



中國石油大學 (华东)
CHINA UNIVERSITY OF PETROLEUM

软件工程



主要内容



第一章 软件工程学概述

第二章 可行性研究

第三章 需求分析

第四章 总体设计

第五章 详细设计

第六章 编码与测试

第七章 软件维护

第八章 面向对象方法学

第九章 面向对象分析设计与实现

第十章 软件项目管理

第九章 面向对象分析设计与实现



第一节 面向对象分析

第二节 面向对象设计

第三节 面向对象实现

第九章 面向对象分析设计与实现



软件生存期各阶段所使用的方法、技术具有高度的连续性，用符合人类认识世界的思维方式来分析、解决问题。将OOA、OOD、OOP有机地集成在一起。

- 面向对象的分析（Object-Oriented Analysis, OOA ）
强调的是对一个系统中的对象特征和行为的定义。建立系统的三类模型。
- 面向对象的设计（Object-Oriented Design, OOD）
与OOA密切配合，顺序实现对现实世界的进一步建模。
- 面向对象的编程（Object-Oriented Program, OOP ）
是面向对象的技术中发展最快的，使用面向对象的程序设计语言，进行编码。

第九章 面向对象分析设计与实现



第一节 面向对象分析

一、面向对象分析的基本过程

1. 概述

面向对象分析就是要解决“做什么”的问题，其基本过程是需求获取 ➡ 建模，该阶段要建立三种模型：

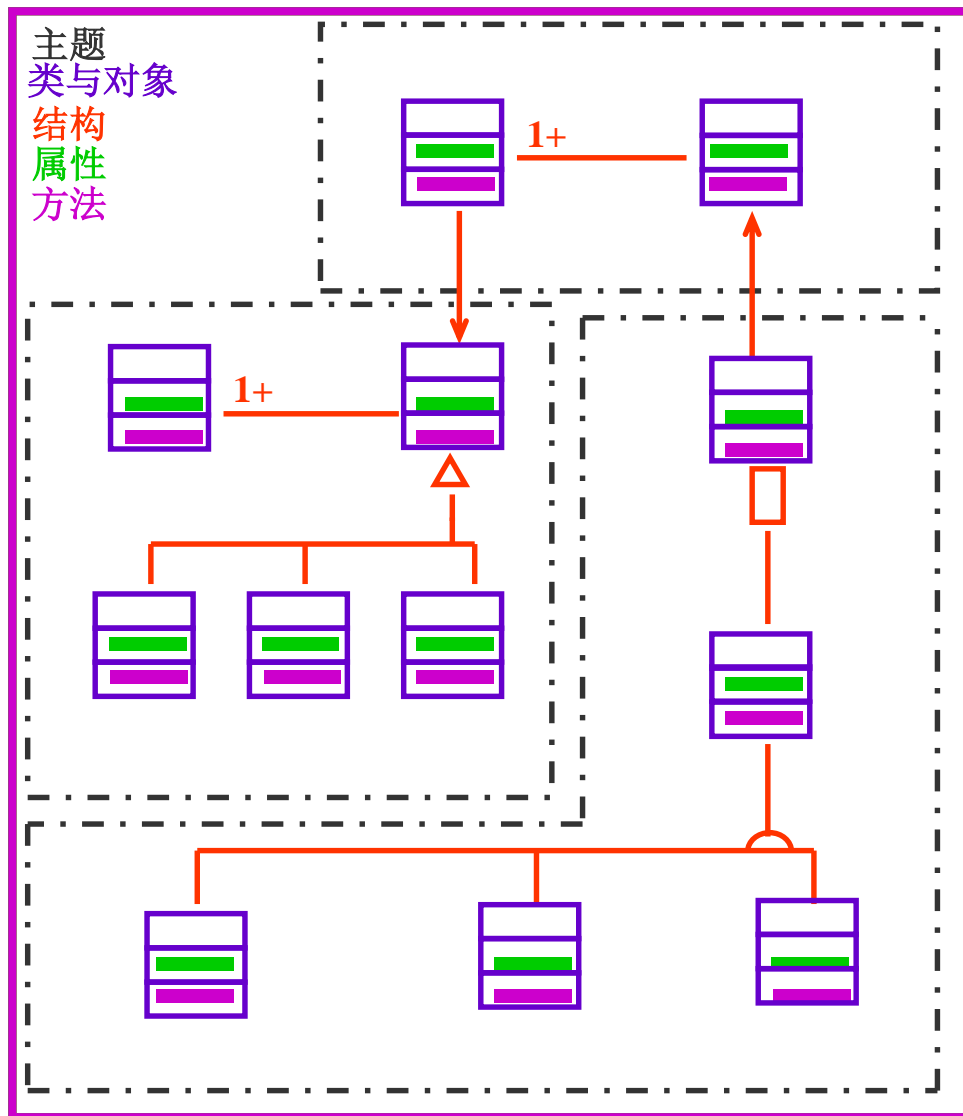
- 对象模型：几乎解决任何一个问题，都需要从客观世界实体及实体间相互关系抽象出对象模型，是最基本、最重要、最核心的。
- 动态模型：当问题涉及交互作用和时序时（如用户界面及过程控制等），动态模型是重要的。
- 功能模型：对于开发大运算量问题（如科学计算、编译系统等）很重要。

第九章 面向对象分析设计与实现



2. 五个层次

- 复杂问题（大型系统）的对象模型通常有五个层次组成：主题层、类与对象层（即UML的“类”）、结构层（即类或对象之间的关系）、属性层和服务层。
- 这5个层次相当于把5张透明胶片叠在一起，每一层显示更多的细节。



第九章 面向对象分析设计与实现



- 主题层

- ✓ 主题是把一组具有较强联系类组织在一起而得到的类的集合，是一种比类和对象抽象层次更高、粒度更大的概念，用以建立系统的高层抽象视图。
- ✓ 在开发大型、复杂系统时，为降低复杂程度，常把系统进一步划分成几个不同的主题，也就是在概念上把系统包含的内容分解成若干范畴。
- ✓ 应该按问题领域而不是用功能分解方法来确定主题。

第九章 面向对象分析设计与实现



二、需求陈述

需求陈述的内容包括：问题范围、功能需求、性能要求、应用环境、假设条件等。

系统分析员必须与用户和领域专家密切配合协同工作，共同提炼和整理用户需求。

第九章 面向对象分析设计与实现



三、建立对象模型

建立对象模型的典型步骤（大体次序）：

- 寻找类和对象
- 识别结构（即确定关联）
- 对于复杂问题则要进一步划分若干主题
- 给类和关联添加属性
- 利用适当的继承关系进一步合并和组织类
- 建立动态模型
- 建立功能模型
- 定义服务

第九章 面向对象分析设计与实现



实例：

自动取款机(ATM)系统：

某银行拟开发一个自动取款机系统，它是一个由自动取款机、中央计算机、分行计算机及柜员终端组成的网络系统。ATM和中央计算机由总行投资购买。总行拥有多台ATM，分别设在全市各主要街道上。分行负责提供分行计算机和柜员终端。柜员终端设在分行营业厅及分行下属的各个储蓄所内。该系统的软件开发成本由各个分行分摊。

第九章 面向对象分析设计与实现



1. 确定类和对象：

第1步：列出所有候选对象(candidates)，它们可能是

- ♠ 物理实体
- ♠ 人或组织
- ♠ 要处理的事件
- ♠ 对象间的活动
- ♠ 抽象概念
- 等等



非正式分析：从需求陈述中挑出

- 名词 → Class-&-Object的候选
- 形容词→确定Attribute的线索
- 动词→Method的候选

第九章 面向对象分析设计与实现



第2步：去粗取精

从ATM需求分析中提出的名词集合

~~银行~~、~~ATM~~、~~系统~~、~~中央计算机~~、~~分行计算机~~、~~柜员终端~~、~~网络~~、
~~总行~~、~~分行~~、~~软件~~、~~成本~~、~~市~~、~~街道~~、~~营业厅~~、~~储蓄所~~、~~柜员~~、
储户、~~现金~~、~~支票~~、~~账户~~、事务、现金兑换卡、~~余额~~、~~磁卡~~、~~分~~
~~行代码~~、~~卡号~~、~~用户~~、~~副本~~、~~信息~~、~~密码~~、~~类型~~、~~取款额~~、~~账单~~、
~~访问~~、~~通信链路~~、~~事务日志~~

筛选时依下列标准删除：

- ♣ 冗余
- ♣ 无关
- ♣ 笼统
- ♣ 属性
- ♣ 实现
- ♣ 操作 既可为名词又可为动词的词，应慎重考虑。

第九章 面向对象分析设计与实现



2. 确定关联

第1步：收集

① 需求陈述中涉及objects的动词短语：

- ATM、中央计算机、分行计算机及柜员终端组成网络
- 总行拥有多台ATM
- ATM设在主要街道上
- 分行提供分行计算机和柜员终端
- 柜员终端设在分行营业厅及储蓄所内
- 分行分摊软件开发成本
- 储户拥有账户
- 分行计算机处理针对账户的事务
- 分行计算机维护账户
- 柜员终端与分行计算机通信
- 柜员输入针对账户的事务
- ATM与中央计算机交换关于事务的信息
- 中央计算机确定事务与分行的对应关系
- ATM读现金兑换卡
- ATM与用户交互
- ATM吐出现金
- ATM打印账单
- 系统处理并发的访问

第九章 面向对象分析设计与实现



② 需求陈述中隐含的关联

- 总行由各个分行组成
- 分行保管账户
- 总行拥有中央计算机
- 系统维护事务日志
- 系统提供必要的安全性
- 储户拥有现金兑换卡

③ 根据问题域知识得出的关联

- 现金兑换卡访问账户
- 分行雇用柜员

第九章 面向对象分析设计与实现



第2步：筛选删除

① 与已删去的类有关的关联

- ~~ATM、中央计算机、分行计算机及柜员终端组成网络~~
- 总行拥有多台ATM
- ~~ATM设在主要街道上~~
- 分行提供分行计算机和柜员终端
- ~~柜员终端设在分行营业厅及储蓄所内~~
- ~~分行分摊软件开发成本~~
- 储户拥有账户
- 总行由各个分行组成
- 现金兑换卡访问账户
- 分行保管账户
- 总行拥有中央计算机
- 分行雇用柜员
- 分行计算机处理针对账户的事务
- 分行计算机维护账户
- 柜员终端与分行计算机通信
- 柜员输入针对账户的事务
- ATM与中央计算机交换关于事务的信息
- 中央计算机确定事务与分行的对应关系
- ATM读现金兑换卡
- ATM与用户交互
- ~~ATM吐出现金~~
- ~~ATM打印账单~~
- ~~系统处理并发的访问~~
- ~~系统维护事务日志~~
- ~~系统提供必要的安全性~~
- 储户拥有现金兑换卡

② 与问题无关的或应在实现阶段考虑的关联

第九章 面向对象分析设计与实现



③ 瞬时事件：关联应该描述问题域的静态结构，不应该是一个瞬时事件。

- ~~ATM、中央计算机、分行计算机及柜员终端组成网络~~
- 总行拥有多台ATM
- ~~ATM设在主要街道上~~
- 分行提供分行计算机和柜员终端
- ~~柜员终端设在分行营业厅及储蓄所内~~
- ~~分行分摊软件开发成本~~
- 储户拥有账户
- 总行由各个分行组成
- 现金兑换卡访问账户
- 分行保管账户
- 总行拥有中央计算机
- 分行雇用柜员
- 分行计算机处理针对账户的事务
- 分行计算机维护账户
- 柜员终端与分行计算机通信
- 柜员输入针对账户的事务
- ATM与中央计算机交换关于事务的信息
- 中央计算机确定事务与分行的对应关系
- ~~ATM读现金兑换卡~~
- ~~ATM与用户交互~~
- ~~ATM吐出现金~~
- ~~ATM打印账单~~
- ~~系统处理并发的访问~~
- ~~系统维护事务日志~~
- ~~系统提供必要的安全性~~
- 储户拥有现金兑换卡

第九章 面向对象分析设计与实现



④ 三元关联：分解为二元关联或限定关联

- 柜员输入针对账户的事务
= 柜员输入事务 + 事务修改账户
- 分行计算机处理针对账户的事务
= 分行计算机处理事务 + 事务修改账户
- ATM与中央计算机交换关于事务的信息
= ATM与中央计算机通信 + 在ATM上输入事务

第九章 面向对象分析设计与实现



⑤ 派生关联：即可用其它关联定义的冗余关联

- 总行拥有多台ATM
- 分行提供分行计算机和柜员终端
- 储户拥有账户
- 总行由各个分行组成
- 分行保管账户
- 总行拥有中央计算机
- 现金兑换卡访问账户
- 储户拥有现金兑换卡
- 分行雇用柜员
- 分行计算机处理事务
- 分行计算机维护账户
- 柜员终端与分行计算机通信
- 柜员输入事务
- 事务修改账户
- ATM与中央计算机通信
- 在ATM上输入事务
- 中央计算机确定事务与分行的对应关系

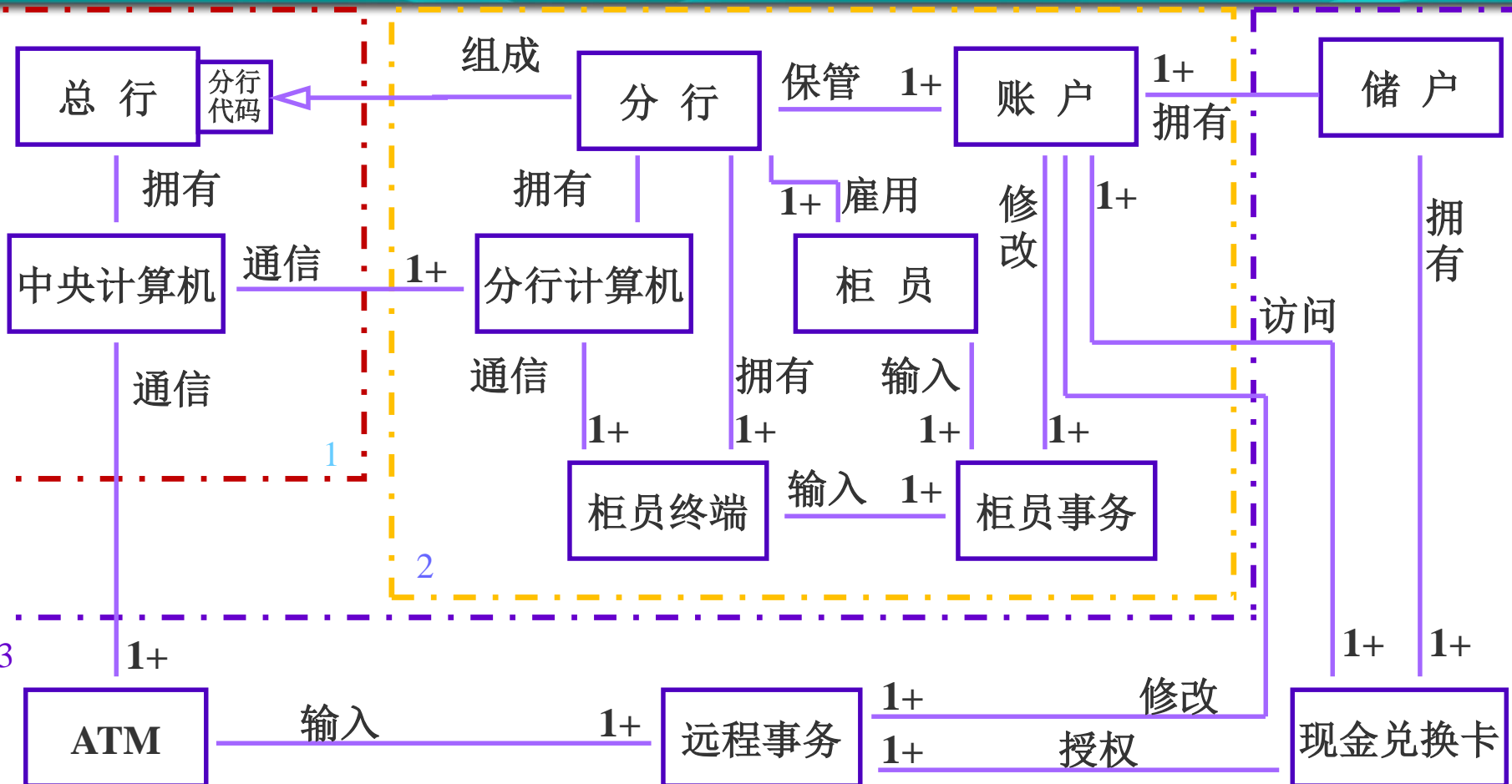
第九章 面向对象分析设计与实现



第3步：进一步完善

- ① 正名：分行提供分行计算机和柜员终端
= 分行拥有分行计算机 + 分行拥有柜员终端
- ② 分解：适当分解前面确定的类与对象, 使其适用于不同的关联事务 = 远程事务 + 柜员事务
- ③ 补充：发现了遗漏的关联就应该及时补上
 - 柜员输入柜员事务
 - 在ATM上输入远程事务
 - 柜员事务输进柜员终端
 - 远程事务由现金兑换卡授权
- ④ 标明重数（可能经常变动，不要花太多时间）

ATM系统原始类图



3. 划分主题

1. 总行 2. 分行 3. ATM

- ① 按问题域而不是按功能分解; ② 主题间的依赖和交互尽可能少

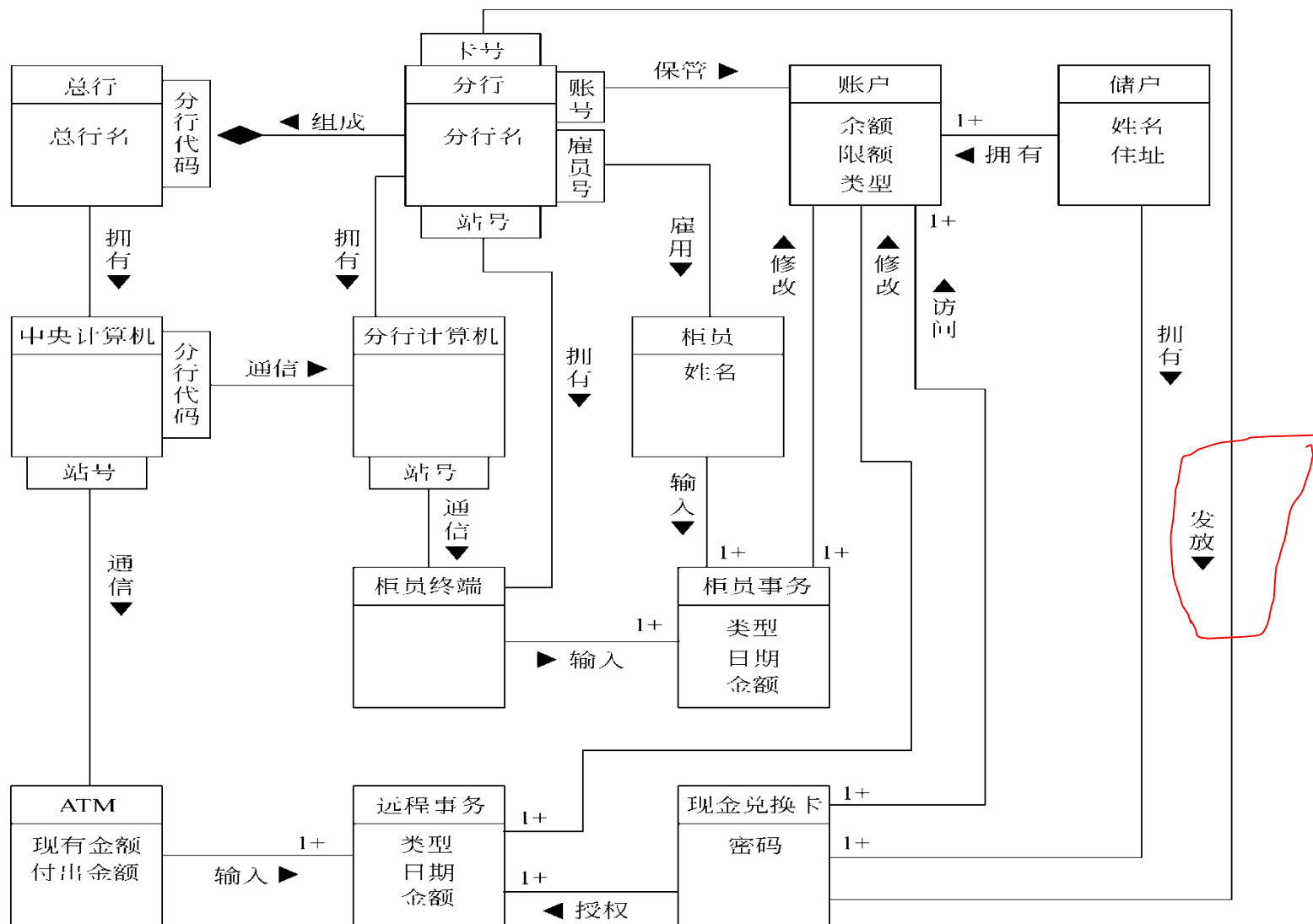
第九章 面向对象分析设计与实现



4. 确立属性

- 在分析过程中应该首先找出最重要的属性，以后再逐渐把其余属性添加进去；
- 应该仅考虑与具体应用直接相关的属性；
- 暂不考虑纯用于实现的 属性；
- 需求陈述中与已确定的 objects 有关的名词、形容词可能是选择的线索。

ATM系统对象模型中的属性



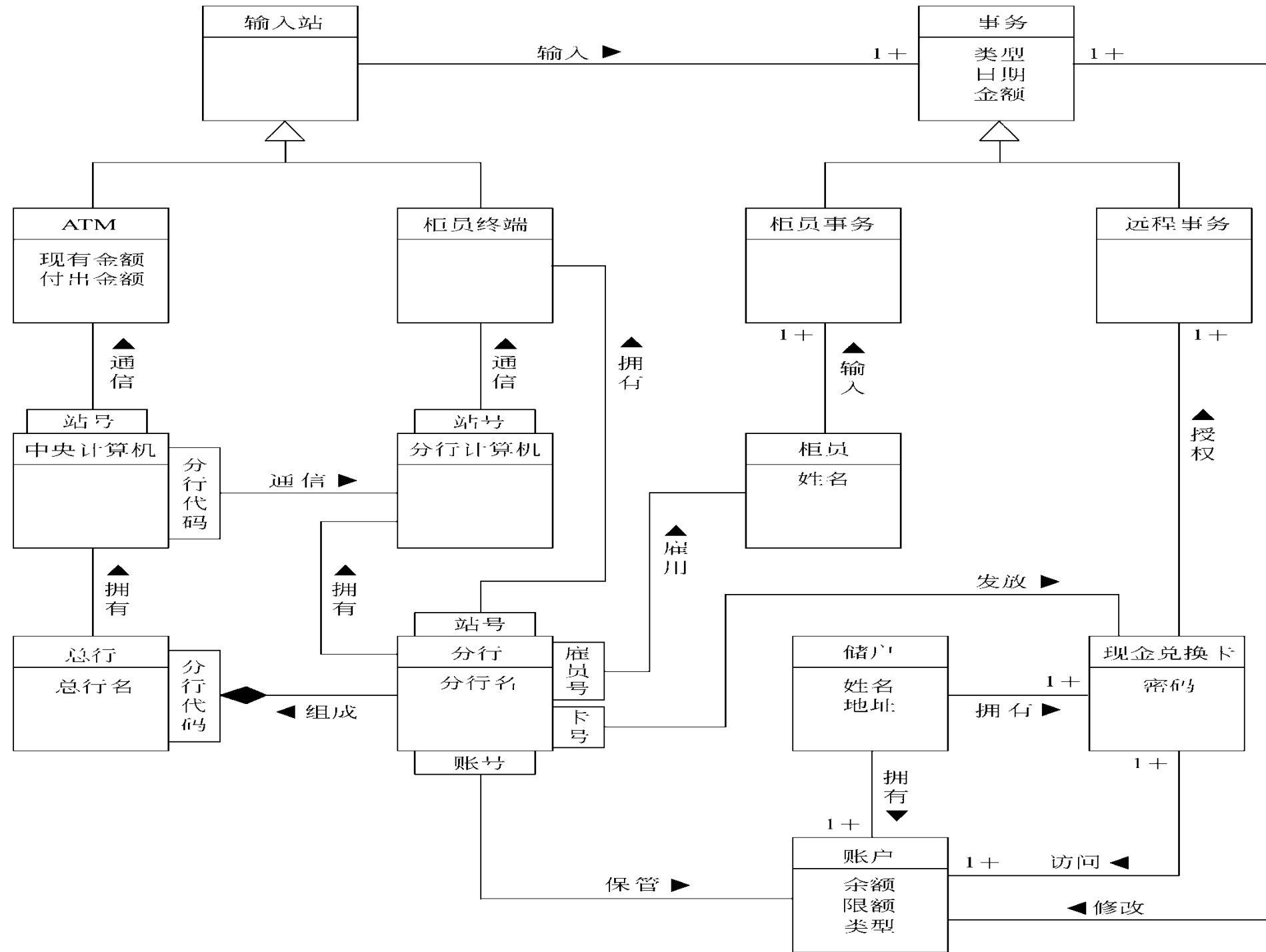
第九章 面向对象分析设计与实现



5. 识别继承关系及其它修改

- 建立继承（即泛化）关系的方法：

- ① 自底向上：将具有相同属性的类向上归纳出父类。如：“远程事务”和“柜员事务”是类似的，泛化出父类“事务”；类似地，“ATM”和“柜员终端”泛化出父类“输入站”。
- ② 自顶向下：将现有类向下细化出子类（但分析阶段避免过度细化）。



第九章 面向对象分析设计与实现



反复修改

- 合并无须分别考虑的若干类。

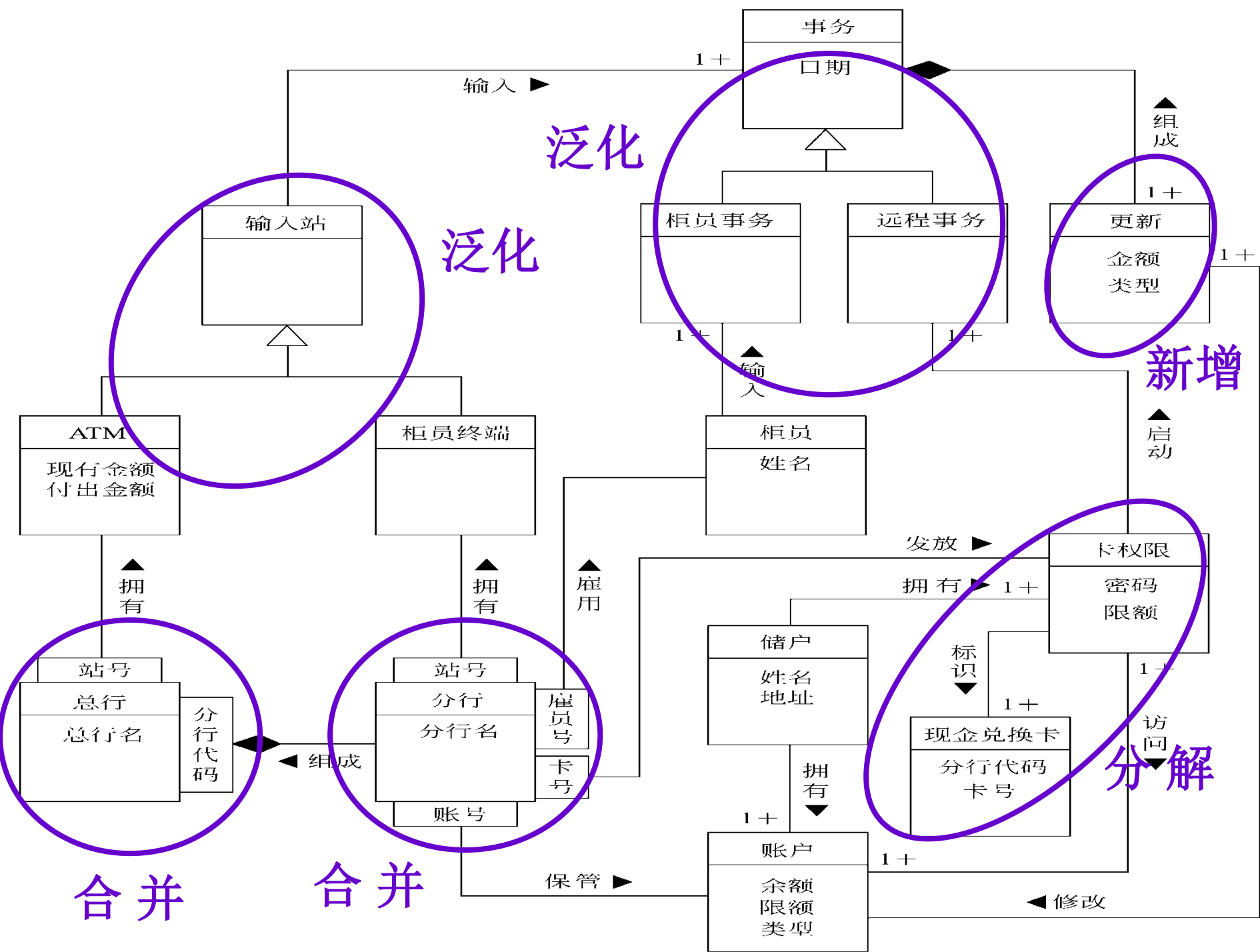
如“分行”“分行计算机”，对于该系统无区分价值，为简单起见，合并；类似地，“总行”与“总行计算机”合并。

- 若某类中具有几个独立的功能，则考虑分为几个类。

如“现金兑换卡”有2个相对独立的功能，它既是鉴别储户使用ATM的权限的卡，又是ATM获得分行代码和卡号等数据的数据载体，为使每个类的功能单一，可分解为“卡权限”和“现金兑换卡”2个类。

- 对于某类中具有自己特有属性的部分，可考虑将之列为独立存在的类，且是原有类的组成。

如事务由更新组成。一个事务包含对账户的若干次更新（即对账户所做的一个动作，如取款、查询等），“更新”虽然代表一个动作，但它有自己的属性（如类型、金额等），应该独立存在，作为一个类。





四、建立动态模型

- 建立动态模型的基本步骤：

- （1）编写典型交互行为的脚本；

- （2）从脚本中提取事件，确定触发每个事件的动作对象以及接受事件的目标对象；

- （3）排列事件发生的次序，确定每个对象可能有的状态及状态间的转换关系，并用状态图描绘他们。

- （4）比较各个对象的状态图，检查他们之间的一致性，确保事件之间的匹配。

第九章 面向对象分析设计与实现



1. 编写脚本

- 脚本描述外部实体与目标系统之间的一个或多个典型的交互过程，以便于对目标系统行为有更具体的认识。
- 脚本中不可能包含每个偶然事件，但至少必须保证不遗漏常见的交互行为。
- 编写脚本时，首先编写正常情况的脚本；然后，考虑特殊情况的脚本；最后，考虑出错情况的脚本。

第九章 面向对象分析设计与实现



例：ATM系统的正常情况脚本

- ATM请储户插卡；储户插入一张现金兑换卡。
- ATM接受该卡并读它上面的分行代码和卡号。
- ATM要求储户输入密码；储户输入自己的密码“1234”等数字。
- ATM请求总行验证卡号和密码；总行要求“39”号分行核对储户密码，然后通知ATM说这张卡有效。
- ATM要求储户选择事务类型(取款、转帐、查询等)；储户选择“取款”。
- ATM要求储户输入取款额；储户输入“600”。
- ATM确认取款额在预先规定的限额内，然后要求总行处理这个事务；总行把请求转给分行，该分行成功地处理完这项事务并返回该帐户的新余额。
- ATM吐出现金并请储户拿走这些现金；储户拿走现金。
- ATM问储户是否继续这项事务；储户回答“不”。
- ATM打印帐单，退出现金兑换卡，请储户拿走它们；储户取走帐单和卡。
- ATM请储户插卡

第九章 面向对象分析设计与实现



例：ATM系统的异常情况脚本

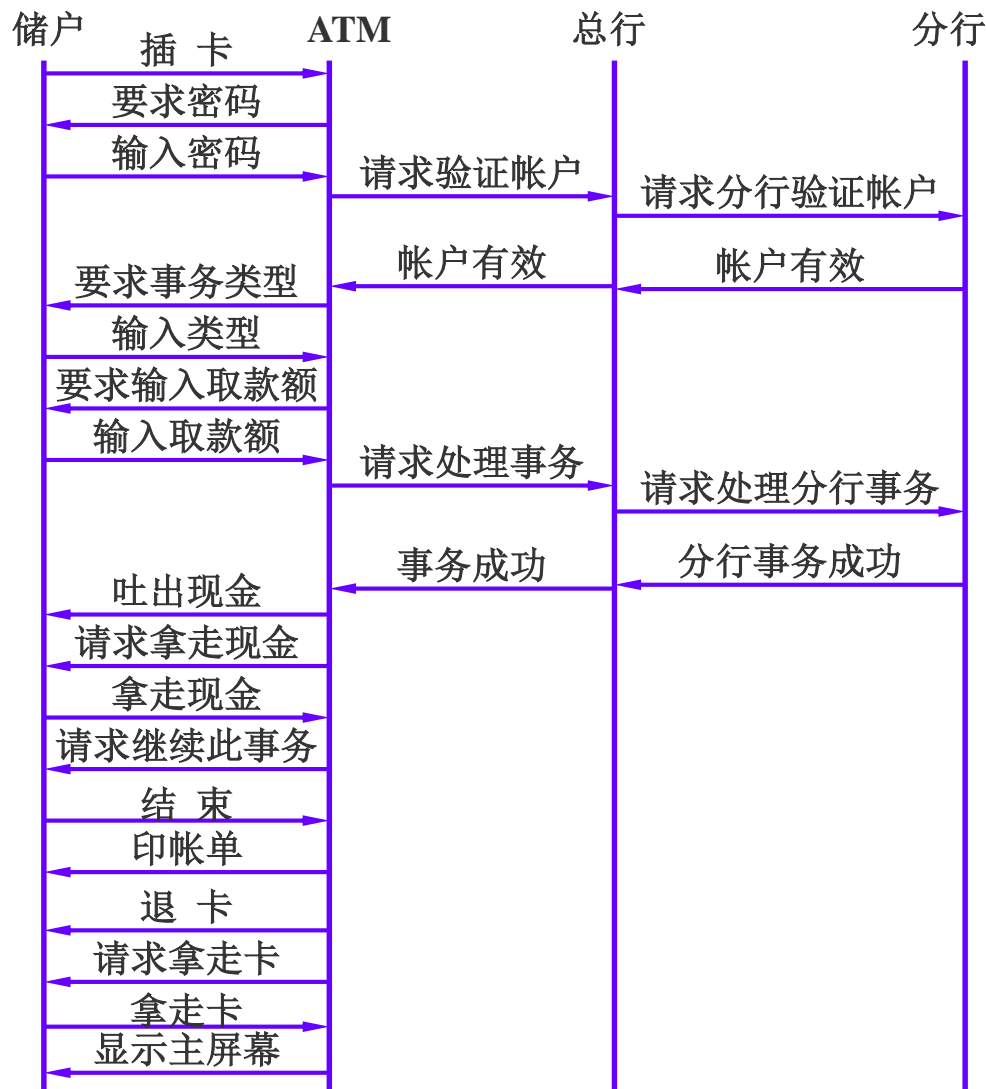
- ATM请储户插卡；储户插入一张现金兑换卡。
- ATM接受该卡并读它上面的分行代码和卡号。
- ATM要求储户输入密码；储户误输入“8888”。
- ATM请求总行验证卡号和密码；总行要求“39”号分行核对储户密码，然后通知ATM拒绝这张卡。
- ATM显示“密码错”，并请储户重新输入密码；储户输入“1234”；
- ATM请总行验证后知此次输入的密码正确。
- ATM要求储户选择事务类型(取款、转帐、查询等)；储户选择“取款”。
- ATM要求储户输入取款额；储户改变主意不想取款了，敲“取消”键。
- ATM退出现金兑换卡，请储户拿走它；储户取走他的卡。
- ATM请储户插卡

第九章 面向对象分析设计与实现



2. 画事件跟踪图

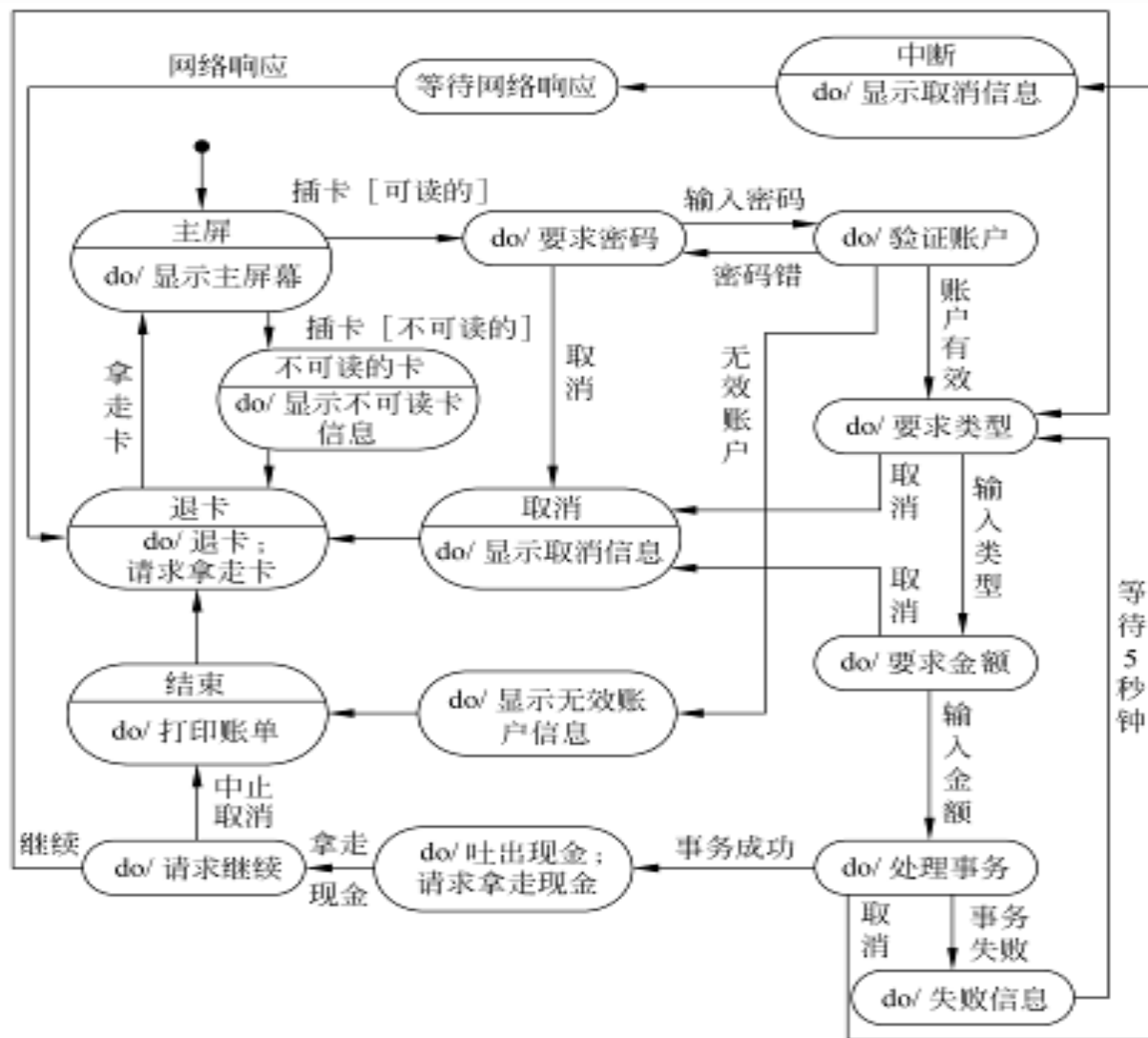
- 用自然语言书写的脚本往往不够简明，有时会有二义性。
- 事件跟踪图中，一条竖线代表一个对象，每个事件用一条水平的箭头线表示，箭头方向从事件的发送对象指向接受对象。



- 系统分析员应该集中精力仅考虑具有重要交互行为的那些类。

- 各个类的状态转换图通过共享事件合并起来，构成系统的动态模型。

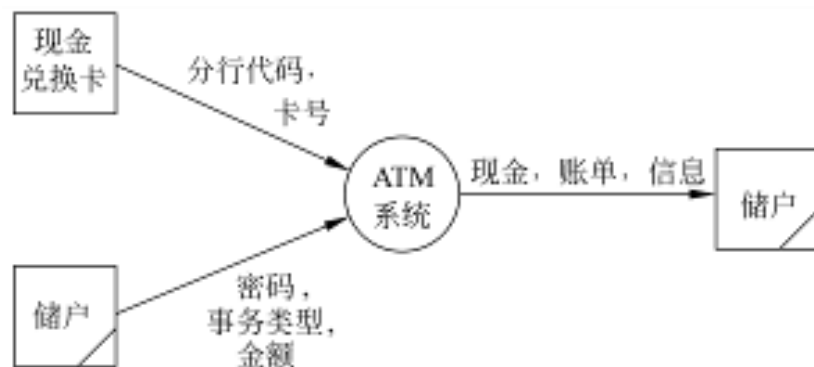
- 检查完整性和一致性。



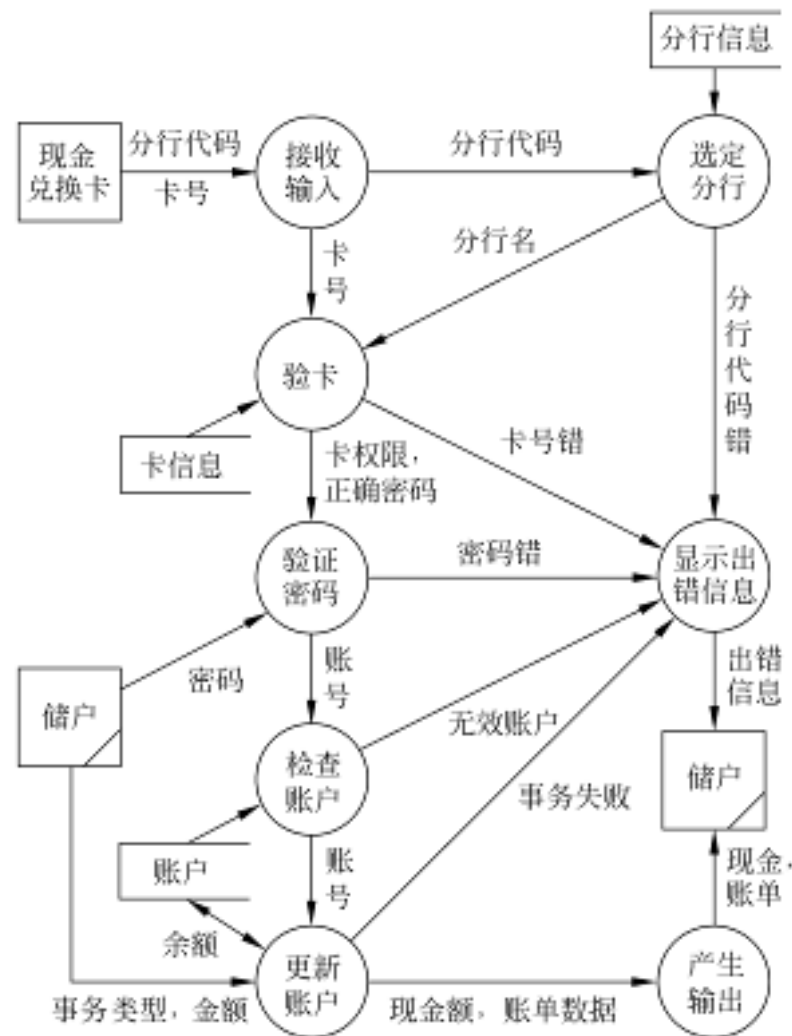
第九章 面向对象分析设计与实现



五、建立功能模型



ATM系统的顶层数据流图



ATM系统的功能级数据流图

第九章 面向对象分析设计与实现



六、定义服务

- 常规行为

假定在每个类中都定义了读、写该类每个属性的操作，无需在类图中显式表示这些常规操作。

- 从事件中导出操作

状态图中发往对象的事件也就是该对象接收到的消息，因此该对象必须有由消息选择符指定的操作，这个操作修改对象状态（即属性值）并启动相应的服务。

- 从数据流图找出服务

数据流图中的每个处理框都与一个对象（也可能是若干个对象）上的操作相对应。

- 利用继承机制减少冗余操作

第九章 面向对象分析设计与实现



第二节 面向对象设计

一、面向对象设计的准则

1. 模块化：模块=对象
2. 抽象：抽出事物的本质特性，暂不考虑其细节，使设计从具体实现方法中超脱。
 - 过程抽象：在SD中已讨论。
 - 数据抽象：Class即是一种抽象数据类型。外界无须知道实现方法，按照类规格说明（即协议）可以使用合法的操作符，利用这些操作对类实例中包含的数据进行操作。
 - 参数抽象：将数据类型作为参数处理。

第九章 面向对象分析设计与实现



3. 信息隐藏：信息隐藏 = 对象的封装

4. 耦合：

- 交互耦合：通过传递message发生。

要求：降低参数个数和参数复杂性；减少objects发送/接收message的个数。as loose as possible

- 继承耦合：as high as possible

5. 内聚：

- 服务内聚：一个服务只完成一个功能；
- 类内聚：一个类只有一个用途，否则分解之；
- 一般-特殊内聚：应该是对相应领域知识的正确抽取。

6. 可重用：

- 尽量使用已有的类
- 设计新类时考虑将来的可重复使用性

第九章 面向对象分析设计与实现



二、启发规则

1. 设计结果清晰易懂，应做到：

- ① 用词一致 —— 按习惯用法命名
- ② 使用已有的协议
- ③ 尽量减少message模式的数目
- ④ 避免模糊定义

2. 一般-特殊结构的深度应适当（约100个classes, 则设计 7 ± 2 层）

3. 设计简单的类:类的定义要明确，避免包含过多的属性和服务。

4. 使用简单的协议

5. 使用简单的服务

6. 把设计变动减至最小

第九章 面向对象分析设计与实现



三、软件重用


1. 概念

也叫复用，是指同一事物不作修改或稍加改动就多次重复使用。广义上，软件重用可分为以下3个层析：

- ① 知识重用（例如软件工程知识的重用）
- ② 方法和标准重用（例如面向对象方法和软件开发规范的重用）
- ③ 软件成分的重用

2. 软件成分的重用

① 代码重用：

- 源码剪贴 —— 无法溯源，无配置管理
- 源代码包含 —— 修改后所有包含此段代码的程序须重新编译
- 继承 —— 无须改动原有代码 

② 设计重用 —— 当移植系统时

③ 分析重用 —— 当需求未变，而系统结构改变时

第九章 面向对象分析设计与实现



四、系统分解

- 系统的主要组成部分称为子系统，大多数系统的面向对象模型在逻辑上可分解为4部分：

- ① 问题域子系统
- ② 人机交互子系统
- ③ 任务管理子系统
- ④ 数据管理子系统



- 这4个子系统相当于面向对象模型的4个垂直切片，有利于降低设计的难度、有利于分工协作、有利于维护人员对系统的理解和维护。

第九章 面向对象分析设计与实现



1. 问题域子系统的设计

基于面向对象分析建立的对象模型进行补充或修改，主要工作可能体现在以下方面：

① 调整需求

② 重用已有的类：

- ✓ 选出可重用的类，标出与本问题无关的属性和服务；
- ✓ 派生出问题域类，标出继承来的属性和服务，无需在问题域类内定义了；
- ✓ 修改与问题域类相关的关联

③ 添加一般化类以建立协议：设计时会发现有些具体类需要有一个公共的协议，即需要定义一组类似的服务，此时可以引入一个附加类（如根类）建立这个协议。

④ 调整继承关系

第九章 面向对象分析设计与实现



2. 人机交互子系统的设计

在对人机交互部分的设计中，使用由原型支持的系统化的设计策略，是成功设计人机交互子系统的关键。

3. 任务管理子系统的设计

基于面向对象分析建立的动态模型。

① 分析并发性：若两个对象之间无交互行为，或它们同时接受事件，则它们本质上是并发的。

② 确定task类型，并分配给适当的软/硬件去执行

- 事件驱动型：主要完成通信工作
- 时钟驱动型：完成周期性工作

第九章 面向对象分析设计与实现



- 优先型：将高优先级或低优先级的任务分离出来先做或后做。
- 关键任务：指关系系统成败的处理，要求高可靠性，应分离考虑，严格测试。
- 协调任务：当系统中存在三个以上tasks时，应增设一个协调任务，用于封装不同tasks之间的协调控制。

设计者在决定到底采用软件还是硬件实现的时候，需要综合考虑一致性、成本、性能等多种因素，还应考虑未来的可扩充性和可修改性。

第九章 面向对象分析设计与实现



4. 数据管理子系统的设计

设置数据管理部分的主要原因：将特定的数据库管理技术与其它部分隔离开来，避免数据存储管理模式的改变对系统的影响。

(1) 设计数据格式

- 文件管理系统
- 关系数据库管理系统
- 面向对象数据库管理系统

(2) 设计相应服务

第九章 面向对象分析设计与实现



五、设计类中的服务

面向对象分析得出的对象模型，通常并不详细描述类中的服务。
面向对象设计则是扩充、完善和细化面向对象分析模型的过程，
设计类中的服务即是工作内容之一。

(1) 确定类中应有的服务

- 需要综合考虑对象模型、动态模型和功能模型，才能正确确定类中应有的服务；
- 面向对象分析得出的对象模型，通常只在每个类中列出很少几个最核心的服务，设计者必须把动态模型中对象的行为以及功能模型中的数据处理，转换为由适当的类所提供的服务。

(2) 设计实现服务的算法

第九章 面向对象分析设计与实现



六、设计关联

在对象模型中，关联是联结不同对象的纽带，它指定了对象间相互访问的路径。

(1) 单向关联



- 由雇员找其所属公司，则设雇主为其属性，即一单向指针。



- 由公司找其下属某一雇员，则有两种方法：
方法1：遍历所有雇员，找雇主匹配且满足特征的雇员。
(省空间)

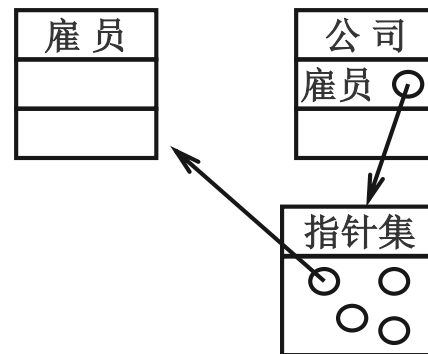
第九章 面向对象分析设计与实现



- 由公司找其下属某一雇员，则有两种方法：

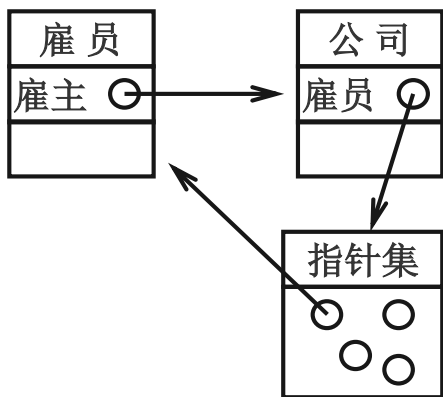
方法2：设公司的属性雇员为一指针集。

（快速）

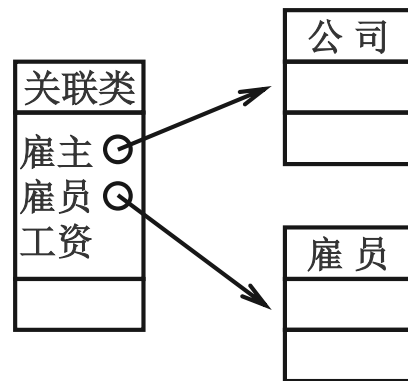


(2) 双向关联

方法1：将上述两种单向关联结合使用。



方法2：另设关联类（特别适用于链属性）



第九章 面向对象分析设计与实现



第三节 面向对象实现

面向对象实现主要包括两项工作：

- 把设计结果翻译成用面向对象程序语言书写的程序；
- 测试并调试面向对象的程序。

一、程序设计风格

1. 提高可重用性

实现阶段的代码重用有两种：

- 项目内代码重用：找出设计中相同或相似的部分，然后利用继承机制共享它们；
- 新项目重用旧项目代码：长远眼光，反复考虑精心设计。

第九章 面向对象分析设计与实现



2. 提高可扩充性

- ① 封装实现策略：实现类的封装，对外只提供公有的接口，将提高今后修改类的数据结构或算法的自由度。
- ② 不要用一个方法遍历多条关联链：违反这条准则将导致方法过分复杂，既不易理解，也不易修改扩充。
- ③ 避免使用多分支语句：不要用来根据对象类型选择应有的行为，否则在增添新类时将不得不修改原有的代码。
- ④ 精心确定公有方法：

公有方法是向公众公布的接口。对这类方法的修改往往会涉及许多其他类，因此，修改公有方法的代价通常都比较高。

第九章 面向对象分析设计与实现



3. 提高健壮性

程序员在编写实现方法的代码时，既应该考虑效率，也应该考虑健壮性。为提高健壮性应该遵守以下几条准则：

- ① 预防用户的操作错误
- ② 检查参数的合法性
- ③ 不要预先确定限制条件

在设计阶段，往往很难准确地预测出应用系统中使用的数据结构的最大容量需求。因此不应该预先设定限制条件。

- ④ 先测试后优化

应该在为提高效率而进行优化之前，先测试程序的性能；经过测试，合理地确定为提高性能应该着重优化的关键部分。

第九章 面向对象分析设计与实现



二、面向对象测试策略

1. 面向对象的单元测试

测试单元为封装的类和对象，但不能孤立地测试单个操作，应把操作作为类的一部分来测试。

2. 面向对象的集成测试

- 集成测试的策略有：

- ① 基于线程的测试(Thread-based testing)

这种策略把响应系统的一个输入或一个事件所需要的那些类集成起来。分别集成并测试每个线程，同时应用回归测试以保证没有产生副作用。

第九章 面向对象分析设计与实现



② 基于使用的测试(Use-based testing)

这种方法首先测试几乎不使用服务器类的那些类（称为独立类），把独立类都测试完之后，再测试使用独立类的下一个层次的类（称为依赖类）。对依赖类的测试一个层次一个层次地持续进行下去，直至把整个软件系统构造完为止。

3. 面向对象的确认测试

类似传统的确认测试和系统测试，集中检查用户可见的动作和用户可识别的输出。可以根据动态模型和描述系统行为的脚本来设计测试用例，用黑盒法。

第九章 面向对象分析设计与实现



小结

- 从面向对象分析到面向对象设计是一个逐渐扩充、完善和细化OOA模型的反复迭代过程；
- 面向对象实现应具有良好的程序设计风格，应在继承传统的测试技术的同时重点关注对象类。



Thank
You