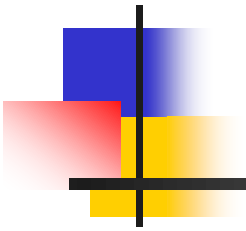
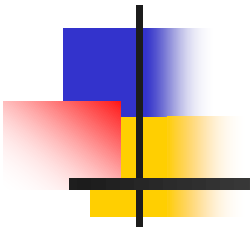
第4章 面向对象的软件分析与设计

目录



目标

- 软件体系结构 4+1建模及UML语言实现
- MVC设计模式
- 基于用例驱动的面向对象软件分析与设计
  - 针对具体的软件体系结构风格
  - 软件体系结构



4.1　4+1模型及UML语言实现

4.2　面向对象的软件工程

4.3　用例图

4.4　活动图

4.5　用户界面设计

4.6　类图

4.7　交互图

4.8　包图

4.9 系统与子系统

4.10 部署图

4.1 4+1模型及UML语言实现

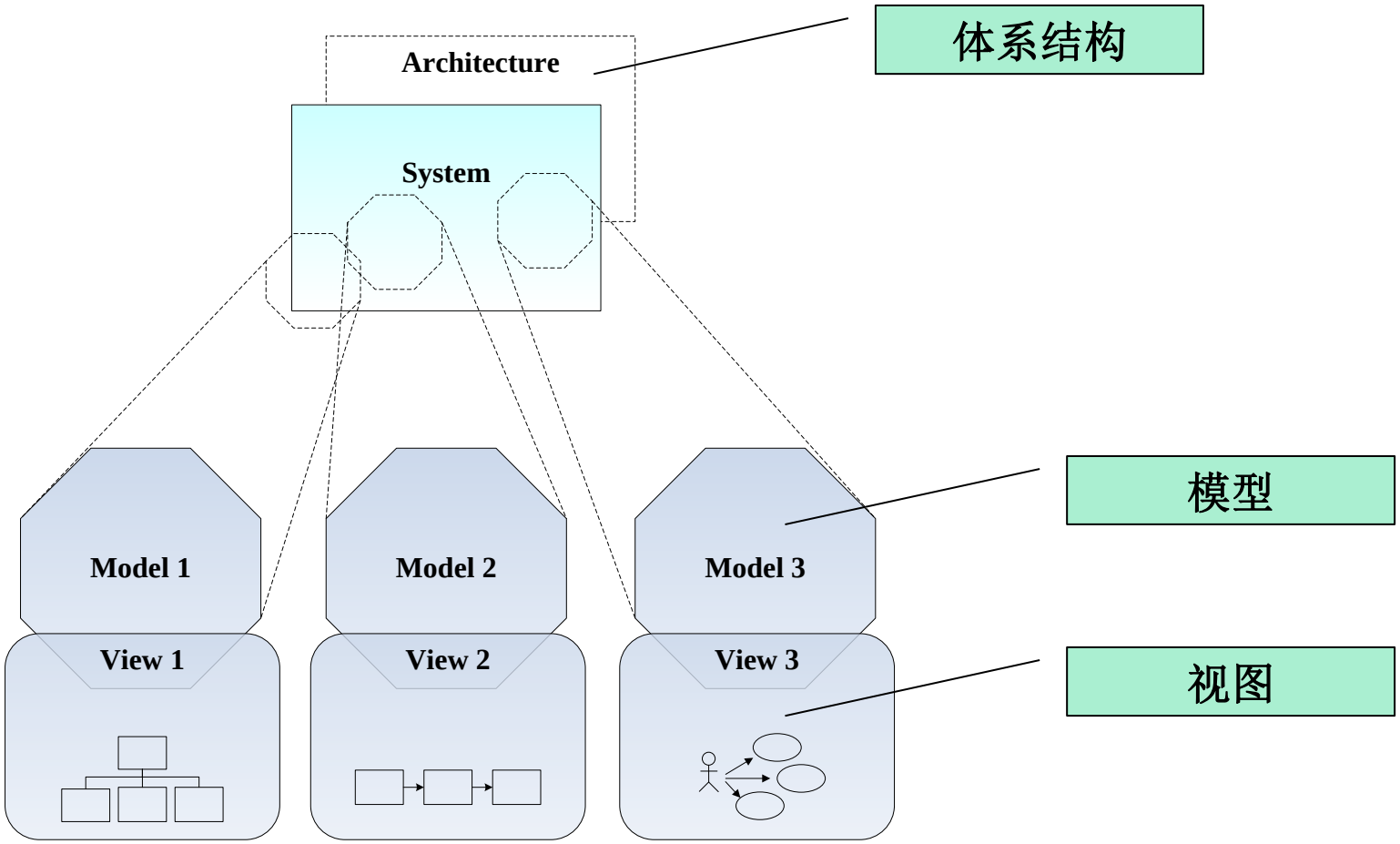
4.1.1 软件体系结构

- 什么是软件体系结构
- 卡内基梅隆大学的软件工程研究所在网站上公开征集软件体系结构的定义，至今已有百余种。

  其中，较有影响力的定义包括：
  - 卡耐基梅隆大学的Garlan于1993年：**构件+连接件+约束**
  - 在2000年发布的IEEE Std1471-2000：软件系统的基本组织，包含构件、构件之间、构件与环境之间的关系，以及相关的设计与演化原则等。

4.1 4+1模型及UML语言实现

4.1.1 软件体系结构



如何理解软件体系结构

（1）软件体系结构是不可见的，它是对系统的高层抽象（建模）

（2）该定义中没有涉及怎么建模，只是告诉我们应该对系统进行建模。
具体的建模方法会告诉我们如何建模。

（3）针对特定的开发过程模型和建模方法，合适的描述语言会有利于软件开发过程。

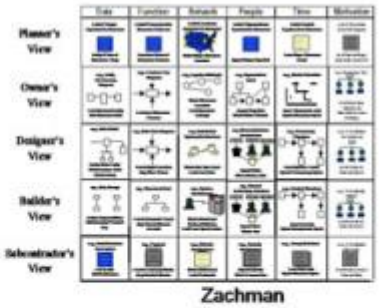
例如：在Rational统一过程（RUP）开发过程模型中，使用UML语言对软件体系结构进行描述。

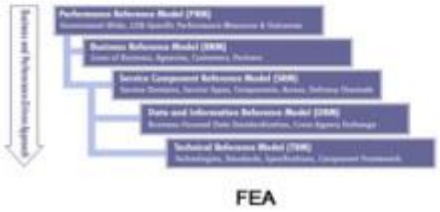
4.1 4+1模型及UML语言实现

4.1. 2 软件体系结构建模方法：怎么建模

- 4+1
- DoDAF
- MODAF
- TOGAF
- Zachman

等等

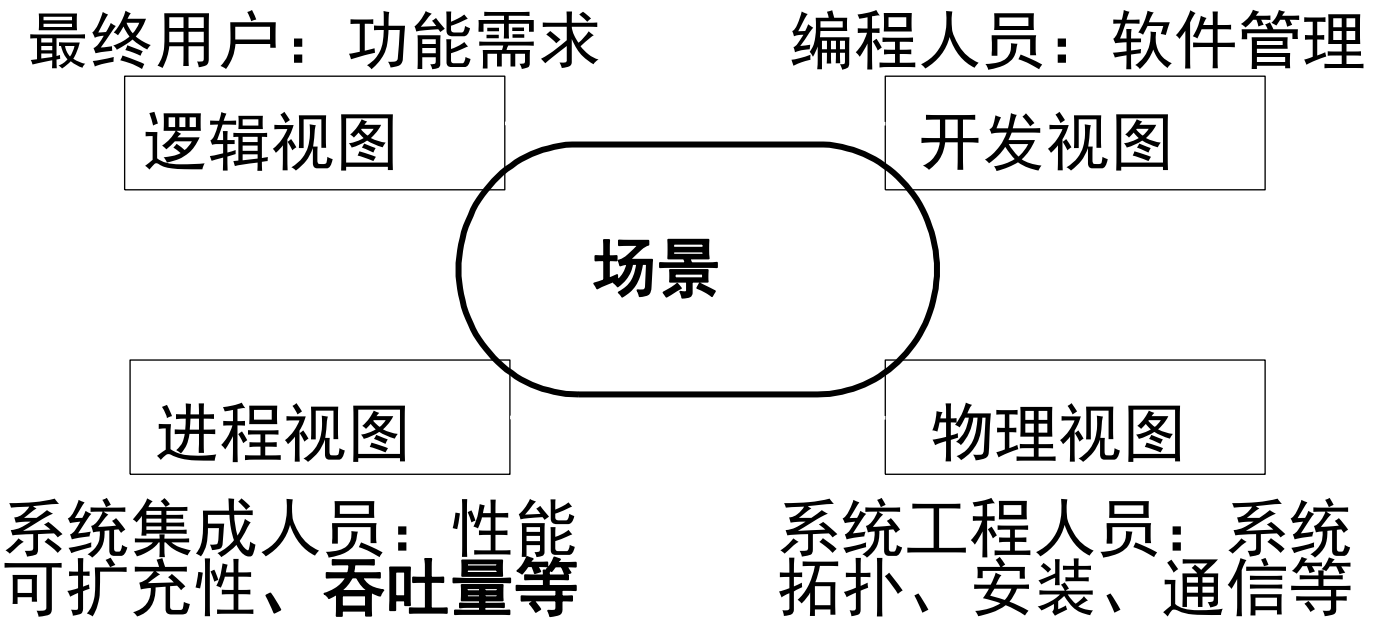




4.1 4+1模型及UML语言实现

4.1.3 4+1模型及UML语言实现（UML建模）

1995年，Philippe Kruchten在《IEEE Software》上发表了题为《The 4+1 View Model of Architecture》的论文，引起了业界的极大关注，并最终被RUP采纳。



4.1.3 4+1模型及UML语言实现（UML建模）

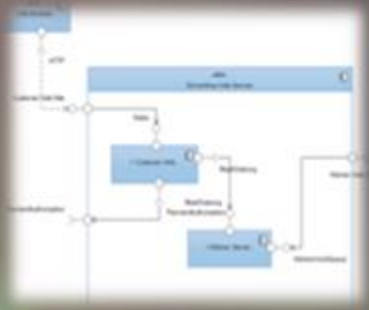
而Rational公司的Booch 从UML语言的角度出发给出了4+1模型的具体实现：

（1）用例视图（场景视图）
（2）设计视图（逻辑视图）
（3） 过程视图（进程视图）
（4） 实现视图（开发视图）
（5） 部署视图（物理视图）

UML：设计视图

逻辑视图

UML：实现视图

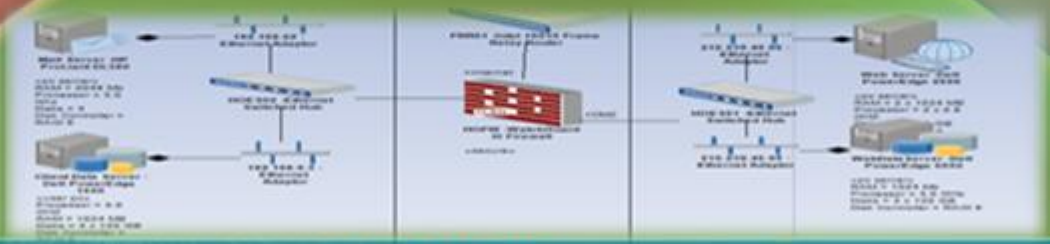


开发视图

进程视图

UML：过程视图

UML：用例视图



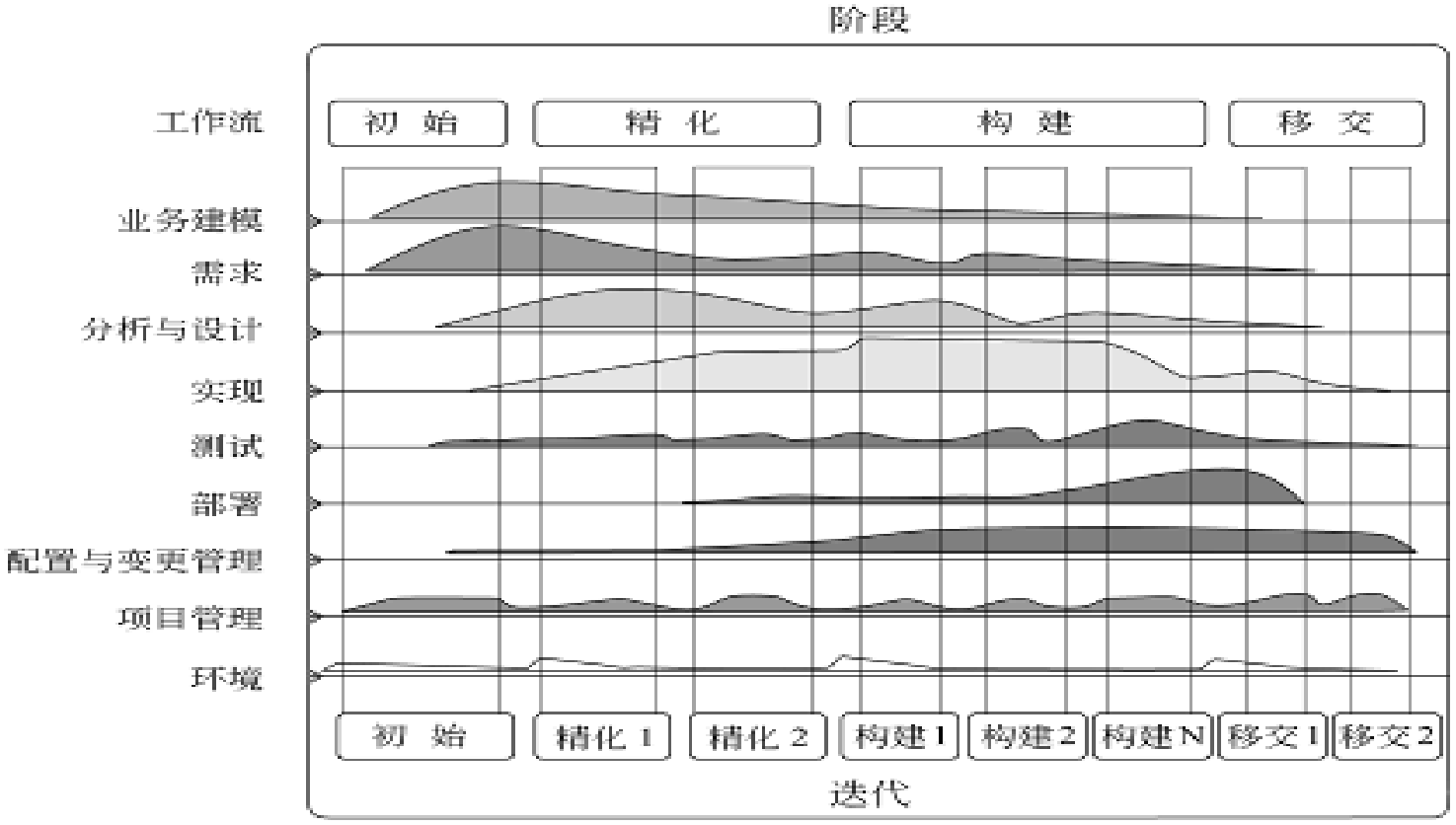
物理视图

UML：部署视图

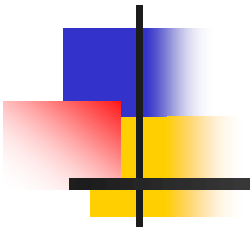
4.1 4+1模型及UML语言实现

4.1.4 4+1模型适用的软件过程模型：不仅仅RUP

RUP：软件生命周期模型也称为软件过程模型



RUP使用UML来制定软件系统的所有视图。



4.1　4+1模型及UML语言实现

4.2　面向对象的软件工程

4.3　用例图

4.4　活动图

4.5　用户界面设计

4.6　类图

4.7　交互图

4.8　包图

4.9 系统与子系统

4.10 部署图

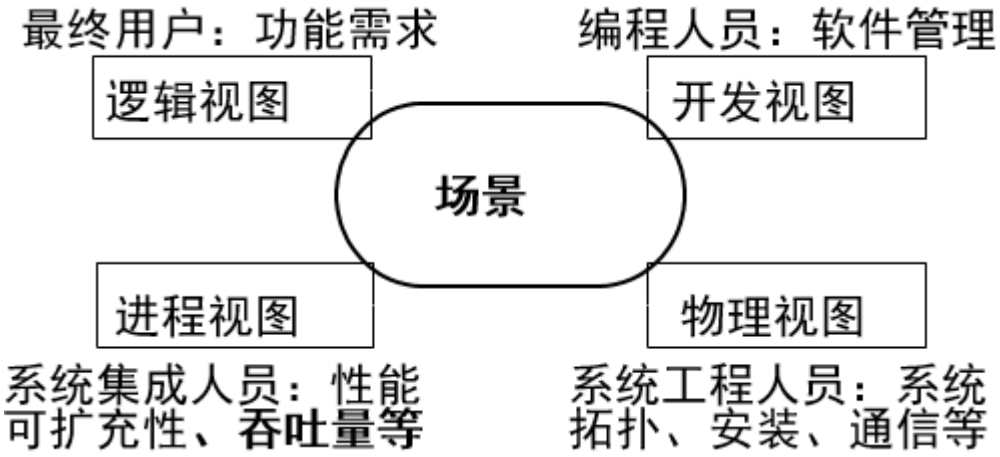
4.2 面向对象的软件工程

4.2.1 软件生命周期模型

瀑布模型、RUP、敏捷开发等

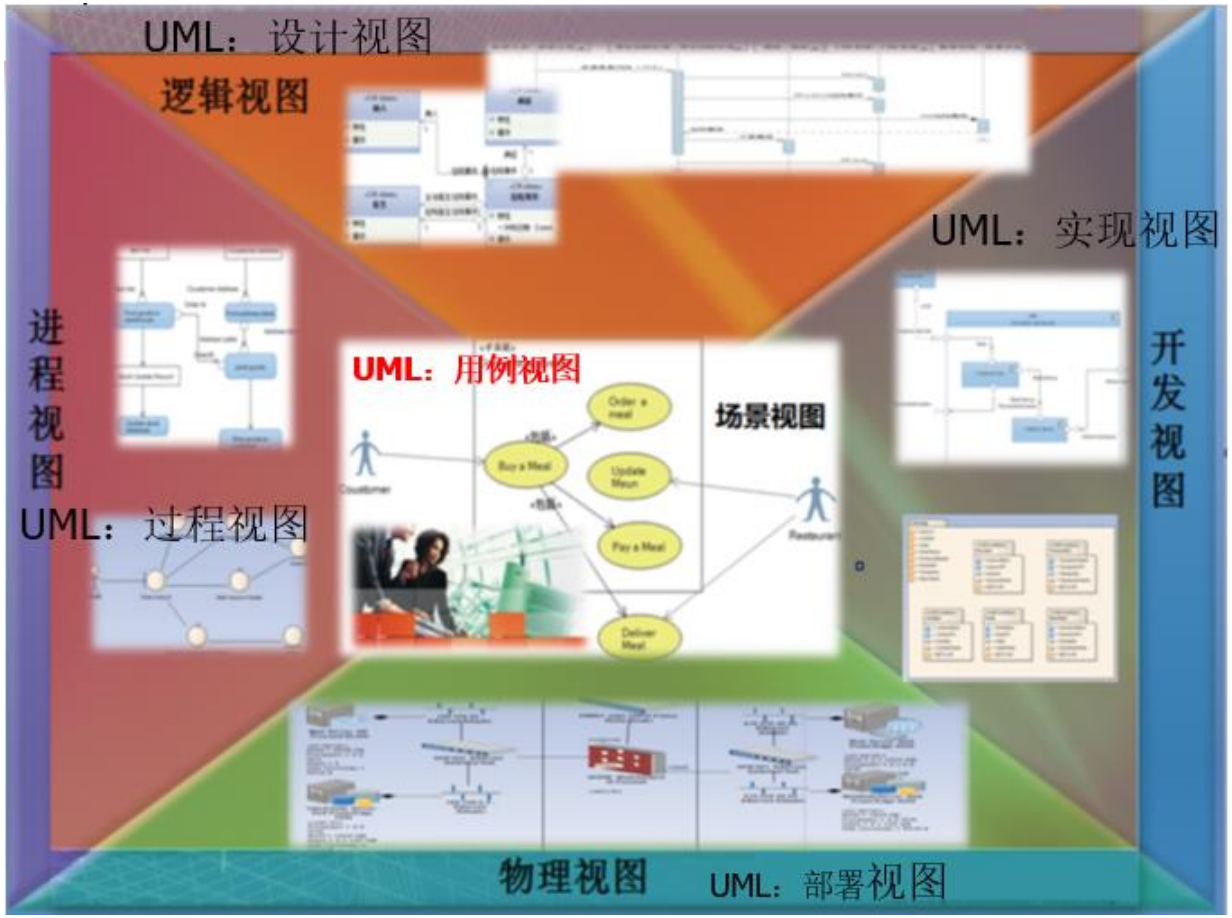
4.2.2 面向对象软件分析与设计（OOAD）

- 是一种思想，这种思想是人们分析问题和解决问题之后总结而成的
- 分析与设计：软件体系结构（ 4+1模型)



4.2.2 面向对象软件分析与设计（OOAD）

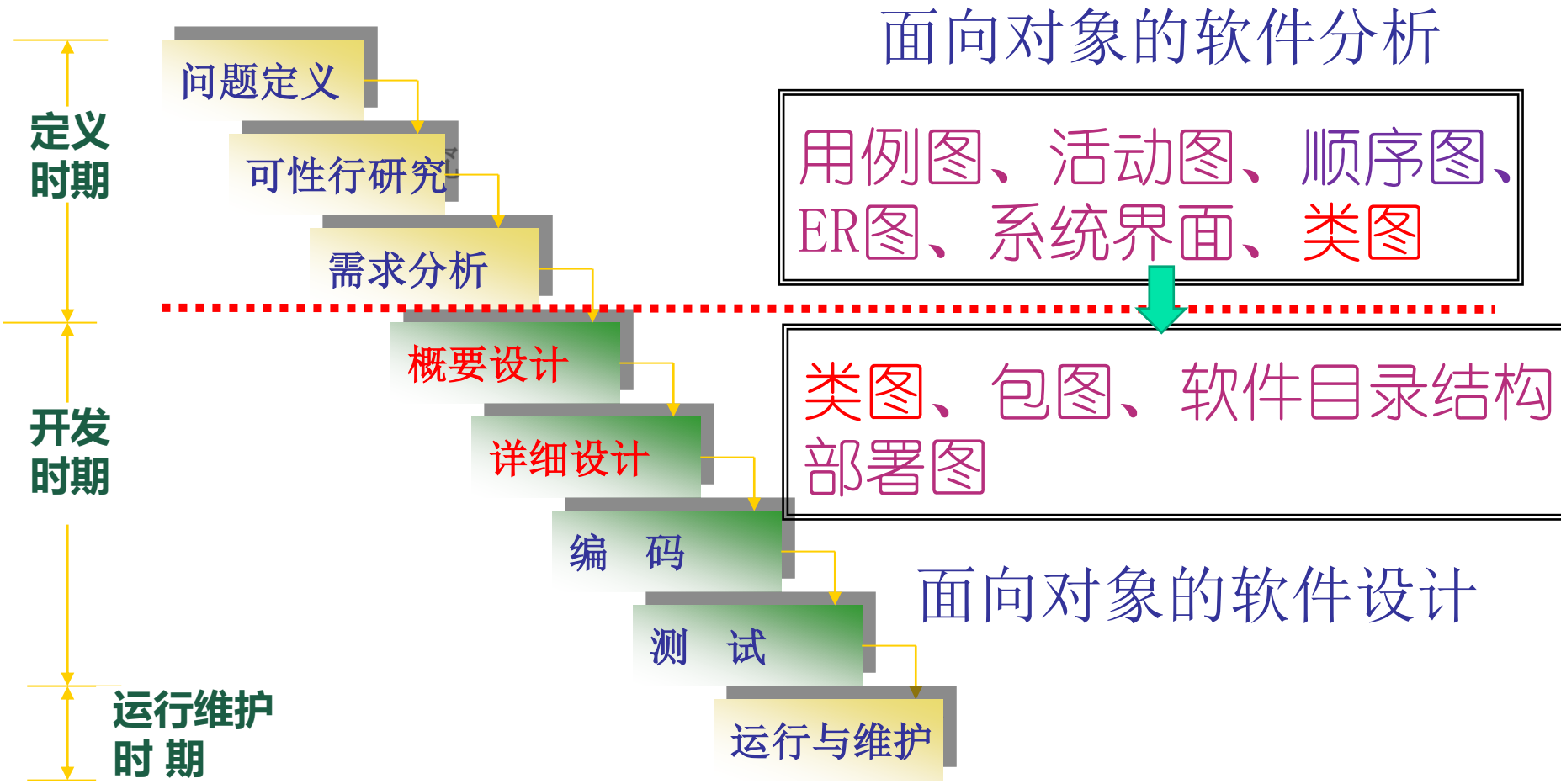
- 怎么建模4+1模型（UML语言实现）：下图



用例图、活动图、顺序图、ER图、系统界面、类图、包图、软件目录结构、部署图

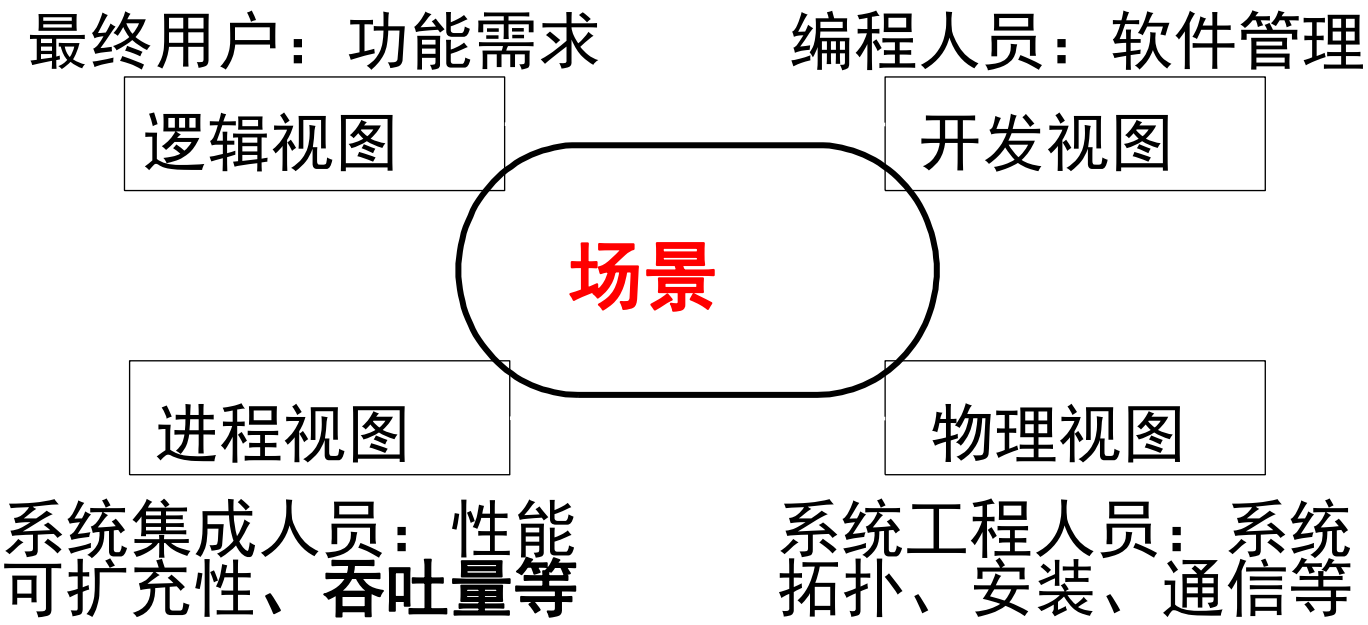
4.2 面向对象的软件工程

4.2.3 如何组织软件体系结构的实现？



4.2 面向对象的软件工程

4.2.4 面向对象软件分析与设计的起点

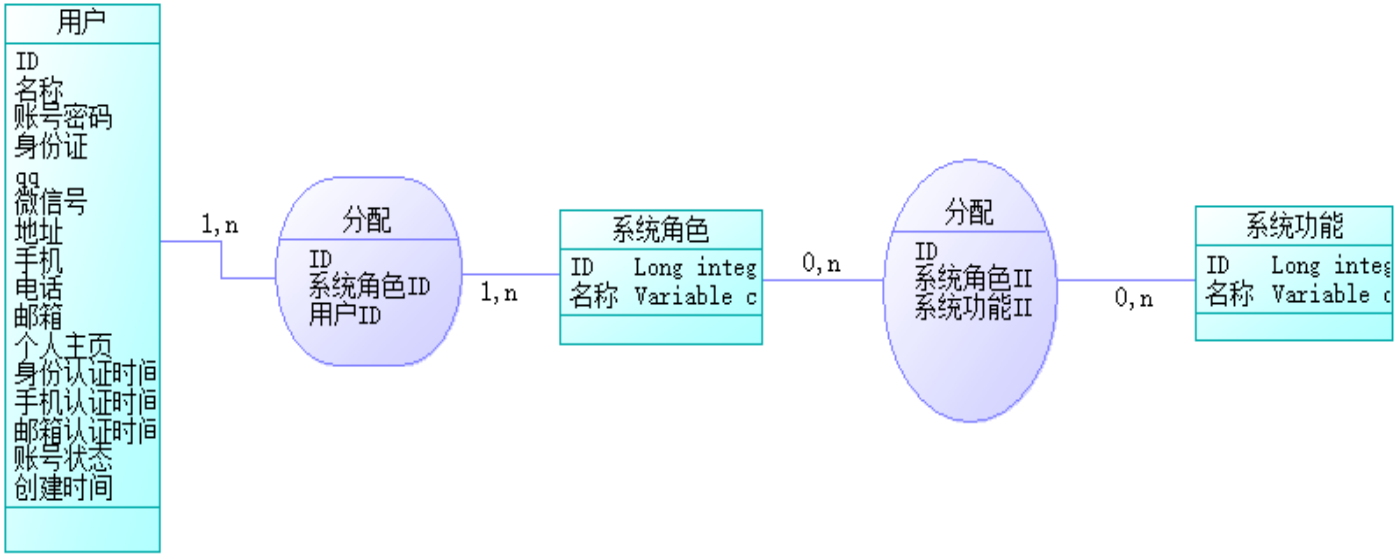


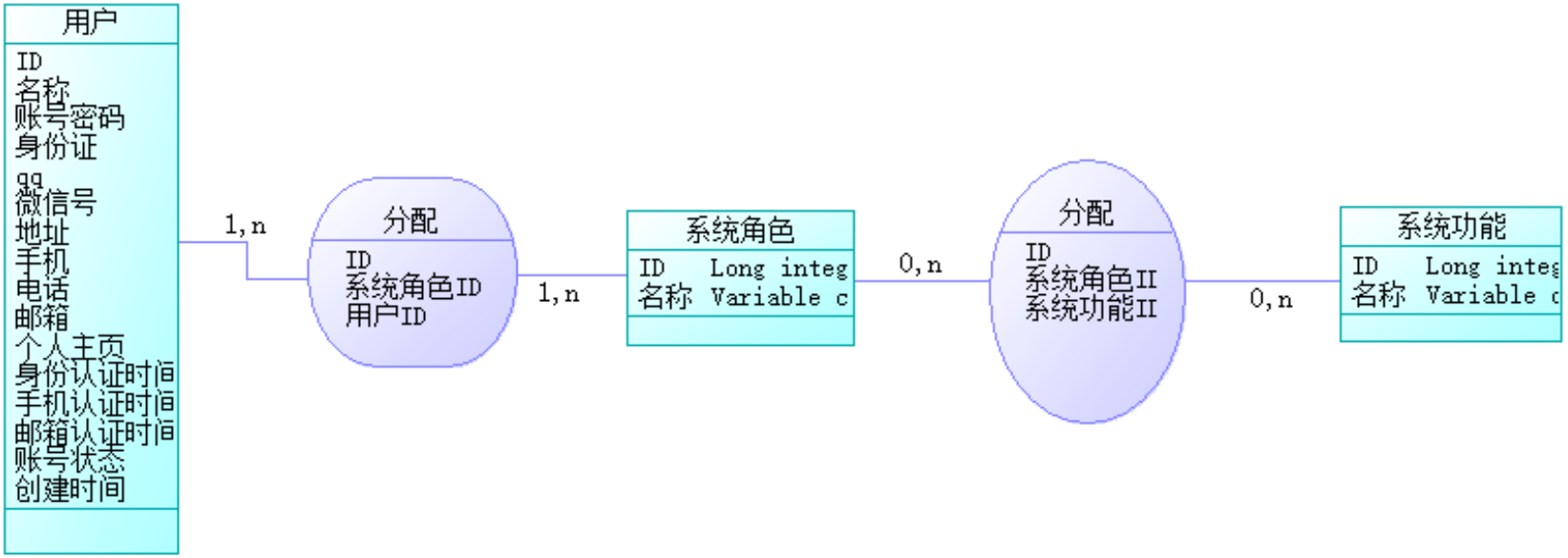
- 场景：用例
- 基于用例驱动的面向对象软件分析与设计

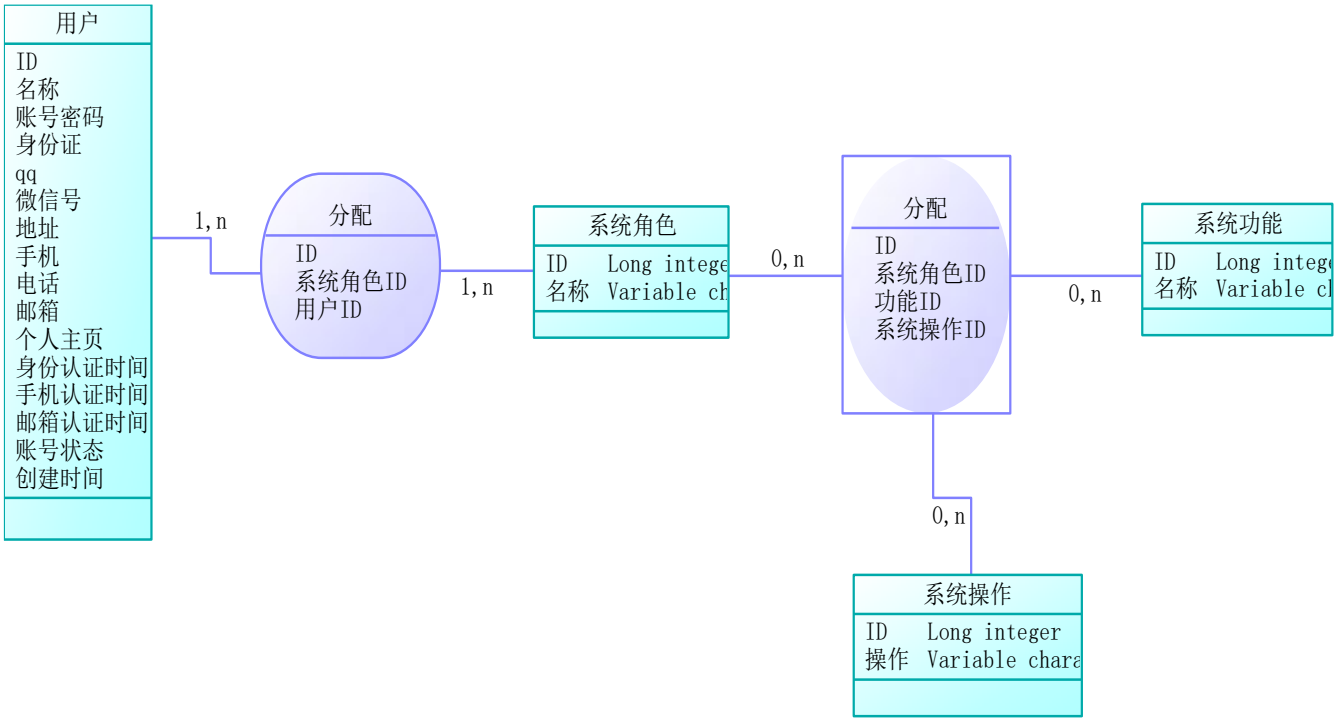
4.2 面向对象的软件工程

4.2.5 如何抽象出软件体系结构的类

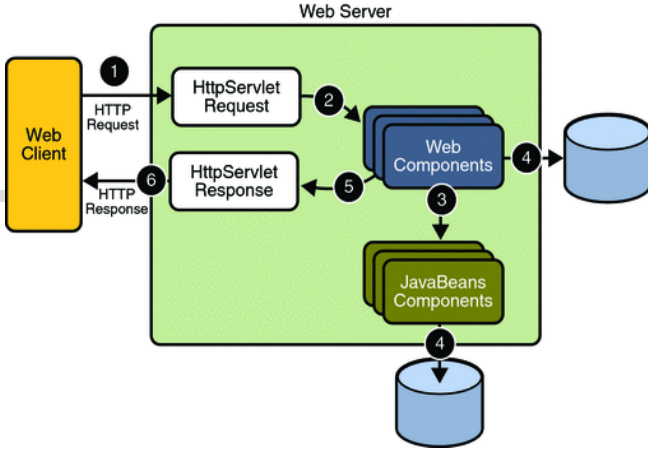
- 管理信息系统(MIS)：图书管理系统、采购系统等
- 抽象出软件体系结构的类：用例+MVC设计模式+ER图。



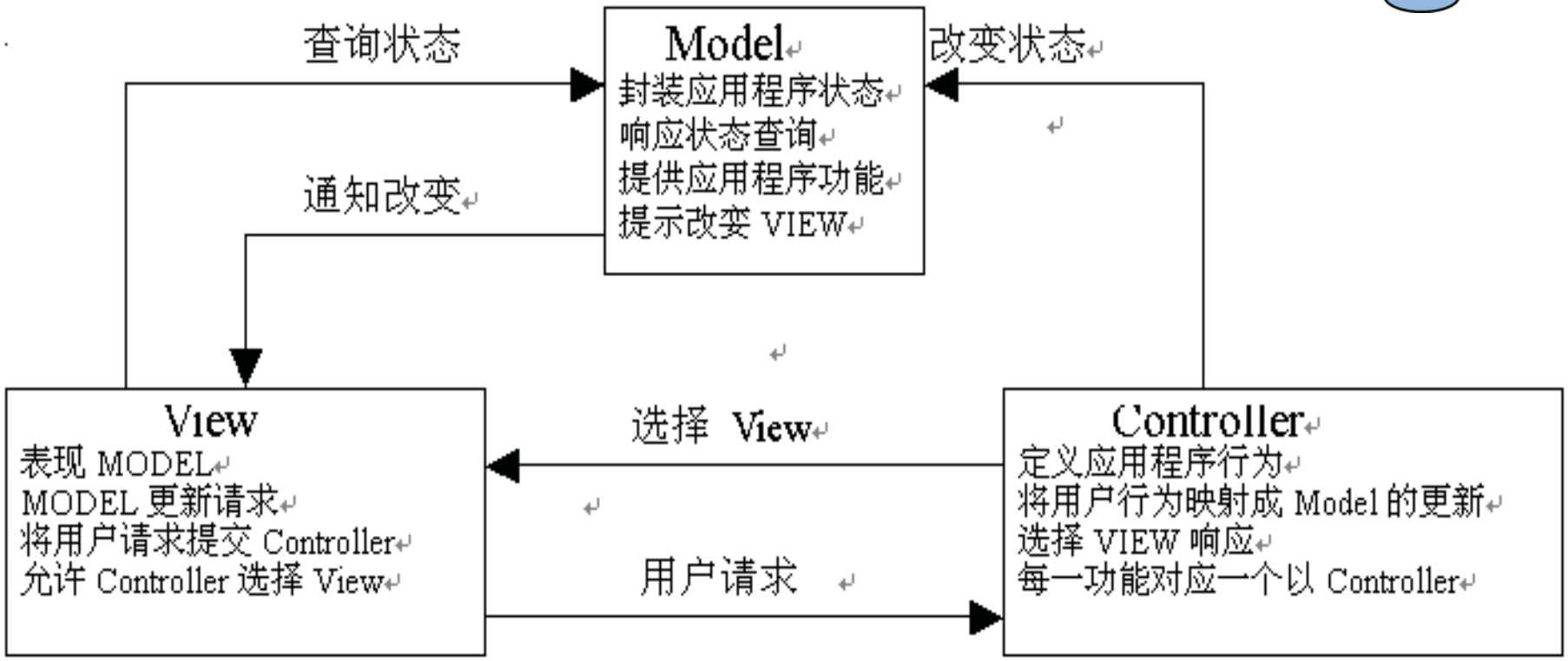

用户
ID
名称
账号密码
身份证
qq
微信号
地址
手机
电话
邮箱
个人主页
身份认证时间
手机认证时间
邮箱认证时间
账号状态
创建时间
1,n
分配
ID
系统角色ID
用户ID
1,n
系统角色
ID Long integ
名称 Variable c
0,n
分配
ID
系统角色II
系统功能II
0,n
系统功能
ID Long inte
名称 Variable


用户
ID
名称
账号密码
身份证
qq
微信号
地址
手机
电话
邮箱
个人主页
身份认证时间
手机认证时间
邮箱认证时间
账号状态
创建时间
1,n
分配
ID
系统角色ID
用户ID
1,n
系统角色
ID Long intege
名称 Variable ch
0,n
分配
ID
系统角色ID
功能ID
系统操作ID
0,n
系统功能
ID Long intege
名称 Variable cl
0,n
系统操作
ID Long integer
操作 Variable chara

4.2 面向对象的软件工程



4.2.6 分析与设计类：**MVC设计模式**



4.2 面向对象的软件工程

4.2.6 分析与设计类：**MVC设计模式**

- MVC设计模式

(1) 概述

MVC设计模式源于SmallTalk－80语言

在80年代初期被许多系统（Macintosh，Windows）所采纳:用户界面

后来伴随着软件设计模式的出现和面向对象技术的成熟，MVC模式也趋于完善，并成为一种典型的面向对象设计模式，它所应用的范围也不仅仅局限在用户界面上。

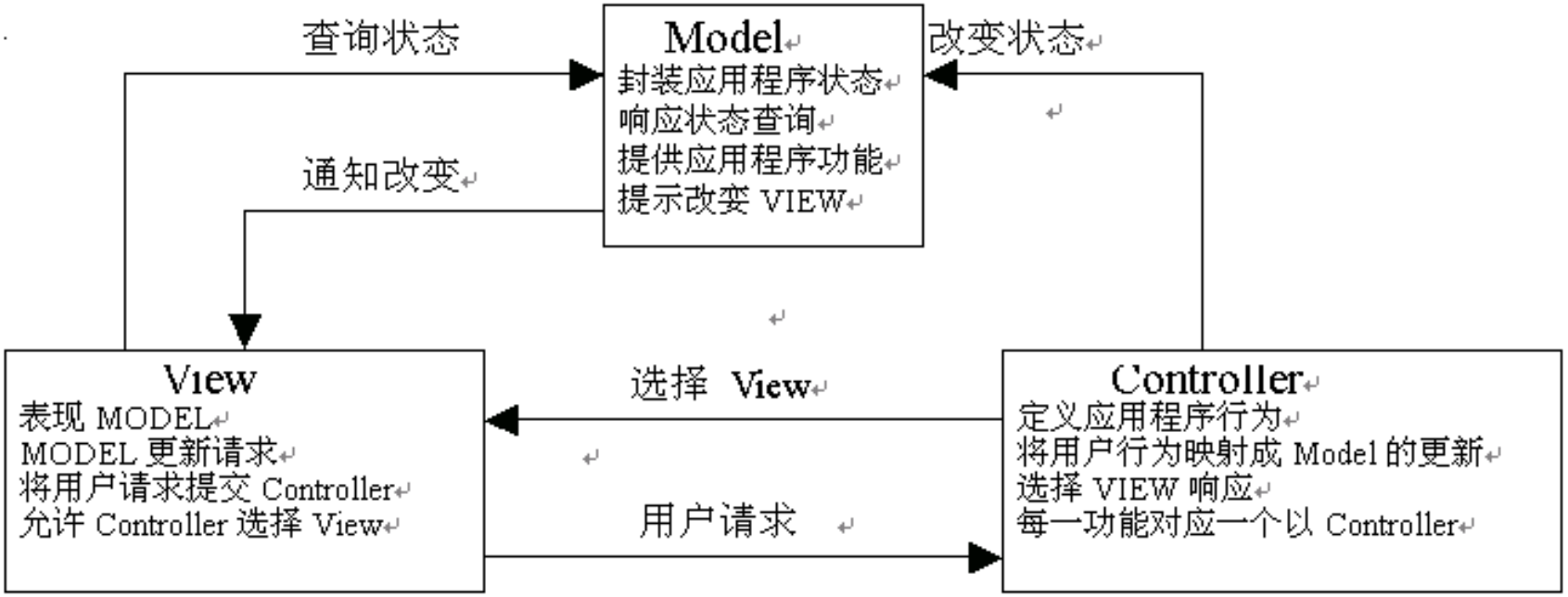
4.2 面向对象的软件工程

4.2.6 分析与设计类：**MVC设计模式**

- MVC设计模式

(2) MVC设计模式

MVC设计模式把应用程序抽象为Model（模型），View（视图），Controller（控制器）三个功能截然不同的部分，如图所示。



4.2 面向对象的软件工程

4.2.6 分析与设计类：MVC设计模式

- Model对象
  - Model对象代表应用程序数据和对这些数据访问和修改的业务逻辑。它是应用程序的核心部分，维护了业务的持久性.
  - 为Controller对象和View对象提供了被Model对象封装的某些应用业务逻辑

4.2 面向对象的软件工程

4.2.6 分析与设计类：MVC设计模式

- View对象
  - View对象代表用户界面。通过访问Model对象中的数据，View对象可视地显示Model对象的状态。
  - View对象与Model对象就如同形式与内容：
    - 当Model对象中的状态发生改变时，View对象所代表的用户界面也会相应地改变，达到内容与形式的一致。
    - 对于同一个Model对象，针对不同的用户请求,可以产生多个View对象。
  - View只是显示数据内容，处理数据的业务逻辑部分留给了Model对象和Controller对象

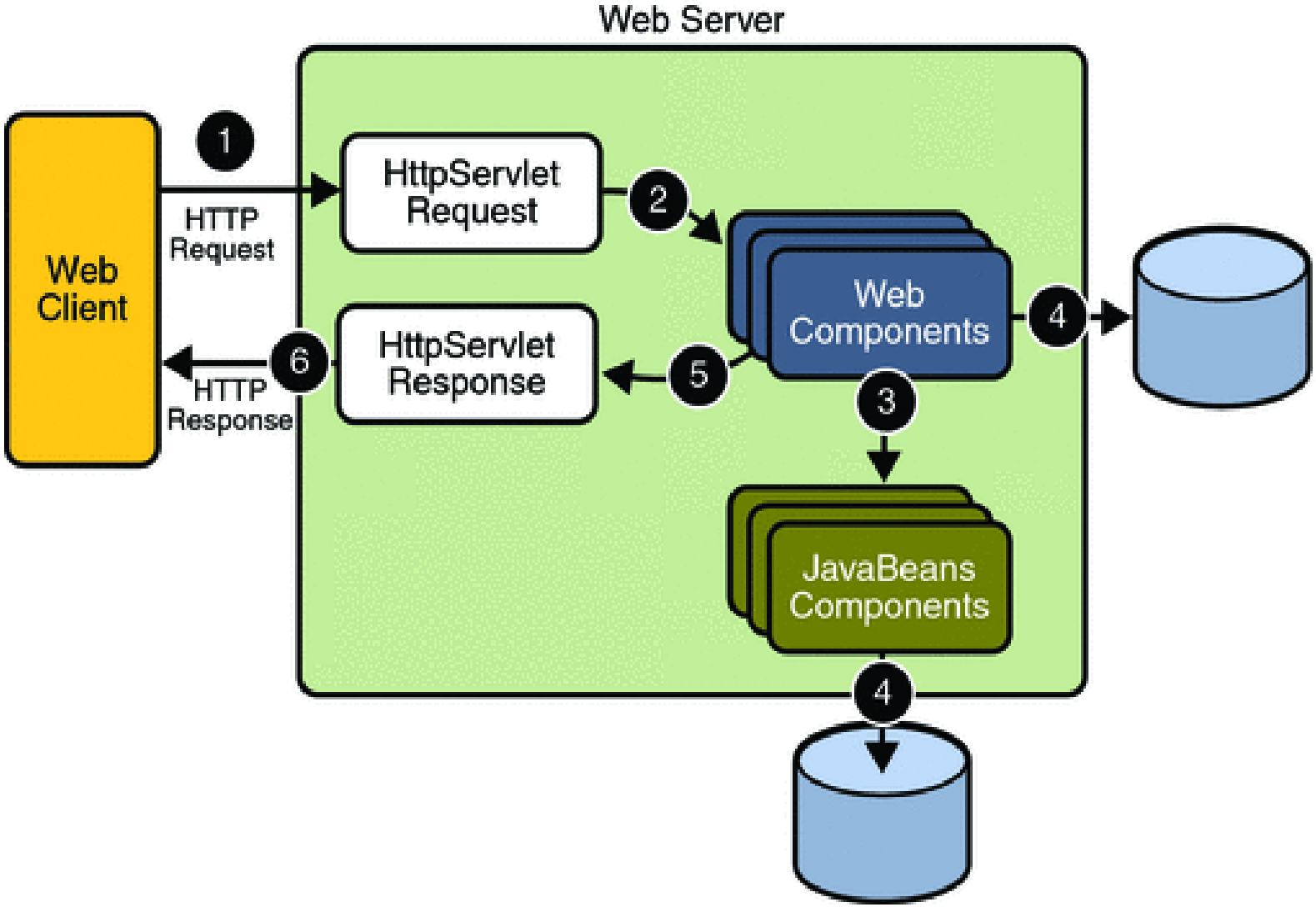
4.2 面向对象的软件工程

4.2.6分析与设计类：**MVC设计模式**

- Controller对象
  - Controller对象定义了应用程序的行为，负责View对象和Model对象之间的同步 。
    - 根据用户对View对象的操作完成对Model对象的更新。
    - 将Model对象状态的改变及时反应到View对象上。

4.2 面向对象的软件工程

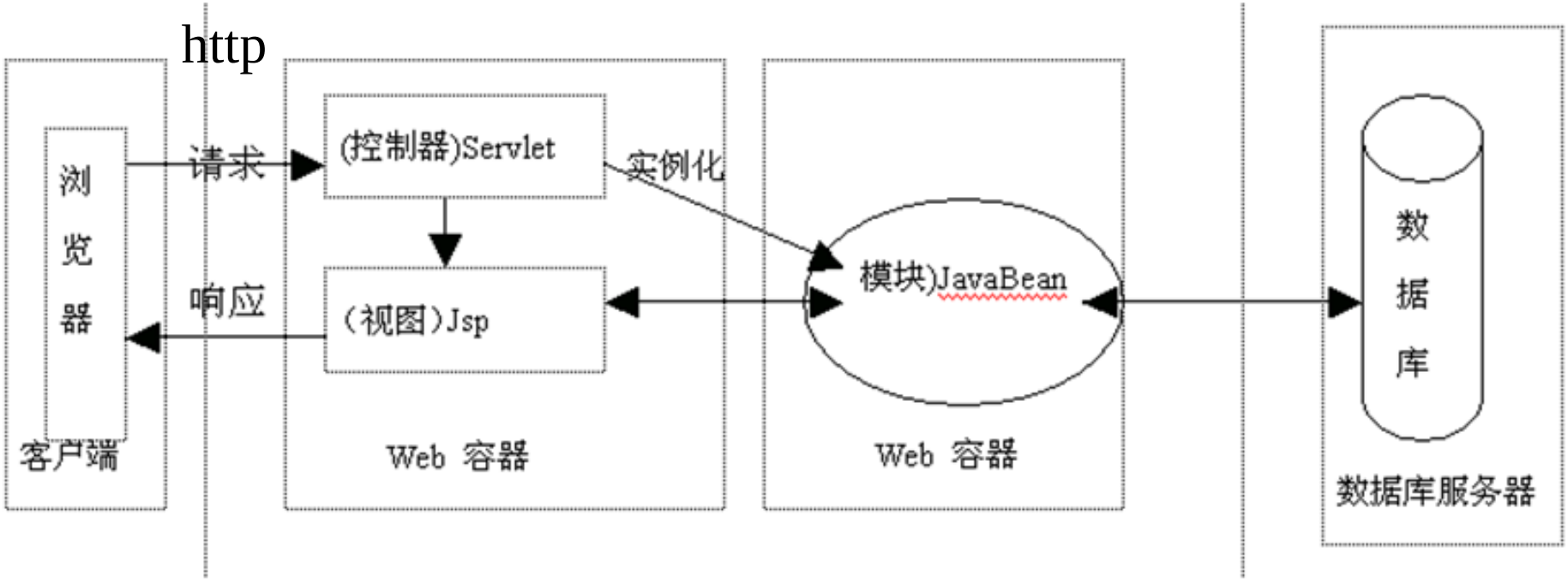
4.2.6 MVC设计模式和Web应用系统



4.2 面向对象的软件工程

4.2.6 MVC设计模式和Web应用系统

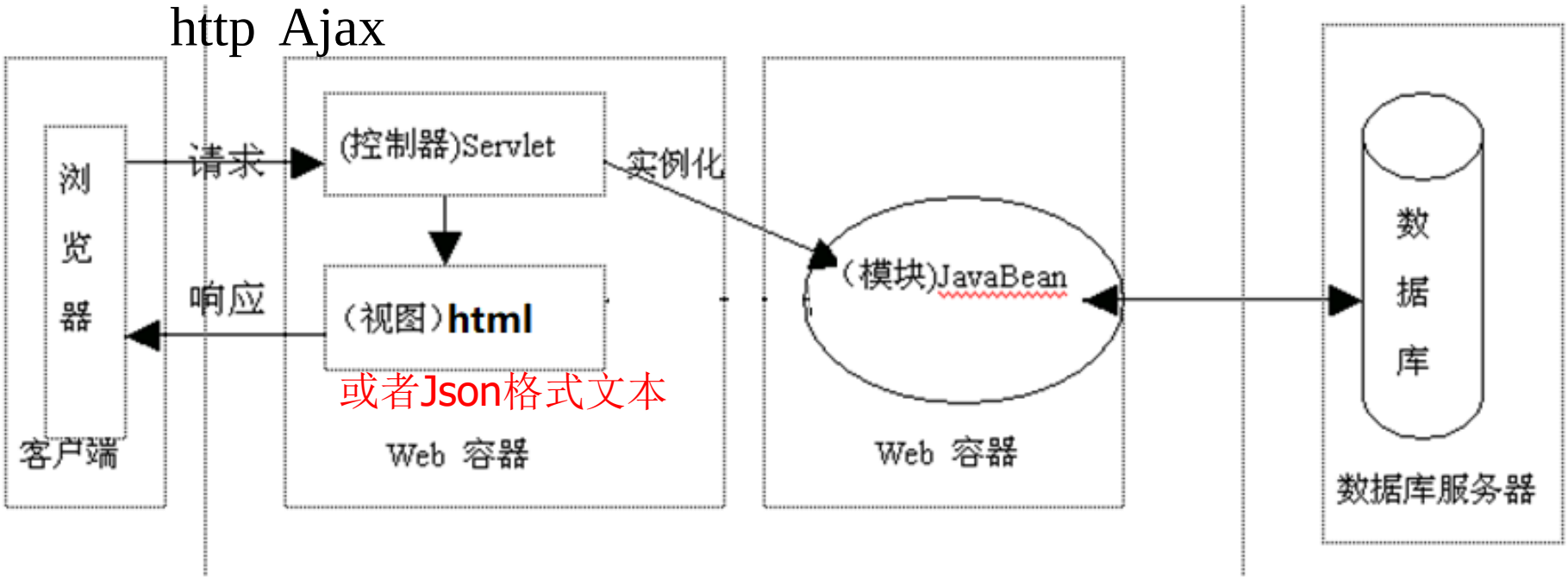
- Jsp Model 2 模型



4.2 面向对象的软件工程

4.2.6 MVC设计模式和Web应用系统

- 修改后的 Jsp Model 2 模型



4.2.7 MVC设计模式例：用户登录用例

需求分析(用户提供)：

1. 用户登录用例界面：

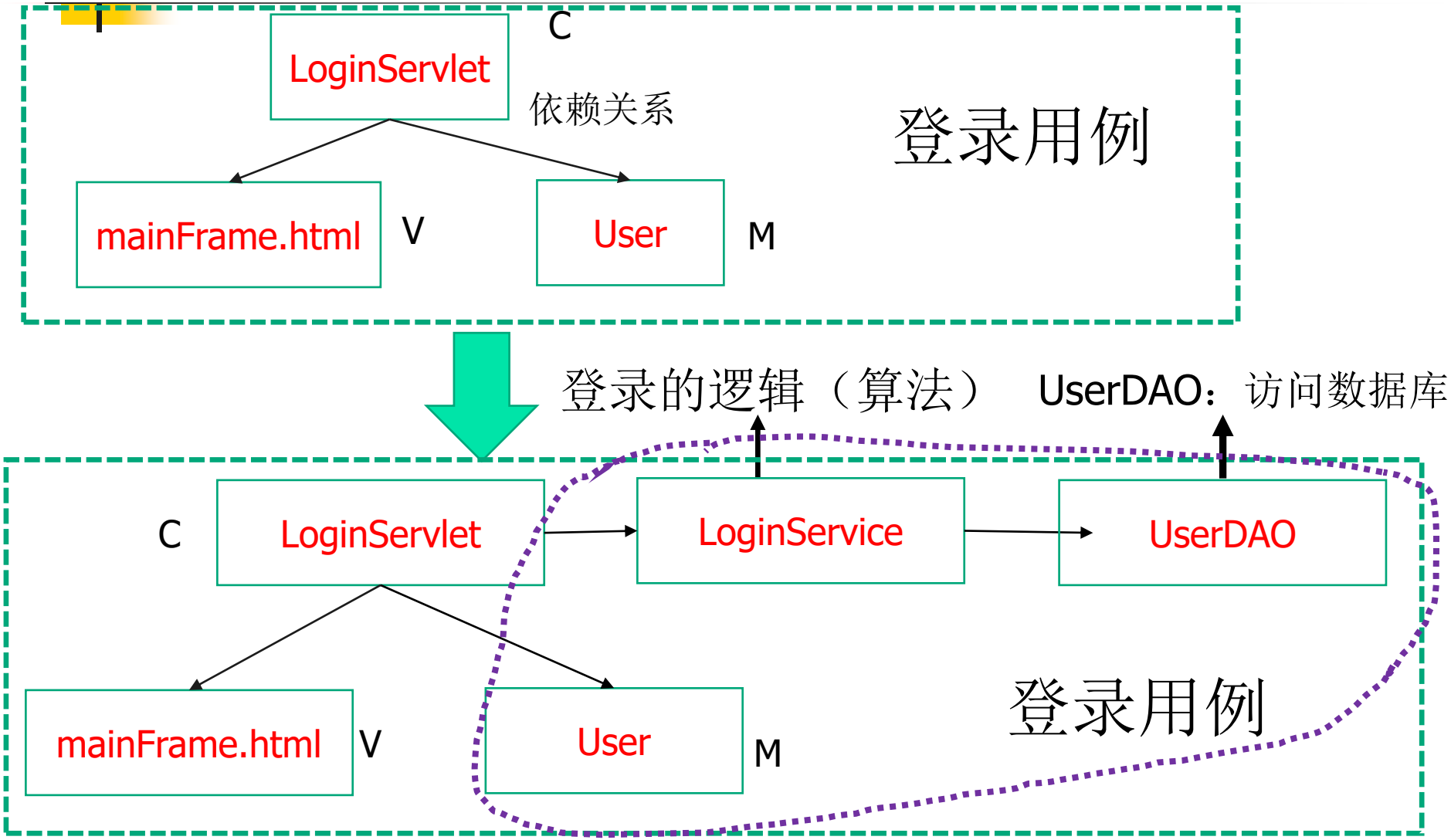
2. 用户登录用例描述：

(1) 用户在登录界面输入用户名、密码.

(2) 如果登录失败，则显示：您填写的用户名或密码错误。

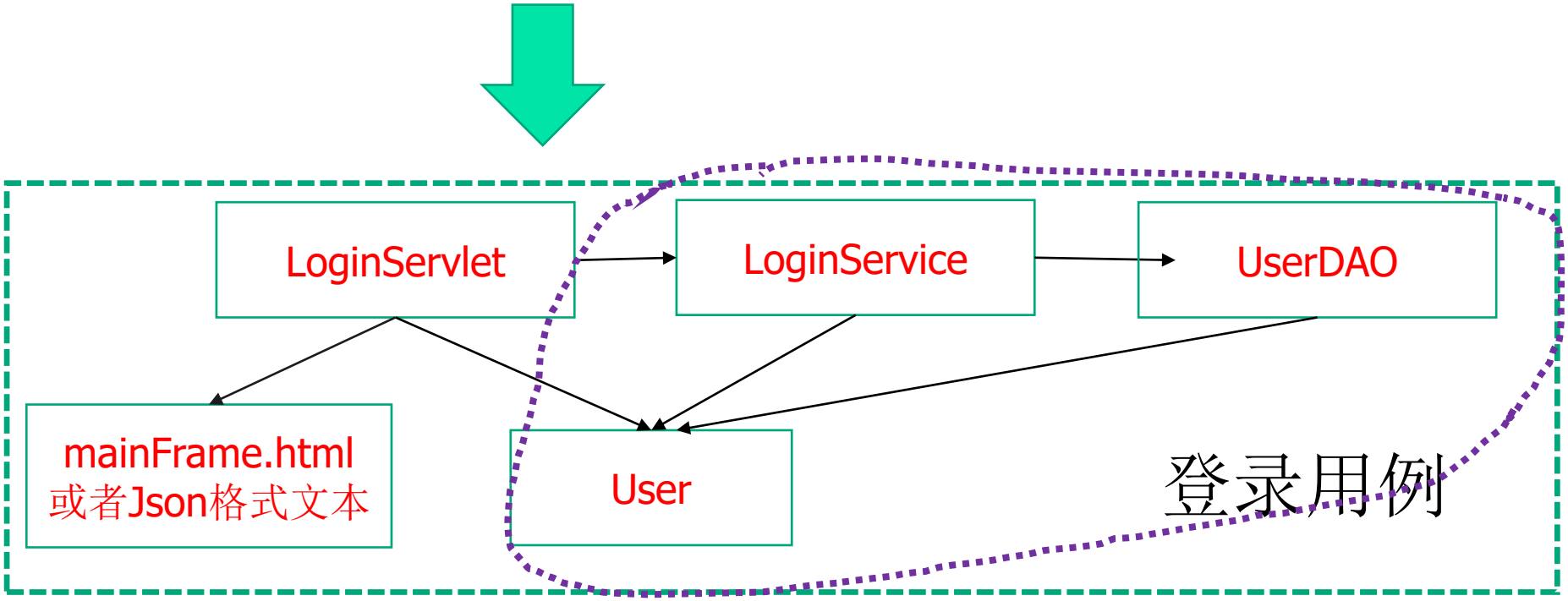
(3)如果登录成功，则进入系统界面 .

用户登录用例(login)：JavaEE Web软件体系结构风格



分层设计原则： 服务类(Service), 访问数据库类(DAO)

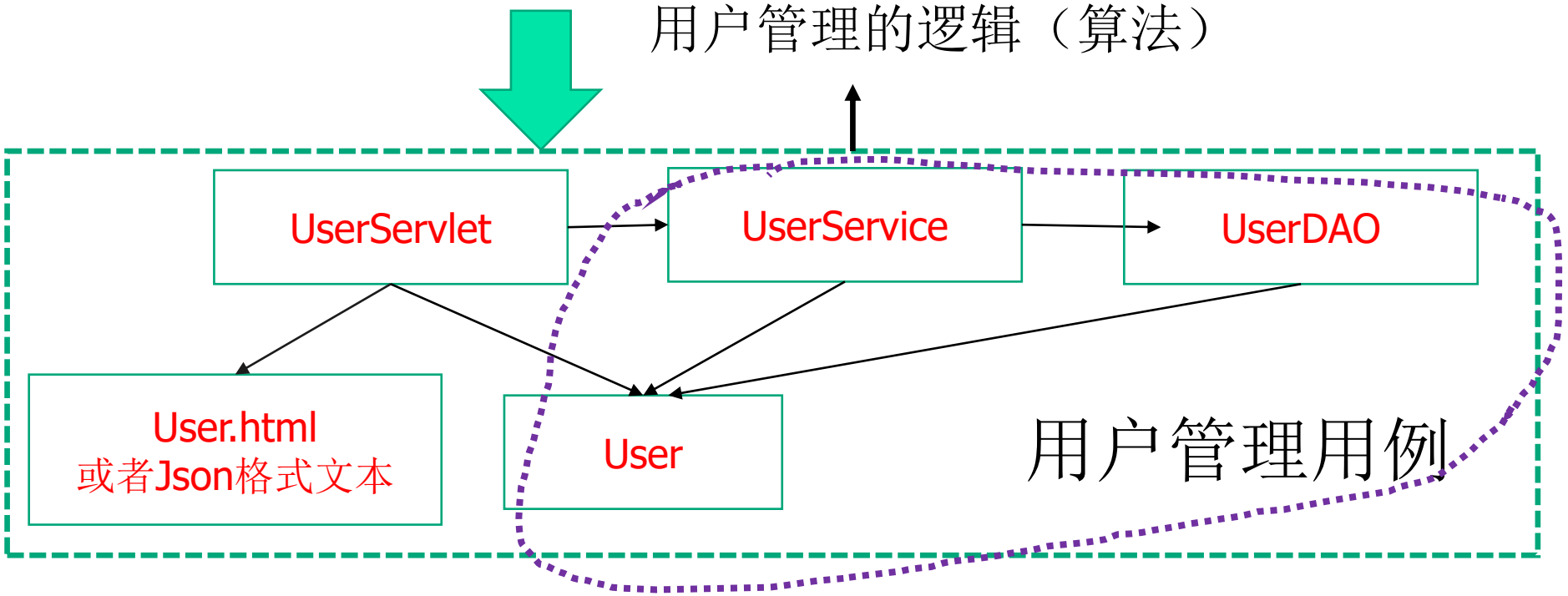
网页登录用例(login)：JavaEE Web软件体系结构风格



完善登录用例涉及的类及类间关系

4.2.7 MVC设计模式例：用户登录用例

JavaEE Web软件体系结构风格



用户管理用例涉及的类及类间关系