上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

软件工程原理与实践
SOFTWARE ENGINEERING

结构化分析与设计

沈备军

饮水思源 · 爱国荣校

结构化方法

- 一种面向数据流的传统软件开发方法
- 以数据流为中心构建软件的分析模型、设计模型和实现模型
- 分为三个步骤：
  - 结构化分析(Structured Analysis，简称SA)
  - 结构化设计(Structured Design，简称SD)
  - 结构化编程(Structured Programming，简称SP)

结构化方法的思想

- 发展历史
  - 提出：20世纪60年代末到70年代初
  - 成熟：20世纪70年代末到80年代中期
- 主要思想：**抽象**与自顶向下的逐层**分解**
  (控制复杂性的两个基本手段)
  - 抽象：从作为整体的软件系统开始(第一层)，每一抽象层次上只关注于系统的输入输出
  - 分解：将系统不断分解为子系统、模块……

大 纲

01-结构化分析

结构化分析方法概述
数据流图
分层数据流图的审查
数据字典
描述基本加工的小说明
实体—关系图

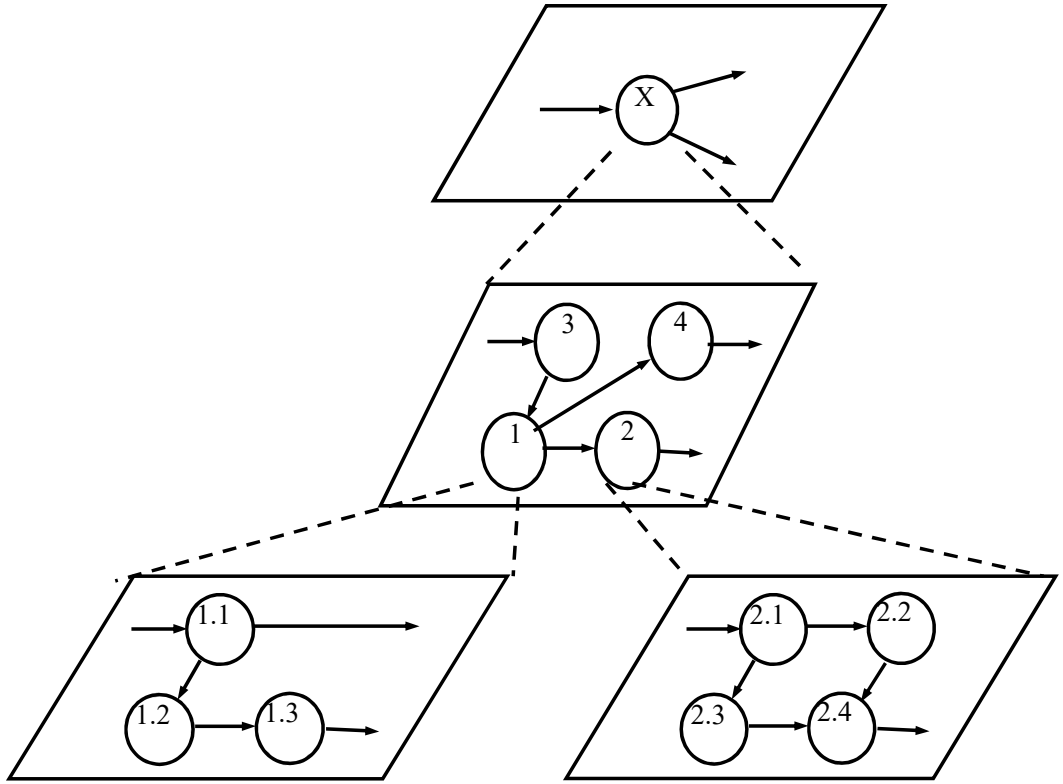
02-结构化设计

结构化设计概述
结构设计—概要设计
过程设计—详细设计

@第3.2章.教材

结构化分析方法中的抽象与分解

- 随着分解层次的增加，抽象的级别越来越低，也越接近问题的解(算法和数据结构)



结构化分析模型

- 数据字典：是模型的核心，它包含了软件使用和产生所有数据的描述
- 数据流图：用于功能建模，描述系统的输入数据流如何经过一系列的加工变换逐步变换成系统的输出数据流
- 实体—关系图：用于数据建模，描述数据字典中数据之间的关系
- 状态转换图：用于行为建模，描述系统接收哪些外部事件，以及在外部事件的作用下的状态迁移情况

大 纲

01-结构化分析

结构化分析方法概述

数据流图

分层数据流图的审查

数据字典

描述基本加工的小说明

实体—关系图

02-结构化设计

结构化设计概述

结构设计—概要设计

过程设计—详细设计

数据流图

- Data Flow Diagram(简称DFD)：描述输入数据流到输出数据流的变换(即加工)过程，用于对系统的功能建模，基本元素包括：

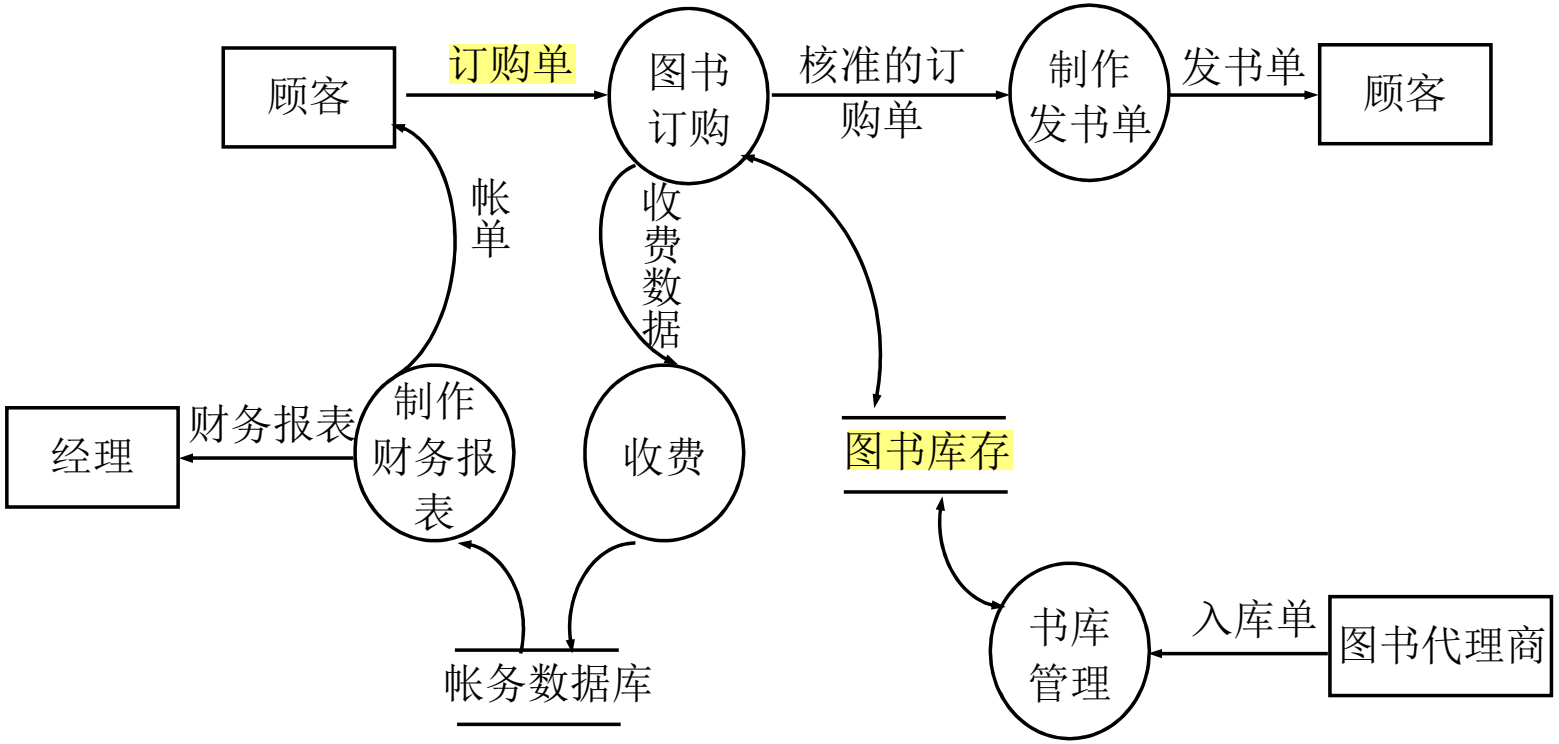
数据流(data flow)：由一组固定成分的数据组成，代表数据的流动方向

加工(process)：描述了输入数据流到输出数据流的变换，即将输入数据流加工成输出数据流

文件(file)：使用文件、数据库等保存某些数据结果供以后使用

源或宿(source or sink)：由一组固定成分的数据组成，代表数据的流动方向

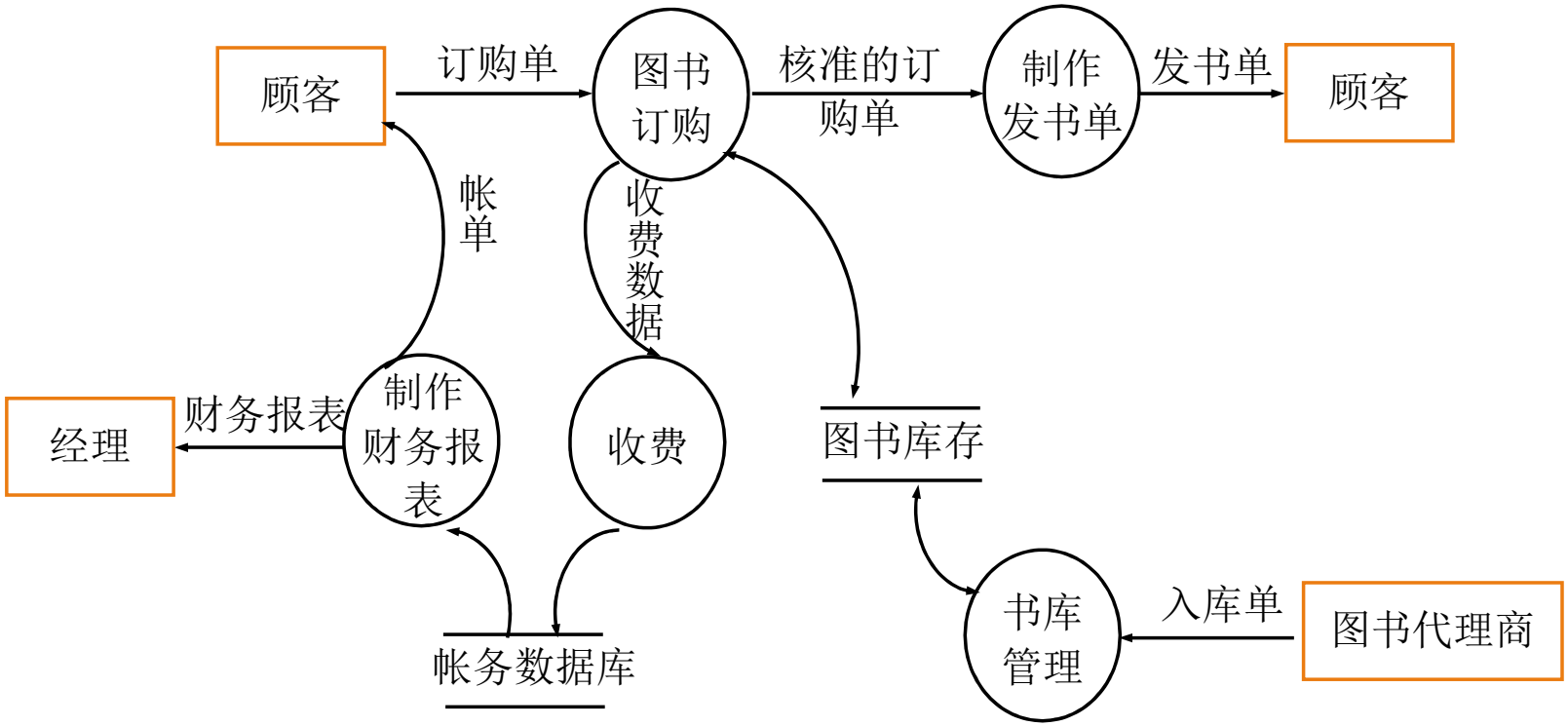
示例：图书订购系统DFD



源或宿 

- 存在于软件系统之外的人员或组织，表示软件系统输入数据的来源和输出数据的去向，因此也称为源点和终点
  - 例如，对一个考务处理系统而言
    - 考生向系统提供报名单(输入数据流)，所以考生是考试系统(软件)的一个源
    - 考务处理系统要将考试成绩的统计分析表(输出数据流)传递给考试中心，所以考试中心是该系统的一个宿
- 源或宿用相同的图形符号表示
  - 当数据流从该符号流出时表示是源
  - 当数据流流向该符号时表示是宿
  - 当两者皆有时表示既是源又是宿

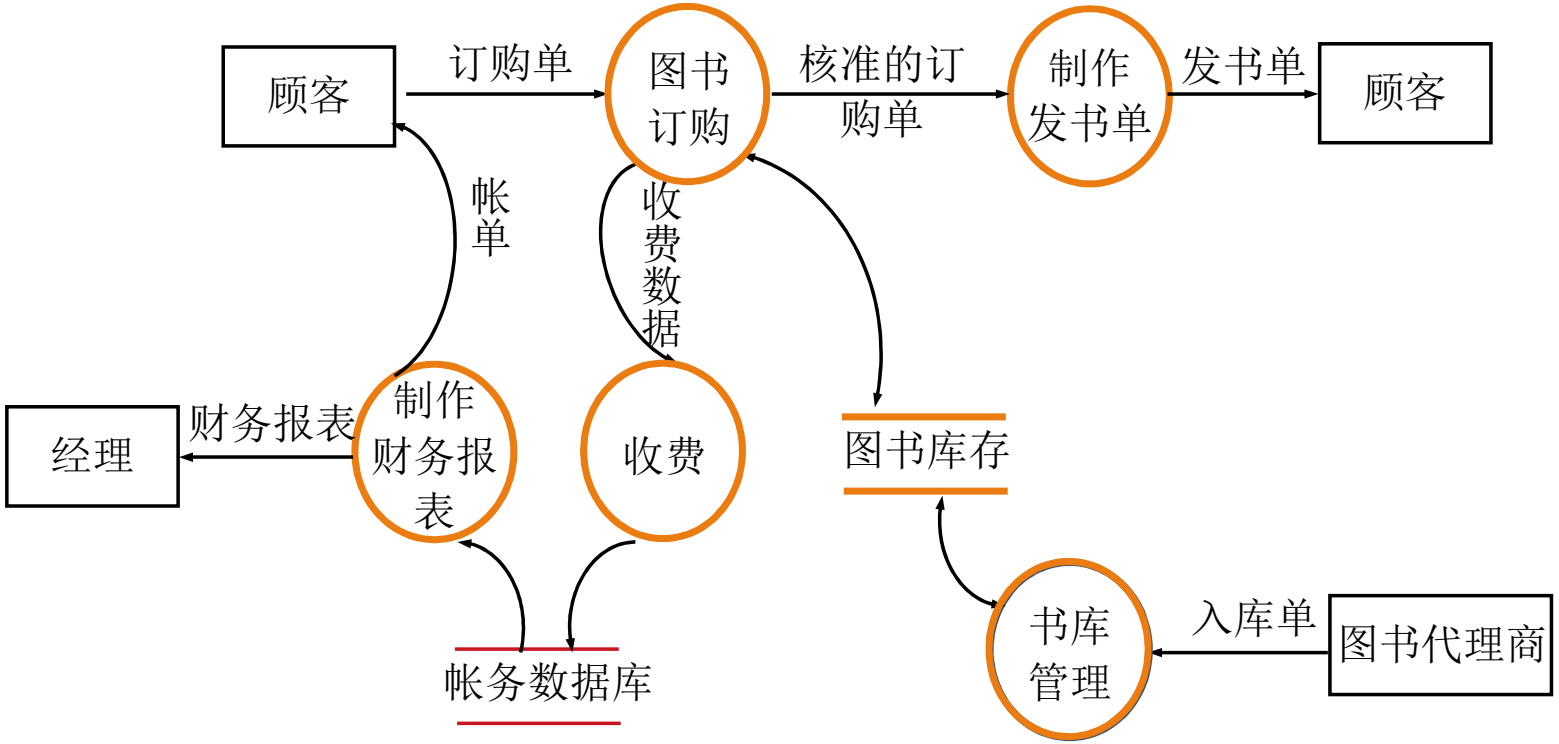
示例：图书订购系统DFD



加工和文件 ○ =

- 加工：描述输入数据流到输出数据流的变换
  - 每个加工用一个定义明确的名字标识
  - 至少有一个输入数据流和一个输出流
  - 可以有多个输入数据流和多个输出数据流
- 文件：保存数据信息的外部单元
  - 每个文件用一个定义明确的名字标识
  - 由加工进行读写
  - DFD中称为文件，但在具体实现时可以用文件系统实现也可以用数据库系统等实现

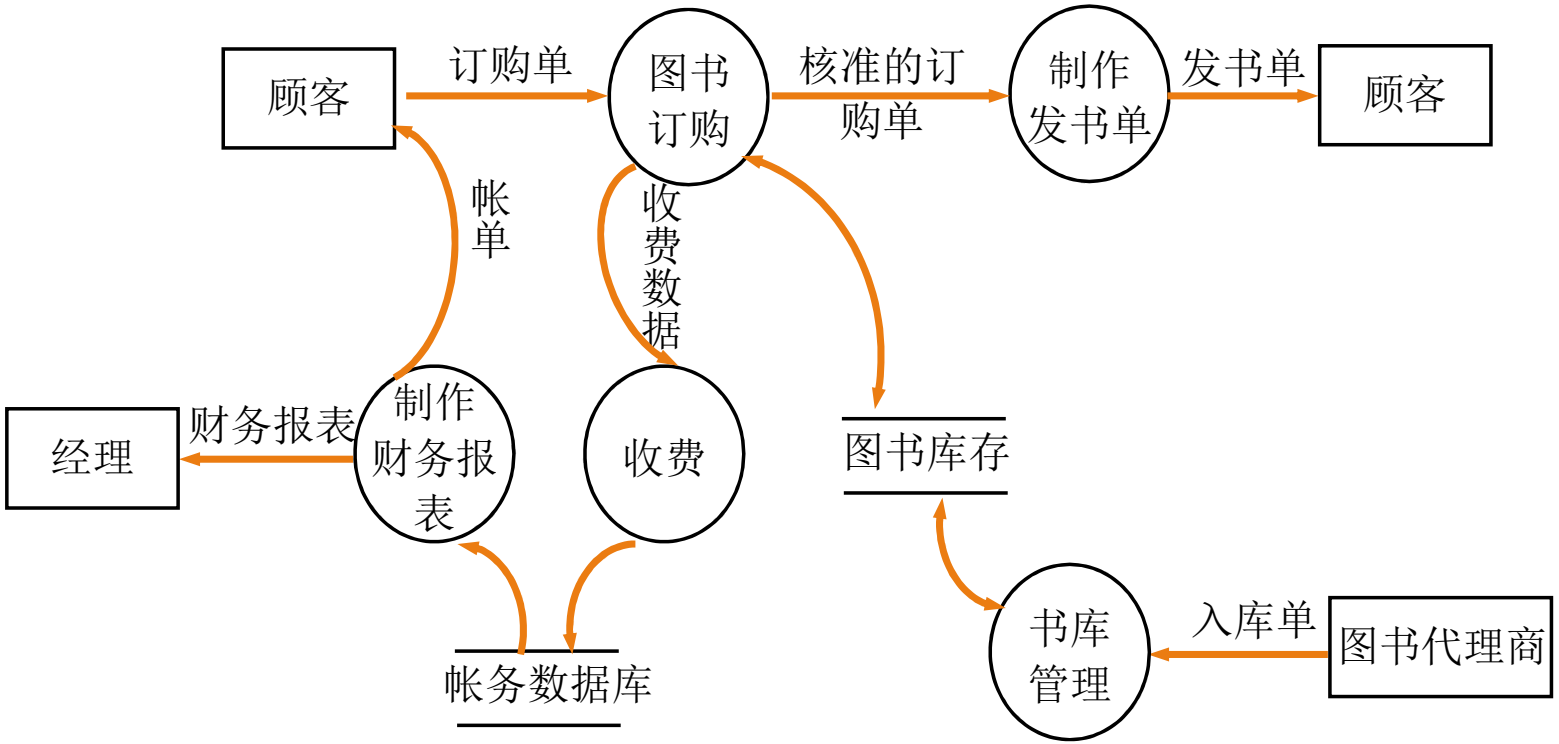
示例：图书订购系统DFD



数据流 →

- 每个数据流用由一组固定成分的数据组成并拥有一个定义明确的名字标识
  - 如：运动会管理系统中，报名单(数据流)由队名、姓名、性别、参赛项目等数据组成
- 数据流的流向
  - 从一个加工流向另一个加工
  - 从加工流向文件(写文件)
  - 从文件流向加工(读文件)
  - 从源流向加工
  - 从加工流向宿

示例：图书订购系统DFD



数据流图的扩充符号

- 描述一个加工的多个数据流之间的关系
  - 星号(*)：表示数据流之间存在“与”关系
    - 所有输入数据流同时存在时，才能进行加工处理
    - 或加工处理的结果是同时产生所有输出数据流
  - 加号(+)：表示数据流之间存在“或”关系
    - 至少存在一个输入数据流时才能进行加工处理
    - 或加工处理的结果是至少产生一个输出数据流
  - 异或(⊕)：表示数据流之间存在“异或”(互斥)关系
    - 必须存在且仅存在一个输入数据流时，才能进行加工处理
    - 或加工处理的结果是产生且仅产生一个输出数据流

对数据流图进行分层

- George Miller在著名的论文“神奇的数字7加减2：我们处理信息的能力的某种限制”中指出：人们在一段时间内的短期记忆似乎限制在5～9件事情之内
- 根据自顶向下逐层分解的思想将数据流图画成层次结构
- 每个层次画在独立的数据流图中，加工个数可大致控制在“7±2”的范围中

数据流图的各个层次

- 顶层图
  - 只有代表整个软件系统的1个加工，描述了软件系统与外界(源或宿)之间的数据流
  - 加工不必编号
- 0层图
  - 顶层图中的加工经分解后的图，只有1张
  - 加工编号分别为1，2，3，…
- X层图
  - 图的编号：若父图中的加工号x分解成某一子图，则该子图号记为“图x”
  - 加工的编号：若父图中的加工号为x的加工分解成某一子图，则该子图中的加工编号分别为x.1、x.2、x.3…

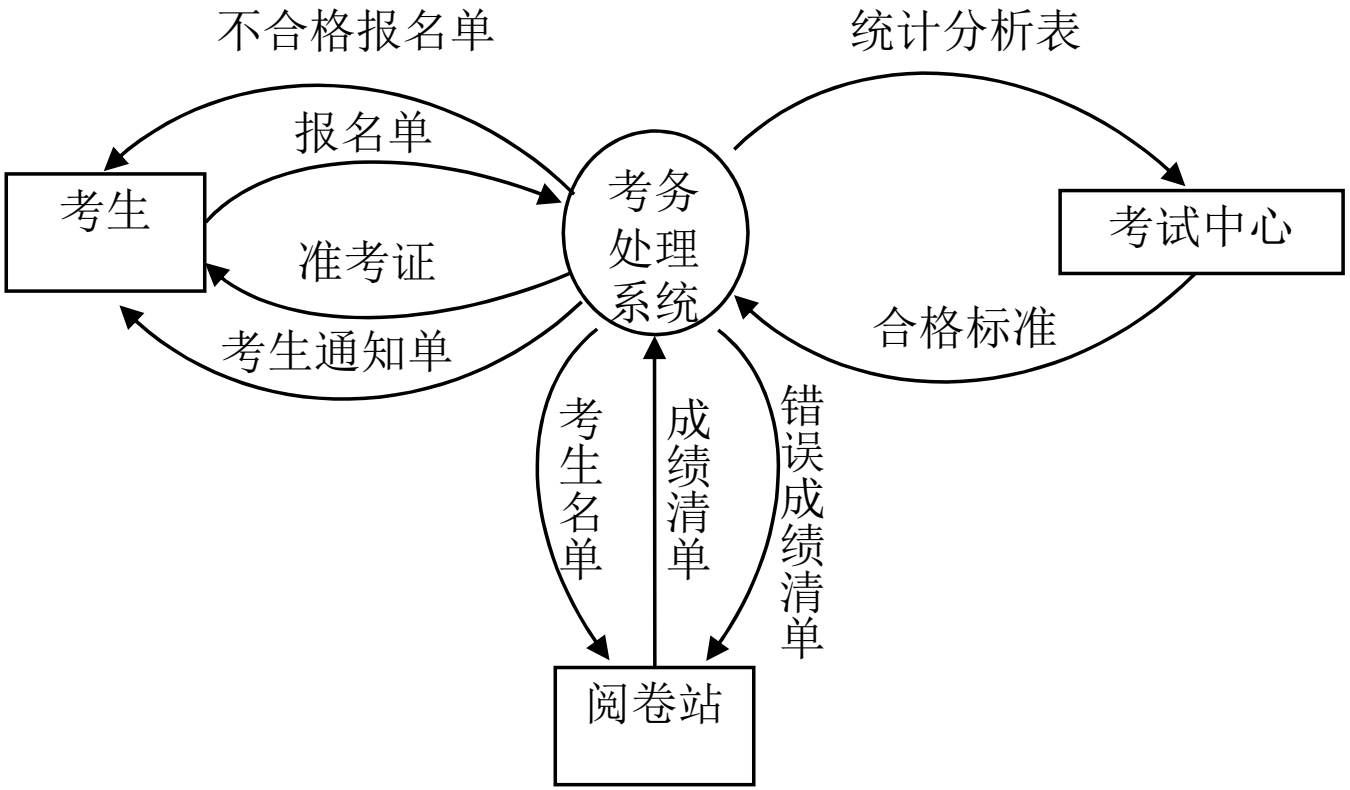
分层数据流图举例

- 简化的资格和水平考试的考务处理系统
  - 分成多个级别，如初级程序员、程序员、高级程序员、系统分析员等，凡满足一定条件的考生都可参加某一级别的考试
  - 考试的合格标准将根据每年的考试成绩由考试中心确定
  - 考试的阅卷由阅卷站进行，因此，阅卷工作不包含在软件系统中

示例：考务处理系统功能需求

- 对考生送来的报名单进行检查
- 对合格的报名单编好准考证号后将准考证送给考生，并将汇总后的考生名单送给阅卷站
- 对阅卷站送来的成绩清单进行检查，并根据考试中心制订的合格标准审定合格者
- 制作考生通知单送给考生
- 进行成绩分类统计(按地区、年龄、文化程度、职业、考试级别等分类)和试题难度分析，产生统计分析表

示例：考务处理系统顶层图

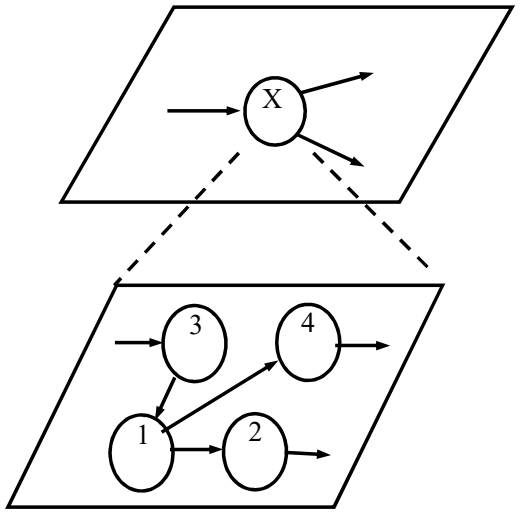


示例：部分数据流的组成

- 报名单 = 地区 + 序号 + 姓名 + 文化程度 + 职业 + 考试级别 + 通信地址
- 正式报名单 = 准考证号 + 报名单
- 准考证 = 地区 + 序号 + 姓名 + 准考证号 + 考试级别 + 考场
- 考生名单 = {准考证号 + 考试级别}
  其中｛w｝表示w重复多次
- 考生名册 = 正式报名单
- 统计分析表 = 分类统计表 + 难度分析表
- 考生通知单 = 准考证号 + 姓名 + 通信地址 + 考试级别 + 考试成绩 + 合格标志

DFD细化 - 确定加工

- 将父图中某加工分解而成的子加工
  - 根据功能分解来确定加工：将一个复杂的功能分解成若干个较小的功能，较多应用于高层DFD中的分解
  - 根据业务处理流程确定加工：分析父图中待分解加工的业务处理流程，业务流程中的每一步都可能是一个子加工
  - 特别要注意在业务流程中数据流发生变化或数据流的值发生变化的地方，应该存在一个加工



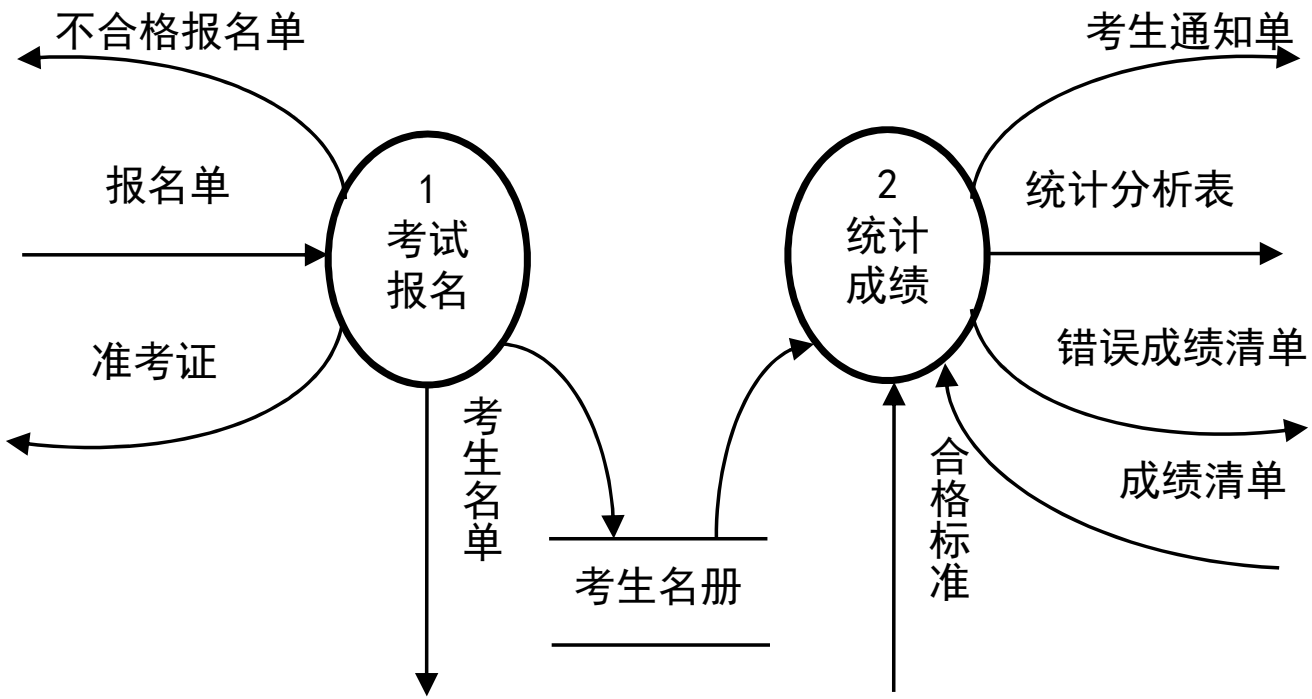
DFD细化 - 确定数据流

- 在父图中某加工分解而成的子图中，父图中相应加工的输入/输出数据流都是且仅是子图边界上的输入/输出数据流
- 分解后的子加工之间应增添相应的新数据流表示加工过程中的中间数据
- 如果某些中间数据需要保存以备后用，那么可以成为流向文件的数据流
- 同一个源或加工可以有多个数据流流向一个加工，如果它们不是一起到达和一起加工的，那么可以将它们分成若干个数据流

DFD细化 - 确定文件

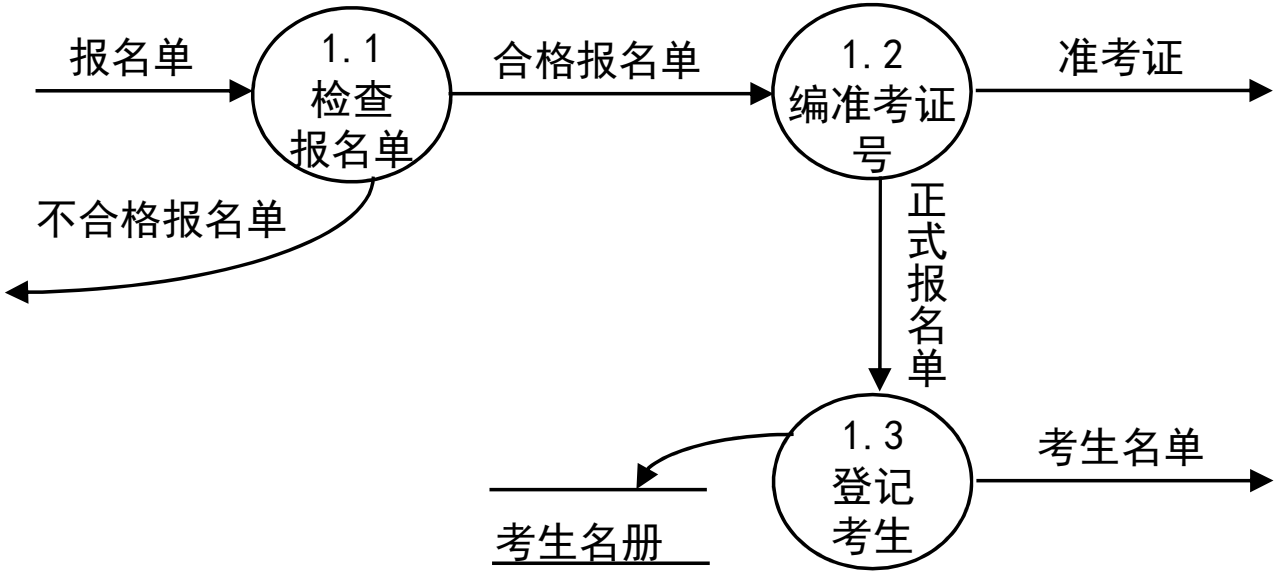
- 如果父图中该加工存在读写文件的数据流，则相应的文件和数据流都应画在子图中
- 在分解子图中，如果需要保存某些中间数据以备后用，则可以将这些数据组成一个新的文件
- 新文件(首次出现的文件)至少应有一个加工为其写入记录，同时至少存在另一个加工来读该文件的记录
- 注意：从父图中继承下来的文件在子图中可能只对其进行读，或只进行写

示例：考务处理系统0层图

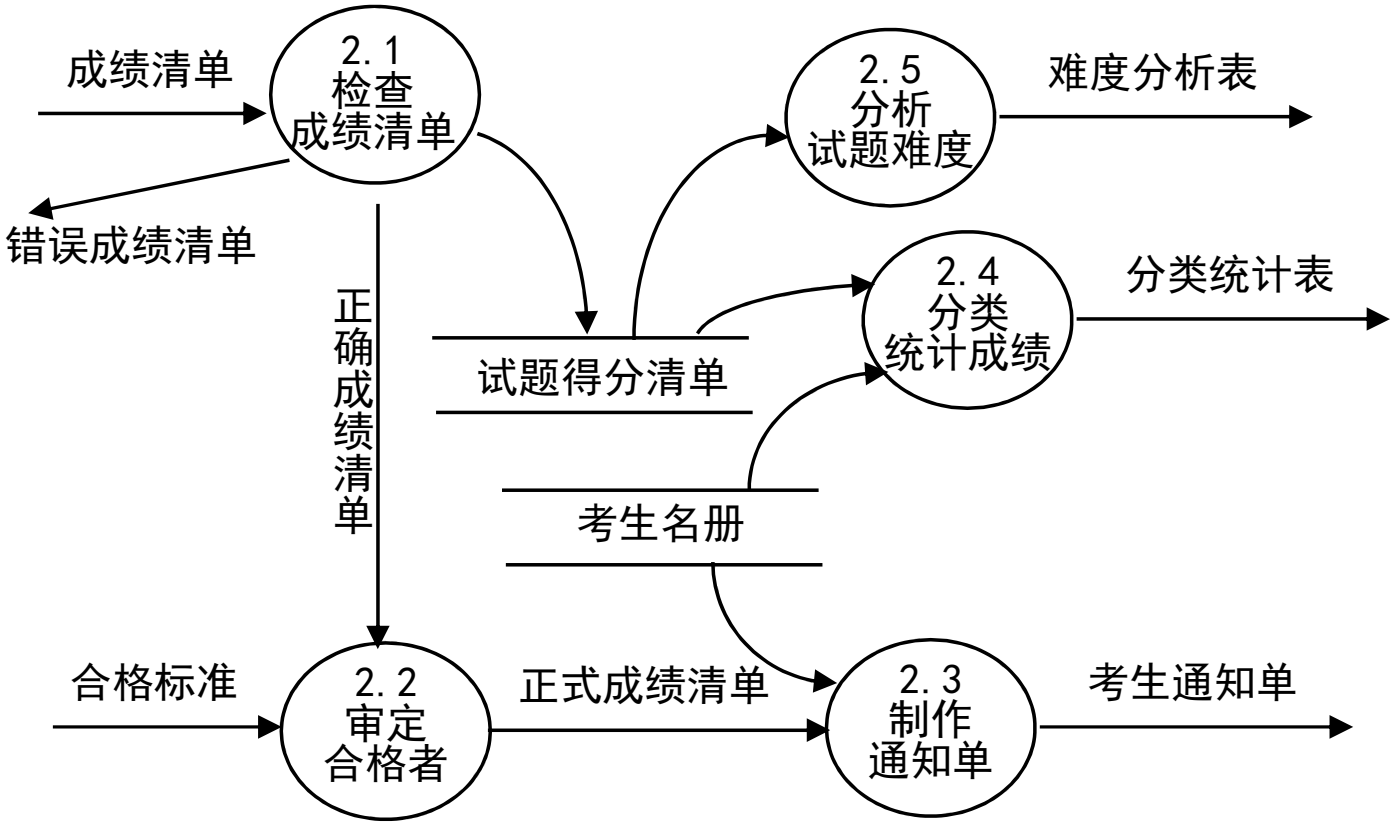


示例：考务处理系统 - 考试报名1子图

- 3个子加工：检查报名单、编准考证号、登记考生
- “合格报名单”和“正式报名单”是新增加的数据流，其它数据流都是加工1原有的
- 在加工1的分解中没有新的文件产生



示例：考务处理系统 - 统计成绩2子图



总结：画分层数据流图的步骤

- 1．画系统的输入和输出
- 2．画系统内部
- 3．画加工内部
- 4．重复第3步，直至每个尚未分解的加工都足够简单(即不必再分解)

DFD练习，画出顶层图和0层图

- A城市的公共工作部门决定开发一个基于Web的坑洼跟踪和修补系统(PHTRS)：
  - 当报告路面有坑洼时，它们被赋予一个标识号，并依据街道地址、大小(1到10)、位置(路中或路边等)、地区(由街道地址确定)和修补优先级(由坑洼的大小确定)储存起来。
  - 工作单数据和每个坑洼有关，数据包括坑洼位置和大小、修理组标识号、修理组的人数、分配的设备、修复时间长度、坑洼状态(正在处理中、已修复、临时修复、未修复)、使用填料的数量和修复的开销(由使用的时间、人数、使用的材料的设备计算得到)。
  - 最后，产生一个损失文件以保存该坑洼造成的损失信息，并包括居民的姓名、地址、电话号码、损失类型和损失钱数。
  - PHTRS是基于Web的系统，可交互地进行所有的查询。

大 纲

01-结构化分析

结构化分析方法概述
数据流图
分层数据流图的审查
数据字典
描述基本加工的小说明
实体—关系图

02-结构化设计

结构化设计概述
结构设计—概要设计
过程设计—详细设计

分层数据流图的审查

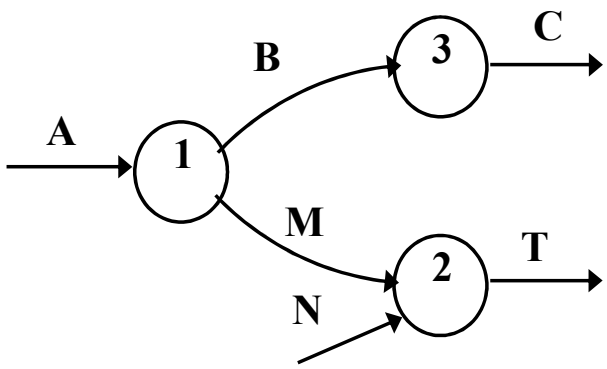
- 检查图中是否存在错误或不合理(不理想)的部分
  - 一致性：分层DFD中不存在矛盾和冲突
  - 完整性：分层DFD本身的完整性，即是否有遗漏的数据流、加工等元素

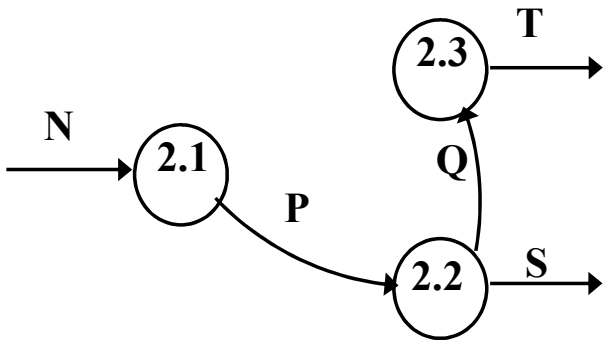
分层数据流图的一致性

- 父图与子图平衡
  - 任何一张DFD子图边界上的输入/输出数据流必须与其父图中对应的加工的输入/输出数据流保持一致
- 数据守恒
  - 一个加工所有输出数据流中的数据，必须能从该加工的输入数据流中直接获得，或者能通过该加工的处理而产生
  - 多余的数据流：加工未使用其输入数据流中的某些数据项

父图与子图不平衡的实例

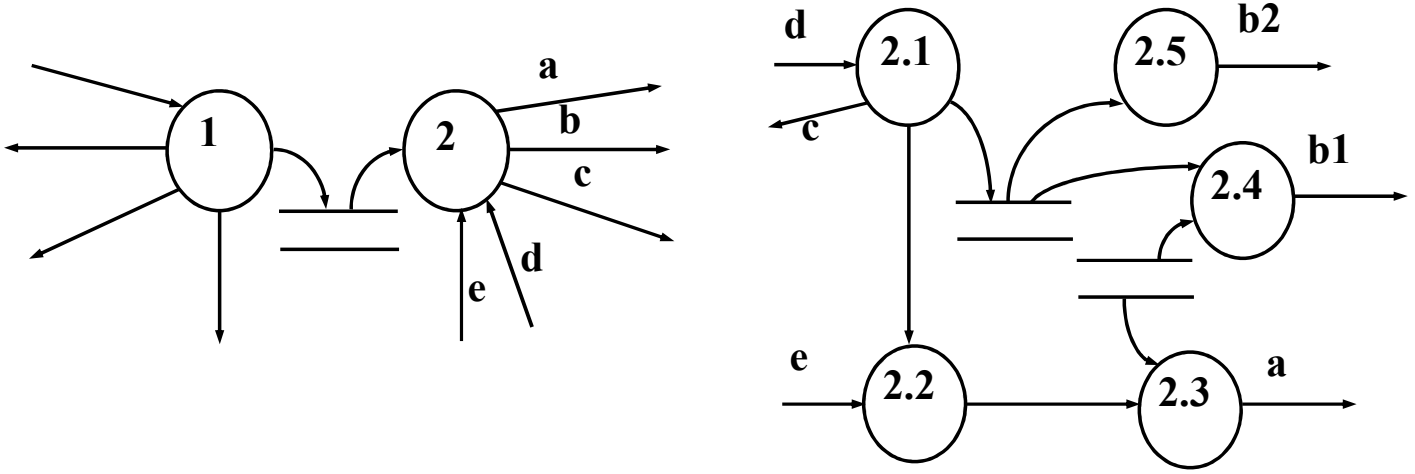
- 加工2的输入数据流有M和N，输出数据流是T
- 而子图(右图)边界上的输入数据流是N，输出数据流是S和T





父图与子图平衡的实例

- 注意：如果父图某加工的一个数据流，对应于子图中几个数据流，而子图中组成这些数据流的数据项全体正好等于父图中的这个数据流，那么它们仍算是平衡的

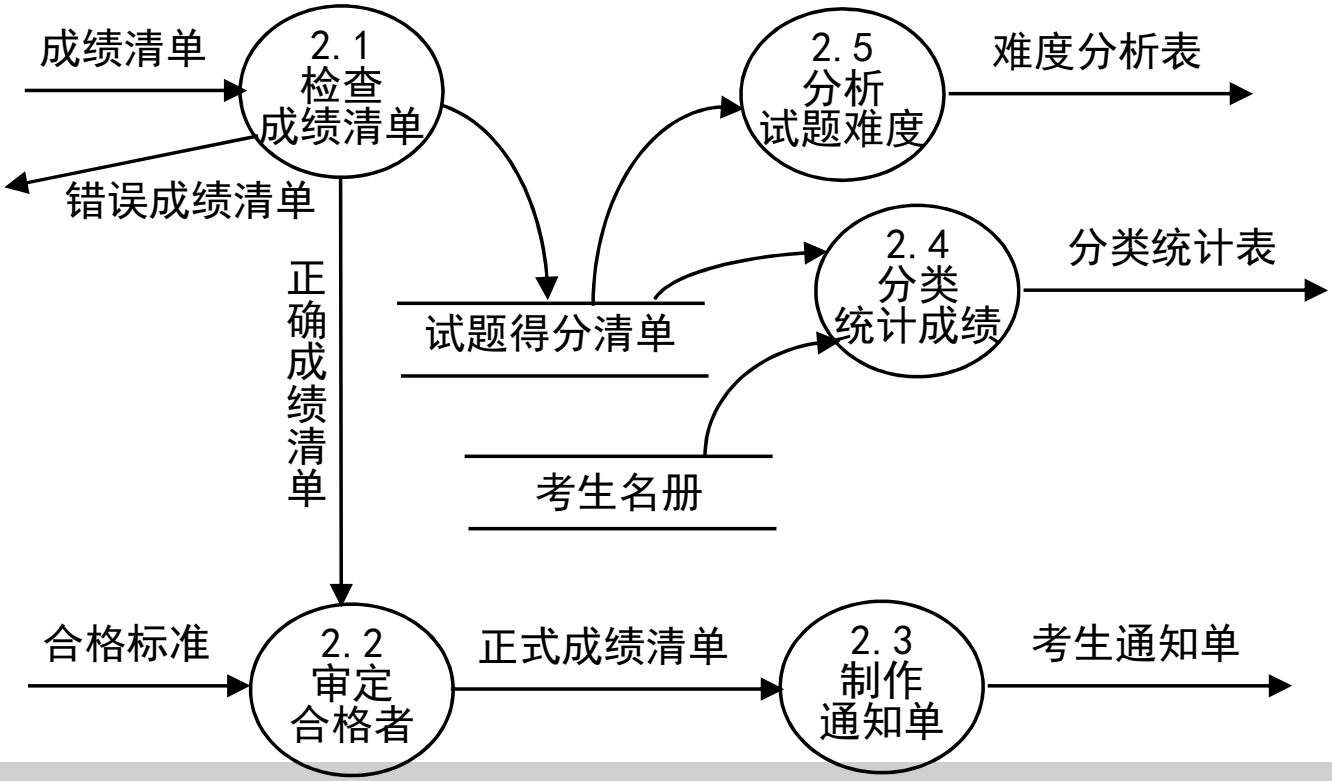


(a) 父图　　(b) 子图

a：考生通知单；b：统计分析表；b1：分类统计表；b2：难度分析表；
c：错误成绩清单；d成绩清单；e合格标准。

数据不守恒的实例

- 由于“正式成绩清单”中缺少“考生通知单”中的姓名、通信地址等数据，这些数据也无法由加工2.3自己产生，因此，加工2.3不满足数据守恒的条件



分层数据流图的完整性

- 每个加工至少有一个输入数据流和一个输出数据流
- 在整套分层数据流中，每个文件应至少有一个加工读该文件，有另一个加工写该文件
- 分层数据流图中的每个数据流和文件都必须命名(除了流入或流出文件的数据流)，并保持与数据字典的一致
- 分层DFD中的每个基本加工(即不再分解子图的加工)都应有一个加工规约

其它需注意的问题-1

- 适当命名：每个数据流、加工、文件、源和宿都应被适应地命名，名字应符合被命名对象的实际含义
  - 名字应反映整个对象(如数据流、加工)，而不是仅反映它的某一部分
  - 避免使用空洞的、含义不清的名字，如数据、信息、处理、统计等
  - 如果发现某个数据流或加工难以命名时，往往是DFD分解不当的征兆，此时应考虑重新分解
- 画数据流而不是画控制流
  - 判断准则：这条线上是否有数据流过

其它需注意的问题-2

- 分解尽可能均匀
  - 理想目标：任何两个加工的分解层数之差不超过1
  - 应尽可能使分解均匀，对于分解不均匀的情况应重新分解
- 先考虑稳定状态，忽略琐碎的枝节
  - 先考虑稳定状态下的各种问题，暂时不考虑系统如何启动、如何结束、出错处理以及性能等问题
- 随时准备重画
  - 对于一个复杂的软件系统，往往要经过反复多次的重画和修改才能构造出完整、合理、满足用户需求的分层DFD
  - 分析阶段遗漏下来的一个错误，到开发后期要化费几百倍代价来纠正这个错误

分解的程度

- 可参照以下几条与分解有关的原则：
  - 7±2
  - 分解应自然，概念上合理、清晰
  - 只要不影响DFD的易理解性，可适当多分解几个加工，以减少层数
  - 一般说来，上层分解得快些 (即多分解几个加工)，下层分解得慢些 (即少分解几个加工)
  - 分解要均匀

大 纲

01-结构化分析

结构化分析方法概述
数据流图
分层数据流图的审查
数据字典
描述基本加工的小说明
实体—关系图

02-结构化设计

结构化设计概述
结构设计—概要设计
过程设计—详细设计

数据字典

- 数据流图与数据字典是密不可分的，两者结合起来构成软件的分析模型
- 数据字典由字典条目组成，每个条目描述DFD中的一个元素
- 数据字典条目包括：
  - 数据流条目
  - 文件条目
  - 加工条目
  - 源/宿条目

数据字典的描述符号

符 号	名 称	举 例
=	定义为	x = … 表示x由…组成
+	与	a + b 表示a和b
[…，…]	或	[a，b] 表示a或b
[…|…]	或	[a|b] 表示a或b
{…}	重复	{a} 表示a重复0或多次
$\{\ldots\}_m^n$	重复	$\{a\}_3^8$ 表示a重复3到8次
(…)	可选	(a) 表示a重复0或1次
"…"	基本数据元素	"a" 表a是基本数据

数据流条目的描述内容

- 名称：数据流名(可以是中文名或英文名)
- 别名：名称的另一个名字
- 简述：对数据流的简单说明
- **数据流组成**：描述数据流由哪些数据项组成
- 数据流来源：描述数据流从哪个加工或源流出
- 数据流去向：描述数据流流入哪个加工或宿
- 数据量：系统中该数据流的总量
  - 如考务处理系统中“报名单”的总量是100000张
  - 或者单位时间处理的数据流数量，如80000张/天
- 峰值：某时段处理的最大数量
  - 如每天上午9：00至11：00处理60000张表单
- 注解：对该数据流的其它补充说明

3
1

数据流组成

- 数据流组成是数据流条目的核心，它列出组成该数据流的各数据项，例如：
  - 培训报名单 = 姓名 + 单位 + 课程
  - 运动员报名单 = 队名 + 姓名 + 性别 + {参赛项目}$^{3}_{1}$
- 当一个数据流的组成比较复杂时，可以将其分解成几个数据流，例如：
  - 课程 = 课程名 + 任课教师 + 教材 + 时间地点
  - 时间地点 = {星期几 + 第几节 + 教室}

文件条目的描述内容

- 名称：文件名
- 别名：同数据流条目
- 简述：对文件的简单说明
- **文件组成**：描述文件的记录由哪些数据项组成(与数据流条目中的文件组成描述方法相同)
- 写文件的加工：描述哪些加工写文件
- 读文件的加工：描述哪些加工读文件
- 文件组织：描述文件的存储方式(顺序、索引)，排序的关键字
- 使用权限：描述各类用户对文件读、写、修改的使用权限
- 数据量：文件的最大记录个数
- 存取频率：描述对该文件的读写频率
- 注解：对该文件的其它补充说明

加工条目的描述内容

- 名称：加工名
- 别名：同数据流条目
- 加工号: 加工在DFD中的编号
- 简述：对加工的功能的简要说明
- 输入数据流：描述加工的输入数据流，包括读哪些文件名
- 输出数据流：描述加工的输出数据流，包括写哪些文件名
- **加工逻辑**：简要描述加工逻辑，或者对加工规约的索引
  - 基本加工的加工逻辑用小说明描述，在加工条目中可填写对加工规约的索引
  - 非基本加工分解而成的DFD子图已反映了它的加工逻辑，不必书写小说明
- 异常处理：描述加工处理过程中可能出现的异常情况，及其处理方式
- 加工激发条件：描述执行加工的条件，如“身份认证正确”，“收到报名单”
- 执行频率：描述加工的执行频率，如，每月执行一次，每天0点执行
- 注解：对加工的其它补充说明

源或宿条目的描述内容

- 名称：源或宿的名(外部实体名)
- 别名：同数据流条目
- 简要描述：对源或宿的简要描述(包括指明该外部实体在DFD中是用作“源”，还是“宿”，还是“既是源又是宿”)
- 输入数据流：描述源向系统提供哪些输入数据流
- 输出数据流：描述系统向宿提供哪些输出数据流
- 注解：对源或宿的其它补充说明

DD练习

- A城市的公共工作部门决定开发一个基于Web的坑洼跟踪和修补系统(PHTRS):
  - 当报告路面有坑洼时，它们被赋予一个标识号，并依据街道地址、大小(1到10)、位置(路中或路边等)、地区(由街道地址确定)和修补优先级(由坑洼的大小确定)储存起来。
  - 工作单数据和每个坑洼有关，数据包括坑洼位置和大小、修理组标识号、修理组的人数、分配的设备、修复时间长度、坑洼状态(正在处理中、已修复、临时修复、未修复)、使用填料的数量和修复的开销(由使用的时间、人数、使用的材料的设备计算得到)。
  - 最后，产生一个损失文件以保存该坑洼造成的损失信息，并包括居民的姓名、地址、电话号码、损失类型和损失钱数。
  - PHTRS是基于Web的系统，可交互地进行所有的查询。

大 纲

01-结构化分析

结构化分析方法概述
数据流图
分层数据流图的审查
数据字典
描述基本加工的小说明
实体—关系图

02-结构化设计

结构化设计概述
结构设计—概要设计
过程设计—详细设计

基本加工的小说明

- 小说明是基本加工的规约说明，应精确地描述用户要求一个加工“做什么”
- 包括加工的激发条件、加工逻辑、优先级、执行频率、出错处理等
- 最基本的部分是加工逻辑，即该加工的输出数据流与输入数据流之间的逻辑关系
- 加工逻辑不是对加工的设计，不涉及数据结构、算法实现、编程语言等与设计和实现有关的细节

加工逻辑的描述方法

- 结构化语言：介于自然语言和形式语言之间的一种半形式语言
- 判定表：适用于加工逻辑包含多个条件，而不同的条件组合需做不同的动作
- 判定树：判定表的变种，它本质上与判定表是相同的，只是表示形式不同

结构化语言

- 没有严格的语法
- 加工规约分为若干个段落，每个段落可分为内外两层：
  - 外层有严格的语法来描述它的控制结构
    - 如结构化英语中可使用if_then_else、while_do、repeat_until、for_do、case等结构
  - 内层可以用自然语言来描述
- 允许使用嵌套结构

“计算信用度”的结构化英语描述

```
Case 1 (No Bounced—Checks in Customer Record):
        Write Exemplary—Customer—Citation to Annual—Summary.
Case 2 (One Bounced—check):
        If Yearly—Average—Balance exceeds $ 1000.
            Remove Bounced—Check from Customer—Record.
        Otherwise.
        Recuce Credit—Limit by 10%.
Case 3 (Multiple Bounced—Checks):
        For each Bounced—Check.
            Recuce Credit—Limit by 15%.
        Set Credit—Rating to Deadbeat.
        Write Scathing—Comment to Annual—Summary.
        Write Customer—Name—and—Address to IRS—Enemies—List.
```

结构化语言书写加工规约注意事项

- 语句力求精炼
- 语句必须易读、易理解、无二义
- 主要使用祈使句，祈使句中的动词要明确表达要执行的动作
- 所有名字必须是数据字典中有定义的名字
- 不使用形容词、副词等修饰语
- 不使用含义相同的动词，如“修改”、“修正”等
- 可以使用常用的算术和关系运算符
- 总之要尽可能**精确、无二义、简明扼要、易理解**

判定表

- 判定表的组成元素
  - 条件桩(Condition Stub)：列出各种条件的对象，如发货单金额，赊欠天数等，每行写一个条件对象
  - 条件条目(Condition entry)：列出各条件对象的取值，条件条目的每一列表示了一个可能的条件组合
  - 动作桩(action stub)：列出所有可能采取的动作，如发出发货单等，每行写一个动作
  - 动作条目(action entry)：列出各种条件组合下应采取的动作

“审批发货单”加工的判定表

发货单金额	＞500	＞500	≤500	≤500
赊欠天数	＞60	≤60	＞60	≤60
发不批准通知	√			
发出批准书		√	√	√
发出发货单		√	√	√
发出赊欠报告			√	

判定表的其它形式

发货单金额	＞500	≤500	—
赊欠天数	＞60	＞60	≤60
发不批准通知	√		
发出批准书		√	√
发出发货单		√	√
发出赊欠报告		√	

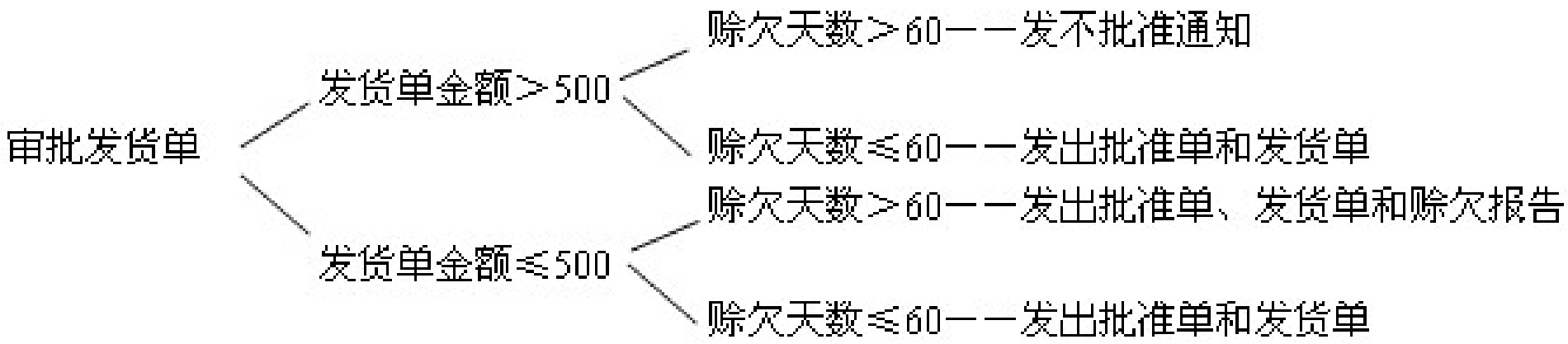
“审批发货单”加工的简化判定表

发货单金额＞500	1	1	0	0
赊欠天数＞60	1	0	1	0
发不批准通知	√			
发出批准单		√	√	√
发出发货单		√	√	√
发出赊欠报告			√	

“审批发货单”加工的另一种判定表

判定树

- 本质上与判定表是相同的，只是表示形式不同
- 例如“审批发货单”加工逻辑的判定树描述入下：



大 纲

01-结构化分析

结构化分析方法概述
数据流图
分层数据流图的审查
数据字典
描述基本加工的小说明
实体—关系图

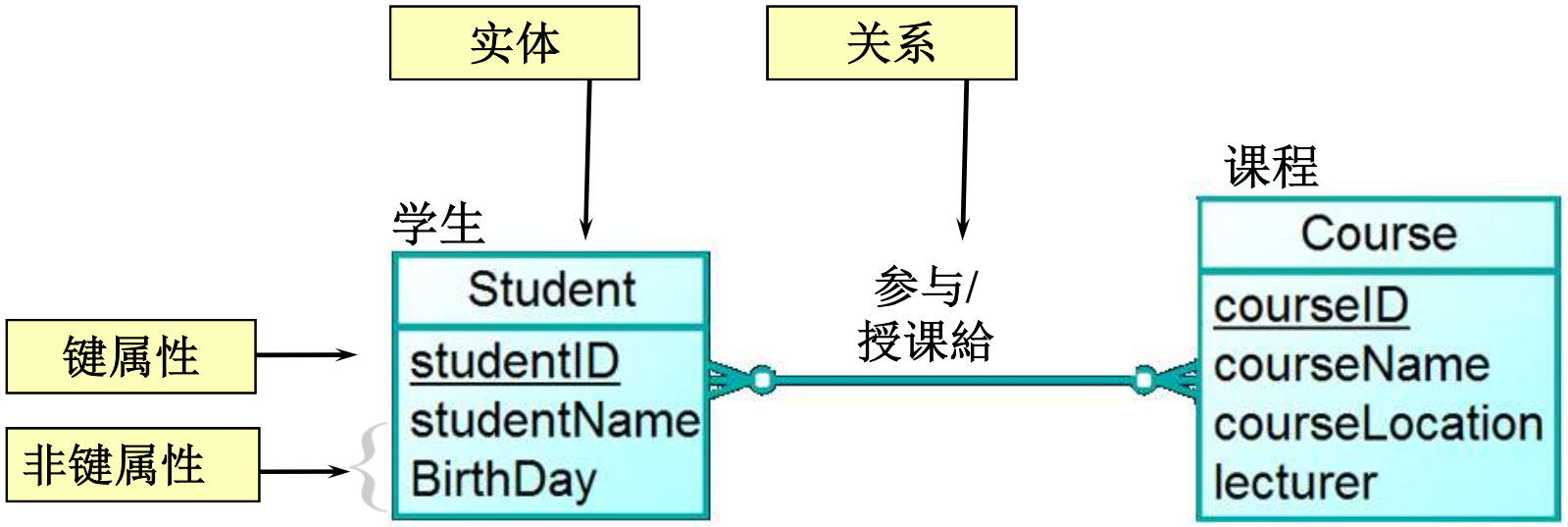
02-结构化设计

结构化设计概述
结构设计—概要设计
过程设计—详细设计

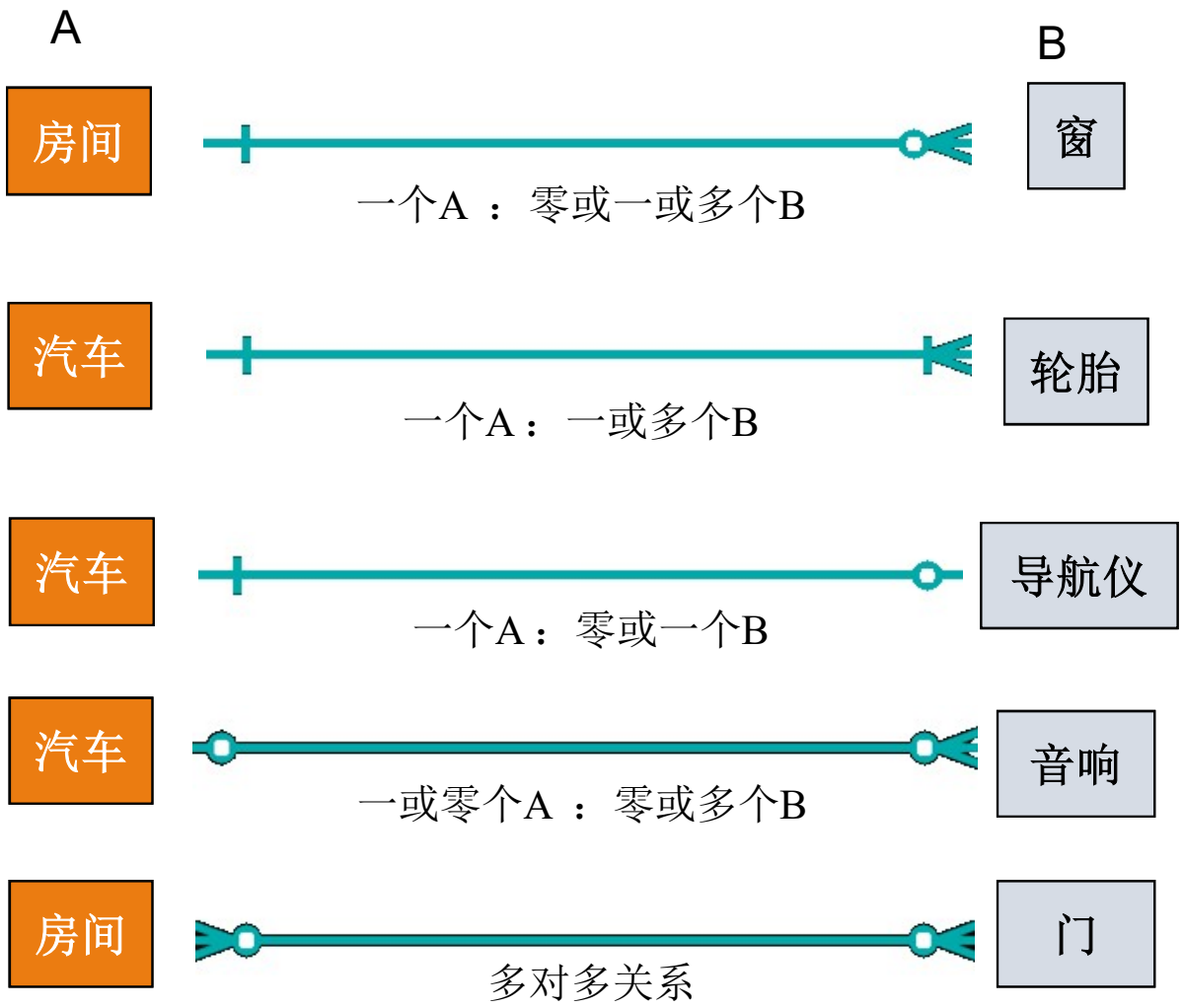
实体—关系图

- 实体—关系图（Entity-Relation Diagram, ERD）用于数据建模，描述数据字典中数据之间的关系
- **实体**是客观存在的数据对象
  - 只封装数据，不封装数据的操作，和OO类不同
  - 例如: 学生, 学校, 事件, 植物
- 实体具有**属性**
  - 例如: 学生具有姓名和地址
- 实体间可存在多种不同**关系**
  - 例如：教师指导学生，学生选课

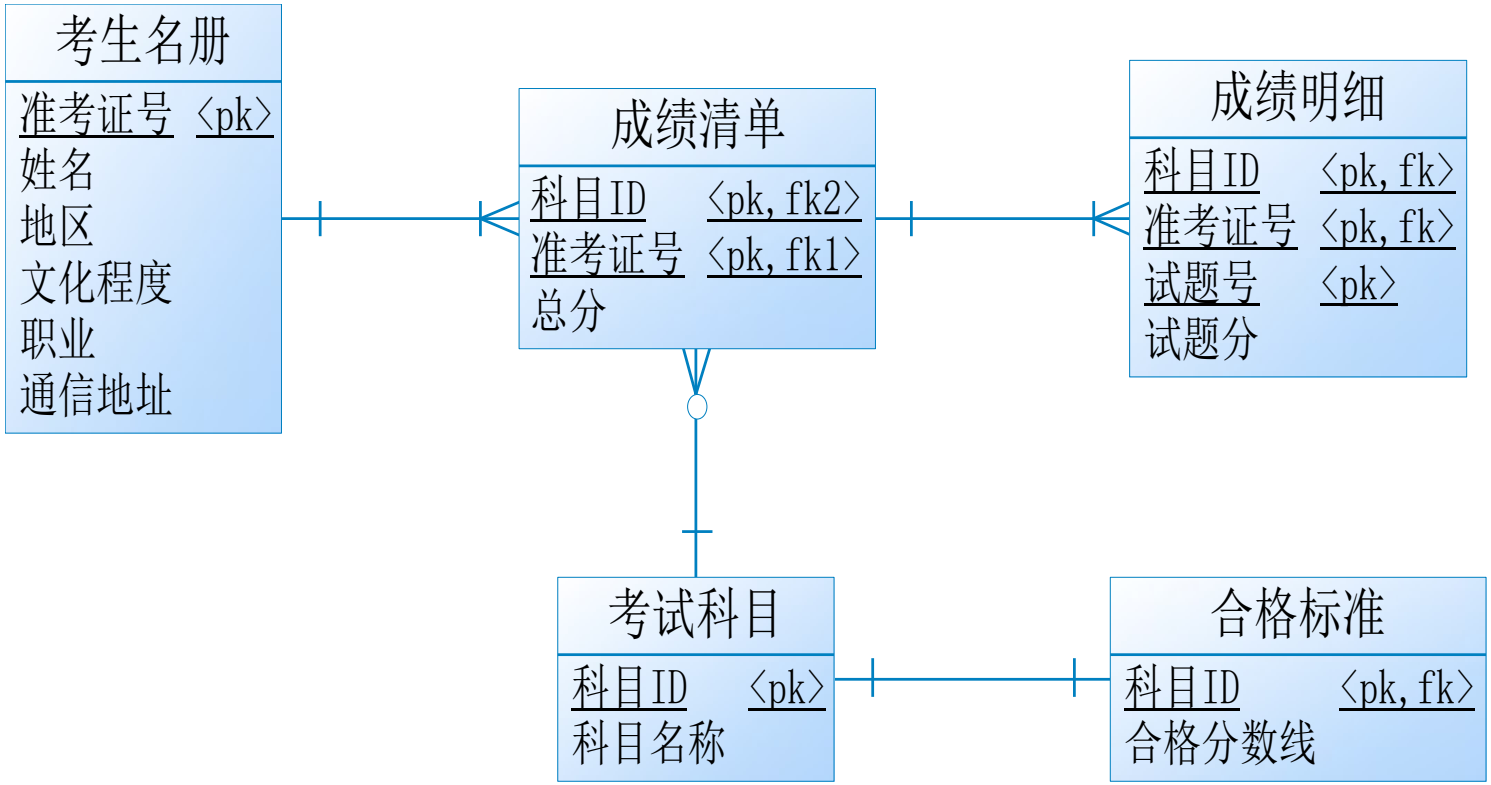
实体和关系


实体
关系
学生
课程
Student
studentID
studentName
BirthDay
Course
courseID
courseName
courseLocation
lecturer
参与/
授课給
键属性
非键属性

关系的基数


A
B
房间
窗
一个A ： 零或一或多个B
汽车
轮胎
一个A： 一或多个B
汽车
导航仪
一个A： 零或一个B
汽车
音响
一或零个A ： 零或多个B
房间
门
多对多关系

ERD 示例


考生名册
准考证号 〈pk〉
姓名
地区
文化程度
职业
通信地址
成绩清单
科目ID 〈pk,fk2〉
准考证号 〈pk,fk1〉
总分
成绩明细
科目ID 〈pk,fk〉
准考证号 〈pk,fk〉
试题号 〈pk〉
试题分
考试科目
科目ID 〈pk〉
科目名称
合格标准
科目ID 〈pk,fk〉
合格分数线

ERD练习

- A城市的公共工作部门决定开发一个基于Web的坑洼跟踪和修补系统(PHTRS):
  - 当报告路面有坑洼时，它们被赋予一个标识号，并依据街道地址、大小(1到10)、位置(路中或路边等)、地区(由街道地址确定)和修补优先级(由坑洼的大小确定)储存起来。
  - 工作单数据和每个坑洼有关，数据包括坑洼位置和大小、修理组标识号、修理组的人数、分配的设备、修复时间长度、坑洼状态(正在处理中、已修复、临时修复、未修复)、使用填料的数量和修复的开销(由使用的时间、人数、使用的材料的设备计算得到)。
  - 最后，产生一个损失文件以保存该坑洼造成的损失信息，并包括居民的姓名、地址、电话号码、损失类型和损失钱数。
  - PHTRS是基于Web的系统，可交互地进行所有的查询。

大 纲

01-结构化分析

结构化分析方法概述
数据流图
分层数据流图的审查
数据字典
描述基本加工的小说明
实体—关系图

02-结构化设计

结构化设计概述
结构设计—概要设计
过程设计—详细设计

结构化设计的内容

- 结构设计—概要设计
  - 逻辑视图 - - 结构图(Structure Chart)
  - 部署视图
  - 进程视图
  - 数据视图 - - 物理数据模型
  - 其它视图
- 过程设计—详细设计
  - 模块的处理过程
    - 流程图，IPO图，N-S图，PAD，PDL等

将结构化分析得到的数据流图映射成软件结构

大 纲

01-结构化分析

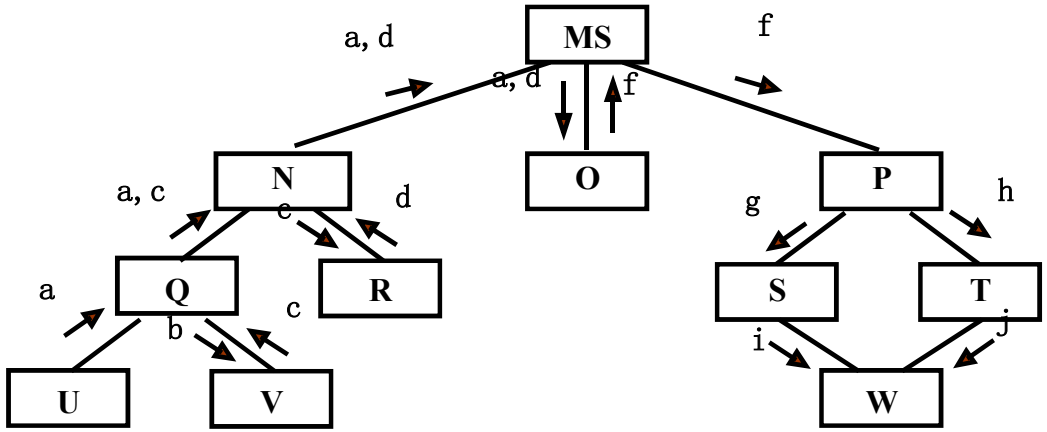
结构化分析方法概述
数据流图
分层数据流图的审查
数据字典
描述基本加工的小说明
实体—关系图

02-结构化设计

结构化设计概述
结构设计—概要设计
过程设计—详细设计

结构设计的工具 - - 结构图

- 用结构图(Structure Chart)来描述软件系统的体系结构
- 描述一个软件系统由哪些模块组成，以及模块之间的调用关系
- 结构图的基本成分有：模块、调用和数据

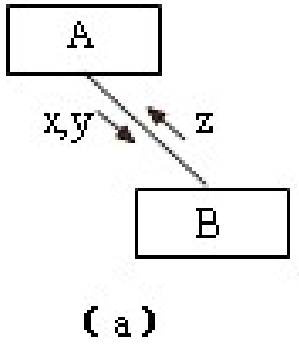


模块(module)

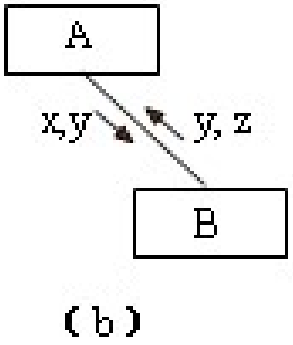
- 模块是指具有一定功能的可以用模块名调用的一组程序语句，包括函数和子程序。
- 一个模块具有其外部特征和内部特征
  - 外部特征包括：模块的接口(模块名、输入/输出参数、返回值等)和模块的功能
  - 内部特征包括：模块的内部数据和完成其功能的程序代码

调用和数据

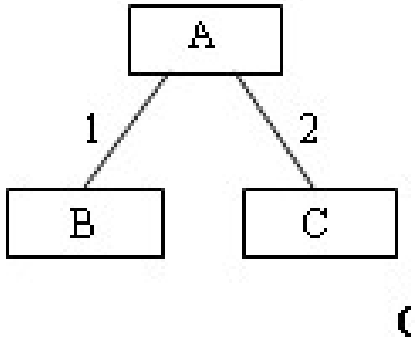
- 调用(call)：用从一个模块指向另一个模块的箭头来表示，其含义是前者调用了后者
  - 为了方便，有时常用直线替代箭头，此时，表示位于上方的模块调用位于下方的模块
- 数据(data)：模块调用时需传递的参数可通过在调用箭头旁附加一个小箭头和数据名来表示



(a)



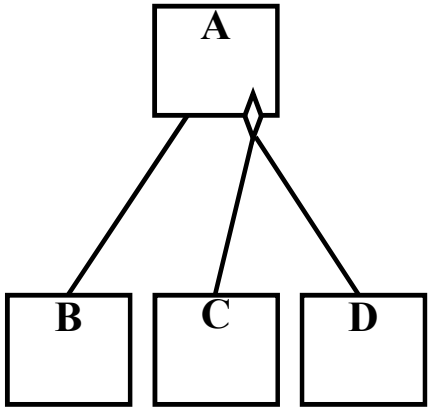
(b)



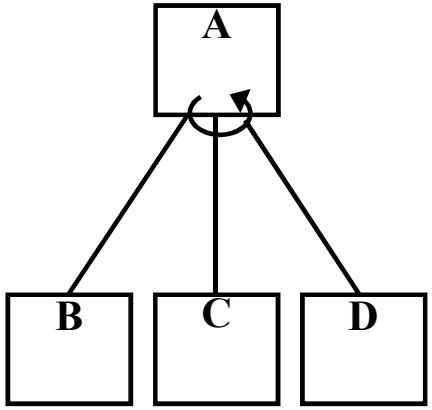
(c)

	输入	输出
1	x,y	z
2	z,w	w

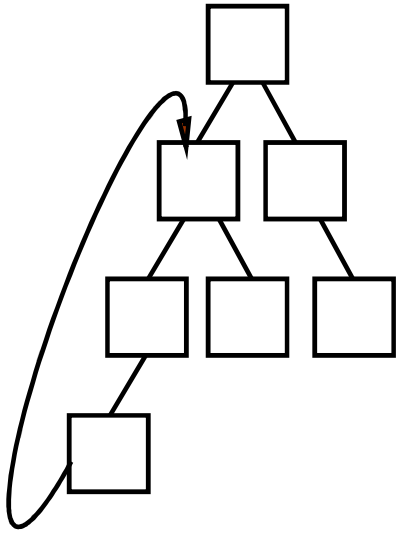
结构图中的辅助符号



条件调用



循环调用



递归调用

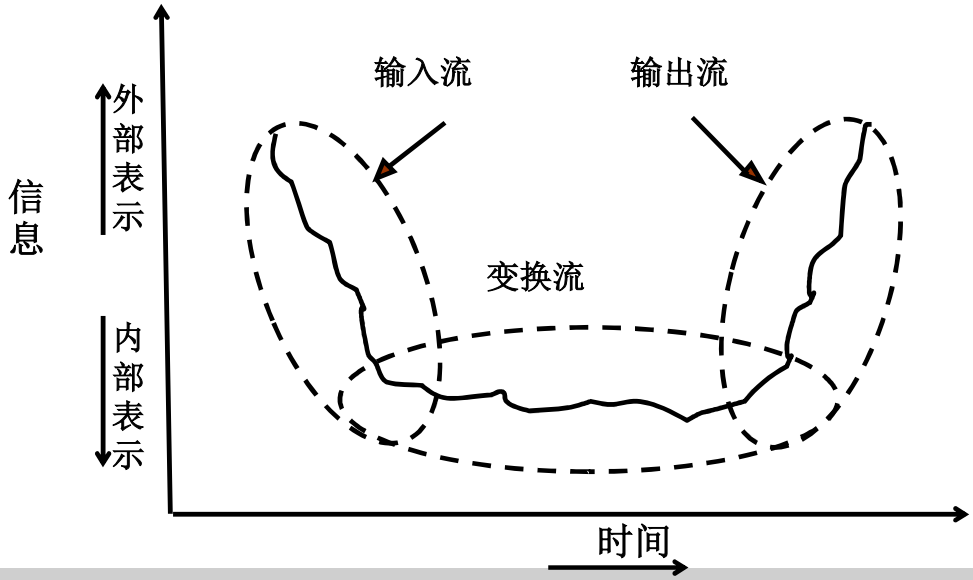
数据流图到结构图的映射

- 结构化设计是将结构化分析的结果(数据流图)映射成软件的体系结构(结构图)

- 将数据流图分为变换型数据流图和事务型数据流图，对应的映射分别称为变换分析和事务分析

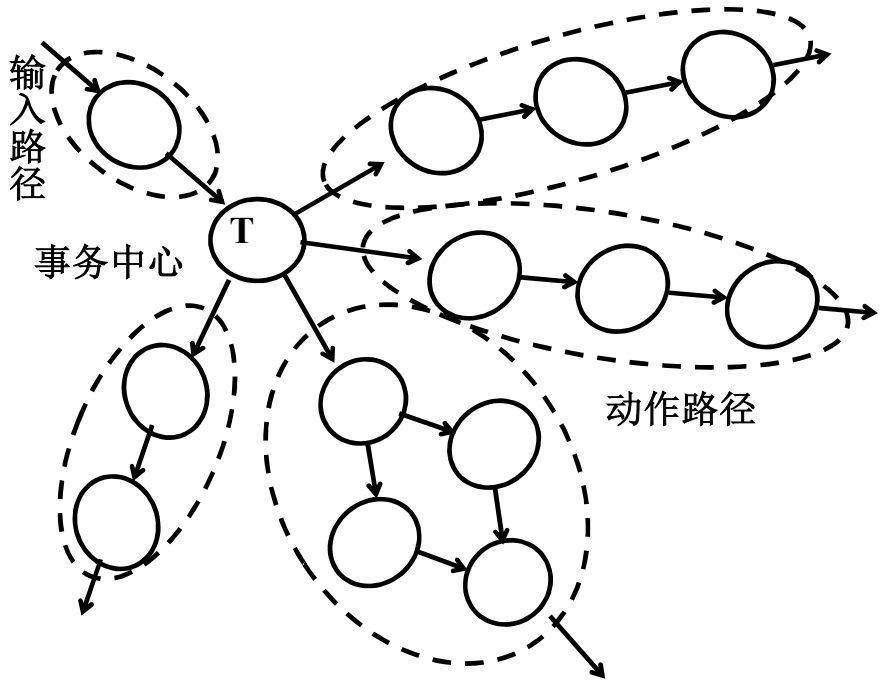
变换流

- 特征：数据流图可明显地分成三部分
  - 输入：信息沿着输入路径进入系统，并将输入信息的外部形式经过编辑、格式转换、合法性检查、预处理等辅助性加工后变成内部形式
  - 变换：内部形式的信息由变换中心进行处理
  - 输出：然后沿着输出路径经过格式转换、组成物理块、缓冲处理等辅助性加工后变成输出信息送到系统外



事务流

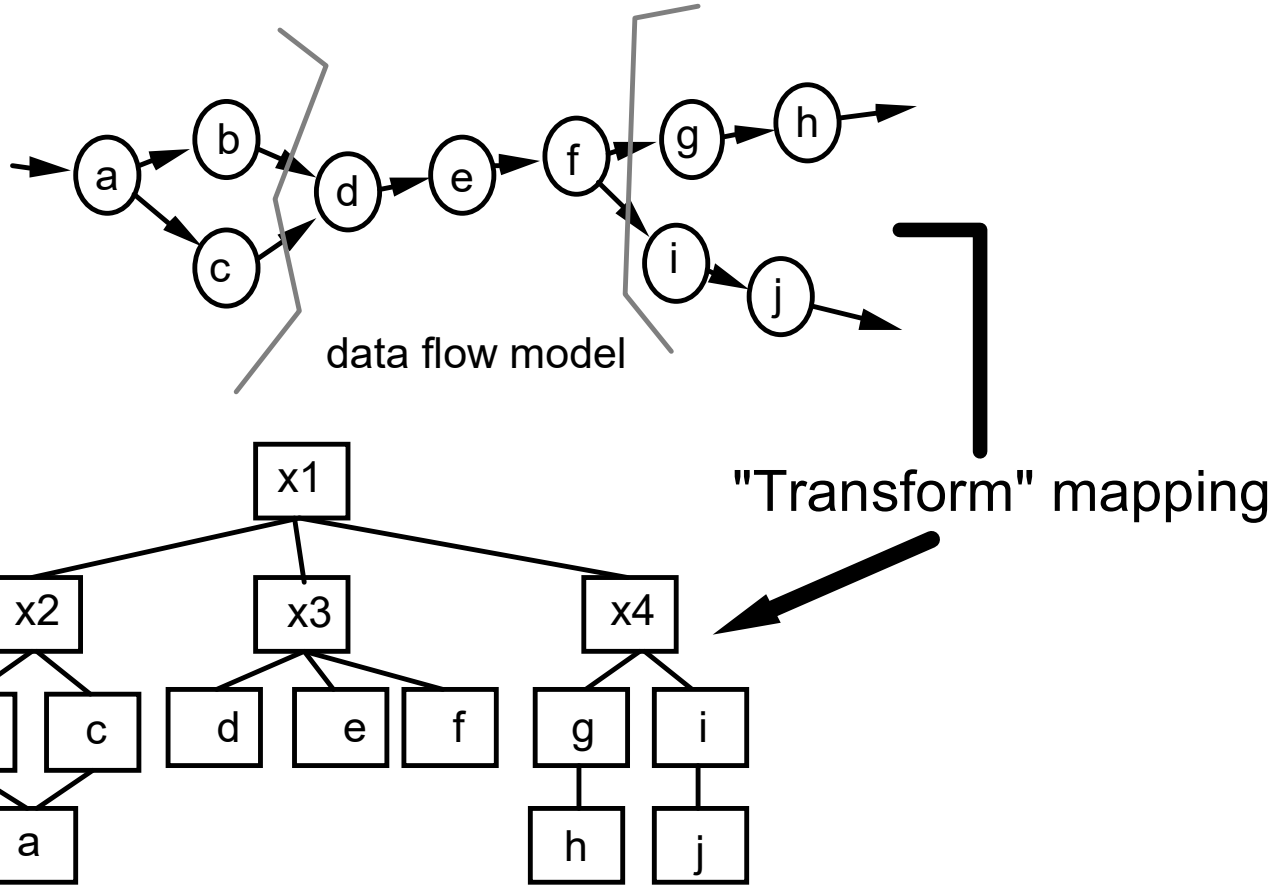
- 特征：数据流沿着输入路径到达一个事务中心，事务中心根据输入数据的类型在若干条动作路径中选择一条来执行
- 事务中心的任务是：接收输入数据(即事务)；分析每个事务的类型；根据事务类型选择执行一条动作路径



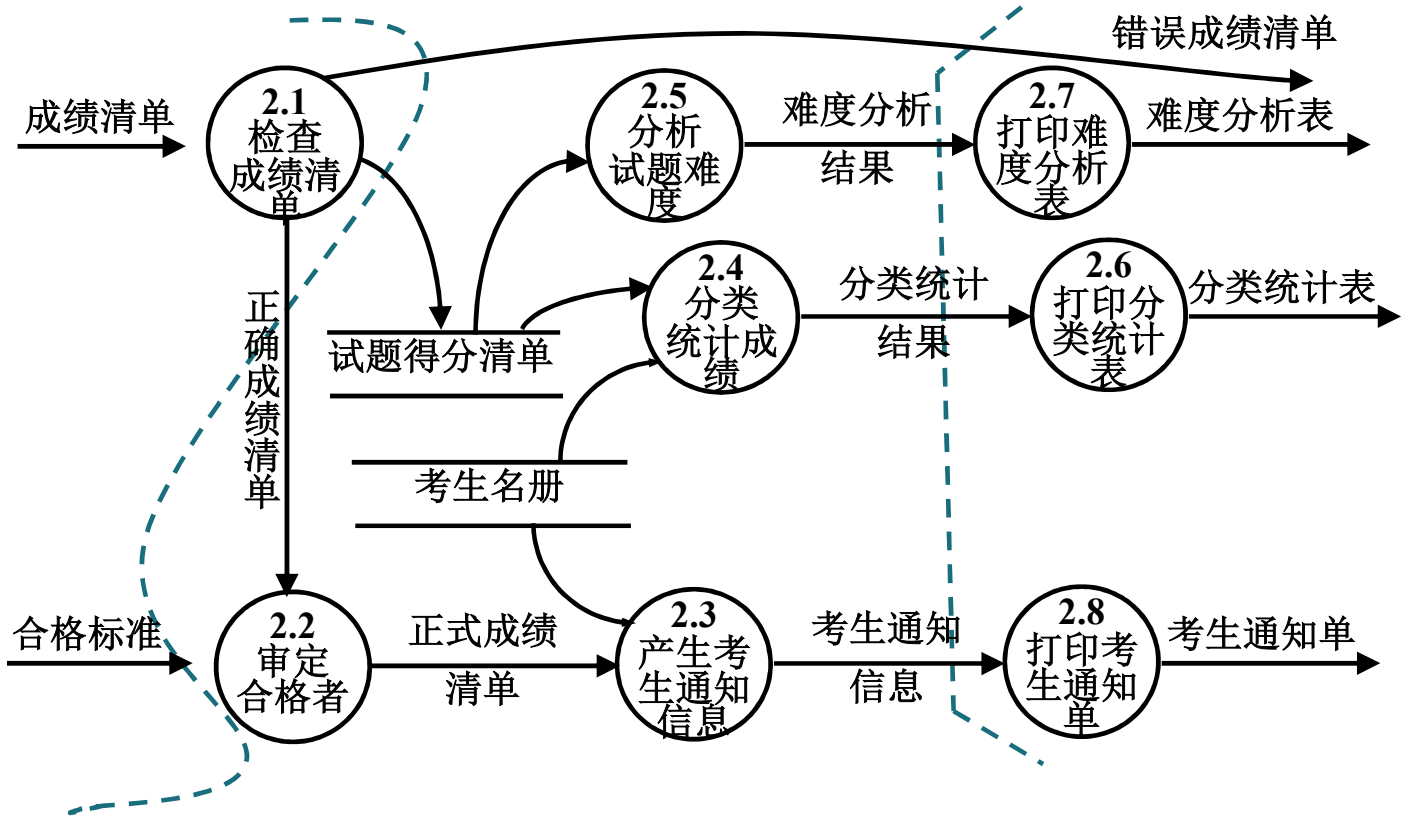
变换分析

- 变换分析的任务是将变换型的DFD映射成初始的结构图，步骤如下：
  - 划定输入流和输出流的边界，确定变换中心
  - 进行第一级分解：将DFD映射成变换型的程序结构
  - 进行第二级分解：将DFD中的加工映射成结构图中的一个适当的模块
  - 标注输入输出信息：根据DFD，在初始结构图上标注模块之间传递的输入信息和输出信息

变换映射

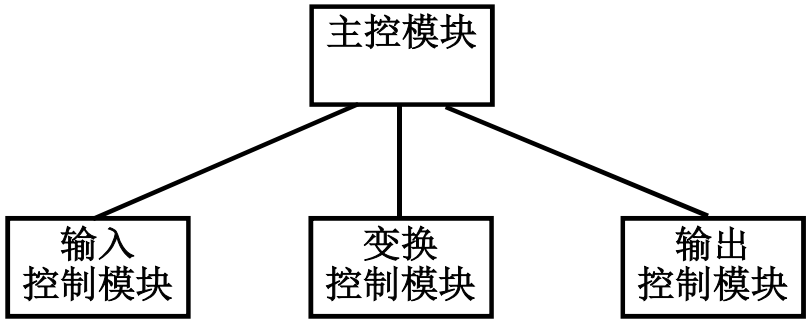

a
b
c
d
e
f
g
h
i
j
data flow model
"Transform" mapping
x1
x2
x3
x4
b
c
d
e
f
g
i
a
h
j

示例：统计成绩子图的输入、输出流边界

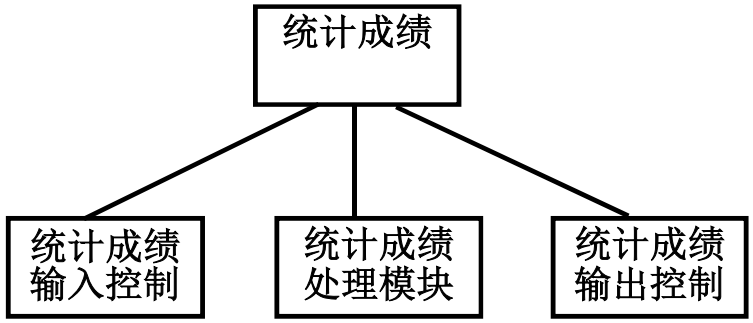


进行第一级分解

- 将DFD映射成变换型的程序结构
- 大型的软件系统第一级分解时可多分解几个模块，以减少最终结构图的层次数
  - 例如，每条输入或输出路径画一个模块，每个主要变换功能各画一个模块

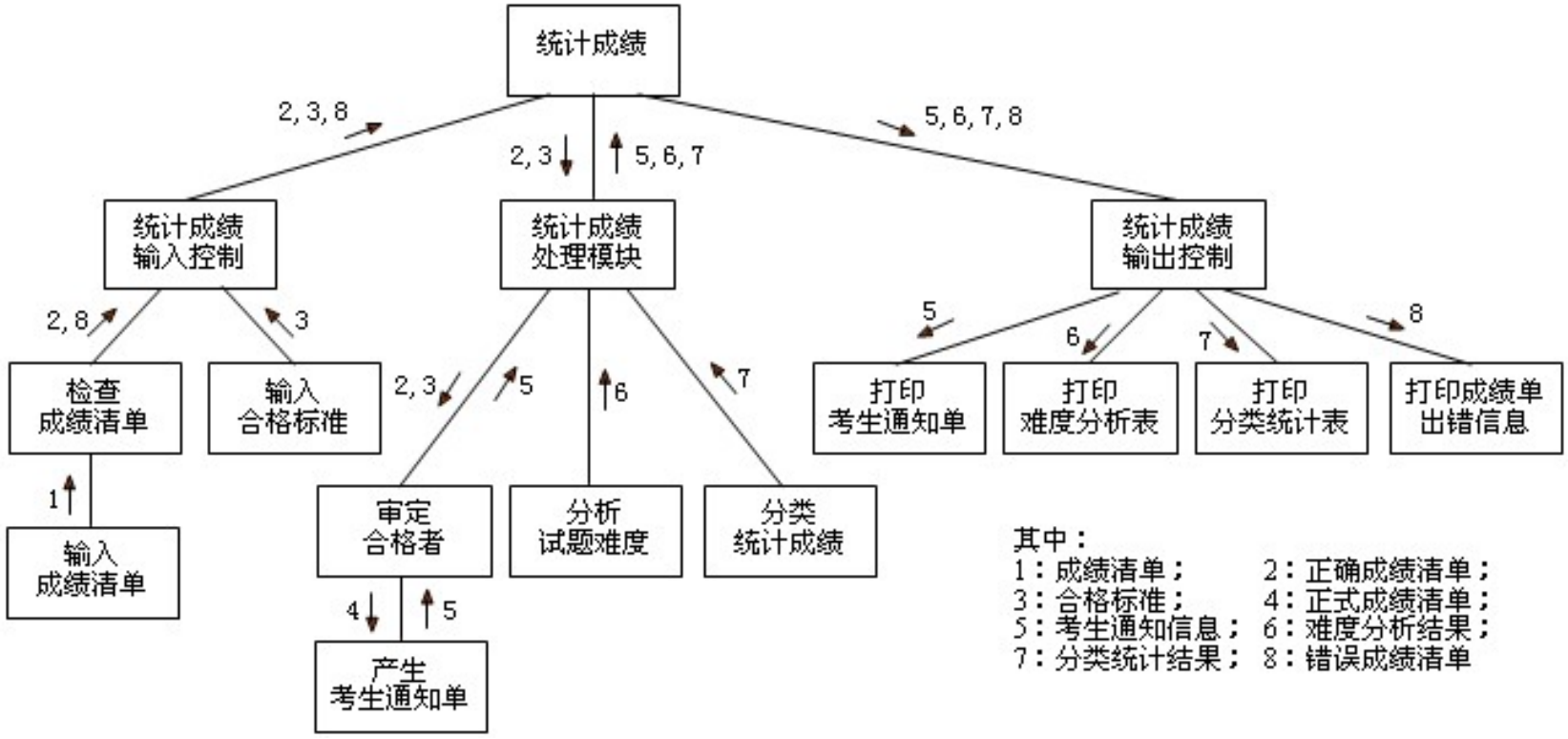


变换型的结构图



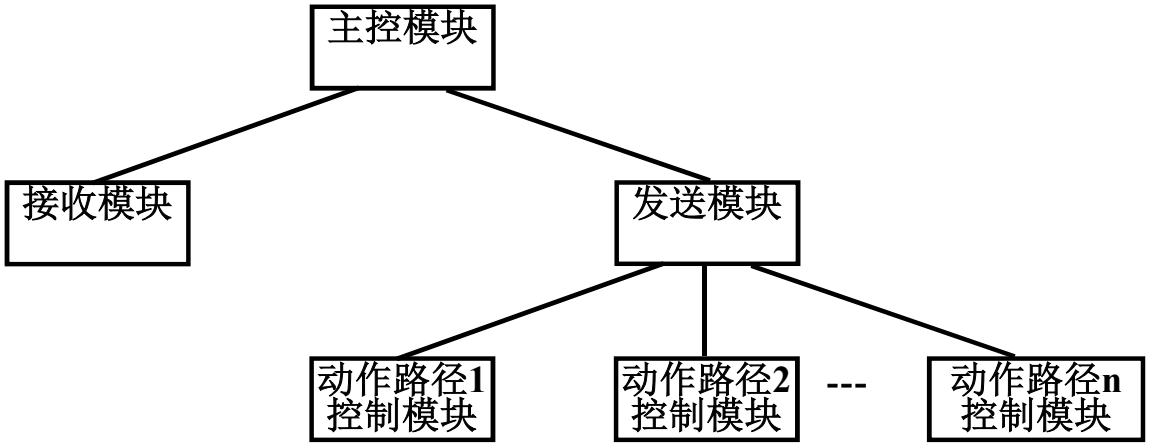
“统计成绩”第一级分解的结构图

“统计成绩”第二级分解的结构图



事务分析

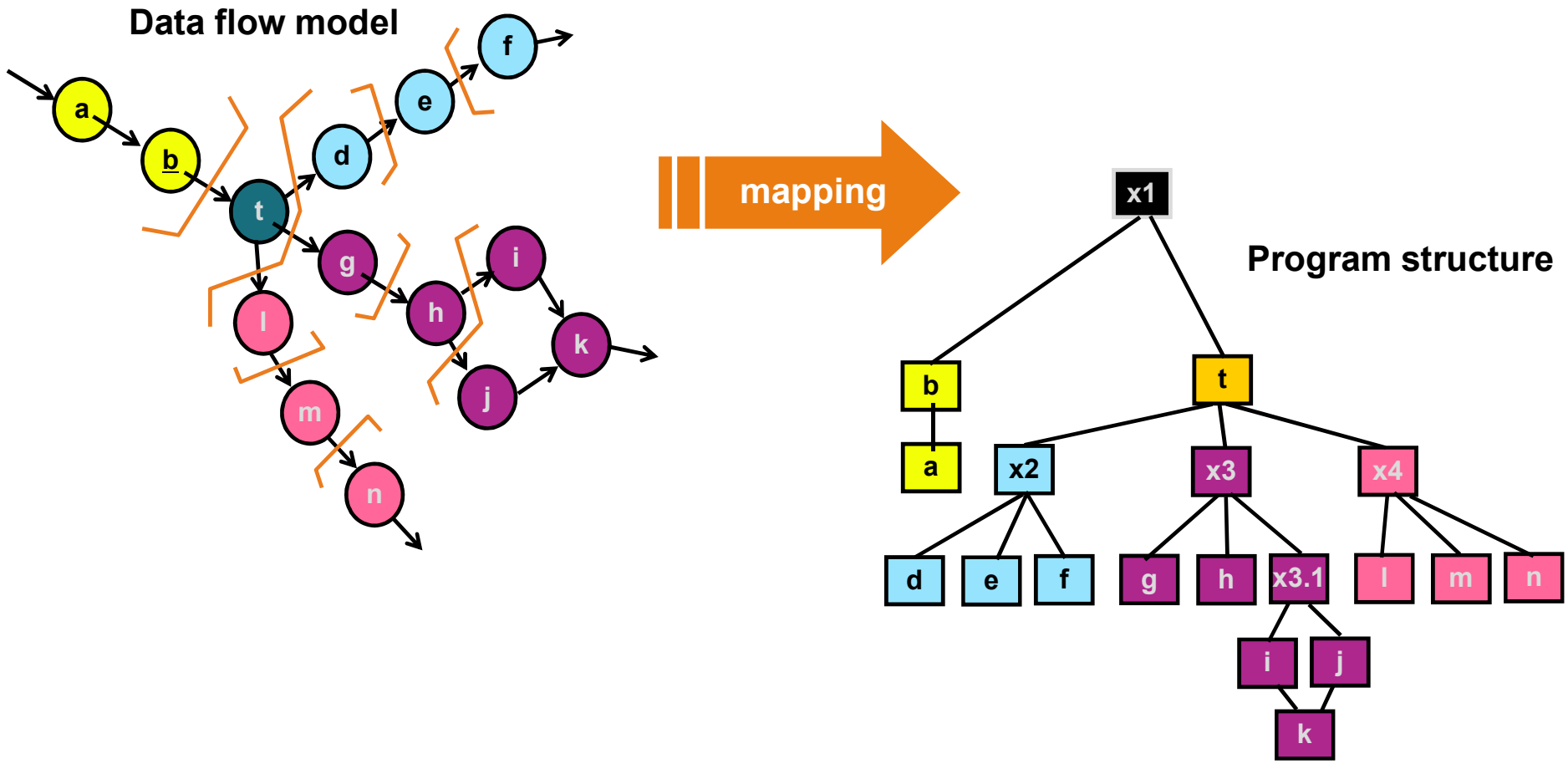
- 将事务型DFD映射成初始的结构图
  - 实例：银行业务中有存款、取款、查询余额、开户、转帐等多种事务，这种软件通常是接收一个事务，然后根据事务的类型执行一个事务处理的功能
- 事务型的结构图包括：
  - 主控模块：完成整个系统的功能
  - 接收模块：接收输入数据(事务)
  - 发送模块：根据输入事务的类型，选择一个动作路径控制模块
  - 动作路径控制模块：完成相应的动作路径所执行的子功能



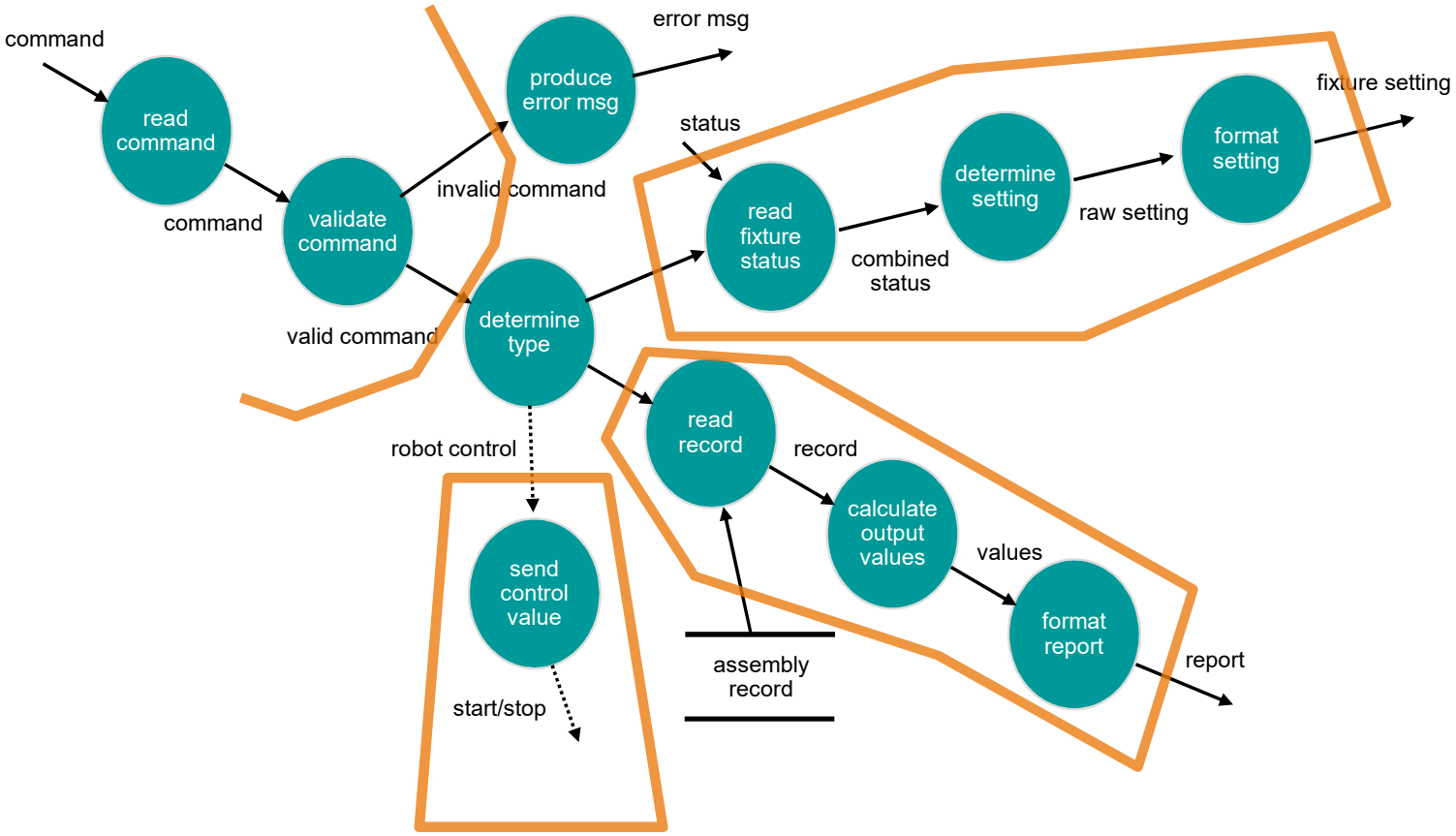
事务分析的步骤

- 在DFD图上确定边界
  - 事务中心
  - 接受部分（包括接受路径）
  - 发送部分（包括全部动作路径）
- 画出SC图框架
  - DFD图的三个部分分别映射为事务控制模块，接受模块和动作发送模块
- 分解和细化接受分支和发送分支

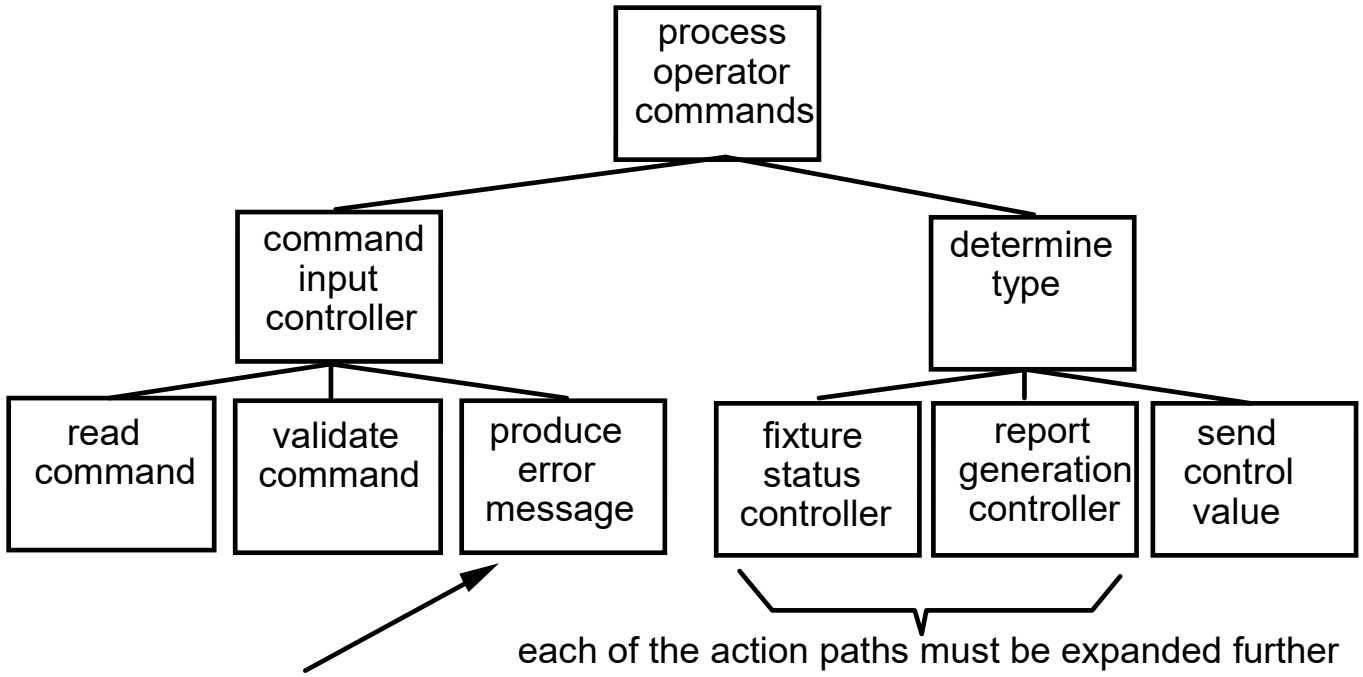
事务映射


Data flow model
a
b
t
d
e
f
g
h
i
k
l
m
j
n
mapping
Program structure
x1
b
t
a
x2
x3
x4
d
e
f
g
h
x3.1
l
m
n
i
j
k

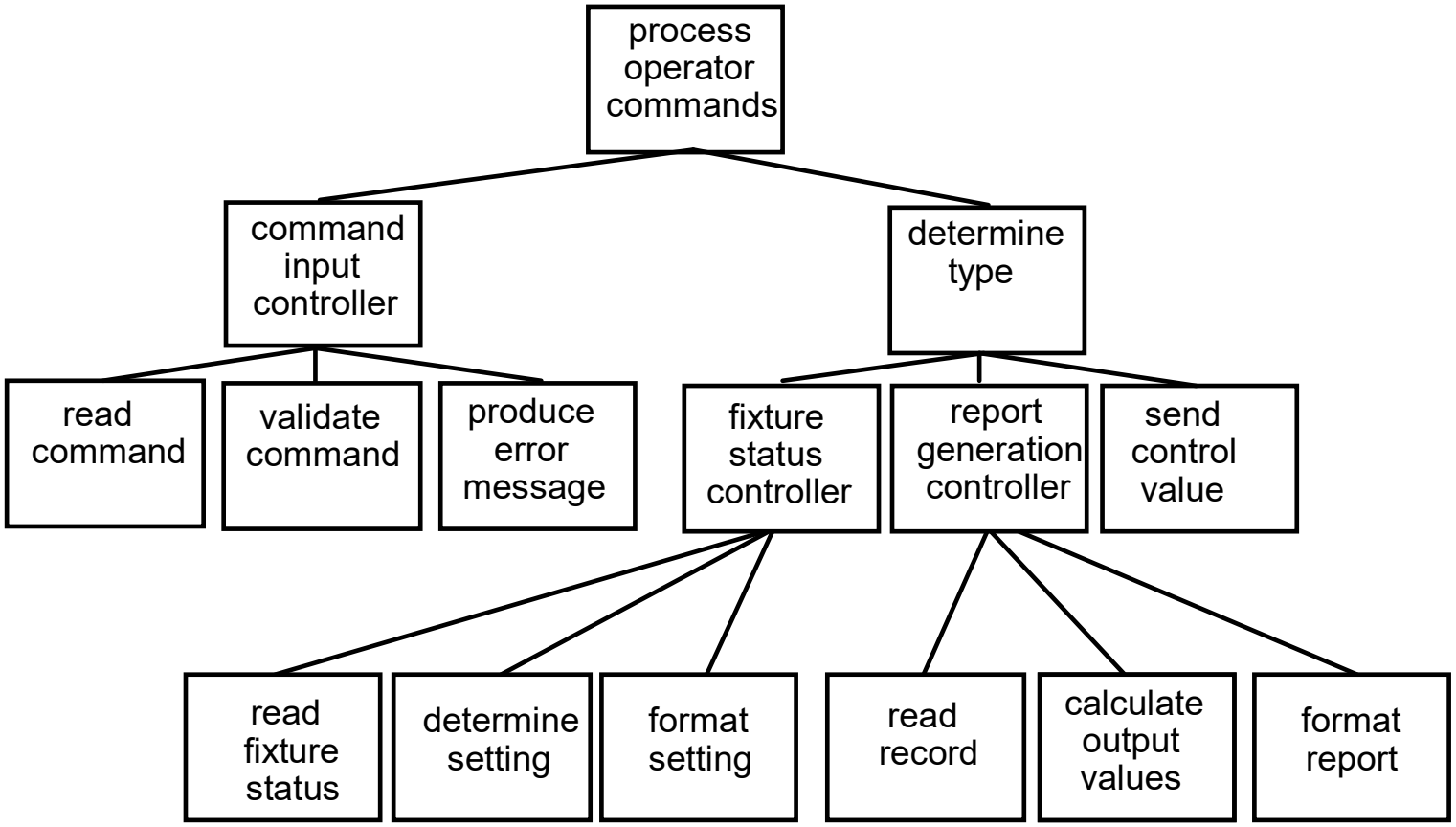
示例：1）划分DFD


command
read command
validate command
command
invalid command
produce error msg
error msg
valid command
determine type
status
read fixture status
combined status
determine setting
raw setting
format setting
fixture setting
robot control
send control value
start/stop
read record
record
calculate output values
values
format report
report
assembly record

示例： 2）画出SC图框架


process operator commands
command input controller
determine type
read command
validate command
produce error message
fixture status controller
report generation controller
send control value
each of the action paths must be expanded further

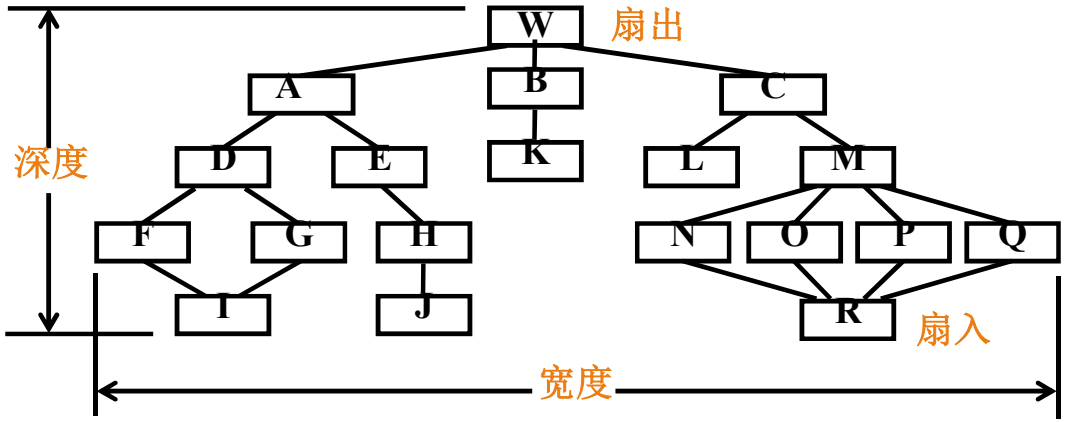
示例：3）细化SC图


process operator commands
command input controller
determine type
read command
validate command
produce error message
fixture status controller
report generation controller
send control value
read fixture status
determine setting
format setting
read record
calculate output values
format report

优化结构图

结构图的相关度量

- 深度：程序结构图中控制的层数，例如图中所示的结构图的深度是5
- 宽度：程序结构图中同一层次上模块总数的最大值，例如图中所示的结构图的宽度为7
- 扇出(fan out)：该模块直接调用的模块数目。例如，例如图中模块M的扇出是4，模块A的是2，模块B的扇出是1
- 扇入(fan in)：能直接调用该模块的模块数目。例如图中模块G的扇入是1，模块I的扇入是2，模块R的扇入是4



相关指标的含义

- 深度和宽度在一定程序上反映了程序的规模和复杂程度
  - 相对而言，如果程序结构图的深度和宽度较大，则说明程序的规模和复杂程度都较大
  - 模块的扇入扇出会影响结构图的深度和宽度，例如减少模块的扇出，可能导致宽度变小而深度增加
- 一个模块的扇出过大通常意味着该模块比较复杂，然而扇出太少，可能导致深度的增加
  - 一般情况，一个模块的扇出以3～9为宜
- 一个模块的扇入表示有多少模块可直接调用它，它反映了该模块的复用(reuse)程度，因此模块的扇入越大越好

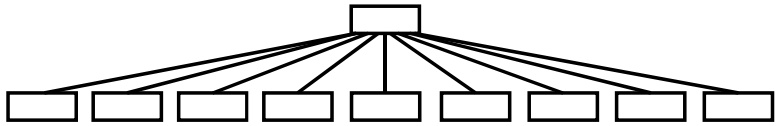
启发式设计策略

按照模块化设计原则，相应的启发式设计策略如下：

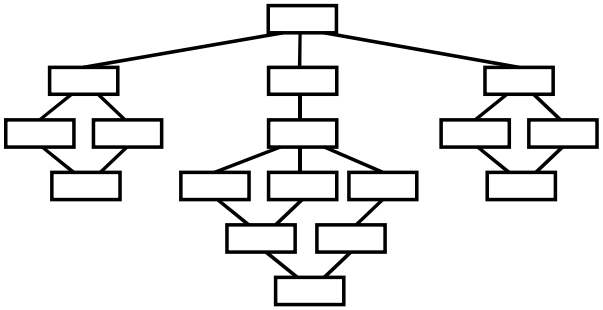
1) 降低耦合度，提高内聚度

2) 避免高扇出，并随着深度的增加，力求高扇入

- 避免如图a那样的“平铺”形态，较好的结构图形态是如图b那样的“椭圆”型



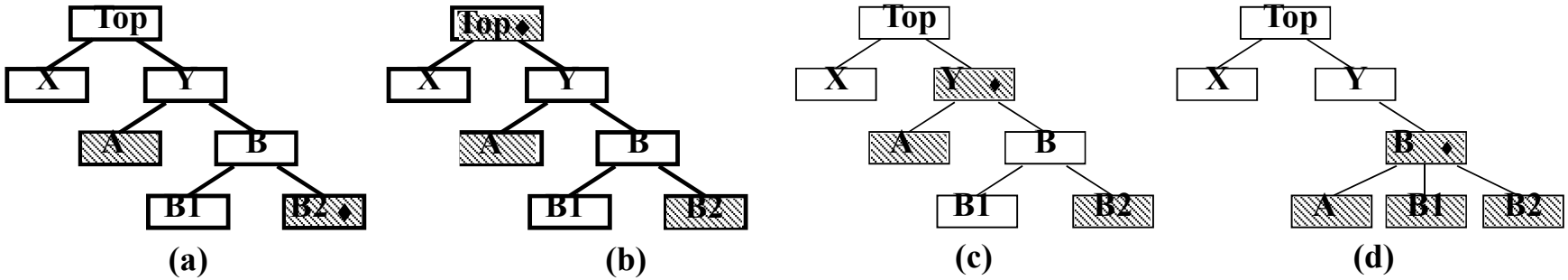
(a)平铺形态



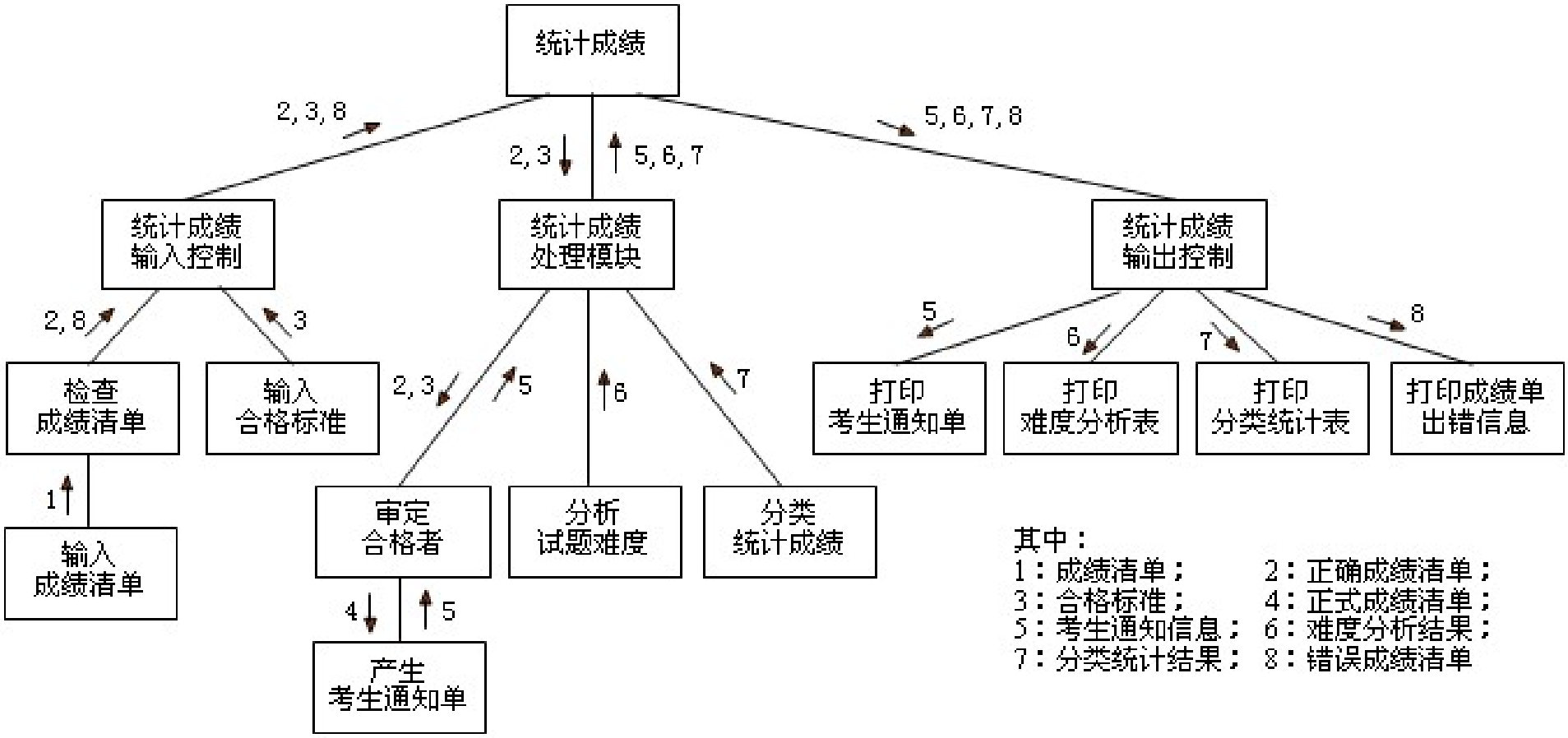
(b)椭圆形态

3) 模块的影响范围应限制在该模块的控制范围内

- 图a中，模块B2的影响范围(模块A)不在其控制范围(模块B2)内
- 图b中，决策控制是在顶层模块，其影响范围(A、B2)在控制范围内，但是从决策控制模块到被控模块之间相差多个层次
- c和d较合适，d为最好

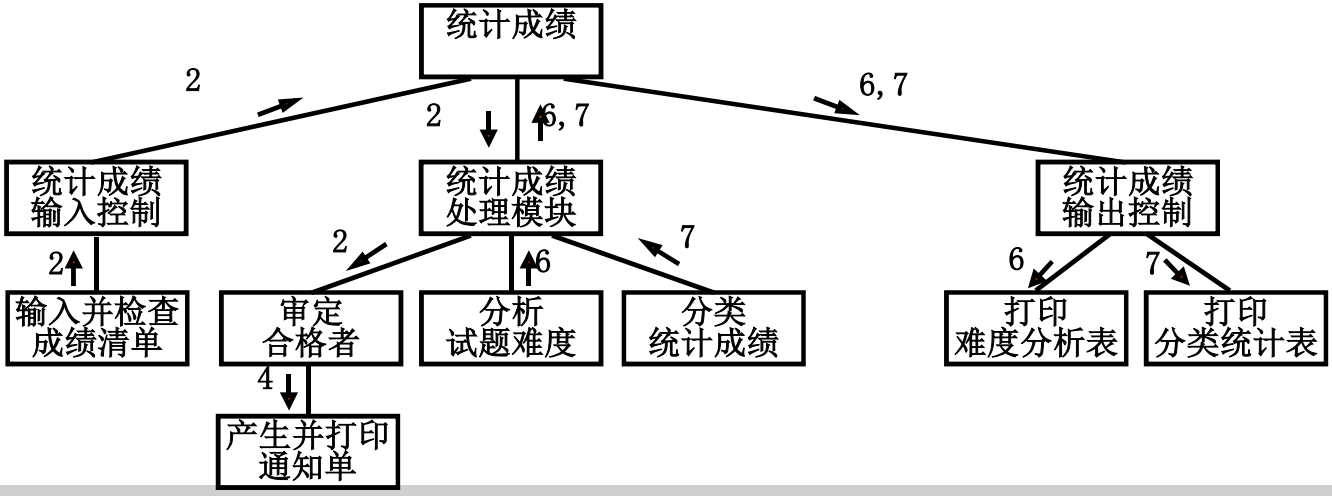


“统计成绩”结构图的改进-0



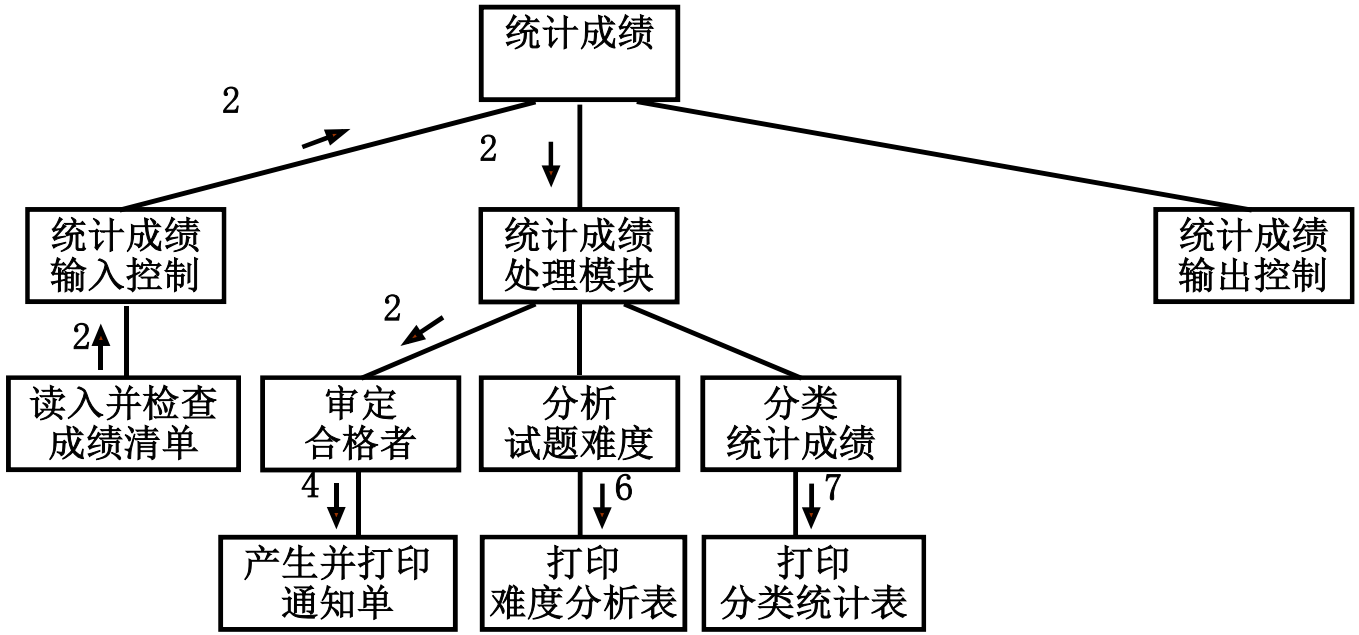
“统计成绩”结构图的改进-1

- 先将一些比较简单的模块合并到与其功能相一致的模块中，以减少耦合度
  - 将“输入成绩清单”、“检查成绩清单”、“打印成绩单出错信息”合并成“输入并检查成绩清单”
  - 将“输入合格标准”与“审定合格者”合并，仍取名“审定合格者”，但它包含读入合格标准功能
  - 将“产生考生通知单”与“打印考生通知单”合并成“产生并打印考生通知单”



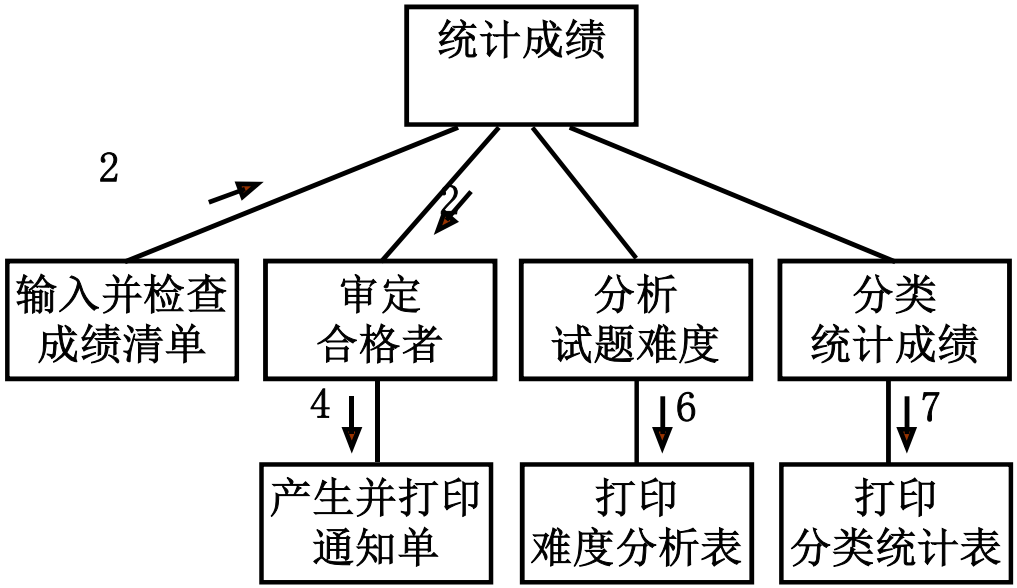
“统计成绩”结构图的改进-2

- 降低模块间的耦合程度
  - 将“打印难度分析表”模块和“打印分类统计表”模块分别作为“分析试题难度”模块和“分类统计成绩”模块的下属模块



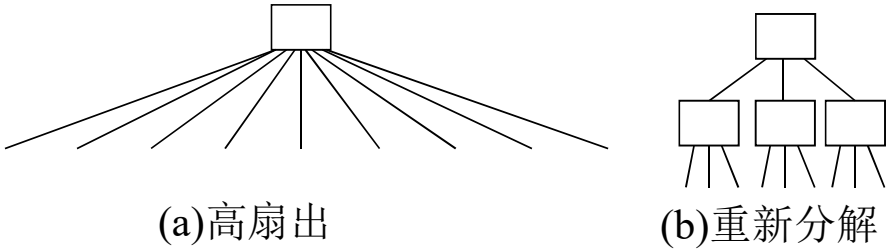
“统计成绩”结构图的改进-3

- 删去“统计成绩输出控制”
- “统计成绩输入控制”模块和“统计成绩处理模块”均为“管道”模块，也可删去



结构图改进技巧

- 减少模块间的耦合度
- 消除重复功能
- 消除“管道”模块
- 模块的大小适中
- 避免高扇出
- 应尽可能研究整张结构图，而不是只考虑其中的一部分



(a)高扇出　　(b)重新分解

高扇出时重新分解

大 纲

01-结构化分析

结构化分析方法概述
数据流图
分层数据流图的审查
数据字典
描述基本加工的小说明
实体—关系图

02-结构化设计

结构化设计概述
结构设计—概要设计
过程设计—详细设计

过程设计

- 目的
  - 确定模块采用的算法和块内数据结构
- 任务：编写软件的“过程设计说明书”
  - 为每个模块确定采用的算法
  - 确定每一模块使用的数据结构
  - 确定模块接口的细节

过程设计的原则

- 清晰第一的设计风格
- 结构化的控制结构
- 逐步细化的实现方法

过程设计工具

- IPO图（Input Process Output Diagram）
- 流程图（Flow Diagram）
- N-S图
- PAD图（Problem Analysis Diagram）
- PDL语言

IPO图

系统名称：订单处理　　设计者：

模块名称：确定能否供货　　日期：

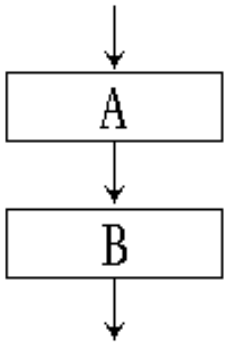
上层调用模块 处 理	可调用下层模块 无
输入： 订单订货量 X 库存信息 Y	输出： 供货标志 I 缺货信息 Z

处理：

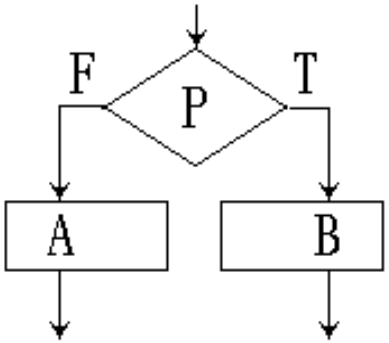
IF　Y—X>0　THEN　I=1　(可供货)

ELSE　I=2　OUTPUT　Z　(缺货)

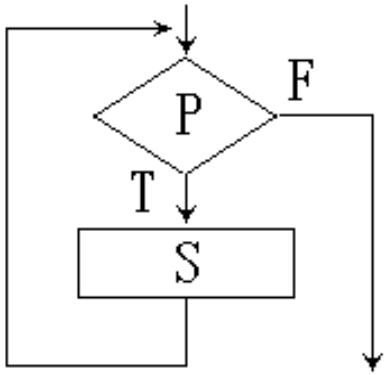
流程图



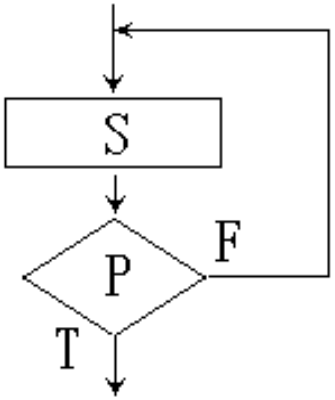
① 顺序型



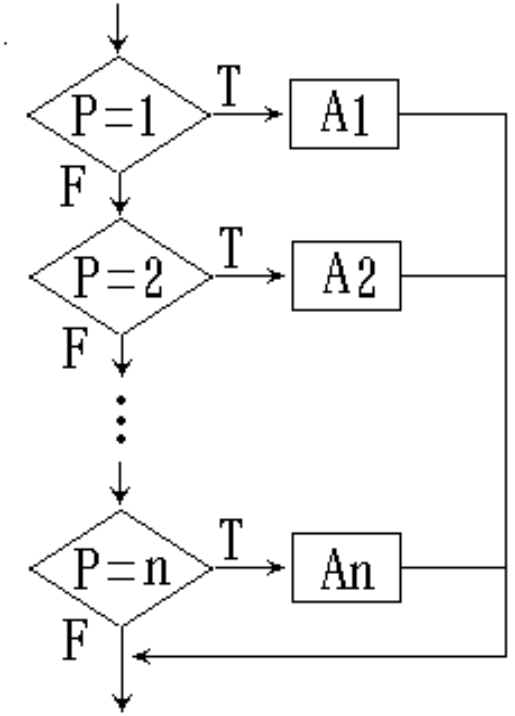
② 选择型



③ 先判定型循环
（DO-WHILE）

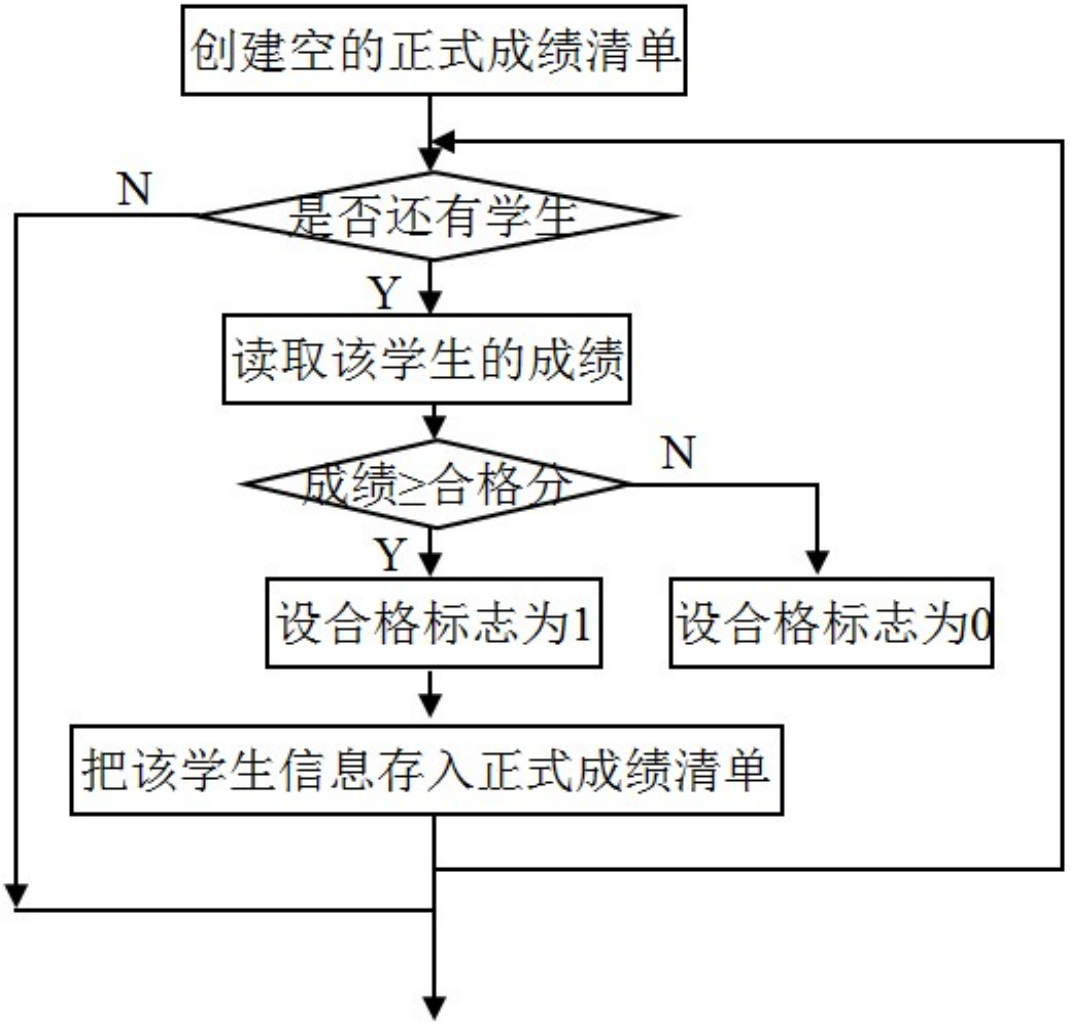


④ 后判定型循环
（DO-UNTIL）



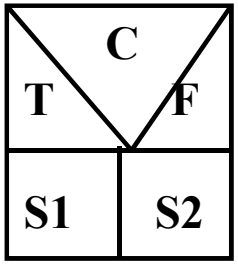
⑤ 多情况选择型
（CASE型）

“审定合格者”模块的程序流程图

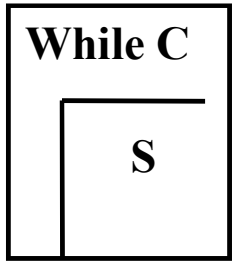


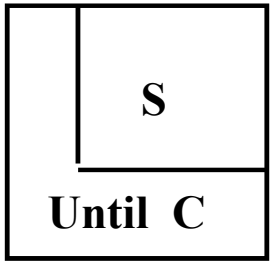
N-S图（盒图）

顺序



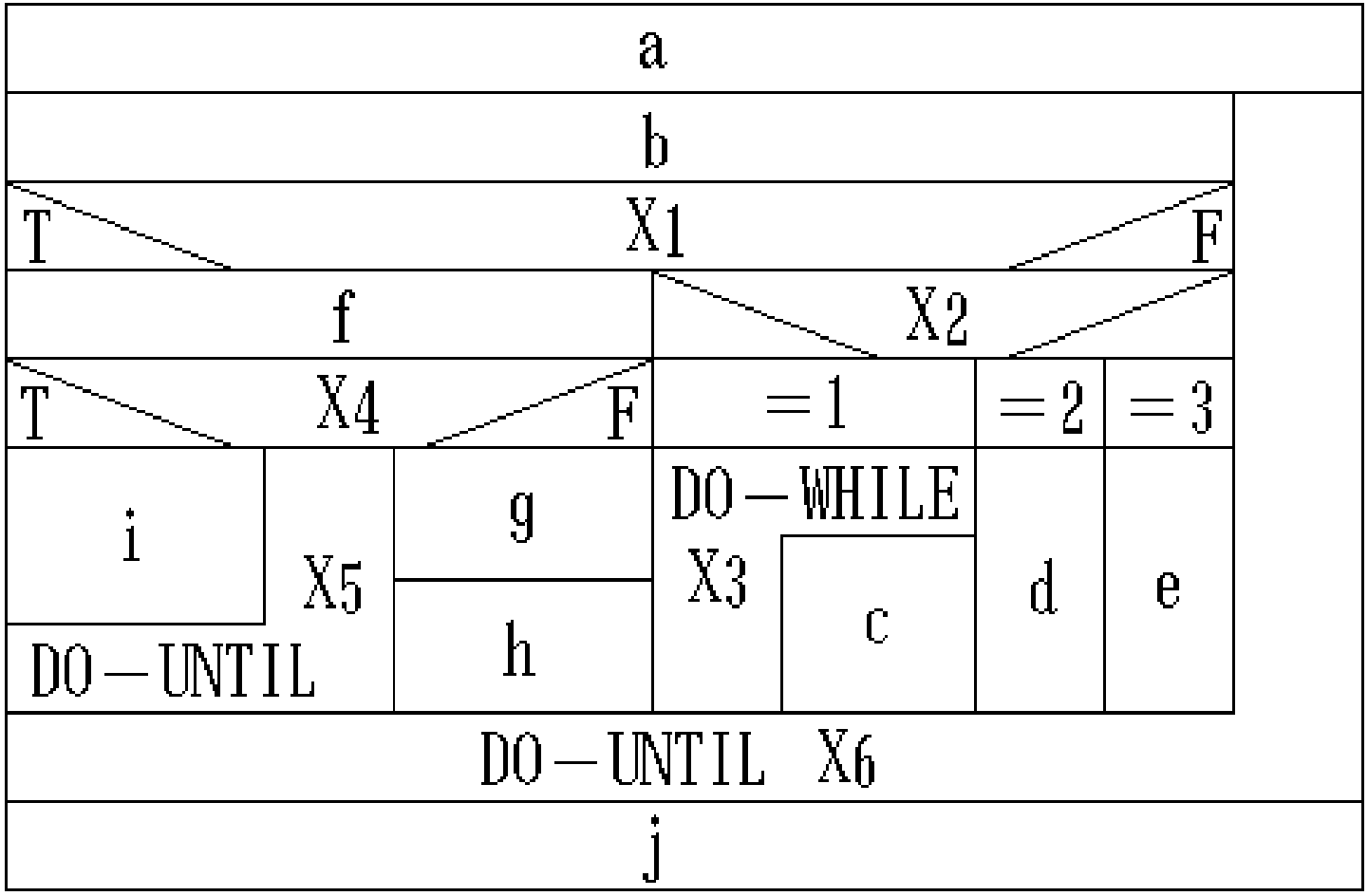
选择



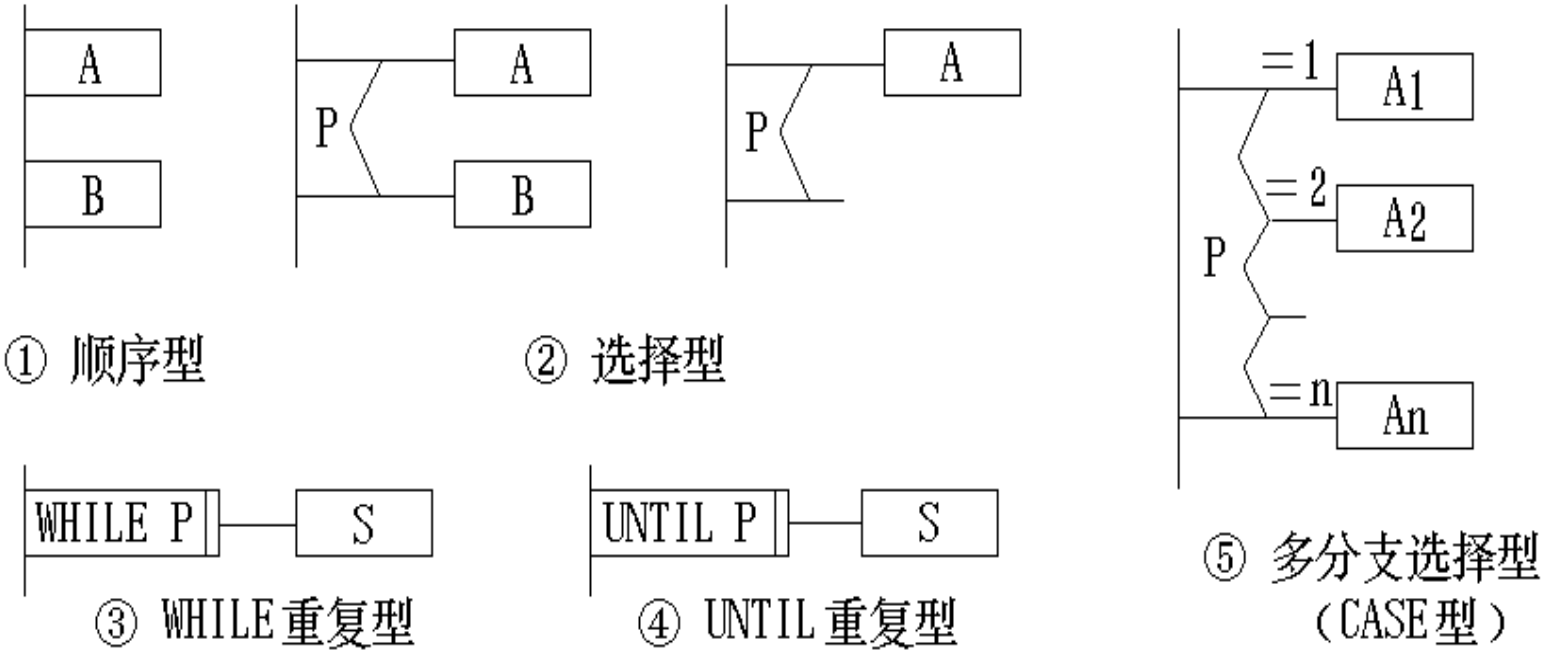


循环

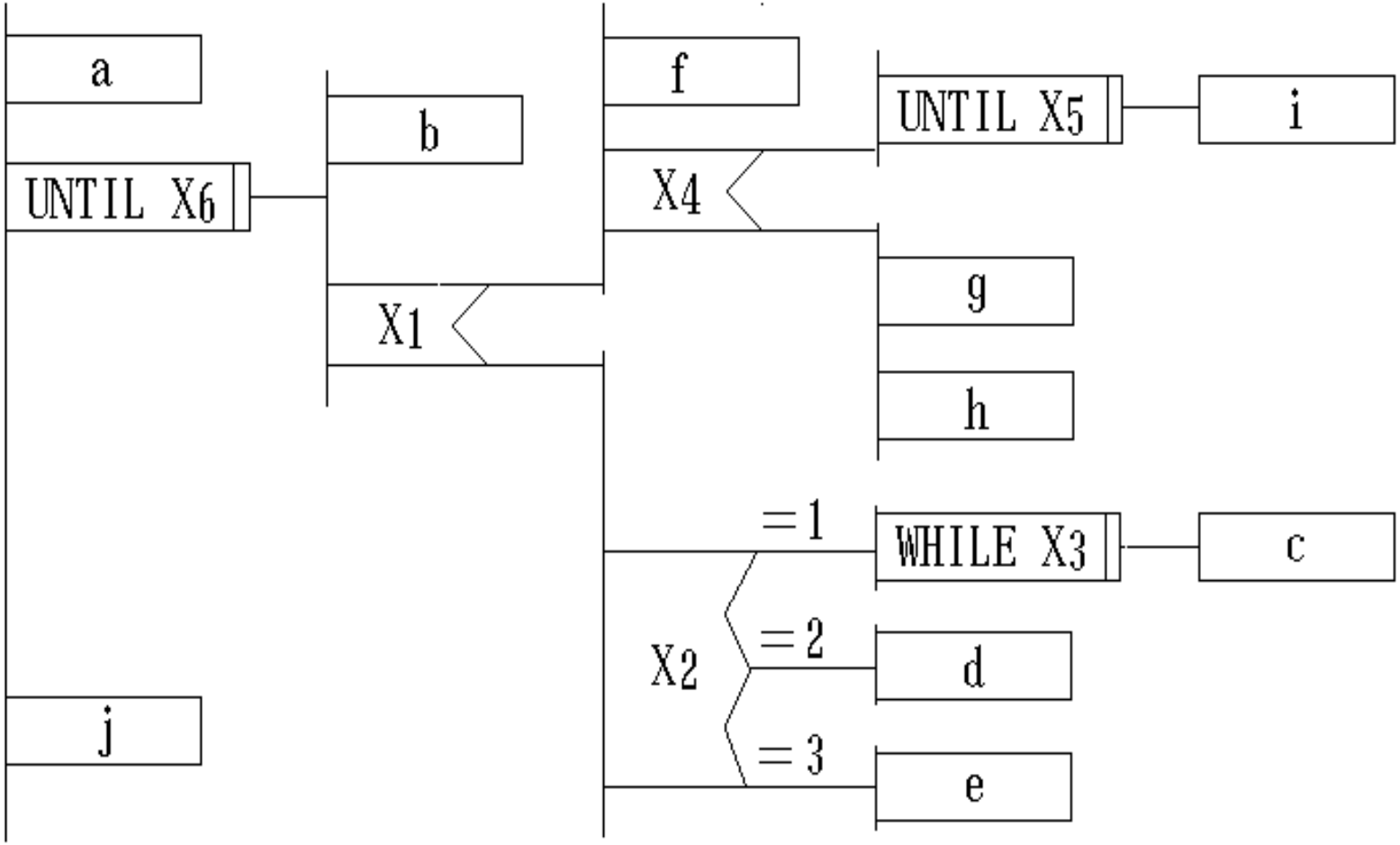
N-S图示例


a
b
T
X1
F
f
X2
T
X4
F
=1
=2
=3
i
X5
g
h
DO—WHILE
X3
c
d
e
DO—UNTIL
DO—UNTIL X6
j

问题分析图(PAD)


A
B
P
A
B
P
A
=1
A1
=2
A2
P
=n
An
① 顺序型
② 选择型
WHILE P
S
UNTIL P
S
③ WHILE重复型
④ UNTIL重复型
⑤ 多分支选择型
(CASE型)

PAD示例



PDL (Program Design Language)

- PDL是一种用于描述功能模块的算法设计和加工细节的语言。称为设计程序用语言。它是一种伪码。
- 伪码的语法规则分为“外语法”和“内语法”。
- PDL具有严格的关键字外语法，用于定义控制结构和数据结构，同时它的表示实际操作和条件的内语法可使用自然语言的词汇。

示例: 拼词检查程序

- PROCEDURE spellcheck IS
  BEGIN
    split document into single words
    look up words in dictionary
    display words which are not in dictionary
    create a new dictionary
  END spellcheck