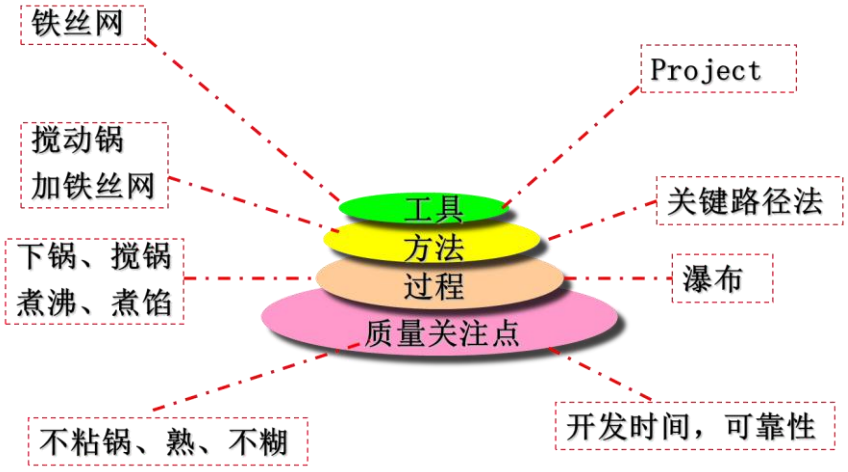
CH01 软件和软件工程

1. 软件的特点：P3
   软件是设计开发的，而不是传统意义上生产制造的；
   软件不会磨损，但是会退化；
   多数软件仍是根据实际顾客需求定制的；
   在软件设计中，大规模的复用才刚刚开始。
2. IEEE 对软件工程的定义：P7
   将系统化的、规范的、可量化的方法应用于软件的开发、运行和维护，即将工程化方法应用于软件；上述方法的研究。
3. 软件工程层次图：P8



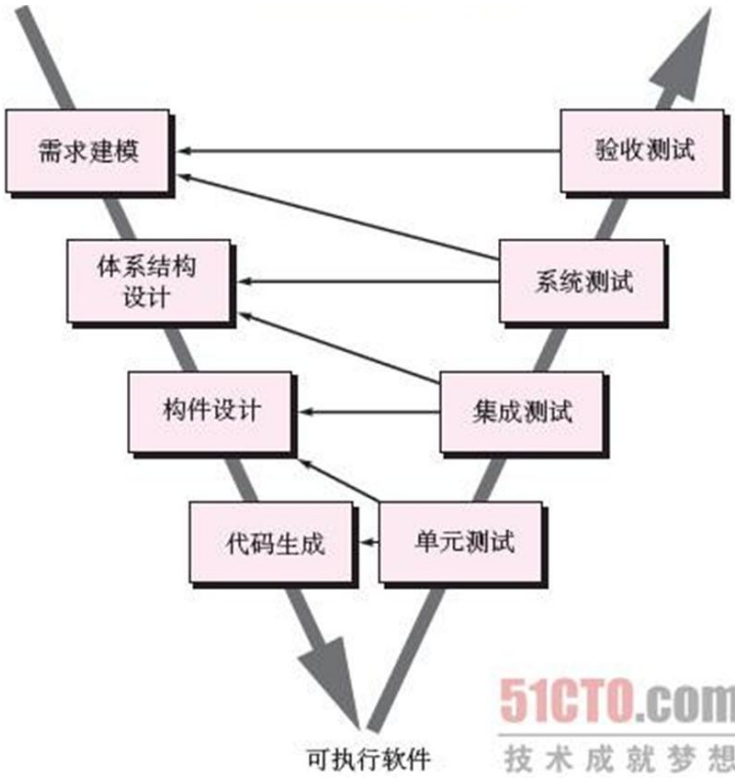
4. 通用软件工程过程框架包括 5 个活动：沟通；策划；建模；构建；部署。
5. 软件神话（为什么是错的？）P13
   1) 管理神话
      - Myth: 我们已经有了一本写满软件开发标准和规程的宝典。它无所不包，囊括了我们可能问到的任何问题。
        Realty: 这本宝典也许的确已经存在，但不能保证它已在实际中采用、反映了软件工程的现状、可以适应不同应用环境、在缩短交付时间的同时还关注保证产品的质量等等。
      - Myth: 如果我们未能按时完成计划，我们可以通过增加程序员人数而赶上进度。
        Reality: 新人加入一个软件项目后，原有开发人员必须牺牲本来的开发时间对新人进行培训，减少了应用于高效开发的时间。
      - Myth: 如果我将一个软件外包给另一家公司，则我可以完全放手不管。
        Reality: 如果开发团队不了解如何在内部管理和控制软件项目，那在外包项目中都会遇到困难。
   2) 客户神话
      - Myth: 有了对项目目标的大概了解，便足以开始编写程序，可以在之后的项目开发过程中逐步充实细节。
        Reality: 虽然通常难以得到综合全面且稳定不变的需求描述，但对项目目标模糊不清的描述将为项目实施带来灾难，只有客户和开发人员之间保持持续有效的沟通才能得到清晰的需求描述。
      - Myth: 虽然项目需求不断变更，但是因为软件是弹性的，因此可以很容易地适应变更。
        Reality: 软件需求的变更引入时机不同，变更造成的影响也不同，如提出的较早，则费用影响较小，但随着时间的推移，变更的代价也迅速增加，因为资源已经分配，设计框架已经建立，此时变更可能会引起剧变，需要添加额外的资源或修改主要设计架构。
   3) 从业者神话
      - Myth: 当我们完成程序并将其交付使用之后，我们的任务就完成了。
        Reality: 软件首次交付顾客使用之后花费的工作更多。
      - Myth: 对于一个成功的软件项目，可执行程序是惟一可交付的成果。
        Reality: 软件配置包括很多内容，可执行程序只是其中之一。各种工作产品（如模型、文档、计划）是

成功实施软件工程的基础，为软件技术支持提供了指导。

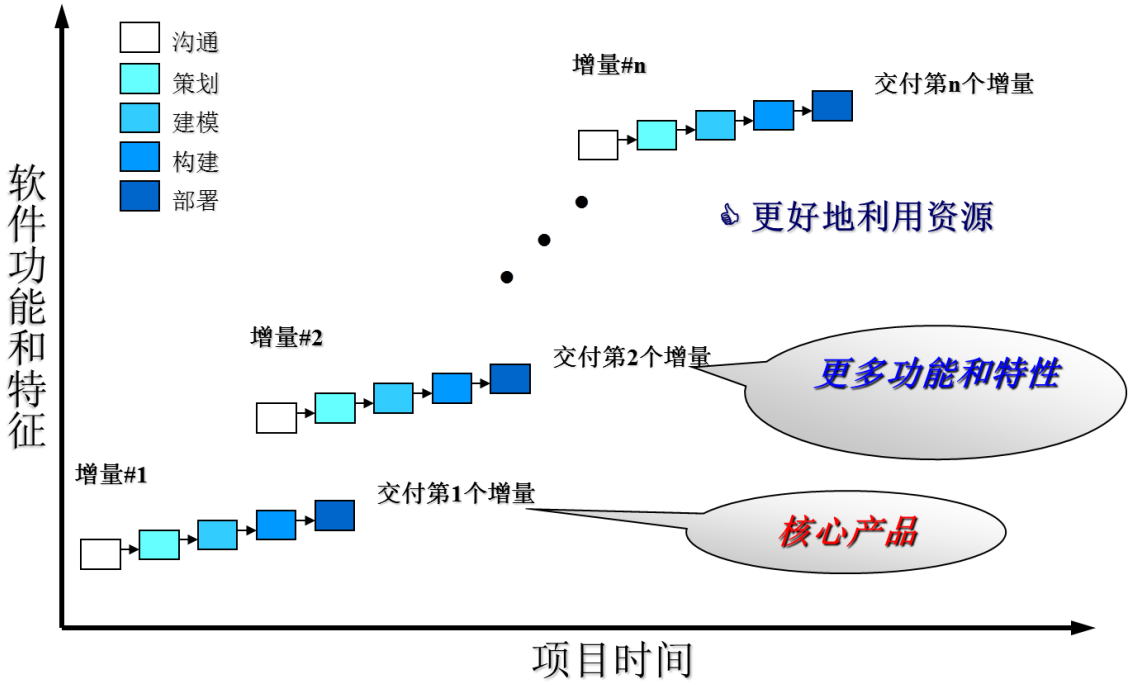
- Myth: 软件工程将导致我们产生大量无用文档，并因此降低工作效率。
  Reality: 软件工程并非以创建文档为目的，而是为了保证软件产品的开发质量。

CH02 过程模型

1. 瀑布模型 P24
   1) 经过的活动：
      - 沟通：项目启动、需求获取
      - 策划：项目估算、进度计划、项目跟踪
      - 建模：分析、设计
      - 构建：编码、测试
      - 部署：交付、支持、反馈
   2) 优缺点：
      - 它提供了一个模板，使得分析、设计、编码、测试与维护工作可以在该模板的指导下有序地展开，避免了软件开发、维护过程中的随意状态。
      - 对于需求确定、变更相对较少的项目，线性顺序模型仍然是一种可以考虑采取的过程模型。采用这种模型，曾经成功地进行过许多大型软件工程的开发。
   3) 存在问题：（为什么瀑布模型有时候会失效？）
      - 实际项目很少严格遵守该顺序。
      - 客户通常难以清楚描述所有需求。
      - 只有到项目接近尾声时，才有可执行程序。
2. V 模型：瀑布模型的变体
   1) 经过的活动：



   2) 优缺点：强调反馈，不将问题留到下一步。
   3) 存在问题：
3. 增量模型：
   1) 经过的活动：



2) 优缺点：

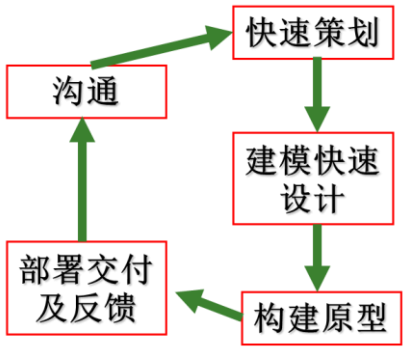
- 优点：
  - 能在较短时间内向用户提交可完成部分工作的产品
  - 用户有较充裕的时间学习和适应新产品
  - 易于保证核心功能正确
  - 可以基于早期版本来获取需求
  - 项目完全失败的风险小。
  - 可以为那些创新的功能开拓市场。
  - 规避了资源缺乏的风险
- 缺点：
  把用户需求转化为功能递增的不同版本可能比较难
  难以确定所有版本共需的公用模块。

3) 存在问题：

- 如果你的客户要求你在一个不可能完成的时间提交产品，向他建议只提交一个或几个增量，此后再提交软件的其他增量。
- 增量和原型的比较：
  任何增量的处理流程均可结合原型实现；增量模型与原型相同，本质上都是迭代的；增量模型强调每个增量都是可操作的；早期的增量是整体的一部分，而原型最终要被抛弃。

4. 原型开发：

1) 经过的活动：



2) 优缺点：
优点：用户能够感受到实际系统；开发者能很快建造出一些东西。

3) 存在问题：

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。

- 客户提出了一些基本功能，但未详细定义输入、处理和输出需求；开发人员可能对开发运行环境、算法效率、操作系统的兼容性和人机交互等情况不确定。这些情况下，采用原型开发。
- 对于要求把一个粗糙的原型系统变为工作产品的压力，建议尽量抵制。这样做的结果往往是产品质量受到损害。

5. 螺旋模型

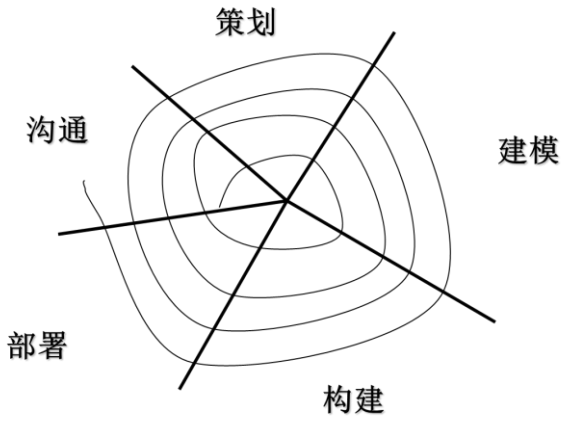
1) 经过的活动：

策划：项目估算、制定进度计划、风险分析；

建模：分析、设计；

构建：编码、测试；

部署：交付、反馈。



2) 优缺点：

- 随着过程进展演化，开发者和客户能够更好地理解和对待每个演化级别上的风险，适合于大型系统及软件的开发；
- 使用原型实现作为降低风险的机制，并在开发的任意阶段均可使用原型实现；
- 更真实的反映了现实世界；
- 如应用得当，能在风险变成问题之前降低它。
- 模型的成功依赖于风险评估的专门技术；
- 是一个较新的模型，功效的确定尚需若干年时间。

3) 存在问题：

6. 协同模型：了解概念

7. 各种模型的比较

模型	优点	缺点
瀑布模型	规范，文档驱动	系统可能不满足客户真正的需求
快速原型	克服了瀑布型的缺点	
增量模型	开发早期回报明确，易于维护	要求开放的软件体系结构
螺旋模型	风险驱动，适用于大型项目开发	风险分析人员需要有经验且经过充分训练

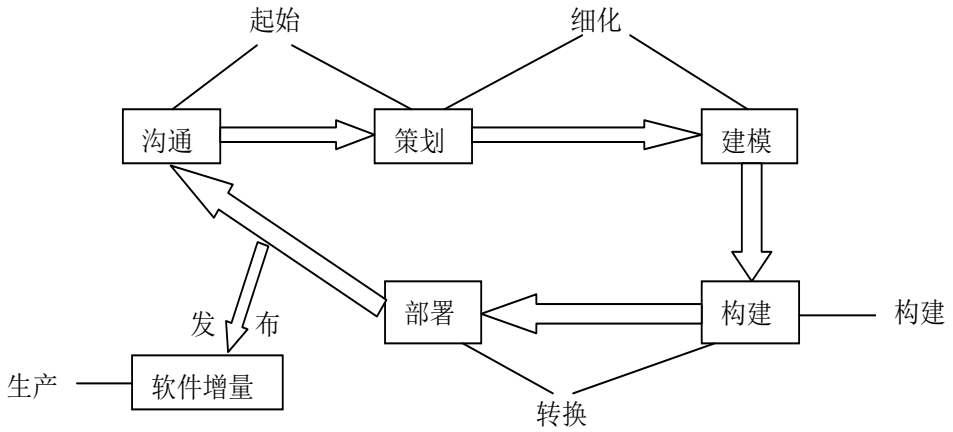
8. 统一过程

1) 宗旨：用例驱动，以架构为核心，迭代并且增量。

2) 统一过程的阶段：

- 起始阶段包括客户沟通和策划活动。
- 细化阶段包括沟通和通用过程模型的建模活动。
- 构建阶段与通用软件过程中的构建活动相同。

- 转换阶段包括通用构建活动的后期阶段以及通用部署（交付和反馈）活动的第一部分。
- 生产阶段与通用过程的部署活动一致。
- 5 个过程并非顺序进行，而是阶段性并发进行。



课后作业：2.3　2.9　2.15

练习

应该采用什么过程模型？

（1）客户不太清楚待开发的系统需要提供什么服务

（2）开发团队了解待开发软件的相关领域知识，尽管此系统庞大，但其较已经开发的系统差异并不大。

（3）软件的功能是把读入的浮点数开平方，所得到的结果应该精确到小数点后 4 位。

（4）开发一个已发布软件的新版本，公司规定了严格的完成期限，并对外公布。

（5）汽车防锁死刹车控制系统

（6）大学记账系统，准备替换一个已存在的系统

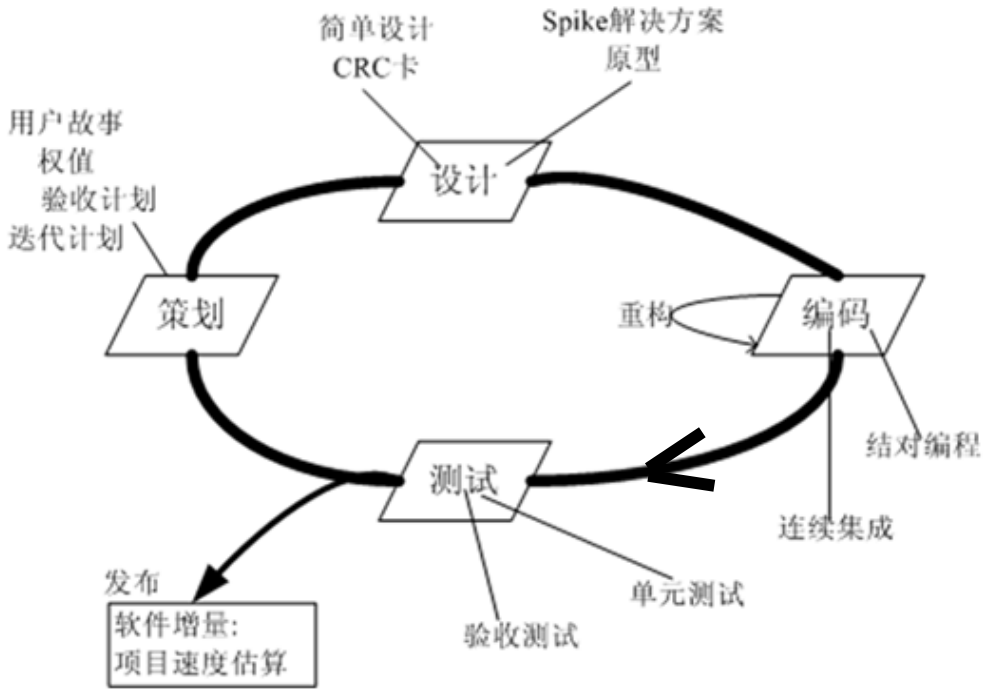
（7）一个位于火车站的交互式火车车次查询系统

答案

(1)原型　（2）瀑布　　（3）瀑布　　（4）增量　（5）螺旋　（6）瀑布　　（7） 原型

CH03 敏捷开发

1. 敏捷的概念：它是一种软件开发方法论，可以应对客户快速变更的需求。它强调以人为核心，采用迭代的方式，循序渐进的开发软件。特别适用于 web 应用开发。
2. 敏捷原则：
   - 通过尽早、持续交付有价值的软件来使客户满意
   - 即使在开发后期，也欢迎需求变更
   - 经常交付可工作软件
   - 业务人员和开发人员必须在一起
   - 围绕受激励的个人构建项目
   - 最有效的信息传递方法是面对面交谈
   - 可工作软件是进度的首要度量标准
   - 提倡可持续的开发速度
   - 不断关注优秀的技能和设计
   - 简单是必要的
   - 好的架构、需求和设计出于自组织团队
   - 定期反省，并相应调整自己的行为
3. 敏捷的特点：
   允许项目团队调整并合理安排任务；理解敏捷开发方法的易变性并制定计划；精简并维持最基本的工作产品；强调增量交付策略；快速向用户提供适应产品和运行环境的可运行软件。
4. 极限编程 XP



1) 策划：
   - 建立一系列描述待开发软件必要特征与功能的“故事”
   - 评估每一个故事（即优先级），并给出以开发周数为度量单位的成本
   - 客户和 XP 团队共同决定如何把故事分组并置于将要开发的下一个发行版本中（下一个软件增量）
   - 形成关于一个发布版本的基本承诺
   - 在第一个版本发布之后，XP 团队计算项目的速度。
     - 对待开发的故事排序方法：所有选定故事在几周内尽快实现；具有最高权值的故事移到进度表前面先实现；高风险故事将首先实现。
     - 项目速度：第一个发行版本中实现的客户故事个数。用于后续发行版本的发布日期和进度安排，调整软件发型内容和最终交付日期。
2) 设计
   - 严格遵循 KIS(Keep it simple)原则
   - 鼓励使用 CRC 卡，CRC（类-责任-协作者）卡确定和组织与当前软件增量相关的面向对象的类。
   - 在某个故事设计中遇到困难时，立即建立这部分设计的可执行原型，实现并评估设计原型（被称为 Spike 解决方案)
   - 鼓励“重构”，重构是以不改变代码外部行为（如函数的输入输出）而改进内部结构（如函数实现方法）的方式来修改软件系统的过程。重构实质就是在编码完成之后改进代码设计。
3) 编码
   - 在编码之前，确定检测本次发布的所有故事的单元测试
   - 鼓励结对编程，这提供了实时解决问题和实时保证质量的机制。
   - 连续集成策略，有助于避免兼容性和接口问题。
4) 测试
   - 每天进行测试
   - “验收测试” 由客户确定，根据本次软件发布中所实现的用户故事而确定。

5. 工业极限编程 IXP
6. 其他敏捷过程模型 P49（了解）

练习：

1、在一个信息系统组织中，你被指派为项目管理者。你的工作是建造一个应用，类似于你的小组以前已经做过的项目，虽然这个规模更大且更复杂一些。需求已由用户写成文档。你会选择哪种软件过程模型？为什么？

2、你被指派为一个大型软件产品公司的项目管理者。你的工作是管理该公司已被广泛使用的字处理软件的新版本的开发。因为竞争激烈，已经规定了严格的期限并对外公布。你会选择哪种软件过程模型？为什么？

3、在一个为遗传工程领域服务的公司中，你被指派为项目管理者。你的工作是管理一个软件产品的开发，该产品

能够加速基因分类的速度。这项工作是面向研究及开发（R&D）的，但其目标是在下一年内生产产品。你会选择哪种软件过程模型？为什么？

4、你被指派为一个小型软件产品公司的项目管理者。你的工作是建造一个具有突破性的产品，该产品结合了虚拟现实的硬件和高超的软件。因为家庭娱乐市场的竞争非常激烈，完成这项工作的压力很大，你会选择哪种软件过程模型？为什么？

答案：

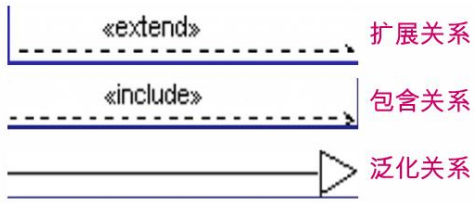
1、 RID(快速模型)或瀑布或增量　　2、 增量或 XP

3、 原型或螺旋　　4、 增量或 XP

CH04/05/06 需求工程

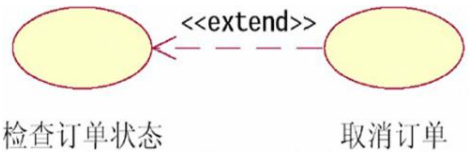
1. 需求工程过程通过执行 7 个活动来实现：起始、导出、精化、协商、规格说明、确认、管理。
   起始阶段的工作：确认共同利益者、识别多种观点、协同合作、Q&A 会议 P68.
2. 用例文本 P74
3. 用例图 P76　　P93

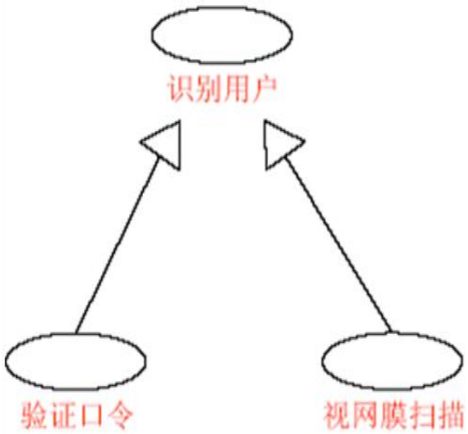
   用例的关系：

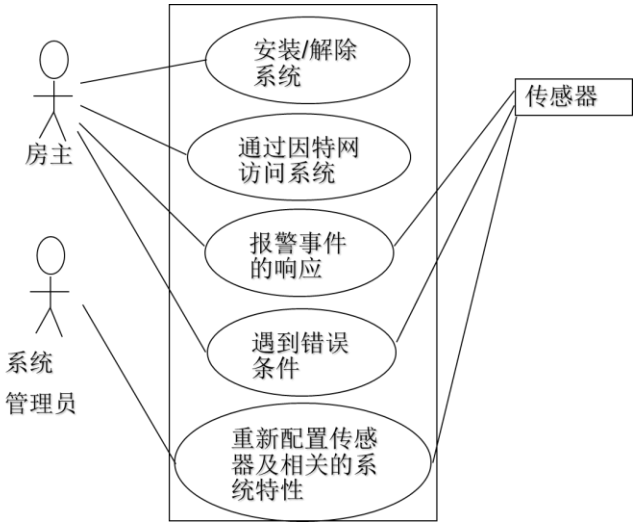


泛化

扩展

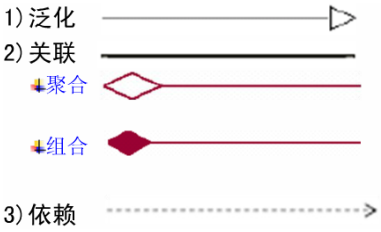






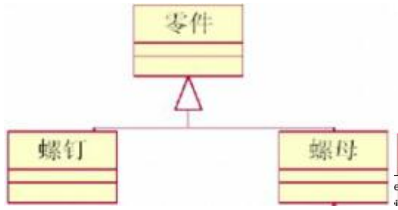
4. 活动图 P94、类图 P77　P102

   类的关系：

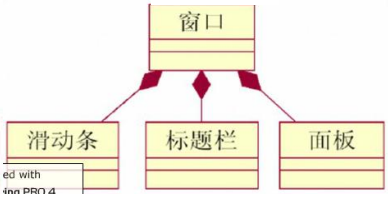


关联

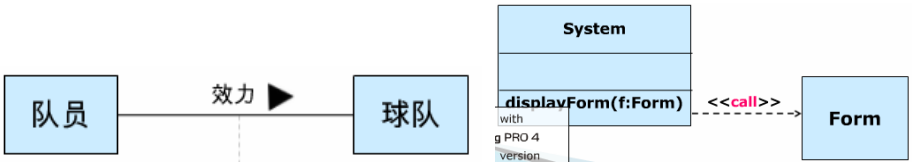
继承和泛化



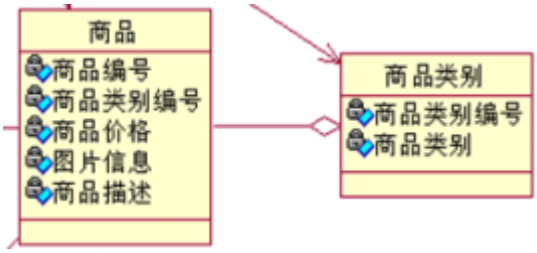
组合

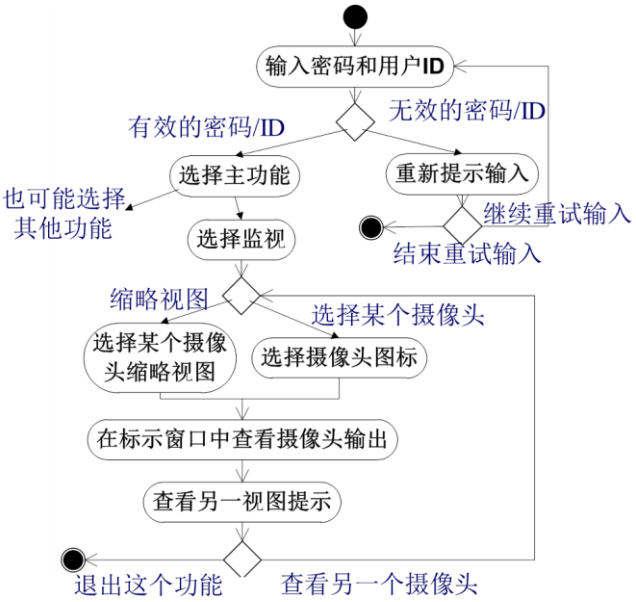


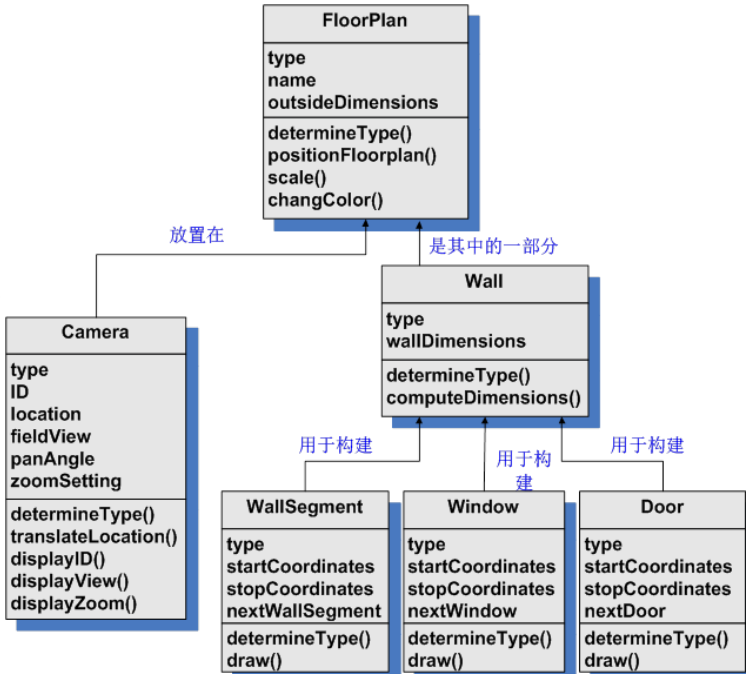
依赖



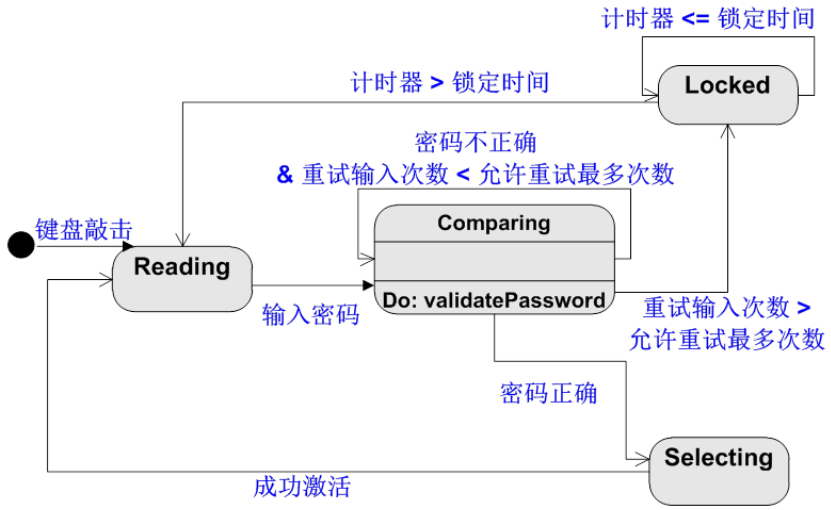
聚合



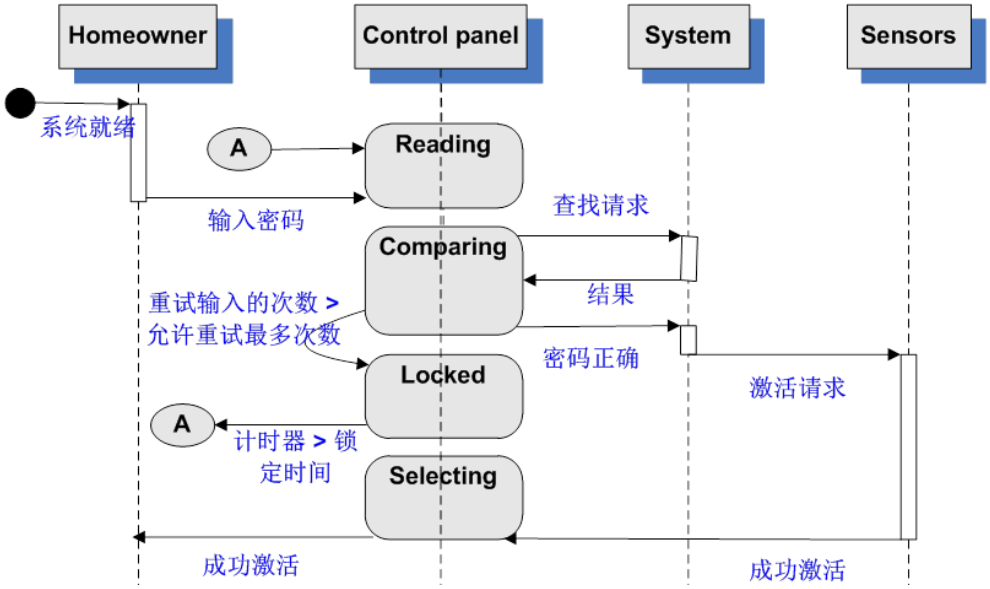




5. 用例模板 P92
6. 识别分析类 P98　　语法分析 P99
7. CRC 建模 P103，提供了一个简单方法，用于识别和组织与系统或产品需求相关的类。CRC 模型实际上是表示类的标准索引卡片的集合，这些卡片分成三部分：
   - 顶部写类名
   - 下面左侧列出类的职责，职责：和类相关的属性和操作
   - 下面右侧列出类的协作关系，协作：信息请求或动作请求
8. 数据流图 DFD　P112　　分层的数据流图
9. 状态图 P119



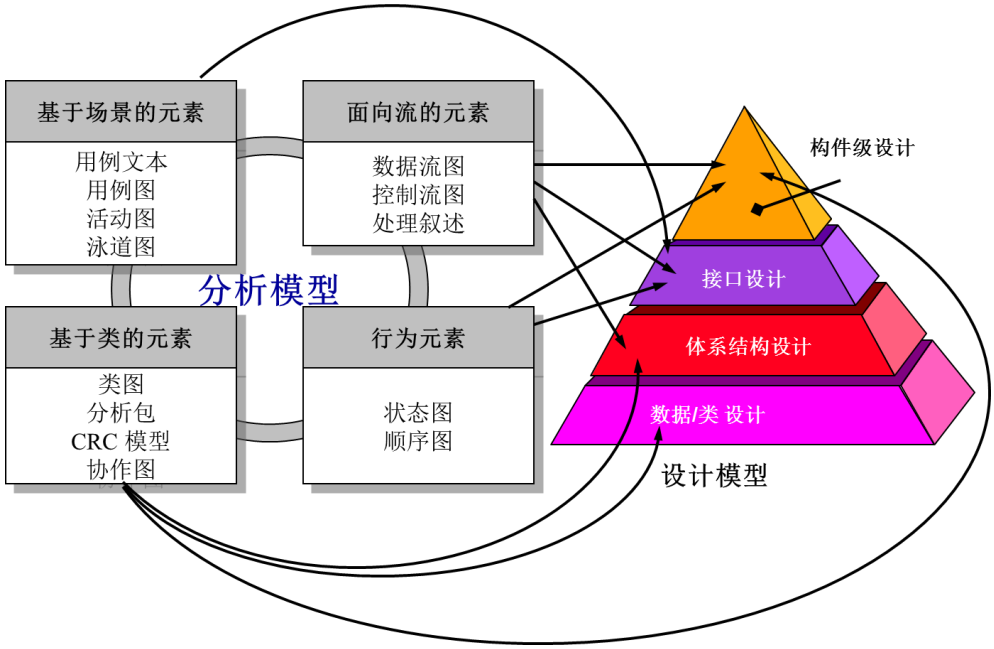
10. 顺序图 P120



课后作业：4.9　5.6

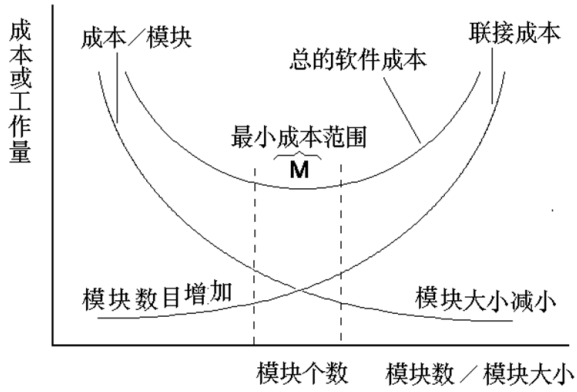
CH07 设计概念

1. 需求模型到设计模型的转换



2. 模块化 P133：将系统划分为相对独立但又有所关联的多个部分。
   1) 按照设计原则将系统划分为若干个较小的模块
      - 相互独立但又相互关联
      - 实际上是系统分解和抽象的过程
   2) 模块是相对独立的程序体
      - 是数据说明、可执行语句等程序对象的集合
      - 单独命名的，并且可以通过名字来访问

        例如：类、过程、函数、子程序、宏等
   3) 模块化就是把程序划分成独立命名且可直接访问的模块，各个模块完成一个子功能，这些模块集成起来构成一个整体，完成指定功能。
   4) 对于一个给定的系统，合适的模块数量是多少？



结论：适度的模块化

- 模块数增加时，模块间的关系也随之增加，接口和集成的工作量也随之增加
- 结论：寻找最佳模块化程度平衡点

5) 模块化设计的好处：模块化设计（以及由其产生的程序）使开发工作更易于规划；可以定义和交付软件增量；更容易实施变更；能够有效的开展测试和调试；可以进行长期维护而没有严重的副作用。

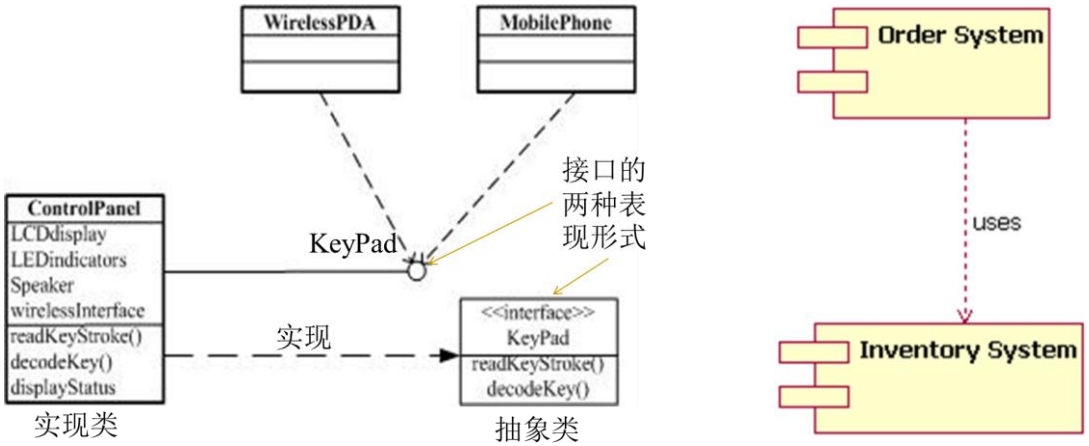
3. 信息隐蔽 P134
   1) 每个模块都尽量对其他模块隐藏自己的内部实现细节
      - 模块内部的数据和过程不允许其它不需要这些信息的模块使用
      - 定义和实施对模块的过程细节和局部数据结构的存取限制
      - 典型的信息隐藏：面向对象的访问控制符
   2) 信息隐藏是实现抽象/模块化机制的基本支撑
   3) 信息隐蔽的目的：将数据结构和处理过程的细节隐蔽在模块接口之后。用户不需要了解模块内部的具体细节。
4. 功能独立 P134：功能独立是模块化、抽象概念和信息隐藏的直接结果。
   1) 功能独立的表现：内聚度与耦合度

      内聚(cohesion)：一个模块内部各个元素彼此结合的紧密程度——尽量高

      耦合(coupling)：模块之间相互关联的程度——尽量低
   2) 功能独立的好处：

      易开发，易维护和测试，减少错误扩散，易复用。
5. 重构 P136：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。
6. 接口设计的 3 个重要元素：
   - 用户界面 (UI)
   - 和其他系统、设备、网络或其他的信息生产者或使用者的外部接口
   - 各种设计构件之间的内部接口.

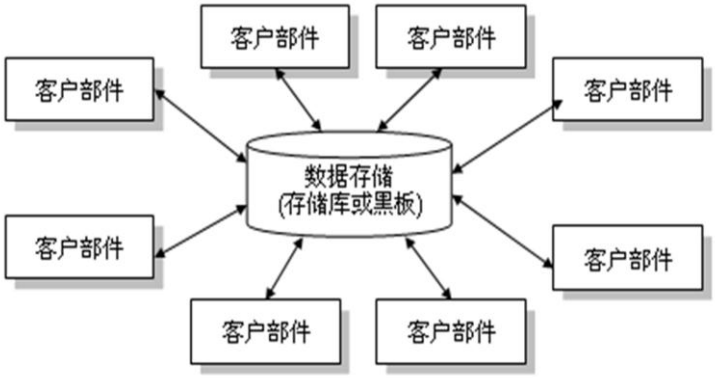
   1) 接口定义：是类、构件或其他分类的外部可见的（公共的）操作说明，而没有内部结构的规格说明。包括：一组描述类的部分行为的操作和提供操作的访问方法。
   2) 接口表示：



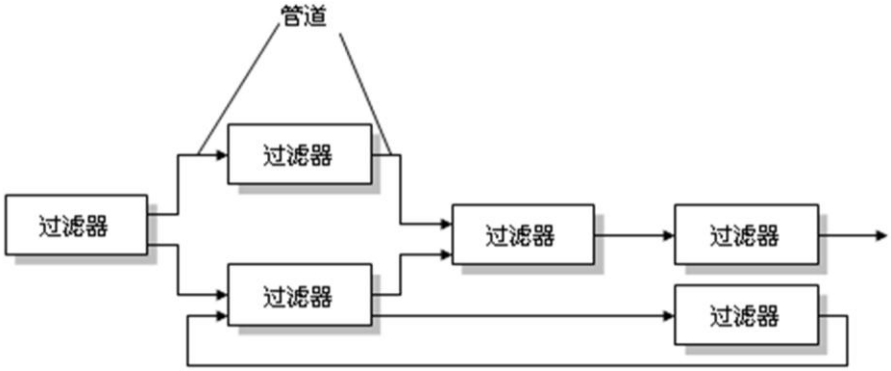
7. UML 构件图 P142:右上

CH08 体系结构设计

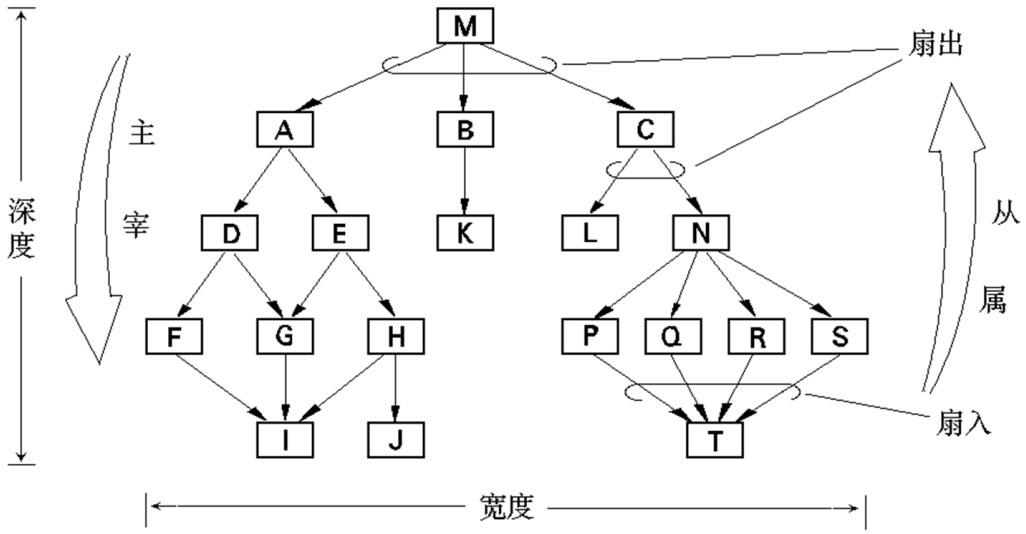
1. 体系结构概念：系统的一个或多个结构，它包括软件构件、这些构件对外可见的属性以及它们之间的相互关系。
2. 体系结构风格（会区分）
   1) 以数据为中心的体系结构
      适用于可重用构件库、大型数据库、搜索引擎等。



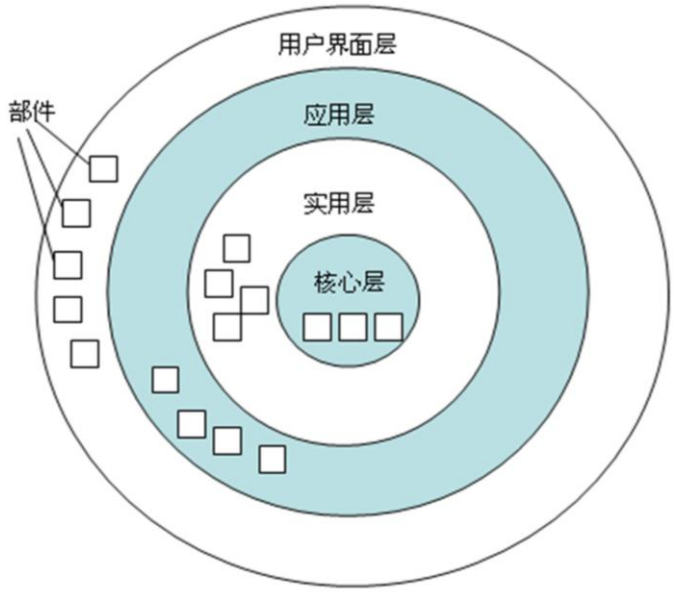
   2) 数据流体系结构



   3) 调用和返回体系结构
      适用于分组交换网络中或给予活动者的系统中。

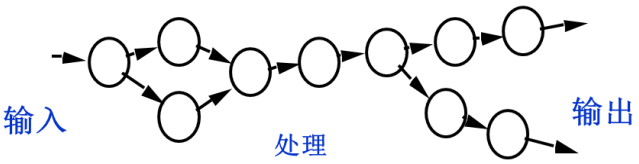


   4) 面向对象体系结构
   5) 层次体系结构



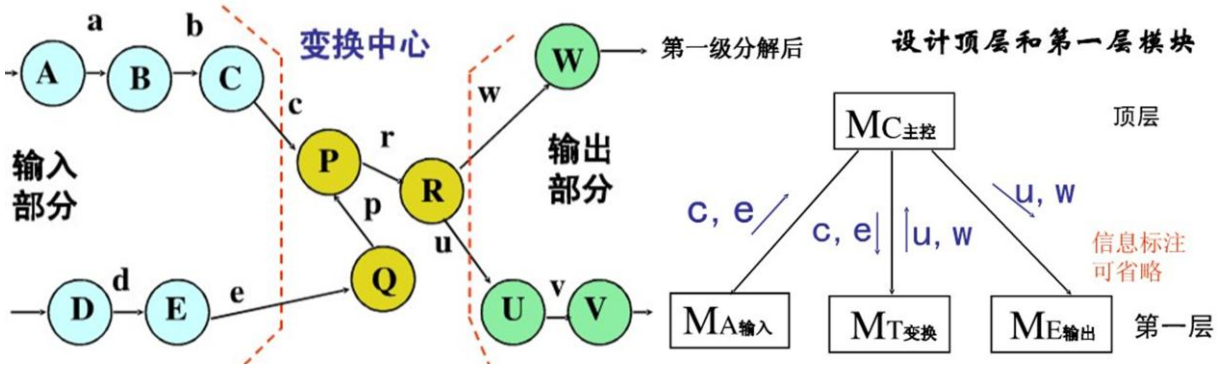
3. 使用数据流进行体系结构映射

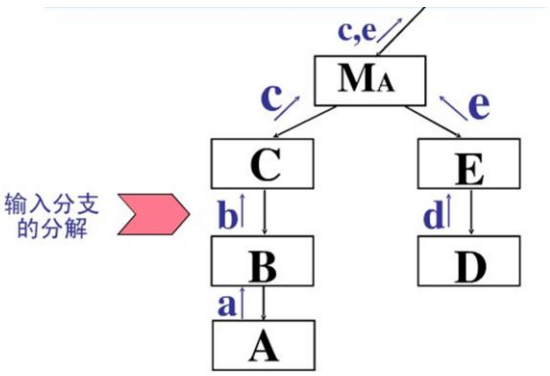
1) 变换型数据处理：

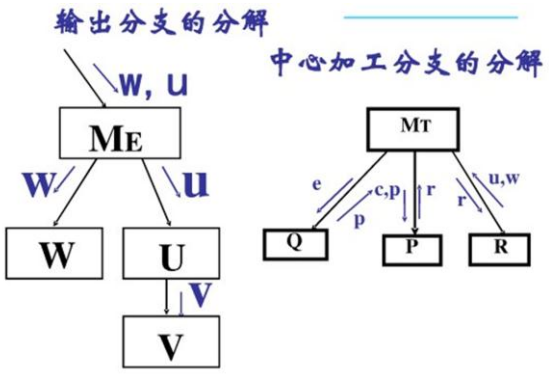


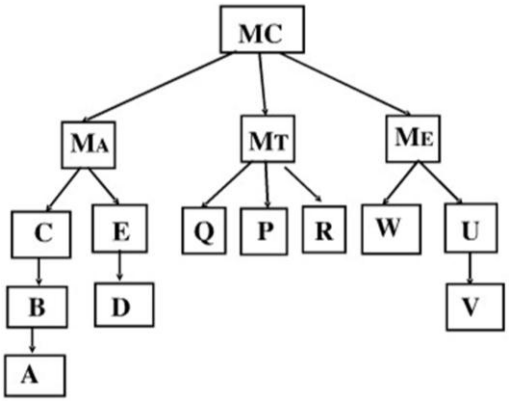
步骤：

(1)区分输入、变换中心输出部分，在 DFD 上标明分界线

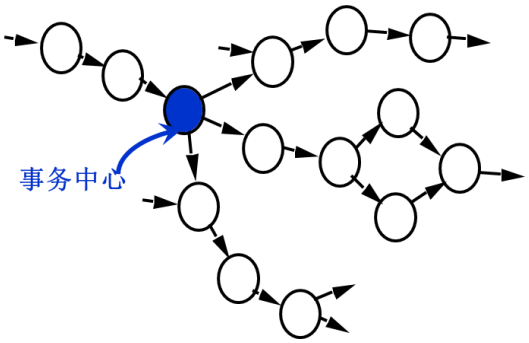




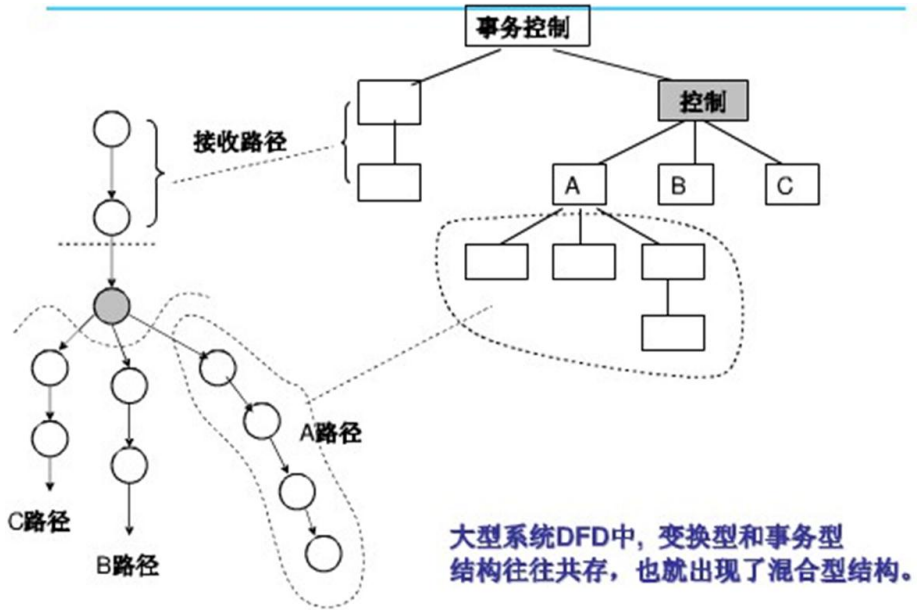




2) 事务型数据处理：



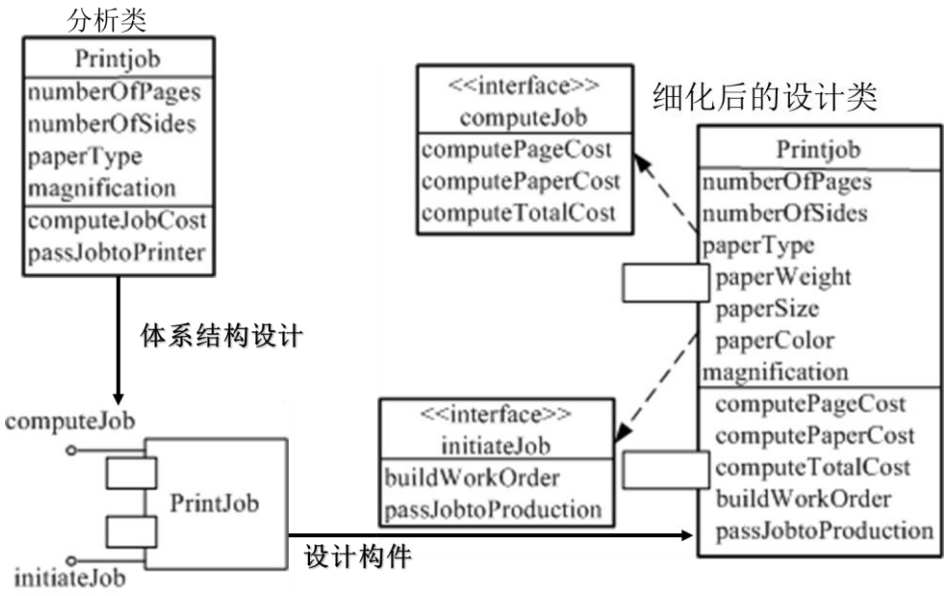
事务分析设计方法步骤：
(1) 精化DFD
(2) 确定确定信息流(DFD)类型
(3) 在DFD上确定事务中心、标识事务中心和每条路经上的流特征。
(4) 将DFD映射到一个适合于进行事务处理的程序结构上。
(5) 精化该事务结构和每条动作路径上的结构。
(6) 精化模型结构。



3) 混合型数据处理：

CH09 构件级设计

1. 构件的概念：OMG 统一建模语言规范的定义：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。
   设计构件的细化 P170



2. 设计基于类的构件

1) 基本设计原则：

- 开闭原则：模块应该对外延具有开放性，对修改具有封闭性，即不应把变更写入代码，从而不必进入代码内部修改。
- Liskov 替换原则：子类可以替换它们的基类。用在有继承时判断有无滥用继承。
- 依赖倒置原则：依赖于抽象，而非具体实现
- 接口分离原则：多个用户专用接口比一个通用接口要好。降低接口间的耦合度。

2) 内聚性：一个模块（构件）内部各成分之间相互关联程度的度量。

级别排序：

- 功能内聚：一个模块内各功能部分都使用了相同的输入数据，或产生了相同的输出数据。
- 分层内聚：高层能访问低层的服务，但低层不能访问高层的服务。
- 通信内聚：访问相同数据的所有操作被定义在一个类中。
- 顺序内聚：将构件或操作按照前者为后者提供输入的方式组合实现了一个操作序列。
- 过程内聚：允许在调用前面的操作或构件之后马上调用后面的构件或操作，即使两者之间没有数据传递。
- 暂时内聚（时间内聚，temporal cohesion)：大多为多功能模块，但模块的各个功能的执行与时间有关，通常要求所有功能必须在同一时间段内执行。例如初始化模块和终止模块。
- 实用内聚（巧合内聚，偶然内聚)：一个模块执行多个完全不相关的动作。

3) 耦合性：构件之间联系的紧密程度，面向对象：类间联系的紧密程度

级别排序：P178

- 内容耦合：包括(1) 一个模块直接访问另一个模块的内部数据，如 GOTO 跳转;(2) 一个模块不通过正常入口转到另一模块内部;(3) 两个模块有一部分程序代码重迭(只可能出现在汇编语言程序中);(4) 一个模块有多个入口。
- 共用耦合（公共耦合)：大量构件都要使用同一个全局变量。
- 控制耦合：一个模块通过传送开关、标志、名字等控制信息，控制选择另一模块的功能。
- 标记耦合：类 B 被声明为类 A 某一操作中的一个参数类型。
- 数据耦合：操作需要传递长串的数据参数。
- 例程调用耦合：一个操作调用另一个操作。
- 类型使用耦合：构件 A 中使用了在构件 B 中定义的一个数据类型。
- 包含或者导入耦合：构件 A 引入或者包含一个构件 B 的包(java)或者内容(C 语言)时。
- 外部耦合：两个构件共享数据格式，通信协议或设备接口。

3. 决策表 P185

课后习题 9.1　9.5　9.6　9.9

练习：

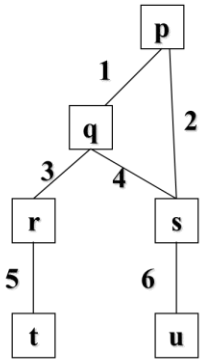
1、一组语句在程序的多处出现，为了节省内存空间把这些语句放在一个模块中，该模块的内聚度是_____的。

模块中所有成分引用共同的数据，该模块的内聚度是_____的。

模块内的某成分的输出是另一些成分的输入，该模块的内聚度是 _____ 的。

模块中所有成分结合起来完成一项任务，该模块的内聚度是 _____ 的。

2、试确定以下各个模块间的耦合类型？



编号	输入	输出
1	飞机类型	状态标志
2	飞机零件清单	
3	功能代码	
4	飞机零件清单	
5	零件编号	零件制造商
6	零件编号	零件名称

模块p,t,和u 访问同一数据库

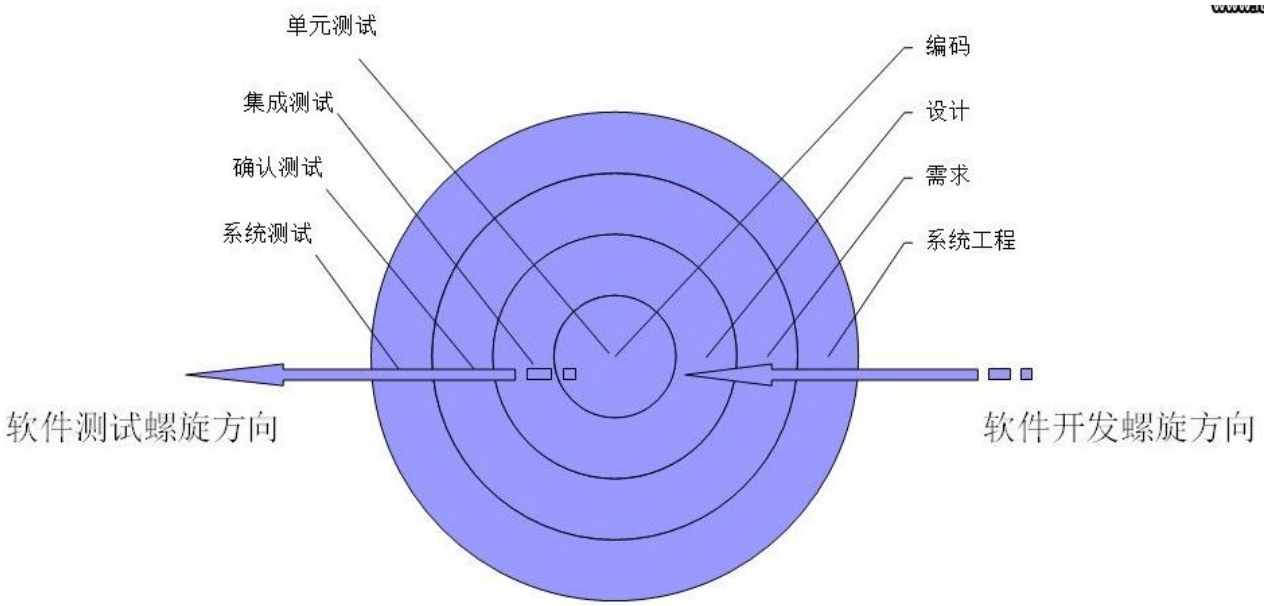
答案：

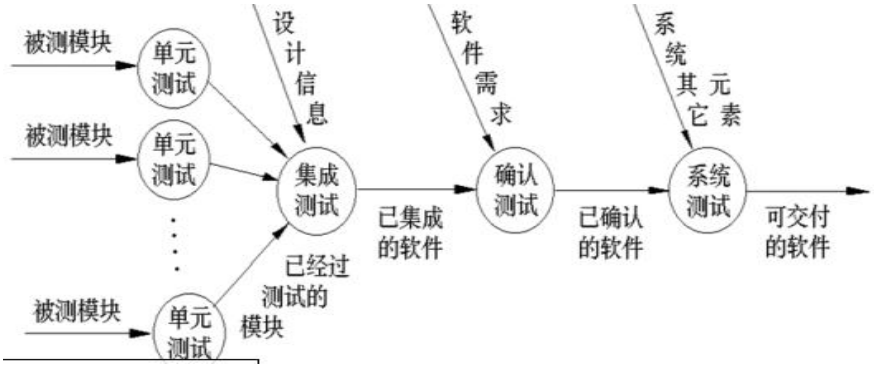
1、实用内聚，通信内聚，顺序内聚，功能内聚

2、PQ 数据耦合；PS 数据耦合（使用清单中所有元素）

CH14 软件测试策略

1. 测试策略（4 个阶段与开发阶段的对应关系）





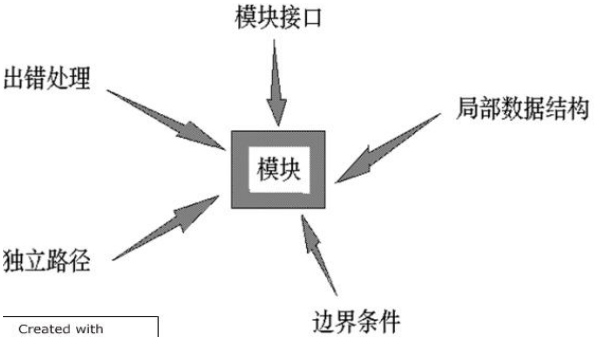
2. 什么时候完成测试？P260

1) 永远不能完成，测试的工作只会从软件工程师身上转移到最终用户身上。用户每次运行程序时，程序就在经受测试。

2) 当时间或资金不够时，测试就结束了。

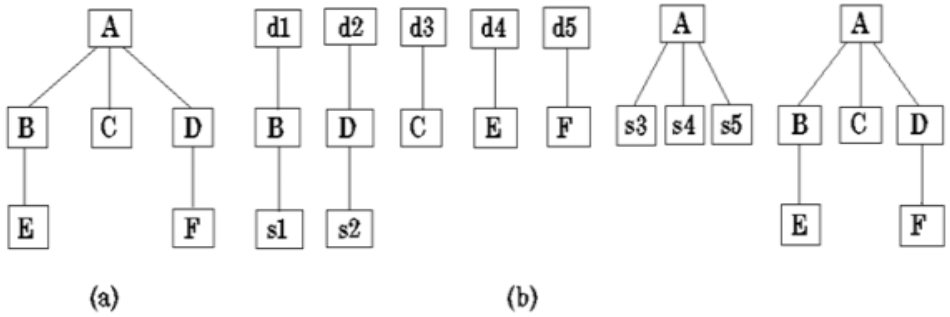
3) 利用统计建模和软件可靠性理论，建立以执行时间为函数的软件故障模型。

3. 单元测试的内容



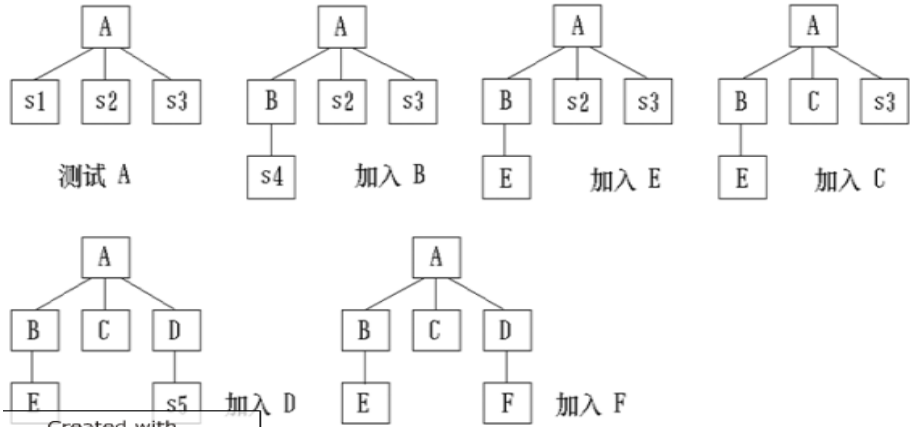
4. 集成测试

1) 一次性组装测试

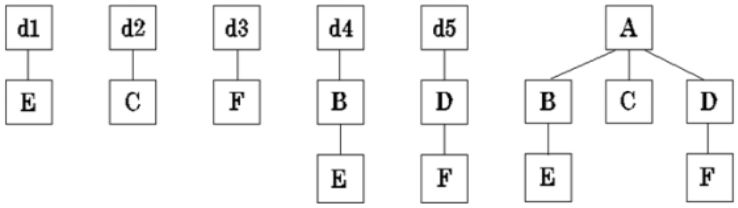


2) 深度组装测试/增值组装测试

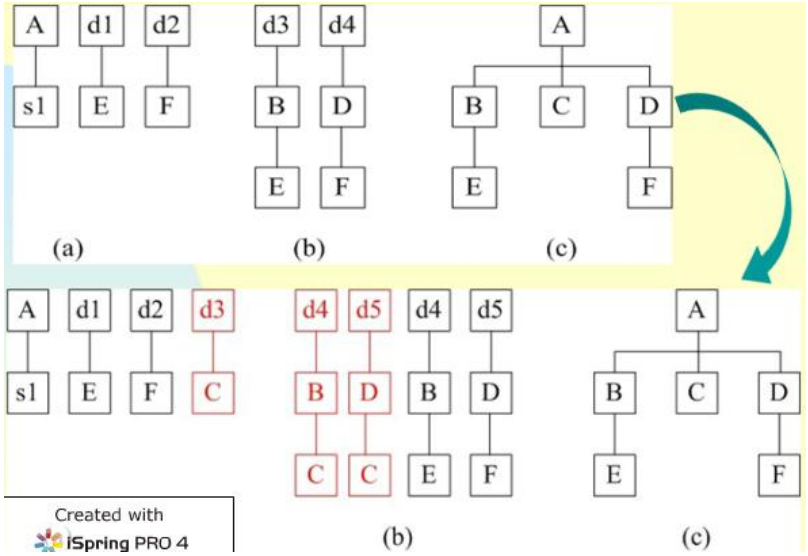
自顶向下

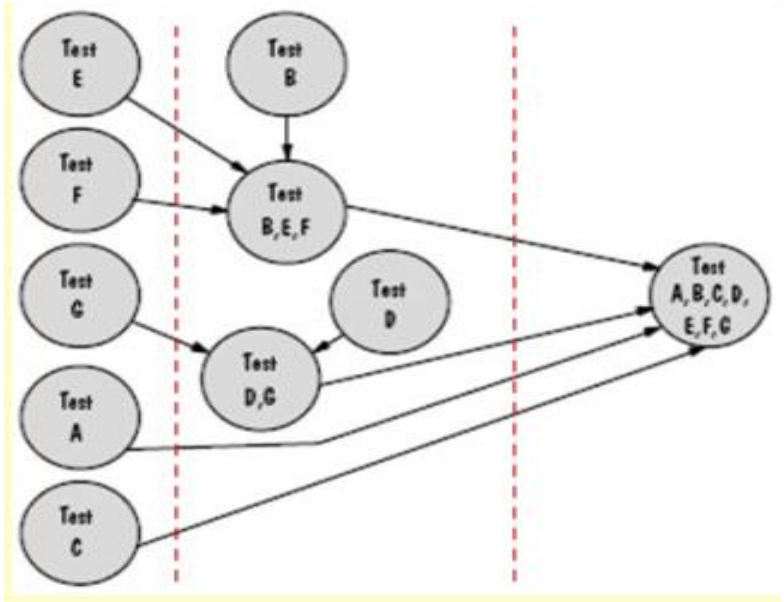


自底向上



3) 混合增值组装测试





4) 回归测试：重新执行已进行测试的某个子集，以确保变更没有传播不期望的副作用。只针对关键模块。关键模块的特征：涉及几个软件需求；在程序的模块结构中位于较高层次；较复杂、较易发生错误；有明确定义的性能要求。

5) 冒烟测试的好处：降低集成风险；提高最终产品的质量；简化错误的诊断和修正；易于评估进展状况。

5. α 测试和 β 测试（会区分）

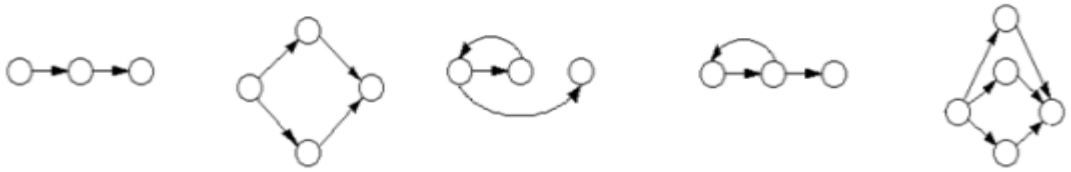
α 测试是由一个用户在开发环境下进行的测试，也可以是公司内部用户在模拟实际操作环境下进行的测试。α 测试在受控环境下进行。

β 测试是由软件的多个用户在实际使用环境下进行的测试。β 测试在一个或多个最终用户场所进行，与 α 测试不同，开发者通常不在场。

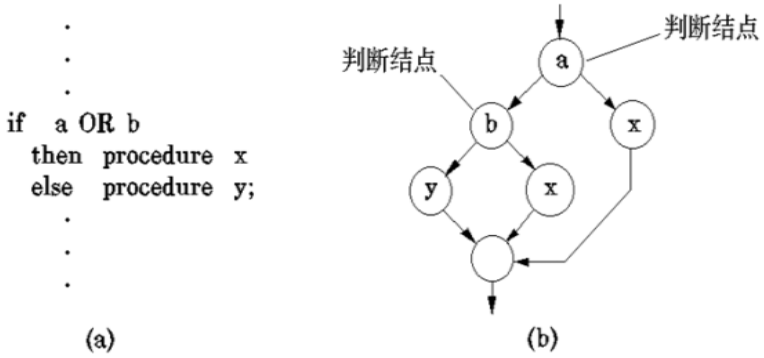
6. 系统测试的概念：将通过确认测试的软件，作为整个基于计算机系统的一个元素，与计算机硬件、外设、某些支持软件、数据和人员等其他系统元素结合在一起，在实际运行环境下，对计算机系统进行一系列的组装测试和确认测试。

CH15/16 测试传统的/面向对象的应用系统

1. 白盒测试 P281：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。
2. 基本路径测试 P281



顺序结构　　IF 选择结构　　WHILE 循环结构　　UNTIL 循环结构　　CASE 多分支结构



(a)　　(b)

程序环路复杂性：

环路复杂度=最少独立路径数=域的数量

环复杂性 V(G)=P+1，其中 P 为包含在流图 G 中的判定结点数，此处判定结点是简单判定结点。

3. 黑盒测试：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。

   主要发现以下错误：不正确或遗漏的功能；接口错误；数据结构或外部数据库访问错误；行为或性能错误；初始化和终止错误。

   1) 等价类划分：把所有可能的输入数据，即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有代表性的数据作为测试用例。
   2) 等价类划分有 2 种情况：
      - 有效等价类：对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：对程序的规格说明来说是不合理的，无意义的输入数据构成的集合。
   3) 划分等价类的原则
      - 如果输入条件指定一个范围，则可以定义一个有效等价类和两个无效等价类。
        如在程序规格说明中规定输入项数可以从 1 到 999，则一个有效等价类是“1≦项数≦999”，两个无效等价类是“项数<1”“项数>999”。
      - 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，此时可确定一个有效等价类和一个无效等价类。
        如在 Pascal 语言中对变量标识符规定为“以字母打头的..串”，那么所有以字母打头的构成有效等价类，否则归于无效等价类。
      - 如果输入条件是一个布尔值，则可以确定一个有效等价类和一个无效等价类。
      - 如果规定输入数据的一组值，且程序要对每个输入值分别处理，此时可为每个输入值确定一个有效等价类，针对这组值确定一个无效等价类，它是所有不允许的输入值的集合。
        如在教师上岗方案中规定对教授、副教授、讲师、助教分别计算分数，做相应处理。因此可以确定 4 个有效等价类是教授、副教授、讲师、助教，所有不符合以上身份人员的输入值集合就是无效等价类。
      - 如果规定了输入数据必须遵守的规则，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（从不同角度违反规则）。
   4) 边界值分析 P288：一种黑盒测试方法，对等价类划分的补充。大量的错误是发生在输入或输出范围的边界上。
4. 如何在测试中区分类间关系
5. 类间测试用例设计 P300（了解）

CH18/19 项目管理

1. 管理涉及的范围：（4 个 P）人员，产品，过程，项目。

2. W^5HH 原则：P330
3. 了解 LOC（面向规模的度量）、FP（面向功能的度量）及其相互调和 P338
4. 测量质量 P343：软件质量的事后度量，包括正确性、可维护性、完整性、可使用性。
   1) 正确性的度量是 KLOC 的差错数；
   2) 可维护性的度量必须采取间接度量，通过平均变更等待时间 MTTC 来度量；
   3) 完整性度量一个系统抗拒对它的安全性攻击的能力，完整性=∑[1-危险性*(1-安全性)]，其中危险性是特定类型的攻击将在一给定时间内发生的概率，安全性是排除特定类型攻击的概率；
   4) 可使用性依据 4 个特征进行度量：
      - 为学习系统所需要的体力上和智力上的技能；
      - 为达到适度有效使用系统所需要的时间；
      - 当软件被某些人适度有效使用时所度量的在生产率方面的净增值；
      - 用户角度对系统的主观评价。
5. 缺陷排除效率 DRE　P344

   DRE=E/(E+D)，其中 E 是软件交付给最终用户前发现的错误数；D 是交付后发现的错误数。

课后习题 19.5　19.7　19.8　19.10　19.11

CH20 软件项目估算

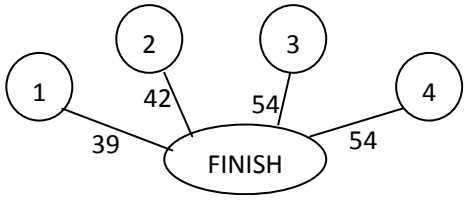
COCOMOII 模型：将项目估算分为三个阶段：应用组装阶段（用对象点 OP 计算），早期设计阶段（用功能点 FP 计算），体系结构后阶段（以 FP 或 LOC 作为估算单位）。

在应用组装阶段，计算对象点：

1) 计算屏幕数、报表数、3GL 构件数；
2) 对每个对象点，按简单、中等、困难进行分类；
3) 根据图 20-6，得到简单、中等、困难的对象点的复杂度权；
4) 计算加权的屏幕、报表、3GL 构件之和，即对象点数 OP；
5) 如果 r%的应用点来自以前的项目的重用，则新对象点 NOP=OP*(100-r)/100；
6) 用生产率能力 PROD 进行调整，涉及两个方面：开发者的经验能力、CASE 成熟度能力（图 20-7）；
7) 估算工作量 E=NOP/PROD。

CH21 项目进度安排

1. 人员与工作量之间的关系 P369
   1) 当几个人共同承担软件开发项目中的某一任务时，人与人之间必须通过交流来解决各自承担任务之间的接口问题，即所谓通信问题。通信需花费时间和代价，会引起软件错误增加，降低软件生产率。
   2) 若两个人之间需要通信，则称这两个人之间存在一条通信路径。
   3) 若一个人单独开发软件，生产率是 5000 行/人年。如果一个小组有 4 个人，则需要 6 条通信路径，每条通信路径上耗费的工作量是 250 行/人年，则小组中每个人的软件生产率降低为 5000-6*250/4=4625 行/人年。
   4) 结论：开发小组不宜太大，成员之间避免太多的通信路径；开发过程中切忌中途加人，除非是对项目有一定了解的人员，否则只会拖延项目。
2. 关键任务



任务 1、2、3、4 分别需要 39、42、54、54 天的时间来完成，而 3、4 的完成时间决定总工程的完成时间，则称 3、4 为关键任务，1、2 的有限延误不会影响。

3. 获得值分析 EVA　P378
   1) 计算：
      - 工作预计成本 BCW：每项工作任务的预计工作量。
      - 所计划的工作的预计成本 BCWS：计划按指定时间完成的每项工作任务的预计工作量之和。
      - 完成预算 BAC：BCW 的总量，是项目总工作量的估计。
      - 计划值 EV=BCWS/BAC。

- 所完成工作的预计成本 BCWP：已经按指定时间完成的工作任务的预计工作量之和。
- 所完成工作的实际成本 ACWP：已经完成的工作任务的实际工作量之和。
- 完成的百分比 EV= BCWS/BAC。
- 进度性能指标 SPI=BCWP/BCWS
- 进度偏差 SV=BCWP-BCWS
- 成本性能指标 CPI=BCWP/ACWP
- 成本偏差 CV=BCWP-ACWP
- 例

工作任务	估计工作量	实际工作量	估计完成日期	实际完成日期
1	5	10	1/25/01	2/1/01
2	25	20	2/15/01	2/15/01
3	120	80	5/15/01	
4	40	50	4/15/01	4/1/01
5	60	50	7/1/01	
6	80	70	9/1/01	

BAC=330 天，为总工作量的估算。

在 4/1/01，任务 1、2、4 已经完成，BCWP 为这些任务的 BCW 之和=70 天。

EV=70/330=21.2%

在 4/1/01，计划完成任务 1、2，所以 BCWS=30 天

SPI=70/30=233%

SV=70-30=40 天，即提前 40 个人日。

ACWP 为任务 1、2、4 的实际工作量之和=80 天

CPI=70/80=87.5%

CV=70-80=-10，即延迟了 10 个人日。

练习：

课后习题 21.12

1.若考察点在 01 年 7 月 1 日，任务 3 已经完成，实际使用了 140 天，计算 BCWP,BCWS,SPI,SV,ACWP,CPI,CV。

工作任务	估计工作量	实际工作量	估计完成日期	实际完成日期
1	5	10	1/25/01	2/1/01
2	25	20	2/15/01	2/15/01
3	120	140	5/15/01	7/1/01
4	40	50	4/15/01	4/1/01
5	60	50	7/1/01	
6	80	70	9/1/01	

答案：

1. BCWP 为 1、2、3、4 任务的 BCW 之和=190 天

   EV=190/330=57.5%

   在 7、1、01，计划完成任务 1、2、3、4、5，所以 BCWS=250 天

   SPI=190/250=76%

   SV=190-250=-60，即延迟 60 个人日

   ACWP 为任务 1、2、3、4 的实际工作量之和=220 天

   CPI=190/220=86.3%

   CV=190-220=-30，即延迟了 30 个人日。