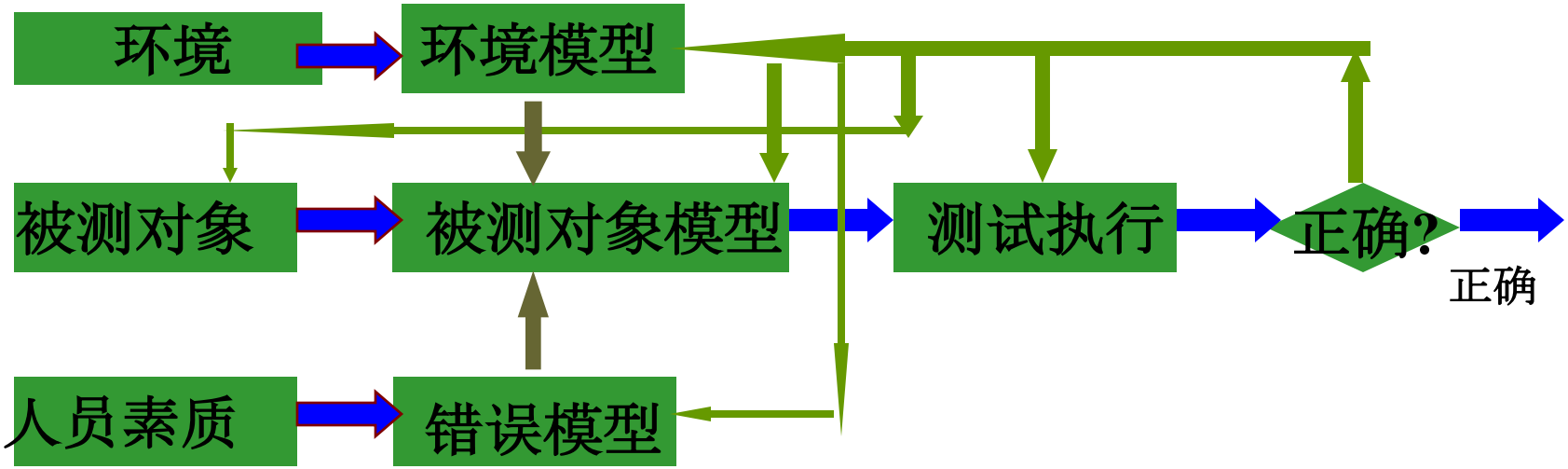
第七章 软件测试技术

1) 测试过程模型

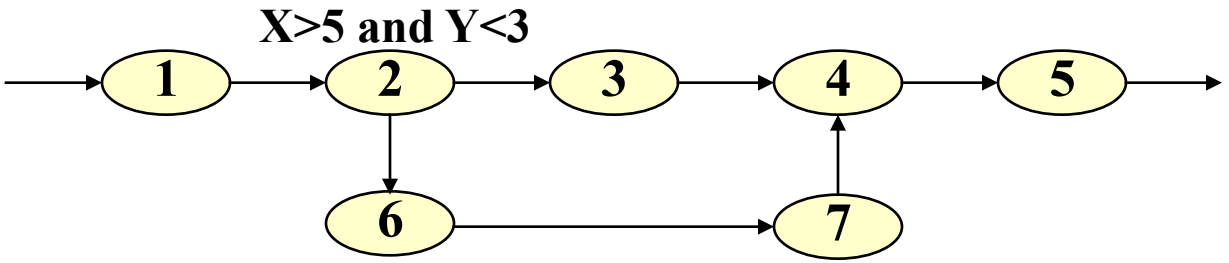


- 软件测试过程所涉及的要素,以及
- 这些要素之间的关系

2）依据程序逻辑结构-白盒测试技术

（1）关于建立被测对象模型

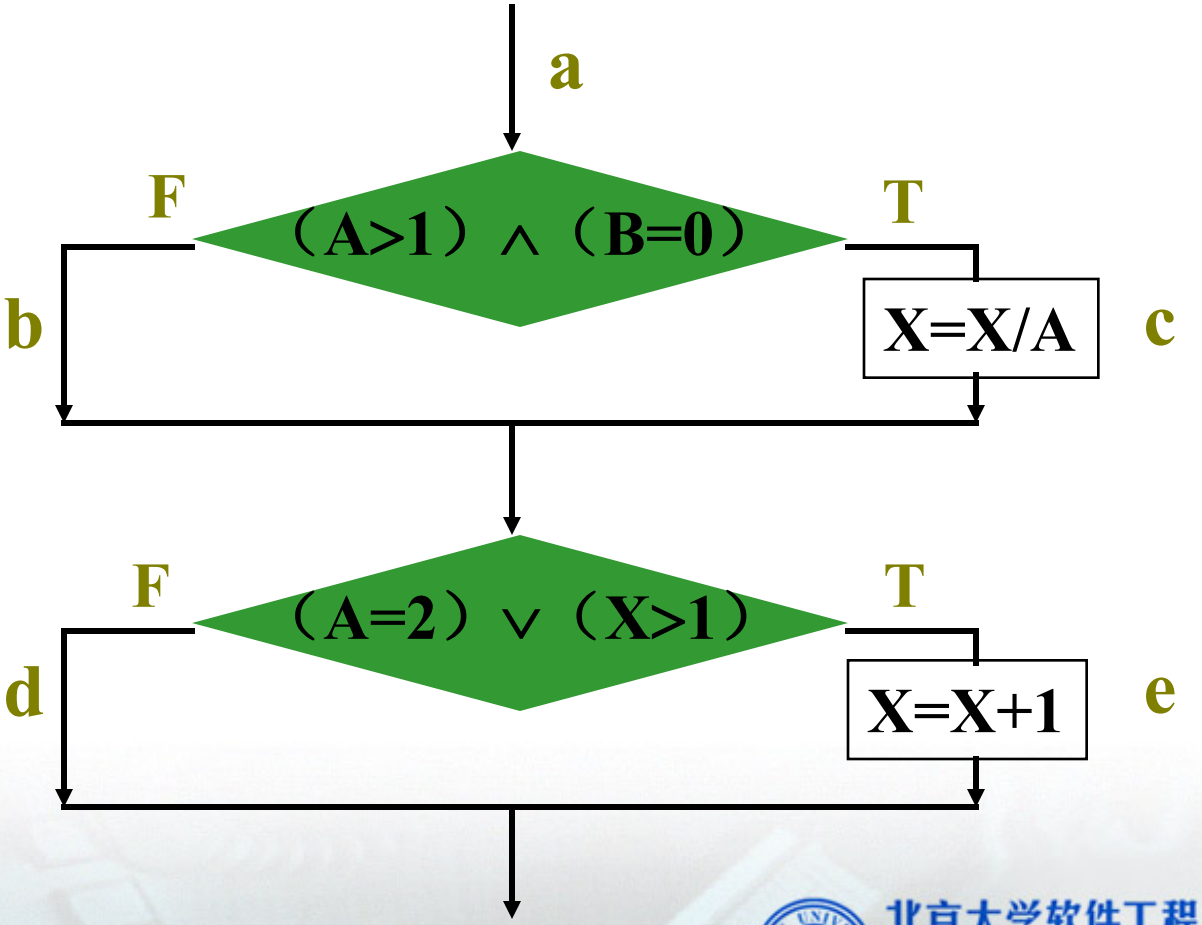
控制流程图:结点/分支/过程块/链路径



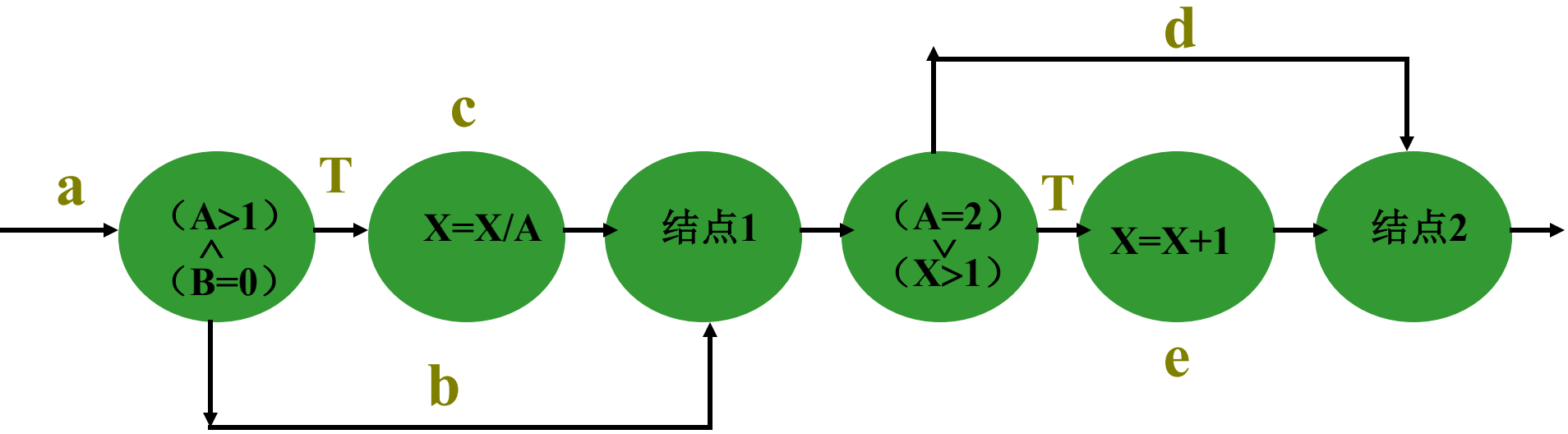
其中：节点1、节点3、节点5、节点6、节点7为过程块

节点2为分支，节点4为结点

例如：以下为一个程序流程图，其中该例子中有两个判断，每个判断都包含复合条件的逻辑表达式。



其控制流程图为：



（2）“完整测试”策略

该控制流程图有4条不同的路径。4条路径可表示为：

L1（a→c→e）简写ace 、L2（a→b→d）简写abd

L3（a→b→e）简写abe 、L4（a→c→d）简写acd

- 路径测试（PX）：执行所有可能的穿过程序的控制流程路径。

一般来说，这一测试严格地限制为所有可能的入口/出口路径。如果遵循这一规定，则我们说达到了100%路径覆盖率。在路径测试中，该策略是最强的，但一般是不可实现的。

针对该例子，要想实现路径覆盖，可选择以下一组测试用例（规定测试用例的设计格式为：【输入的（A，B，X），输出的（A，B，X）】）。

测 试 用 例	覆盖路径
【（2，0，4），（2，0，3）】	L1
【（1，1，1），（1，1，1）】	L2
【（1，1，2），（1，1，3）】	L3
【（3，0，3），（3，0，1）】	L4

● 语句测试（P1）：至少执行程序中所有语句一次。如果遵循这一规定，则我们说达到了100%语句覆盖率（用C1表达）。

在该例子中，只要设计一种能通过路径ace的测试用例，就覆盖了所有的语句。所以可选择测试用例如下：

【（2，0，4），（2，0，3）】 覆盖L1

语句覆盖是最弱的逻辑覆盖准则。

问题：就该程序而言，如果两个判断的逻辑运算写错，例如，第一个判断中的逻辑运算符“∧”错写成了“∨”，或者第二个判断中的逻辑运算符“∨”错写成了“∧”，利用上面的测试用例，仍可覆盖所有4个可执行路径，而发现不了判断中逻辑运算符出现的错误。

• 分支测试（P2）：至少执行程序中每一分支一次。如果遵循这一规定，则我们说达到了100%分支覆盖率（用C2表示）。

分支覆盖是一种比语句覆盖稍强的逻辑覆盖。但若程序中分支的判定是由几个条件联合构成时，它未必能发现每个条件的错误。

例如对于以上例子，如果选择路径L1和L2，就可得到实现分支覆盖的测试用例：

【（2，0，4），（2，0，3）】 覆盖L1

【（1，1，1），（1，1，1）】 覆盖L2

如果选择路径L3和L4，还可得另一组可用的测试用例：

【（2，1，1），（2，1，2）】覆盖L3

【（3，0，3），（3，1，1）】覆盖L4

问题：分支覆盖还不能保证一定能查出在判断的条件中存在的错误。例如，在该例子中，若第二个分支X>1错写成X<1，利用上述两组测试用例进行测试，无法查出这一错误。因此，需要更强的逻辑覆盖准则去检验判定的内部条件。

- 条件组合测试

条件组合测试是一种具有更强逻辑覆盖的测试。

条件组合测试，就是设计足够的测试用例，使每个判定中的所有可能的条件取值组合至少执行一次。如果遵循这一规定，则我们说就实现了条件组合覆盖。只要满足了条件组合覆盖，就一定满足分支覆盖。

在条件组合覆盖技术发展过程中，最初，在设计测试用例时，人们只考虑使分支中各个条件的所有可能结果至少出现一次。但发现该测试技术未必能覆盖全部分支。例如，在上图的例子中，程序段中有四个条件：A>1，B=0，A=2，X>1。

条件A＞1 取真值标记为T1，取假值标记为F1

条件B=0 取真值标记为T2，取假值标记为F2

条件A=2 取真值标记为T3，取假值标记为F3

条件X＞1 取真值标记为T4，取假值标记为F4

在设计测试用例时，要考虑如何选择测试用例实现T1、F1、T2、F2、T3、F3、T4、F4的全部覆盖。

例如，可设计如下测试用例实现条件覆盖：

测试用例	通过路径	条件取值	覆盖分支
【（1，0，3），（1，0，4）】	L3	F1 T2 F3 T4	b,e
【（2，1，1），（2，1，2）】	L3	T1 F2 T3 F4	b,e

从上面的测试用例，可以看到该组测试用例虽然实现了判定中各条件的覆盖，但没有实现分支覆盖，因为该组测试用例只覆盖了第一个判断的取假分支和第二个判断的取真分支。为此，人们又进一步提出了条件组合覆盖技术。

例如，在该例子中，前一个判定有4种条件组合：

（1）(A＞1)，(B=0)，标记为 T1 、T2 ；

（2）(A＞1)，(B≠0)，标记为 T1 、F2, ；

（3）(A≤1)，(B=0)，标记为 F1 、T2 ；

（4）(A≤1)，(B≠0)，标记为 F1 、F2 ；

后一个判定又有4种条件组合：

（5）(A=2)，(X＞1)，标记为 T3、T4 ；

（6）(A=2)，(X≤1)，标记为 T3、F4 ；

（7）(A≠2)，(X＞1)，标记为 F3、T4 ；

（8）(A≠2)，(X≤1)，标记为 F3、F4。

因此，要满足条件组合覆盖，设计的测试用例必须满足以下16种条件组合：

（1）(A＞1)，(B=0)，(A=2)，(X＞1)，可标记为 T1 、T2、T3、T4；
（1）(A＞1)，(B=0)，(A=2)，(X≤1)，可标记为 T1 、T2、T3、F4；
（1）(A＞1)，(B=0)，(A≠2)，(X＞1)，可标记为 T1 、T2、F3、T4
（1）(A＞1)，(B=0)，(A≠2)，(X≤1)，可标记为 T1 、T2、F3、F4。
（2）(A＞1)，(B≠0)，(A=2)，(X＞1)，可标记为 T1 、F2, T3、T4；
（2）(A＞1)，(B≠0)，(A=2)，(X≤1)，可标记为 T1 、F2、T3、F4；
（2）(A＞1)，(B≠0)，(A≠2)，(X＞1)，可标记为 T1 、F2、F3、T4；
（2）(A＞1)，(B≠0)，(A≠2)，(X≤1)，可标记为 T1 、F2、F3、F4。
（3）(A≤1)，(B=0)，(A=2)，(X＞1)，可标记为 F1 、T2、T3、T4；
（3）(A≤1)，(B=0)，(A=2)，(X≤1)，可标记为 F1 、T2、T3、F4；
（3）(A≤1)，(B=0)，(A≠2)，(X＞1)，可标记为 F1 、T2、F3、T4；
（3）(A≤1)，(B=0)，(A≠2)，(X≤1)，可标记为 F1 、T2、F3、F4。
（4）(A≤1)，(B≠0)，(A=2)，(X＞1)，可标记为 F1 、F2、T3、T4；
（4）(A≤1)，(B≠0)，(A=2)，(X≤1)，可标记为 F1 、F2、T3、F4；
（4）(A≤1)，(B≠0)，(A≠2)，(X＞1)，可标记为 F1 、F2、F3、T4；
（4）(A≤1)，(B≠0)，(A≠2)，(X≤1)，可标记为 F1 、F2、F3、F4。

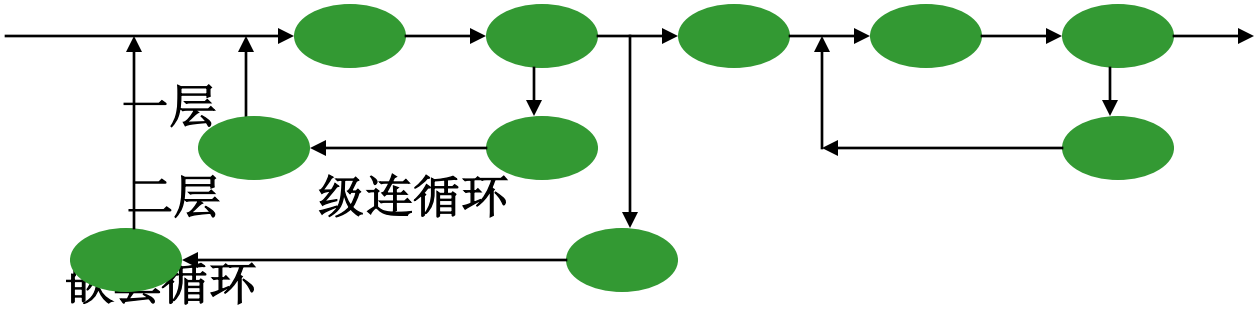
可以采用以下四组测试数据，从而实现条件组合覆盖。

测试用例	覆盖条件	覆盖组合号	通过路径
【（2，0，4），（2，0，3）】	T1 T2 T3 T4	1、5	L1
【（2，1，1），（2，1，2）】	T1 F2 T3 F4	2、6	L3
【（1，0，3），（1，0，4）】	F1 T2 F3 T4	3、7	L3
【（1，1，1），（1，1，1）】	F1 F2 F3 F4	4、8	L2
【（3，0，3），（3，0，1）】	T1 T2 F3 F4	1、8	L4

•••

这组测试用例实现了分支覆盖，也实现了条件的所有可能取值的组合的覆盖。

（3）循环情况的路径选取



还要考虑循环变量的具体情况

关键路径的选取

主要功能路径
没有功能的路径
最短路径

…

3） 功能测试-基于规格说明的测试

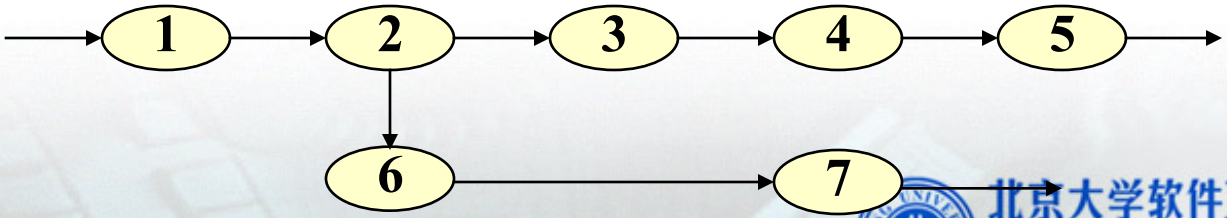
3.1 事务流测试技术

（1）基本概念：

①事务：以用户的角度所见的一个工作单元。

一个事务由一系列操作组成。其中某些操作可含有系统执行成分，或含有设备执行成分。

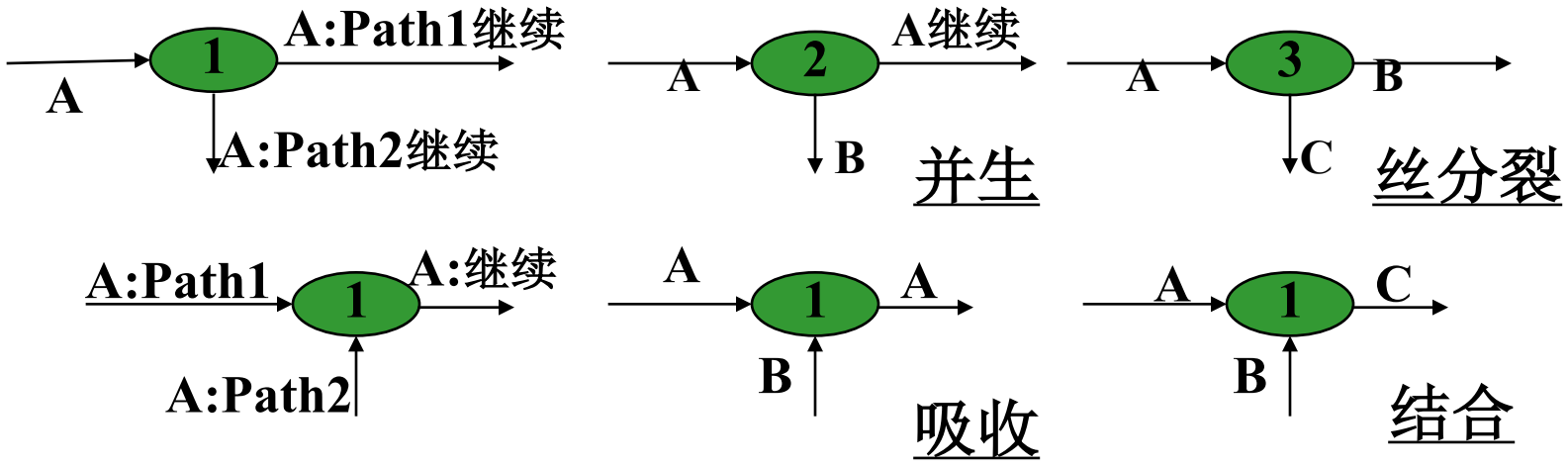
②事务处理流程（图）：系统行为的一种表示方法，为功 能测试建立了软件动作模式。其中使用了白盒测试中的一些概念，例如：分支，结点，链等。



（2）与程序控制流程图的比较：

①事务流图是一种数据流图，即从操作应用的历史，观察数据对象。

②事务流图中的判定；“抽象”了一个复杂的过程。



③事务流图存在“中断”，把一个过程等价地变换为具有繁多出口的链支。

测试设备:路径分析器,测试用例数据库,

测试执行调度器, 路径敏化问题...

（3）测试步骤

第一步：获取事务流程图，即建立被测对象模型；

第二步：浏览与复审

主要对事务进行分类，为设计用例奠 定基础；

第三步：用例设计

涉及：覆盖策略，事务选取，路径敏化等；

第四步：测试设备开发:

路径分析器,测试用例数据库,

测试执行调度器, ...

第五步：测试执行；

第六步：测试结果比较。

3.2 等价类划分技术

(1) 基本概念

①等价类：输入域的一个子集，在该子集中，各个输入数据对于揭示程序中的错误都是等效的。即：以等价类中的某代表值进行的测试，等价于对该类中其他取值的测试。

②有效等价类：指那些对于软件的规格说明书而言，是合理的、有意义的输入数据所构成的集合。

-用于实现功能和性能的测试。

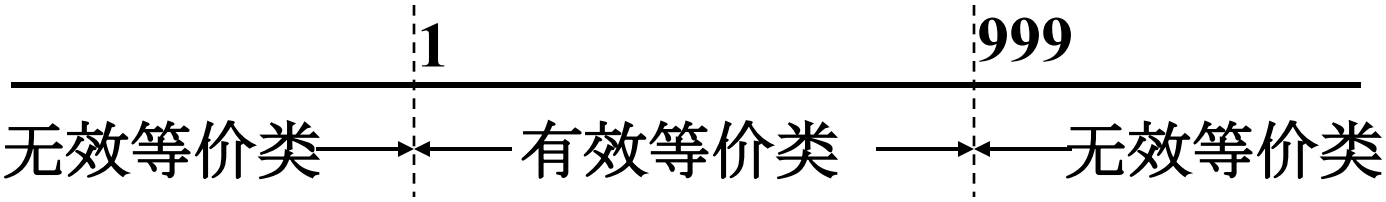
③无效等价类：指那些对于软件的规格说明书而言，是不合理的、无意义的输入数据所构成的集合。

-用于测试那些所实现的功能和性能不符合规格说明书的要求。

（2）等价类划分原则（指南）

① 如果输入条件规定了输入数据的取值范围或值的个数，则可以确定一个有效等价类和二个无效等价类。例如：

输入条件："...项数可以是1到999..."



② 如果输入条件规定了输入值的集合，或规定了"必须如何"的条件，则可以确定一个有效等价类和一个无效等价类。例如："标识符是一字母打头的...串。"则字母打头的--为一个有效等价类，而其余的—为一个无效等价类

③ 如果输入条件是一个布尔量，则可以确定一个有效等价类和一个无效等价类。

④ 如果输入条件规定了输入数据的一组值，而且软件要对每个输入值进行处理，则可以为每一个输入值确定一个有效等价类，为所有不允许的输入值确定一个无效等价类。

⑤ 如果输入条件规定了输入数据必须遵循的规则，则可以确定一个有效等价类（符合规则），和若干个无效等价类。

例如：“语句必须以；号结束”

注意：如果在已确定的等价类中各元素在软件中的处理方式不同，则应根据需要对等价类进一步进行划分。

（3） 测试用例设计

在确定了等价类之后，建立等价类表：

输入条件	有效等价类	无效等价类
……	……	……
……	……	……

（4） 实例研究

某一8位计算机，其十六进制常数的定义为：以0x或0X开头的数是十六进制整数，其值的范围是-7f至7f（大小写字母不加区别），如0x13，0X6A，-0x3c

第一步：建立等价类表

输入条件	有效等价类	无效等价类
十六进制整数	1、 0x或0X开头1-2位数字串	4、非0x或非-开头的串
		5、含有非数字且（a,b,c,d,e,f）以外字符
		6、多于5个字符
	2、以- 0x开头的1-2位数字串	7、-后跟非0的多位串
		8、-0后跟数字串
		9、-后多于3个数字
	3、在-7f至7f之间	10、小于-7f
		11、大于 7f

第二步：为有效等价类设计测试用例

测试用例	期望结果	覆盖范围
0x23	显示有效输入	1，3
-0x15	显示有效输入	2，3

第三步：为无效等价类至少设计一个测试用例

测试用例	期望结果	覆盖范围
2	显示无效输入	4
G12	显示无效输入	5
123311	显示无效输入	6
-1012	显示无效输入	7
-011	显示无效输入	8
-0134	显示无效输入	9
-0x777	显示无效输入	10
0x87	显示无效输入	11

3.3 软件测试步骤

（1）单元测试

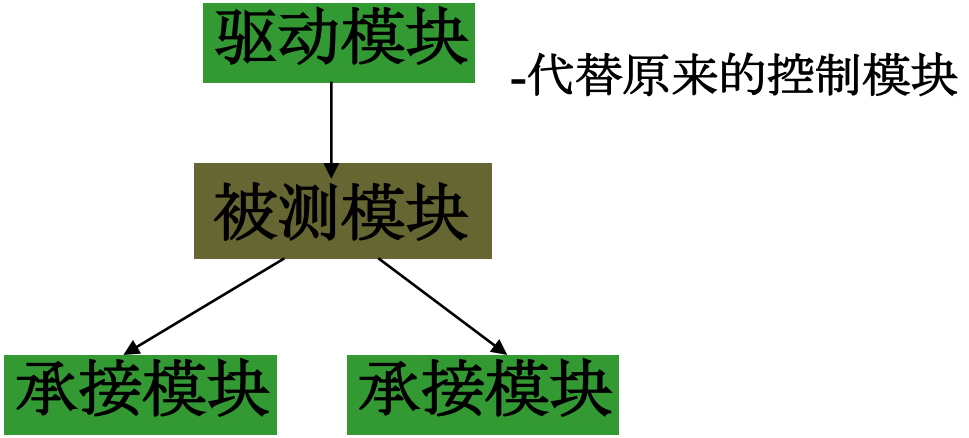
（2）集成测试

集成测试是一种软件集成化技术

- 方式:自顶向下或自底向上
- 设计测试设备

  驱动模块

  承接模型



（3）有效性测试