软件工程：第十章 面向对象设计

导学目标

- 了解面向对象设计的目的、设计准则以及启发式规则
- **掌握软件重用的三个层次、可重用的成分以及重用的好处**
- 了解系统分解中的4个部分以及两种交互方式

第一节 面向对象设计的准则及启发式规则

- 分析：提取整理用户的需求，建立问题域精确模型(OOA)
- 设计：转变需求为系统实现方案，建立求解域模型(OOD)
- 在实际的软件开发中，分析和设计的界限是模糊的
- 分析和设计的活动是一个反复迭代的过程
- 面向对象方法学在概念和表示方法上的一致性，保证了在各项开发活动之间的平滑(无缝)过渡，领域专家和开发人员能够比较容易地跟踪整个系统开发过程，这是面向对象方法与传统方法比较起来所具有的一大优势

1、面向对象设计准则

1. 模块化
    - 对象就是模块
2. 抽象
    - 类抽象、提高可重用性
3. 信息隐蔽
    - 通过封装性实现、提高模块独立性
4. 弱耦合
    - 在面向对象方法中，耦合是指不同对象之间相互关联的紧密程度
    - 对象间耦合分为两大类：交互耦合和继承耦合
    - **交互耦合**：对象间通过消息连接实现(松散)
        - 降低消息连接复杂度(减少参数个数，降低参数复杂度
        - 减少信息数
    - **继承耦合**：一般类和特殊类之间耦合(紧密)
        - 有继承关系基类和派生类是系统中粒度更大模块
5. 强内聚
    - 服务内聚：只完成一个功能
    - 类内聚：一个类只有一个用途，否则分解
    - 一般特殊内聚：设计合理，是对领域知识正确抽取
6. 可重用性
    - 尽量利用已有类(类库、已创建类)
    - 创建新类考虑以后可重用性

2、启发式规则

1. 设计结果清晰易懂
    - 用词一致
    - 使用已有的协议
    - 较少消息模式的数目
    - 避免模糊的定义
2. 一般-特殊结构的深度要适当
    - 应该是类层级中包含的层次数适当
3. 设计简单的类
    - 避免包含过多的属性
    - 有明确的定义
    - 简化对象之间的合作关系
    - 不要提供太多服务
4. 使用简单的协议
    - 经验表明，通过复杂消息相互关联的对象是紧耦合的，对一个对象的修改往往导致其他对象的修改
5. 使用简单的服务
    - 本类中的操纵尽可能简单
6. 把设计变动减至最小

第二节 软件重用

重用也叫再用或复用，是指同一事物不做修改或稍加改动就多次重复使用

1、软件重用的三个层次

- 知识重用
- 方法和标准的重用
- 软件成分的重用

2、软件成分重用的三个层次

- 代码重用
    - 调用库中的模块
- 设计重用
    - 重用某个软件系统的设计模型
- 分析重用
    - 重用某个系统的分析模型

3、典型的可重用软件的成分

- 项目计划
- 成本估计
- 体系结构
- 需求模型和规格说明
- 设计

- 源代码
- 用户文档和技术文档
- 用户界面
- 数据
- 测试用例

3、类构件重用的三种方式

- 实例重用
  - 使用者无须了解实现细节就可以使用适当的构造函数创建类的实例
- 继承重用
  - 提供了一种安全地修改已有类构件
- 多态重用
  - 降低了消息连接的复杂程度，提供了简便可靠的软构件机制

4、软件重用的好处

- 质量
  - 每一次重用，软件质量都会有所改善
- 生产率
  - 软件开发全过程时间较少，带来生产率提高
- 成本
  - 软件重用可以减少净成本

第三节 系统分解

1、系统分解的定义

把系统分解成若干个比较小的部分，然后在分别设计每个部分

系统的主要组成部分是子系统，各个子系统之间应该具有尽可能简单、明确的接口，再划分和设计子系统时，应当尽量减少子系统之间的依赖

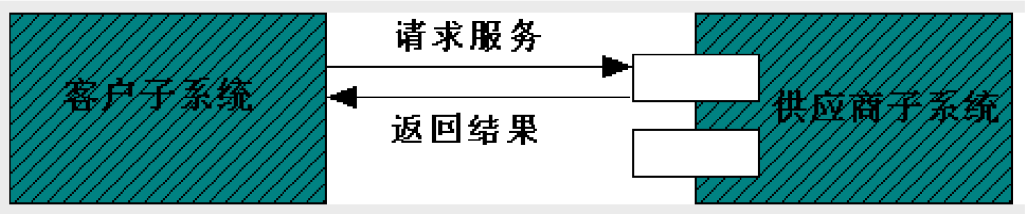
2、面向对象设计模型，逻辑上由4大部分组成

- 问题域
  - 直接负责实现客户需求子系统
- 人机交互
  - 实现用户界面子系统包括可复用的GUI子系统
- 任务管理

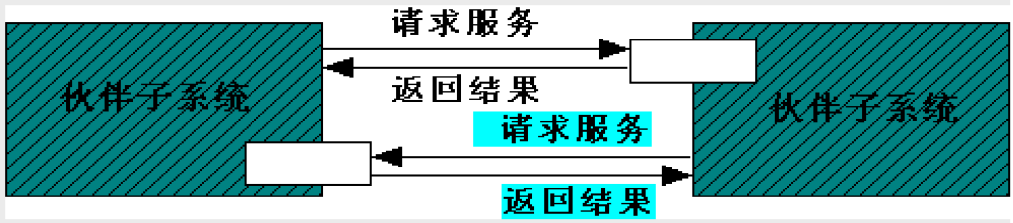
    - 确定各类任务，把任务分配给适当的硬件或软件去执行
- 数据管理
    - 负责对象的存储和检索的子系统

3、子系统间交互方式

- 客户 - 供应商关系
    - “客户”子系统了解“供应商”子系统接口，反之则不能



- 平等伙伴关系
    - 各子系统都有可能调用其它子系统，或为其它子系统提供服务。交互方式复杂，各子系统需要了解彼此接口



第四节 设计问题域子系统

1、设计基础

面向对象分析所得出的问题域模型

2、设计任务

对问题域模型做相应的补充或者修改

3、补充或者修改的内容

1. 调整需求
    - 用户需求或外部环境变化
    - 分析模型不完整、不准确
2. 重用已有的类
    - 根据问题解决的需要，把从类库或其他来源得到既存类增加到问题解决方案中去
    - 重用已有类的典型过程：
        - 标出候选类中对本问题无用的属性和服务
        - 在重用类和问题域类之间添加泛化关系
        - 标出问题域类从已有类继承来的属性和服务
        - 修改与问题域类的关联
3. 把问题域类组合到一起
    - 设计者往往通过引入一个根类而把问题域类组合到一起
4. 添加一般化类

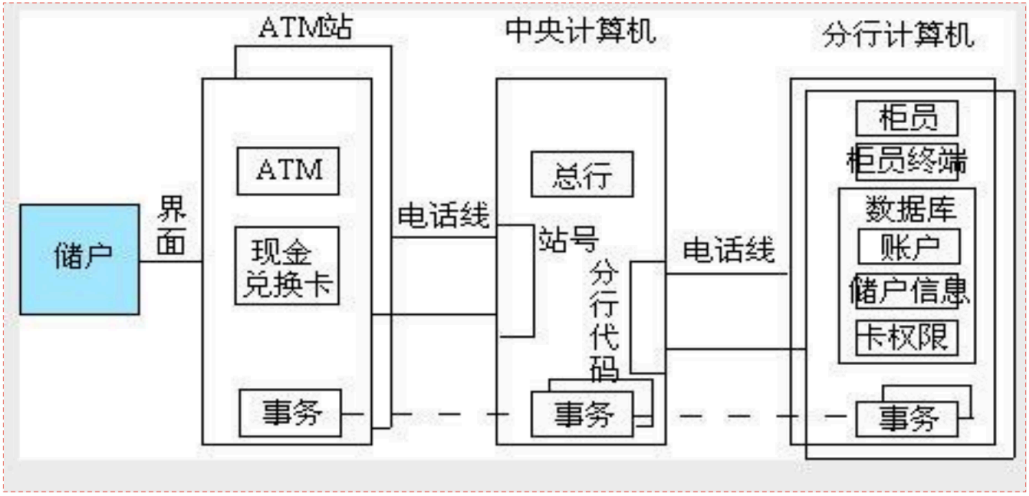
- 某些特殊类要求一组类似的服务，应加入一般化的类，定义为所有特殊类共用的一组服务名，服务都是虚函数；在特殊类中定义其实现

5. 调整继承的层次
    - 在OOA阶段建立的对象模型中可能包括多继承关系，但实现时使用程序设计语言可能只有单继承，需对分析结果修改

4、示例

ATM系统实例

- ATM系统问题域子系统划分成三个更小的子系统，分别是ATM站子系统、中央计算机子系统、分行计算机子系统
- 星型拓扑结构
- 以专用电话线连接



第五节 设计人机交互子系统

1、设计基础

在面向对象分析过程中对用户需求做的初步分析

2、设计任务

确定人机交互细节，窗口报表形式，命令层次等

3、设计策略：

1. 分类用户
    - 通常可以按照技能水平、职位、所属集团进行分类
2. 描述用户
    - 应该将使用系统的每类用户情况详细的记录下来
3. 设计命令的层次
    - 研究现有的人机交互的含义和准则
    - 确定初始的命令层次
    - 精化命令层次
4. 设计人机交互类

第六节 设计任务管理子系统

在实际系统中，许多对象之间往往存在相互依赖关系。设计工作的一项重要内容就是，确定哪些是必须同时动作的对象，哪些是相互排斥的对象。进一步设计任务管理子系统

系统总有许多并发行为，需按照各自行为的协调和通信关系，划分各种任务(进程)，简化并发行为的设计和编码

确定各类任务，把任务分配给适当的硬件和软件去执行

根据动态模型去分析、定义

1、分析并发性

- 并发对象
  - 无交互行为的对象
  - 同时接受事件的对象
- 定义任务
  - 检查各个对象的状态图，找没并发对象的路径(任何时候路径中只有单个对象是活跃的)，称控制线
  - 通过分离出控制线设计任务
- 并发任务分配
  - 每个任务分配到独立的处理器
  - 配到相同处理器，通过操作系统提供并发支持

2、设计任务管理子系统

- 确定事件驱动型任务
  - 指睡眠任务(不占用cpu)，某个事件发生，任务被触发，醒来做相应处理，又回到睡眠状态
- 确定时钟驱动型任务
  - 按特定时间间隔去触发任务进行处理。如某些设备需要周期性获取数据
- 确定优先任务
  - 高优先级任务分离出独立的任务，在规定时间内完成
- 确定关键任务
  - 严格可靠性、精心设计和编码、严格测试
- 确定协调任务
  - 三个以上任务，引入协调任务，控制封装任务间协调
- 尽量减少任务数
  - 任务多，设计复杂、不易理解、难维护
- 确定资源需求
  - 计算系统载荷，每秒处理业务数，处理一个业务花费时间，估算所需cpu (或其他固件)处理能力
  - 综合考虑，确定哪些任务硬件实现，哪些任务软件实现

第七节 设计数据管理子系统

1、选择数据存储管理的模式

1. 文件管理系统
    - 成本低、简单
    - 操作级别低，不同操作系统的文件系统差别大
2. 关系数据库管理系统
    - 优点：
        - 提供了各种最基本的数据管理功能
        - 为多种应用提供一致的接口
        - 标准化的语言
    - 缺点：
        - 运行开销大
        - 不能满足高级应用的需求
        - 与程序设计语言连接不自然
3. 面向对象数据库管理系统
    - 扩展的关系型数据库：
        - 增加抽象数据类型，继承等机制
        - 增加了创建管理类和对象的通用服务
    - 扩展的面向对象语言
        - 扩充了面向对象程序设计语言的语法和功能
        - 增加数据库存储和管理对象机制

2、设计数据管理子系统

1. 设计数据格式
    - 设计数据格式与数据存储管理模式密切相关
    - 文件系统：达到第一范式、减少文件数量、减小存储空间
    - 关系型数据库管理系统：达到第三范式、满足性能及存储需求
    - 面向对象数据库管理系统：对于扩展性关系型数据库使用与关系型数据库相同的方法；对于扩展性面向对象程序设计语言，不需要规范化属性的步骤
2. 设计相应的服务
    - 文件系统：打开文件、记录定位、检索记录、更新
    - 关系型数据库管理系统：访问那些数据库表、怎样访问所需要的行、怎样检索出旧值、怎样用现有值更新
    - 面向对象数据库管理系统：对于扩展性关系型数据库使用与关系型数据库相同的方法；对于扩展性面向对象程序设计语言，不需要增加服务设计数据管理子系统

3、示例：ATM系统

- 永久性数据存储在分行计算机
    - 保持数据一致性、完整性、满足并发性
- 用商品化关系数据库管理系统

    - 每个事务不可分割，事务封锁账户
- ATM系统中需存储对象主要是账户类对象，两种方法
    - 每个对象自己保存自己
        - 账户类对象接到“存储自己”通知，把自身存储起来
    - 由数据管理子系统负责存储对象
        - 账户类对象接到“存储自己”通知，向数据管理子系统发消息，由数据管理子系统将状态保存起来