Software Quality Assurance and Testing

Software College Northeastern University

Chapter 4 Black-Box Testing

wuchenni@qq.com

1. （1）

有效等价类：

Name：非空， ≤ 20 个字母字符

Age：0 ≤ age ≤120，整数

无效等价类：

Name：空，＞20 个，包含非字母字符

Age：＞120，＜0，非整数

（2）

name ， age	Expected output
Amy ， 30	Amy， 30
30，empty string	name is empty string， 30
Amyamyamyamyamyamyamy，30	Amyamyamyamyamyamyam，30
Amy1，30	name contains analphabetic characters
Amy，16	age is out of range
Amy，123	age is out of range

2. Code：{99,100,500,998,999}，Qty：{1,2,50,99,100}

二者相互组合形成测试用例并给出对应的预期结果即可。

3.（1）

Conditions	1	2	3	4	5
1. Deductible met?	Y	Y	Y	Y	N
2. Type of visit	D	D	H	H	-
3. Preferred Provider?	Y	N	Y	N	-
Actions					
1. Reimburse 65%	X				
2. Reimburse 95%			X		
3. Reimburse 50%		X			
4. Reimburse 80%				X	
5. No reimbursement					X

（2）每一列转换成 1 条测试用例即可。

4.（1）Y1：平年；Y2：闰年；M1：月在 2、4、6、8、9、11 月份中；M2：月在 5、7、10、12 月份中；M3 月是 3 月；M4 月是 1 月；D1：除 1 日的任何一个日期；D2：日期是 1 日。

年在？	-	-	-	-	Y1	Y2
月在？	M1-M4	M1	M2	M4	M3	M3
日在？	D1	D2	D2	D2	D2	D2
年减 1				√		
月减 1		√	√		√	√
月份置为 12				√		
日期减 1	√					
日期置为 31		√		√		
日期置为 30			√			
日期置为 29						√
日期置为 28					√	

（2）每一列转换成 1 条测试用例即可。

5.（1）C1: the first argument is an existing file in the sender's home directory

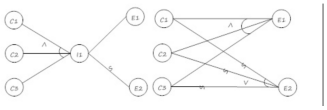
C2: the second argument is the name of the receiver's file server

C3: the third argument is the receiver's userid

E1: the file is successfully sent

E2: the sender obtains an error message

因果图可以是以下的任何一种形式：

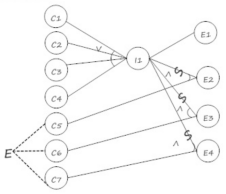


（2）

C1	0	1	1	1
C2	-	0	1	1
C3	-	-	0	1
E1				√
E2	√	√	√	

将决策表的每一列转换成具体的测试用例即可。

6.（1）C1：落点在棋盘外；C2：落点与起点不构成日字形；C3：落点处有自己方棋子；C4：落点方向邻近交叉点有棋子；C5：不属于 1-4 且落点处无棋子；C6：不属于 1-4 且落点处为对方棋子（非老将）；C7：不属于 1-4 且落点处为对方老将；E1：不移动棋子；E2：移动棋子；E3：移动棋子并去除对方棋子；E4：移动棋子并提示战胜对方，游戏结束。



（2）

C1	0	0	0	0	0	0	0	1
C2	0	0	0	0	0	0	1	-
C3	0	0	0	0	0	1	-	-
C4	0	0	0	0	1	-	-	-
C5	0	0	0	1	-	-	-	-
C6	0	0	1	0	-	-	-	-
C7	0	1	0	0	-	-	-	-
E1	0	0	0	0	1	1	1	1
E2	0	0	0	1	0	0	0	0
E3	0	0	1	0	0	0	0	0
E4	0	1	0	0	0	0	0	0

将决策表的每一列转换成具体的测试用例即可。

7. 试验次数=3**（2-1）+1=4

选择 $L_4(2^3)$，0：不显示/隐藏；1：显示

测试用例：

编号	输入	预期结果
1	上部不显示，中部隐藏，底部隐藏	隐藏整个页面
2	上部隐藏，中部显示，底部显示	只显示中部和底部
3	上部显示，中部隐藏，底部显示	只显示上部和底部
4	上部显示，中部显示，底部隐藏	只显示上部和中部

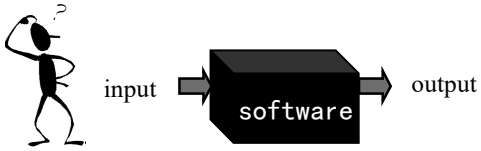
Contents

Chapter 4

Black-Box Testing

4.1 Introduction

- An approach to testing where the program is considered as a 'black-box'.
- The program test cases are based on the system specification.
- Test planning can begin early in the software process.



4.1 Introduction

- The advantages of black-box testing:
  - The test is unbiased because the designer and the tester are independent of each other.
  - The tester does not need knowledge of any specific programming languages.
  - The test is done from the user's point of view, not the designer's.
  - Test cases can be designed as soon as the specifications are completed.

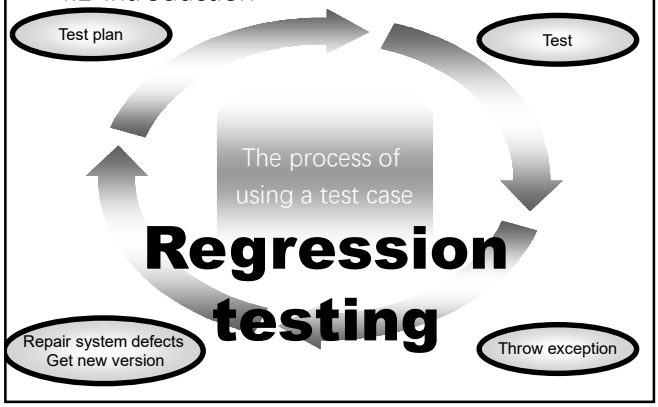
4.1 Introduction

- In theory, if black-box testing is used to find program errors, all kinds of possible input conditions and output conditions should be needed. But it is impossible for us to do that.

4.1 Introduction

- Review:
  - What is test case?
  - Operating processes
  - Regression testing

4.1 Introduction



4.1 Introduction

- Test case must be designed follow these characteristics :
  - Validity
  - Reusability
  - Easy organized
  - Measurability
  - Manageability

4.1 Introduction

- How to design a test case?
  - Understand the design of the product, specification, scenario
  - A test case should try to cover one or more situation
  - Do not copy the specification
  - Localized software testing
  - State
  - To decompose the test cases

4.1 Introduction

- Test methods
  - Equivalence partitioning
  - Boundary value analysis
  - Decision tables
  - Cause-effect graphing
  - Orthogonal array testing
  - Scenario testing
  - …

4.2 Equivalence Partitioning

- Input data and output results often fall into different classes where all members of a class are related.
- Each of these classes is an equivalence partition or domain where the program behaves in an equivalent way for each class member.
- Test cases should be chosen from each partition.

4.2 Equivalence Partitioning

- **Equivalence class definition**
  - It is the sub congregation of an input domain or an output domain.
  - Reasonable suppose： For tester, typical data are equal to other data in the same equivalence class.

4.2 Equivalence Partitioning

- Partition situation
  - Valid equivalence class
    - For specification, it is a congregation consists of reasonable and meaningful data.
    - It can be used to check whether the application realized the predicted functions and performance.
  - Invalid equivalence class
    - For specification, it is a congregation consists of unreasonable and meaningless data.
    - It can be used to check whether the invalid data is handled correctly.

4.2 Equivalence Partitioning

- Guidelines for generating equivalence classes for variables:
  - Range: one class with values inside the range and two with values outside the range.
  - Speed∈[60…90]

4.2 Equivalence Partitioning

- Guidelines for generating equivalence classes for variables:
  - String: At least one containing all legal strings and one containing all illegal strings. Legality （合法性）is determined based on constraints on the length and other semantic features of the string.

4.2 Equivalence Partitioning

- Guidelines for generating equivalence classes for variables
  - Enumeration: each value in a separate class.
  - auto_color∈{red, blue, green}

4.2 Equivalence Partitioning

- Guidelines for generating equivalence classes for variables:
  - Array: One class containing all legal arrays, one containing only the empty array, and one containing arrays larger than the expected size.

4.2 Equivalence Partitioning

Note

规定范围和个数，有效一个无效俩。

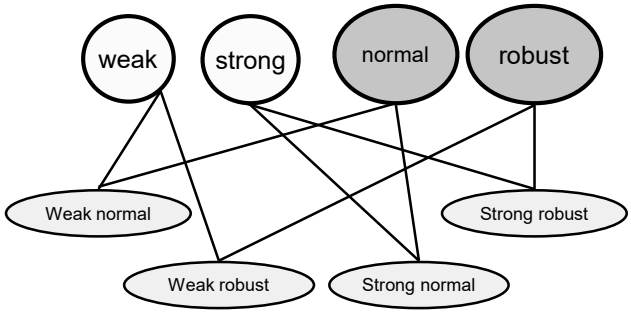
必须条件布尔量，有效无效一没错。

不同输入不同待，有效多来无效单。

输入数据有限制，有效单一无效多。

等价划分综合断，类中有类细定度。

4.2 Equivalence Partitioning



4.2 Equivalence Partitioning

- Single fault
  - Assumption in reliability theory: failures are only rarely the result of the simultaneous occurrence of two (or more) faults.

4.2 Equivalence Partitioning

- The steps of design test case
  - Find out equivalence classes
  - Build a table and list all equivalence classes
  - Every equivalence class has a unique No.
  - Design a new test case, try to make it include those exclusive equivalence classes which doesn't be included now, repeat this step until all valid equivalence classes are included.
  - Design a new test case, try to make it include an invalid equivalence class only which doesn't be included now, repeat this step until all invalid equivalence classes are included.

4.2 Equivalence Partitioning

- Example 1: Triangle problem
- It receive there integers a, b, c as input. If a, b, c satisfy these conditions below:
  - C1. 1≤a ≤200
  - C2. 1≤b ≤200
  - C3. 1≤c ≤200
  - C4. a < b + c
  - C5. b < a + c
  - C6. c < a + b
- Four possible outputs: Not a Triangle, Scalene（不等边的）, Isosceles（等腰的）, and Equilateral（等边的）.
- If no input value is satisfied then there will be a message. Example , c is out of range
- If a, b ,c satisfy the condition C1, C2,C3 then we can get four outputs exclusively:
  - If three sides have the same value then output "equilateral"
  - If only two sides have the same value then output "isosceles"
  - If any two sides have different value then output "scalene"
  - If one condition among C4, C5 and C6 isn't be satisfied than output "not a triangle"

4.2 Equivalence Partitioning

- Step1: find out equivalence classes
  - R1={<a, b, c>: the triangle with sides a ,b, and c is equilateral（等边的）}
  - R2={<a, b, c>: the triangle with sides a, b, and c is isosceles（等腰的）}
  - R3={<a, b, c>: the triangle with sides a, b, and c is scalene（不等边的）}
  - R4={<a, b, c>: the triangle with sides a, b, and c not form a triangle}

4.2 Equivalence Partitioning

- Step2 select test cases

Test cases	a	b	c	Expected output
WN1	5	5	5	Equilateral
WN2	2	2	3	Isosceles
WN3	3	4	5	Scalene
WN4	4	1	2	Not a triangle

SR1	-1	-1	5	a、b out of range
SR2	5	-1	-1	b、c out of range
SR3	-1	5	-1	a、c out of range
SR4	-1	-1	-1	a、b、c out of range

4.2 Equivalence Partitioning

- Example 2: Equivalence Class Test Cases for the NextDate Function （1812≤year≤2015 ）
  - Step 1: find out equivalence classes
    - Valid equivalence class:
    - M1 = {month: 1≤month≤12}
    - D1 = {date: 1≤date≤31}
    - Y1 = {year: 1812≤year≤2015}
    - Invalid equivalence class:
    - M2 = {month: month<1}
    - M3 = {month: month>12}
    - D2 = {date: date<1}
    - D3 = {date: date>31}
    - Y2 = {year: year<1812}
    - Y3 = {year: year>2015}

4.2 Equivalence Partitioning

- Strong robust equivalence class for NextDate function

Test case ID	month	date	year	Expected output
SR1	-1	15	1912	Invalid input
SR2	6	-1	1912	Invalid input
SR3	6	15	1811	Invalid input
SR4	-1	-1	1912	Invalid input
SR5	6	-1	1811	Invalid input
SR6	-1	15	1811	Invalid input
SR7	-1	-1	1811	Invalid input

4.2 Equivalence Partitioning

- **练习：前亚利桑那州境内的一位步枪销售商销售密苏里州制造的步枪机、枪托和枪管。枪机（Lock）卖45美元，枪托（Stock）卖30美元，枪管（Barrel）卖25美元。销售商每月至少要销售一支完整的步枪，且生产限额是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。每访问一个镇子之后，销售商都给密苏里州步枪制造商发出电报，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，通知－1个枪机被售出。这样步枪制造商就知道当月的销售情况，并计算销售商的佣金如下：销售额不到（含）1000美元的部分为10%，1000（不含）～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。**

4.2 Equivalence Partitioning

- Exercise : commission problem test case
  - Input domain equivalence class
  - Output domain equivalence class

4.2 Equivalence Partitioning

- Input equivalence class:
  - Valid equivalence class:
    - L1 = {Lock: 1≤Lock≤70}
    - L2 = {Lock = －1}
    - S1 = {Stock: 1 ≤Stock≤80}
    - B1 = {Barrel : 1 ≤ Barrel ≤90}
  - Invalid equivalence class:
    - L3 = {Lock : Lock = 0 or Lock < －1}
    - L4 = {Lock : Lock > 70}
    - S2 = {Stock : Stock < 1}
    - S3 = {Stock : Stock > 80}
    - B2 = {Barrel : Barrel < 1}
    - B3 = {Barrel: Barrel > 90}

4.2 Equivalence Partitioning

- Output equivalence class:
  - Sales = 45×Lock ＋30×Stock＋25×Barrel
  - S1 = {<Lock，Stock，Barrel>: Sales≤1000}
  - S2 = {< Lock，Stock，Barrel >: 1000< Sales ≤1800}
  - S3= {< Lock，Stock，Barrel >: Sales >1800}

4.2 Equivalence Partitioning

- **练习**
  - 有一个报表处理系统，要求用户输入处理报表的日期。假如日期限制在2000年1月至2020年12月，即系统只能对该段时期内的报表进行处理。如果用户输入的日期不在这个范围内，则显示“错误信息”。并且此系统规定日期由年月的6位数字组成，前四位代表年，后两位代表月。
  - 要求:
    - 写出日期的等价列划分。
    - 生成测试用例。

4.2 Equivalence Partitioning

- 划分等价类

输入	合理等价类	不合理等价类
报表日期	① 6位数字字符	② 有非数字字符
		③ 少于6个数字字符
		④ 多于6个数字字符
年份范围	⑤ 2000-2020	⑥ 小于2000
		⑦ 大于2020
月份范围	⑧ 1-12	⑨ 等于0
		⑩ 大于12

4.2 Equivalence Partitioning

- 生成测试用例

覆盖等价类	输入	预期结果
①、⑤、⑧	201006	正常处理
②无效，5有效，8有效	200a0b	错误信息
③，5，8	20102	错误信息
④，5，8	2011112	错误信息
⑥，1，8	198802	错误信息
⑦，1，8	203011	错误信息
⑨，1，5	200000	错误信息
⑩，1，5	202013	错误信息

4.3 Boundary Value Analysis

- Boundary value analysis focuses on the boundary of the input space to identify test cases.
- The rationale（基本原理） behind boundary value testing is that errors tend to occur near the extreme values of an input variable.
  - e.g. loop conditions (< instead of ≤), counters

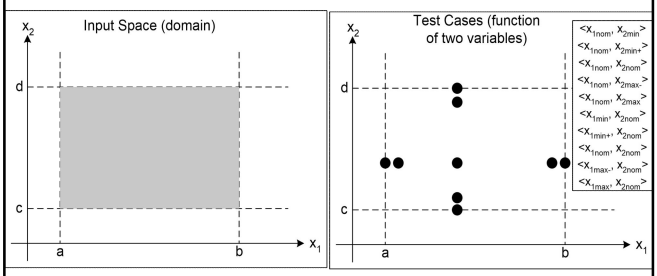
4.3 Boundary Value Analysis

- Basic idea:
  - use input variable values at their minimum (min), just above the minimum (min+), a nominal value (nom), just below their maximum (max-), and at their maximum (max)

4.3 Boundary Value Analysis

- Attention:
  - Boundary Value Analysis is based Single fault.

4.3 Boundary Value Analysis



4.3 Boundary Value Analysis

- Limitations of Boundary Value Analysis
  - Boundary value analysis works well when the program to be tested is a function of several independent variables that represent bounded physical quantities.
  - e.g. NextDate test cases are inadequate (little stress on February, dependencies among month, day, and year)
  - e.g. variables refer to physical quantities, such as temperature, air speed, load etc.

4.3 Boundary Value Analysis

- Robustness Testing
  - Simple extension of boundary value analysis
  - A value slightly greater than the maximum (max+) and a value slightly less than the minimum (min-)
  - Focuses on the expected outputs
  - e.g. exceeding load capacity of a public elevator
  - Forces attention on exception handling

4.3 Boundary Value Analysis

- Triangle problem
  - Three inputs: a、b、c

$1 \leqslant a \leqslant 200$

$1 \leqslant b \leqslant 200$

$1 \leqslant c \leqslant 200$

a = {1，2，100，199，200}

b = {1，2，100，199，200}

c = {1，2，100，199，200}

4.3 Boundary Value Analysis

Triangle problem test case

Test case	A	B	C	Expected output
1	100	100	1	Isosceles
2	100	100	2	Isosceles
3	100	100	100	Equilateral
4	100	100	199	Isosceles
5	100	100	200	Not a triangle
6	100	1	100	Isosceles
7	100	2	100	Isosceles
8	100	100	100	Equilateral
9	100	199	100	Isosceles
10	100	200	100	Not a triangle
11	1	100	100	Isosceles
12	2	100	100	Isosceles
13	100	100	100	Equilateral
14	199	100	100	Isosceles
15	200	100	100	Not a triangle

At minimum

Just above the minimum

Nominal value

Below their maximum

At their maximum

4.3 Boundary Value Analysis

- NextDate function problem
  - NextDate is a function has three variables (month, date, and year), it return the next date which user input. Month, date and year are all integer, and satisfy these conditions below:

$1 \leqslant month \leqslant 12$

$1 \leqslant date \leqslant 31$

$1812 \leqslant year \leqslant 2015$

month = {1，2，6，11，12}

date = {1，2，15，30，31}

year = {1812，1813，1912，2014，2015}

4.3 Boundary Value Analysis

Next Date function test case

Test case	month	date	year	Expected output
1	6	15	1812	6/16/1812
2	6	15	1813	6/16/1813
3	6	15	1912	6/16/1912
4	6	15	2014	6/16/2011
5	6	15	2015	6/16/2012
6	6	1	1912	6/2/1912
7	6	2	1912	6/3/1912
8	6	15	1912	6/16/1912
9	6	30	1912	7/1/1912
10	6	31	1912	Invalid input
11	1	15	1912	1/16/1912
12	2	15	1912	2/16/1912
13	6	15	1912	6/16/1912
14	11	15	1912	11/16/1912
15	12	15	1912	12/16/1912

At minimum

Just above the minimum

Nominal value

Below their maximum

At their maximum

4.3 Boundary Value Analysis

- 练习
- 有一个报表处理系统，要求用户输入处理报表的日期。假如日期限制在2000年1月至2020年12月，即系统只能对该段时期内的报表进行处理。如果用户输入的日期不在这个范围内，则显示“错误信息”。并且此系统规定日期由年月的6位数字组成，前四位代表年，后两为代表月。
  - 要求：用边界值分析法进行测试并生成测试用例。

4.3 Boundary Value Analysis

- 年 = { 2000,2001,2010,2019,2020 }
- 月 = { 1,2,6,11,12 }

测试用例	月	年	预期输出
1	1	2010	201001
2	2	2010	201002
3	6	2010	201006
4	11	2010	201011
5	12	2010	201012
6	6	2000	200006
7	6	2001	200106
8	6	2019	201906
9	6	2020	202006

Keystone

- What is black-box testing?
- Guidelines for generating equivalence classes for variables
- How to design test case using equivalence classes method?
- What is boundary value analysis?

Summary

- 等价类测试的弱形式不如强形式测试全面。
- 无效值会引起运行错误的时候（实现语言是强类型），则没有必要做健壮形式的测试。
- 错误条件很重要的时候，健壮测试很重要。
- 边界值测试是等价类测试的一种补充，两者结合可以加强测试效果。
- 决策表技术可以解决变量之间依赖的问题。
- 要进行多次尝试，确认最合适的等价类划分。

4.4 Decision Tables

- Of all the functional testing methods, those based on decision tables are the most rigorous, because decision tables themselves enforce logical rigor.
- They are ideal for describing situations in which a lot of combinations of actions are taken under varying sets of conditions.

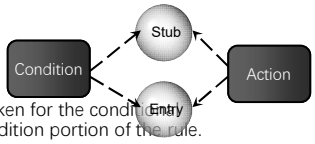
4.4 Decision Tables

『Reading Guideline "Decision Table

choice \ rule		1	2	3	4	5	6	7	8
question	tired	Y	Y	Y	Y	N	N	N	N
	interesting	Y	Y	N	N	Y	Y	N	N
	confused	Y	N	Y	N	Y	N	Y	N
suggestion	repeat					√			
	continue						√		
	skip							√	√
	rest	√	√	√	√				

4.4 Decision Tables

- Four portions of decision table:
  - Stub portion（question）
  - Entry portion（suggestion）
  - Condition portion
  - Action portion
- Rules indicate which actions are taken for the conditions circumstances indicated in the condition portion of the rule.



4.4 Decision Tables

• Decision table

stub	rule1	rule2	rule3、4	rule5	rule6	rule7、8
c1	T	T	T	F	F	F
c2	T	T	F	T	T	F
c3	T	F	—	T	F	—
a1	×	×		×		
a2	×				×	
a3		×		×		
a4			×			×

4.4 Decision Tables

- Attention: C3 in Rule3
  - "–" or "n/a" means conditions are careless or not available.

4.4 Decision Tables

• Steps of constructs a decision table
- Determine the rules number.
  - if there are n conditions, there must be 2^n rules. (True/False, Yes/No, 0/1)
- List all condition stubs and action stubs
- Enter condition entries
- Enter action entries, To obtain initial decision table , Then simplify decision table
  - If there are two more rules have the same action, and the conditions are very similar, it could be merged.
  - The Don't Care entry has two major interpretations: the condition is irrelevant, or the condition does not apply.

4.4 Decision Tables

A simplified decision table **" Reading Guideline"**

choice \ rule		1-4	5	6	7	8
question	tired	Y	N	N	N	N
	interesting	—	Y	Y	N	N
	confused	—	Y	N	Y	N
suggestion	repeat		√			
	continue			√		
	skip				√	√
	rest	√				

4.4 Decision Tables

• Decision table of triangle problem

c1: a、b、c get a triangle?	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: c=b?	—	Y	N	Y	N	Y	N	Y	N
a1: not a triangle	×								
a2: scalene									×
a3: isosceles					×		×	×	
a4: equilateral		×							
a5:impossible			×	×		×			

4.4 Decision Tables

• Refined Decision Table for the Triangle Problem

c1: a <b + 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 <b + a?	—	—	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
Rules count	32	16	8	1	1	1	1	1	1	1	1
a1: not a triangle	×	×	×								
a2: scalene											×
a3: isosceles							×		×	×	
a4: equilateral				×							
a5:impossible					×	×		×			

4.4 Decision Tables

- Test Case for the Triangle Problem

Test Case ID	a	b	c	Expected output
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	3	**等腰三角形**
DT10	3	2	2	**等腰三角形**
DT11	3	4	5	**不等边三角形**

4.4 Decision Tables

• NextDate Function（First Try）

Equivalence classes：
M1 = {month:month has 30 days}
M2 = {month:month has 31 days}
M3 = {month:month is February}
D1 = {day:1≤day≤28}
D2 = {day:day=29}
D3 = {day:day=30}
D4 = {day:day=31}
Y1 = {year:year is a leap year}
Y2 = {year:year is a common year}

Conditions				
C1: month in M1?	T			
C2: month in M2?		T		
C3: month in M3?			T	
C4: day in D1?				
C5: day in D2?				
C6: day in D3?				
C7: day in D4?				
C8: year in Y1?				
a1: impossible				
a2: next date				

4.4 Decision Tables

Conditions	Rule1	Rule2	Rule3
C1: month in M1?	T	–	–
C2: month in M2?	–	T	–
C3: month in M3?	–	–	T
Rules count	4	4	4
a1			

Conditions	**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: month in M1?	**T**	T	T	**T**	**T**	T	F	**F**	**T**	T	F	**F**
C2: month in M2?	**T**	T	F	**F**	**T**	T	**T**	**T**	**T**	F	**T**	**F**
C3: month in M3?	**T**	F	T	**F**	**T**	F	**T**	**F**	**T**	**T**	**T**	**T**
Rules count	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
a1												

4.4 Decision Tables

Conditions	1.1	1.2	1.3	1.4	2.3	2.4	3.4	
C1: month in M1?	T	T	T	T	F	F	F	F
C2: month in M2?	T	T	F	F	T	T	F	F
C3: month in M3?	T	F	T	F	T	F	T	F
Rules count	1	1	1	1	1	1	1	1
a1: Impossible	×	×	×		×			×

4.4 Decision Tables

•We can use the complete decision table to solve redundancy and inconsistencies.

•A redundant decision table

Conditions	**1-4**	**5**	**6**	**7**	**8**	**9**
C1	**T**	**F**	**F**	**F**	**F**	**T**
C2	–	**T**	**T**	**F**	**F**	**F**
C3	–	**T**	**F**	**T**	**F**	**F**
a1	×	×	×	–	–	×
a2	–	×	×	×	–	–
a3	×	–	×	×	×	×

4.4 Decision Tables

•A inconsistent decision table

Conditions	1-4	5	6	7	8	9
C1	T	F	F	F	F	T
C2	–	T	T	F	F	F
C3	–	T	F	T	F	F
a1	×	×	×	–	–	–
a2	–	×	×	×	–	×
a3	×	–	×	×	×	–

- Attention:
  - Rule4 and Rule9 are inconsistent.
  - Decision table is uncertain.
  - Be careful to use the “Don’t Care” entry.

4.4 Decision Tables

- NextDate Function（First Try）

Equivalence classes：

M1 = {month:month has 30 days}
M2 = {month:month has 31 days}
M3 = {month:month is February}
D1 = {day:1≤day≤28}
D2 = {day:day=29}
D3 = {day:day=30}
D4 = {day:day=31}
Y1 = {year:year is a leap year}
Y2 = {year:year is a common year}

Conditions				
C1: month in M1?	T			
C2: month in M2?		T		
C3: month in M3?			T	
C4: day in D1?				
C5: day in D2?				
C6: day in D3?				
C7: day in D4?				
C8: year in Y1?				
a1: Too many days in the month				
a2: Can not in a leap year				
a3: next date				

4.4 Decision Tables

- NextDate Function（Second Try）
  - Equivalence classes
    - M1 = {month： month has 30 days}
    - M2 = {month： month has 31 days}
    - M3 = {month： month is February}
    - D1 = {day： 1≤day≤28}
    - D2 = {day： day=29}
    - D3 = {day： day=30}
    - D4 = {day： day=31}
    - Y1 = {year： year is 2000}
    - Y2 = {year： year is a leap year}
    - Y3 = {year： year is a common year}

4.4 Decision Tables

- NextDate Function（Second Try Test Cases）

Conditions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C1：month in	M1	M1	M1	M1	M2	M2	M2	M2	M3	M3	M3	M3	M3	M3	M3	M3
C2：day in	D1	D2	D3	D4	D1	D2	D3	D4	D1	D1	D1	D2	D2	D2	D3	D4
C3：year in	—	—	—	—	—	—	—	—	Y1	Y2	Y3	Y1	Y2	Y3	—	—
Rule count	3	3	3	3	3	3	3	3	1	1	1	1	1	1	3	3
a1：impossible				×										×	×	×
a2：increment day	×	×			×	×	×			×		×				
a3：reset day			×					×	×		×		×			
a4：increment month			×					?	×		×		×			
a5：reset month								?								
a6：increment year								?								

4.4 Decision Tables

- Decision table of NextDate function（third time analysis）
  - M1 = {month： month has 30 days}
  - M2 = {month： month has 31 days per，except Dec.}
  - M3 = {month： month is December}
  - M4 = {month： month is February}
  - D1 = {day： 1≤day≤27}
  - D2 = {day： day=28}
  - D3 = {day： day=29}
  - D4 = {day： day=30}
  - D5 = {day： day=31}
  - Y1 = {year： year is a leap year}
  - Y2 = {year： year is a common year}

Third try

	C1：month in	C2：day in	C3：year in	a1：impossible	a2：increment day	a3：reset day	a4：increment month	a5：reset month	a6：increment year
1	M1	D1	—		×				
2	M1	D2	—		×				
3	M1	D3	—		×				
4	M1	D4	—			×	×		
5	M1	D5	—	×					
6	M2	D1	—		×				
7	M2	D2	—		×				
8	M2	D3	—		×				
9	M2	D4	—		×				
10	M2	D5	—			×	×		
11	M3	D1	—		×				
12	M3	D2	—		×				
13	M3	D3	—		×				
14	M3	D4	—		×				
15	M3	D5	—			×		×	×
16	M4	D1	—		×				
17	M4	D2	Y1		×				
18	M4	D2	Y2			×	×		
19	M4	D3	Y1			×	×		
20	M4	D3	Y2	×					
21	M4	D4	—	×					
22	M4	D5	—	×					

	C1：month in	C2：day in	C3：year in	a1：impossible	a2：increment day	a3：reset day	a4：increment month	a5：reset month	a6：increment year
1-3	M1	D1 D2 D3	—		×				
4	M1	D4	—			×	×		
5	M1	D5	—	×					
6-9	M2	D1 D2 D3 D4	—		×				
10	M2	D5	—			×	×		
11-14	M3	D1 D2 D3 D4	—		×				
15	M3	D5	—			×		×	×
16	M4	D1	—		×				
17	M4	D2	Y1		×				
18	M4	D2	Y2			×	×		
19	M4	D3	Y1			×	×		
20	M4	D3	Y2	×					
21-22	M4	D4 D5	—	×					

4.4 Decision Tables

ID	month	day	year	Expected output
1-3	4	15	2001	4/16/2001
4	4	30	2001	5/1/2001
5	4	31	2001	impossible
6-9	1	15	2001	1/16/2001
10	1	31	2001	2/1/2001
11-14	12	15	2001	12/16/2001
15	12	31	2001	1/1/2002
16	2	15	2001	2/16/2001
17	2	28	2004	2/29/2004
18	2	28	2001	3/1/2001
19	2	29	2004	3/1/2004
20	2	29	2001	impossible
21-22	2	30	2001	impossible

4.4 Decision Tables

- Guidelines and observations
  - The decision table technique is indicated for applications characterized by any of the following
    - prominent If-Then-Else logic
    - logical relationships among input variables
    - calculations involving subsets of the input variables
    - cause and effect relationships between inputs and outputs
  - Decision tables don't scale up very well
  - As with other techniques, iteration helps. The first set of conditions and actions you identify may be unsatisfactory. Use it as a stepping stone, and gradually improve on it until you are satisfied with a decision table.

4.4 Decision Tables

- 练习：
- 某货运站收费标准如下：如果收件地点在本省，则快件每公斤5元，慢件每公斤3元；如果收件地点在省外，则在20公斤以内（含20公斤）快件每公斤7元，慢件每公斤5元，而超过20公斤时，快件每公斤9元，慢件每公斤7元。请用决策表方法解决此问题。

4.4 Decision Tables

- 第一步：确定规则的数目。
- 条件：
  - （1）收件地在本省?
  - （2）是快件?
  - （3）重量不超过20公斤?
  - 根据公式计算$2^3 = 8$
  - 所以应有8条规则。
- 第二步：列出所有的条件桩和行动桩。
- 第三步：填入条件条目
- 第四步：填入行动条目

4.4 Decision Tables

	1	2	3	4	5	6	7	8
收件地在本省?	Y	Y	Y	Y	N	N	N	N
是快件?	Y	Y	N	N	Y	Y	N	N
重量不超过20公斤?	Y	N	Y	N	Y	N	Y	N
每公斤3元			×	×				
每公斤5元	×	×					×	
每公斤7元					×			×
每公斤9元						×		

4.4 Decision Tables

- 第五步：化简决策表

	1	2	3	4	5	6
收件地在本省?	Y	Y	N	N	N	N
是快件?	Y	N	Y	Y	N	N
重量不超过20公斤?	—	—	Y	N	Y	N
每公斤3元		×				
每公斤5元	×				×	
每公斤7元			×			×
每公斤9元				×		

4.5 Cause-Effect Graphing

- Review:
- Equivalence partitioning and Boundary value analysis
- Decision tables
- Effective method — Cause- Effect Graphing

4.5 Cause-Effect Graphing

- What is cause-effect graphing?
- Notation used in cause-effect graphing
- The steps of cause-effect graph design
- How to use cause-effect graph design test case?

4.5 Cause-Effect Graphing

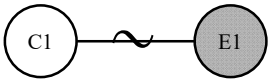
- Cause-Effect Graphing （*因果图*）
  - A cause-effect graph is a visual representation of a logical relationship among inputs and outputs that can be expressed as a Boolean expression.
  - A cause is any condition in the requirements that may effect the program output.
  - An effect is the response of the program to some combination of input conditions.

4.5 Cause-Effect Graphing

- Notation used in cause-effect graphing
- The relations between causes and effects

（1）Implication（*等价*）：if C1 is 1，then E1 is 1 also，else E1 is 0.

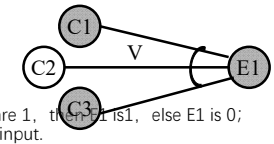
（2）Not （*非*）：if C1 is 1，then E1is 0，else E1 is 1.



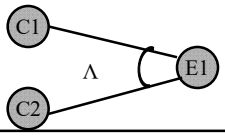
4.5 Cause-Effect Graphing

- The relations between causes and effects

（3）Or *（或）*： if C1or C2 or C3 is 1，then E1 is 1，else E1 is 0；“Or” may have arbitrary number of inputs.



（4）And *（与）*： if both C1 and C2 are 1，then E1 is1，else E1 is 0；“And” may have arbitrary number of input.

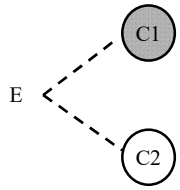


4.5 Cause-Effect Graphing

- Constraint（*约束*）
  - In practical issues, the input states also exist certain dependencies, called constraint.
  - There are constraints among outputs states .
  - These are represented as edges labeled with the constraint symbol using a dashed line（*虚线*）.

4.5 Cause-Effect Graphing

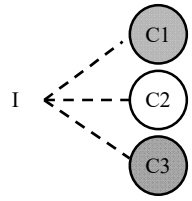
•Exclusive (E)*（异）*: either C1 or C2



C1	C2
0	0
1	0
0	1

4.5 Cause-Effect Graphing

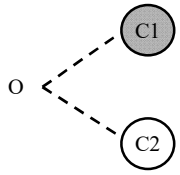
•Inclusive (I)*（或）*: at least C1 or C2 or C3



C1	C2	C3
0	0	1
0	1	0
1	0	0
1	1	0
1	0	1
0	1	1
1	1	1

4.5 Cause-Effect Graphing

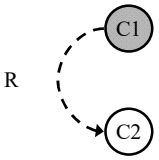
• One and only one (O)*（唯一）*: one, and only one, of C1 and C2



C1	C2
1	0
0	1

4.5 Cause-Effect Graphing

•Requires (R)*（要求）*: C1 requires C2

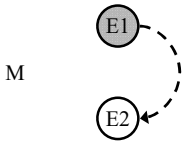


C1	C2
1	1
0	1
0	0

4.5 Cause-Effect Graphing

•Constraint among effects

• Masking (M)*（强制）*: E1 masks E2



E1	E2
0	0
0	1
1	0

4.5 Cause-Effect Graphing

• The steps of cause-effect graph design

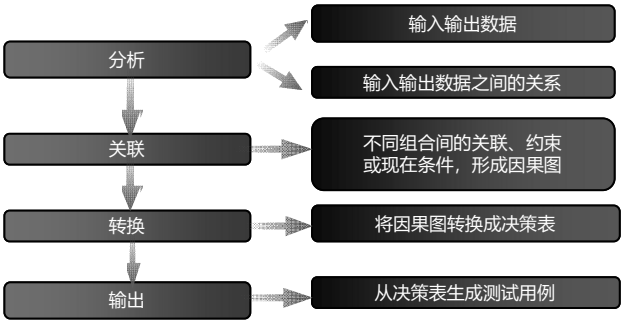
1.Identify causes and effects by reading the requirements. Each cause and effect is assigned a unique identifier. Note that an effect can also be a cause for some other effect.

2. Express the relationship between causes and effects using a cause effect graph.

3.Transform the cause-effect graph into a limited entry decision table, hereafter referred to simply as decision table.

4.Generate test cases from the decision table.

4.5 Cause-Effect Graphing



4.5 Cause-Effect Graphing

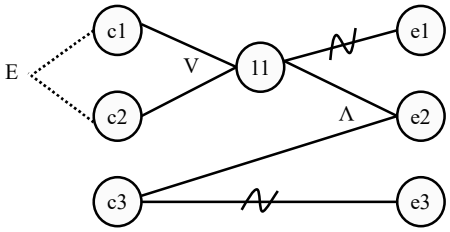
- Example 1
- The first character of input must be *'#'* or *'*'*;the second character of input must be ***number***. In this condition, the ***document will be modified***. If **the first character *isn't '#'* or *'*'*** then give the ***information N***. If the second character isn't number then give the ***information M***.
- Request
  - Please give the causes and effects
  - Find out the relationship between causes and effects
  - Find out the relationship between cause and cause
  - Draw cause-effect graph
  - Change the cause-effect graph to decision table
  - According to this judgment table, design test cases

4.5 Cause-Effect Graphing

- Answer
  - Cause
    - c1— the first character is #
    - c2 —the first character is *
    - c3 —the second character is number
  - Effect
    - e1— give the information N
    - e2— modify the document
    - e3— give the information M

4.5 Cause-Effect Graphing

- Answer
  - Cause-effect graph



4.5 Cause-Effect Graphing

	1	2	3	4	5	6	7	8
Cause								
c1	1	1	1	0	0	1	0	0
c2	1	1	0	1	0	0	1	0
c3	1	0	0	0	0	1	1	1
11			1	1	0	1	1	0
Action								
e1			0	0	1	0	0	1
e2			0	0	0	1	1	0
e3			1	1	1	0	0	0
Test case			#A	*B	GT	#3	*6	A1

4.5 Cause-Effect Graphing

- Design test cases

ID	Input	Expected results
1	#3	Modify the document
2	#A	Give the information M
3	*6	Modify the document
4	*B	Give the information M
5	A1	Give the information N
6	GT	Give the information M and N

4.5 Cause-Effect Graphing

- Example 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：**

（1）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。

（2） 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。

（3）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。”

4.5 Cause-Effect Graphing

（1）Analysis the specification and find out causes and effects

cause： 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

建立中间结点，表示处理中间状态

11. 该找5角（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. 可找5角（应当找5角并且售货机有零钱找）
14. 钱已付清

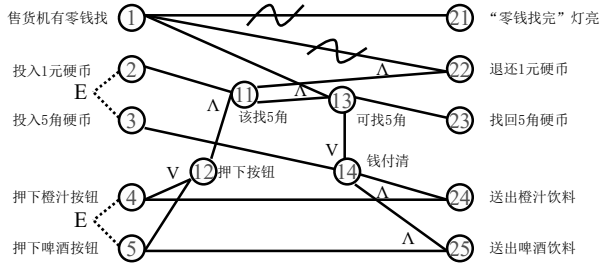
effect： 21. 售货机〖零钱找完〗灯亮
22. 退还1元硬币
23. 找回5角硬币
24. 送出橙汁饮料
25. 送出啤酒饮料

4.5 Cause-Effect Graphing

（2）Draw CEG. Cause nodes on the left, effect nodes on the right

（3）CEG

（4）Because 2 and 3, 4 and 5 can not happen at the same time, so there are E constraints.



4.5 Cause-Effect Graphing

（5）Change to judgment table

序号		1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1	2
条件	①	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	1	1	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	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	1	1	0	0	1	1	0	0	1	1	0	0
	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	0
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	

4.5 Cause-Effect Graphing

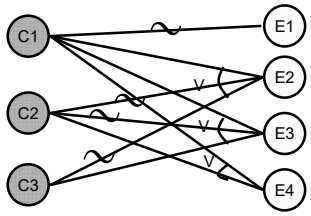
- 练习：
- 某公司对客户有一定的折扣政策，公司软件的一个模块的需求说明书中描述“……当交易额小于等于5万元时折扣为0，当交易额大于5万元时才有折扣，如果交易的客户在三个月内无欠款，则折扣为15%；如果交易用户在三个月内有欠款，若该用户是三年以上的老客户，则折扣为10%；若该客户不是三年以上的老客户，则折扣为5%。”

4.5 Cause-Effect Graphing

- 原因：
  - C1：交易额大于5万元
  - C2：三个月无欠款
  - C3：三年以上老客户
- 结果：
  - E1：无折扣
  - E2：折扣=5%
  - E3：折扣=10%
  - E4：折扣=15%

4.5 Cause-Effect Graphing

- 因果图



4.5 Cause-Effect Graphing

- 决策表

C1	0	1	1	1
C2	—	0	0	1
C3	—	0	1	—
E1	1	0	0	0
E2	0	1	0	0
E3	0	0	1	0
E4	0	0	0	1

4.5 Cause-Effect Graphing

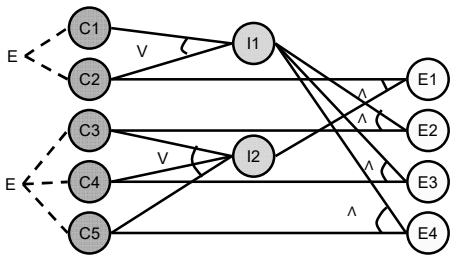
- 练习：
- 有一个处理单价为2元5角钱的盒装饮料的自动售货机软件。若投入2元5角硬币，按下“可乐”、“雪碧”或“绿茶”按钮，相应的饮料就送出来。若投入的是三个1元硬币，在送出饮料的同时退还5角硬币。请用因果图法找出原因和结果，并给出因果图及判定表，最后根据判定表给出测试用例。

4.5 Cause-Effect Graphing

- 原因：
  - C1：投入2元5
  - C2：投入3元
  - C3：按下可乐
  - C4：按下雪碧
  - C5：按下绿茶
- 结果：
  - E1：退还5角
  - E2：送出可乐
  - E3：送出雪碧
  - E4：送出绿茶

4.5 Cause-Effect Graphing

- 因果图



4.5 Cause-Effect Graphing

C1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
C3	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
C4	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	
C5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
E1										×	×		×				
E2													×				
E3											×						
E4										×							
C1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
C3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1
C4	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	
C5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
E1																	
E2					×												
E3			×														
E4		×															

4.5 Cause-Effect Graphing

•测试用例

编号	输入值	预期结果
1	投入3元按下绿茶	找5角并送出绿茶
2	投入3元按下雪碧	找5角并送出雪碧
3	投入3元按下可乐	找5角并送出可乐
4	投入2元5按下绿茶	送出绿茶
5	投入2元5按下雪碧	送出雪碧
6	投入2元5按下可乐	送出可乐

4.5 Cause-Effect Graphing

•Disadvantages of CEG test case design:

- Sometimes, it is difficult to find out the cause-effect relation between cause of input conditions and results from specification.
- Usually the causal relationship is very large, resulting in the use of CEG test obtained an amazing amount of test cases. It will bring a heavy burden to software testing.

•Solution

- Use orthogonal test to design test case.

4.6 Orthogonal Array Testing

- 引例:
- 微软Power Point 程序的打印测试，需要考虑4个因素，每个因素又有多个选项。
  - 打印范围：分全部、当前幻灯片、给定范围。
  - 打印内容：分幻灯片、讲义、备注页、大纲视图。
  - 打印颜色/灰度：分彩色、灰度、黑白
  - 打印效果：分幻灯片加框和幻灯片不加框

4.6 Orthogonal Array Testing

•Orthogonal array testing is a systematic, statistical way of testing.

•Orthogonal arrays could be applied in

- •user interface testing,
- •system testing,
- •regression testing,
- •configuration testing ,
- •performance testing.

4.6 Orthogonal Array Testing

•The type of experimental design methods

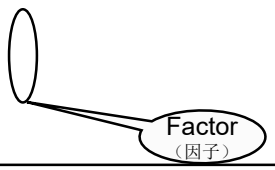
•Comprehensive test

- •Single factor test
- •Orthogonal test
  - •OTDM（Orthogonal Test Design Method）
  - •According to Galois Theory

4.6 Orthogonal Array Testing

•Example：

- In order to improve a certain chemical products conversion rate, three factors which may affect the conversion rate are selected , Reaction temperature（A）, Reaction time（B）, alkali quantity（C）, and the experiment range of them show below:
  - A: 80°C ~ 90°C
  - B: 90 min ~ 150 min
  - C: 5% ~ 7%



4.6 Orthogonal Array Testing

- Test goal:
  - Find out the conversion rate is affected by A, B, and C
  - Which is main condition? Which is secondary condition?
  - Determine the most appropriate production conditions

4.6 Orthogonal Array Testing

- Experiment with several representative values of A , B and C.
  - A: A1= 80 ℃ , A2= 85 ℃ , A3= 90 ℃
  - B: B1=90min, B2=120min, B3=150min
  - C: C1=5%, C2=6%, C3=7%
- **Level** *（水平）*: Number of the values

Selected value of A, B，C is called “level”

4.6 Orthogonal Array Testing

- Two-dimensional forms (level and factor)

Factors \ Levels	1	2	3
temperature （A）	80℃	85 ℃	90 ℃
time （B）	90min	120min	150min
alkali quantity (C)	5%	6%	7%

4.6 Orthogonal Array Testing

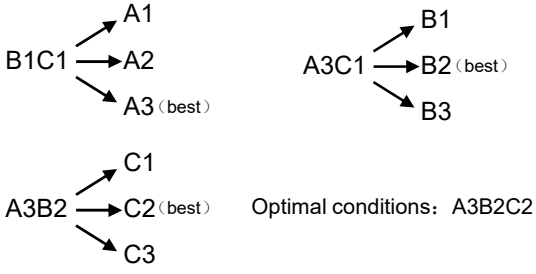
- Comprehensive test
  - Combination of all factors and levels
  - A1B1C1, A1B1C2, A1B1C3⋯ ⋯
  - 3^3=27 times experiments are needed
  - Advantages
    - The relationship between factors and levels
    - Achieve good test results.
  - Disadvantages
    - If there are a lot of factors and levels , then the experiment times is unacceptable. For example, factor amount is 6, and every level of them are 5, then experiment times is 5^6=15626

4.6 Orthogonal Array Testing

- Single factor test for many times
  - Fixed a number of other factors in the same circumstances, At a certain stage, a factor which has changed, so as to arrive at an optimal level of the parameters, Finally, the best combination of parameters, which is closest to the actual situation outcome.

4.6 Orthogonal Array Testing

- Single factor test for many times



Optimal conditions：A3B2C2

4.6 Orthogonal Array Testing

- Single factor test for many times
  - Advantage
    - times=level+（factors-1）×（levels-1）=3+2×2=7（No repeat condition）.
    - Less experiment times
    - If there is no "Factor" interaction between the circumstances, the results will be right fundamentally .
    - In most cases, it can be operated easily than the comprehensive test.
  - Disadvantage
    - If there are correlations between factors , larger deviations occur.

4.6 Orthogonal Array Testing

- Orthogonal test design
  - Effective balanced the advantages of comprehensive test method and single factor test.
  - Select a typical, representative test points from a comprehensive points "level values of factors". It can reflect the full circumstances.
  - Through a series of orthogonal design forms to achieve those tables called "orthogonal table."

4.6 Orthogonal Array Testing

- Orthogonal table
  - is the key of orthogonal test design.
  - is form designed by a set of strict rules.
  - is defined by $L_{Runs}(Levels^{Factors})$:
    - Runs: the number of rows in the array.
    - Factors: the number of columns in an array.
    - Levels: the maximum number of values that can be taken on by any single factor.
- Experiment times= $\sum$ (level amount of every factor-1) +1

4.6 Orthogonal Array Testing

- Orthogonal test - L_9 (3^4) orthogonal array

Row NO	Column No			
	1	2	3	4
	Level			
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	2
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

4.6 Orthogonal Array Testing

- Common orthogonal table:
  - $L_8(2^7)$, $L_9(3^4)$, $L_{16}(4^5)$, $L_{49}(7^8)$, $L_8(4^12^4)$, $L_{18}(3^66^1)$, $L_{16}(2^13^7)$,
- $L_8(2^7)$
- $L_8(4^12^4)$
- Orthogonal table can search or use tools to generate.

4.6 Orthogonal Array Testing

- Steps of orthogonal test case design
  - Extract functional description, and construct the factors state table
    - Divided the functions of requirements, According to software specifications Identify the impact of the operation of functional objects and external factors, take them as factors.
    - Take the value of each factor as a state.
    - Select weighted, generate factors analysis table (based on the factors, status, and frequency of testing required to determine the weight) .

4.6 Orthogonal Array Testing

- Steps of orthogonal test case design
  - Select proper orthogonal table based on the factors, levels, the number of rows and experiment times, select the one has Minimum number of rows.
  - Using orthogonal table structure test data sets.

4.6 Orthogonal Array Testing

	A	B	C		Test No.	Level combination	Experiment condition		
	1	2	3	4			temperature（℃）	time（min）	alkali quantity（%）
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

4.6 Orthogonal Array Testing

- Which is the better orthogonal array?
  - We have a system with 5 independent variables (A, B,C, D, and E).
  - Variables A and B each 2 possible values.
  - Variables C and D each have 3 possible values.
  - Variable E has 6 possible values.
  - A. $L_{49}(7^8)$ B. $L_{18}(3^6 6^1)$

4.6 Orthogonal Array Testing

- 练习
- 使用正交试验法，为下面的基本信息查询选取测试用例。
- 其中：
  - 姓 名： 1-不填 0-填写
  - 身份证： 1-不填 0-填写
  - 手机号： 1-不填 0-填写

基本信息查询

姓　　名：	
身 份 证 号 码：	
手 机 号 码：	

4.6 Orthogonal Array Testing

- 参考正交表如下：

L_4 (2^3)

0	0	0
0	1	1
1	0	1
1	1	0

L_5 (2^4)

0	0	0	0
1	0	1	0
1	1	0	0
0	0	1	1
1	1	1	1

4.6 Orthogonal Array Testing

- 答案：
  - 第一步：计算出需要的测试用例个数。
    - n = ∑（每列的水平数-1）+1
    - = (2 - 1) + (2 - 1) + (2 - 1) +1
    - = 3×（2 - 1）+1
    - =4
    - 需要4次实验即可
  - 第二步：根据实验的水平数，和因子个数选择合适的正交表，我们选择第一个。

4.6 Orthogonal Array Testing

- 第三步：根据正交表写出测试用例。
  - 正交表的每一列对应一个因子（变量），每个取值对应的水平值，即变量的取值。

序号	姓名	身份证	手机	预期结果
1	不填	不填	不填	缺少输入信息
2	不填	填	填	缺少姓名
3	填	不填	填	缺少身份证号码
4	填	填	不填	信息正确

4.7 Scenario Testing

- **Scenario testing**
  - It is a software testing activity that uses **scenario** cases, or simply **scenarios**, which are based on a hypothetical story to help a person think through a complex problem or system for a testing environment.

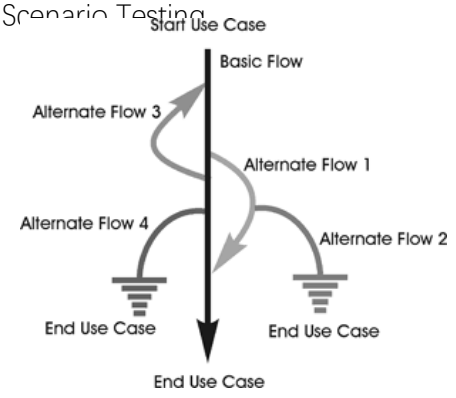
4.7 Scenario Testing

- The ideal scenario has five key characteristics:
  - it is a story
  - motivating
  - credible
  - complex
  - easy to evaluate

4.7 Scenario Testing

- Important part of use case is the flow of events:
  - Basic flow of events
    - should cover what "normally" happens when the use case is performed.
  - Alternate flows events
    - covers behavior of an optional or exceptional character relative to normal behavior, and also variations of the normal behavior.

4.7 Scenario Testing



4.7 Scenario Testing

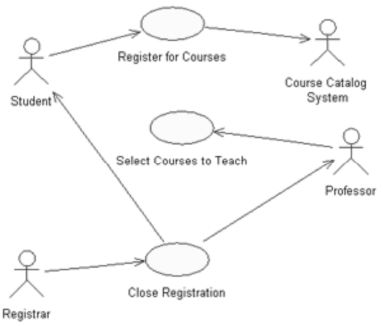
- Scenario for the use case
  - Scenario 1: Basic flow
  - Scenario 2: Basic flow ->Alternate flow 1
  - Scenario 3: Basic flow ->Alternate flow 1 ->Alternate flow 2
  - Scenario 4: Basic flow ->Alternate flow 3
  - Scenario 5: Basic flow ->Alternate flow 3 ->Alternate flow 1
  - Scenario 6: Basic flow ->Alternate flow 3 ->Alternate flow 1 ->Alternate flow 2
  - Scenario 7: Basic flow ->Alternate flow 4
  - Scenario 8: Basic flow ->Alternate flow 3 ->Alternate flow 4

4.7 Scenario Testing

- A three-step process for generating test cases from a fully-detailed use case
  - For each use case, generate a full set of use-case scenarios.
  - For each scenario, identify at least one test case and the conditions that will make it "execute".
  - For each test case, identify the data values with which to test.

4.7 Scenario Testing

- Use case: A university course registration system



4.7 Scenario Testing

- Register For Courses —— Basic flow
  - **1.Logon:** This use case starts when a Student accesses the Wylie University Web site. The system asks for, and the Student enters, the student ID and password.
  - **2.Select 'Create a Schedule' :** The system displays the functions available to the student. The student selects 'Create a Schedule' .

4.7 Scenario Testing

- Register For Courses —— Basic flow
  - **3.Obtain Course Information:** The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the Student.
  - **4.Select Courses:** The Student selects four primary course offerings and two alternate course offerings from the list of available course offerings.

4.7 Scenario Testing

- Register For Courses —— Basic flow
  - **5.Submit Schedule:** The student indicates that the schedule is complete. For each selected course offering on the schedule, the system verifies that the Student has the necessary prerequisites.
  - **6.Display Completed Schedule:** The system displays the schedule containing the selected course offerings for the Student and the confirmation number for the schedule.

4.7 Scenario Testing

- Register For Courses —— Alternate flows
  - **1.Unidentified Student:** In Step 1 of the Basic Flow, Logon, if the system determines that the student ID and/or password is not valid, an error message is displayed.
  - **2.Quit:** The Course Registration System allows the student to quit at any time during the use case. The Student may choose to save a partial schedule before quitting. All courses that are not marked as "enrolled in" are marked as "selected" in the schedule. The schedule is saved in the system. The use case ends.

4.7 Scenario Testing

- Register For Courses —— Alternate flows
  - **3.Unfulfilled Prerequisites, Course Full, or Schedule Conflicts:** In Step 5 of the Basic Flow, Submit Schedule, if the system determines that prerequisites for a selected course are not satisfied, that the course is full, or that there are schedule conflicts, the system will not enroll the student in the course. A message is displayed that the student can select a different course. The use case continues at Step 4, Select Courses, in the basic flow.

4.7 Scenario Testing

- Register For Courses —— Alternate flows
  - **4.Course Catalog System Unavailable:** In Step 3 of the Basic Flow, Obtain Course Information, if the system is down, a message is displayed and the use case ends.
  - **5.Course Registration Closed:** If, when the use case starts, it is determined that registration has been closed, a message is displayed, and the use case ends.

4.7 Scenario Testing

- Step One: Generate Scenarios
  - Read the use-case textual description and identify each combination of basic and alternate flows – the scenarios.

4.7 Scenario Testing

- Scenario for the Register for Courses Use Case

Scenario Name	Starting flow	alternate
Scenario 1-successful registration	Basic flow	
Scenario 2-Unidentified student	Basic flow	A1
Scenario 3-User quits	Basic flow	A2
Scenario 4-Course catalog system unavailable	Basic flow	A4
Scenario 5-Registration closed	Basic flow	A5
Scenario 6-Cannot enroll	Basic flow	A3

4.7 Scenario Testing

- Step Two: Identify Test Cases and create a scenario matrix
  - There should be at least one test case for each scenario, but there will probably be more.
  - For example, if the textual description for an alternate flow is written in a very cursory way, like the description below.
    - A3. Unfulfilled Prerequisites, Course Full, or Schedule Conflicts

4.7 Scenario Testing

- 下面显示的是场景矩阵的通用格式，其中各行代表各个测试用例，各列代表测试用例的信息。
- V (valid): 表示有效的条件，可执行基本流。
- I (invalid): 表示在这种条件下将激活备选流。
- n/a: 表示不适用于测试用例。

4

Scenario Matrix

Test Case ID	Scenario/ Condition	Student ID	Password	Courses selected	Prerequisites fulfilled	Course Open	Schedule Open	Expected Result
RC 1	Scenario 1- successful registration	V	V	V	V	V	V	Schedule and confirmation number displayed
RC 2	Scenario 2- unidentified student	I	N/A	N/A	N/A	N/A	N/A	Error message; back to login screen
RC 3	Scenario 3- valid user quits	V	V	N/A	N/A	N/A	N/A	Login screen appears
RC 4	Scenario 4- course registration system unavailable	V	V	N/A	N/A	N/A	N/A	Error message; back to step 2
RC 5	Scenario 5- registration closed	V	V	N/A	N/A	N/A	N/A	Error message; back to step
RC 6	Scenario 6- cannot enroll -- course full	V	V	V	V	I	V	Error message; back to step 3
RC 7	Scenario 6- cannot enroll -- prerequisite not fulfilled	V	V	V	I	V	V	Error message; back to step 4
RC 8	Scenario 6- cannot enroll -- schedule conflict	V	V	V	V	V	I	Error message; back to step 4

4.7 Scenario Testing

- Step Three: Identify the data values with which to test.

Test Case ID	Scenario/ Condition	Student ID	Password	Courses selected	Prerequisites fulfilled	Course Open	Schedule Open	Expected Result
RC 1	Scenario 1- successful registration	jheumann	abc123	M101 E201 S101	Yes	Yes	Yes	Schedule and confirmation number displayed
RC 2	Scenario 2- unidentified student	Jheuman1	N/A	N/A	N/A	N/A	N/A	Error message; back to login screen
RC 3	Scenario 3- valid user quits	jheumann	abc123	N/A	N/A	N/A	N/A	Login screen appears
RC 4	Scenario 4- course registration system unavailable	jheumann	abc123	N/A	N/A	N/A	N/A	Error message; back to step 2
RC 5	Scenario 5- registration closed	jheumann	abc123	N/A	N/A	N/A	N/A	Error message; back to step 2

4.7 Scenario Testing

- 练习：
- 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：

（1）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。

（2）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。

（3）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。”

4.7 Scenario Testing

- 答案：
  - 基本流:
    - 投入5角硬币，押下〖橙汁〗或〖啤酒〗的按钮，则相应的饮料就送出来。
  - 备选流：
    - 1、〖零钱找完〗的红灯没亮，这时在投入1元硬币并押下按钮后，在送出饮料的同时退还5角硬币。
    - 2、〖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。

4.7 Scenario Testing

- 场景：
  - 场景1：基本流
  - 场景2：备选流1
  - 场景3：备选流2
- 测试用例：

编号	场景	投入硬币	〖橙汁〗或〖啤酒〗按钮	〖零钱找完〗红灯亮	预期输出
1	场景1	5角硬币	按一个按钮	无关	饮料送出
2	场景2	1元硬币	按一个按钮	没亮	饮料送出找5角硬币
3	场景3	1元硬币	按一个按钮	亮	饮料不送出退回1元硬币