第五章 结构化设计

1.何谓设计

一种软件开发活动，定义实现需求规约所需的软件结构.

设计目标：依据需求规约，在一个抽象层上建立系统软件模型，包括软件体系结构（数据和程序结构），以及详细的处理算法，产生设计规格说明书.

即：要回答如何解决问题－给出软件解决方案

结构化设计分为：

（1）总体设计：确定系统的整体模块结构，即系统实现所需 要的软件模块以及这些模块之间的调用关系。

（2）详细设计：详细描述模块。

2. 实现软件设计的目标对结构化设计方法的需求

（1）提供可体现“原理 / 原则”的一组术语 (符号)，形成一个特定的抽象层，用于表达设计中所使用的部件。

（2）依据术语所形成的“空间”，给出表达软件模型工具。

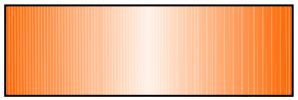
（3）给出设计的过程指导。

3. 结构化设计方法

（1）在总体设计层

① 引入了两个术语 / 符号

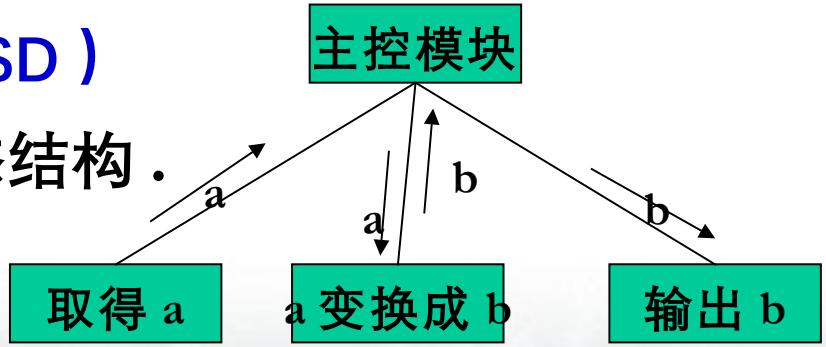
模块 : 一种可独立标识的软件成分 .



调用 : 模块间的一种关系，模块 A 为了完成其任务必须依赖其他模块 .

② 引入了模块结构图（ MSD ）

用于表达软件系统的静态结构 .



③ 过程指导

为了实现设计目标，总体设计的具体任务是：

将 DFD 转化为 MSD

分二步实现：

第一步：如何将 DFD 转化为初始的 MSD

分类：变换型数据流图

事务型数据流图

变换设计

事务设计

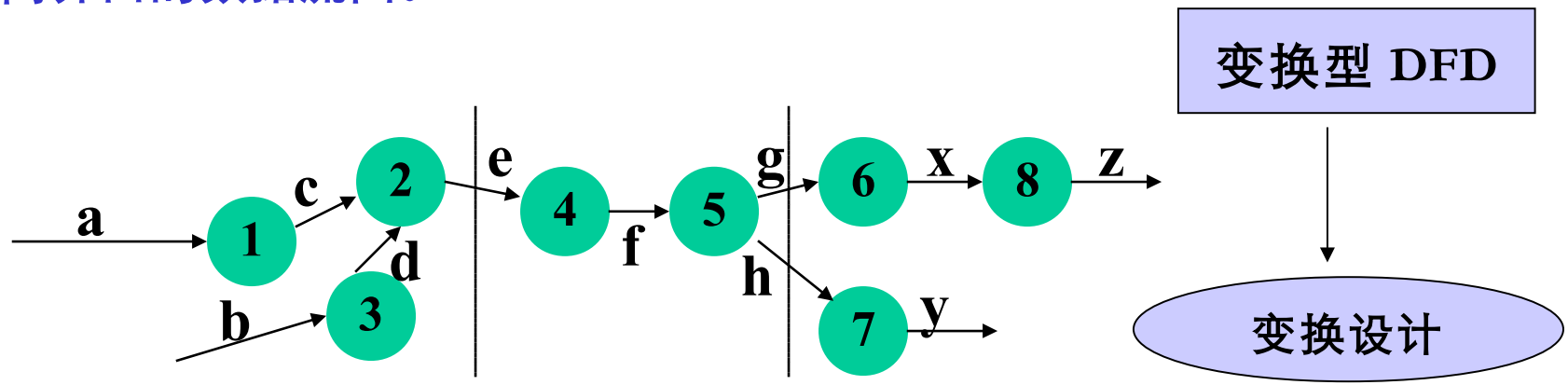
第二步：如何将初始的 MSD 转化为最终可供详细设计使用的 MSD

4. 总体设计第一步：DFD→ 初始的MSD

（1）数据流图分类

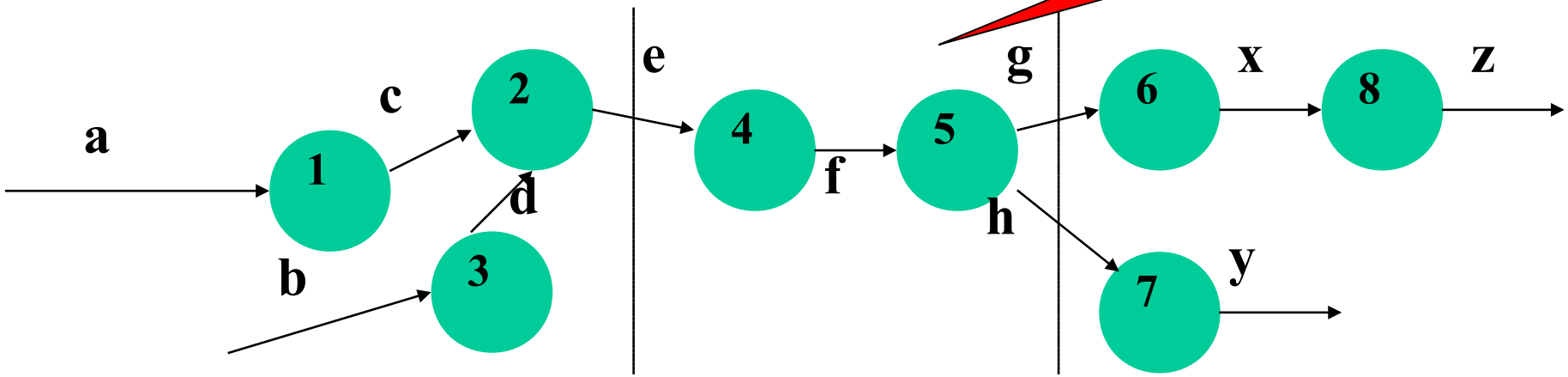
- 变换型数据流图

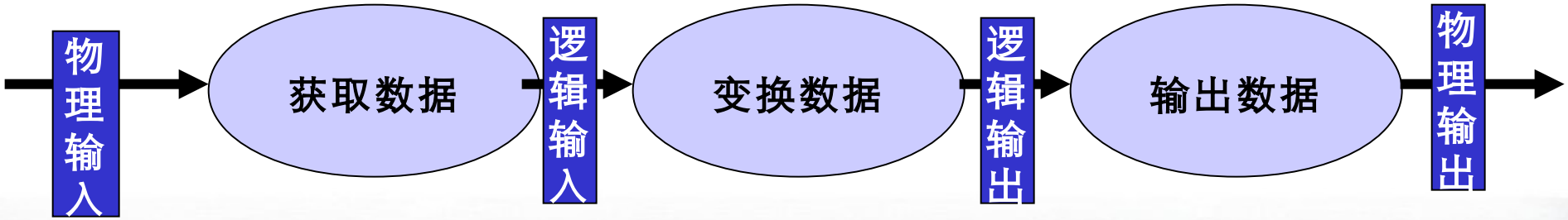
具有较明显的输入部分和变换部分之间的界面、变换部分和输出部分之间界面的数据流图。



变换型 DFD

问题：
逻辑输入：e
逻辑输出：g,h
物理输入：a, b
物理输出：z,y

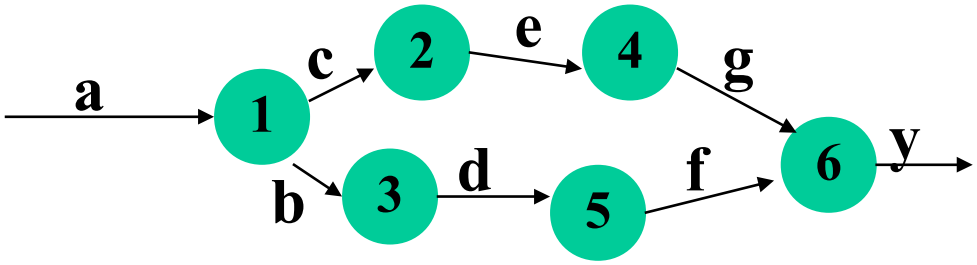


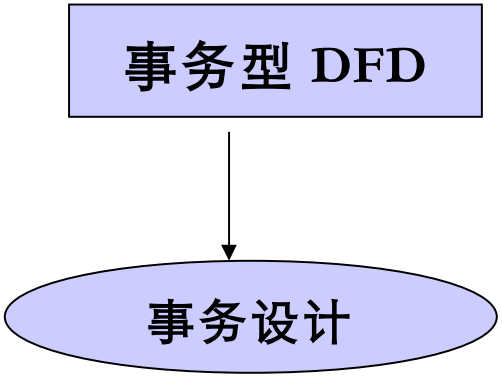


- 逻辑输入：离物理输入最远、仍被看成系统输入的数据流
- 逻辑输出：离物理输出最远、仍被看成系统输出的数据流

- **事务型数据流图：**

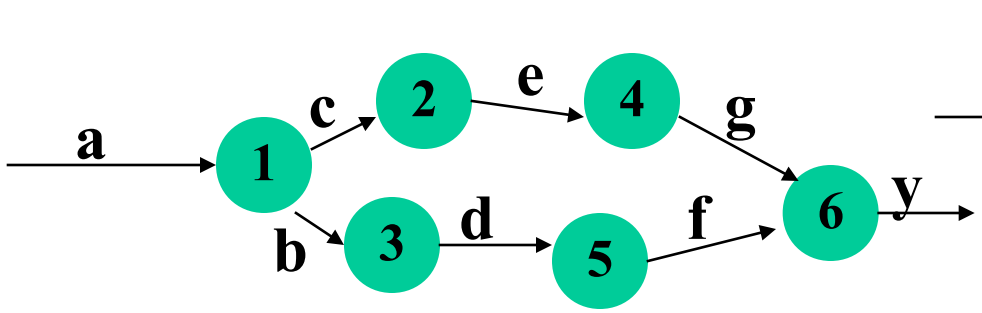
数据到达一个加工（例如下图 1），该加工根据输入数据的值，在其后的若干动作序列（称为一个事务）中选出一个来执行，这类数据流图称为事务型数据流图。

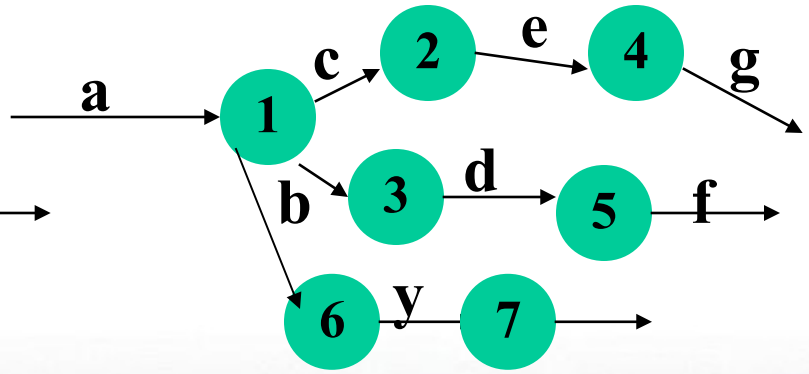




事务型 DFD

- 事务型 DFD 完成下述任务
  - 1）接受输入数据
  - 2）分析并确定对应的事务
  - 3）选取与该事务对应的一条活动路径
- 事务型 DFD 和变换型 DFD 的区别
  - 原则上所有 DFD 都可以看成是变换型 DFD
  - 一般而言，接受 1 个输入数据，分成多条路径





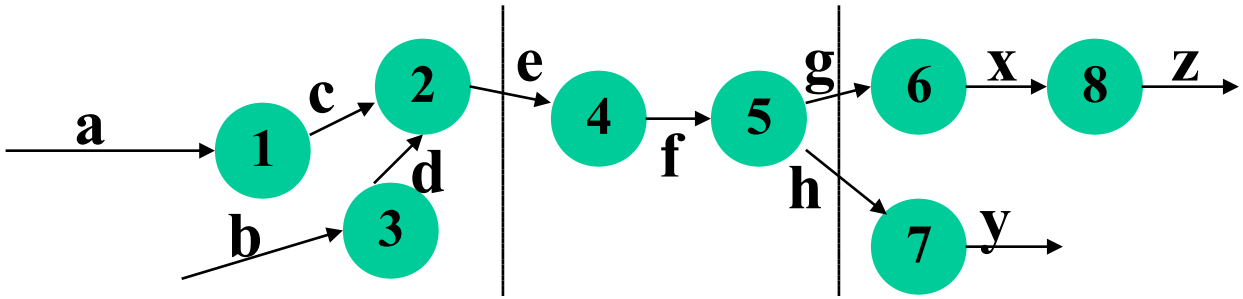
（2）变换设计的基本步骤

①第 1 步：设计准备—复审并精化系统模型

- 为了确保系统的输入数据和输出数据符合实际情况而复审其语境
- 为了确保是否需要进一步精化系统的 DFD 图而复审其语境

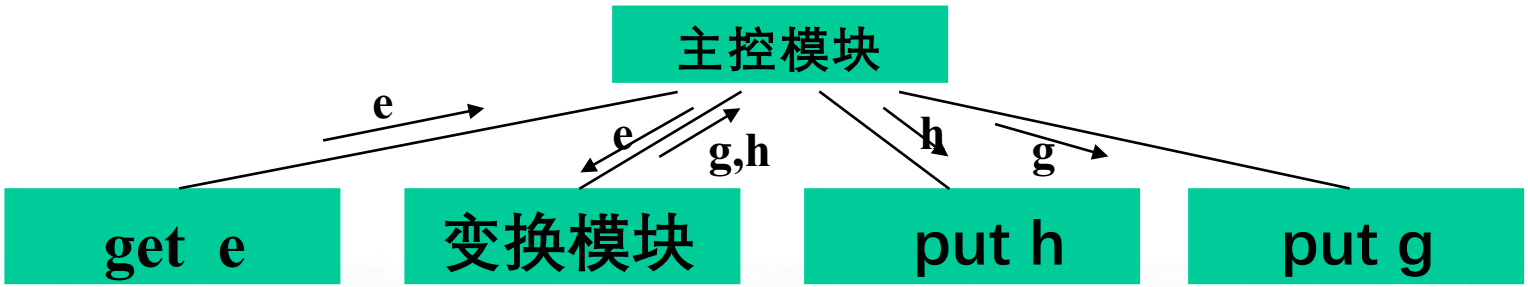
②第 2 步：确定输入、变换、输出这三部分之间的边界

- 根据加工的语义和相关的数据流，确定系统的逻辑输入和逻辑输出

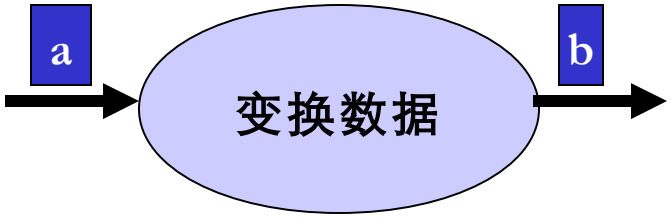


③ 第 3 步：第一级分解—系统模块结构图顶层和第一层的设计

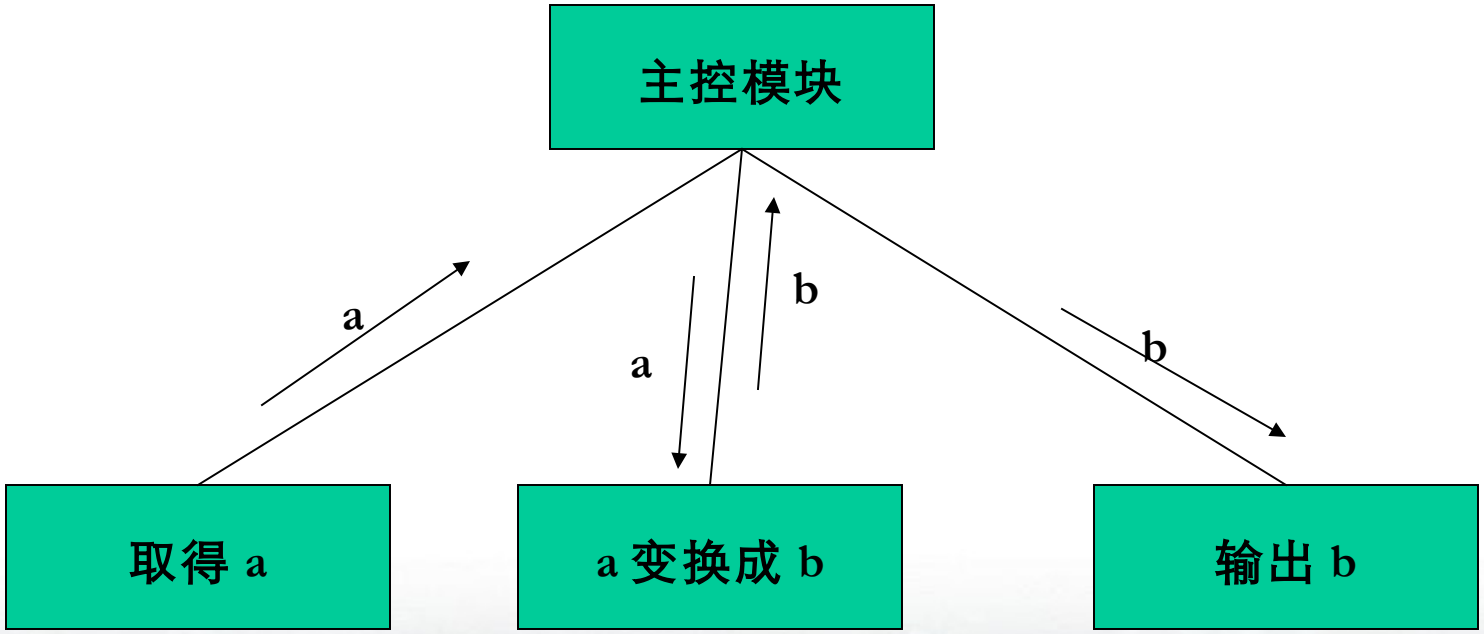
- 主模块：位于最顶层，一般以所建系统的名字命名，其任务是协调控制第一层模块
- 输入模块部分：为主模块提供加工数据，有几个逻辑输入就设计几个输入模块
- 变换模块部分：接受输入模块部分的数据，并对内部形式的数据加工，产生系统所有的内部输出数据
- 输出模块部分：将变换模块产生的输出数据，以用户可见的形式输出。有几个逻辑输出，就设计几个输出模块



最简单的变换型 DFD

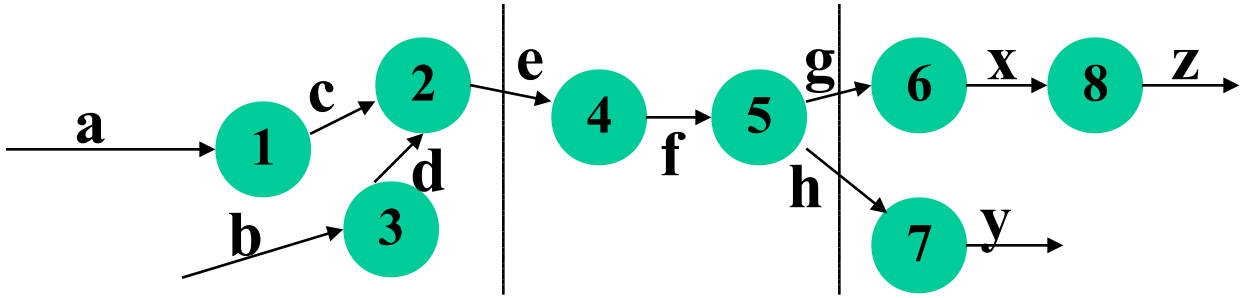


逻辑输入 = 物理输入
逻辑输出 = 物理输出



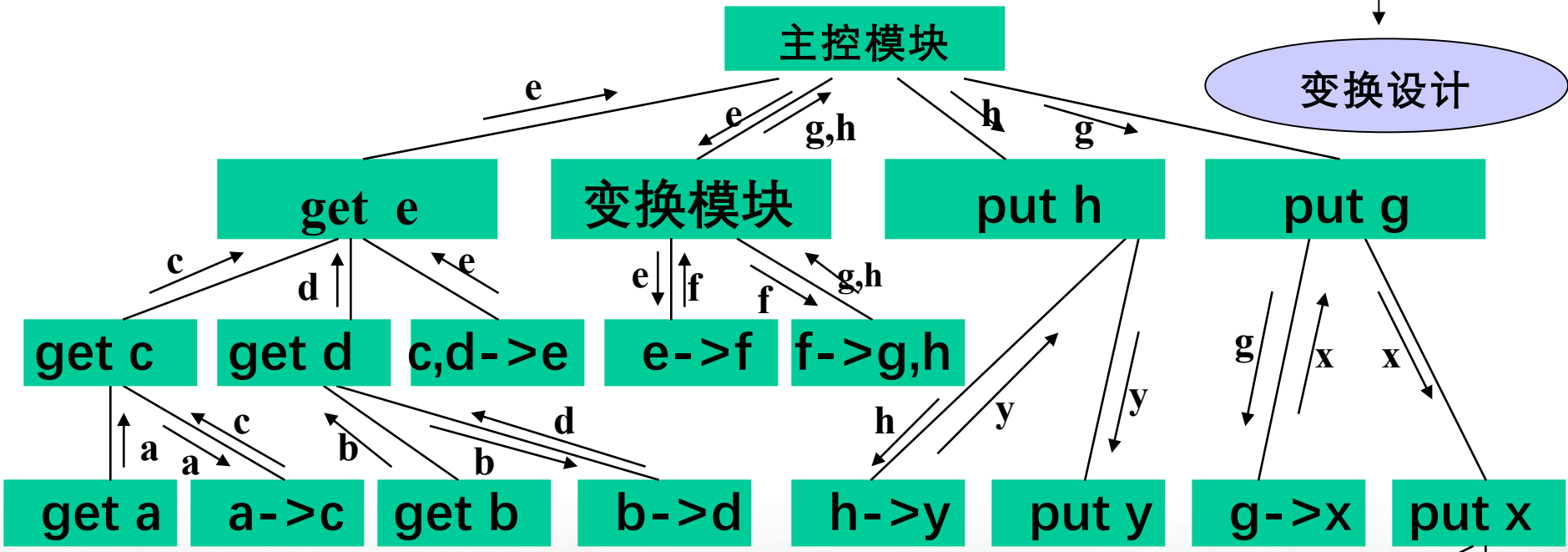
④ 第 4 步：第二级分解—自顶向下，逐步求精

- 对每一个输入模块设计其下层模块
  - 接收数据模块（即输入模块）
  - 把接收的数据变换成它的上级模块所需的数据（即变换模块）
  - 直到输入模块为物理输入，则细化停止
- 对每一个输出模块设计其下层模块
  - 将得到的数据向输出形式进行转换
  - 将转换后的数据进行输出
  - 直到输出模块是物理输出，则细化停止
- 对变化模块进行分解（无通用法则）



变换型 DFD

变换设计



图的宽度不好控制☺

（3）事务设计的基本步骤

①第 1 步：设计准备—复审并精化系统模型

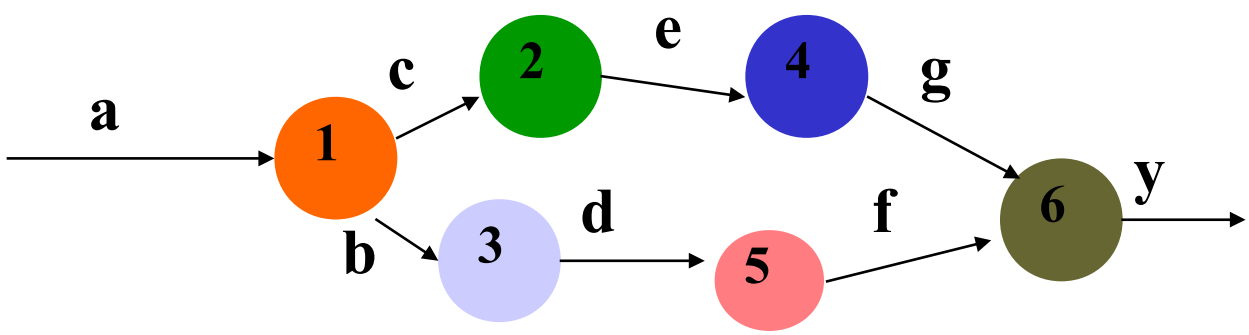
- 为了确保系统的输入数据和输出数据符合实际情况而复审其语境
- 为了确保是否需要进一步精化系统的 DFD 图而复审其语境

②第 2 步：确定事务处理中心

③第 3 步：第一级分解—系统模块结构图顶层和第一层的设计

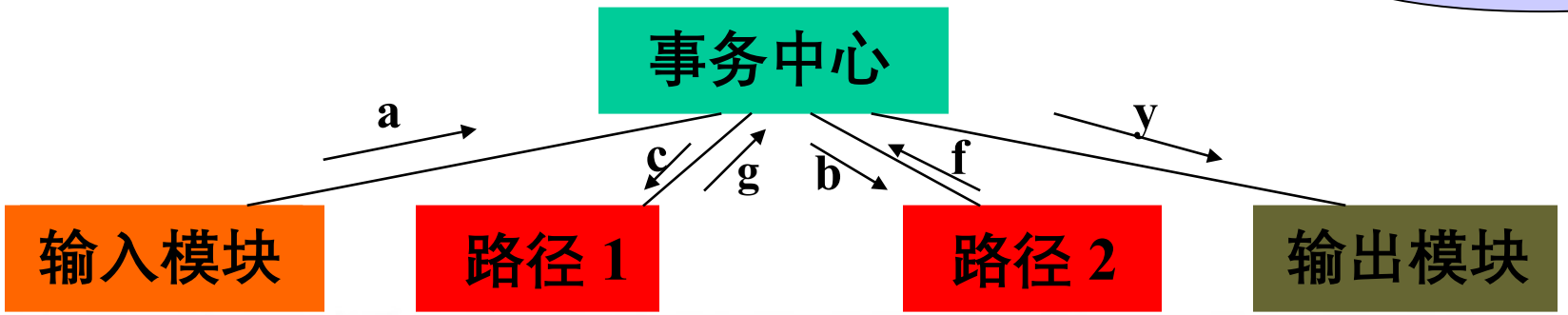
- 首先，为事务中心设计一个主模块
- 然后，为每一条活动路径设计一个事务处理模块
- 对其输入部分设计一个输入模块
- 如果一个事务数据流图的活动路径集中于一个加工，则设计一个输出模块，否则第一层不设计输出模块

第一级分解举例：



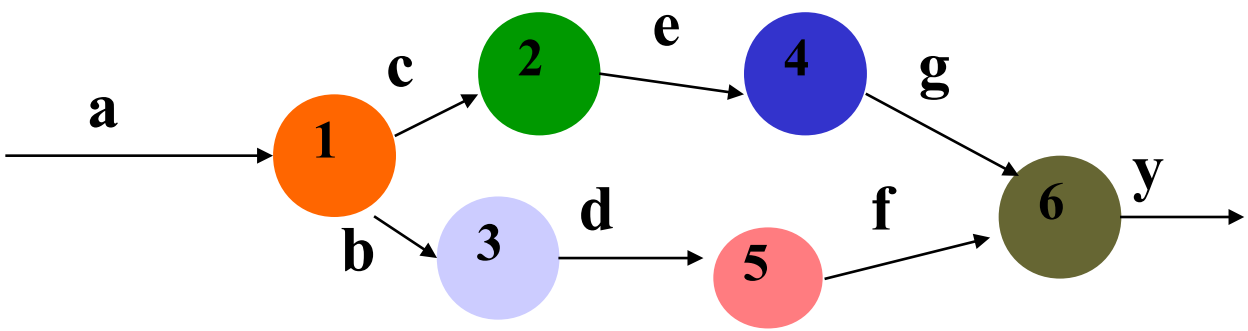
事务型 DFD

事务设计



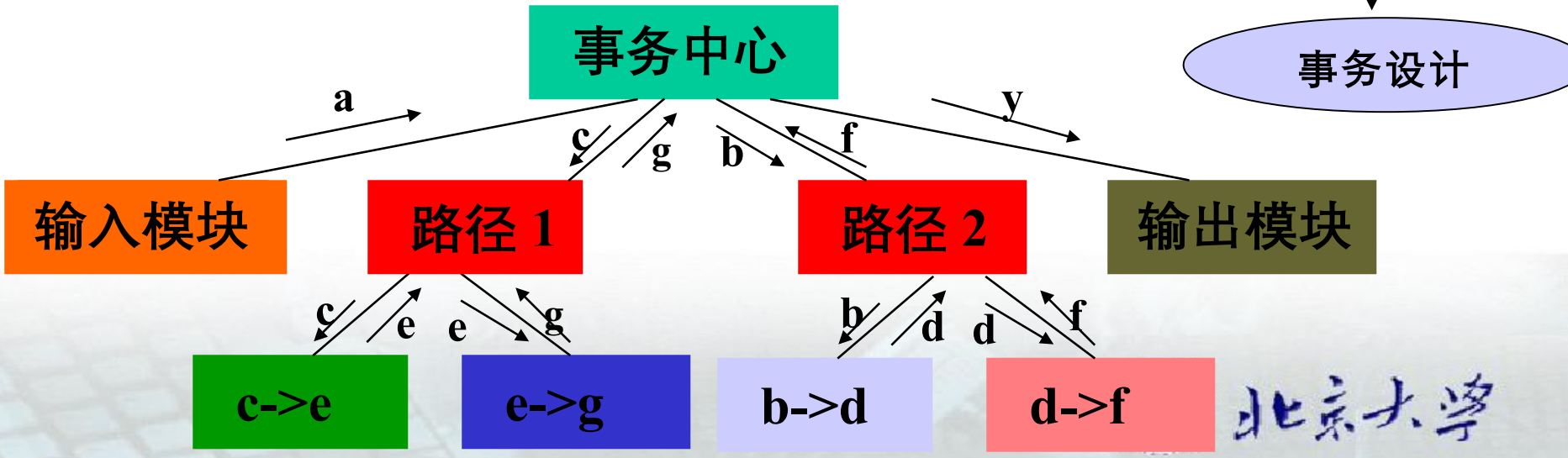
④ 第 4 步：第二级分解—自顶向下，逐步求精

- 对于输入模块、输出模块的细化，如同变化设计的细化过程
- 对各条路径模块的细化，无设计法则



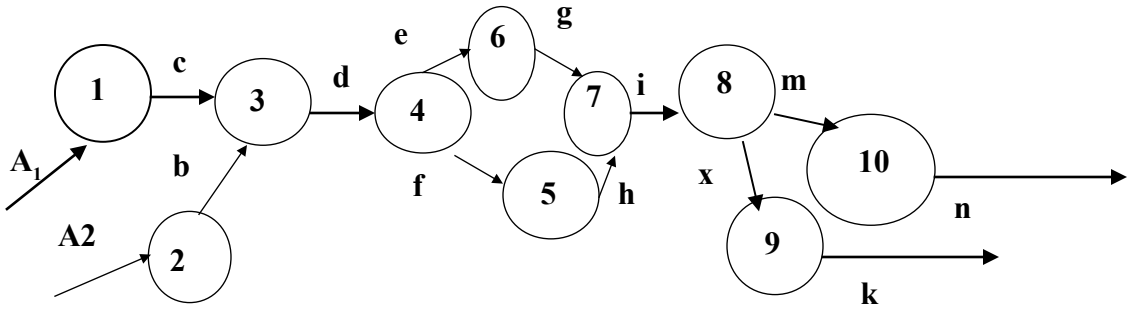
事务型 DFD

事务设计

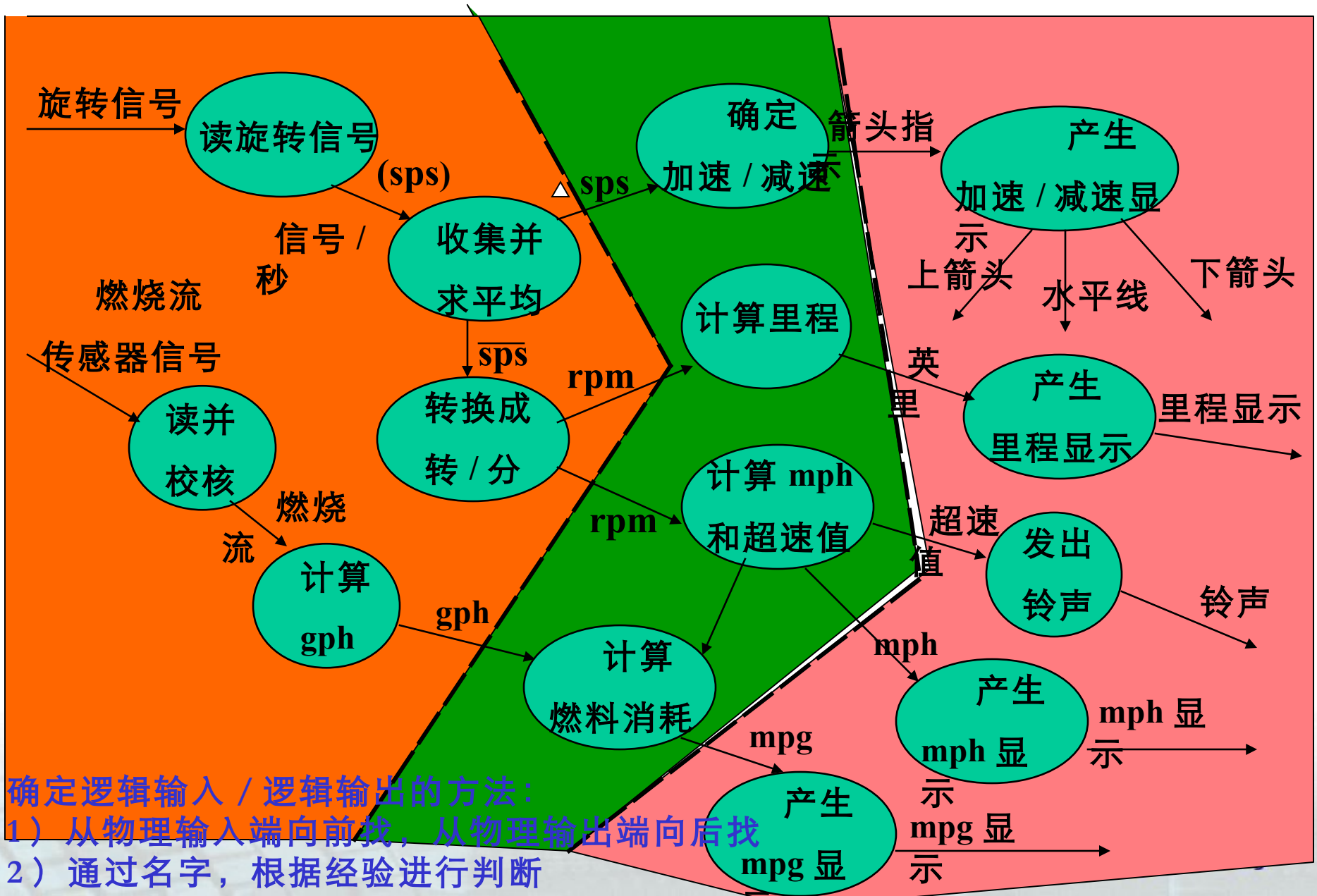


总体设计第一步

- DFD→ 初始的 MSD
  - 一个系统的 DFD，通常是变换型数据流图和事务型数据流图的组合
  - 自动的变换设计
  - 自动的事务设计



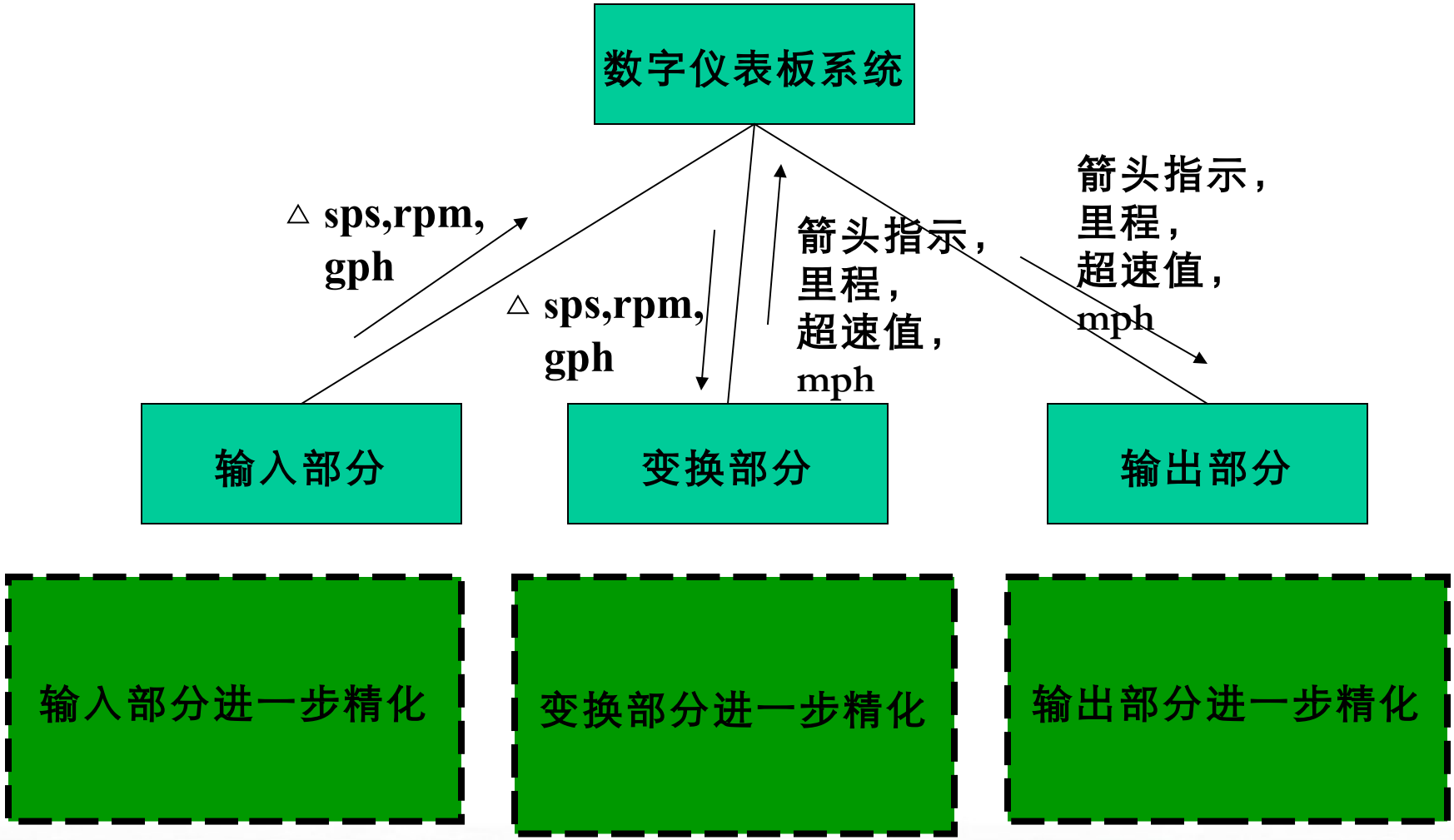
例子：数字仪表板系统



确定逻辑输入 / 逻辑输出的方法：

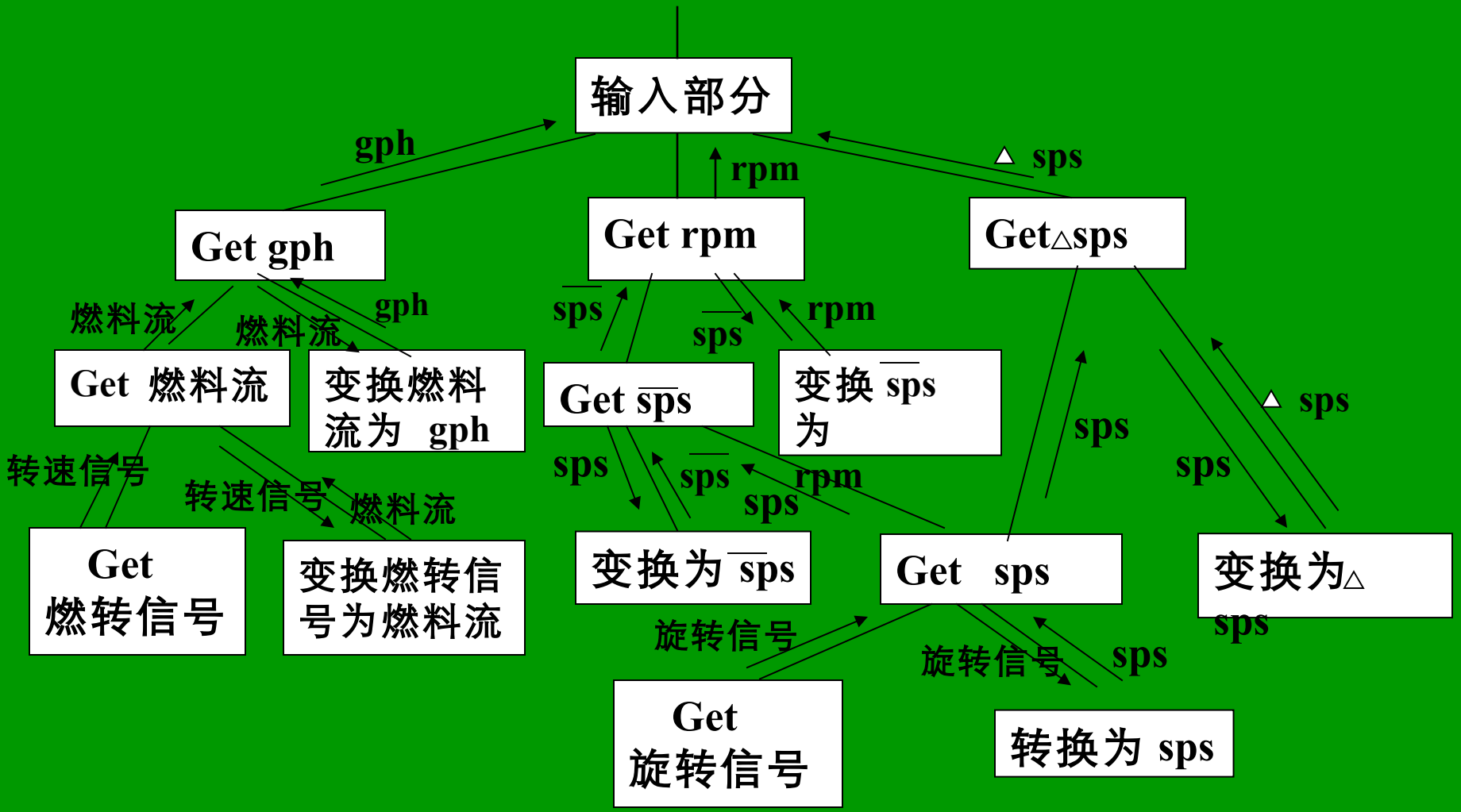
1）从物理输入端向前找，从物理输出端向后找

2）通过名字，根据经验进行判断

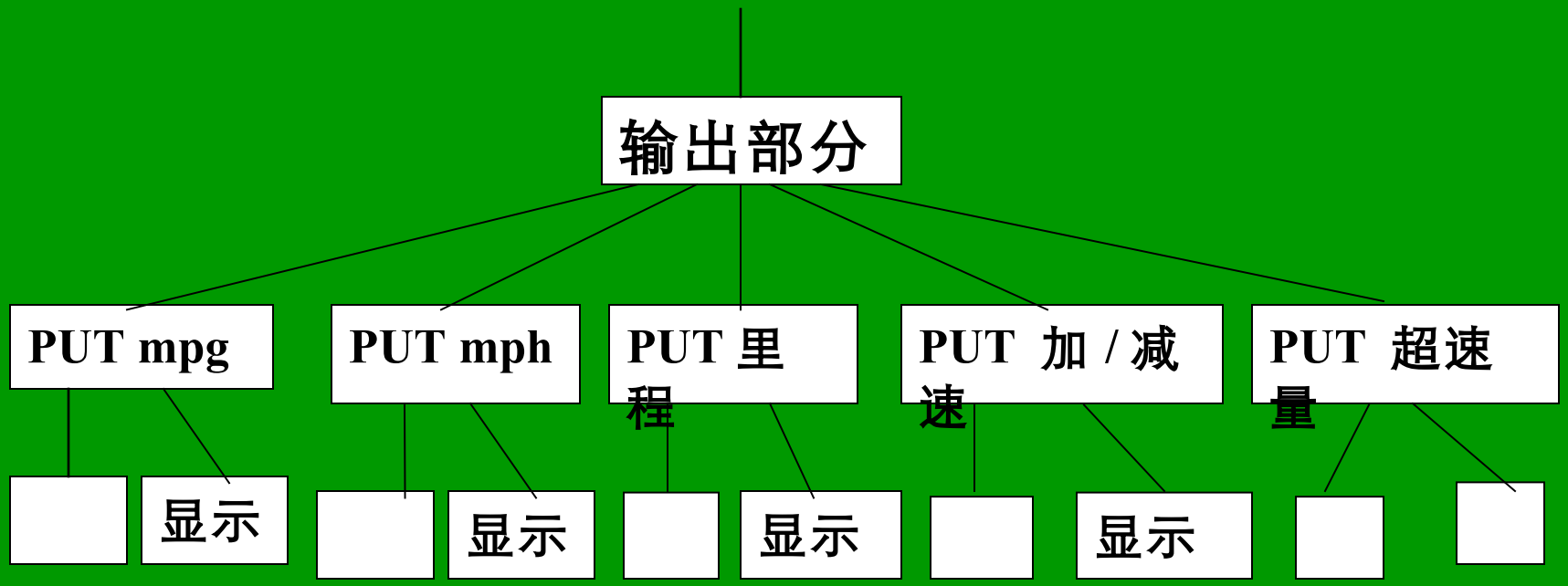


其中： sps 为转速的每秒信号量； $\overline{sps}$ 为 sps 的平均值； Δsps 为 sps 的瞬变化值； rpm 为每分钟转速； mph 为每小时英里数； gph 为每小时燃烧的燃料加仑数； m 为行进里程。

输入部分进一步精化


输入部分
gph
rpm
△sps
Get gph
Get rpm
Get△sps
燃料流
燃料流
gph
sps
sps
rpm
Get 燃料流
变换燃料流为 gph
Get $\overline{sps}$
变换 $\overline{sps}$ 为
转速信号
转速信号
燃料流
sps
sps
rpm
sps
sps
sps
△sps
Get 燃转信号
变换燃转信号为燃料流
变换为 $\overline{sps}$
Get sps
变换为△sps
旋转信号
旋转信号
sps
Get 旋转信号
转换为 sps

输出部分进一步精化



5. 总体设计第二步：将初始的 MSD 转化为最终可供详细设计 使用的 MSD

概念：模块，模块化

基于模块化原理 - **高内聚 低耦合，**

模块 和 模块化

- 模块：执行一个特殊任务的一组例程和数据结构
  - 接口：给出可由其他模块和例程访问的对象
    - 常量，变量，数据类型，函数
  - 实现：接口的实现（模块功能的执行机制）
    - 私有量，过程描述，源程序代码
- 模块化：把系统分解成若干模块的过程
  - 50 多年的历史
  - 软件的单个属性，使得程序能够被理性的管理
    - Myers,G. *Composite Structured Design*, 1978

为什么要模块化？（1）

- 设 C(x) 是定义问题 x 复杂性的函数， E(x) 是定义解决问题 x 所需要的工作量，那么，对于两个问题 p1 和 p2 ，如果

C(p1)>C(p2)

- 那么

E(p1)>E(p2)

解释：解决困难问题需要花费更多的时间

为什么要模块化？（2）

- 人们又发现了另外一个有趣的特征：

$$C(p1+p2)>C(p1)+C(p2)$$

- 由上页结论：

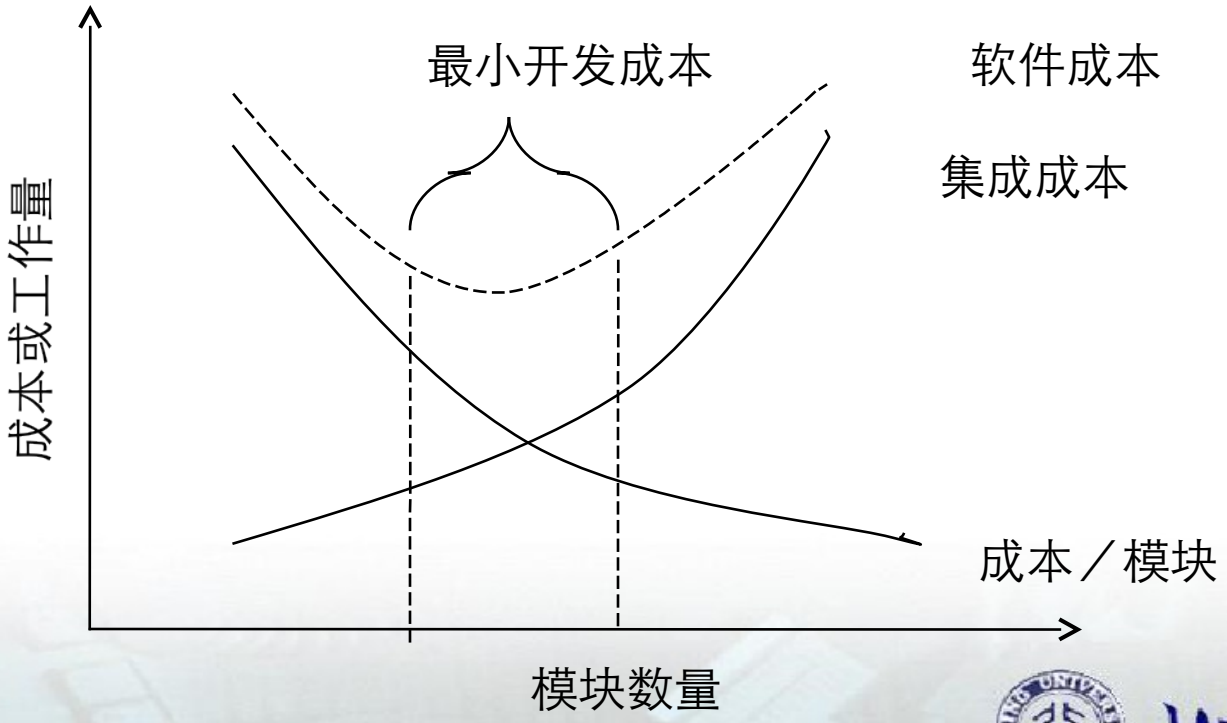
$$If\ C(g1)>C(g2)\ Then\ E(g1)>E(g2)$$

- 所以：

$$E(p1+p2)>E(p1)+E(p2)$$

为什么要模块化？（3）

- 一个理想的情况
  - 如果我们能够无限制地划分软件，那么开发它所需的工作量可以变得非常小，乃至可以忽略！
- 但是，这个结论是错误的
  - 随着模块数量的增长，集成模块所需的工作量（成本）也在增长。



模块化的评价

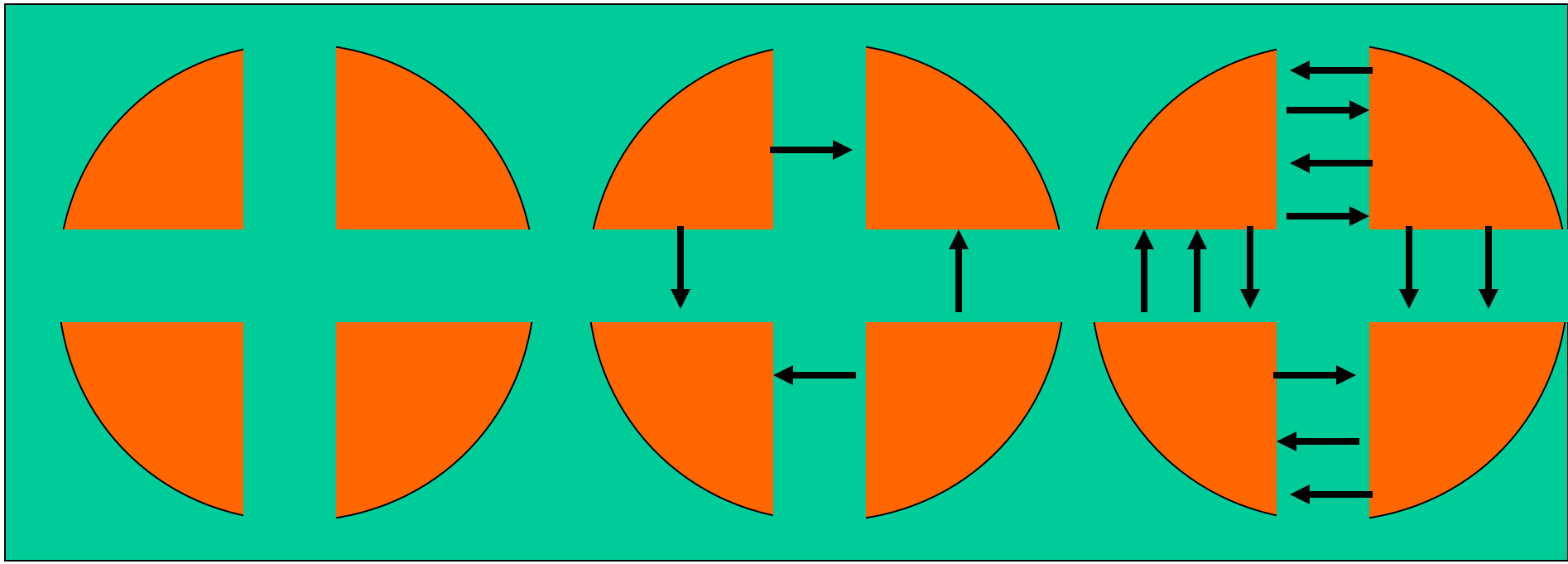
- 基本原则

高内聚，低耦合

- 概念和分类
  - 耦合
  - 内聚
- 启发式规则

耦 合（1）

- **定义：**不同模块之间相互依赖程度的度量



无耦合　　松散耦合　　紧密耦合

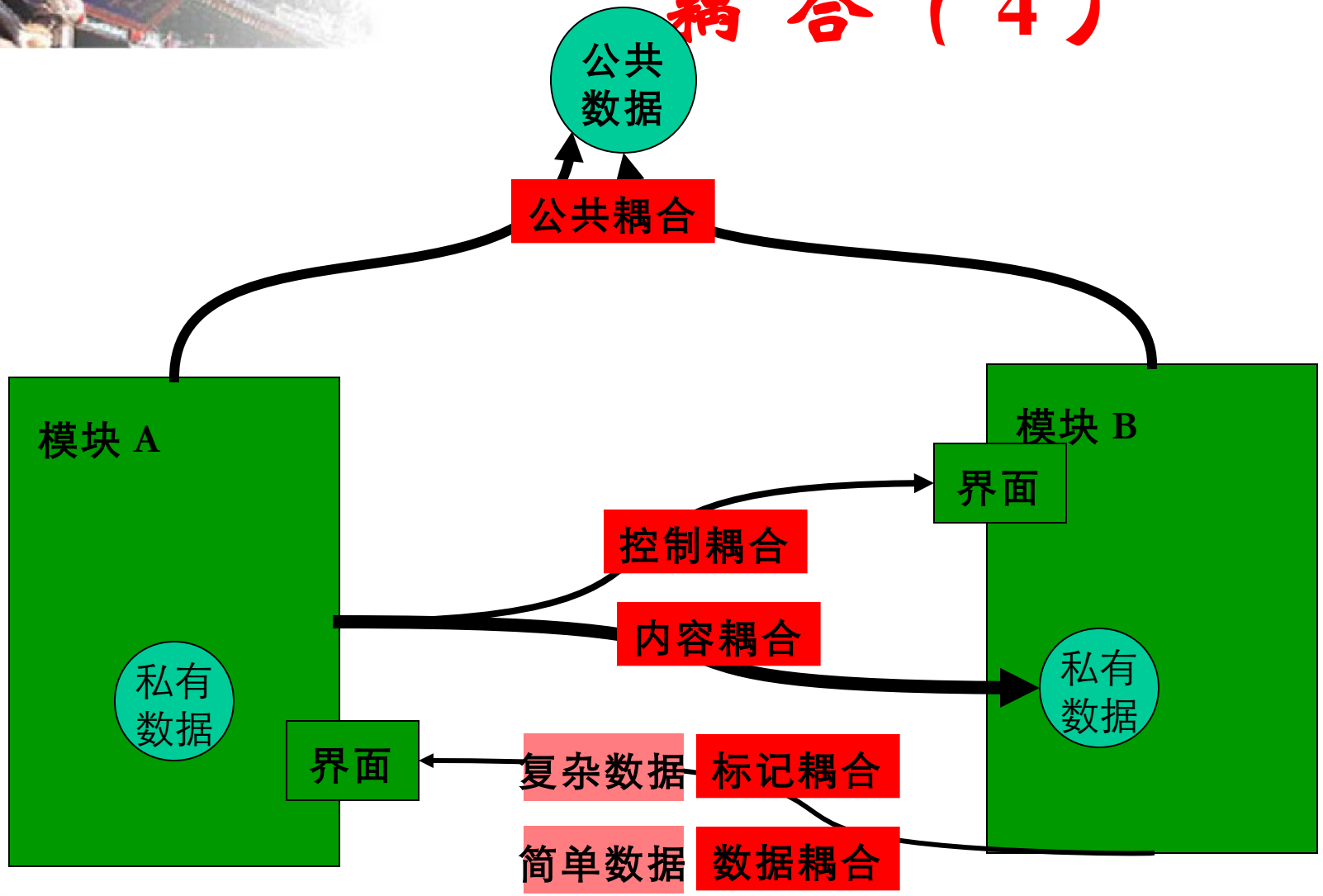
耦 合（2）

- 耦合的强度所依赖的因素：
  - 一个模块对另一个模块的引用
  - 一个模块向另一个模块传递的数据量
  - 一个模块施加到另一个模块的控制的数量
  - 模块之间接口的复杂程度
    - 整数，数组，控制信号……

耦 合（3）

- 耦合类型：（由强到弱）
  - 内容耦合：一个模块直接修改或操作另一个模块的数据。
  - 公共耦合：两个以上的模块共同引用一个全局数据项。
  - 控制耦合：一个模块向另一模块传递一个控制信号，接受信号的模块将依据该信号值进行必要的活动。
  - 标记耦合：两个模块至少有一个通过界面传递的公共参数，包含内部结构，如数组，字符串等。
  - 数据耦合：模块间通过参数传递基本类型的数据。

耦合（4）


公共数据
公共耦合
模块 A
模块 B
界面
控制耦合
内容耦合
私有数据
私有数据
界面
复杂数据
标记耦合
简单数据
数据耦合

耦合（5）

- 原则：

如果模块间必须存在耦合，就尽量使用数据耦合，少用控制耦合，限制公共耦合的范围，坚决避免使用内容耦合。

内 聚（1）

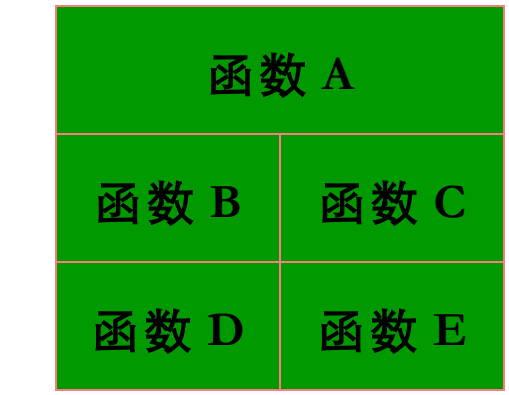
- 定义：一个模块之内各成分之间相互依赖程度的度量。

- 好的设计满足：
  - 模块的功能单一
  - 模块的各部分都和模块的功能直接相关
  - 高内聚

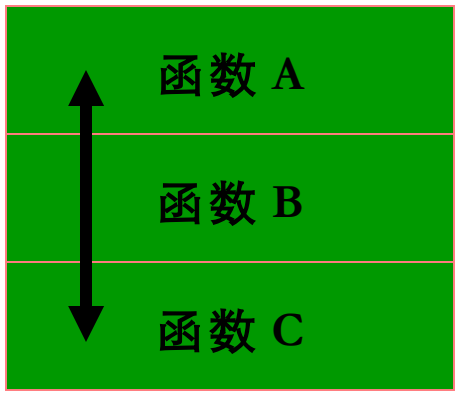
内 聚（2）

- 内聚类型：（由低到高）
  - 偶然内聚：一个模块之内各成分之间没有任何关系。
  - 逻辑内聚：几个逻辑上相关的功能放在同一模块中。
  - 时间内聚：一个模块完成的功能必须在同一时间内完成，而这些功能只是因为时间因素关联在一起。
  - 过程内聚：处理成分必须以特定的次序执行。
  - 通信内聚：各成分都操作在同一数据集或生成同一数据集。
  - 顺序内聚：各成分与一个功能相关，且一个成分的输出作为 另一成分的输入。
  - 功能内聚：模块的所有成分对完成单一功能是最基本的，且该模块对完成这一功能而言是充分必要的。

内 聚（3）

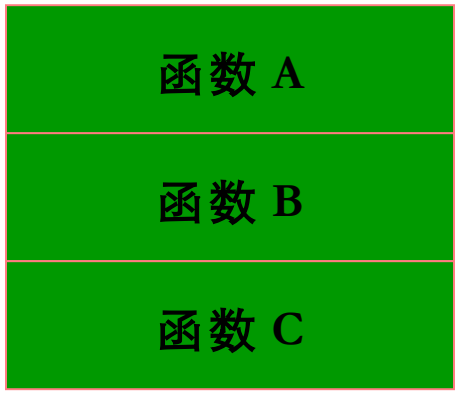


偶然内聚：
各部分互不相关

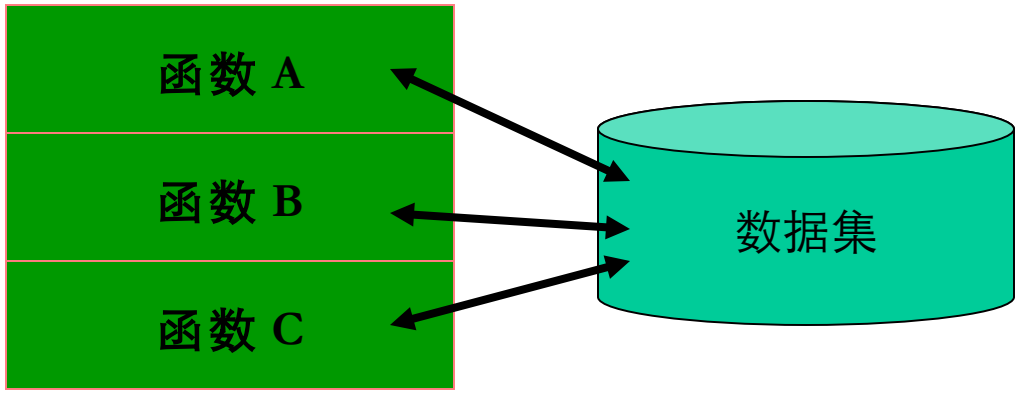


逻辑内聚：
相似的功能

时间内聚：
和时间有关的功能

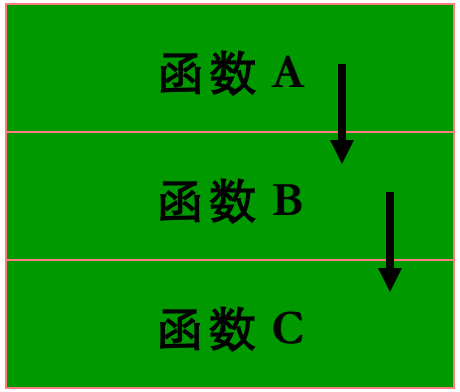


过程内聚：
按次序执行功能

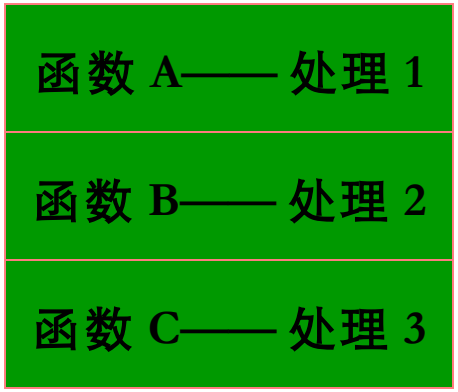


通信内聚：
访问同一数据

内 聚（4）



顺序内聚：
一个部分的输出作为下一部分的输入



功能内聚：
充分而必要的功能

启发式规则（1）

- 什么叫做“启发式”？
  - 根据设计准则，从长期的软件开发实践中，总结出来的规则。
  - 既不是设计目标，也不是设计时应该普遍遵循的原理
- 常见的六种启发式规则
  - 改进软件结构，提高模块独立性；
  - 模块规模适中 - 每页 60 行语句；
  - 深度、宽度、扇入和扇出适中；
  - 模块的作用域力争在控制域之内；
  - 降低模块接口的复杂性；
  - 模块功能应该可以预测。

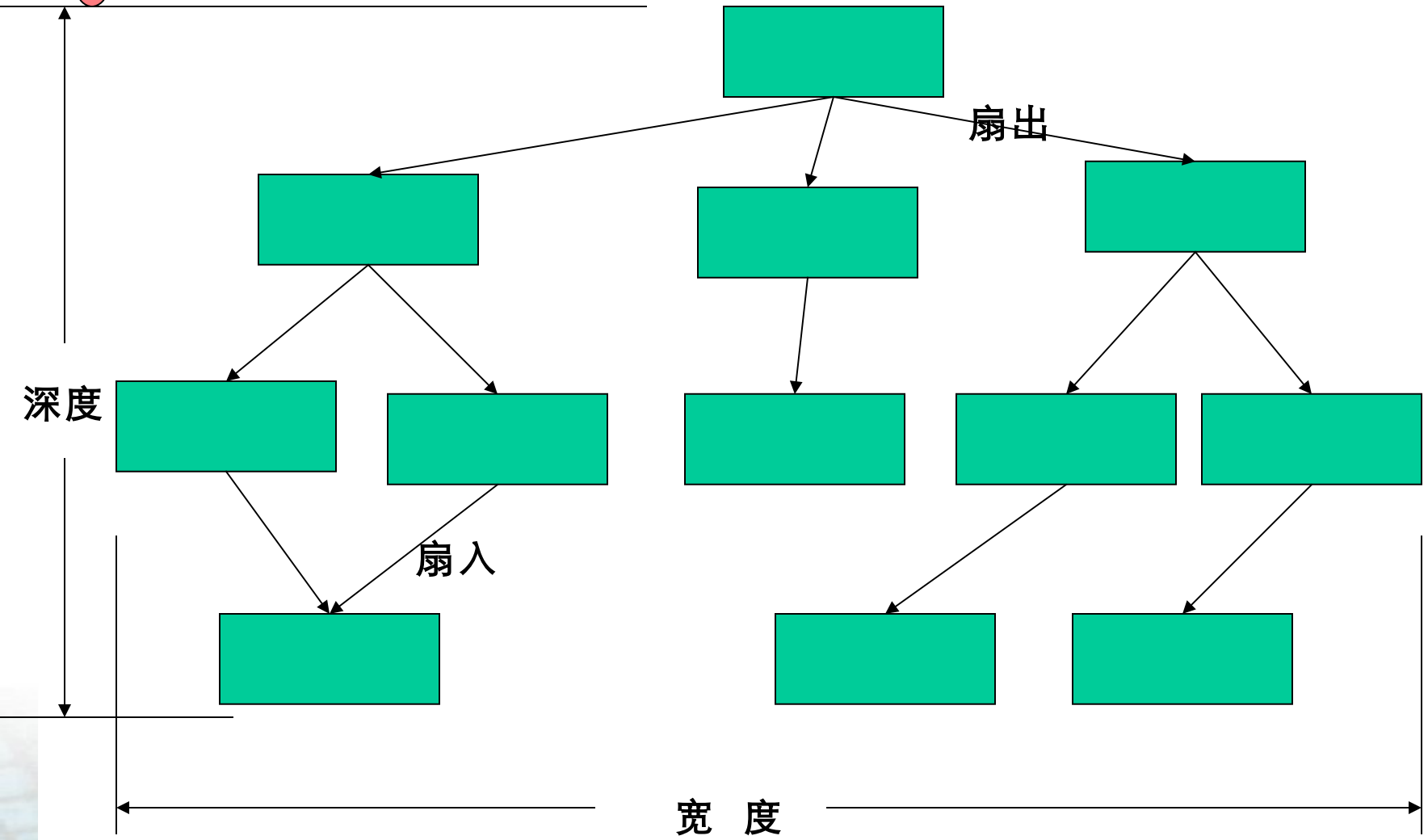
改进软件结构，提高模块独立性

- 通过模块的分解和合并，力求降低耦合，提高内聚。
  - 例：多个模块公用的子功能可以独立形成一个模块，供这些模块调用。

模块规模适中，每页 60 行语句

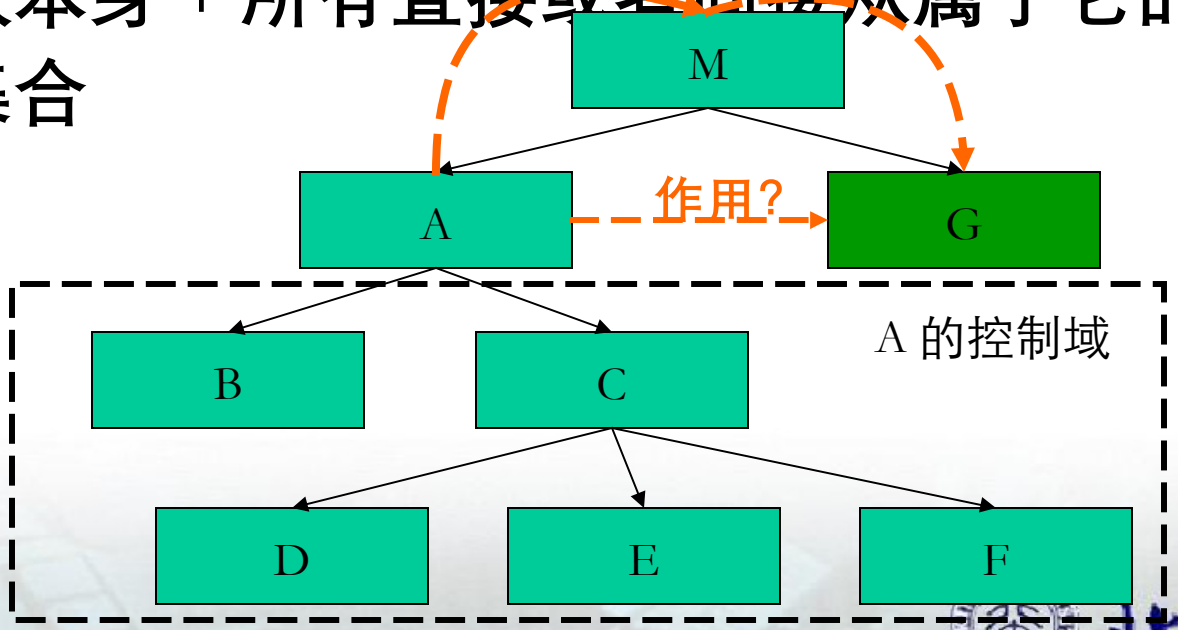
- 模块最好能够写在一页纸内（60 行）
  - 心理学研究表明：模块语句 >30 之后，可理解性迅速下降。
- 方法
  - 进一步分解过大的模块
  - 将频繁调用的小模块合并到上级模块中

深度、宽度、扇入和扇出适中



模块的作用域力争在控制域之内

- 作用域
  - 受该模块内一个判定影响的所有模块的集合
- 控制域
  - 模块本身 + 所有直接或者间接从属于它的模块的集合



降低模块接口的复杂性

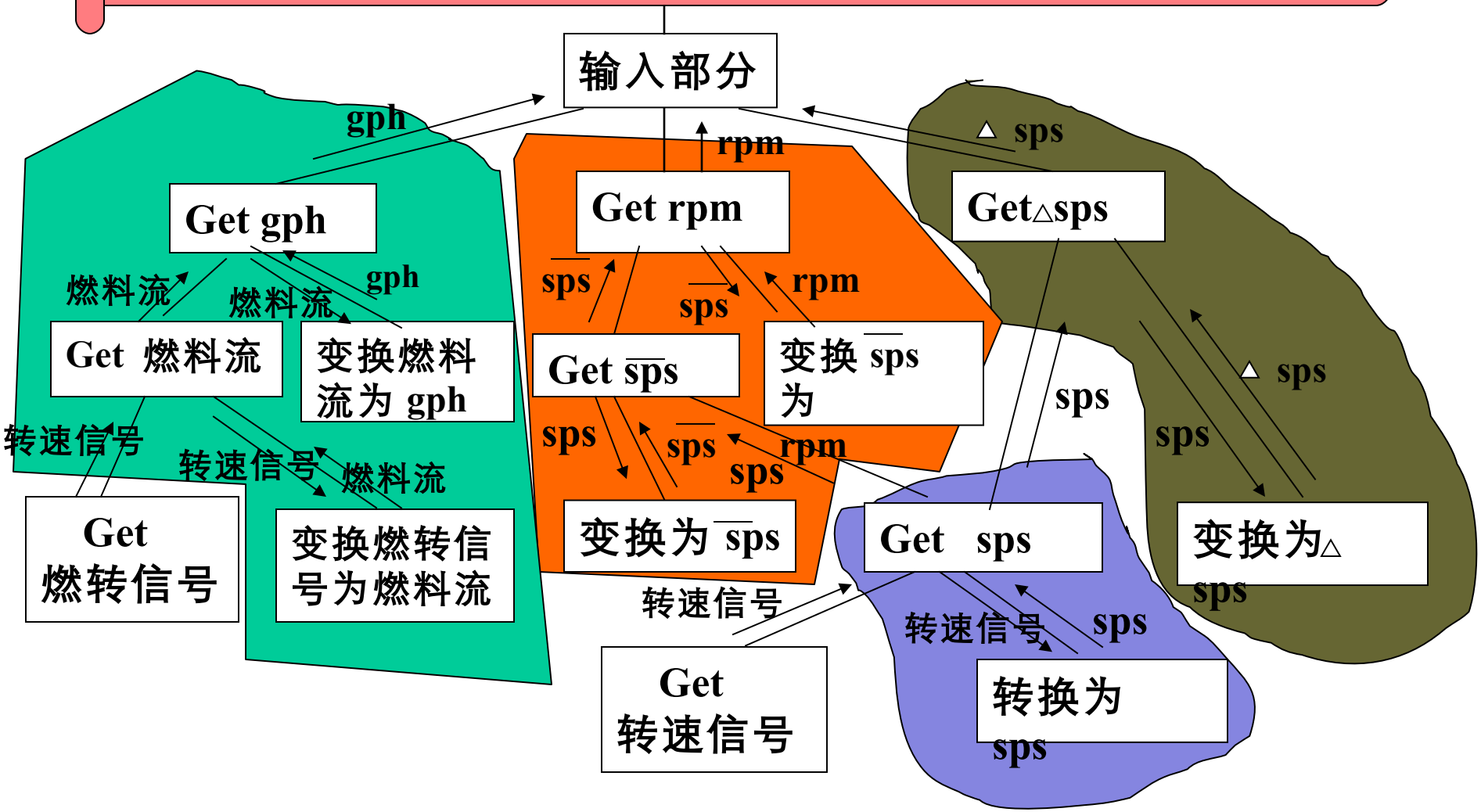
- 使得信息传递简单并且和模块的功能一致
- 接口复杂或不一致往往导致紧耦合和低内聚
- 例子：求 A x^2+B x + C=0 的根
  - QUAD-ROOT(TBL,X)
    - 数组 TBL 传送方程系数，数组 X 回送求得的根
  - QUAD-ROOT(A,B,C,Root1,Root2)

模块功能应该可以预测

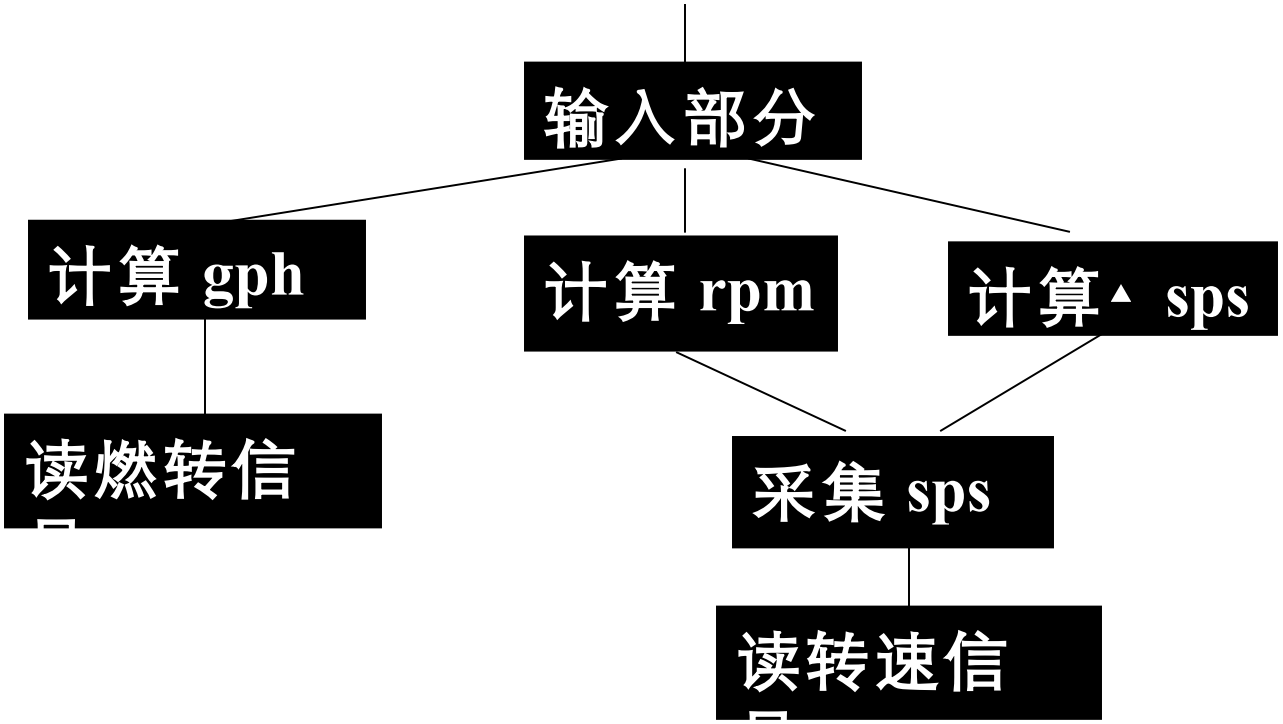
- 什么叫做“功能可以预测”？

- 什么样的模块功能不可预测？
  - 模块带有内部状态→输出取决于该状态

改进软件结构，提高模块独立性



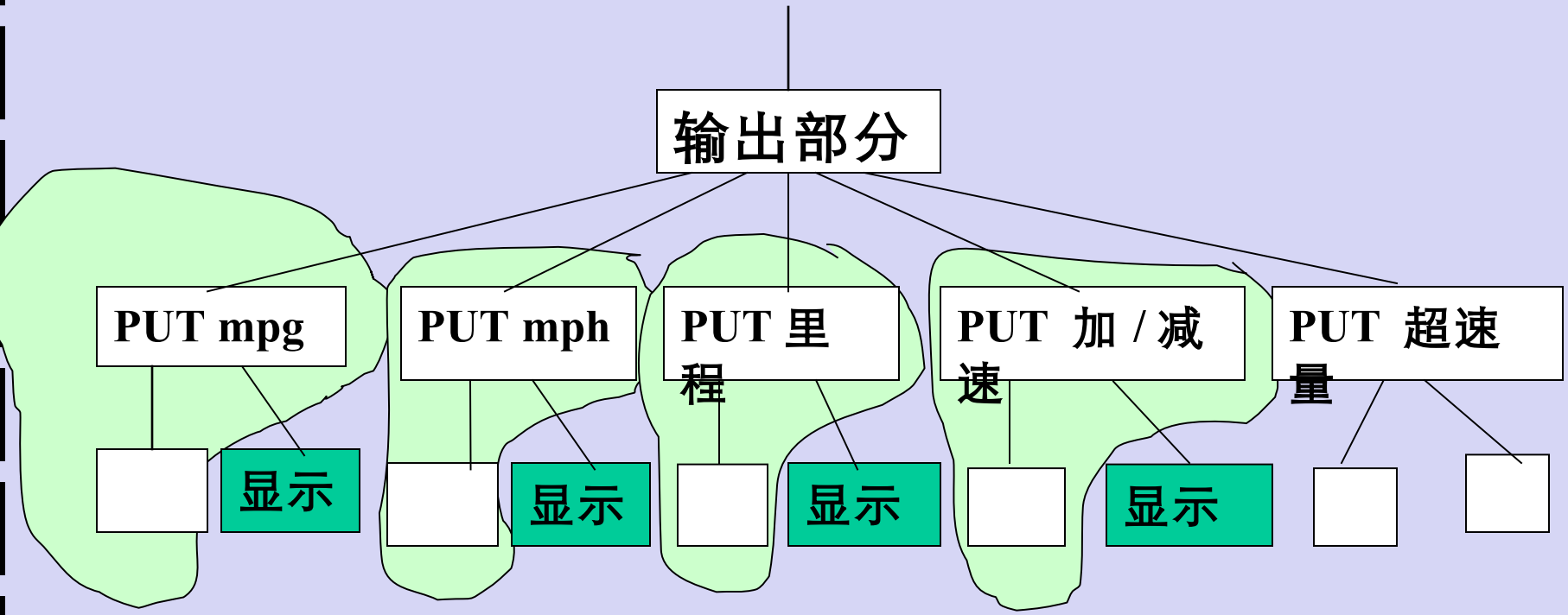
输入部分的精化



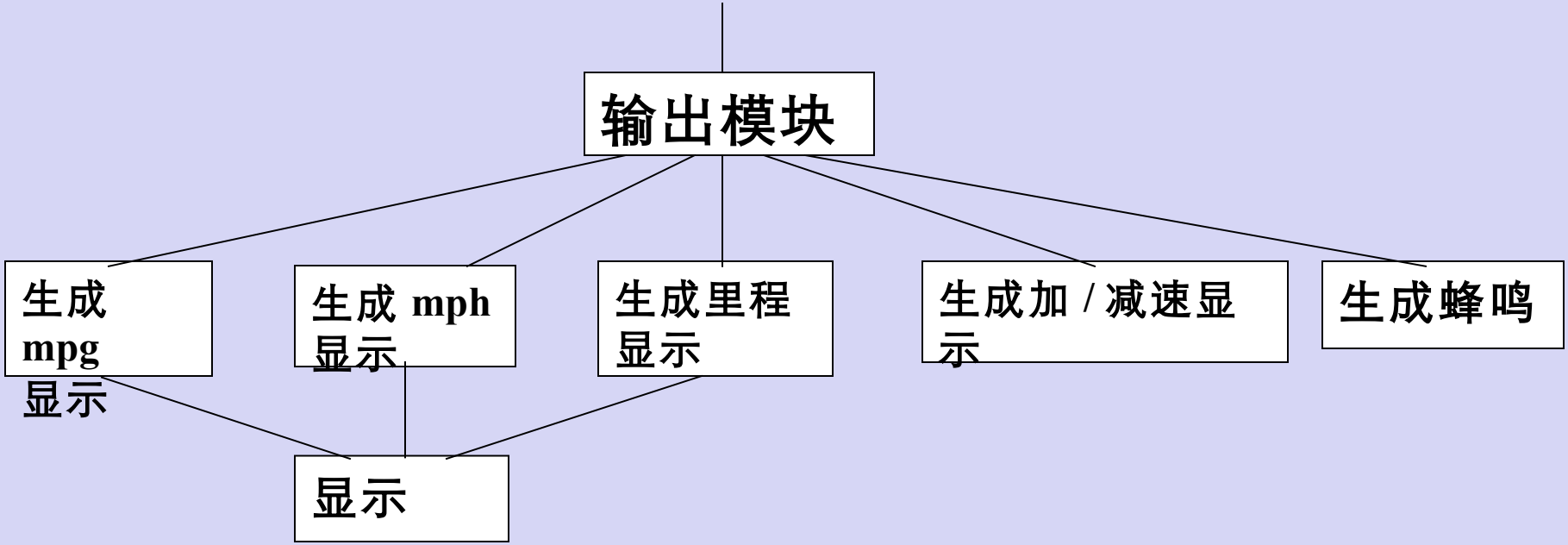
其中： sps 为转速的每秒信号量； sps 为 sps 的平均值； sps 为 sps 的瞬变化值； rpm 为每分钟转速； mph 为每小时英里数； gph 为每小时燃烧的燃料加仑数； rpm 为行进里程。

输出部分进一步精化

把相同或类似的物理输出合并为一个模块，以减少模块之间的关联。



输出部分的精化

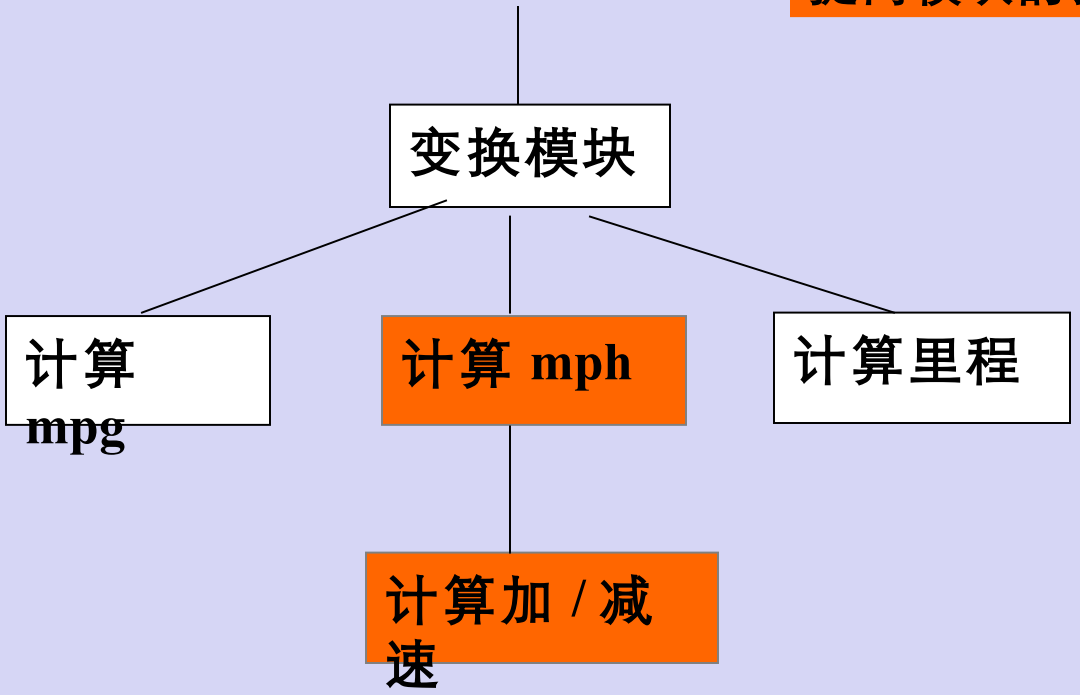


变换部分的精化

- 对于变换部分的求精，是一项具有挑战性的工作。其中主要是根据设计准则，并要通过实践，不断地总结**经验**，才能设计出合理的模块结构。

变换部分的精化

把“确定加/减速”的模块放在“计算速度 mph”模块下面，则可以减少模块之间的关联，提高模块的独立性。



总体设计总结

- 将一个给定的 DFD 转换为初始的模块结构图
  - 基本上是一个 “机械” 的过程
  - 一般体现不了设计人员的创造力
- 优化设计
  - 将一个初始的模块结构图转换为最终的模块结构图
  - 对设计人员将是一种挑战
  - 其结果将直接影响软件系统开发的质量。