**软件工程课程复习**

1. 什么是软件危机？表现？
   1. 软件危机与软件工程：
      1. 20世纪60年代末，由于软件规模扩大、软件复杂性提高导致软件开发开发周期长、成本高、质量差和维护困难，爆发了软件危机，为了克服软件危机，1968年北大西洋公约组织（NATO）提出了软件工程的概念，试图将工程化的方法应用于软件开发。
      2. 许多软件项目不能满足客户的要求
      3. 许多软件项目超出预算和时间安排
   2. 表现：
      1. 对软件开发成本和进度的估计常常很不准确。
      2. 用户对“已完成的”软件系统不满意的现象经常发生。
      3. 软件产品的质量往往靠不住。
      4. 软件常常是不可维护的。
      5. 软件通常没有适当的文档资料。
      6. 软件成本在计算机系统总成本中所占的比例逐年上升。
      7. 软件开发生产率提高的速度，既跟不上硬件的发展速度，也远远跟不上计算机应用迅速普及深入的趋势。
      8. 还有软件开发和维护有关的很多问题
2. 软件组成：
   1. 程序
   2. 文档
   3. 数据
3. 管理的目标：

（1）达到项目预期的软件产品功能和性能要求。也就是软件产品达到了用户已认可的需求规格说明的要求。

（2）时限要求。项目应在合同规定的期限内完成。

（3）项目开销限制在预算之内。

1. 软件项目管理涉及的几个主要方面（4P），了解每个管理的内容。
2. People 人员管理

人员管理涉及：利益相关方、团队负责人、团队集体

1. Product 产品管理

项目经理必须在项目开始时就明确项目的以下三个目标，才能着手项目管理的各项工作，如项目估算、风险分析、项目计划的制定等。

1. Process 过程管理

过程在软件工程项目中是重要的因素，它决定着项目中开展哪些活动以及对活动的要求和开展活动的顺序。

1. Project 项目管理

项目管理的任务是如何利用已有的资源，组织实施既定的项目，提交给用户适用的产品。

1. 软件工程方法学包括三要素，及其作用
   1. 软件工程**方法**为软件开发提供了 “如何做” 的技术;
   2. 软件**工具**为软件工程方法提供了自动的或半自动的软件支撑环境;
   3. **过程**是为了获得高质量的软件所需要完成的一系列任务框架，它规定了完成各项任务的工作步骤
2. 软件工程的目标？
   1. 运用先进的软件开发技术和管理方法来提高软件的质量和生产率
   2. 也就是要以较短的周期、较低的成本生产出高质量的软件产品，并最终实现软件的工业化生产。
3. 软件规模估算：
   1. 功能点方法（function point）简称FP方法
   2. 软件规模估算的功能点方法：
      1. 外部输入
      2. 外部输出
      3. 内部逻辑文件
      4. 外部接口文件
      5. 外部查询
   3. 调节因子
   4. 交付功能点

经过调节因子调节后的功能点值被称为交付功能点(delivered function point，DFP)

DFP = CAF × UFP

CAF：复杂度调节因子

UFP: 未调节功能点

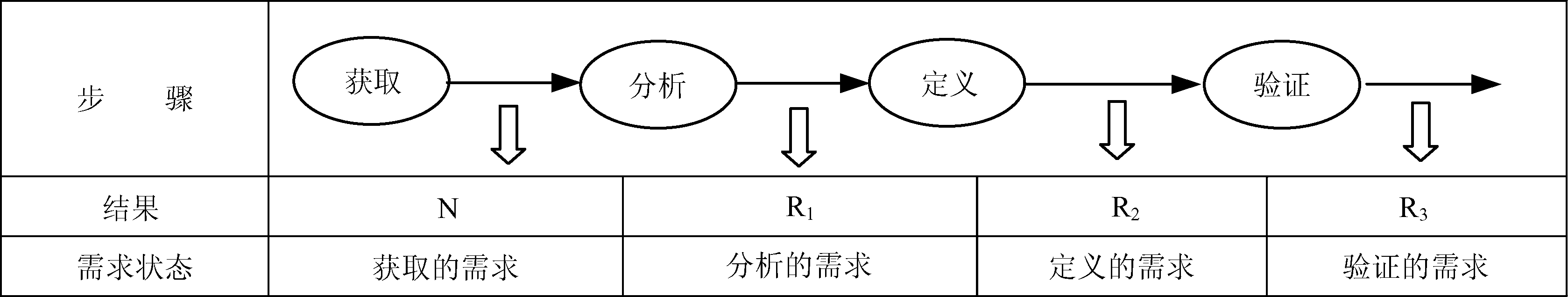
* 1. 交付功能点与软件规模

计算出的功能点的值可以代表软件的规模，也可作为估算成本的依据。

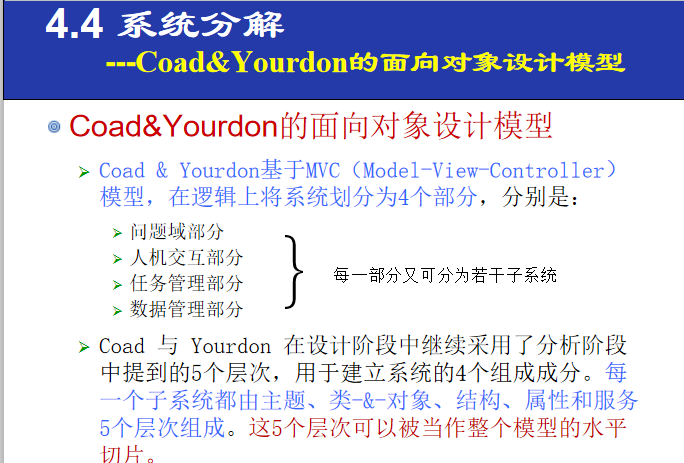
软件的规模可用交付的源代码行数(delivered lines of code，DLOC)来表示。

1. 软件质量包括哪些方面？功能性、可靠性、可使用性、效率、可维护性和可移植性
2. 软件需求分析阶段的任务：

可以把软件需求分析阶段的工作分为4个步骤，即获取需求、分析需求、定义需求和验证需求，如图所示。



1. Coad&Yourdon的面向对象设计模型：



1. 需求分析、概要设计等各阶段在做什么？
   1. 需求分析：
      1. 准确地回答：“目标系统必须做什么”这个问题。
      2. 分析内容：对目标系统提出完整、准确、清晰、具体的要求。对用户提出的需求进行分析并给出详细的定义。
      3. 输出（软件需求说明书、或系统功能说明书、初步的系统用户手册）：提交管理机构评审
   2. 可行性：
      1. 明确回答：“上一个阶段所确定的要解决的问题是否有行得通的解决办法”。
      2. 输出（可行性研究报告）：从技术、经济和社会因素等方面研究论证各方案的可行性。 为决策提供依据。
   3. 概要设计：
      1. 概括地回答：“怎样实现目标系统?”
      2. 设计内容：设计程序的体系结构，即确定程序的组成模块，确定模块间的关系。
      3. 输出文档：软件的概要设计说明书。
   4. 详细设计：
      1. 详细回答：“应怎样具体实现这个系统？”
      2. 设计内容：详细地设计每个模块，确定实现模块功能所需要的算法和数据结构。
      3. 输出文档：软件的详细设计说明书
   5. 程序编码和单元测试：
      1. 实现：正确的、容易理解的、容易维护的程序模块。
      2. 输出文档：源程序、详尽的程序说明和单元测试报告。
   6. 集成测试和系统测试：
      1. 集成测试实现：对模块的组装和连接测试(及相应的调试)，使软件达到预定的要求。
      2. 系统测试实现：根据需求规格说明，逐项测试并确认
      3. 输出文档：测试计划、详细测试方案以及实际测试结果等。
   7. 维护
      1. 改正性维护：诊断和改正在使用过程中发现的软件错误
      2. 适应性维护：修改软件以适应环境的变化
      3. 完善性维护：根据用户的要求改进或扩充软件，使它更完善
      4. 预防性维护：修改软件为将来的维护活动预先做准备
   8. 开发过程中的典型文档
      1. 软件可行性分析报告
      2. 软件需求规格说明书
      3. 项目计划
      4. 软件设计说明书
      5. 软件测试计划
      6. 用户手册



1. 包及其依赖性：
   1. 包
2. 包（package）又可称为层或子系统，是表示组织类的一种方式，用于划分应用程序的逻辑模型。
3. 包是高度相关的类的聚合，这些类本身是内聚的，但相对于其他聚合来说又是松散耦合的。
4. 包可以嵌套。外层包可以直接访问包括在它的嵌套包中的任何类。
5. 包还可以导入其他包，例如，在包A中导入了包B，这意味着包A或者包A的元素可以引用包B或者包B的元素。因此，虽然一个类只属于一个包，但是它可以被导入其他包。
6. 包的导入操作会引入包之间的依赖性以及它们的元素之间的依赖性。
   1. 依赖性:
7. 如果包A的一些成员在某种程度上引用了包B的某些成员（包A导入了包B的一些成员），这隐含着双重含义，有依赖性。
8. 本质上，两个包之间的依赖性来自于两个包中类之间的依赖性。
9. 类之间的循环依赖性是个特别棘手的问题，但大多数情况下可以通过重新设计避免循环依赖性。
10. 通过增加新包来消除包之间的循环依赖性
11. 软件需求的3个层次：
12. 业务需求：客户对软件的高层目标要求。
13. 用户需求：用户使用软件必须达到的要求和完成的任务。通常在用例(use case)或方案脚本(scenario)中加以说明。
14. 功能和非功能需求。规定了开发人员必须实现的需求，它的实现将满足上述业务需求和用户需求。通常以需求规格说明(requirement specification)的形式给以详尽描述
15. 面向对象设计过程与准则：

（1）模块化

（2）抽象

（3）信息隐藏

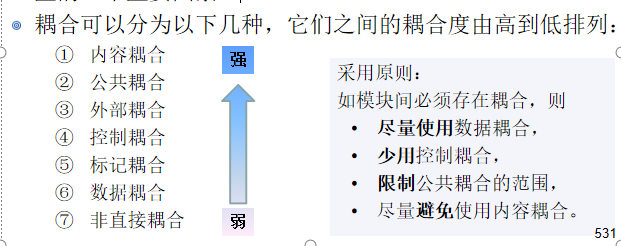
（4）弱耦合 （可参见教材P80）

（5）强内聚 （可参见教材P82）

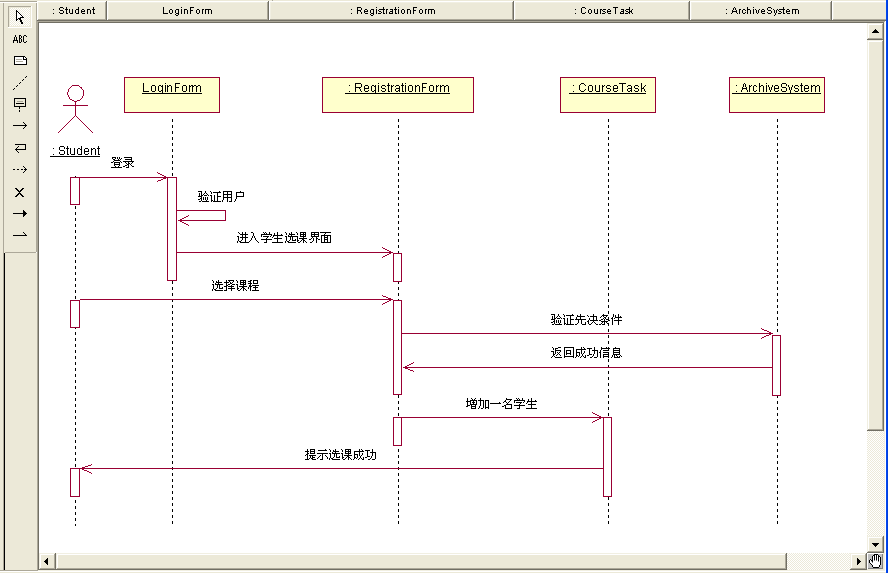
（6）可重用

什么是耦合：耦合是软件各模块之间相互关联的度量。它取决于各个模块之间接口的复杂程度、调用模块的方式以及哪些信息通过接口。耦合是影响软件复杂程度和设计质量的一个重要因素。

耦合可以分为几种？它们之间的耦合度由高到低排列

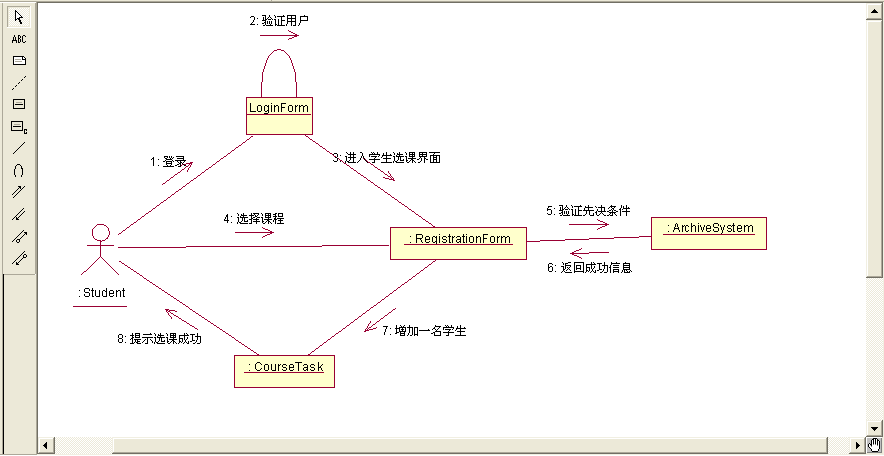


1. 面向对象的分析模型：用例图、类图、顺序图和包图
2. UML中动态模型的描述工具：顺序图、通信图和状态图
   1. 顺序图
3. 用例图中的事件流是由文本表示的，事件流描述的是用例实现的过程，也称为场景（scenarios），可以用顺序图表示场景。
4. 顺序图按照时间顺序显示对象之间的交互关系。它描述场景中的对象和类以及在完成场景中定义的功能时对象间要交换的信息



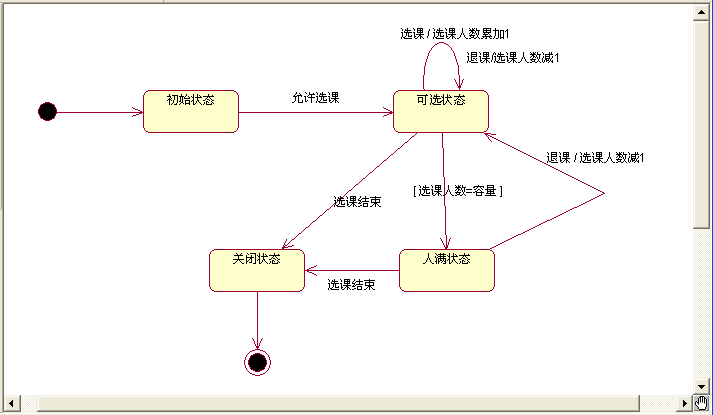
* 1. 通信图

1. 协作图也称通信图，是顺序图的另一种表示形式，用于描述相互协作的对象间的交互关系和链接关系。
2. 一般情况下，当表示涉及很多对象的模型时，协作图比顺序图更形象。
3. 另外，与顺序图不同，对象之间的实线可能表明这些对象的类之间需要关联



* 1. 状态图

1. 状态图由对象的各个状态和连接这些状态的转换组成。
2. 通常，用一张状态图描绘一类对象的行为，它确定了由事件序列引出的状态序列。
3. 不是任何一个类都需要有一张状态图描绘它的行为，只针对具有明显的状态特征并且具有比较复杂的状态—事件—响应行为的类，才需要画状态图。



1. 面向对象分析中需要创建的3个模型：用例、对象、动态模型
   1. 用例模型：用例和场景表示的功能模型；（是基础和依据）
   2. 对象模型：用类和对象表示的静态模型；（是核心）
   3. 交互模型：由状态图和顺序图表示的动态模型。（涉及交互时很重要 ）
2. 设计模式：

总体来说设计模式分为三大类：

创建型模式，共五种：工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式。

结构型模式，共七种：适配器模式、装饰器模式、代理模式、外观模式、桥接模式、组合模式、享元模式。

行为型模式，共十一种：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式。

需要知道你熟悉的几个设计模式的作用、使用场景等等。

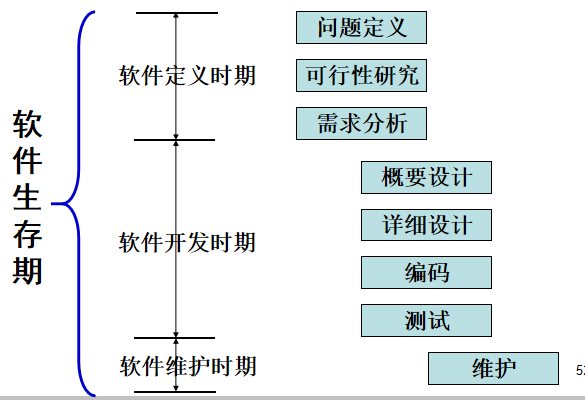
1. 软件生存期
2. 软件定义
   * 1. 确定总目标和可行性
     2. 导出策略和系统功能
     3. 估计资源和成本
     4. 制定工程进度表
3. 软件开发
   1. 任务：具体设计和实现软件定义时期定义的软件
   2. 执行人：系统设计员，高级程序员，程序员，测试工程师和辅助人员等
   3. 阶段划分：分为概要设计、详细设计、编码和单元测试、集成测试和系统测试。前两个阶段为系统设计，后两个阶段为系统实现
4. 运行维护
   1. 目的：使软件能持久地满足用户的需要
   2. 4类维护活动：

改正性维护：诊断和改正在使用过程中发现的软件错误

适应性维护：修改软件以适应环境的变化

完善性维护：根据用户的要求改进或扩充软件，使它更完善

预防性维护：修改软件为将来的维护活动预先做准备



1. 对象模型的5个层次：主题层(也称为范畴层)、类-对象层、结构层、属性层和服务层
2. 对象模型类分类：实体类、边界类、控制类
   1. 实体类表示系统将跟踪的持久信息；
   2. 边界类表示参与者与系统之间的交互；
   3. 控制类负责用例的实现
3. 对象模型中类之间的关系：普通关联（一对一、一对多、多对多）、聚合、泛化、依赖等
4. 面向对象分析的模型:
5. 用例模型：用例和场景表示的功能模型；（是基础和依据）
6. 对象模型：用类和对象表示的静态模型；（是核心）
7. 交互模型：由状态图和顺序图表示的动态模型。（涉及交互时很重要 ）
8. 软件生存模型也叫软件过程模型,典型的有:
9. 瀑布模型
10. 快速原型模型
11. 增量模型
12. 螺旋模型
13. 喷泉模型
14. 统一过程
15. 基于构件的开发模型
16. 敏捷过程

注意各位模型的优缺点，PPT中有详细描述

瀑布模型的特点：

阶段间的顺序性和依赖性

推迟实现的观点（是瀑布模型的重要指导思想）

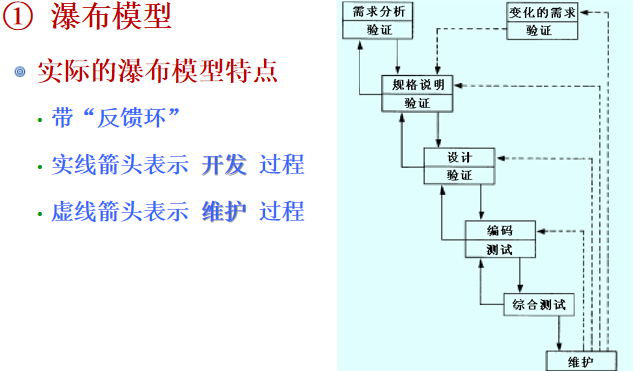
质量保证的观点（重视文档及对文档评审，以便尽早发现问题，改正错误）

实际的瀑布模型特点：

带“反馈环”

实线箭头表示 开发 过程

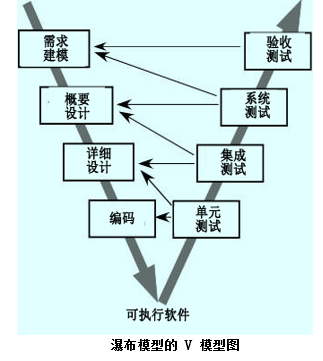
虚线箭头表示 维护 过程



瀑布模型的V模型特点:

瀑布模型的一个变体

V模型描述了测试阶段的活动与开发阶段相关活动（包括需求建模、概要设计、详细设计、编码）之间的关系。



瀑布模型的优点、缺点及使用范围

优点：

强制开发人员采用规范化的方法。

严格规定每个阶段必须提交的文档。

每个阶段交出的所有产品都必须经过验证。

缺点：完全依赖书面的规格说明，如需求规格说明与用户需求之间有差异，则可能导致最终开发出的软件产品不能真正满足用户的需要；

适用范围：只适用于项目开始时需求已确定的情况。

1. 能分析业务画出对应的用例图、顺序图、状态图等
2. 三层C/S体系结构将整个系统分成：
3. 表示层

是应用系统的用户界面部分，担负着用户与应用程序之间的对话功能。它用于检查用户从键盘等输入的数据，显示应用程序输出的数据，一般采用图形用户界面（graphic user interface， GUI）。

1. 应用逻辑层

是应用系统的主体部分，包含具体的业务处理逻辑。通常在功能层中还包含有确认用户对应用和数据库存取权限的功能以及记录系统处理日志的功能

1. 数据层

主要包括数据的存储及对数据的存取操作，一般选择关系型数据库管理系统

1. 三层C/S体系结构特点：

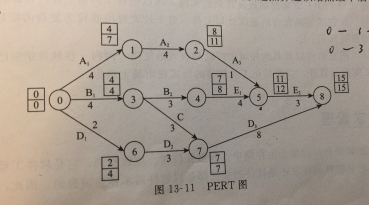
为了克服两层的缺点，增加了应用服务器。可以将整个应用逻辑驻留在应用服务器上，而只有表示层存在于客户机上。

1. 瘦客户机模型。
   1. 特点：数据管理部分和应用逻辑都在服务器上执行，客户机只负责表示部分。
   2. 缺点：
      1. 它将繁重的处理负荷都放在了服务器和网络上，服务器负责所有的计算，这将增加客户机和服务器之间的网络流量。
      2. 目前个人计算机所具有的处理能力在瘦客户机模型中用不上。
2. B/S体系结构：
3. 优点：
4. 基于B/S体系结构的软件，系统安装、修改和维护全 在服务器端解决;
5. B/S体系结构还提供了异种机、异种网、异种应用服 务的联机、联网和统一服务的最现实的开放性基础
6. 缺点：
7. B/S体系结构缺乏对动态页面的支持能力，没有集成有效的数据库处理功能。
8. 采用B/S体系结构的应用系统，在数据查询等响应速度上，要远远地低于C/S体系结构。
9. B/S体系结构的数据提交一般以页面为单位，数据的动态交互性不强，不利于在线事务处理（OLTP）应用
10. 客户机/服务器体系结构：

客户机/服务器（client/server，C/S）体系结构是基于资源不对等，且为实现共享而提出来的，由服务器、客户机和网络三部分组成。

1. PERT图：
2. 计划评审技术（program evaluation and review techique，PERT）也称网络图方法，或简称PERT图方法，它的另一名称是关键路径法（critical path method，CPM）
3. 有向的箭头作为边表示子任务，它是有名称（即子任务名）、有长度（即完成此项子任务所需的时间）的向量；
4. 编号的圆圈作为结点，它应该是子任务向量的始发点或指向点；从与结点相连的边可以看出任务间的依赖关系。
5. 若干条边和若干个结点构成了网状图，我们根据相互衔接的子任务形成的路径，进行路径长度的计算、比较和分析，从而实现项目工期的控制。

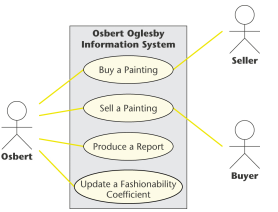
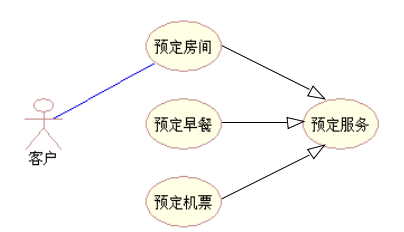
课本P327页，图13-11。



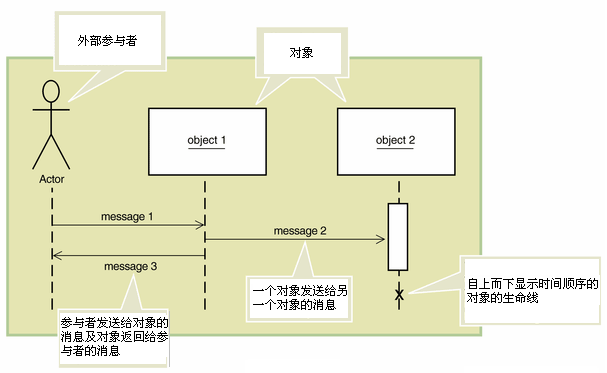
最早启动时间：从前往后推

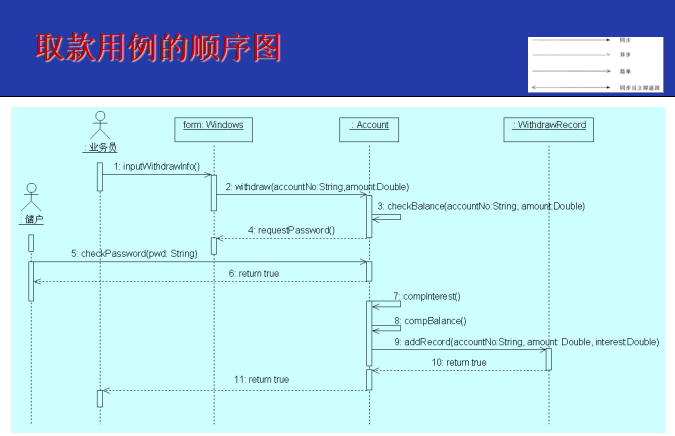
最迟启动时间：从后往前推

1. 用例图、顺序图、通信图、状态图、活动图、构件图、部署图
   1. 用例图：
      1. 执行者
      2. （由系统边界（一个矩形）封闭的）一组用例
      3. 执行者和用例之间的关联
      4. 用例间关系

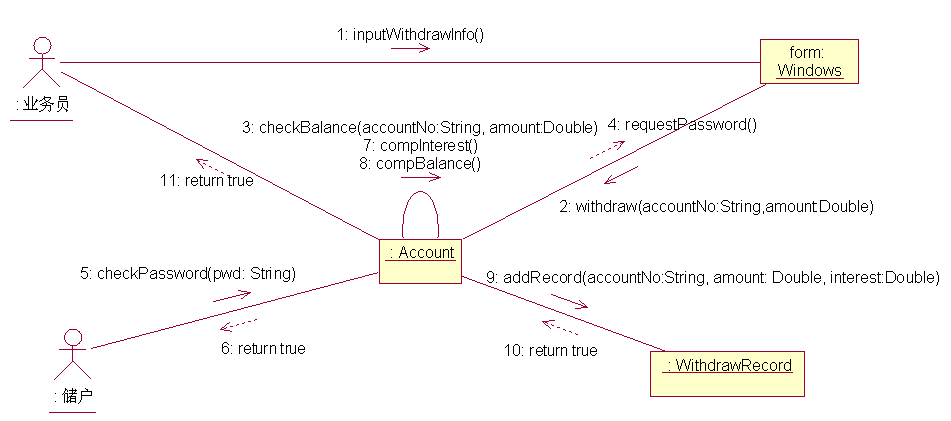
 

* 1. 顺序图：描述对象之间的动态交互关系，着重表现对象间消息传递的时间顺序。

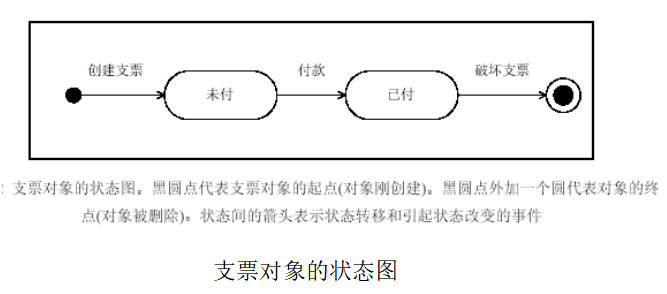




* 1. 通信图是顺序图的一种变化形式，用于描述相互协作的对象间的交互关系和链接关系。

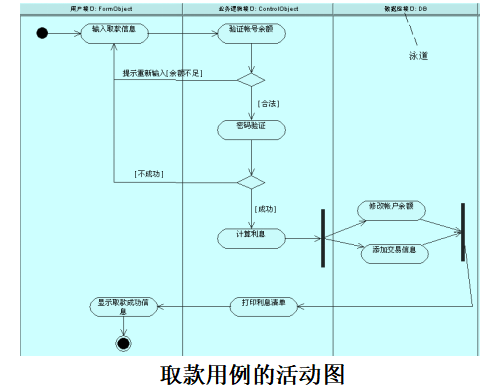


* 1. 状态图：描述一个特定对象的所有可能的状态以及引起状态转换的事件。大多数面向对象技术都用状态图表示单个对象在其生命期中的行为。一个状态图包括一系列状态、事件以及状态之间的转移。

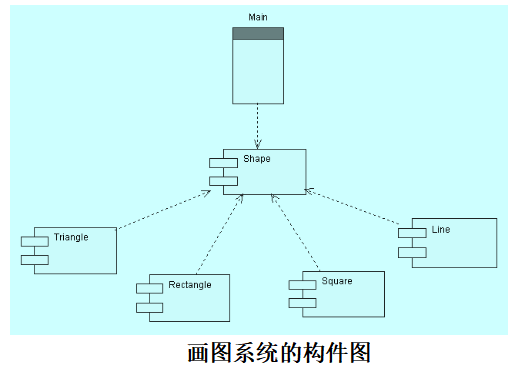


* 1. 活动图用来捕捉用例的活动，使用框图的方式显示动作及其结果

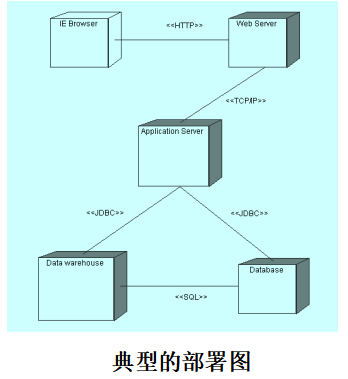
1. 活动图是一个流图，描述了从活动到活动的流。
2. 它是另一种描述交互的方式，它描述采取何种动作，动作的结果是什么(动作状态改变)，何时发生(动作序列)，以及在何处发生(泳道)。



* 1. 构件图描述软件构件及构件之间的依赖关系，显示代码的静态结构。



* 1. 部署图描述处理器、设备和连接，它显示系统硬件的物理拓扑结构及在此结构上执行的软件



1. 面向对象模块设计准则：

（1）模块化

（2）抽象

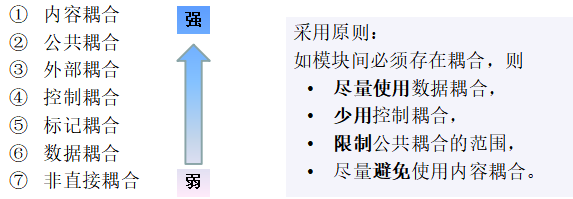
（3）信息隐藏

（4）弱耦合 （可参见教材P80）

（5）强内聚 （可参见教材P82）

（6）可重用

1. 各种内聚及耦合分类
   1. 强内聚 （可参见教材P82）
   2. 弱耦合 （可参见教材P80）
   3. 耦合
      1. 什么是耦合：耦合是软件各模块之间相互关联的度量。它取决于各个模块之间接口的复杂程度、调用模块的方式以及哪些信息通过接口。耦合是影响软件复杂程度和设计质量的一个重要因素。
      2. 耦合可以分为以下几种，它们之间的耦合度由高到低排列：



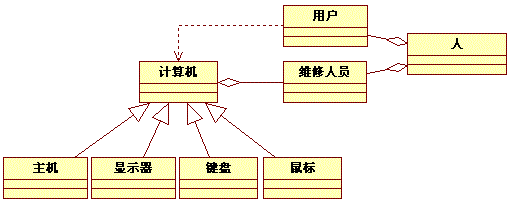
* + 1. 交互耦合、继承耦合

1. 代码规范问题

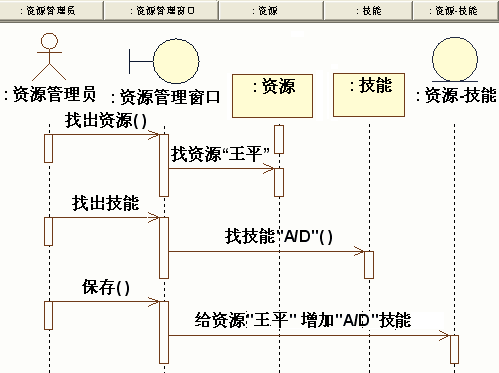
规范涉及：版面、注释、标识符命名、变量使用、代码可测性、程序效率、质量保证、代码编译、单元测试、程序版本与维护。

**综合类举例：**

1. 找出并说明下面类图中的错误。



1. 参考下图，说明“资源管理窗口”和“资源-技能”类至少应包含哪些操作。



资源管理窗口类包含操作：

1. 找出资源（）
2. 找出技能（）
3. 保存（）

资源-技能类包含操作：

1. 给资源”王平“能加”A/D“技能

3.画出能正确表示下面叙述的类图（括号内为类名）。

 “一个雇员（Employee）最多由一个经理（Manager）管理，某些经理管理多个雇员，某些经理不管理任何雇员”。



4.设计一个饮料自动售货机系统，其主要功能是向顾客出售饮料，同时供应商需要向其中放置饮料，收银员需要向其中放置零钱和收回营业收入。画出该系统的用例图。

