，那么，在 ATM 提款机界面上提示用户“您使用的银行卡无效!”，3 秒钟后，自动退出该银行卡。

备选流 2：如果用户输入的密码错误，则提示用户“您输入的密码无效，请重新输入”。

备选流 3：如果用户连续 3 次输入错误密码，ATM 提款机吞卡，并且 ATM 提款机的界面恢复到初始状态。此时，其他提款人可以继续使用其他的合法的银行卡在 ATM 提款机上提取现金。

备选流 4：用户输入错误的密码后，也可以按“退出”键，则银行卡自动退出。

备选流 5：如果用户输入的单笔提款金额超过单笔提款上限，ATM 提款机界面提示“您输入的金额错误，单笔提款上限金额是 1500RMB，请重新输入”。

备选流 6：如果用户输入的单笔金额，不是以 50RMB 为单位的，那么提示用户“您输入的提款金额错误，请输入以 50 为单位的金额”。

备选流 7：如果用户在 24 小时内提取的金额大于 4500RMB，则 ATM 提款机提示用户，“24 小时内只能提取 4500RMB，请重新输入提款金额”(输入提取的金额超过了系统的设定的限制)。

备选流 8：如果用户输入正确的提款金额，ATM 提款机提示用户确认后，用户取消提款，则 ATM 提款机自动退出该银行卡。

备选流 9：如果 ATM 提款机中余额不足，则提示用户，“抱歉，ATM 提款机中余额不足”，3 秒钟后，自动退出银行卡。

备选流 10：如果用户银行户头中的存款小于提款金额，则提示用户“抱歉，您的存款余额不足!”，3 秒钟后，自动退出银行卡。

备选流 11：如果用户没有取走现金，或者没有拔出银行卡，ATM 提款机不做任何提示，直接恢复到界面的初始状态。

(3) 根据基本流和备选流生成场景。

以“取款”用例为例，“取款”用例的事件流如下：

基本流：预设提取金额(100 元、200 元、500 元、1000 元)；

备选流 2：ATM 内没有现金；

备选流 3：ATM 内现金不足；

备选流 4：PIN 有误；

备选流 5：账户不存在/账户类型有误；

备选流 6：账面金额不足。

根据“取款”用例的事件流，生成的场景有：

场景 1：成功的取款：基本流；

场景 2：ATM 内没有现金：基本流，备选流 2；

场景 3：ATM 内现金不足：基本流，备选流 3；

场景 4：PIN 有误(还有输入机会)：基本流，备选流 4；

场景 5：PIN 有误(不再有输入机会)：基本流，备选流 3，备选流 4；

场景 6：账户不存在/账户类型有误：基本流，备选流 5；

场景 7：账户余额不足：基本流，备选流 6。

(4) 对每一个场景生成对应的测试用例设计。

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。通过从确定执行用例场景所需的数据元素入手构建矩阵。对于每个场景，至少要确定包含执行场景所需的适当条件的测试用例，如表 3.36 所示。

表 3.36 ATM 的测试用例矩阵表示

编号	场景/条件	PIN	账号	选择的金额	账面金额	ATM 内的金额	预 期 结 果
1	场景 1：成功提款	V	V	V	V	V	成功提款
2	场景 2：ATM 内没有现金	V	V	V	V	1	提款选项不可用，用例结束
3	场景 3：ATM 内现金不足	V	V	V	V	1	警告消息，返回基本流步骤 6——输入金额
4	场景 4：PIN 有误(还有不止一次输入机会)	1	V	n/a	V	V	警告消息，返回基本流步骤 4，输入 PIN
5	场景 4：PIN 有误(还有一次输入机会)	1	V	n/a	V	V	警告消息，返回基本流步骤 4，输入 PIN
6	场景 4：PIN 有误(不再有输入机会)	1	V	n/a	V	V	警告消息，卡予以保留，用例结束

3.6 正交实验法

利用因果图来设计测试用例时，作为输入条件的原因与输出结果之间的因果关系，有时很难从软件需求规格说明中得到。因果关系往往非常多，导致利用因果图而得到的测试用例数目多得惊人，给软件测试带来了沉重的负担。为了有效地、合理地减少测试的工时与费用，可利用正交试验法进行测试用例的设计。

正交实验法就是利用排列整齐的正交表来对试验进行整体设计、综合比较、统计分析，实现通过少数的实验次数找到较好的生产条件，以达到最佳生产工艺效果，这种试验设计法是从大量的试验点中挑选适量的具有代表性的点，利用已经造好的表格——正交表来安排试验并进行数据分析的方法。正交表能够在因素变化范围内均衡抽样，使每次试验都具有较强的代表性，由于正交表具备均衡分散的特点，保证了全面实验的某些要求，这些试验往往能够较好或更好地达到实验的目的。

1. 利用正交实验法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,把它们当作因子;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,并为确定每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地、正确地确定取值,以确保测试用例的设计做到完整与有效。

(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。

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。

在使用正交实验法时,要考虑到被测系统中要准备测试的功能点,而这些功能点就是要获取的因子或因素,但每个功能点要输入的数据按等价类划分有多个,也就是每个因素的输入条件,即状态或水平值。

2. 正交表的构成

(1) 行数(Runs):正交表中的行的个数,即试验的次数,也是我们通过正交实验法设计的测试用例的个数。

(2) 因素数(Factors):正交表中列的个数,即我们要测试的功能点。

(3) 水平数(Levels):任何单个因素能够取得的值的最大个数。正交表中的包含的值为从 0 到数“水平数-1”或从 1 到“水平数”。即要测试功能点的输入条件。

(4) 正交表的形式如下。

$L_{行数}(水平数^{因素数})$

(5) 正交表的表示方法。

① 用 L 代表正交表,常用的有 $L_8(2^7)$、$L_9(3^4)$、$L_{16}(4^5)$、$L_8(4\times2^4)$等。

② $L_8(2^7)$ 7 表示正交表的列数,2 为因子的水平数,8 表示正交表的行数。

③ $L_{16}(2\times3^7)$,有 7 列是 3 水平的,有 1 列是 2 水平的——做 16 个试验最多可以考察 1 个 2 水平的因子和 7 个 3 水平的因子。

④ 行数(即试验次数) $=\sum$(每列水平数 -1) $+1$。如:

$L_8(2^7)$,如图 3.19 所示。

列号 试验号	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2	1	1	1	2	2	2	2
3	1	2	2	1	1	2	2
4	1	2	2	2	2	1	1
5	2	1	2	1	2	1	2
6	2	1	2	2	1	2	1
7	2	2	1	1	2	2	1
8	2	2	1	2	1	1	2

图 3.19 正交表构成图

【实例】

为提高某化工产品的转化率，选择了三个有关因素进行条件试验：反应温度(A)、反应时间(B)和用碱量(C)，并确定了它们的试验范围如下：

A：80 ～90℃

B：90～150min

C：5%～7%

试验的目的是搞清楚因子 A、B、C 对转化率有什么影响，哪些是主要的，哪些是次要的，从而确定最适生产条件，即温度、时间及用碱量各为多少才能使转化率最高。

在试验范围内都选了三个水平(即各因素的不同状态)，如图 3.20 所示。

A：A_1=80℃，A_2=85℃，A_3=90℃

B：B_1=90min，B_2= 120min，B_3= 150min

C：C_1=5%，C_2=6%，C_3=7%

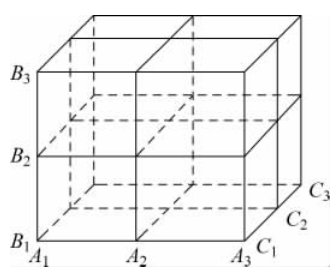

图 3.20 实例图

取三因子所有水平之间的组合，即 $A_1B_1C_1$、$A_1B_1C_2$、$A_1B_1C_3$、…、$A_3B_3C_3$，共有 3^3=27 次试验。用图 3.20 表示立方体的 27 个节点。

全面试验法对各因子与指标间关系的剖析比较清楚，但试验次数太多。特别是当因子数目多，每个因子的水平数目也很多时，试验量非常大。如选 6 个因子，每个因子取 5 个水平时，全面试验法需 5^6=15 625 次试验，这实际上是不可能实现的。

从全面试验的点中选择具有典型性、代表性的点，使试验点在试验范围内分布得很均匀，能反映全面情况。但我们又希望试验点尽量少，为此还要具体考虑一些问题。如上例，对应于 A 有 A_1、A_2、A_3 共 3 个平面，对应于 B、C 也各有 3 个平面，共 9 个平面。则这 9 个

平面上的点都应当一样多，即对每个因子的每个水平都要同等看待。具体来说，每个平面上都有 3 行、3 列，要求在每行、每列上的点一样多。如图 3.21 所示。

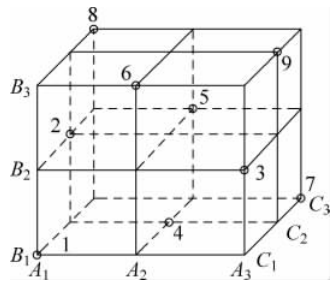

图 3.21　改进后的实例图

9 个平面中每个平面上恰好有 3 个点，而每个平面的每行每列都有且仅有 1 个点，总共 9 个点。这样的试验方案，试验点分布均匀，试验次数也不多，如表 3.37 所示。

表 3.37　正交矩阵

实验号	水平组合	实 验 条 件		
		温度/℃	时间/min	加减量/%
1	$A_1B_1C_1$	80	90	5
2	$A_1B_2C_2$	80	120	6
3	$A_1B_3C_3$	80	150	7
4	$A_2B_1C_2$	85	90	6
5	$A_2B_2C_3$	85	120	7
6	$A_2B_3C_1$	85	150	5
7	$A_3B_1C_3$	90	90	7
8	$A_3B_2C_1$	90	120	5
9	$A_3B_3C_2$	90	150	6

正交表必须满足以下两个性质：

(1) 对于表中任何一列，其所含各种水平的个数都相同。

(2) 在表的任何两列中，所有各种可能的数对出现的次数都相同。

课后习题

1. 程序规定；输入三个整数作为三边的边长构成三角形。当此三角形为一般三角形、等腰三角形、等边三角形时，分别计算。用等价类划分方法为该程序进行测试用例设计。

2. 保险公司计算保费费率的程序。

某保险公司的人寿保险的保费计算方式为：投保额×保险费率。其中，保险费率依点数不同而有别，10 点及 10 点以上保险费率为 0.6%，10 点以下保险费率为 0.1%；而点数又是由投保人的年龄、性别、婚姻状况和抚养人数来决定，具体规则如表 3.38 所示。

表 3.38 基本表

年 龄			性 别		婚 姻		抚养人数
20～39	40～59	其他	M	F	已婚	未婚	1 人扣 0.5 点，最多扣
6 点	4 点	2 点	5 点	3 点	3 点	5 点	3 点(四舍五入取整)

请用等价类划分法，进行弱健壮性测试，设计测试用例。

3. 有二元函数 $f(x,y)$，其中 $x\in[1900,2100]$，$y\in[1,12]$，采用边界值分析法设计测试用例。

4. 有函数 $f(x,y,z)$，其中 $x\in[1900,2100]$，$y\in[1,12]$，$z\in[1,31]$。请写出该函数采用边界值分析法设计的测试用例。

5. 某软件的一个模块的需求规格说明书中描述如下情况。

(1) 年薪制员工：严重过失，扣年终风险金的 4%；过失，扣年终风险金的 2%。

(2) 非年薪制员工：严重过失，扣当月薪资的 8%；过失，扣当月薪资的 4%。请绘制出因果图和判定表，并给出相应的测试用例。

6. 有一个处理单价为 1 元 5 角钱的盒装饮料的自动售货机软件，若投入 1 元 5 角硬币，按下“可乐”、“雪碧”或“橙汁”按钮，相应的饮料就送出来。若投入的是 2 元硬币，在送出相应的饮料同时退还 5 角硬币。

请绘制出因果图和判定表，并给出相应的测试用例。

7. PriorDate 函数。该函数要求输入 3 个变量 month、day 和 year，输出该日期之前一天的日期。使用判定表法进行测试用例设计。

8. 三角形问题决策表法设计测试用例。

(1) 确定规则个数(有 4 个条件，每个条件两个取值，有 $2^4=16$ 种规则)；

(2) 列出所有的条件桩和动作桩；

(3) 填入输入项；

(4) 填入动作项，得到初始决策表；

(5) 简化(合并相似规则)；

(6) 设计测试用例。

9. 以中国象棋中走马的测试用例设计为例学习因果图的使用方法。