软件工程

第5-1章 结构化分析

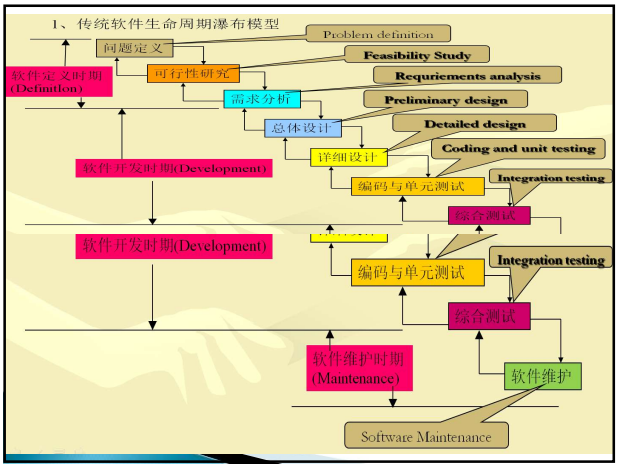
@教材3.2节

结构化分析

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）

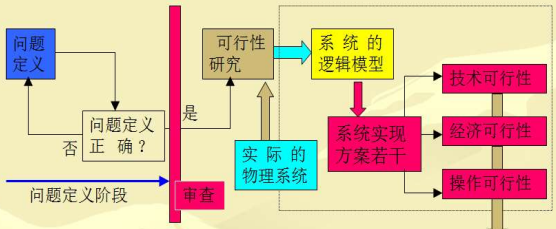
1. **结构化分析方法概述**
2. **面向数据流的软件建模**
3. **分层数据流图的审查**
4. **数据字典**
5. **描述基本加工的小说明**
6. **实体—关系图**
7. **面向数据流的需求分析过程**

1、传统软件生命周期瀑布模型



2、 可行性研究的任务

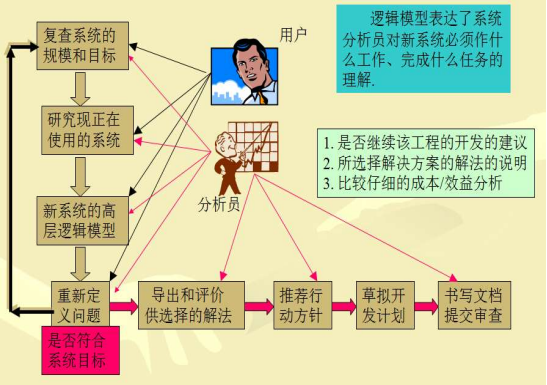
用最小的代价在尽可能短的时间内确定问题是否能够解决。



可行性研究阶段任务(示意图)

可行性研究需要的时间长短取决于工程的规模,一般来说,其成本只能占预期的工程总成本的5%~10%.

3、可行性研究的步骤

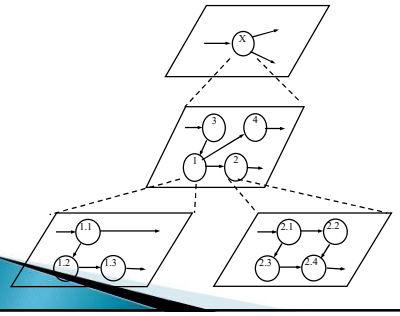


1、结构化方法概述

- 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：
  - 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

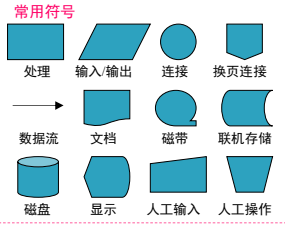
2、 面向数据流的软件建模

（1） 系统流程图(System Flow Diagram)

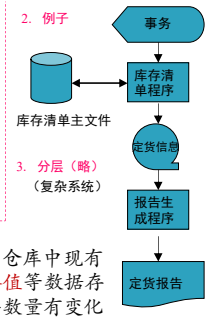
系统流程图：描绘物理系统的工具，其基本思想是用图形符号以黑盒子形式描绘系统里面的每个部件（程序、文件、数据库、表格、人工过程等），表达的是信息在系统各部件之间流动的情况。

系统物理模型！

1. 常用符号



2. 例子



3. 分层（略）（复杂系统）

某装配厂有一座存放零件的仓库，仓库中现有各种零件的数量和每种零件的库存临界值等数据存放在库存清单主文件中。当仓库中零件数量有变化时，应及时修改库存清单主文件，如果哪种零件的库存量少于它的库存量临界值，则应该报告给采购部门以便定货，规定每天向采购部门送一次定货报告。

（2）数据流图

数据流图是系统分析的重要手段，也是设计与实现目标系统的基础。

系统逻辑模型描述分三个方面：

System Logistic Model

- 数据流图(DFD Data Flow Digraph)
- 数据字典(DD DataDictionary)
- 加工/处理说明(IPO)

以下从三个方面介绍数据流图：

- 数据流图的定义
- 如何画数据流图
- 数据流图的用途

数据流图的定义

- 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：

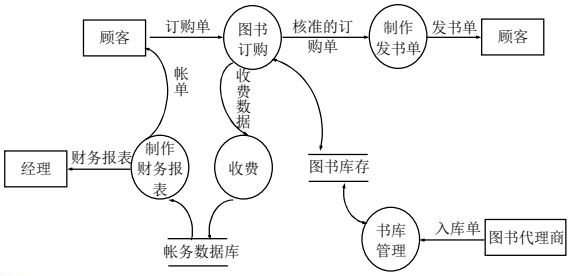
→ 数据流(data flow)：由一组固定成分的数据组成，代表数据的流动方向

○ 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（数据存储）

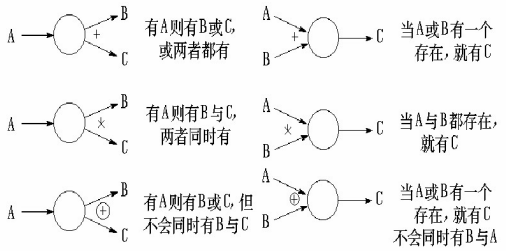
□ 源或宿(source or sink)：由一组固定成分的数据组成，代表数据的流动方向

示例：图书订购系统DFD

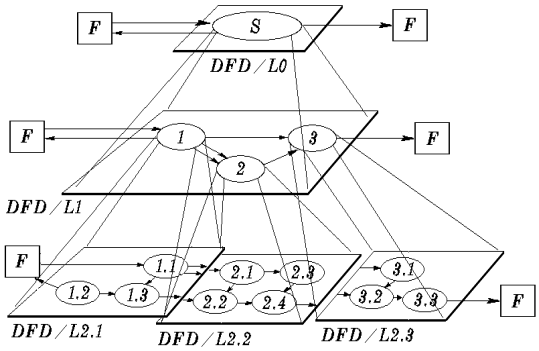


数据流图的扩充符号

- 描述一个加工的多个数据流之间的关系
  - 星号(*)、加号(+)、异或(⊕)



- 如何画数据流图
  - 基本方法：自顶向下逐层分解。
  - 分解原则：上层是下层的抽象，下层是上层的分解。
  - 直到所有的处理都足够简单，不必再分解为止。通常把这种不能再分解的加工称为-----“基本处理”



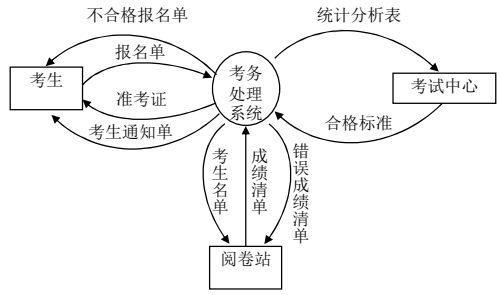
分层的数据流图1

- **在多层数据流图中，顶层流图仅包含一个处理，它代表被开发系统。它的输入流是该系统的输入数据，输出流是系统所输出数据**
- **底层流图是指其处理不需再做分解的数据流图，它处在最底层**
- **中间层流图则表示对其上层父图的细化。它的每一处理可能继续细化，形成子图。**

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。

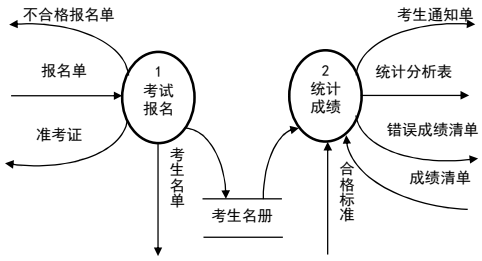
DFD细化－确定数据流

- 在父图中某加工分解而成的子图中，子图的输入/输出数据流要**保持一致性**
- 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（数据存储）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（数据存储）

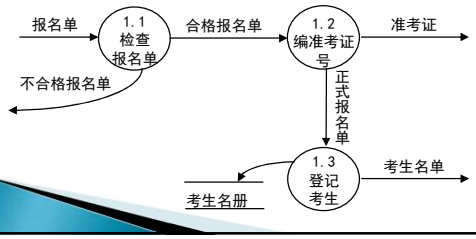
- 如果父图中该加工存在读写文件的数据流，子图中保持一致性
- 在分解子图中，如果需要保存某些中间数据以备后用，则可以创建新的文件（存储）
- 新文件(首次出现的文件)至少应有一个写入和一个加工来读写文件的记录
- 从父图中继承下来的文件在子图中可能只进行读，或只写

示例：考务处理系统0层图

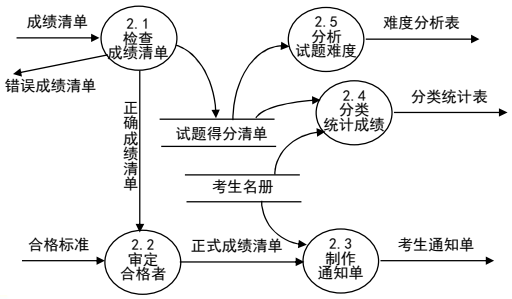


示例：考务处理系统－考试报名1子图

- 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)

注意：

（1）**若作下一步分解时考虑的是实现策略就停止分解，例如：循环；**

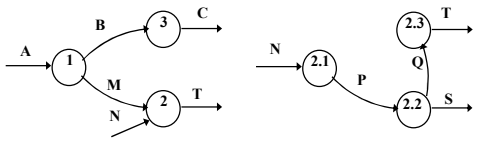
（2）**把握主线忽略枝节**。

3、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
- **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

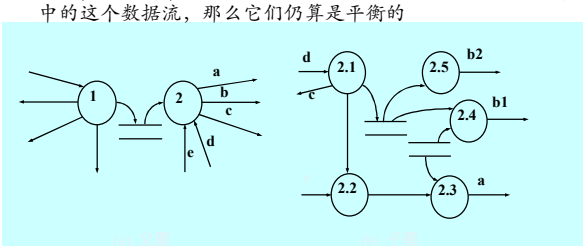
父图与子图不平衡的实例

- 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

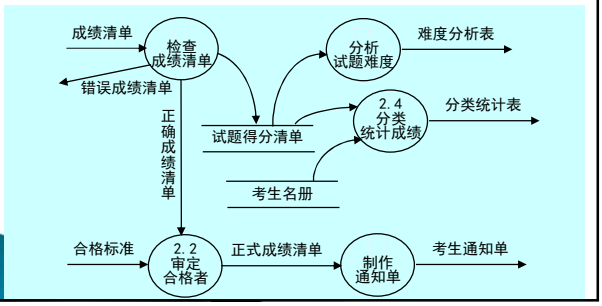
- 注意：如果父图某加工的一个数据流，对应于子图中几个数据流，而子图中组成这些数据流的数据项全体正好等于父图中的这个数据流，那么它们仍算是平衡的



a：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、源和宿都应被适当地命名，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

- 数据流图的用途

1、利用数据流图作为交流的工具。

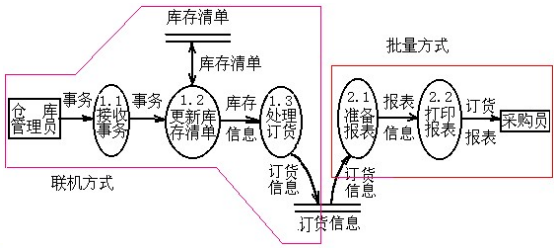
· 审查开发人员对现有系统的认识和对新系统的设想。

· 使用户了解开发人员对目标系统的设想。

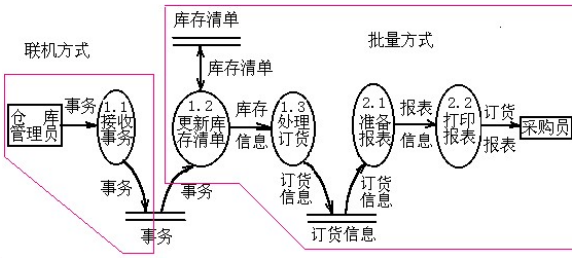
2、利用数据流图作为分析和设计的工具。

3、进一步设计的依据。

例如，在数据流图上画出多组自动化边界，来考虑系统不同的物理实现。



方案二



4、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、文件、加工、源或宿

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

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（存储）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

5、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

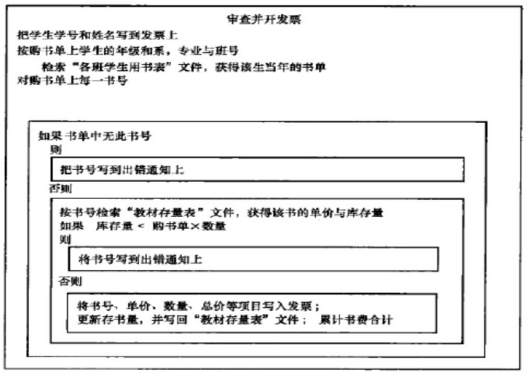
加工逻辑的描述方法

- **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**

审查并开发票的加工逻辑：



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

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			√	

“审批发货单”加工的判定表

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		√	

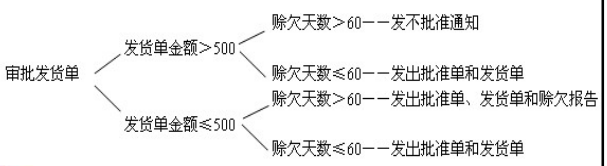
“审批发货单”加工的简化判定表

发货单金额≤500	0	0	1	1
发货单金额>500	1	1	0	0
赊欠天数≤60	0	1	0	1
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

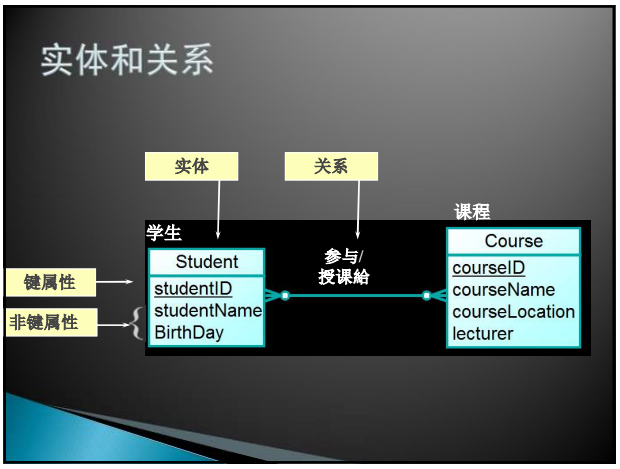
判定树

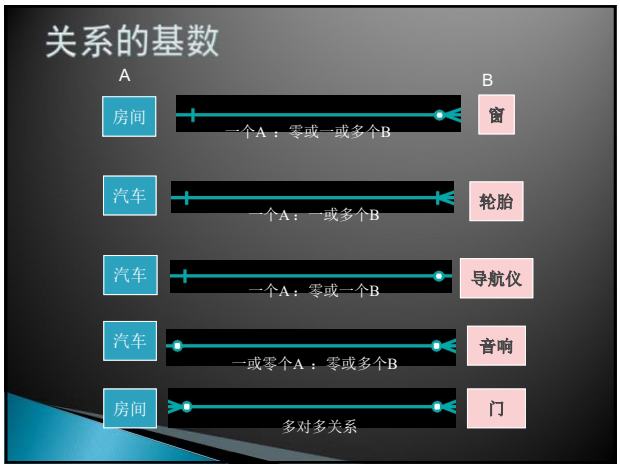
- **本质上与判定表是相同的，只是表示形式不同**
- **例如“审批发货单”加工逻辑的判定树描述入下：**

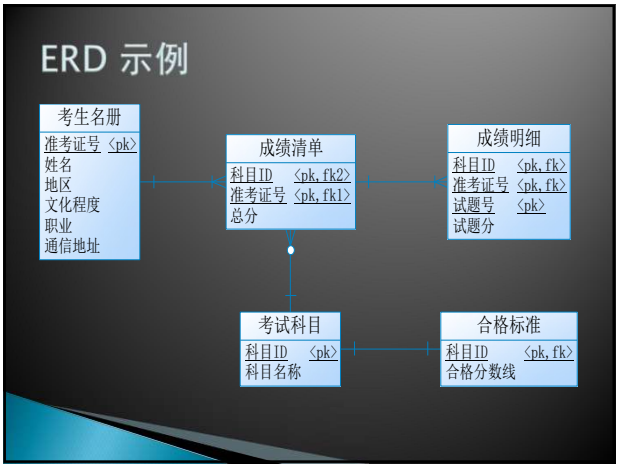


6、实体—关系图

- **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

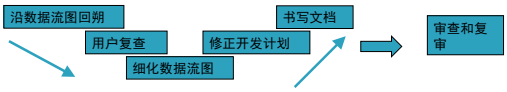






7 面向数据流的需求分析过程

- 结构化分析方法就是面向数据流自顶向下逐步求精进行需求分析的方法。
- 可行性研究已经得出了目标系统的高层数据流图，需求分析的目标之一就是把数据流和数据存储定义到元素级。



- 从数据流图的输出端着手分析，系统的基本功能是产生这些输出，输出数据决定了系统必须具有的最基本的组成元素。

需求分析的任务：

基本任务是准确地回答“系统必须做什么?”这个问题。对目标系统提出完整、准确、清晰、具体的要求。

1、 确定系统的综合需求

功能需求、性能需求、可靠性和可用性需求、出错处理需求、接口需求、约束、逆向需求、将来可能的需求。

2、分析系统的数据要求

分析系统数据要求通常采用建立数据模型的方法，实体关系ER(Entity Relation)模型。

3、导出系统的逻辑模型

- 综合上述分析的结果可以导出系统的详细的逻辑模型，对系统需求表达主要内容。
- 结构化分析方法（Structured Analysis）：数据流图、数据字典、处理说明等描述系统的逻辑模型。

4、 修正系统开发计划
5、 开发原型系统
6、 写需求规格说明书

内容: 信息描述
结构化分析方法（SA）：
ＤＦＤ数据流图
ＤＤ数据字典
ＥＲ实体－联系模型
处理说明ＩＰＯ
功能描述
……

作业:

为方便储户，某银行拟开发计算机储蓄系统。储户填写的存款单或取款单由业务员键入系统，如果是存款，系统记录存款人姓名、住址、存款类型、存款日期、利率等信息，并印出存款单给储户；如果是取款，系统计算利息并印出利息清单给储户。

1) 请画出数据流图（DFD);

2) 定义数据字典（DD）存款单,利息清单等5个;