

# 结构化编程 I - 思想

# Outline

- 结构化编程思想
  - 思想和模型
- 数据流图
- 结构图
- 流程图

# 结构化方法

- 思想
  - 自顶向下逐步求精
  - 算法+数据结构
- 模型
  - 数据流图
  - 结构图
  - 流程图

# Outline

- 结构化编程思想
  - 思想和模型
  - 数据流图
  - 结构图
  - 流程图
- 编译器的结构
- 编程的现实考量

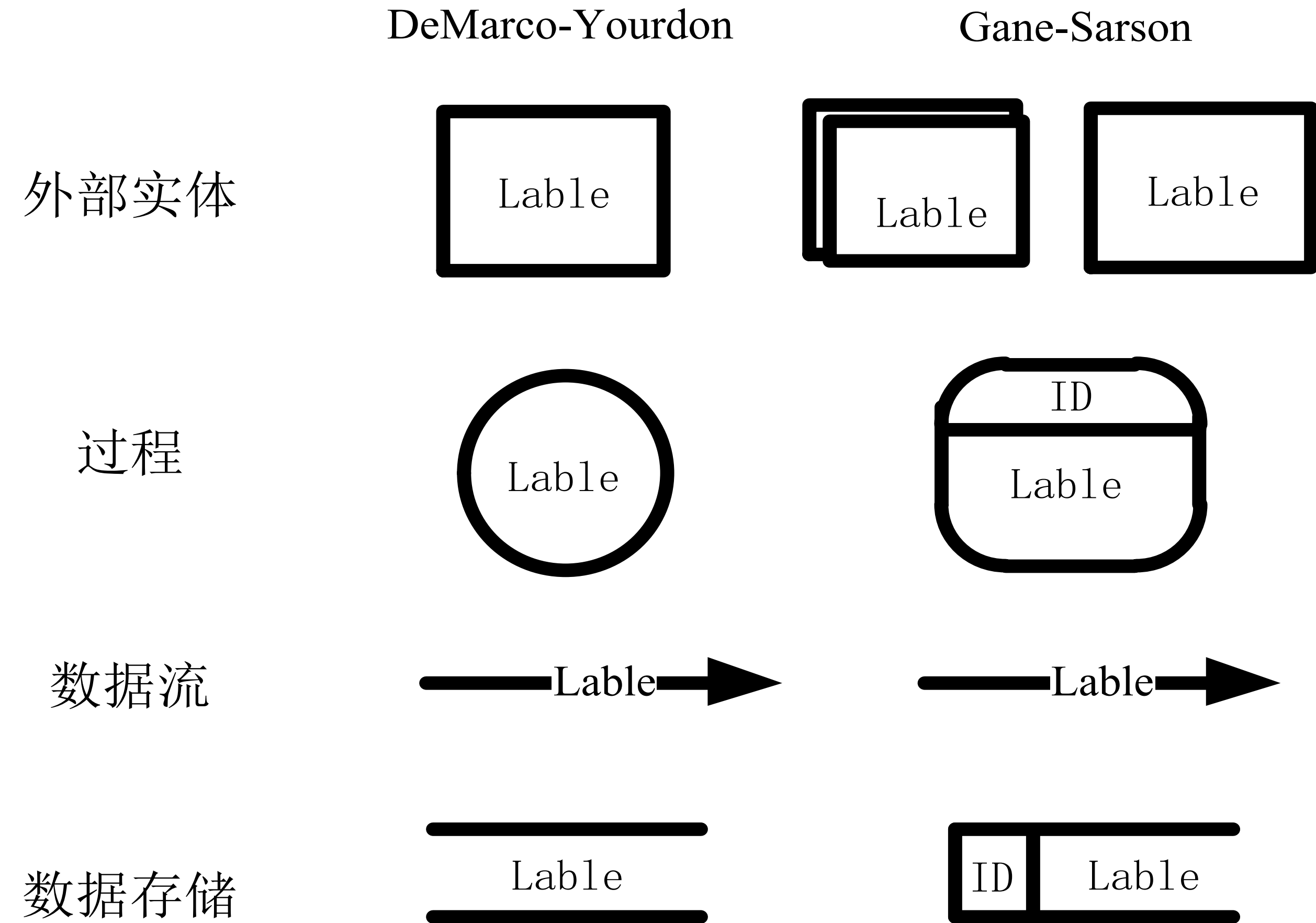
# 数据流图的世界观



所有的计算系统都是信息的处理和转换。

# 过程与数据

- 将系统看做是过程的集合；
- 过程就是对数据的处理：
  - 接收输入，进行数据转换，输出结果
  - 代表数据对象在穿过系统时如何被转换
- 可能需要和软件系统外的实体尤其是人进行交互
- 数据的变化包括：
  - 被转换、被存储、或者被分布



# Flow Modeling Notation

# 外部实体

- 数据的生产者或者消费者
  - 人、设备、传感器
  - 计算机系统
- 数据总是从某处来，然后流向其它的地方



# 过程

- 数据的转换器
  - 计算纳税金额、计算面积、格式化报告、显示图表
- 数据总是被处理然后完成某项业务功能

# 数据流

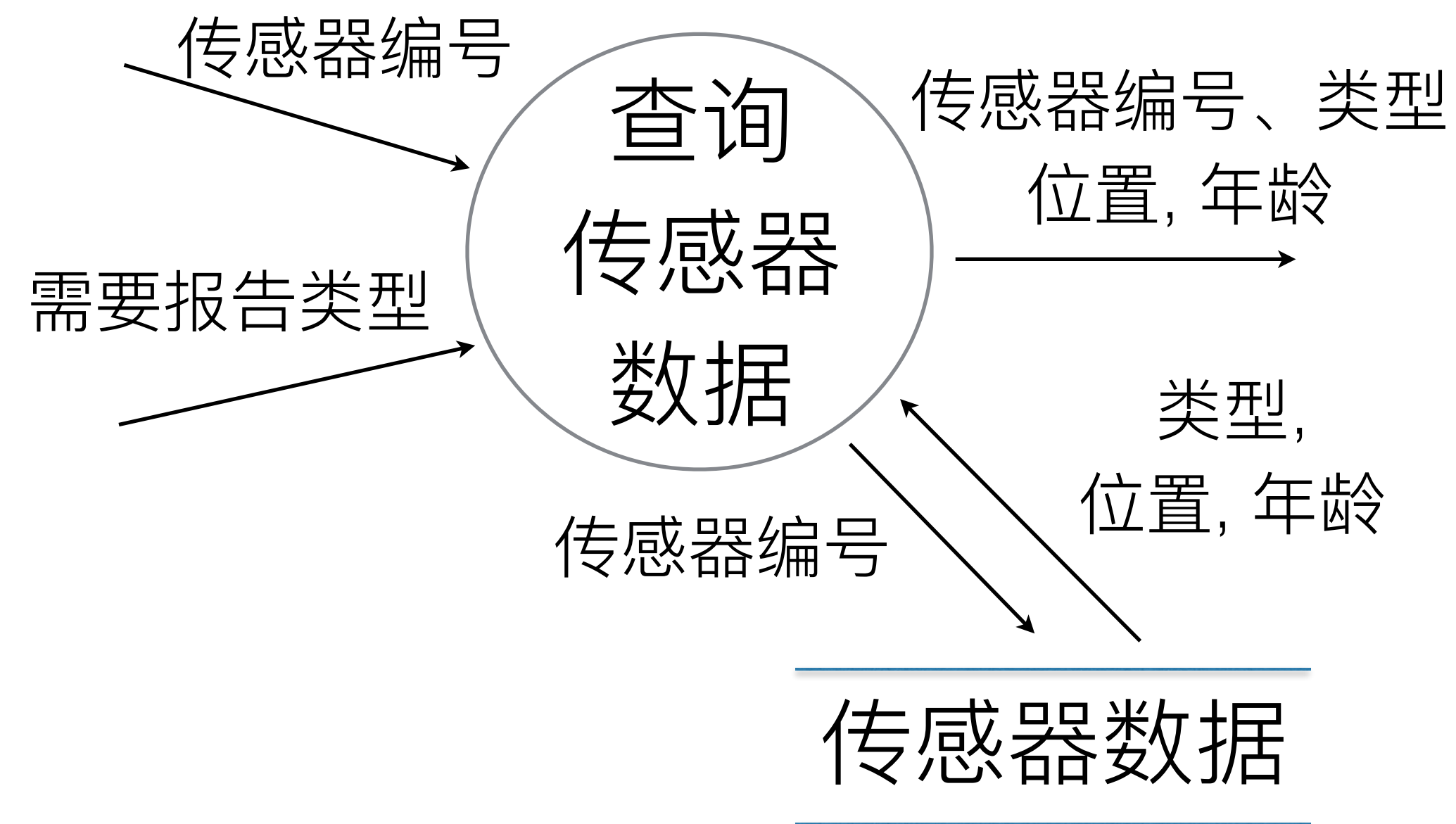
- 通过系统的数据流总是从输入被转换为输出



-

# 数据的存储

- 数据有时候会被存储起来为以后使用做准备。

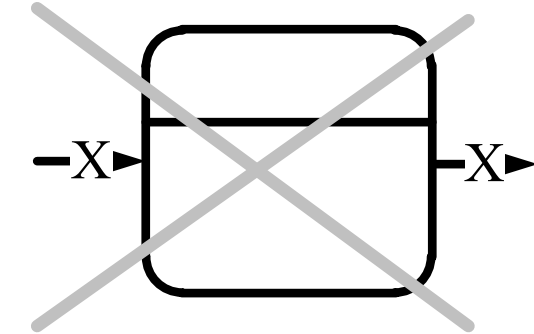
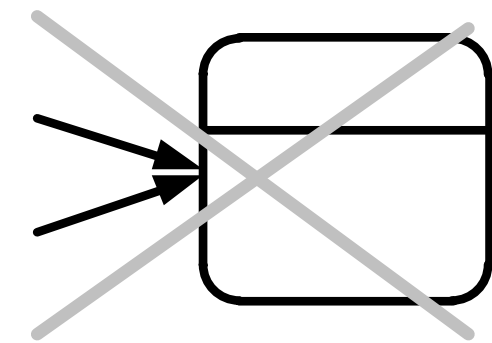
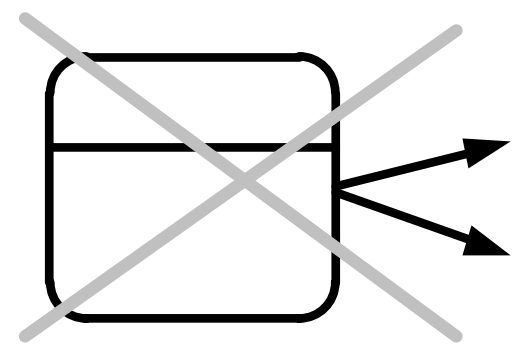


•

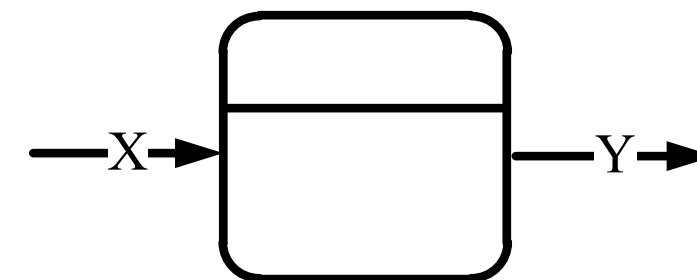
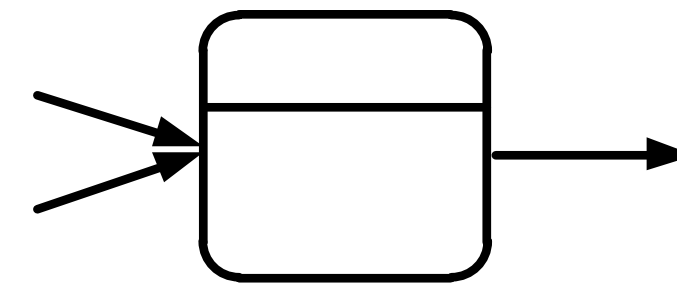
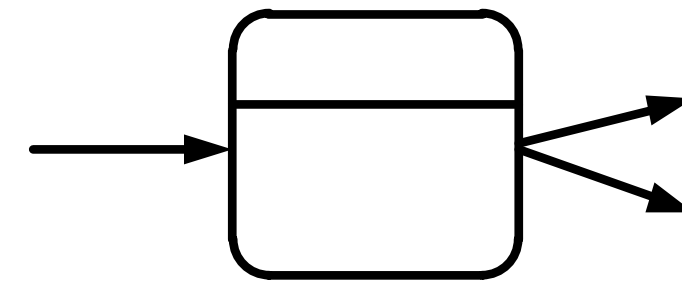
# 语法规则

- 过程是对数据的处理，必须有输入，也必须有输出，输入数据集应该和输出数据集存在差异

错误的数据流



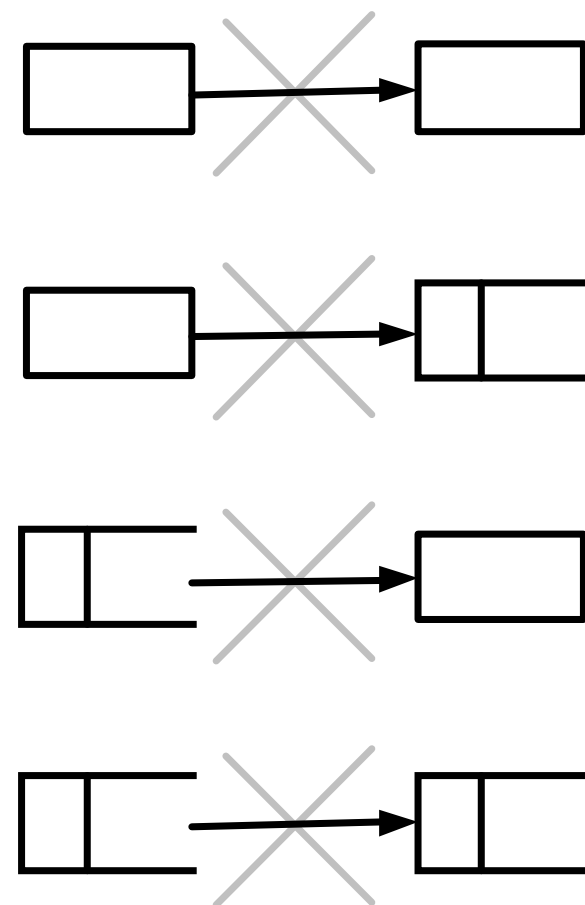
正确的数据流



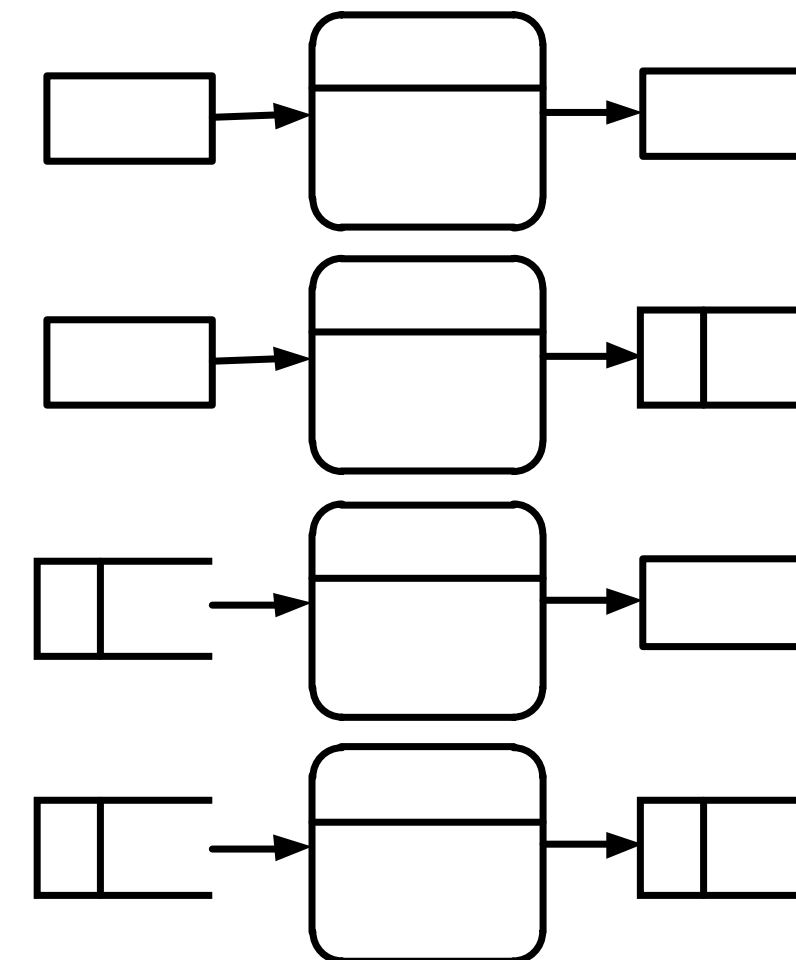
# 语法规则

- 数据流是必须和过程产生关联的，它要么是过程的数据输入，要么是过程的数据输出
- 所有的对象都应该有一个可以唯一标示自己的名称。

错误的数据流



正确的数据流



# 理解数据流图

- 处理
  - 并不一定是程序。
  - 它可以是一系统程序、单个程序或程序的一个模块，甚至可以是人工处理过程；
- 数据存储
  - 并不等同于一个文件。
  - 它可以是一个文件、文件的一部分、数据库元素或记录的一部分；它代表的是静态的数据。
- 数据流
  - 也是数据，是动态的数据。

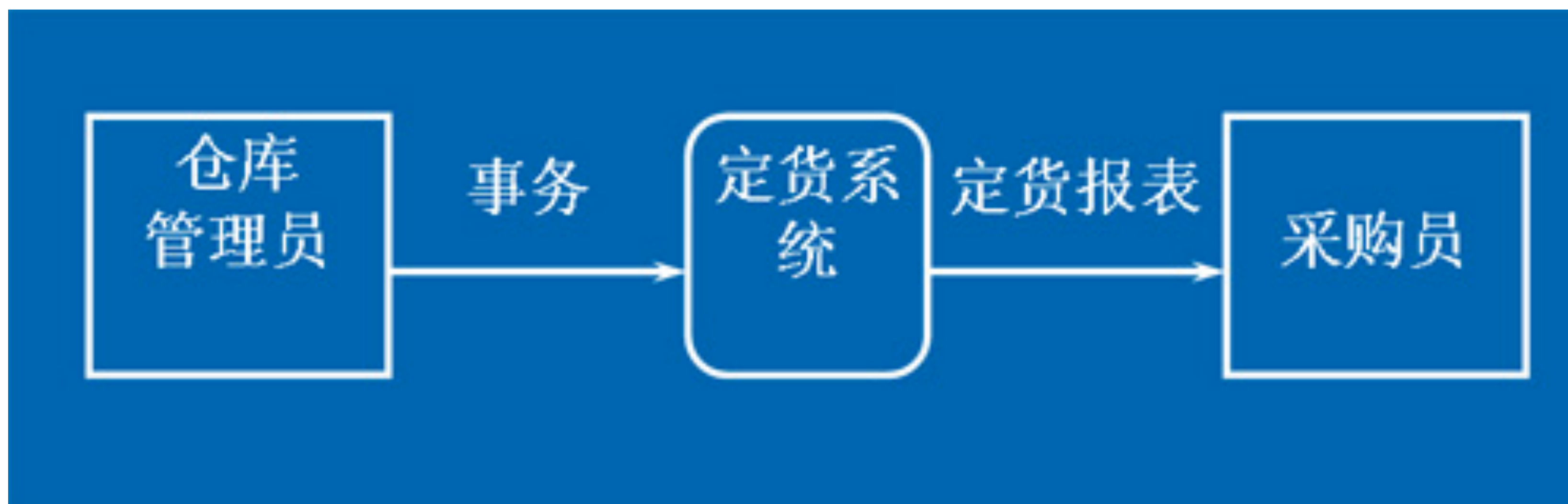
# 案例——订货系统

- 假设一家工厂的采购部门每天需要一张定货报表。报表按零件编号排序，表中列出所有需要再次定货的零件。
- 对于每个需要再次定货的零件应该列出下述数据：零件编号、零件名称、定货数量、目前价格、主要供应商、次要供应商。
- 零件入库或出库称为事务，通过放在仓库中的CRT终端把事务报告给定货系统。当某种零件的库存数量少于库存临界值时就应该再次定货。

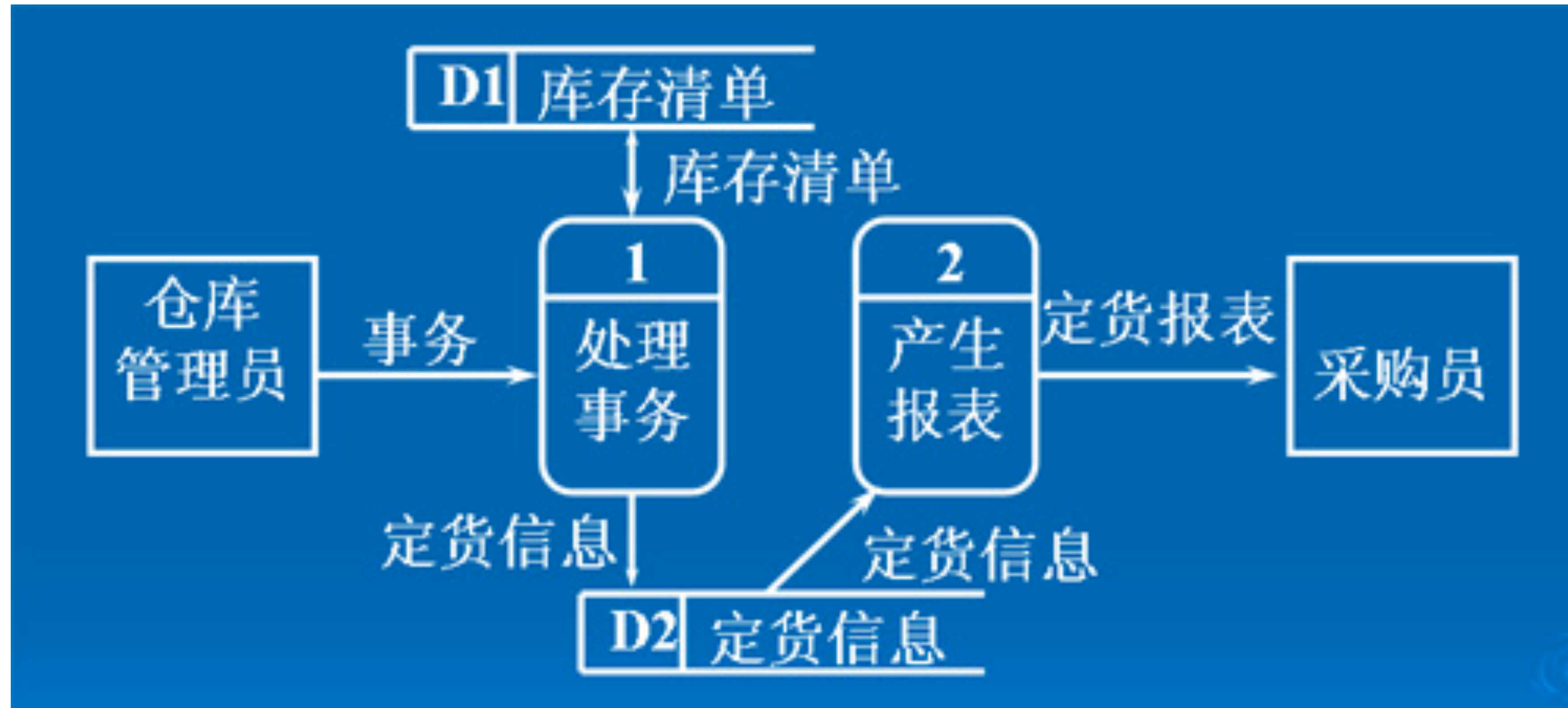
# 案例

- 源点/终点（外部实体）
  - 采购员
  - 仓库管理员
- 数据处理
  - 产生报表
  - 处理事务
- 数据流
  - 定货报表
    - 零件编号
    - 零件名称
    - 定货数量
    - 目前价格
  - 主要供应商
  - 次要供应商
  - 事务
    - 零件编号
    - 事务类型
    - 数量
  - 数据存储
    - 定货信息（见定货报表）
    - 库存清单
      - 零件编号
      - 库存量
      - 库存量临界值

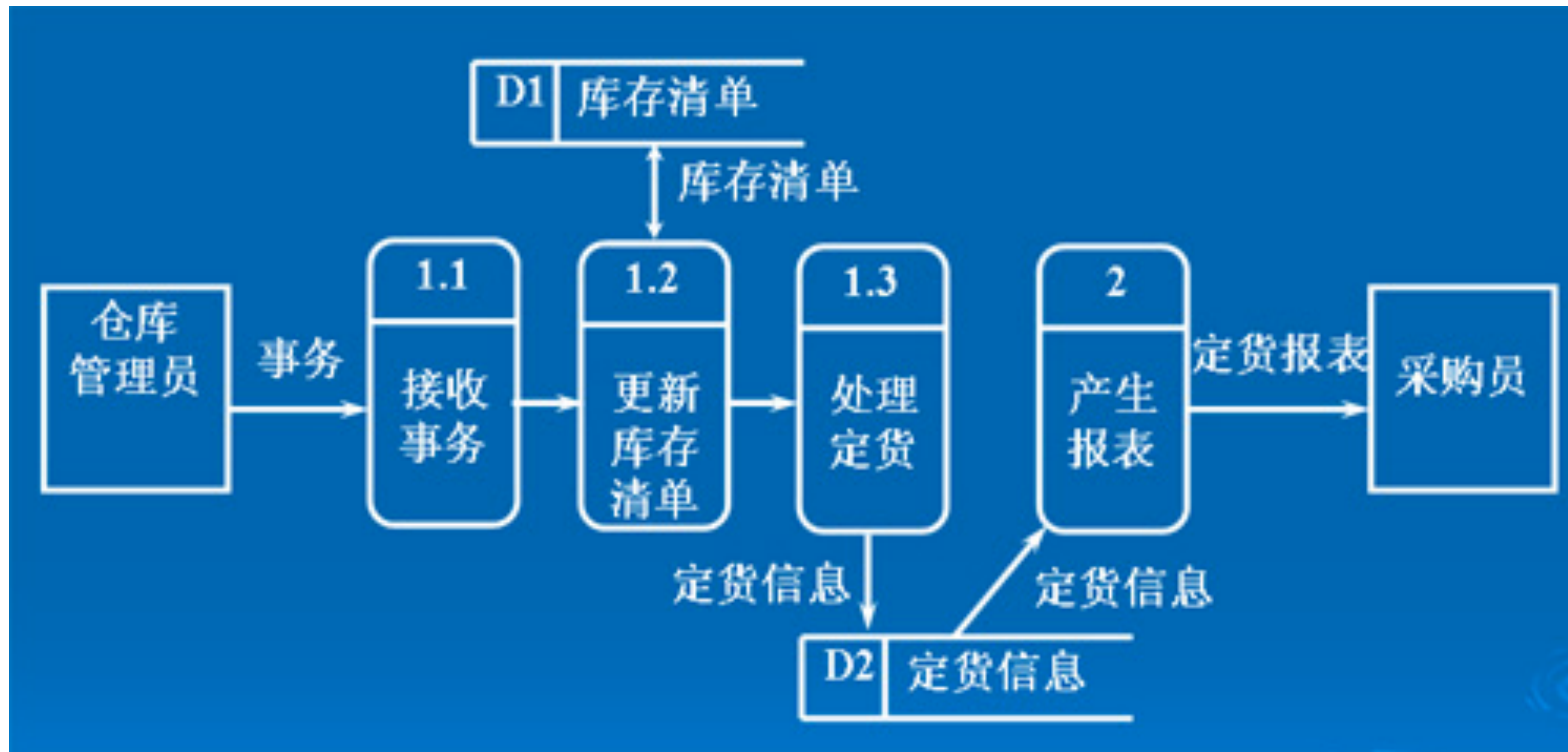




最概括的系统模型

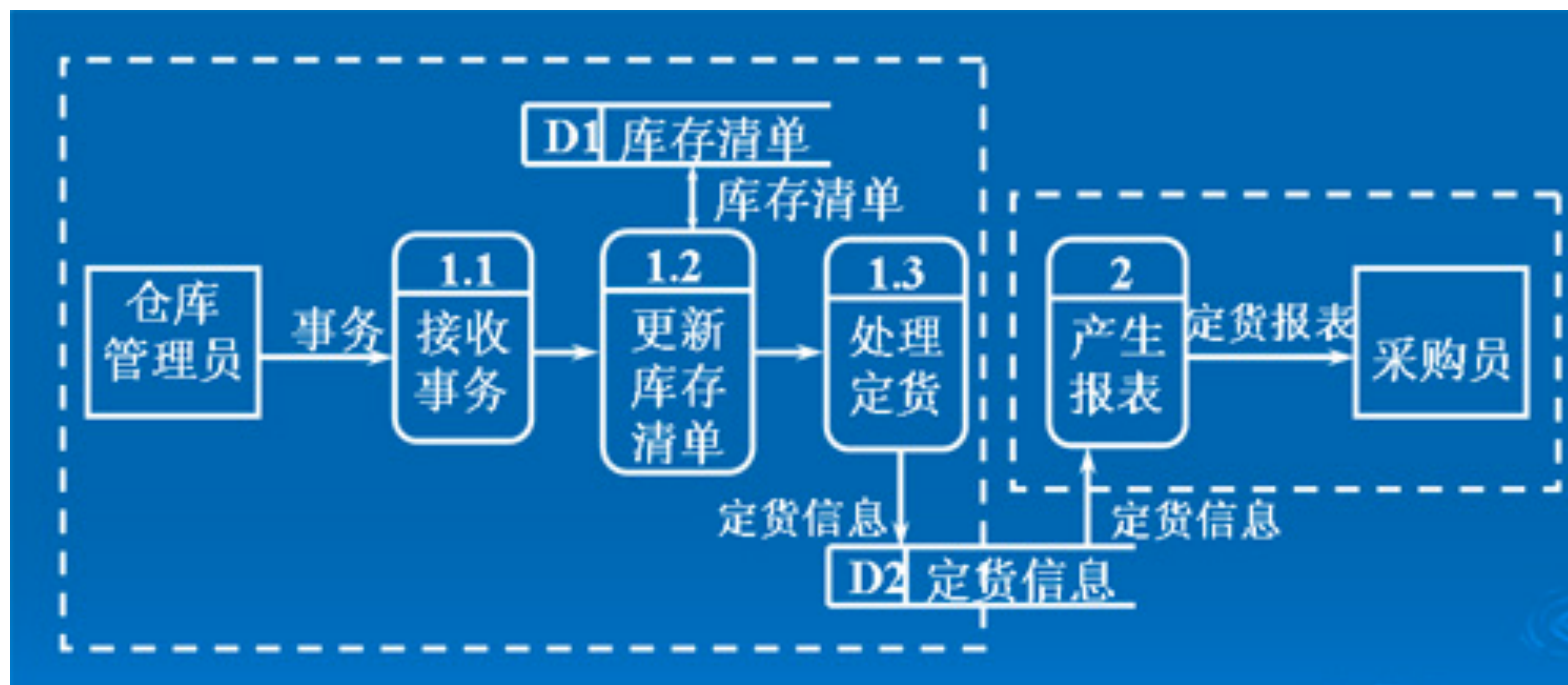


细化数据存储和数据流

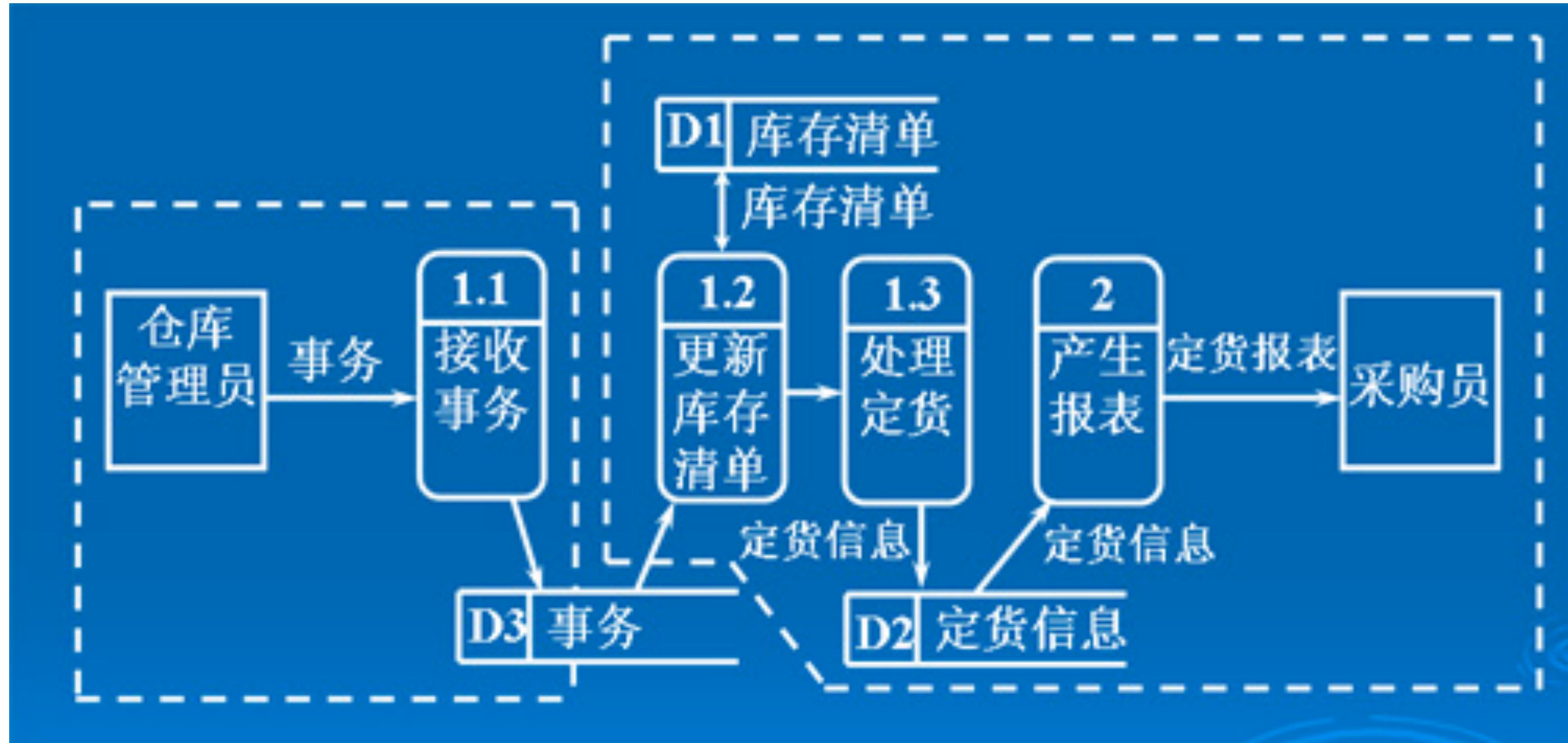


子过程





边界



细化

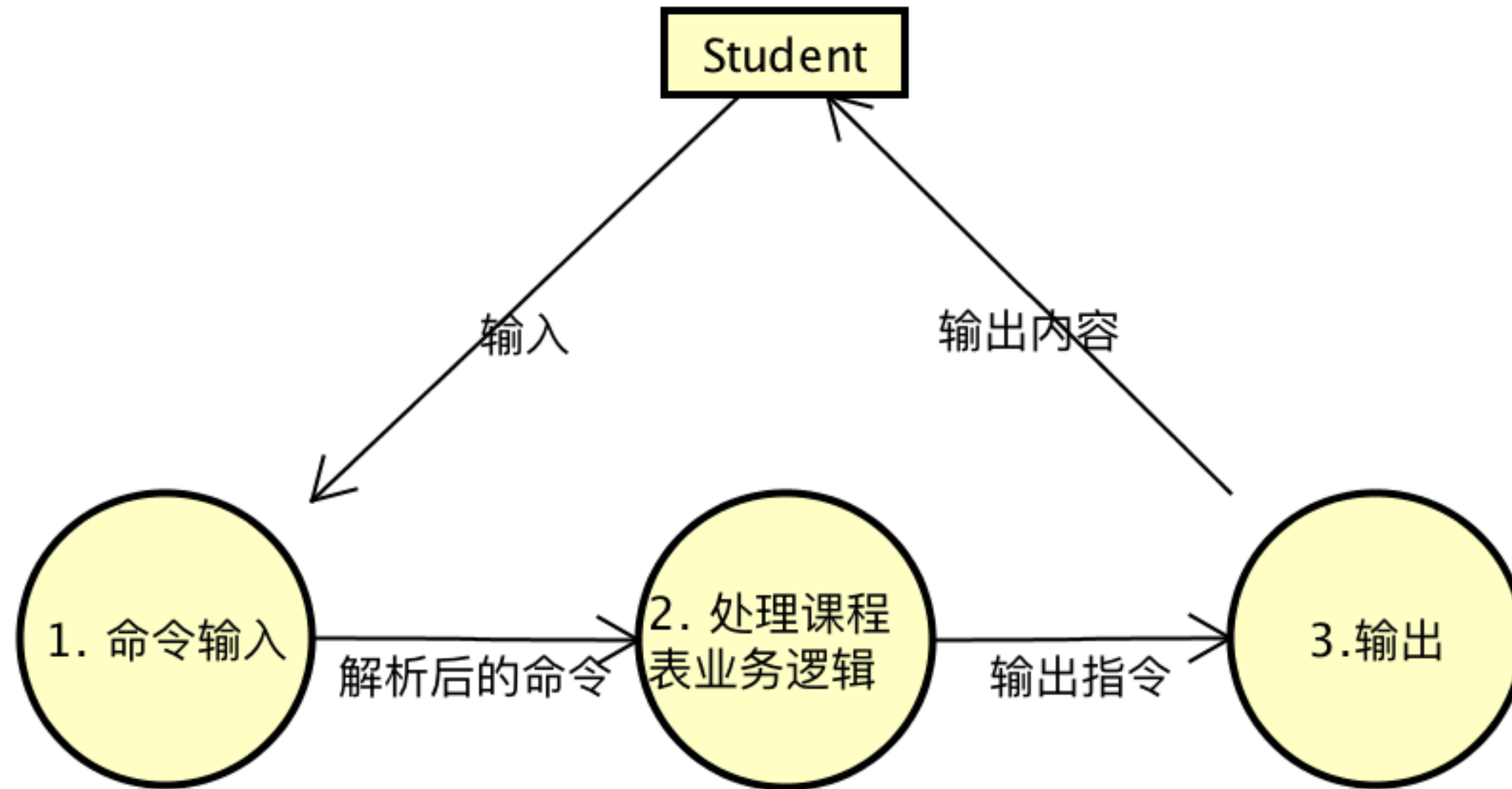
# 课程表案例

- 建一个课程表。
  - 星期四；三，四节；计算与软件工程；仙2-407；
- 通过命令行方式完成对课程的增、删、改、查、显示。
  - Add 星期四；三，四节；计算与软件工程；仙2-407； //如果成功 显示“已添加到文件中”
  - Remove 星期四；三，四节；计算与软件工程；仙2-407； //如果成功 显示“已从文件删除”
  - Update 星期四；三，四节；计算与软件工程；仙2-408； //如果成功 显示“已更新文件”
  - Find 星期四；三，四节； //如存在课程 显示 “课程名； 上课地点”
  - Show //显示所有课程， 按照时间排序
- 数据保存在文件里。
  - CurriculumSchedule.txt

# 课程表案例分析

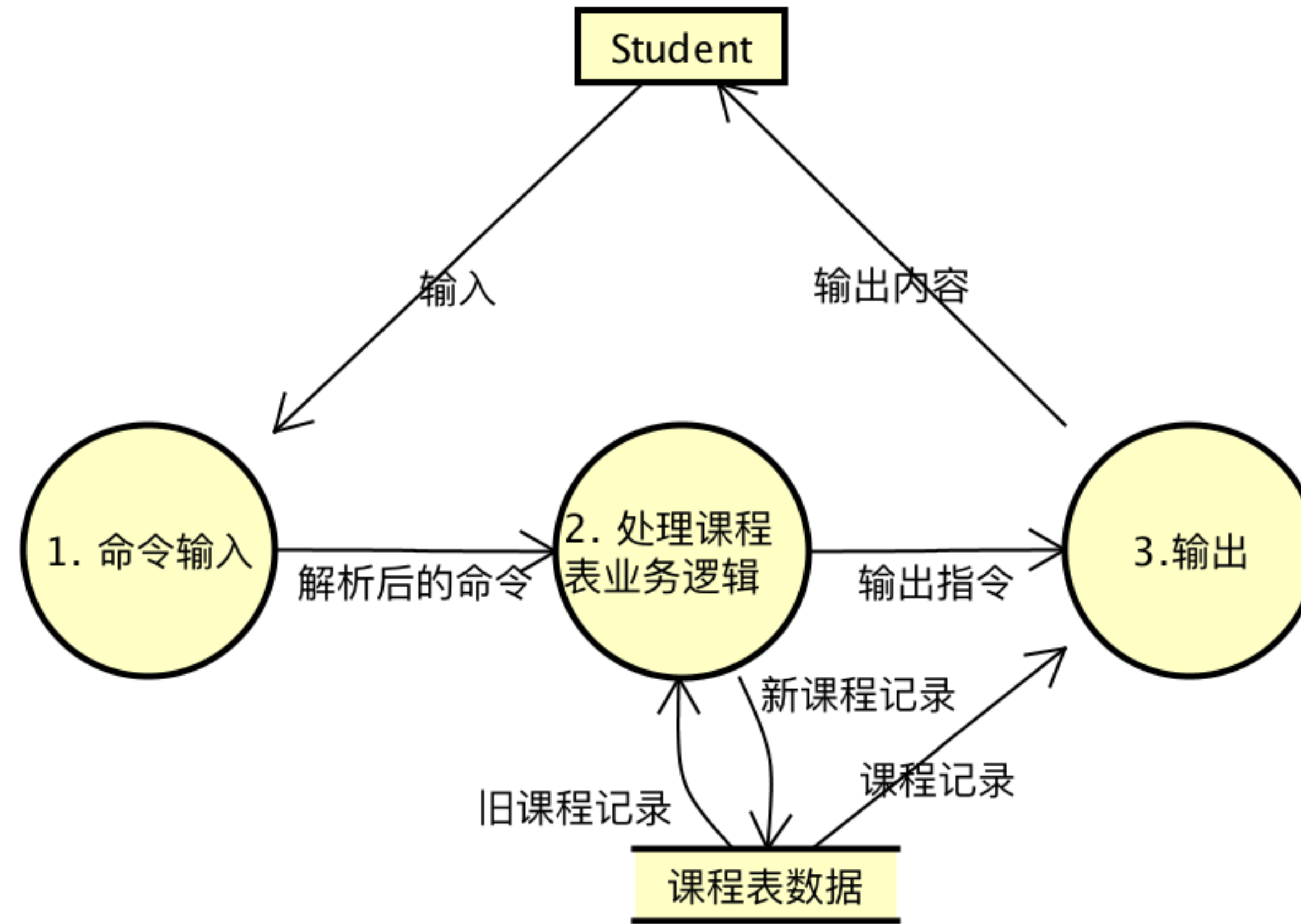
- 源点/终点（外部实体）
  - 学生
- 数据处理
  - 命令输入
    - 输入
    - 分析输入
  - 处理课程表业务逻辑
    - 增
    - 删
    - 改
    - 查
- 显示
  - 输出
    - 生成输出内容
    - 文件输出
    - 控制台输出
- 数据流
  - 命令
- 数据存储
  - 课程表数据
  - 文件地址



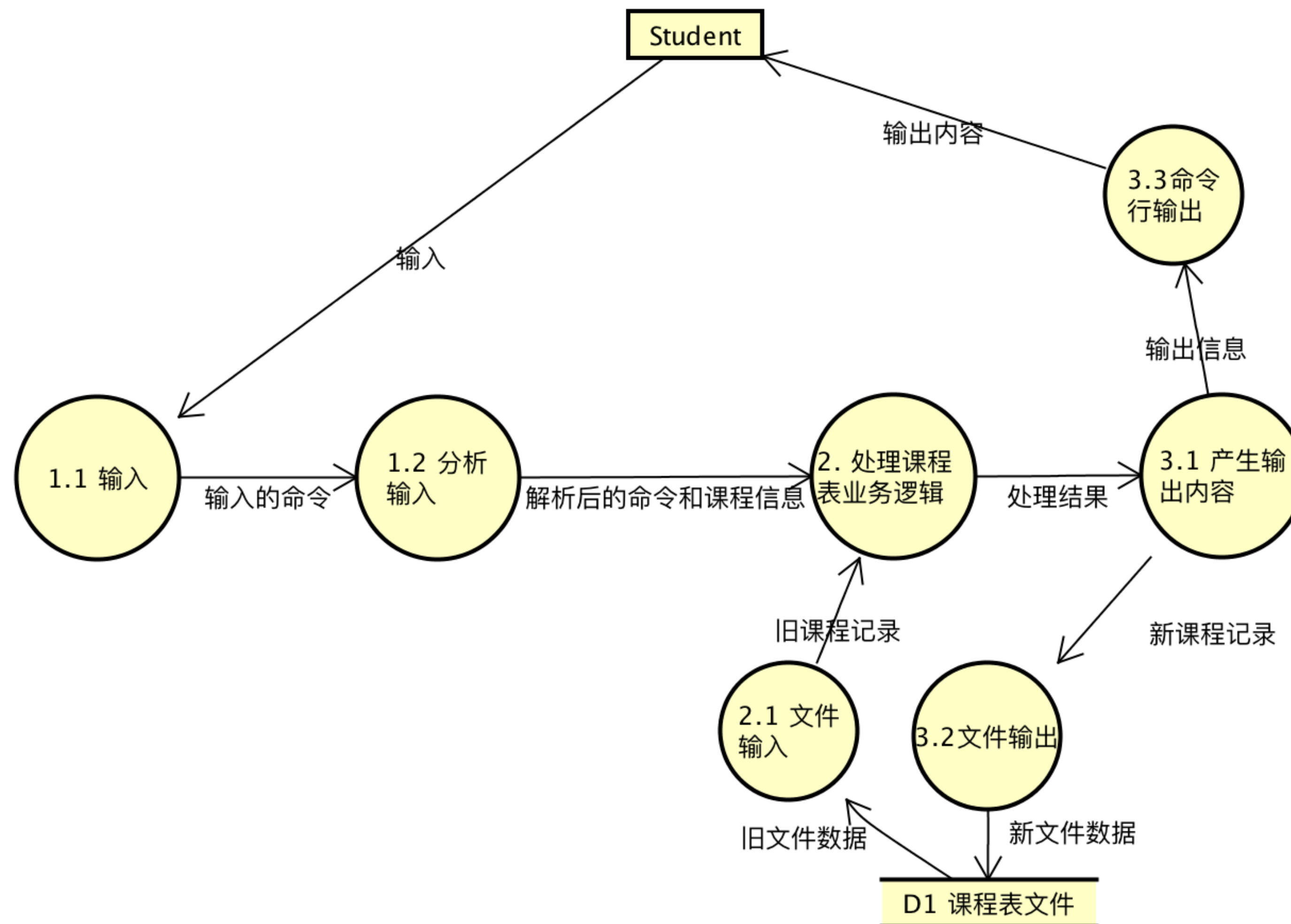


## 数据流图 - 概括





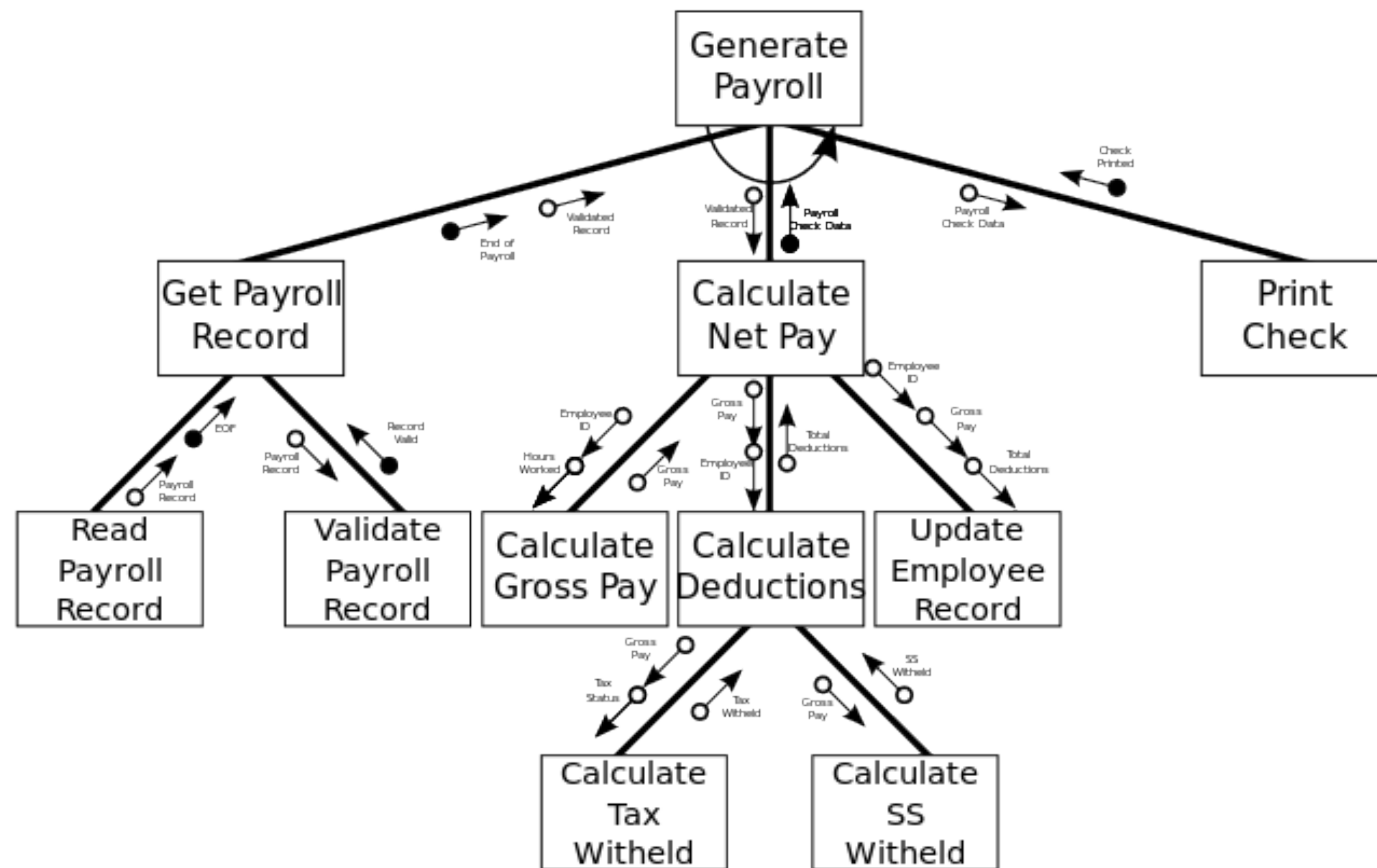
# 数据流图 - 细化



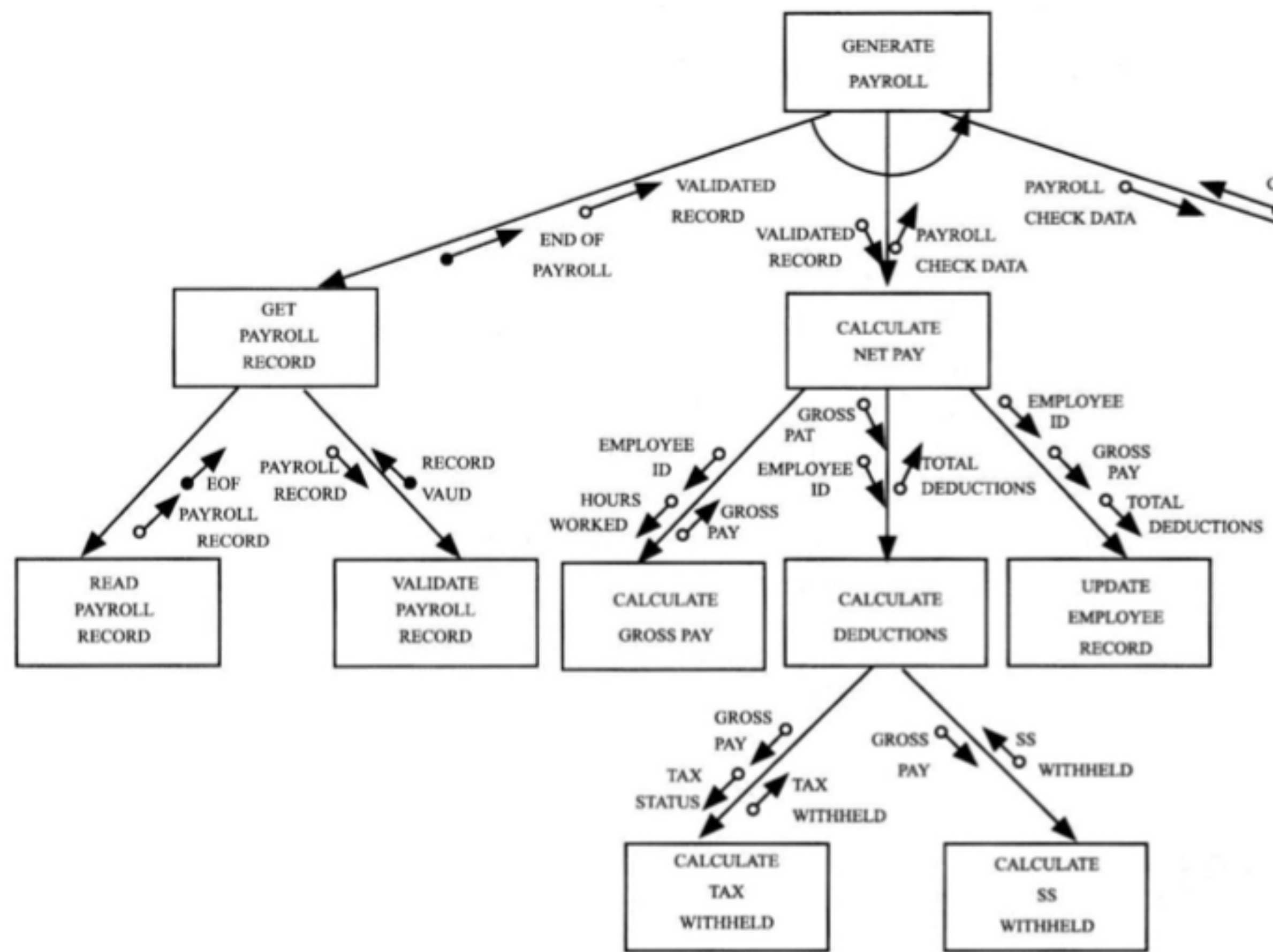
# 数据流图 - 子过程

# Outline

- 结构化编程思想
  - 思想和模型
  - 数据流图
  - 结构图
  - 流程图



结构图 (Structured Chart)



Modules



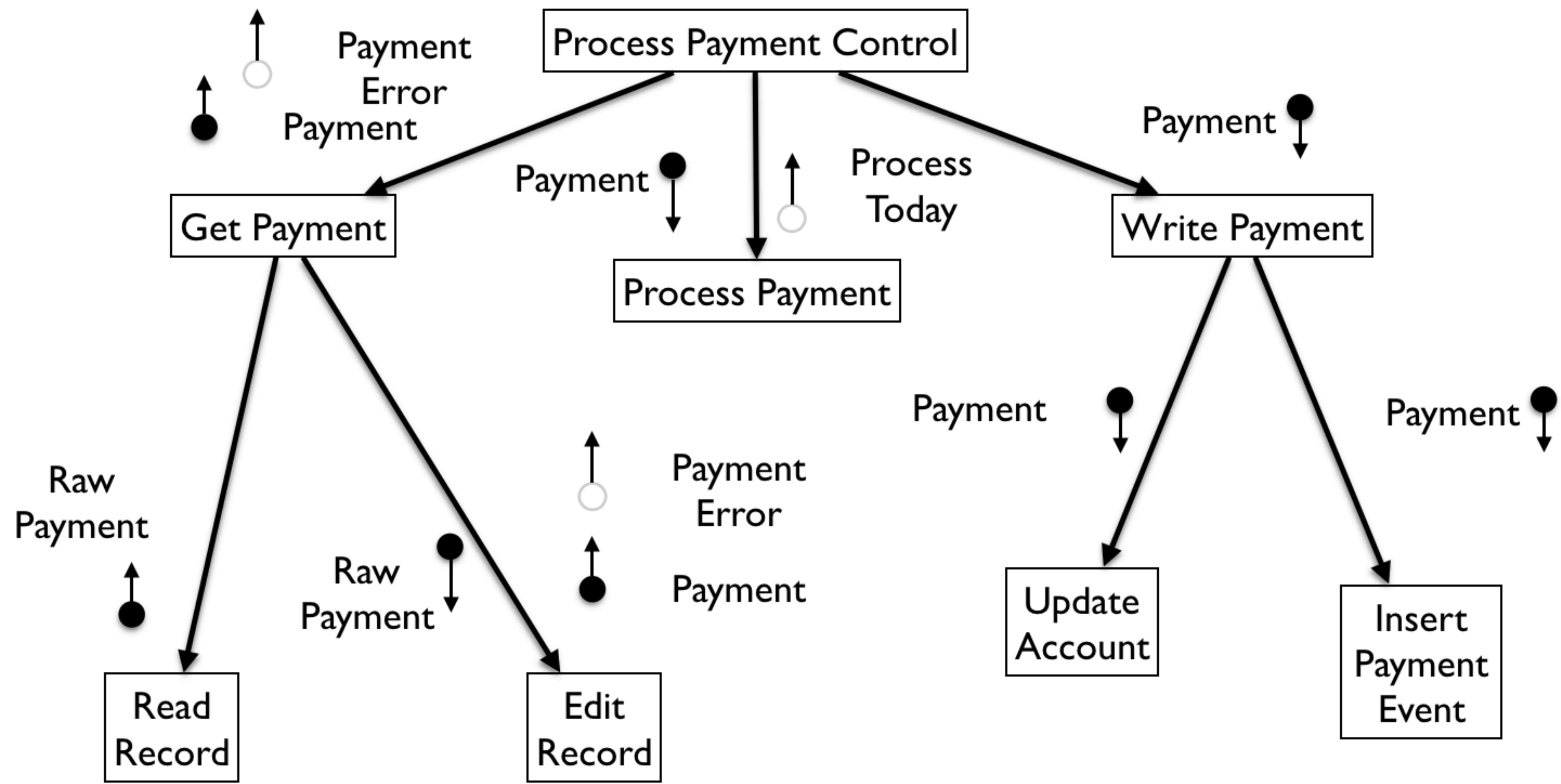
Library modules

————— Module call

○ ——— Data

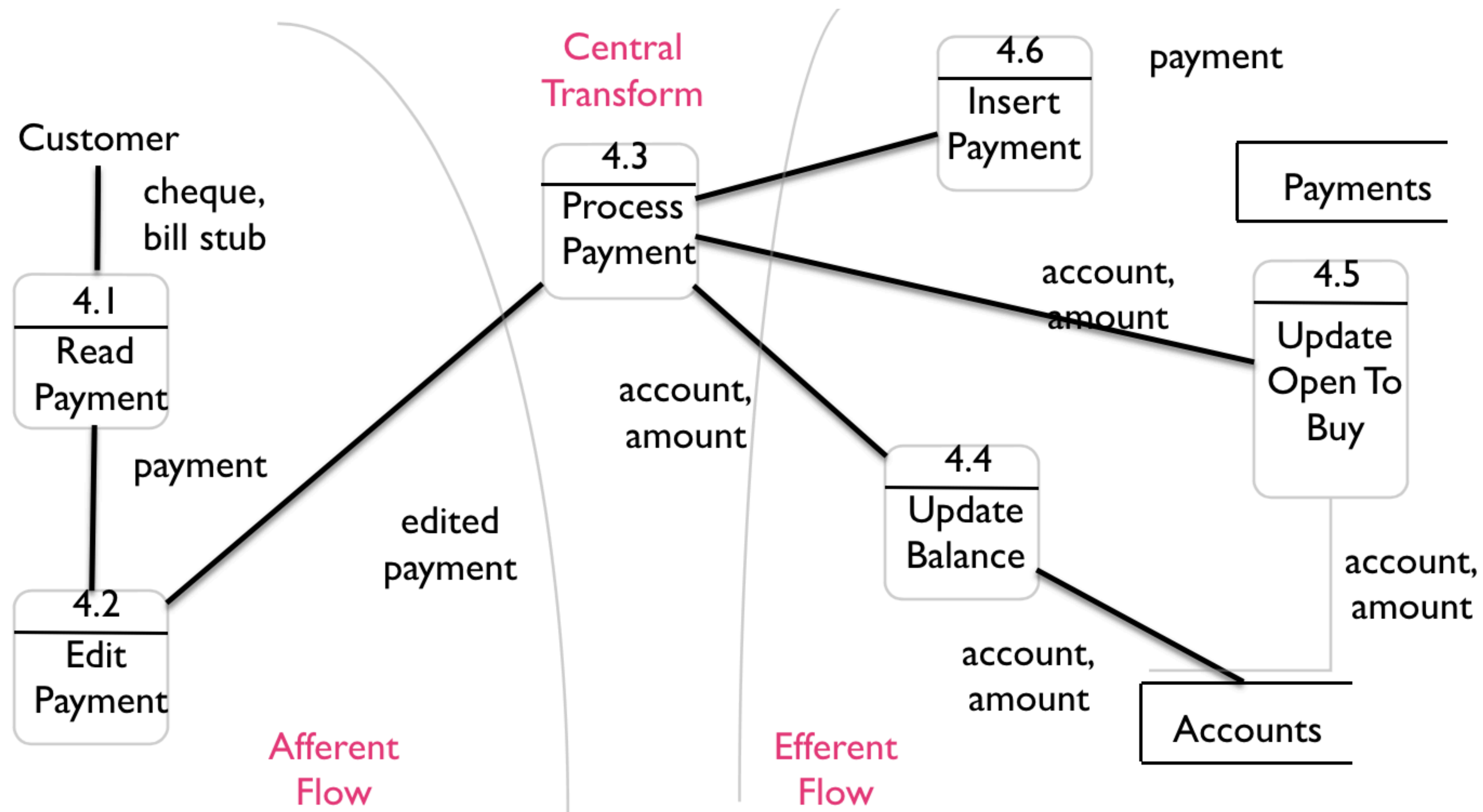
○ ——— Flag

结构图



结构图



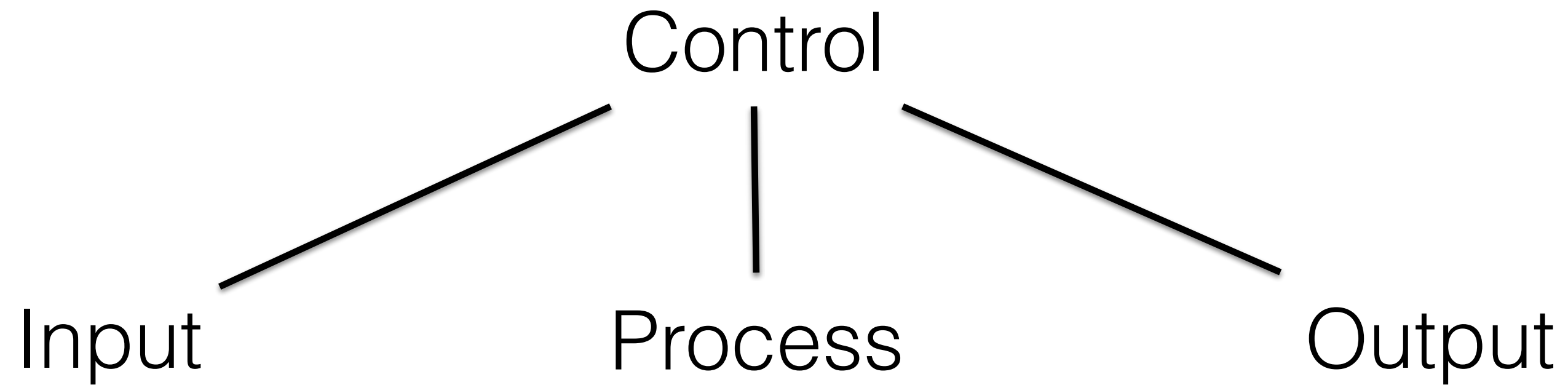
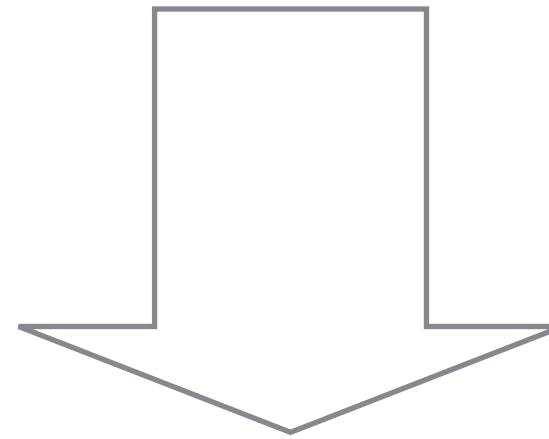


Modularity and Information hiding!

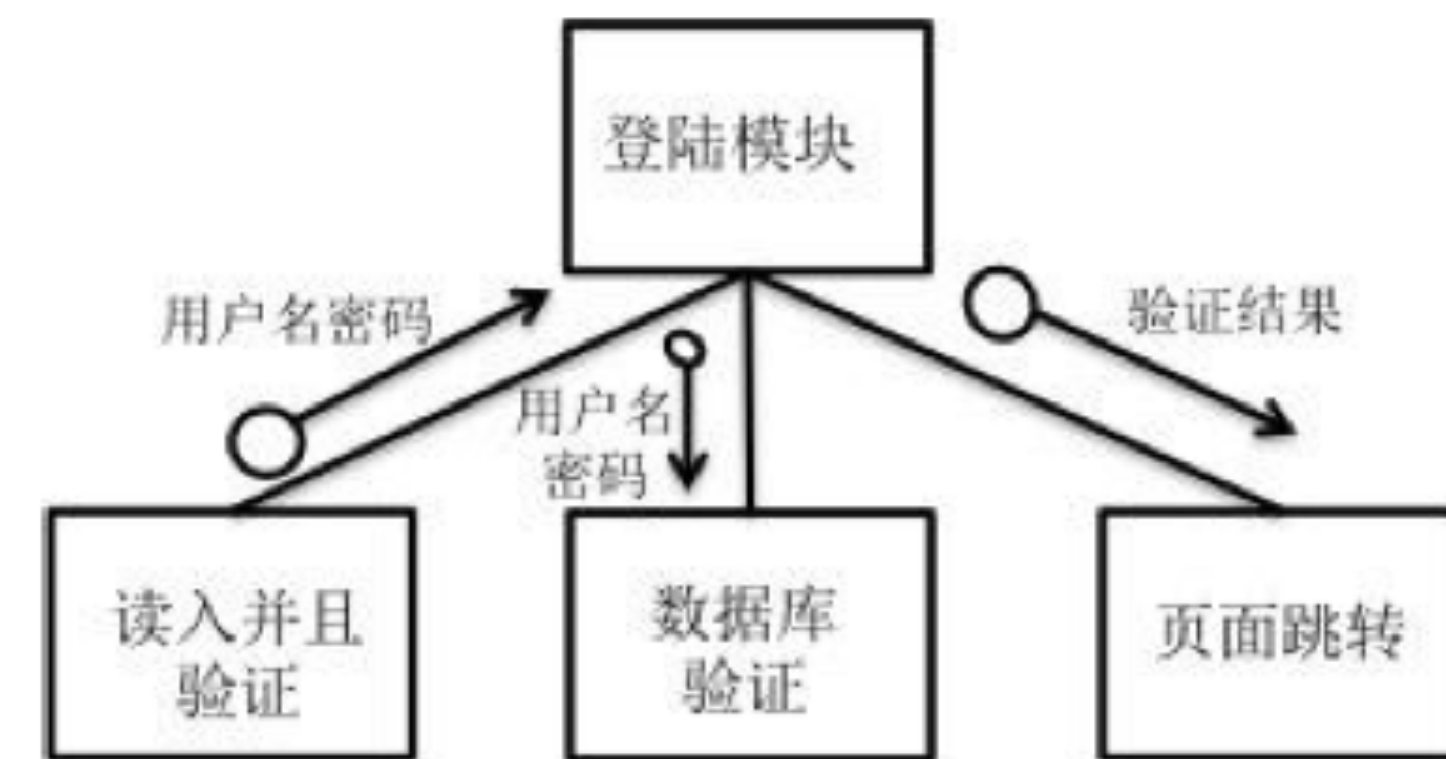
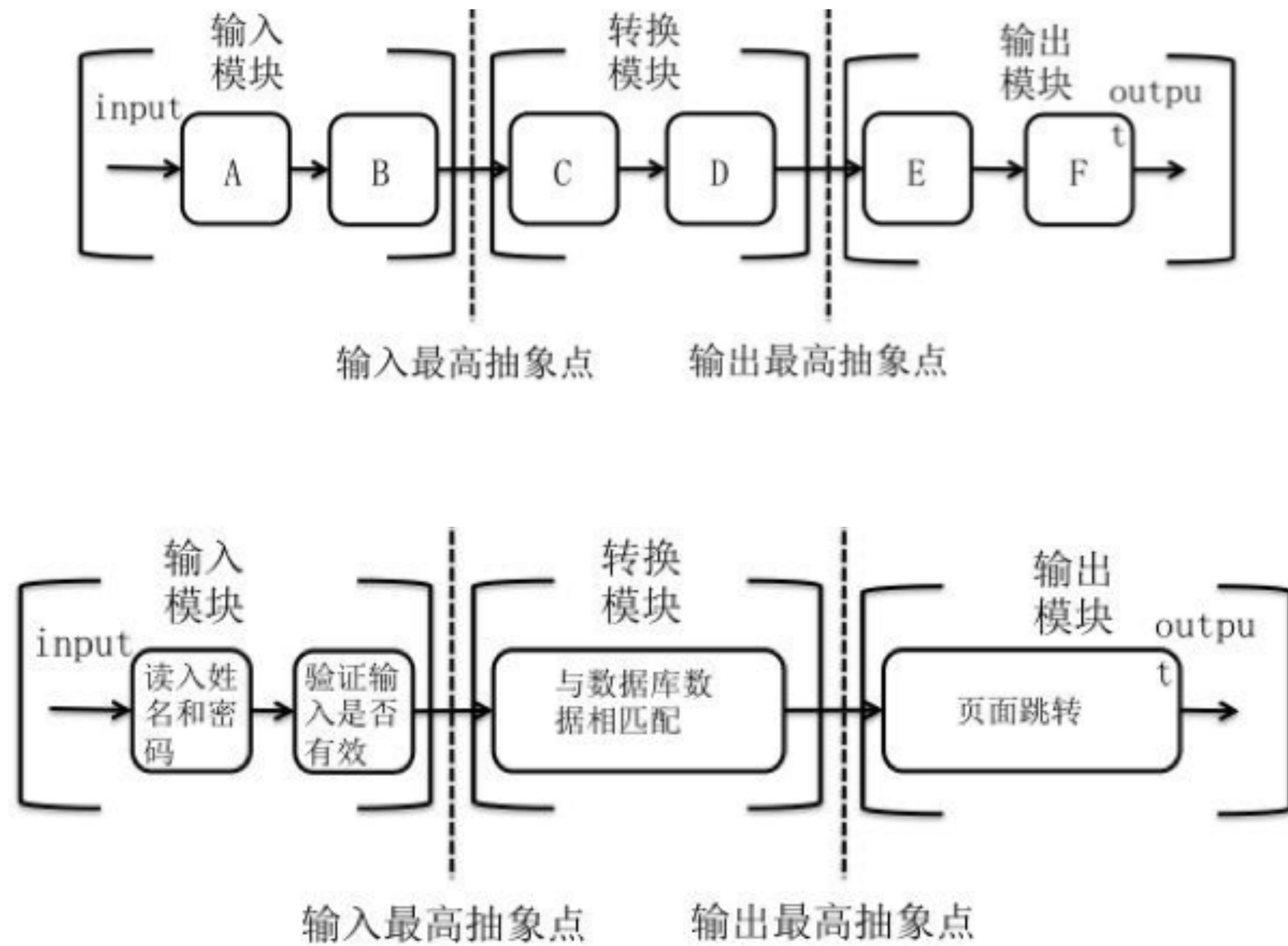
# 转换分析

# 结构图

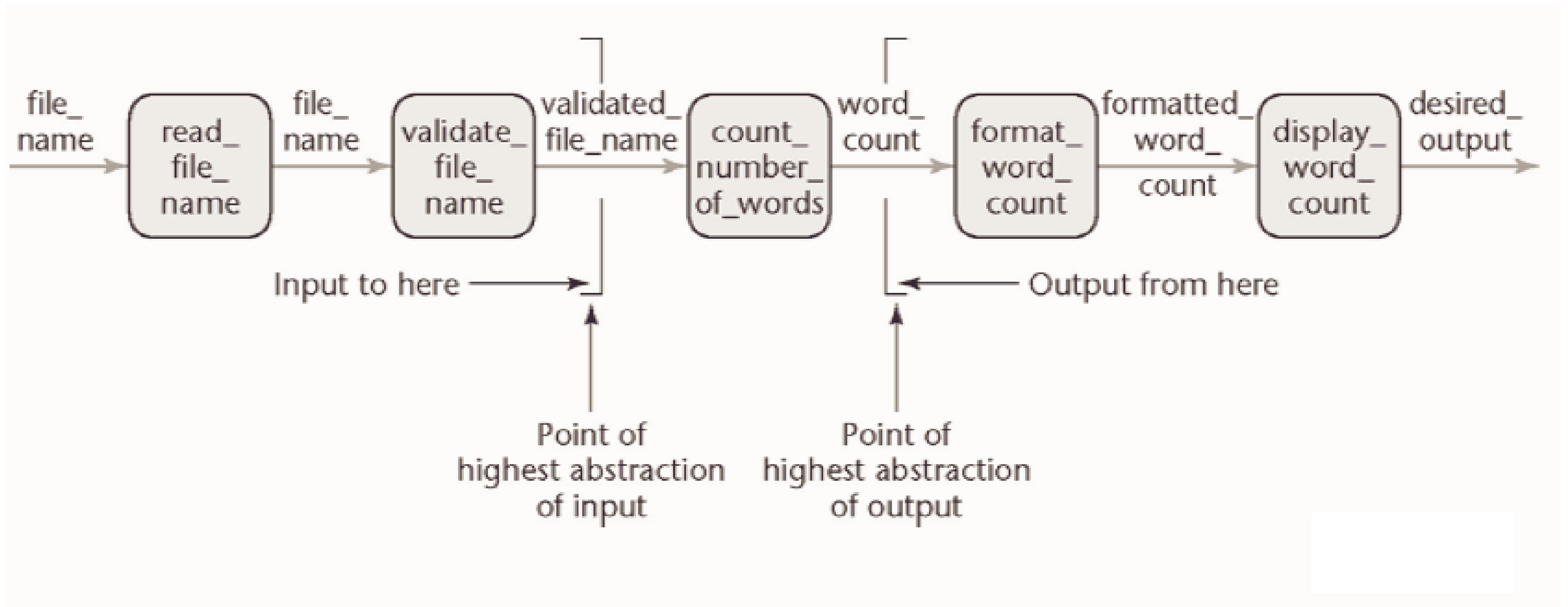
Input (Afferent Flow) — Process Central Transform — Output (Efferent Flow)



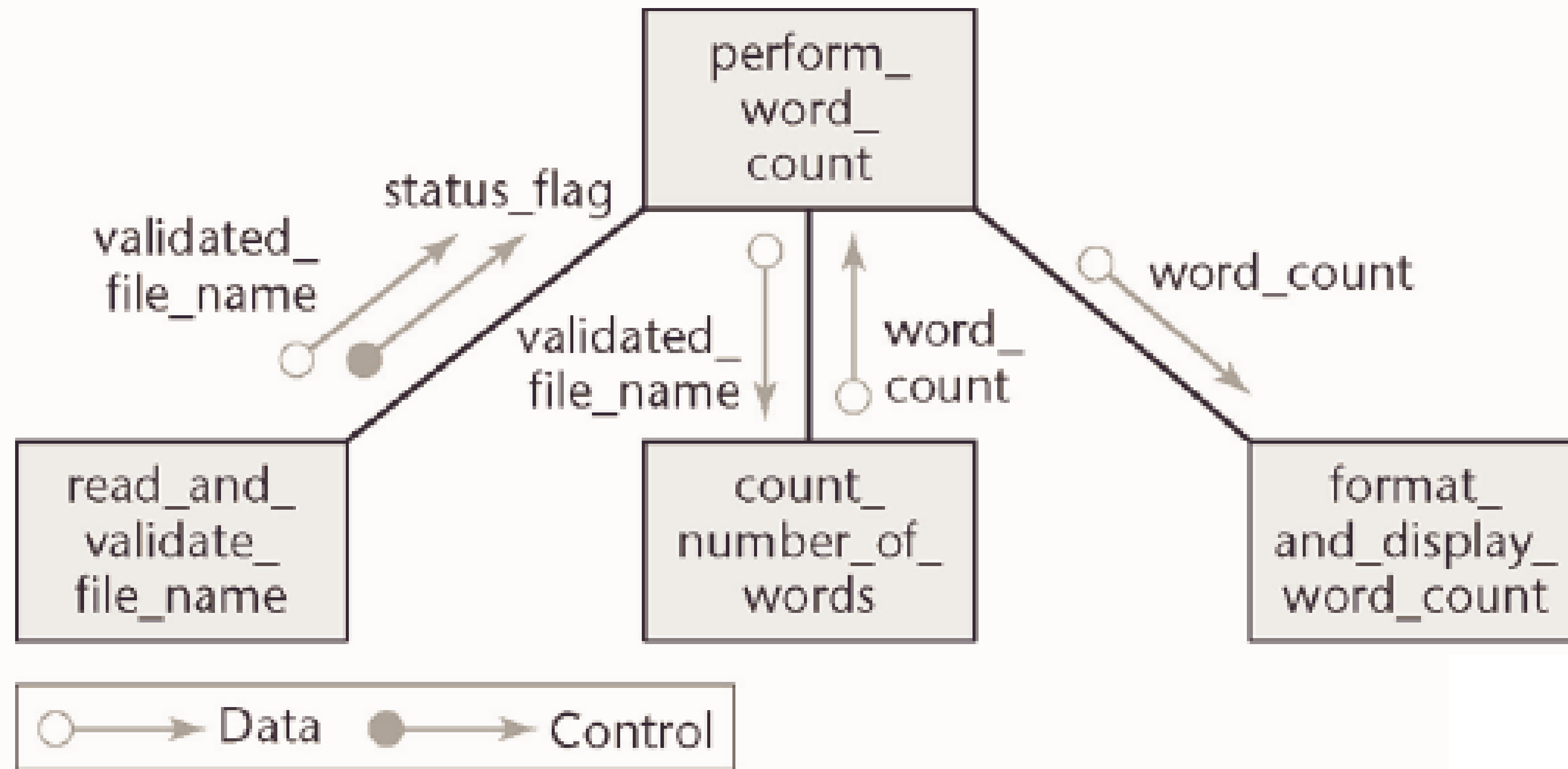




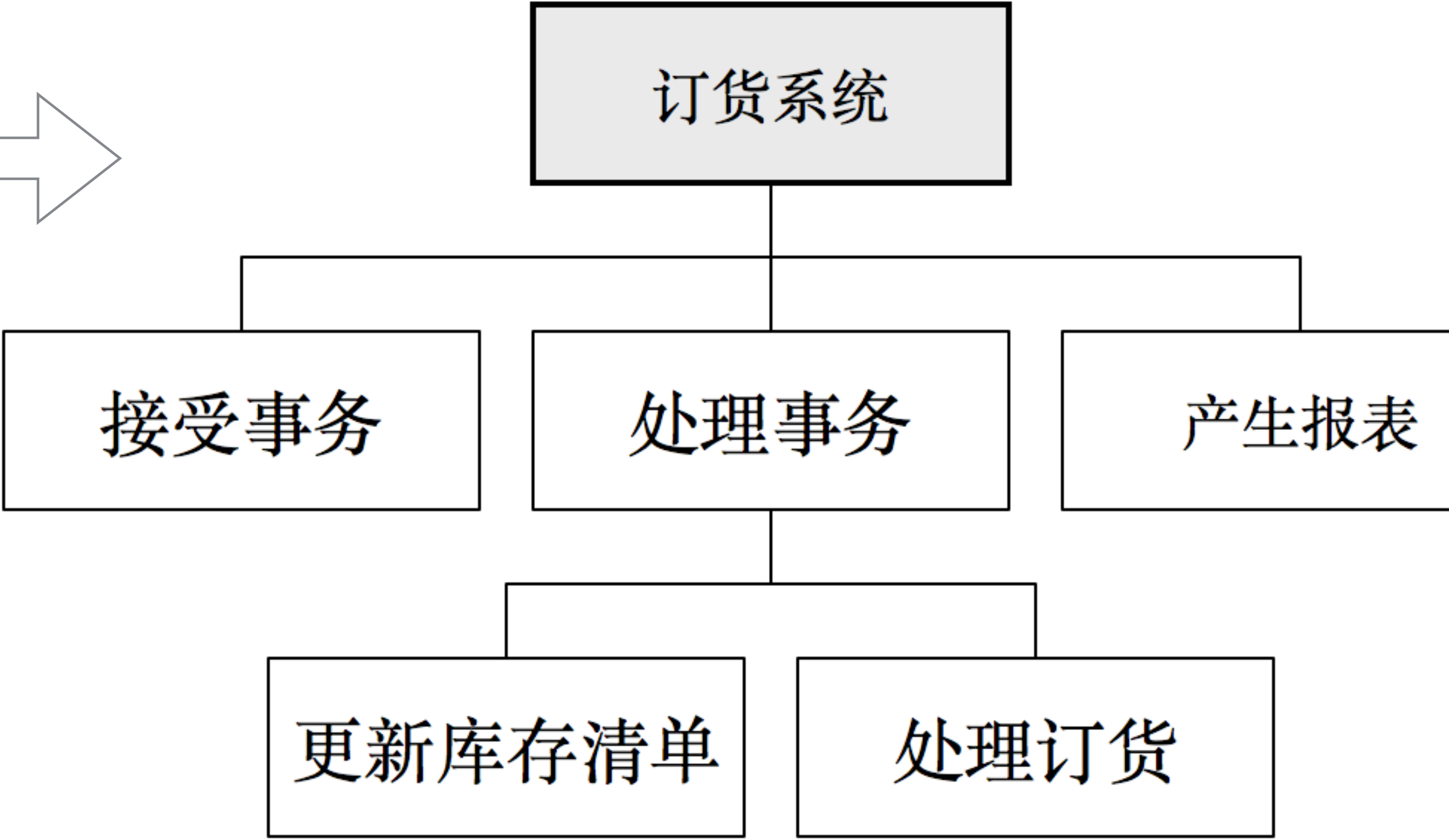
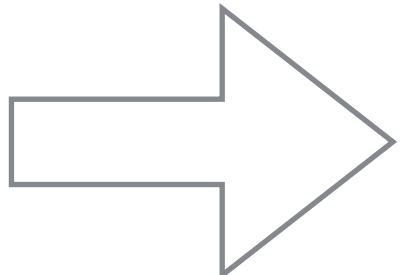
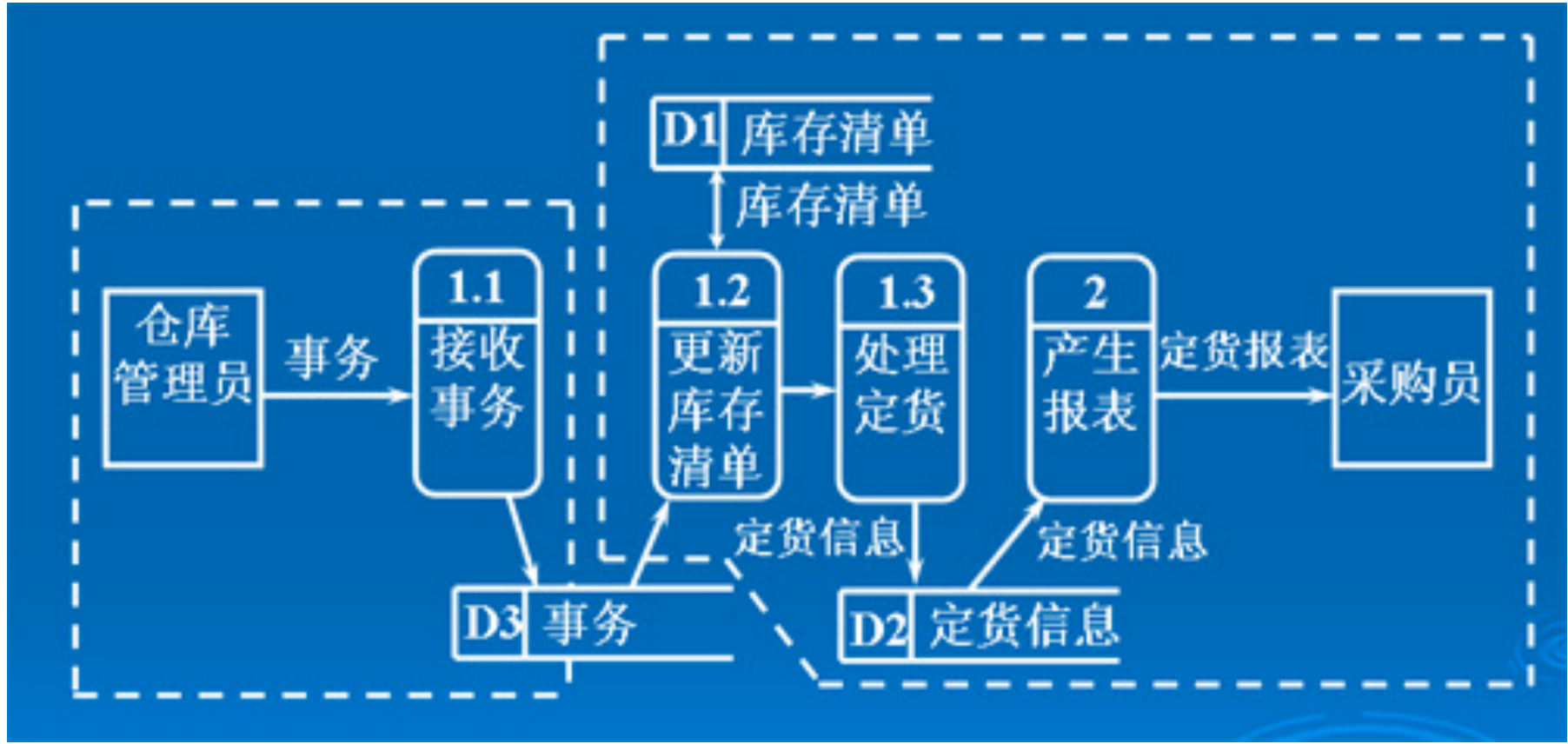
# 数据流图向结构图的转换



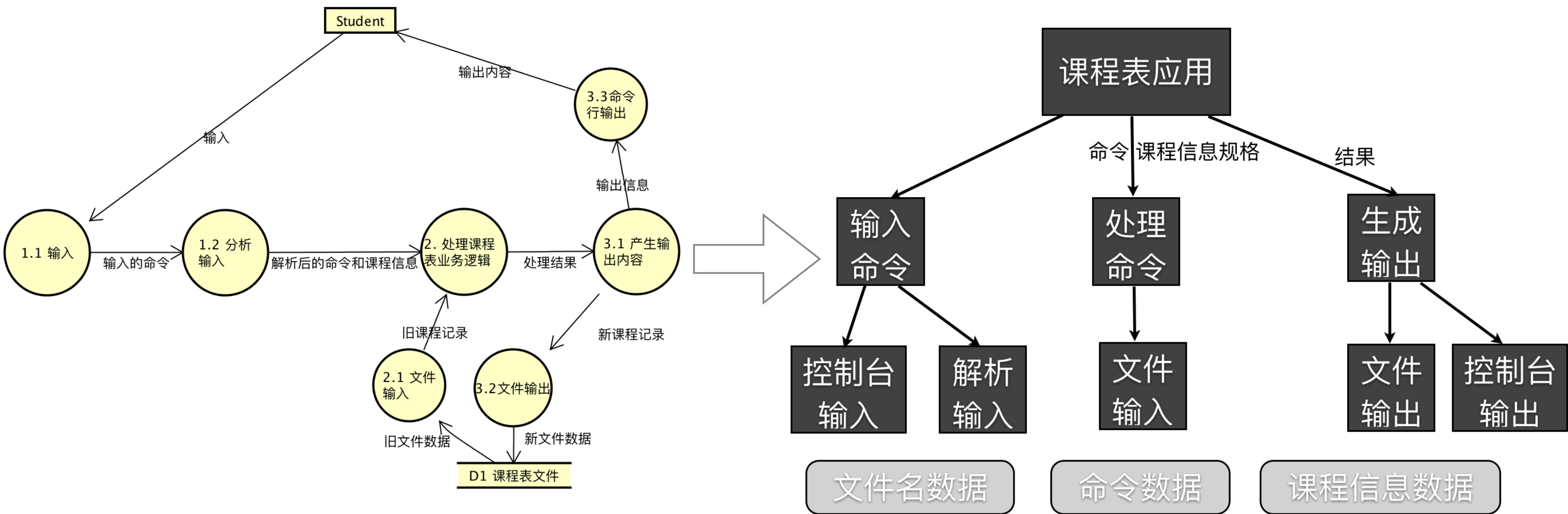
案例



案例



订货系统的结构图



课程表结构图

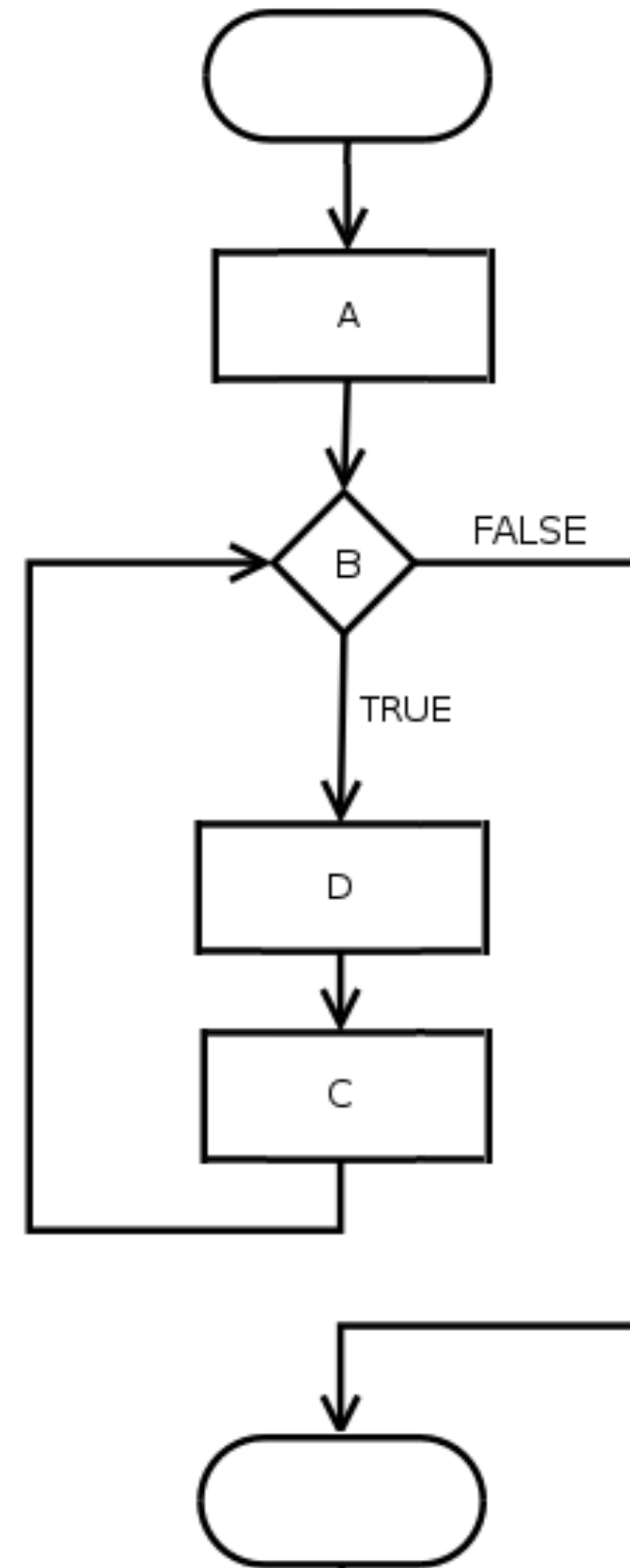
# Outline

- 结构化编程思想
  - 思想和模型
  - 数据流图
  - 结构图
  - 流程图

# 算法

- 增加课程
  - 检查是否可以增加
    - 如果可以增加
      - 增加课程信息
      - 显示已增加
    - 否则
      - 显示已经存在课程，无法添加

for(A;B;C)  
D;



流程图



# 数据结构

- 课程表数据
  - 数组 or ArrayList
  - 全局变量
- 文件地址
  - char[] or String
  - 文件名作为常量

# 总结： Structured Programming

- 行为视角
  - 首先根据行为来分解
  - 接着设计数据来配合行为
    - 全局数据