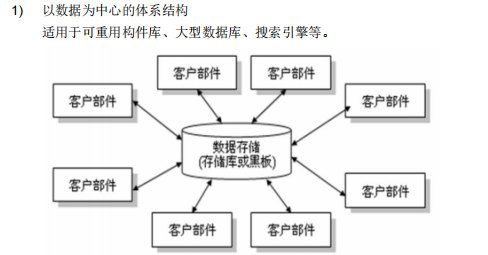
**选择题与名词解释**

**1) 体系结构：**

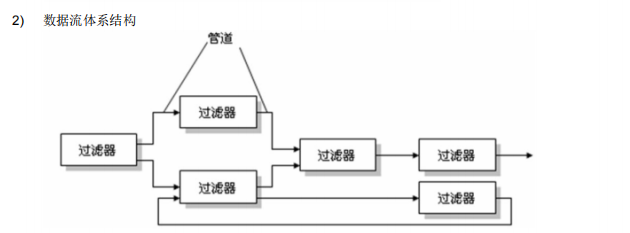
**概念：**系统的一个或多个结构，它包括软件构件、这些构件对外可见的属性以及它们之间的相互关系

**风格(选择题)：**

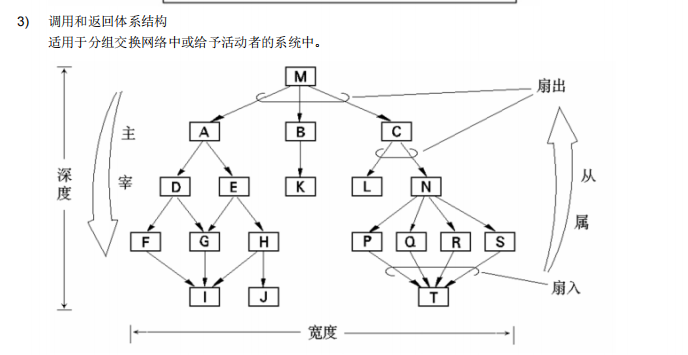
**❑ 以数据为中心的体系结构**



**❑数据流体系结构**

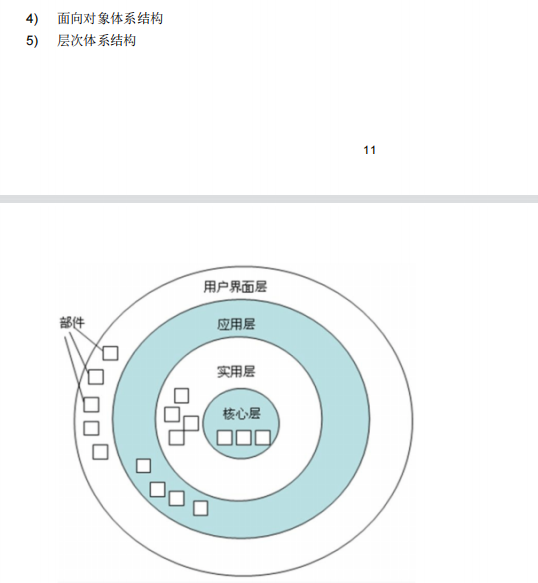


**❑ 调用和返回体系结构**

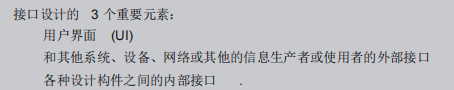


**❑ 面向对象体系结构**

**❑ 层次体系结构**



**2) 接口：**类、构件或其他分类的外部可见的（公共的）操作说明，而没有内部结构的规格说明。包括：一组描述类的部分行为的操作和提供操作的访问方法。



**4) 构建：**系统中模块化的、可部署的和可替换的部件，该部件封装了实现并暴露一组接口。构件包括一组协作的类，也可以包含一个单独的类。

**5) CRC模型**：表示类的标准索引卡片的集合，这些卡片分成三部分：

1.顶部写类名

2.下面左侧列出类的职责，职责：和类相关的属性和操作

3.下面右侧列出类的协作关系，协作：信息请求或动作请求

**6) 重构：**为简化设计而进行的重组。使用这样一种方式改变软件系统的过程：不改变代码设计的外部行为，而是改变其内部结构。

**7) 软件风险管理：**

**概念：**量化每个风险的不确定性和损失程度，采取行动降低或消除风险

**目的：**

**❑ 提高项目的成功率；**

**❑ 保证风险发生时的及时反应；**

**❑ 增加团队的健壮性；**

**❑ 帮助项目经理抓住工作重点；**

**方法：**

**❑ 风险识别**

**❑ 风险估算**

**❑ 风险缓解，检测和管理**

**8) 软件配置管理(SCM)：**

**概念：**协调软件开发以最大限度的减少混乱的技术称为配置管理,是贯穿于整个【软件过程】中的保护性活动。

**目的：**保证软件项目生成的产品在软件生命周期中的完整性和一致性**方法：**

**❑** 标识变更

**❑** 控制变更

**❑** 保证恰当的实施变更

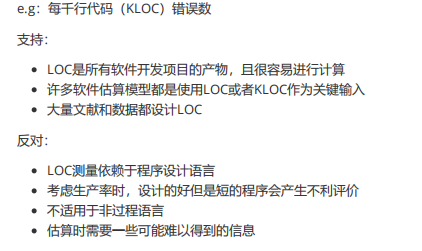
**❑** 向利益相关者报告变更

**9) 软件估算与度量：**

**❑ 面向规模度量(LOC测量)：**

通过对质量和生产率的测量进行规范化后得到，这些测量都是根据开发过的软件的规模得到的。

可以自选规范化值，根据表中包含的基本数据，每个项目都能得到一组简单的面向规模的度量

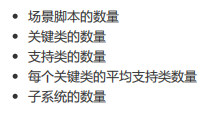


**❑ 面向功能度量(FP测量)：**

以功能测量数据作为规范化的值

**❑ 面向对象度量：**

**基于以下参数**

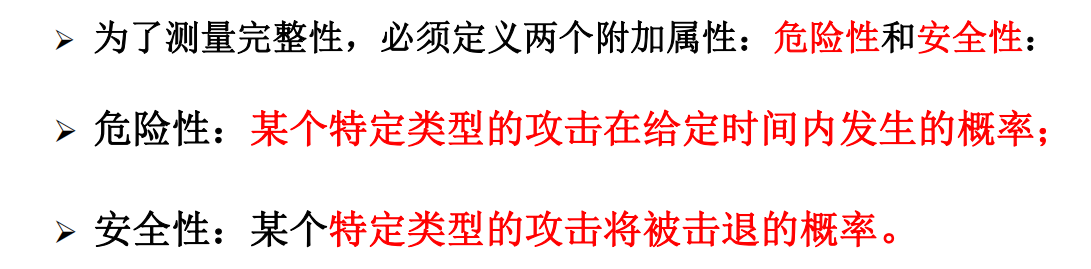


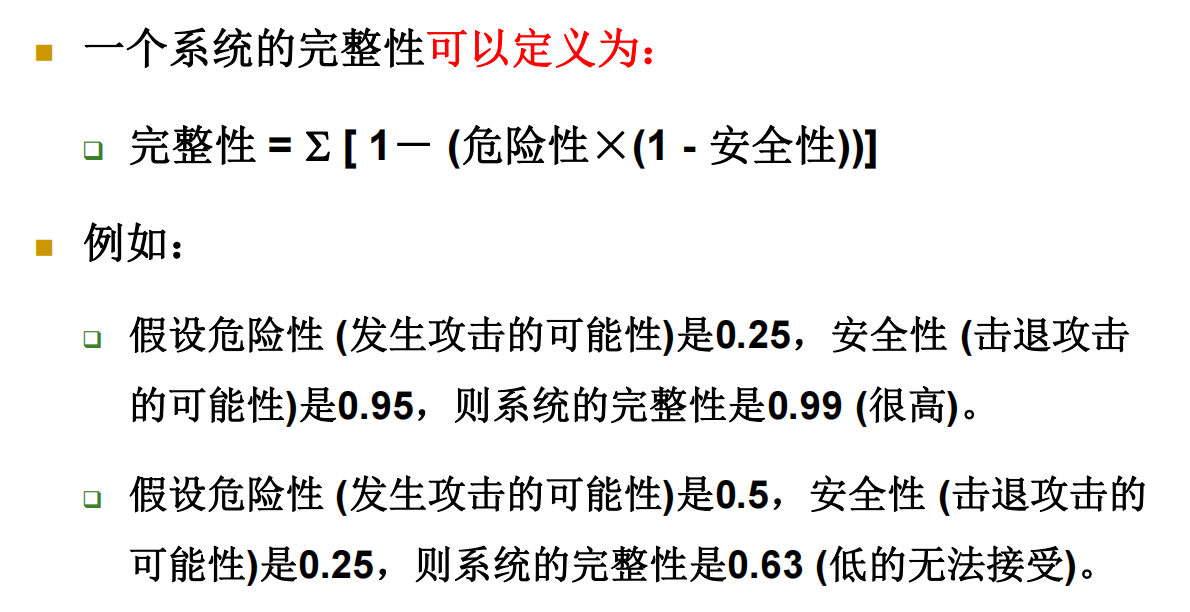
**10) 软件质量度量**

**❑ 测量指标：**

**❑ 正确性：软件完成所需的功能的程度。最常用的测量：每千行（KLOC）的缺陷数，缺陷是按某标准时间段计数的，比如1年。**

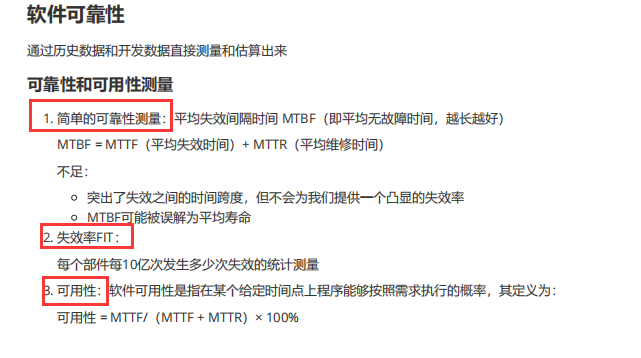
**❑ 可维护性：遇到错误时程序能够被修改的容易程度，只能采取间接测量（度量方法：平均变更时间）**

**❑ 完整性：在安全方面的抗攻击能力。**



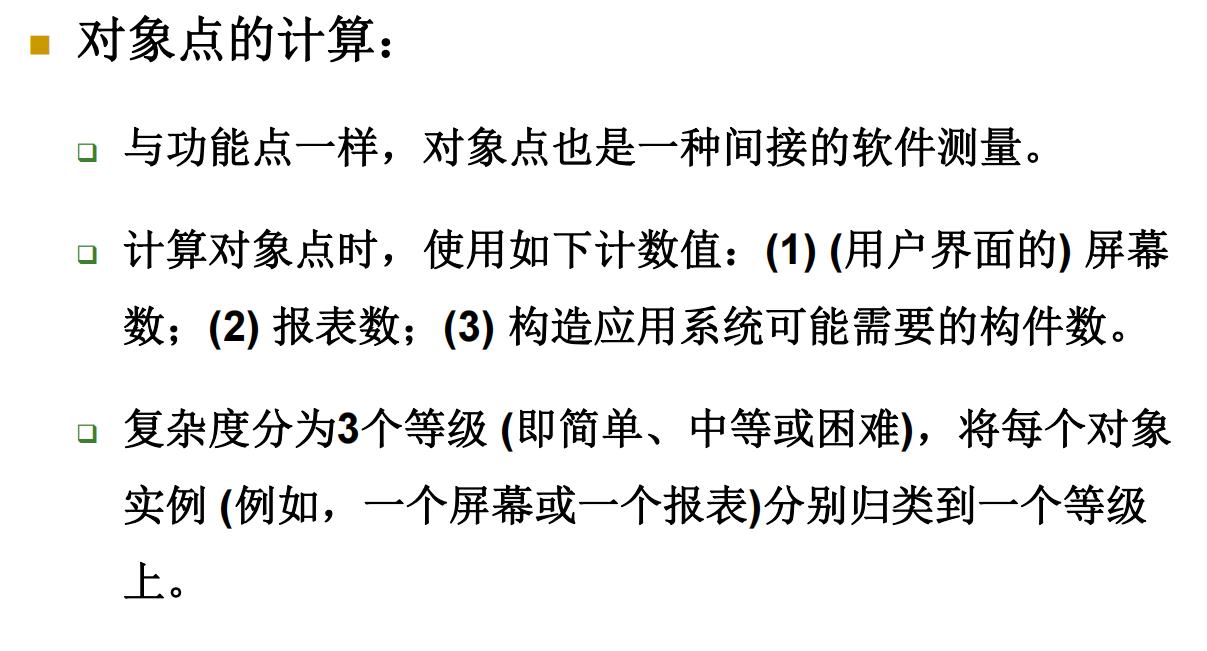
**❑ 可用性：使用的容易程度**

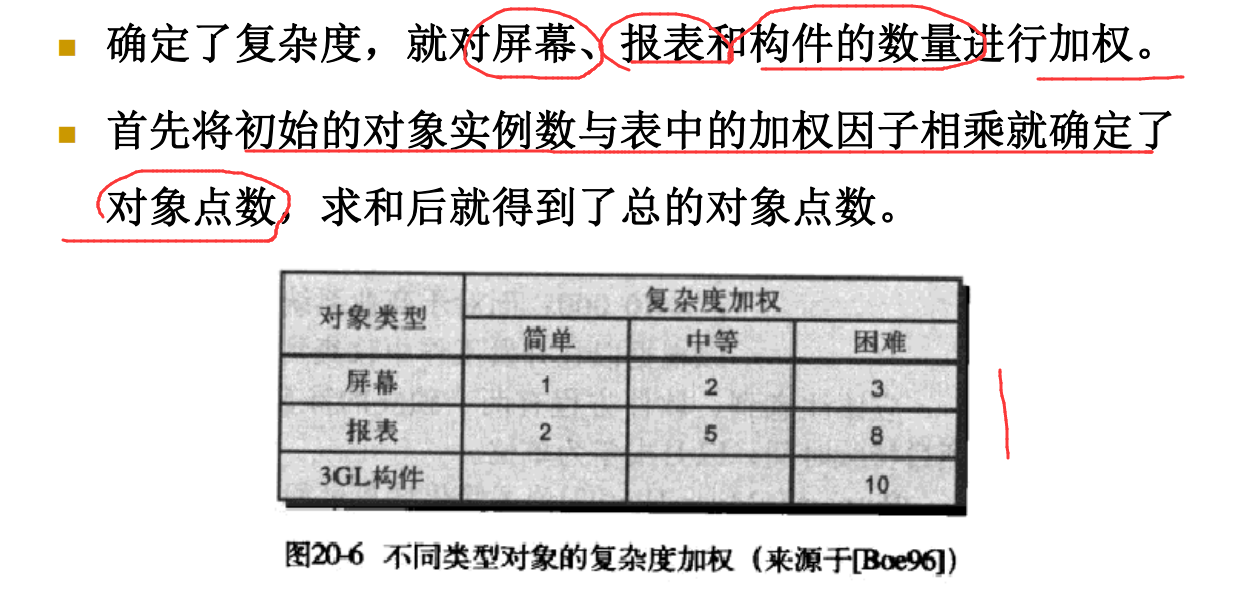
**❑ 可靠性**



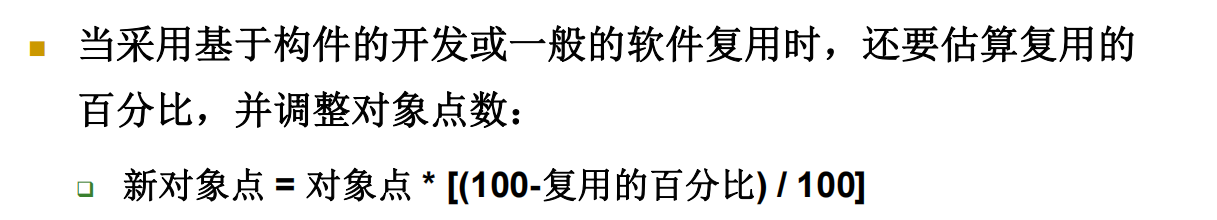
**❑ COCOMO II模型(经验估算模型)：**

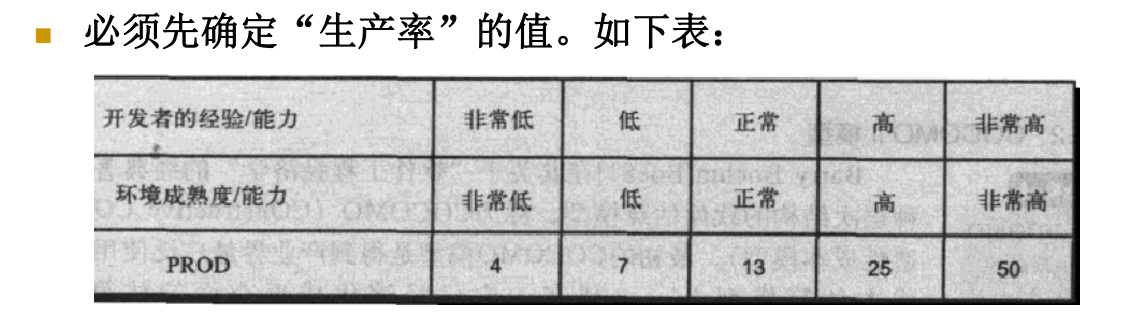
**step1:计算对象点**



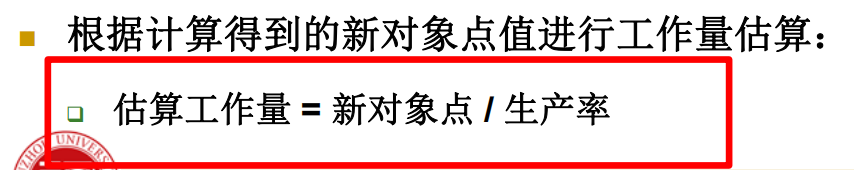


**step2：调整对象点**

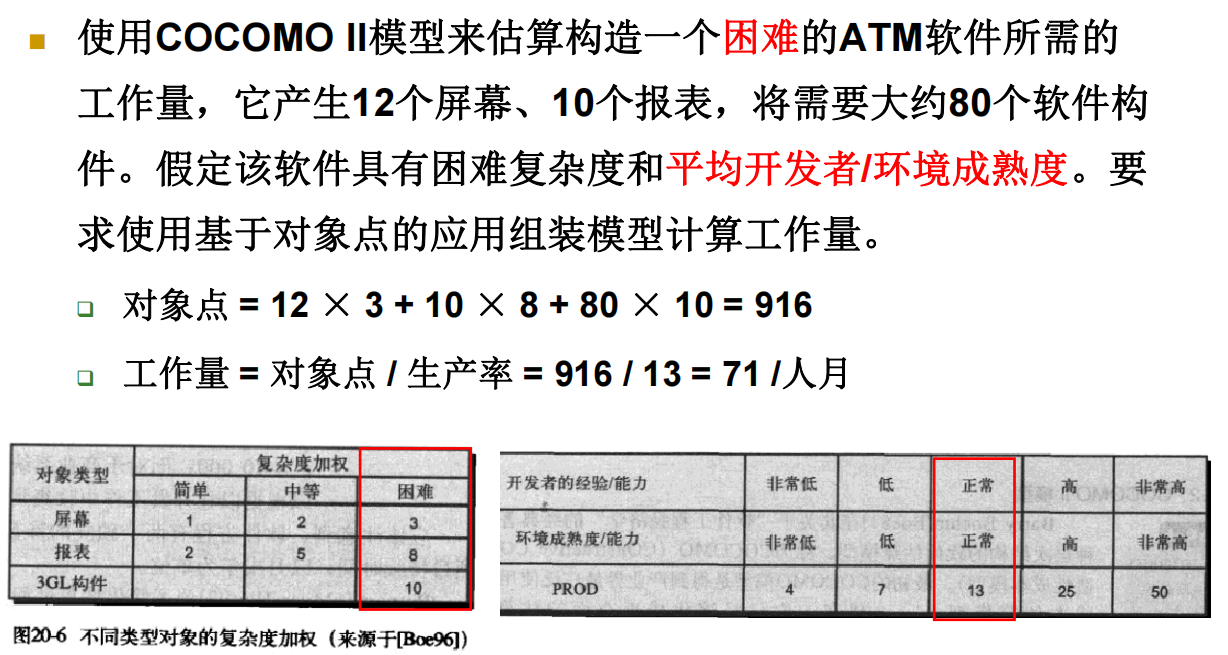
**step3；确定生产率**



**step4:计算工作量(单位：人/月)**

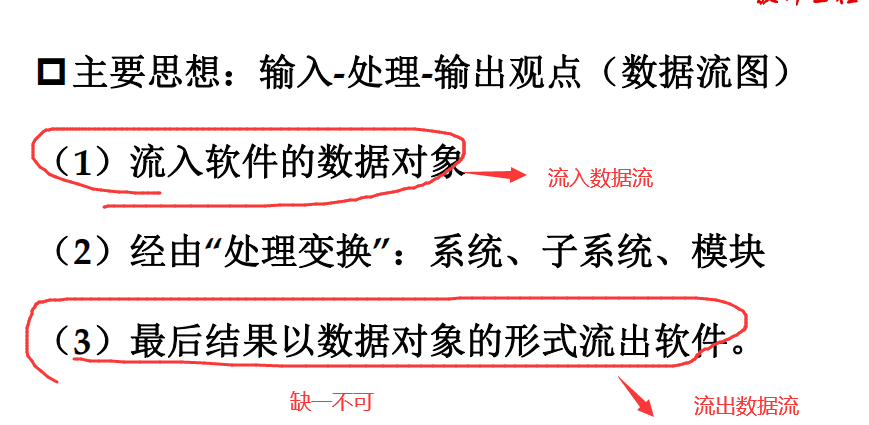


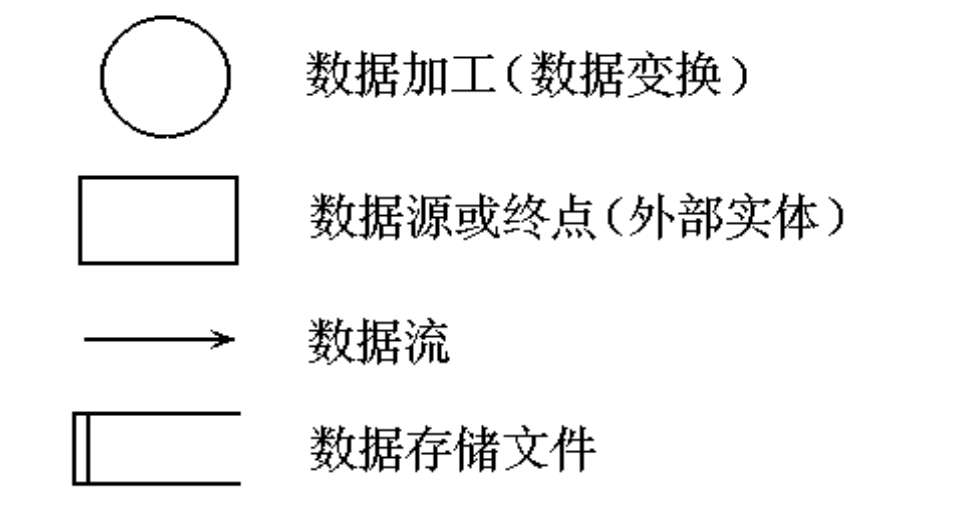
**例子：**

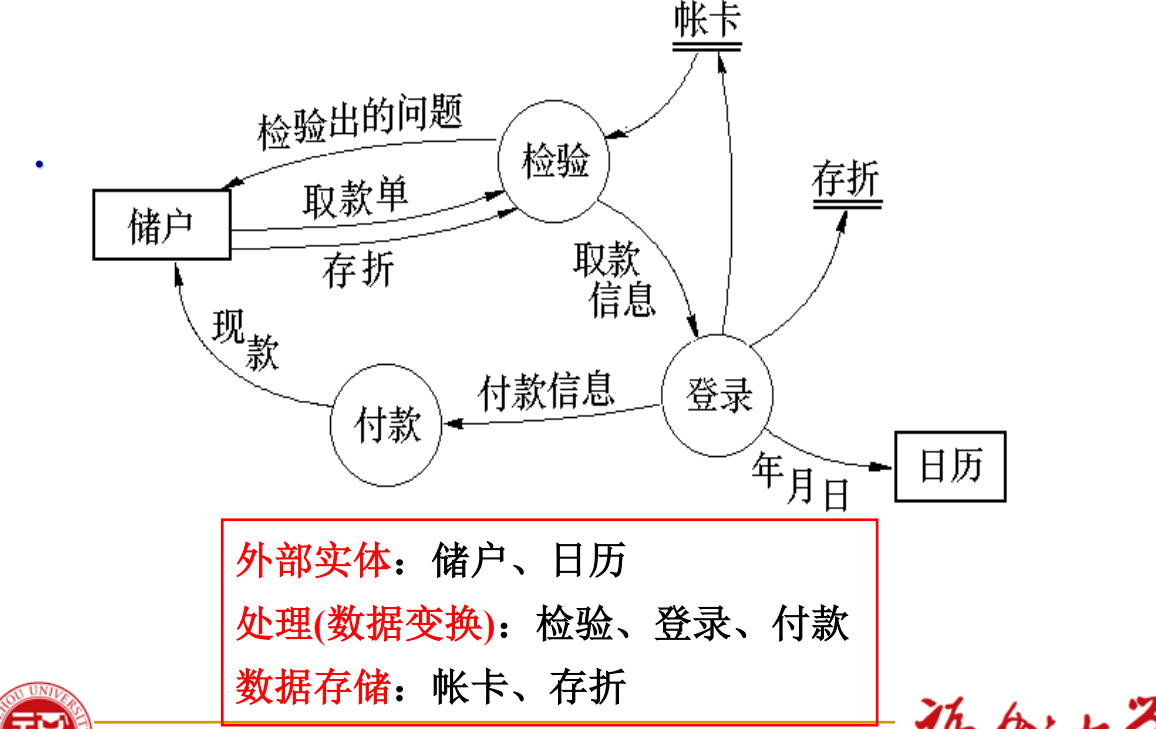


**11) 数据流图DFD**

**用于：结构化建模(面向数据流的建模方法)：数据流建模（数据流图）+控制流建模(状态图)**





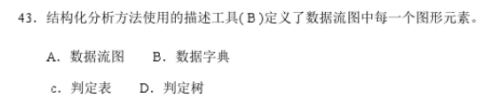


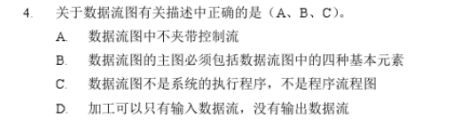
**例题：**

IMG_256



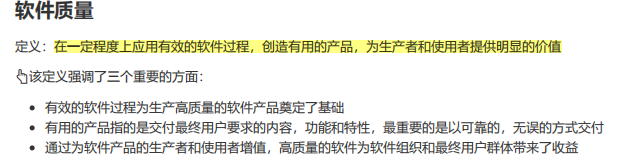






**12) 软件质量管理**

**定义：**



**13) 常见的开发模型(过程模型)定义、特点、适用场合：**

**瀑布，增量，演化，并发**

**❑ 瀑布模型(线性顺序模型)：**

**定义：**瀑布模型又称为经典生命周期，它提出了一个系统的、顺序的软件开发方法，从用户需求规格说明开始，通过沟通、策划、建模、构建和部署的过程，最终提供完整的软件支持。

**特点：**1.结构简单

2.阶段间具有顺序性和依赖性

3.推迟实现的观点：前面步骤完成后才考虑实现

4.质量保证的观点：每一阶段都需要有文档以及经过评审

**适用场合：**需求确定、变更相对较少的项目



**❑ 瀑布模型变种-V模型：**

**定义：**本质是瀑布模型增加了一系列测试（质量保证动作）

**特点：**1.将瀑布模型的活动细化为动作

2.验证过程动作与开发过程动作相对应

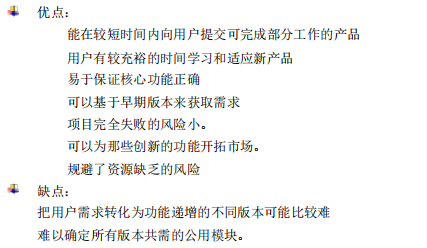
3.强调反馈，不将问题留到下一步

**适用场合：**需求确定、变更相对较少的项目

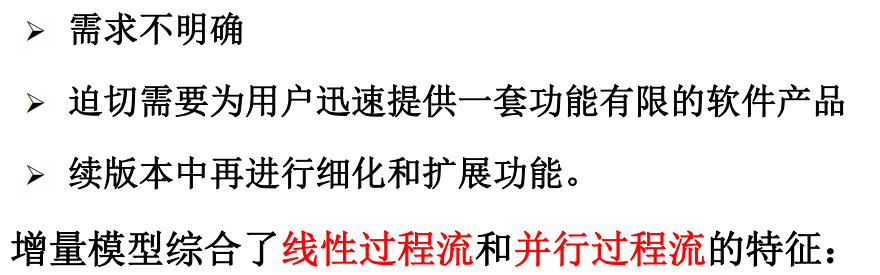
**❑ 增量模型(增量过程模型)or变体-迭代开发：**

**定义：**综合了线性过程流和并行过程流的特征，以一种演化过程流的方式生产出软件的可交付增量的过程模型

**特点：**



**适用场合：**

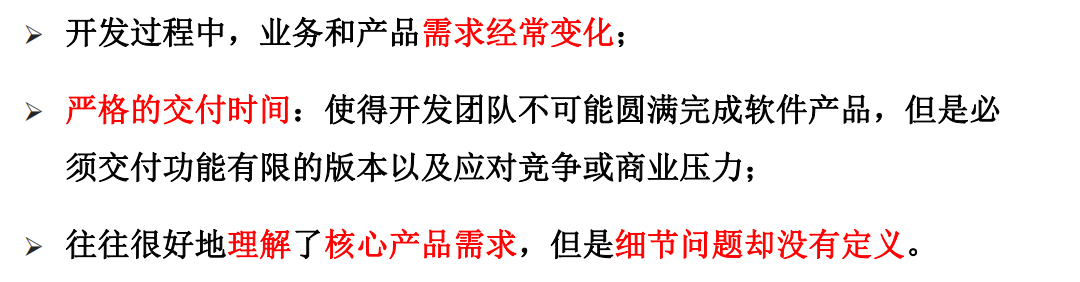


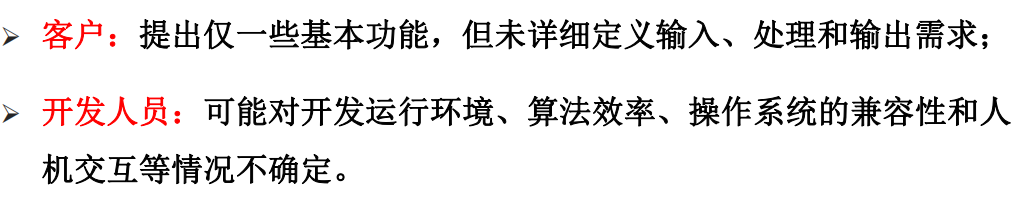
**❑ 演化模型1-原型开发：**

**定义：**首先提出一个大概能实现用户需求的原型，接着每次迭代产生软件的一个更完整的版本，最后丢弃原型。

**特点：**用户能够感受到实际系统；开发者能很快建造出一些东西。但开发者没有考虑整体软件质量和长期的可维护性；开发者往往在实现过程中采用折中的手段。

**适用场合：**

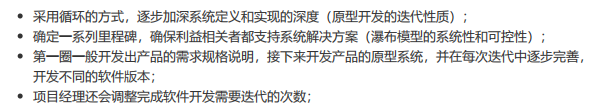




**❑ 演化模型2-螺旋模型**

**定义：**是风险驱动的软件开发模型：结合原型法和瀑布模型的特征。

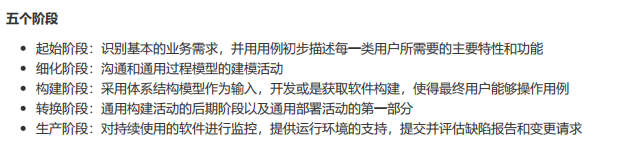
**特点：**



**适用场合：**大型项目开发

**❑ 统一过程：**

**定义：**



**特点：用例驱动，以架构为核心，迭代并且增量**

**适用场合：大型项目**

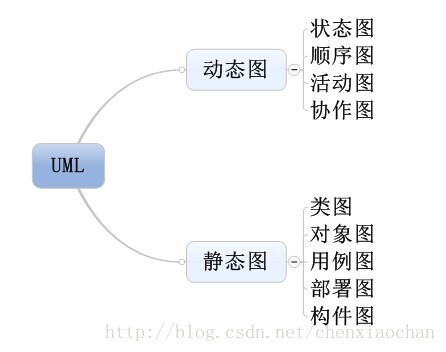
**❑ 敏捷开发：**

**定义：它是一种软件开发方法论，可以应对客户快速变更的需求。**

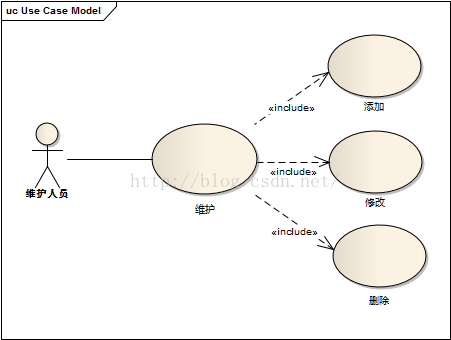
**特点：以人为核心，采用迭代的方式，循序渐进的开发软件**

**适用场合：特别适用于 web 应用开发。**

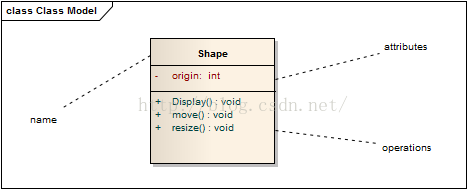
**14) 9种UML图(概念，作用，分别在什么阶段用到)**



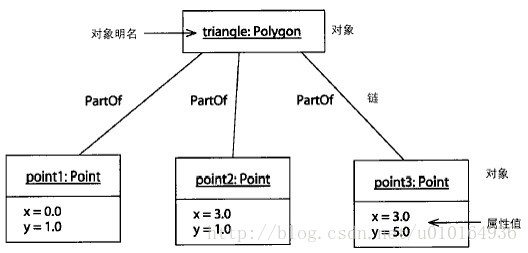
**❑ 用例图：定义了系统的功能需求。从系统的外部看系统功能，并不描写叙述系统内部对功能的详细实现。**



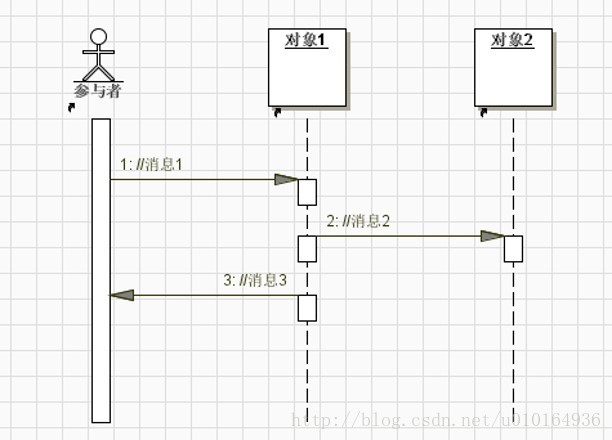
**❑ 类图：对静态结构的描写叙述，用来定义系统中类和类之间的关系。**



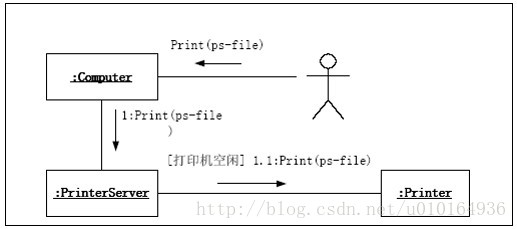
**❑ 对象图：表示类的对象实例。**



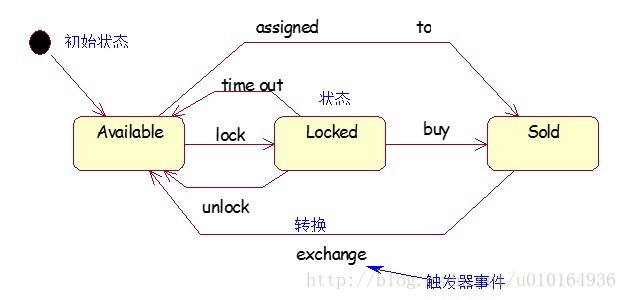
**❑ 顺序图(序列图)：强调消息时间顺序的交互图，描写叙述类以及类相互交互以完毕期望行为的消息。**



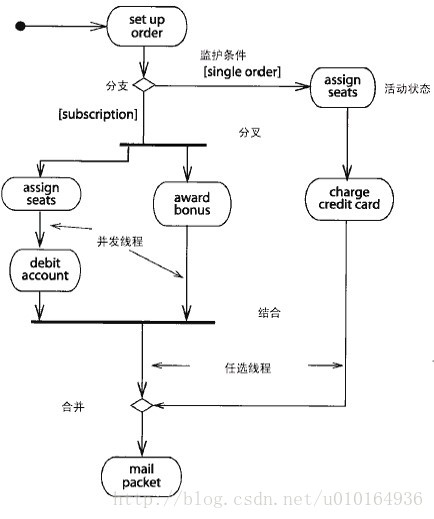
**❑ 协作图：**协作图是动态视图的另一种表现形式，强调参加交互的各对象的组织，侧重于说明哪些对象之间有消息的传递。因此，协作图只对相互间有交互作用的对象和这些对象间的关系建模，而忽略了其他对象和联系。



**❑ 状态图：**一个类对象所可能经历的所有历程的模型图。由对象的各个状态和连接这些状态的转换组成。



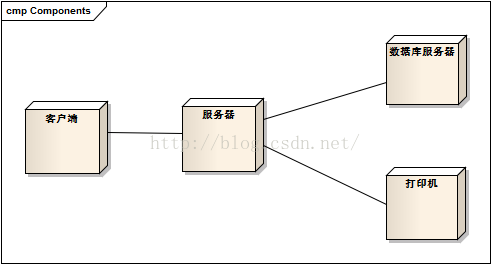
**❑ 活动图：**状态图的一个变体，用来描述执行算法的工作流程中涉及的活动。



**❑ 构件图：**用于描写叙述各种软件构件之间的依赖关系。以便通过这些依赖关系来估计对系统构件的修改给系统可能带来的影响。



**❑ 部署图：用来显示系统中软件和硬件的物理架构。**



**❑ 包图：用于描述包的元素以及与包之间的依赖**



**各个阶段设计的UML图：**

**❑ 需求阶段：**用例图描写叙述需求

**❑ 分析阶段：**明确解决问题的细节，主要采用类图来描述静态结构，采用顺序图，协作图，活动图，状态图来描述系统动态行为；

**❑ 设计阶段：**给出解决方案，主要采用类图、包图，对类的接口进行设计；

**❑ 实现阶段：**构件图，部署图

**❑ 测试阶段：【**单元测试】使用类图和类的规格说明书。【集成测试】使用类图，包图，构件图和协作图。【系统测试】使用用例图来测试系统的功能；

**❑ 集成和交付阶段：**主要采用构件图、包图、部署图；

**15) 设计概念：**

**❑ 抽象：**现实世界中一定事物、状态或过程之间总存在着某些相似的方面(共性)。把这些相似的方面集中和概括起来，暂时忽略它们之间的差异，这就是抽象。(分为【数据抽象】和【过程抽象】)

**❑ 体系结构：**如何组织程序构件，相当于整个房子的平面设计图 。（3个属性：结构特性、外部功能特性、相关系统族）

**❑ 模式：**经过验证的，用于解决在特定环境下、重复出现的、特定问题的解决方案 。

**❑ 关注点分离：**将关注点分割为更小的关注点，以便于用更小的工作量和时间解决一个问题

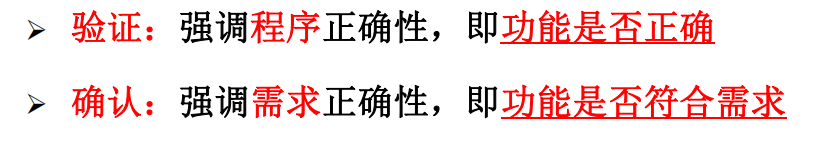
**❑ 模块化：**关注点分离最常见的表现，按照设计原则将系统划分为若干个较小的模块。

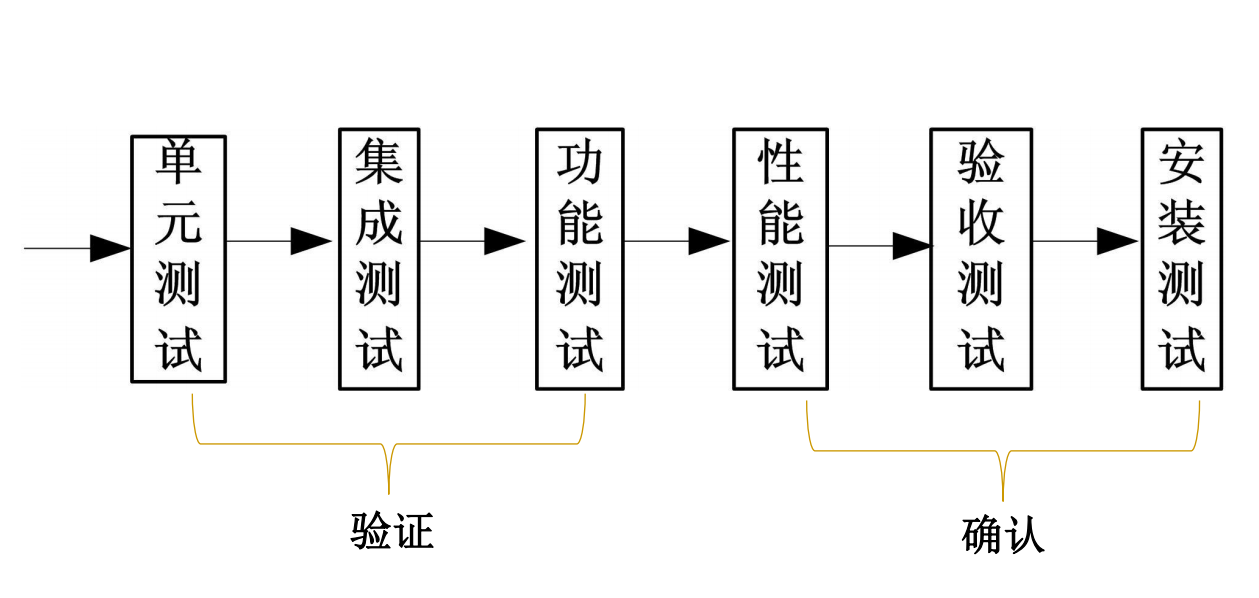
**❑ 信息隐蔽：**在一个模块接口后面隐藏数据结构的细节和过程的处理，模块的用户不需要知道细节。

**❑ 功能独立：**关注点分离，模块化，抽象和信息隐蔽概念的直接产物。(两条标准：【内聚性】和【耦合性】)

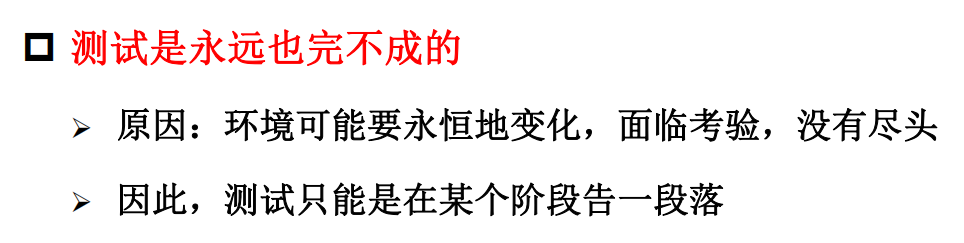
**16) 测试流程(选择题)**







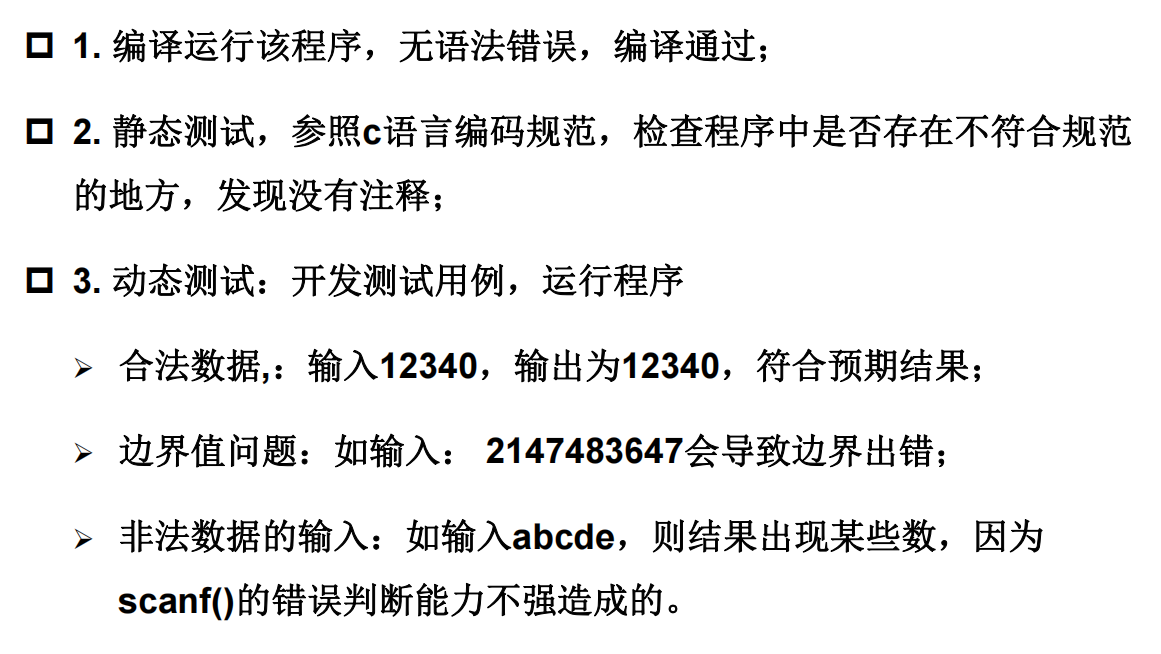
**17) 测试何时完成：**



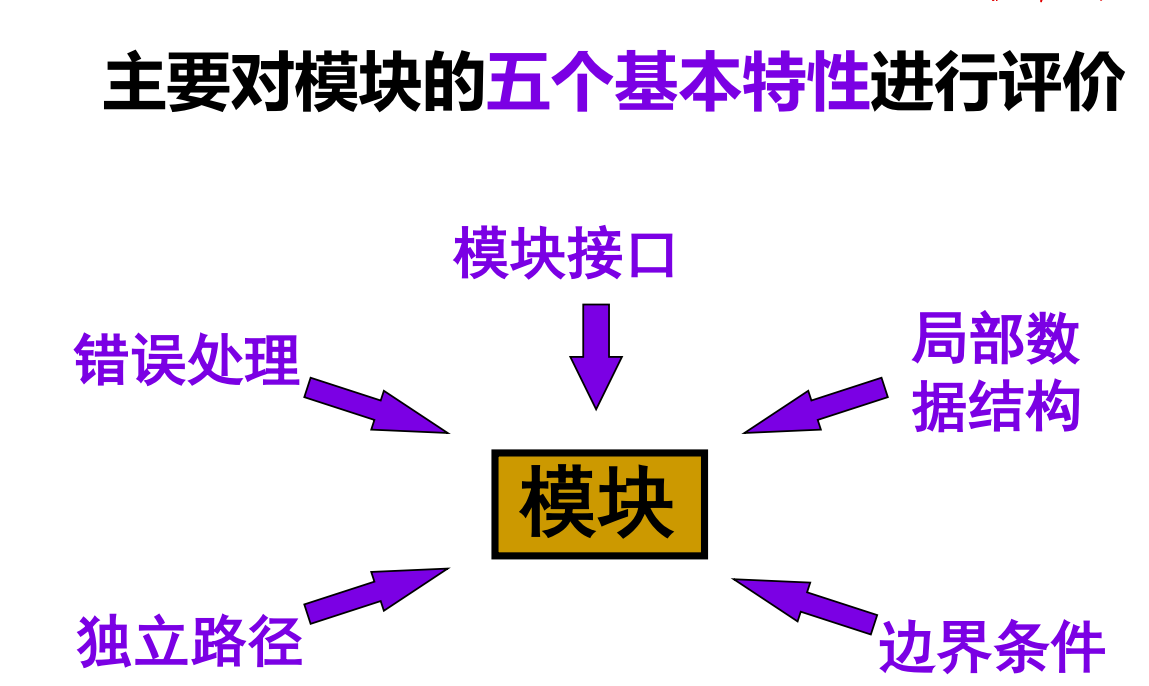
**18) 传统软件测试测试：**

**❑ 单元：结构化开发方法中，最小可被测试的模块**

**❑ 单元测试=静态测试+动态测试**

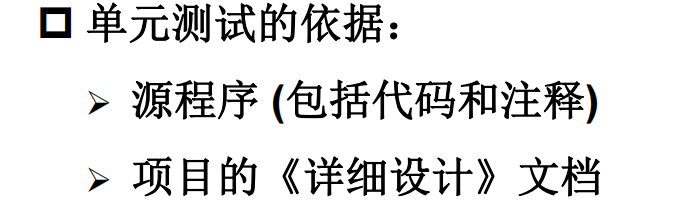


**❑ 单元测试评价指标:**



**❑ 何时进行【单元测试】：编码完成后**

**❑ 单元测试依据：**



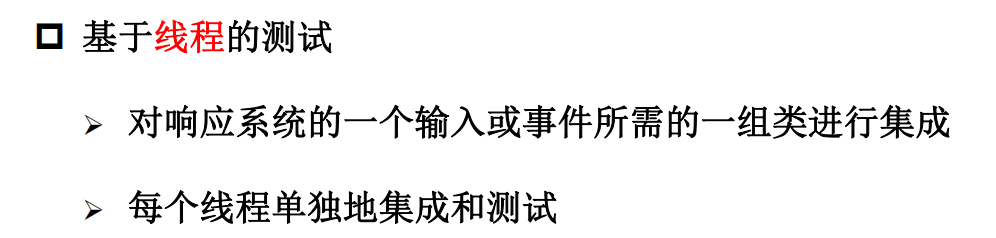
**❑ 集成测试：将通过测试的单元模块组装成系统或子系统，进行测试**

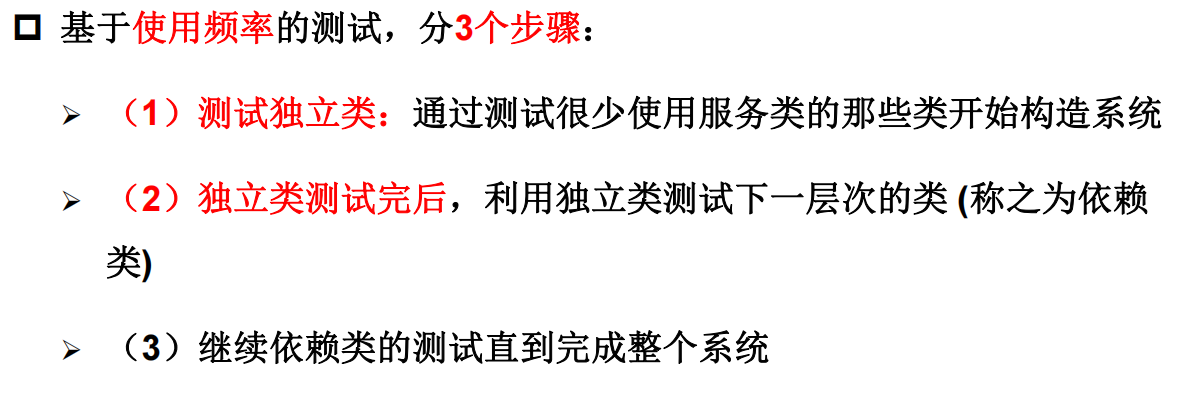
**19) 面向对象测试方法与传统测试方法差异：**

**❑ 最小测试单位：“模块”(唯一)VS“类”(以操作为驱动，不唯一)**

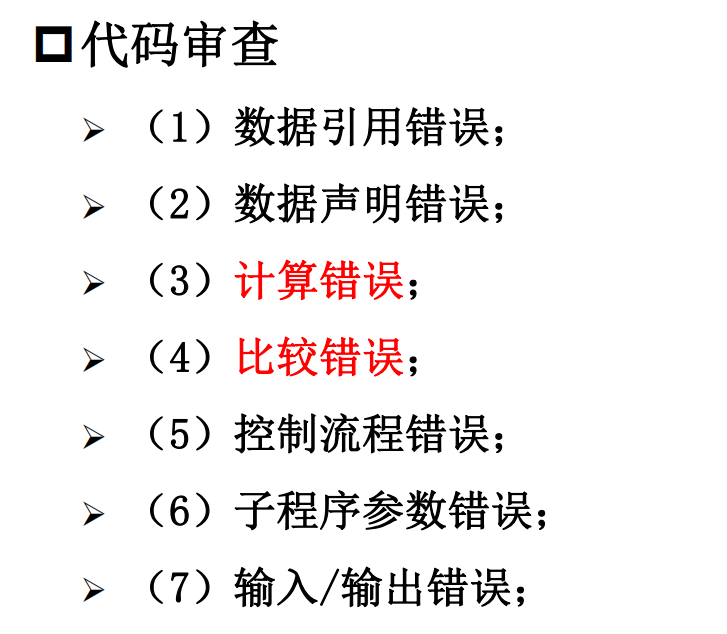
**❑ 结构：“模块”有层次关系，“类”地位对等**

**❑ 集成测试两种策略：基于线程的测试、基于使用频率的测试**





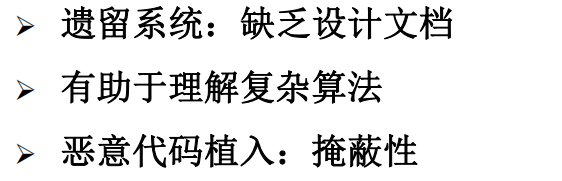
**17) 功能测试(又称【有效性测试】)-白盒测试策略：把测试对象看做一个透明盒子，测试人员了解程序的内部结构-【结构测试】**

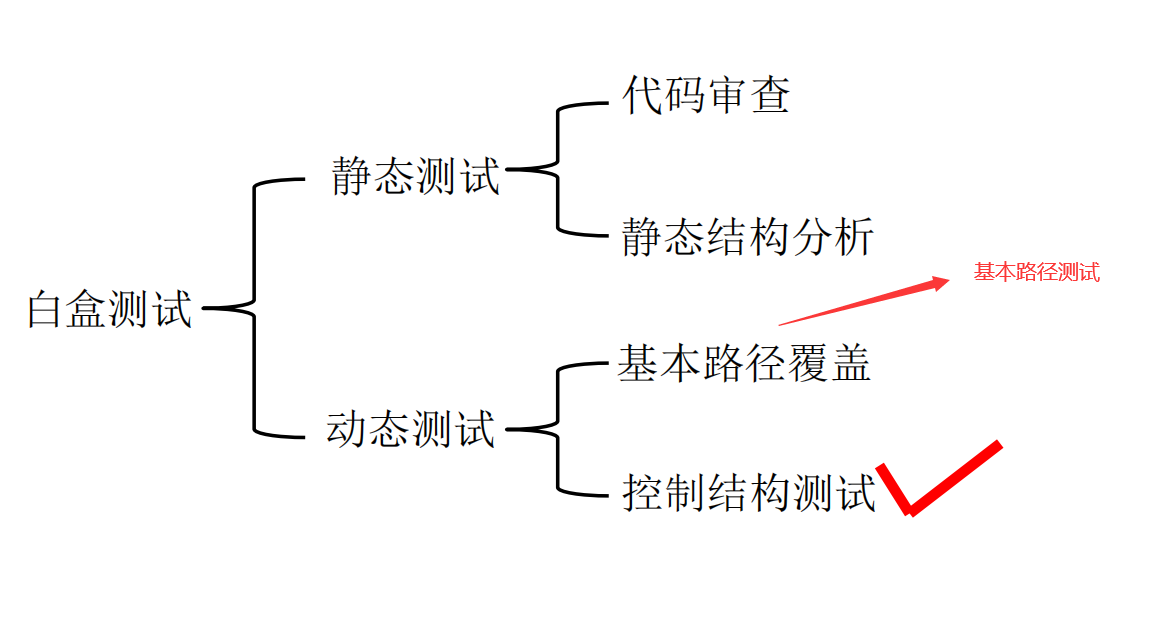


七种错误的实际例子：第三部分 P219-227

**目的：从源程序中“还原”出设计模型**

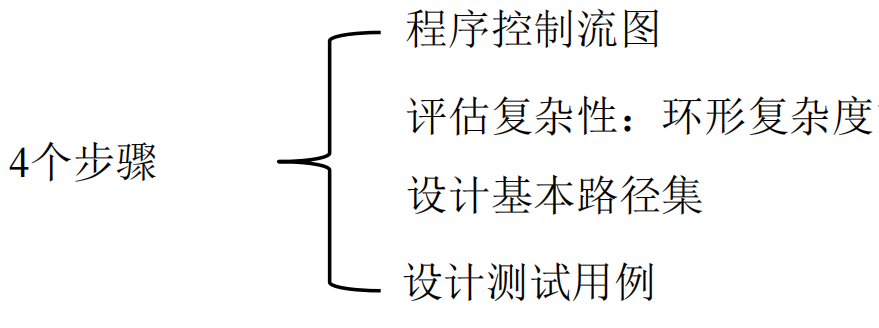
**应用场景：**



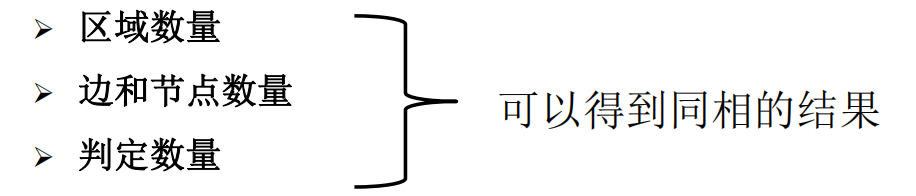


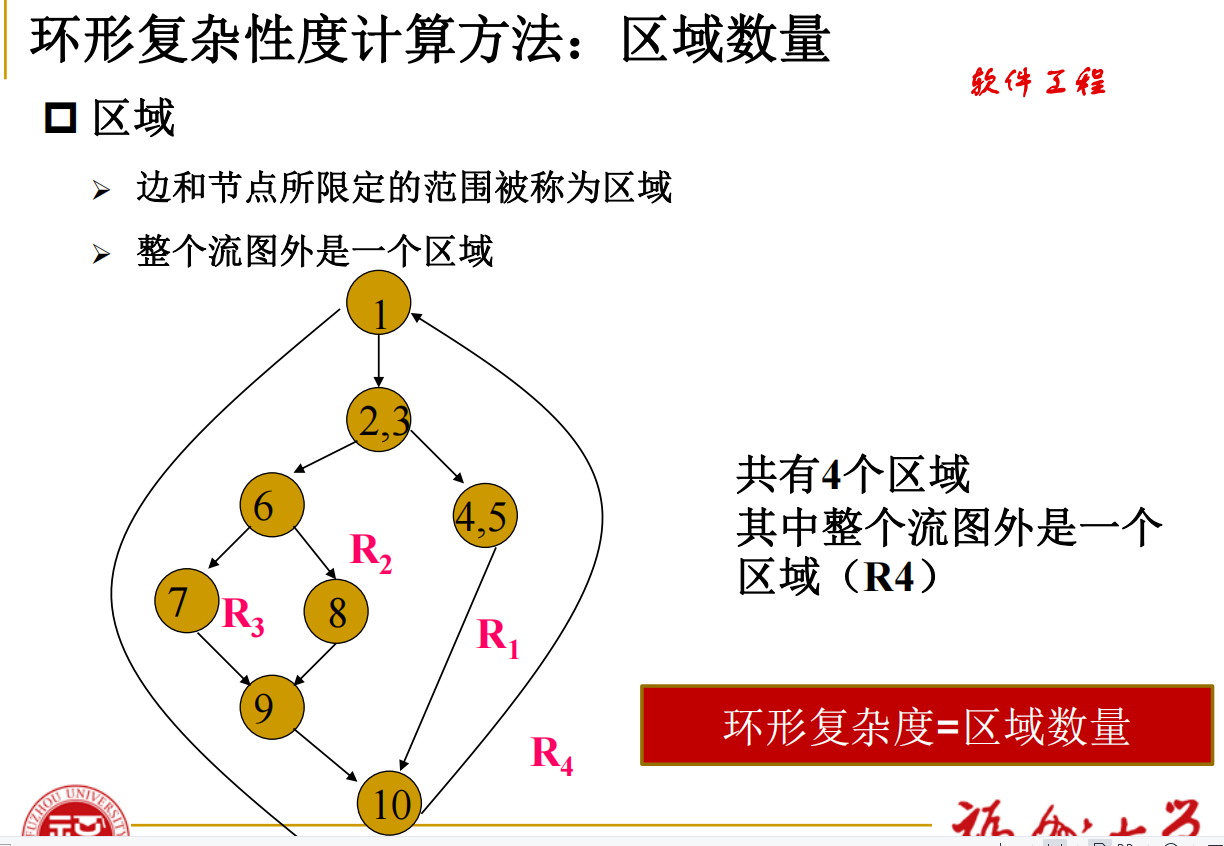
**18) 基本路径测试：**设计足够多的测试用例，要保证每一个可执行语句至少执行一次

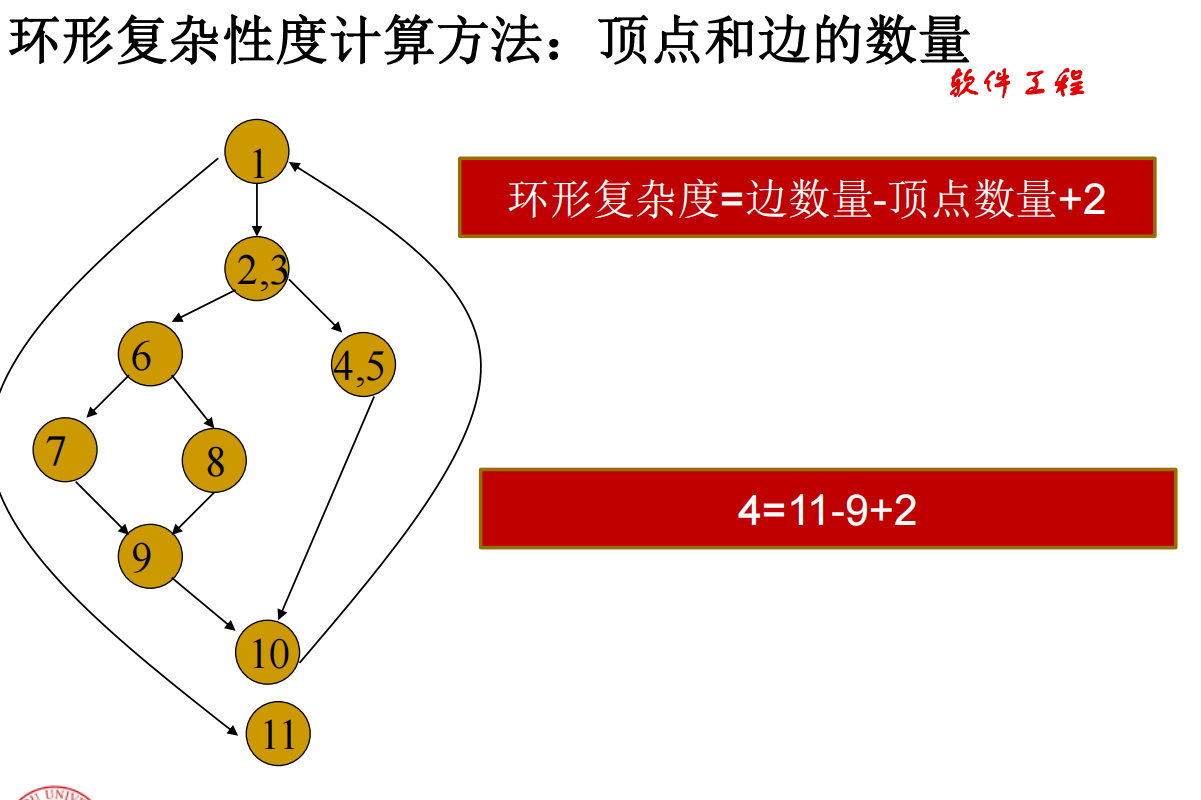
**四个步骤**

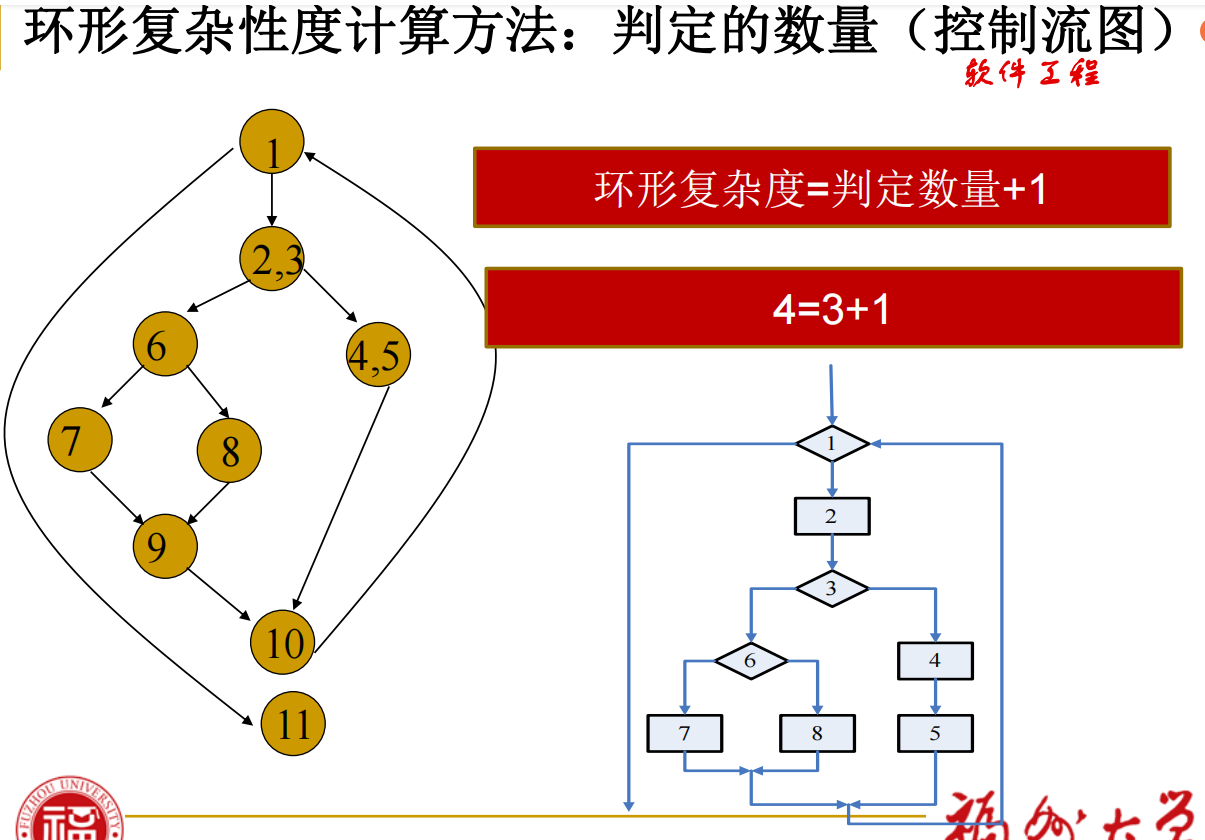


**❑ 程序逻辑复杂性：用【环形复杂度】评估**

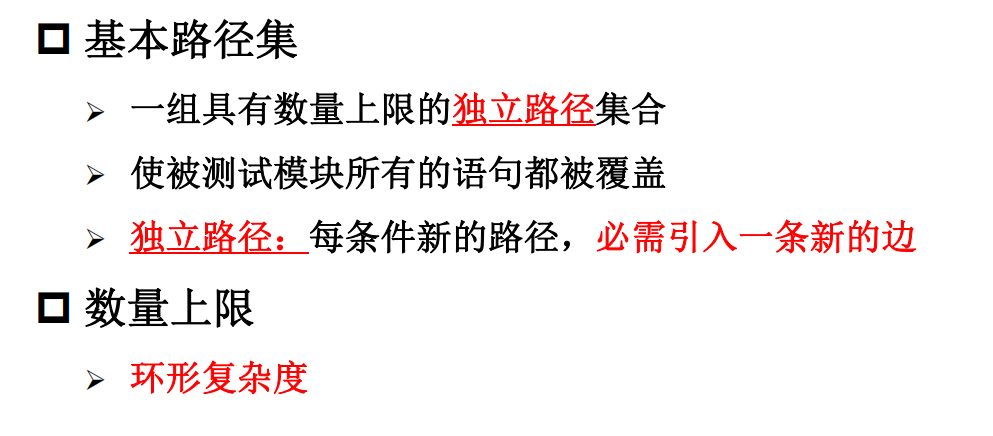


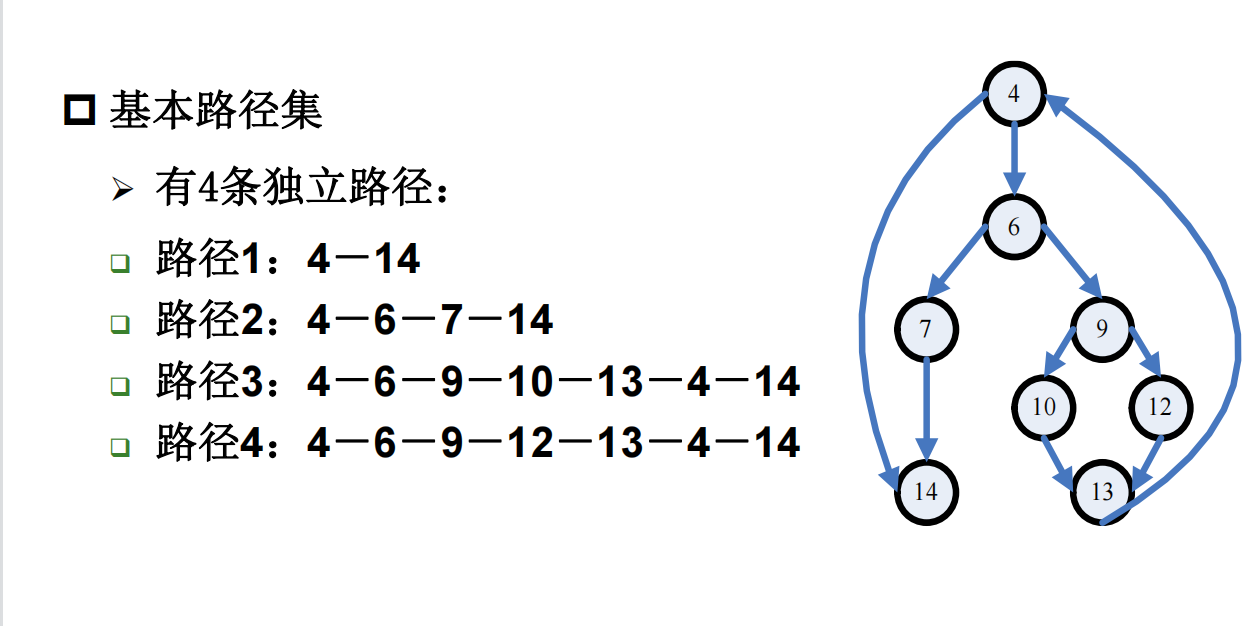




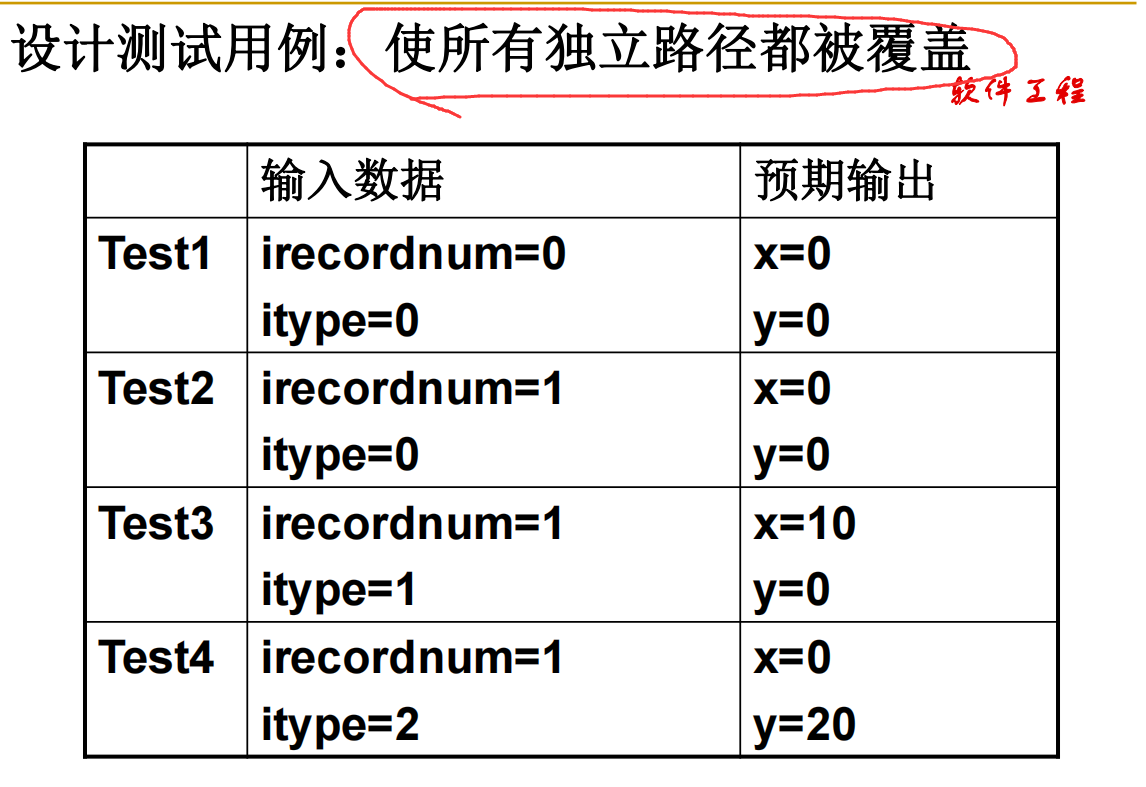


**❑ 基本路径集**





**❑ 设计测试用例：**

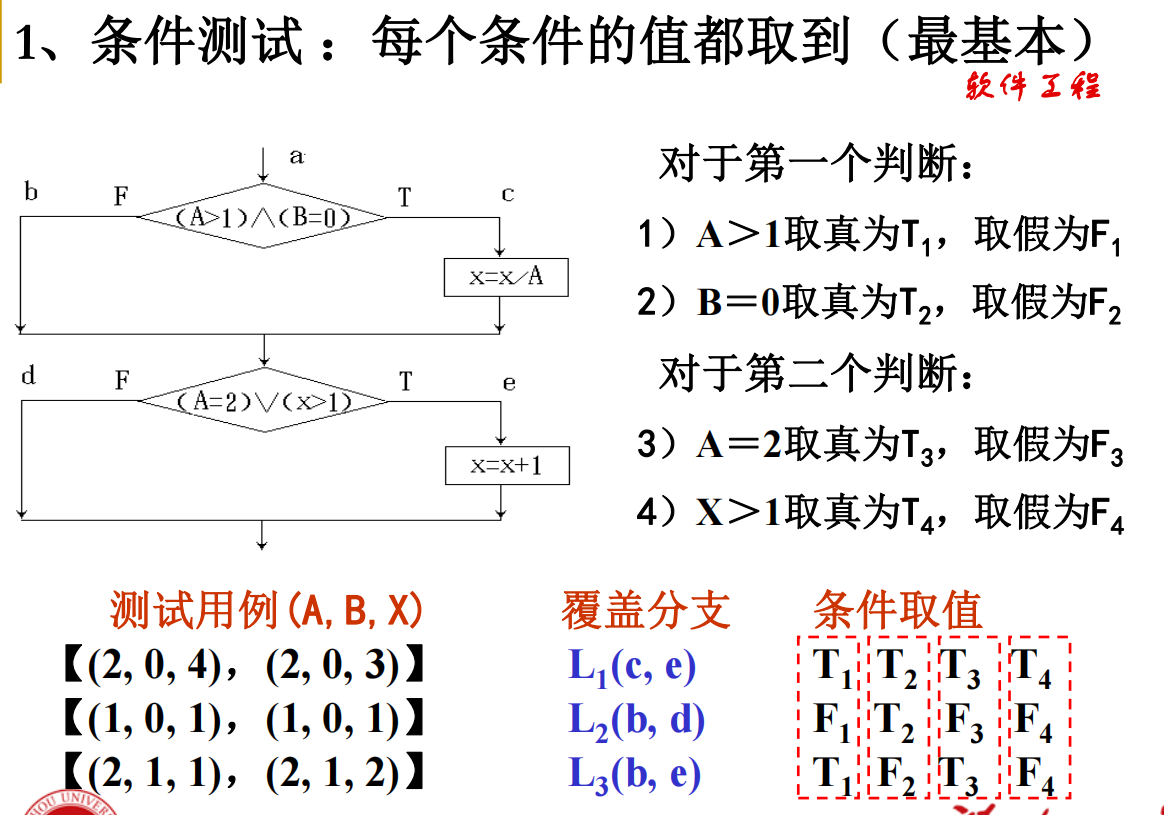


**19) 控制结构测试：**

** ❑ 条件测试：**

**➢ 基于模块中包含的逻辑条件，设计测试用例**

**➢ 每个判定的每个条件的可能取值至少执行一次**

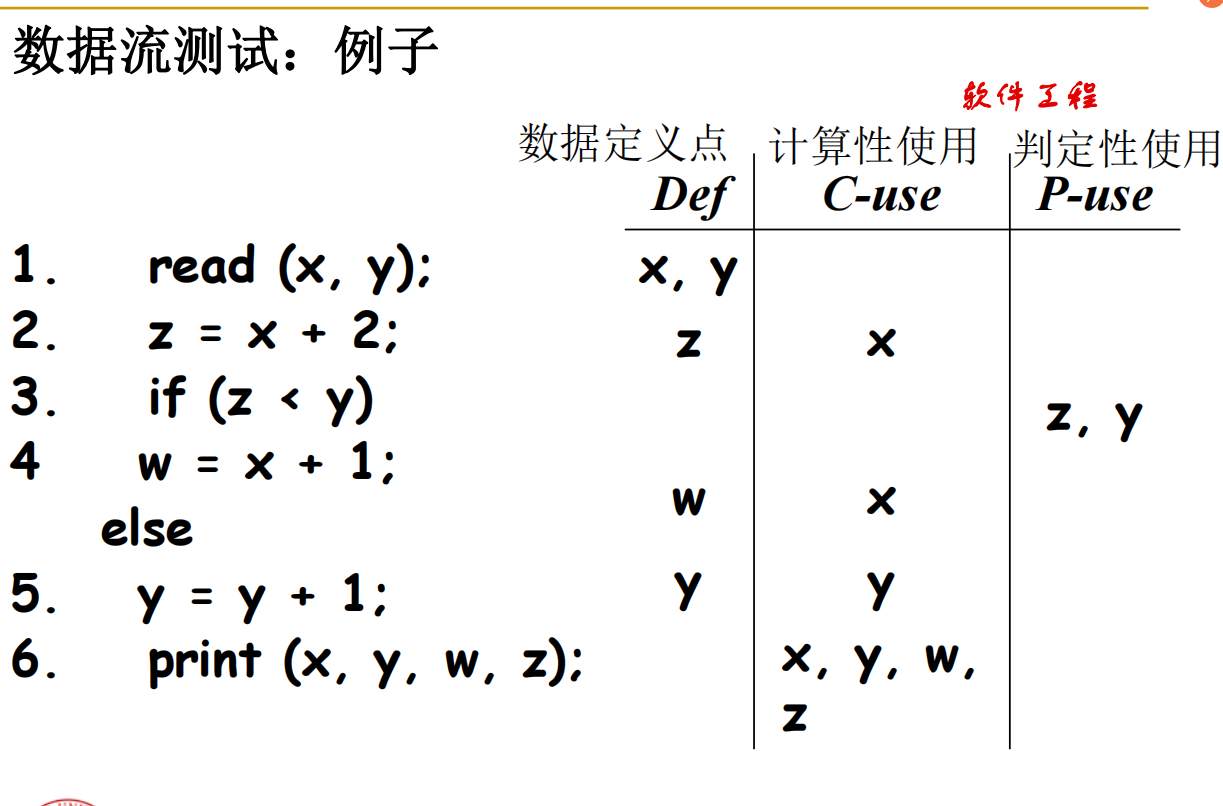


** ➢基本测试+语句测试：附加条件：保证每条语句至少执行一次**

**➢基本测试+分支测试：保证每个分支至少执行一次**

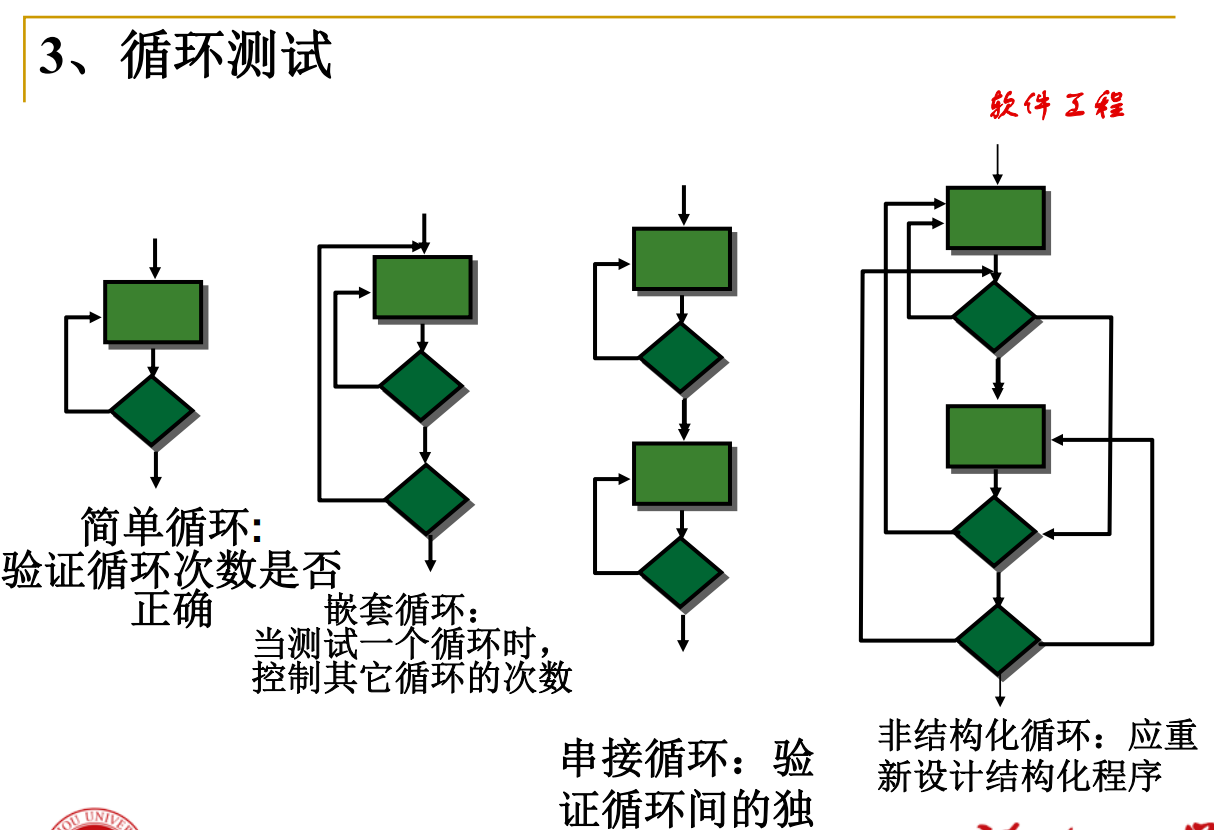
**❑ 数据流测试：**

**➢ 根据变量的定义和使用位置来选择测试路径**



** ❑ 循环测试：**

**➢ 侧重于循环的有效性**



**18) 功能测试-黑盒测试策略：把测试对象看做一个黑盒子，测试人员不考虑程序内部的逻辑结构和内部特性，只依据程序的需求规格说明书，检查程序的功能是否符合它的功能说明。-【功能测试】**

**应用：**

**➢ 集成测试**

**➢ 确认测试**

**➢ 系统测试**