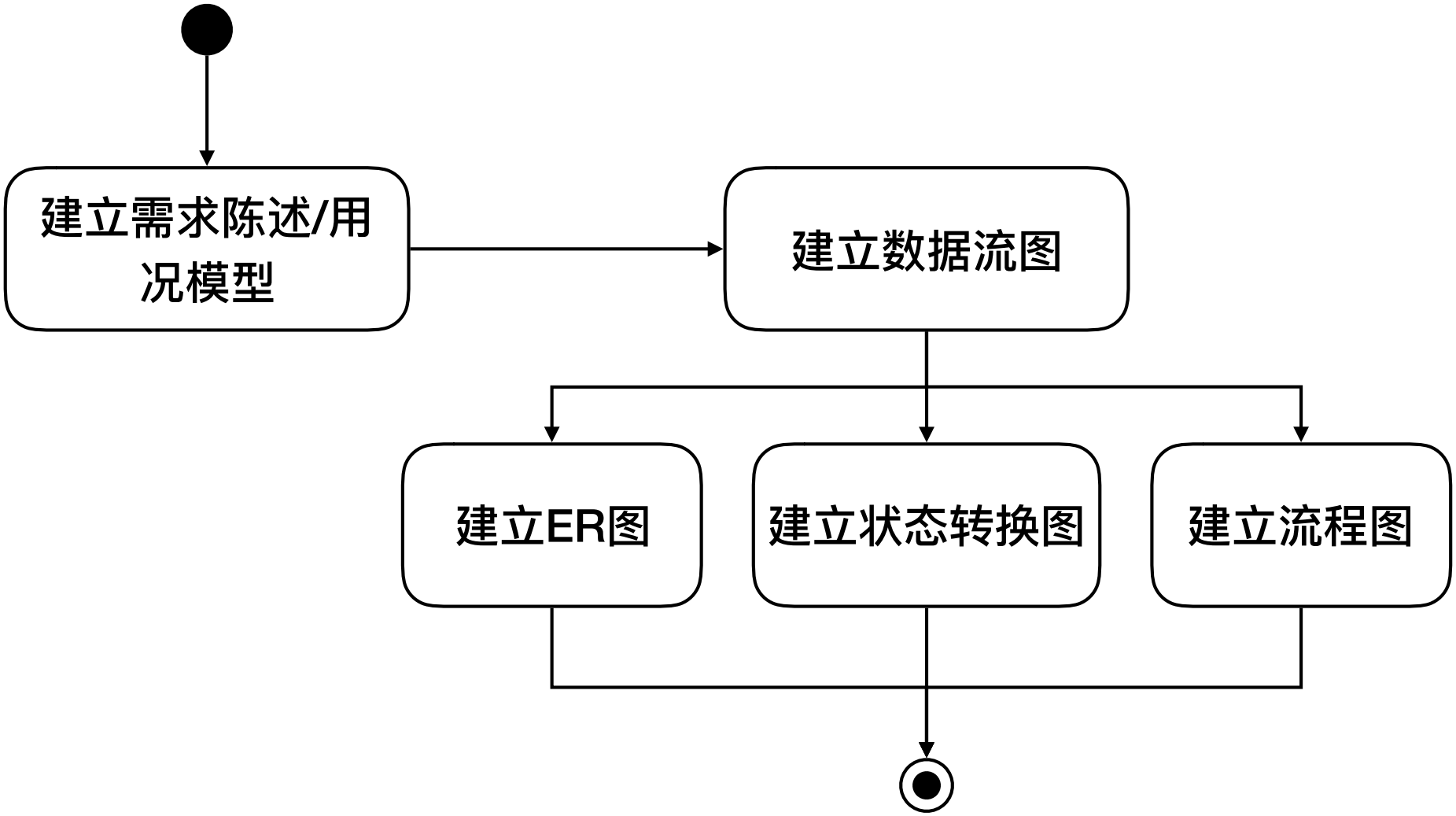
北京科技大学
University of Science and Technology Beijing

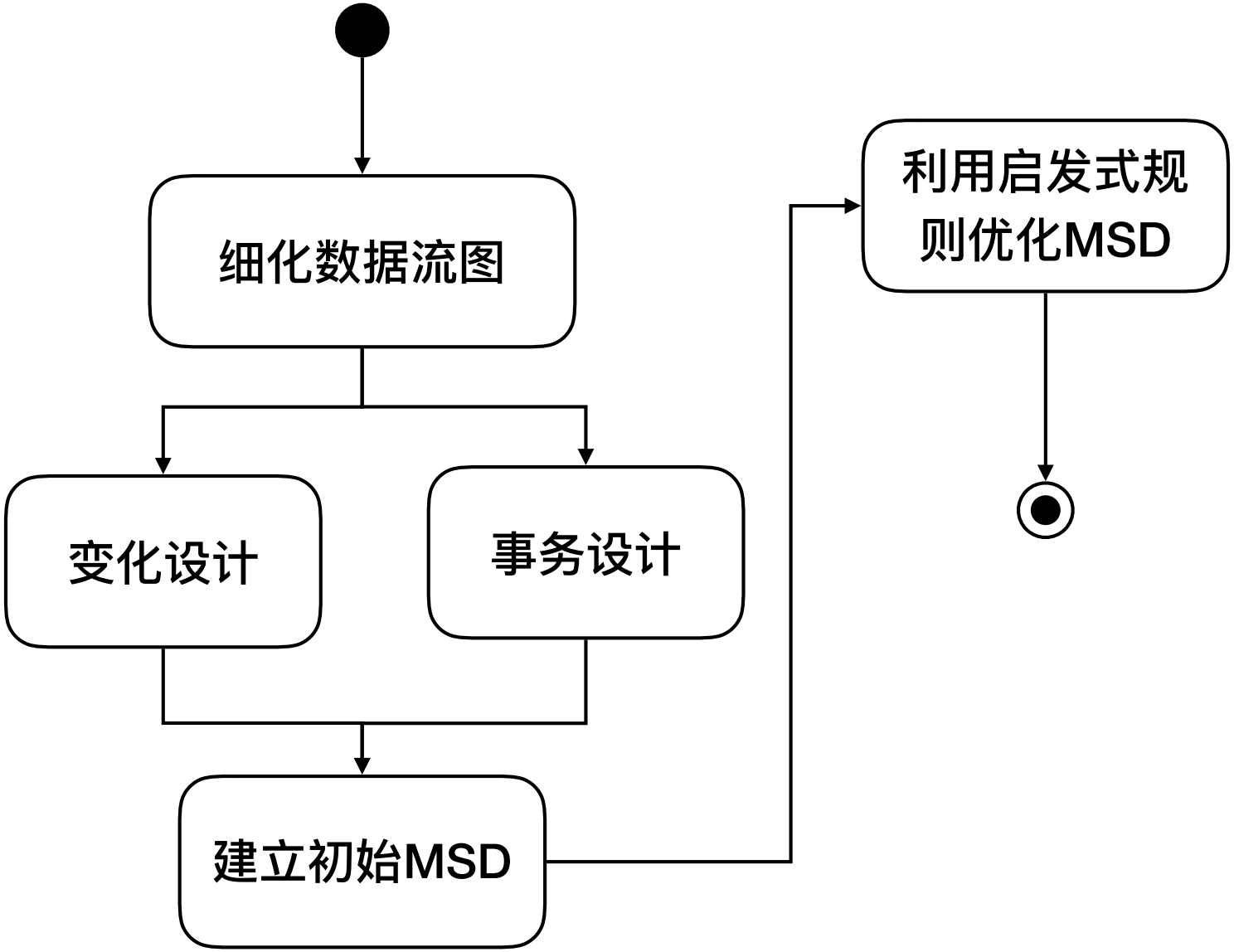
软件工程

结构化分析和设计方法

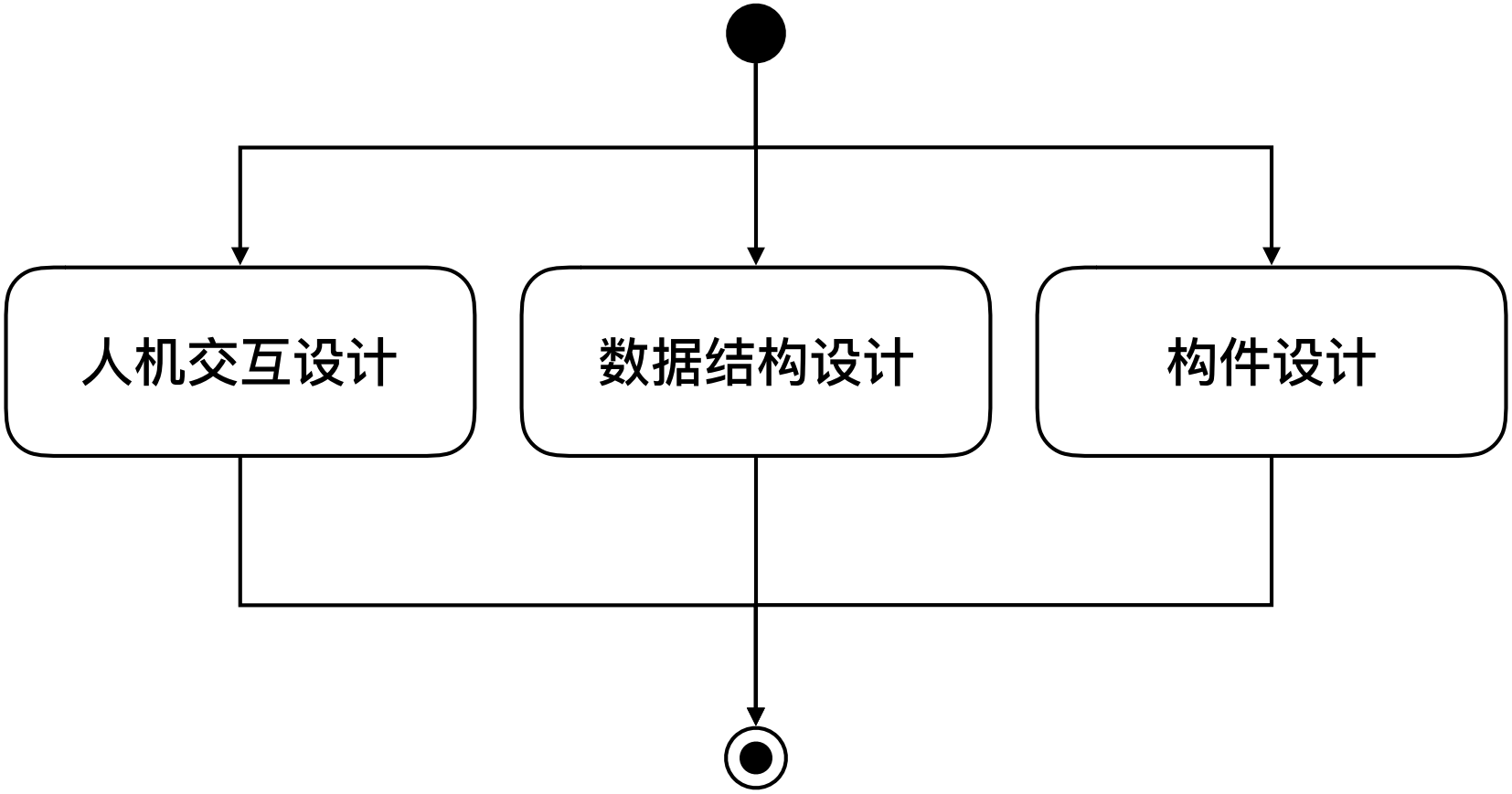
结构化分析过程



结构化总体设计过程



结构化详细设计过程



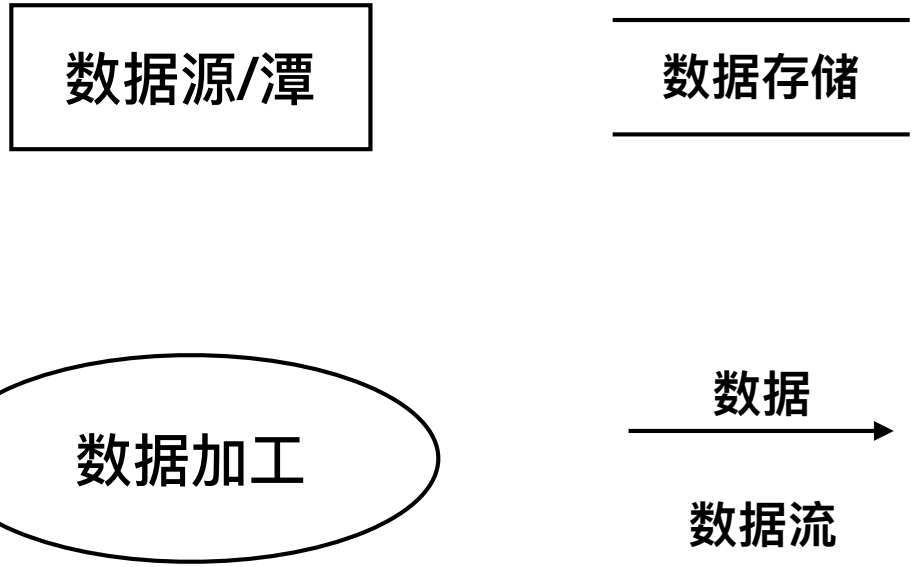
结构化分析和设计工具

数据流图

- 数据流图
  - 结构化方法中一种非常重要的模型
  - 描绘信息流和数据从输入移动到输出的过程中所经受的变换
  - 贯穿分析和设计
- 基本概念
  - 数据加工：系统中对数据的加工处理，可以看作是一个函数，将输入变为输出
  - 数据流：刻画系统中数据的流向
  - 数据源、数据潭：刻画系统边界，是数据的发起和终结
  - 数据存储：系统内部存储数据的部分

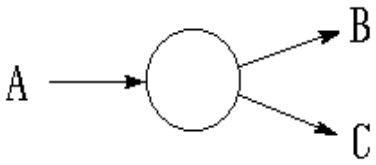
北京科技大学
University of Science and Technology Beijing

数据流图

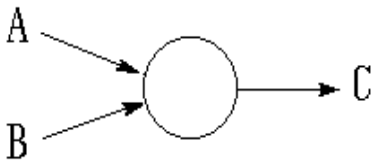


数据流图不是流程图！！

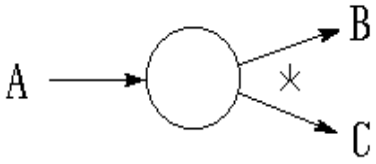
数据流图



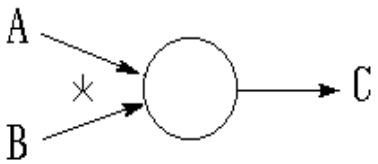
有A则有B或C，
或两者都有



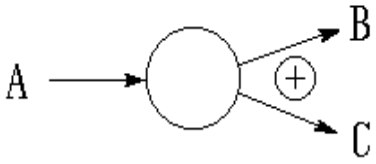
当A或B有一个
存在，就有C



有A则有B与C，
两者同时有

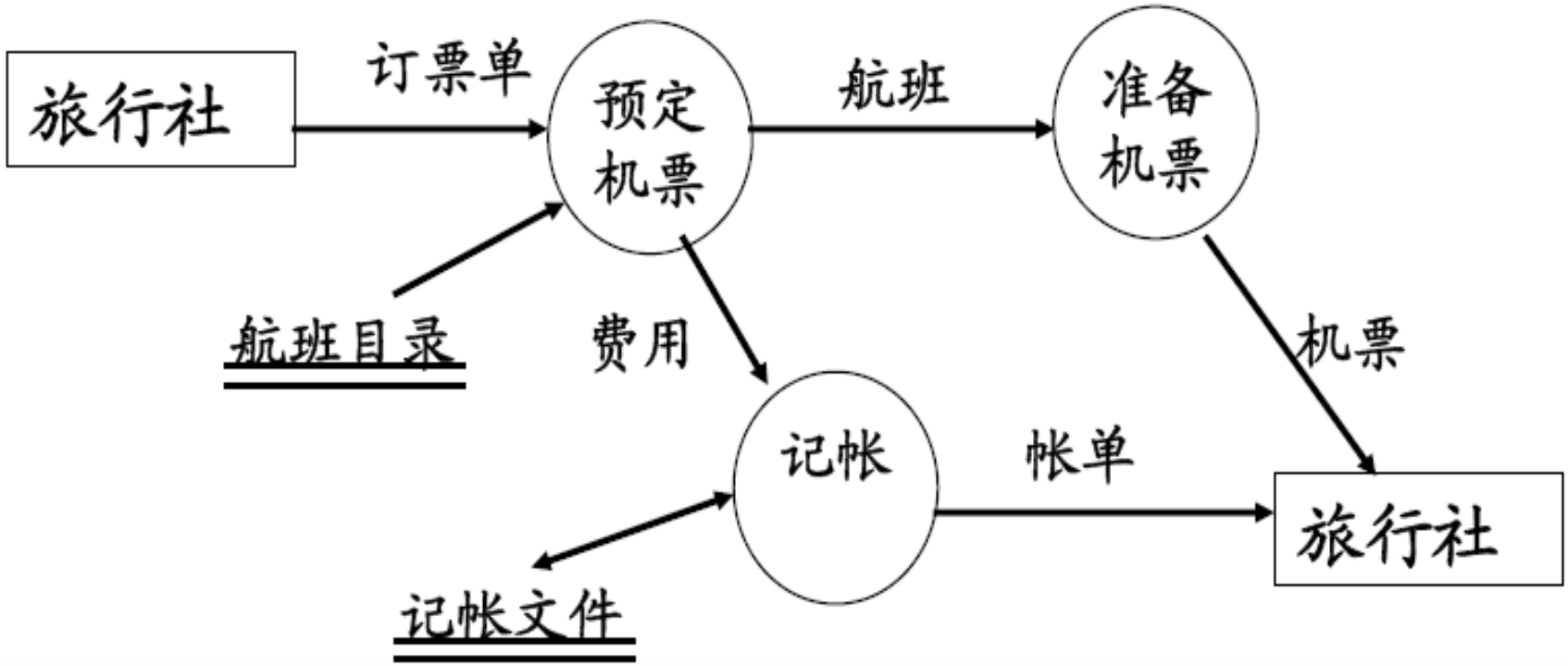


当A与B都存在，
就有C



有A则有B或C，但
不会同时有B与C

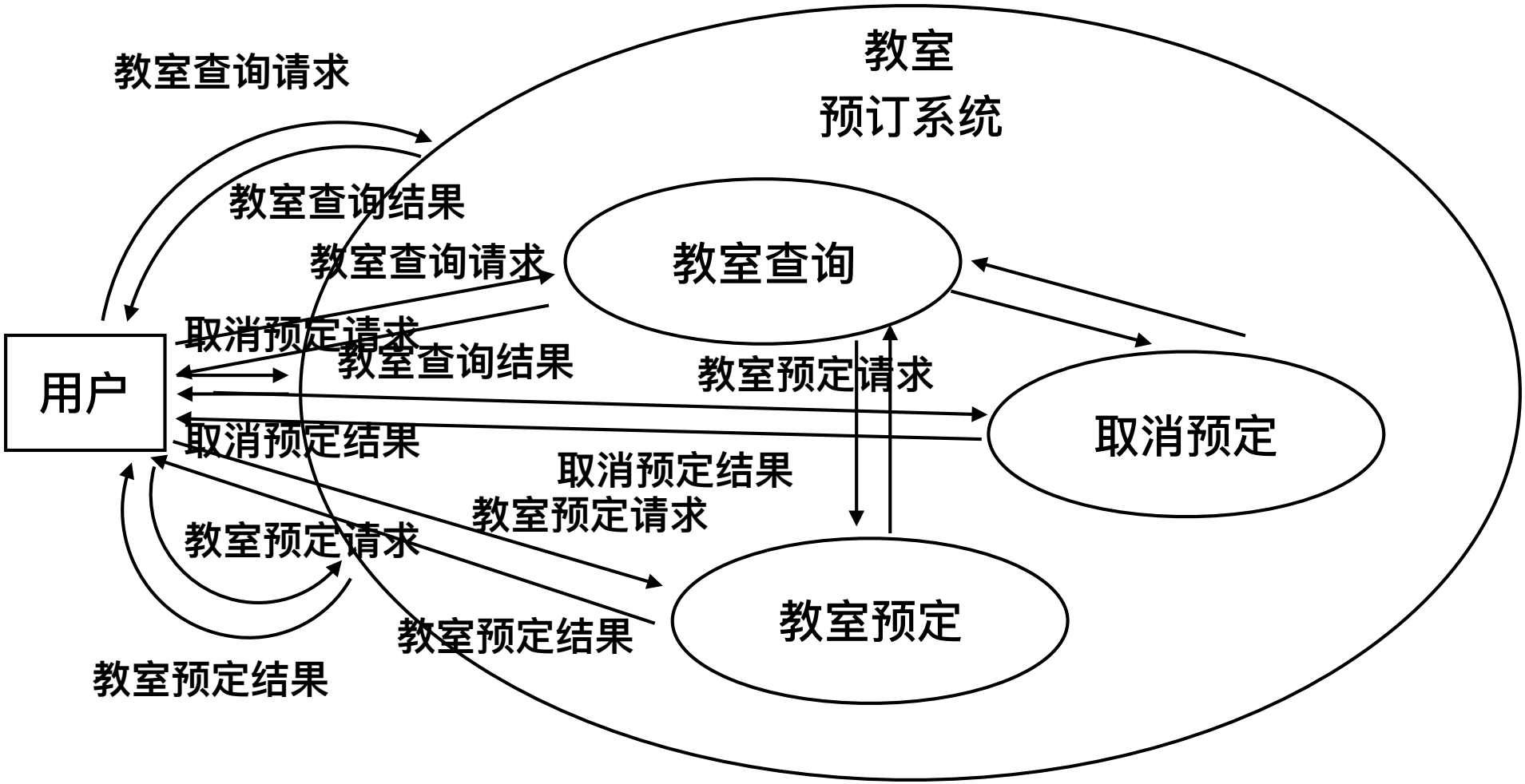
数据流图



数据流图

- 建模过程
  - 定义原始数据加工，识别数据源和数据潭，建立数据流
  - 分解数据加工，识别数据存储，重新分配数据流
  - 重复第二步，自顶向下逐步求精

数据流图


教室查询请求
教室
预订系统
教室查询结果
教室查询请求
教室查询
取消预定请求
用户
教室查询结果
教室预定请求
取消预定
取消预定结果
取消预定结果
教室预定请求
教室预定请求
教室预定
教室预定结果
教室预定结果

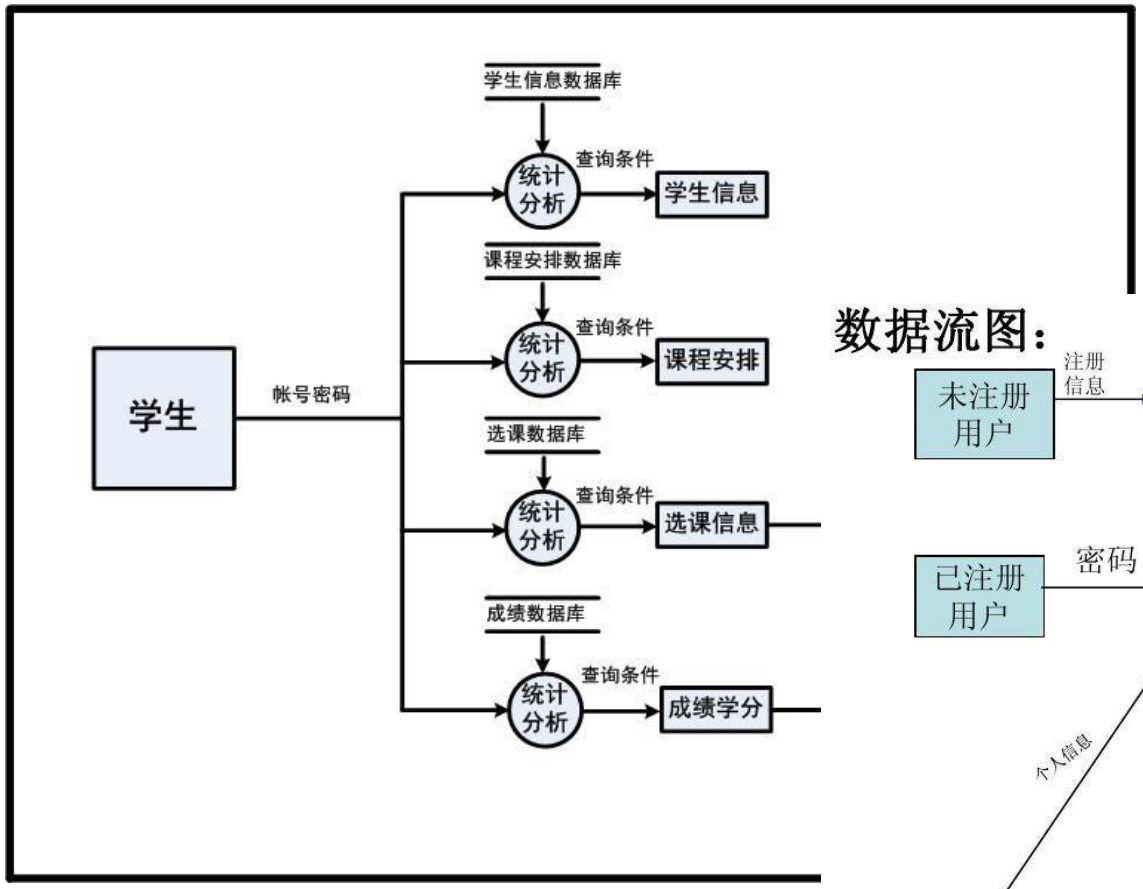
数据流图

- 细化数据流图的注意事项
  - 把一个数据加工P分解后得到新的数据加工{P1,P2,…}，连接在P上的数据流f必须也连接在某个Pi上
    - 除非f也被分解了
  - 数据流f分解成{f1,f2,…}，则f上传递的信息必须等价于f1，f2,…上的信息，即f=f1+f2+…
    - 如果分解前图中没有的数据和信息，分解后也不能有

数据流图

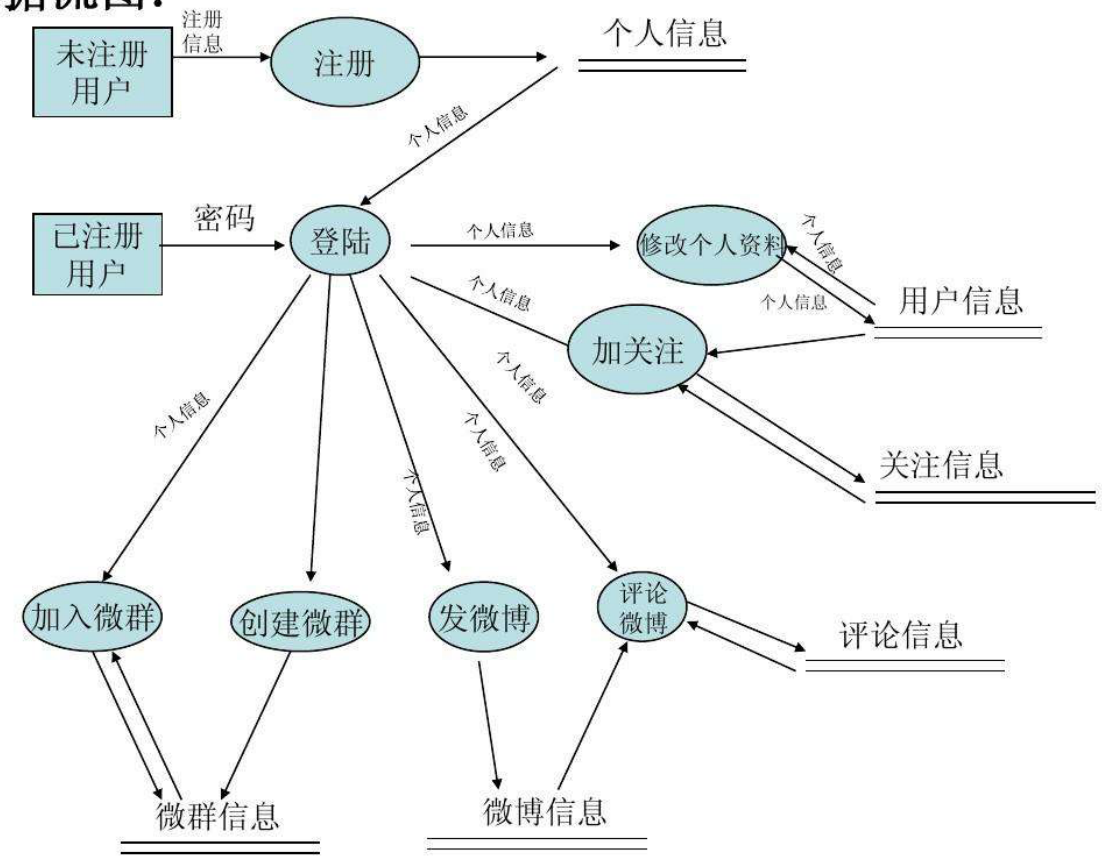
- 注意事项
  - 数据流图不是流程图，数据加工之间不存在控制先后关系
  - 前继数据流所包含的信息应该能够导出后继数据流上的信息
  - 分解数据加工时，要保证数据流分解的一致性

数据流图



不好的数据流图！

数据流图：



数据流图

- 数据流图中的所有数据都必须进一步定义
  - 数据词典
    - 一种使用类似于正则表达式的方法定义数据结构的方法
      - 例如:
        - 订票请求=车次+等级+时间日期+数量
        - 登录请求=用户名+密码
  - E-R图

数据词典

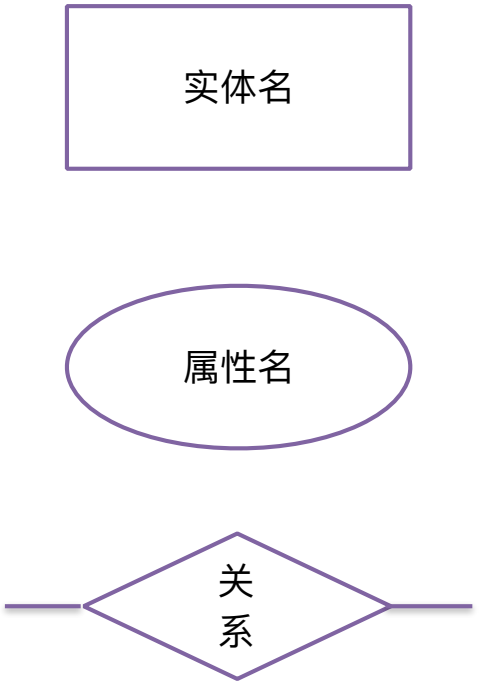
- 数据词典包含的内容
  - 数据项描述={数据项名，数据项含义说明，别名，数据类型，长度，取值范围，取值含义，与其他数据项的逻辑关系}
  - 数据结构描述={数据结构名，含义说明，组成:{数据项或数据结构}}
  - 数据流描述={数据流名，说明，数据流来源，数据流去向，组成:{数据结构}，平均流量，高峰期流量}
  - 数据存储描述={数据存储名，说明，编号，流入的数据流，流出的数据流，组成:{数据结构}，数据量，存取方式}
  - 处理过程描述={处理过程名，说明，输入:{数据流}，输出:{数据流}，处理:{简要说明}}

E-R图

- E-R图
  - 一种数据建模的方法
  - 采用“实体”、“属性”和“关系”作为最基本的概念
    - 现在已经被类图替代

E-R图

- E-R图
  - 实体——表示一个数据对象
  - 属性——刻画实体中包含的数据项
  - 关系——表示实体之间的关系



E-R 图

- 数据对象
  - 是需要被目标系统所理解的复合信息的表示。它具有若干不同特征或属性的信息。
  - 数据对象可以是外部实体,事物, 角色,行为或事件, 组织单位, 地点或结构。
  - 数据对象只封装了数据，没有包含作用于这些数据上的操作。
    - 包含操作的数据对象是什么?

E-R 图

- 属性
  - 定义了数据对象的特征。它可用来：
    - 为数据对象的实例命名；
    - 描述这个实例；
    - 建立对另一个数据对象的另一个实例的引用。
  - 为了唯一地标识数据对象的某一个实例，定义数据对象中的一个属性或几个属性为关键码 (key)
    - 例如在“学生”数据对象中用“学号”做关键码，它可唯一地标识一个“学生”数据对象中的实例。

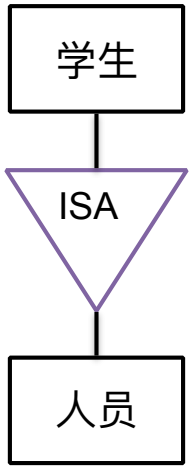
E-R 图

- 关系
  - 各个数据对象的实例之间有关联。
    - 如一个学生“张鹏”选修两门课程“软件工程”与“计算机网络”，学生与课程的实例通过“选修”关联起来。
  - 关联有三种：一对一 (1:1)； 一对多(1:m)；多对多(n:m)
    - 基数：基数表明了“重复性”。如 1 位教师带学生班的 30 位同学，就是 1:m 的关系。
    - 可选和必选：也有 1 位教师带 0 位同学的情形，所以实例关联有是“可选”还是“必须”之分。基数下限为0时，表示可选，默认为必选。

E-R 图

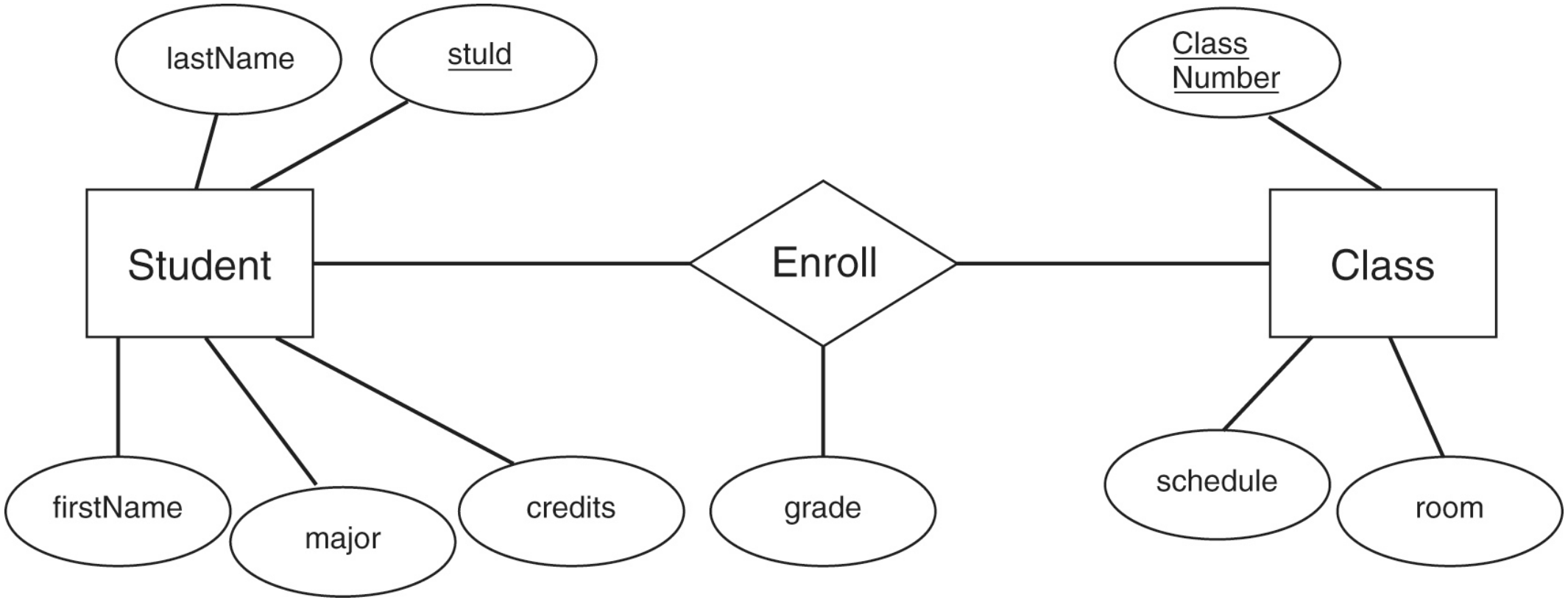
- 关系

- 一般-特殊关系

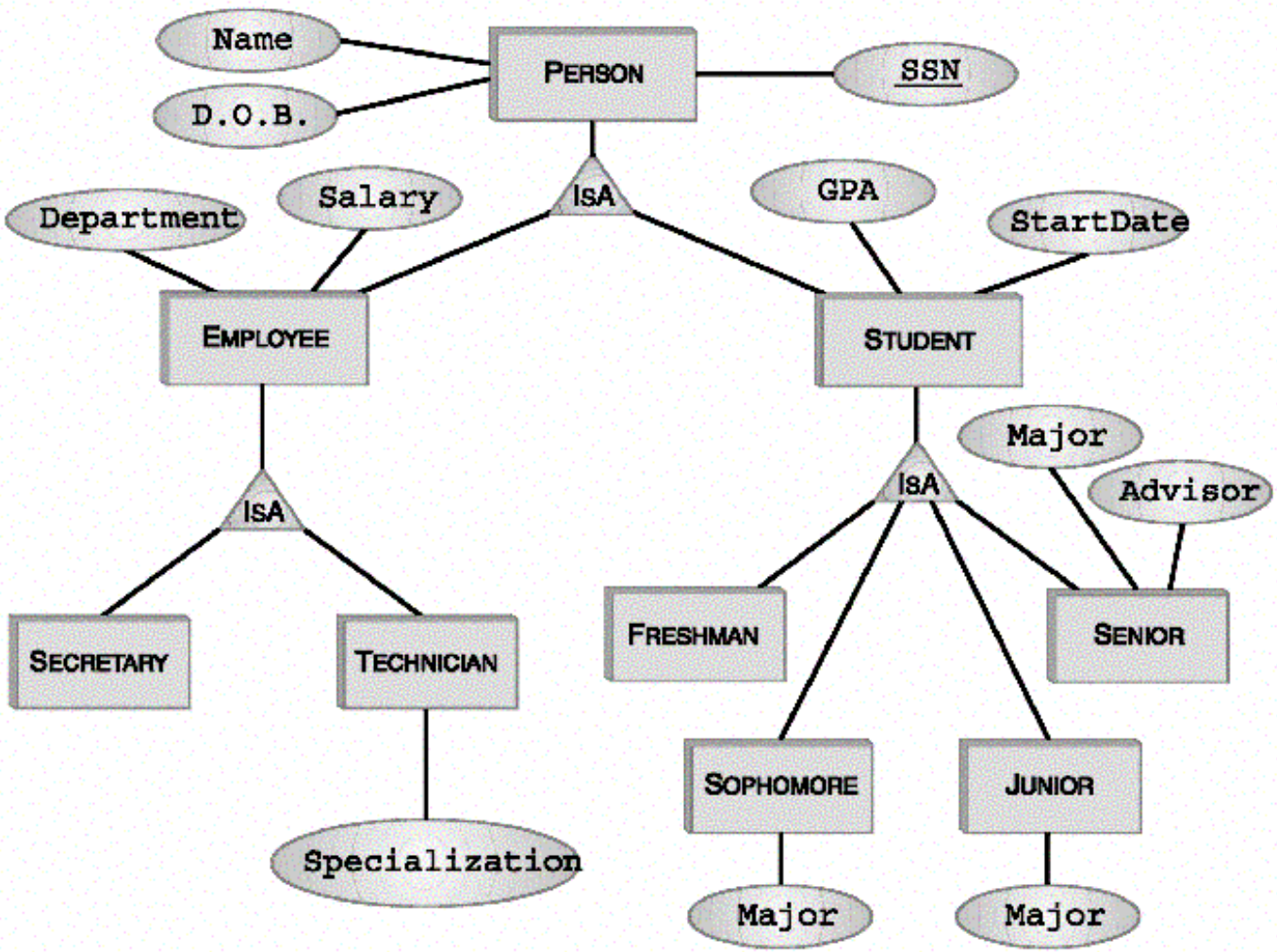


《数据库系统概念》

E-R 图


lastName
stuld
Class Number
Student
Enroll
Class
firstName
major
credits
grade
schedule
room

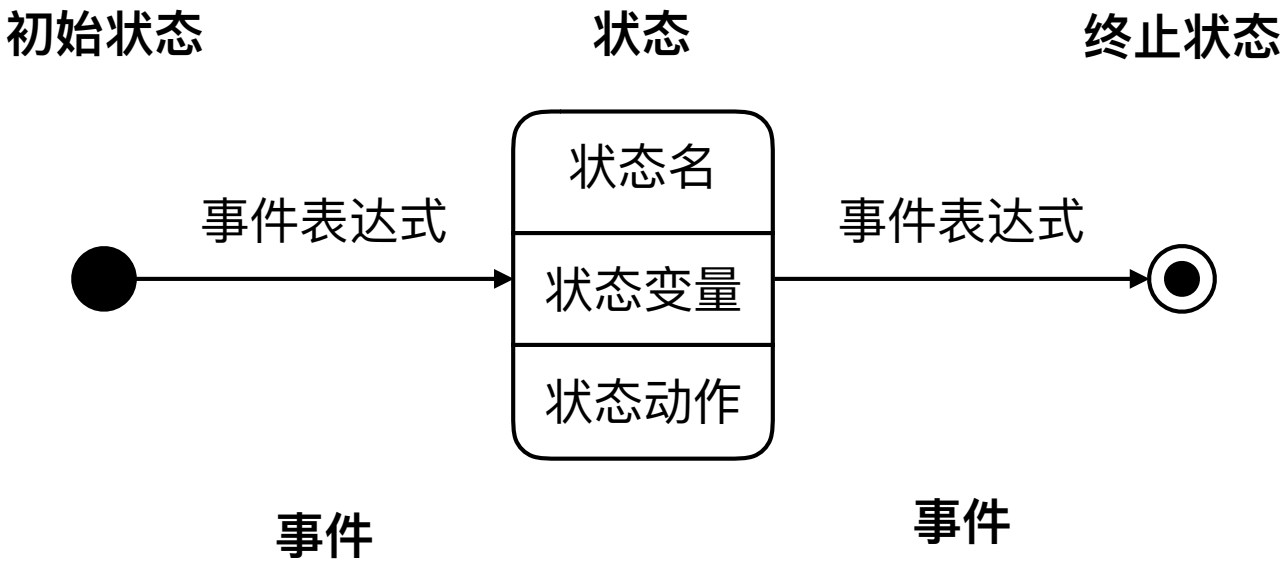
E-R图


Name
D.O.B.
PERSON
SSN
ISA
Department
Salary
GPA
StartDate
EMPLOYEE
STUDENT
ISA
Major
Advisor
ISA
SECRETARY
TECHNICIAN
FRESHMAN
SENIOR
Specialization
SOPHOMORE
JUNIOR
Major
Major

状态转换图

- 状态转换图
  - 通过描述系统的状态以及引起系统状态转换的事件来刻画系统行为的工具
  - 核心元素
    - 状态——可以理解为系统运行过程中所处的一个阶段
    - 事件（状态转移）——导致系统状态发生转换的事件

状态转换图

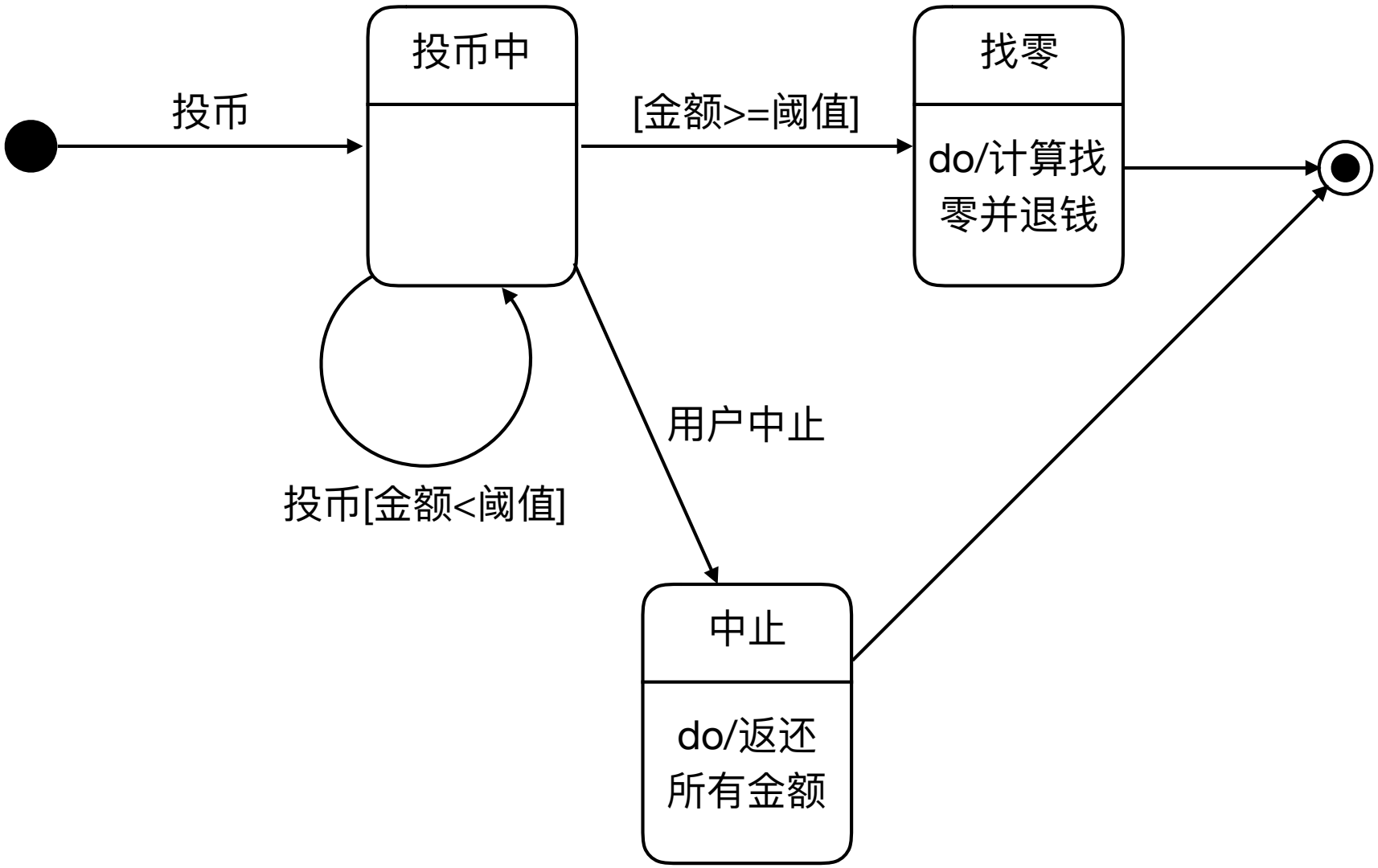


状态变量: (*var* '=' *expression*)*

状态动作: (('entry' | 'do' | 'exit') **'/'** *action*)*

事件表达式: *name* '[' *guard condition* ']' '/' *action*

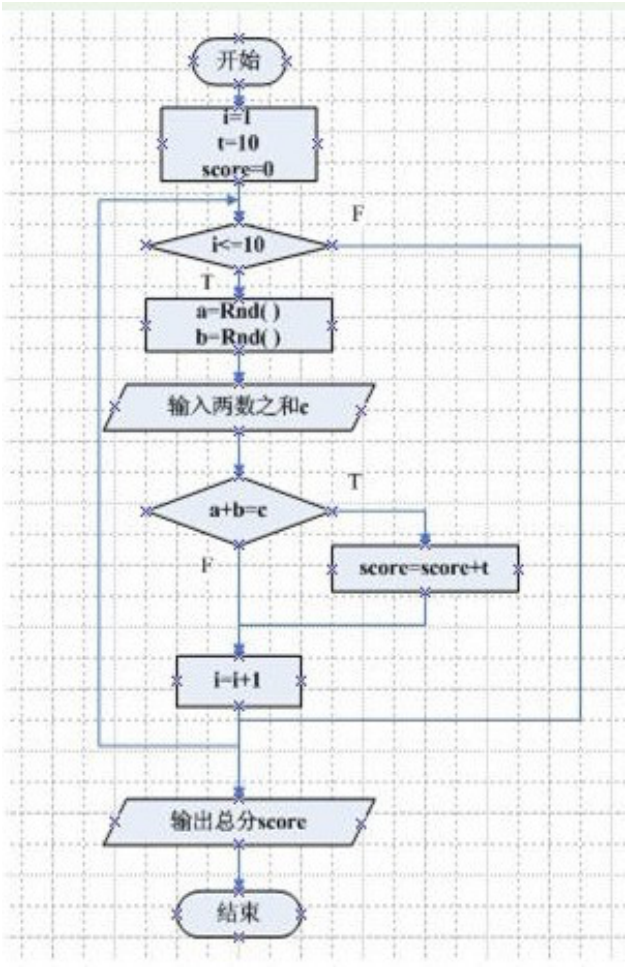
状态转换图



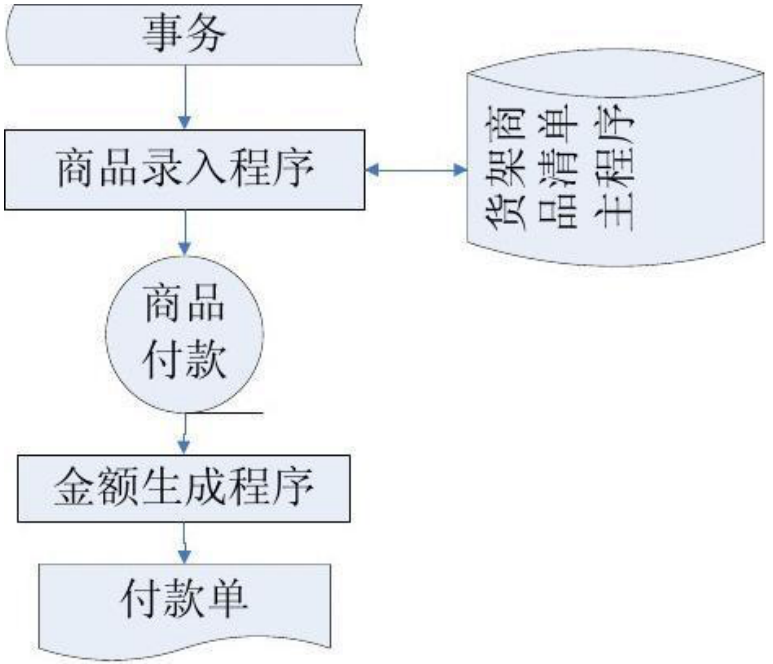
流程图

- 流程图
  - 一种刻画业务流程和系统流程的工具（被活动图替代）
    - 程序流程图——描述控制流（常用于详细设计）
    - 系统流程图——描述数据流（常用于需求分析）

流程图



程序流程图



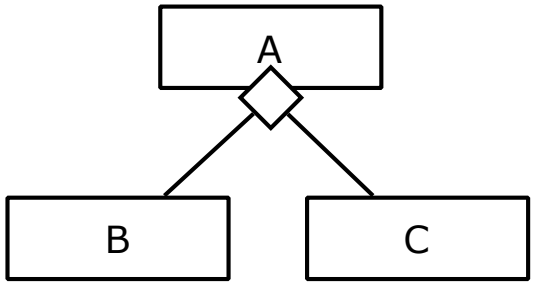
系统流程图

模块结构图

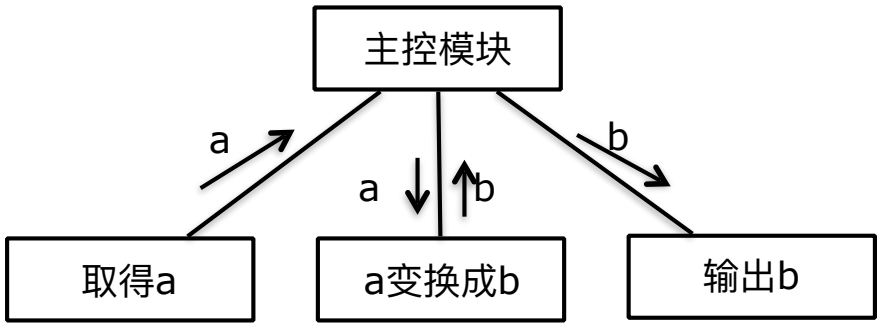
- 模块结构图（Module Structure Diagram，MSD）
  - 一种用于软件总体设计（软件体系结构设计）的工具，用来刻画软件系统的结构
  - 核心元素
    - 模块——一种可独立标识的软件成分
    - 调用关系——模块间的一种关系，模块A为了完成其任务必须依赖其他模块

模块结构图

- 符号
  - 模块
  - 调用



表示根据一个判定来调用B或C



模块结构图

面向数据流的设计

- 目标
  - 根据数据流图产生模块结构图
  - 核心技术
    - 事务设计——将事务型数据流图转化为模块结构图
    - 变换设计——将变换型数据流图转化为模块结构图

结构化分析和设计的特点

- 以数据处理、模块、实体等计算机概念为核心对系统进行分析和设计，便于开发人员理解
- 在不同的阶段采用不同的工具和符号体系建立相应的模型，因此这些符号需要进行相应的映射

自学内容

- 数据流图的建立方法（读书+作业+辅导文档）
- 数据字典的详细定义（读书+作业）
- 状态转换图的用法（读书）
- 将数据流图转化成模块结构图的方法（读书+作业+辅导文档）
- 详细设计的相关工具（读书+作业）

北京科技大学
University of Science and Technology Beijing

启发式设计规则

启发式规则

- 1. 改进软件结构提高模块独立性
  - 设计出软件的初步结构以后，应该审查分析这个结构，通过模块分解或合并，力求降低耦合提高内聚。
  - 例如，多个模块公有的一个子功能可以独立成一个模块，由这些模块调用；有时可以通过分解或合并模块以减少控制信息的传递及对全程数据的引用，并且降低接口的复杂程度。

启发式规则

- 2. 模块规模应该适中
  - 经验表明，一个模块的规模不应过大，最好能写在一页纸内（通常不超过60行语句）。有人从心理学角度研究得知，当一个模块包含的语句数超过30以后，模块的可理解程度迅速下降
    - 过大的模块往往是由于分解不充分，但是进一步分解必须符合问题结构，一般说来，分解后不应该降低模块独立性
    - 过小的模块开销大于有效操作，而且模块数目过多将使系统接口复杂

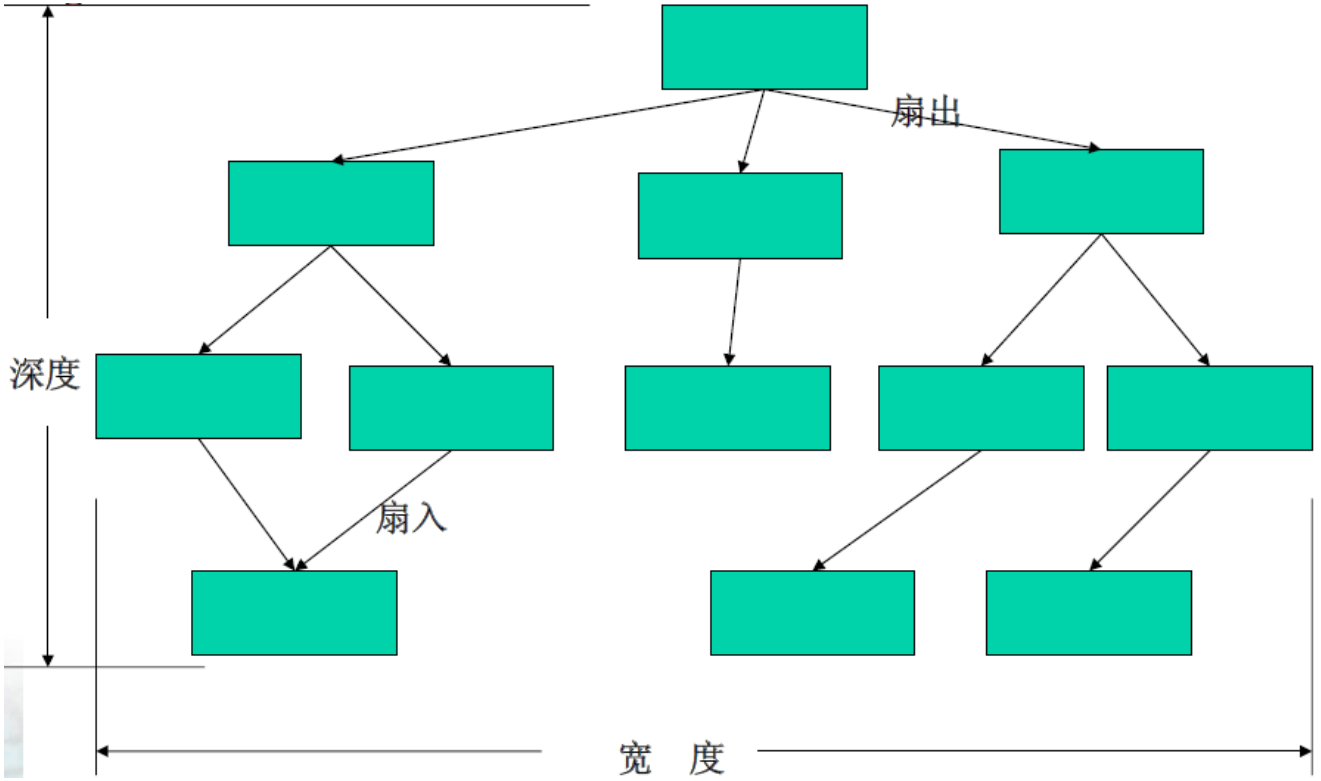
启发式规则

- 3. 深度、宽度、扇出和扇入都应适当
  - 深度表示软件结构中控制的层数，它往往能粗略地标志一个系统的大小和复杂程度
    - 深度和程序长度之间应该有粗略的对应关系，当然这个对应关系是在一定范围内变化的。如果层数过多则应该考虑是否有许多管理模块过分简单了，能否适当合并。
  - 宽度是软件结构内同一个层次上的模块总数的最大值
    - 一般说来，宽度越大系统越复杂。对宽度影响最大的因素是模块的扇出。

启发式规则

- 4. 深度、宽度、扇出和扇入都应适当
  - 扇出是一个模块直接控制(调用)的模块数目
    - 扇出过大意味着模块过分复杂，需要控制和协调过多的下级模块；扇出过小(例如总是1)也不好。
      - 经验表明，一个设计得好的典型系统的平均扇出通常是3或4(扇出的上限通常是5～9)。
  - 扇入表明一个模块的有多少个上级模块直接调用它
    - 扇入越大则共享该模块的上级模块数目越多，这是有好处的，但是，不能违背模块独立原理单纯追求高扇入。

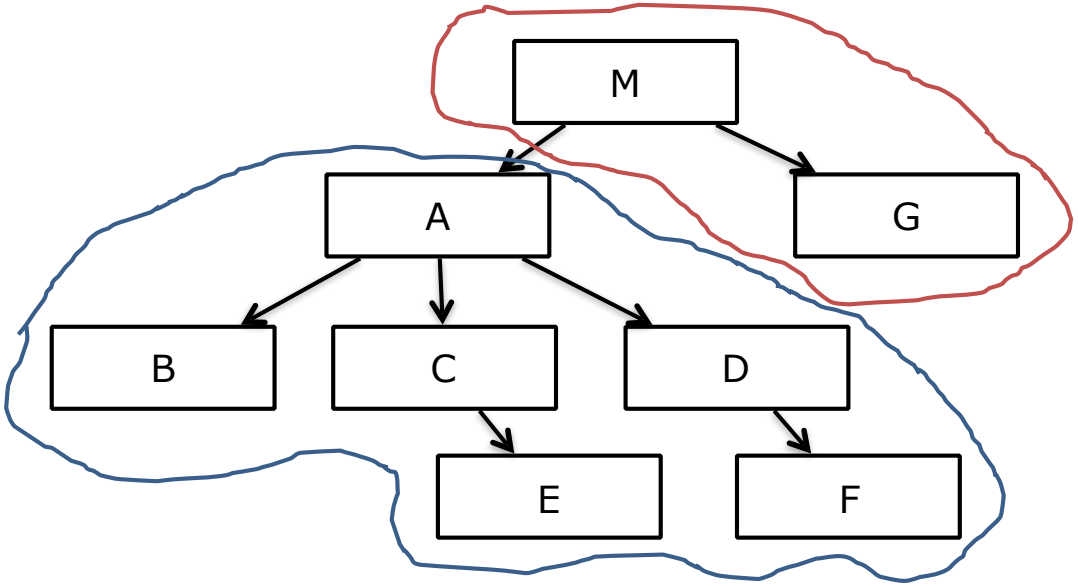
启发式规则



好的设计：顶层扇出比较高，中层扇出较少，底层扇入到公共的实用模块中去(底层模块有高扇入)。

启发式规则

- 5. 模块的作用域应该在控制域之内
  - 作用域定义为受该模块内一个判定影响的所有模块的集合。
  - 控制域是这个模块本身以及所有直接或间接从属于它的模块的集合。



启发式规则

- 6. 力争降低模块接口的复杂程度
  - 模块接口复杂是软件发生错误的一个主要原因。应该仔细设计模块接口，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(即看起来传递的数据之间没有联系)，是紧耦合或低内聚的征兆，应该重新分析这个模块的独立性。
- 7. 设计单入口单出口的模块
  - 这条启发式规则警告软件工程师不要使模块间出现内容耦合。当从顶部进入模块并且从底部退出来时，软件是比较容易理解的，因此也是比较容易维护的。

启发式规则

- 8. 模块功能应该可以预测
  - 可预测性：只要输入的数据相同就产生同样的输出
    - 带有内部状态的模块功能可能是不可预测的，因为它的输出可能取决于这个内部状态（对外不可见）。这样的模块既不易理解又难于测试和维护。