软件工程基础

第四章 系统设计

苏 生
susheng@uestc.edu.cn

信息与软件工程学院
电子科技大学

本章学习目标

1. 掌握软件设计的概念、内容与过程
2. 掌握体系架构设计与构件设计方法
3. 掌握程序流程图与顺序图

第四章 系统设计

4.1 系统设计概念

4.2 软件体系架构设计

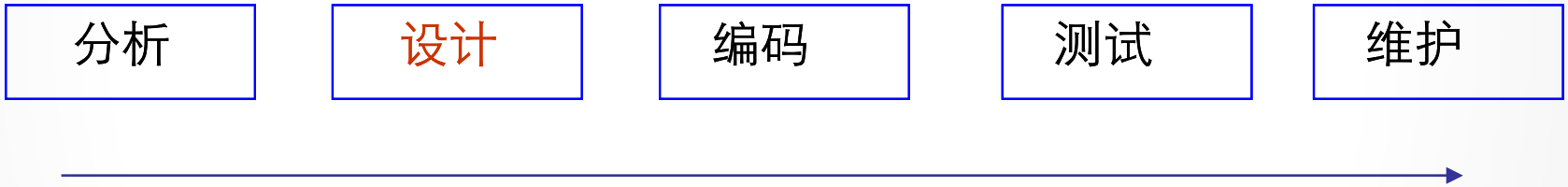
4.3 构件级设计

4.4 程序流程图

4.5 用户界面设计

4.1 软件设计概念

- 在[IEEE610.12-90]中，软件设计定义为软件系统或组件的架构、构件、接口和其他特性的定义过程及该过程的结果

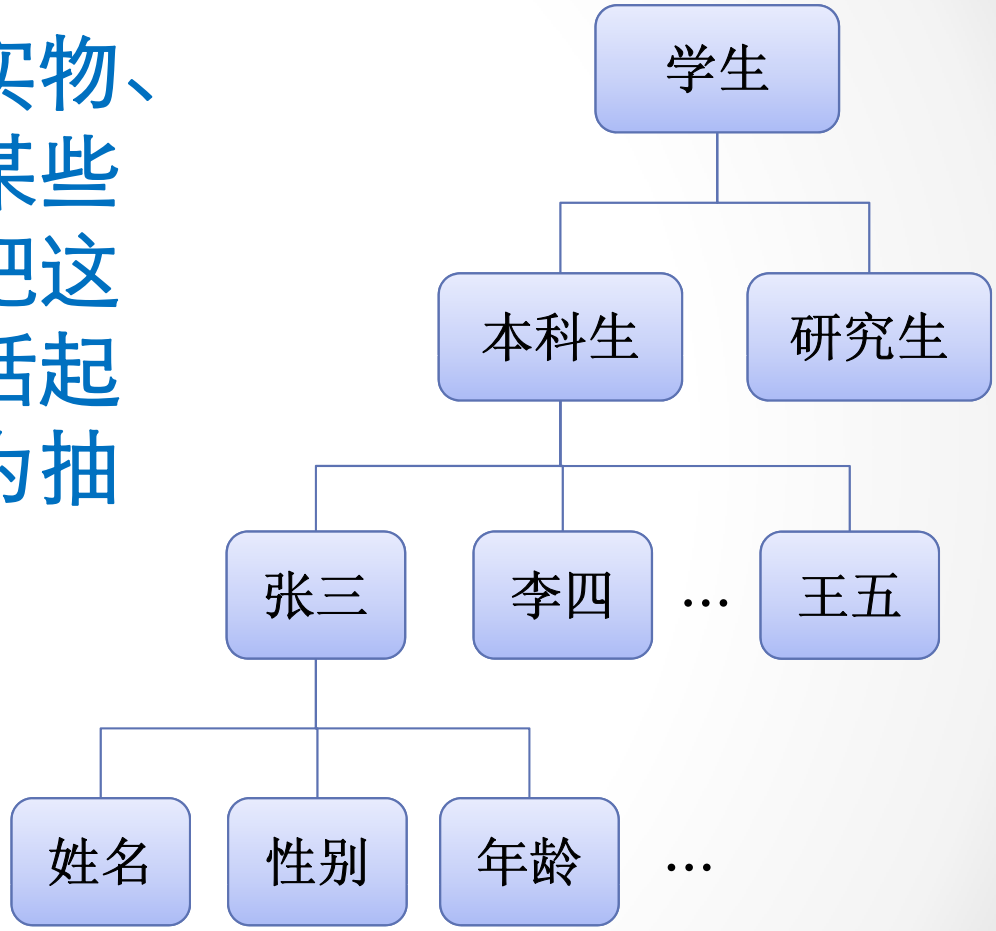


软件设计概念

- 软件设计概念
  - 抽象
  - 体系架构
  - 模块化
  - 信息隐藏
  - 求精（细化）
  - 功能独立
  - 重构

软件设计概念—抽象

抽象：现实世界中一定实物、状态或过程之间总存在某些相似的方面（共性），把这些相似的方面集中和概括起来，忽略其差异性，称为抽象。

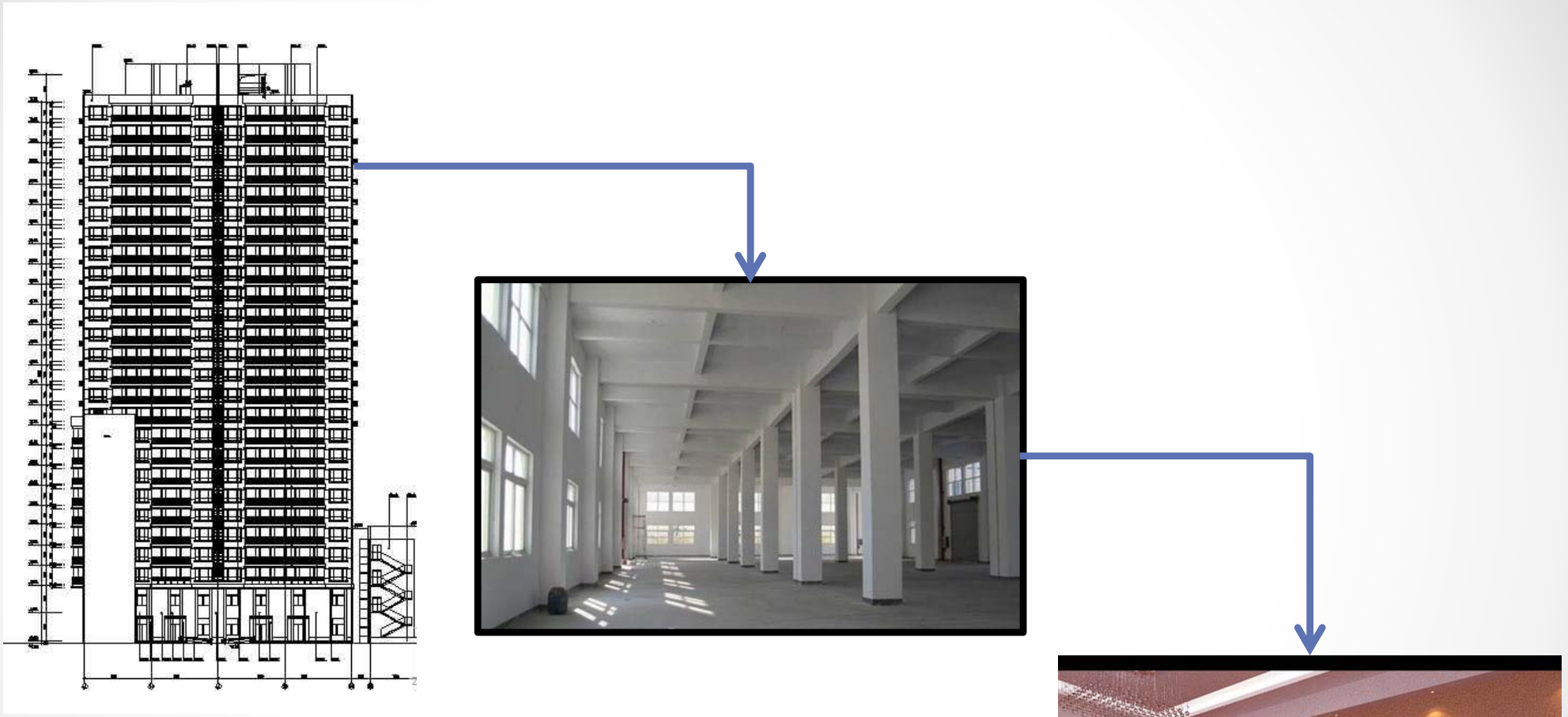


数据抽象

软件设计概念—体系架构

- 含义：软件的整体结构和这种结构为系统提供概念完整性的方式
- 体系架构内容
  - 结构特性－构件、构件被封装方式、构件间交互
  - 外部功能特性－如何满足性能需求、可靠性需求…
  - 模式性－抽取相似系统的重复性模式

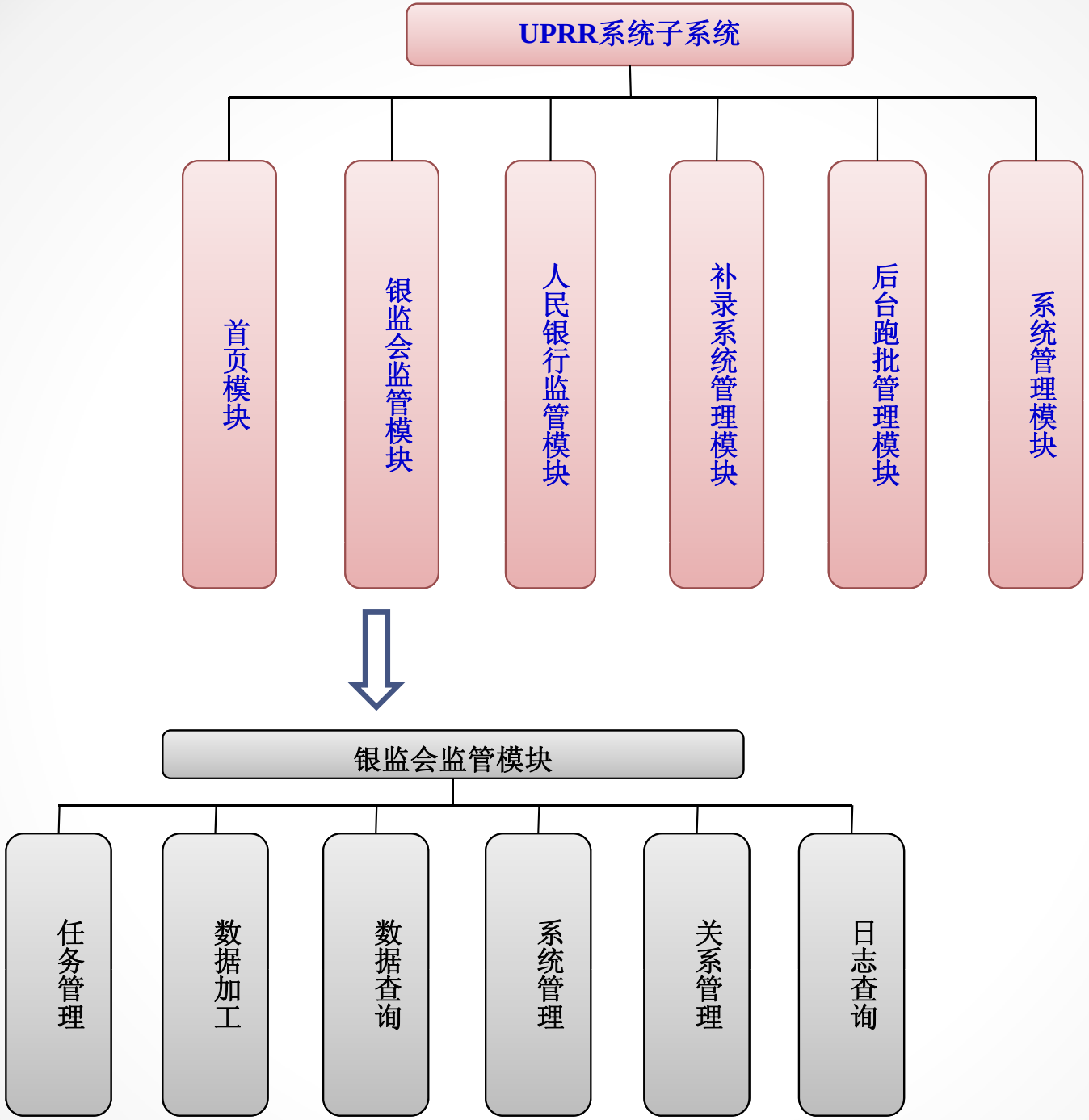
软件设计概念—体系架构



建筑体系架构

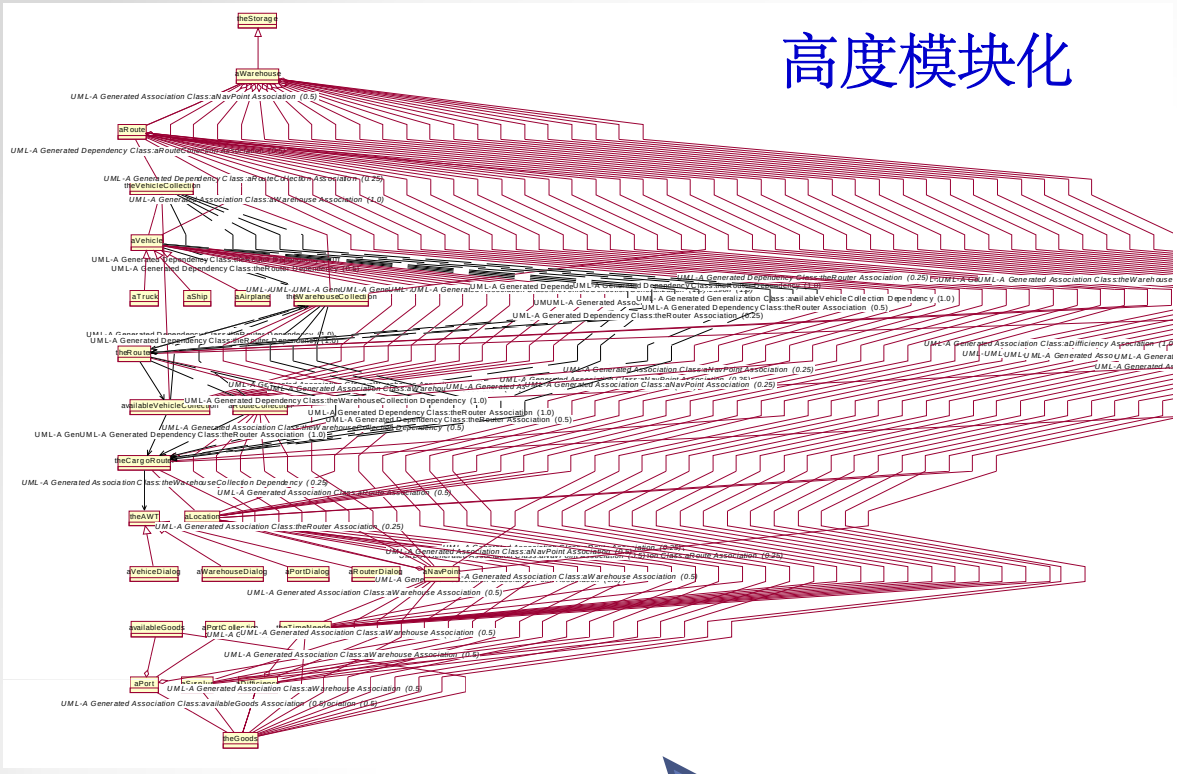
软件设计概念—模块化

- 含义：软件被划分为命名和功能相对独立的多个组件（通常称为模块），通过这些组件的集成来满足问题的需求
- 软件的模块性：程序可被智能管理的单一属性

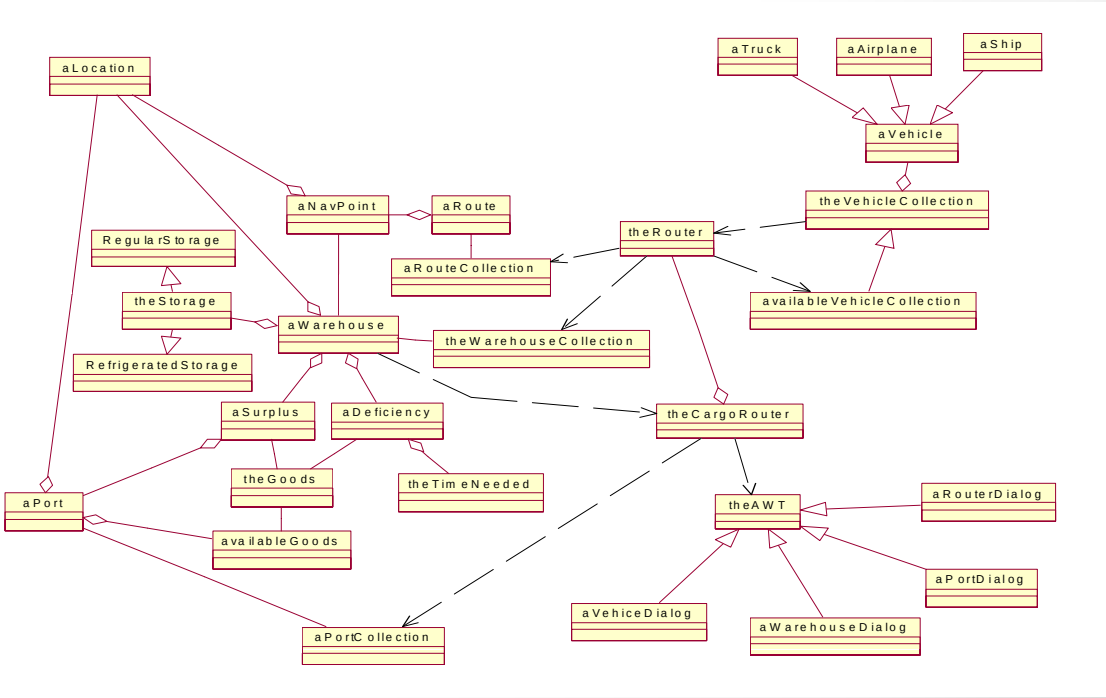

UPRR系统子系统
首页模块
银监会监管模块
人民银行监管模块
补录系统管理模块
后台跑批管理模块
系统管理模块
银监会监管模块
任务管理
数据加工
数据查询
系统管理
关系管理
日志查询

模块化

高度模块化



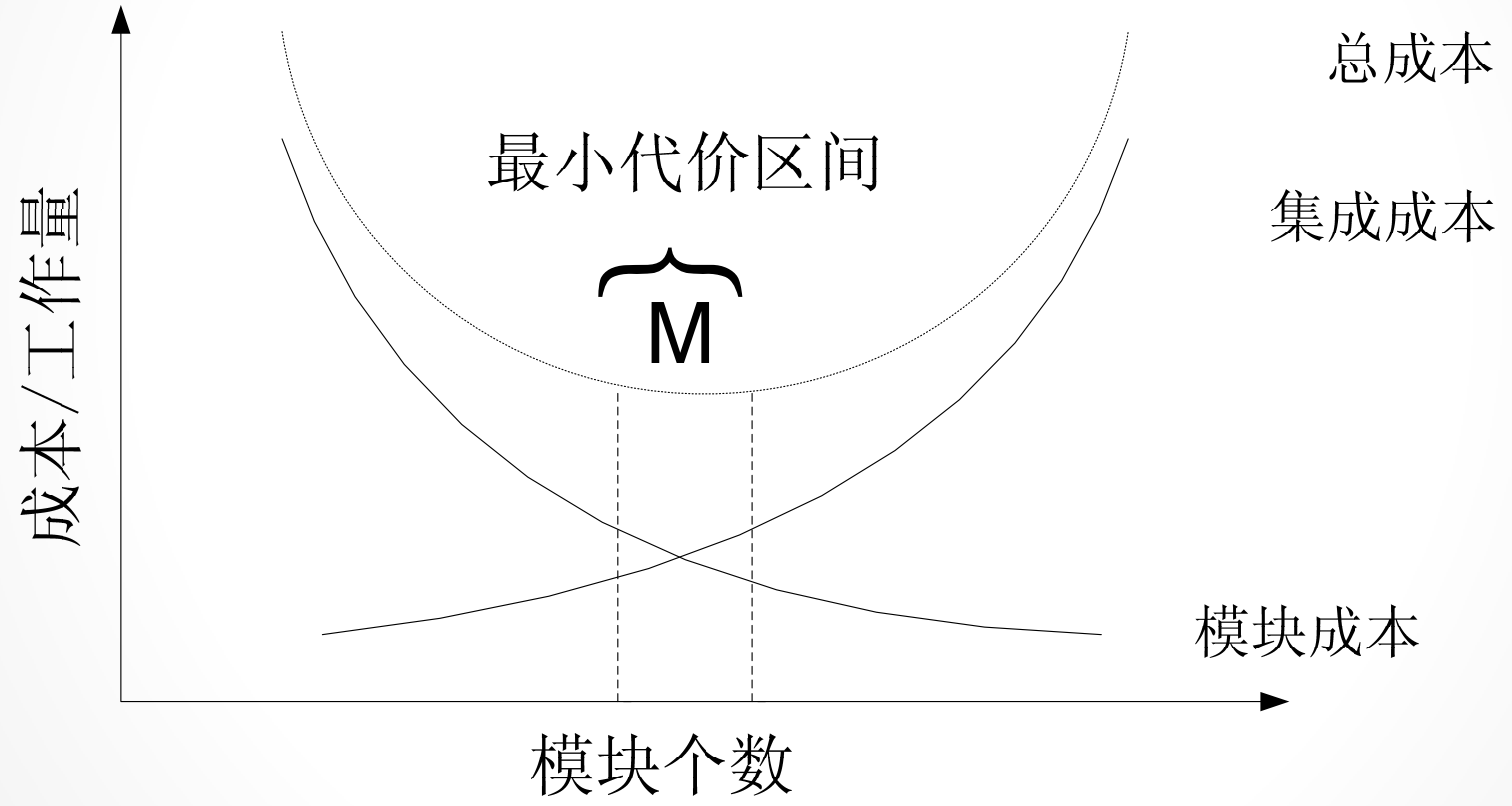
一般模块化



模块越多越好?

软件设计概念—模块化

- 模块化和软件成本



软件设计概念—信息隐藏

- 含义：模块应该具有彼此相互隐藏的特性，模块定义和设计时应当保证模块内的信息（过程和数据）不可以被不需要这些信息的其他模块访问
- *抽象*有助于定义构成软件的过程（或信息）实体。
- *信息隐藏*原则定义和隐藏了模块内的过程细节和模块内的本地数据结构。
- 举例：私有变量与方法

软件设计概念—求精

- 含义：逐步求精的过程
- *抽象*使设计师确定过程和数据，对外部隐藏底层细节
- *求精*有助于设计者在设计过程中揭示底层细节

软件设计概念—求精

用筛选法求100以内的素数。所谓的筛选法，就是从2到100中去掉2，3，5，7的倍数，剩下的就是100以内的素数

首先写出一个程序框架：

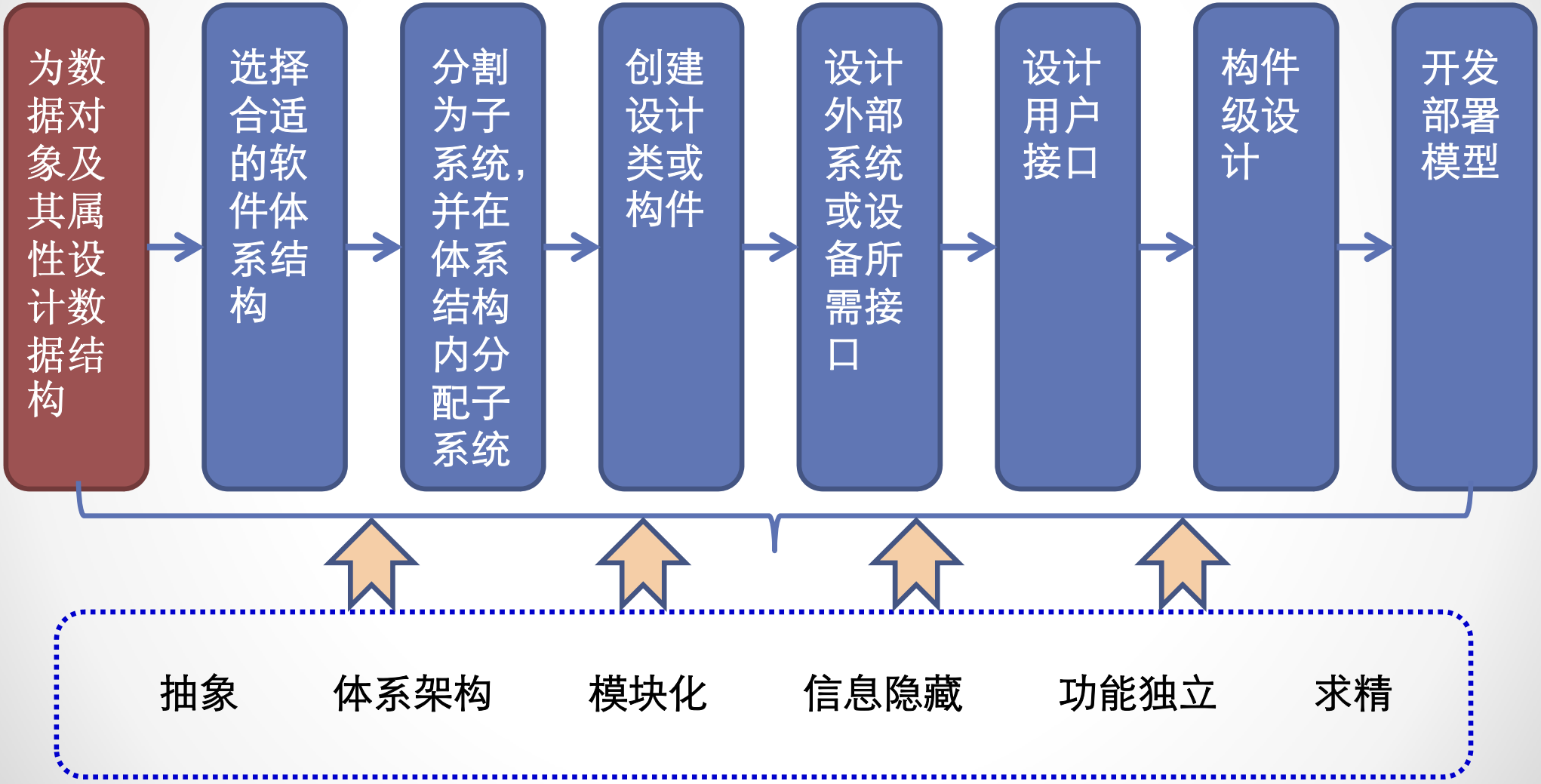
```
main()
{
    建立2到100的数据A[]，其中A[i]=i; ……..……1
    建立2到10的素数表B[]，存放2到10以内的素数; ……2
    若A[i]=i是B[]中任一数的倍数，则剔出A[i]; ……..……3
    输出A[]中所有没有被剔出的数; ……4
}
```

软件设计概念—求精

上述程序框架中的每一个加工语句都可以细化

```
main()
{
    建立2到100的数据A[]，其中A[i]=i; ……..……1
    for(i=2;i<=100;i++) A[i]=i;
    建立2到10的素数表B[]，存放2到10以内的素数; ……2
    B[1]=2; B[1]=3; B[1]=5; B[1]=7;
    若A[i]=i是B[]中任一数的倍数，则剔出A[i]; ……..……3
    for(j=1;j<=4;j++)
    检查A[]中所有数能否被B[j]整除并将其从A[]剔除; …3.1
    输出A[]中所有没有被剔出的数; ……..……4
    for(i=2;i<=100;i++)
    若A[i]没有被剔除，则输出之;……..……4.1
}
```

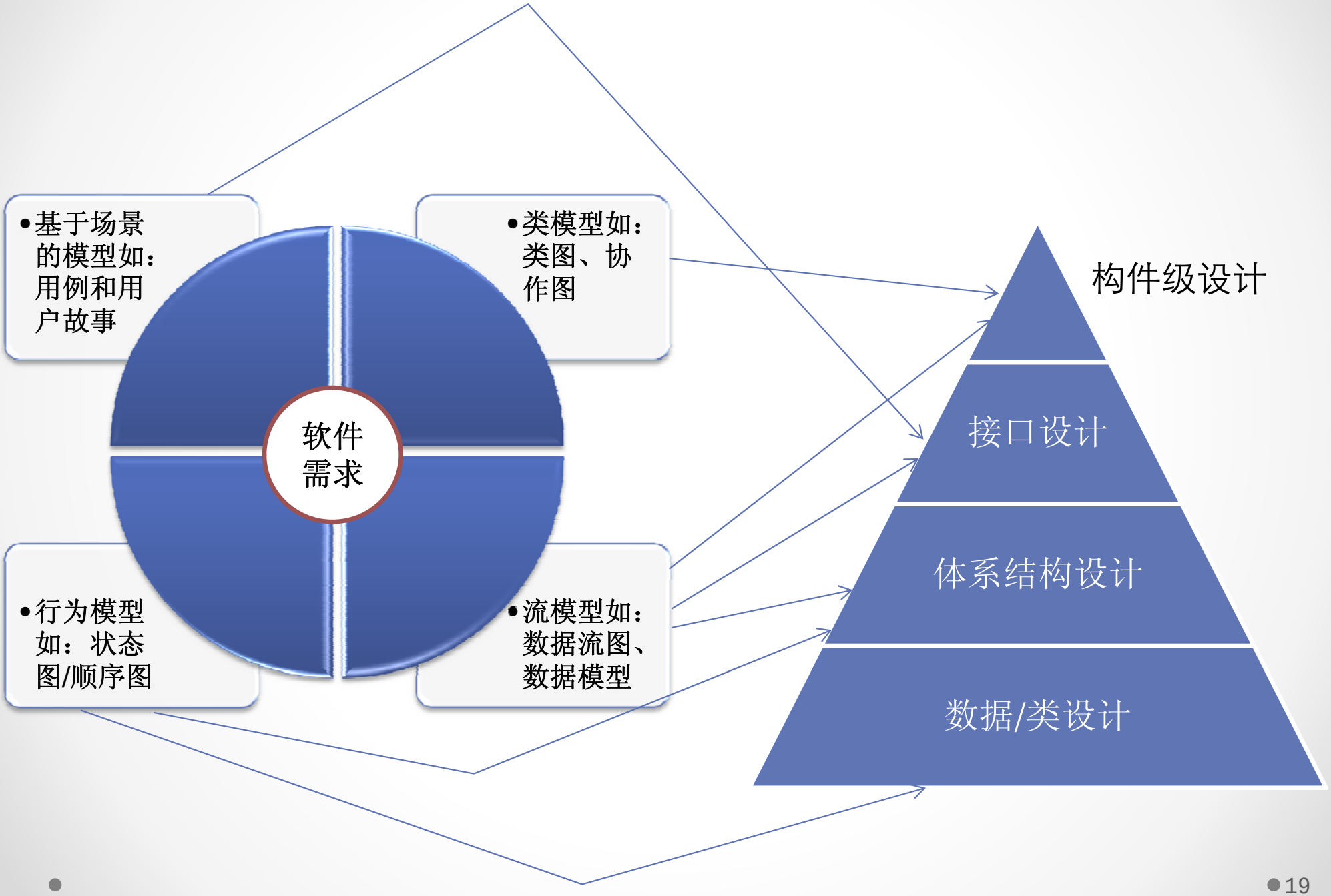
软件设计过程


为数据对象及其属性设计数据结构
选择合适的软件体系结构
分割为子系统，并在体系结构内分配子系统
创建设计类或构件
设计外部系统或设备所需接口
设计用户接口
构件级设计
开发部署模型
抽象
体系架构
模块化
信息隐藏
功能独立
求精

软件设计内容

- 设计模型
  - 数据设计
  - 体系结构设计
  - 接口设计
  - 构件级设计
  - 部署级设计

需求模型到设计模型的转换



软件设计质量

- 质量属性 **HP-FURPS**
  - 功能性**functionality**
  - 易用性 **usability**
  - 可靠性 **reliability**
  - 性能 **performance**
  - 可支持性 **supportability**

第四章 系统设计

4.1 系统设计概念

4.2 软件体系架构设计

4.3 构件级设计

4.4 程序流程图

4.5 用户界面设计

4.2 软件体系架构设计

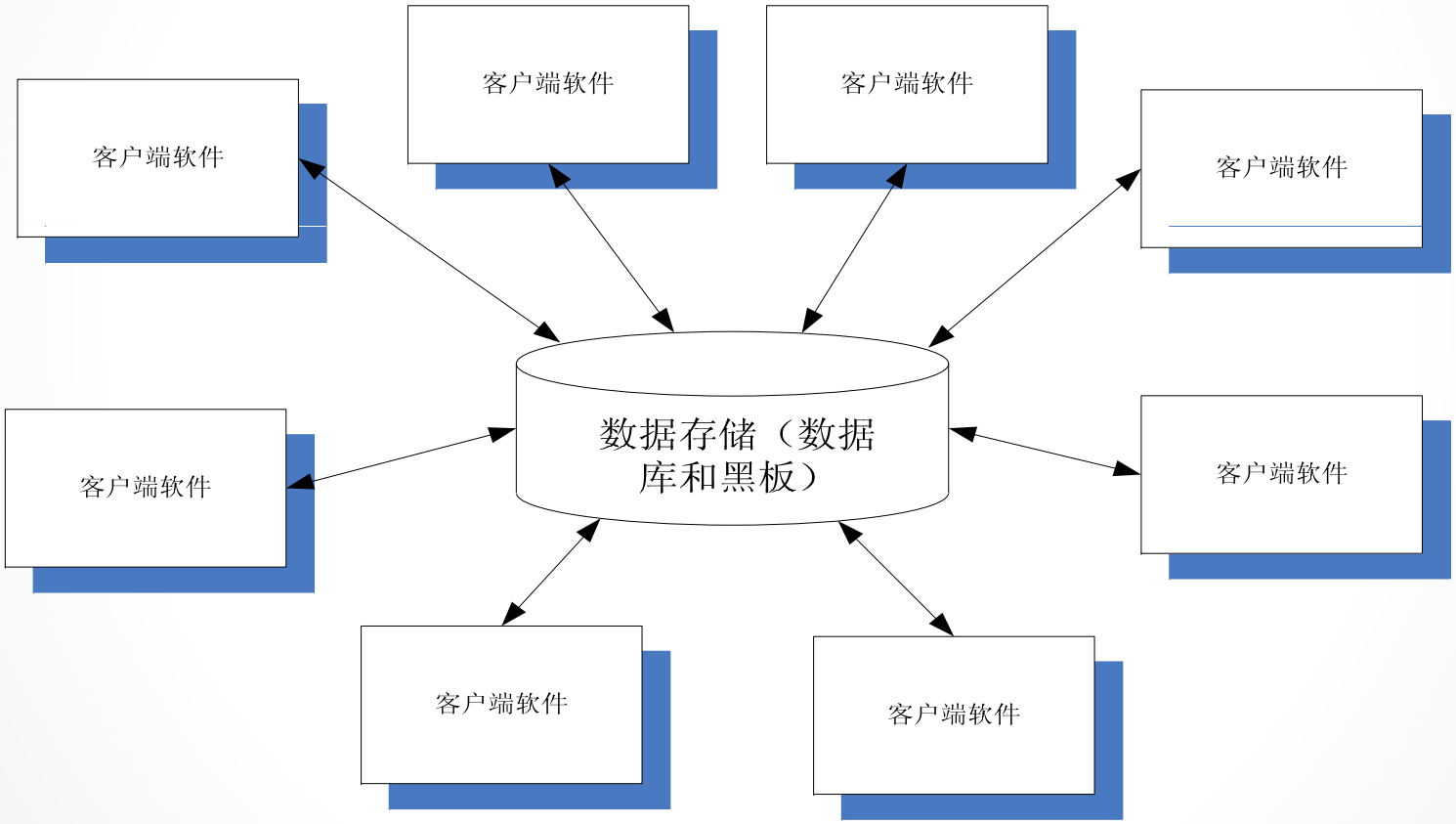
- 什么是体系架构
  - 系统的一个或多个结构，包括软件构件、构件的外部可见属性以及它们之间的相互关系
  - 一种表达，使能够：对设计在满足既定需求方面的有效性进行分析；在设计变更相对容易的阶段，考虑体系结构可能的选择方案；降低与软件构造相关的风险

软件体系架构设计

- 体系架构设计的目的
  - 标识关键构建/子系统
  - 描述关系/交互/构建接口
  - 指导软件设计，并桥接软件设计和实现
  - 标识关键技术/实现工具

软件体系架构设计

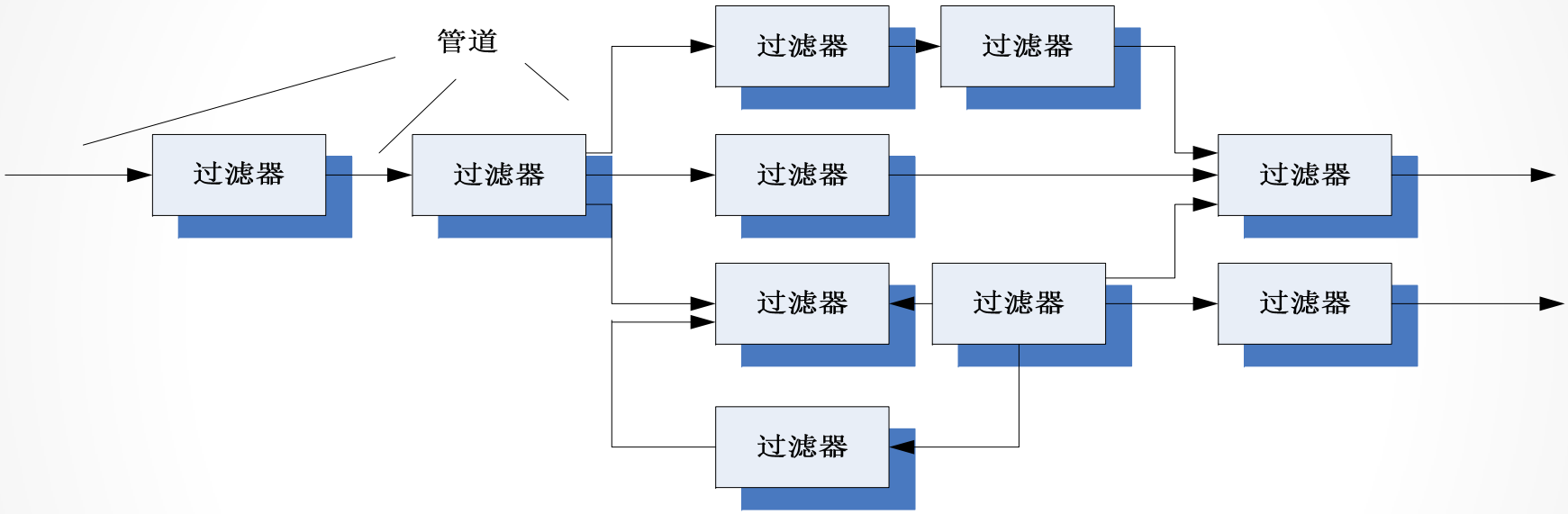
- 体系架构风格



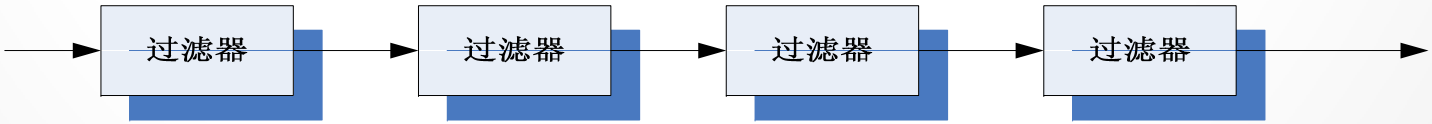
以数据为中心的体系结构

软件体系架构设计

- 体系架构风格



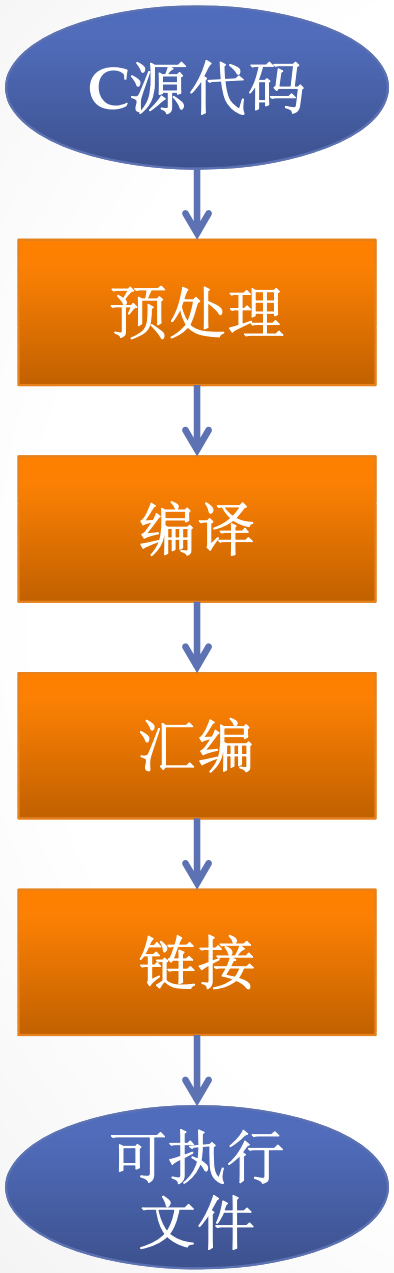
（a）管道和过滤器



(b) 批处理序列

数据流体系结构

管道过滤器架构例子---C程序编译器



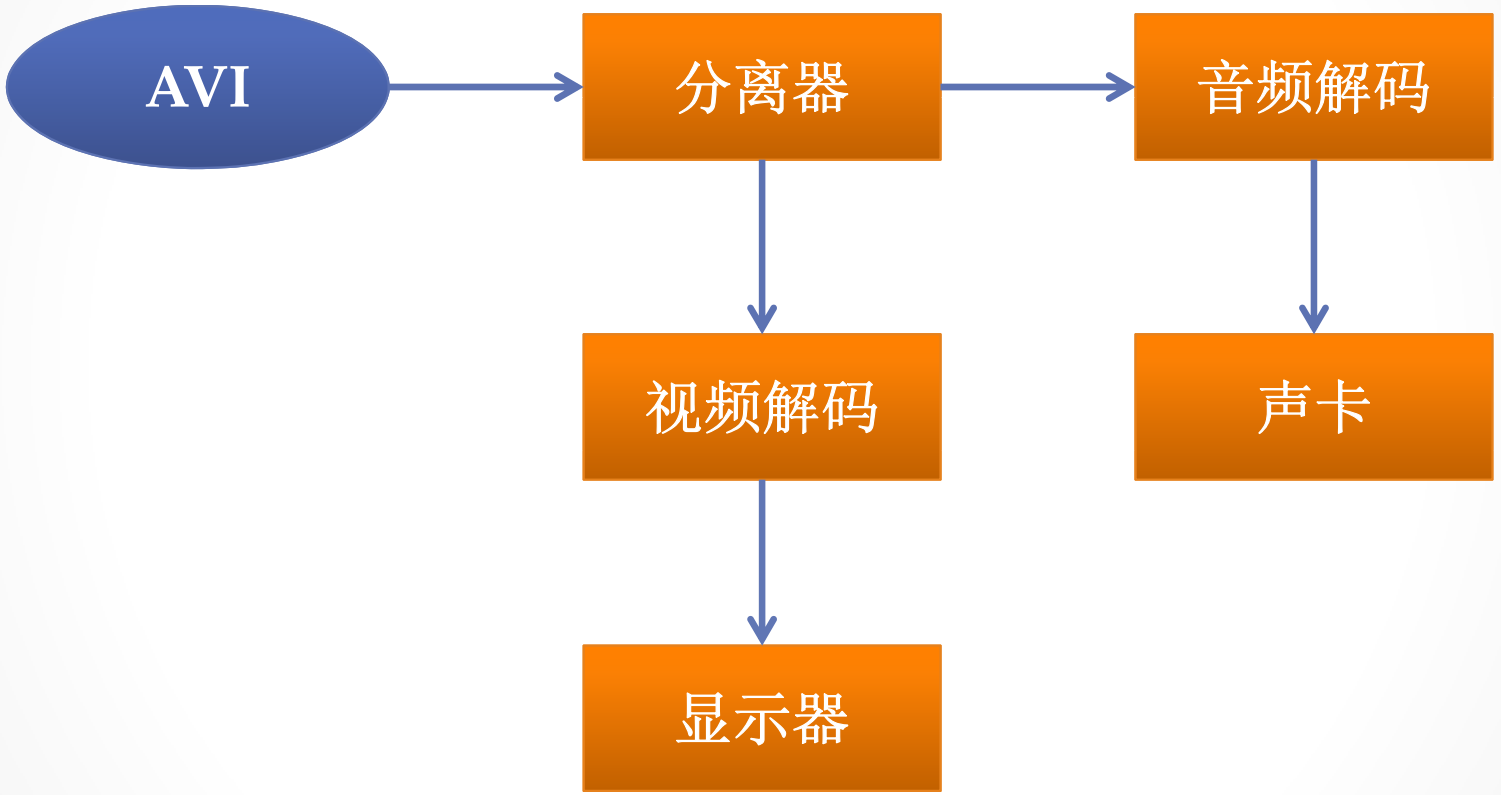
宏展开和去掉注释等工作

进行词法分析、语法分析、语义分析、代码优化和代码产生

把汇编代码转换成机器指令，生成目标文件

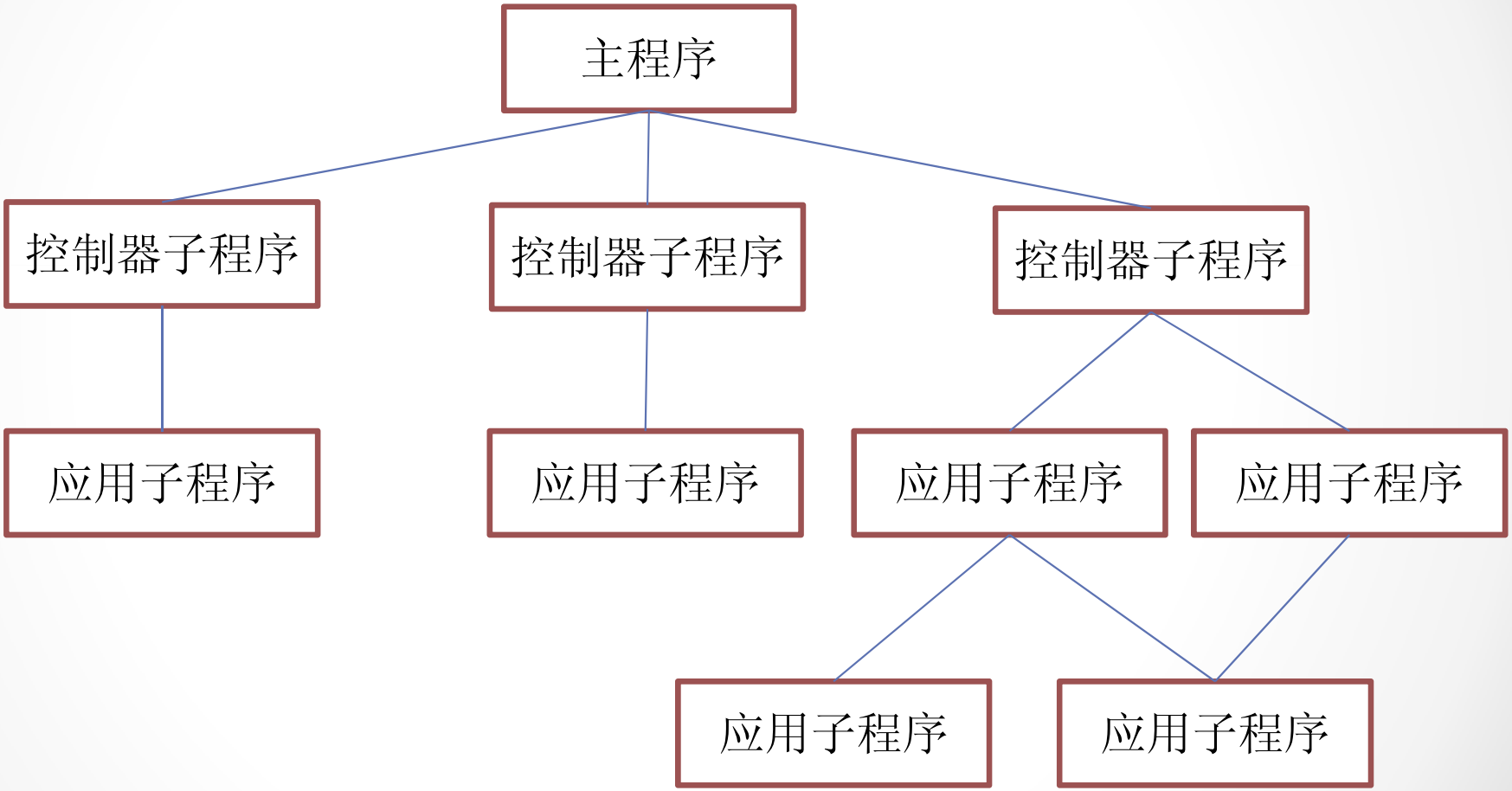
负责把多个目标文件、静态库和共享库链接成可执行文件/共享库

管道过滤器架构例子---媒体播放器



软件体系架构设计

- 体系架构风格



主程序/子程序体系结构

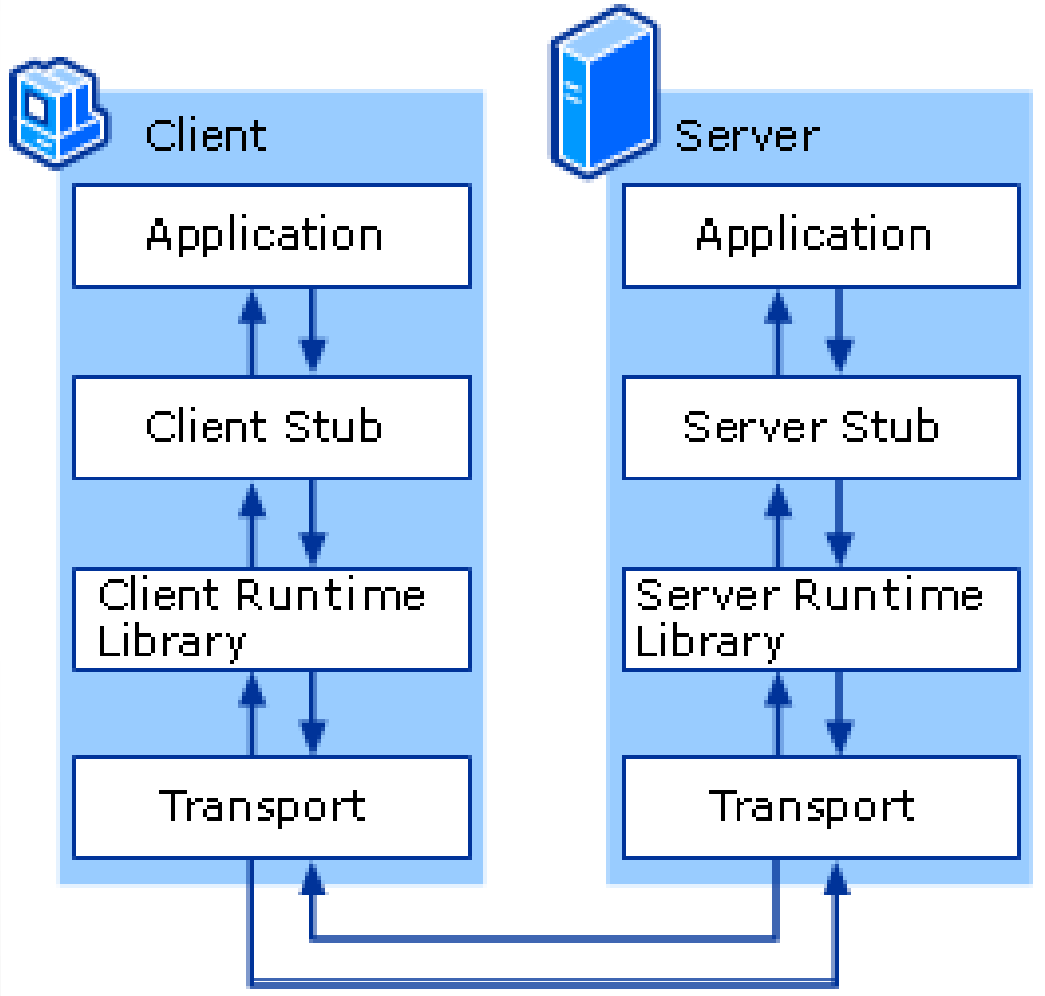
主程序/子程序体系结构例子

1. 类A的方法M调用类A的方法N
2. 类A的方法M调用类B的方法N
3. Ajax回调函数

```
$.ajax({
    type: “post”,
    url: “../../...”, /*请求执行的地址（即：调用的后台方法）*/
    cache: true,
    data: “xxx”,
    dataType: “json”,
    success: function (data) {.....; /*程序语句*/},
}
```

软件体系架构设计

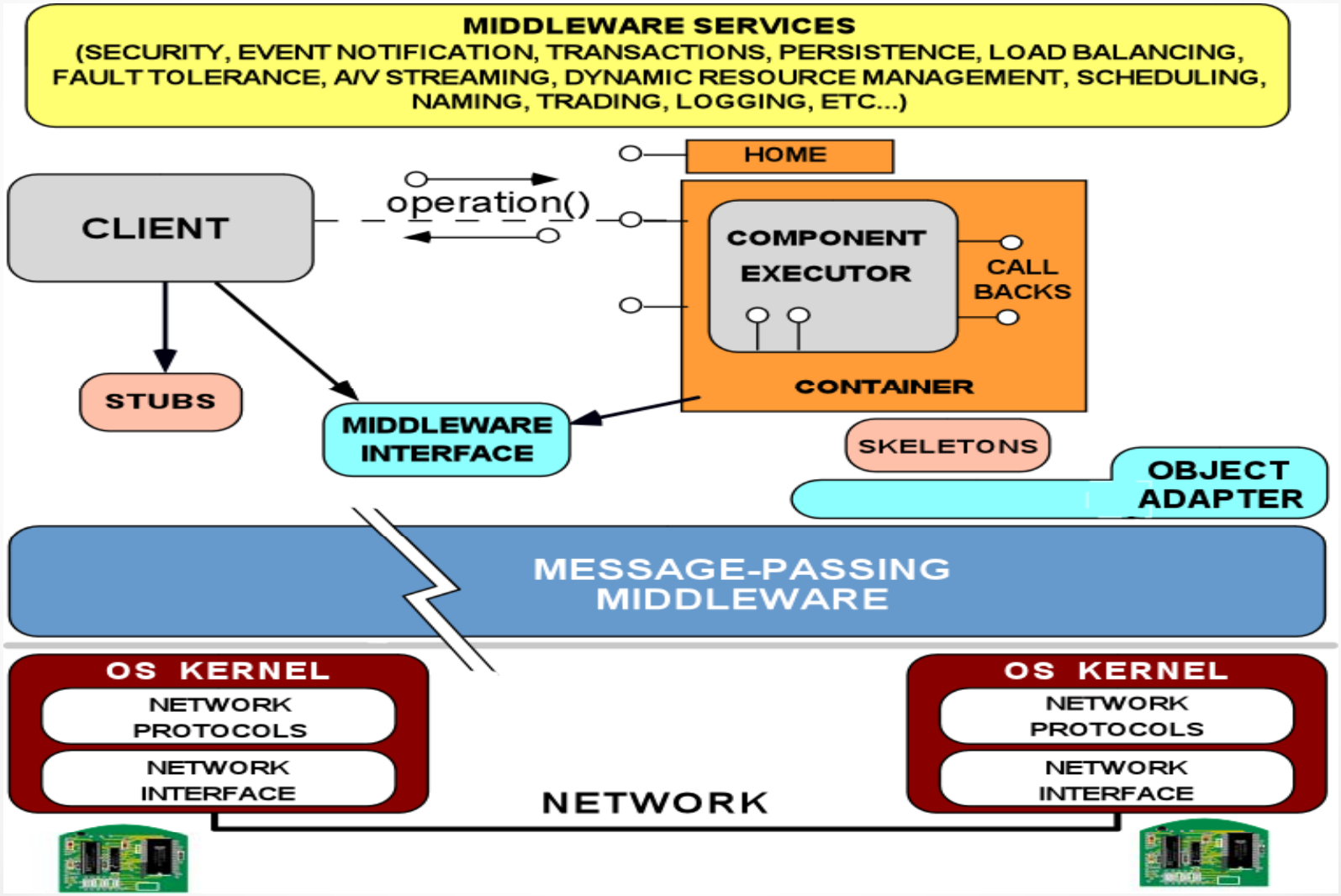
- 体系架构风格



远程过程调用体系结构

软件体系架构设计

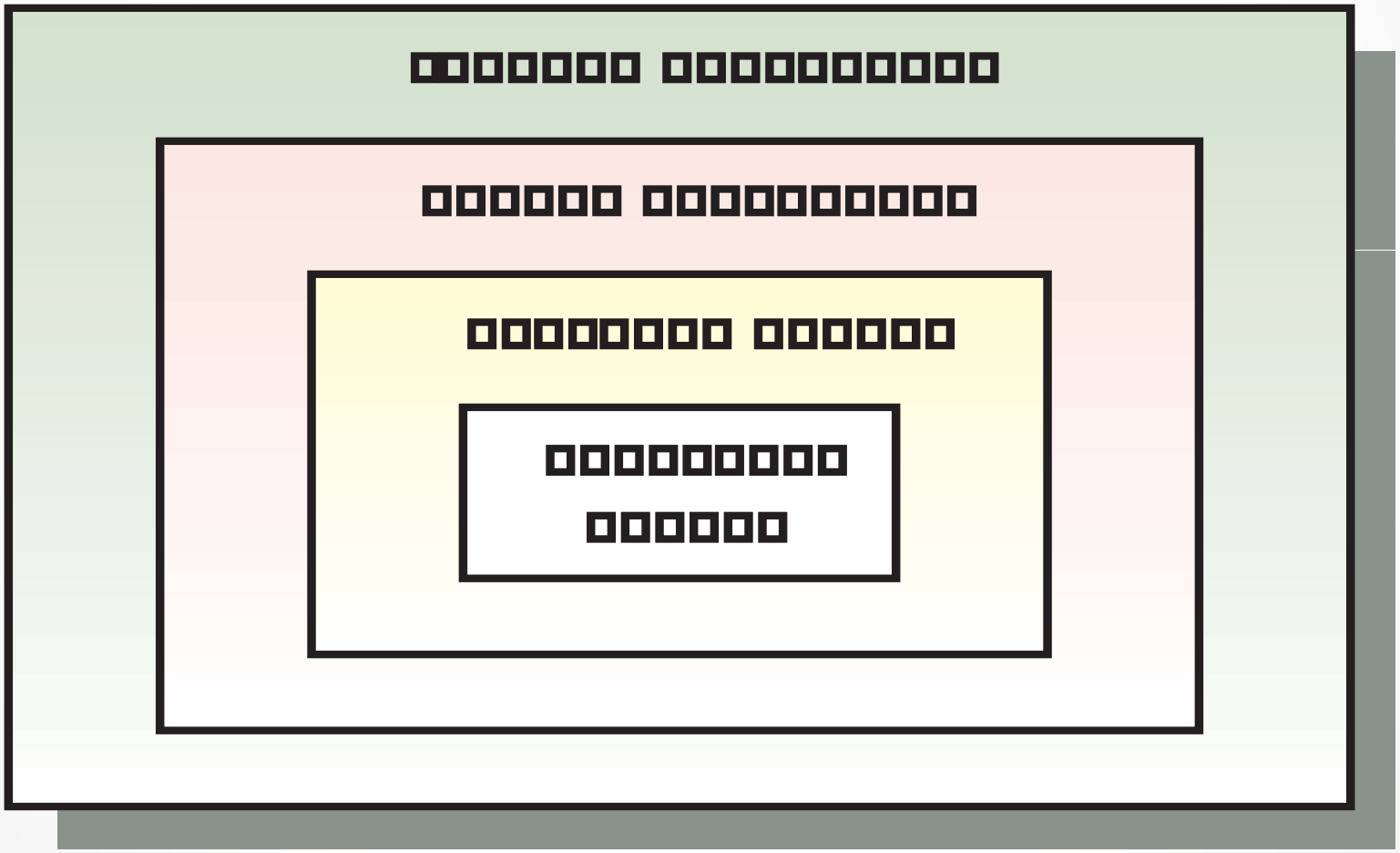
- 体系架构风格



公共对象请求代理体系结构（CORBA）

软件体系架构设计

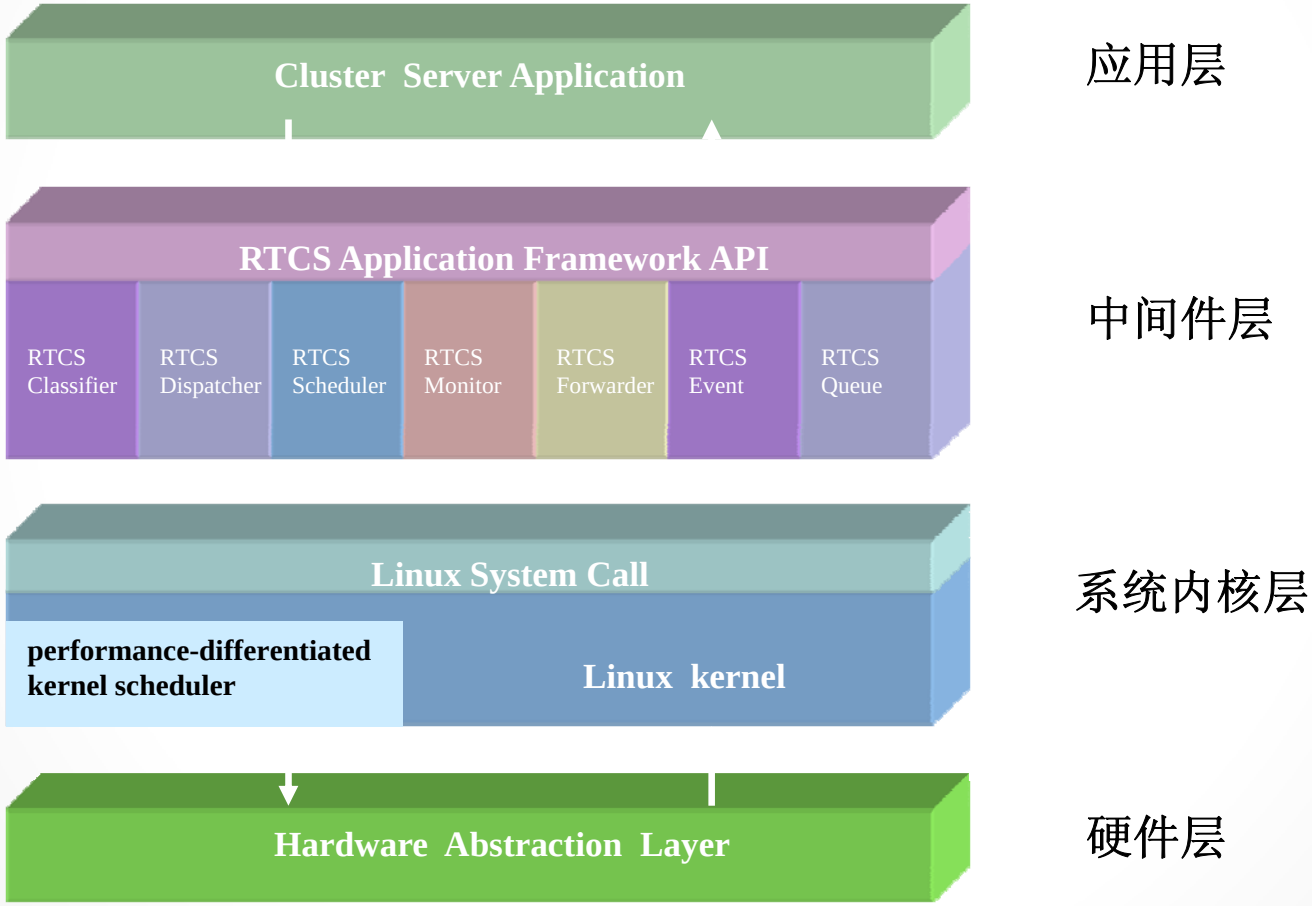
- 体系架构风格



层次体系架构

软件体系架构设计

- 体系架构风格

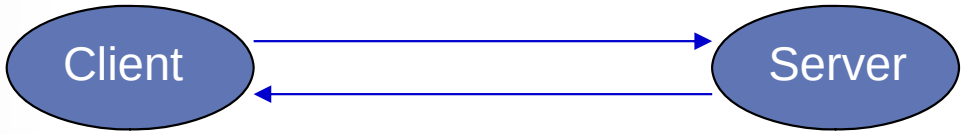


层次体系架构

软件体系架构设计

- 体系架构风格

客户端->服务器

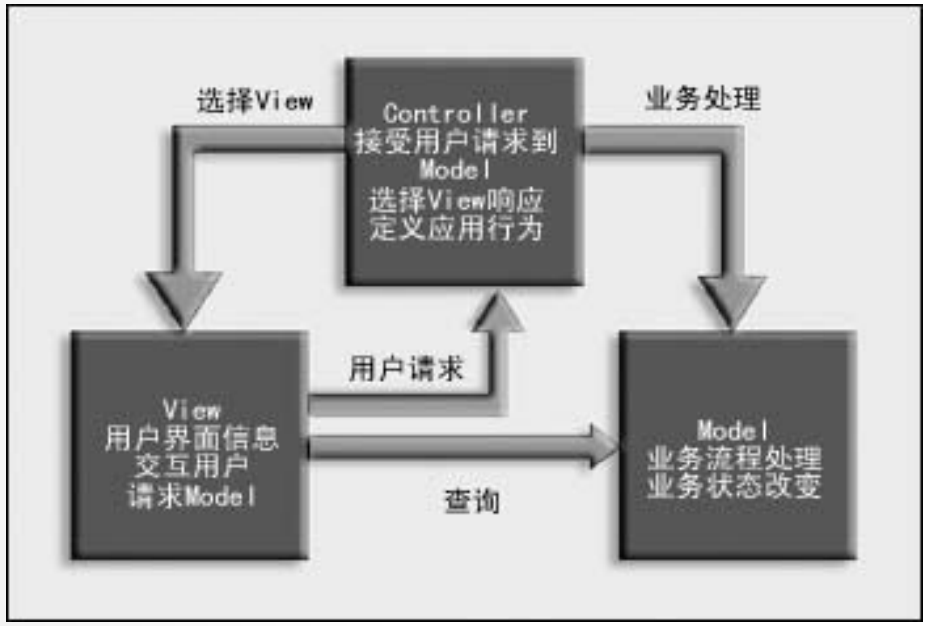


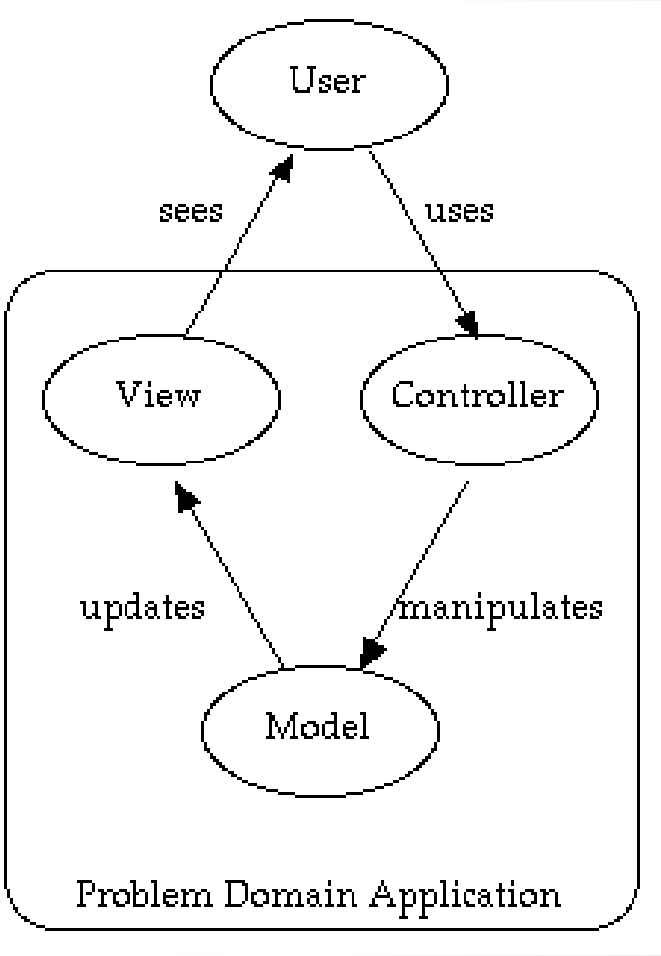
客户端->web 服务器->应用服务器

层次体系架构

软件体系架构设计

- 体系架构风格

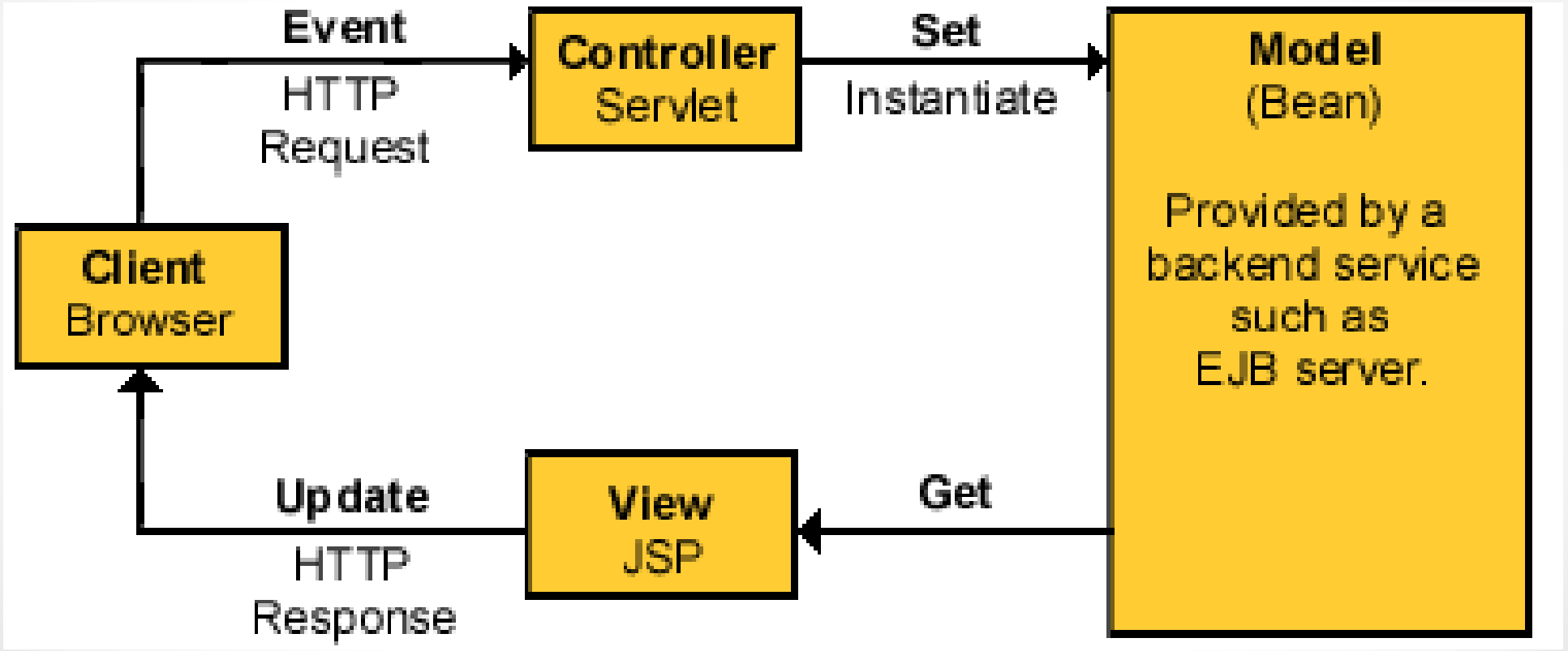




层次体系架构

软件体系架构设计

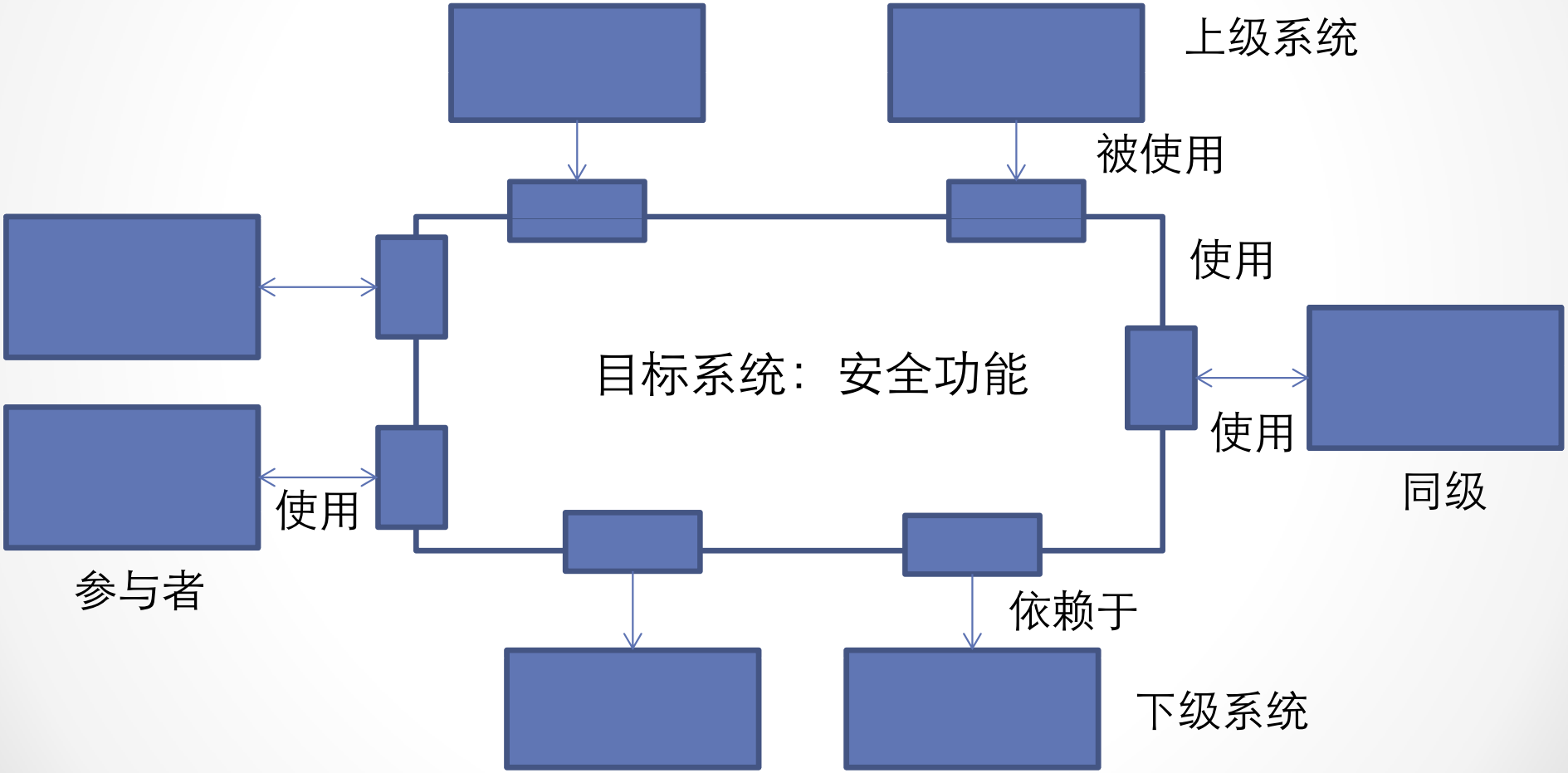
- 体系架构风格



层次体系架构

软件体系架构设计

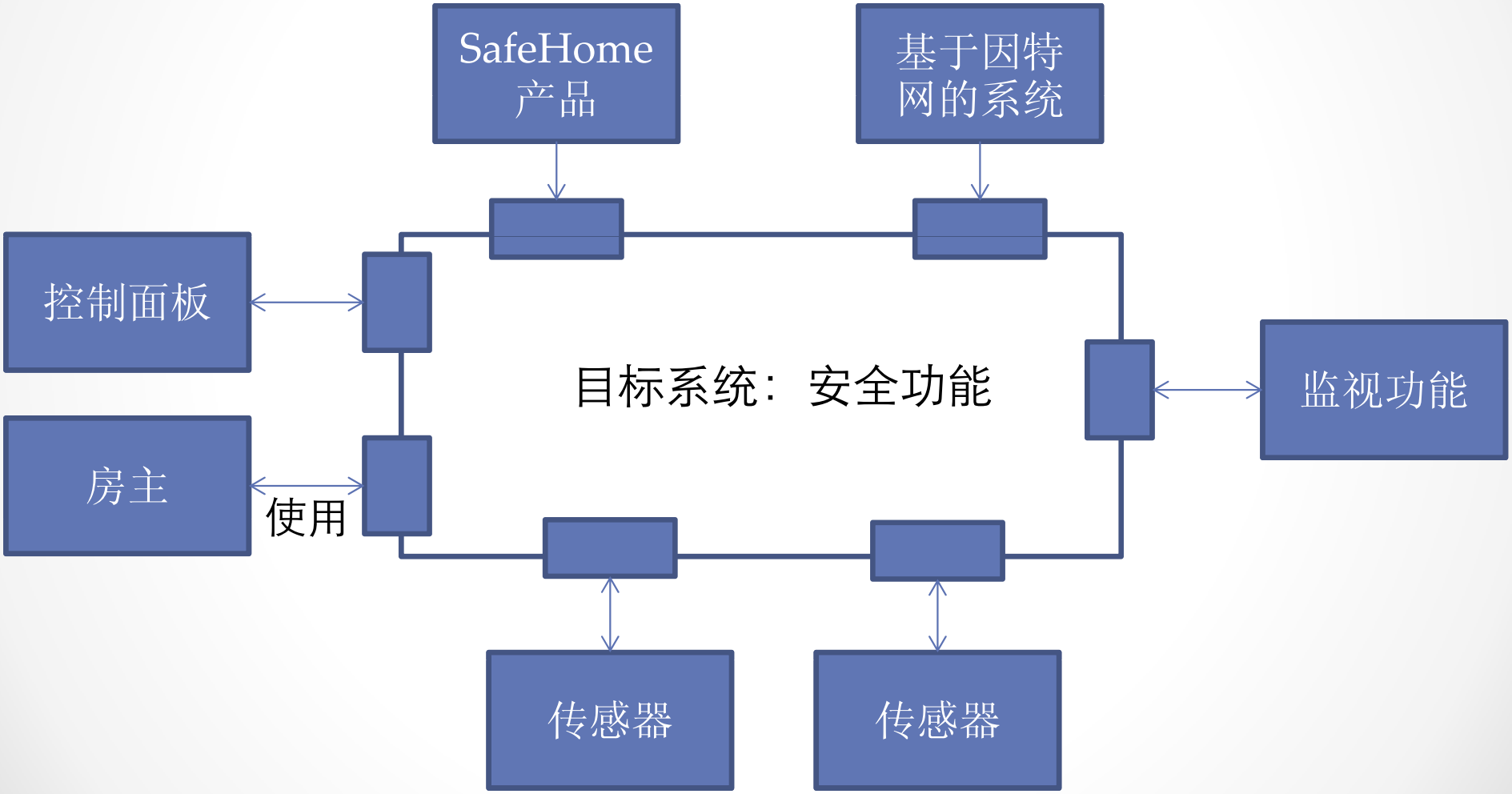
- 体系架构设计步骤



第一步 体系结构环境图

软件体系架构设计

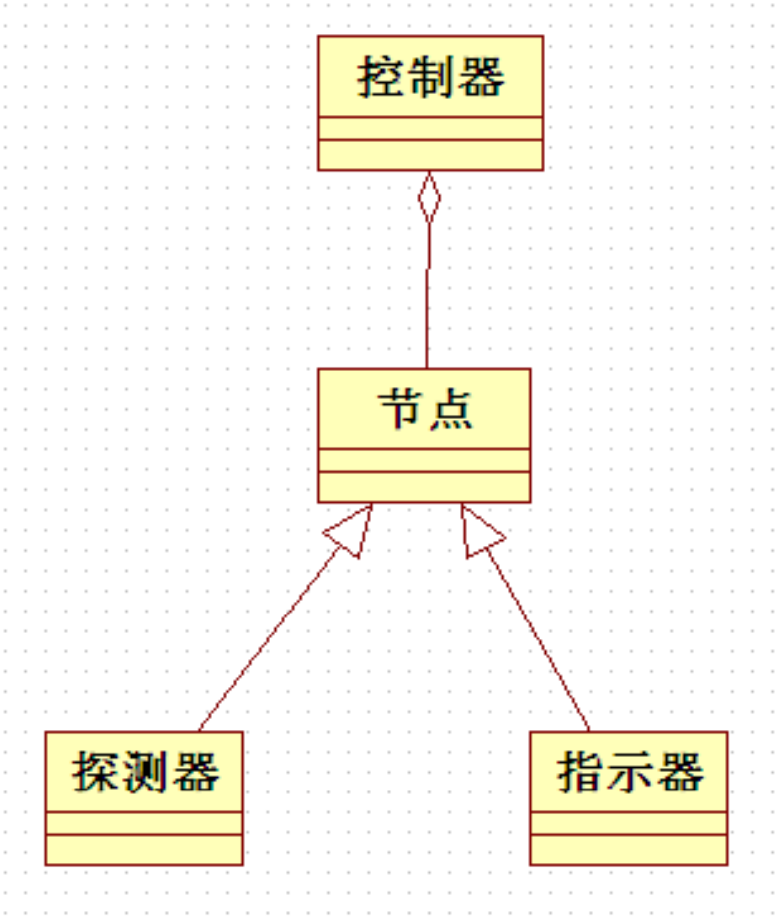
- 体系架构设计步骤



第一步 体系结构环境图

软件体系架构设计

- 体系架构设计步骤

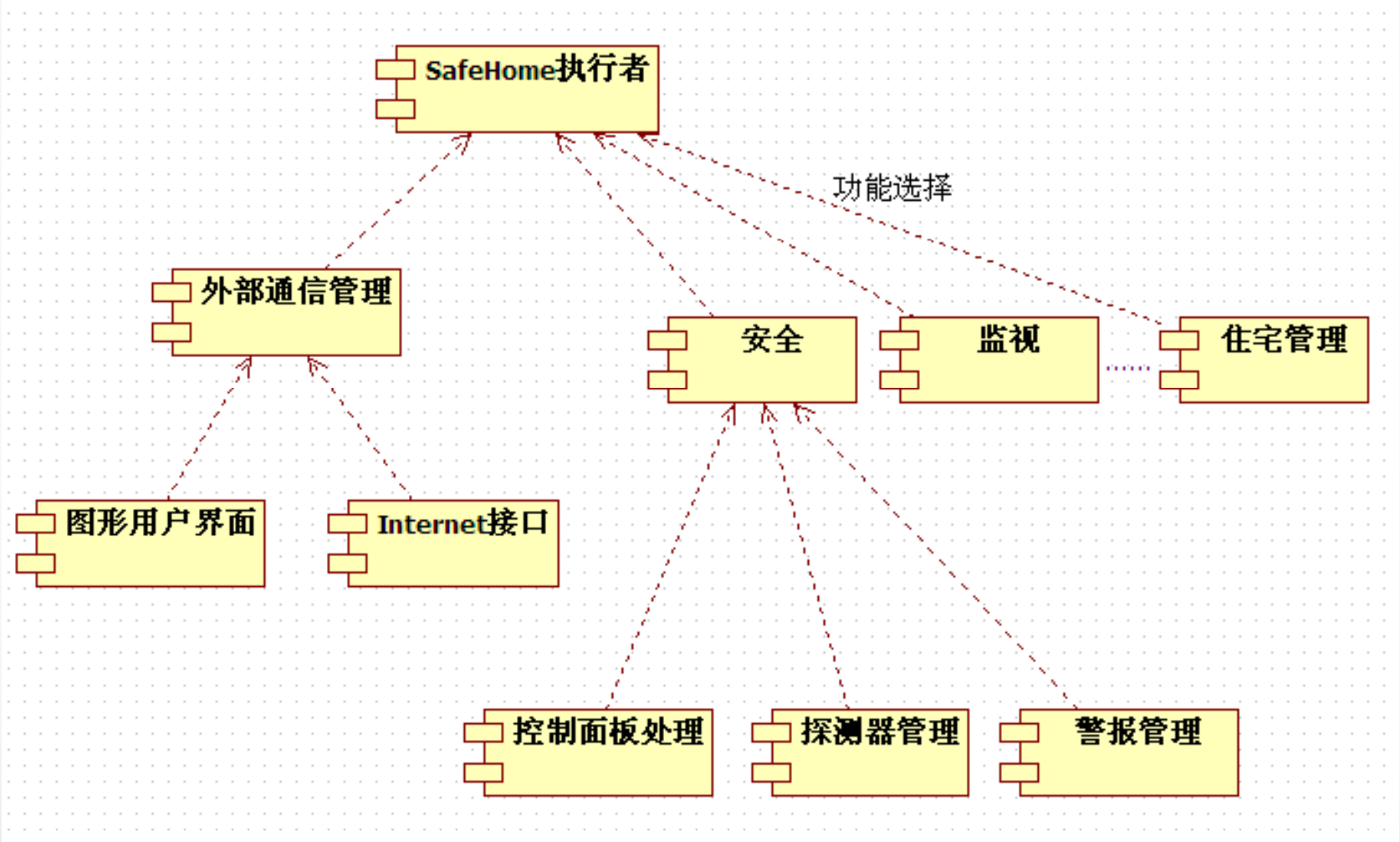


SafeHome安全功能

第二步 定义原型

软件体系架构设计

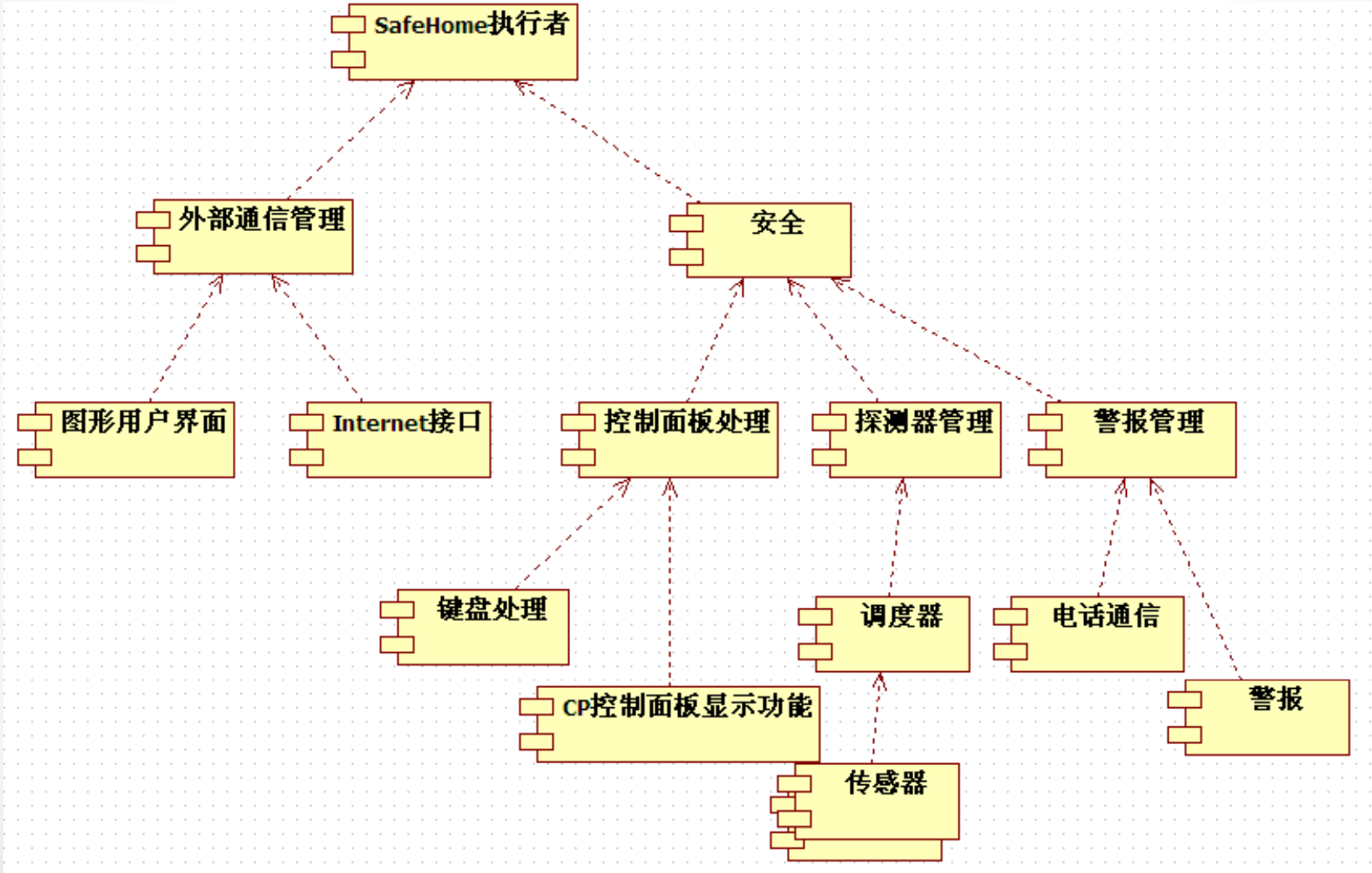
- 体系架构设计步骤



第三步 将体系结构精细化为构件

软件体系架构设计

- 体系架构设计步骤



第四步 描述系统实例

第四章 系统设计

4.1 系统设计概念

4.2 软件体系架构设计

4.3 构件级设计

4.4 程序流程图

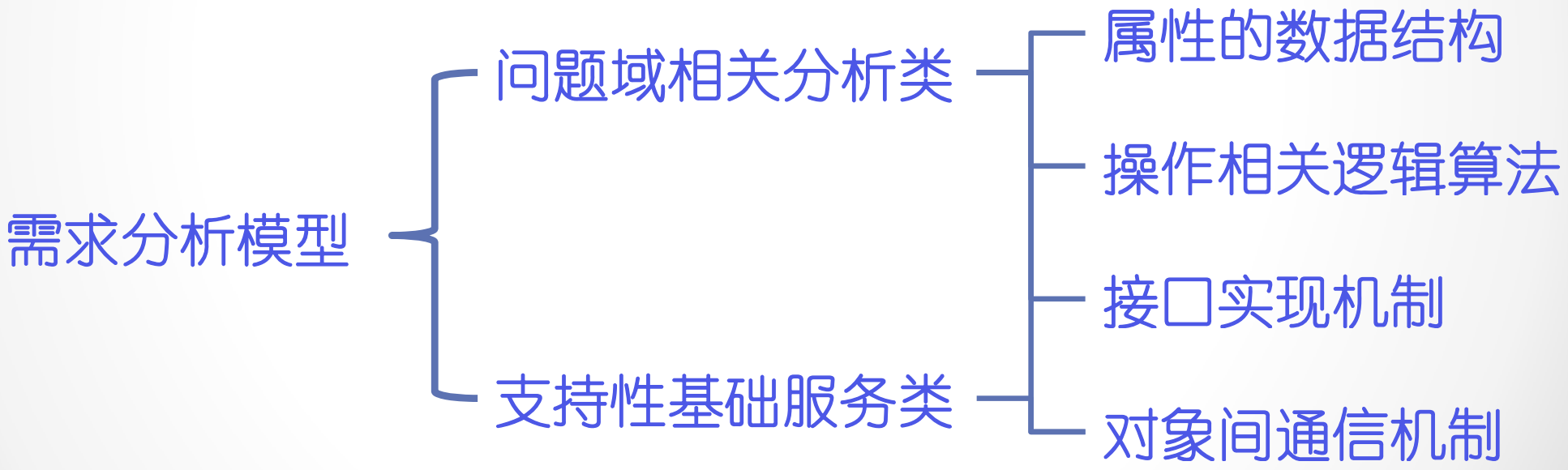
4.5 用户界面设计

构件级设计

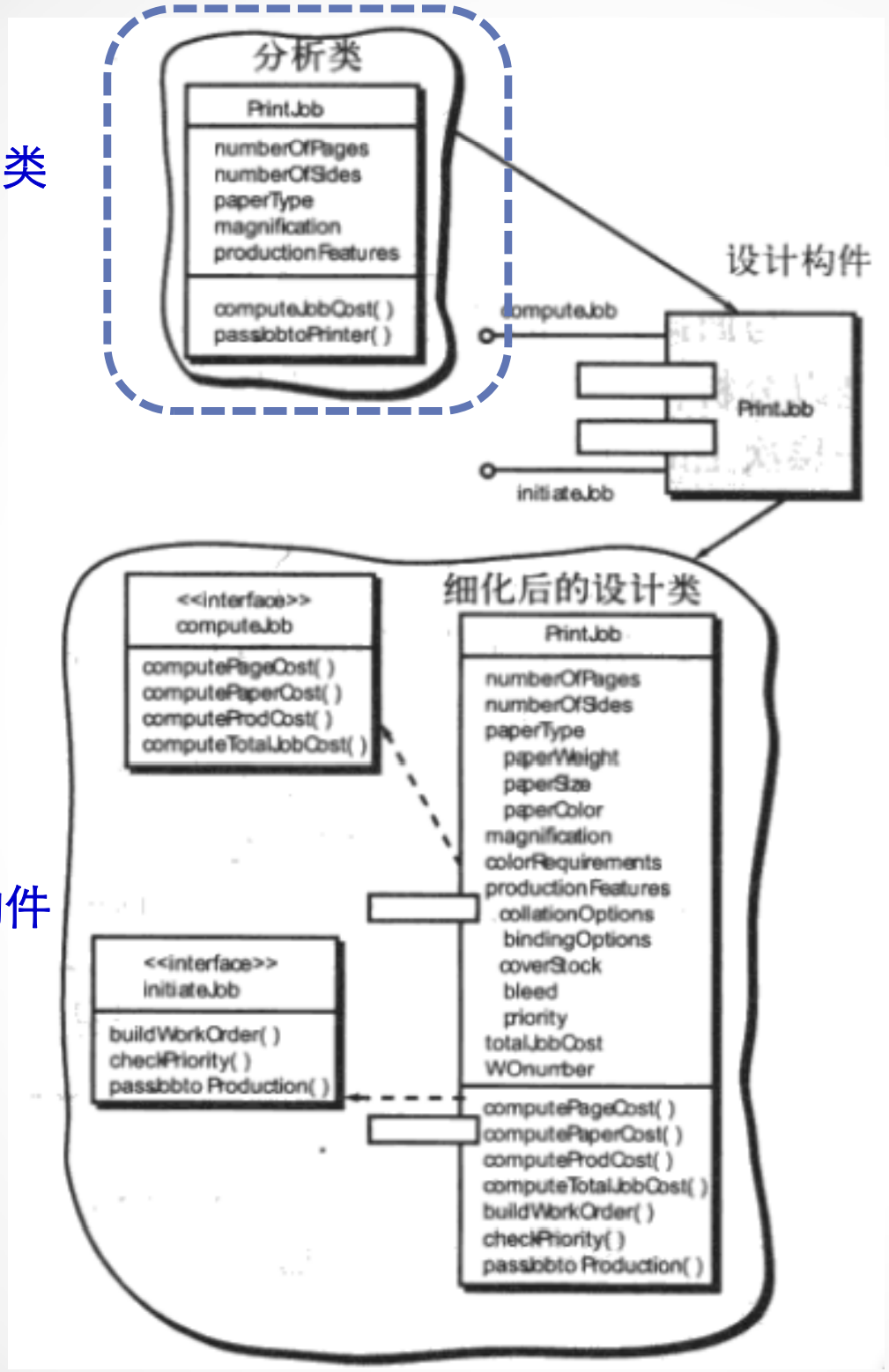
- 构件：系统中模块化的、可以部署和可以替换的部件，该部件封装了实现并暴露一组接口
- 构件存在于软件体系架构中
- 构件与其它构件以及软件外部实体通信

构件级设计

- 面向对象的构件：一组协作的类，每个类都详细描述，还需定义接口



需求分析类



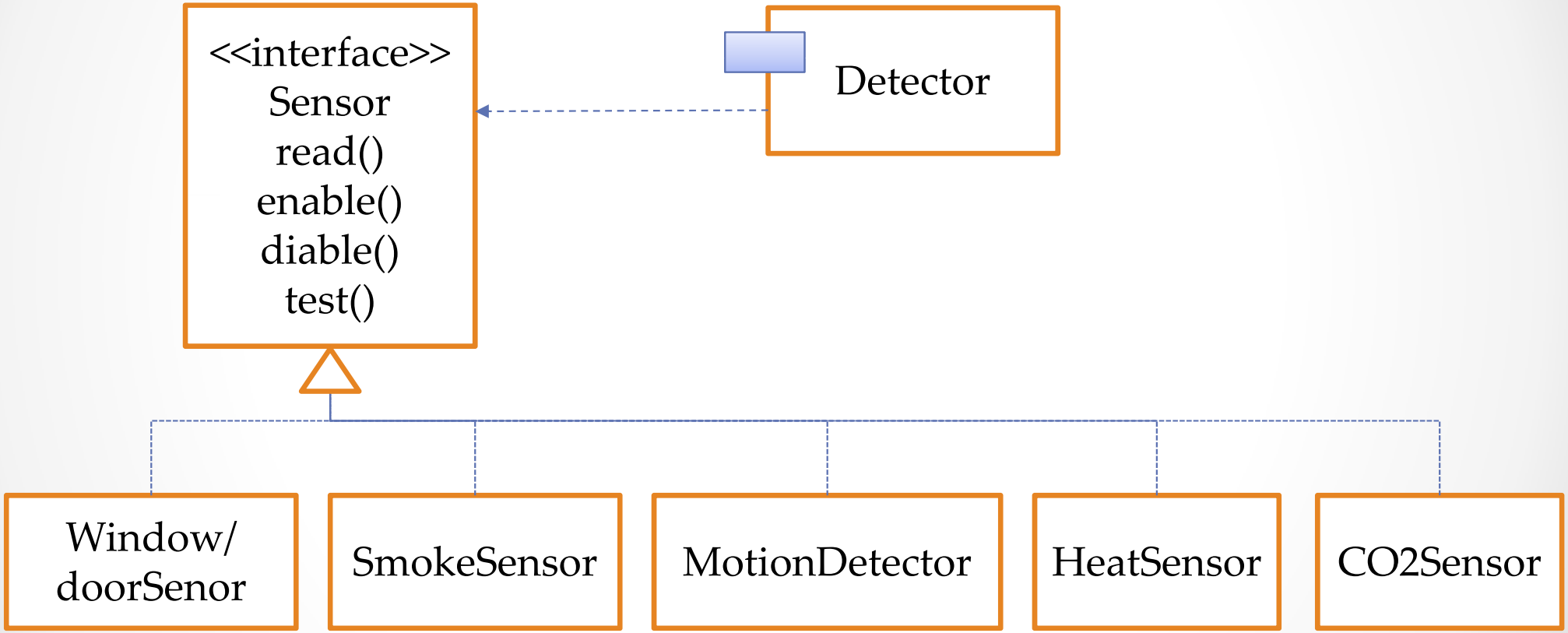
简化表示的设计构件

细化后的设计构件

目的：为高级车间构造软件，搜集前台的客户需求，对印刷业务定价，然后把印刷任务交给自动生产设备

构件级设计原则

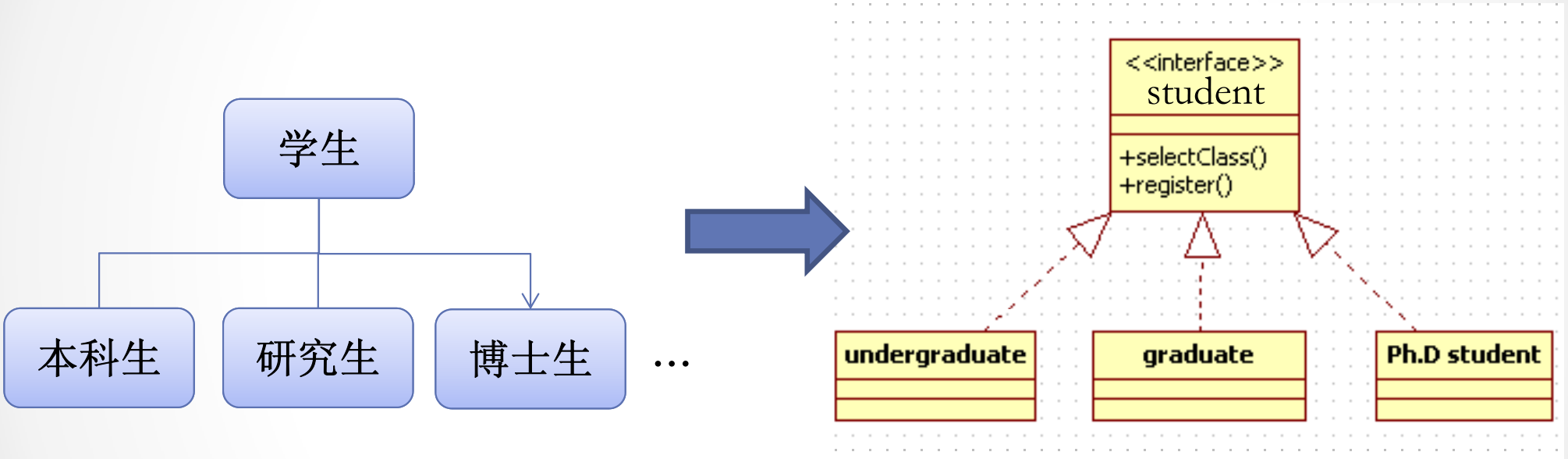
- 面向对象的构件设计原则



开闭原则：构件应该对外延具有开放性，对修改具有封闭性

构件级设计原则

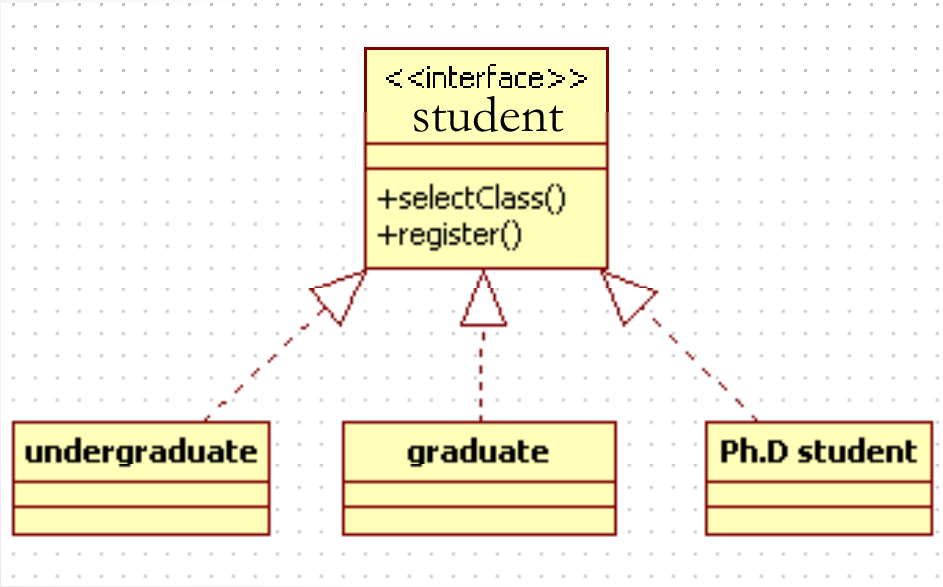
- 面向对象的构件设计原则



开闭原则：构件应该对外延具有开放性，对修改具有封闭性

构件级设计原则

- **Liskov替换原则：子类可以替换它们的父类**
- 源自父类的任何子类必须遵守父类与使用该父类的构件之间的隐含约定




```
Student s = new Undergraduate();
```

```
List allList = new ArrayList();
Collections.sort(allList);
```

构件级设计原则

- 接口分离原则：多个客户专用接口比一个通用接口好
- 为每个主要客户类型都设计一个特定接口

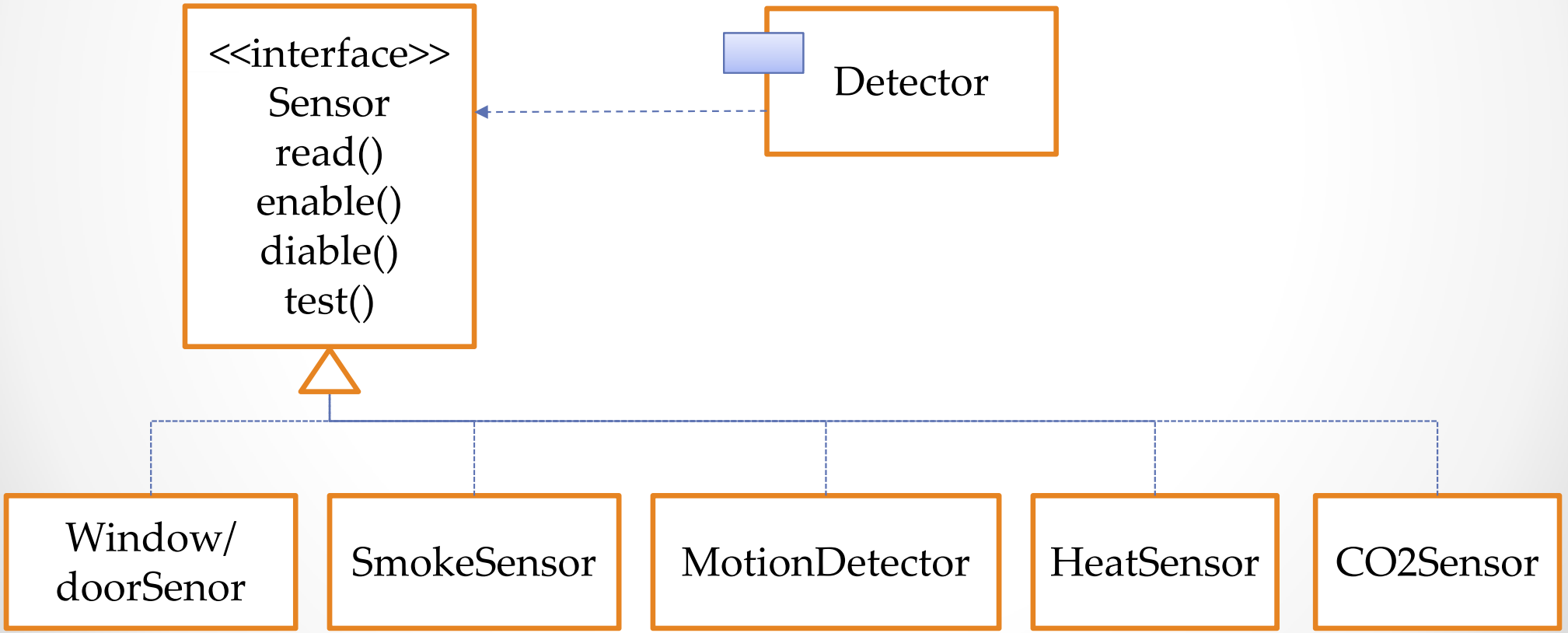
SafeHome系统：

安全接口： **placeDevice(), showDevice(), groupDevice(), removeDevice()**

监视接口：**placeDevice(), showDevice(), groupDevice(), removeDevice(), showFOV(), showDeviceID()**

构件级设计原则

- 依赖倒置原则：依赖于抽象，而非具体实现
- **Java**中，含有抽象方法的类称为抽象类，不能实例化



构件级设计原则

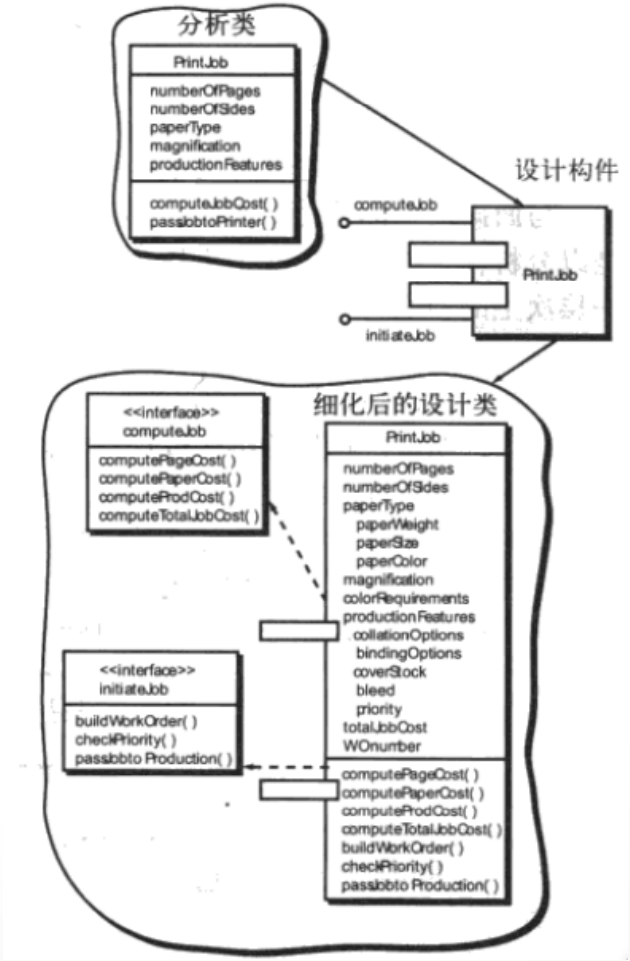
- 打包原则（多个构件或类形成包）
  - 发布复用等价性原则：将可复用的类分组打包成能够管理和控制的包并作为一个更新的版本，而不是对每个类分别进行升级
  - 共同封装原则：多个类根据内聚性打包，将变更限制在包中
  - 共同复用原则：不能一起复用的类不能被分到一组

构件级设计原则

- 内聚性
  - 功能内聚：一个模块只完成一组特定操作并返回结果
  - 分层内聚：在多层结构中，高层能访问低层的服务，但低层不能访问高层的服务
  - 通信内聚：访问相同数据的所有操作被定义在一个类中

构件级设计步骤

- 步骤1：标识出所有与问题域相对应的设计类：将需求分析类和体系架构构件进行精细化



构件级设计步骤

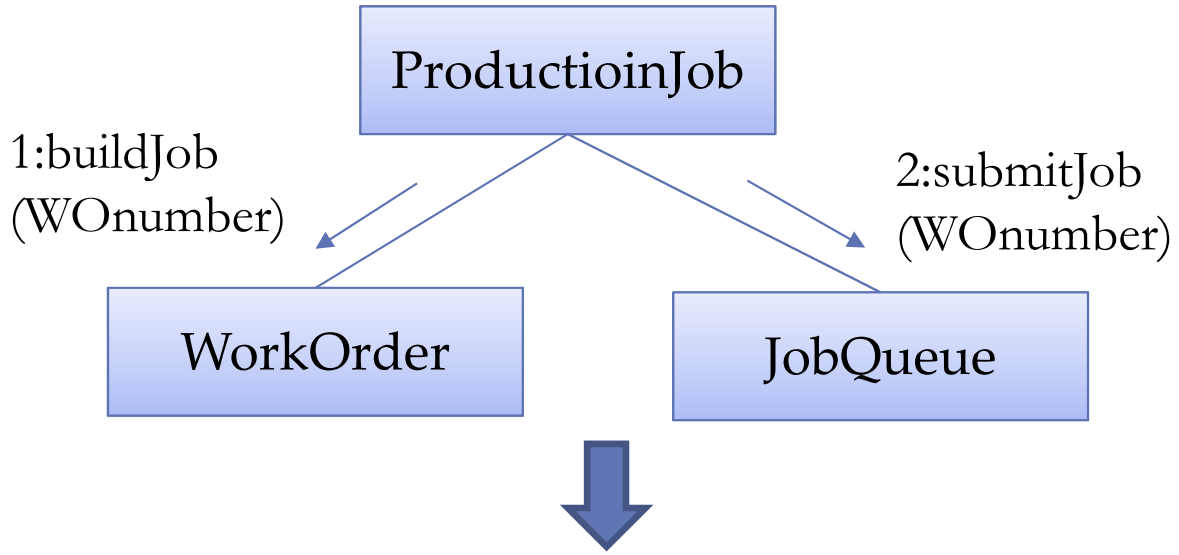
- 步骤2：确定所有与基础设施域相对应的设计类
  - 通用用户界面GUI
  - 操作系统构件
  - 对象和数据管理构件

构件级设计步骤

- 步骤3：细化所有不需要作为可复用构件的设计类，详细描述实现类需要的所有接口、属性和操作

构件级设计步骤

- 步骤3a：在类或构件协作时说明消息的细节，主要为对象间传递的消息

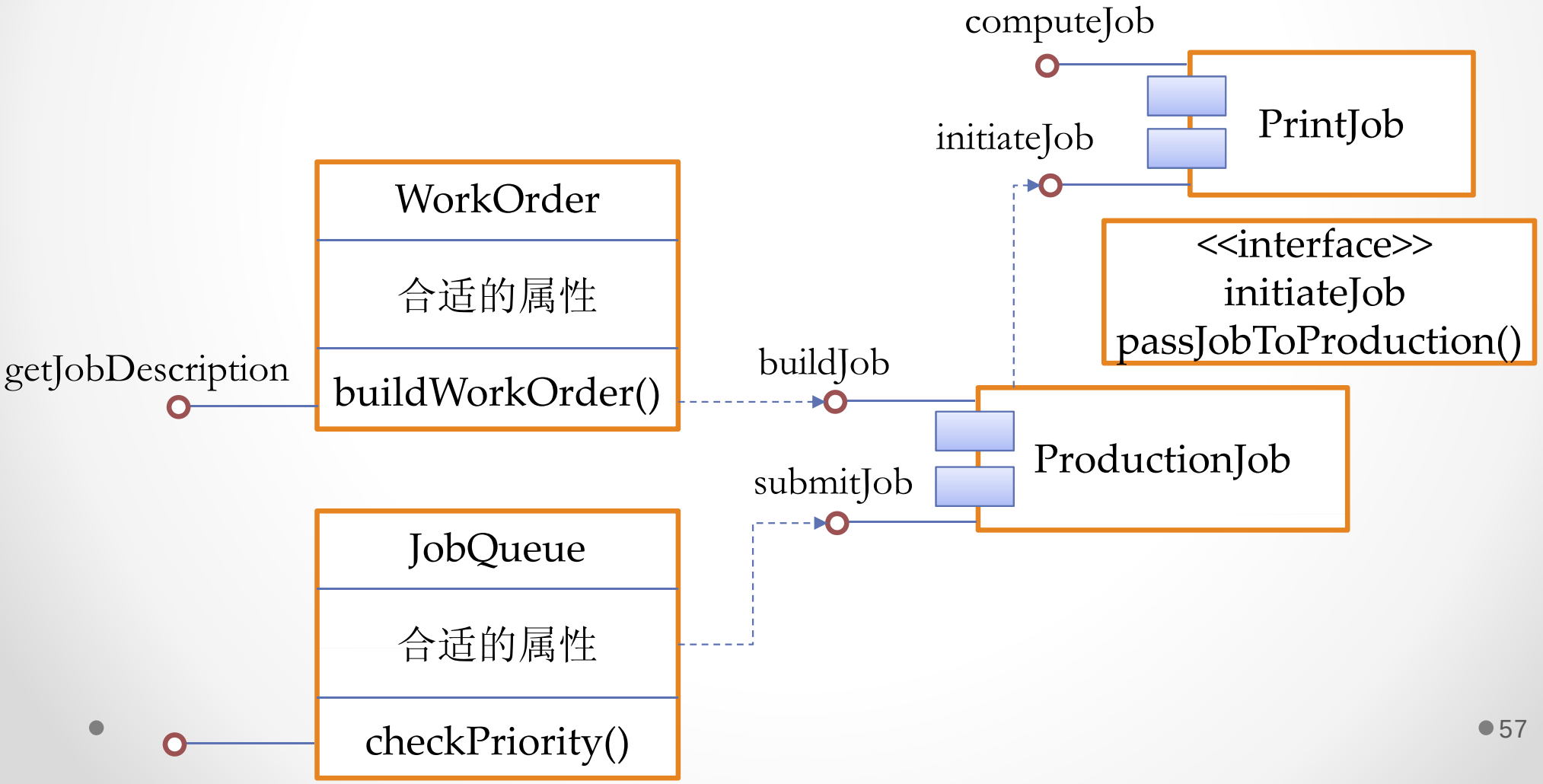


[guard condition]sequence expression(return value):=message name(argument list)

消息发出之前应满足的条件	消息发送序号	消息处理类的返回值	消息处理操作名	消息处理操作参数列表

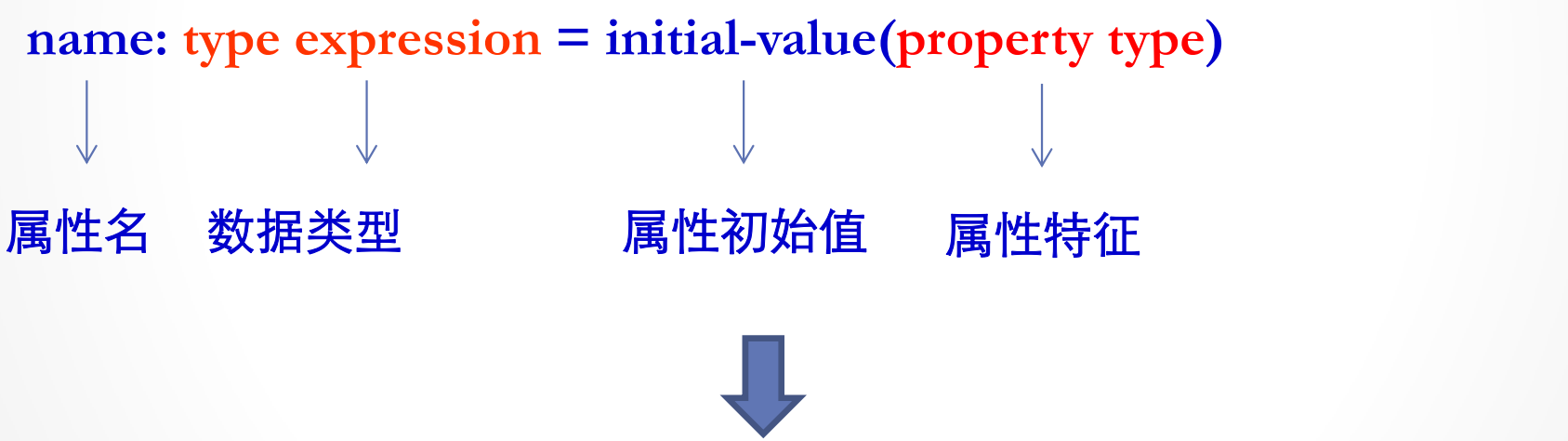
构件级设计步骤

- 步骤3b：为每个构件确定适当的接口，为设计类定义的操作可以归结为一个或者多个抽象类。

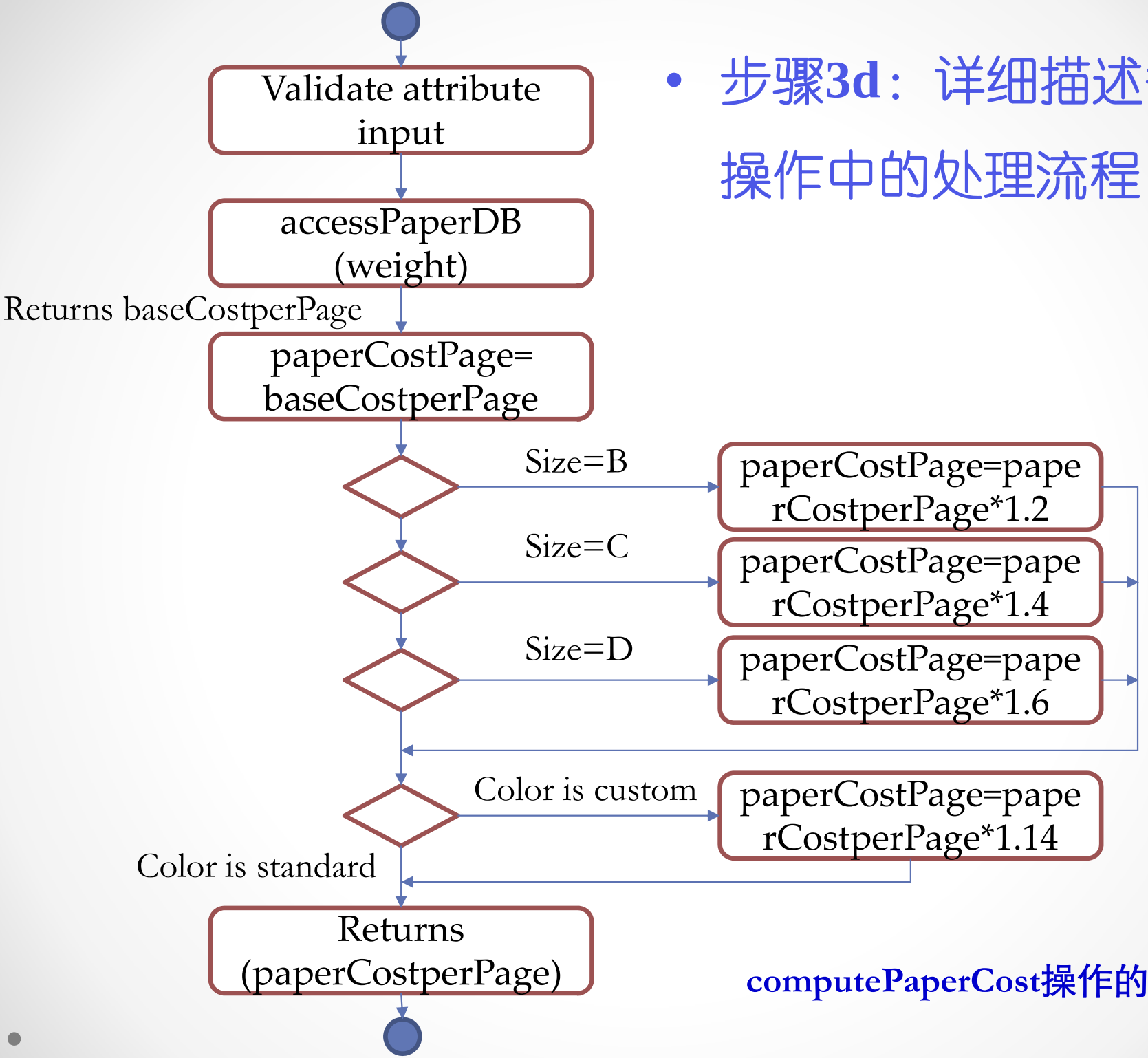


构件级设计步骤

- 步骤3c：细化属性，定义实现属性所需要的数据类型和数据结构



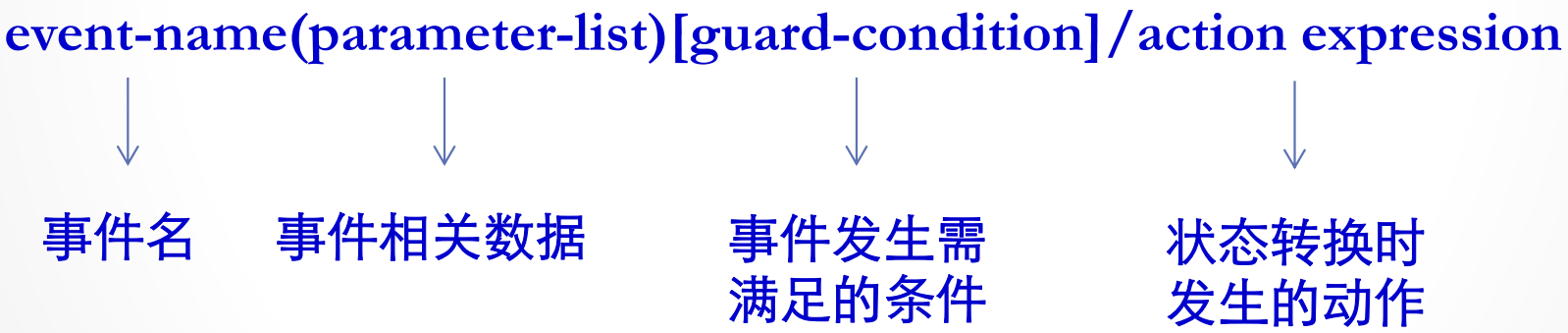
- 步骤3d：详细描述每个操作中的处理流程

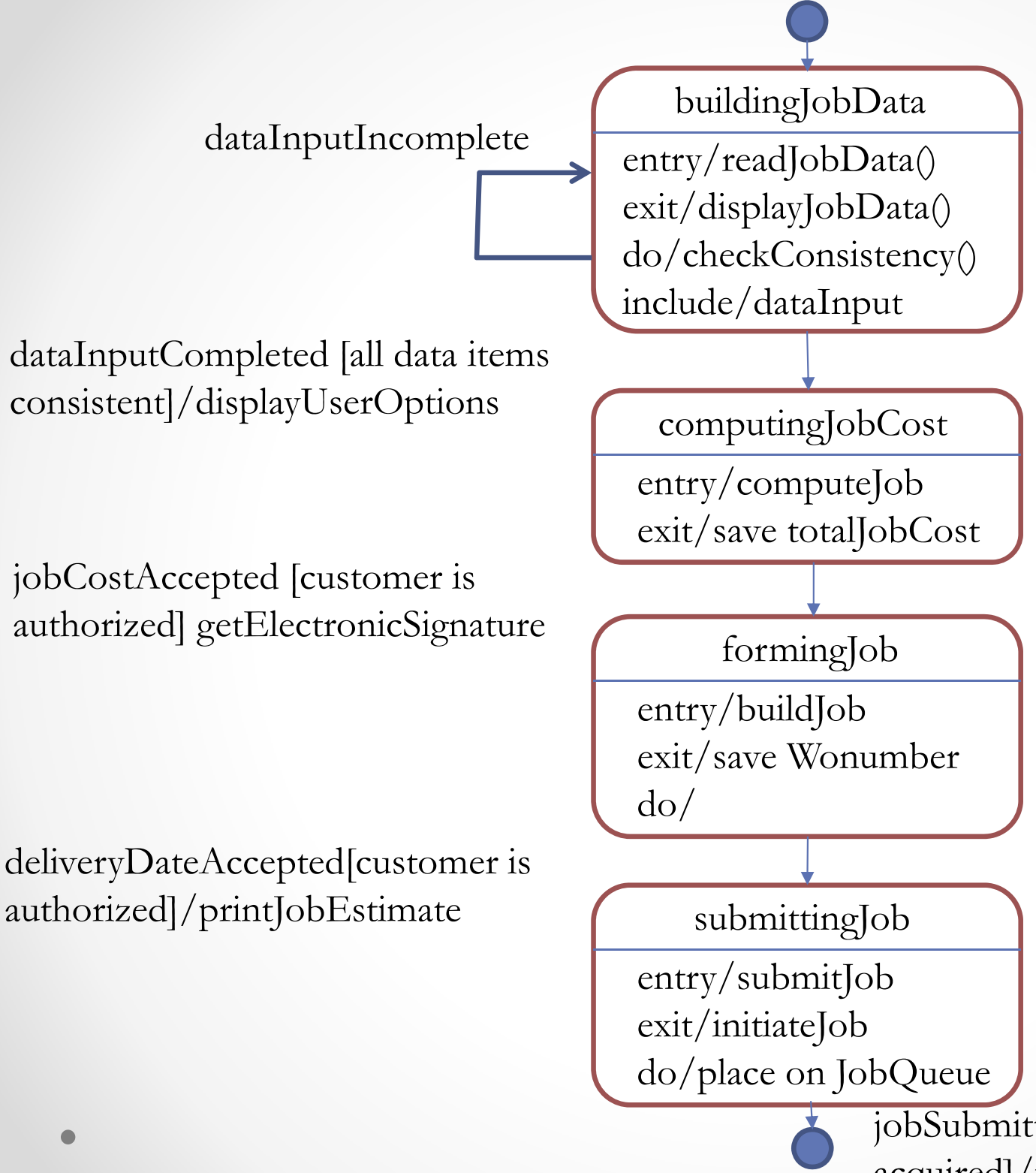


computePaperCost操作的活动图

构件级设计步骤

- 步骤4：说明持久数据源（数据库和文件）并确定管理数据源所需的类
- 步骤5：开发并细化类或构件的行为





PrintJob类的状态图

构件级设计步骤

- 步骤6：细化部署图以提供额外的实现细节
- 步骤7：考虑每个构件的设计表示，并时刻考虑其他可选方案，主要为重构与迭代

比较

- 比较软件体系架构设计与构件级设计
  - 软件体系架构设计属于宏观，构件级设计属于微观
  - 软件体系架构设计重视风格，构件级设计重视设计原则
  - 软件体系架构可从需求描述导出，构件设计可从需求描述与体系架构导出

第四章 系统设计

4.1 系统设计概念

4.2 软件体系架构设计

4.3 构件级设计

4.4 程序流程图

4.5 用户界面设计

4.4.1 程序流程图

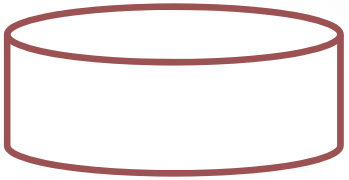
程序流程图又称程序框图，是用统一规定的标准符号描述程序运行具体步骤的图形表示。

程序框图的设计是在处理流程图的基础上，通过对输入输出数据和处理过程的详细分析，将计算机的主要运行步骤和内容标识出来。

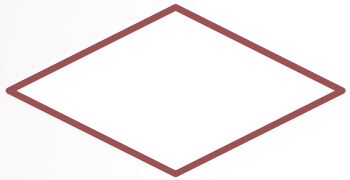
程序框图的优点是采用规范的符号，画法简单清晰。

程序流程图的符号

程序处理过程



数据存储



判断决策

预定义过程

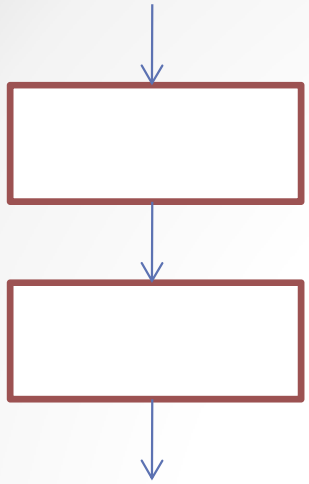
文档

开始

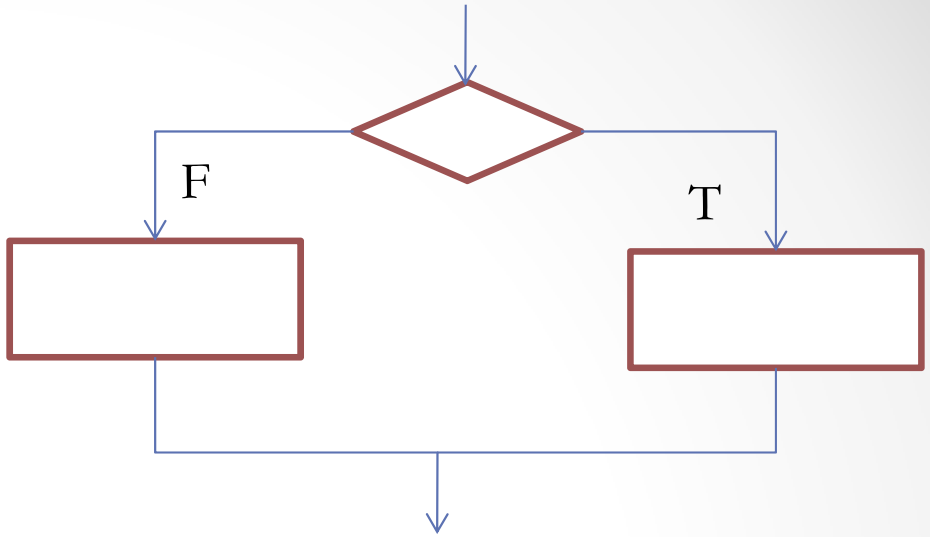
多文档

结束

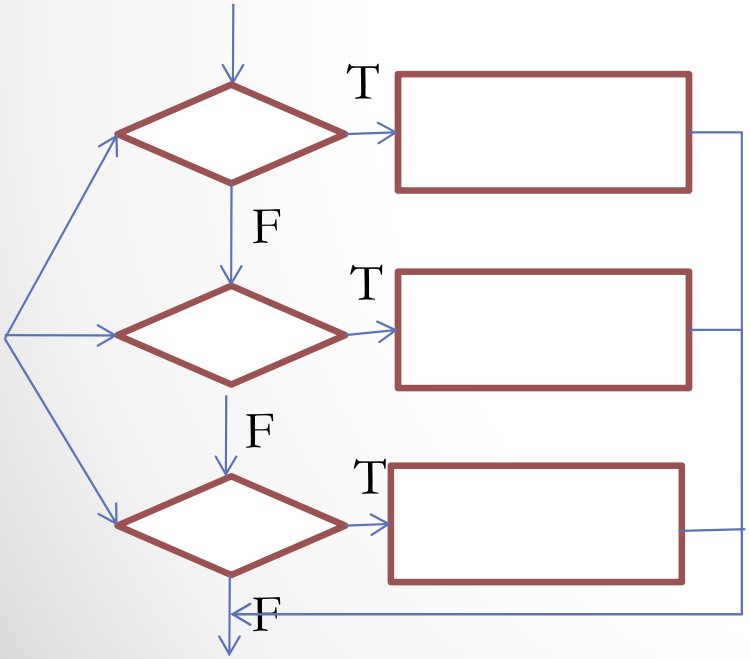
路径



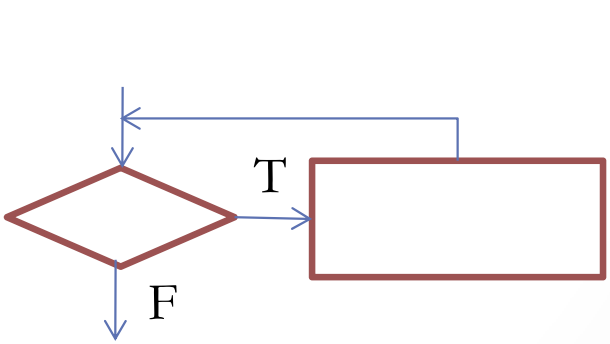
顺序型

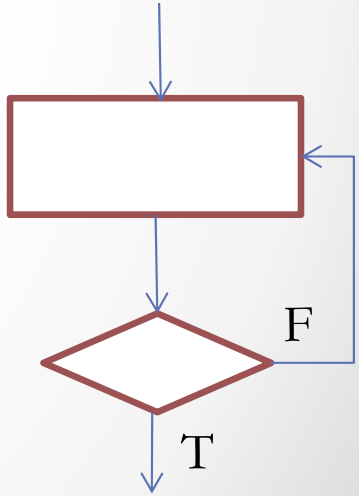


if-then-else型



选择型





重复型

画一个SafeHome的流程图

报警功能的流程图

第四章 系统设计

4.1 系统设计概念

4.2 软件体系架构设计

4.3 构件级设计

4.4 程序流程图

4.5 用户界面设计

界面设计原则

- 黄金原则
  - 用户操纵控制
  - 减少 用户的记忆负担
  - 保持界面一致

界面设计原则

- 用户操纵控制
  - 为用户主动避免不必要或不期望的动作－拼写检查模式
  - 提供灵活的交互－使用鼠标画图，而不是键盘
  - 允许用户交互被中断或撤销－**word**撤销功能
  - 用户定制－宏定义，批处理
  - 使用户与内部技术细节隔离开
  - 通过屏幕直接交互对象－智能手机，**ipad**

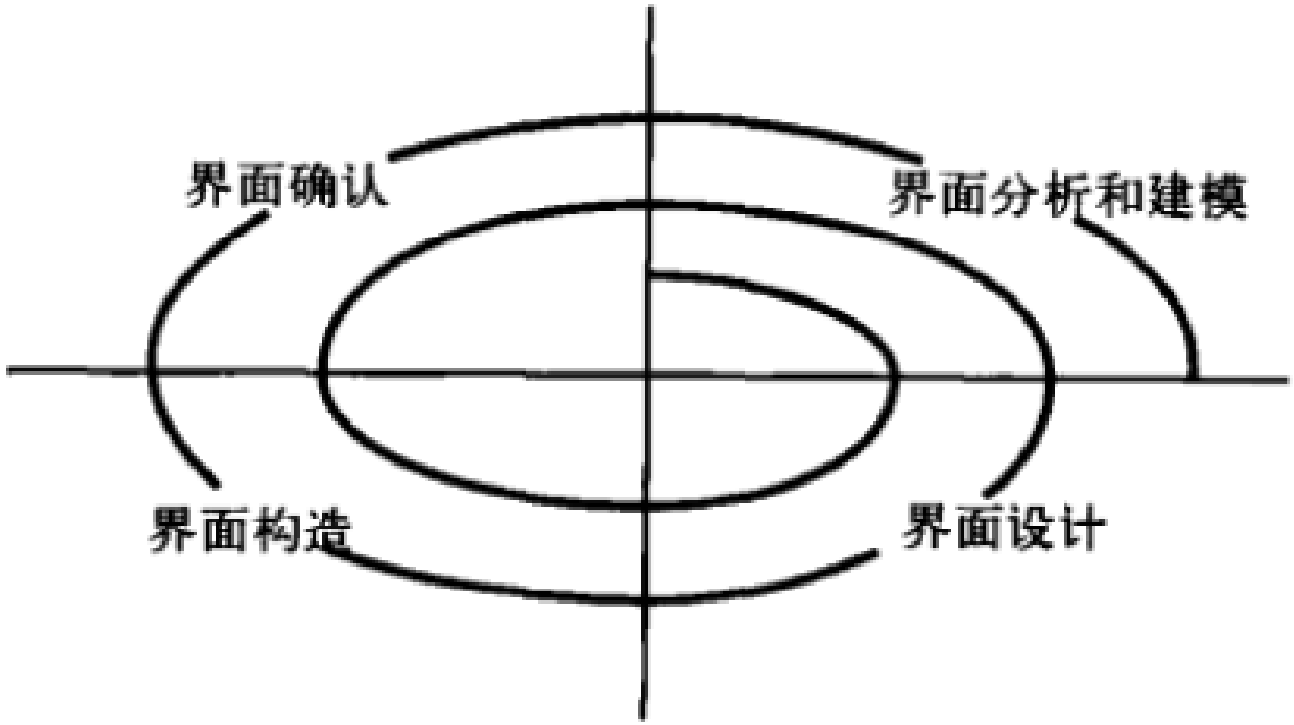
界面设计原则

- 减轻用户记忆负担
  - 减少对短期记忆的要求－可视的提示，历史记录
  - 建议有意义的缺省－**reset**，浏览器首页
  - 快捷方式
  - 界面布局基于真实世界象征－管理系统中的单据
  - 层层展开方式揭示信息－多层窗口

界面设计原则

- 保持界面一致
  - 允许用户将当前任务放入有意义的环境中－让用户了解当前界面来自何处，有什么新途径转入到新任务
  - 在应用系统家族内保持一致性－**office**软件界面
  - 保持过去的使用经验－**Ctrol+S**表示存储

界面设计过程



螺旋过程：

1、界面分析与建模

2、界面设计

3、界面构造

4、界面确认

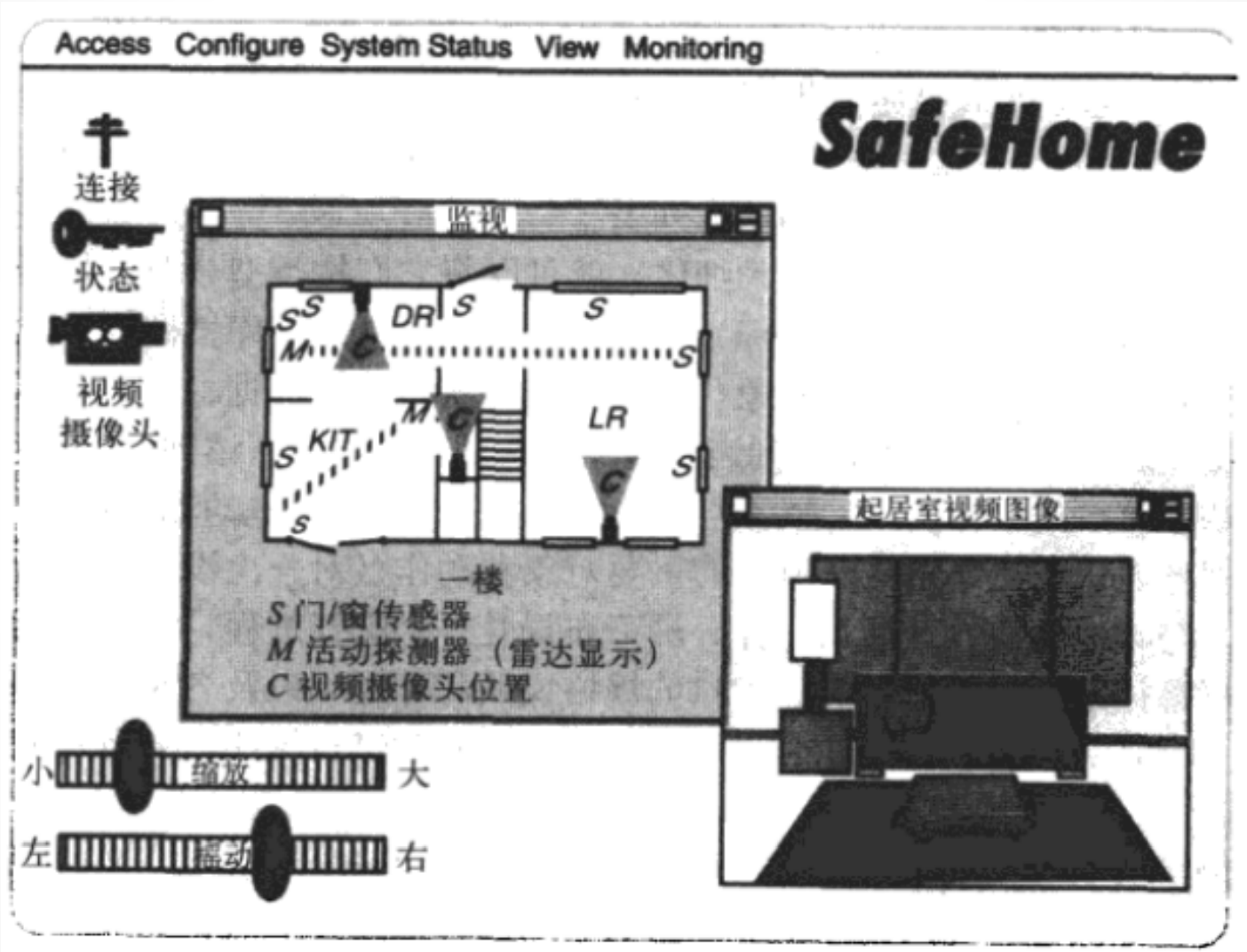
界面设计过程

- 界面分析与建模
  - 用户分析－用户、销售、市场、支持四种角色
  - 任务分析与建模－用例、过程
  - 显示内容分析－文字、图形、音视频
  - 工作环境分析－嵌入式显示和大屏幕显示

界面设计步骤

- 界面设计步骤
  - 使用界面分析中获得的信息，定义界面对象与动作
  - 定义导致用户界面状态发生变化的事件，并对行为建模
  - 描述每个界面状态，就像最终用户看到的那样
  - 简要说明用户如何从界面提供的信息来揭示系统状态

界面设计步骤



视频监控屏幕初步设计

Question?