软件测试

软件测试的定义、目的

发现程序错误 - - 直接目标

检查系统是否满足需求 - - 期望目标

改进开发过程 - - 附带目标

开发过程模型

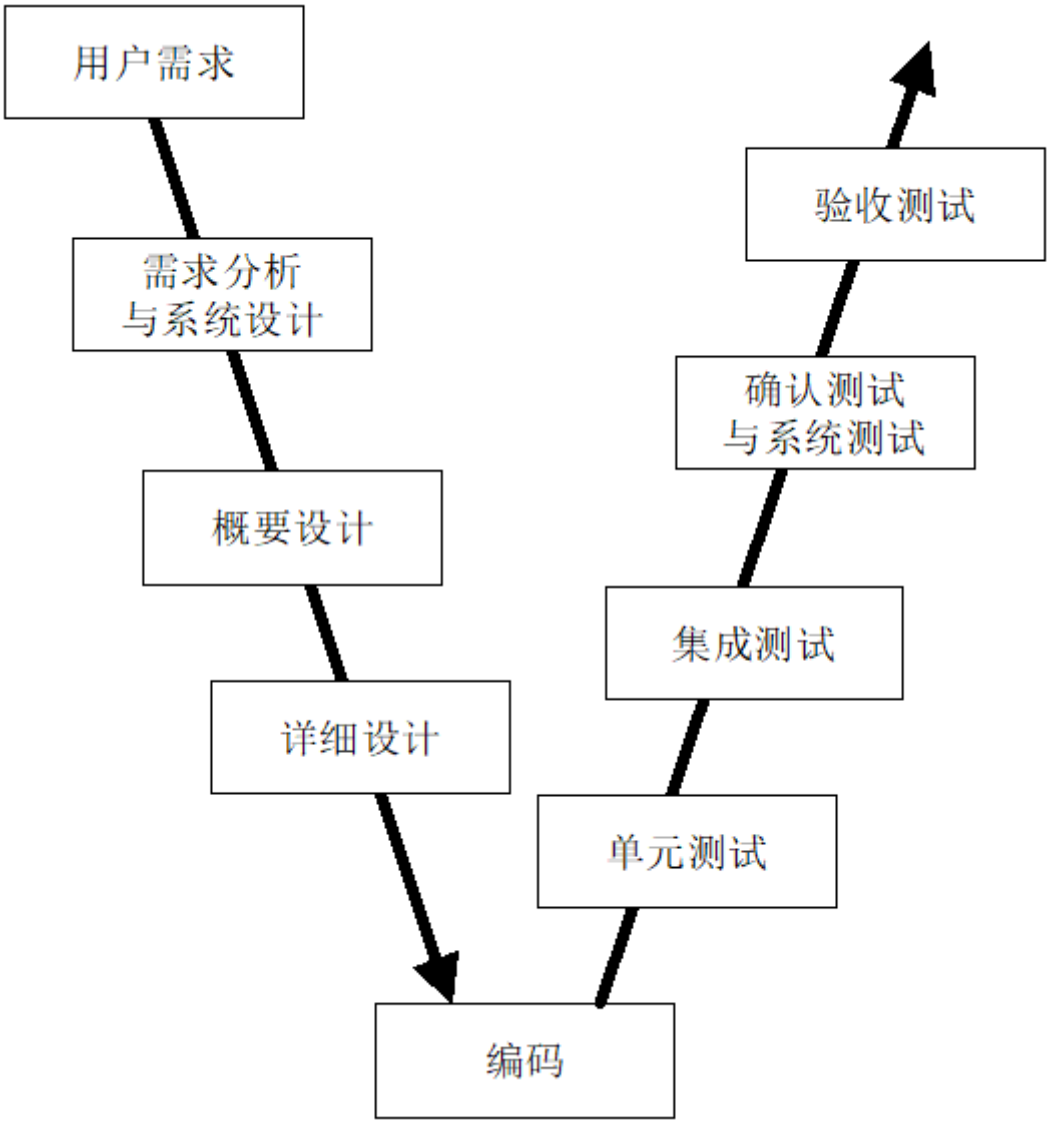
瀑布模型、快速模型、增量模型、螺旋模型

模型名称	技术特点	适用范围
瀑布模型	1、简单，分阶段，阶段间存在因果关系 2、各个阶段完成后都有评审 ，允许反馈，不支持用户参与，要求预先确定需求	需求易于完善定义且不易变更的软件系统
快速原型模型	1、不要求需求预先完备定义，支持用户参与 2、支持需求的渐进式完善和确认，能够适应用户需求的变化	需求难以确定、动态变化的软件系统
增量模型	1、软件产品是被增量式地一块块开发的 2、允许开发活动并行和重叠	技术风险较大、用户需求较为稳定的软件系统
螺旋模型	结合瀑布模型、快速原型模型的思想，引进了风险分析活动	需求难以获取和确定、软件开发风险较大的软件系统

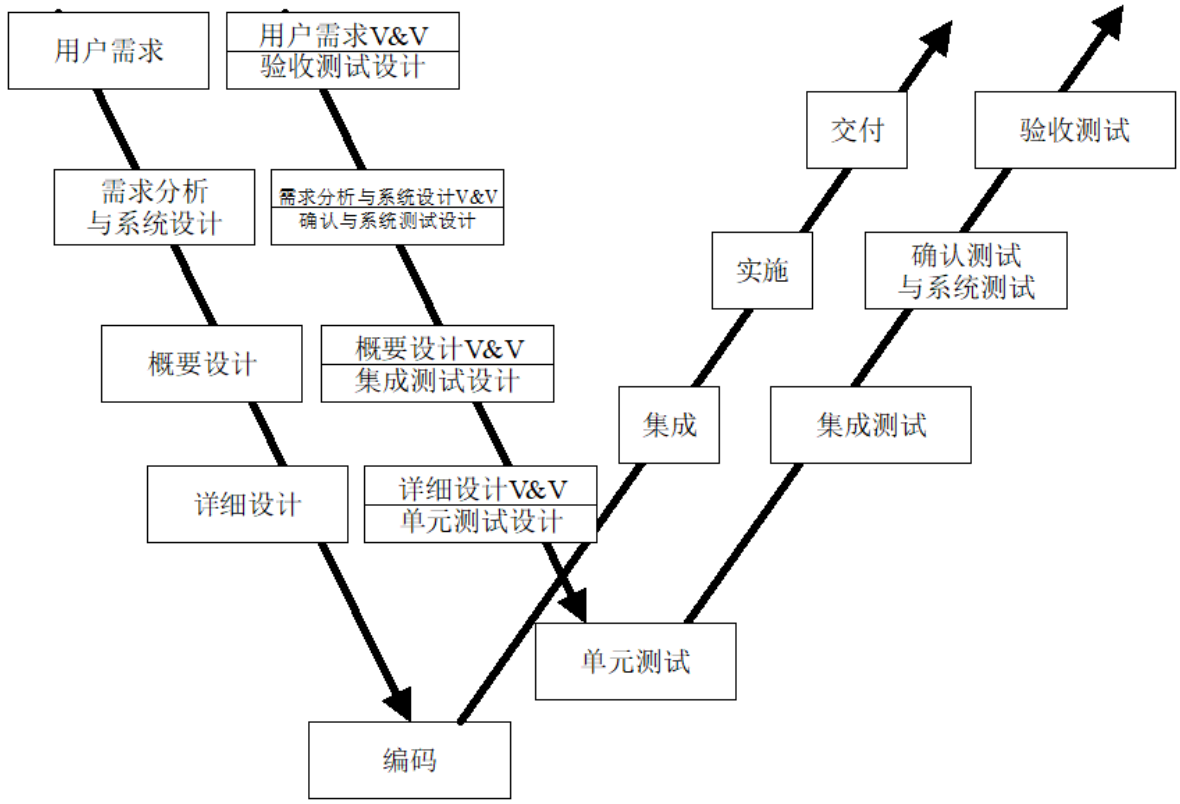
测试过程模型

V模型、W模型、H模型

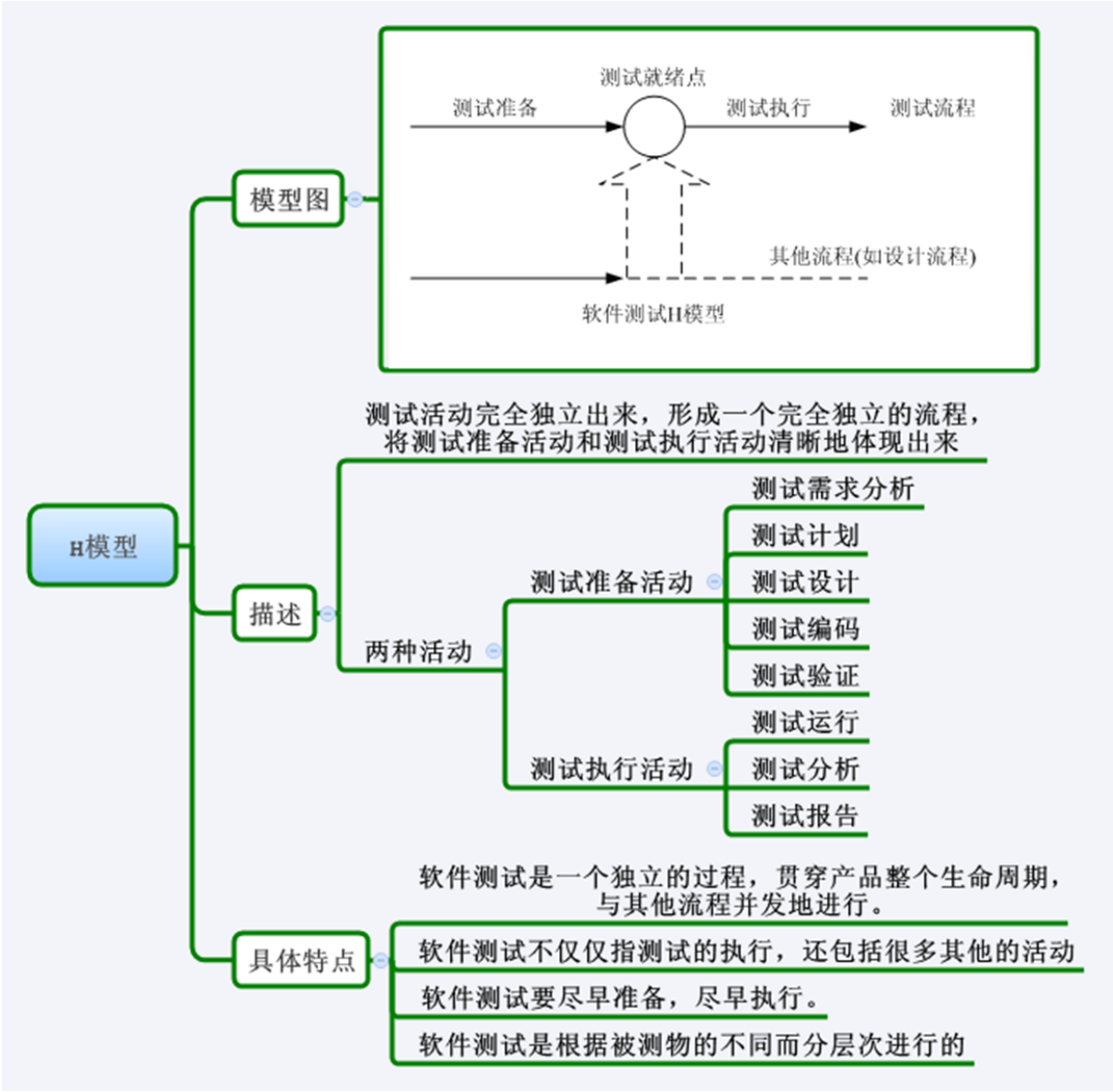
V模型



W模型


用户需求
用户需求V&V
验收测试设计
需求分析
与系统设计
需求分析与系统设计V&V
确认与系统测试设计
概要设计
概要设计V&V
集成测试设计
详细设计
详细设计V&V
单元测试设计
编码
单元测试
集成
集成测试
实施
确认测试
与系统测试
交付
验收测试

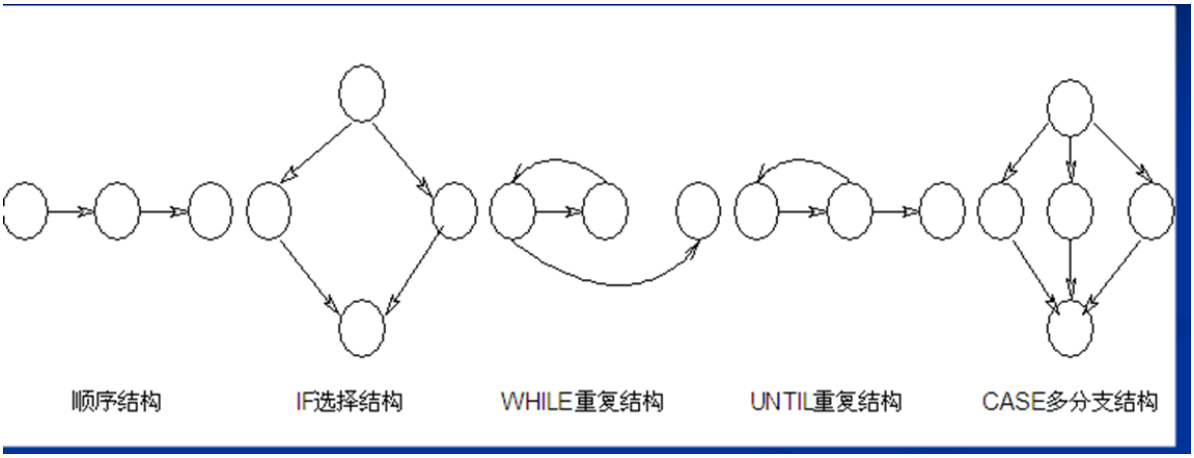
H模型



白盒测试

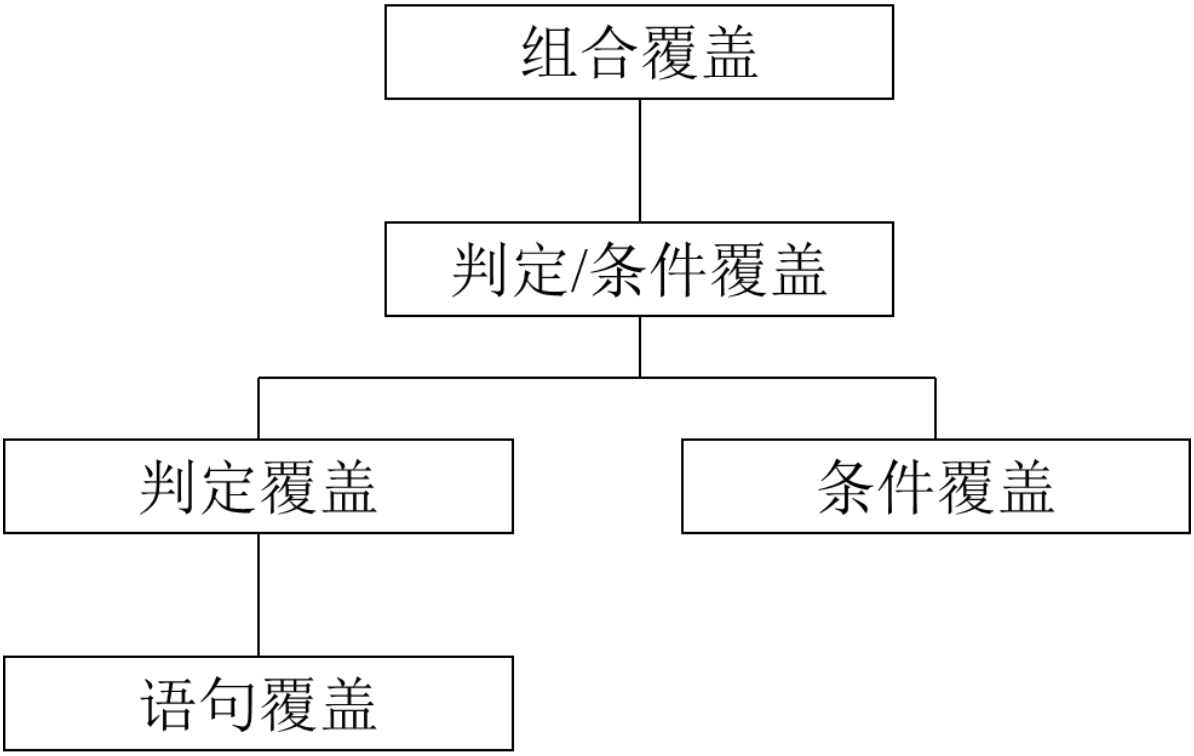
白盒测试法检查程序内部逻辑结构，对所有逻辑路径进行测试，是一种穷举路径的测试方法。

控制流图



逻辑控制法

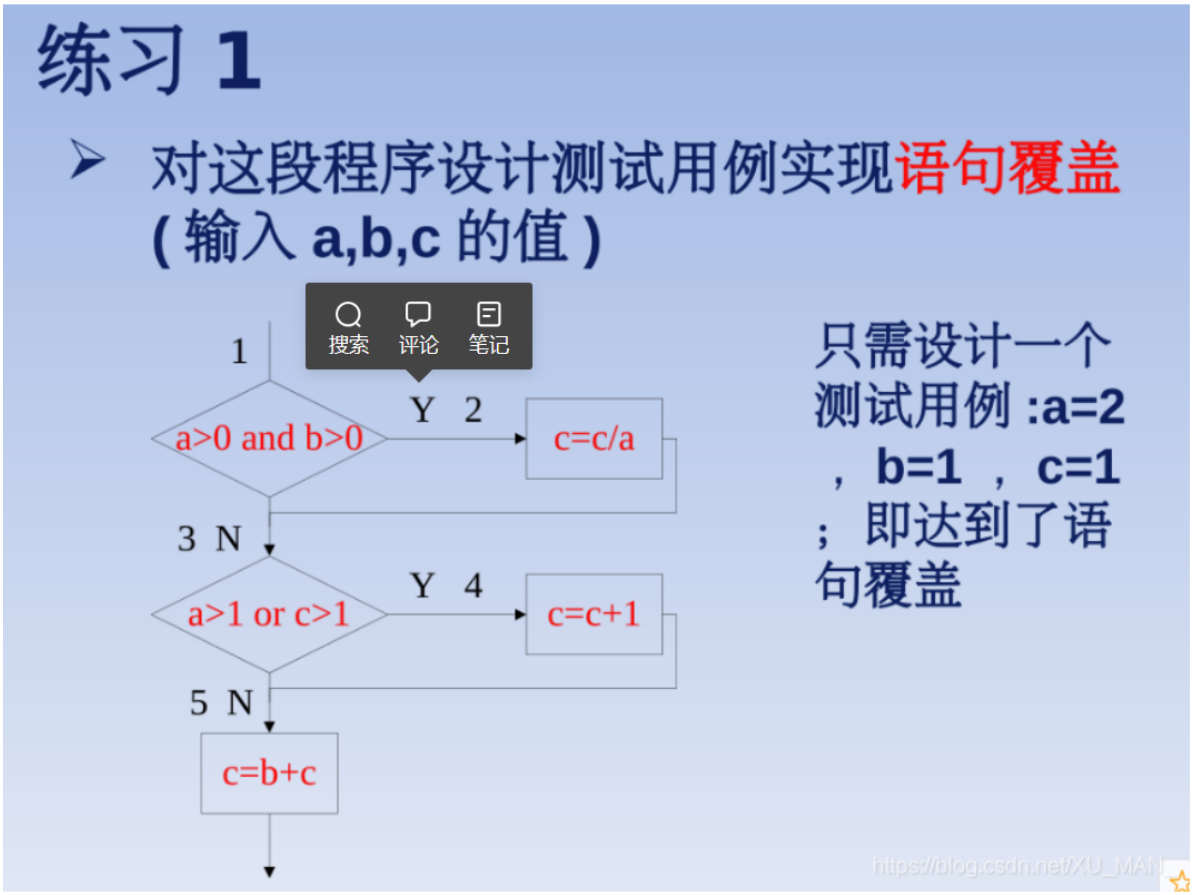
语句覆盖、判定（或分支）覆盖、条件覆盖、判定/条件覆盖、组合覆盖、路径覆盖和修正条件判定覆盖



语句覆盖：所有的可执行语句得到执行;
判定覆盖：每个判定取到真值和假值;
条件覆盖：判定中的每个条件取到真值和假值;
判定-条件覆盖：既满足判定覆盖又满足条件覆盖
组合-条件覆盖：判定中的条件的组合全部覆盖：
路径覆盖：程序的可能执行路径都覆盖。

语句覆盖

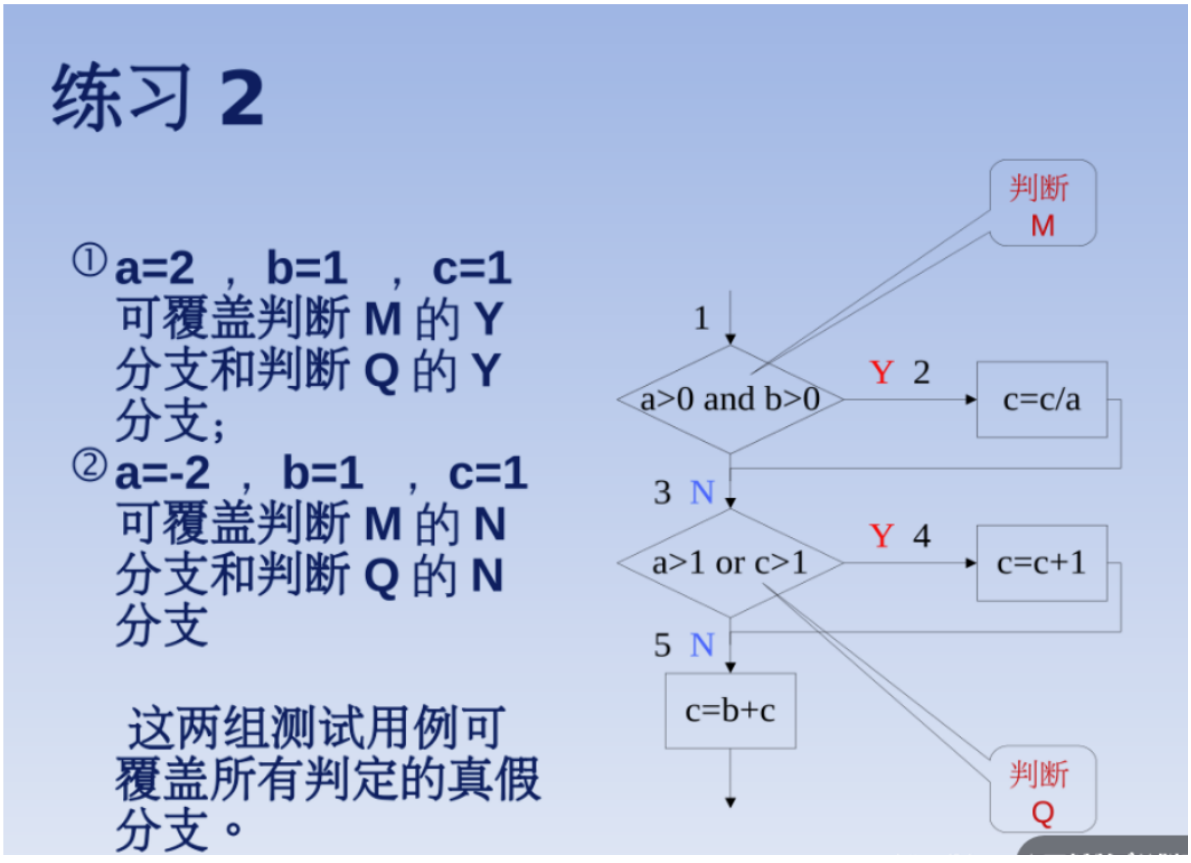
设计测试用例，运行被测程序，**使程序中每个可执行语句至少被执行一次。**只需要一个测试用例就可以测试所有语句



判定（或分支）覆盖

设计测试用例，运行被测程序，**使得程序中每一个判断的取真分支和取假分支至少执行一次。**即判断真假值均曾被 满足。

判定覆盖同时也实现了语句覆盖，看似判定覆盖比语句覆盖更强一些，但仍然无法确定判定内部条件的错误



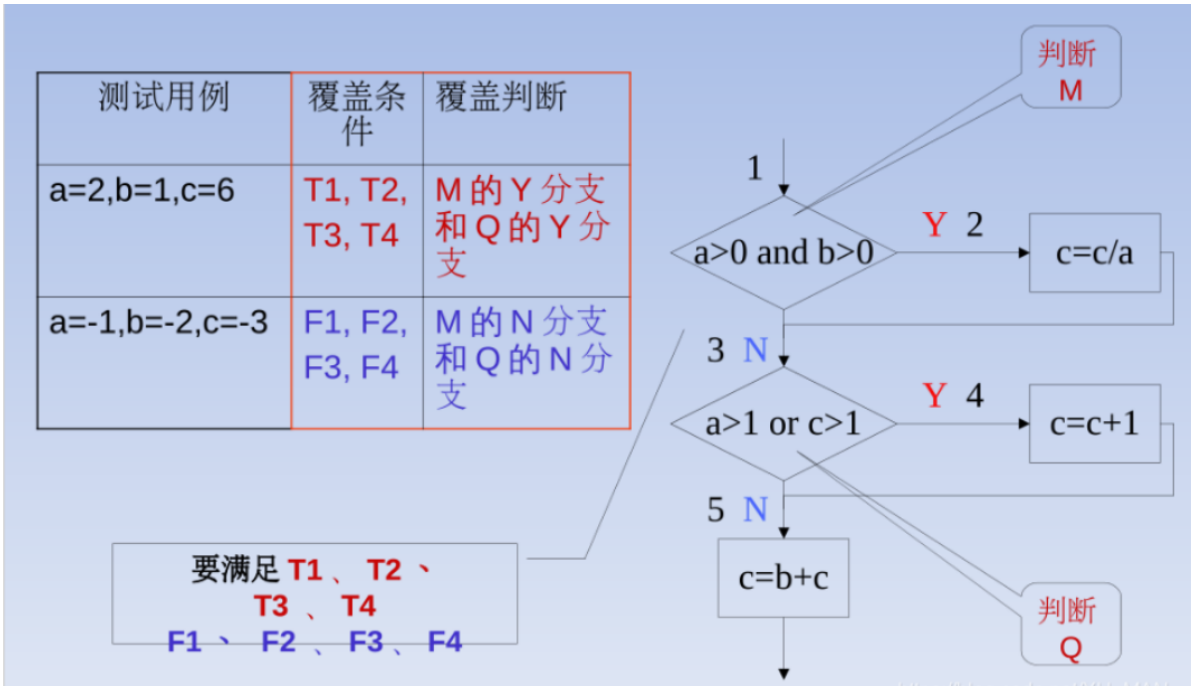
条件覆盖

设计测试用例，**执行被测程序以后要使每个判断中每一个条件的可能取值至少满足一次。**

判定-条件覆盖

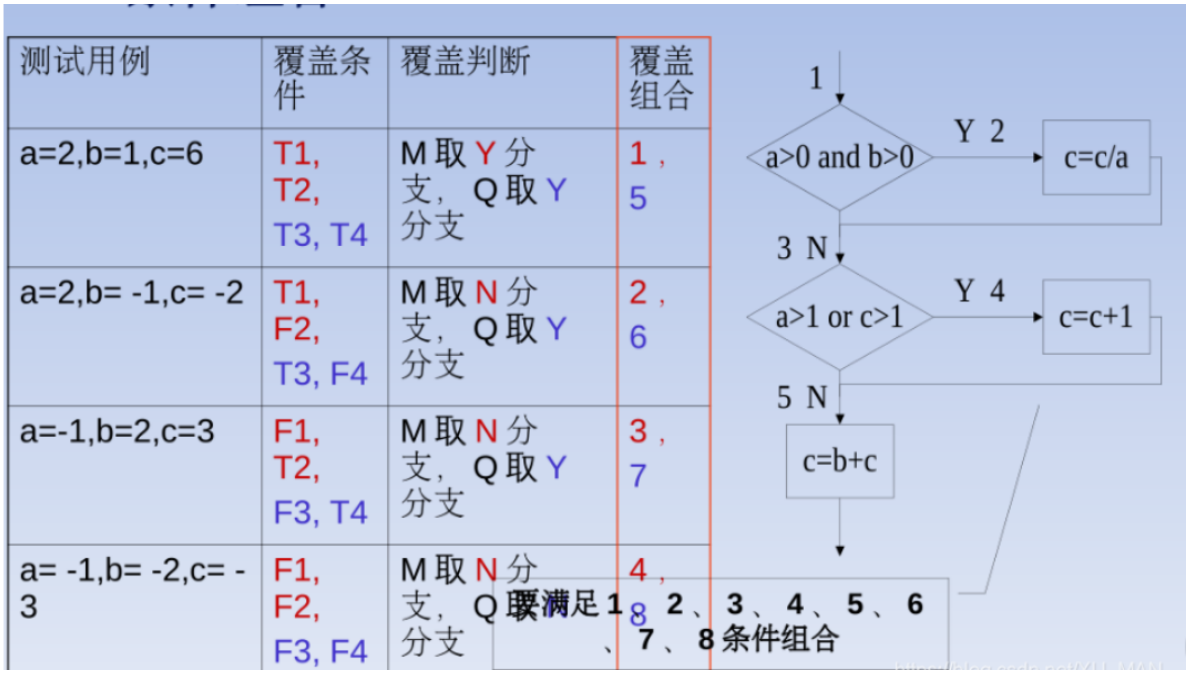
设计足够的测试用例，**使得判断条件中的所有条件可能至少执行一次取值，同时，所有判断的可能结果至少执行一次。**

判定-条件覆盖测试了各个判定中的所有条件的取值，但实际上，编译器在检测含有多个条件的逻辑表达式时，某些情况下的某些条件将会被其他条件覆盖，所以判定-条件也不一定能完全检查出逻辑表达式中的错误



组合-条件覆盖（多条件覆盖）

设计足够的测试用例，**使得每一个判定之中所有可能的条件取值组合至少执行一次**



路径覆盖

设计所有的测试用例，来覆盖程序中所有可能的执行路径

黑盒测试

黑盒测试定义

只是依据软件的规格说明书，运行软件，输入测试数据，根据运行结果，检验该软件的功能是否实现并符合要求、性能等其它特性是否满足用户需要。黑盒测试是一种从用户观点出发，基于规格说明的测试。又叫**功能测试**、**数据驱动测试**。

测试用例定义

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**预期结果**。（设计阶段）

等价类划分法(重点)

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量具有代表性的数据作为测试用例。

等价类划分可分为两种不同的情况：有效等价类和无效等价类

等价类划分法设计测试用例的步骤：

1、分析输入数据形式（数据类型、数据长度、约束条件）

2、划分输入数据（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**】、无效等价类【与有效等价类的定义相反】）

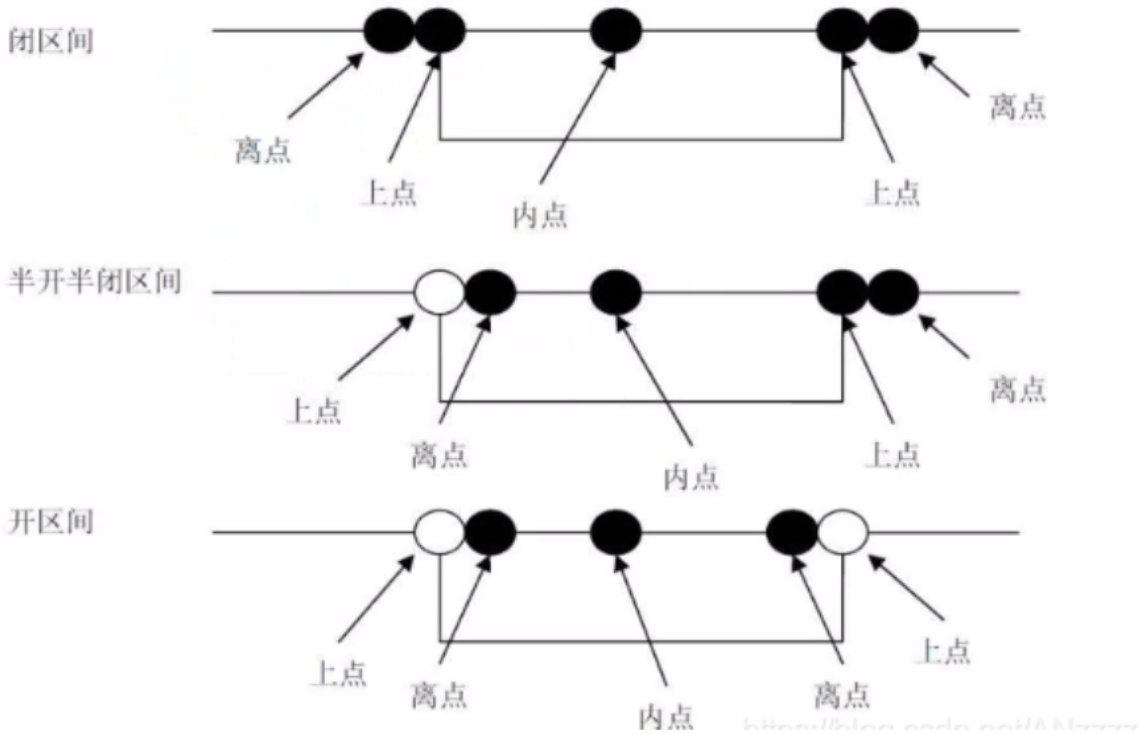
3、设计测试数据

注意：一条用例尽可能覆盖夺得有效等价类

4、设计测试用例

注意：一条用例尽可能覆盖夺得有效等价类

边界值分析法(重点)



边界值分析法设计测试用例的步骤：

1.分析输入数据形式（数据类型、数据长度、约束条件）

2.使用规则划分有效等价类和无效等价类

3.确认区间、上点、离点、内点（即各个边界值）

4.设计测试用例，覆盖有效等价类

注意：一条用例尽可能覆盖夺得有效等价类

5.设计测试用例，覆盖无效等价类

注意：一条用例只能覆盖一个无效等价类

eg：

1、 分析输入数据形式（数据类型、数据长度、约束条件）
商品价格：数字，（0,100），必填，整数
顾客付款：数字，（0,100），必填
找零：数字，[0,100),必填，由50元、10元、5元、1元组成

2、 找出输入数据【整型数据】或数据长度【非整型数据】 的边界值

输入项	区间	上点	离点	边界值
商品价格	（0，100）	0，100	1，99	0，1，99，100
顾客付款	（0，100）	0，100	1，99	0，1，99，100
找零	[0,100）	0，100	-1，99	-1，0，99，100

3.设计测试数据

输入项	测试数据
商品价格	0
	1
	99
	100
顾客付款	0
	1
	99
	100
找零	-1
	0
	1
	2
	4
	5
	6
	9
	10
	11
	49
	50
	51
	99
	100

4、 设计测试用例

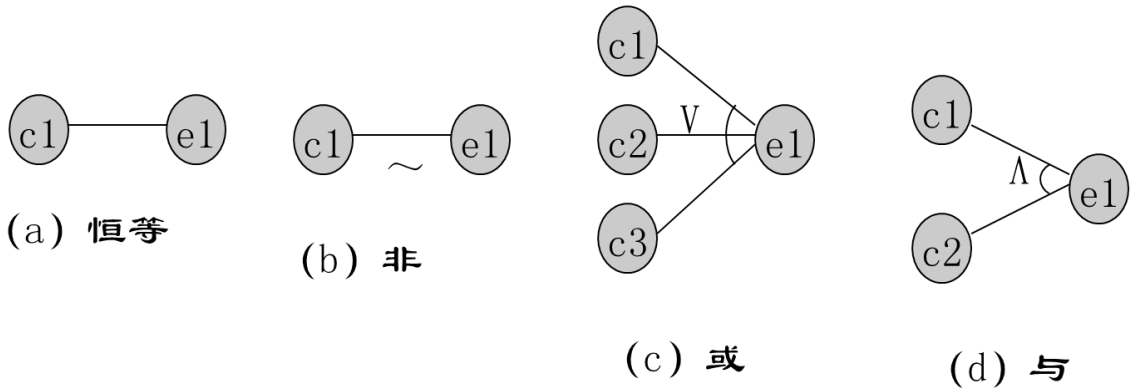
用例编号	测试项	操作步骤	数据	预期结果	实际结果
XZXS001	商品价格	1.进入对应页面 2.输入商品价格为数据列数据 3.输入顾客付款：99 ， 找零：0 4.点击【保存】按钮	1.空 2.1 3.99 4.100	1提示：请输入数据 4.提示：请输入合法数据 2-3.提示：输入正确	
XZXS002	顾客付款	1.进入对应页面 2.输入顾客付款为数据列数据 3.输入商品价格：99 找零：0 4.点击【保存】按钮	1.空 2.1 3.99 4.100	1提示：请输入数据 4.提示：请输入合法数据 2-3.提示：输入正确	
XZXS003	商店找零	1.进入对应页面 2.输入找零为数据列数据 3.输入商品价格：99 顾客付款：99 4.点击【保存】按钮	1.-1 2.0 3.1 4.2 5.4 6.5 7.6 8.9 9.10 10.11 11.49 12.50 13.51 14.99 15.100	1、15提示：请输入数据 2.提示：请输入合法数据 3-5.提示：主要使用面额为1元的货币 6-8.提示：主要使用面额为5元的货币 9-11.提示：主要使用面额为10元的货币 12-14.提示：主要使用面额为50元的货币	

决策表法

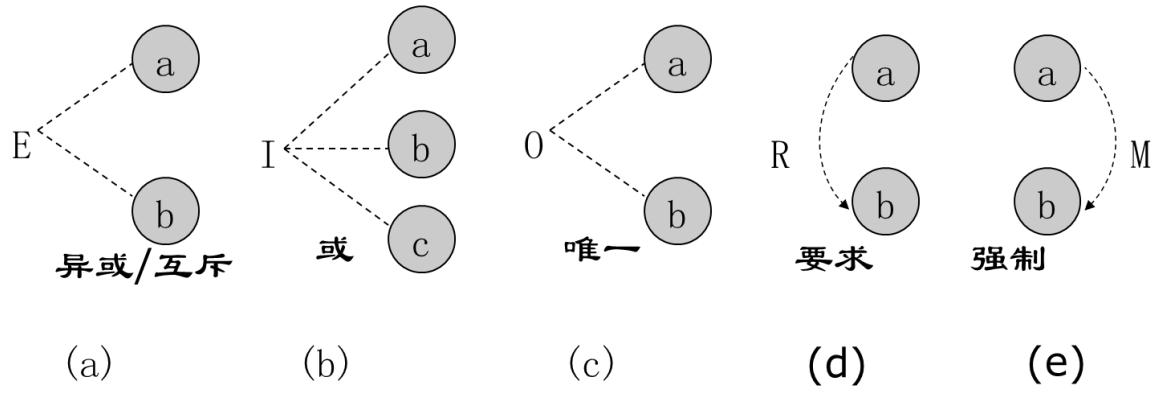
因果图法

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

关系：



约束：



(a) (b) (c) (d) (e)

对于输入条件的约束有4种——

E约束（异或/互斥）：a和b中最多有一个可能为1，即a和b不能同时为1；

I约束（或）：a、b、c中至少有一个必须是1，即a、b、c不能同时为0；

O约束（唯一）：a和b必须有一个且仅有一个为1；

R约束（要求）：a是1时，b必须是1；

对于输出条件的约束只有M约束——

M约束（强制）：若结果a是1，则结果b强制为0。

因果图法设计测试用例步骤：

1、分析所有可能的输入和输出

2、找出输入与输出之间的对应关系

3、画出因果图

4、把因果图转换成判定表

5、设计测试用例

eg:

某程序规定：

1、若输入数据中的第一个字符是“$”或“@”，第二个字符是数字或“%”，则在此情况下进行文件的更新；

2、若第一个字符不是“$”或“@”，则给出信息A；

3、若第二个字符不是数字或“%”，则给出信息B。

问题：

试用因果图法为该程序设计测试用例

1、 分析所有可能的输入和输出

输入

C1：第一个字符是“$”

C2：第一个字符是“@”

C3：第二个字符是数字

C4：第二个字符是“%”

输出

E1：信息A

E2：信息B

E3：文件更新

E4：信息AB

2、 找出输入与输出之间的对应关系

E1：~ (C1∨C2) ∧ (C3∨C4) === Z3∧Z2

E2： (C1∨C2)) ∧ ~ (C3∨C4) === Z1∧Z4

E3：(C1∨C2) ∧ (C3∨C4) === Z1∧Z2

E4：~ (C1∨C2) ∧ ~ (C3∨C4) === Z3∧Z4

中间结果：

Z1：C1∨C2

Z2：C3∨C4

Z3：~Z1

Z4：~Z2

3、 画出因果图

4、 把因果图转换成判定表

	1	2	3	4	5	6	7	8	9	
条件	C1	0	1	0	0	0	1	1	0	0
C2	0	0	1	0	0	0	0	1	1	
C3	0	0	0	1	0	1	0	1	0	
C4	0	0	0	0	1	0	1	0	1	
中间结果	Z1		1	1			1	1	1	1
Z2				1	1	1	1	1	1	
Z3	1			1	1					
Z4	1	1	1							

	1	2	3	4	5	6	7	8	9	
结果	E1				1	1				
E2		1	1							
E3						1	1	1	1	
E4	1									

5、设计测试用例

用例ID	测试输入	预期结果
1	第一个字符不是“$”或“@”，第二个字符不是数字或“%”	输出信息AB
2	第一个字符是“$”，第二个字符不是数字或“%”	输出信息B
3	第一个字符是“@”，第二个字符不是数字或“%”	输出信息B
4	第一个字符不是“$”或“@”，第二个字符是数字	输出信息A
5	第一个字符不是“$”或“@”，第二个字符是“%”	输出信息A
6	第一个字符是“$”，第二个字符是数字	文件更新
7	第一个字符是“$”，第二个字符是“%”	文件更新
8	第一个字符是“@”，第二个字符是数字	文件更新
9	第一个字符是“@”，第二个字符是“%”	文件更新

场景法(重点)

通过用例场景描述用例执行的路径，从用例开始到结束遍历这条路径上所有基本流和备选流。

场景法设计测试用例步骤：

1、根据说明，描述出程序的基本流及各项备选流

2、根据基本流和各项备选流生成不同的场景

3、对每一个场景生成相应的测试用例

4、补充测试数据

eg：

用户进入一个网上商城进行购物，选择商品后加入购物车，进行在线购买，这时需要使用帐号密码登录，登录成功后，付款，输入交易密码，交易成功后，生成订购单，完成整个购物过程。

请使用场景分析法针对该需求设计测试用例，需包含如下四个步骤：

步骤一：根据说明，描述出程序的基本流及各项备选流

基本流：	进入网上商城→选择商品→放入购物车→登录账号→付款→输入付款密码→生成订购凭证
备选流1：	商品缺货
备选流2：	账户不存在，提示账户名或密码错误，请重新输入
备选流3：	登陆密码错误，提示账户名或密码错误，请重新输入
备选流4：	错误超过三次，提示错误超过三次，商城账户被锁定
备选流5：	付款密码错误，提示密码错误，请重新输入
备选流6：	错误超过三次，支付账户被锁定
备选流7：	账户余额不足

步骤二：根据基本流和各项备选流生成不同的场景

场景	场景说明	场景组合
场景1：	购买成功，生成订购凭证	基本流
场景2：	商品缺货	基本流+备选流1
场景3：	账户不存在	基本流+备选流2
场景4：	登陆密码错误	基本流+备选流3
场景5：	错误次数超过三次，商城账户被锁定	基本流+备选流4
场景6：	付款密码错误	基本流+备选流5
场景7：	错误次数超过三次，支付账户被锁定	基本流+备选流5+备选流6
场景8：	账户余额不足	基本流+备选流7

步骤三：对每一个场景生成相应的测试用例

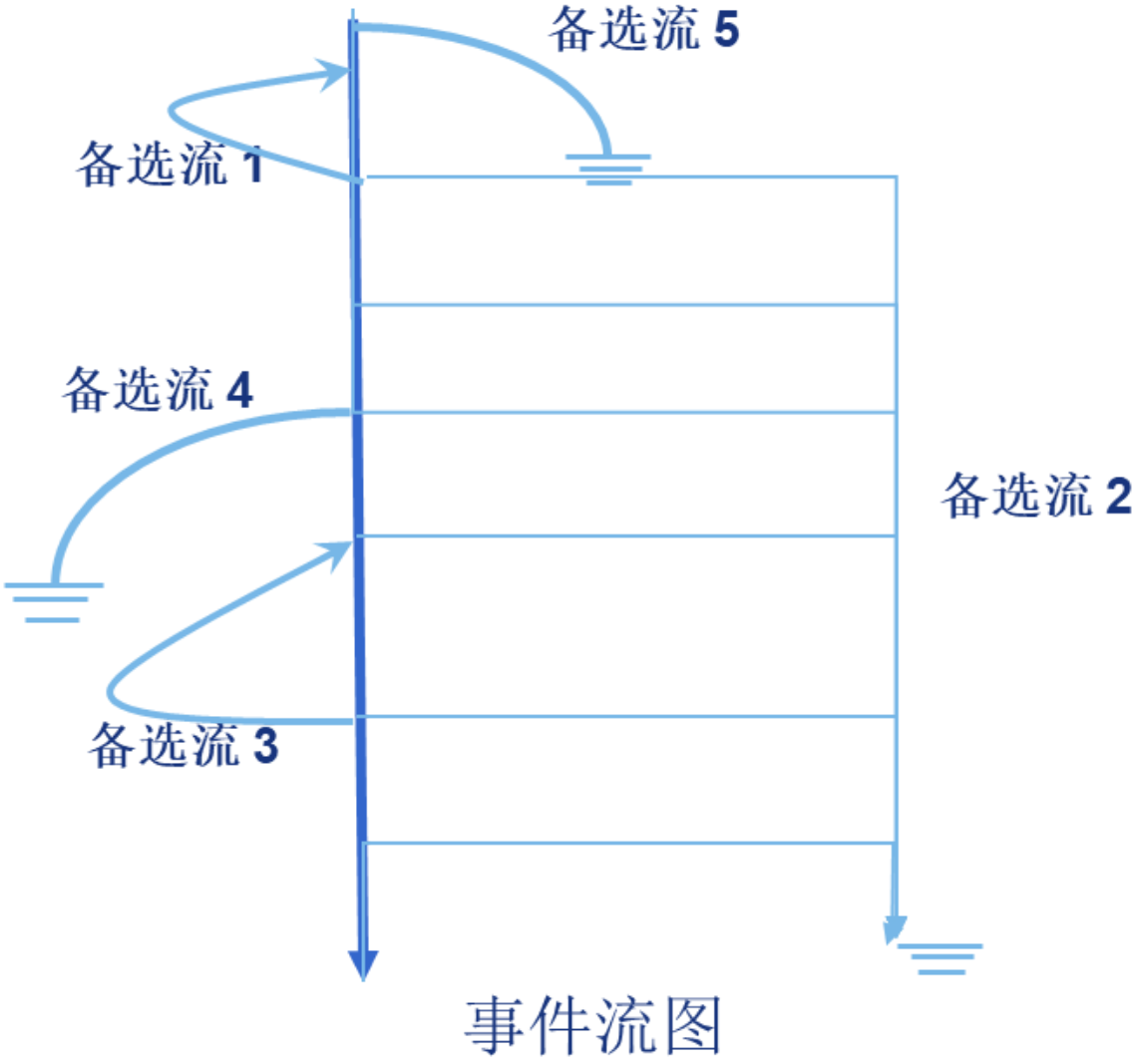
用例ID	场景/条件	购买数量	库存量	登陆账号	登陆密码	支付密码	消费总额	账户余额	预期结果
1	场景1：购买成功，生成订购凭证	V	V	V	V	V	V	V	购买成功，生成订购凭证，用例结束
2	场景2：商品缺货	I	I	n/a	n/a	n/a	n/a	n/a	商品缺货,购买失败，返回基本流步骤2
3	场景3：账户不存在	V	V	I	n/a	n/a	n/a	n/a	提示账户名或密码错误，请重新输入,返回基本流步骤4
4	场景4：登陆密码错误	V	V	V	I	n/a	n/a	n/a	提示账户名或密码错误，请重新输入,返回基本流步骤4
5	场景5：错误次数超过三次，商城账户被锁定	V	V	V	V	I	n/a	n/a	提示错误超过三次，商城账户被锁定，用例结束
6	场景6：付款密码错误	V	V	V	V	I	n/a	n/a	提示密码错误，请重新输入,返回基本流步骤4
7	场景7：错误次数超过三次，支付账户被锁定	V	V	V	V	I	n/a	n/a	提示错误超过三次，支付账户被锁定，用例结束
8	场景8：账户余额不足	V	V	V	V	V	I	I	提示账户余额不足，返回基本流步骤5

步骤四：补充测试数据

假设有效账号为：23041，密码为：1234，交易密码为：5678；黑名单账号为：23001；液体胶库存为0；无尘粉笔库存为2，单价为10元；

用例ID	场景/条件	购买数量	库存量	登陆账号	登陆密码	支付密码	消费总额	账户余额	预期结果
1	场景1：购买成功，生成订购凭证	2	2	23041	1234	5678	20	30	购买成功，生成订购凭证，用例结束
2	场景2：商品缺货	2	0	23041	1234	5678	20	30	商品缺货,购买失败，返回基本流步骤2
3	场景3：账户不存在	2	2	23001	1234	5678	20	30	提示账户名或密码错误，请重新输入,返回基本流步骤4
4	场景4：登陆密码错误	2	2	23041	1111	5678	20	30	提示账户名或密码错误，请重新输入,返回基本流步骤4
5	场景5：错误次数超过三次，商城账户被锁定	2	2	23041	1111	5678	20	30	提示错误超过三次，商城账户被锁定，用例结束
6	场景6：付款密码错误	2	2	23041	1234	5656	20	30	提示密码错误，请重新输入,返回基本流步骤4
7	场景7：错误次数超过三次，支付账户被锁定	2	2	23041	1234	5656	20	30	提示错误超过三次，支付账户被锁定，用例结束
8	场景8：账户余额不足	2	2	23041	1234	5678	20	20	提示账户余额不足，返回基本流步骤5

事件流图



单元测试

集成测试

集成测试方法

静态测试、动态测试。

静态测试：指对概要设计的测试。

动态测试：结合白盒测试、黑盒测试的灰盒测试。

系统测试

是将已经集成好的软件系统，作为整个计算机系统的一个元素，与计算机硬件、外设、某些支持软件、数据和人员等其他系统元素结合在一起，在实际运行（使用）环境下，对计算机系统进行系列的测试活动。

功能测试

根据需求规格说明书和测试需求列表，验证产品功能是否符合产品的需求规格。

不管软件内部的实现逻辑。

功能测试步骤：

1、对每个明确的功能需求进行标号

2、对可能隐含的功能需求进行标号

3、对可能出现的功能异常进行分类分析并标号

4、对前面3个步骤得到的功能需求进行分级

5、对每个功能进行测试分析（此处测试用例设计方法包括规格导出法、等价类划分法、边界值分析法、因果图、判定表、场景法、基于风险的测试、错误推测法）

6、脚本化和自动化

此外还有web功能测试和移动应用功能测试。

性能测试

用来测试软件在集成系统中的运行性能。

性能测试没有完全的标准定义，从广义上来说，压力测试、负载测试、并发测试、大数据量测试、配置测试、可靠性测试、强度测试等等均属于性能测试范畴。

性能测试混合了黑盒测试和白盒测试。

验收测试

在软件产品完成了功能测试和系统测试之后、产品发布之前所进行的软件测试活动。它是技术测试的最后一个阶段，也称为交付测试。

用户群验收测试

组织不同使用层次用户对软件产品进行需求确认的验收测试过程。

测试管理基础

缺陷管理

缺陷：软件中存在的各种问题，都为缺陷，简称bug

自动化测试

自动化软件测试工具分类——

测试管理工具: TestDirector、ClearQuest、Ouality Center

性能测试工具: LoadRunner、JMeter、SilkPerformer

功能测试工具: WinRunner、UnifiedFunctionalTesting、QuickTestProfessional

测试执行案例设计

用例执行结果

1.当用例还尚未被执行时，是No Test未执行状态

2.当执行结果与预期结果相符时，是Pass通过状态

3.当执行结果与预期结果不符时，是Fail失败状态

4.当因为软件有缺陷而妨碍了用例步骤的执行，且该缺陷并不是我们的测试点，则用例是Block阻碍状态。

5.当用例正在执行中，但是需要耗较多时间去观察其结果，是Investigate观察中状态。

缺陷

二八定理：80%的缺陷出现在 20%的模块。

并非所有的缺陷都需要修复

缺陷生命周期—状态

缺陷状态	描述
New	测试中新报告的软件缺陷，等待分派
Open	已确认的缺陷，等待开发人员修改
Fixed	已经被开发人员修改的缺陷，等待测试人员校验
Rejected	不是缺陷或不需要修复
Reopen	没有修复，重新打开返回开发人员
Closed	已经被测试人员确认得到正确修复，可以关闭

缺陷的等级

缺陷严重程度	描述
4--致命	软件无法运行，或者软件的主要功能丧失，或者很大可能性会造成严重不良后果
3--严重	软件的次要功能丧失，或者主要功能在一些特定情况下会出错，比如金额计算等
2--一般	软件在某些情况下会出错，但是造成的后果影响不大
1--轻微	在某些情况下会出错，但是造成的后果影响很小

测试计划

测试计划——测试设计——测试开发——测试评估——测试执行

测试计划制定过程：分析和测试软件需求——定义测试策略——定义测试环境——定义测试管理——编写和审核测试计划