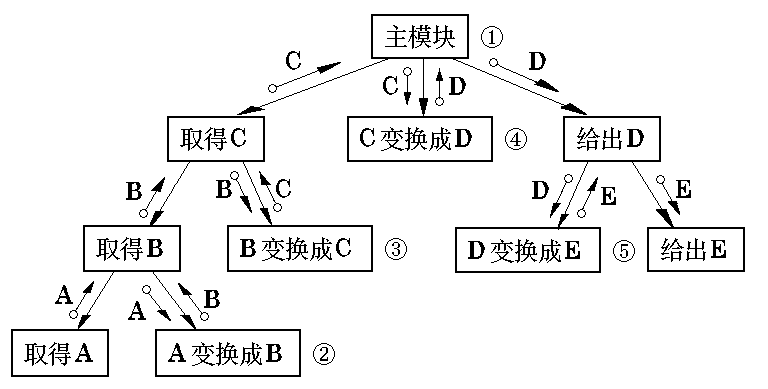
# 结构化软件设计

### 系统功能结构图

* 1. **功能结构图的基本结构**
     1. 传入模块：从下属模块获得数据，处理之后传送给上级模块
     2. 传出模块：从上属模块获得数据，处理之后传送给下级模块
     3. 变换模块：加工模块，从上级模块取得数据，处理之后传送给上级模块
     4. 协调模块：协调管理下属模块
  2. 分类
     1. **变换处理型**
        1. 更常见
        2. 处理问题的过程
           1. 取得、**变换**、给出数据



* + - 1. 例图：



* + 1. **事务处理型**
       1. 事务和事务处理中心：一个数据流处理节点，引发一个或者多个相同的处理，将处理结果返回给该节点，该结点称为事务处理中心：
       2. **事务是最小的工作单位**
          1. **原子性：**事务处理不可继续分割
          2. **一致性：**数据一致性
          3. **隔离性：**事务执行独享资源
          4. **持久性：**事务处理的结果被持久化保存
       3. 例图：



* 1. 分解：
     1. 原子模块：结构图中不能再分解的底层模块
     2. **完全因子分解系统**：
        1. 系统加工处理由原子模块完成，非原子模块进行控制和协调
        2. 最理想

### 变换映射

* 1. **映射**：将数据流图映射为程序结构图，称为映射
  2. 变换映射是一组设计步骤
  3. 步骤
     1. 复审并评估分析模型；【需求要全部被分析模型涵盖】
     2. **复审并重画数据流图；**【优化数据流图】
        1. 自顶向下
        2. **箭头只表示数据流，而非控制流**
        3. **省略异常处理和开始结束的考虑**，只考虑主要的加工处理逻辑
        4. 数据流进入和离开一个加工的时候，仔细标记，不要重名
        5. 可使用逻辑运算符
     3. 确定数据流图中的变换和事务特征；
     4. 区分有效（逻辑）输入、有效（逻辑）输出和中心变换部分，即标明数据流的边界；
        1. **中心变换部分：多股数据流汇集**
        2. **逻辑输入：**
           1. 从物理输入不断进行处理和移动，直到数据流不再被看做系统的输入，则前一个数据流就是系统的逻辑输入
           2. 依然是系统输入，但是离物理输入最远
        3. **逻辑输出**
           1. 从物理输出端开始向系统之间寻找，找到离物理输入最远但是依然看作系统输出的数据流
     5. 进行一级“因子化”分解，设计顶层和第一层模块；
        1. 设计顶层和第一层模块
        2. **顶层模块：【主模块】**
           1. 控制和协调的作用
           2. 程序名字来命名，和中心变换相对应
           3. 调用下层模块
        3. 第一层【中层】
           1. 控制和变换工作
           2. 每个逻辑输入设计输入模块，逻辑输出设计输出模块
           3. 为中心变换设计一个变换模块，将逻辑输入转换为逻辑输出
           4. 第一层和主模块之间传送的数据与数据流图相对应
     6. 进行二级“因子化”分解，设计中、下层模块；
        1. 设计中下层模块
        2. **每个变换型加工**--映射为程序中的输入模块、输出模块、变换模块
        3. 从变换中心的边界开始，将变换依次映射，直到外部实体
     7. 利用一些启发式原则来改进系统的初始结构图，直到得到符合要求的结构图为止。

### 事务映射

* 1. 步骤
     1. 复审系统分析模型
     2. 重画数据流图
     3. **确定是否具有事务流特征**
        1. 一个加工，将一个输入数据发散为多条数据流
     4. 确定事务中心及流特征：
        1. 事务中心通常位于几条操作路径的起始点上
        2. 事务源：带有**请求性质**的数据流
        3. 事务中心：处理事务源的加工，后续加工并列在事务中心的控制下
     5. 进行事务映射
        1. 映射到一个程序结构上，包含一个输入分支和一个分类事务处理分支
        2. 分支事务处理分支包含一个调度模块，用来调度和控制下属的事务处理模块
        3. 程序结构图分层：
           1. P层（主模块）==>
           2. T层（事务层，只能是取得事物、处理事务、给出结果三个模块）==>
           3. A层（操作模块）==>
           4. D层（操作模块之下的细分，细节模块）
           5. 事务模块可以调用多个操作模块、操作模块可以调用多个细节模块
     6. 因子化分解和细化
     7. 优化系统结构
  2. **变换-事务混合型结构**
     1. **变换型为主，事务型为辅**

### 优化系统功能结构图

* 1. 优化原则
     1. 模块**功能的完善化**
        1. 功能实现部分
        2. 出错处理部分——如何返回错误处理的信息和标志
        3. 完成数据加工之后返回给调用者本模块执行是否正确结束的标志
     2. **消除重复功能，**改善软件结构
        1. 模块的相似：
           1. 完全相似：结构上完全相似，但是数据类型上不一致
           2. 局部相似：考虑功能之间的耦合和内聚
     3. **模块的作用范围应在控制范围之内**
        1. 模块控制范围：本身和所有的从属模块
        2. 模块作用范围：指模块内一个判定的作用范围，受其影响的魔窟啊都属于该判定的作用范围
        3. 判定的范围包含在判定所在模块的控制范围——结构简单
        4. 所有收到一个判定影响的模块从属于该判定的所在模块
     4. **尽可能减少高扇出结构** 
        1. 上层扇出比较高，中层扇出较少，底层扇入到有高扇入的公用模块中。
        2. 模块的扇出指模块调用子模块的个数——2~5
        3. 扇入过大的非公共模块，可进一步分析进行功能分解
     5. **避免或减少使用病态联接**
     6. **模块的大小要适中**
     7. **设计功能可预测的模块，避免过分受限制的模块**
     8. **软件包应满足设计约束和可移植性**

### 设计后处理

### 详细设计