统一建模语言UML

内容

- UML是什么
- UML有什么用
- UML的历史
- UML语言体系
- 建模与UML
- UML工具

内容

- **UML是什么**
- UML有什么用处
- UML的历史
- UML语言体系
- 建模与UML
- UML工具

UML是什么

- **U**nified **M**odeling **L**anguage
- UML是一种绘制软件蓝图的标准语言。
- 可用UML对软件密集型系统的制品进行可视化、详述、构造和文档化。

UML是什么

- UML是一种可视化的建模语言，它能够让系统构造者用标准的、易于理解的方式建立起能够表达其设计思想的系统模型，便于不同的人之间有效的共享和交流设计成果。
- UML不是编程语言，UML是建模语言，UML用于对所要构建的软件系统进行建模，编程语言一般用于实现所建立的系统模型。

内容

- UML是什么
- **UML有什么用处**
- UML的历史
- UML语言体系
- 建模与UML
- UML工具

UML有什么用处

- UML统一了各种不同的软件分析和设计的建模语言，成为系统建模语言的事实上的标准
- 可视化建模，一图胜千言
- UML表述内容贯穿软件开发的生命周期
- UML可从不同角度描述系统
- 帮助我们认识和解决问题
- 有助于人员之间的沟通
- 更早地发现错误
- 获取设计结果
- 为最后的代码生成提供依据

内容

- UML是什么
- UML有什么用处
- **UML的历史**
- UML语言体系
- 建模与UML
- UML工具

UML的历史

- 面向对象建模语言产生于1970年代中期
- 1990年代初，建模语言数量达到50多种
  - 众多语言各有千秋，缺乏通用性
  - 语言之间的细小差别妨碍了用户的交流
- 1990年代中期，三位主要的OO建模大师：Booch,Rumbaugh,Jacobson开始致力于建立统一建模语言的工作

三位面向对象大师

Grady Booch

- Rational公司首席科学家，Booch方法发明人
- 提出了面向对象软件工程的概念，将Ada的工作扩展到整个面向对象领域
- Booch1993比较适合于系统的设计和构造

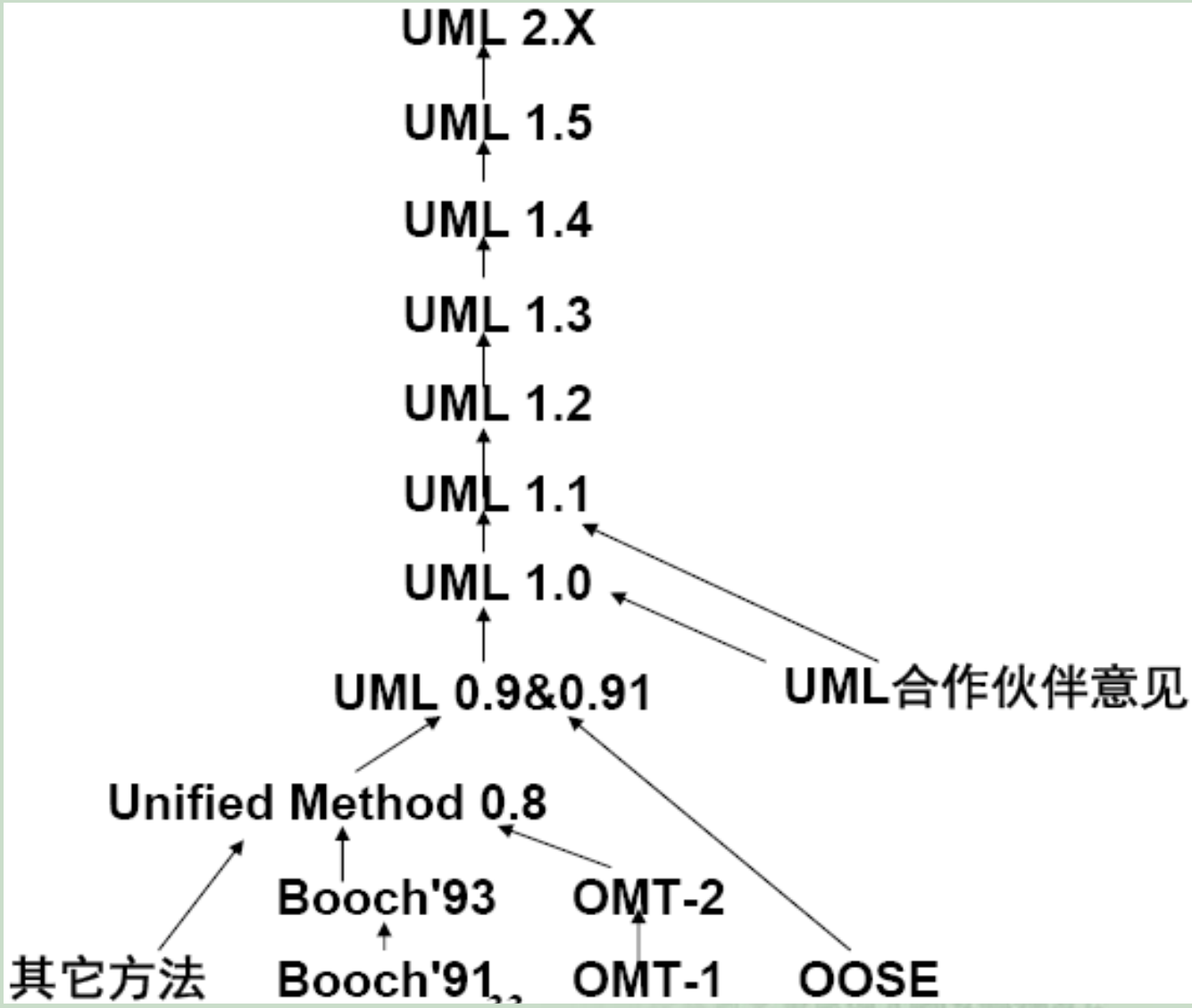
Jim Rumbaugh

- Rational公司高级研究员
- 提出了面向对象建模技术(OMT)，该方法用对象模型、动态模型、功能模型和用例模型来支持软件开发的全过程
- OMT-2特别适用于分析和描述以数据为中心的信息系统

Ivar Jacobson

- Rational公司副总裁，提出了面向对象软件工程（OOSE）方法
- OOSE的最大特点是面向用例的，用例(use-case)贯穿了整个开发过程
- OOSE比较适合于商业工程和需求分析

UML的发展历程


UML 2.X
UML 1.5
UML 1.4
UML 1.3
UML 1.2
UML 1.1
UML 1.0
UML 0.9&0.91
UML合作伙伴意见
Unified Method 0.8
Booch'93
OMT-2
其它方法
Booch'91
OMT-1
OOSE

UML

UML的设计目标：

- 运用面向对象概念来构造系统模型
- 建立起从概念模型直至可执行体之间明显的对应关系
- 着眼于那些有重大影响的问题
- 创建一种对人和机器都适用的建模语言

UML

UML的主要特点：

- 统一的标准, 已经被OMG接受为标准建模语言
- 支持面向对象开发
- 可视化建模, 表达能力很强大
- 独立于开发过程, 可以适用于不同软件过程
- 独立于编程语言
- 概念明确, 表示简洁, 结构清晰

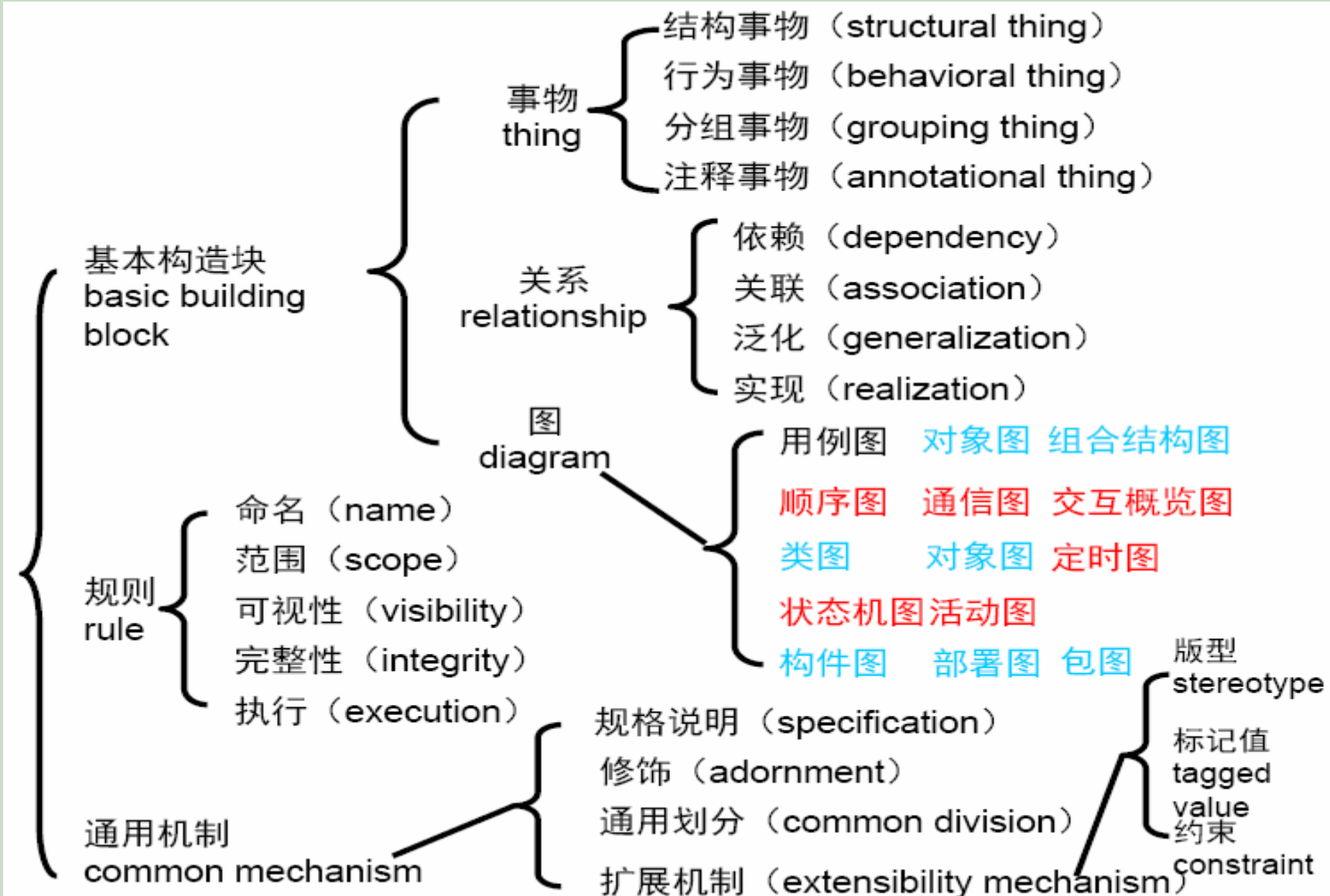
内容

- UML是什么
- UML有什么用处
- UML的历史
- **UML语言体系**
- 建模与UML
- UML工具

UML语言体系

- UML的构成
- UML中的图
- UML的视图

UML的构成


基本构造块
basic building block
事物
thing
结构事物（structural thing）
行为事物（behavioral thing）
分组事物（grouping thing）
注释事物（annotational thing）
关系
relationship
依赖（dependency）
关联（association）
泛化（generalization）
实现（realization）
图
diagram
用例图 对象图 组合结构图
顺序图 通信图 交互概览图
类图 对象图 定时图
状态机图 活动图
构件图 部署图 包图
规则
rule
命名（name）
范围（scope）
可视性（visibility）
完整性（integrity）
执行（execution）
通用机制
common mechanism
规格说明（specification）
修饰（adornment）
通用划分（common division）
扩展机制（extensibility mechanism）
版型
stereotype
标记值
tagged value
约束
constraint

UML中的图

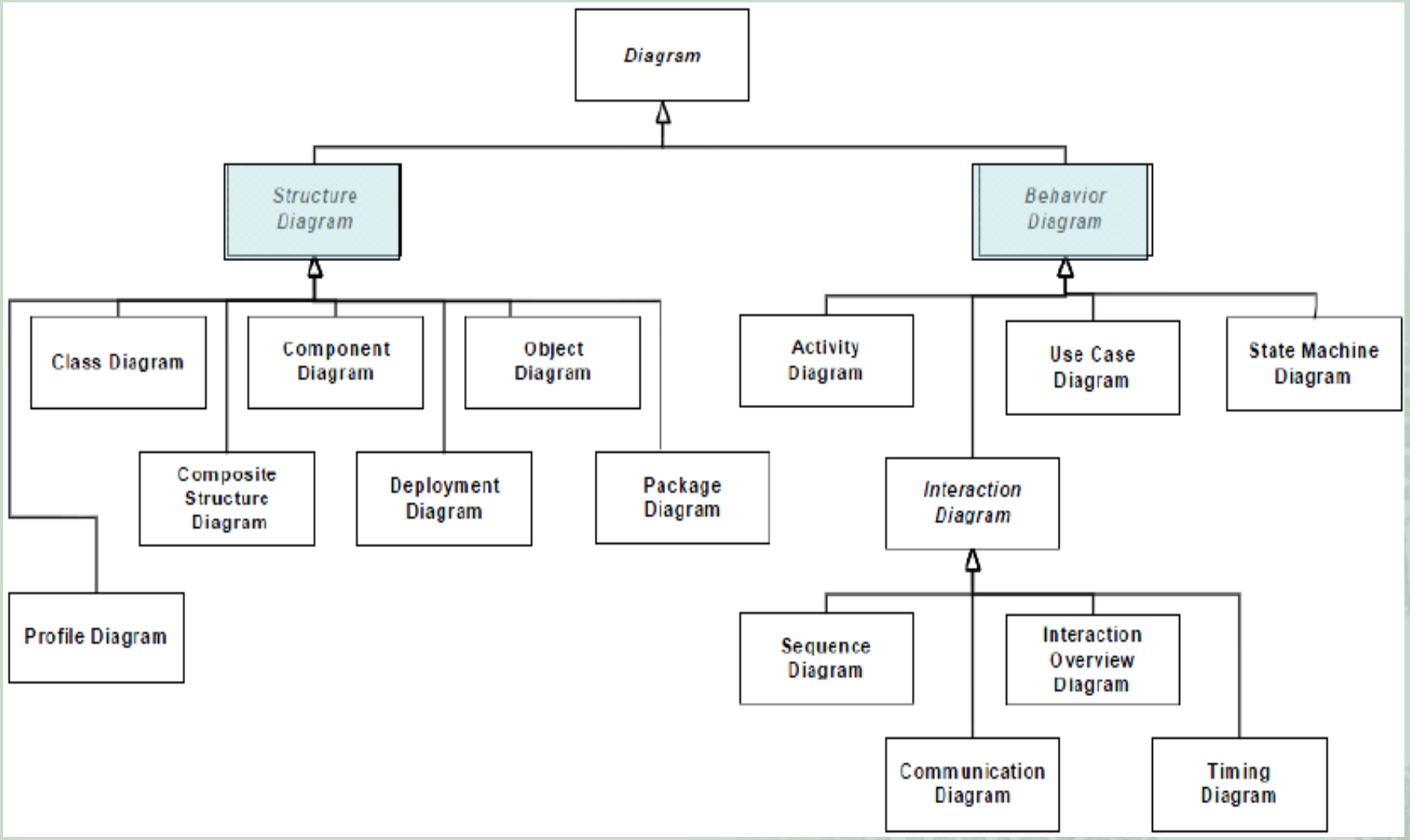


Diagram
Structure Diagram
Behavior Diagram
Class Diagram
Component Diagram
Object Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Profile Diagram
Activity Diagram
Use Case Diagram
State Machine Diagram
Interaction Diagram
Sequence Diagram
Interaction Overview Diagram
Communication Diagram
Timing Diagram

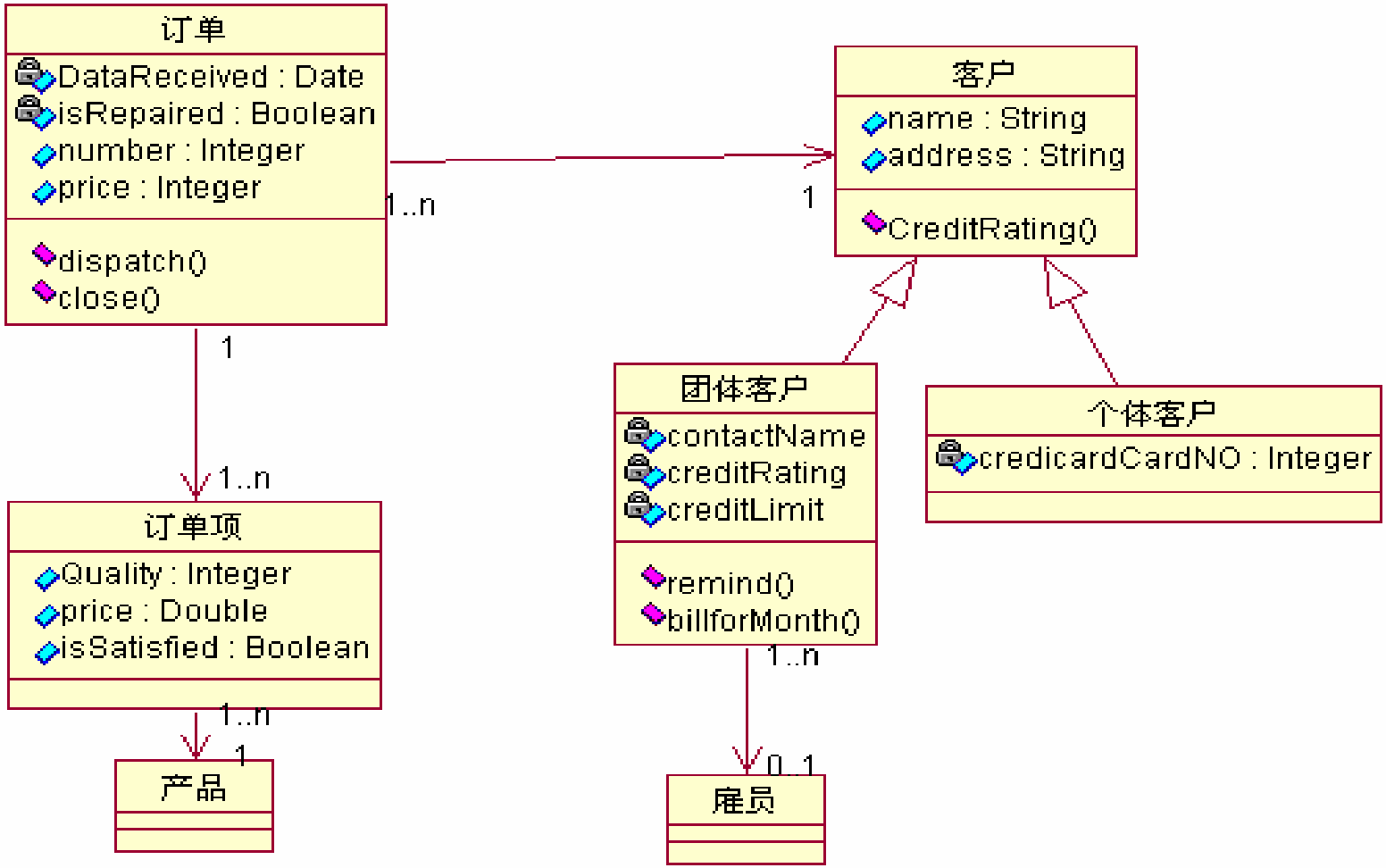
结构图（Structure Diagram）

- 类图（Class Diagram）
- 构件图（Component Diagram）
- 对象图（Object Diagram）
- 部署图（Deployment Diagram）
- 组合结构图（Composite Structure Diagram）
- 包图（Package Diagram）
- Profile Diagram

类图（Class Diagram）

- 描述类和类之间的关系。
- 可以说是UML中的核心，是使用UML建模时最常用的图。
- 在软件开发的不同阶段使用的类图具有不同的抽象层次。

类图示例

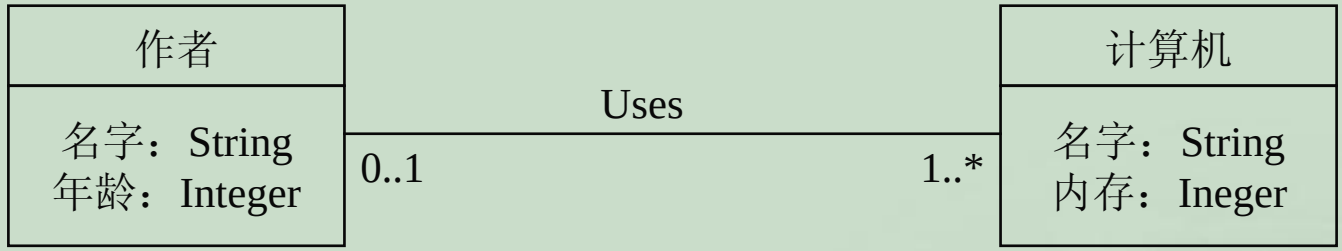

订单
DataReceived : Date
isRepaired : Boolean
number : Integer
price : Integer
dispatch()
close()
客户
name : String
address : String
CreditRating()
1..n
1
1
1..n
订单项
Quality : Integer
price : Double
isSatisfied : Boolean
1..n
1
产品
团体客户
contactName
creditRating
creditLimit
remind()
billforMonth()
1..n
0..1
雇员
个体客户
credicardCardNO : Integer

对象图（Object Diagram）

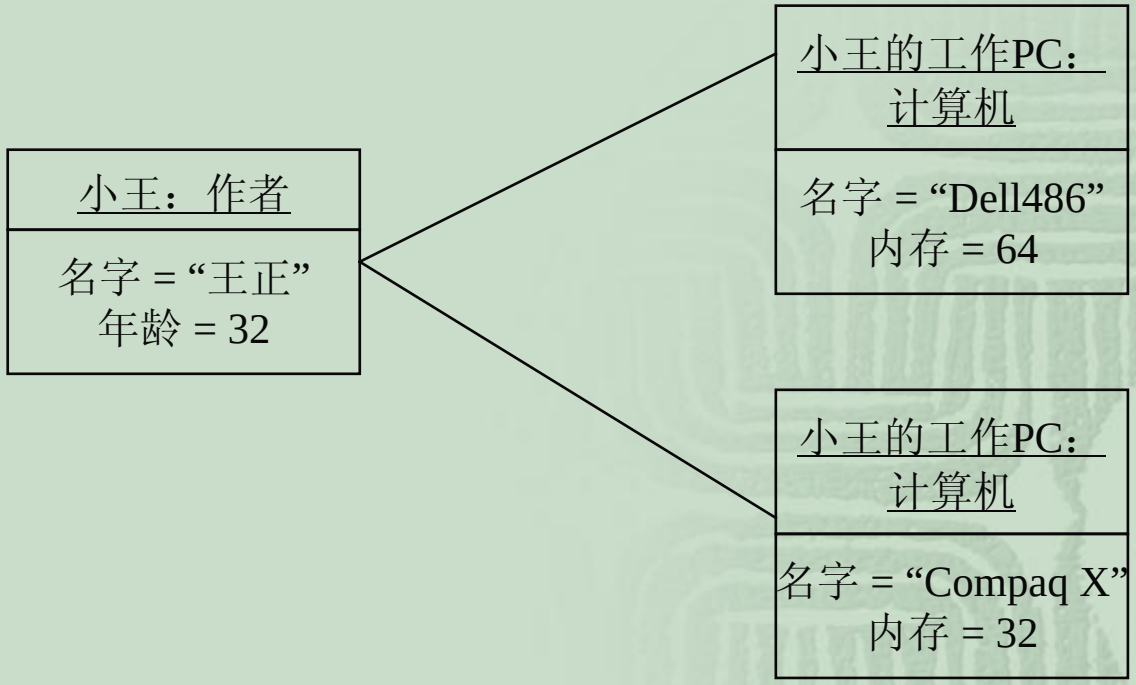
- 对象图是类图的一种实例化。
- 一张对象图表示的是与其对应的类图的一个具体实例，即系统在某一时期或者某一特定时刻可能存在的具体对象实例以及它们相互之间的具体关系。
- 对象图是系统在某一时刻的快照

对象图示例

类图


作者
名字：String
年龄：Integer
Uses
0..1
1..*
计算机
名字：String
内存：Ineger

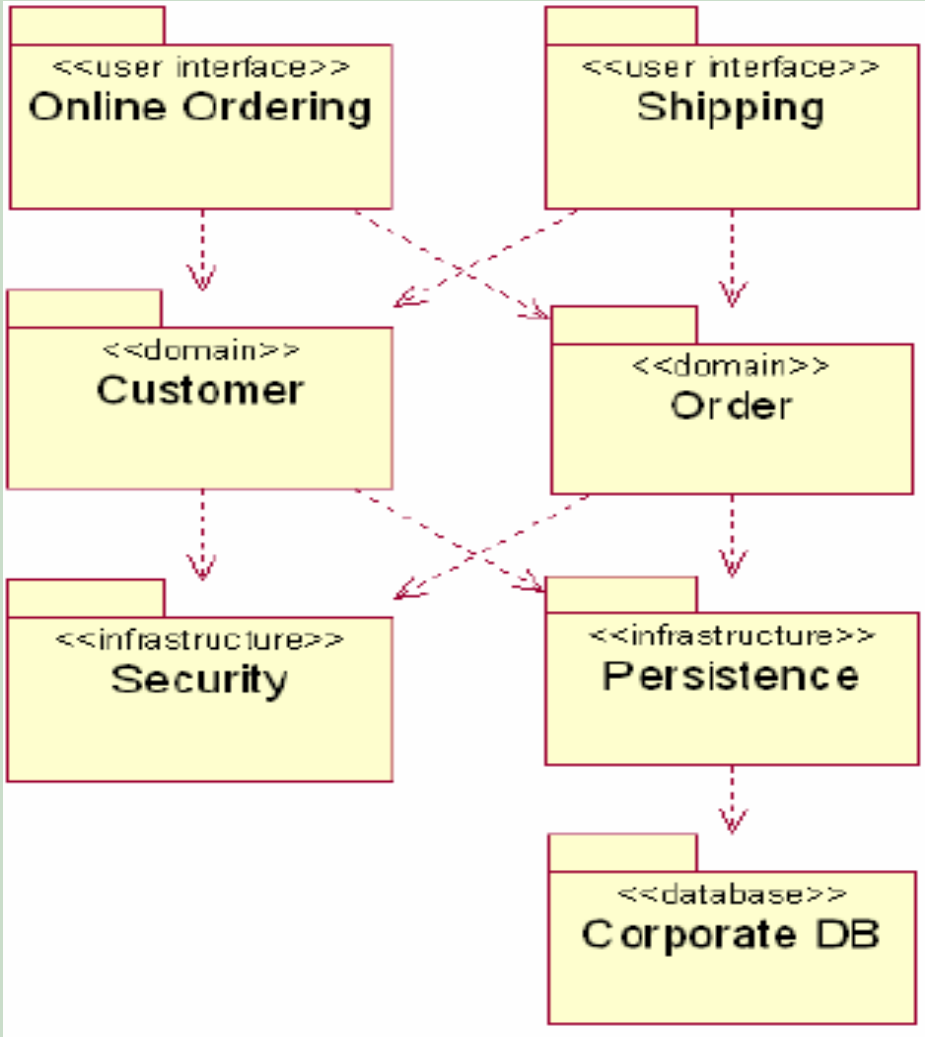
对象图


小王：作者
名字 = “王正”
年龄 = 32
小王的工作PC：
计算机
名字 = “Dell486”
内存 = 64
小王的工作PC：
计算机
名字 = “Compaq X”
内存 = 32

包图（Package Diagram）

- 包是类的集合。
- 包图所显示的是类的包以及这些包之间的依赖关系。
- 如果两个包中的任意两个类之间存在依赖关系，则这两个包之间存在依赖关系。
- 包的依赖是不传递的。

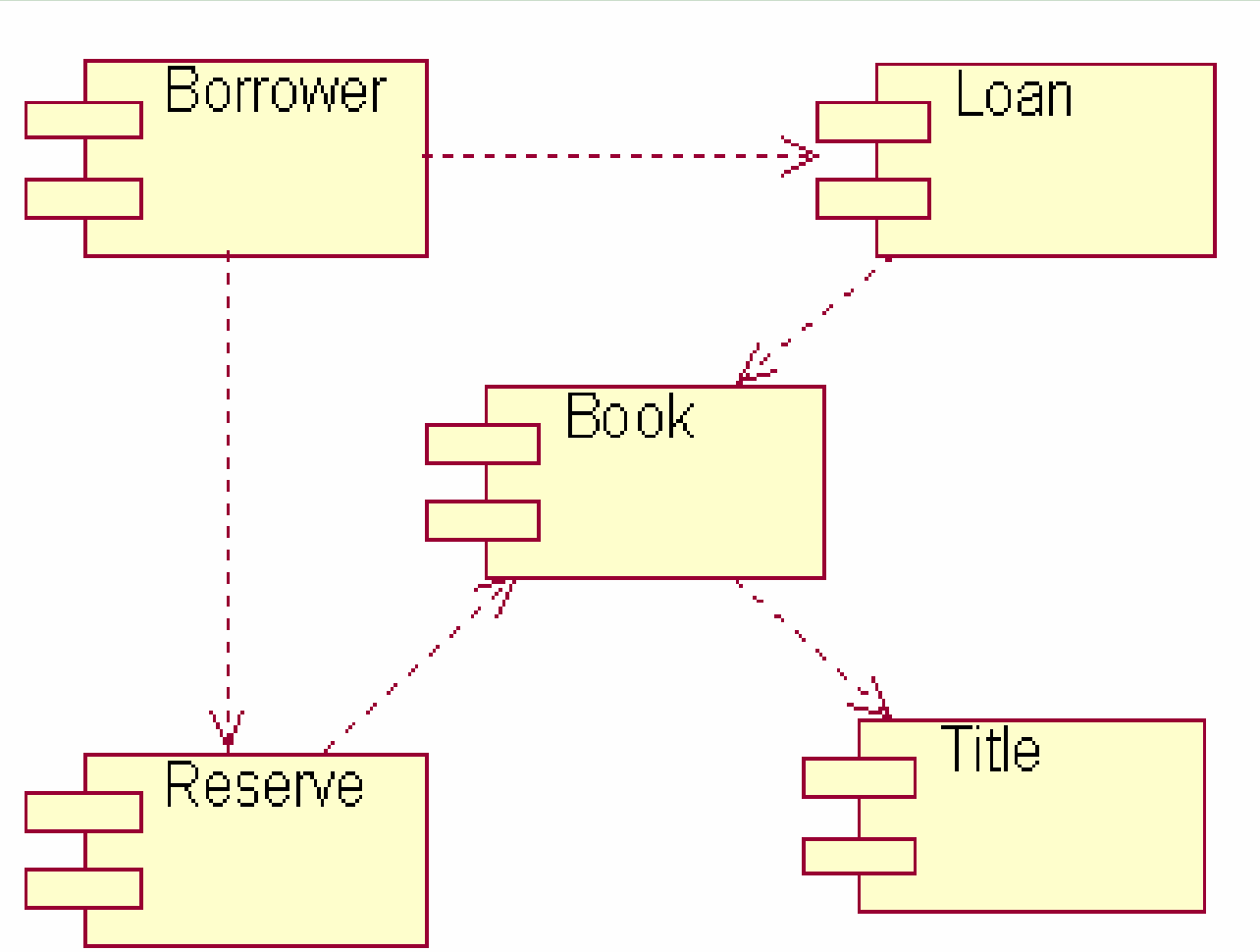
包图示例


<<user interface>>
Online Ordering
<<user interface>>
Shipping
<<domain>>
Customer
<<domain>>
Order
<<infrastructure>>
Security
<<infrastructure>>
Persistence
<<database>>
Corporate DB

构件图（Component Diagram）

- 构件图描述软件构件以及它们之间的依赖关系。
- 构件图便于人们分析和发现当修改某个构件时可能对哪些构件产生影响，以便对它们做相应的修改或更新。
- 构件可以是源代码构件、二进制目标码构件、可执行构件或文档构件。

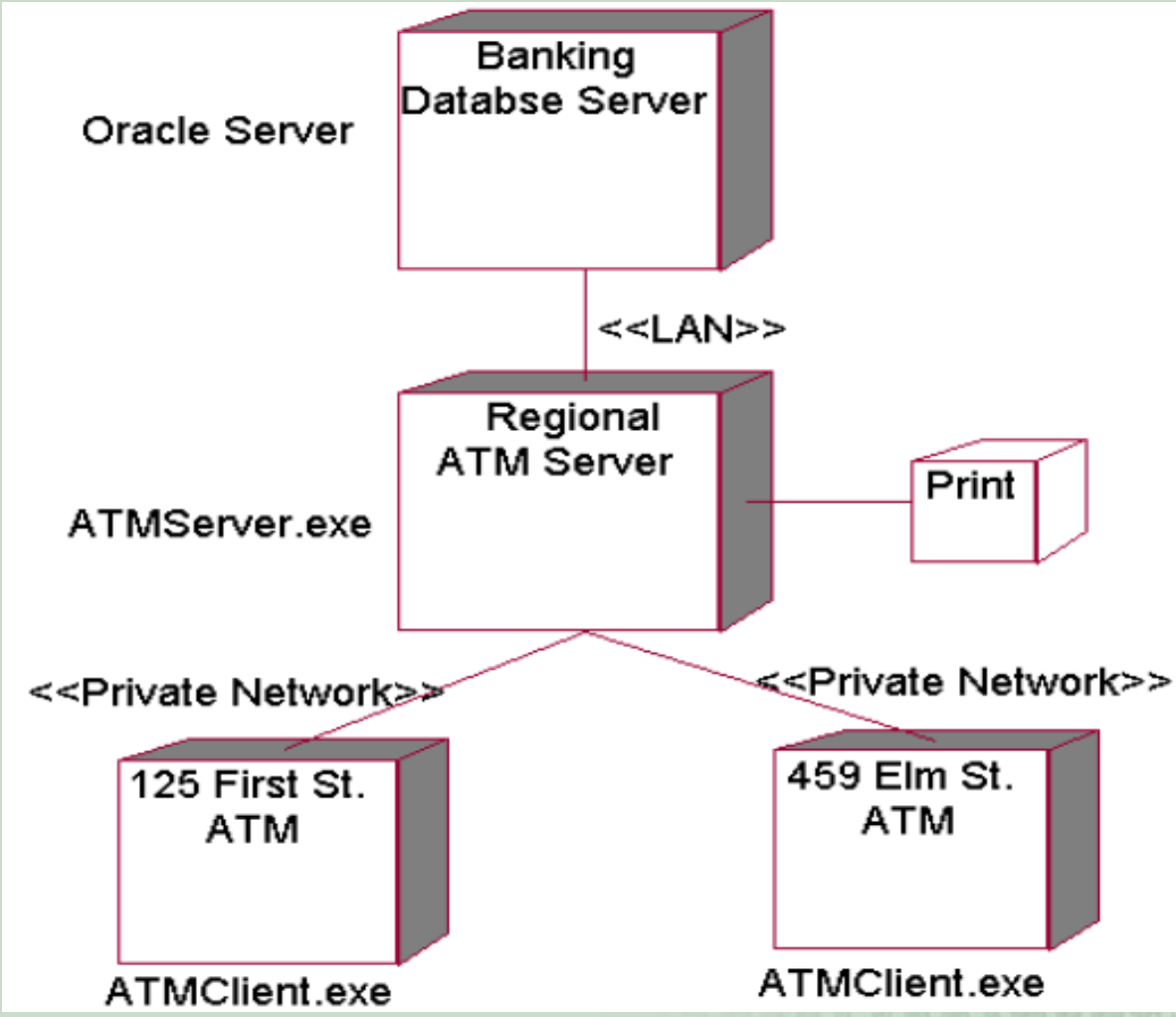
构件图示例



部署图（Deployment Diagram）

- 部署图描述系统中硬件和软件的物理部署情况和系统体系结构。
- 在部署图中，用结点表示实际的物理设备，并根据它们之间的连接关系，将相应的结点连接起来，并说明其连接方式。在结点里面，说明分配给该结点上运行的可执行构件或对象，从而说明哪些软件单元被分配在哪些结点上运行。

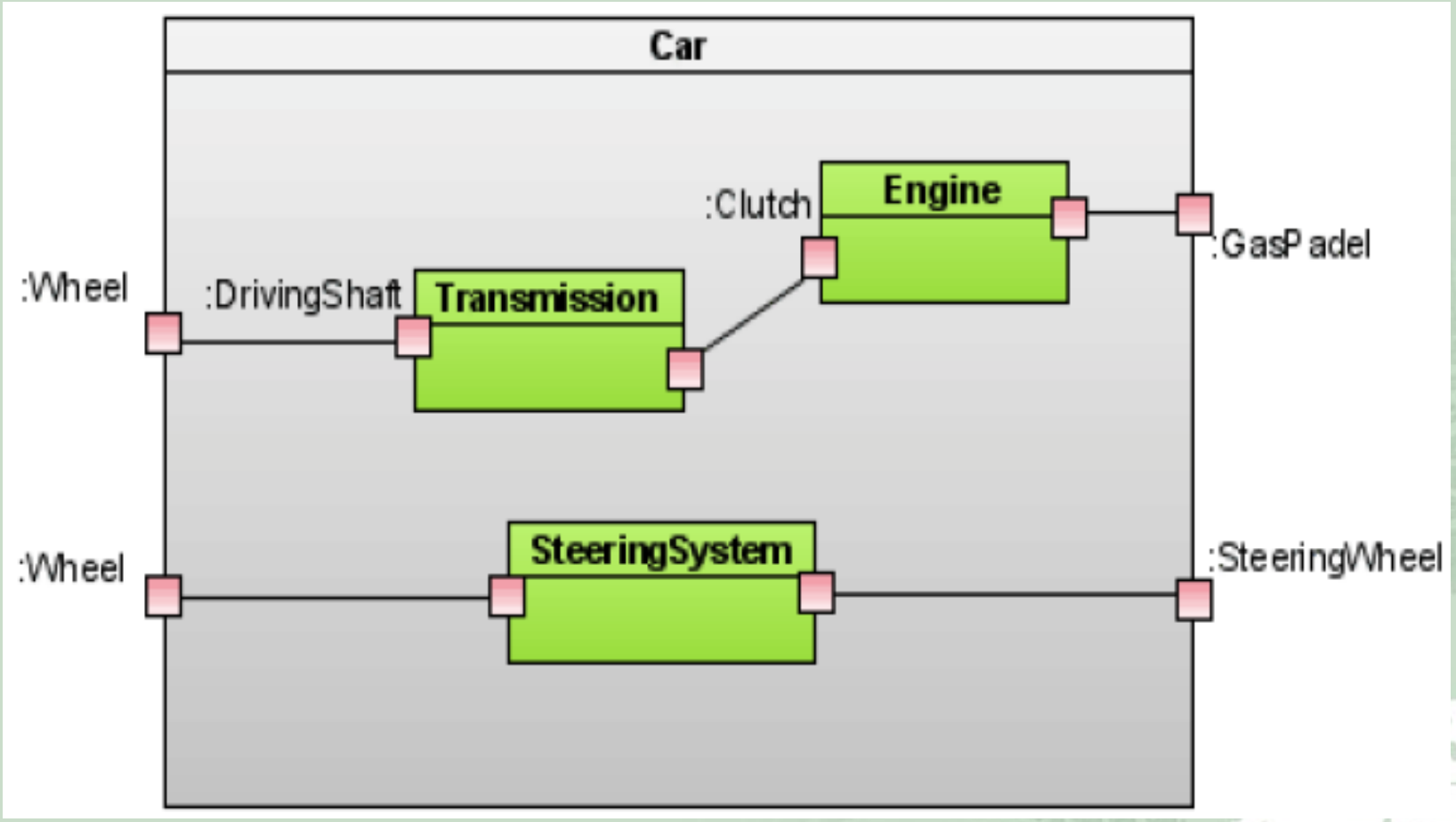
部署图示例


Banking
Databse Server
Oracle Server
<<LAN>>
Regional
ATM Server
ATMServer.exe
Print
<<Private Network>>
<<Private Network>>
125 First St.
ATM
459 Elm St.
ATM
ATMClient.exe
ATMClient.exe

组合结构图
（Composite Structure Diagram）

- 表示一个模型元素的内部构造及协作关系。
- 既能表示封装结构，也使图形更直观。

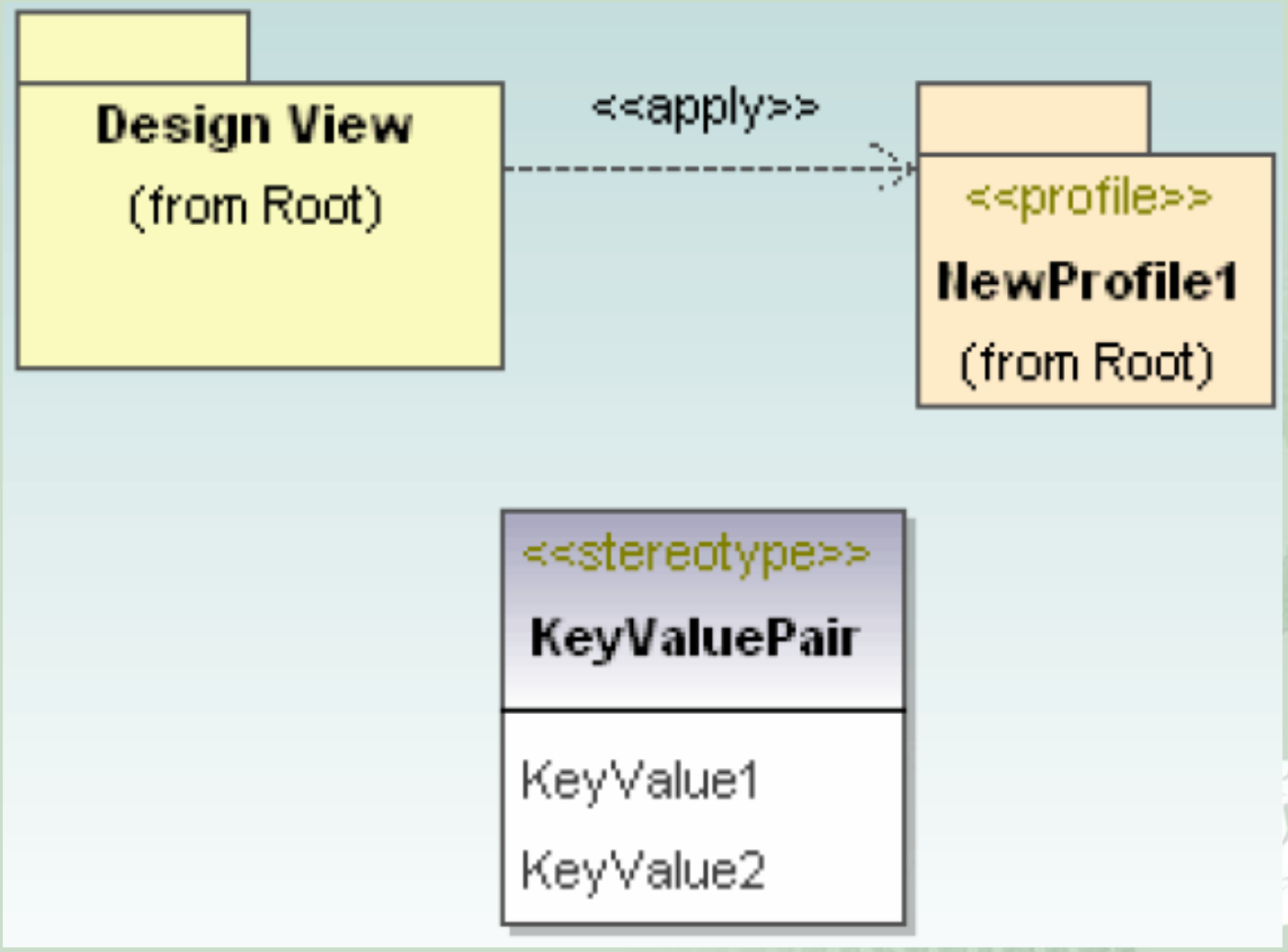
组合结构图示例


Car
:Clutch
Engine
:GasPadel
:Wheel
:DrivingShaft
Transmission
:Wheel
SteeringSystem
:SteeringWheel

Profile Diagram

- 描述Profile与其使用者之间的关系。

Profile Diagram示例



行为图（Behavior Diagram）

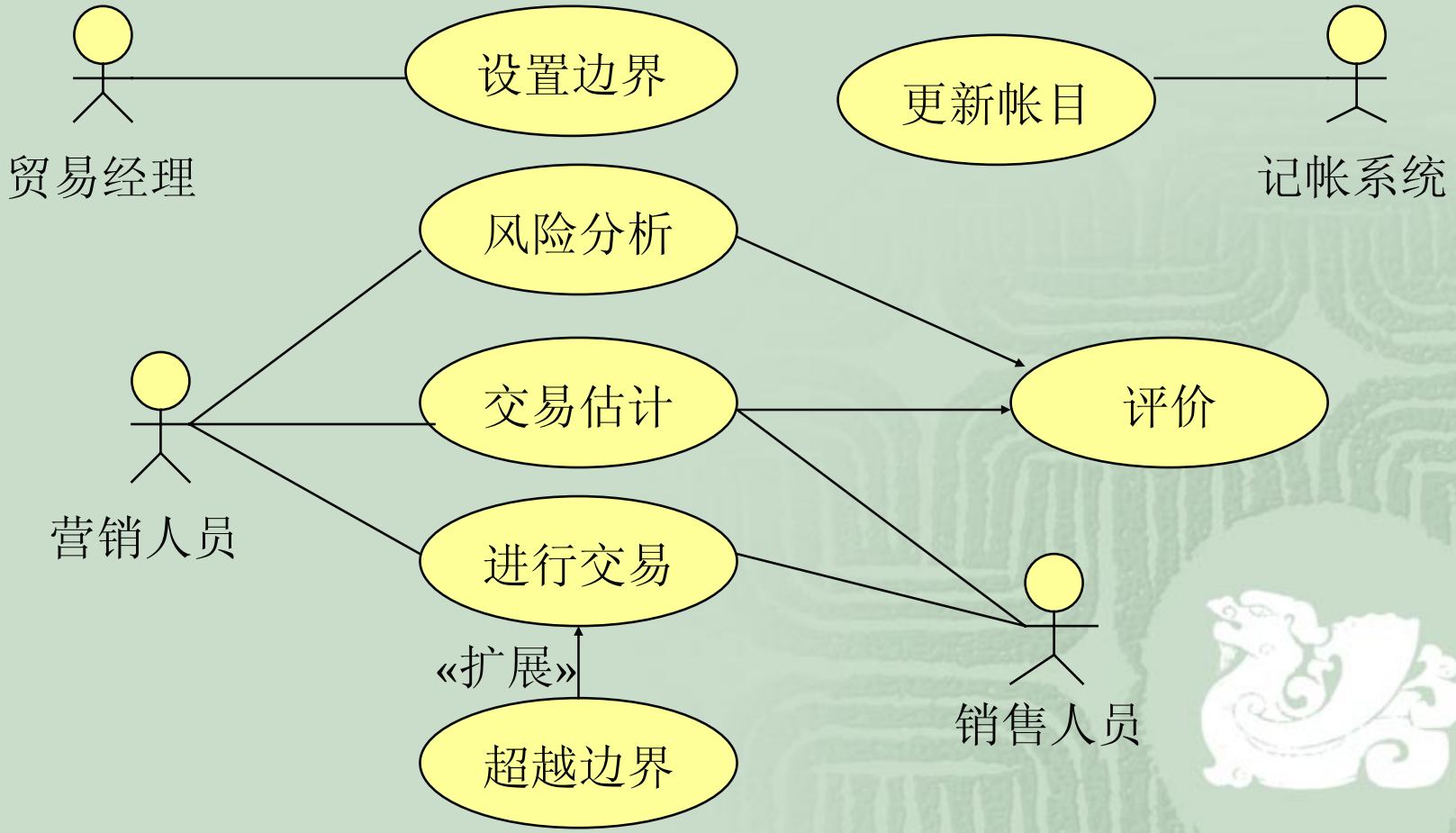
- 活动图（Activity Diagram）
- 用例图（Use Case Diagram）
- 状态机图（State Machine Diagram）
- 交互图（Interaction Diagram）
  - 通信图（Communication Diagram）
  - 顺序图（Sequence Diagram）
  - 定时图（Timing Diagram）
  - 交互概览图（Interaction Overview Diagram）

用例图（Use Case Diagram）

用例图描述系统外部的参与者与系统的用例之间的某种联系。

- 所谓用例是指对系统提供的功能（或称系统的用途）的一种描述；
- 参与者是那些可能使用这些用例的人或外部系统；
- 用例和参与者之间的联系描述了“谁使用哪个用例”。

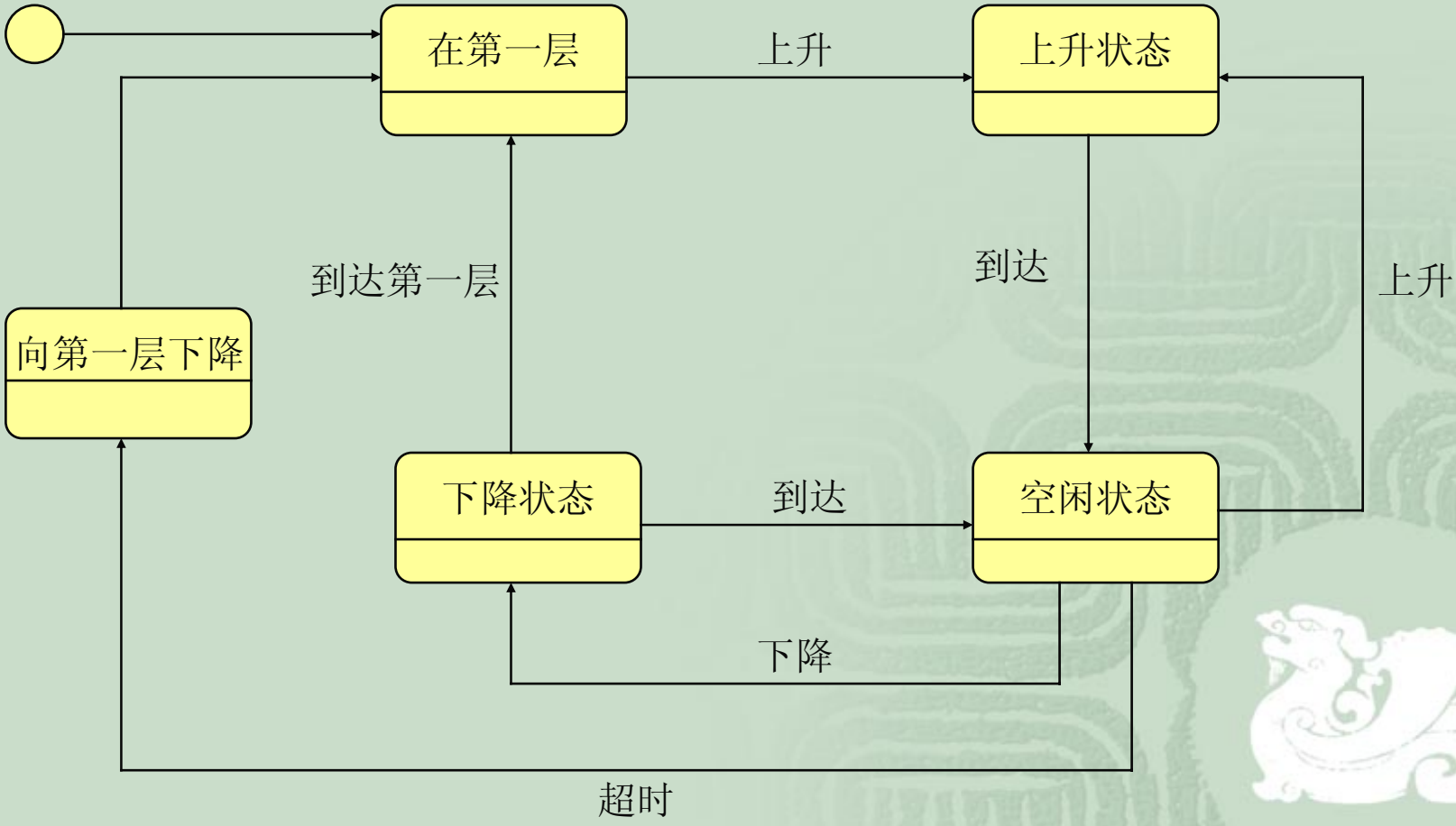
用例图示例



状态机图
（State Machine Diagram）

- 一个状态机描述了一种对象或一种交互在其生命期中响应各种事件而经历的一系列状态转换，以及对事件的反应效果，如动作和活动。

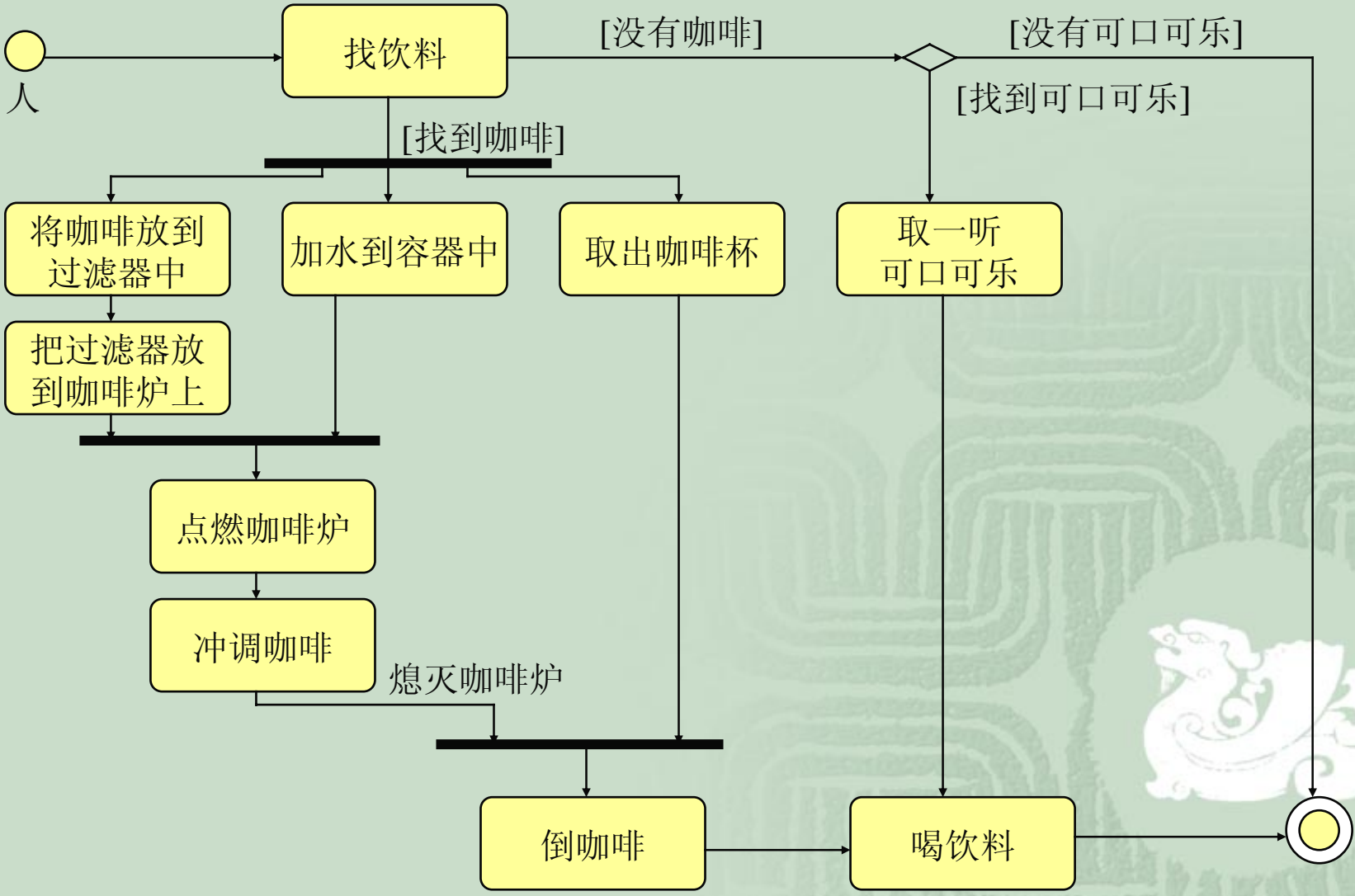
状态机图示例



活动图（Activity Diagram）

- 活动图描述系统中各种活动的执行顺序，通常用于描述一个操作中所要进行的各项活动的执行流程。同时，它也常被用来描述一个用例的处理流程，或者某种交互流程。
- 活动图由一些活动组成，图中同时包括了对这些活动的说明。当一个活动执行完毕之后，控制将沿着控制转移箭头转向下一个活动。活动图中还可以方便地描述控制转移的条件以及并行执行等要求。

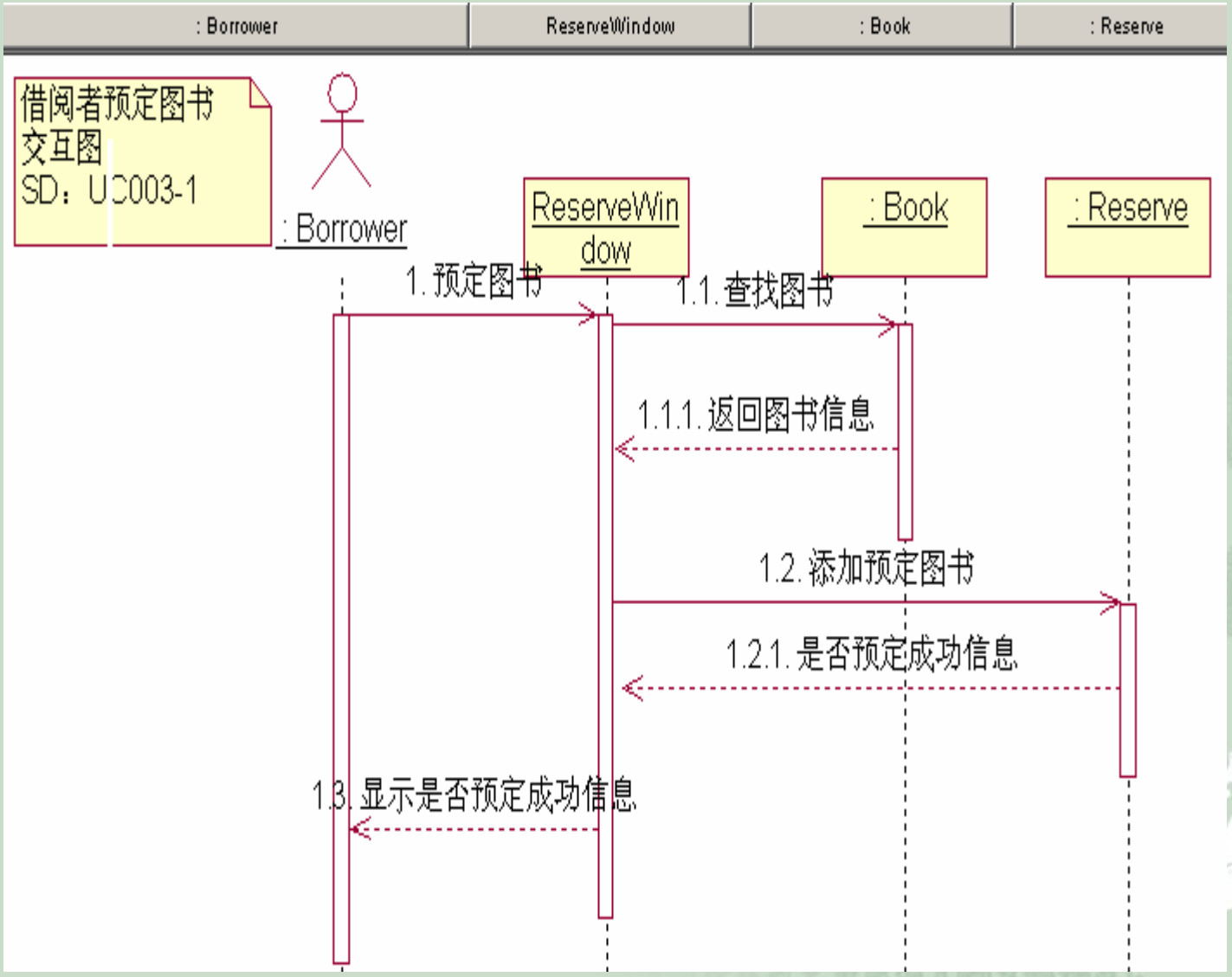
活动图示例


人
找饮料
[没有咖啡]
[没有可口可乐]
[找到可口可乐]
[找到咖啡]
将咖啡放到过滤器中
加水到容器中
取出咖啡杯
取一听可口可乐
把过滤器放到咖啡炉上
点燃咖啡炉
冲调咖啡
熄灭咖啡炉
倒咖啡
喝饮料

顺序图（Sequence Diagram）

- 顺序图描述对象之间动态的交互关系，着重体现对象间消息传递的时间顺序。
- 顺序图描述了这些对象随着时间的推移相互之间交换消息的过程。

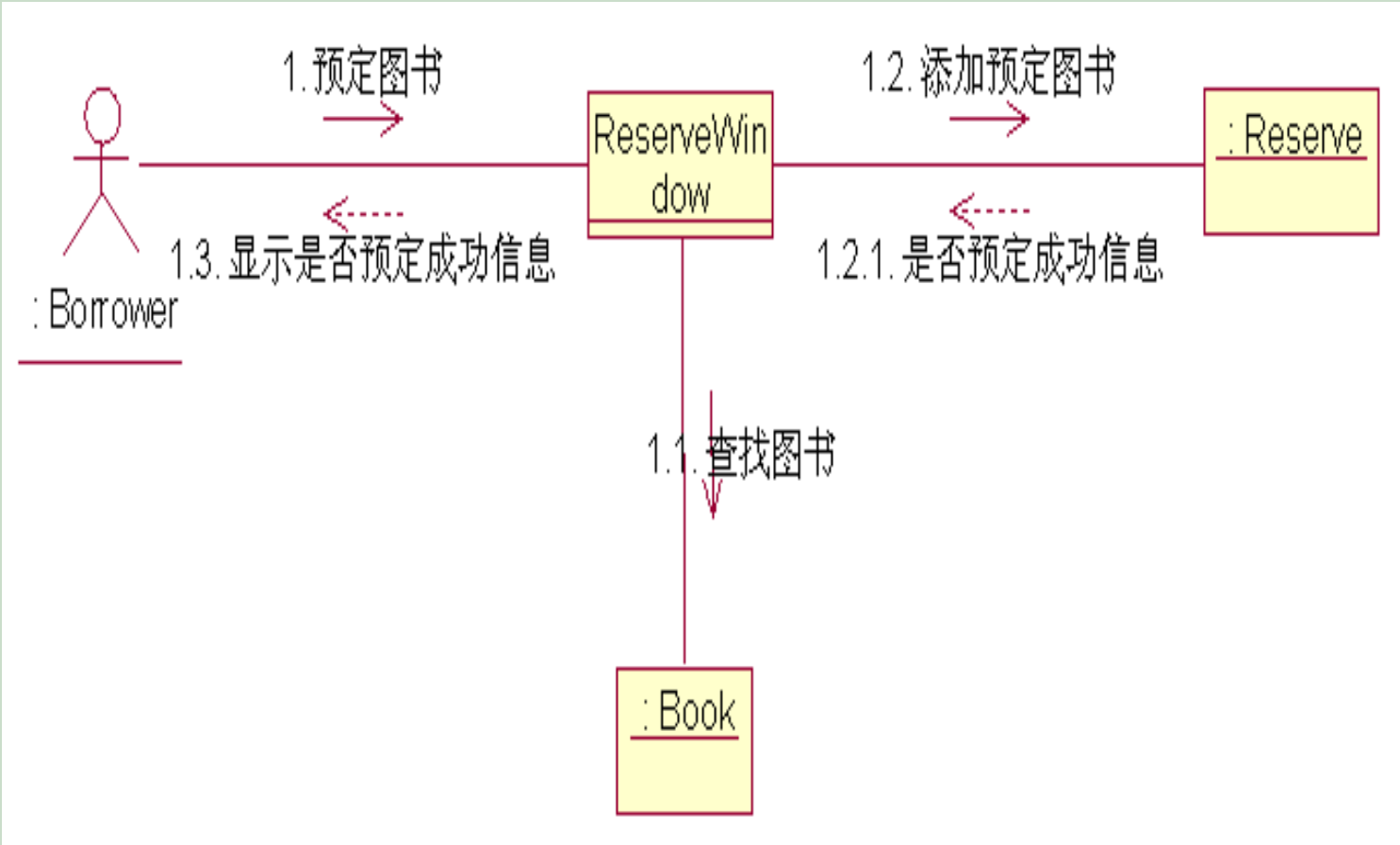
顺序图示例


: Borrower
ReserveWindow
: Book
: Reserve
借阅者预定图书
交互图
SD：UC003-1
: Borrower
ReserveWin
dow
: Book
: Reserve
1. 预定图书
1.1. 查找图书
1.1.1. 返回图书信息
1.2. 添加预定图书
1.2.1. 是否预定成功信息
1.3. 显示是否预定成功信息

通信图
（Communication Diagram）

- 通信图描述对象之间动态的交互关系，着重体现对象间的链接，而不专门突出这些消息发送的时间顺序。

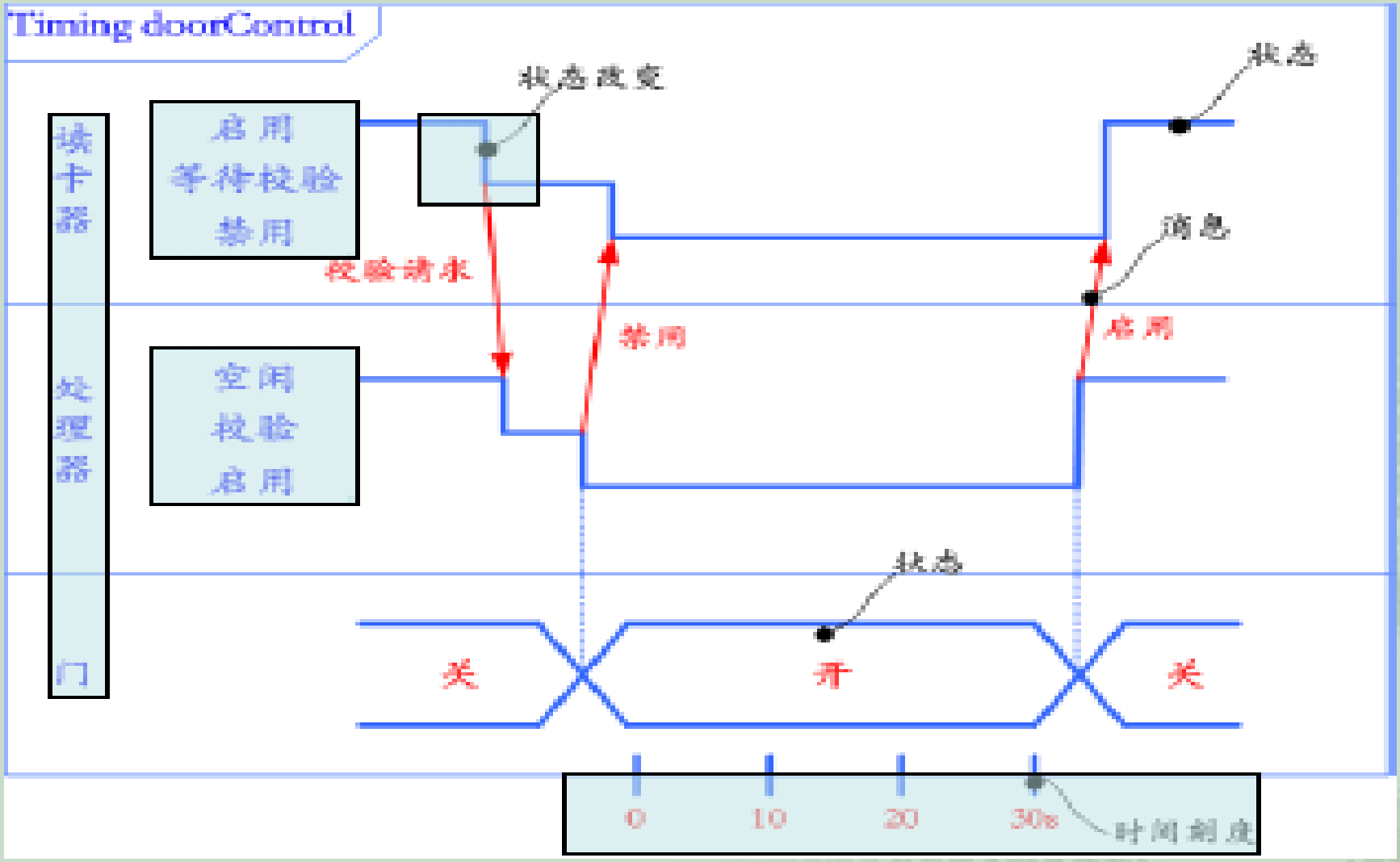
通信图示例


1.预定图书
1.2.添加预定图书
ReserveWin
dow
: Reserve
1.3.显示是否预定成功信息
1.2.1.是否预定成功信息
: Borrower
1.1.查找图书
: Book

定时图（Timing Diagram）

- 定时图描述对象之间动态的交互关系，着重体现与交互元素的状态转换或条件变化有关的详细时间信息。
- 定时图表示对象处于某一状态中的持续时间，强调时间对系统状态变化的影响。

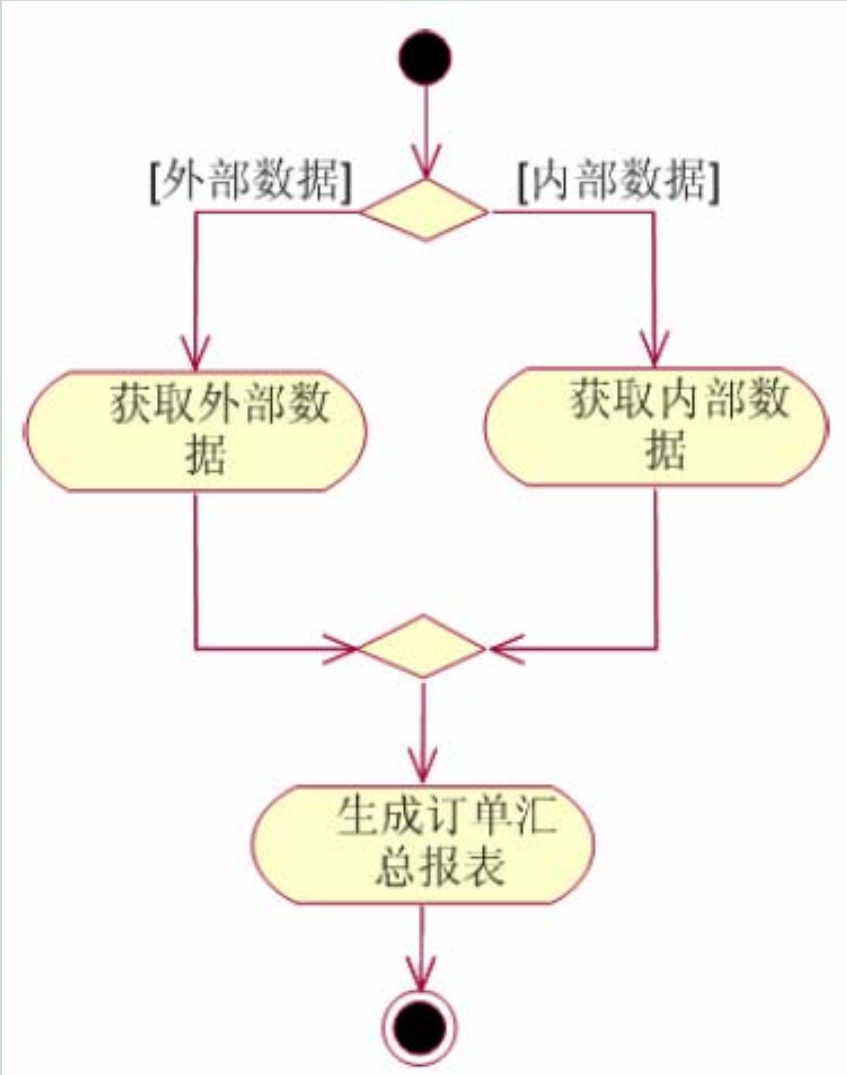
定时图示例


Timing doorControl
读卡器
处理器
门
启用
等待校验
禁用
空闲
校验
启用
校验请求
状态改变
状态
消息
禁用
启用
状态
关
开
关
0
10
20
30s
时间刻度

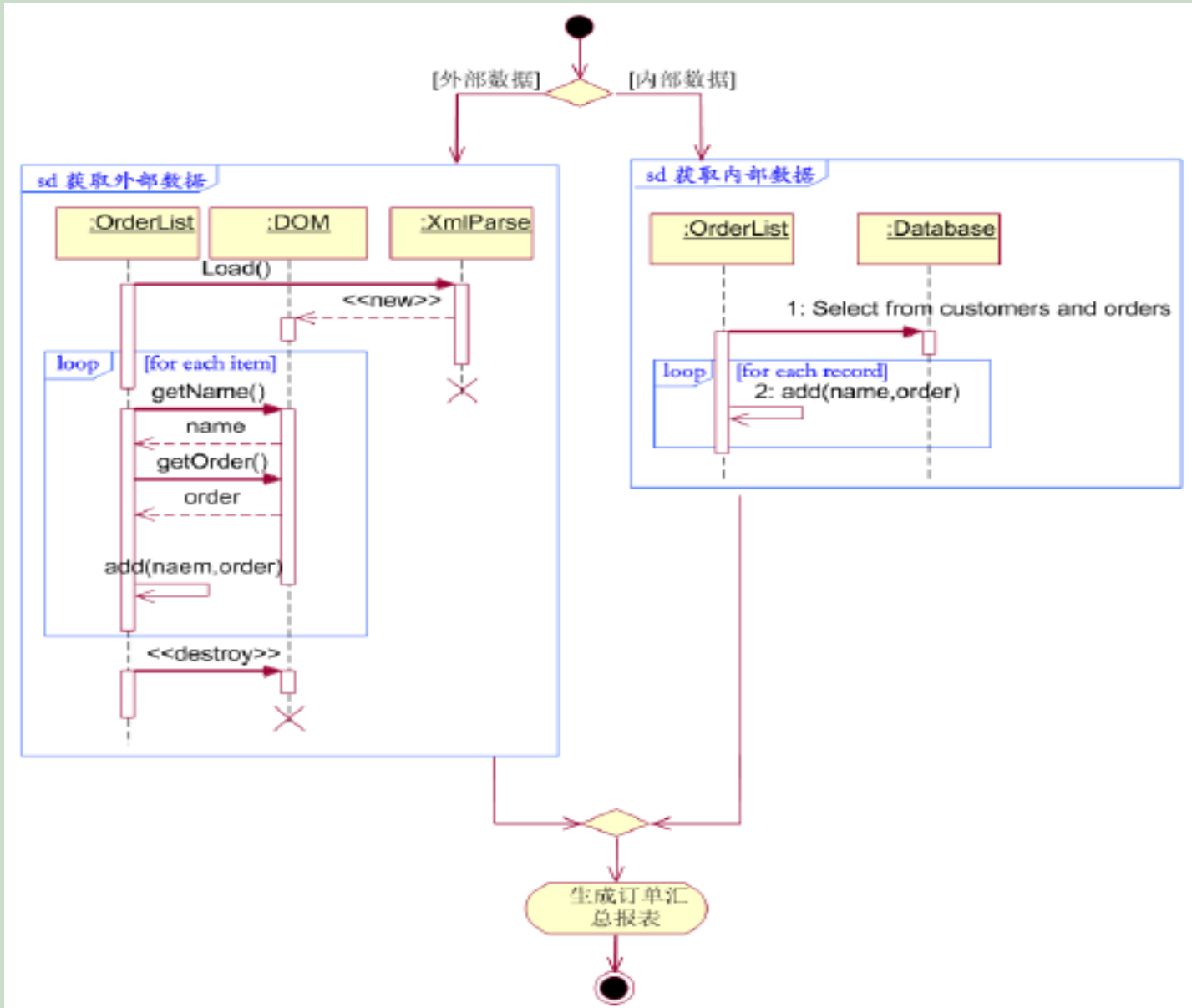
交互概览图（Interaction Overview Diagram）

- 交互概览图用于从总体上显示交互序列之间的控制流。
- 交互概览图中用顺序图来替换活动图中的活动。

用活动图表述主线



用顺序图描述细节


[外部数据]
[内部数据]
sd 获取外部数据
:OrderList
:DOM
:XmlParse
Load()
<<new>>
loop
[for each item]
getName()
name
getOrder()
order
add(naem,order)
<<destroy>>
sd 获取内部数据
:OrderList
:Database
1: Select from customers and orders
loop
[for each record]
2: add(name,order)
生成订单汇
总报表

需重点关注的图

- 用例图
- 类图
- 状态机图
- 顺序图和通信图
- 活动图

UML的视图

- 可视化、详述、构造和文档化一个软件密集型系统，要求从几个角度去观察系统。
- 各种人员在不同时间以不同方式看系统。
- 系统体系结构也许是最重要的制品，它可以驾驭不同的观点，并在整个项目的生命周期内控制对系统的增量迭代式开发。
- 软件体系结构不仅关心结构和行为，而且关心用法、功能、性能、弹性、复用、可理解性、经济与技术约束及其折中，以及审美。

UML的视图

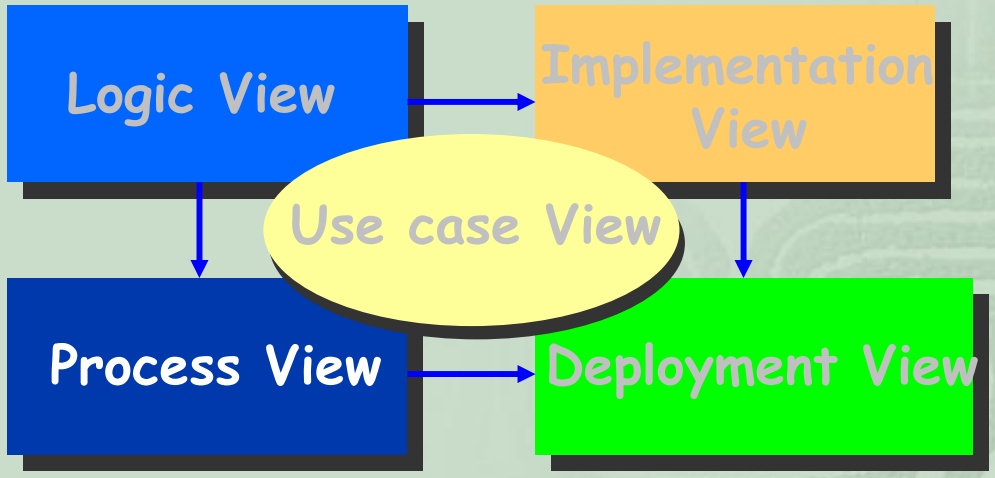
- UML用五个互连的视图来描述软件密集型系统的体系结构。
- 每一种视图从一个特定的方面对系统体系结构进行投影。
- 一个完整的系统模型由多个视图来共同描述。

UML的4+1 视图模型

用例视图： 描述系统应该具备的功能

逻辑视图：
描述系统的静态结构和动态行为

进程视图：
描述系统的并发与同步结构



实现视图：
描述系统的代码组件的组织结构

部署视图：
描述系统的具体部署

UML的视图

- 五种视图中的每一种都可单独使用，使不同的人员能关注于他们最为关心的体系结构问题。
- 五种视图也会互相作用，如部署视图中的结点拥有实现视图的构件，而这些构件又表示了逻辑视图和进程视图中的类、接口、协作以及主动类的物理实现。

内容

- UML是什么
- UML有什么用处
- UML的历史
- UML语言体系
- **建模与UML**
- UML工具

建模与UML

UML的应用领域：

- 软件密集型系统建模：企业信息系统、银行与金融服务、电信、运输、国防、航天、零售、医疗、电子、科学、基于Web的分布式服务等。
- 非软件系统建模：法律系统的工作流程、医保系统的结构和行为等。

建模与UML

- 模型是对一个对象或物体的简化表示
- 在软件开发过程中，模型主要用来描述问题域和软件域
  - 问题域：业务、业务规则、业务流程、工作流程
  - 软件域：软件组成、软件结构、软件部署

需要哪些模型

可能需要在不同阶段从不同角度建模，例如：

- 从使用者角度，系统用来做什么、如何使用；
- 从设计者角度，为完成功能系统需要哪些对象、如何组织、怎么交互；
- 从实现者角度，每个对象什么样、对象间如何组装。

有效使用UML

在UML建模原则的指导下，选择合适的建模元素和图，并使用必要的文字描述，构建系统的模型。

UML可建立的模型种类

按产生模型的阶段性分类：

- 业务模型
- 需求模型
- 设计模型
- 实现模型
- 数据库模型

UML可建立的模型种类

按模型的用途分类：

- 功能模型 --- 用例图
- 静态模型 --- 类图
- 动态模型 --- 顺序图，活动图，状态机图

内容

- UML是什么
- UML有什么用处
- UML的历史
- UML语言体系
- 建模与UML
- **UML工具**

UML工具

- Rational Rose（功能强大，市场占有率高）
- StarUML
- Enterprise Architect
- ArgoUML
- Visio
- ……
- 太多了，更多的请访问

http://www.umlchina.com/Tools/Newindex1.htm

内容

- UML是什么
- UML有什么用
- UML的历史
- UML语言体系
- 建模与UML
- UML工具