CHAPTER 4

Architecture Design

软件需求　vs　软件设计

软件需求：解决“做什么”

软件设计：解决“怎么做”　“如何做”

前，因

后，果

软件设计是把软件需求变换成为软件的具体解决方案

将用户要求转换为一个具体的设计方案，即决定系统“怎样做”

软件设计：总体设计，详细设计

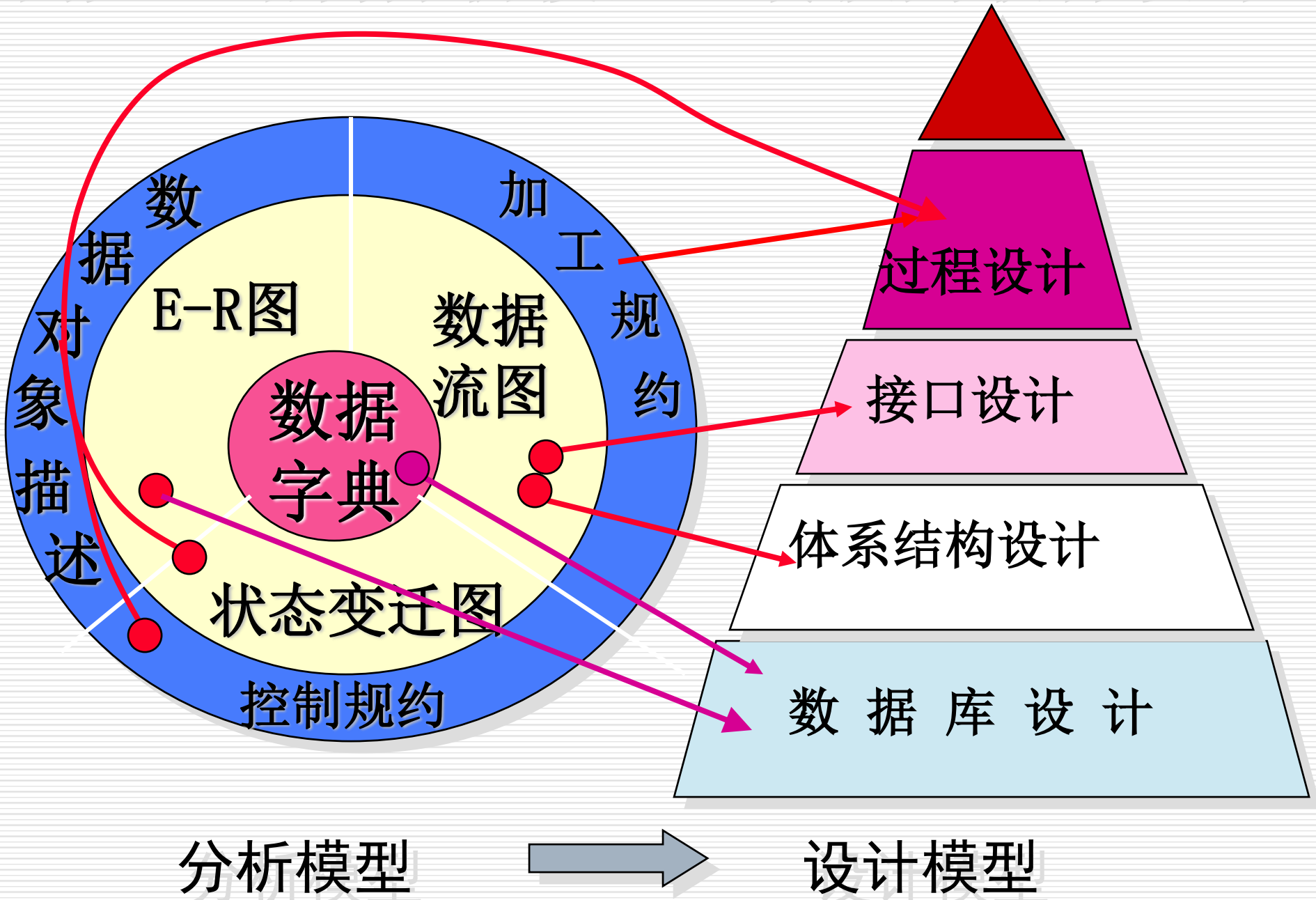
软件需求 vs 软件设计

任务：

软件需求：分析模型

软件开发：设计模型

例如：结构分析模型，转换为软件设计



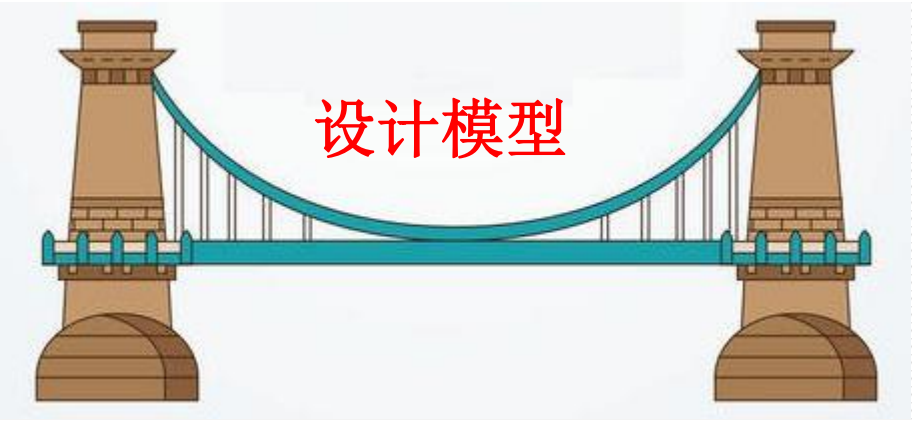
什么是软件设计，为什么要设计？

- 面向特定的软件需求，探求可行的、尽可能最优的软件解决方案的过程。
  - ✓ 软件解决方案表示为软件设计模型

循序渐进、不断精化，依靠经验，综合比较

- 桥梁

需求模型



实现模型

细节

总体设计 vs 详细设计

(1) **总体设计（概要设计）**

确定软件的结构以及各组成成分(子系统或模块)之间的相互关系。

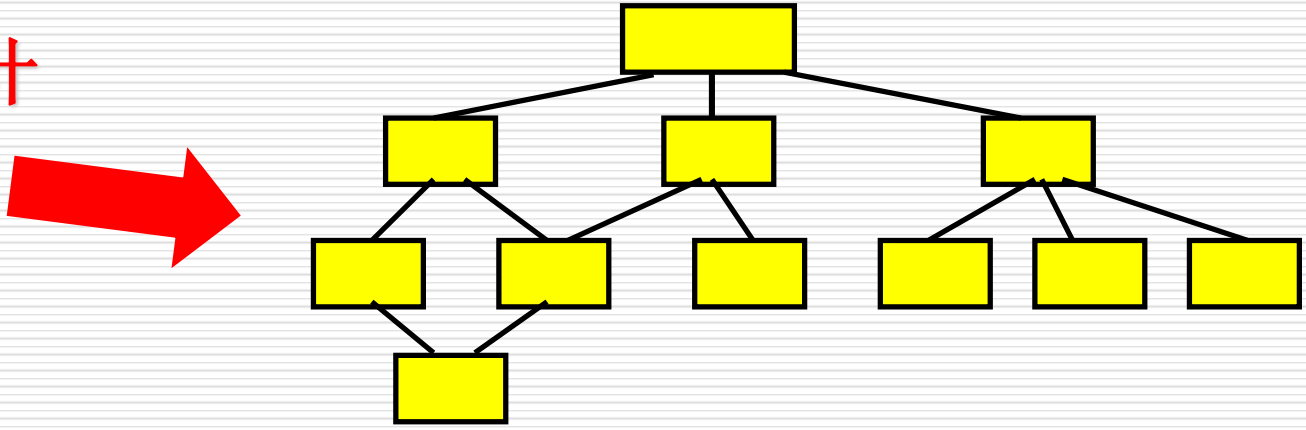
(2) **详细设计**

确定模块内部的算法和数据结构，产生描述各模块程序过程的详细文档。

总体，细节；战略，战术；宏观，微观；…

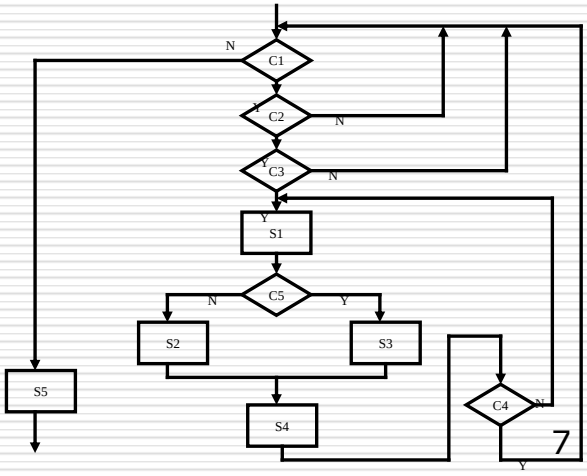
总体设计 vs 详细设计

总体设计



详细设计

模块？



软件设计的任务（具体）

总体设计的任务：将复杂系统按功能分成模块、确定每个模块的功能和模块之间的调用关系、块间传递的信息、评价模块结构的质量。

概要设计文档主要有：概要设计说明书（或设计规格说明书）、数据库设计说明书

详细设计的任务：为每个模块进行详细的算法设计，用某种图形、表格、语言等工具将每个模块处理过程的详细算法描述出来；为模块内的数据结构进行设计；对数据库进行物理设计（确定数据库的物理结构）；确定其他设计。

详细设计文档主要有：详细设计说明书

总体设计的任务和重要性

- 将系统划分成模块
- 决定每个模块的功能
- 决定模块的调用关系
- 决定模块的界面，即模块间传递的数据

重要性：站在全局的高度，从抽象的层次，设计软件的总体实现方案，确定软件最合理的结构，从而以较低的成本，指导后续软件设计和实现，保障软件质量。

Different aspects of design

- ***Architecture design:***
  - The division into subsystems and components,
    - How these will be connected.
    - How they will interact.
    - Their interfaces.
- ***Class design:***
  - The various features of classes.
- ***User interface design***
- ***Algorithm design:***
  - The design of computational mechanisms.
- ***Protocol design:***
  - The design of communications protocol.

详细设计

“总体设计” 同义词

- ✓ Architecture Design
- ✓ General Design
- ✓ 软件体系结构设计
- ✓ 软件总体设计
- ✓ 软件概要设计
- ✓ 软件初步设计
- ✓ logical design
- ✓ high-level design

What is Software Architecture?

A definition:

Software Architecture is the global organization of a software system, including

- the division of software into subsystems/components,
- deciding on how these subsystems interact, and
- the definition of their interfaces.

(free after) Object Oriented Software Engineering
T. C. Lethbridge & R. Laganière
McGraw Hill, 2001

What is Software Architecture?

Classic Definitions 1

An architecture is the set of significant decisions about

✓ the organization of a software system,

✓ the selection of the structural elements and their interfaces by which the system is composed, together with their behaviour as specified in the collaborations among those elements,

✓ the composition of these structural and behavioural elements into progressively larger subsystems,

✓ the architectural style that guides this organization

The UML Modeling Language User Guide, Addison-Wesley, 1999
Booch, Rumbaugh, and Jacobson

What is Software Architecture?

Classic Definitions 2

The structure of the components of a program/system,

their interrelationships, and principles and guidelines governing their design and evolution over time.

David Garlan and Dewayne Perry
April 1995 IEEE Transactions on Software Engineering

What is Software Architecture?

Definition 3

Architecture of software is a collection of design decisions that are expensive to change.

Alexander Ran, Nokia Research
September 2001 European Conference on Software Engineering

"The things that are fixed"

什么是软件体系结构

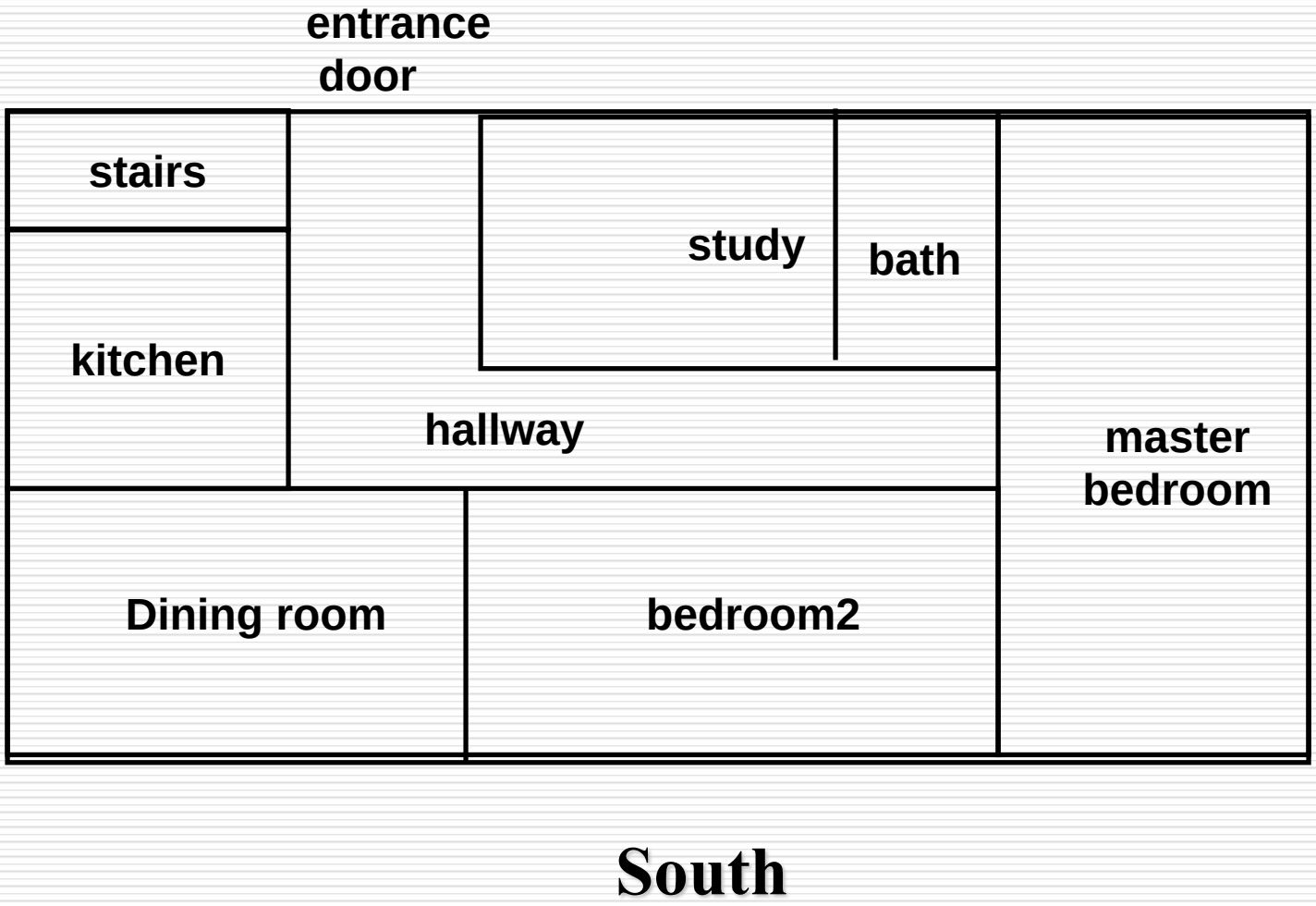
- 软件体系结构定义了软件局部和总体计算部件的构成。从整体看，软件体系结构是由结构和功能各异、相互作用的部件集合，按照层次构成的。
- 软件体系结构定义了组成部件之间的相互作用关系。
- 软件体系结构定义了构成系统的合成原理、方法、原则
- 软件体系结构定义了构成系统应该遵守的约束的条件。

联想硬件体系结构，冯-诺依曼 计算机系统结构

造房子的例子

- **Requirements**
  - The house should have two bedrooms, a study, a kitchen, a bathroom, living area.
  - The overall distance occupants walk every day should be minimized
  - The use of daylight should be maximized
- **Assumptions**
  - Most of the walking will be from
    - the entrance door to the kitchen for groceries
    - the kitchen to the dining area before and after meals
    - between bedrooms and bath
  - Occupants spend most of their time in living/dining area and the master bath

房子的平面设计图


entrance
door
stairs
kitchen
study
bath
hallway
master
bedroom
Dining room
bedroom2
South

书本（论文）目录

基于语法和语义结合的源代码精确搜索方法（小论文）

总体设计的任务

- 确定软件的组成
- 确定各组成部分的相互关系
- 确定软件的运行模式
- 确定软件的若干原则

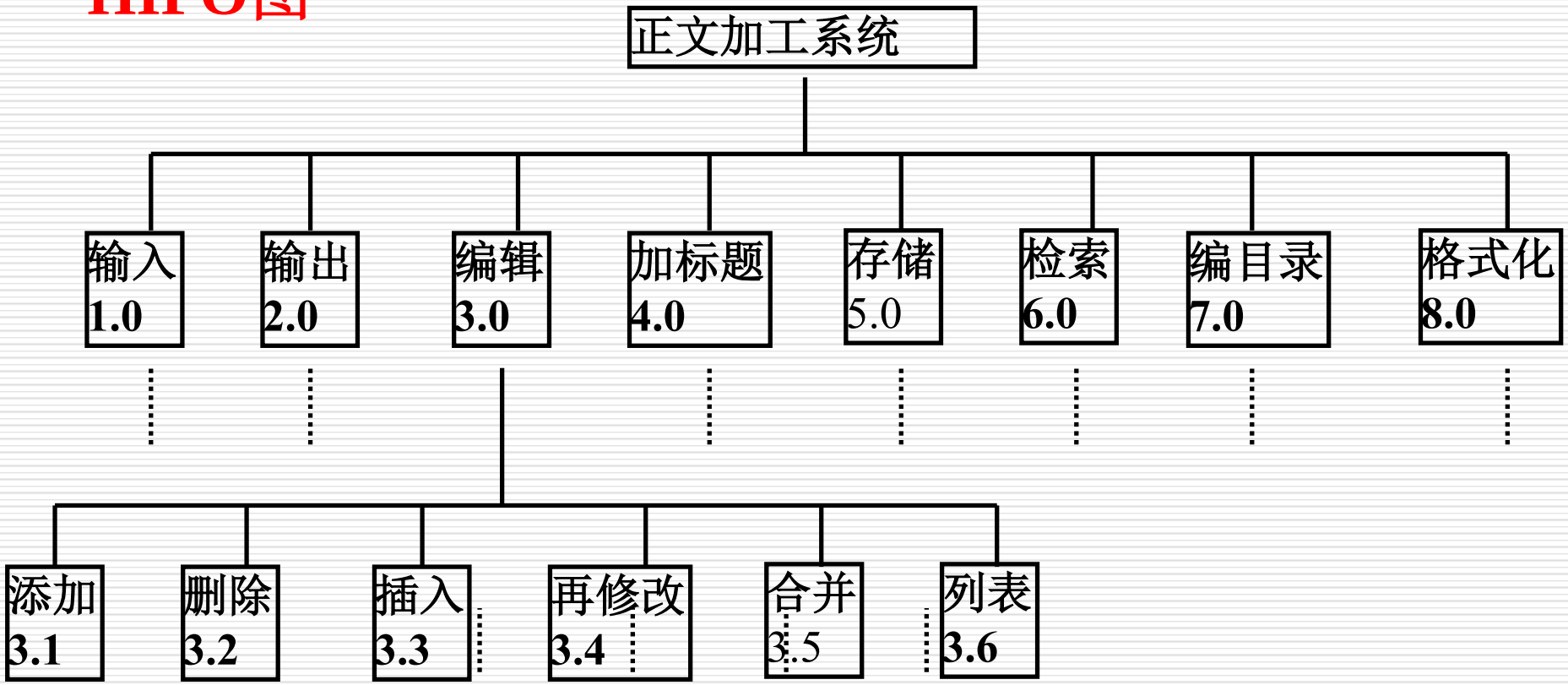
总体设计的关键技术

- 工具 — 如何描述软件的总体结构
- 方法 — 用什么方法从问题结构导出软件结构
- 评估准则 — 什么样的软件结构是“最优的”

描述方法

Notation: Box, arrow (line), text

HIPO图



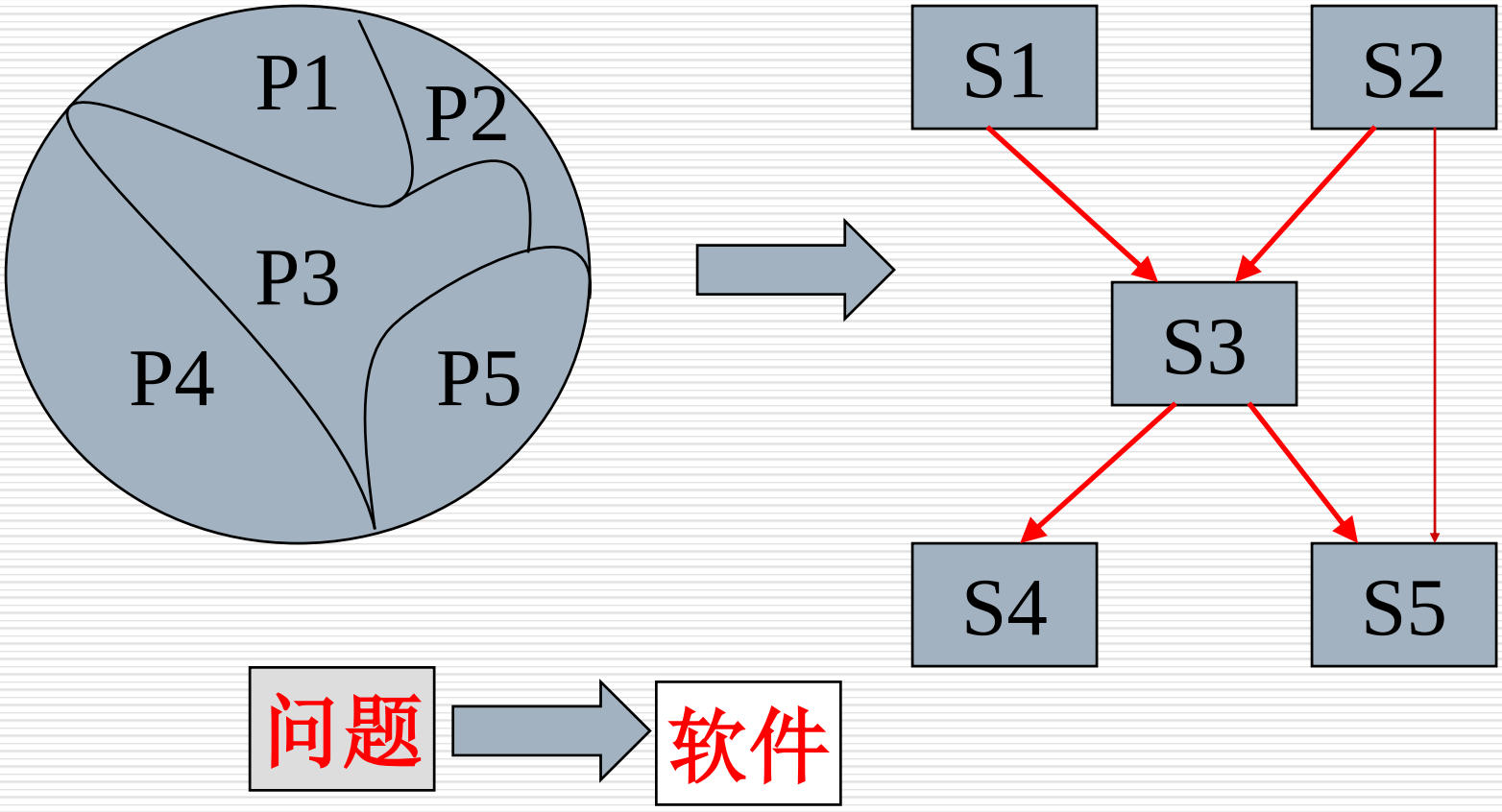
软件设计方法

- 结构化设计方法(SD)
- 面向对象的设计方法(OOD)
- 面向数据结构的设计方法(JSD方法)

评估准则

1. 经验启发式规则
2. 模块化
3. 抽象
4. 信息隐蔽
5. 信息局部化

怎样进行总体设计？



总体结构的设计方法

如何导出软件总体结构？

如何进行总体结构设计？

1. 根据经验
2. 根据模式和风格
3. 根据DFD
4. 根据对象模型
5. ……

（1）根据经验划分子系统结构

总原则：根据待解决的问题特点
和用户需求

经验方法

Methods:

- ✓by function
- ✓by organization,
- ✓by component
- ✓by process
- ✓by object
- ✓by data
- ✓by menu
- ✓by control
- ✓by domain
- ✓…or mixed

system structuring based on function

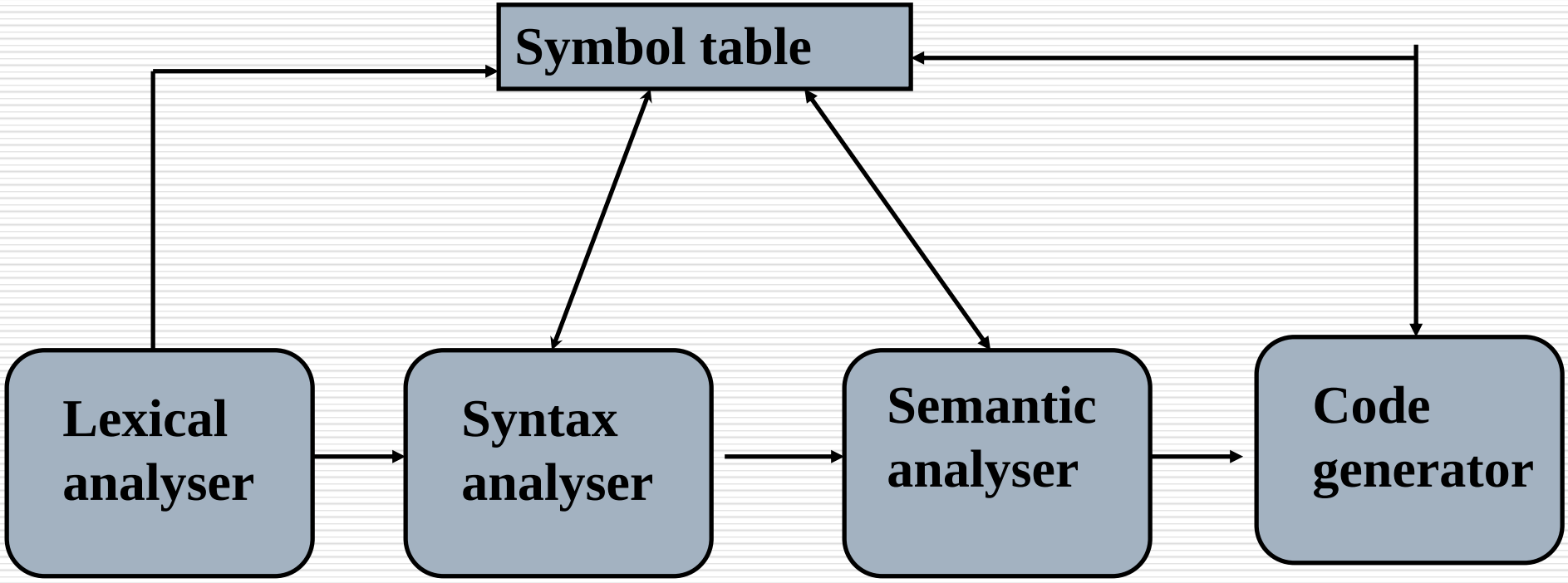
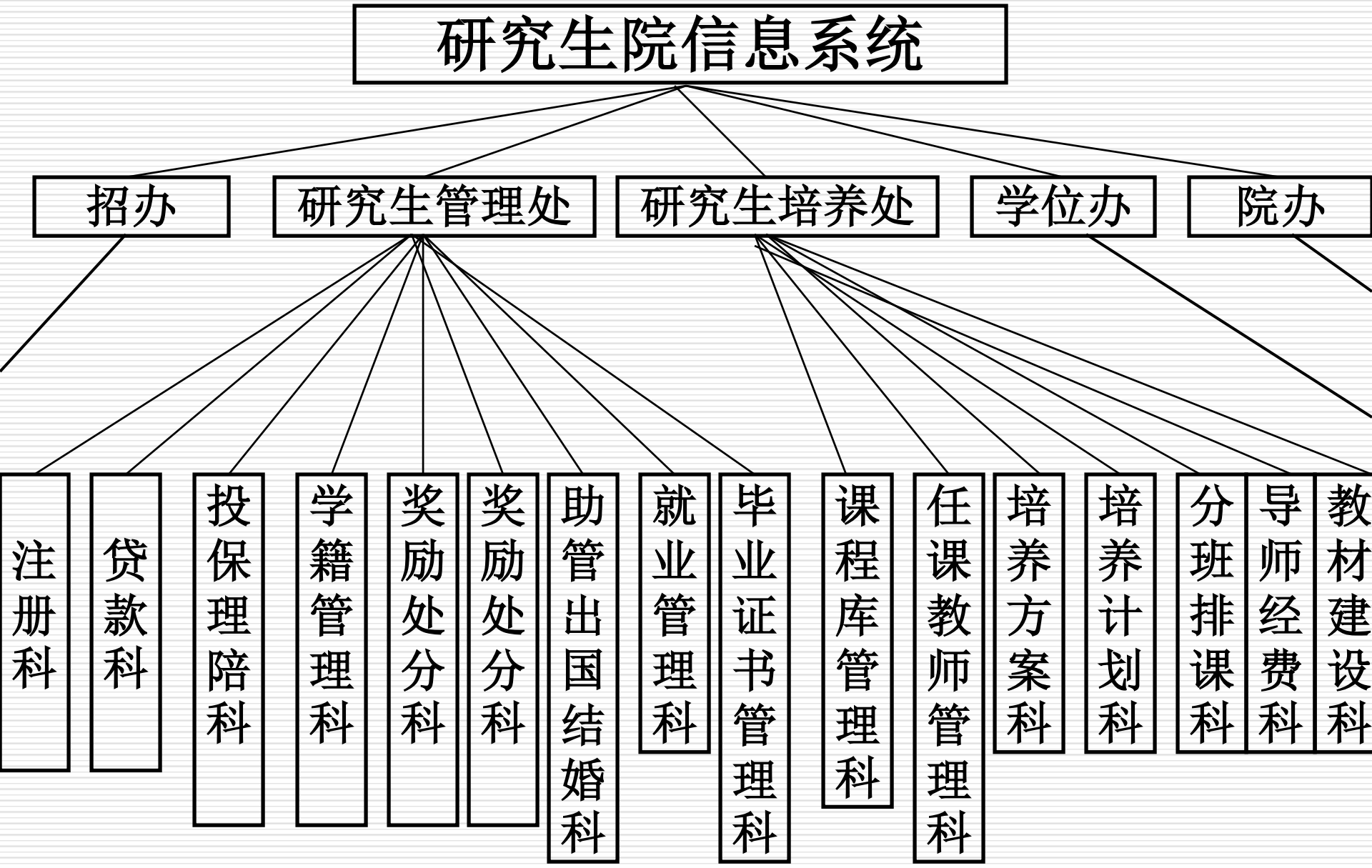


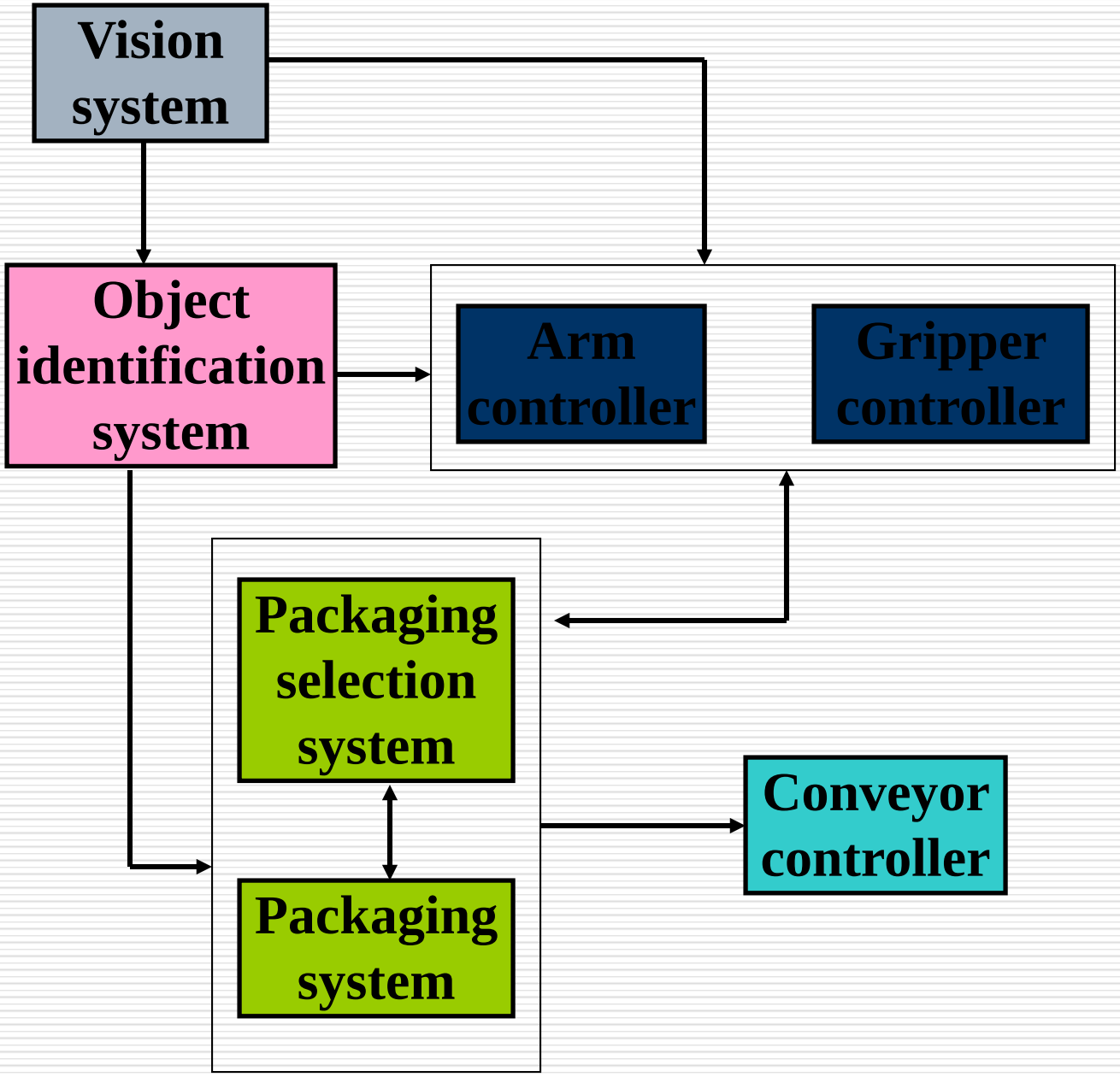
Figure: Compiler architecture

system structuring based on organization



system structuring based on components（硬件）

Block diagram of Packing robot system



system structuring based on process

system structuring based on object

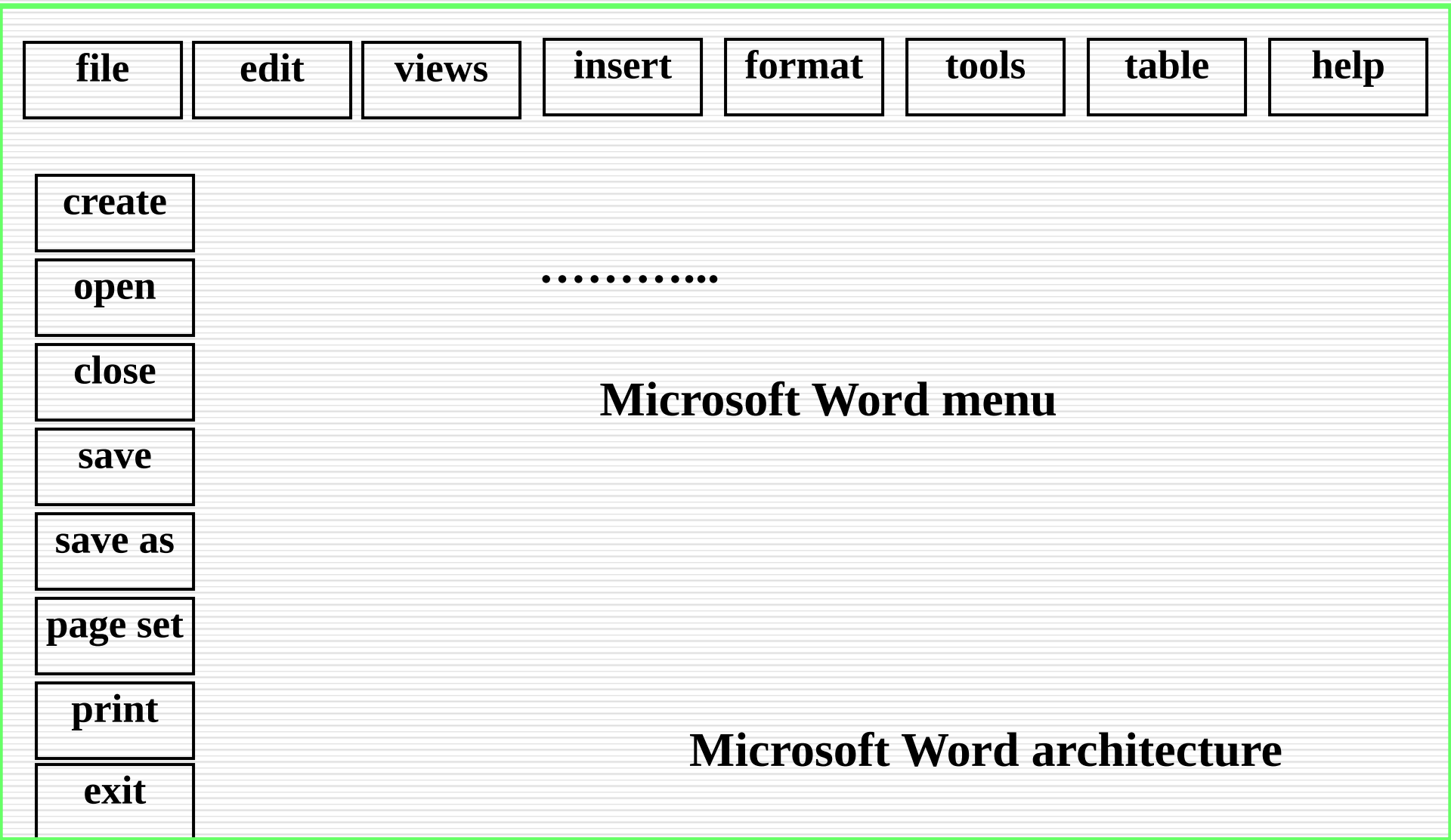
- **object model (system static model) discussed before**
- **use it where identifying objects easily**
- **game software**

War game:

soldier, weapon, tank, plane, camp, city, power, communication, support,

system structuring based on menu

Window program, Web program, visual program



- **system structuring based on control**

One sub-system has overall responsibility for control, and starts and stops other sub-system.

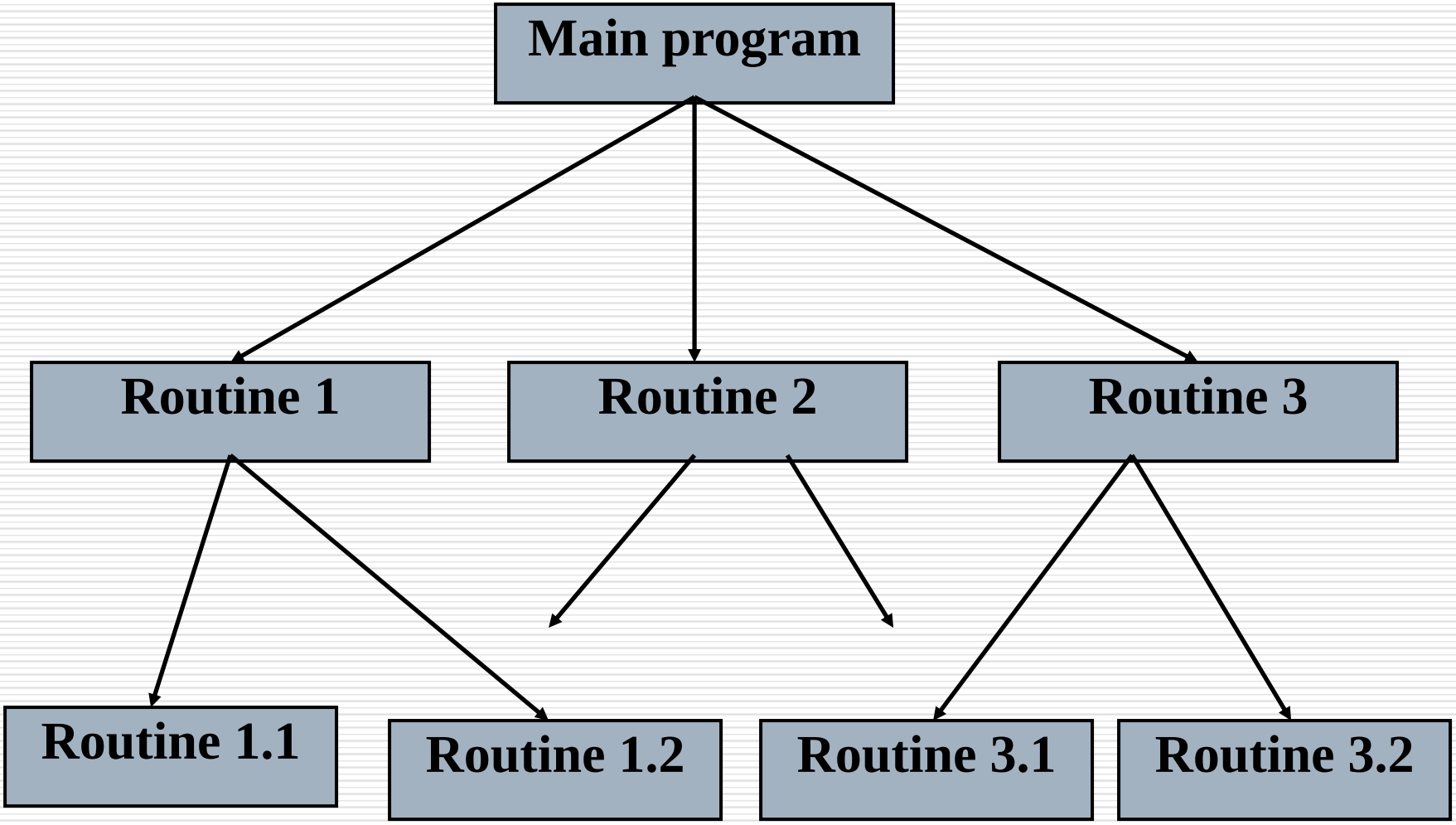
Centralized control model fall into two class depending on whether the controlled sub-system execute sequentially or in parallel.

- The call-return model

  This is familiar top-down subroutine call model.

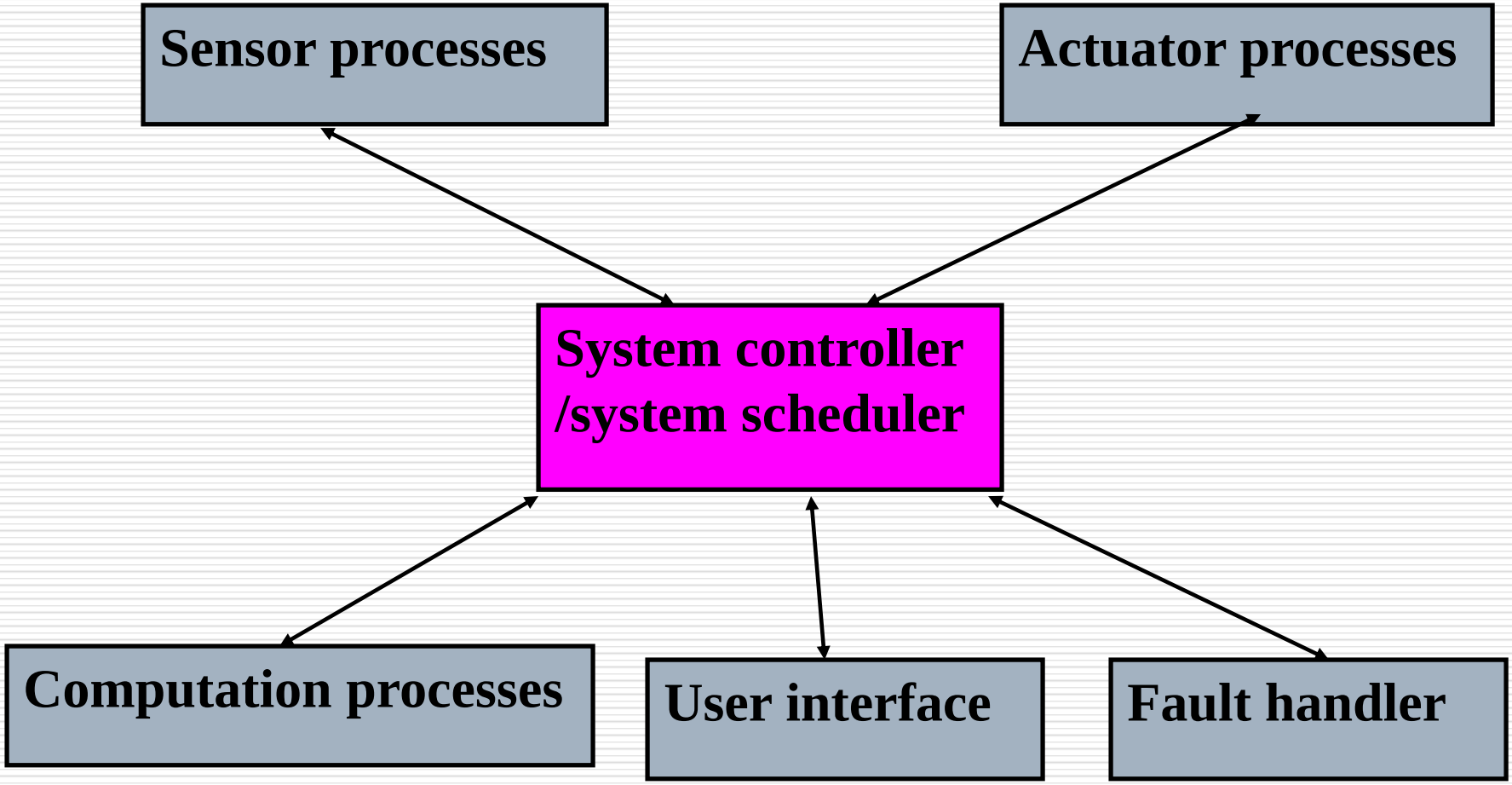
- The manager model

  This is applicable to concurrent system.



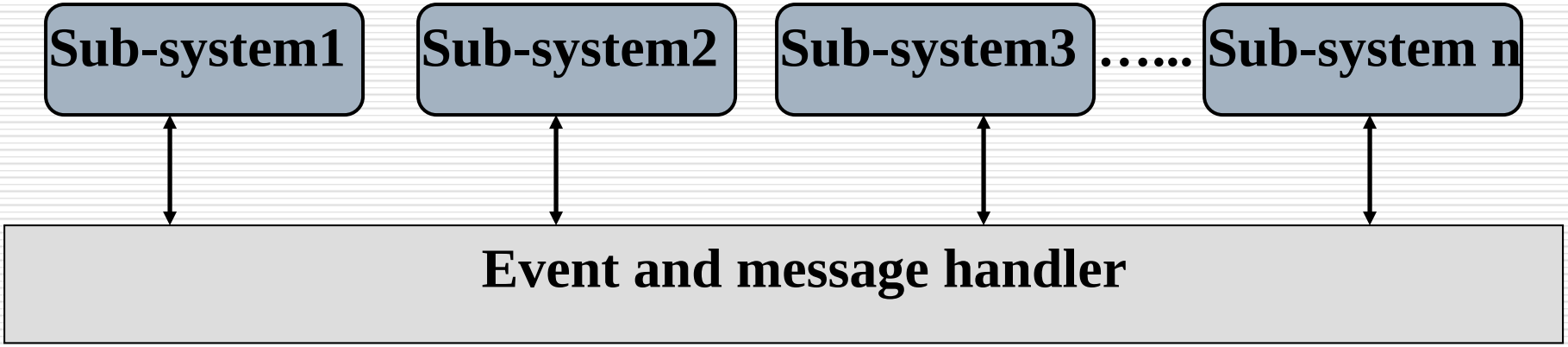
Call-return model in programming language, such as Pascal, C, Ada, …

program dynamics, not program structure



A centralized control model for a real-time system

on event and message

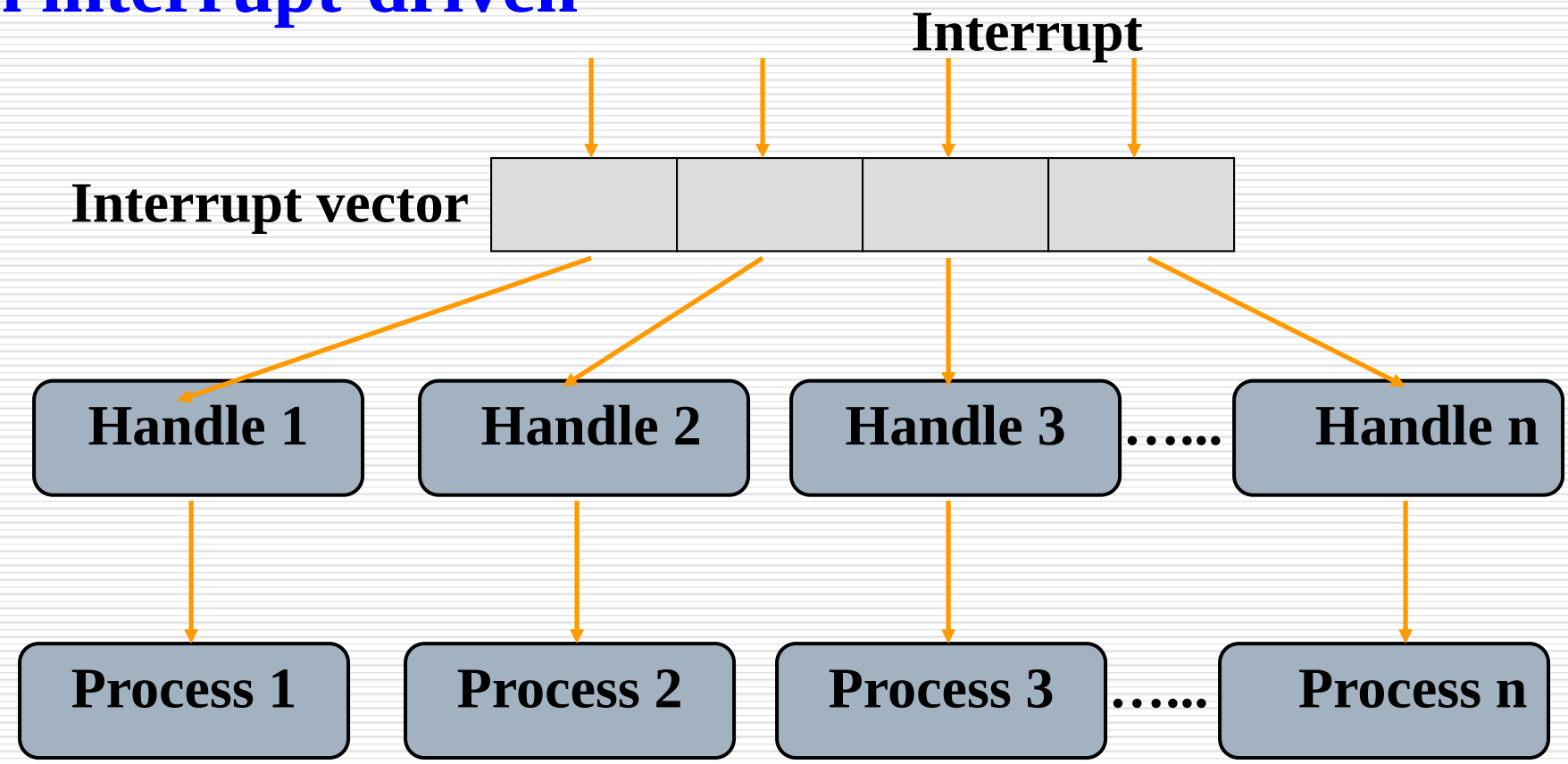


A event broadcast to all sub-system. Any sub-system handle such event and responds to it selectively.

Broadcast model is effective in integrating sub-systems distributed across different computers on a network.

This model may cause conflict when ...

on interrupt-driven



This model allows very fast responses to events, so it is often used in real-time system, but it is complex to program and difficult to validate.

(2) architectural pattern or style

相关概念

- 体系结构风格(Architecture Styles)

  表示软件系统的一种特别的基本结构，以及相关的构造方法

- 设计模式(Design Patterns)

  构造型模式、结构型模式、行为型模式

- 框架(Framework)

  另一种研究和构造软件体系结构的方法，更多的是关于应用领域问题的已建立的系统结构。

相关概念

- 公认的、被多次使用的系统结构被称为结构风格、设计模式、(设计)框架。
- 如果说一门工程技术的成熟表现在其基本设计构件的提出和系统化，那么体系结构的风格、模式、框架就是软件工程中的基本构件。

相关概念

对体系结构风格的理解

- 结构风格以结构组织为特性定义了一个软件系统族，表达了部件以及部件之间的关系。
- 体系结构风格通过组件应用的限制及其与构建有关的组成和设计规则来表现组件和组件之间的关系。
- 体系结构风格表示了软件系统的一种特别的基本结构，以及相关的构造方法。
- 体系结构风格应该使一些对软件构成带有整体性、普遍性、一般性的结构和结构关系的方法。在设计中，遵循这些风格的构成原则，对软件的开发和维护十分有益。

相关概念

广泛提及的体系结构风格：

- 管道和过滤器 （Pipe and Filters）
- 数据抽象或对象（Data Abstraction and Object Orientted）
- 隐式调用/消息（Event-Based/Implicit Invocation）
- 层次（Layered System）
- 仓库（Repositories）
- 解释器（Interpreters）
- 过程控制（Process Control）
- 分布式系统（Distributed System）
- 客户/服务器（Client/Server）
- 主程序/子程序（Main/Sub Programs）
- 状态转换（State Transition）
- 专用领域（Domain Specific Styles）

最基本的模式

单机

多机

网络
- 局域网
- Internet
- Web

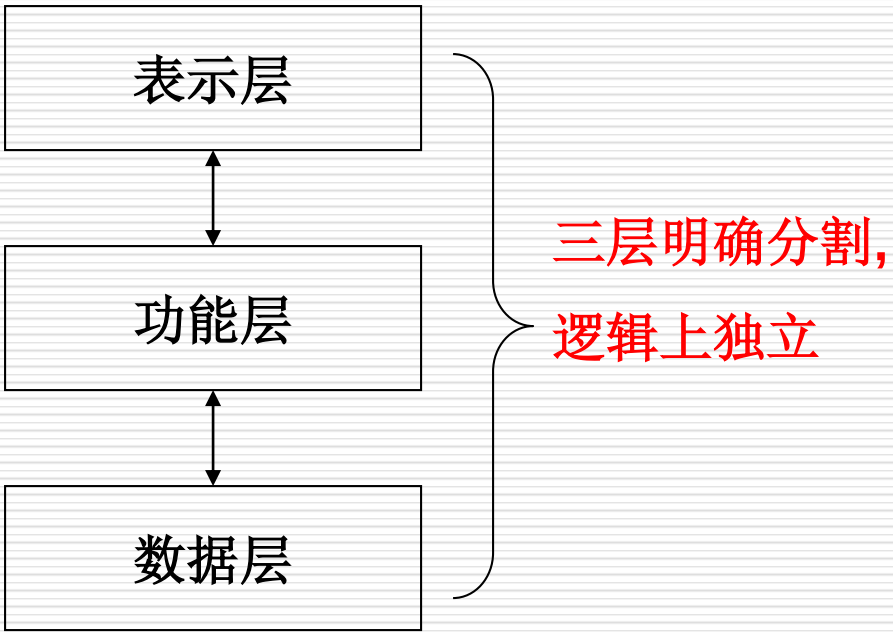
集中式　　银行系统

分布式　　火车票预定系统

客户机/服务器体系结构

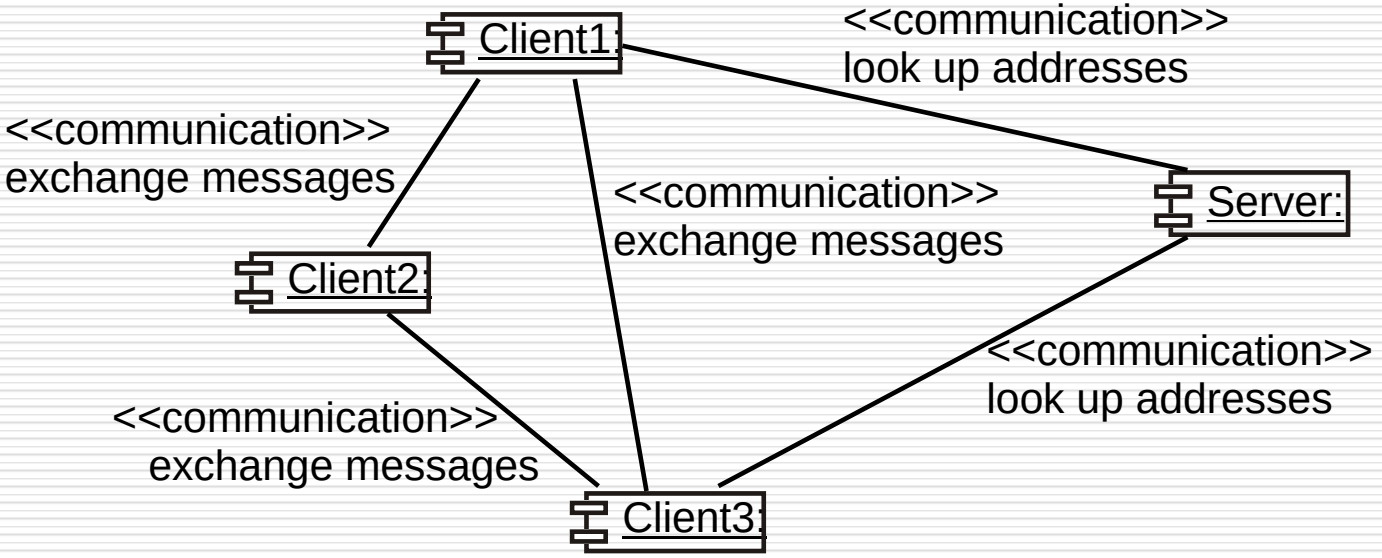
应用系统的组成：

- 显示逻辑部分(表示层):实现与用户交互
- 应用处理部分(功能层):进行具体运算和数据处理
- 数据管理部分(数据层):对数据库中数据进行查询、修改、更新等任务

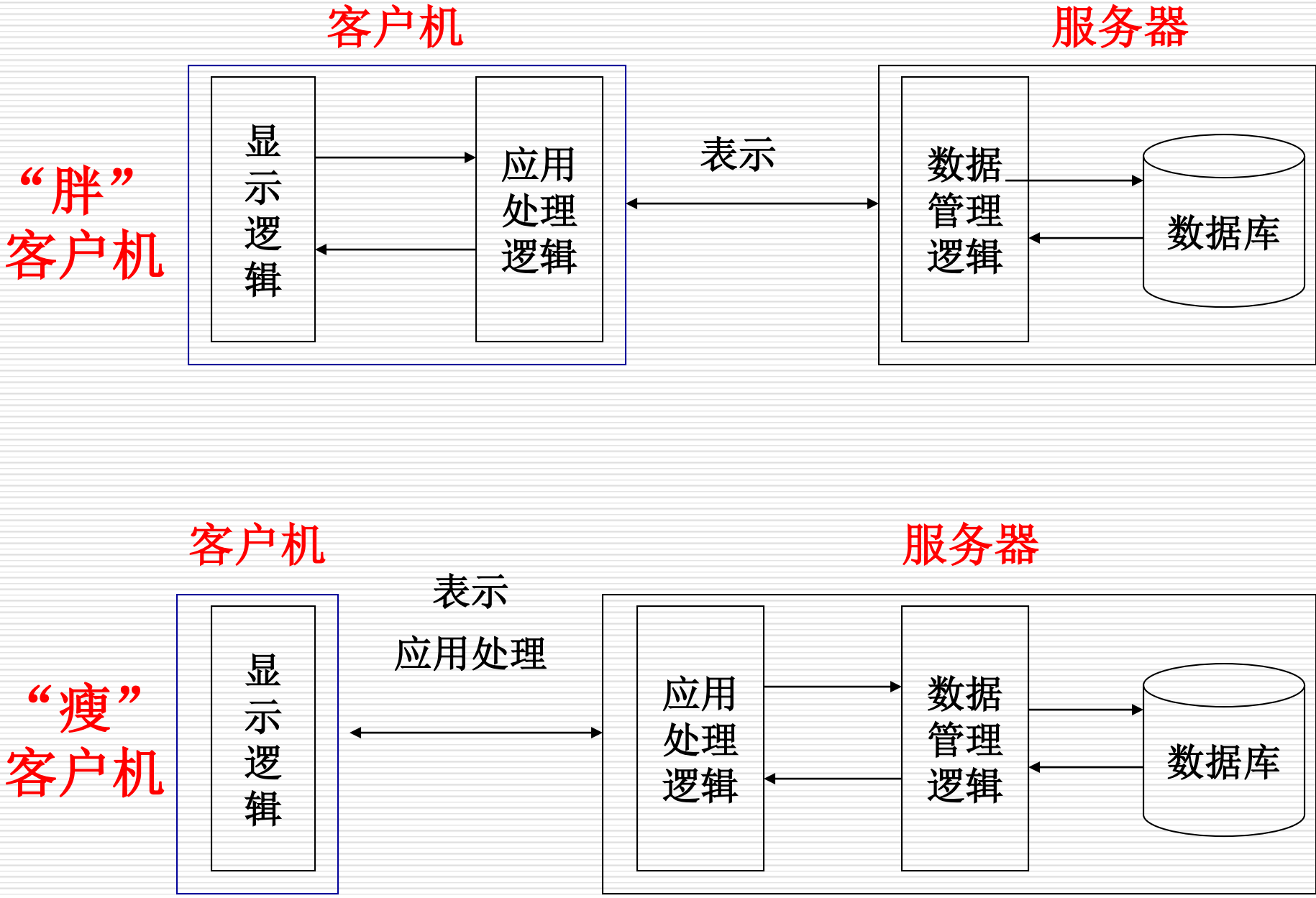


应用分层

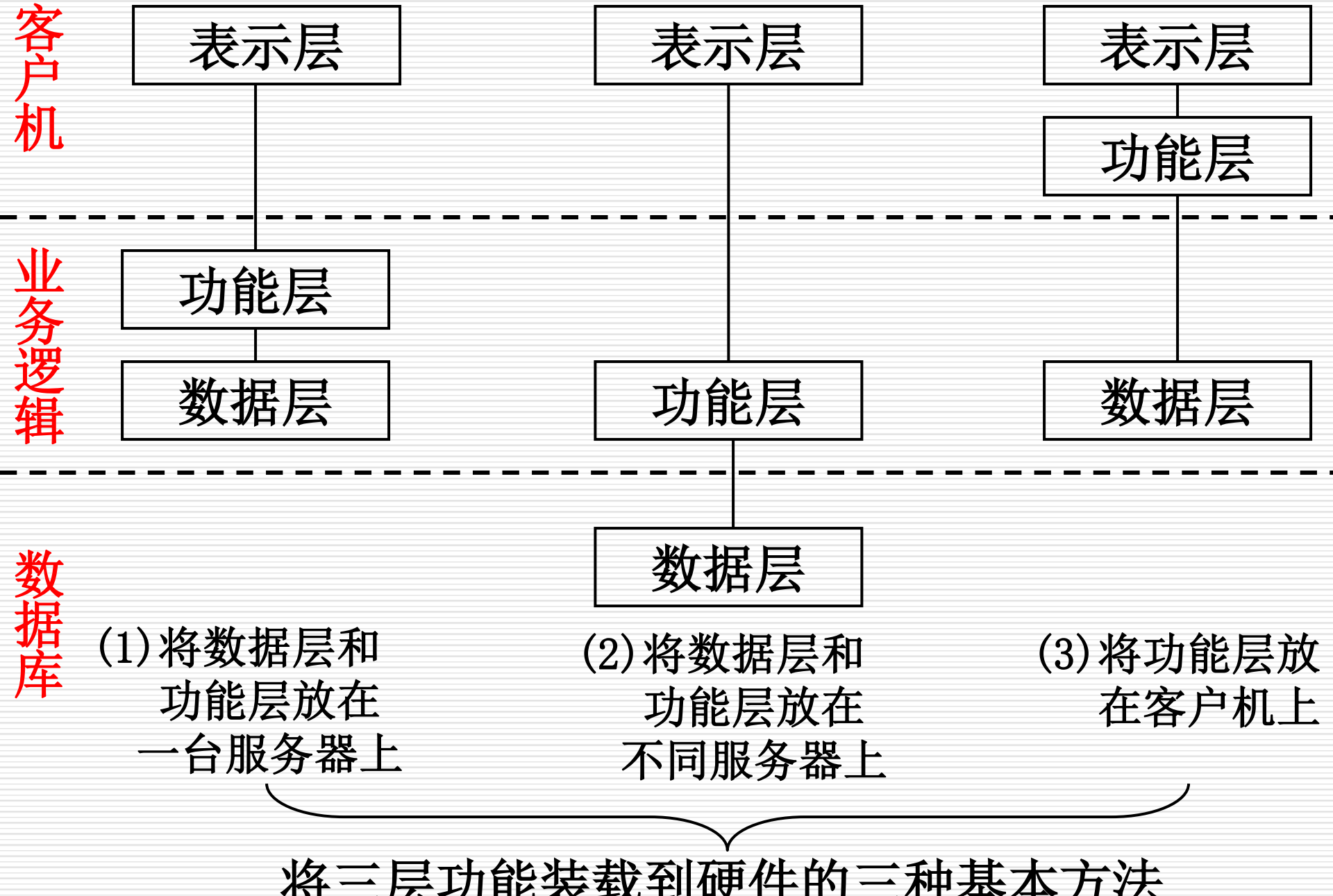
An example of a distributed system



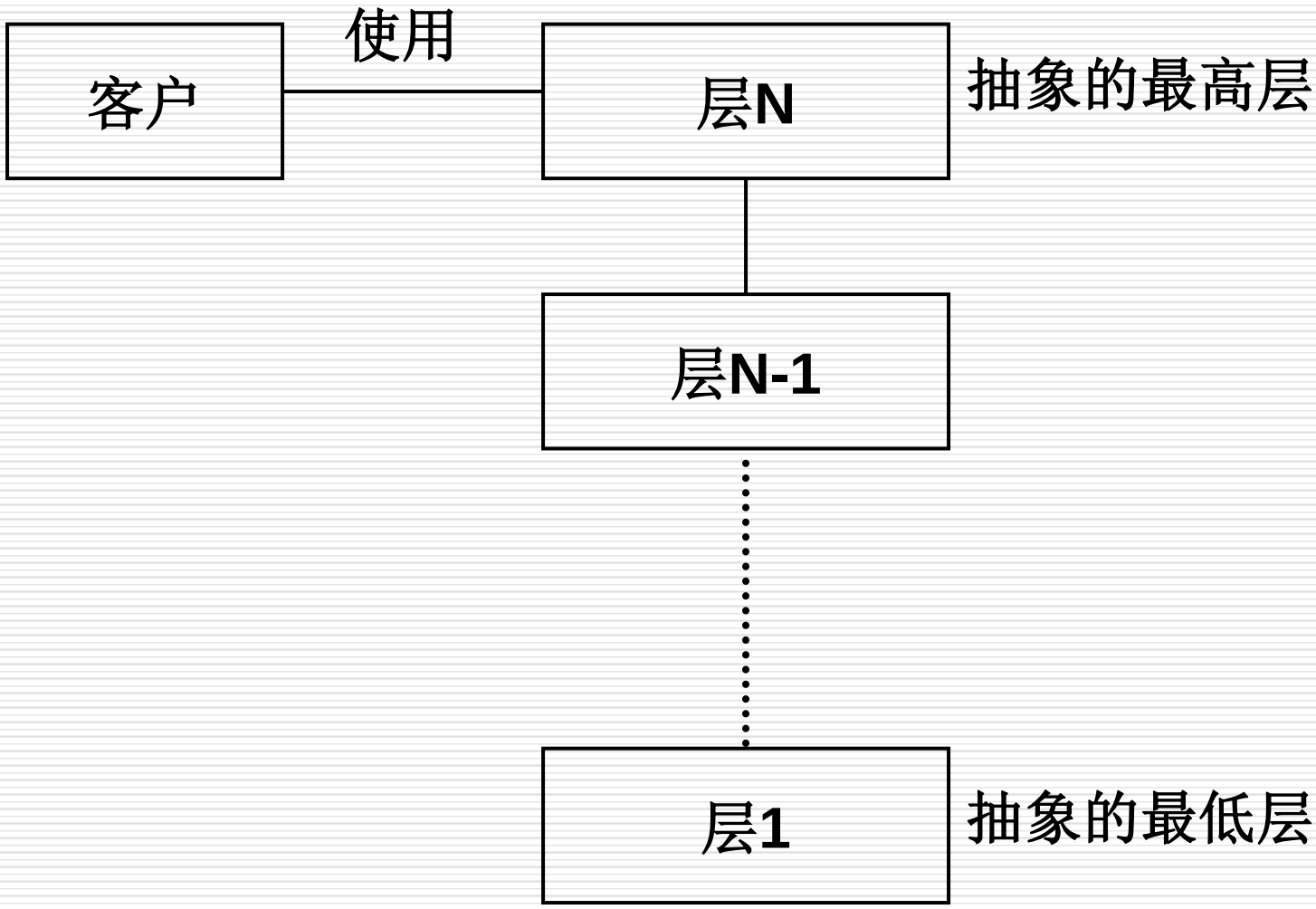
瘦客户机和胖客户机

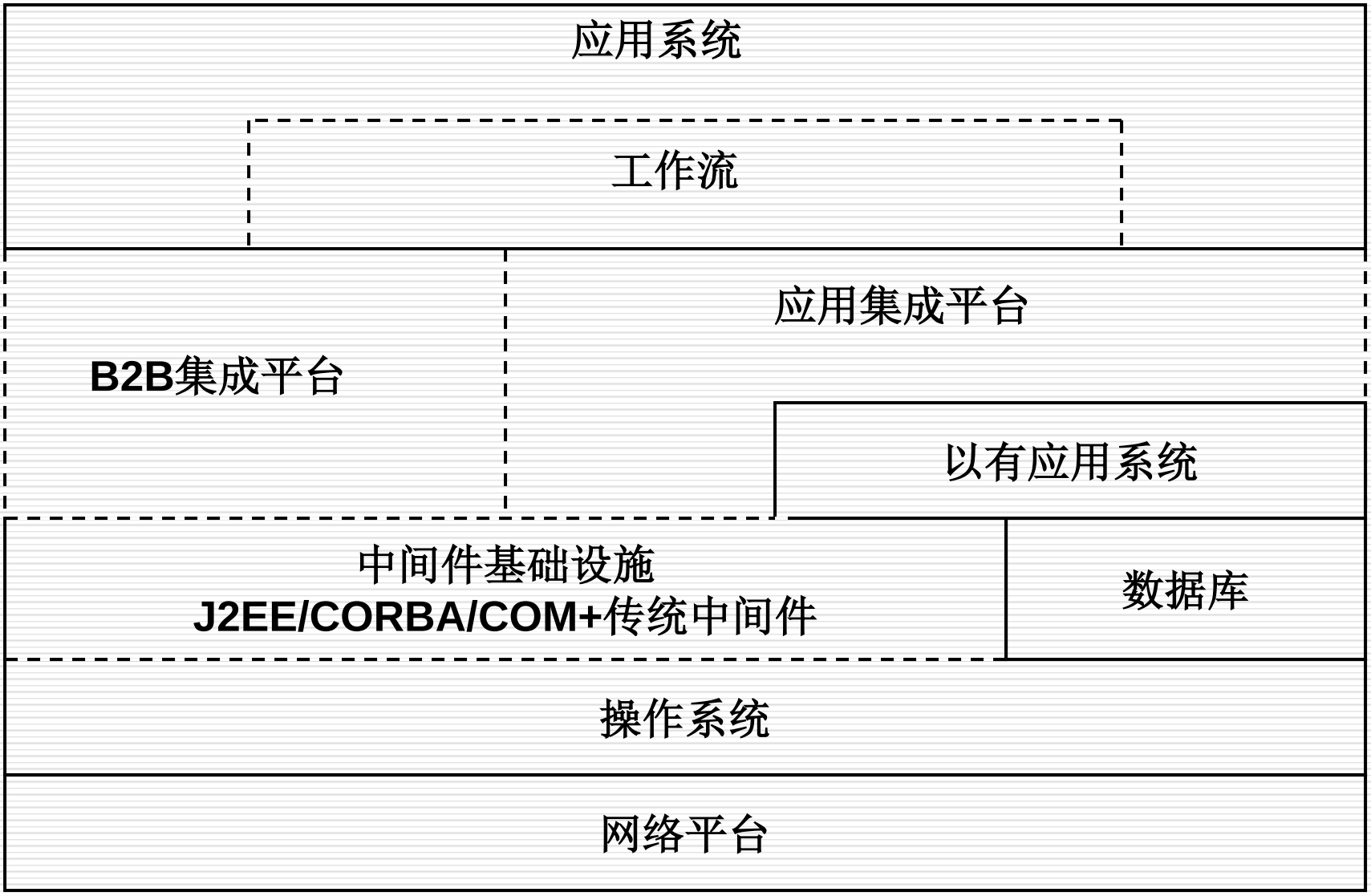

客户机
服务器
“胖”
客户机
显示逻辑
应用处理逻辑
表示
数据管理逻辑
数据库
客户机
服务器
“瘦”
客户机
显示逻辑
表示
应用处理
应用处理逻辑
数据管理逻辑
数据库

三层B/S的基本硬件结构



层次模型的结构关系

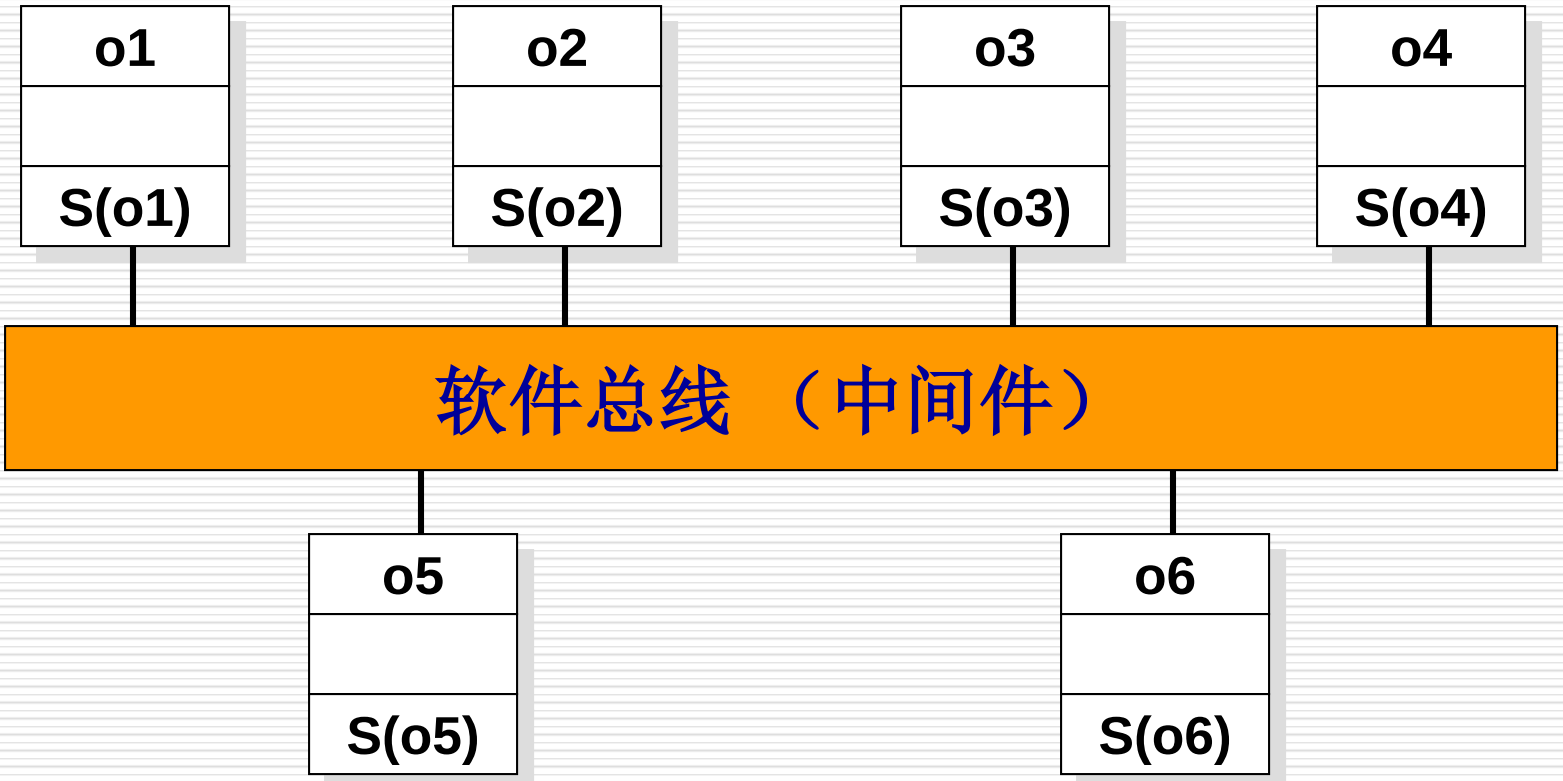




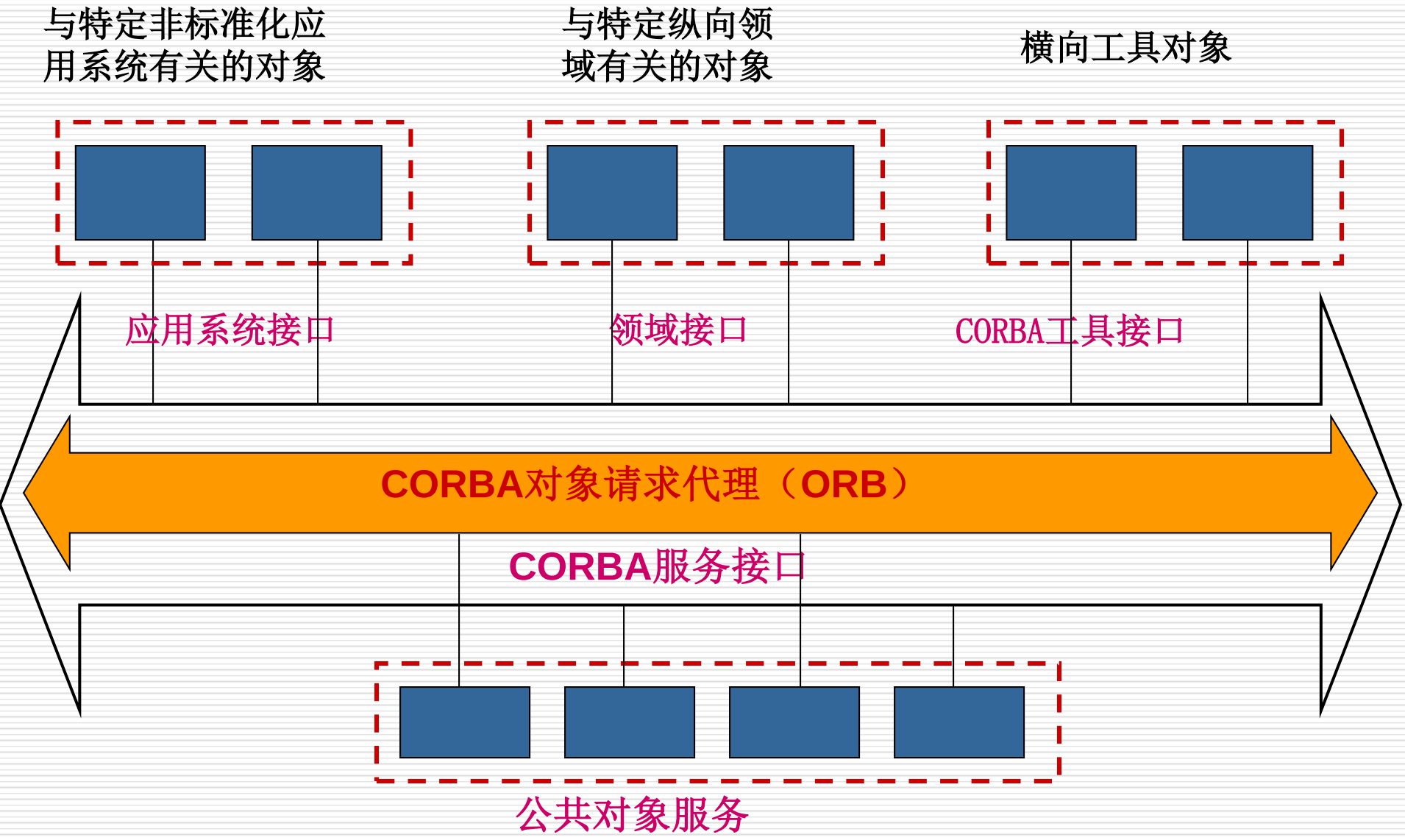
面向电子商务的应用体系结构图

分布式对象体系结构

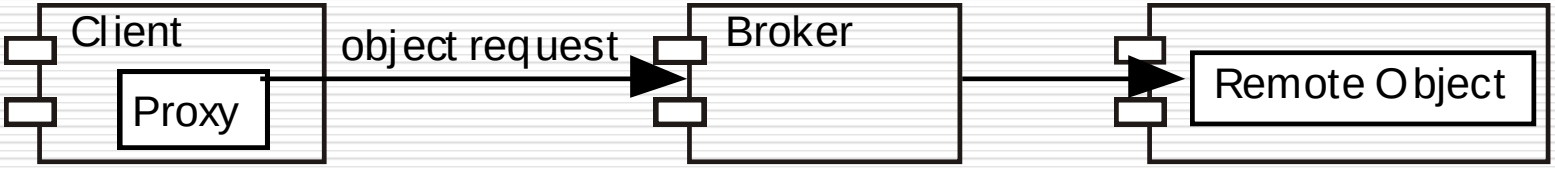
- 基本系统组件是对象，提供一组服务，对外给出服务的接口
- 对象之间不存在客户机与服务器的界限，接受服务者扮演客户机角色，提供服务者就是服务器
- 对象可能分布在网络的多台计算机上，通过中间件相互通信



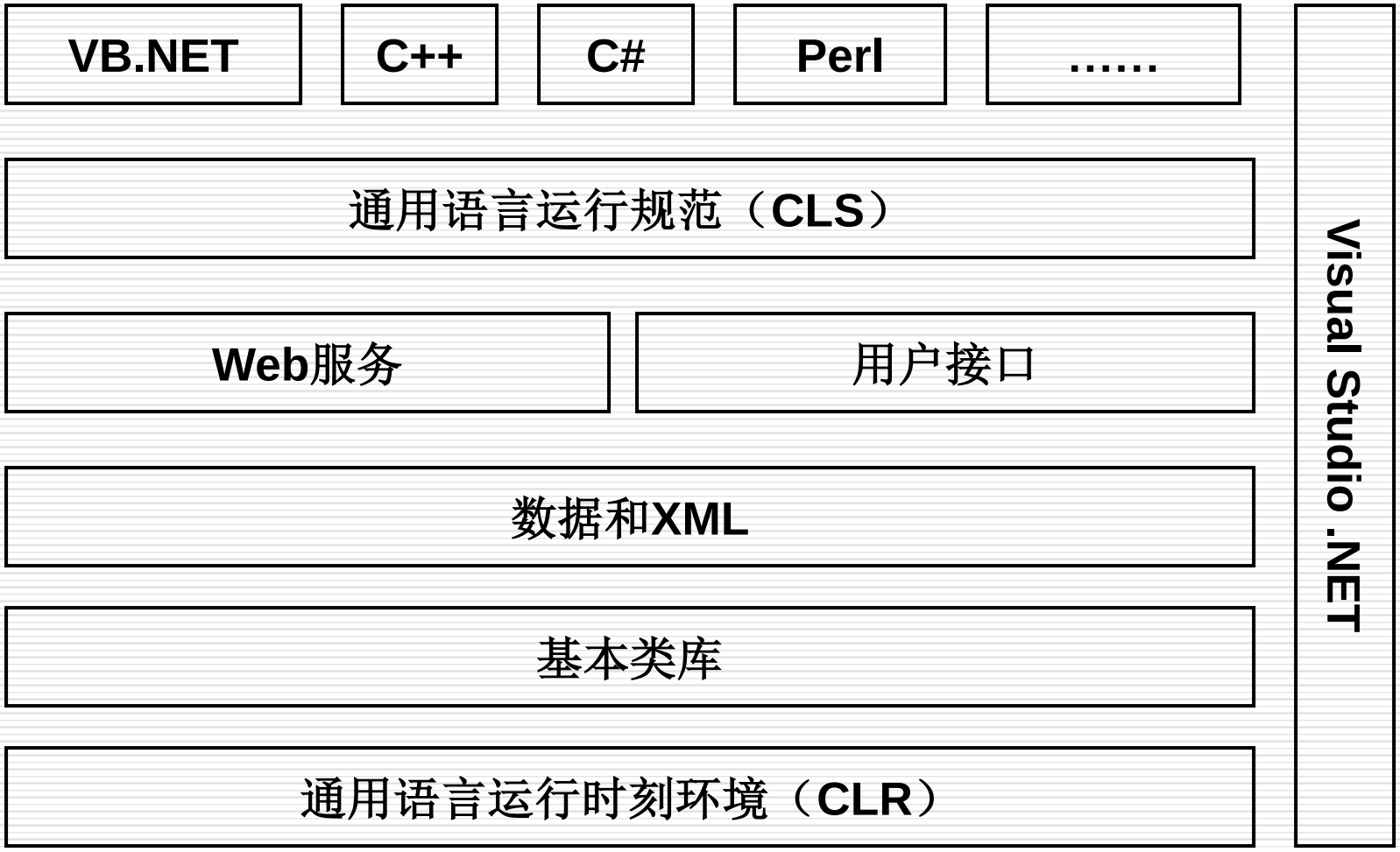
OMG体系结构和服务参考模型



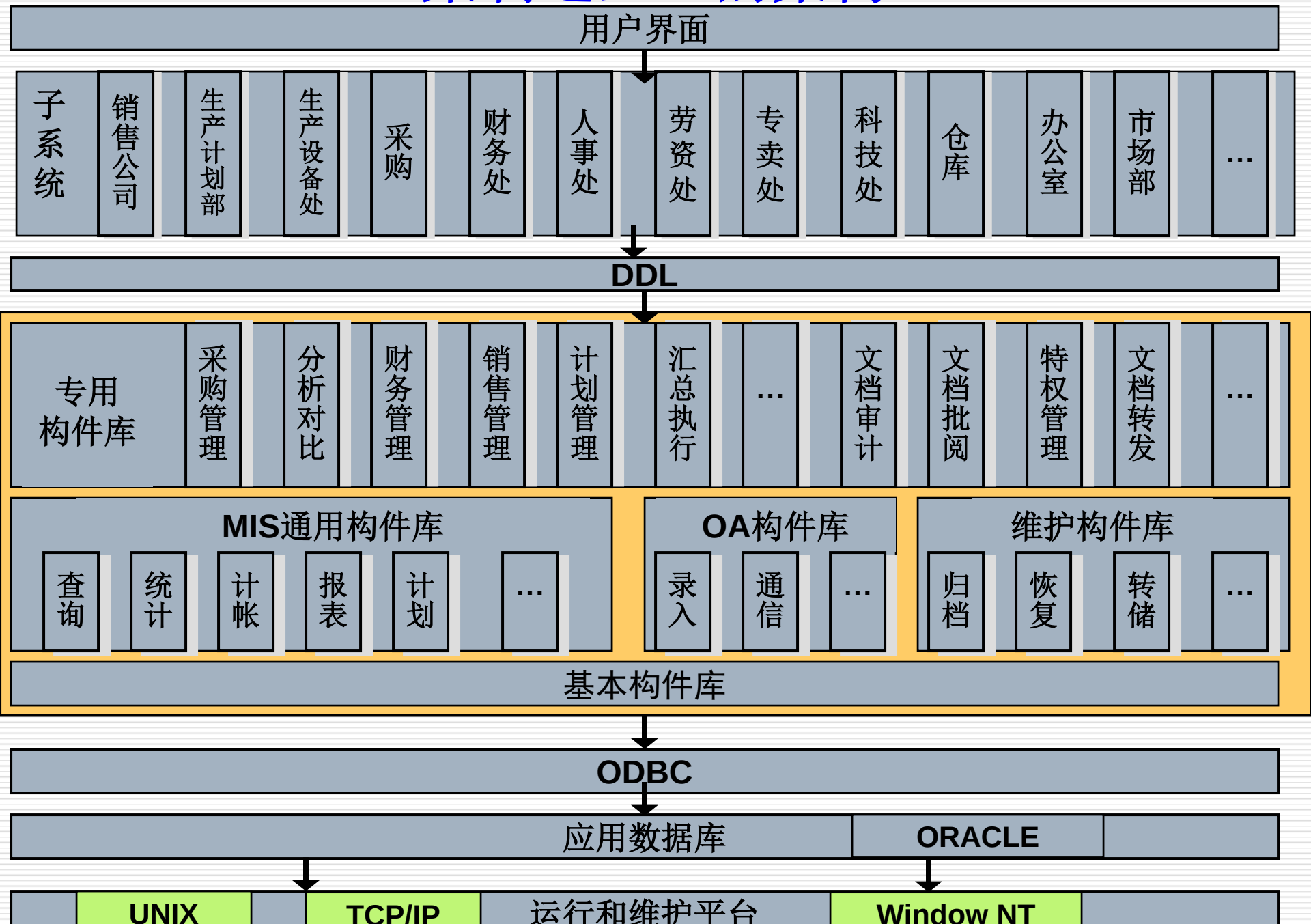
Example of a Broker system



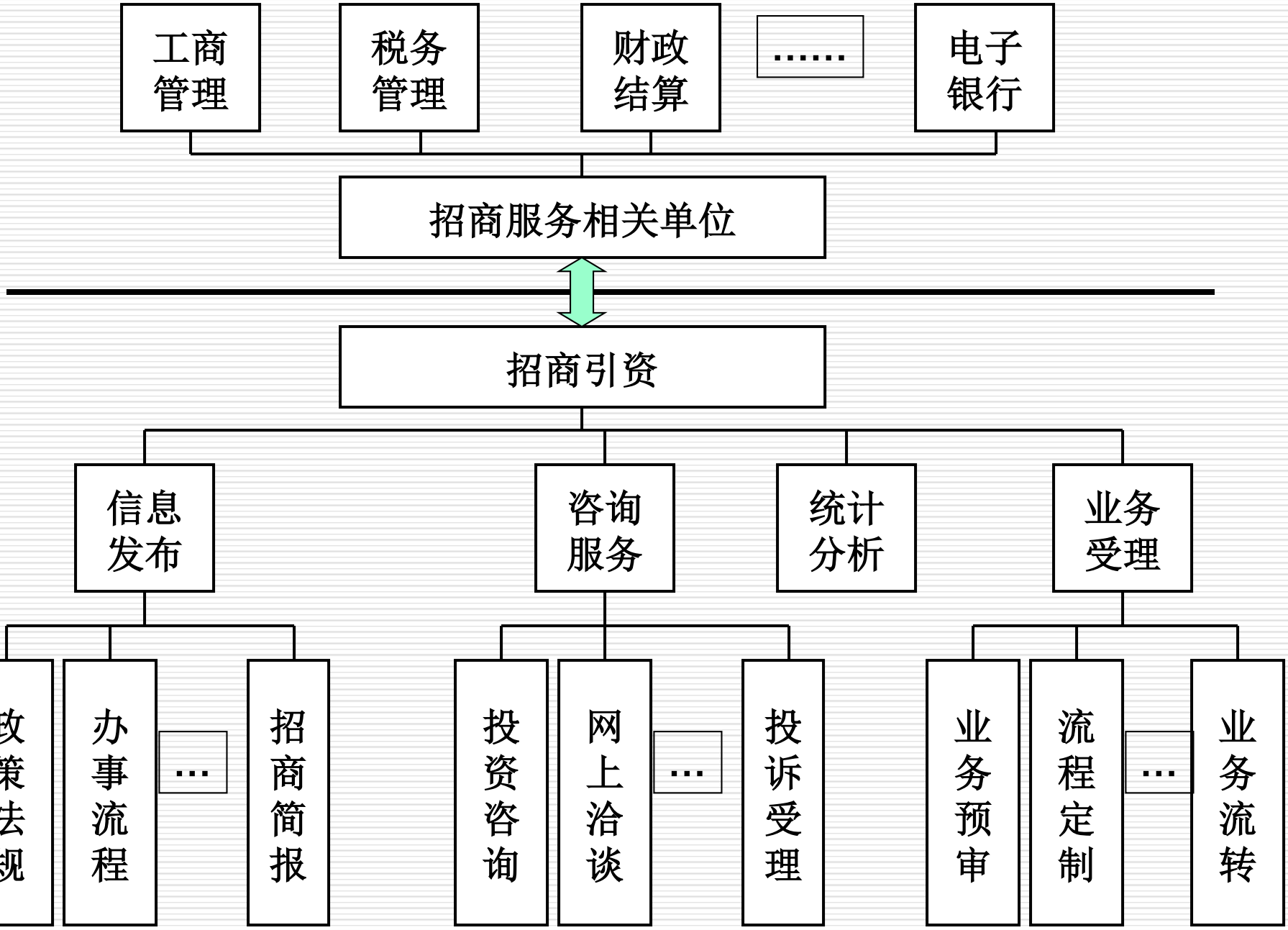
.NET体系


VB.NET
C++
C#
Perl
……
通用语言运行规范（CLS）
Web服务
用户接口
数据和XML
基本类库
通用语言运行时刻环境（CLR）
Visual Studio .NET

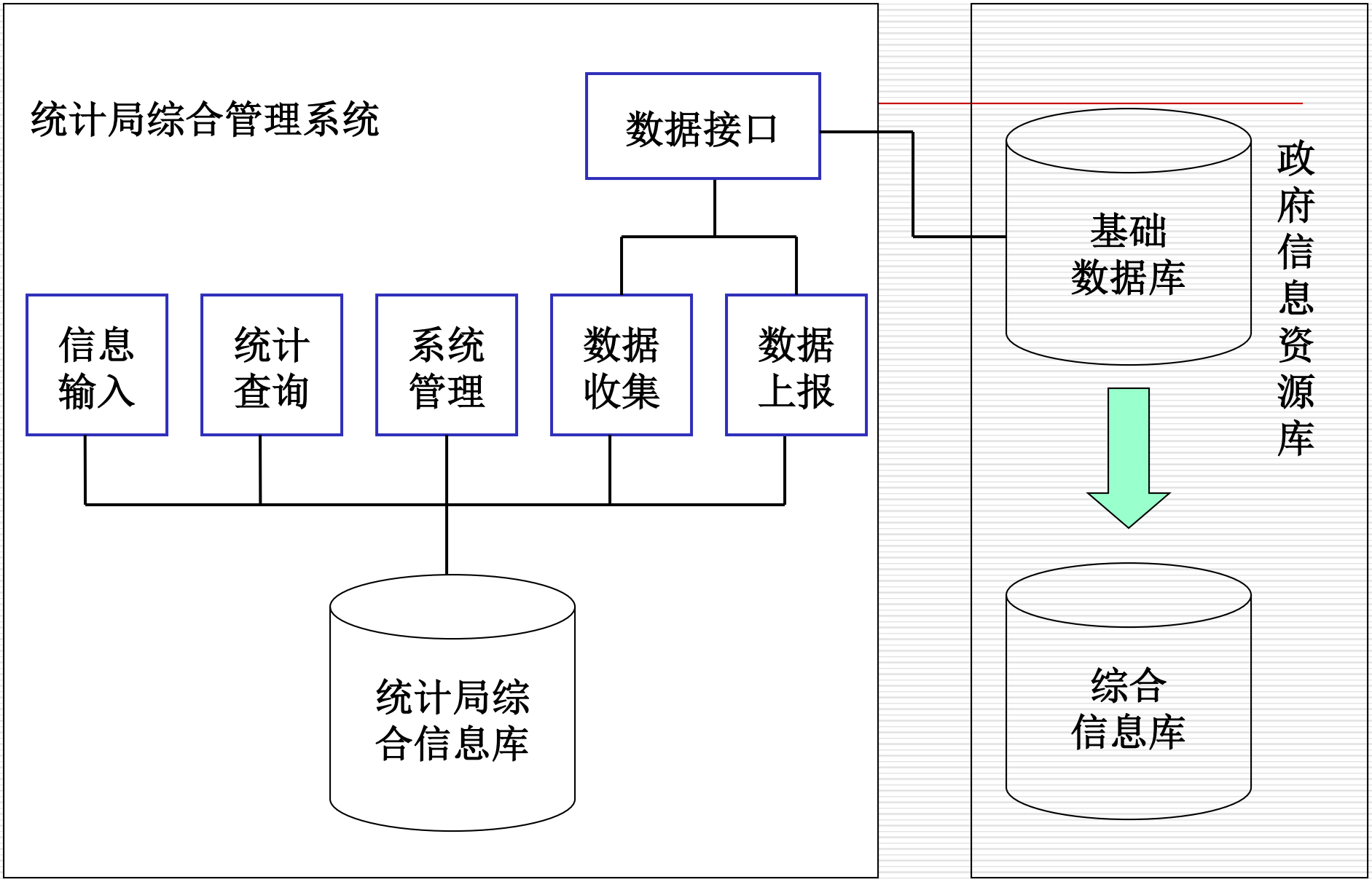
某制造企业的架构


用户界面
子系统
销售公司
生产计划部
生产设备处
采购
财务处
人事处
劳资处
专卖处
科技处
仓库
办公室
市场部
…
DDL
专用构件库
采购管理
分析对比
财务管理
销售管理
计划管理
汇总执行
…
文档审计
文档批阅
特权管理
文档转发
…
MIS通用构件库
查询
统计
计帐
报表
计划
…
OA构件库
录入
通信
…
维护构件库
归档
恢复
转储
…
基本构件库
ODBC
应用数据库
ORACLE
UNIX
TCP/IP
运行和维护平台
Window NT

领域框架示例之一：政府招商引资

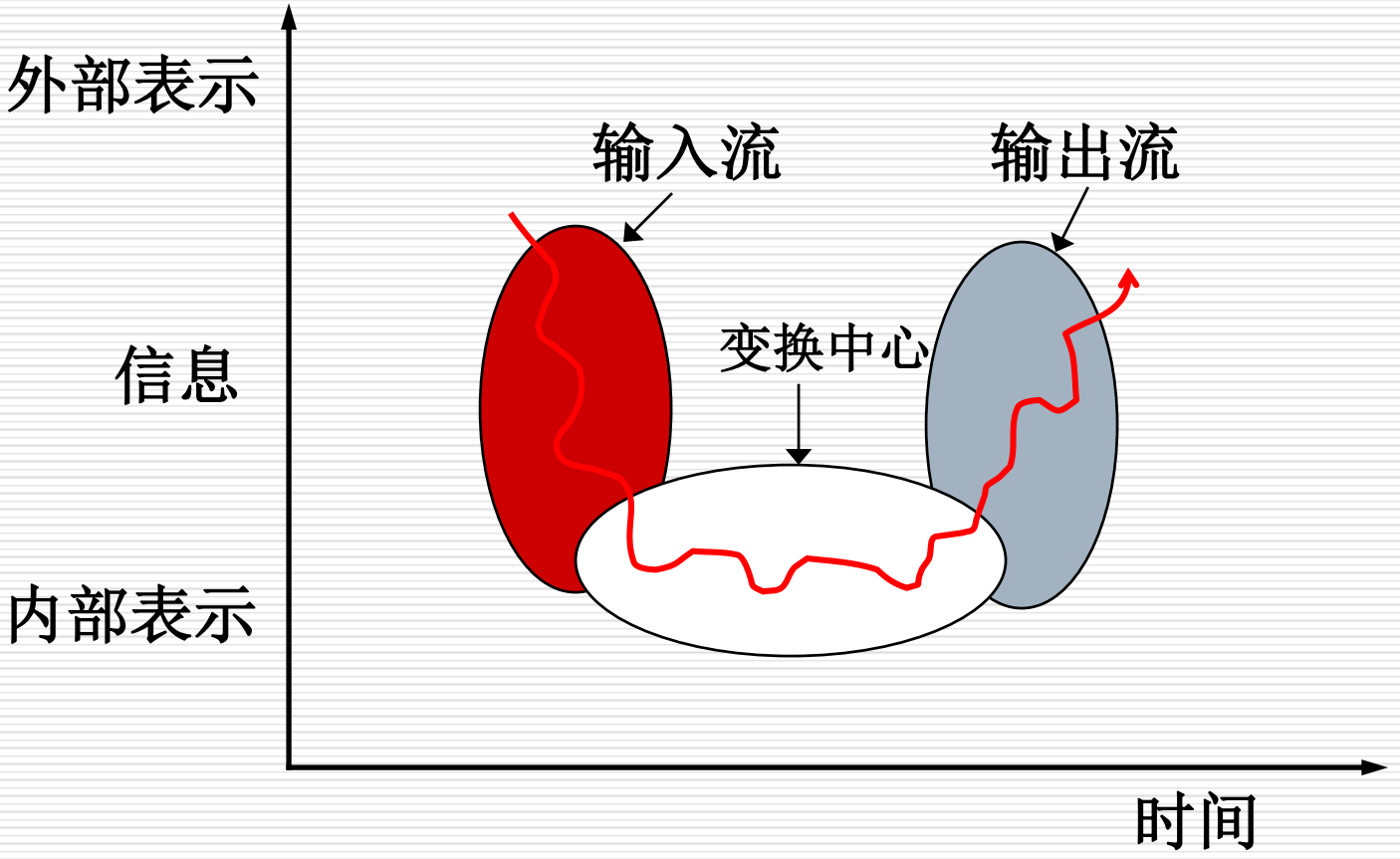


领域框架示例之二：统计局



(3) 由DFD图导出总体结构

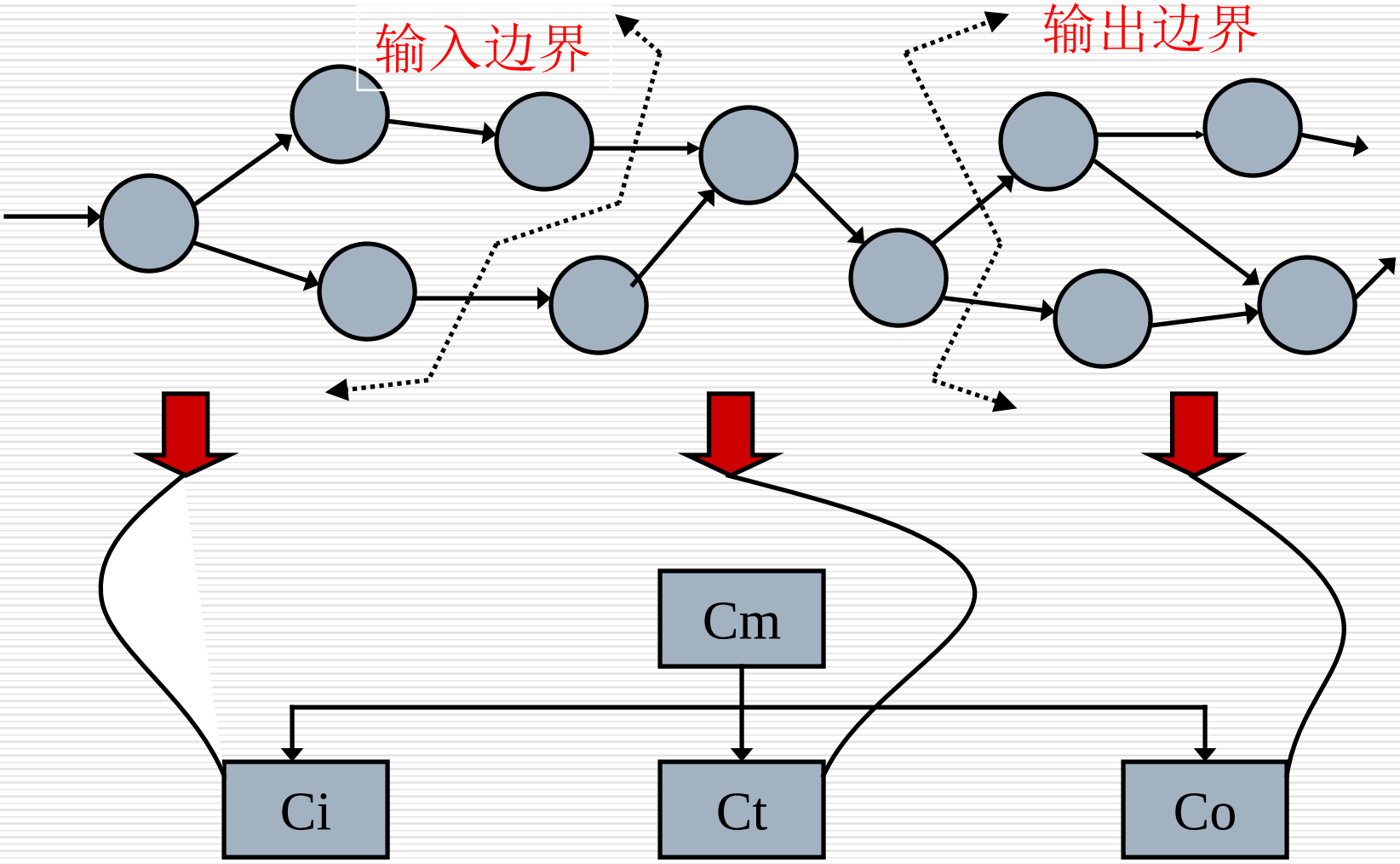
原理



I → P → O

自顶向下，逐步细化

变换原理



基本步骤

✓ 步骤：

最顶层的控制模块Cm协调下述从属的控制功能：

（1）输入信息处理控制模块Ci，协调对所有输入数据的接收；

（2）变换中心控制模块Ct，管理对内部形式的数据的所有操作；

（3）输出信息控制模块Co，协调输出信息的产生过程。

✓ 掌握层次，不断进行

✓使用设计度量和启发式规则对得到的软件结构进一步精化。

变换举例

一个汽车数字仪表板的设计

假设仪表板的功能如下：

（1）通过模／数（A／D）转换实现传感器和微处理机接口；

（2）在发光二极管（LCD）面板上显示数据；

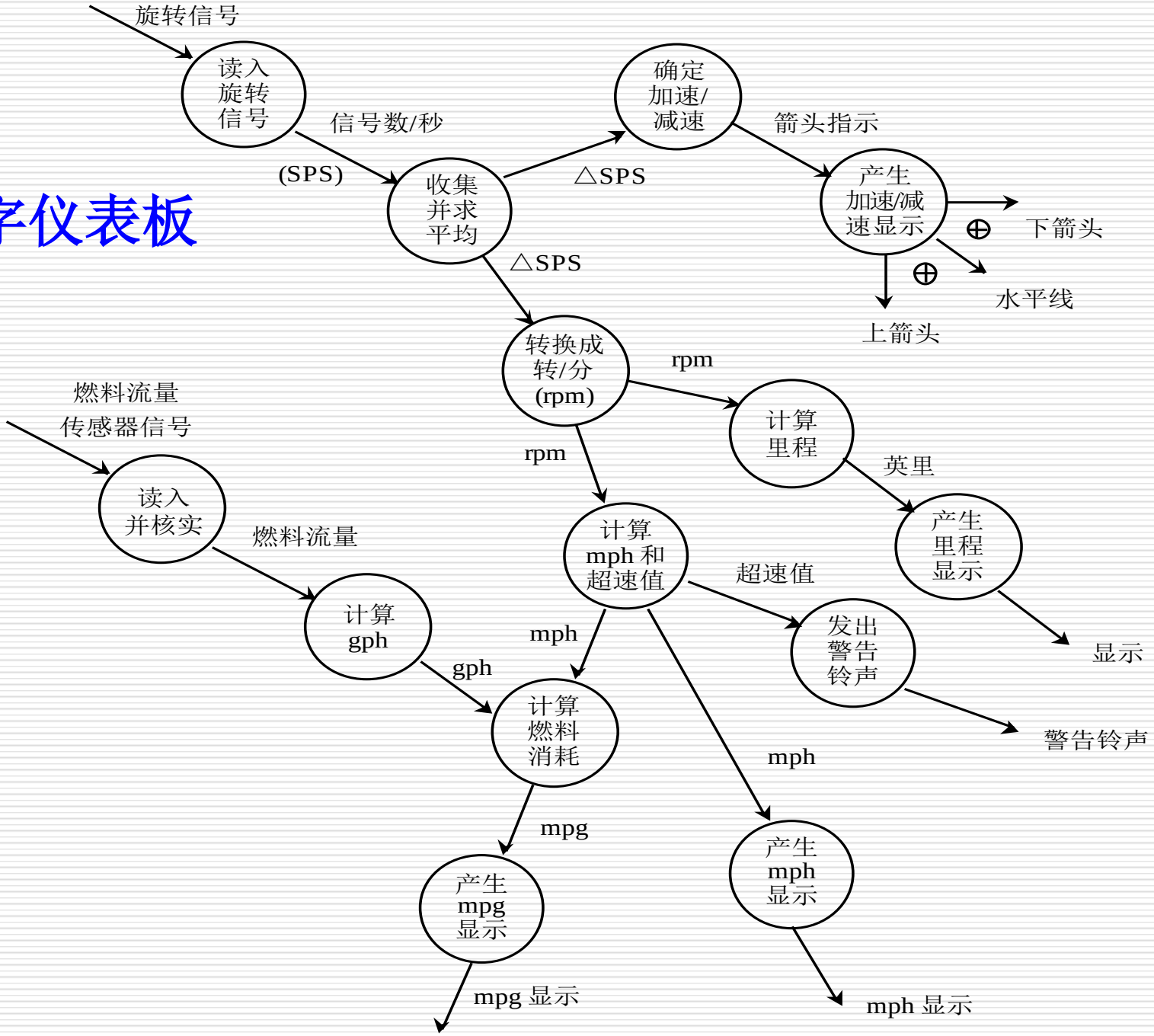
（3）指示每小时英里数（mph），行驶的里程，每加仑油行驶的英里数（mpg）等等；

（4）指示加速或减速；

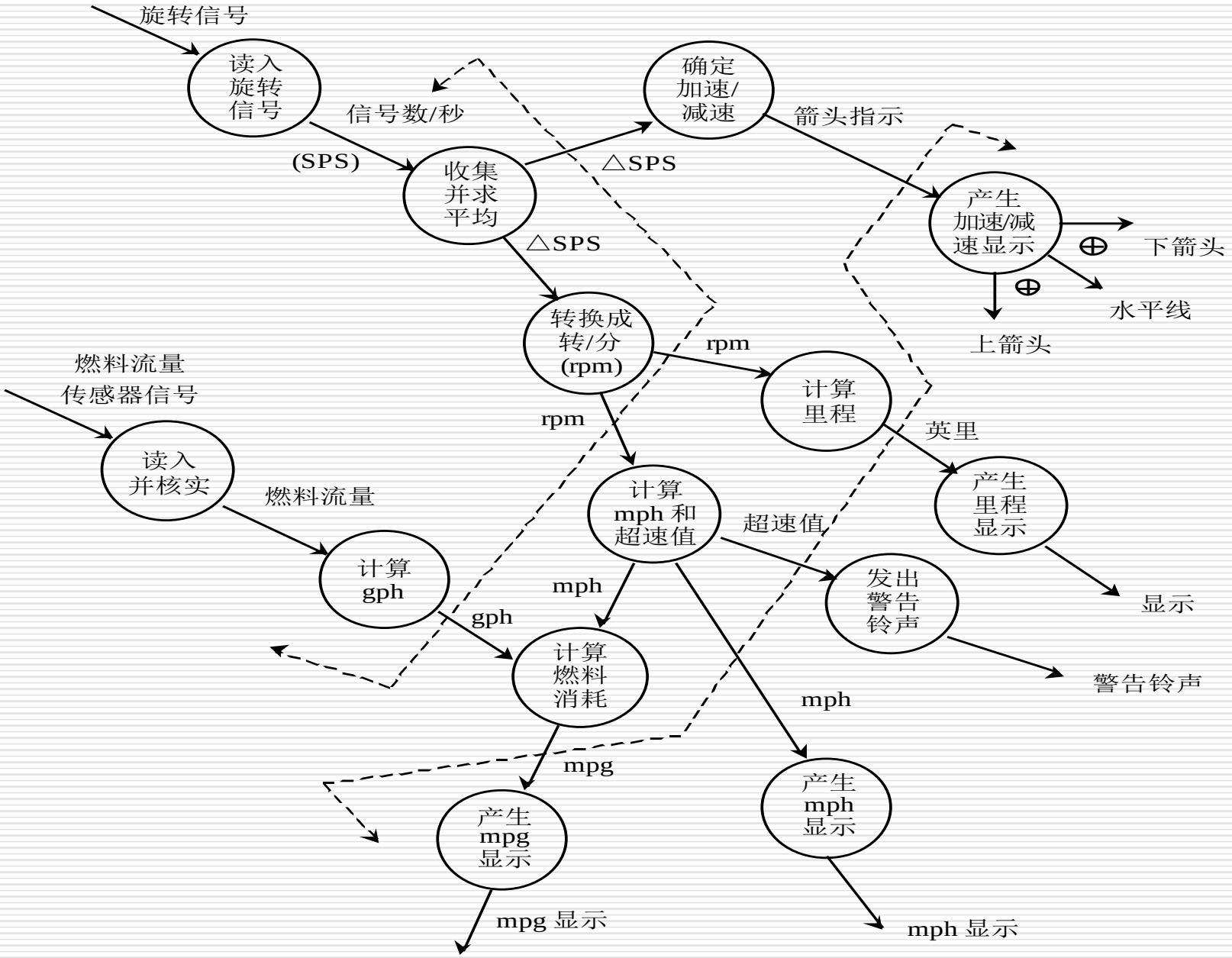
（5）超速警告：如果车速超过55英里／小时，则发出超速警告铃声。

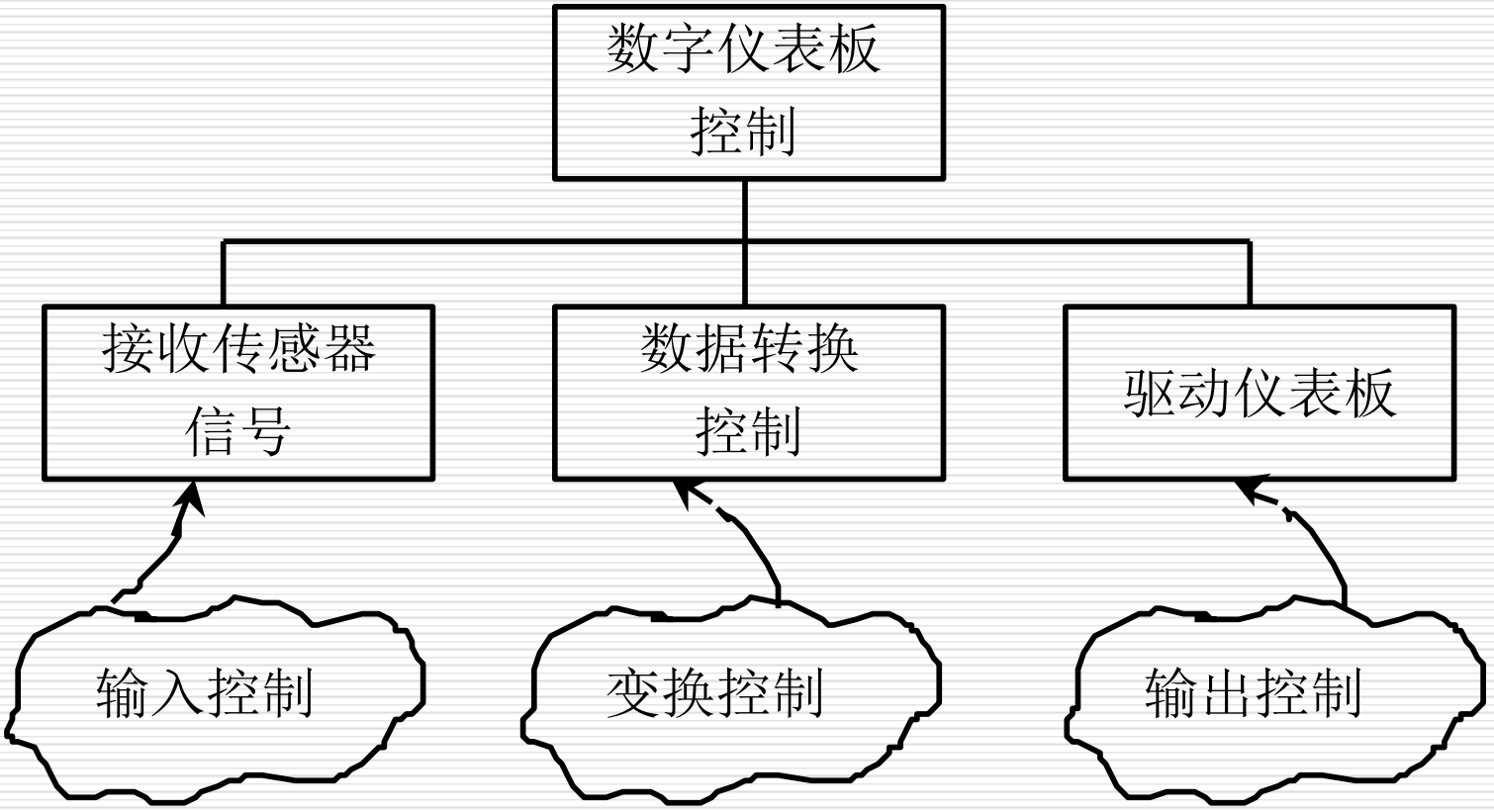
在软件需求分析阶段，应该对上述每项性能和其它要求进行全面的分析，并建立起相应的文档资料，得出数据流图。

汽车数字仪表板 DFD图



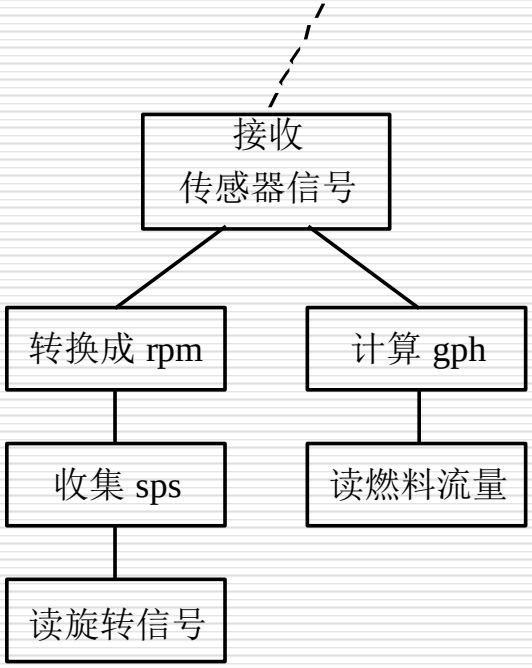
确定输入流和输出流的边界，从而孤立出变换中心

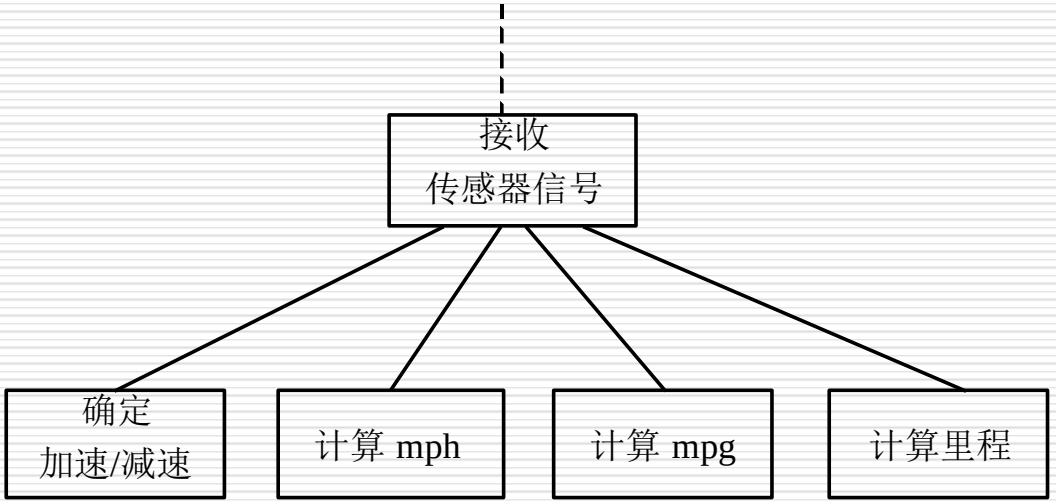




第一级分解的结果

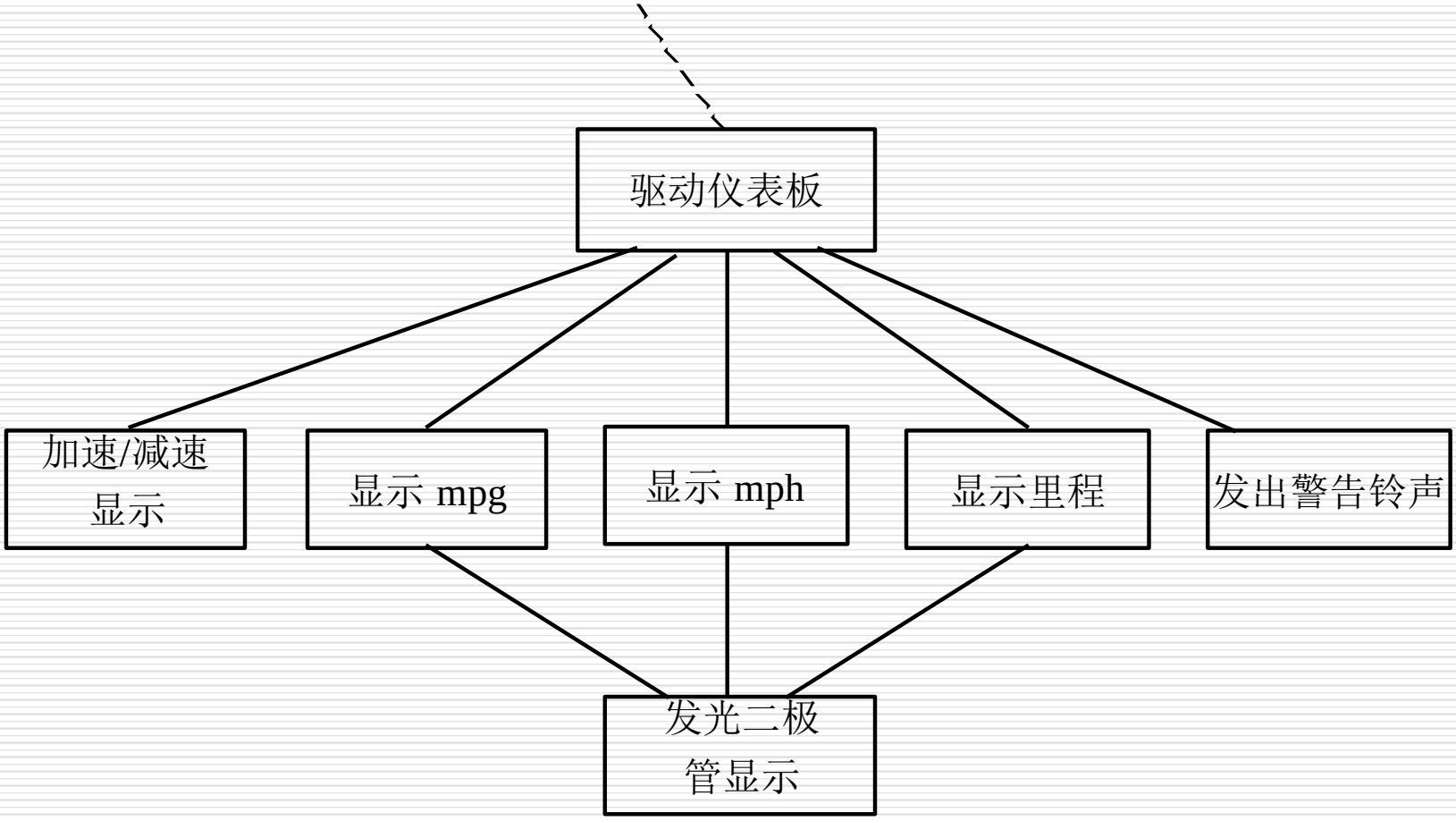
第二级分解的结果：



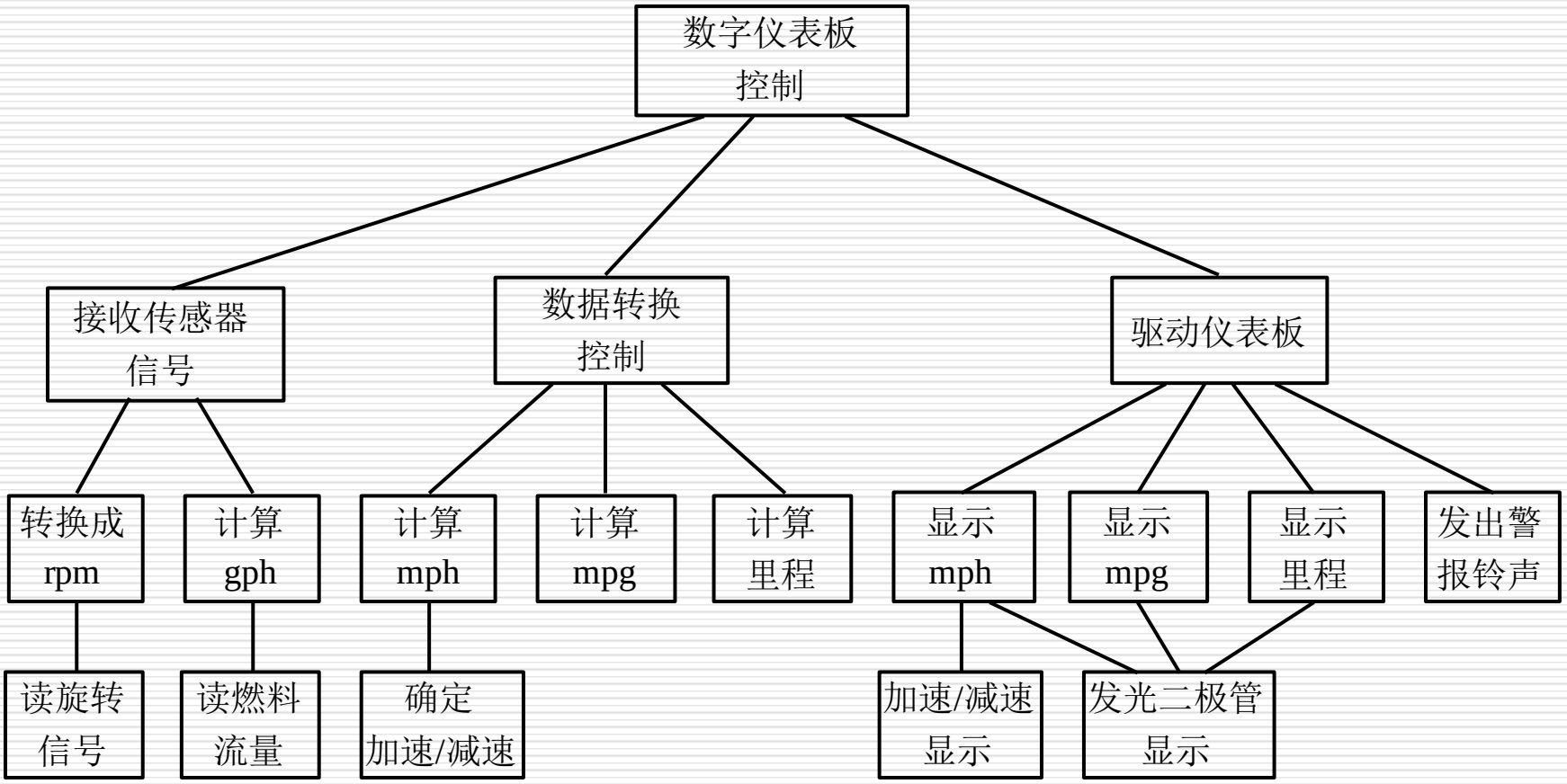


未经精化的输入结构

未经精化的变换结构

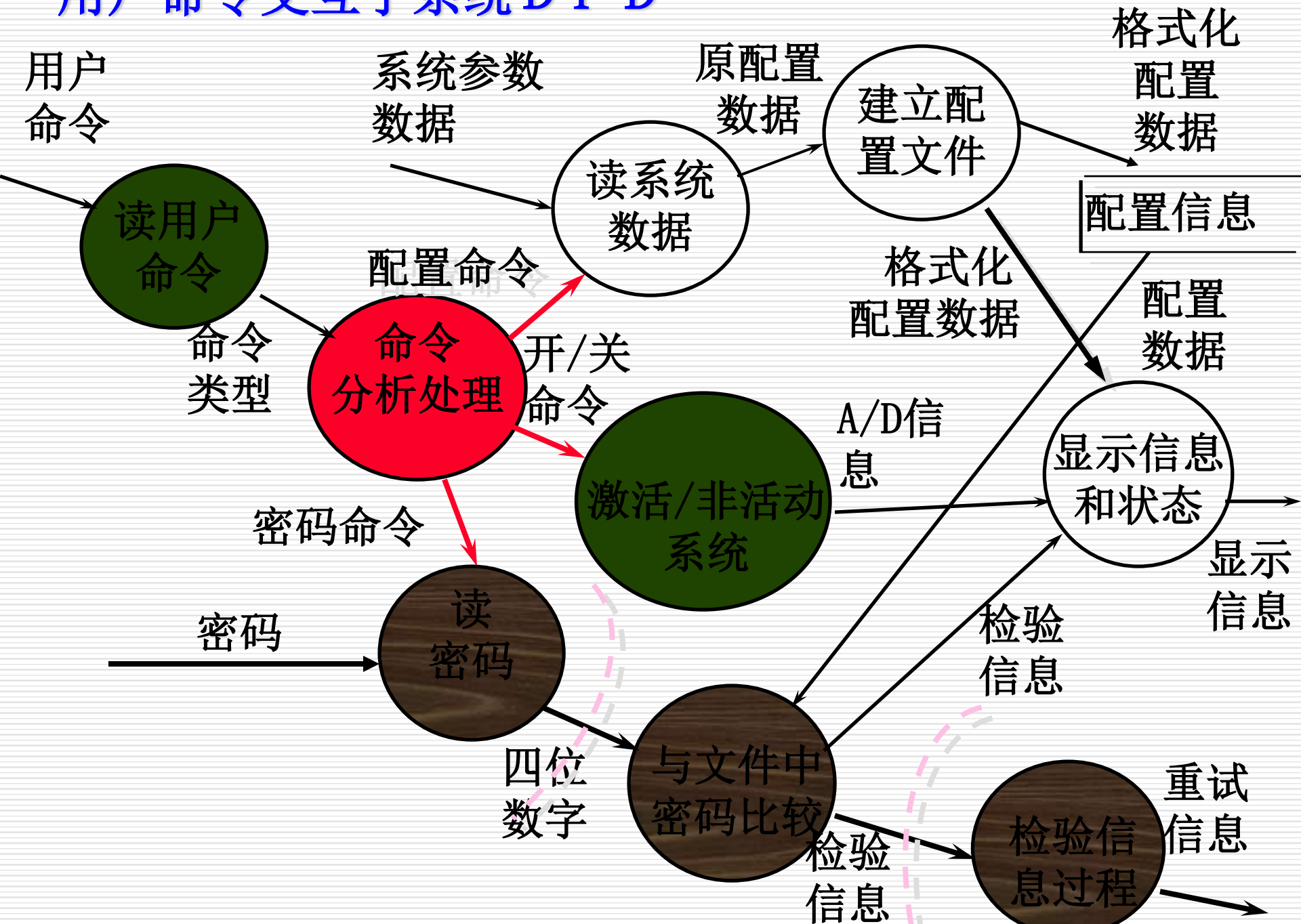


未经精化的输出结构

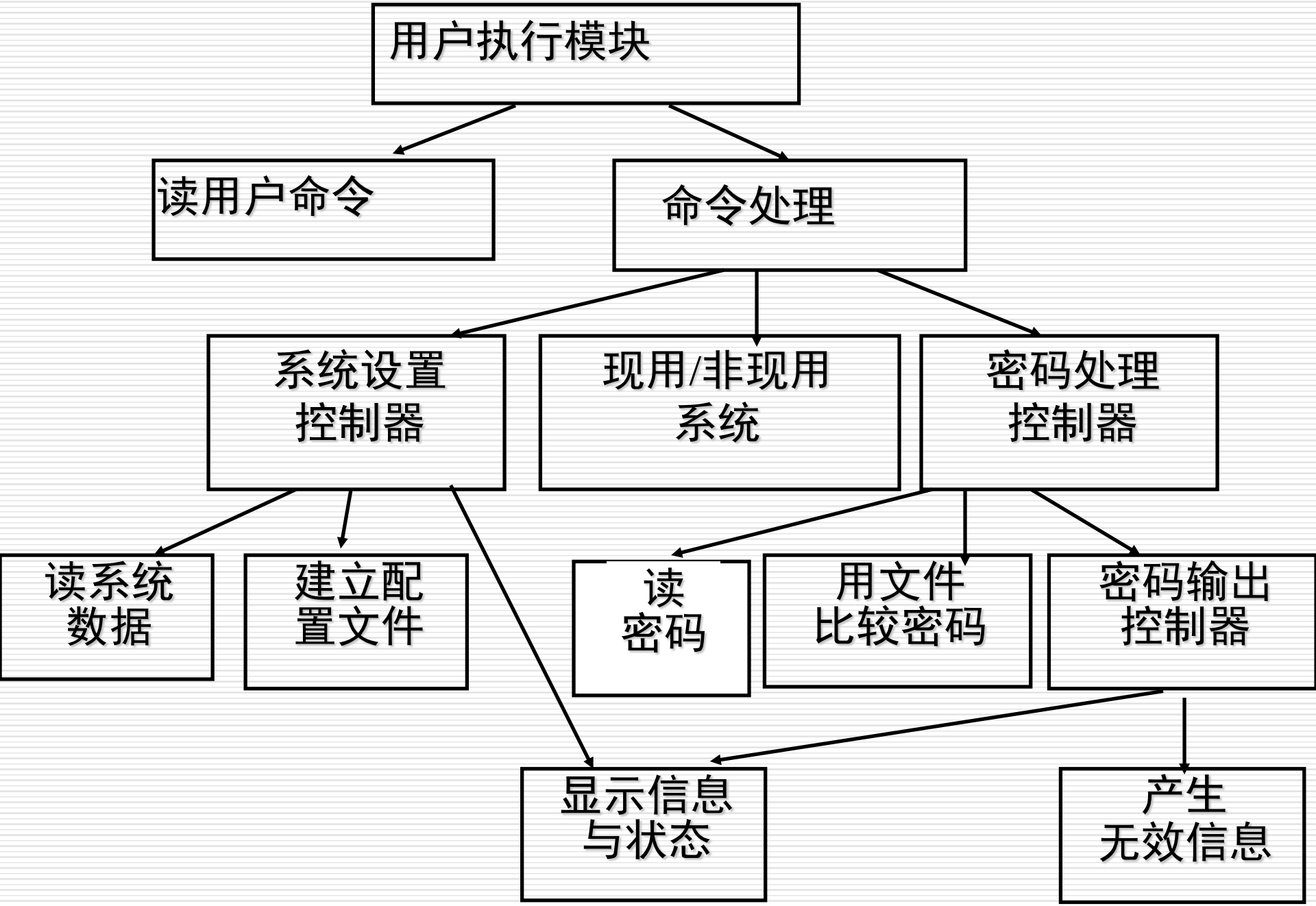


精化的数字仪表板系统的软件结构

用户命令交互子系统ＤＦＤ



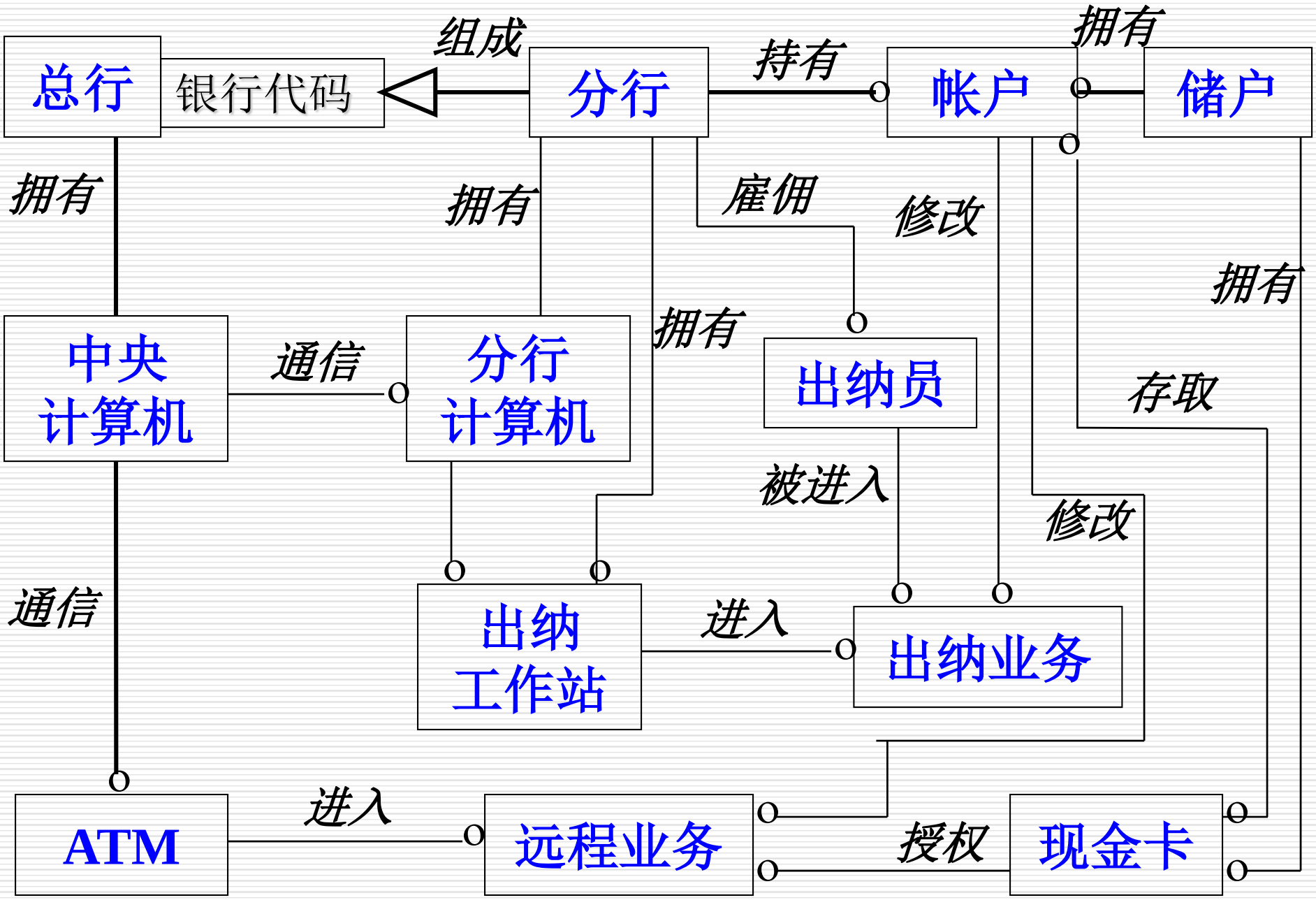
用户命令交互子系统的SC



(4) 由类图导出总体结构

UML中的关键类图就是总体结构

特别是包含“主题”的类图



ATM系统的初始对象图

下一个问题

什么是好的软件体系结构？

Design principles of architecture

- ✓ 抽象（abstraction）
- ✓ 封装（encapsulation）
- ✓ 信息隐蔽(information hiding)
- ✓ 模块化（modularization ）
- ✓ 注意点分散(Separation of Concerns)
- ✓ 耦合和内聚（ couple and cohesion ）
- ✓ 策略和实现的分离（separation of police and implementation)
- ✓ 接口和实现的分离（separation of interface and implementation)
- ✓ 分而制之(Divide-and-conquer)
- ✓ 层次化（hierarchy)