第3章 黑盒测试及其实例

黑盒测试的作用主要是根据功能需求来测试程序是否按照预期工作，通过黑盒测试可以确定软件所实现的功能是否符合规格说明，也可以用来证明软件代码是否有错误和缺陷。本章将会对常见的几种黑盒测试技术基本原理进行介绍，并通过案例阐述应用这些测试技术的方法。

视频讲解

3.1 黑盒测试概述

软件产品必须具备一定的功能，通过这些功能为用户提供服务。软件产品的功能是为了满足用户的实际需求而设计的，在软件交付给用户使用前，所有的功能都需要经过验证，确定软件能够真正满足用户的需求。功能测试一般采用黑盒测试方法，将软件程序或系统看作一个不能打开的黑盒子，测试人员无须了解程序的内部结构，而是直接根据程序输入与输出之间的关系确定测试数据，推断测试结果的正确性。提高测试用例发现错误的能力和减少测试用例的冗余，是黑盒测试技术研究的重要问题。

黑盒测试关心的是软件的输入和输出，其主要测试依据是需求文档。黑盒测试是一种从用户角度出发的测试。软件的黑盒测试被用来证实软件功能的正确性和可操作性，而并不会破坏被测对象的数据信息，其主要试图发现下列几类错误。

(1) 功能不正确或遗漏。

(2) 界面错误。

(3) 数据库访问错误。

(4) 性能错误。

(5) 初始化和终止错误等。

在实际的操作执行中，黑盒测试可以被分为静态黑盒测试和动态黑盒测试。静态黑盒测试主要检查和审阅产品说明书，从中查验软件的缺陷。当软件测试人员第一次接到需要审查的产品说明书时，最容易做的就是把自己当作用户，设身处地为用户着想，此时了解用户所想是非常重要的。同时，测试人员还需要审查和测试同类软件，这样有助于制定测试条件和测试方法，暴露被设计者忽略的潜在问题。动态黑盒测试主要测试软件在使用过程中的实际行为，测试工作就是进行输入、接收输出、检验结果。有效的动态黑盒测试需要产品说明书和其他文档，了解输入什么将会得到什么，或者操作什么将会得到的结果又是什么。当清楚了软件的输入和输出之后，就要选择和设计测试用例，要把导致测试工作量异常的错误选择，以及测试目标选定错误等降至最低。因此，准确评估风险，把不可穷尽的可能性减少到可以控制的范围是非常重要的。

从理论上讲，黑盒测试只有采用穷举输入测试，把所有可能的输入都作为测试情况考虑，才能查出程序中所有的错误。实际上测试情况有无穷多个，人们不仅要测试所有的输入，还要对那些不合法但可能的输入进行测试。这样看来，完全测试是不可能的，所以要进行有针对性的测试，通过制定测试案例指导测试的实施，保证软件测试有组织、按步骤，以及有计划地进行。黑盒测试行为必须能够量化，只有量化才能真正保证软件质量，而测试用例就是将测试行为具体量化的方法。典型的黑盒测试用例设计方法包括等价类划分法、边界值分析法、决策表法、因果图法等。接下来将对这些测试方法进行介绍和讨论，并给出运用的实例。

3.2 等价类划分法

视频讲解

3.2.1 等价类划分法的概述

等价类划分是一种典型的黑盒测试方法，其主要根据特定的准则或关系将输入域划分为若干子集（被称为等价类）。这一划分所采用的准则或关系可能是不同的计算结果，也可能是某种基于控制流或数据流的关系，或者是根据系统能否实现接收和处理来区分的有效输入和无效输入（例如，超出范围的输入值将不被接受，应产生错误信息或触发错误处理程序）。从每个等价类中提取一个或若干个测试用例，聚合起来就形成了测试用例集合。

等价类指的是输入域的某个子集，在该子集中，各个输入数据对接入程序中的错误都是等效的，并且还可以被进一步合理假定：测试某个等价类的代表值就等于对这一类的其他值进行测试。

如果测试某一等价类中的一个数据时发现了错误，这一等价类中的其他数据也能发现同样的错误；反之，如果测试某一类中的一个数据没有发生错误，则这一类中的其他数据也不会被查出错误（除非等价类中的某些数据同时还属于另一等价类，因为几个等价类之间是可能相交的）。这样就可以把全部的输入数据合理地划分为若干个等价类，在每一个等价类中取一个数据作为测试的输入条件，从而把无限的穷举输入转化为有限的等价类并在其中选择具有代表性的数据进行输入，用少量的有代表性的测试数据来取得较好的测试结果。

3.2.2 常见的等价类划分法

本节主要从等价类的划分、等价类划分的原则及等价类划分表的建立等 3 方面对等价类进行说明。

1. 等价类的划分

等价类的划分有两种不同的情况：有效等价类和无效等价类。

有效等价类指的是对程序的规格说明来说是合理的、有意义的输入数据所构成的集合。利用有效等价类可检验程序是否实现了规格说明中所规定的功能和性能。

无效等价类与有效等价类的定义恰巧相反，是那些对程序的规格说明来说是不合理的或无意义的输入数据所构成的集合。

设计测试用例时，要同时考虑这两种等价类。具体到项目中，无效等价类至少应有一个，也可能有多个。因为软件不仅要能接收合理的数据，也要能经受各种意外的考验，如用户错误的输入等，这样的测试才能确保软件具有更高的可靠性。

等价类的划分需要在认真研读需求规格说明的基础上进行,它不仅可以被用来确定测试用例中的数据输入输出的精确取值范围,也可以被用来准备中间值状态、与时间相关的数据以及接口参数等。在有明确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。等价类划分方法还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,会使测试对象得到同样的反应行为。

2. 等价类划分的原则

常见的划分等价类的参考包括按区间划分、按数值划分、按数值集合划分、按限制条件或规划划分、按处理方式划分等。

下面给出 6 条划分等价类的原则。

(1) 在输入条件规定了取值范围或值个数的情况下,可以确立一个有效等价类和两个无效等价类。

(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,可以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。

(4) 在规定了输入数据的一组值(假定有 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。

(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)。

(6) 在确知已划分的等价类中,各元素在程序中处理方式不同的情况下,应再将该等价类进一步地划分为更小的等价类。

3. 等价类划分表的建立

在确立了等价类后,可以建立等价类表,列出所有已被划分出的等价类。

此处举一个等价类表的典型例子。一个软件中要求用户输入以年、月表示的日期,假定日期的输入范围限定在 2000 年 1 月至 2100 年 12 月之间,并且规定日期由 6 位数字字符组成,前 4 位表示年,后 2 位表示月,那么对应的"日期输入格式检查"这一功能的等价类表可以设计为如表 3-1 所示。

表 3-1 等价类表

输 入 条 件	有效等价类	无效等价类
日期的类型及长度	① 6 位数字字符	④ 包含非数字字符
		⑤ 小于 6 位数字字符
		⑥ 大于 6 位数字字符
年份范围	② 在 2000～2100 之间	⑦ 小于 2000
		⑧ 大于 2100
月份范围	③在 01～12 之间	⑨ 等于 00
		⑩ 大于 12

3.2.3 等价类划分法的测试用例

依据表 3-1 所示建立的等价类表,可以从划分出的等价类中按以下步骤确定测试用例。

(1) 为每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覆盖的有效等价类。重复这一步,最后使得所有有效等价类均被测试用例覆盖。

(3) 设计一个新的测试用例,使其只覆盖一个无效等价类。重复这一步使所有无效等价类均被覆盖。

继续用这个日期输入的例子来演示测试用例的设计。第(1)步为每个等价类规定编号,这一工作已经在等价类表中完成了;继续反复迭代地完成第(2)步,以形成有效类所对应的测试用例,如表 3-2 所示;反复迭代完成第(3)步以形成无效类所对应的测试用例如表 3-3 所示。

表 3-2 设计的有效类测试用例

测试数据	期望结果	覆盖的有效等价类
200001	输入有效	①②③
205506	输入有效	①②③
210012	输入有效	①②③

表 3-3 设计的无效类测试用例

测试数据	期望结果	覆盖的无效等价类
20a734	输入无效	④
20207	输入无效	⑤
20200624	输入无效	⑥
199912	输入无效	⑦
210101	输入无效	⑧
202000	输入无效	⑨
202020	输入无效	⑩

表 3-2 和表 3-3 中列出了 3 个覆盖所有有效等价类的用例,以及 7 个覆盖所有无效等价类的用例。

等价类与测试用例之间的关系可以是每一个测试用例覆盖一个特定的等价类(如表 3-3 中无效类的测试用例就都是一对一的),也可以是一个测试用例对应多个等价类(如表 3-2 中有效类的测试用例都是一对多的)。

请记住,等价分配的目标是把可能的测试用例组合缩减到仍然足以满足软件测试需求为止。因为选择不完全测试就要冒一定的风险,所以必须仔细选择分类。

关于等价分配需要说明的是,这样做有可能不客观。测试同一个复杂程序的两个软件测试员可能会制定出两组不同的等价区间,只要审查等价区间的人都认为它们足以覆盖测试对象就可以了。

3.3 边界值分析法

视频讲解

3.3.1 边界值分析法的概述

边界值分析法主要是在变量的输入域边界上或边界附近选择测试用例。其基本原理是许多错误倾向于发生在输入的极值附近,而非输入域的中间部分,因此,针对边界情况设计测试用例能够更有效地发现错误。例如,在做三角形计算时,要输入平面内三角形的 3 个边长 A、B 和 C。这 3 个数值应当满足 $A>0$、$B>0$、$C>0$、$A+B>C$、$A+C>B$、$B+C>A$,

这样才能在平面上构成三角形。但如果把6个不等式中的任何一个大于号“>”错写成大于或等于号“≥”,那就不能在平面上构成三角形。这类问题恰恰就出现在容易被疏忽的边界附近。这里所说的边界是相对于输入等价类和输出等价类而言,稍高于其边界值及稍低于其边界值的一些特定情况。该技术的一个扩展是稳健性测试,即在变量的输入域外部选择测试用例以测试程序针对未期望的错误输入的稳健性。

边界值分析法是一种补充等价类划分法的黑盒测试方法,它不是选择等价类中的任意元素,而是选择等价类边界的测试用例。实践证明,这些测试用例往往能取得很好的测试效果。边界值分析法不仅重视输入范围边界,也会从输出范围中导出测试用例。

通常情况下,软件测试所包含的边界条件有数字、字符、位置、质量、大小、速度、方位、尺寸、空间等几种类型;对应的边界值应该有最大/最小、首位/末位、上/下、最大/最小、最快/最慢、最高/最低、最短/最长、空/满等情况。

用边界值分析法设计测试用例,应遵循以下几条原则。

(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值,以及刚刚超越这个范围边界的值作为测试输入数据。

(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1的数、比最大个数多1的数等作为测试数据。

(3) 根据规格说明的每个输出条件,使用前面的原则(1)。

(4) 根据规格说明的每个输出条件,使用前面的原则(2)。

(5) 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。

(6) 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边界上的值作为测试用例。

(7) 分析规格说明,找出其他可能的边界条件。

表3-4给出了一些常见的确定边界值附近数据的方法。

表3-4 确定边界值附近数据的几种方法

项	边界值附近数据	测试用例的设计思路
字符	起始-1个字符/结束+1个字符	假设一个文本输入区域要求允许输入1～255个字符,输入1个和255个字符作为有效等价类;不输入字符(0个)和输入256个字符作为无效等价类
数值范围	开始位-1/结束位+1	如数据输入域为1～999,其最小值为1,而最大值为999,则0和1000刚好在边界值附近。从边界值方法来看,要测试4个数据:0、1、999、1000
空间	比零空间小一点/比满空间大一点	如测试数据的存储,使用比剩余磁盘空间大几千字节的文件作为测试的边界条件附近值

3.3.2 边界条件

边界条件是一种特殊情况,因为程序的开发从根本上不怀疑边界会有问题。奇怪的是,程序在处理大量中间数值时都是对的,但是很可能会在边界处出现错误。下面的一段源代码说明了在一个极简单的程序中边界条件问题是如何产生的。

```
1    rem create a 10 element integer array
2    rem initialize each element to - 1
3    dim data(10)as integer
4    dim i as integer
5    for i = 1 to 10
6      data(i) =  - 1
7    next i
8    end
```

这段代码的意图是创建包含 10 个元素的数组，并为数组中的每一个元素赋初值－1。看起来其相当简单，它建立了包含 10 个整数的数组 data 和一个计数值 i。for 循环的计数器是 1～10，数组中从第 1 个元素到第 10 个元素均被赋予数值－1。那么边界问题在哪儿呢？

在大多数编程语言脚本中，都应当以声明的范围来定义数组，在本例中定义语句是 dim data(10)as integer，创建的第一个元素是 data(0)而不是 data(1)。该程序实际上创建了一个 data(0)～ data(10)共包含 11 个元素的数组。程序进行 1～10 的循环将数组元素的值初始化为－1，但是由于数组的第一个元素 data(0)的索引号是 0 而不是 1，因此它并没有被初始化。程序执行完毕后，数组应值如下：

```
data(0) = 0          data(4) = - 1          data(8) = - 1
data(1) = - 1        data(5) = - 1          data(9) = - 1
data(2) = - 1        data(6) = - 1          data(10) = - 1
data(3) = - 1        data(7) = - 1
```

很明显，data(0)的值是 0 而不是－1。如果这位程序员以后忘记了，或者其他程序员不知道这个数据数组是如何初始化的，那么他就可能会用到数组的第 1 元素 data(0)，并理所当然地认为它的值是－1。诸如此类的问题在软件开发中很常见，在复杂的大型软件中，这就可能导致极其严重的软件缺陷。

3.3.3 次边界条件

上面讨论的普通边界条件是容易被找到的，它们在相关文档中有定义，或者在使用软件的过程中被确定。而有些边界在软件内部存在，最终用户几乎看不到，但是软件测试时仍有必要检查这类边界问题。这样的边界条件被称为次边界条件或者内部边界条件。

寻找这样的边界不要求软件测试员具有程序员那样阅读源代码的能力，但是要求他们大体了解软件的工作方式。像 2 的 N 次方和 ASCII 码表就是这样的典型例子。

1. 2 的 N 次方

计算机和软件的计数基础是二进制数，其用位(bit)来表示 0 和 1，1 字节(byte)由 8 位组成，一个字符(word)由 2 字节组成等。表 3-5 中列出了常用的 2 的乘方单位及其范围或值。

表 3-5 软件中 2 的乘方

术　语	范围或值	术　语	范围或值
位	0 或 1	千	1024
双位	0～15	兆	1048576
字节	0～255	亿	1073741824
字	0～65535	万亿	1099511627776

表 3-5 中所列的取值和取值范围是作为边界条件的重要数据。除非软件向用户提出这些范围,否则在需求文档中通常不会被指明,但在测试的时候需要考虑这些边界会否产生软件缺陷,在建立等价区间时,要考虑是否需要包含 2 的 N 次方边界条件。

例如,如果软件接受用户输入 1～1000 范围内的数字,明显可知输入域的合法区间中包含 1 和 1000,也许还要有 2 和 999。但考虑到输入的数字必然是以某种数据类型存储在计算机中的,那么为了覆盖任何可能的 2 的 N 次方边界,还要考虑包含邻近双位边界的 14、15 和 16,以及邻近字节边界的 254、255 和 256。

2. ASCII 码表

ASCII 码是不同计算机在相互通信时共同遵守的字符编码国际通行标准,它使用单字节对常见的文本字符进行了编码,从 0 开始连续排列数字,每个数字对应一个字符。如表 3-6 所示就是从 ASCII 码表中节选的部分内容。

表 3-6　部分 ASCII 码表

字符	ASCII 值	字符	ASCII 值	字符	ASCII 值	字符	ASCII 值
/	47	8	56	@	64	y	121
0	48	9	57	A	65	z	122
1	49	:	58	B	66	{	123

在 ASCII 码表中,常用的 0～9 数字所对应的 ASCII 值是 48～57,大写字母 A～Z 对应 65～90,小写字母对应 97～122。按照 ASCII 码顺序,字符“/”在数字 0 的前面,而字符“:”在数字 9 的后面;字符“@”在大写字母 A 的前面,而字符“[”在大写字母 Z 的后面;字符“`”在小写字母 a 的前面,而字符“{”在小写字母 z 的后面。这些情况都代表了次边界条件。

测试进行文本输入或文本转换的软件,在定义数据区间包含哪些值时,如果测试的文本框只接受用户输入字符 A～Z 和 a～z,就应该在非法区间中包含 ASCII 表中这些字符前后的值。

3.3.4　边界值分析法的测试用例

现有一个标准化考试批阅学生试卷,产生成绩报告的程序。其规格说明如下:程序的输入文件由 80 个字符的记录组成,如图 3-1 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 1～3 个字符为题目编号(取值为 1～999)。第 10～59 个字符给出第 1～50 题的答案(每个合法字符表示一个答案)。该组的第 2 个,第 3 个……记录相应为第 51～100 题,第 101～150 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 1～9 字符给出学生姓名及学号,第 10～59 字符列出的是甲所做的第 1～50 题的答案。若试题数超过 50,则第 2 个,第 3 个……记录分别给出他的第 51～100 题,第 101～150 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。

① 按学号排列的成绩单,列出每个学生的成绩、名次。

② 按学生成绩排列的成绩单。

③ 平均分数及标准偏差的报告。

④ 试题分析报告。按试题号排序,列出各题学生答对的百分比。

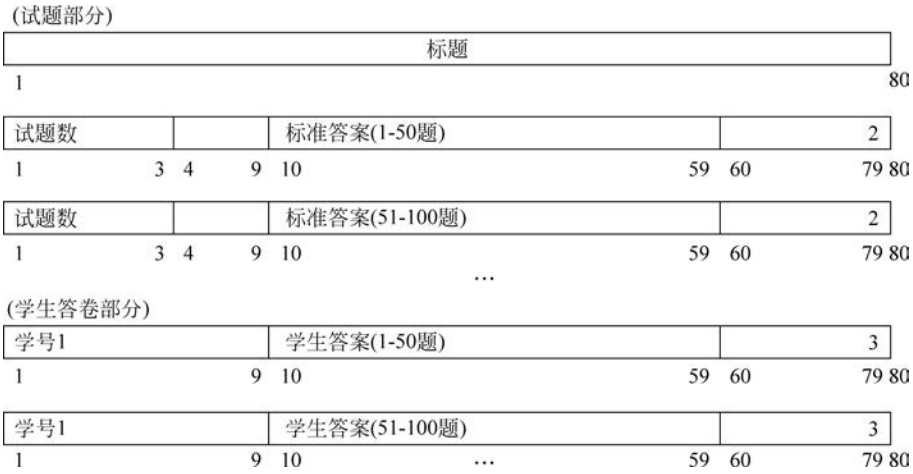


图 3-1　记录结构图

解答:分别考虑输入条件和输出条件以及边界条件。给出如表 3-7 所示的输入条件及相应的测试用例。

表 3-7　输入条件测试用例表

输 入 条 件	测 试 用 例
输入文件	空输入文件
标题	没有标题 标题只有 1 个字符 标题有 80 个字符
试题数	试题数为 1 试题数为 50 试题数为 51 试题数为 100 试题数为 0 试题数含非数字字符
标准答案记录	没有标准答案记录,有标题 标准答案记录多于 1 个 标准答案记录少于 1 个
学生人数	0 个学生 1 个学生 200 个学生 201 个学生
学生答题	某学生只有 1 个回答记录,但有 2 个标准答案记录 该学生是文件中的第 1 个学生 该学生是文件中的最后 1 个学生(记录数出错的学生)
学生答题	某学生只有 2 个回答记录,但有 1 个标准答案记录 该学生是文件中的第 1 个学生(记录数出错的学生) 该学生是文件中的最后 1 个学生

续表

输 入 条 件	测 试 用 例
学生成绩	所有学生的成绩都相同 每个学生的成绩都不相同 部分学生的成绩相同 (检查是否能按成绩正确排名次) 有个学生 0 分 有个学生 100 分

输出条件及相应的测试用例如表 3-8 所示。

表 3-8　输出条件测试用例表

输 入 条 件	测 试 用 例
输出报告 a、b	有个学生的学号最小(检查按序号排列是否正确) 有个学生的学号最大(检查按序号排列是否正确) 适当的学生人数,使产生的报告刚好满 1 页(检查打印页数) 学生人数比刚才多出 1 人(检查打印换页)
输出报告 c	平均成绩 100 平均成绩 0 标准偏差为最大值(有一半得 0 分,其他 100 分) 标准偏差为 0(所有成绩相等)
输出报告 d	所有学生都答对了第 1 题 所有学生都答错了第 1 题 所有学生都答对了最后 1 题 所有学生都答错了最后 1 题 选择适当的试题数,使第 4 个报告刚好打满 1 页 试题数比刚才多 1,使报告打满 1 页后,刚好剩下 1 题未打

视频讲解

3.4　决 策 表 法

3.4.1　决策表法的概念

决策表(也称判定表)是软件工程实践中的重要工具,主要用在软件开发的详细设计阶段,能表示输入条件的组合以及与每一输入组合相对应的动作组合。

决策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。在所有的黑盒测试方法中,基于决策表的测试是最严格也最具有逻辑性的测试方法。在软件行业发展的初期,决策表就已被用作编写程序的辅助工具了。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,使其指导开发的程序能针对不同的逻辑条件组合值分别执行不同的操作。

决策表的优点在于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,利用决策表能够设计出完整的测试用例集合。

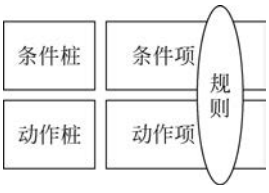


图 3-2　决策表构造形式

1. 决策表的组成

决策表的构造形式如图 3-2 所示,其通常由以下 5 部分组成。

条件桩:列出问题的所有条件。通常认为其列出的条件的

次序无关紧要。

动作桩：列出问题规定可能采取的操作。这些操作的排列顺序没有被约束。

条件项：列出针对它所列条件的取值，在所有可能情况下的真假值。一般来说，条件项的个数数量庞大，例如，问题有 5 个条件，每个条件有 2 个取值，条件项的个数就是 $2^5 = 32$ 个。

动作项：列出在条件项的各种取值情况下应该采取的动作，动作项的数目与条件项相等。

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。在决策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，也就确定了其包含有多少条规则，也确定了条件项和动作项有多少列。

2. 决策表的构造

可以依据软件的规格说明，按照如下步骤建立决策表。

(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)，则有 2^n 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，制定初始决策表。

(5) 简化决策表，合并相似的规则或者相同的动作。

建立决策表后，可针对决策表中的每一列有效规则设计一个测试用例，用于对程序进行黑盒测试。

实际使用决策表时需要将相似的规则简化与合并。若表中有两条以上规则具有相同动作，且在条件项之间存在相似关系，就可以对其进行合并。如图 3-3 所示，条件项的前两个条件取值一致，只有第 3 个条件取值不同，这表明前两个条件分别取真值和假值时，无论第 3 个条件取何值都要执行同一种操作，故这两条规则可以合并。合并后第 3 个条件项用符号“—”表示与取值无关，其通常被称为“无关条件”。与此类似，可进一步合并具有相同动作的规则。

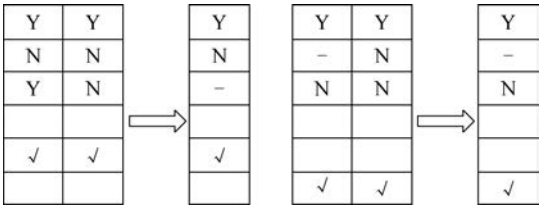


图 3-3　规则的合并

以支票的借记功能为例，输入是借记金额、账户类型和当前余额，输出是新的余额和操作代码。账户类型是邮政(“p”)类型或柜台(“c”)类型。操作代码是“D&L”“D”“S&L”或“L”，分别对应“处理借记并发送信件”“只处理借记”“冻结账户和发送信件”和“只发送信件”。该功能的描述如下：如果账户中有足够的金额或者新的余额在授权透支的范围内，则处理借记。如果新的余额超过了授权透支的范围，则不处理借记，如果是一个邮政账户则进行冻结。邮政账户的所有交易都会发送信件，非邮政账户如果有足够的资金也会发送信件(即账户将不再是信贷)。

那么，从功能描述中可以得到的条件桩(C)一共有 3 个。C1：账户中有足够金额；C2：

新的透支余额在授权范围内；C3：账户是邮政类型。得到所有的动作桩(A)也是一共 3 个。A1：处理借记；A2：冻结账户；A3：发送信件。

据此建立决策表如表 3-9 所示。

表 3-9　支票借记功能的决策表

		判定规则							
		1	2	3	4	5	6	7	8
条件桩	C1：账户中有足够金额	F	F	F	F	T	T	T	T
	C2：新的透支余额在授权范围内	F	F	T	T	F	F	T	T
	C3：账户是邮政类型	F	T	F	T	F	T	F	T
动作桩	A1：处理借记	F	F	T	T	T	T	*	*
	A2：冻结账户	F	T	F	F	F	F	*	*
	A3：发送信件	T	T	T	T	F	T	*	*

在本决策表中，每一列是一个判定规则。实际操作中也可以把整个决策表转置过来，用行的形式来显示这些判定规则。

该表包含了两部分。在第一个部分中，每个判定规则对应多个条件。“T”表示对使用的判定规则来说条件必须是真的；“F”表示对使用的判定规则来说条件必须是假的。第二部分中，每个判定规则对应多个动作。“T”表示动作将被执行；“F”表示动作不会被执行；“*”表示该条件的组合是无效的，因此该判定规则没有对应的动作。如果两个或多个列都包含了一个不会影响结果的布尔条件，则可以对其进行合并。本例对应的决策表有 8 条判定规则，其中 6 条是有效的，因此可得到 6 个测试输入条件组合。

最后从决策表中一次选择一个或者多个有效的判定规则来导出测试用例，确保当前导出的这些规则未被已生成的测试用例覆盖，以确定测试判定规则的条件和动作的输入值，测试用例其他的输入变量取任意有效值，确定预期结果，重复上述步骤直到达到覆盖要求为止，然后汇集这些测试用例成为对应该项待测功能的一个测试集。

3.4.2　决策表法的测试用例

问题要求：对功率大于 50 马力的机器、维修记录不全或已运行 10 年以上的机器，应给予优先的维修处理。这里假定，“维修记录不全”和“优先维修处理”均已在别处有更严格的定义。请建立决策表。

解题步骤如下。

(1) 确定规则的个数。

这里有 3 个条件，每个条件有两个取值，故应有 $2^3=8$ 种规则。

(2) 列出所有的条件桩和动作桩，如表 3-10 所示。

表 3-10　条件桩和动作桩

条件桩	功率大于 50 马力吗？
	维修记录不全吗？
	运行超过 10 年吗？
动作桩	进行优先处理
	作其他处理

(3) 填入条件项。

可从最后 1 行条件项开始逐行向上填满。如第三行是 YNYNYNYN,第二行是 YYNNYYNN。

(4) 填入动作桩和动作项。

这样得到如表 3-11 所示的初始决策表。

表 3-11　初始决策表

条件桩和动作桩		判定规则							
		1	2	3	4	5	6	7	8
条件桩	功率大于 50 马力吗?	Y	Y	Y	Y	N	N	N	N
	维修记录不全吗?	Y	Y	N	N	Y	Y	N	N
	运行超过 10 年吗?	Y	N	Y	N	Y	N	Y	N
动作桩	进行优先处理	√	√	√		√		√	
	作其他处理				√		√		√

(5) 简化决策表。

根据合并相似规则,初始决策表中的部分规则可以进行合并。例如规则 1 和规则 2 可以合并,规则 3、5、7 可以合并,规则 6、8 可以合并。合并后得到简化决策表,如表 3-12 所示。

表 3-12　简化决策表

条件桩和动作桩		判定规则			
		1	2	3	4
条件桩	功率大于 50 马力吗?	Y	—	Y	N
	维修记录不全吗?	Y	—	N	—
	运行超过 10 年吗?	—	Y	N	N
动作桩	进行优先处理	√	√		
	作其他处理			√	√

3.4.3　决策表的适用范围

决策表能把复杂的问题按各种可能发生的情况一一列举出来,简明而易于理解,也可避免遗漏。但决策表不能表达重复执行的动作,例如循环结构。

决策表测试法适用于具有以下特征的应用程序。

(1) 规格说明本身即以决策表的形式给出,或很容易转换成决策表。

(2) 条件的排列顺序不影响执行哪些操作。

(3) 规则的排列顺序不影响执行哪些操作。

(4) 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。

(5) 如果某一规则要执行多个操作,这些操作的执行顺序无关紧要。

(6) if-then-else 逻辑突出。

(7) 输入变量之间存在逻辑关系。

(8) 涉及输入变量子集的计算;输入与输出之间存在因果关系。

(9) 适用于使用决策表设计测试用例的条件。

3.5 因果图法

3.5.1 因果图法的概念

视频讲解

如果程序的输入条件之间相互存在联系,那么就会使情况变得复杂,因为要检查输入条件的组合情况并不是一件容易的事情,即使把所有输入条件划分为等价类,它们之间的组合情况也相当多,难以分析。因此,必须考虑采用因果图法,这种方法可以针对多种条件组合、产生多个动作的情况来设计测试用例。

因果图法是用逻辑式描述程序的输入条件(原因)和输出条件(结果),同时,用制约条件描述输入条件间的依赖关系的一种方法,其特征在于图式记述。因果图法是软件测试中一种重要的方法,它是由美国 IBM 公司的 EIemendorf 在吸收硬件测试中自动生成逻辑组合电路测试等技术的基础上于 1973 年提出的,作为功能测试常用方法之一,它将功能说明书转换为形式化表达。

因果图法基于这样一种思想:一些程序的功能可以用决策表的形式来表示,并根据输入条件的组合情况规定相应的操作。因此,可以考虑为决策表中每一列设计一个测试用例,以便测试程序在输入条件的某种组合下的输出十分正确。概括地说,因果图法就是从程序规格说明书的描述中找出因(输入条件)和果(输出结果或程序状态的变化)的关系,通过因果图转换为决策表,最后为决策表中的每一列设计一个测试用例。这种方法能够考虑到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,适合检查程序输入条件的各种组合情况,通过映射同时发生相互影响的多个输入来确定判定条件。

3.5.2 因果图的画法

因果图法是一种黑盒测试方法,它能够帮助人们按照一定的步骤高效地选择测试用例,同时还能指出需求规格说明书中存在的不完整性和二义性。

1. 输入条件与输出结果之间的因果关系

如图 3-4 描述了输入条件与输出结果间的关系,即"因果关系"。这种关系共有 4 种:恒等、非、或、与。

恒等:若原因出现,则结果出现;若原因不出现,则结果也不出现。例如,若 a=1,b=1;若 a=0,则 b=0。

非:若原因出现,则结果不出现;若原因不出现,则结果出现。例如,若 a=1,则 b=0;若 a=0,则 b=1。

或:若几个原因中有一个出现,则结果出现;若几个原因都不出现,则结果不出现。例如,若 a=1 或 b=1 或 c=1,则 d=1;若 a=b=c=0,则 d=0。

与:若几个原因都出现,结果才出现;若其中有一个原因不出现,则结果不出现。例如,若 a=b=c=1,则 d=1;若 a=0 或 b=0 或 c=0,则 d=0。

2. 输入或输出的约束关系

如图 3-5 所示,输入状态之间还可能存在某些依赖关系,或输出结果之间相互制约,这被称为约束。其所描述的这种制约关系一般可被分为 5 类:互斥、包含、唯一、要求和屏蔽。

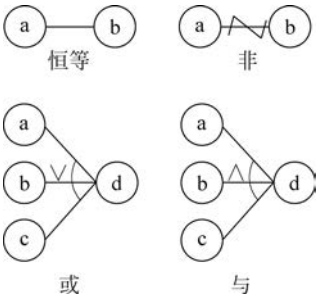


图 3-4　因果图逻辑符号

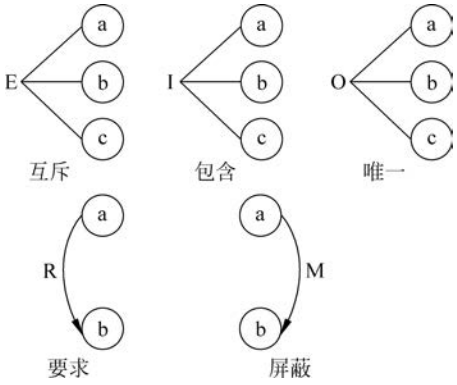


图 3-5　因果图的约束符号

其中，输入条件的约束有以下 4 类。

互斥(E)：表示 a、b、c 这 3 个原因不会同时成立，最多有一个可能成立。

包含(I)：表示 a、b、c 这 3 个原因中至少有一个必须成立。

唯一(O)：表示 a、b、c 中必须有一个成立，且仅有一个成立。

要求(R)：表示当 a 出现时，b 必须也出现。例如，若 a=1，则 b 必须为 1。

而其中的输出条件约束类型只有 1 种，即屏蔽(M)：若 a=1，则 b 必须为 0；而当 a=0 时，b 的值不定。

3. 因果图法设计测试用例的步骤

(1) 分析在程序规格说明的描述中哪些是原因，哪些是结果，并给每个原因和结果赋予一个标识符。原因常常是输入条件或是输入条件的等价类，而结果则是输出条件。

(2) 分析在程序规格说明的描述中语义的内容，并将其表示成连接各个原因与各个结果的“因果图”。

(3) 标明约束条件。由于语法或环境的限制，有些原因和结果的组合情况是不可能出现的。为表明这些特定的情况，可在因果图上使用若干个标准的符号标明约束或限制条件。

(4) 把因果图转换成决策表。

(5) 把决策表中每一列拿出来作为依据，设计测试用例。

因果图生成的测试用例(局部，组合关系下的)包括了所有输入数据的取 True 与取 False 的情况，其构成的测试用例数目达到最少，且测试用例数目随输入数据数目的增加而增加。在较为复杂的问题中，这个方法常常十分有效，它能有力地帮助测试者确定测试用例。当然，如果开发项目在设计阶段就采用了决策表，那么也就不必再画因果图了，此时可以直接利用决策表设计测试用例。

3.5.3　因果图法的测试用例

图 3-6 所示的是交通一卡通自动充值模拟系统，其需求描述如下。

(1) 系统只接收 50 元或 100 元纸币，一次充值只能使用一张纸币，一次充值金额只能为 50 元或 100 元。

(2) 若输入 50 元纸币，并选择充值 50 元，完成充值后退卡，提示充值成功。

(3) 若输入 50 元纸币，并选择充值 100 元，提示输入金额不足，并退回 50 元。

(4) 若输入 100 元纸币，并选择充值 50 元，完成充值后退卡，提示充值成功，找零 50 元。

(5) 若输入 100 元纸币,并选择充值 100 元,完成充值后退卡,提示充值成功。

(6) 若输入纸币在规定时间内不选择充值按钮,退回输入的纸币,并提示错误。

(7) 若选择充值按钮后不输入纸币,提示错误。

图 3-6　交通一卡通自动充值模拟系统

下面给出交通一卡通自动充值系统的测试用例设计过程。

1) 条件之间的制约及组合关系

根据上述描述,输入条件(原因)如下。

(1) 投币 50 元(1)。

(2) 投币 100 元(2)。

(3) 选择充值 50 元(3)。

(4) 选择充值 100 元(4)。

输出(结果)如下。

(1) 完成充值、退卡(a)。

(2) 提示充值成功(b)。

(3) 找零(c)。

(4) 提示错误(d)。

2) 明确所有条件之间的制约关系及组合关系

条件之间的制约关系及组合关系如图 3-7 所示。

3) 画出因果图

为了描述得更清楚,这里将每种情况单独画一个因果图说明。

(1) 条件 1 和条件 3 可以组合,输出 a 和 b 的组合,也就是投币 50 元,充值 50 元,会输出完成充值、退卡,提示充值成功的结果。其因果图如图 3-8 所示。

(2) 条件 1 和条件 4 可以组合,输出 c 和 d 的组合,也就是投币 50 元,充值 100 元,会输出找零、提示错误的结果。其因果关系如图 3-9 所示。

(3) 条件 2 和条件 3 可以组合,输出 a、b、c 的组合,也就是投币 100 元,充值 50 元,会输出找零、完成充值、提示充值成功的结果。其因果图如图 3-10 所示。

(4) 条件 2 和条件 4 可以组合,输出 a 和 b 的组合,也就是投币 100 元,充值 100 元,会输出完成充值、退卡,提示充值成功的结果。其因果图如图 3-11 所示。

(5) 条件 1、2、3、4 均可以单独出现,其因果图如图 3-12 所示。

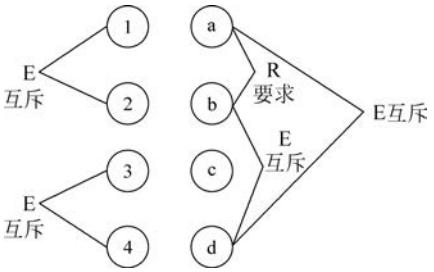


图 3-7　条件之间的约束关系

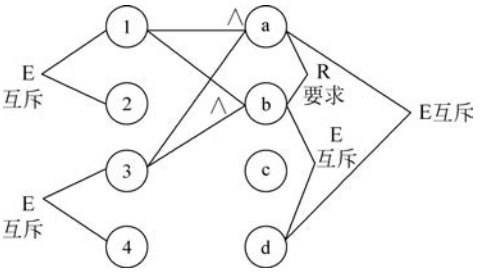


图 3-8　条件 1 和条件 3 的组合

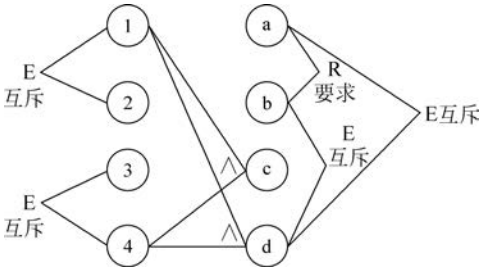


图 3-9　条件 1 和条件 4 的组合

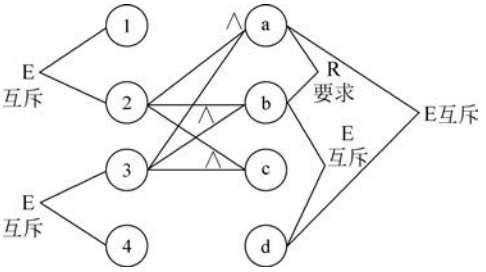


图 3-10　条件 2 和条件 3 的组合

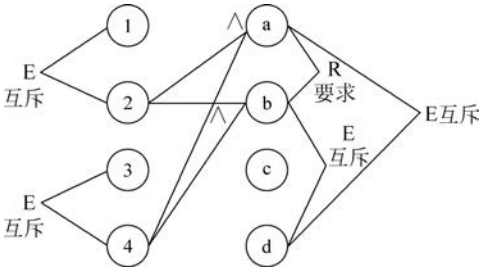


图 3-11　条件 2 和条件 4 的组合

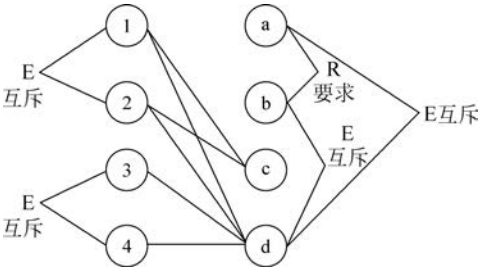


图 3-12　各条件单独出现

4）根据因果图写出决策表

根据上面的因果图，写出对应的决策表，如表 3-13 所示。

表 3-13　决策表

		判 定 规 则							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
条件桩	1. 投币 50 元	1	1			1			
	2. 投币 100 元			1	1		1		
	3. 选择充值 50 元	1		1				1	
	4. 选择充值 100 元		1		1				1
动作桩	a. 完成充值、退卡	1		1	1				
	b. 提示充值成功	1		1	1				
	c. 找零		1	1		1	1		
	d. 错误提示		1			1	1	1	1

5) 根据决策表写出测试用例

根据上面的决策表,写出对应的测试用例,如表 3-14 所示。

表 3-14　交通一卡通自动充值模拟系统测试用例

编号	用 例 说 明	预 期 结 果
1	投币 50 元 选择充值 50 元	正确充值 50 元,提示充值成功后退卡
2	投币 50 元 选择充值 100 元	系统提示错误并退回 50 元
3	投币 100 元 选择充值 50 元	正确充值 50 元,提示充值成功后退卡,并找回 50 元
4	投币 100 元 选择充值 100 元	正确充值 100 元,提示充值成功后退卡
5	投币 50 元	系统提示错误并退回 50 元
6	投币 100 元	系统提示错误并退回 100 元
7	选择充值 50 元	系统提示错误
8	选择充值 100 元	系统提示错误

视频讲解

3.6　其他黑盒测试方法

3.6.1　正交试验法

正交试验设计起源于科学试验,它应用依据 Galois 理论导出的正交表,从大量试验条件中挑选出适量的、有代表性的条件来合理地安排试验。运用这种方法安排的试验具有“均匀分散、整齐可比”的特点。其均匀分散性使试验点均衡地分布在试验范围内,让每个试验点有充分的代表性;而其整齐可比性则使试验结果在分析应用上十分方便,可以用于估计各因素对指标的影响,找出影响事物变化的主要因素。实践证明,正交试验设计是一种解决多因素试验问题卓有成效的方法。

利用因果图来设计测试用例时,有时很难从软件需求规格说明中得到作为输入条件的原因与输出结果之间的因果关系。这类因果关系往往非常庞大,于是便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。

1. 正交试验方法

正交试验法就是使用已经造好了的表格——正交表来安排试验并进行数据分析的一种方法。它简单易行并且能使计算表格化,应用性较好。下边通过一个例子来说明正交试验法。

为提高某化工产品的转化率,选择三个有关因素进行条件试验,即反应温度(A),反应时间(B),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,从而确定最佳生产条件,即温度、时间及用碱量各为多少才能使转化率最高。这里,对因子 A、

B 和 C，在试验范围内都选三个水平，分别如下。

A：A1＝80℃，A2＝85℃，A3＝90℃。

B：B1＝90min，B2＝120min，B3＝150min。

C：C1＝5％，C2＝6％，C3＝7％。

当然，在正交试验设计中，因子可以是定量的，也可以是定性的。而定量因子各水平间的距离可以相等，也可以不相等。这个三因子各三个水平的条件试验，通常有两种试验方法。

（1）取三因子所有水平之间的组合，即 A1B1C1，A1B1C2，A1B2C1，…，A3B3C3，共有 $3^3=27$ 次试验。用图 3-13 表示立方体的 27 个节点可以清晰地展示这种关系。这种试验法叫作全面试验法。

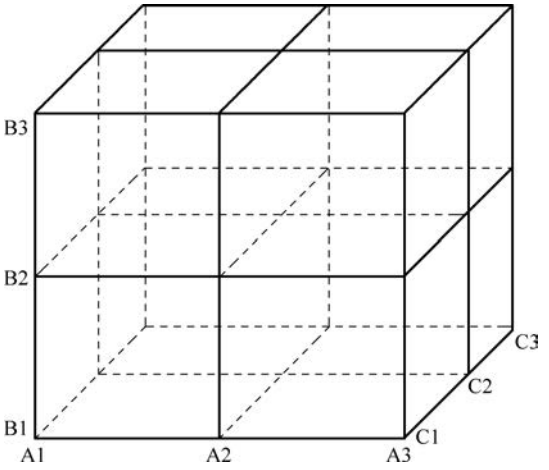
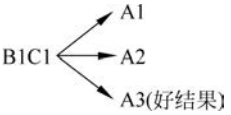


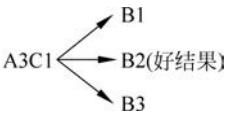
图 3-13　全面试验法取点

全面试验对各因子与指标间的关系剖析得比较清楚，但其所需试验次数太多。特别是当因子数目多，每个因子的水平数目也很多时，试验量会非常大。如选 6 个因子，每个因子取 5 个水平时，如做全面试验，则需 $5^6=15625$ 次试验，这实际上是不可能实现的。但如果应用将要介绍的正交试验法，则只做 25 次试验就行了，而且从某种意义上讲，这 25 次试验能够代表 15625 次试验。

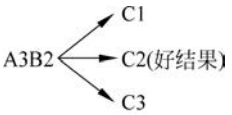
（2）简单对比法，即变化一个因素而固定其他因素，如首先固定 B、C 于 Bl、Cl，使 A 变化如下所示。



（3）如得出结果 A3 最好，则固定 A 于 A3，C 还是 C1，使 B 变化如下所示。



（4）得出结果以 B2 为最好，则固定 B 于 B2，A 于 A3，使 C 变化如下所示。



试验结果以 C2 为最好。于是可以认为最好的工艺条件是 A3B2C2。

这种方法也有一定的效果,但缺点很多。首先这种方法的选点代表性很差,如按上述方法进行试验,试验点完全分布在一个角上,而在一个很大的范围内没有选点,因此这种试验方法并不全面,所选的工艺条件 A3B2C2 不一定是 27 个组合中最好的。其次,用这种方法比较条件好坏时,是把单个的试验数据拿来进行数值上的简单比较,而试验数据中必然包含着误差成分,所以单个数据的简单比较不能剔除误差,这就必然造成结论的不稳定。

简单对比法的最大优点就是试验次数少,例如,6 因子 5 水平试验,在不重复时,只用 5+(6-1)×(5-1)=5+5×4=25 次试验就可以了。

考虑兼顾这两种试验方法的优点,可以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。但同时人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 Al、A2、A3 这 3 个平面,对应于 B、C 也各有 3 个平面,共 9 个平面。则这 9 个平面上的试验点都应当一样多,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。这样,做出如图 3-14 所示的设计,试验点用⊙表示。从图中可看出,在 9 个平面中每个平面上都恰好选出 3 个点,而每个平面的每行每列都有 1 个点,而且只有 1 个点,总共 9 个点。这样的试验方案,试验点的分布得很均匀,试验次数也不多。

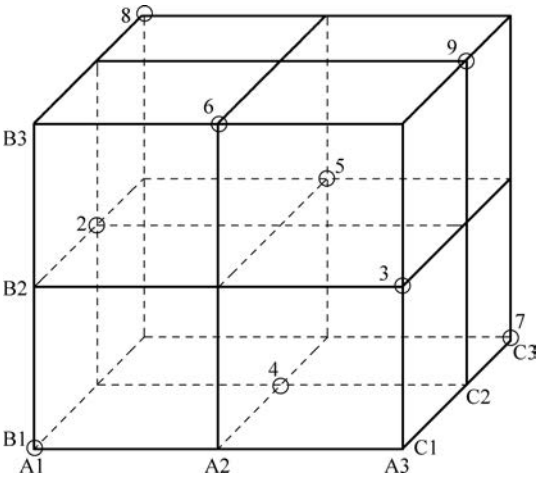


图 3-14　正交试验设计案例

当因子数和水平数都不太大时,尚可通过做图的办法来选择分布很均匀的试验点。但是因子数和水平数多了,做图的方法就不行了。试验工作者在长期的工作中总结出一套办法,创造出所谓的正交表。按照正交表来安排试验,既能使试验点分布得很均匀,又能减少试验次数,而且计算分析简单,能够清晰地阐明试验条件与指标之间的关系。用正交表来安排试验及分析试验结果,这种方法叫正交试验法。

一般用 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。此符号各数字的意义如下。

例如:$L_8(2^7)$,其中,7 为此表列的数目(最多可安排的因子数);2 为因子的水平数;8 为此表行的数目(试验次数)。

又例如:$L_{18}(2\times3^7)$,有 7 列是 3 水平的,有 1 列是 2 水平的,$L_{18}(2\times3^7)$的数字告诉我们,用它来安排试验,做 18 个试验最多可以考察 1 个 2 水平因子和 7 个 3 水平因子。

在行数为 mn 型的正交表中(m,n 是正整数),试验次数(行数)=(每列水平数−1)+1,如 $L_8(2^7)$,8=7×(2−1)+1,利用上述关系式可以从所要考察的因子水平数来决定最低的试验次数,进而选择合适的正交表。例如要考察 5 个 3 水平因子及一个 2 水平因子,则起码的试验次数为 5×(3−1)+1×(2−1)+1=12(次),这就是说,要在行数不小于 12,既有 2 水平列又有 3 水平列的正交表中选择,故 $L_{18}(2\times3^7)$较为适合。正交表具有两条性质:每一列中各数字出现的次数都一样多;任何两列所构成的各有序数对出现的次数都一样多。

例如,在 $L_9(3^4)$中(如表 3-15 所示),各列中的 1、2、3 都各自出现 3 次;任何两列,例如第 3、4 列,所构成的有序数对从上向下共有 9 种,既没有重复也没有遗漏。其他任何两列所构成的有序数对也是这 9 种各出现一次,这便反映了试验点分布的均匀性。

表 3-15　$L_9(3^4)$正交表

行　　号	列号			
	1	2	3	4
	行号			
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

试验方案应该如何设计呢?安排试验时,只要把所考察的每一个因子任意地对应于正交表的一列(一个因子对应一列,不能让两个因子对应同一列),然后把每列的数字“翻译”成对应因子的水平。这样,每一行的各水平组合就构成了一个试验条件(不考虑没安排因子的列)。对于上例,因子 A、B、C 都是 3 水平的,试验次数要不少于 3×(3−1)+1=7(次),可考虑选用 $L_9(3^4)$正交表。因子 A、B、C 可任意地对应于 $L_9(3^4)$正交表的某三列,例如 A、B、C 分别放在 1、2、3 列,然后试验按行进行,顺序不限,每一行中各因素的水平组合就是每一次的试验条件,从上到下就是这个正交试验的方案,如表 3-16 所示。这个试验方案的几何解释正好是正交试验设计图例。

表 3-16　试验方案

行号	列号					试验号	水平组合	试验条件		
	A	B	C					温度(℃)	时间(min)	加碱量(%)
	1	2	3	4						
1	1	1	1	1		1	A1B1C1	80	90	5
2	1	2	2	2		2	A1B2C2	80	120	6
3	1	3	3	3		3	A1B3C3	80	150	7
4	2	1	2	3		4	A2B1C2	85	90	6
5	2	2	3	1	⇨	5	A2B2C3	85	120	7
6	2	3	1	2		6	A2B3C1	85	150	5
7	3	1	3	2		7	A3B1C3	90	90	7
8	3	2	1	3		8	A3B2C1	90	120	5
9	3	3	2	1		9	A3B3C2	90	150	6

3个3水平的因子做全面试验需要$3^3=27$次试验,现用$L_9(3^4)$正交表来设计试验方案只要做9次,工作量减少了2/3,而在一定意义上却能代表27次试验。

2. 正交试验测试用例设计步骤

(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作因子的状态。利用正交试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。然后,对软件需求规格说明中的功能要求进行划分,把整体的、概要性的功能要求层层分解与展开,分解成具体的、有相对独立性的基本功能要求,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作用的大小、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。

利用正交试验法设计测试用例,与使用等价类划分、边界值分析、因果图等方法相比有以下优点:节省测试工作工时;可控制生成的测试用例的数量;测试用例具有一定的覆盖率。

正交试验法在软件测试中是一种有效的方法,例如在平台参数配置方面,要选择最好的组合方式,每个参数可能就是一个因子,参数的不同取值就是水平,这样就可以采用正交试验法设计出最少的测试组合,达到有效的测试目的。

3.6.2 场景法

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了事件流。这种在软件设计领域的思想也可被引入到软件测试中,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,同时也使测试用例更容易被理解和执行。

场景测试使用软件与用户或其他系统之间的交互序列模型来测试软件的使用流程。测试条件是在测试中覆盖基本场景和可选场景(即用户和系统交互的事件流用序列组成一个场景)。

提出这种测试思想的是Rational公司,并在RUP2000中文版中有详尽的解释和应用。

1. 基本流和备选流

用例场景用来描述流经用例的路径,其将从用例开始到结束遍历这条路径上所有基本流和备选流,如图3-15所示。

基本流:采用黑直线表示,是经过用例的最简单路径,表示无任何差错,程序从开始执行到结束。

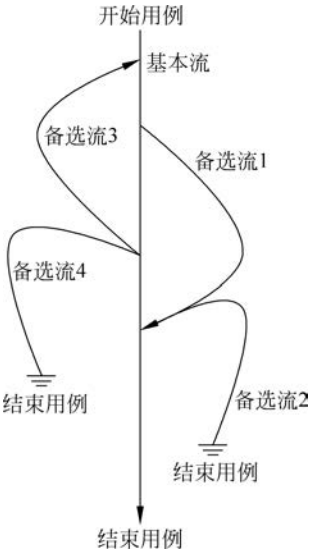


图3-15 基本流和备选流

备选流：采用不同颜色表示，一个备选流可以从基本流开始，在某个特定条件下执行，然后重新加入到基本流中，也可以起源于另一个备选流，或结束用例，不再加入到基本流中。

按照图 3-15 中所示的每个经过用例的路径，可以确定以下不同的用例场景。

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、6 和 8 只考虑了备选流 3 循环执行一次的情况。

需要说明的是，为了能清晰地说明场景，此处所举的例子都非常简单，在实际应用中，测试用例很少能如此简单。

2. 应用场景法进行测试的步骤

（1）根据规格说明，描述出程序的基本流和各个备选流。

（2）根据基本流和各个备选流生成不同的场景。

（3）对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进行复审，去掉多余的测试用例，对每一个测试用例确定测试数据。

3. 场景法测试案例

下面将以经典的 ATM 为例，介绍使用场景法来设计测试用例的过程。ATM 的取款流程场景分析如图 3-16 所示，其中灰色框构成的流程为基本流。

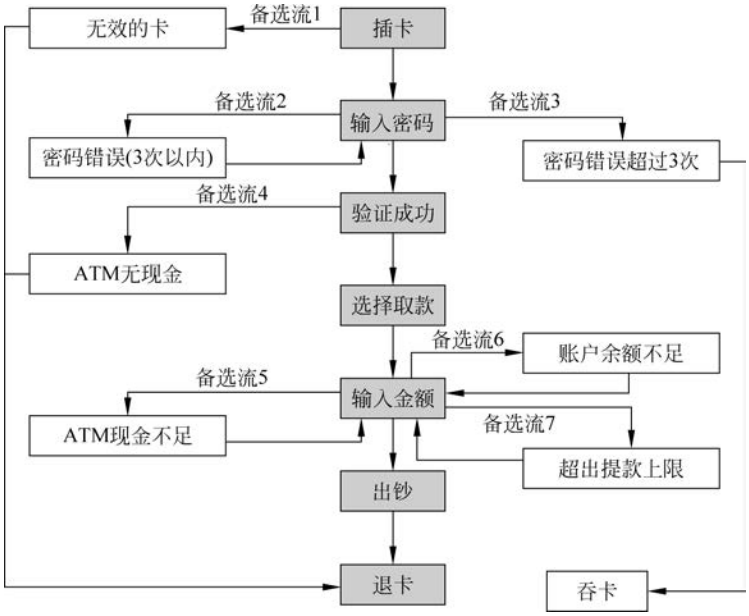


图 3-16　ATM 取款流程场景法分析

根据图中分析,得到该程序用例场景如表 3-17 所示。

表 3-17　用例场景

场　景	描　述	路　径	
场景 1	成功提款	基本流	
场景 2	无效卡	基本流	备选流 1
场景 3	密码错误 3 次以内	基本流	备选流 2
场景 4	密码错误超过 3 次	基本流	备选流 3
场景 5	ATM 无现金	基本流	备选流 4
场景 6	ATM 现金不足	基本流	备选流 5
场景 7	账户余额不足	基本流	备选流 6
场景 8	超出提款上限	基本流	备选流 7

接下来设计测试用例,并将其覆盖每个用例场景,如表 3-18 所示。

表 3-18　场景法测试案例

用例号	场景	账　号	密码	操　作	预 期 结 果
1	场景 1	621226××××××××3481	123456	插卡,取 500 元	成功取款 500 元
2	场景 2	—	—	插入一张无效卡	系统退卡,显示该卡无效
3	场景 3	621226××××××××3481	111111	插卡,输入密码	系统提示密码错误,请求重新输入
4	场景 4	621226××××××××3481	111111	插卡,输入密码超过 3 次	系统提示密码输入错误超过 3 次,卡被吞掉
5	场景 5	621226××××××××3481	123456	插卡,选择取款	系统提示 ATM 无现金,退卡
6	场景 6	621226××××××××3481	123456	插卡,取款 2000 元	系统提示现金不足,返回输入金额界面
7	场景 7	621226××××××××3481	123456	插卡,取款 3000 元	系统提示账户余额不足,返回输入金额界面
8	场景 8	621226××××××××3481	123456	插卡,取款 3500 元	系统提示超出取款上限(3000 元),返回输入金额界面

3.6.3　错误猜测法

1. 基本概念

错误猜测法又被称为错误推测法,是基于测试人员对以往项目测试中曾经发现的缺陷、故障或失效数据,在对软件错误原因进行的分析基础上设计测试用例,用于预测错误、缺陷和失效发生的技术。错误推测的结构化方法是基于测试人员丰富的经验,对软件错误的产生原因进行分析,构建缺陷或故障列表,并尝试设计产生缺陷或故障的测试用例的方法。错

误推测法能否成功主要取决于测试人员的技能。使用错误推测法与其他测试技术相结合，可以使测试人员更好地了解系统功能及其工作方式。根据对系统的理解，经验丰富的测试人员可以通过假设和推测找到系统无法正常运行的原因，发现更多的缺陷，有效地提高软件测试的效率。

2. 软件错误类型

软件错误是指软件的期望运行结果与实际运行结果之间存在差异的问题。错误推测法是测试人员在拥有丰富经验和知识，在对软件错误原因分析的基础上，预判错误和缺陷的技术。软件的错误分类在错误推测法的使用中尤为重要，其可以帮助测试人员根据软件错误的分类提前列出可能出现的缺陷，构建相关测试用例。

软件错误可分为以下类型。

1）软件需求错误

软件需求错误包括但不限于：

（1）软件需求不合理。

（2）软件需求不全面、不明确。

（3）需求中包含逻辑错误。

（4）需求分析的文档有误。

2）功能和性能错误

功能和性能错误包括但不限于：

（1）需求规格说明中规定的功能实现不正确、存在未实现或冗余的情况。

（2）性能未满足规定的要求。

（3）为用户提供的信息不准确。

（4）异常情况处理有误。

3）软件结构错误

软件结构错误包括但不限于：

（1）程序控制流或控制顺序有误。

（2）处理过程有误。

4）数据错误

数据错误包括但不限于：

（1）数据定义或数据结构有误。

（2）数据存取或数据操作有误。

5）软件实现和编码错误

软件实现和编码错误包括但不限于：

（1）编码错误或按键错误。

（2）违反编码要求和标准，例如语法错误、数据名错误、程序逻辑有误等。

6）软件集成错误

软件集成错误包括但不限于：

（1）软件的内部接口或外部接口有误。

（2）软件各相关部分在时间配合、数据吞吐量等方面不协调。

对于上述软件错误来说，软件结构错误、数据错误与功能和性能错误出现的频次较高也

最为普遍,所以更需要测试者给予充分的重视。

3. 估算错误数量的方法

为了保证错误推测法的有效实施,可以预先对错误数量进行估算。测试人员通过解软件中可能存在的错误数量,能够运用错误推测法有效地推测程序中所有可能存在的各种错误。以下介绍可以估算程序中可能存在错误数量的两个方法。

1) Seeding 模型估算法

1972 年 H. D. Mills 在估算软件的工作中引入了 Seeding 模型,用来预测及计算程序中可能存在的错误数量。Seeding 模型的工作原理为,在开始排错工作前,排错人员并不知道软件中的错误总数,此时可将软件中含有的未知错误数据记为 N,在此基础上,人为向程序中添加 N_t 个错误。经过 t 个月的排错工作以后,检查排错的清单,将排错类型分为两类,一类为程序中原有的错误,数量记为 n,另一类则是由排错人员人工插入的错误,数量记为 n_t。则预估该软件中错误总数 N 的方法为:

$$N=\frac{n}{n_t}N_t$$

从理论上来说,此方法可以直观地帮助排错人员估计程序中的错误数量,但在实际应用中,由于排错人员无法确定程序中出现错误的数量和内容,可能会导致新添加的错误与原有错误重复,所以其结果不一定准确。因此,此方法的效果并不理想。

Hyman 在 Mills 提出的 Seeding 模型基础上进行了改进,其成果被软件行业广泛应用。Hyman 估算方法为设置 A、B 两组测试人员相互独立地对某个软件进行测试,记 A 组人员和 B 组人员测得的错误数分别为 i 和 j 个,两组测试人员共同测试出的错误数为 k,软件错误数的估算值 $\hat{N}$ 与这三个量的关系如下:

$$\hat{N}=\frac{i*j}{k}$$

通过理论推导证明了 Hyman 估算方法的可靠性。

2) Shooman 模型估算法

Shooman 模型是一种通过估算错误产生的频度来保证软件可靠性的方法。估算错误产生的频度主要体现为估算平均失效等待时间 MTTF。因此,Shooman 模型估算 MTTF 的公式为:

$$\mathrm{MTTF}=\frac{I_\mathrm{T}}{K(E_\mathrm{T}-E_\mathrm{c}(t))}$$

其中,K 为经验常数;E_T 是测试之前程序中的原有故障总数;I_T 是程序长度(机器指令条数或简单汇编语句条数);t 是测试(包括排错)的时间;$E_\mathrm{C}(t)$是在 0~t 期间内检出并排除的故障总数。此公式中的 K 及 E_T 可通过两次以上不同的互相独立功能测试的结果进一步估算得到。在实际应用中,此方法估算 E_T 值和 K 的值较为困难,由此可能会导致实验数据存在误差。

Shooman 模型主要应用于软件的开发阶段,而 Seeding 模型主要应用于软件的测试阶段,通过对错误数量的估算能够有效地提升软件可靠性。

3.7 黑盒测试方法的选择

视频讲解

测试用例的设计方法不是单独存在的，具体到每个测试项目里都会用到多种方法，每种类型的软件、每种测试用例设计的方法都有各自的特点，针对不同软件利用这些黑盒方法非常重要。在实际测试中，往往是综合使用各种方法才能有效地提高测试效率和测试覆盖度，这就需要认真掌握这些方法的原理，积累更多的测试经验，以有效地提高测试水平。

以下是各种测试方法选择的综合策略，可供读者在实际应用过程中参考。

（1）首先进行等价类划分，包括输入条件和输出条件的等价类划分，将无限测试变成有限测试，这是减少工作量和提高测试效率最有效的方法。

（2）在任何情况下都必须使用边界值分析方法。经验表明，用这种方法设计出的测试用例发现程序错误的能力最强。

（3）可以用错误推测法追加一些测试用例，这需要依靠测试工程师的智慧和经验。

（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，则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决策表驱动法。

（6）对参数配置类的软件，要用正交试验法选择较少的组合方式达到最佳效果。

（7）对业务流清晰的系统，可以利用场景法贯穿整个测试案例过程，在案例中综合使用各种测试方法。

3.8 本章小结

黑盒测试又被称为功能测试，它主要关注被测软件功能的实现，而不是其内部逻辑。在黑盒测试中，被测对象的内部结构、运作情况对测试人员而言是不可见的，测试人员把被测软件系统看成一个黑盒子，并不关心盒子的内部结构和内部特性，而只关注输入数据和输出结果，以此检查软件产品是否符合它的功能说明。

本章重点介绍了几种常用的黑盒测试方法，并给出了相应的案例说明。

（1）等价类划分法。等价类划分法把程序的输入域划分为若干部分，然后从每个部分中选取少数具有代表性的数据作为测试用例，每一类代表性数据在测试中的作用等价于这一类中的其他值。

（2）边界值分析法。边界值分析法是一种补充等价类划分法的黑盒测试方法，它不是选择等价类中的任意元素，而是选择等价类边界值来编写测试用例。

（3）因果图法。因果图法是一种黑盒测试方法，它从自然语言书写的程序规格说明书中寻找因果关系，即寻找输入条件与输出和程序状态的改变，通过因果图产生决策表。

（4）决策表法。决策表是分析和表达多逻辑条件下执行不同操作情况的工具，其可以把复杂的逻辑关系和多种条件组合的情况表达得比较明确。

（5）其他黑盒测试方法，如正交实验法、场景法和错误猜测法等。

不同的黑盒测试方法有不同的应用场合，需要根据实际项目进行选择。

3.9 习　　题

1. 以下叙述中,不正确的是(　　)。
   A. 黑盒测试可以检测软件行为、性能等特性是否满足要求
   B. 黑盒测试可以检测软件是否有人机交互上的错误
   C. 黑盒测试依赖于软件内部的具体实现,如果实现发生了变化,则需要重新设计用例
   D. 黑盒测试用例设计可以和软件实现同步进行
2. 黑盒测试不能发现(　　)。
   A. 功能错误或者遗漏　　B. 输入输出错误
   C. 执行不到的代码　　D. 初始化和终止错误
3. 以下关于黑盒测试方法选择策略的叙述中,不正确的是(　　)。
   A. 首先进行等价类划分,因为这是提高测试效率最有效的方法
   B. 任何情况下都必须使用边界值分析,因为这种方法发现错误能力最强
   C. 如果程序功能说明含有输入条件组合,则一开始就需要错误推测法
   D. 如果没有达到要求的覆盖准则,则应该补充一些测试用例
4. 以下关于等价类划分法的叙述中,不正确的是(　　)。
   A. 如果规定输入值 a 的范围为 1～99,那么可得到两个等价类,即有效等价类{a|1≤a≤99},无效等价类{a|a<1 或者 a>99}
   B. 如果规定输入值 s 的第一个字符必须为数字,那么可得到两个等价类,即有效等价类{s|s 的第一个字符是数字},无效等价类{s|s 的第一个字符不是数字}
   C. 如果规定输入值 x 取值为 1,2,3 三个数之一,那么可得到 4 个等价类,有效等价类{x|x=l},{x|x=2},{x|x=3},无效等价类{x|x≠1,2,3}
   D. 如果规定输入值 i 为奇数,那么可得到两个等价类,即有效等价类{i|i 是奇数},无效等价类{i|i 不是奇数}
5. 以下关于等价类划分法的叙述中,不正确的是(　　)。
   A. 如果规定输入值 stringl 必须以'\0'结束,那么可得到两个等价类,即有效等价类{string1|string1 以'\0'结束},无效等价类{string1|stringl 不以'\0'结束)
   B. 如果规定输入值 intl 取值为 1、－1 两个数之一,那么可得到 3 个等价类,即有效等价类{int1|int1=1}、{int1|intl=－1},无效等价类{int1|int1≠1 并且 int1≠－1}
   C. 如果规定输入值 int2 的取值范围为－10～9,那么可得到两个等价类,即有效等价类{int2|－10≤－int2≤9},无效等价类{int2|int2 小于－10 或者 int2 大于 9}
   D. 如果规定输入值 int3 为质数,那么可得到两个等价类,即有效等价类{int3|int3 是质数},无效等价类{int3|int3 不是质数}
6. 以下关于边界值测试法的叙述中,不正确的是(　　)。
   A. 边界值分析法不仅重视输入域边界,也必须考虑输出域边界
   B. 边界值分析法是对等价类划分方法的补充
   C. 发生在输入输出边界上的错误比发生在输入输出范围内部的错误要少
   D. 测试数据应尽可能选取边界上的值,而不是等价类中的典型值或任意值

7. 假设需要计算员工的奖金。它不能是负数，但计算结果可以为零。奖金基于工作年限进行计算。分类情况有：小于或等于2年，超过2年但不到5年，超过5年，但不到10年，10年或更长。为计算奖金而覆盖所有有效等价类所需的最小测试用例数是(　　)。

A. 3　　B. 5　　C. 2　　D. 4

8. 健身应用程序测量每天的步数，并提供反馈以激励用户坚持健身。

不同的步数反馈如下。

最多1000：沙发土豆。

1000以上，最多2000：懒汉。

2000以上，最多4000：坚持。

4000以上，最多6000：不错。

6000以上：太棒了。

以下测试输入将获得最高的等价类划分覆盖率的是(　　)。

A. 0,1000,2000,3000,4000　　B. 1000,2001,4000,4001,6000

C. 123,2345,3456,4567,5678　　D. 666,999,2222,5555,6666

9. 植物的每日光照记录仪根据植物暴露在阳光下的时长(低于3小时、3小时到6小时或高于6小时)以及平均日照强度(极低、低、中、高)的组合生成日照评分。

根据下面的测试用例：

时　长		强　度	分　数
T1	1.5	极低	10
T2	7.0	中	60
T3	0.5	极低	10

为确保覆盖所有有效输入的等价类划分，至少需要增加测试用例(　　)。

A. 1　　B. 2　　C. 3　　D. 4

10. 视频应用具有以下需求，该应用允许播放以下显示分辨率的视频：

① 640×480；　② 1280×720；　③ 1600×1200；　④ 1920×1080。

以下哪个测试用例表是应用等价类测试技术来测试此需求的结果(　　)。

A. 验证应用是否可以在分辨率1920×1080(1个测试)的显示器上播放视频

B. 验证应用是否可以在分辨率640×480和1920×1080(2个测试)的显示器上播放视频

C. 验证应用是否可以在需求中的每个显示分辨率上播放视频(4个测试)

D. 验证应用是否可以在需求中的任一分辨率上播放视频(1个测试)

11. 智能家庭应用软件可以测量上一周的平均温度，并根据这个温度向住户反馈环境的舒适性。

不同平均温度范围(最接近的0℃)下住户提供的反馈如下。

最高10℃：冰凉。

11℃～15℃：冷。

16℃～19℃：凉爽。

20℃～22℃：有点热。

22℃以上：热得出汗。

使用两点边界值,以下测试输入提供的边界覆盖率最好的是(　　)。

A. 0℃,11℃,20℃,22℃,23℃

B. 9℃,15℃,19℃,23℃,100℃

C. 10℃,16℃,19℃,22℃,23℃

D. 14℃,15℃,18℃,19℃,21℃,22℃

12. 速度控制和报告系统具有以下特征。

如果以 50km/h 或更低的速度行驶,什么都不会发生。

如果行驶速度超过 50km/h,但速度不超过 55km/h,将收到警告。

如果行驶速度超过 55km/h,但速度不超过 60km/h,将被罚款。

如果行驶速度超过 60km/h,驾照将被暂停。

基于两点边界值分析,最可能被识别(km/h)的是(　　)。

A. 0,49,50,54,59,60(km/h)　　B. 50,55,60(km/h)

C. 49,50,54,55,60,62(km/h)　　D. 50,51,55,56,60,61(km/h)

13. 对一个超速罚款系统使用决策表判定。针对规则 R1 和规则 R4 生成两个测试用例,如下所示。

规　则		R1	R4
条件	速度＞50km/h	T	F
	学校区域	T	F
动作	罚款￥250	F	F
	坐牢	T	F

以下是额外的测试用例。

规　则		DT1	DT2	DT3	DT4
输入	速度＞50km/h	55km/h	44km/h	66km/h	77km/h
	学校区域	T	T	T	F
期望结果	罚款￥250	F	F	F	T
	坐牢	T	F	T	F

以下测试用例可以获得完整决策表的完全覆盖率(包括根据规则 R1 和 R4 生成的两个测试用例)的是(　　)。

A. DT1,DT2　　B. DT2,DT3　　C. DT2,DT4　　D. DT3,DT4

14. 如果公司员工在公司工作超过一年并达到个人业绩的目标,则可获得奖金。

以下决策表旨在测试支付奖金的逻辑而设计:

		T1	T2	T3	T4	T5	T6	T7	T8
条件									
COND1	合同超过 1 年?	是	没有	是	没有	是	没有	是	没有
COND2	同意目标?	没有	没有	是	是	没有	没有	是	是
COND3	实现目标?	没有	没有	没有	没有	是	是	是	是
行动									
	奖金支付?	没有	没有	没有	没有	没有	没有	是	没有

在上述的决策表中，(　　)等用例会因为在真实的情况下不会发生而被删除。

A. T1 和 T2　　B. T3 和 T4　　C. T7 和 T8　　D. T5 和 T6

15. 以下关于决策表测试法的叙述中，不正确的是(　　)。

A. 决策表由条件桩、动作桩、条件项和动作项组成

B. 决策表依据软件规格说明建立

C. 决策表需要合并相似规则

D. n 个条件可以得到最多 n^2 个规则的决策表

16. 根据输出对输入的依赖关系设计测试用例的黑盒测试方法是(　　)。

A. 等价类划分法　　B. 因果图法

C. 边界值分析法　　D. 场景法

17. 以下关于选择黑盒测试的测试方法的叙述中，不正确的是(　　)。

A. 在任何情况下都要采用边界值分析法

B. 必要时用等价类划分法补充测试用例

C. 可以用错误推断法追加测试用例

D. 如果输入条件之间不存在组合情况，则应采用因果图法

18. 阅读下列说明，按要求回答问题。

某商店为购买不同数量商品的顾客报出不同的价格，其报价规则如下表所示。

购 买 数 量	单价(单位：元)
头 10 件(第 1 件到第 10 件)	30
第二个 10 件(第 11 件到第 20 件)	27
第三个 10 件(第 21 件到第 30 件)	25
超过 30 件	22

如买 11 件需要支付 10×30＋1×27＝327 元，买 35 件需要支付 10×30＋10×27＋10×25＋5×22＝930 元。

现为该商家开发一个软件，输入为商品数 C(1≤C≤100)，输出为应付的价钱 P。

【问题 1】

请采用等价类划分法为该软件设计测试用例(不考虑 C 为非整数的情况)。

【问题 2】

请采用边界值分析法为该软件设计测试用例(不考虑稳健性测试，即不考虑 C 不在 1～100 或者是非整数的情况)。

【问题 3】

列举除了等价类划分法和边界值分析法以外的其他常见的黑盒测试用例设计方法。

19. 阅读下列说明，按要求回答问题。

某商店的货品价格 P 都不大于 20 元(且为整数)，假设每次顾客每次付款为 20 元且每次限购一件商品，现有一个软件能在每位顾客购物后给出找零钱的最佳组合(以找给顾客货币张数最少为标准)。

假定此商店的找零货币面值只包括：10 元(N10)、5 元(N5)、1 元(N1)等 3 种。

【问题 1】

请采用等价类划分法为该软件设计测试用例(不考虑 P 为非整数的情况)并填入到下

表中(<<N1,2>>表示 2 张 1 元,若无输出或输出非法则填 N/A)。

序号	输入(商品价格 P)	输出(找零钱的组合)
1	20($P=20$)	N/A
2	18(任意 $15<P<20$)	<<N1,2>>
3		
4		
5		
6		
7		
8		
9		
10		

【问题 2】

请采用边界值分析法为该软件设计测试用例。

【问题 3】

请给出决策表法进行测试用例设计的主要步骤。