



# 黑盒测试

## Black-box testing

# 提纲

- ◆ 黑盒测试概述
- ◆ 黑盒测试技术
- ◆ 黑盒测试举例

# 提纲

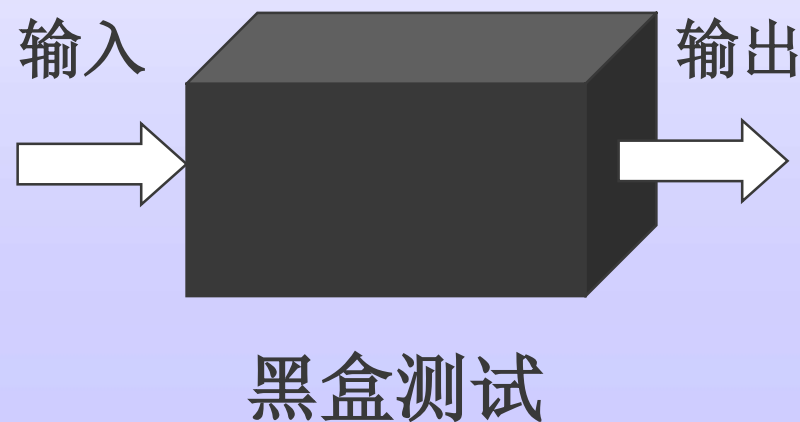
- ◆ 黑盒测试概述
- ◆ 黑盒测试技术
- ◆ 黑盒测试举例

# 1. 黑盒测试概述

- ◆ 黑盒测试概念
- ◆ 通过测试和失败测试
- ◆ 黑盒测试的应用范围
- ◆ 黑盒测试的过程
- ◆ 黑盒测试可能发现的错误
- ◆ 黑盒测试的优缺点
- ◆ 黑盒测试分类

# 1.1 黑盒测试概念

- ◆ **黑盒测试**——是基于需求说明书的软件测试，在这种测试下，不需要了解软件的内部结构，以及软件代码的具体实现。
- ◆ 在不了解软件结构和内部代码的情况下测试软件（软件处于运行状态）称为动态黑盒测试
- ◆ 动态黑盒测试也叫**行为测试**，因为测试的是软件在使用过程中的实际行为




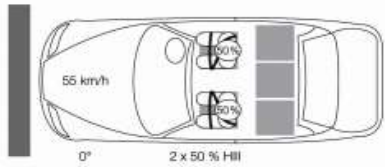
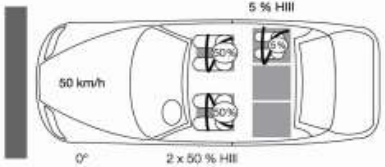
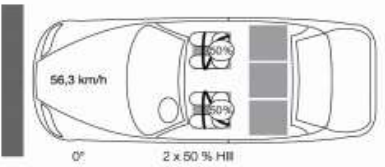
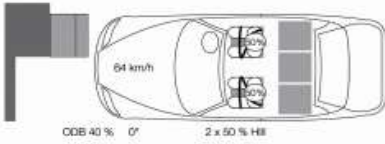
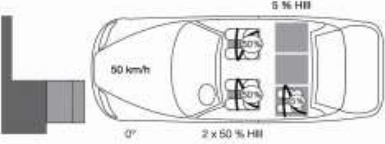
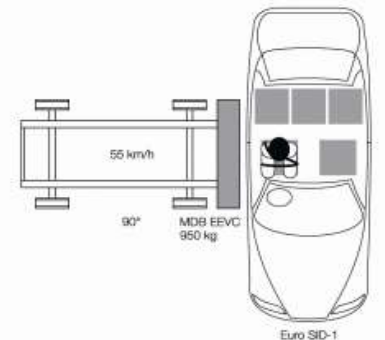
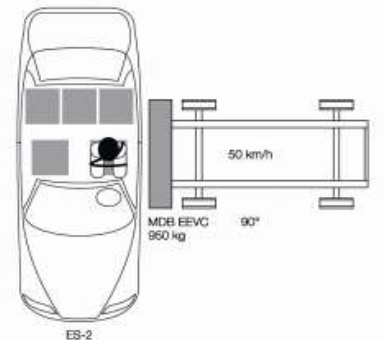
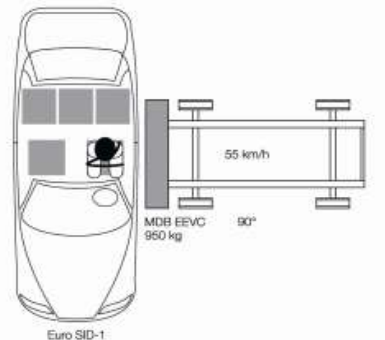


# 1.1黑盒测试概念(cont.)

- ◆ 有效的黑盒测试需要关于软件行为的定义，即软件需求规格说明书，这是动态黑盒测试的基础
- ◆ 如果被测软件没有需求规格说明书，则只能将软件产品本身当作需求规格说明书来看待，然后对软件上的所有功能进行测试；
- ◆ 在这种情况下，由于测试比较盲目，将造成遗漏测试数据或分支，但仍然可以找出一些软件缺陷。

# 1.2 通过测试和失败测试



JNCAP 	C-NCAP 	KNCAP 
		
		
		

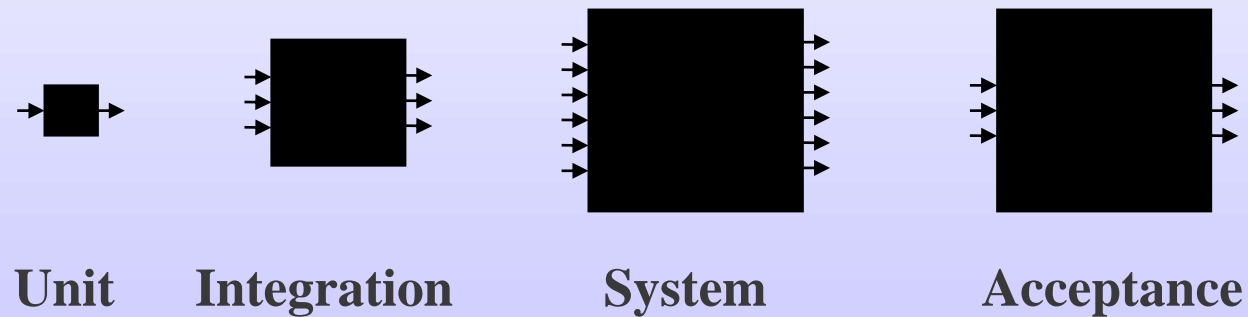
## 1.2 通过测试和失败测试

- ◆ **通过测试**：对软件的**正常**功能进行测试，保证软件能够完成它声称的功能。一般选择**有效**测试用例进行测试
- ◆ **失败测试**：又称为**迫使出错**测试，采取各种措施和手段使软件出错。目的是测试软件在各种极端情况下运行的状况。一般选择**无效**测试用例进行测试。
- ◆ 在设计和执行测试用例时，应**先进行通过测试**，在确认软件基本功能正常的情况下再进行失败测试。



## 1.3 黑盒测试的应用范围

- ◆ 黑盒测试基于软件的基本需求，主要是针对软件功能、性能等的测试。
- ◆ 黑盒测试可以应用到系统开发的各个阶段，包括单元测试、集成测试、系统测试以及验收测试



黑盒测试的应用

# 1.4 黑盒测试的过程

## ◆ 黑盒测试的基本过程：

1. 分析需求规格说明书；
2. 基于需求规格说明书选择有效输入确认软件能够正确处理；选择无效输入验证软件是否能够处理它们；
3. 确定预期输出结果；
4. 用选择的输入构建测试用例；
5. 进行测试；
6. 比较实际输出结果和预期输出结果；
7. 确认被测试软件的正确性。

## 1.5 黑盒测试可能发现的错误

- ◆ 黑盒测试可以发现的错误：
  - 不正确的或遗失的功能
  - 接口错误
  - 数据结构或外部数据库访问错误
  - 性能缺陷
  - 初始化和终止错误

## 1.6 黑盒测试的优缺点

### ◆ 优点

正规的黑盒测试可以指导测试人员选择高效和有效发现软件缺陷的测试子集。

### ◆ 缺点

黑盒测试不能确认对软件测试的程度。无论多么聪明和勤奋的测试人员，都不可能通过黑盒测试测试程序的所有路径。

## 1.7 黑盒测试分类

- ◆ 按测试目的分：功能性测试及非功能性测试
- ◆ 功能性测试用于验证被测试软件的功能正确性、一致性
- ◆ 非功能性测试用于验证软件的非功能属性，比如性能测试等
- ◆ 由于非功能性测试的存在，黑盒测试较白盒测试有更广泛的应用

# 提纲

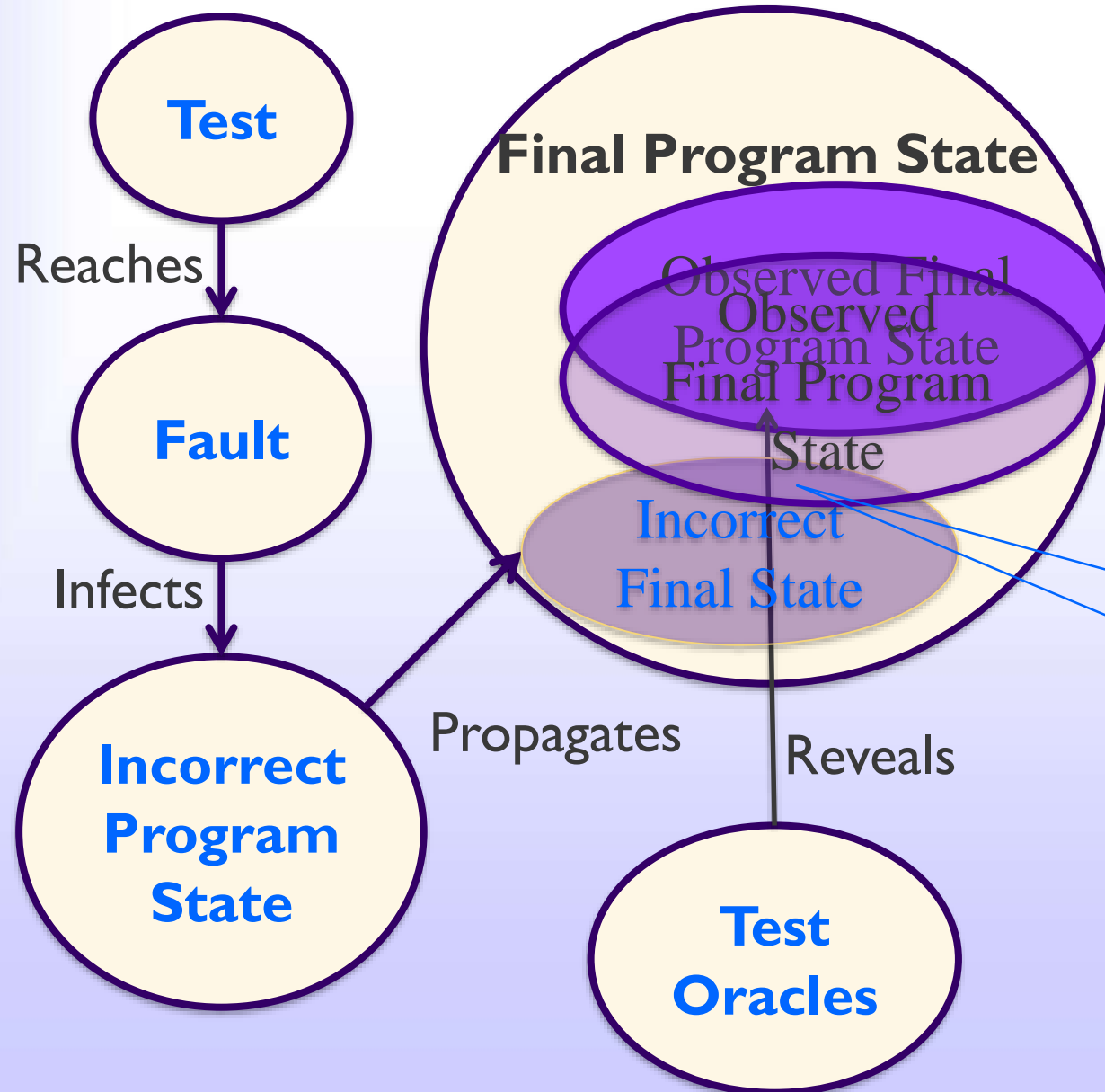
- ◆ 黑盒测试概述
- ◆ 黑盒测试技术
- ◆ 黑盒测试举例

# Fault & Failure Model (RIPR)

## Four conditions necessary for a failure to be observed

1. **Reachability** : The location or locations in the program that contain the fault must be reached
2. **Infection** : The state of the program must be incorrect
3. **Propagation** : The infected state must cause some output or final state of the program to be incorrect
4. **Reveal** : The tester must observe part of the incorrect portion of the program state

# RIPR Model



- Reachability
- Infection
- Propagation
- Revealability

目标: Getting the overlap as big as possible and use the cost as small as possible.

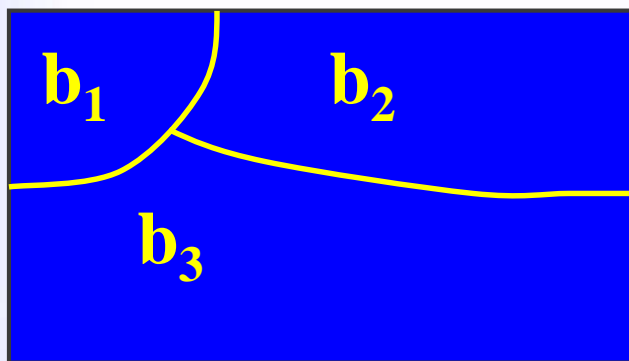


## 2 黑盒测试技术

- ◆ 等价类划分 (Equivalence partitioning)
- ◆ 边界值分析 (Boundary value analysis)
- ◆ 决策表 (Decision table)
- ◆ 组合测试
- ◆ 用例/场景测试
- ◆ 状态测试 ( State transition tables )
- ◆ 基于经验的测试

## 2.1 等价类划分

- ◆ 把所有可能的输入数据，即程序的输入域D划分成若干部分，然后从每部分中选取少数有代表性的数据做为测试用例。

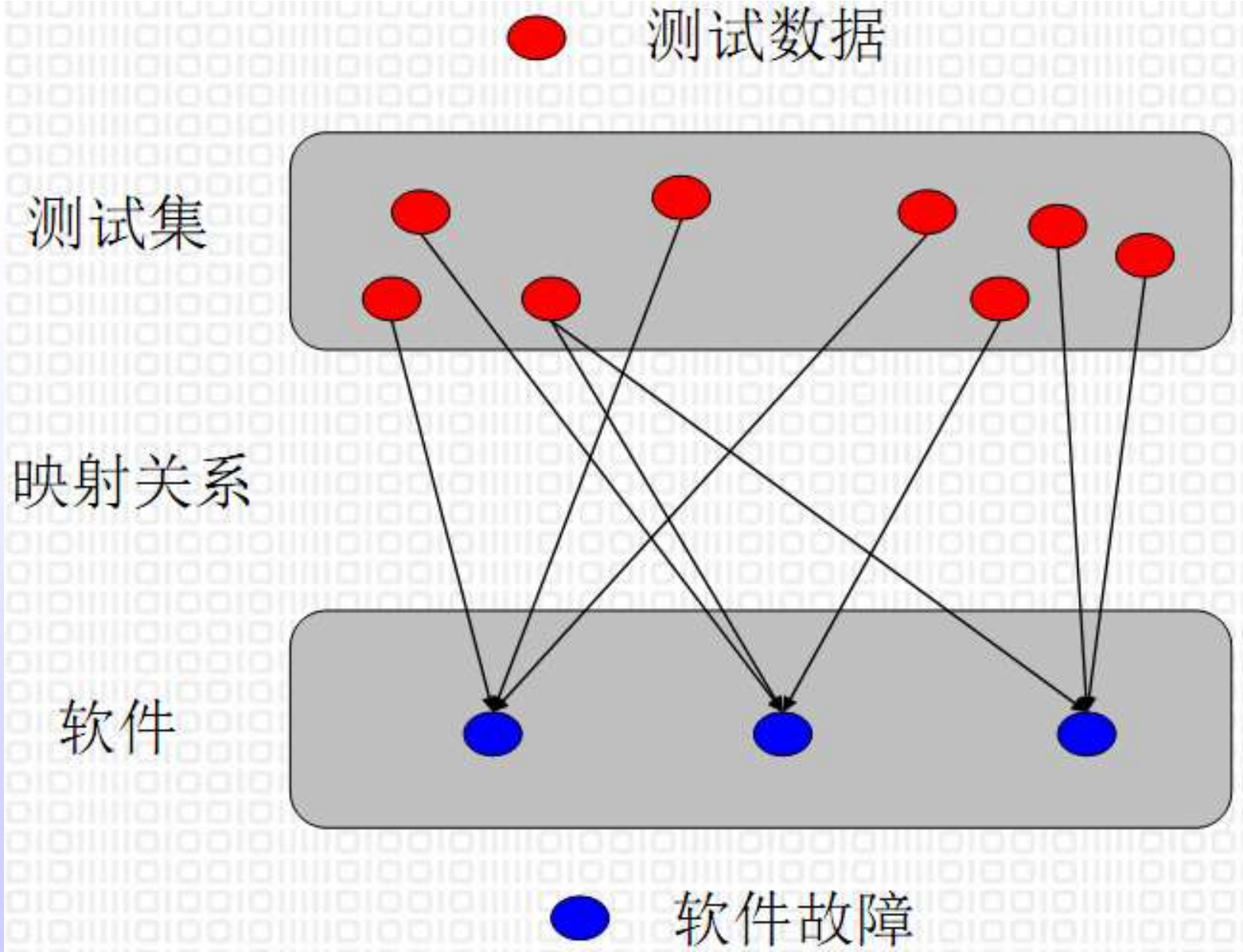


$$b_i \cap b_j = \Phi, \forall i \neq j$$

$$\cup b_i = D$$

- ◆ 设计测试用例要经历分类（列出等价类表）和选取测试用例（抽象）两步。
  - 分类：将输入域按照具有相同特性或者类似功能进行分类
  - 抽象：在各子类中抽象出相同特性并用实例来表征这个特性

# 等价类测试基本思想



## 2.1.1等价类

- ◆ 等价类划分将产生等价类
- ◆ 等价类中的各个数据对于揭露程序中的错误都是等效的。意味着：
  - 如果等价类中的一个测试用例发现缺陷，那么其它用例都将发现同样的软件缺陷
  - 如果等价类中的一个测试用例没有发现缺陷，那么其它用例也不会发现软件缺陷
- ◆ 等价类可以在保证软件合理测试的情况下大幅度减少测试用例

## 2.1.1等价类

- ◆ 等价类的划分有两种不同的情况：

① **有效等价类**：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。

目的：检查是否实现了规定的功能和性能

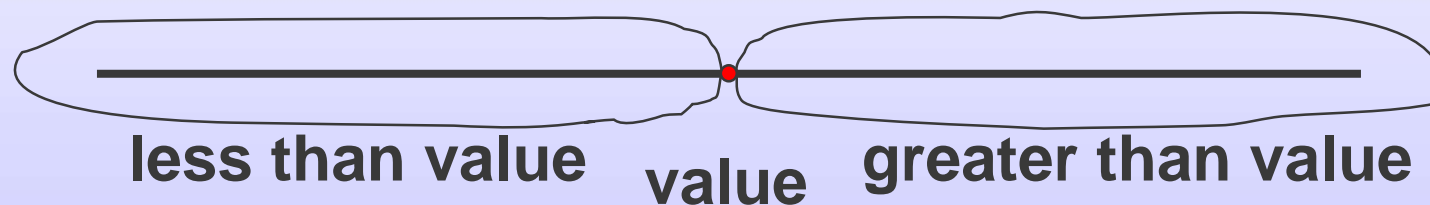
② **无效等价类**：对于程序的规格说明来说，是不合理的，无意义的输入数据构成的集合。

目的：测试系统的容错性，检查在错误输入时，是否有合理响应

- ◆ 在设计测试用例时，要同时考虑有效等价类和无效等价类的设计。
- ◆ 可以按**区间、数值、数值集合、限制条件或规则等**划分等价类

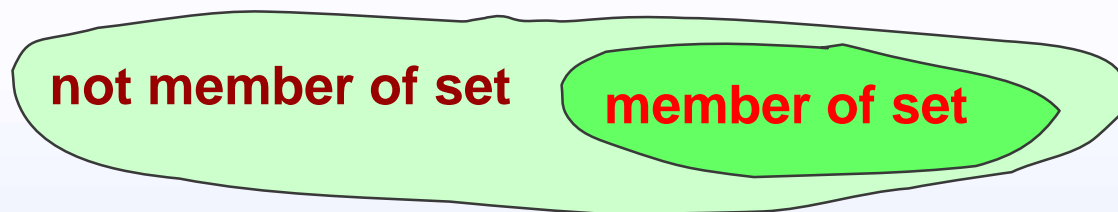
## 2.1.2 划分等价类的方法(1)

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。

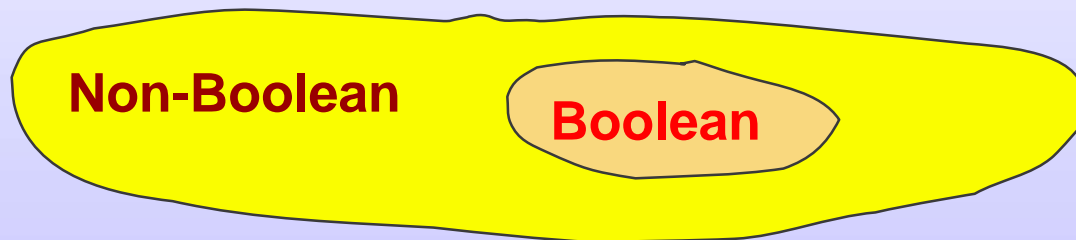


## 2.1.2 划分等价类的方法(2)

- ◆ 在输入条件规定了输入值的集合或者规定了“必须如何”的条件的情況下，可以确立一个有效等价类和一个无效等价类。



- ◆ 在输入条件是一个布尔量的情況下，可确定一个有效等价类和一个无效等价类



## 2.1.2 划分等价类的方法(3)

- ◆ 在程序要对不同情况的输入值（假设有n种情况）分别处理时，可确立n个有效等价类和一个无效等价类。

个人月收入(x)	税率
<b><math>x &lt; 3500</math></b>	<b>0%</b>
<b><math>3500 \leq x &lt; 5000</math></b>	<b>5%</b>
<b><math>5000 \leq x &lt; 10000</math></b>	<b>10%</b>
<b>...</b>	
<b><math>&gt; 80000</math></b>	<b>45%</b>

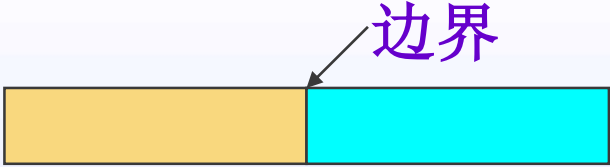


## 2.2 边界值分析

- ◆ 边界测试
- ◆ 次边界测试
- ◆ 默认值、零值和空白输入
- ◆ 无效数据

# 2.2.1 边界测试

◆ 等价类区间之间的边缘即为边界，边界测试是指对等价类区间边界附近的数据进行测试



◆ 在实际的软件中，通常会在需求规格说明书中规定操作界限的边缘条件。

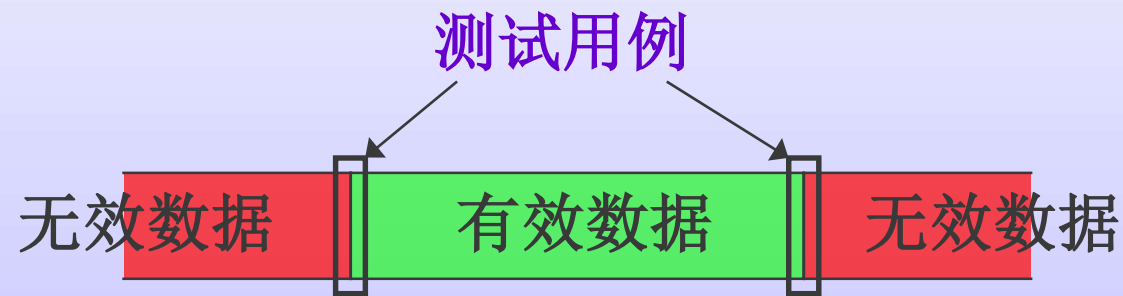
## Excel 规范与限制

### 工作表和工作簿规范与限制

功能	最大限制
打开的工作簿个数	受可用内存和系统资源的限制
工作表上的总行数 和列数	1,048,576 行乘以 16,384 列
列宽	255 个字符
行高	409 磅
分页符	水平方向和垂直方向各 1,026 个
单元格可以包含的 字符总数	32,767 个字符
页眉或页脚中的字 符数	255

## 2.2.1.1 测试边界条件

- ◆ 根据边界条件划分等价类区间：
  1. 第一个区间包含有效数据，选择的测试用例为边界内部最后的一两个数据点
  2. 第二个区间包含无效数据，选择测试边界之外的一两个数据点



## 2.2.1.2 测试边界条件举例

1. 如果允许输入数据范围为1-99，就输入0， 1， 99， 100等
2. 如果文本输入域允许输入1~255个字符，那么就输入0， 1， 254， 255， 256个字符做测试
3. 如果存贮文件的最大大小为2G，那么就存贮0G， 1.9G， 2G， 2.1G的文件

## 例：排序问题中的边界条件

（对含有10个整数的数组进行升序排序）

- ① 序列为空  $A=\{\}$
- ② 序列中只有一个数据  $A=\{1\}$
- ③ 序列中有最多个数的数据  $A=\{0,9,8,3,1,7,2,6,4,5\}$
- ④ 序列中数据个数超出最多个数1个  $A=\{0,9,8,3,1,7,2,6,4,5,10\}$
- ⑤ 已经有序的序列  $A=\{0,2,3,7,9\}$
- ⑥ 刚好逆序的序列  $A=\{9,7,5,4,2,1\}$
- ⑦ 序列中所有数据相等  $A=\{6,6,6,6,6\}$
- ⑧ .....

## 2.2.2 次边界测试

- ◆ 软件内部设立的边界条件
- ◆ 需要大体了解软件的工作方式
- ◆ 典型例子
  - 2的乘方
  - ASCII码表

## 2.2.2.1 2的乘方

- ◆ 计算机内部数据采用二进制表示，在2的乘方位置自动形成内部边界

2的乘方	范围
1	0或1
4	0 ~ 15
8	0 ~ 255
16	0 ~ 65,535
32	0 ~ 4,294,967,295

## 2.2.2.1 2的乘方

- ◆ 2的乘方边界问题例子：32位的Unix操作系统和Linux操作系统时间溢出问题
- ◆ C语言中用 `time_t` 来代表时间和日期，记载从1970-01-01到某个时间所经历的秒数。
- ◆ `time_t` 是int型，在32位编译器中，int以32位存储，第一位是符号位，其余31位用来存数字，31位数字可以存储的最大值为2147483647
- ◆ 从1970-01-01开始，31位数字可以表示的秒数最多可以用到2038-01-19 03:14:07
- ◆ 解决：用64位表示`time_t`类型



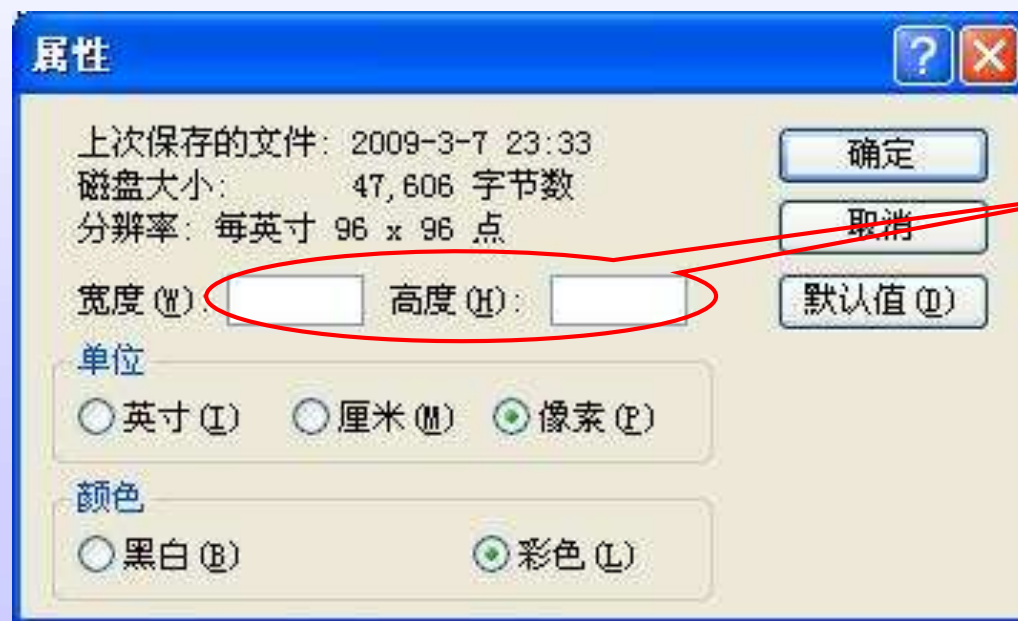
## 2.2.2.2 ASCII Table

Character	ASCII Value	Character	ASCII Value
<b>Null</b>	<b>0</b>	<b>B</b>	<b>66</b>
<b>Space</b>	<b>32</b>	<b>Y</b>	<b>89</b>
<b>/</b>	<b>47</b>	<b>Z</b>	<b>90</b>
<b>0</b>	<b>48</b>	<b>[</b>	<b>91</b>
<b>1</b>	<b>49</b>	<b>`</b>	<b>96</b>
<b>2</b>	<b>50</b>	<b>a</b>	<b>97</b>
<b>9</b>	<b>57</b>	<b>b</b>	<b>98</b>
<b>;</b>	<b>58</b>	<b>y</b>	<b>121</b>
<b>@</b>	<b>64</b>	<b>z</b>	<b>122</b>
<b>A</b>	<b>65</b>	<b>{</b>	<b>123</b>

一段示例代码：  
`if(ch>=47 && ch<=57)  
{  
 //这是一个数字  
}`

## 2.2.3 默认值、零值、空白测试

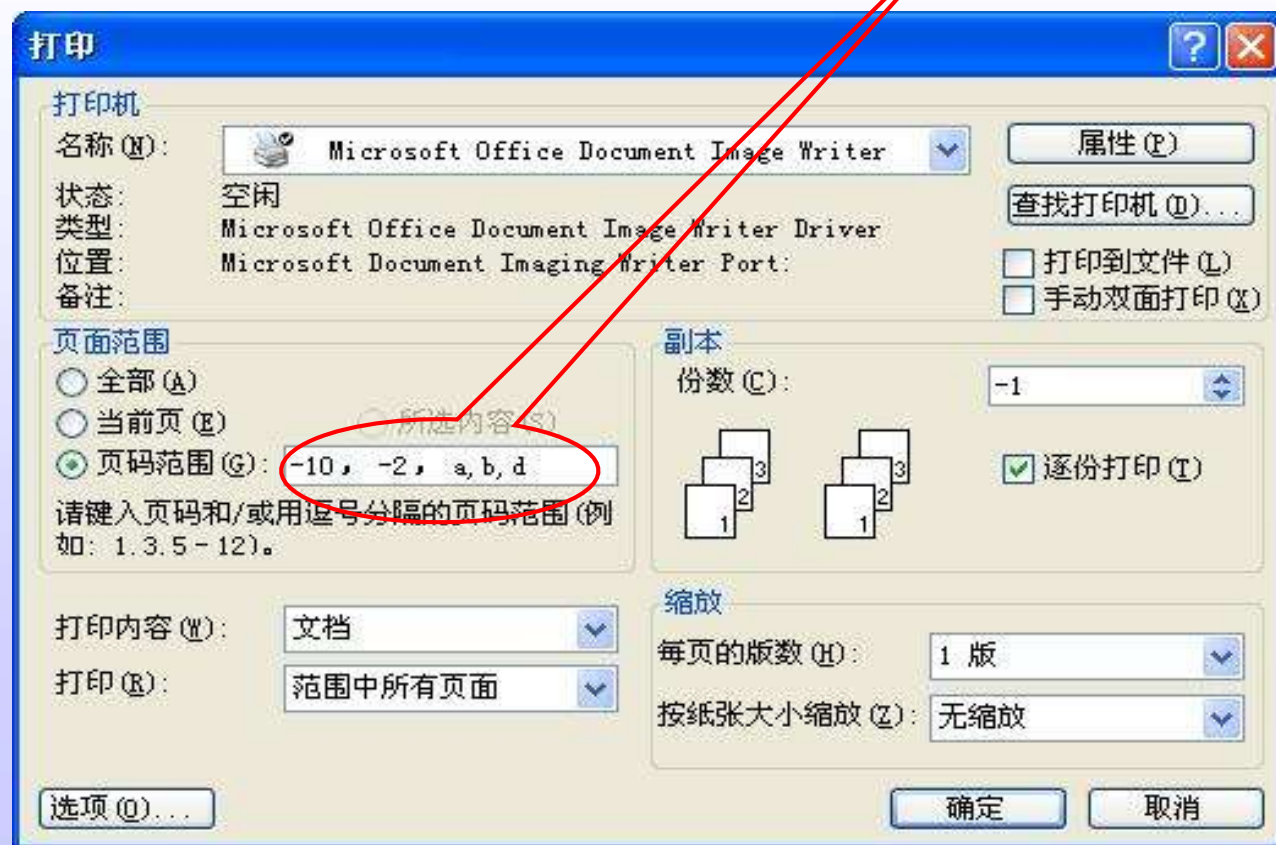
- ◆ 一定要考虑建立默认值、零值、空白等条件，这些值通常在软件中特殊处理，要建立单独的等价区间。



空白输入

## 2.2.4 非法、错误等垃圾数据测试

- ◆ 错误数据测试，这是失败测试的对象
- ◆ 比如，打印对话框的页码范围内输入“-10”，“-2”，“b”，“d”等非法数据



## 2.3 决策表测试

- ◆ 也叫判定表
- ◆ 等价类划分和边界值分析方法，通常只考虑测试单因素的输入条件，未考虑到各种输入条件之间的关系以及组合情况
- ◆ 实际情况更多的是各种输入条件之间可能存在关联，由输入条件的组合确定系统应该完成的任务
- ◆ 若输入条件和输出条件都是只有0和1两个取值，可以采用决策表方法进行测试用例设计

## 2.3.1 决策表测试的一般形式

- ◆ 决策表是一种利用表格形式表达系统业务规则的工具。
- ◆ 决策表测试的优点是可以生成测试条件的各种组合

	规则1	规则2	...	规则p
条件1				
条件2				
...				
条件m				
动作				
动作1				
动作2				
...				
动作n				

## 2.3.2 决策表测试的组成

- ◆ 决策表由条件、动作和规则构成
  - 条件（**Condition**）：代表软件的各种输入条件，通常认为条件的先后顺序与动作无关
  - 动作（**Actions**）：代表依据某种输入条件组合系统采取的行动
  - 规则（**Rules**）：定义了唯一的条件组合以及由此引发的系统动作，即决策表中的列

## 2.3.3决策表举例

Printer troubleshooter

		Rules							
Conditions	Printer does not print	Y	Y	Y	Y	N	N	N	N
	A red light is flashing	Y	Y	N	N	Y	Y	N	N
	Printer is unrecognised	Y	N	Y	N	Y	N	Y	N
Actions	Check the power cable			●					
	Check the printer-computer cable	●		●					
	Ensure printer software is installed	●		●		●		●	
	Check/replace ink	●	●			●	●		
	Check for paper jam		●		●				

## 2.4 组合测试

- ◆ 多个输入参数、每个参数有多个取值的情况下，采用组合测试方法
- ◆ 比如，某网站需要进行的测试参数：
  - OS: Win/MacOS/Unix/Android/iOS
  - 浏览器: IE/Chrome/QQ浏览器/FireFox
  - 屏幕分辨率: 600\*800/240\*400/1024\*768/1920\*1680
  - .....
- ◆ 对所有参数的全部组合进行测试是不现实的

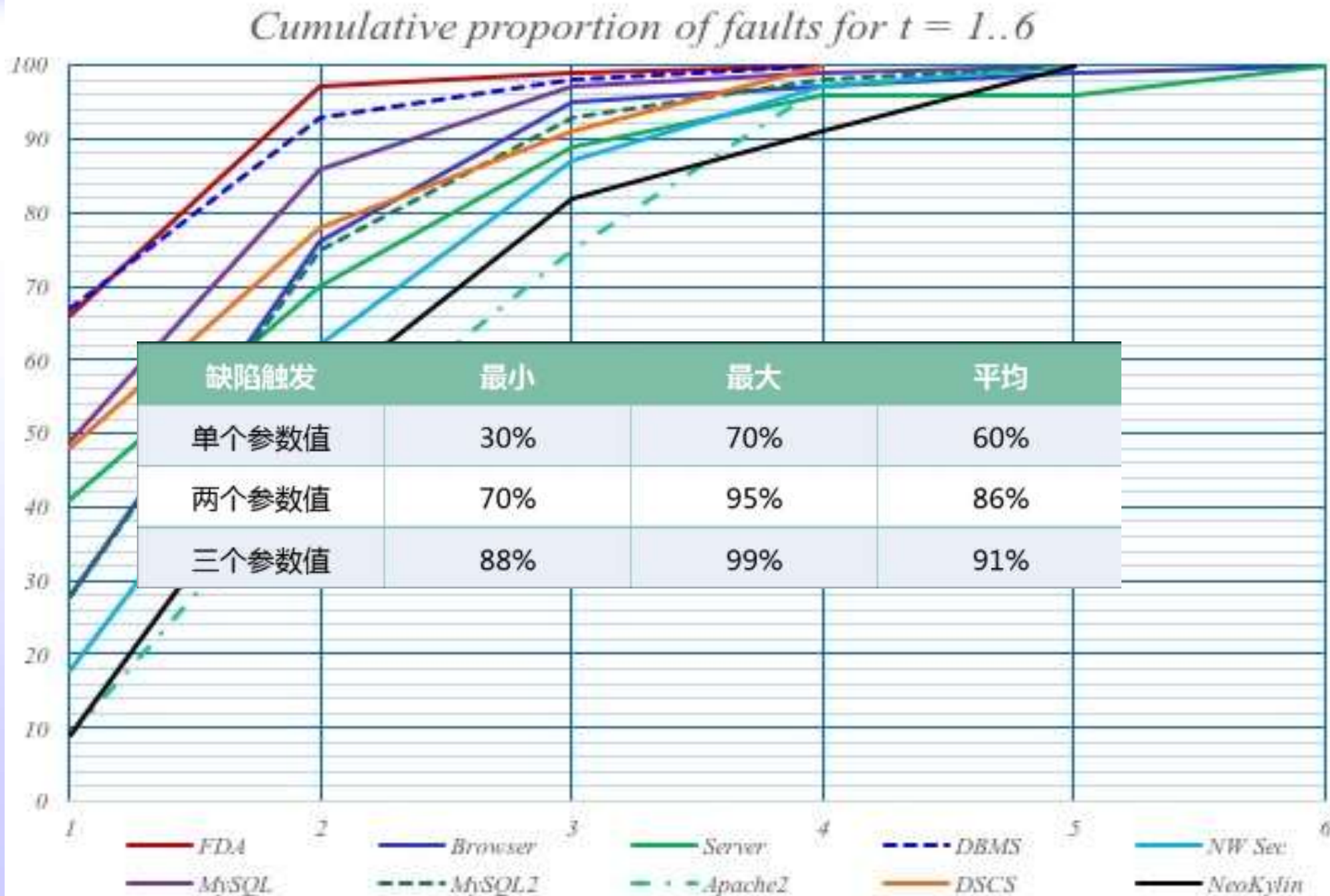


## 2.4 组合测试

- ◆ 通常采用的是两两组合（2-way, pair-wise）：任意两个参数的任意组合都出现
  - 例：假设有4个参数，每个参数有3个取值，其全部组合情况是  $3*3*3*3=81$  种，其两两组合情况最少可以只有9种，右图
- ◆ 也可以是三三组合、四四组合(3-way, 4-way), .....
- ◆ 研究表明，两两组合的模块覆盖率可达93.5%，判断覆盖可达83%

序号	A	B	C	D
1	1	1	1	2
2	1	2	2	3
3	1	3	3	1
4	2	1	2	1
5	2	2	3	2
6	2	3	1	3
7	3	1	3	3
8	3	2	1	1
9	3	3	2	2

不同组合强度下，发现缺陷的能力：一些不同研究人员的结果



## 两两组合测试举例：在线购物网站测试

- ◆ 根据不同条件，有不同的操作界面和功能：
  - ① 登录方式：未登录、第一次登录、正常登录
  - ② 会员状态：非会员、会员、VIP会员、雇员
  - ③ 折扣：无、假日95折、会员9折、VIP会员8折
  - ④ 物流方式：标准、快递、加急
- ◆ 完全组合，数量： $3*4*4*3=144$ 种组合
- ◆ 两两组合：共有17种组合

序号	登录方式	会员状态	折扣	物流
1	未登录	雇员	假日95折	快递
2	正常登录	会员	假日95折	加急
3	第一次登录	VIP会员	无	标准
4	未登录	非会员	VIP会员8折	标准
5	正常登录	非会员	无	快递
6	未登录	雇员	无	加急
7	第一次登录	非会员	假日95折	加急
8	第一次登录	VIP会员	会员9折	快递
9	未登录	会员	无	标准
10	正常登录	雇员	VIP会员8折	标准
11	未登录	VIP会员	VIP会员8折	加急
12	未登录	雇员	会员9折	标准

可以使用工具自动生成t-way组合  
如微软PICT(<http://www.pairwise.org/tools.asp>)

15	第一次登录	雇员	VIP会员8折	快递
16	第一次登录	会员	会员9折	快递
17	正常登录	会员	VIP会员8折	标准

# PICT使用实例

- ◆ 创建输入文件， a plain-text model file

Login: 未登录, 第一次登录, 正常登录  
Member: 非会员, 会员, VIP会员, 雇员  
Sale: 无, 假日95折, 会员9折, VIP会员8折  
Transport: 标准, 快递, 加急

- ◆ 命令行执行

Usage: pict model [options]

Options:

/o:N - Order of combinations (default: 2)

/d:C - Separator for values (default: ,)

.....

## 去除不合理组合

PLATFORM: x86, ia64, amd64

CPU: Single, Dual, Quad

RAM: 128MB, 1GB, 4GB, 64GB

OS: NT4, Win2K, WinXP, Win2K3

IE: 4.0, 5.0, 5.5, 6.0

```
IF [PLATFORM] in {"ia64", "amd64"} THEN  
[OS] in {"WinXP", "Win2K3"};  
IF [PLATFORM] = "x86" THEN [RAM] <>  
"64GB";
```

1	PLATFORM	CPU	RAM	OS	IE
2	ia64	Single	64GB	Win2K3	5
3	x86	Dual	4GB	Win2K	5
4	ia64	Quad	4GB	WinXP	5.5
5	x86	Quad	128MB	NT4	6
6	amd64	Single	128MB	WinXP	4
7	amd64	Dual	64GB	WinXP	6
8	x86	Quad	1GB	Win2K	4
9	x86	Dual	1GB	NT4	5.5
10	ia64	Dual	4GB	Win2K3	4
11	x86	Single	1GB	Win2K3	6
12	amd64	Quad	128MB	Win2K3	5.5
13	amd64	Quad	1GB	WinXP	5
14	x86	Single	128MB	Win2K	5.5
15	x86	Dual	128MB	NT4	5
16	ia64	Quad	64GB	WinXP	5.5
17	ia64	Dual	1GB	Win2K3	6
18	x86	Single	4GB	NT4	4
19	x86	Quad	4GB	Win2K	6
20	amd64	Dual	4GB	WinXP	4
21	ia64	Single	128MB	Win2K3	5.5
22	amd64	Dual	64GB	WinXP	4
23	x86	Dual	4GB	WinXP	4

## 2.5 用例/场景测试

- ◆ **场景**是用户具体使用软件时的一个片断。**用例**是UML设计中用来表达用户需求的一种手段，这里把用例定义为用户与系统之间的一次交互。实际上，用例也是使用软件的一个场景
- ◆ 基于场景的测试方法就是设计各种测试用例来**反映软件系统****在现实中的典型应用、极端应用**等情况，从实际使用的角度来测试软件



## 2.6 状态测试

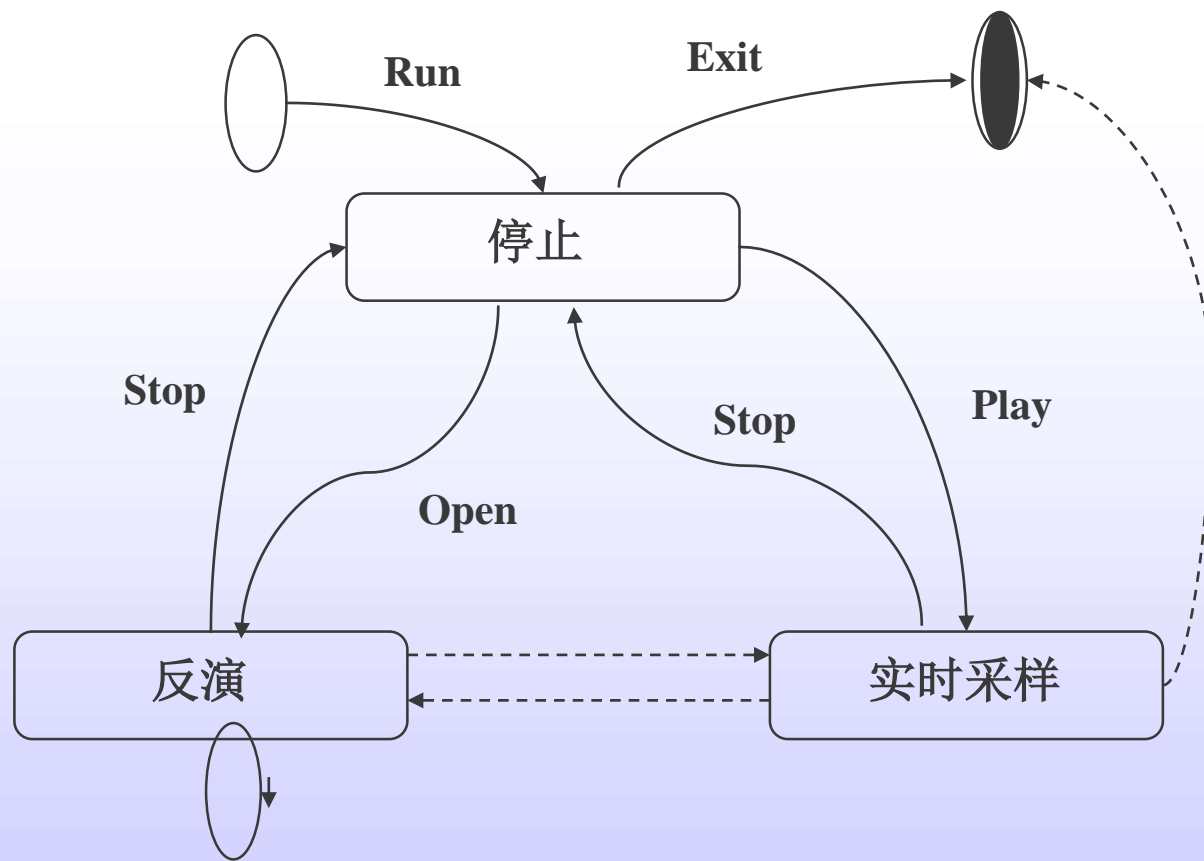
- ◆ **软件状态**是指软件当前所处的情况或者模式
- ◆ 改变了软件的状态，软件的代码会进入到不同的处理分支。  
软件状态测试相当于测试软件的不同分支路径
- ◆ 使用状态转换表



## 2.6.1 建立状态转换图

- ◆ 进行状态测试的第一步是建立状态转换图。状态图包括以下项目：
  1. 软件可能进入的每一种独立状态（如果不能确定，那么认为是）
  2. 从一种状态转入另一种状态所需的输入和条件
  3. 进入和退出某种状态时的设置条件及输出结果

## 2.6.2 状态转换图举例



信号采集系统状态转换图

## 2.7 基于经验的测试

- ◆ 除了上面讲解的各种正规测试技术之外，还有一些测试技术是基于经验的测试技术，这些测试技术对于克服杀虫剂现象是有益处的

## 2.7.1 像新手一样的测试

- ◆ 像新手一样的测试，由于新手不遵循任何规则、次序、也不做任何假定，在这种非常规的使用方式下往往会暴露出软件考虑不周的地方

## 2.7.2 在已经找到缺陷的地方再找找

◆ 这样做有两个原因：

1. 找到软件缺陷越多，就说明那里的软件缺陷越多，需要再继续查找
2. 许多程序员倾向于只修复报告出来的软件缺陷，但这个缺陷可能是由其他根源的缺陷引起的，再测试这个缺陷可能会发现其他问题

## 2.7.3 凭借经验、直觉和预感

- ◆ 要想成为真正的软件测试人员，积累经验是不可代替的
- ◆ 经验和直觉是不可言传的，必须经过长期实践的积累
- ◆ 随着在测试工作中的逐步提高，学习测试不同类型和规模的产品，就会得到各种提示和技巧以便更加有效地找出软件缺陷

# 提纲

- ◆ 黑盒测试概述
- ◆ 黑盒测试技术
- ◆ 黑盒测试举例

### 3. 黑盒测试举例

使用等价类划分技术设计测试用例



## 3.1 用户注册信息合理性测试

### 测试场景：

申请某网站账号时，要求输入用户名、密码及确认密码。要求：

- 用户名：长度**4-16**位，可以使用英文字母、数字、“-”、“\_”，首字符必须为字母或数字
- 密码：长度**6-16**位，可以使用英文字母、数字、“-”、“\_”，区分大小写
- 确认密码与密码完全一致

# 创建等价类表：

输入条件	有效等价类	无效等价类
用户名	4-16位 (1)	少于4位 (8)
		多于16位 (9)
	首字符为字母 (2)	首字符不是字母或数字 (10)
	首字符为数字 (3)	
	英文字母/数字/-/_ 的组合 (4)	含有其它字符 (11)
密码	6-16位 (5)	少于6位 (12)
		多于16位 (13)
	英文字母/数字/-/_ 的组合 (6)	含有其它字符 (14)
确认密码	与密码完全相同 (7)	与密码内容相同，但字母大小写不同 (15)

# 设计测试用例

原则：

- (1)设计测试用例，使其尽可能多地覆盖尚未覆盖的有效等价类，重复这一步骤，直到所有有效等价类均被测试用例所覆盖；
- (2)对于无效等价类：设计一新测试用例，使其只覆盖一个无效等价类（避免遗漏某个错误），重复这一步骤直到所有无效等价类均被覆盖；

## 确定测试用例：

序号	用户名	密码	确认密码	覆盖等价类	预期输出
1	abc_123	abc_111	abc_111	1, 2, 4, 5, 6, 7	注册成功
2	123_abc	111_abc	111_abc	1, 2, 3, 5, 6, 7	注册成功
3	a_1	abc_111	abc_111	8, 2, 4, 5, 6, 7	用户名错误
4	abcdefgh012 3456789	abc_111	abc_111	9, 2, 4, 5, 6, 7	用户名错误
5	_abc123	abc_111	abc_111	10, 1, 4, 5, 6, 7	用户名错误
6	abc#123	abc_111	abc_111	11, 1, 2, 5, 6, 7	用户名错误
7	abc_123	a12	a12	12, 1, 2, 4, 6, 7	密码错误
8	abc_123	abcdefgh0123456789	abcdefgh0123456789	13, 1, 2, 4, 6, 7	密码错误
9	abc_123	abc#123	abc#123	14, 1, 2, 4, 5, 7	密码错误
10	abc_123	abc_111	aBc_111	15, 1, 2, 4, 5, 6	密码错误

## 3.2 根据输入判断三角形的形状

测试场景：读入3个整数作为一个三角形的3条边的长度值。程序判断三角形形状：不等边的、等腰的、等边的。

确定输入数据与三角形形状的关系：

- 设3条边分别为A, B, C。要构成三角形的3条边，必须满足：
  - $A > 0$ ,  $B > 0$ ,  $C > 0$ , 且  $A + B > C$ ,  $B + C > A$ ,  $A + C > B$ ;
- 如果是等腰的，则  $A = B$ , 或  $B = C$ , 或  $A = C$ ;
- 如果是等边的，则  $A = B$ , 且  $B = C$ , 且  $A = C$ 。

## 创建等价类表：

输入条件	有效等价类	无效等价类
是否三角形的三条边	$(A > 0)$ (1) $(B > 0)$ (2) $(C > 0)$ (3) $(A + B > C)$ (4) $(B + C > A)$ (5) $(A + C > B)$ (6)	$(A \leq 0)$ (7) $(B \leq 0)$ (8) $(C \leq 0)$ (9) $(A + B \leq C)$ (10) $(B + C \leq A)$ (11) $(A + C \leq B)$ (12)
是否等腰三角形	$(A = B)$ (13) $(B = C)$ (14) $(C = A)$ (15)	$(A \neq B) \ \&\& \ (B \neq C)$ $\&\& \ (C \neq A)$ (16)
是否等边三角形	$(A = B) \ \&\& \ (B = C) \ \&\& \ (C = A)$ (17)	$(A \neq B)$ (18) $(B \neq C)$ (19) $(C \neq A)$ (20)

# 确定测试用例：

序号	[A, B, C]	覆盖等价类	输出
1	[3, 4, 5]	(1), (2), (3), (4), (5), (6)	一般三角形
2	[0, 1, 2]	(7)	不构成三角形
3	[1, 0, 2]	(8)	
4	[1, 2, 0]	(9)	
5	[1, 2, 3]	(10)	
6	[1, 3, 2]	(11)	
7	[3, 1, 2]	(12)	
8	[3, 3, 4]	(1), (2), (3), (4), (5), (6), (13)	等腰三角形
9	[3, 4, 4]	(1), (2), (3), (4), (5), (6), (14)	
10	[3, 4, 3]	(1), (2), (3), (4), (5), (6), (15)	
11	[3, 4, 5]	(1), (2), (3), (4), (5), (6), (16)	非等腰三角形
12	[3, 3, 3]	(1), (2), (3), (4), (5), (6), (17)	是等边三角形
13	[3, 4, 4]	(1), (2), (3), (4), (5), (6), (14), (18)	非等边三角形
14	[3, 4, 3]	(1), (2), (3), (4), (5), (6), (15), (19)	
15	[3, 3, 4]	(1), (2), (3), (4), (5), (6), (13), (20)	

## 练习：报表日期

- ◆ 设某公司要打印2001~2005年的报表，报表日期为6位数字组成，其中，前4位为年份，后两位为月份。



# 第一步：划分等价类

输入及外部条件	有效等价类	无效等价类
报表日期的类型及长度	6位数字字符①	有非数字字符 ④ 少于6个数字字符 ⑤ 多于6个数字字符 ⑥
年份范围	在2001~2005之间②	小于2001 ⑦ 大于2005 ⑧
月份范围	在1~12之间③	小于1 ⑨ 大于12 ⑩

## 第二步：为有效等价类设计测试用例

对表中编号为①②③的3个有效等价类用一个测试用例覆盖：

测试数据	期望结果	覆盖范围
200105	输入有效	等价类①②③

第三步：为每个无效等价类至少设计一个测试用例

测试数据	期望结果	覆盖范围
001MAY	输入无效	等价类④
20015	输入无效	等价类⑤
2001001	输入无效	等价类⑥
200001	输入无效	等价类⑦
200801	输入无效	等价类⑧
200100	输入无效	等价类⑨
200113	输入无效	等价类⑩

# 黑盒测试中的研究课题

- ◆ 如何根据需求说明、UML等自动生成测试用例？
- ◆ 如何进行GUI、Web的自动化测试？

# 课堂作业

- ◆ 变量的命名规则规定如下：变量名的长度不多于10个字符，第一个字符必须为英文字母或下划线，其他字母可以英文字母、数字以及下划线的任意组合。

现需要对变量名的有效性进行测试，请用等价类划分法：

（1）进行等价类划分； （2）设计测试用例。



# 课后作业

1. 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。比如输入month、day和year分别为2003、2、28，则输出“2003年3月1日”。要求输入变量month、day和year均为整数值，并且满足下列条件：

✓条件1  $1 \leq \text{month} \leq 12$

✓条件2  $1 \leq \text{day} \leq 31$

✓条件3  $1912 \leq \text{year} \leq 2050$

现需要对NextDate函数进行测试，请用等价类划分法：（1）进行等价类划分；（2）设计测试用例。

# 课后作业

2. 系统功能：根据参保车辆各方面的不同情况（如下），确定保险费标准。要求针对下列因素，使用PICT工具给出此系统的pair-wise组合测试用例：

- ① 驾驶记录：过去5年内没有违规，过去3年内没有违规，过去3年内违规小于3次，过去3年内违规3次或3次以上，过去1年内违规3次或3次以上
- ② 汽车型号：一般国产车，高档国产车，进口车，高档进口车
- ③ 使用汽车的方式：出租车，商务车，私家车
- ④ 所住的地区：城市中心地带，市区，郊区，农村
- ⑤ 受保的项目：全保，自由组合，最基本保险
- ⑥ 司机的驾龄： $\leq 1$ 年， $\leq 3$ 年， $\leq 5$ 年， $\leq 10$ 年， $> 10$ 年
- ⑦ 保险方式：首次参保，第二次参保，连续受保( $\geq 3$ 次)
- ⑧ 约束条件：(1)驾龄少于5年的，不能驾驶出租车；(2)“高档国产车”“高档进口车”必须“全保”



The End  
Any Question?

