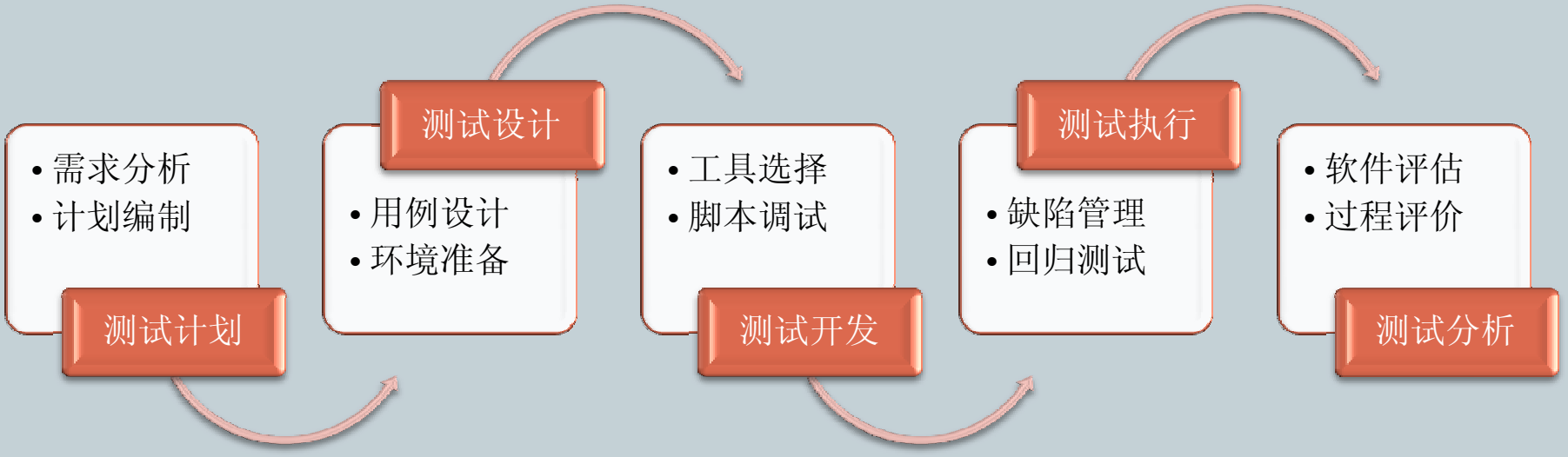
测试技术应用与实践

中国软件评测中心

2010年5月

测试流程图


• 需求分析
• 计划编制
测试计划
测试设计
• 用例设计
• 环境准备
• 工具选择
• 脚本调试
测试开发
测试执行
• 缺陷管理
• 回归测试
• 软件评估
• 过程评价
测试分析

测试设计&测试开发

1. 测试用例框架设计
2. 测试用例设计方法
3. 测试用例的管理
4. 测试工具的使用

测试用例框架设计

- 什么是测试用例？
  - 为达到最佳的测试效果或高效的揭露隐藏的错误而精心设计的少量测试数据。
  - 测试用例是为某个特定测试目标而设计的，它是测试操作过程序列、条件、期望结果及相关数据的一个特定的集合。

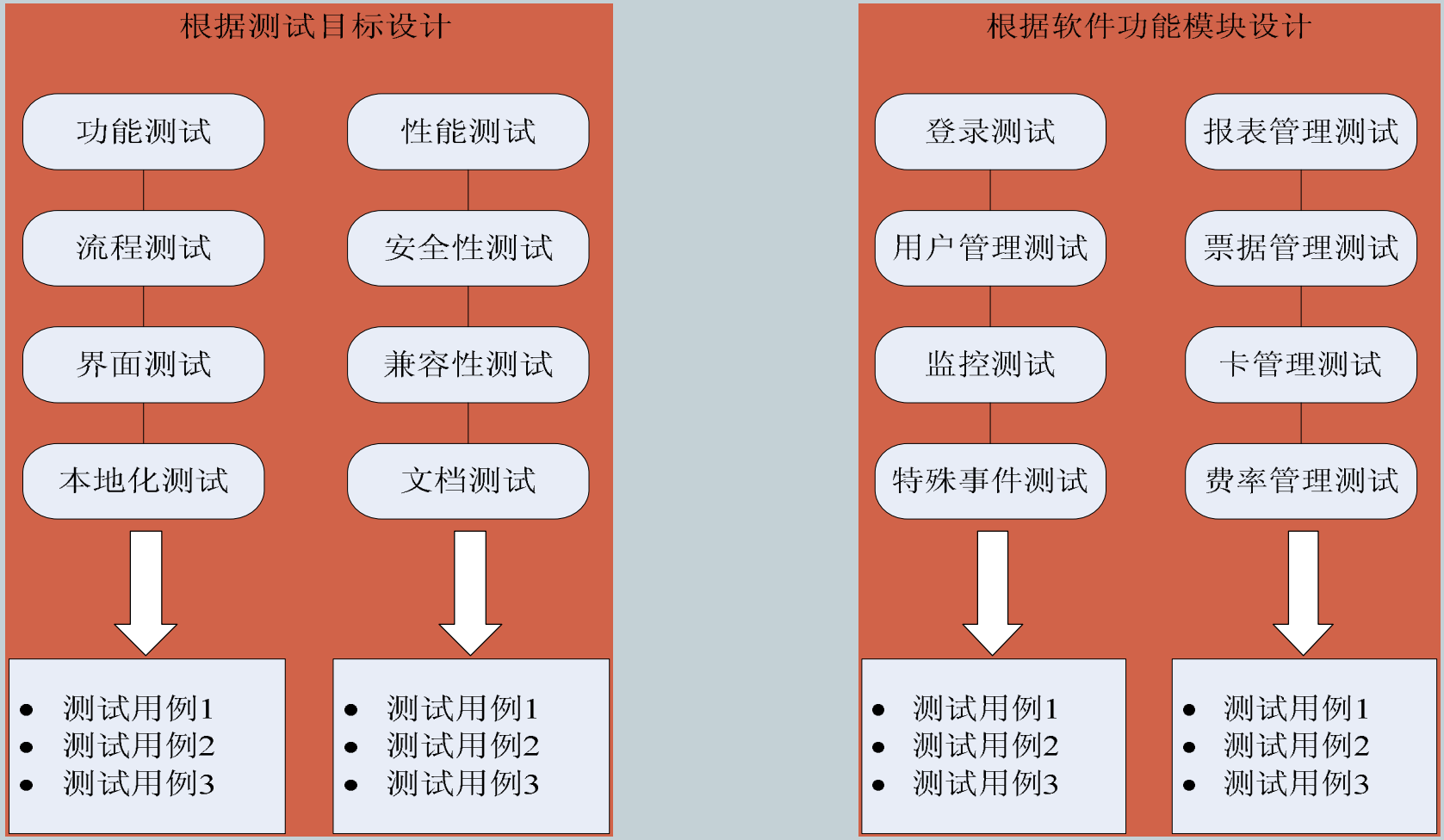
测试用例框架设计

- 为什么需要测试用例
  - 测试用例是”操作指导书”
  - 帮助实施有效的测试，使测试重点突出，目的明确
  - 测试用例的复用性
  - 测试用例是知识积累和知识传递的过程
  - 测试用例体现测试的计划性和组织性
  - 软件质量评估的重要依据

测试用例框架设计

- 测试用例设计考虑因素
  - 需求目标
  - 用户场景
  - 软件设计文档
  - 测试的方法
  - 测试的对象
  - 软件的技术

测试用例框架设计


根据测试目标设计
功能测试
流程测试
界面测试
本地化测试
性能测试
安全性测试
兼容性测试
文档测试
测试用例1
测试用例2
测试用例3
测试用例1
测试用例2
测试用例3
根据软件功能模块设计
登录测试
用户管理测试
监控测试
特殊事件测试
报表管理测试
票据管理测试
卡管理测试
费率管理测试
测试用例1
测试用例2
测试用例3
测试用例1
测试用例2
测试用例3

测试用例框架设计

- 测试用例组成（5W1H1R）
  - 测试目标：Why
  - 测试对象：What
  - 测试环境：Where
  - 测试前提：When
  - 输入数据：Which
  - 操作步骤：How
  - 输出结果：Result

测试设计&测试开发

1. 测试用例框架设计
2. 测试用例设计方法
3. 测试用例的管理
4. 测试工具的使用

测试用例设计方法

- 白盒测试用例设计
- 黑盒测试用例设计
- 性能测试用例设计

测试用例设计方法

- 白盒测试用例设计方法
  - 代码测试方法
    - 语句覆盖
    - 判定覆盖
    - 条件覆盖
    - 判定/条件覆盖
    - 条件组合覆盖
    - 基本路径覆盖

测试用例设计方法

- 白盒测试用例设计方法
  - 程序代码的审查
    - 业务逻辑的审查
    - 算法的效率
    - 代码风格
    - 编程规则

测试用例设计方法

- 黑盒测试用例设计方法
  - 等价类法
  - 边界值法
  - 因果图法
  - 判定表驱动法
  - 场景法
  - 错误推测法

（一）等价类划分

- 等价类划分的办法是把程序的输入域划分成若干部分，然后从每个部分中选取少数代表性数据当作测试用例。

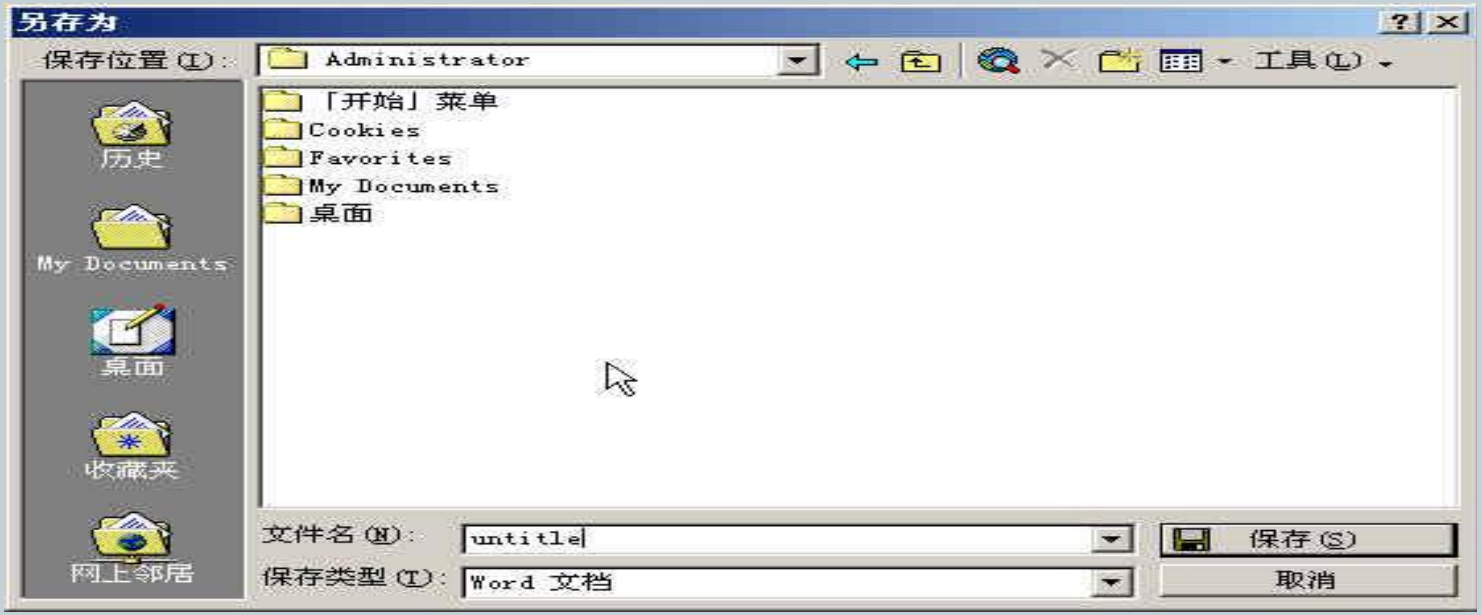
怎样划分等价类

1) 如果输入条件规定了取值的范围或值的个数，则可确定一个有效等价类和两个无效等价类；
2) 如果一个输入条件说明了一个“必须成立”的情况，则可划分一个有效等价类和一个无效等价类；
3) 如果输入条件规定了输入数据的一组可能的值，而且程序是用不同的方式处理每一种值，则可为每一种值划分一个有效等价类，并划分一个无效等价类；
4) 在确立了等价类之后，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
○ ○ ○	○ ○ ○	○ ○ ○
○ ○ ○	○ ○ ○	○ ○ ○

示例：等价区间如何划分？

- Windows文件名可以包含除了、 / ： ·? “”<>和\之外的任意字符。文件名长度是1—255个字符。



示例：三角形的等价划分

题干：根据下面给出的规格说明，利用等价类划分的方法，给出足够的测试用例。“一个程序读入3个整数，把这三个数值看做一个三角形的3条边的长度值。这个程序要打印出信息，说明这个三角形是不等边的、是等腰的、还是等边的。”

Step1：分析三角形的特点

- 3条边分别为A，B，C。满足：A>0，B>0，C>0，且A+B>C，B+C>A，A+C>B；
- 等腰需满足A=B，或B=C，或A=C ；
- 等边需满足A=B，且B=C，且A=C ；

Step2：列出三角形的等价类列表

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

Step3：设计三角形的测试用例

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）	不能构成三角形
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）	

（二）边界值分析

- 边界值分析法是一种补充等价划分的测试用例设计技术，它不是选择等价类的任意元素，而是选择等价类边界的测试用例。

边界值设计遵守的原则

1) 如果输入条件规定了取值范围，应以该范围的边界内及刚刚超范围的边界外的值作为测试用例；
2) 若规定了值的位数，分别以最大位数、最小位数及稍小于位数、稍大于最大位数作为测试用例；
3) 针对每个输出条件使用前面的第1）和2）条原则；
4) 分析规格说明，找出其他的可能边界条件。

示例：程序片段

```
Private Sub Command1_Click()
    If Val(Text1.Text) >= 99 Or Val(Text1.Text) <= -99 Then
        MsgBox ("输入的参数值必须大于-99同时小于99!")
     Else
     If Val(Text2.Text) >= 99 Or Val(Text2.Text) <= -99 Then
         MsgBox ("输入的参数值必须大于-99同时小于99!")
      Else
          Text3.Text = Val(Text1.Text) + Val(Text2.Text)
      End If
      End If
   End Sub
```

示例：边界条件缺陷的程序

1：Rem Create a 10 element integer array
2：Rem Initialize each element to -1
3：Dim data(10) As Integer
4：Dim I As Integer
5：For I=1 TO 10
6：data(i)=-1
7：Next i
8：End

边界问题会在哪儿呢？

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

data(0)=0

问题：X的边界值有哪些？

10<= X <=100

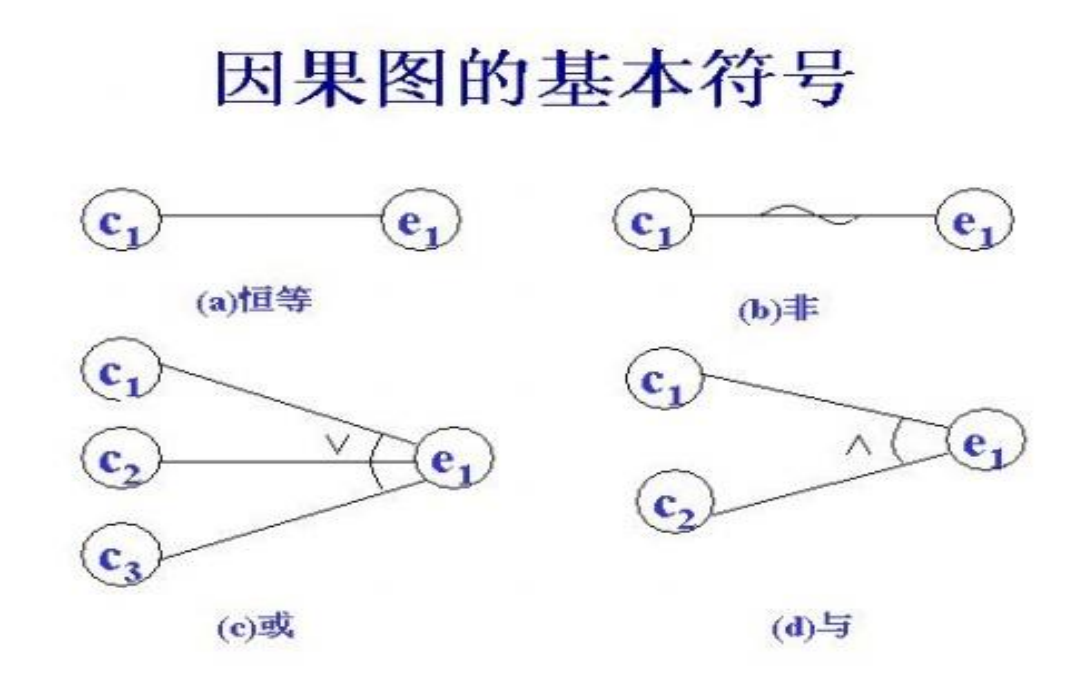
（三）因果图法

- 因果图方法的思路是：从用自然语言书写的程序规格说明的描述中找出因（输入条件）和果（输出或程序状态的改变），通过因果图转换为判定表。

因果图法的设计步骤

1) 分析程序规格说明的描述中，哪些是原因，哪些是结果；
2) 分析程序规格说明的描述中语义的内容，并将其表示成连接各个原因与各个结果的“因果图”；
3) 由于语法或环境的限制，有些原因和结果的组合情况是不可能出现的。为表明这些特定的情况，在因果图上使用若干个特殊的符号标明约束条件；
4) 把因果图转换成判定表；
5) 为判定表中每一列表示的情况设计测试用例。

因果图的基本符号和说明



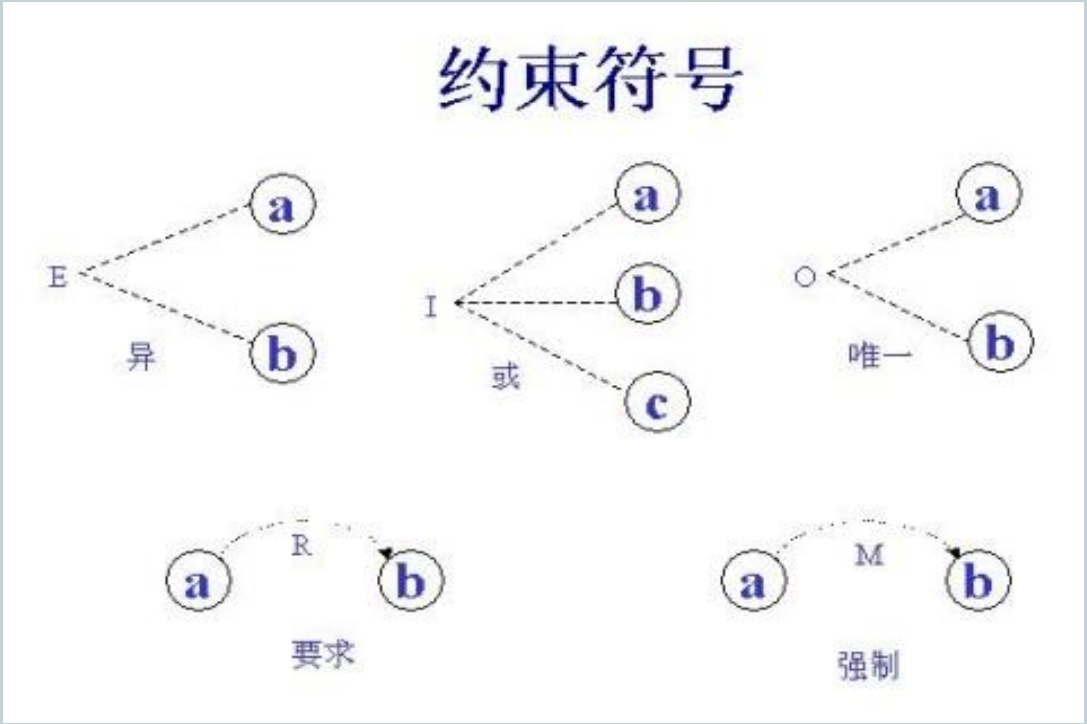
恒等：若原因出现，则结果出现；若原因不出现，则结果也不出现。

非(～)：若原因出现，则结果不出现；若原因不出现，结果反而出现。

或(∨)：若几个原因中有1个出现，则结果出现；若几个原因都不出现，则结果不出现。

与(∧)：若几个原因都出现，结果才出现。若其中有1个原因不出现，则结果不出现。

因果图的约束符号和说明



E（互斥）：表示a、b两个原因不会同时成立，两个中最多有一个可能成立。

I（包含）：表示a、b、c这3个原因中至少有一个必须成立。

O（惟一）：表示a和b当中必须有一个，且仅有一个成立。

R（要求）：表示当a出现时，b必须也出现。a出现时不可能b不出现。

M（屏蔽）：表示当a是1时，b必须是0。而当a为0时，b的值不定。

示例：自动售货机

- 产品说明书：有一个处理单价为1元钱的盒装饮料的自动售货机软件。若投入1元硬币，按下“可乐”、“雪碧”、或“红茶”按钮，相应的饮料就送出来。若投入的是2元硬币，在送出饮料的同时退还1元硬币。

Step1：原因和结果

原因：

投入1元钱
投入2元钱
按可乐按钮
按雪碧按钮
按红茶按钮

结果：

退还1元钱
送出可乐饮料
送出雪碧饮料
送出红茶饮料

Step2：因果图

输入条件（原因）

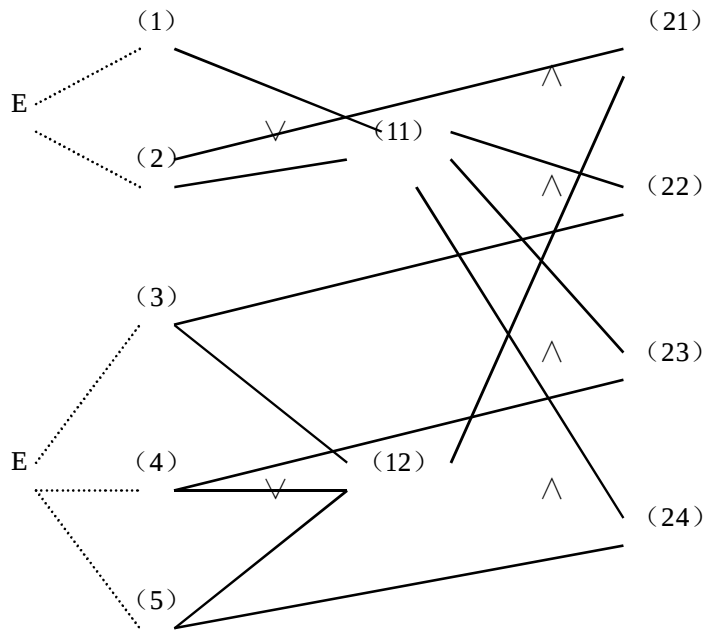
投入1元硬币

投入2元硬币

按“可乐”按钮

按“雪碧”按钮

按“红茶”按钮



输出条件（结果）

退还1元硬币

送出“可乐”饮料

送出“雪碧”饮料

送出“红茶”饮料

Step3：测试用例

			1	2	3	4	5	6	7	8	9	10	11
输入	投入 1 元硬币	（1）	1	1	1	1	0	0	0	0	0	0	0
	投入 2 元硬币	（2）	0	0	0	0	1	1	1	1	0	0	0
	按“可乐”按钮	（3）	1	0	0	0	1	0	0	0	1	0	0
	按“雪碧”按钮	（4）	0	1	0	0	0	1	0	0	0	1	0
	按“红茶”按钮	（5）	0	0	1	0	0	0	1	0	0	0	1
中间结点	已投币	（11）	1	1	1	1	1	1	1	1	0	0	0
	已按钮	（12）	1	1	1	0	1	1	1	0	1	1	1
输出	退还 1 元硬币	（21）	0	0	0	0	1	1	1	0	0	0	0
	送出“可乐”饮料	（22）	1	0	0	0	1	0	0	0	0	0	0
	送出“雪碧”饮料	（23）	0	1	0	0	0	1	0	0	0	0	0
	送出“红茶”饮料	（24）	0	0	1	0	0	0	1	0	0	0	0

（四）判定表驱动法

- 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。它可以把复杂的逻辑关系和多种条件组合的情况表达得较明确。
- 判定表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。

判定表组成

			规则 1	2	3	4	5	6	7	8	9	10	11
输入	投入 1 元硬币	（1）	1	1	1	1	0	0	0	0	0	0	0
	投入 2 元硬币	（2）	0	0	0	0	1	1	1	1	0	0	0
	按“可乐”按钮	（3）	1	0	0	0	1	0	0	0	1	0	0
	按“雪碧”按钮	（4）	0	1	0	0	0	1	0	0	0	1	0
	按“红茶”按钮	（5）	0	0	1	0	0	0	1	0	0	0	1
中间结点	已投币	（11）	1	1	1	1	1	1	1	1	0	0	0
	已按钮	（12）	1	1	1	0	1	1	1	0	1	1	1
输出	退还 1 元硬币	（21）	0	0	0	0	1	1	1	0	0	0	0
	送出“可乐”饮料	（22）	1	0	0	0	1	0	0	0	0	0	0
	送出“雪碧”饮料	（23）	0	1	0	0	0	1	0	0	0	0	0
	送出“红茶”饮料	（24）	0	0	1	0	0	0	1	0	0	0	0

- 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。
- 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。
- 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。
- 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。

判定表建立

- 判定表的建立因该依据软件规格说明，步骤如下：
  - 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2^n种规则。
  - 列出所有的条件桩和动作桩。
  - 填入条件项。
  - 填入动作项。制定初始判定表。
  - 简化、合并相似规则或者相同动作。

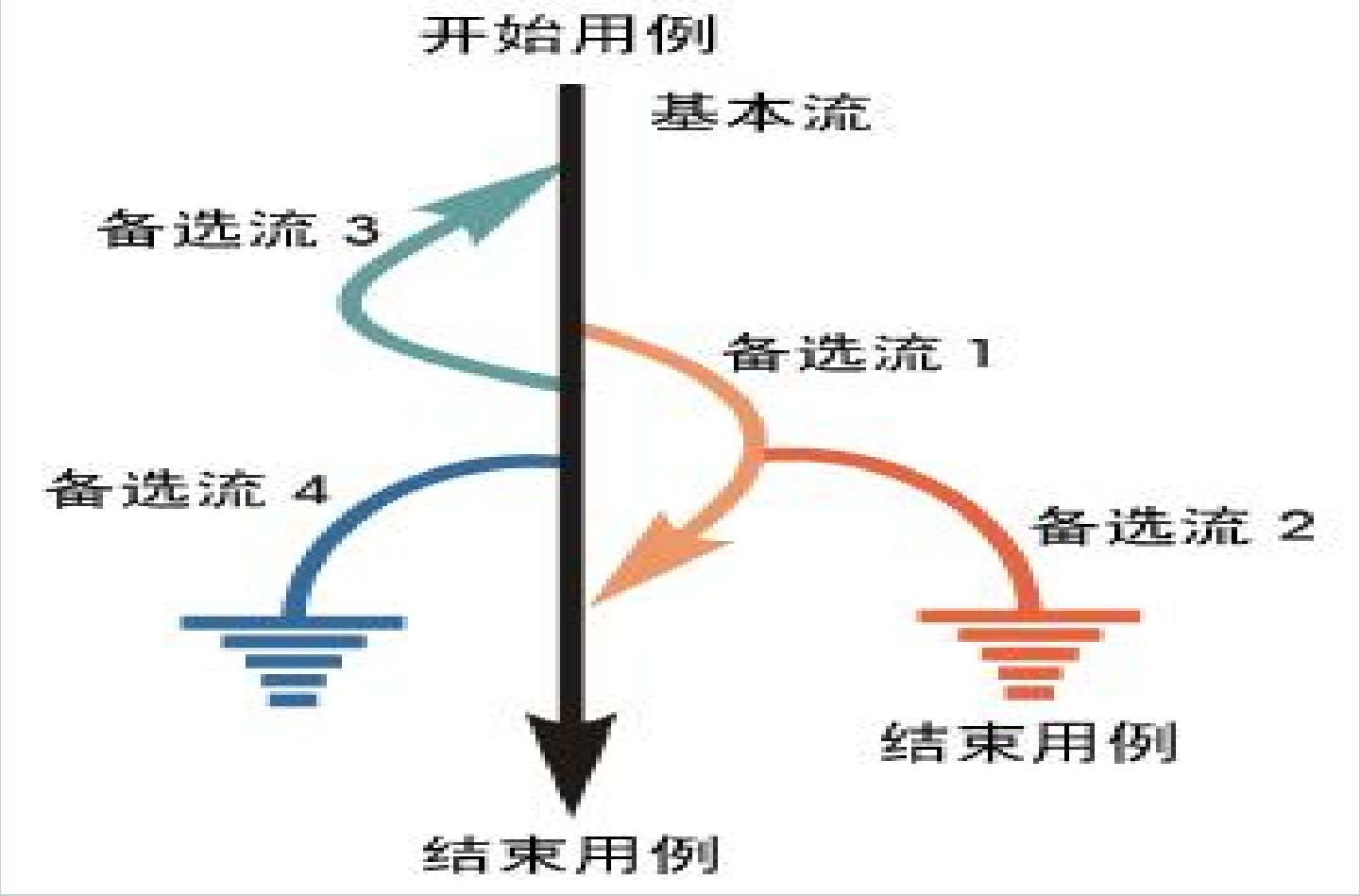
判定表建立

- 适合使用判定表设计测试用例的条件：
  - 规格说明以判定表形式给出，或很容易转换成判定表。
  - 条件的排列顺序不影响执行哪些操作。
  - 规则的排列顺序不影响执行哪些操作。
  - 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。
  - 如果某一规则要执行多个操作，这些操作的执行顺序无关紧要。

（五）场景法

- 用例场景是通过描述流经用例的路径来确定的过程，这个流经过程要从用例开始到结束遍历其中所有基本流和备选流。

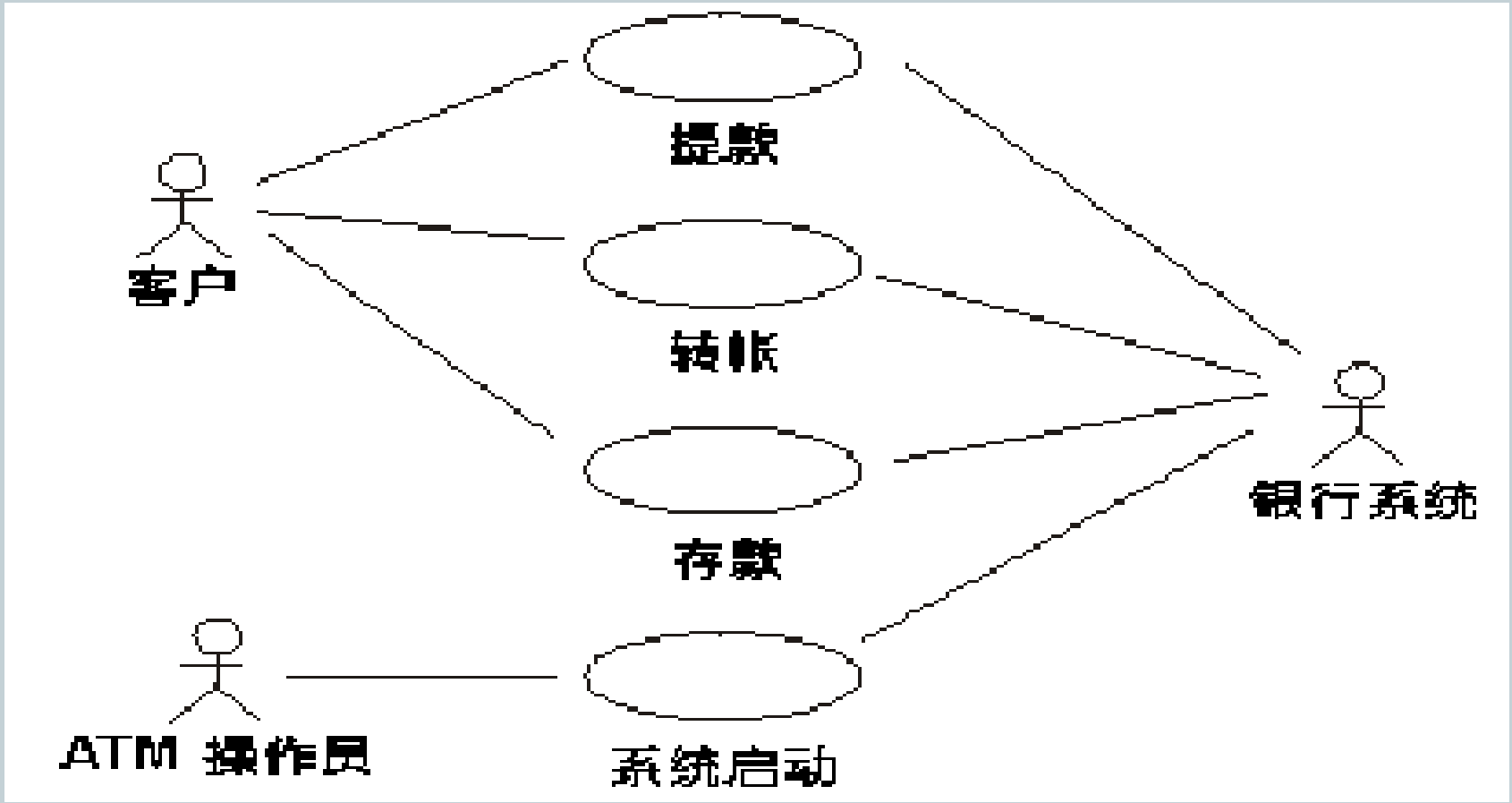
用例场景



用例场景描述

- 场景1 基本流
- 场景2 基本流 备选流1
- 场景3 基本流 备选流1 备选流2
- 场景4 基本流 备选流3
- 场景5 基本流 备选流3 备选流1
- 场景6 基本流 备选流3 备选流1 备选流2
- 场景7 基本流 备选流4
- 场景8 基本流 备选流3 备选流4

示例：ATM（业务模型）



Step1：ATM基本流（一）

- 步骤1：准备提款- 客户将银行卡插入ATM 机的读卡机；
- 步骤2：验证银行卡- ATM 机从银行卡的磁条中读取帐户代码，并检查它是否属于可以接收的银行卡；
- 步骤3：输入PIN - ATM 要求客户输入PIN 码；
- 步骤4：验证帐户代码和PIN - 验证帐户代码和PIN 以确定该帐户是否有效以及所输入的PIN 对该帐户来说是否正确；
- 步骤5：ATM 选项- ATM 显示在本机上可用的各种选项。在此事件流中，银行客户通常选择“提款”；

Step1： ATM基本流（二）

- 步骤6：输入金额- 要从ATM 中提取的金额；
- 步骤7：授权-ATM 通过将卡ID、PIN、金额以及帐户信息作为一笔交易发送给银行系统来启动验证过程；
- 步骤8：出钞- 提供现金；
- 步骤9：收据- 打印收据并提供给客户。ATM 还相应地更新内部记录；
- 步骤10：返回银行卡- 银行卡被返还。

Step2：ATM备选流（一）

备选流 1 – 银行卡无效	在基本流步骤 2 中 – 验证银行卡，如果卡是无效的，则卡被退回，同时会通知相关消息。
备选流 2 – ATM 内没有现金	在基本流步骤 5 中 – ATM 选项，如果 ATM 内没有现金，则“提款”选项将无法使用。
选流 3 – ATM 内现金不足	在基本流步骤 6 中– 输入金额，如果 ATM 机内金额少于请求提取的金额，则将显示一则适当的消息，并且在步骤 6 – 输入金额处重新加入基本流。
选流 4 – PIN 有误	在基本流步骤 4 中– 验证帐户和 PIN，客户有三次机会输入 PIN。 如果 PIN 输入有误，ATM 将显示适当的消息；如果还存在输入机会，则此事件流在步骤 3 – 输入 PIN 处重新加入基本流。 如果最后一次尝试输入的 PIN 码仍然错误，则该卡将被 ATM 机保留，同时 ATM 返回到准备就绪状态，本用例终止。
选流 5 – 帐户不存在	在基本流步骤 4 中 – 验证帐户和 PIN，如果银行系统返回的代码表明找不到该帐户或禁止从该帐户中提款，则 ATM 显示适当的消息并且在步骤 9 – 返回银行卡处重新加入基本流。

Step2：ATM备选流（二）

选流 6 – 帐面金额不足	在基本流步骤 7 – 授权中，银行系统返回代码表明帐户余额少于在基本流步骤 6 – 输入金额内输入的金额，则 ATM 显示适当的消息并且在步骤 6 – 输入金额处重新加入基本流。
选流 7 – 达到每日最大的提款金额	在基本流步骤 7– 授权中，银行系统返回的代码表明包括本提款请求在内，客户已经或将超过在 24 小时内允许提取的最多金额，则 ATM 显示适当的消息并在步骤 6 – 输入金额上重新加入基本流。
选流 x – 记录错误	如果在基本流步骤 10 – 收据中，记录无法更新，则 ATM 进入“安全模式”，在此模式下所有功能都将暂停使用。同时向银行系统发送一条适当的警报信息表明 ATM 已经暂停工作。
选流 y – 退出	客户可随时决定终止交易（退出）。交易终止，银行卡随之退出。
选流 z – “翘起”	ATM 包含大量的传感器，用以监控各种功能，如电源检测器、不同的门和出入口处的测压器以及动作检测器等。在任一时刻，如果某个传感器被激活，则警报信号将发送给警方而且 ATM 进入“安全模式”，在此模式下所有功能都暂停使用，直到采取适当的重启/重新初始化的措施。

Step3：ATM（场景）

场景 1 - 成功的提款	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 - 帐户不存在/帐户类型有误	基本流	备选流 5
场景 7 - 帐户余额不足	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。

Step4：ATM（测试用例矩阵）

TC（测试用例）ID 号	场景/条件	PIN	帐号	输入的金额（或选择的金额）	帐面金额	ATM 内的金额	预期结果
CW1.	场景 1 - 成功的提款	V	V	V	V	V	成功的提款。
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

Step5：ATM（测试用例数据）

TC（测试用例）ID 号	场景/条件	PIN	帐号	输入的金额或选择的金额	帐面金额	ATM 内的金额	预期结果
CW1.	场景 1 – 成功的提款	4987	809 – 498	50.00	500.00	2,000	成功的提款。帐户余额被更新为 450.00
CW2.	场景 2 – ATM 内没有现金	4987	809 – 498	100.00	500.00	0.00	提款选项不可用，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

（六）错误推测法

- 错误推测法就是基于经验和直觉推测程序中所有可能存在的各种错误，从而有针对性地设计测试用例的方法。

功能测试用例设计方法

- 功能测试用例设计策略
  - 首先进行等价类划分，包括输入条件和输出条件的等价划分，将无限测试变成有限测试，这是减少工作量和提高测试效率的最有效方法
  - 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
  - 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。
  - 对于业务流清晰的系统，可以利用功能图法贯穿整个测试案例过程，在案例中综合使用各种测试方法。
  - 可以用错误推测法追加一些测试用例，这需要依靠测试工程师的智慧和经验。
  - 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。

测试用例设计方法

- 性能测试用例设计
  - 80-20原理
  - 任务分布图
  - 交易混合图
  - 用户概况分析

（一）80-20原理

- 测试强度估算

80～20原理：每个工作日中80％的业务在20％的时间内完成。

每年业务量集中在8个月，每个月20个工作日，每个工作日8小时即每天80%的业务在1.6小时完成。

举例：

- 去年全年处理业务约100万笔，其中15％的业务处理中每笔业务需对应用服务器提交7次请求；其中70％的业务处理中每笔业务需对应用服务器提交5次请求；其余15％的业务处理中每笔业务需对应用服务器提交3次请求。根据以往统计结果，每年的业务增量为15％，考虑到今后3年业务发展的需要，测试需按现有业务量的两倍进行。

80-20原理

- 测试强度估算

每年总的请求数为：

(100x15%x7+100x70%x5+100x15%x3)x2=1000万次/年

每天请求数为：1000/160=6.25万次/天

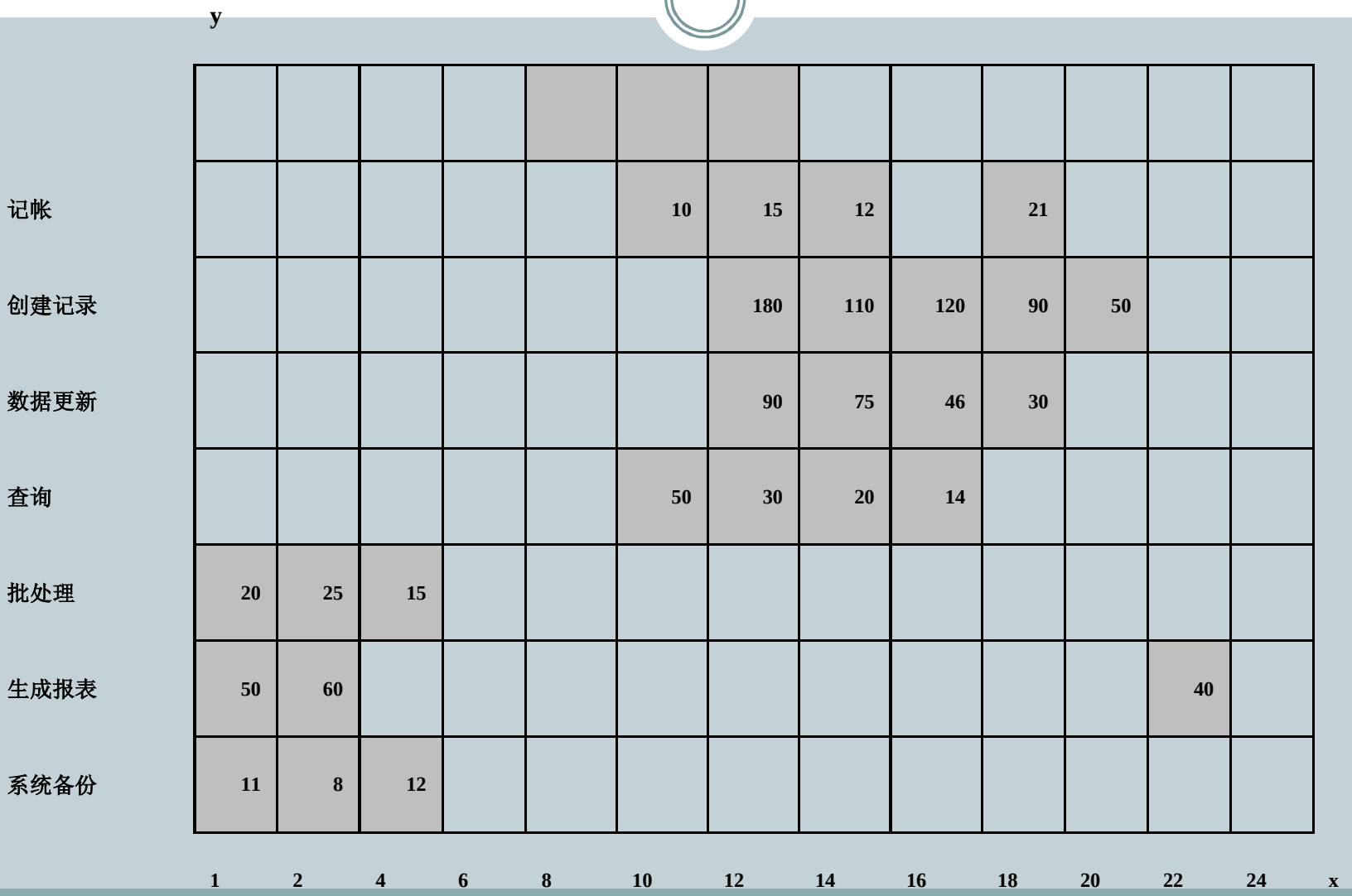
每秒请求数为：(62500x80%)/(8x20%x3600)=8.68次/秒

即服务器处理请求的能力应达到9次/秒

（二）任务分布图

- 任务分布图
  - 有哪些交易任务
  - 在一天的某些特定时刻系统都有哪些主要操作

任务分布图


y
记帐
创建记录
数据更新
查询
批处理
生成报表
系统备份
10 15 12 21
180 110 120 90 50
90 75 46 30
50 30 20 14
20 25 15
50 60 40
11 8 12
1 2 4 6 8 10 12 14 16 18 20 22 24 x

（三）交易混合图

- 交易混合图
  - 高峰期有哪些操作？
  - 中间件操作有多少？数据库操作有多少？
  - 如果任务失败，那么商业风险有多少？

交易混合图

选择的标准：

高吞吐量　　高数据库I/O　　高商业风险

交易名称	日常业务	高峰期业务	Web服务器负载	数据库服务器负载	风险
登陆	70/hr	210/hr	高	低	大
开一个新帐号	10/hr	15/hr	中等	中等	小
生成订单	130/hr	180/hr	中等	中等	中
更新订单	20/hr	30/hr	中等	中等	大
发货	40/hr	90/hr	中等	高	大

（四）用户概况分析

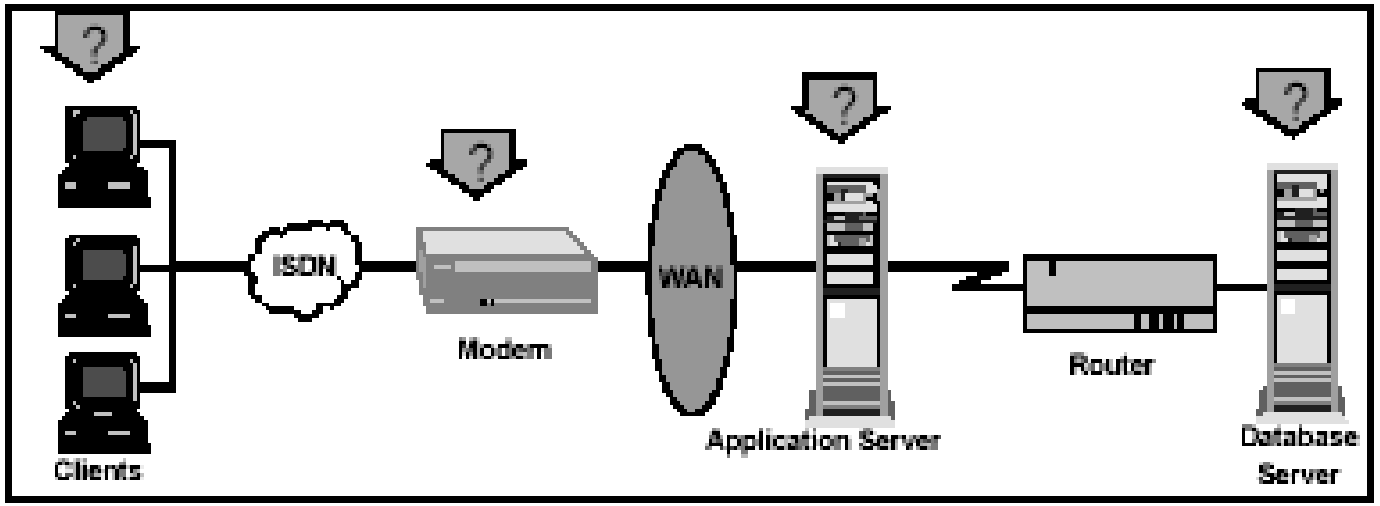
- 用户概况分析
  - 哪些任务是每个用户都要执行的?
  - 针对不同角色的用户，他们的任务是什么?
  - 针对每个用户，不同任务的比例如何?

用户概况分析

	定票部门（170）	飞行部门（50）	经理（30）
输入订单	100	25	
更新订单	50	10	
计算飞行里程		70	5
计算销售			8

性能测试用例设计方法

- 体现在四个方面
  - 环境的模拟
  - 数据的模拟
  - 用户行为的模拟
  - 监控的指标（采用分段排除法）



测试用例设计方法

- 可靠性测试用例设计
  - 屏蔽用户操作错误：考察对用户常见的误操作的提示和屏蔽情况
  - 错误提示的准确性：对用户的错误提示准确程度
  - 错误是否导致系统异常退出：有无操作错误引起系统异常退出的情况
  - 数据可靠性：系统应对输入的数据进行有效性检查，对冗余的数据进行过滤、校验和清洗，保证数据的正确性和可靠性
  - 异常情况的影响：考察数据和系统的受影响程度，若受损，是否提供补救工具，补救的情况如何

系统测试用例设计

- 安全性测试用例设计
  - 用户权限限制：考察对不同的用户权限限制情况。
  - 用户和密码封闭性：对于相应用户及密码进行次数限制。
  - 数字证书认证：系统的证书发放中心能够实现为所有系统用户发放一个标识其身份的数字证书，用于实现本系统用户的身份认证，包括个人证书、服务器证书和设备证书三种类型。
  - 留痕功能：系统是否有操作日志，操作日志记录的操作情况的全面性和准确性，是否包括主要要素如操作员、操作日期、使用模块等。
  - 数据备份与恢复手段：系统是否提供备份及恢复功能，备份手段如何，是否可自定义备份策略。
  - 数据传输安全性：对于有特殊安全要求的数据传输，应对传输的数据进行必要的加密处理。
  - 其他：如跨站脚本攻击、Session和Cookie的管理

测试设计&测试开发

1. 测试用例框架设计
2. 测试用例设计方法
3. 测试用例的管理
4. 测试工具的使用

测试用例的审查

- 测试用例书写标准（ANSI/IEEE 829-1983）
  - 标志符
  - 测试项
  - 测试环境
  - 输入说明
  - 输出说明
  - 测试用例之间的关联

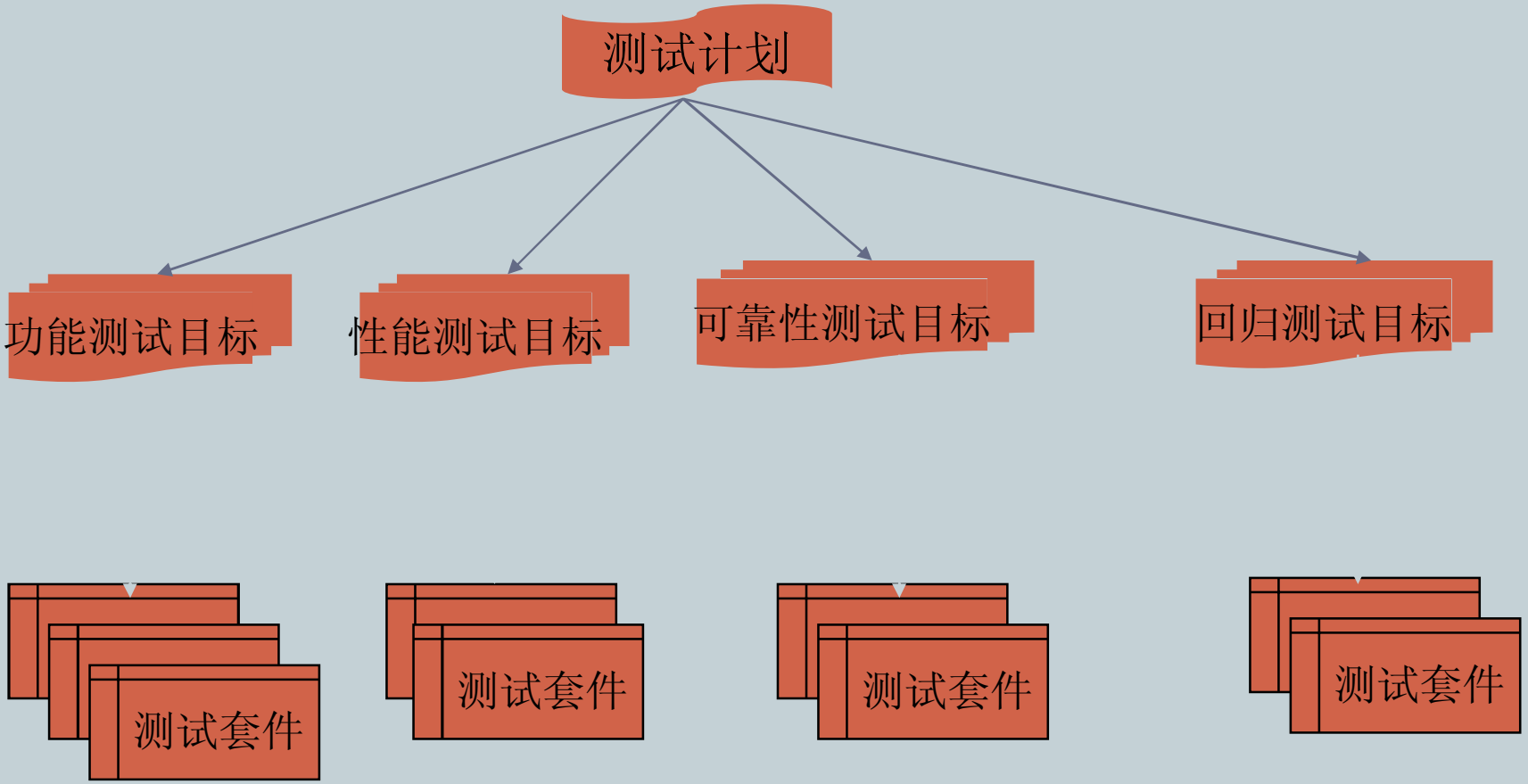
几种用例方式

测试用例的审查

- 测试用例评审要点
  - 总体上，分析测试用例设计思路是否符合业务逻辑、符合技术设计的逻辑、是否和系统架构、组件等建立完全的映射关系
  - 局部上，抓住测试的难点和系统的关键点，从不同角度向测试用例的设计者提问
  - 细节上，检查是否遵守测试用例的编写规范或模板，每项元素是否描述清楚

测试用例评审检查表

测试套件的创建



测试套件的创建

- 测试套件组织方法
  - 按照程序的功能模块组织
  - 按照测试用例的类型组织
  - 按照测试用例的优先级组织

测试设计&测试开发

1. 测试用例框架设计
2. 测试用例设计方法
3. 测试用例的管理
4. 测试工具的使用

测试工具需求

- 测试工具的优势
  - 速度高、效率高
  - 可重复运行
  - 可增强测试的稳定性和可靠性
  - 可提高测试的准确度和精确度

测试工具需求

- 测试工具类别
  - 白盒测试工具
    - 静态测试工具

    语法规则或规范检查、评价代码质量
    - 动态测试

    插入监测代码、分析语句性能、内存使用等
  - 黑盒测试工具
    - 捕捉和回放、自动比较

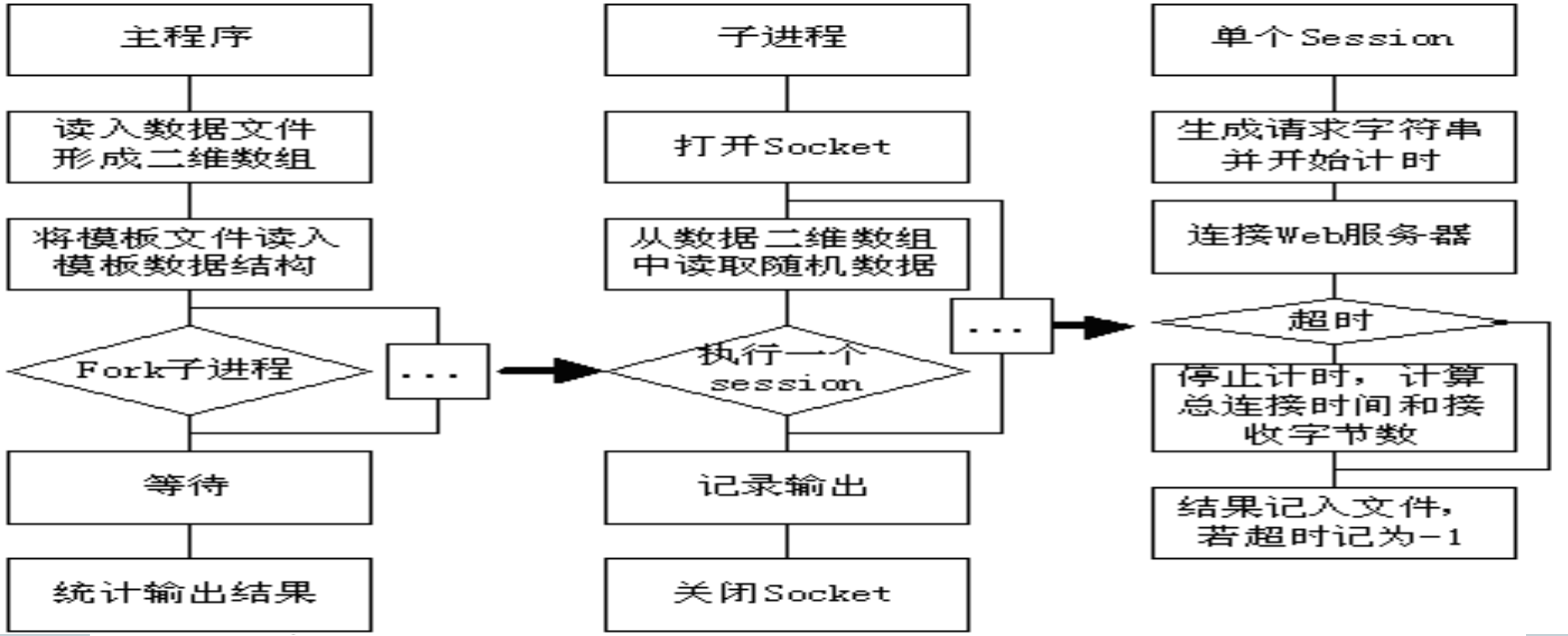
测试工具需求

- 黑盒测试工具的原理

功能测试验证是否应用程序准确完成所有需求?

测试工具的选择

- WEB应用性能测试工具开发
  - 模板文件、数据文件
  - 性能测试程序
  - 结果处理程序



测试工具的选择

- 选择依据
  - 跨平台和环境的兼容性
  - 易使用，针对性强
  - 支持脚本语言
  - 对程序界面中对象的识别能力
  - 测试工具的集成能力
  - 图表分析能力

测试工具的选择

- 主流测试工具厂商
  - HP-Mercury

    www.mercuryinteractive.com
  - IBM-Rational

    www.rational.com
  - Compuware

    www.compuware.com

测试工具的选择

- 测试工具种类
  - 功能测试工具
  - 性能测试工具
  - 白盒测试工具
  - 嵌入式测试工具
  - 测试管理工具
  - 缺陷管理工具
  - 配置管理工具

主要测试工具

测试脚本的开发

- 测试脚本
  - 结构化脚本
  - 共享脚本
  - 数据驱动脚本
  - 关键字驱动脚本

结构	成本	收益	净收益
手工测试（基准）	1	1	0
线性脚本（录制和回放）	8.3	11	2.7
数据驱动脚本	8.4	18	9.6
框架结构	9.8	15	5.2
关键字驱动	11.6	19	7.4