



第八章

面向对象开发方法

[返回总目录](#)

教学目的

本章主要介绍信息系统的面向对象开发方法

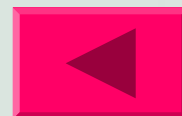
- 介绍面向对象的一些概念
- 介绍UML面向对象分析和设计

教学要求

- 了解面向对象的一些概念
- 了解UML面向对象分析
- 了解UML面向对象设计

面向对象开发方法

- 面向对象开发方法概述
- 使用UML进行面向对象分析与建模
- 使用UML进行面向对象设计与动态建模
- 使用UML进行面向对象开发实例
- 小结



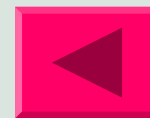
面向对象开发方法

第一节 信息系统设计概述



面向对象开发方法概述

- ❖ 对象、属性、方法和封装
- ❖ 类、概化和特性
- ❖ 对象/类的关系
- ❖ 消息和消息的发送
- ❖ 多态性



对象、属性、方法和封装

对象 (Object)

- 某种存在的，或者能够被看到、触摸或者以其他方式感觉到的事物，可是人、地点、事物或事件
- 一个学生、教师、供应商和客户都是人为对象
- 一个特定的办公室、建筑物和房间都是地点对象
- 一个产品、车辆、设备、录像带或者显示器上的一个窗口都是事物对象
- 一个订单、支付、发票、注册和预约等都是事件对象

对象、属性、方法和封装

属性 (Attribute)

- 是关于一个对象相关特征的数据
- 对于每个客户，属性将指定那个特定客户的值，如4123，王宏，吉林市长春路123号，64578910，中等
- 每个单独的客户就是一个对象实例

对象、属性、方法和封装

对象实例 (Object Instance)

- 由描述特定的人、地点、事物或者事件的属性值构成
- 随着技术的进步，对象除了上面列举的4类对象，还有位图、图片、声音和视频等

对象、属性、方法和封装

行为 (Behavior)

- 指的是对象可以做的事情，以及在对象数据（或属性）上执行的功能
- 在面向对象环境中，对象的行为通常被称为方法、操作或者服务
- 如电话可以应答、拨号、挂断，还可以执行其他与其相关的操作

对象、属性、方法和封装

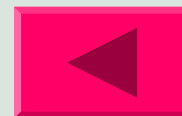
面向对象的原理

- 对象单独地负责执行任何在其数据（或属性）上操作的功能或者行为
- 如你可以修改账户的姓名和地址等
- 对象属性和行为都被封装在那个对象的内部，访问或修改对象属性只能通过对象的行为来实现。

对象、属性、方法和封装

封装 (Encapsulation)

- 是几项内容一起被打包成一个单元（也称作信息隐蔽）
- 在对象模型中对象实例用包含对象实例名称的矩形表示
- 对象实例名字由唯一地标识对象的属性值、冒号以及对象所属的类的名称构成



类、概化和特化

类（Class）是共享相同属性和行为的对象集合
类有时也被称作对象类

学生

只显示名称

学生
学号
姓名
年龄
.....

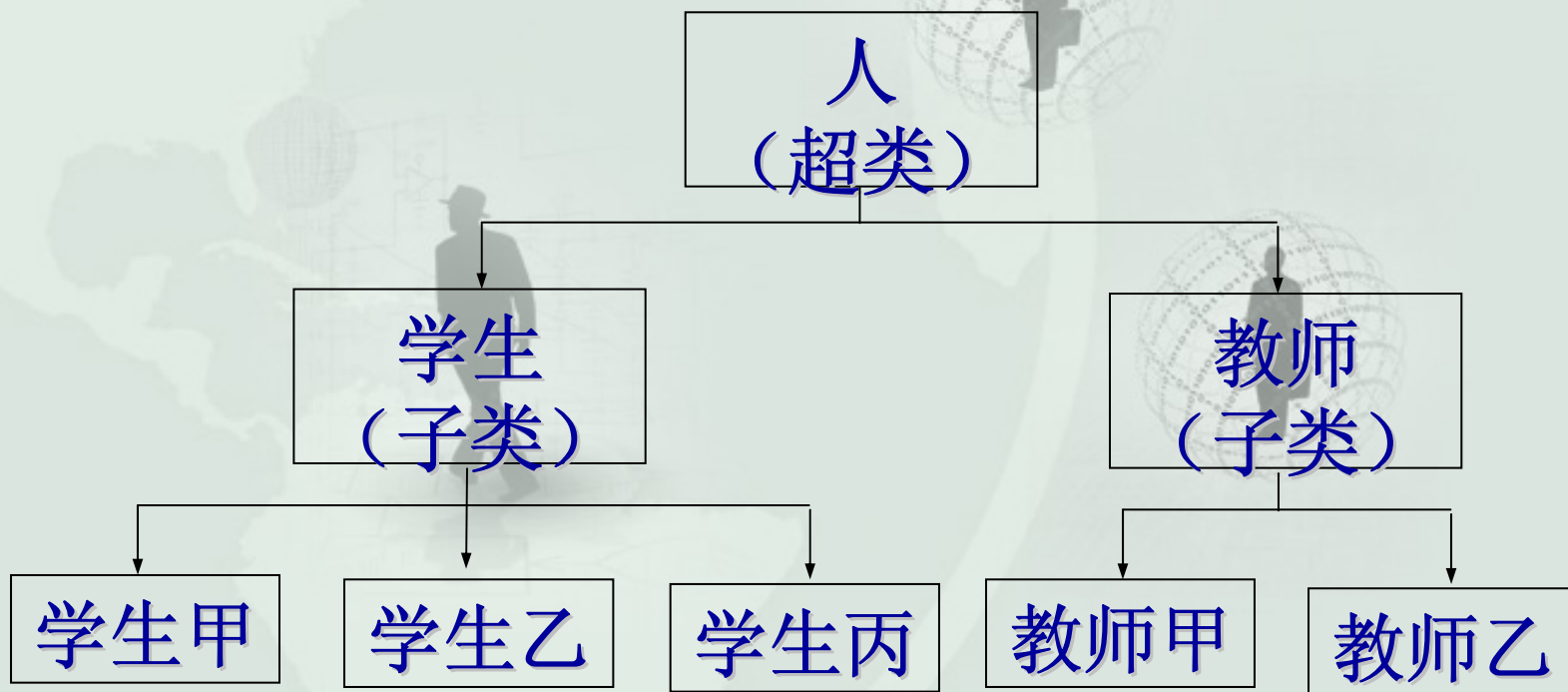
显示名称和属性

学生
学号
姓名
年龄
.....
Talk ()
Walk ()

显示名称、属性和行为

类、概化和特化

继承（Inheritance）是指在一个对象类中定义的方法和/或属性可以被另一个对象类继承或复用。



继承的关系

类、概化和特化

概化/特化 (Generalization/specialization)

是一种技术

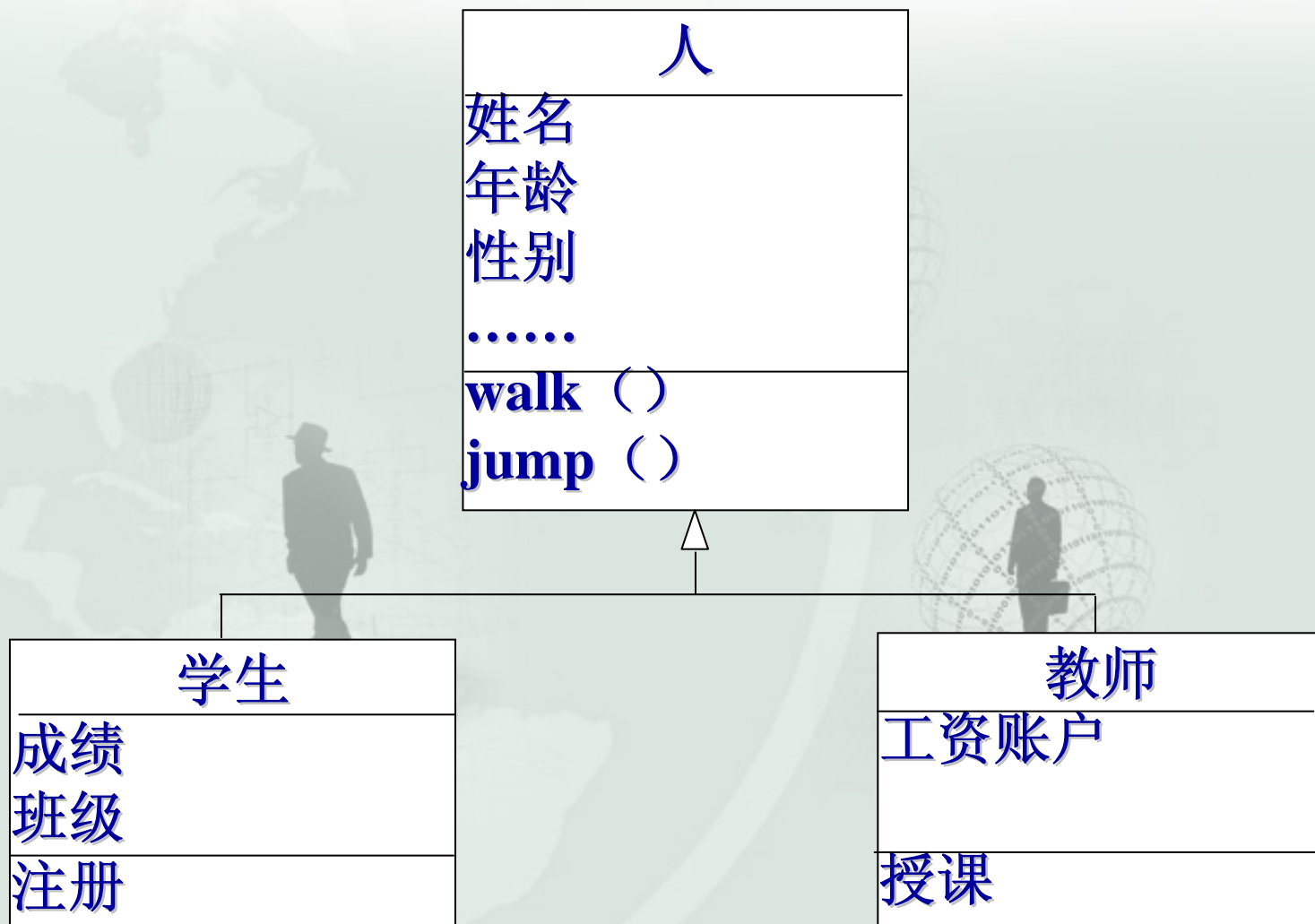
几类对象类的公共属性和行为被组合成类，称为超类（如“人”）

超类的属性和方法可以被子类继承。

类、概化和特化

- 超类（**super type**）是包含一个或多个对象子类的公共属性和行为的实体，也称为父类或抽象类
- 子类（**subtype**）是一个对象类，它从一个超类继承属性和行为并可能包含自身所特有的属性和行为，也称为儿子类
如果它位于继承层次的最底层，也称为实类
- 子类和超类的关系是“**is a**”

类、概化和特化



使用UML表示概化/特化关系

对象、类的关系

- 对象/类关系（**Object/Class Relationship**）是一种存在于一个或多个对象/类之间的自然业务联系

对象/类关联和重数符号

重数	UML重数 记号	关联的含义
一个	1或空白	一个教师为一个且仅为一个学校工作
零个或一个	0..1	一个教师具有一个配偶或没有配偶
零个或多个	0..**	一个教师可以不教学生，也可以教多名学生
一个或多个	1..*	一个大学提供一门及以上的课程
特定范围	1..3	一个教师安排1、3或3门课程

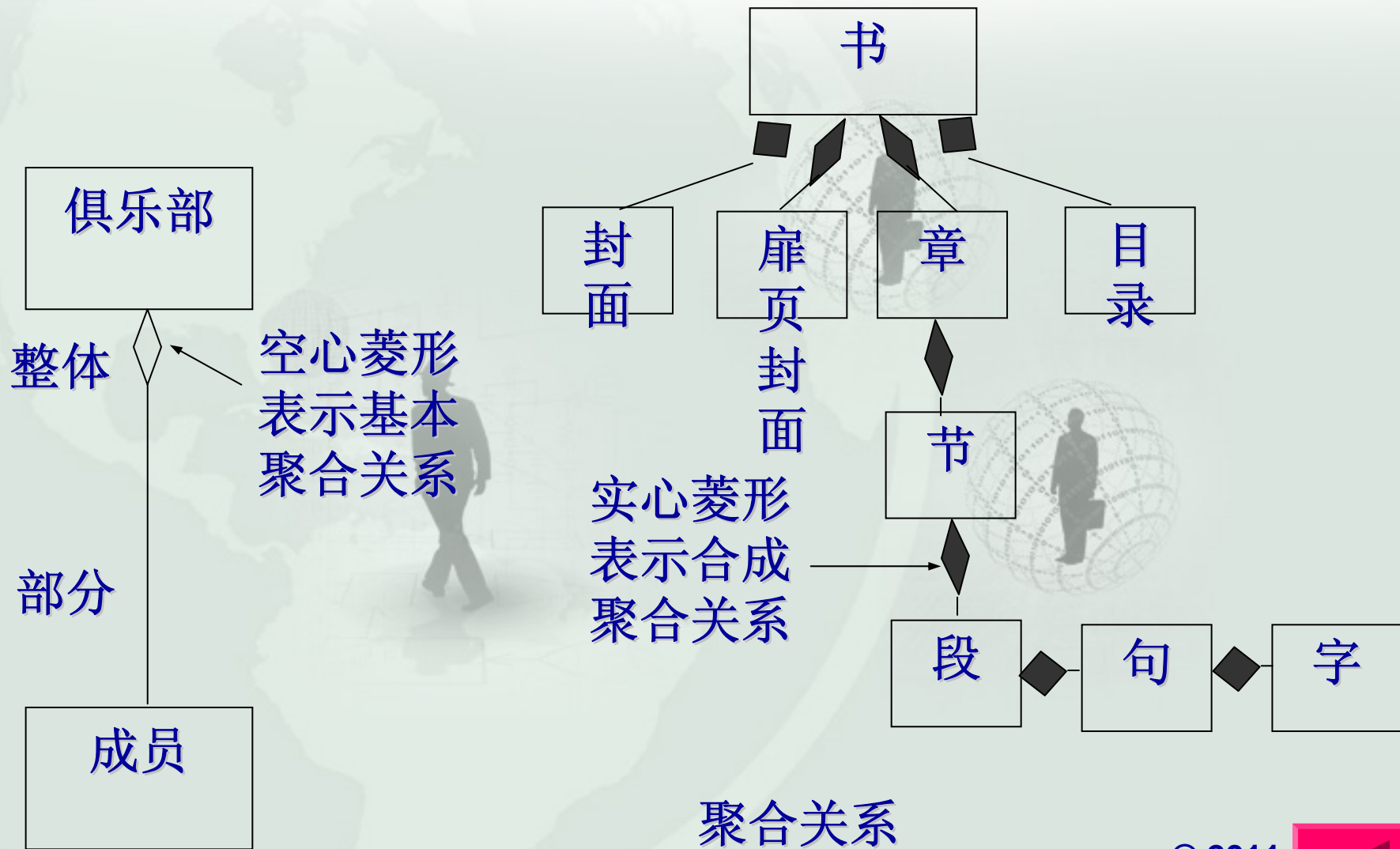
对象、类的关系

- 对象/类之间的特殊种类的关系称为聚合（**Aggregative**）
- 聚合是一种关系，其中一个较大的“整体”类包含一个或多个较小的“部分”类
- 相反地，一个较小的“部分”类是一个较大的“整体”类的一部分

对象、类的关系

- 通过聚合关系可以分解复杂的对象，并给其中的每个对象分配行为和属性
- 聚合关系又分为基本聚合和合成（**Composition**）聚合
- 合成是一种强形式的聚合，“整体”对象完全拥有“部分”对象，“部分”对象只能与一个“整体”对象有关联，“整体”不存在了，“部分”也就不存在了。

对象、类的关系



消息和消息发送

- 消息（**Message**）是当一个对象调用另一个对象的方法（行为）以请求信息或者某些动作时发出的通信
- 如当客户对象检查订单的当前状态时，发送一条消息给订单对象，通过调用订单对象显示状态行为

消息和消息发送

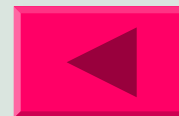


多态性

- 一个与消息密切相关的重要概念是多态性（**Polymorphism**）
- 多态性是指多种形式，即不同的对象可以以不同的形式响应同样的消息
- 当超类的行为需要被子类的行为重载时，就需要在面向对象应用中使用多态性

多态性

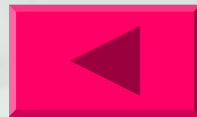
- 重载（**Override**）是一种技术，其中子类使用它自己的属性或行为，而非从父类继承的属性或行为
- 多态性和消息发送的关系是请求服务的对象知道要请求什么服务（或行为）和从哪个对象请求，但是请求对象不需要关心行为如何实现。



面向对象开发方法

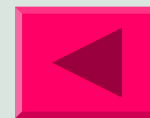
第二节

使用UML进行面向对象分析与建模



面向对象开发方法概述

- ❖ 面向对象分析
- ❖ 用例建模
- ❖ 用例活动建模
- ❖ 教学管理的静态结构图



面向对象分析

- 面向对象分析（**OOA**）是用面向对象方法开发管理信息系统的一个重要阶段
- 一般需要建立**3**个系统模型：对象静态模型、对象动态模型和系统功能模型
- 这**3**个模型称为系统的分析模型
- 系统分析主要是获取需求，建立需求模型

面向对象分析

面向对象分析活动:

(1) 获取领域知识

(2) 定义系统功能

- 定义系统的功能模型可以采用数据流图，也可以采用用例图，在 **UML** 中采用用例图表示。

(3) 确定合适的类

- 根据需求陈述以及领域知识和经验确定系统的类。

(4) 建立类的静态模型

- 对象类的静态模型描述了系统的静态结构，包括构成系统的类和对象，它们的属性、操作以及这些对象之间的联系，通常用类图和对象图表示。

面向对象分析

面向对象分析活动:

(5) 描述对象的动态行为

系统的动态行为模型包括对象状态模型和交互行为模型，对象状态模型由状态图和活动图组成，对象交互模型由协作图和时序图组成

(6) 验证

- 专家对前面完成的模型作静态验证，以便确定系统的正确性

(7) 给出基本的用户界面原型

- 给出系统整体结构的原型，包括主窗口的内容、窗口之间的导航等

面向对象分析

面向对象分析：

- 面向对象分析过程从了解用户需求开始
- 需求分析阶段的工作是在客户和软件开发人员之间沟通基本的需求
- 与问题领域的专家进行讨论，分析领域的业务范围、业务规则和业务处理过程
- 明确系统的责任、范围和边界
- 确定系统的需求，形成需求陈述

面对对象分析

需求陈述的内容:

问题范围

功能需求

性能需求

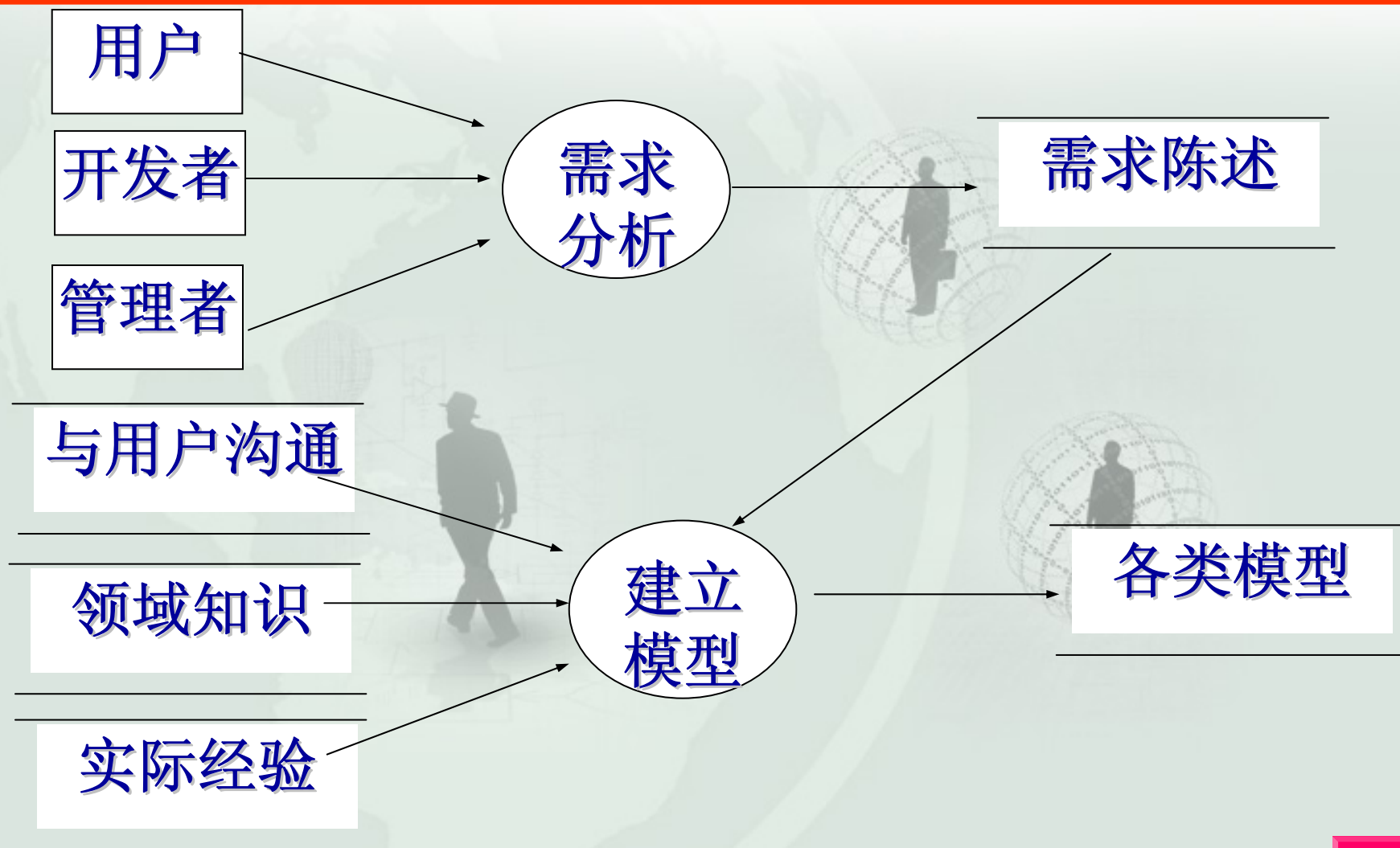
应用环境及假设条件等

- 需求陈述应该阐明系统“做什么”而不是“怎样做”
- 它应该描述用户的需求而不是提出解决问题的方法
- 书写需求陈述要尽力做到语法正确，而且应该慎重选用名词、动词、形容词和同义词

面向对象分析

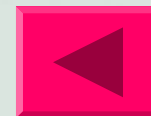
- 系统分析员应该深入理解用户需求
- 抽象出目标系统的本质属性
- 并用模型准确地表示出来
- 通过对象建模记录确定的对象、对象封装的数据和行为以及对象之间的关系

面对对象分析



用例建模

- ❖ 用例建模概述
- ❖ 用例图使用符号
- ❖ 用例建模步骤
- ❖ 教学管理系统用例建模



用例建模概述

- 用例建模是用于描述一个系统功能的建模技术
- 用例可用于获取新系统的客户需求
- 可以作为对已有系统进行升级时的指南
- 一个系统的用例模型由若干用例图组成
- 在用例模型中，功能以用例表示，每个用例指明一个完整的功能

用例建模概述

- 用例建模主要有两个产物
- 一是用例图
 - 它用来描述系统用例、参与者及其之间的关系
- 二是用例描述
 - 它是业务事件以及用户如何同系统交互以完成任务的文字描述。



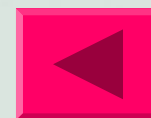
用例建模使用符号

用例图符号



用例建模步骤

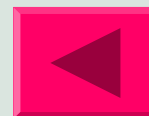
- ❖ 明确系统范围和边界
- ❖ 确定执行者和用例
- ❖ 对用例进行描述
- ❖ 定义用例之间的关系
- ❖ 分层绘制用例图
- ❖ 审核用例建模



用例建模步骤

(1) 明确系统的范围和边界

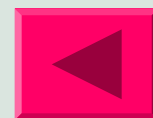
- 系统分析人员和客户要进行反复的交流，以便尽早确定系统的范围和边界
- 进而确定系统的责任、功能和性能
- 清楚地回答系统应该做什么，不应该做什么
- 本系统与哪些外部事物发生联系，发生哪些联系等



用例建模步骤

(2) 确定系统的执行者 (**Actor**) 和用例 (**Use case**)

- 执行者是指在系统外部与系统交互的人或其他系统，他以某种方式参与系统内用例的执行，即执行者执行用例
- 参与者可以是人、组织、信息系统、外部设备和时间概念
- 用例是对系统用户功能需求的描述，表达了系统的功能和提供的服务
- 一个用例表示系统中一个与特定执行者相关的完整的功能
- 用例通过关联与执行者连接，关联指出一个用例与哪些执行者交互



用例建模步骤

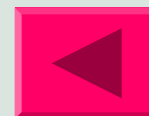
(3) 对用例进行描述

- 用例描述是一个系统（或子系统、类、接口）做什么，而不是描述怎么做
- 用清晰、用户容易理解的语言进行描述，是一份关于执行者与用例如何进行交互的简明、一致的规约
- 用例描述应包括
 - 用例的目的（功能）
 - 该用例在什么情况下哪个执行者启动执行
 - 用例与执行者之间交互哪些消息来通知对方做出决定
 - 交互的主消息流及因此被使用或修改的实体
 - 用例中可供选择的异常事件流
 - 用例结束标志

用例建模步骤

(4) 定义用例之间的关系

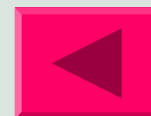
- 在用例模型中，用例之间也有关联
- 用例之间的关联主要有4种
 - 继承关联
 - 扩展关联
 - 包含关联
 - 使用关联



用例建模步骤

(5) 分层绘制用例图

- 对用例进行分解，绘出下一层次的用户图
- 用例图有一系列分层的用例图组成



用例建模步骤

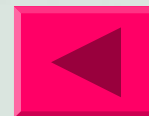
(6) 审核用例建模

- 对已经建立的用例图进行审核，确定系统的用例图
- 用例图在高层确定了系统必须处理的业务的范围
- 用例在系统生命周期的需求阶段进行定义，并在整个生命周期中不断进行细化
- 用例辅助确定对象或系统行为，设计界面和代码说明，并作为测试系统的计划



教学管理系统用例建模

- ❖ 整体用例建模
- ❖ 教务管理用例建模
- ❖ 排课管理用例建模



教学管理系统用例建模

● 整体用例建模

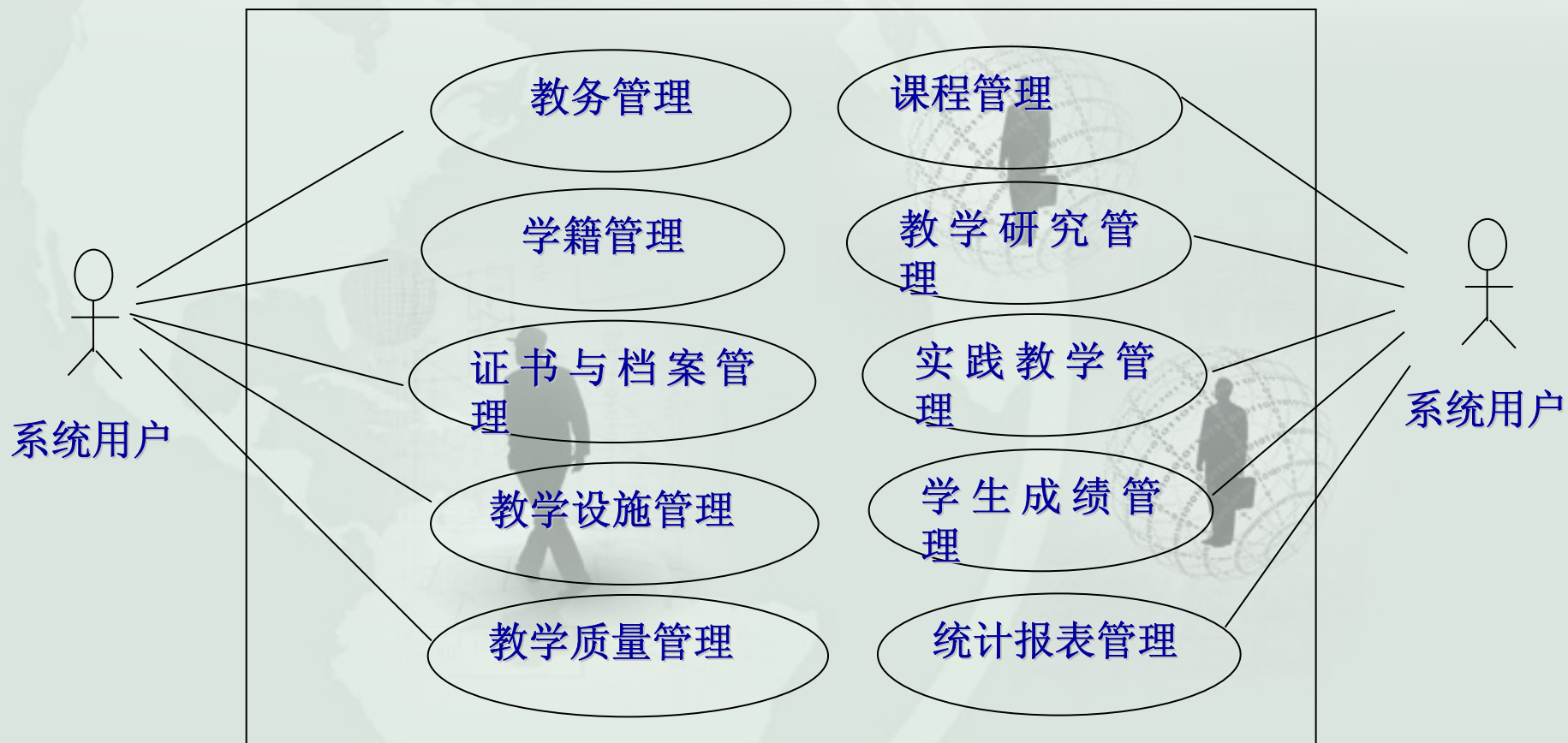
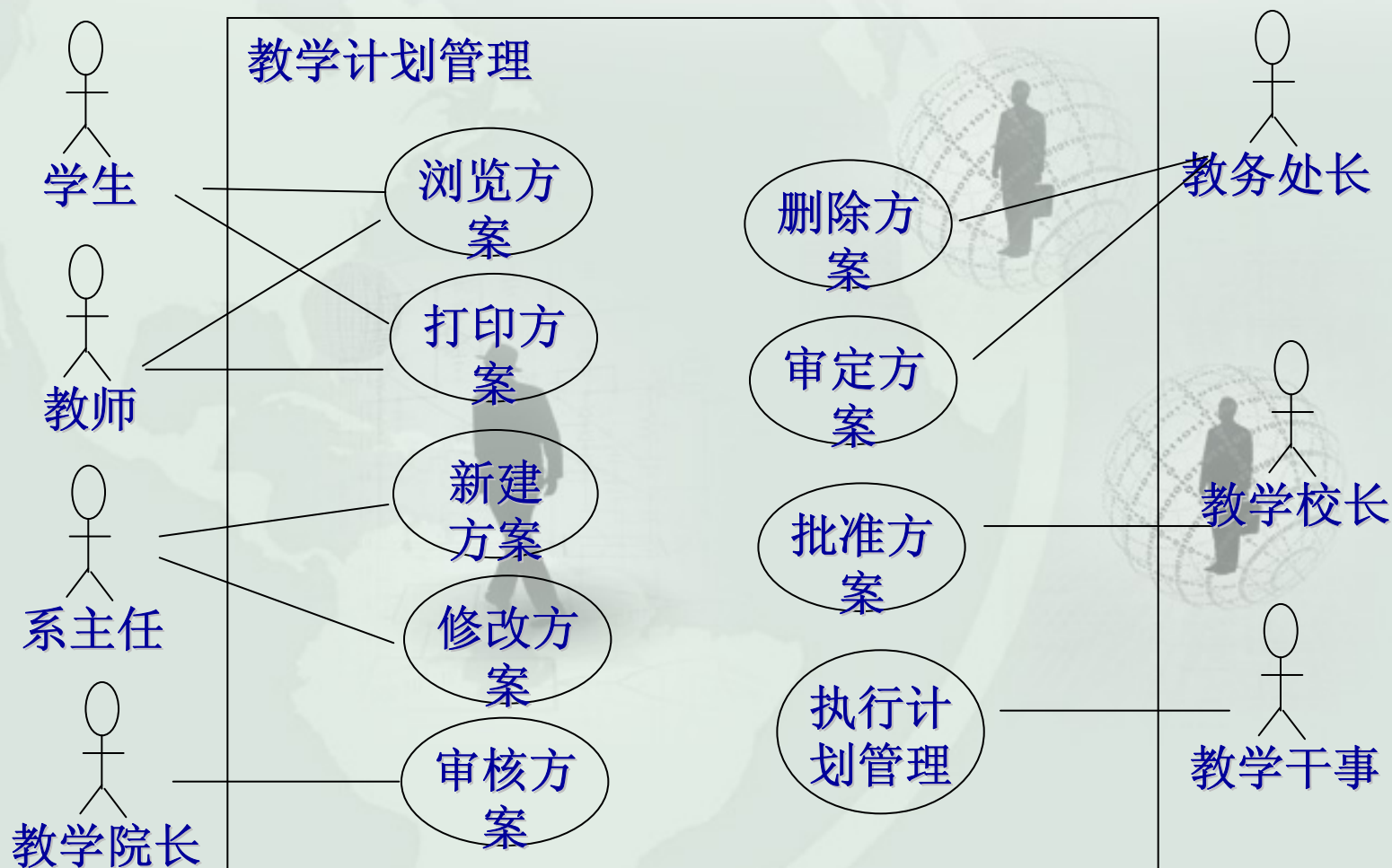


图8教学管理系统顶层用例图

教学管理系统用例建模

• 教务管理用例建模



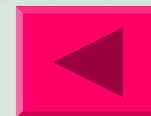
教学管理系统用例建模

• 排课管理用例建模



用例活动建模

- ❖ 用例活动建模概述
- ❖ 用例活动建模符号
- ❖ 用例活动建模步骤

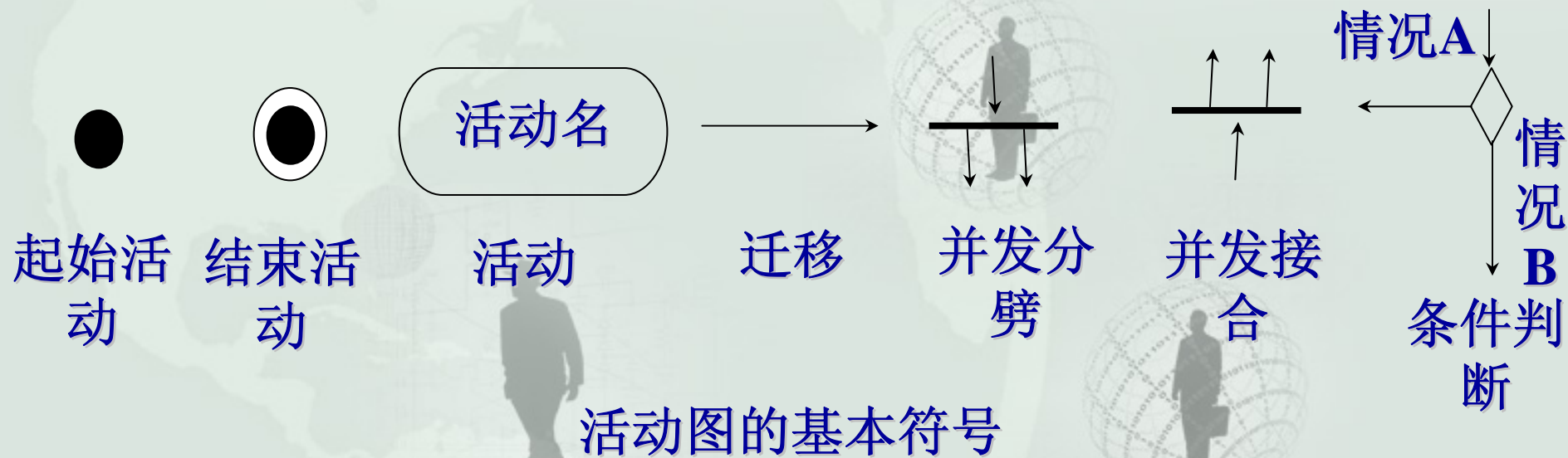


用例活动建模概述

- **UML**提供的活动图描述动作流和并发处理行为
- 它既可以用于系统分析阶段，也可以用于设计阶段
- 对于每个用例，至少可以构造一个活动图
- 如果用例很长、包含复杂的逻辑，则可以构造多个活动图
- 系统分析员可以使用活动图更好地理解用例步骤的流程和顺序



用例活动建模符号



用例活动建模步骤

(1) 标识需要活动图的用例

- 一个系统用例模型包含多幅用例图，每幅图又包含多个用例
- 一般情况下，不需要对每个用例绘制活动图，只有当实现该用例的步骤繁杂或者有特殊需要的时候才会画它
- 首先确定建模的内容，即要对哪个用例建立活动图
- 在教学管理系统中对课程管理用例进行案例分析，选课管理子系统需要画出活动图



用例活动建模步骤

(2) 建模每一个用例的主路径

- 在创建用例的活动图时，需要先确定该用例一条明确的执行工作流程，建立活动图的主路径
- 以该路径为主线进行补充、扩展和完善
- 在选课管理用例中，从管理员给出权限到选课成功组成了该活动图的基本执行轨迹



用例活动建模步骤

(3) 建模每一个用例的从路径

- 首先根据主路径分析其他可能出现的工作流的情况
- 这些可能是活动图中还没有建模的其他活动
- 可能是处理错误
- 或者是执行其他的活动
- 然后对不同进程中并发执行的活动进行处理。



用例活动建模步骤

(4) 添加泳道来标识活动的事务分区

- 泳道是将活动用线条分成一些纵向的矩形，这些矩形称为泳道
- 每个矩形属于一个特定的对象或部门（子系统）责任区
- 使用泳道可以把活动按照功能或所属对象的不同进行组织
- 属于同一个对象的活动都放在同一个泳道内，对象或子系统的名字放在泳道的顶部
- 在选课管理中列出学生和管理员两个对象
- 添加泳道的活动图是一个比较完整的活动图

用例活动建模步骤

(5) 改进高层的活动

- 对于一个复杂的系统，需要将描述系统不同部分的活动图按照结构层次关系进行排列
- 在一个活动图中，其中的一些活动可以分解为若干个子活动或动作，这些子活动或动作可以组成一个新的活动图
- 采用结构层次的表示方法，可以在高层只描述几个组合活动，其中每个组合活动的内部行为在展开的低一层活动图上进行描述，这些便于突出主要问题
- 如在学生选课前，管理员设定学生选课的学分上限
- 然后开放选课



用例活动建模步骤

(6) 进一步对细节进行完善

- 对前面的活动图进行补充和完善



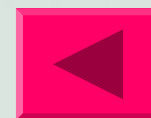
教学管理静态结构图

建立静态模型的步骤:

- 确立对象类
- 确定关联
- 给类和关联增添属性
- 利用继承合并和组织类

教学管理静态结构图

- ❖ 确定对象类
- ❖ 组合对象并确定关系
- ❖ 画出对象类图



确定对象类

❁ 找出候选的类

❁ 筛选出正确的对象类



确定对象类

- 可以采用**CRC**（类—责任—协作者）技术来分析，找出在当前问题域中的候选对象类。
- 可用非正式分析方法
 - 以用自然语言书写的需求陈述、用例文本表述或活动图描述为依据
 - 把陈述中的名词作为对象类的候选者
 - 用形容词作为确定属性的线索
 - 把动词作为服务（操作）候选者
 - 用这种简单方法确定的候选者是非常不准确的，必须经过更进一步的严格筛选。



确定对象类

筛选时依据下列标准，删除不正确或不必要的对象类：

- (1) 冗余
- (2) 无关
- (3) 笼统
- (4) 属性
- (5) 操作
- (6) 实现



组合对象并确定其关系

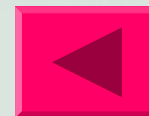
- 确定了系统的业务对象，就要组织这些对象，并记录对象之间的主要关系
- 类图主要用来表示对象及其关联关系，在此图中包括重数、关联关系、泛化关系及聚合关系
- 两个或多个对象之间的相互依赖、相互作用的关系就是关联
- 分析确定关联，可使分析员考虑问题域的边缘情况，有助于发现那些尚未被发现的对象类。

组合对象并确定其关系

❁ 初步确定关联

❁ 确定泛化关系

❁ 确定聚合关系



组合对象并确定其关系

初步确定关联

- 在需求陈述中使用的描述性动词或动词词组，通常表示关联关系
- 在初步确定关联时，大多数关联可以通过直接提取需求陈述中的动词词组而得出
- 通过分析需求陈述，还能发现一些在陈述中隐含的关联
- 分析员还应该与用户及领域专家讨论问题域实体间的相互依赖、相互作用关系，根据领域知识再进一步补充一些关联

组合对象并确定其关系

确定泛化关系

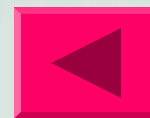
- 可采用自顶向下方法把现有类细化成更具体的子类，这模拟了人类的演绎思维过程
- 从应用域中能明显看出应该做的自顶向下的具体化工作
- 带有形容词修饰的名词词组往往暗示了一些具体类
- 在分析阶段应该避免过度细化
- 可以自底向上，归纳具体类的关系，形成父类，这模拟了人类的归纳思维过程



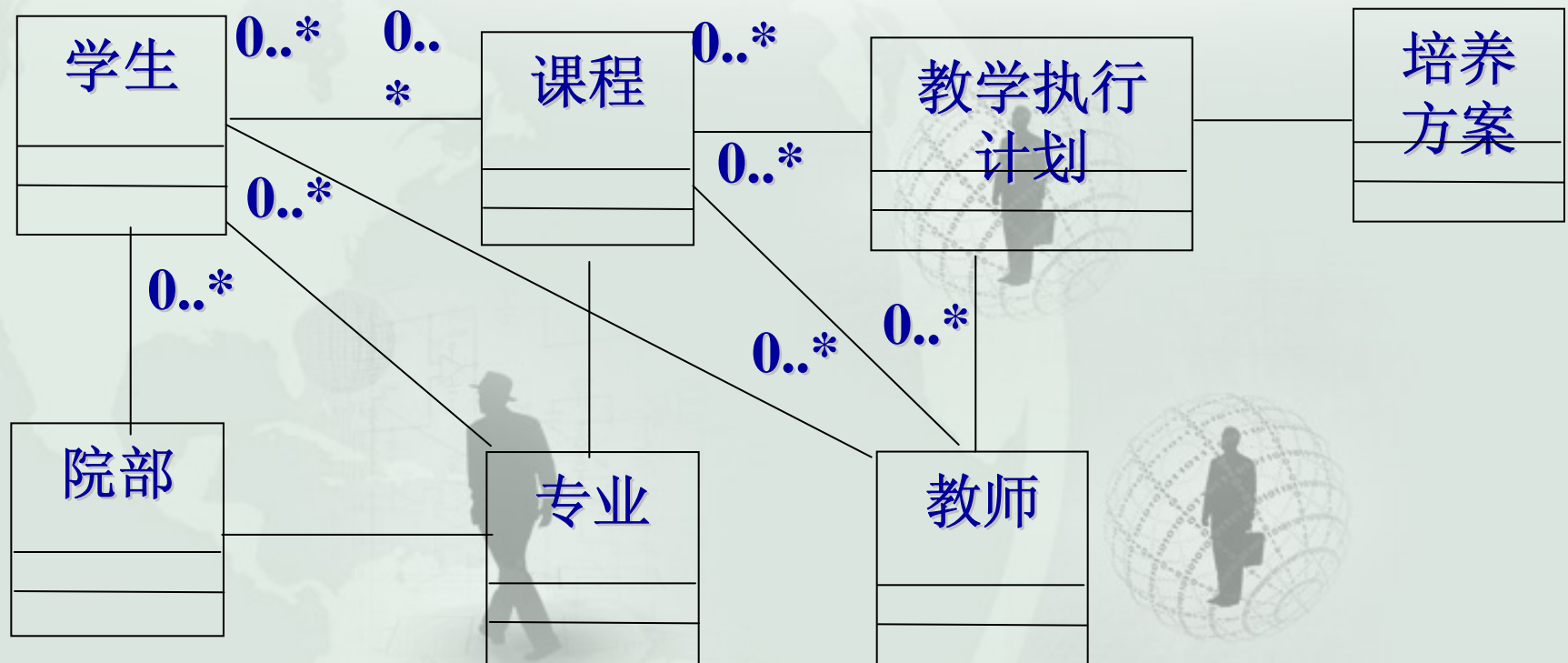
组合对象并确定其关系

确定聚合关系

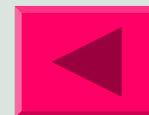
- 聚合是一类关联关系，其中一个对象是另一个对象的一部分
- 它经常被称为整体/部分关系
- 聚合关系是不对称的



画出对象类图



教学管理子系统部分类图



面向对象开发方法

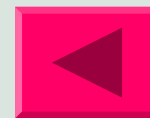
第三节

使用UML进行面向对象设计与动态建模



使用UML进行面向对象设计与动态建模

- ❖ 面向对象设计的准则
- ❖ 面向对象设计的内容
- ❖ 系统动态模型的设计
- ❖ 系统体系结构设计



面向对象设计准则

(1) 模块化

- 面向对象软件开发模式，很自然地支持了把系统分解成模块的设计原理，对象就是模块
- 它是把数据结构和操作这些数据的方法紧密地结合在一起所构成的模块

(2) 抽象

- 抽象表示对规格说明的抽象 (**Abstraction by Specification**) 和参数化抽象 (**Abstraction by Parametrization**)

(3) 信息隐藏

- 在面向对象方法中，信息隐藏通过对象的封装性实现
- 类结构分离了接口与实现，从而支持了信息隐藏
- 对于类的用户来说，属性的表示方法和操作的实现算法都应该是隐藏的

面向对象设计准则

(4) 弱耦合

- 在面向对象方法中，对象是最基本的模块
- 耦合主要指不同对象之间相互关联的紧密程度
- 弱耦合是优秀设计的一个重要标准，因为这有助于使得系统中某一部分的变化对其他部分的影响降到最低程度
- 对象不可能是完全孤立的，当两个对象必须相互联系、相互依赖时，应该通过类的协议（即公共接口）实现耦合，而不应该依赖于类的具体实现细节

面向对象设计准则

(5) 强内聚

- 设计中使用的一个构件内的各个元素，对完成一个定义明确的目的所做出的贡献程度
- 在设计时应该力求做到高内聚

在面向对象设计中存在下述**3**种内聚：

1) 服务内聚

一个服务应该完成一个且仅完成一个功能。

2) 类内聚

一个类应该只有一个用途。

3) 一般—特殊内聚

设计出的一般—特殊结构，应该符合多数人的概念

面向对象设计准则

(6) 可重用

- 软件重用是提高软件开发生产率和目标系统质量的重要途径
- 重用也叫再用或复用
- 是指同一事物不作修改或稍加改动就多次重复使用。重用是从设计阶段开始的

重用有两方面的含义

- 一是尽量使用已有的类（包括开发环境提供的类库及以往开发类似系统时创建的类）
- 二是如果确实需要创建新类，则在设计这些新类的协议时，应该考虑将来的可重复使用性

面向对象设计准则

软件成分的重用可以进一步划分成以下**3**个级别：

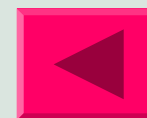
1) 代码重用。

2) 设计结果重用

- 设计结果重用指的是重用某个软件系统的设计模型（即求解域模型）
- 这个级别的重用有助于把一个应用系统移植到完全不同的软硬平台上

3) 分析结果重用

- 这是一种更高级别的重用，即重用某个系统的分析模型
- 这种重用特别适用于用户需求未改变，但系统体系结构发生根本变化的场合



面向对象设计内容

系统设计一般包括

系统对象设计

- 对象设计是为每个类的属性和操作做出详细的设计，并设计连接类及其关联类间的消息规约

系统体系结构设计

- 问题域子系统
- 人一机交互子系统
- 任务管理子系统
- 数据管理子系统设计

系统设计的优化及审查

面向对象设计内容

系统体系结构设计——问题域子系统

面向对象设计仅需从实现角度对问题域模型作一些补充或修改，主要是增添、合并或分解对象类、属性及服务，调整继承关系等

- (1) 调整需求
- (2) 重用已有的类
- (3) 把问题域类组合在一起
- (4) 增添一般化类以建立协议
- (5) 调整继承层次

面向对象设计内容

系统体系结构设计——人-机交互子系统设计

包括指定窗口和报表的形式、设计命令层次等内容

(1) 设计人一机交互界面的准则

(2) 设计人一机交互子系统的策略

(3) 设计命令层次

面向对象设计内容

系统体系结构设计——任务管理子系统设计

- 确定哪些是必须同时操作的对象
 - 哪些是相互排斥的对象
 - 进一步设计任务管理子系统
- (1) 分析并发性
- 通过检查各个对象的状态图及其交换的事件，能够把若干个非并发的对象归并到一条控制线中
 - 在计算机系统中用任务（**Task**）实现控制线

面向对象设计内容

系统体系结构设计——任务管理子系统设计

(2) 设计任务管理子系统

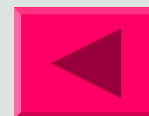
- 确定各类任务并把任务分配给适当的硬件或软件去执行
- 确定事件驱动型任务、时钟驱动任务、优先任务、关键任务并协调任务
- 确定资源需求，使用多处理器或固件，主要是为了满足高性能的需求
- 设计者必须通过计算系统载荷来估算所需要的**CPU**（或其他固件）的处理能力。

面向对象设计内容

系统体系结构设计——数据管理子系统设计

数据管理子系统是系统存储或检索对象的基本设施，它建立在某种数据存储管理系统之上，并且隔离了数据存储管理模式

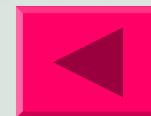
- (1) 选择数据存储管理模式
- (2) 设计数据管理子系统



系统动态模型的设计

❖ 对象交互模型

❖ 对象状态模型



对象交互模型设计

❁ 顺序图建模

❁ 协作图建模



对象交互模型设计

顺序图建模

顺序图（**Sequence Diagram**）也称为时序图

它描述系统中对象之间的交互行为，它强调消息的时间顺序，即对象间消息的发送和接收顺序

时序图还揭示了一个特定场景的交互，即系统执行期间发生在某个时间的对象之间的特定交互，它适合描述实时系统中的时间特性和时间约束

时序图中包括：类角色、生命线、激活期和消息

对象交互模型设计

顺序图建模

- 对象之间的通信可以同步进行，也可以异步进行
- UML中消息的类型

简单消息



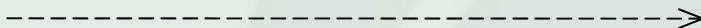
同步消息



异步消息



返回消息



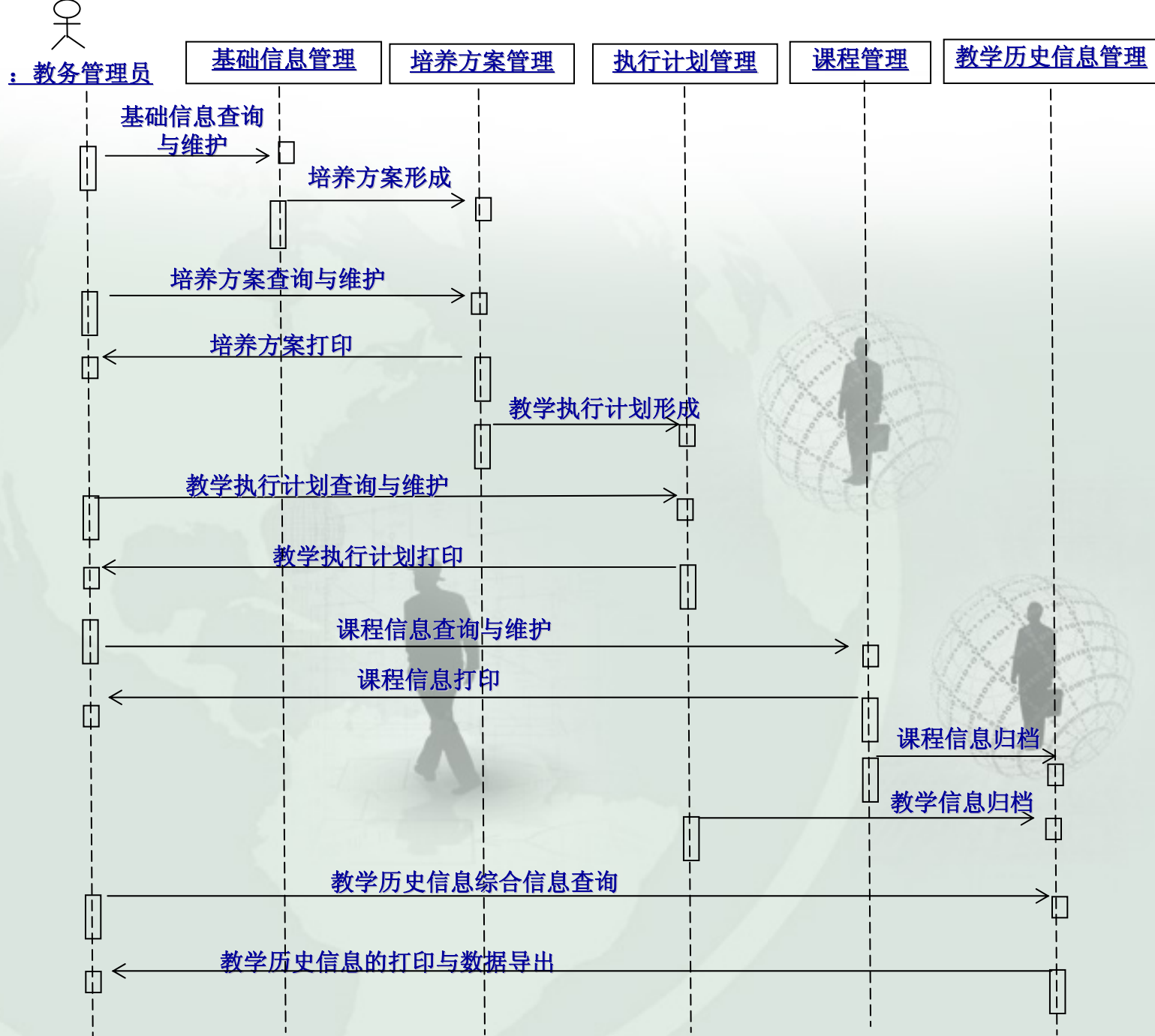
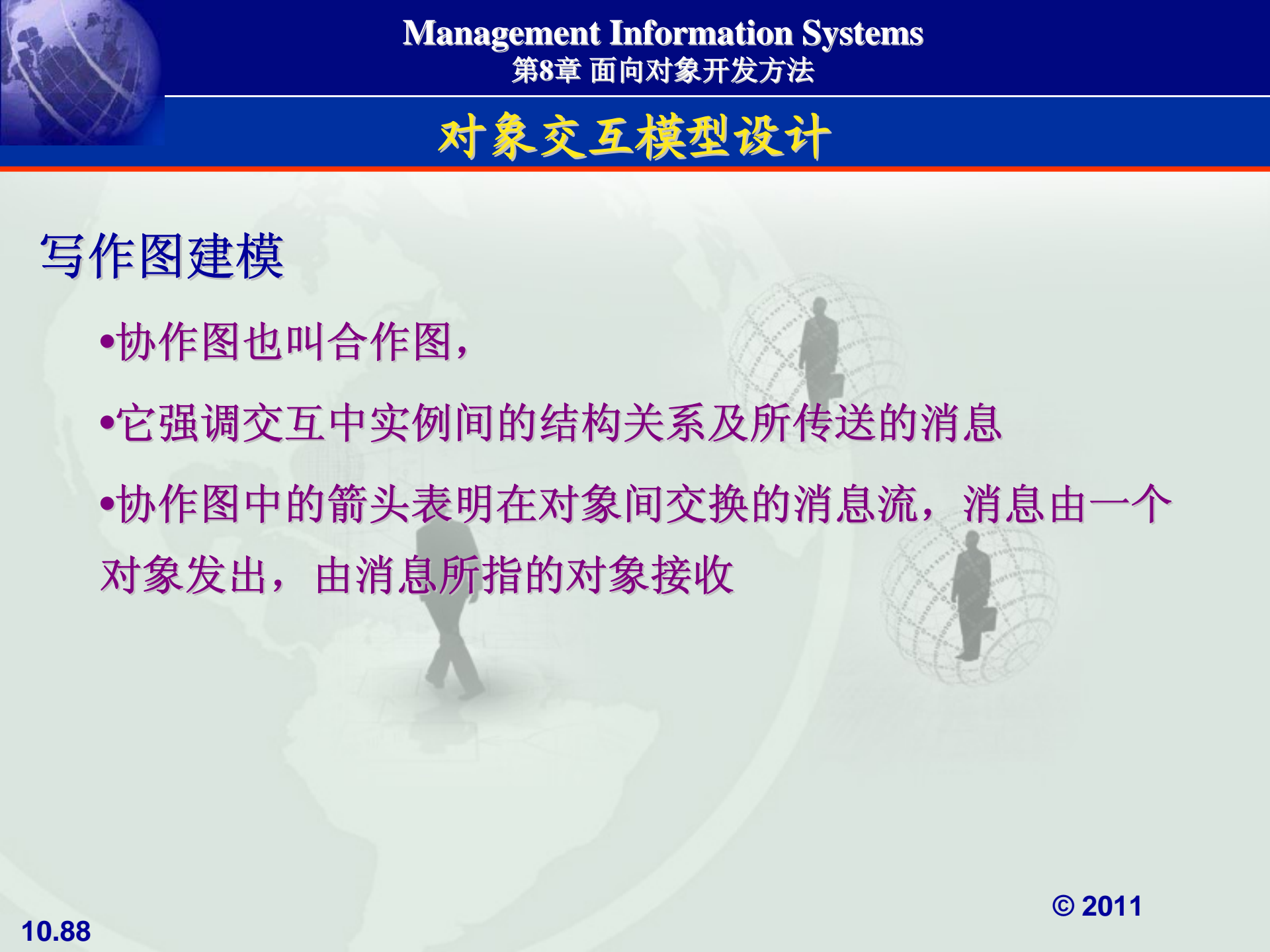


图8. 教务管理时序图



对象交互模型设计

写作图建模

- 协作图也叫合作图，
 - 它强调交互中实例间的结构关系及所传送的消息
 - 协作图中的箭头表明在对象间交换的消息流，消息由一个对象发出，由消息所指的对象接收
- 

对象交互模型设计

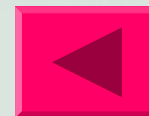


对象状态模型设计

❁ 状态图使用符号

❁ 状态图绘制步骤

❁ 状态图实例



对象状态模型设计

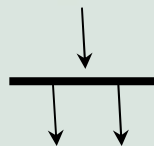
状态图符号



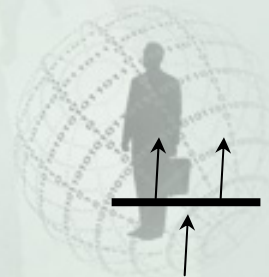
起始状态



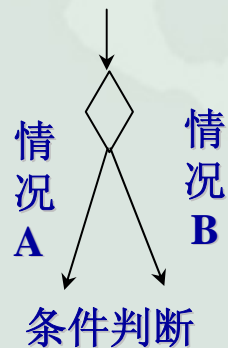
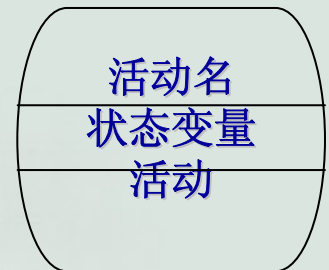
结束状态



并发分劈



并发接合



发送信号



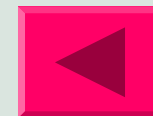
接收信号



状态

迁移

状态图使用的符号



对象交互模型设计

状态图绘制步骤

- 确定状态图描述的主体
 - 可以是一个整个系统、一个用例、一个类或一个对象
- 确定状态图描述的范围，明确起始状态和结束状态
- 确定描述主体在其生存期的各种稳定状态，包括高层状态和可能的子状态
- 确定状态的序号，对这些稳定状态按其出现顺序编写序号
- 取得触发状态迁移的事件，该事件可以触发状态进行迁

对象交互模型设计

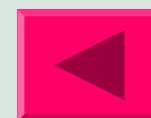
状态图绘制步骤

- 附上必要的动作，把动作附加到相应的迁移线上。
- 简化状态图。
- 确定状态图的可实现性。
- 确定有无死锁。
- 审核状态图



对象交互模型设计

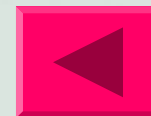
状态图绘制实例



系统体系架构设计

❖ 硬件系统体系架构

❖ 软件系统体系架构

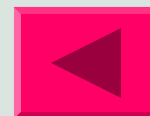


系统体系架构设计

硬件系统体系架构

硬件系统架构主要有两种

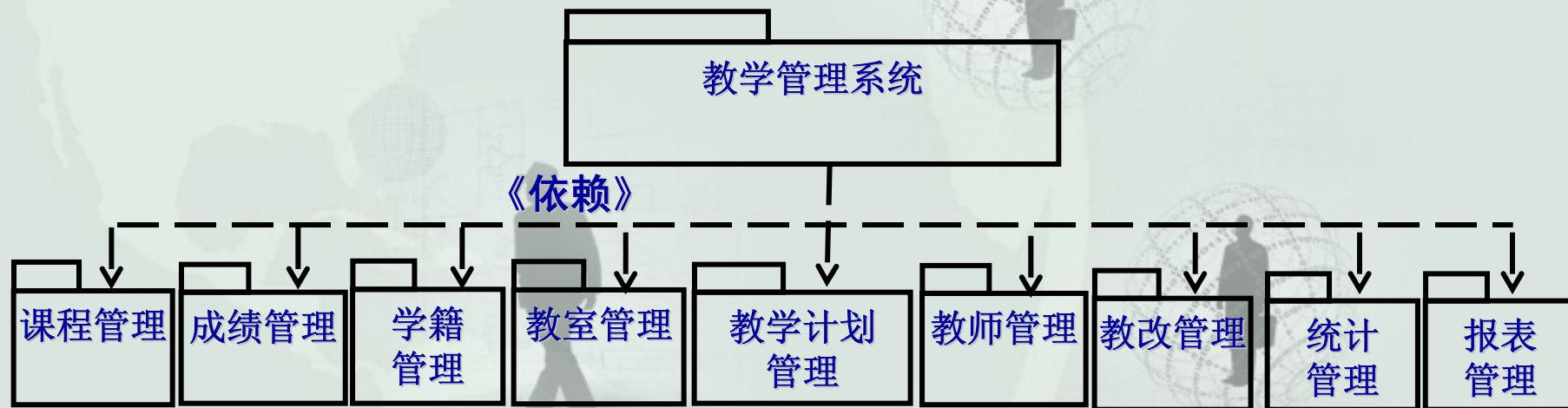
- 客户机/服务器模式(**Client/Server, C/S**)
- **Web**浏览器/服务器模式(**Browser/Server, B/S**)
- 高校教学管理信息系统架构采用**C/S**与**B/S**相结合的模式
- 硬件系统体系结构图可以用配置图表示



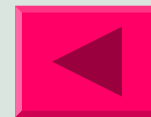
系统体系架构设计

软件系统体系架构

面向对象的软件系统体系结构一般采用包图进行描述



“教学管理系统”的包（子系统）层次结构



面向对象开发方法

第四节

使用UML进行面向对象开发实例



使用UML进行面向对象开发实例

❖ 系统分析与建模

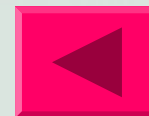
❖ 面向对象设计



系统分析与建模

❁ 用例建模

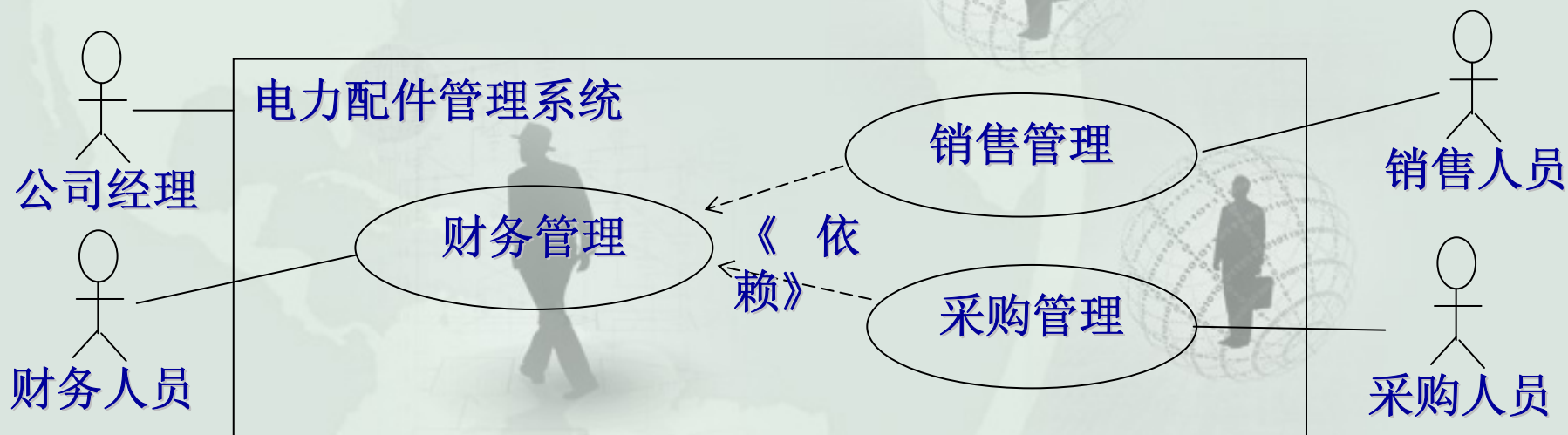
❁ 对象建模



系统分析与建模

用例建模

电力配件管理包括销售管理、采购管理和财务管理

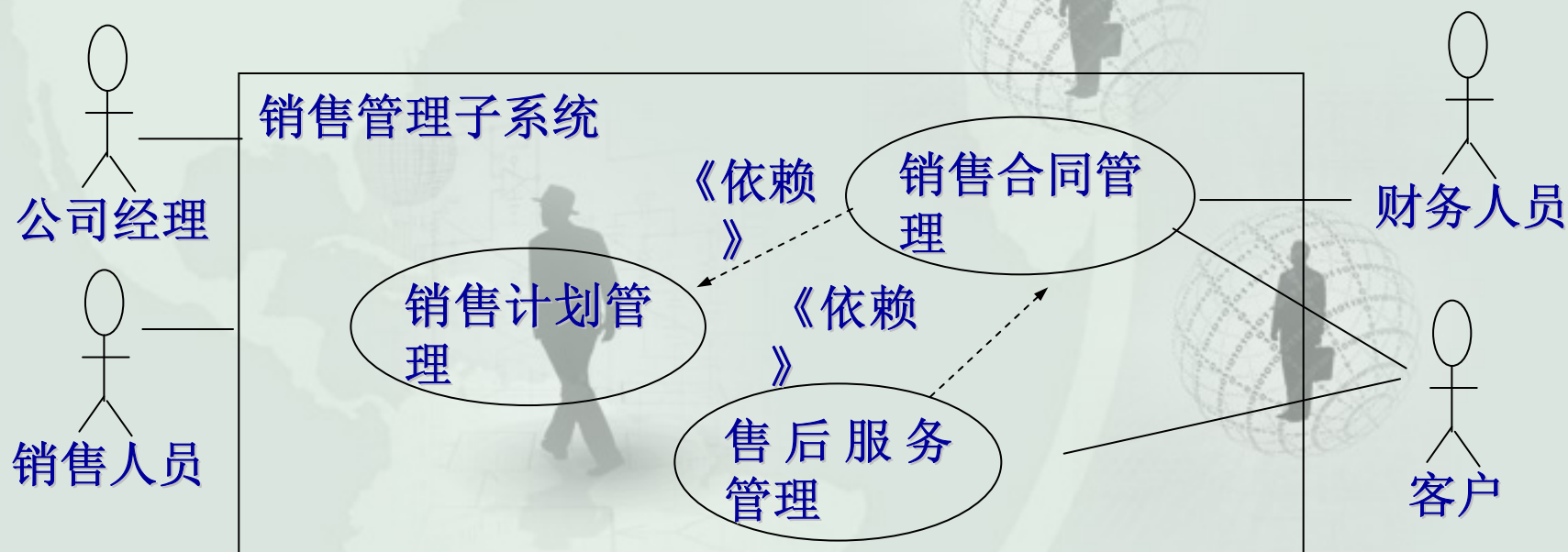


电力配件管理用例图

系统分析与建模

用例建模

销售管理子系统



销售管理子系统用例图

系统分析与建模

用例建模

销售管理子系统

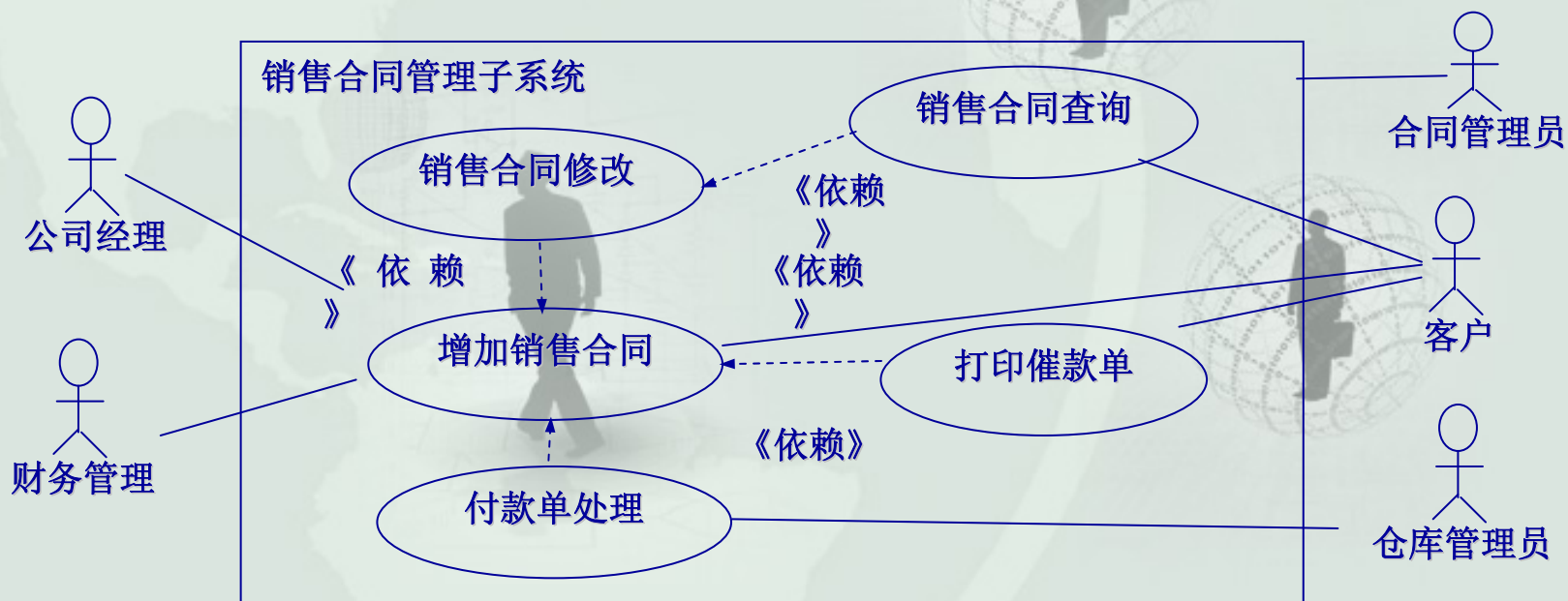


图8. 销售合同管理子系统用例图

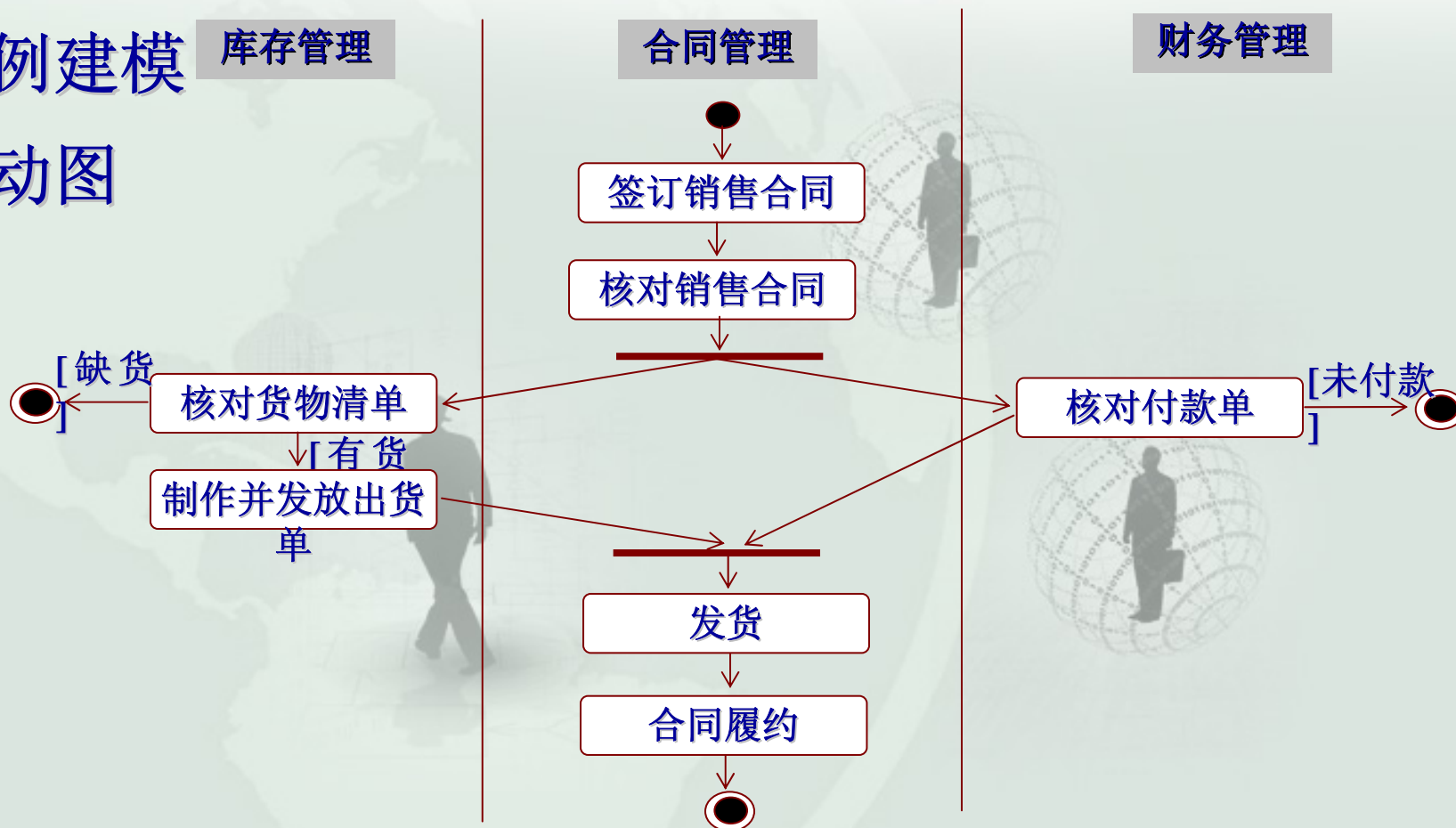
系统分析与建模

用例描述

- 用例编号：**00010101**。
- 用例名：增加销售合同。
- 执行者：直接执行者—合同管理员。涉及的执行者有合同管理员、客户、仓库管理员、财务管理系统、公司经理等。
- 描述：合同管理员将与客户签订的销售合同的详细内容录入管理系统中，用于对销售合同进行统计、查询等，便于监控正在执行的合同。
- 前置条件：已经签署了销售合同。
- 后置条件：对销售合同进行处理。
- 过程描述：合同管理员输入标识码，系统识别其有效性；初始化一个新的销售合同，输入一个新的销售合同编号；将与客户签订的销售合同的详细内容录入到系统中；退出系统。
- 与其他用例的关联：过程描述中的身份验证用例；编号自动生成用例。
- 异常事件处理：标识码有效性检查失败；身份验证失败；编号也可以由合同管理员手工录入，系统进行唯一性检验。出现错误，允许重新输入。

系统分析与建模

用例建模 活动图



销售合同从签订到履约的活动图

系统分析与建模

对象模型

- (1) 找出系统的实体类
- (2) 找出系统的边界类
- (3) 找出系统控制类
- (4) 找出销售管理子系统的属性和操作

面向对象设计

❁ 系统体系结构设计

❁ 系统详细设计



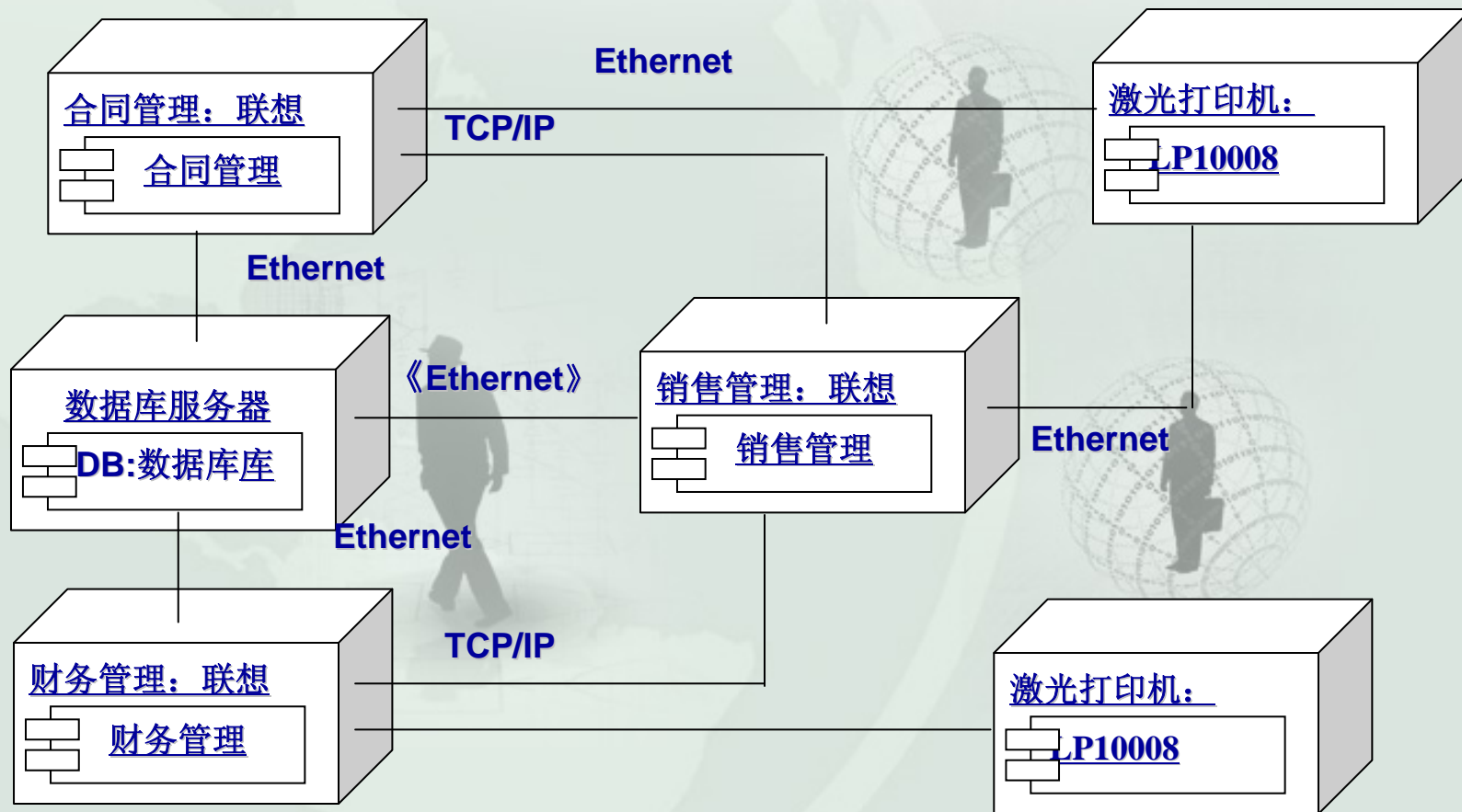
面向对象设计

系统体系结构设计——硬件系统体系结构建模

- (1) 确定结点。确定该系统使用的硬件设备：计算机终端、服务器和打印机等。根据软件体系结构的功能要求：需要一个网络数据库服务器、客户机和打印机
- (2) 描述结点属性。该系统各结点计算机的性能指标
- (3) 确定各结点驻留的构件。作为客户机的结点分别有“销售管理”构件、“采购管理”构件和“财务管理”构件，网络数据库服务器结点驻留数据库对象。
- (4) 确定各节点之间的联系

面向对象设计

系统体系结构设计——硬件系统体系结构建模



“销售管理子系统”配置图

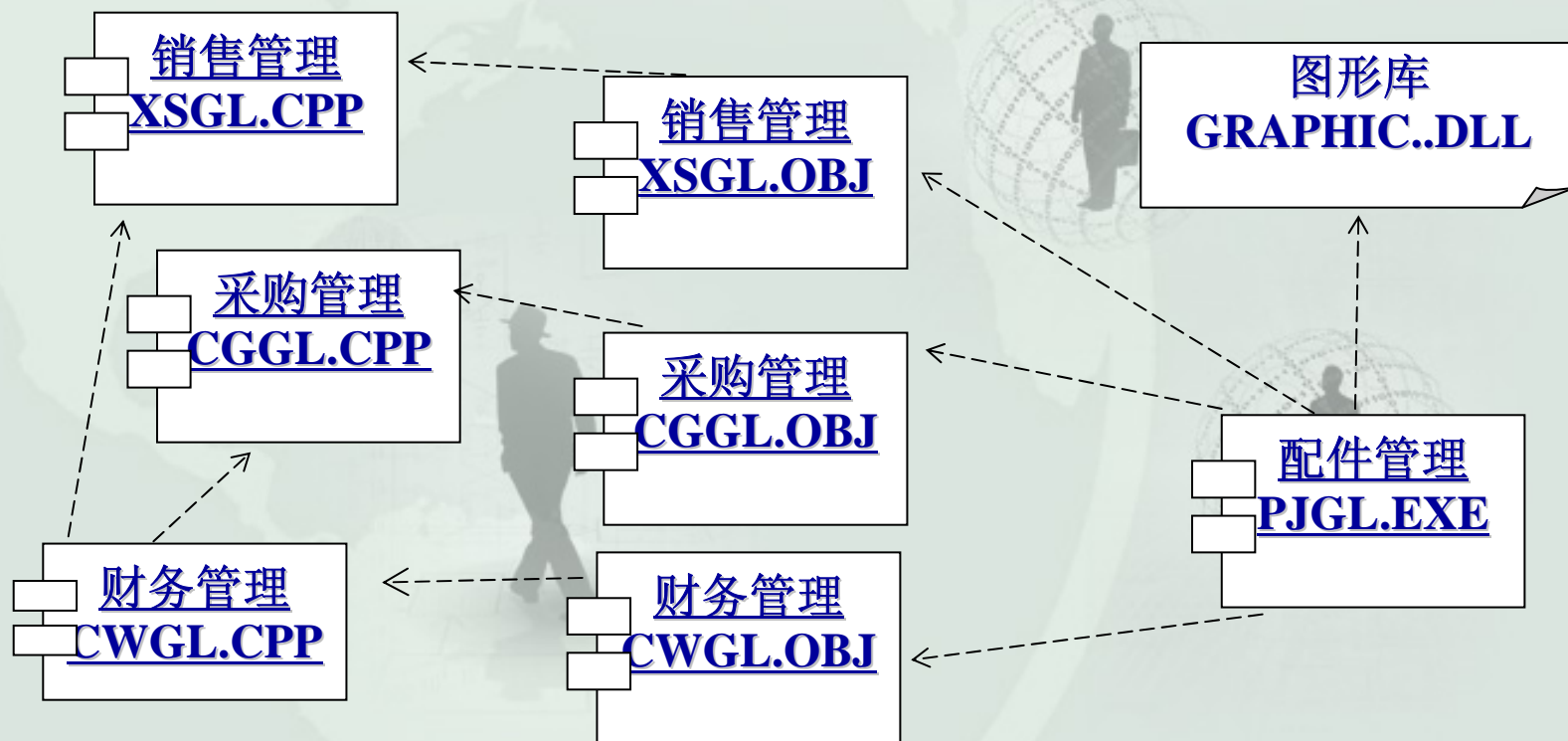
面向对象设计

系统体系结构设计——软件系统体系结构建模

- 在**UML**中包可以由构件组成
- 构件图描述构件及其相互依赖关系，由构件、接口及构件之间的关系构成
- 构件是逻辑体系结构（类、对象及其关系和协作）中定义的概念和功能在物理体系结构中的实现，它通常是开发环境中的实现性文件
- 一个由**C++**语言编写的配件管理系统，其源代码形成可执行代码的过程构件图

面向对象设计

系统体系结构设计——软件系统体系结构建模



源代码形成可执行代码过程的构件图

系统详细设计

系统的详细设计即对象设计

它对所有类的属性和操作都进行详尽地描述，并设计连接类及其相关关联间的消息规约

主要设计有

- 对象接口设计
- 设计算法和数据结构
- 确认子系统
- 子系统间的通信规约
- 给编写代码的程序员一个清晰的规范说明

小 结

- 系统的静态模型由类图、对象图、包图、构件图和配置图组成
- 系统的动态模型由对象交互模型和对象的状态模型组成，用来描述系统的动态行为，显示对象在系统运行期间不同时刻的动态交互
- 在**UML**中对象交互模型由顺序图和合作图进行描述，对象的状态模型由状态图和活动图进行描述
- 在面向对象方法中，对象之间通过发送消息来相互通信，消息分为简单消息、同步消息、异步消息和返回消息

