

测试用例设计白皮书

| | |
|--------------------|--|
| FILEID: | VINCETEST-20070101 |
| VERSION: | 1.0 |
| AUTHOR: | Vince |
| DATE: | 2007-01-01 |
| FILE STATE: | <input type="checkbox"/> DRAFT <input type="checkbox"/> MODIFY <input checked="" type="checkbox"/> RELEASE |
| CONTACT: | vincetest@163.com |

VINCETEST

版权所有 侵权必究

目 录

| | |
|---------------------|----|
| 1. 概述 | 3 |
| 2. 测试用例基本概念..... | 4 |
| 2.1. 测试用例的定义..... | 4 |
| 2.2. 测试用例的特征..... | 4 |
| 2.3. 测试用例组成元素..... | 4 |
| 2.4. 测试用例设计原则..... | 4 |
| 3. 测试用例设计方法..... | 5 |
| 3.1. 等价类划分方法..... | 5 |
| 3.2. 边界值分析方法..... | 11 |
| 3.3. 错误推测方法..... | 19 |
| 3.4. 因果图方法..... | 19 |
| 3.5. 判定表驱动分析方法..... | 24 |
| 3.6. 正交实验设计方法..... | 31 |
| 3.7. 功能图分析方法..... | 32 |
| 3.8. 场景设计方发..... | 33 |
| 4. 测试用例设计综合策略..... | 37 |

1. 概述

Grenford J. Myers 在《The Art of Software Testing》一书中提出：一个好的测试用例是指很可能找到迄今为止尚未发现的错误的测试，由此可见测试用例设计工作在整个测试过程中的地位，我们不能只凭借一些主观或直观的想法来设计测试用例，应该要以一些比较成熟的测试用例设计方法为指导，再加上设计人员个人的经验积累来设计测试用例，二者相结合应该是非常完美的组合。本文所介绍的测试用例设计方法对于测试设计人员将是一个很好的方法指导，当然看完本文也未必能设计出好的测试用例，有了好的方法作为指导后需要更多的实践经验加以巩固和提炼。只有将测试设计思想与丰富的实践经验相融合才能设计出高质量的测试用例，相信你能行！

本文描述的范围：测试用例基本概念、测试用例设计方法、测试用例设计综合策略。

关键词：测试用例、等价类划分、边界值分析、错误推测、因果图、判定表驱动分析、正交实验、功能图分析、场景设计

读者对象：测试设计人员、测试人员

参考文献：

1. 《计算机软件测试技术》 郑人杰
2. 《The Art of Software Testing》 Grenford J. Myers

2. 测试用例基本概念

2.1. 测试用例的定义

测试用例是为特定的目的而设计的一组测试输入、执行条件和预期的结果。测试用例是执行的最小实体。简单地说，测试用例就是设计一个场景，使软件程序在这种场景下，必须能够正常运行并且达到程序所设计的执行结果。

2.2. 测试用例的特征

1. 最有可能抓住错误的；
2. 不是重复的、多余的；
3. 一组相似测试用例中最有效的；
4. 既不是太简单，也不是太复杂。

2.3. 测试用例组成元素

1. 用例 ID；
2. 用例名称；
3. 测试目的；
4. 测试级别；
5. 参考信息；
6. 测试环境；
7. 前提条件；
8. 测试步骤；
9. 预期结果；
10. 设计人员。

2.4. 测试用例设计原则

1. 测试用例的代表性：能够代表并覆盖各种合理的和不合理的、合法的和非法的、边界的和越界的以及极限的输入数据、操作和环境设置等。
2. 测试结果的可判定性：即测试执行结果的正确性是可判定的，每一个测试用例都应有相应的期望结果。
3. 测试结果的可再现性：即对同样的测试用例，系统的执行结果应当是相同的。

文斯测试技术研究中心：<http://blog.csdn.net/vincetest>

3. 测试用例设计方法

3.1. 等价类划分方法

一. 方法简介

1. 定义

是把所有可能的输入数据,即程序的输入域划分成若干部分(子集),然后从每一个子集中选取少数具有代表性的数据作为测试用例。该方法是一种重要的,常用的黑盒测试用例设计方法。

2. 划分等价类:

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它值的测试,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。等价类划分可有两种不同的情况:有效等价类和无效等价类。

1) 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中所规定的功能和性能。

2) 无效等价类

与有效等价类的定义恰巧相反。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,也可能有多个。

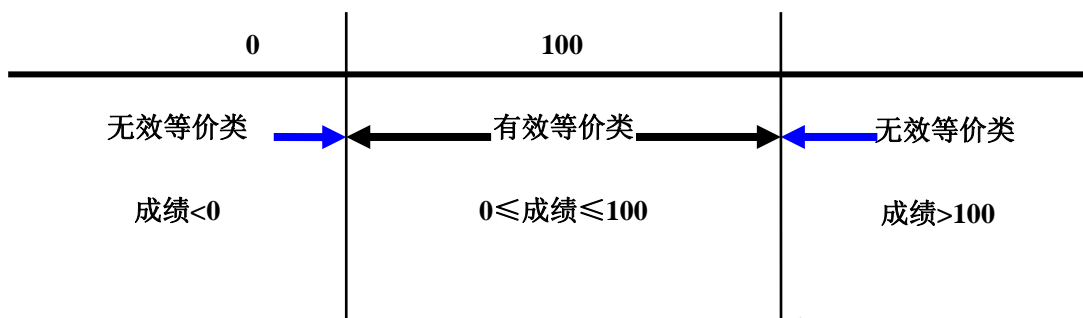
设计测试用例时,要同时考虑这两种等价类。因为软件不仅要能接收合理的数据,也要能经受意外的考验,这样的测试才能确保软件具有更高的可靠性。

3. 划分等价类的标准:

- 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:完备性;
- 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”。

4. 划分等价类的方法

- 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 $0 \sim 100$;



- 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。
- 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处理中的方式不同的情况下,则应再将该等价类进一步的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. 实战演习

1. 某程序规定:“输入三个整数 a 、 b 、 c 分别作为三边的边长构成三角形。通过程序判定所构成的三角形的类型,当此三角形为一般三角形、等腰三角形及等边三角形时,分别作计算 ...”。用等价类划分方法为该程序进行测试用例设计。(三角形问题的复杂之处在于输入与输出之间的关系比较复杂。)

分析题目中给出和隐含的对输入条件的要求:

- (1) 整数 (2) 三个数 (3) 非零数 (4) 正数
 (5) 两边之和大于第三边 (6) 等腰 (7) 等边

如果 a 、 b 、 c 满足条件 (1) ~ (4)，则输出下列四种情况之一：

- 1) 如果不满足条件 (5)，则程序输出为“非三角形”。
- 2) 如果三条边相等即满足条件 (7)，则程序输出为“等边三角形”。
- 3) 如果只有两条边相等、即满足条件 (6)，则程序输出为“等腰三角形”。
- 4) 如果三条边都不相等，则程序输出为“一般三角形”。

列出等价类表并编号

| | | 有效等价类型 | 号码 | 无效等价类 | 号码 | |
|------|---------|-------------------------|--------------|---|---|----|
| | | | | | | |
| 输入条件 | 输入三个整数 | 整数 | 1 | 一边为非整数 | $\left\{ \begin{array}{l} a \text{ 为非整数} \\ b \text{ 为非整数} \\ c \text{ 为非整数} \end{array} \right.$ | 12 |
| | | | | | | 13 |
| | | | | | | 14 |
| | | | | 两边为非整数 | $\left\{ \begin{array}{l} a, b \text{ 为非整数} \\ b, c \text{ 为非整数} \\ a, c \text{ 为非整数} \end{array} \right.$ | 15 |
| | | | | | | 16 |
| | | | | | | 17 |
| | | | | | | 18 |
| | | | | | | |
| | | 三个数 | 2 | 只给一边 | $\left\{ \begin{array}{l} \text{只给 } a \\ \text{只给 } b \\ \text{只给 } c \end{array} \right.$ | 19 |
| | | | | | | 20 |
| | | | | | | 21 |
| | | | | 只给两边 | $\left\{ \begin{array}{l} \text{只给 } ab \\ \text{只给 } b, c \\ \text{只给 } ac \end{array} \right.$ | 22 |
| | | | | | | 23 |
| | | | | | | 24 |
| | | | | | | 25 |
| | | | | | | |
| | | 非零数 | 3 | 一边为零 | $\left\{ \begin{array}{l} a \text{ 为 } 0 \\ b \text{ 为 } 0 \\ c \text{ 为 } 0 \end{array} \right.$ | 26 |
| | | | | | | 27 |
| | | | | | | 28 |
| | | | | 二边为零 | $\left\{ \begin{array}{l} a, b \text{ 为 } 0 \\ b, c \text{ 为 } 0 \\ a, c \text{ 为 } 0 \end{array} \right.$ | 29 |
| | | | | | | 30 |
| | | | | | | 31 |
| | | | | | | 32 |
| | | | | | | |
| | | 正数 | 4 | 一边 <0 | $\left\{ \begin{array}{l} a<0 \\ b<0 \\ c<0 \end{array} \right.$ | 33 |
| | | | | | | 34 |
| | | | | | | 35 |
| | | | | 二边 <0 | $\left\{ \begin{array}{l} a<0 \text{ 且 } b<0 \\ a<0 \text{ 且 } c<0 \\ b<0 \text{ 且 } c<0 \end{array} \right.$ | 36 |
| | | | | | | 37 |
| | | | | | | 38 |
| | | | | | | 39 |
| | | | | | | |
| 输出条件 | 构成一般三角形 | $a+b>c$ | 5 | $\left\{ \begin{array}{l} a+b<0 \\ a+b=0 \\ b+c<a \\ b+c=a \\ a+c<b \\ a+c=b \end{array} \right.$ | 40 | |
| | | | | | 41 | |
| | | $b+c>a$ | 6 | | 42 | |
| | | | | | 43 | |
| | | $a+c>b$ | 7 | | 44 | |
| | | | 45 | | | |
| | 构成等腰三角形 | $a=b$ $b=c$ $a=c$ | 8 9 10 | | | |
| | | 且两边之和大于第三边 | | | | |
| | | | | | | |
| | 构成等腰三角形 | $a=b=c$ | 11 | | | |

覆盖有效等价类的测试用例：

| a | b | c | 覆盖等价类号码 |
|---|---|---|------------------|
| 3 | 4 | 5 | (1) -- (7) |
| 4 | 4 | 5 | (1) -- (7), (8) |
| 4 | 5 | 5 | (1) -- (7), (9) |
| 5 | 4 | 5 | (1) -- (7), (10) |
| 4 | 4 | 4 | (1) -- (7), (11) |

覆盖无效等价类的测试用例：

| a | b | c | 覆盖等价类号码 | a | b | c | 覆盖等价类号码 |
|-----|-----|-----|---------|----|----|----|---------|
| 2.5 | 4 | 5 | 12 | 0 | 0 | 5 | 29 |
| 3 | 4.5 | 5 | 13 | 3 | 0 | 0 | 30 |
| 3 | 4 | 5.5 | 14 | 0 | 4 | 0 | 31 |
| 3.5 | 4.5 | 5 | 15 | 0 | 0 | 0 | 32 |
| 3 | 4.5 | 5.5 | 16 | -3 | 4 | 5 | 33 |
| 3.5 | 4 | 5.5 | 17 | 3 | -4 | 5 | 34 |
| 4.5 | 4.5 | 5.5 | 18 | 3 | 4 | -5 | 35 |
| 3 | | | 19 | -3 | -4 | 5 | 36 |
| | 4 | | 20 | -3 | 4 | -5 | 37 |
| | | 5 | 21 | 3 | -4 | -5 | 38 |
| 3 | 4 | | 22 | -3 | -4 | -5 | 39 |
| | 4 | 5 | 23 | 3 | 1 | 5 | 40 |
| 3 | | 5 | 24 | 3 | 2 | 5 | 41 |
| 3 | 4 | 5 | 25 | 3 | 1 | 1 | 42 |
| 0 | 4 | 5 | 26 | 3 | 2 | 1 | 43 |
| 3 | 0 | 5 | 27 | 1 | 4 | 2 | 44 |
| 3 | 4 | 0 | 28 | 3 | 4 | 1 | 45 |

2. 设有一个档案管理系统，要求用户输入以年月表示的日期。假设日期限定在 1990 年 1 月~2049 年 12 月，并规定日期由 6 位数字字符组成，前 4 位表示年，后 2 位表示月。现用等价类划分法设计测试用例，来测试程序的“日期检查功能”。

1) 划分等价类并编号,下表等价类划分的结果

| 输入等价类 | 有效等价类 | 无效等价类 |
|----------|-----------------|---------------------------------------|
| 日期的类型及长度 | ①6 位数字字符 | ②有非数字字符 ③少于 6 位数字字符 ④多于 6 位数字字符 |
| 年份范围 | ⑤在 1990~2049 之间 | ⑥小于 1990 ⑦大于 2049 |
| 月份范围 | ⑧在 01~12 之间 | ⑨等于 00 ⑩大于 12 |

- 2) 设计测试用例，以便覆盖所有有效等价类在表中列出了 3 个有效等价类，编号分别为①、⑤、⑧，设计的测试用例如下：

| 测试数据 | 期望结果 | 覆盖的有效等价类 |
|--------|------|----------|
| 200211 | 输入有效 | ①、⑤、⑧ |

- 3) 为每一个无效等价类设计一个测试用例，设计结果如下：

| 测试数据 | 期望结果 | 覆盖的无效等价类 |
|---------|------|----------|
| 95June | 无效输入 | ② |
| 20036 | 无效输入 | ③ |
| 2001006 | 无效输入 | ④ |
| 198912 | 无效输入 | ⑥ |
| 200401 | 无效输入 | ⑦ |
| 200100 | 无效输入 | ⑨ |
| 200113 | 无效输入 | ⑩ |

3. NextDate 函数包含三个变量：month、day 和 year，函数的输出为输入日期后一天的日期。例如，输入为 2006 年 3 月 7 日，则函数的输出为 2006 年 3 月 8 日。要求输入变量 month、day 和 year 均为整数值，并且满足下列条件：

① $1 \leq \text{month} \leq 12$

② $1 \leq \text{day} \leq 31$

③ $1920 \leq \text{year} \leq 2050$

1) 有效等价类为:

$M1 = \{\text{月份}: 1 \leq \text{月份} \leq 12\}$

$D1 = \{\text{日期}: 1 \leq \text{日期} \leq 31\}$

$Y1 = \{\text{年}: 1812 \leq \text{年} \leq 2012\}$

2) 若条件 ① ~ ③ 中任何一个条件失效, 则 `NextDate` 函数都会产生一个输出, 指明相应的变量超出取值范围, 比如 “month 的值不在 1-12 范围当中”。显然还存在着大量的 `year`、`month`、`day` 的无效组合, `NextDate` 函数将这些组合作统一的输出: “无效输入日期”。其无效等价类为:

$M2 = \{\text{月份}: \text{月份} < 1\}$

$M3 = \{\text{月份}: \text{月份} > 12\}$

$D2 = \{\text{日期}: \text{日期} < 1\}$

$D3 = \{\text{日期}: \text{日期} > 31\}$

$Y2 = \{\text{年}: \text{年} < 1812\}$

$Y3 = \{\text{年}: \text{年} > 2012\}$

弱一般等价类测试用例

| 月份 | 日期 | 年 | 预期输出 |
|----|----|------|-----------------|
| 6 | 15 | 1912 | 1912 年 6 月 16 日 |

强一般等价类测试用例同弱一般等价类测试用例

注: 弱——有单缺陷假设; 健壮——考虑了无效值

(一) 弱健壮等价类测试

| 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 1~12 中 |
| WR3 | 13 | 15 | 1912 | 月份不在 1~12 中 |
| WR4 | 6 | -1 | 1912 | 日期不在 1~31 中 |
| WR5 | 6 | 32 | 1912 | 日期不在 1~31 中 |
| WR6 | 6 | 15 | 1811 | 年份不在 1812~2012 中 |
| WR7 | 6 | 15 | 2013 | 年份不在 1812~2012 中 |

文斯测试技术研究中心: <http://blog.csdn.net/vincetest>

(二) 强健壮等价类测试

| 用例 ID | 月份 | 日期 | 年 | 预期输出 |
|-------|----|----|------|------------------|
| SR1 | -1 | 15 | 1912 | 月份不在 1~12 中 |
| SR2 | 6 | -1 | 1912 | 日期不在 1~31 中 |
| SR3 | 6 | 15 | 1811 | 年份不在 1812~2012 中 |
| SR4 | -1 | -1 | 1912 | 两个无效一个有效 |
| SR5 | 6 | -1 | 1811 | 两个无效一个有效 |
| SR6 | -1 | 15 | 1811 | 两个无效一个有效 |
| SR7 | -1 | -1 | 1811 | 三个无效 |

4. 佣金问题等价类测试用例，它是根据佣金函数的输出值域定义等价类，来改进测试用例集合。

输出销售额 ≤ 1000 元 佣金 10%

$1000 < \text{销售额} \leq 1800$ 佣金 $= 100 + (\text{销售额} - 1000) * 15\%$

销售额 > 1800 佣金 $= 220 + (\text{销售额} - 1800) * 20\%$

| 测试用例 | 枪机(45) | 枪托(30) | 枪管(25) | 销售额 | 佣金 |
|------|--------|--------|--------|------|-----|
| 1 | 5 | 5 | 5 | 500 | 50 |
| 2 | 15 | 15 | 15 | 1500 | 175 |
| 3 | 25 | 25 | 25 | 2500 | 360 |

根据输出域选择输入值，使落在输出域等价类内，可以结合弱健壮测试用例结合。

3.2. 边界值分析方法

一. 方法简介

1. 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。
2. 与等价划分的区别
 - 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。
3. 边界值分析方法的考虑：

长期的测试工作经验告诉我们，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。

4. 常见的边界值

- 1) 对 16-bit 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

5. 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 Print-Line 可以用来输出错误信息。

2) 等价类划分：

I. 可以考虑作出如下划分：

- 输入 (i)<0 和 (ii)>=0
- 输出 (a)>=0 和 (b) Error

II. 测试用例有两个：

- 输入 4，输出 2。对应于 (ii) 和 (a)。
- 输入 -10，输出 0 和错误提示。对应于 (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。由此得到以下测试用例：

- 输入 {最小负实数}
- 输入 {绝对值很小的负数}
- 输入 0
- 输入 {绝对值很小的正数}

文斯测试技术研究中心：<http://blog.csdn.net/vincetest>

➤ 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

| 项 | 边界值 | 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允许输入 1 个到 255 个字符，输入 1 个和 255 个字符作为有效等价类；输入 0 个和 256 个字符作为无效等价类，这几个数值都属于边界条件值。 |
| 数值 | 最小值-1/最大值+1 | 假设某软件的数据输入域要求输入 5 位的数据值，可以使用 10000 作为最小值、99999 作为最大值；然后使用刚好小于 5 位和大于 5 位的数值来作为边界条件。 |
| 空间 | 小于空余空间一点/大于满空间一点 | 例如在用 U 盘存储数据时，使用比剩余磁盘空间大一点（几 KB）的文件作为边界条件。 |

7) 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：

- a) 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。

| 项 | 范围或值 |
|-----------|----------------------------------|
| 位 (bit) | 0 或 1 |
| 字节 (byte) | 0 ~ 255 |
| 字 (word) | 0~65535 (单字) 或 0~4294967295 (双字) |
| 千 (K) | 1024 |
| 兆 (M) | 1048576 |
| 吉 (G) | 1073741824 |

- b) 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。下表中列出了一些常用字符对应的 ASCII 码值。

| 字符 | ASCII 码值 | 字符 | ASCII 码值 |
|------------|----------|---------|----------|
| 空 (null) | 0 | A | 65 |
| 空格 (space) | 32 | a | 97 |
| 斜杠 (/) | 47 | Z | 90 |
| 0 | 48 | z | 122 |
| 冒号 (:) | 58 | 单引号 (') | 96 |
| @ | 64 | | |

- c) 其它边界值检验

6. 基于边界值分析方法选择测试用例的原则

- 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的值, 以及刚刚超越这个范围边界的值作为测试输入数据。

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。

比如, 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。

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。

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。

二. 实战演习

1. 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:

| | | | | | | | |
|----------|-----|-----------------|--|-------|--|-------|---|
| (试题部分) | | | | | | | |
| 标 题 | | | | | | | |
| 1 80 | | | | | | | |
| 试题数 | | 标准答案 (1~50 题) | | | | | 2 |
| 1 | 3 4 | 9 10 | | 59 60 | | 79 80 | |
| 试题数 | | 标准答案 (51~100 题) | | | | | 2 |
| 1 | 3 4 | 9 10 | | 59 60 | | 79 80 | |
| | | | | | | | |
| (学生答卷部分) | | | | | | | |
| 学号 1 | | 学生答案 (1~50 题) | | | | | 3 |
| 1 | | 9 10 | | 59 60 | | 79 80 | |
| 学号 1 | | 学生答案 (51~100 题) | | | | | 3 |
| 1 | | 9 10 | | 59 60 | | 79 80 | |
| | | | | | | | |

- 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—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 标题只有一个字符 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 试题数含有非数字字符 |
| 标准答案记录 | 没有标准答案记录，有标题 标准答案记录多于一个 标准答案记录少一个 |
| 学生人数 | 0 个学生 1 个学生 200 个学生 201 个学生 |
| 学生答题 | 某学生只有一个回答记录，但有两个标准答案记录 该学生是文件中的第一个学生 该学生是文件中的最后一个学生（记录数出错的学生） |
| 学生答题 | 某学生有两个回答记录，但只有一个标准答案记录 该学生是文件中的第一个学生（记录数出错的学生） 该学生是文件中的最后一个学生 |
| 学生成绩 | 所有学生的成绩都相等 每个学生的成绩都不相等 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表。

| 输出条件 | 测试用例 |
|----------|--|
| 输出报告 a、b | 有个学生的学号最小（检查按序号排序是否正确） 有个学生的学号最大（检查按序号排序是否正确） 适当的学生人数，使产生的报告刚好满一页（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（有一半的 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一题 所有学生都答错了第一题 所有学生都答对了最后一题 所有学生都答错了最后一题 选择适当的试题数，是第四个报告刚好打满一页 试题数比刚才多 1，使报告打满一页后，刚好剩下一题未打 |

2. 三角形问题的边界值分析测试用例

在三角形问题描述中，除了要求边长是整数外，没有给出其它的限制条件。在此，我们将三角形每边边长的取值范围值设定为[1, 100]。

说明：如果程序规格说明中没有显式地给出边界值，则可以在设计测试用例前先设定取值的下限值和上限值。

| 测试用例 | a | b | c | 预期输出 |
|--------|-----|-----|-----|-------|
| Test1 | 60 | 60 | 1 | 等腰三角形 |
| Test2 | 60 | 60 | 2 | 等腰三角形 |
| Test3 | 60 | 60 | 60 | 等边三角形 |
| Test4 | 50 | 50 | 99 | 等腰三角形 |
| Test5 | 50 | 50 | 100 | 非三角形 |
| Test6 | 60 | 1 | 60 | 等腰三角形 |
| Test7 | 60 | 2 | 60 | 等腰三角形 |
| Test8 | 50 | 99 | 50 | 等腰三角形 |
| Test9 | 50 | 100 | 50 | 非三角形 |
| Test10 | 1 | 60 | 60 | 等腰三角形 |
| Test11 | 2 | 60 | 60 | 等腰三角形 |
| Test12 | 99 | 50 | 50 | 等腰三角形 |
| Test13 | 100 | 50 | 50 | 非三角形 |

3. NextDate 函数的边界值分析测试用例

在 NextDate 函数中, 隐含规定了变量 month 和变量 day 的取值范围为 $1 \leq \text{month} \leq 12$ 和 $1 \leq \text{day} \leq 31$, 并设定变量 year 的取值范围为 $1912 \leq \text{year} \leq 2050$ 。

| 测试用例 | month | day | year | 预期输出 |
|--------|-------|-----|------|------------------|
| Test1 | 6 | 15 | 1911 | 1911.6.16 |
| Test2 | 6 | 15 | 1912 | 1912.6.16 |
| Test3 | 6 | 15 | 1913 | 1913.6.16 |
| Test4 | 6 | 15 | 1975 | 1975.6.16 |
| Test5 | 6 | 15 | 2049 | 2049.6.16 |
| Test6 | 6 | 15 | 2050 | 2050.6.16 |
| Test7 | 6 | 15 | 2051 | 2051.6.16 |
| Test8 | 6 | -1 | 2001 | day 超出[1...31] |
| Test9 | 6 | 1 | 2001 | 2001.6.2 |
| Test10 | 6 | 2 | 2001 | 2001.6.3 |
| Test11 | 6 | 30 | 2001 | 2001.7.1 |
| Test12 | 6 | 31 | 2001 | 输入日期超界 |
| Test13 | 6 | 32 | 2001 | day 超出[1...31] |
| Test14 | -1 | 15 | 2001 | Mouth 超出[1...12] |
| Test15 | 1 | 15 | 2001 | 2001.1.16 |
| Test16 | 2 | 15 | 2001 | 2001.2.16 |
| Test17 | 11 | 15 | 2001 | 2001.11.16 |
| Test18 | 12 | 15 | 2001 | 2001.12.16 |
| Test19 | 13 | 15 | 2001 | Mouth 超出[1...12] |

3.3. 错误推测方法

一. 方法简介

1. 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。

2. 错误推测方法的基本思想：

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：

- I. 程序是否把空格作为回答

- II. 在回答记录中混有标准答案记录

- III. 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

- IV. 有两个学生的学号相同

- V. 试题数是负数。

- 3) 再如，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：

- I. 输入的线性表为空表；

- II. 表中只含有一个元素；

- III. 输入表中所有元素已排好序；

- IV. 输入表已按逆序排好；

- V. 输入表中部分或全部元素相同。

二. 实战演习

暂无

3.4. 因果图方法

一. 方法简介

1. 定义：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。

2. 因果图法产生的背景：

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

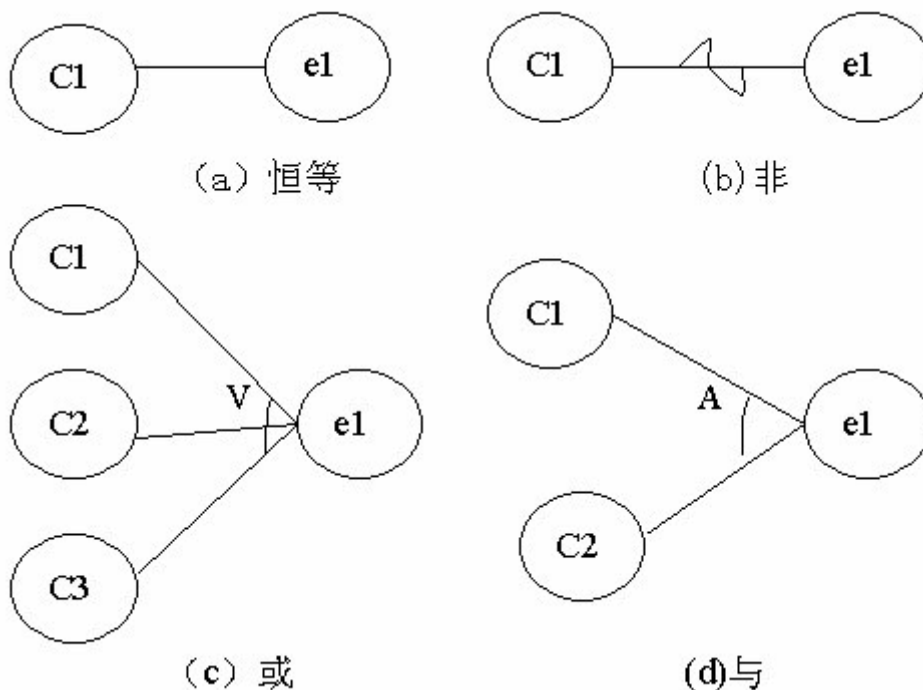
文斯测试技术研究中心：<http://blog.csdn.net/vincetest>

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）。

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

4. 因果图概念

1) 关系

①恒等：若 ci 是 1，则 ei 也是 1；否则 ei 为 0。

②非：若 ci 是 1，则 ei 是 0；否则 ei 是 1。

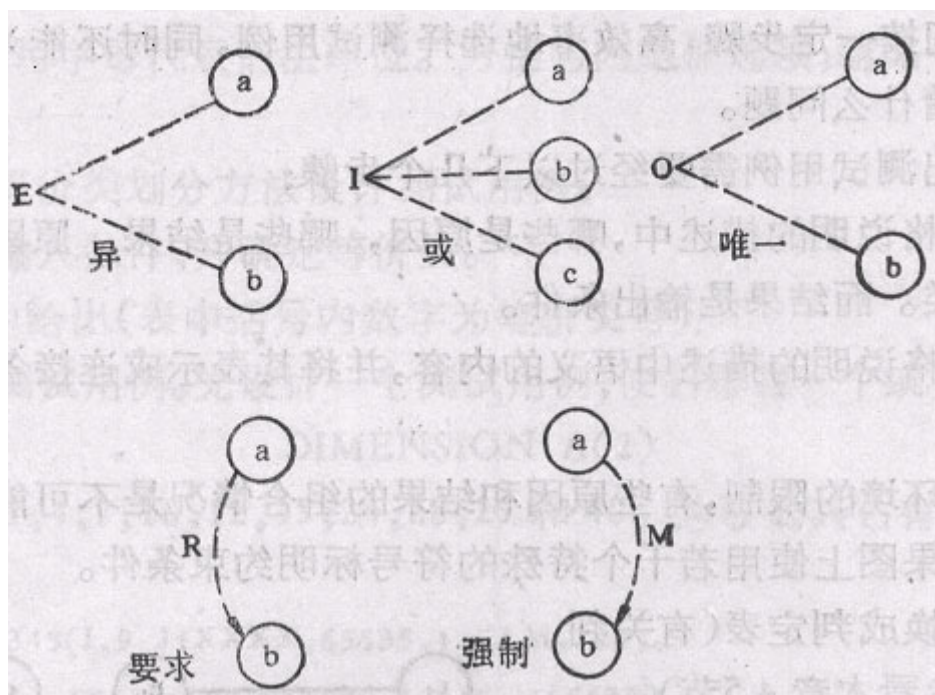
③或：若 c1 或 c2 或 c3 是 1，则 ei 是 1；否则 ei 为 0。“或”可有任意个输入。

④与：若 c1 和 c2 都是 1，则 ei 为 1；否则 ei 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。



A. 输入条件的约束有以下 4 类:

- ① 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。
- ② 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。
- ③ O 约束 (唯一): a 和 b 必须有一个, 且仅有 1 个为 1。
- ④ 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。

B. 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。

5. 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。

二. 实战演习

1. 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：

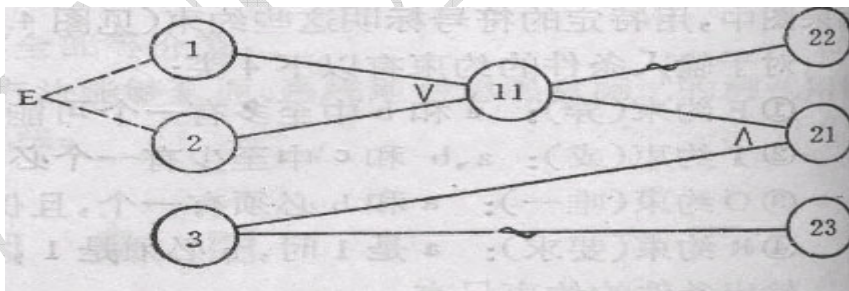
- 1——第一列字符是 A；
- 2——第一列字符是 B；
- 3——第二列字符是一数字。

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。

| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|--------|----|---|---|----|----|----|----|----|----|
| 条件（原因） | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| | 2 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| | 3 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| | 11 | | | 1 | 1 | 1 | 1 | 0 | 0 |
| 动作（结果） | 22 | | | 0 | 0 | 0 | 0 | 1 | 1 |
| | 21 | | | 1 | 0 | 1 | 0 | 0 | 0 |
| | 23 | | | 0 | 1 | 0 | 1 | 0 | 1 |
| 测试用例 | | | | A3 | AM | B5 | BN | C2 | DY |
| | | | | A8 | A7 | B4 | B1 | X6 | P; |

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，这是我們所需要的数据。

2. 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：

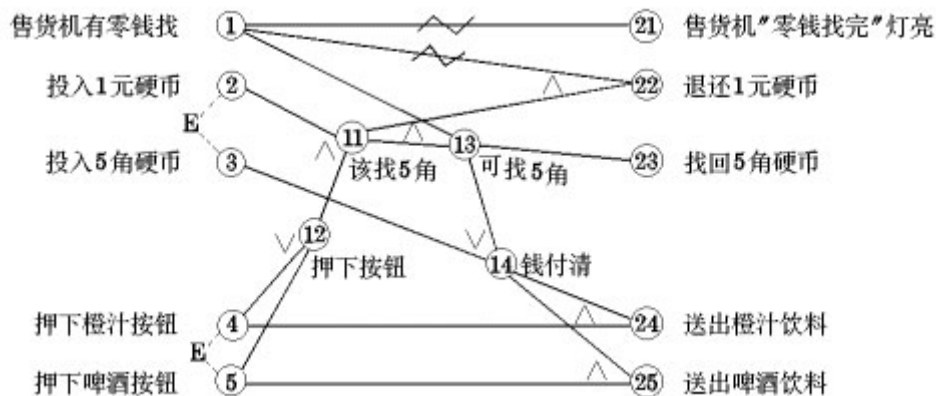
1. 售货机有零钱找
2. 投入 1 元硬币
3. 投入 5 角硬币
4. 押下橙汁按钮
5. 押下啤酒按钮

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:

| 序号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
|------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 条件 | ① | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | ② | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | ③ | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| | ④ | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | ⑤ | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 中间结果 | ⑪ | | | | | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | 1 | 1 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | |
| | ⑫ | | | | | 1 | 1 | 0 | | 1 | 1 | 0 | 1 | 1 | 0 | | | | | | 1 | 1 | 0 | | 1 | 1 | 0 | | 1 | 1 | 0 | |
| | ⑬ | | | | | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | |
| | ⑭ | | | | | 1 | 1 | 0 | | 1 | 1 | 1 | 0 | 0 | 0 | | | | | | 0 | 0 | 0 | | 1 | 1 | 1 | | 0 | 0 | 0 | |
| 结果 | ㉑ | | | | | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | 1 | 1 | 1 | | 1 | 1 | 1 | | 1 | 1 | 1 | |
| | ㉒ | | | | | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | 1 | 1 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | |
| | ㉓ | | | | | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | |
| | ㉔ | | | | | 1 | 0 | 0 | | 1 | 0 | 0 | 0 | 0 | 0 | | | | | | 0 | 0 | 0 | | 1 | 0 | 0 | | 0 | 0 | 0 | |
| | ㉕ | | | | | 0 | 1 | 0 | | 0 | 1 | 0 | 0 | 0 | 0 | | | | | | 0 | 0 | 0 | | 0 | 1 | 0 | | 0 | 0 | 0 | |
| 测试用例 | | | | | | Y | Y | Y | | Y | Y | Y | | Y | Y | | | | | | Y | Y | Y | | Y | Y | Y | | Y | Y | | |

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。

3.5. 判定表驱动分析方法

一. 方法简介

1. 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。
2. 判定表的优点

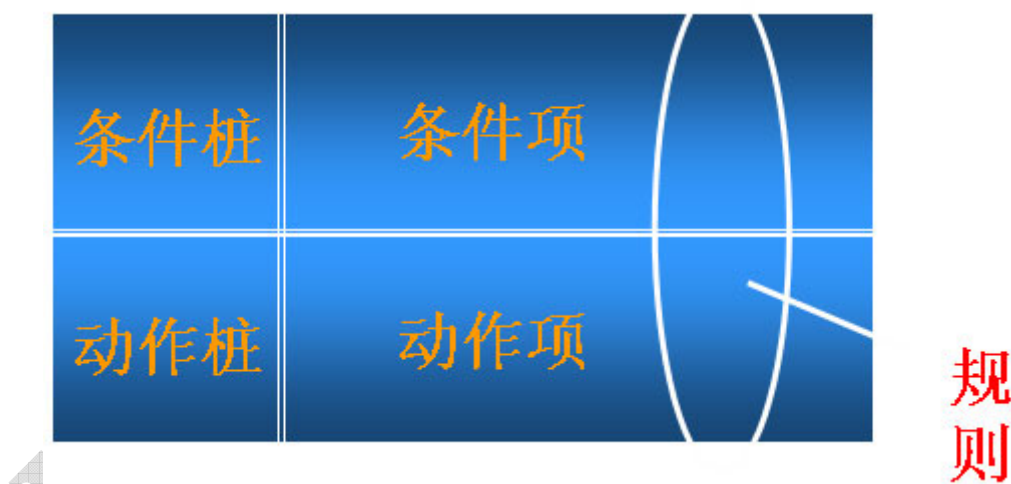
能够将复杂的问题按照各种可能的情况全部列举出来，简明并避免遗漏。因此，利用判定表能够设计出完整的测试用例集合。

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。判定表很适合于处理这类问题。

3. “阅读指南”判定表

| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|----|-------|---|---|---|---|---|---|---|---|
| 问题 | 觉得疲倦? | Y | Y | Y | Y | N | N | N | N |
| | 感兴趣吗? | Y | Y | N | N | Y | Y | N | N |
| | 糊涂吗? | Y | N | Y | N | Y | N | Y | N |
| 建议 | 重读 | | | | | √ | | | |
| | 继续 | | | | | | √ | | |
| | 跳下一章 | | | | | | | √ | √ |
| | 休息 | √ | √ | √ | √ | | | | |

4. 判定表通常由四个部分组成如下图所示。



- 1) 条件桩 (Condition Stub): 列出了问题得所有条件。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。

5. 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

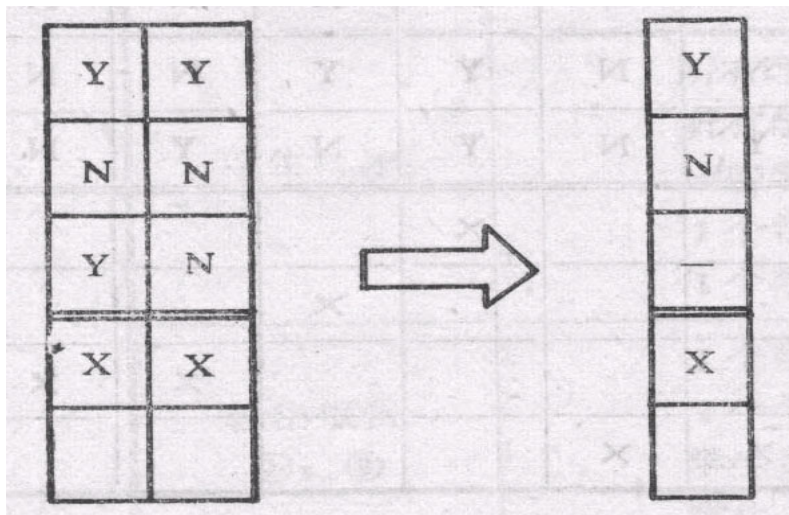
文斯测试技术研究中心: <http://blog.csdn.net/vincetest>

就有多少条规则,既条件项和动作项有多少列。

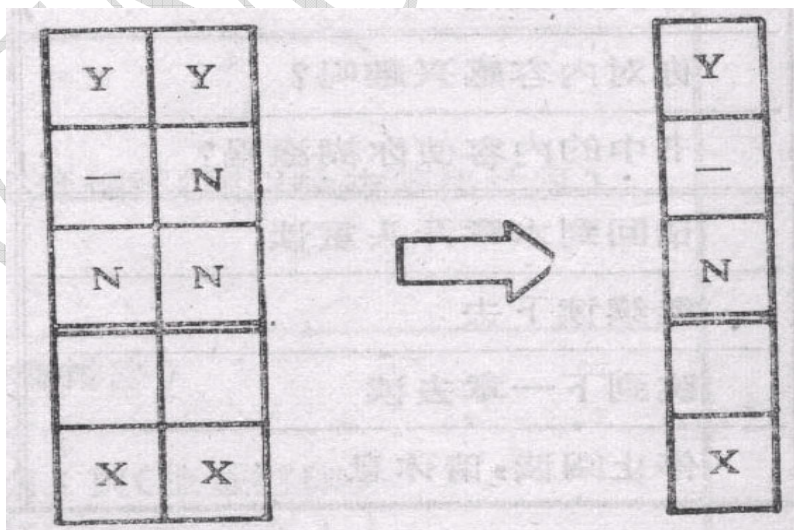
- 2) 化简: 就是规则合并有两条或多条规则具有相同的动作, 并且其条件项之间存在着极为相似的关系。

6. 规则及规则合并举例

- 1) 如下图左端, 两规则动作项一样, 条件项类似, 在 1、2 条件项分别取 Y、N 时, 无论条件 3 取何值, 都执行同一操作。即要执行的动作与条件 3 无关。于是可合并。“—”表示与取值无关。



- 2) 与上类似, 下图中, 无关条件项“—”可包含其他条件项取值, 具有相同动作的规则可合并。



3) 化简后的读书指南判定表

| | | 1 | 2 | 3 | 4 |
|--------|------------|---|---|---|---|
| 问 题 | 你觉得疲倦吗? | - | - | Y | N |
| | 你对内容感兴趣吗? | Y | Y | N | N |
| | 书中内容使你糊涂吗? | Y | N | - | - |
| 建 议 | 请回到本章开头重读 | x | | | |
| | 继续读下去 | | X | | |
| | 跳到下一章去读 | | | | x |
| | 停止阅读, 请休息 | | | x | |

7. 判定表的建立步骤: (根据软件规格说明)

- 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), 故有 $2n$ 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)。

二. 实战演习

1. 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。请建立判定表。

解答:

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 $2*2*2=8$ 种规则。

②列出所有的条件桩和动作桩:

| | |
|--------|--------------|
| 条 件 | 功率大于 50 马力吗? |
| | 维修记录不全吗? |
| | 运行超过 10 年吗? |
| 动 作 | 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: Y N Y N Y N

文斯测试技术研究中心: <http://blog.csdn.net/vincetest>

YN, 第二行是: YYNNYYNN 等等。

④填入动作桩和动作顶。这样便得到形如图的初始判定表。

| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|--------|--------------|---|---|---|---|---|---|---|---|
| 条 件 | 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 | 进行优先处理 | x | x | X | | X | | X | |
| | 作其他处理 | | | | X | | x | | x |

初始判定表

⑤化简。合并相似规则后得到图。

| | | 1 | 2 | 3 | 4 | 5 |
|--------|--------------|---|---|---|---|---|
| 条 件 | 功率大于 50 马力吗? | Y | Y | Y | N | N |
| | 维修记录不全吗? | Y | N | N | - | - |
| | 运行超过 10 年吗? | - | Y | N | Y | N |
| 动 作 | 进行优先处理 | x | x | | X | |
| | 作其他处理 | | | x | | x |

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: $1 \leq \text{日期} \leq 27$ }

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

| 输入变量间存在大量逻辑关系的NextData决策表 | | | | | | | | | | | | | |
|--|--|----------------------------------|----------------------------------|--|----------------------------------|--|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|
| | 1~3 | 4 | 5 | 6~9 | 10 | 11~14 | 15 | 16 | 17 | 18 | 19 | 20 | 21~22 |
| C ₁ 月份在 C ₂ 日期在 | M ₁ D ₁ D ₂ D ₃ | M ₁ D ₄ | M ₁ D ₅ | M ₂ D ₁ D ₂ D ₃ D ₄ | M ₂ D ₅ | M ₃ D ₁ D ₂ D ₃ D ₄ | M ₃ D ₅ | M ₄ D ₁ | M ₄ D ₂ | M ₄ D ₂ | M ₄ D ₃ | M ₄ D ₃ | M ₄ D ₅ |
| C ₃ 年在 | — | — | — | — | — | — | — | — | Y ₁ | Y ₂ | Y ₁ | Y ₂ | — |
| 行为 | | | | | | | | | | | | | |
| a ₁ : 不可能 | | | × | | | | | | | | | × | × |
| a ₂ : 日期+1 | × | | | × | | × | | × | × | | | | |
| a ₃ : 日期复位 | | × | | | × | | × | | | × | × | | |
| a ₄ : 月份+1 | | × | | | × | | | | | × | × | | |
| a ₅ : 月份复位 | | | | | | | × | | | | | | |
| a ₆ : 年+1 | | | | | | | × | | | | | | |

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： $1 \leq \text{month} \leq 12$ 和 $1 \leq \text{day} \leq 31$ ），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：

- 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: { $1 \leq \text{day} \leq 26$ }

D2: {day=27}

D3: {day=28}

D4: {day=29}

D5: {day=30}

D6: {day=31}

- 3) 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：

I. 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。

II. 在任一个条件都不满足时，要执行操作 2。

III. 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：

| | 规则 1 | 规则 2 | 规则 3 | 规则 4 |
|------|------|------|------|------|
| 条件 1 | Y | Y | N | N |
| 条件 2 | Y | - | N | - |
| 条件 3 | N | Y | N | - |
| 条件 4 | N | Y | N | Y |
| 操作 1 | x | x | | |
| 操作 2 | | | x | |
| 操作 3 | | | | x |

根据规则说明得到的判定表

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)。

| | 规则 5 | 规则 6 | 规则 7 | 规则 8 |
|------|------|------|------|------|
| 条件 1 | - | N | Y | Y |
| 条件 2 | - | Y | Y | N |
| 条件 3 | Y | N | N | N |
| 条件 4 | N | N | Y | - |
| 默许操作 | x | x | x | x |

默许的规则

2) 判定表的优点和缺点

- I. 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。
- II. 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:

- ①规格说明以判定表形式给出,或很容易转换成判定表。
- ②条件的排列顺序不会也不影响执行哪些操作。
- ③规则的排列顺序不会也不影响执行哪些操作。
- ④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。
- ⑤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。

3.6. 正交实验设计方法

一. 方法简介

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。往往因果关系非常庞大,以至于据此因果图而得到的测试用例数目多的惊人,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,可利用正交实验设计方法进行测试用例的设计。

正交实验设计方法:依据 Galois 理论,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。

二. 实战演习

暂无

3.7. 功能图分析方法

一. 方法简介

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.)

1. 功能图

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。

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.

二. 实战演习

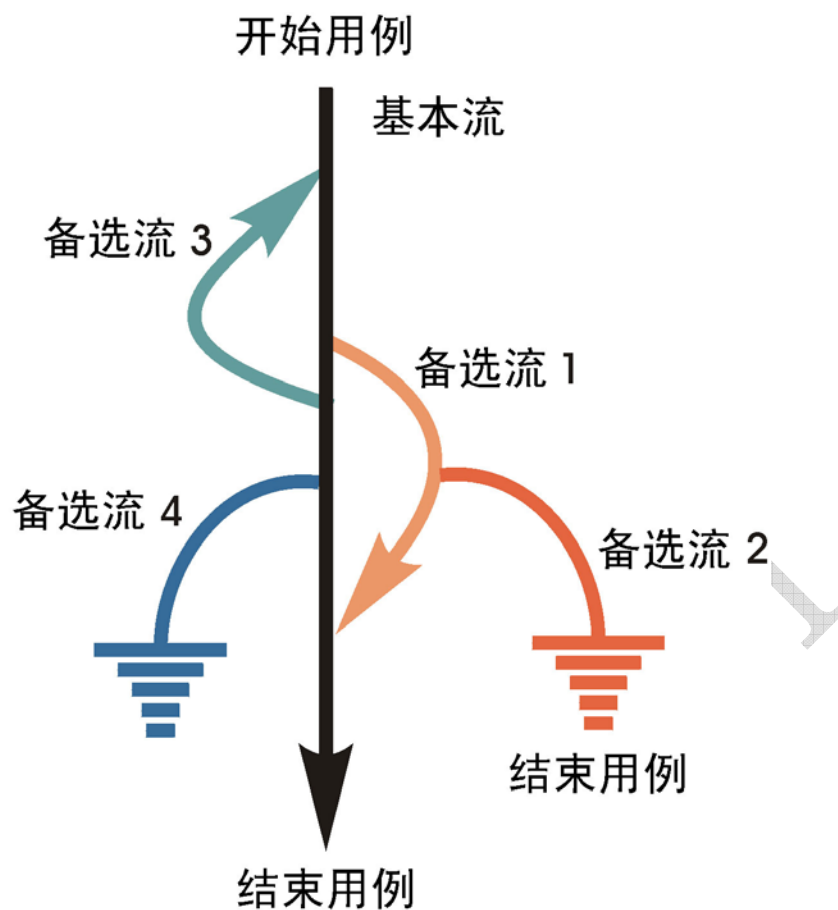
暂无

3.8. 场景设计方发

一. 方法简介

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。

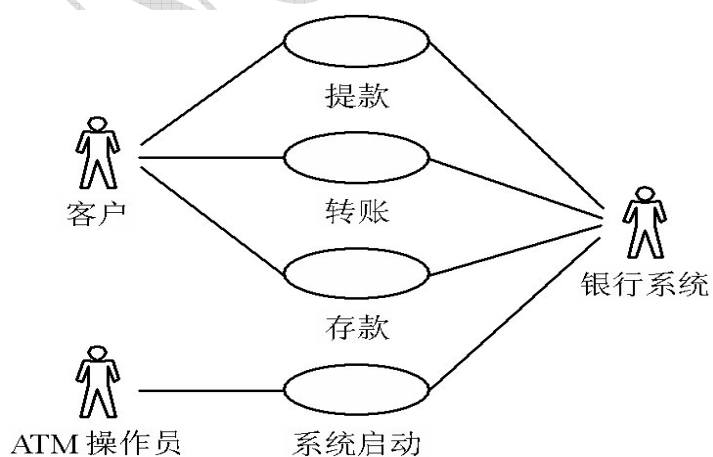
基本流和备选流:如下图所示,图中经过用例的每条路径都用基本流和备选流来表示,直黑线表示基本流,是经过用例的最简单的路径.备选流用不同的色彩表示,一个备选流可能从基本流开始,在某个特定条件下执行,然后重新加入基本流中(如备选流 1 和 3);也可能起源于另一个备选流(如备选流 2),或者终止用例而不再重新加入到某个流(如备选流 2 和 4)。



二. 实战演习

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。

表 3-8 场景设计

| | | |
|-----------------------|-----|-------|
| 场景 1——成功提款 | 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 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 |

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中）以及预期结果。

表 3-9 测试用例表

| TC（测试用例）ID 号 | 场景/条件 | PIN | 账号 | 输入（或选择）的金额 | 账面金额 | ATM 内的金额 | 预期结果 |
|--------------|-------------------------|-----|----|------------|------|----------|-----------------------|
| CW1 | 场景 1：成功提款 | V | V | V | V | V | 成功提款 |
| CW2 | 场景 2：ATM 内没有现金 | V | V | V | V | I | 提款选项不可用，用例结束 |
| CW3 | 场景 3：ATM 内现金不足 | V | V | V | V | I | 警告消息，返回基本流步骤 6，输入金额 |
| CW4 | 场景 4：PIN 有误（还有不止一次输入机会） | I | V | n/a | V | V | 警告消息，返回基本流步骤 4，输入 PIN |
| CW5 | 场景 4：PIN 有误（还有一次输入机会） | I | V | n/a | V | V | 警告消息，返回基本流步骤 4，输入 PIN |
| CW6 | 场景 4：PIN 有误（不再有输入机会） | I | V | n/a | V | V | 警告消息，卡予保留，用例结束 |

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，如表 3-10 所示。

表 3-10 测试用例表

| TC（测试用例）ID 号 | 场景/条件 | PIN | 账号 | 输入（或选择）的金额（元） | 账面金额（元） | ATM 内的金额（元） | 预期结果 |
|--------------|-------------------------|------|---------|---------------|---------|-------------|-----------------------|
| CW1 | 场景 1：成功取款 | 4987 | 809-498 | 50.00 | 500.00 | 2 000 | 成功取款。账户余额被更新为 450.00 |
| CW2 | 场景 2：ATM 内没有现金 | 4987 | 809-498 | 100.00 | 500.00 | 0.00 | 取款选项不可用，用例结束 |
| CW3 | 场景 3：ATM 内现金不足 | 4987 | 809-498 | 100.00 | 500.00 | 70.00 | 警告消息，返回基本流步骤 6，输入金额 |
| CW4 | 场景 4：PIN 有误（还有不止一次输入机会） | 4978 | 809-498 | n/a | 500.00 | 2 000 | 警告消息，返回基本流步骤 4，输入 PIN |
| CW5 | 场景 4：PIN 有误（还有一次输入机会） | 4978 | 809-498 | n/a | 500.00 | 2 000 | 警告消息，返回基本流步骤 4，输入 PIN |
| CW6 | 场景 4：PIN 有误（不再有输入机会） | 4978 | 809-498 | n/a | 500.00 | 2 000 | 警告消息，卡予保留，用例结束 |

4. 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。