第三章

软件需求分析

什么是需求分析

- 需求分析是软件定义时期的最后一个阶段
- 基本任务是准确地回答“系统必须做什么?”
- 需求分析的任务
- 确定系统必须完成哪些工作，即对目标系统提出完整、准确、清晰、具体的要求
  - 还不是确定系统怎样完成它的工作
- 在需求分析阶段结束之前，系统分析员应该写出软件需求规格说明书，以书面形式准确地描述软件需求

- 在分析软件需求和书写软件需求规格说明书的过程中，分析员和用户都起着关键的、必不可少的作用
  - 只有用户才真正知道自己需要什么，但是他们并不知道怎样用软件实现自己的需求，用户必须把他们对软件的需求尽量准确、具体地描述出来
  - 分析员知道怎样用软件实现人们的需求，但是在需求分析开始时他们对用户的需求并不十分清楚，必须通过与用户沟通获取用户对软件的需求

需求分析的要求

- 需求分析和规格说明是一项十分艰巨复杂的工作
  - 用户与分析员之间需要沟通的内容非常多
    - 在双方交流信息的过程中很容易出现误解或遗漏，也可能存在二义性
- 整个需求分析过程中
  - 要采用行之有效的通信技术
  - 要保证开展集中、细致工作过程
  - 必须严格审查验证需求分析的结果

需求分析的工作准则

- 四个“必须”

(1)必须理解并描述问题的信息域，

- 应该建立数据模型

(2)必须定义软件应完成的功能

- 建立功能模型

(3)必须描述作为外部事件结果的软件行为

- 建立行为模型

(4)必须对描述信息、功能和行为的模型进行分解，用层次的方式展示细节

需求分析的任务

1.确定对系统的综合要求

- 功能需求
  - 指定系统必须提供的服务
  - 通过需求分析应该划分出系统必须完成的所有功能
- 性能需求
  - 指定系统必须满足的定时约束或容量约束，通常包括
    - 速度(响应时间)、信息量速率、主存容量、磁盘容量、安全性等方面的需求
- 可靠性和可用性需求
  - 定量地指定系统的可靠性
  - 可用性与可靠性密切相关，量化了用户可以使用系统的程度
- 出错处理需求
  - 说明系统对环境错误应该怎样响应
    - 例如，如果它接收到从另一个系统发来的违反协议格式的消息，应该做什么?
    - 注意，上述这类错误并不是由该应用系统本身造成的
    - 在某些情况下，“出错处理”指的是当应用系统发现它自己犯下一个错误时所采取的行动
      - 应该有选择地提出这类出错处理需求
        - 软件开发的目的是开发出正确的系统，而不是用无休止的出错处理代码掩盖自己的错误
      - 总之，对应用系统本身错误的检测应该仅限于系统的关键部分，而且应该尽可能少

需求分析的任务（CONT.）

1.确定对系统的综合要求（cont.）

- 接口需求
  - 描述应用系统与它的环境通信的格式
    - 常见的接口需求有：用户接口需求；硬件接口需求；软件接口需求；通信接口需求
- 约束
  - 描述在设计或实现应用系统时应遵守的限制条件
  - 在需求分析阶段提出这类需求，并不是要取代设计(或实现)过程，只是说明用户或环境强加给项目的限制条件
    - 常见的约束有：精度；工具和语言约束；设计约束；应该使用的标准；应该使用的硬件平台
- 逆向需求
  - 说明软件系统不应该做什么
  - 理论上有无限多个逆向需求
    - 应该仅选取能澄清真实需求且可消除可能发生的误解的那些逆向需求
- 将来可能提出的要求
  - 应该明确地列出那些虽然不属于当前系统开发范畴，但是据分析将来很可能会提出来的要求
    - 其目的在于，在设计过程中对系统将来可能的扩充和修改预做准备，以便一旦确实需要时能比较容易地进行这种扩充和修改

需求分析的任务（CONT.）

2.分析系统的数据要求

- 任何一个软件系统本质上都是信息处理系统，系统必须处理的信息和系统应该产生的信息是软件设计的核心任务
  - 将分析系统的数据要求作为软件需求分析的一个重要任务
  - 分析系统的数据要求通常采用建立数据模型的方法
- 分析系统的数据要求，要注意
  - 正确、高效地对复杂数据进行建模和表达
    - 复杂的数据由许多基本的数据元素组成，数据结构表示数据元素之间的逻辑关系
    - 利用数据字典可以全面准确地定义数据，但是数据字典的缺点是不够形象直观
    - 可利用图形工具辅助描绘数据结构，提高建模的可理解性
      - 常用的图形工具有层次方框图和Warnier图
  - 对数据结构进行规范化，提高可用性、实用性和适用性
    - 软件系统经常使用各种长期保存的信息，这些信息通常以一定方式组织并存储在数据库或文件中，为减少数据冗余，避免出现插入异常或删除异常，简化修改数据的过程，通常需要把数据结构规范化

需求分析的任务（CONT.）

3.导出系统的逻辑模型

- 综合之前两项分析的结果（系统的综合要求、系统的数据要求）可以导出
  - 系统的详细的逻辑模型，通常用数据流图、实体-联系图、状态转换图、数据字典和主要的处理算法描述

4.修正系统开发计划

- 根据在分析过程中获得的对系统的更深入更具体的了解，可以比较准确地估计系统的成本和进度，修正以前制定的开发计划

与用户沟通获取需求的方法

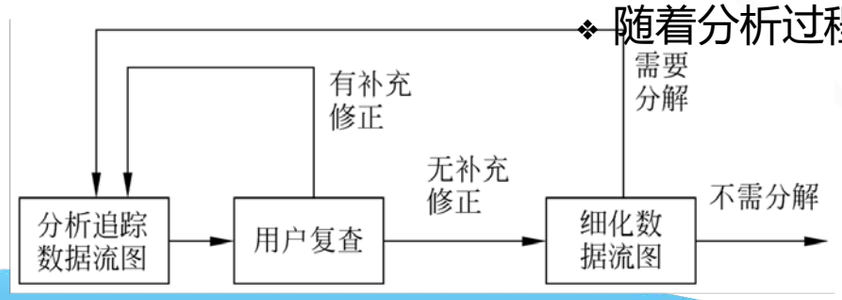
- 访谈
  - 是最早开始使用的获取用户需求的技术，也是迄今为止仍然广泛使用的需求分析技术
  - 访谈的两种基本形式
    - 正式访谈
      - 正式访谈时，系统分析员将提出一些事先准备好的具体问题
    - 非正式访谈
      - 分析员将提出一些用户可以自由回答的开放性问题，以鼓励被访问人员说出自己的想法
  - 当需要调查大量人员的意见时，可向被调查人分发调查表
    - 经过仔细考虑写出的书面回答可能比被访者对问题的口头回答更准确
    - 分析员仔细阅读收回的调查表，然后再有针对性地访问一些用户，以便向他们询问在分析调查表时发现的新问题
  - 使用情景分析技术
    - 对用户将来使用目标系统解决某个具体问题的方法和结果进行分析
    - 能在某种程度上演示目标系统的行为，从而便于用户理解，而且还可能进一步揭示出一些分析员目前还不知道的需求
    - 由于情景分析较易为用户所理解，使用这种技术能保证用户在需求分析过程中始终扮演一个积极主动的角色
      - 需求分析的目标是获知用户的真实需求
      - 让信息的唯一来源，即用户，起积极主动的作用对需求分析工作获得成功至关重要

与用户沟通获取需求的方法（CONT.）

- 面向数据流自顶向下细化需求
  - 软件系统本质上是把输入数据转变成需要的输出信息
    - 数据决定了需要的处理和算法，数据是需求分析的出发点
  - 在需求分析阶段要定义在可行性研究阶段被忽略的数据元素
    - 结构化分析方法即是面向数据流自顶向下逐步细化需求的方法
      - 通过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
        - 为了达到此目标，通常从数据流图的输出端开始分析
          - 因为数据输出是系统的基本功能的实际表达，输出数据决定了系统必须具有的最基本的组成元素
          - 需要向用户和其他有关人员请教，从回答中使分析员对目标系统的认识更加深入具体，从而将系统中更多的数据元素划分出来
        - 通常把分析过程中得到的有关数据元素的信息记录在数据字典中，把对算法的简明描述记录在IPO图中
          - 通过分析而补充的数据流、数据存储和处理，应该添加到数据流图的适当位置
        - 数据流图是帮助复查上述分析的有力工具
          - 从输入端开始，分析员借助数据流图、数据字典和IPO图向用户解释输入数据是怎样一步一步地转变成输出数据
            - 上述解释集中反映了通过前面的分析工作分析员所获得的对目标系统的认识
            - 1通过分析员与用户反复交流核实的复查过程可验证已知的元素，补充了未知的元素，填补了文档中的空白

与用户沟通获取需求的方法（CONT.）

- 面向数据流自顶向下细化需求
  - 软件系统本质上是把输入数据转变成需要的输出信息
    - 数据决定了需要的处理和算法，数据是需求分析的出发点
  - 在需求分析阶段要定义在可行性研究阶段被忽略的数据元素
    - 结构化分析方法即是面向数据流自顶向下逐步细化需求的方法
      - 通过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
        - 数据流图是帮助复查上述分析的有力工具
          - 从输入端开始，分析员借助数据流图、数据字典和IPO图向用户解释输入数据是怎样一步一步地转变成输出数据
          - 上述解释集中反映了通过前面的分析工作分析员所获得的对目标系统的认识
          - 通过分析员与用户反复交流核实的复查过程可验证已知的元素，补充了未知的元素，填补了文档中的空白
  - 反复进行上述分析过程，分析员可深入地定义系统中的数据和系统应该完成的功能
    - 为了追踪更详细的数据流，分析员应该把数据流图扩展到更低的层次。通过功能分解可以完成数据流图的细化
  - 对数据流图细化之后得到一组新的数据流图，不同的系统元素之间的关系即可变得更清楚
    - 必要时可在数据字典中增加一些新条目，并且产生新的或精化的算法描述
    - 随着分析过程的进展，经过问题和解答的反复循环，最终得到对系统数据和功能要求的满意了解



与用户沟通获取需求的方法（CONT.）

- 简易的应用规格说明技术
  - 使用传统的访谈或面向数据流自顶向下细化方法定义需求时，交互性较差，用户处于被动地位而且往往有意无意地与开发者区分“彼此”，导致之前两种方法的效果有时并不理想
  - 为了解决上述问题，可采用面向团队的需求收集法，称为简易的应用规格说明技术
  - 该方法提倡用户与开发者密切合作，共同标识问题，提出解决方案要素，商讨不同方案并指定基本需求
    - 目前简易的应用规格说明技术已经成为信息系统领域使用的主流技术

与用户沟通获取需求的方法（CONT.）

- 简易的应用规格说明技术的典型过程

(1)进行初步的访谈，通过用户对基本问题的回答，初步确定待解决的问题的范围和解决方案

(2)开发者和用户分别写出“产品需求”，选定会议的时间和地点，并选举一个负责主持会议的协调人

- 邀请开发者和用户双方组织的代表出席会议，并在开会前预先把写好的产品需求分发给每位与会者
- 要求每位与会者在开会的前几天认真审查产品需求，并且列出作为系统环境组成部分的对象、系统将产生的对象以及系统为了完成自己的功能将使用的对象
- 要求每位与会者列出操作这些对象或与这些对象交互的服务(即处理或功能)
  - 还应该列出约束条件(例如，成本、规模、完成日期)和性能标准(例如，速度、容量)
  - 不要期望每位与会者列出的内容都是毫无遗漏的，但是尽可能追求准确地表达出每个参会者对目标系统的认识

与用户沟通获取需求的方法（CONT.）

- 简易的应用规格说明技术的典型过程

(3)会议开始后，讨论的第一个问题是，是否需要这个新产品，一旦大家都同意确实需要这个新产品，每位与会者就应该把他们在会前准备好的列表展示出来供大家讨论
- 可以把这些列表抄写在大纸上钉在墙上，或者写在白板上挂在墙上
- 理想的情况是：表中每一项都能单独移动，这样就能方便地删除或增添表项，或组合不同的列表
- 在此阶段，严格禁止批评与争论

(4)在展示了每个人针对某个议题的列表之后，大家共同创建一张组合列表
- 在组合列表中消去了冗余项，加入了在展示过程中产生的新想法，但是并不删除任何实质性内容
- 在针对每个议题的组合列表都建立起来之后，由协调人主持讨论这些列表
- 组合列表将被缩短、加长或重新措辞，以便更准确地描述将被开发的产品
- 讨论的目标是，针对每个议题(对象、服务、约束和性能)都创建出一张意见一致的列表

(5)一旦得出了意见一致的列表，就把与会者分成更小的小组
- 每个小组的工作目标是为每张列表中的项目制定小型规格说明
- 小型规格说明是对列表中包含的单词或短语的准确说明

(6)每个小组都向全体与会者展示他们制定的小型规格说明，供大家讨论
- 通过讨论可能会增加或删除一些内容，或进一步做精化工作

(7)在完成了小型规格说明之后，每个与会者都制定出产品的一整套确认标准，并把自己制定的标准提交会议讨论，以创建出意见一致的确认标准

(8)由一名或多名与会者根据会议成果起草完整的软件需求规格说明书

与用户沟通获取需求的方法（CONT.）

- 简易的应用规格说明技术的典型过程
  - 简易的应用规格说明技术并不能解决需求分析阶段遇到的所有问题
  - 但是此方法具备以下优点
    - 开发者与用户不分彼此，齐心协力，密切合作
    - 即时讨论并求精
  - 具备导出规格说明的具体步骤和指导方法

与用户沟通获取需求的方法（CONT.）

- 快速建立软件原型
  - 是最准确、最有效、最强大的需求分析技术
  - 目的
    - 演示目标系统主要功能的可运行的程序
  - 方法要点
    - 实现用户看得见的功能(例如，屏幕显示或打印报表)
    - 省略目标系统的“隐含”功能(例如，修改文件)
  - 快速原型的特点
    - 快速
      - 快速原型的目的是尽快向用户提供一个可在计算机上运行的目标系统的模型，以使用户和开发者在目标系统应该“做什么”这个问题上尽快达成共识
      - 原型的某些缺陷是可以忽略的，只要这些缺陷不严重地损害原型的功能，不会使用户对产品的行为产生误解，即可进行忽略
    - 容易修改
      - 如果当前原型不是用户所需要的，就必须根据用户的意见迅速地修改它，构建出新版原型，以更好地满足用户需求
      - 可避免在实际开发软件产品时，原型的“修改—试用—反馈”过程可能重复多遍，进而耗时过多，延误软件开发时间

与用户沟通获取需求的方法（CONT.）

- 快速建立软件原型的方法和工具
  - 第四代技术
    - 包括众多数据库查询和报表语言、程序和应用系统生成器以及其他非常高级的非过程语言
    - 第四代技术使得软件工程师能够快速地生成可执行的代码，是较理想的快速原型工具
  - 可重用的软件构件
    - 即使用一组已有的软件构件(也称为组件)来装配(而不是从头构造)原型
    - 软件构件可以是数据结构(或数据库)，或软件体系结构构件(即程序)，或过程构件(即模块)
    - 必须把软件构件设计成能在不知其内部工作细节的条件下重用
    - 应该注意，现有的软件可以被用作“新的或改进的”产品的原型
  - 形式化规格说明和原型环境
    - 基于已有的多种形式化规格说明语言和工具，可用其替代自然语言规格说明
      - 目前形式化语言的倡导者正在开发交互式环境，以便可以调用自动工具把基于形式语言的规格说明翻译成可执行的程序代码，使用户使用可执行的原型代码去进一步精化形式化的规格说明

分析建模与规格说明

- 分析建模
  - 为了更好地理解复杂事物，可采用建立事物模型的方法
- 结构化分析实质上是一种创建模型的活动
  - 为了开发出复杂的软件系统，系统分析员应该
    - 从不同角度抽象出目标系统的特性
    - 使用精确的表示方法构造系统的模型
    - 验证模型是否满足用户对目标系统的需求
    - 并在设计过程中逐渐把和实现有关的细节加进模型中
    - 直至最终用程序实现模型
- 需求分析过程应该建立3种模型
  - 数据模型
    - 实体-联系图，描绘数据对象及数据对象之间的关系，是用于建立数据模型的图形
  - 功能模型
    - 数据流图，描绘当数据在软件系统中移动时被变换的逻辑过程，指明系统具有的变换数据的功能，是建立功能模型的基础
  - 行为模型
    - 状态转换图(简称为状态图)，指明了作为外部事件结果的系统行为
    - 状态转换图描绘了系统的各种行为模式(称为“状态”)和在不同状态间转换的方式
    - 状态转换图是行为建模的基础

- 所谓模型，就是为了理解事物而对事物做出的一种抽象，是对事物的一种无歧义的书面描述
  - 通常，模型由一组图形符号和组织这些符号的规则组成

分析建模与规格说明（CONT.）

- 软件需求规格说明
  - 通过需求分析除了创建分析模型之外，还应该写出软件需求规格说明书
  - 需求规格说明书是需求分析阶段得出的最主要的文档
  - 要求
    - 通常用自然语言完整、准确、具体地描述系统的数据要求、功能需求、性能需求、可靠性和可用性要求、出错处理需求、接口需求、约束、逆向需求以及将来可能提出的要求
      - 自然语言的规格说明具有容易书写、容易理解的优点，为大多数人所欢迎和采用
  - 为了消除用自然语言书写的软件需求规格说明书中可能存在的不一致、歧义、含糊、不完整及抽象层次混乱等问题，可采用形式化方法描述用户对软件系统的需求

实体 - 联系图

- 为了把用户的数据要求清楚、准确地描述出来，系统分析员通常建立一个概念性的数据模型(也称为信息模型)
- 概念性数据模型是一种面向问题的数据模型，是按照用户的观点对数据建立的模型
  - 描述了从用户角度看到的数据，反映了用户的现实环境，并且与在软件系统中的实现方法无关
- 数据模型中包含3种相互关联的信息
  - 数据对象
  - 数据对象的属性
  - 数据对象彼此间相互连接的关系

实体 - 联系图

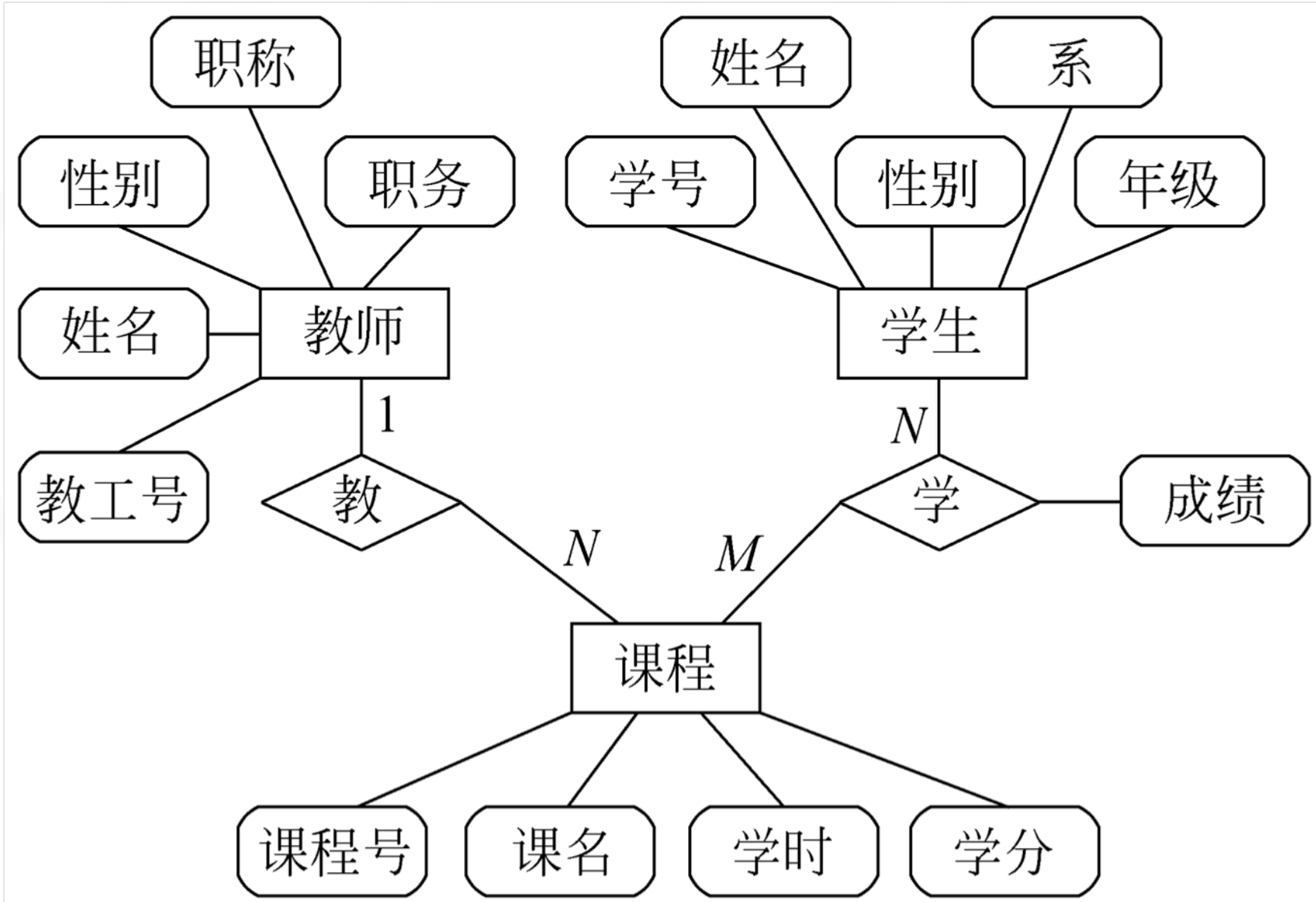
- 数据对象
  - 是对软件必须理解的复合信息的抽象
    - 所谓复合信息是指具有一系列不同性质或属性的事物，仅有单个值的事物(例如，宽度)不是数据对象
  - 数据对象可以是
    - 外部实体(例如，产生或使用信息的任何事物)
    - 事物(例如，报表)
    - 行为(例如，打电话)
    - 事件(例如，响警报)
    - 角色(例如，教师、学生)
    - 单位(例如，会计科)
    - 地点(例如，仓库)
    - 结构(例如，文件)等
  - 数据对象彼此间是有关联的
    - 例如，教师“教”课程，学生“学”课程，教或学的关系表示教师和课程或学生和课程之间的一种特定的连接
  - 数据对象只封装了数据而没有对施加于数据上的操作的引用
    - 即是数据对象与面向对象范型(参见本书第9章)中的“类”或“对象”的显著区别
- 可以由一组属性来定义的实体都可以被认为是数据对象

实体 - 联系图

- 数据对象的属性
  - 属性定义了数据对象的性质
  - 必须把一个或多个属性定义为“标识符”
    - 即，当希望找到数据对象的一个实例时，可以用标识符属性作为“关键字”(通常简称为“键”)。
  - 应该根据对所要解决的问题的理解，来确定特定数据对象的一组合适的属性

实体 - 联系图

- 数据对象的关系，即联系
  - 数据对象彼此之间相互连接的方式称为联系，也称为关系
  - 联系可分为3种类型
    - 一对一联系(1：1)
      - 例如，一个部门有一个经理，而每个经理只在一个部门任职，则部门与经理的联系是一对一的
    - 一对多联系(1：N)
      - 例如，某校教师与课程之间存在一对多的联系“教”，即每位教师可以教多门课程，但是每门课程只能由一位教师来教
    - 多对多联系(M：N)
      - 例如，图3.2表示学生与课程间的联系(“学”)是多对多的，即一个学生可以学多门课程，而每门课程可以有多个学生来学
  - 联系也可能有属性
    - 例如，学生“学”某门课程所取得的成绩，既不是学生的属性也不是课程的属性
    - 由于“成绩”既依赖于某名特定的学生又依赖于某门特定的课程，所以是学生与课程之间的联系“学”的属性


职称
性别
职务
姓名
教师
1
教工号
教
N
姓名
系
学号
性别
年级
学生
N
学
成绩
M
课程
课程号
课名
学时
学分

实体 - 联系图

- 实体-联系图的符号
  - 可使用实体-联系图(entity-relationship diagram)来建立数据模型，简称为E-R图
    - 相应地可把用E-R图描绘的数据模型称为E-R模型
  - E-R图中包含了3种基本成分
  - 实体(即数据对象)
    - 通常用矩形框代表
  - 关系
    - 用连接相关实体的菱形框表示
  - 属性
    - 用椭圆形或圆角矩形表示
    - 直线把实体(或关系)与其属性连接起来

数据规范化

- 原因
  - 软件系统经常使用各种长期保存的信息，这些信息通常以一定方式组织并存储在数据库或文件中，为减少数据冗余，避免出现插入异常或删除异常，简化修改数据的过程，通常需要把数据结构规范化
- 通常用“范式(normal forms)”定义消除数据冗余的程度
  - 第一范式(1 NF)数据冗余程度最大，第五范式(5 NF)数据冗余程度最小
    - 范式级别越高，存储同样数据就需要分解成更多张表，因此，“存储自身”的过程也就越复杂。
  - 随着范式级别的提高，数据的存储结构与基于问题域的结构间的匹配程度也随之下降，
    - 在需求变化时数据的稳定性较差
  - 范式级别提高则需要访问的表增多，因此性能(速度)将下降
    - 从实用角度看来，在大多数场合选用第三范式都比较恰当

数据规范化

- 范式
  - 通常按照属性间的依赖情况区分规范化的程度
  - 属性间依赖情况满足不同程度要求的为不同范式
    - 满足最低要求的是第一范式
    - 在第一范式中再进一步满足一些要求的为第二范式
    - 其余依此类推
  - 第一范式
    - 每个属性值都必须是原子值，即仅仅是一个简单值而不含内部结构
  - 第二范式
    - 满足第一范式条件
    - 且每个非关键字属性都由整个关键字决定(而不是由关键字的一部分来决定)
  - 第三范式
    - 符合第二范式的条件
    - 每个非关键字属性都仅由关键字决定
    - 且一个非关键字属性不能仅仅是对另一个非关键字属性的进一步描述
      - 即一个非关键字属性值不依赖于另一个非关键字属性值

状态转换图

- 在需求分析过程中应该建立起软件系统的行为模型
- 状态转换图(简称为状态图)
  - 通过描绘系统的状态及引起系统状态转换的事件，来表示系统的行为
  - 状态图还指明了作为特定事件的结果系统将做哪些动作(例如，处理数据)
- 状态图提供了行为建模机制，可以满足需求分析的第3条分析准则要求
  - 必须描述作为外部事件结果的软件行为

状态转换图

- 状态
  - 是任何可以被观察到的系统行为模式
  - 一个状态代表系统的一种行为模式
- 状态规定了系统对事件的响应方式
  - 系统对事件的响应
    - 既可以是做一个(或一系列)动作
    - 也可以是仅仅改变系统本身的状态
    - 还可以是既改变状态又做动作
- 在状态图中定义的状态主要有
  - 初态(即初始状态)
  - 终态(即最终状态) 和
  - 中间状态
- 在一张状态图中只能有一个初态，而终态则可以有0至多个
- 状态图既可以表示系统循环运行过程，也可以表示系统单程生命期
  - 当描绘循环运行过程时，通常并不关心循环是怎样启动的
  - 当描绘单程生命期时，需要标明初始状态(系统启动时进入初始状态)和最终状态(系统运行结束时到达最终状态)

状态转换图

- 事件
  - 是在某个特定时刻发生的事情
  - 是对引起系统做动作或(和)从一个状态转换到另一个状态的外界事件的抽象
    - 例如，内部时钟表明某个规定的时间段已经过去，用户移动或点击鼠标等都是事件
- 在状态转换图中，事件是引起系统做动作或(和)转换状态的控制信息

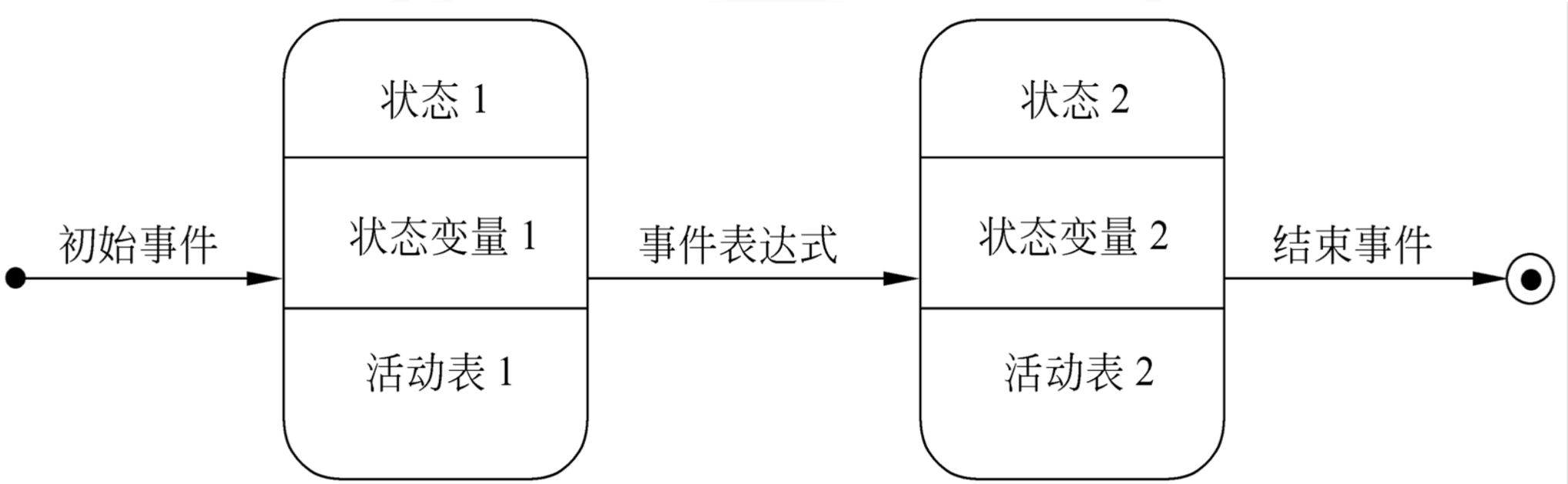
状态转换图

- 符号表达
  - 初态用实心圆表示
  - 终态用一对同心圆(内圆为实心圆)表示
  - 中间状态用圆角矩形表示，可以用两条水平横线把它分成上、中、下3个部分
    - 上面部分为状态的名称，是必须有的
    - 中间部分为状态变量的名字和值，是可选的
    - 下面部分是活动表，是可选的
- 活动表的语法格式
  - 事件名(参数表)/动作表达式、
    - “事件名”可以是任何事件的名称
      - 在活动表中经常使用下述3种标准事件：entry，exit和do
      - entry事件指定进入该状态的动作
      - exit事件指定退出该状态的动作
      - do事件则指定在该状态下的动作
    - 需要时可以为事件指定参数表
    - 活动表中的动作表达式描述应做的具体动作

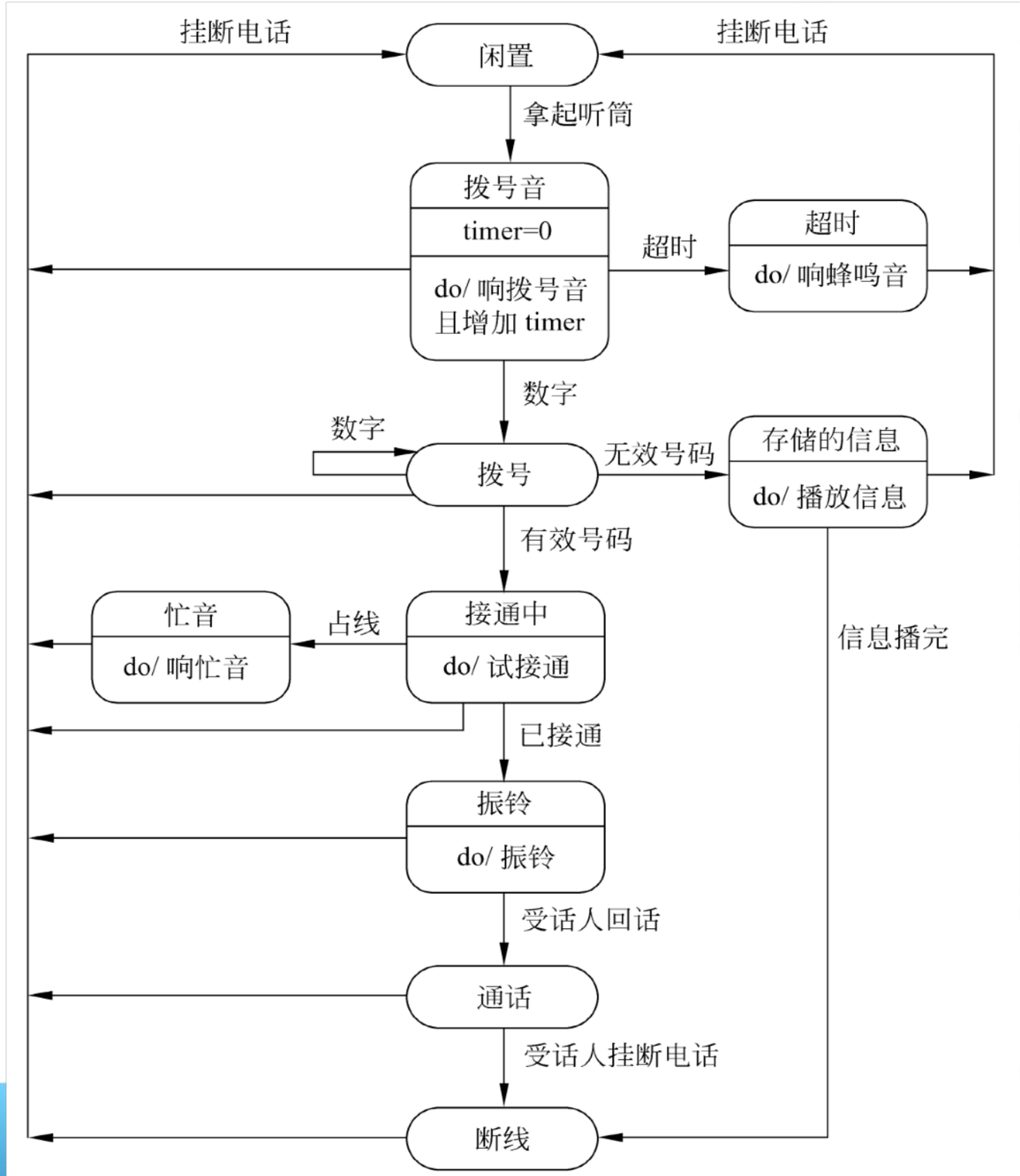
状态转换图

- 符号表达（cont.）
  - 状态图中两个状态之间带箭头的连线称为状态转换，箭头指明了转换方向
  - 状态转换通常是由事件触发的
    - 在这种情况下应在表示状态转换的箭头线上标出触发转换的事件表达式
    - 如果在箭头线上未标明事件，则表示在源状态的内部活动执行完之后自动触发转换
  - 事件表达式的语法如下：
    - 事件说明［守卫条件］ / 动作表达式
    - 事件说明的语法为：事件名(参数表)。
    - 守卫条件是一个布尔表达式
    - 如果同时使用事件说明和守卫条件，则当且仅当事件发生且布尔表达式为真时，状态转换才发生
    - 如果只有守卫条件没有事件说明，则只要守卫条件为真状态转换就发生
    - 动作表达式是一个过程表达式，当状态转换开始时执行该表达式

状态转换图



状态转换图 - 示例



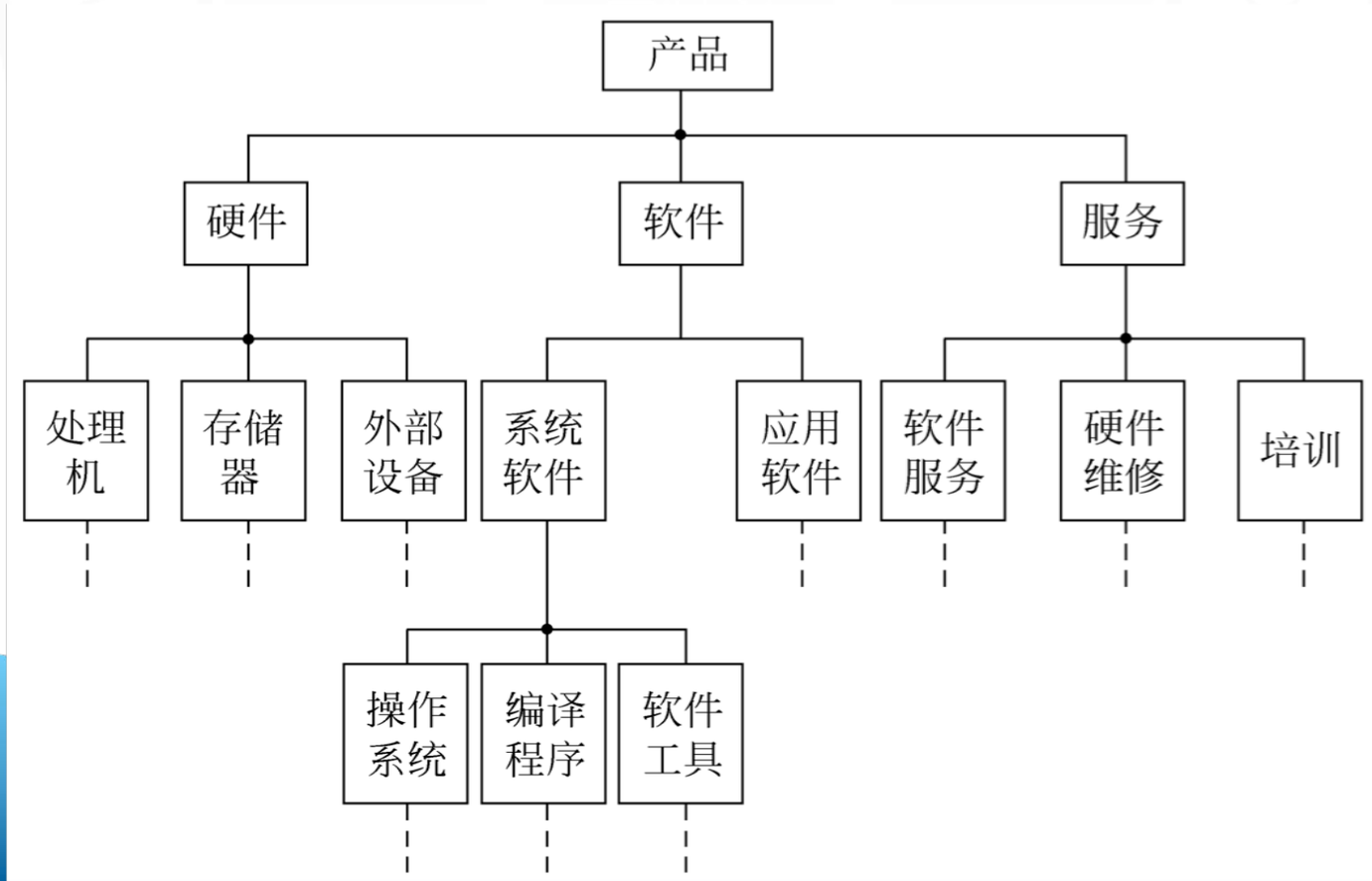
- 图中表明
  - 没有人打电话时电话处于闲置状态；
  - 有人拿起听筒则进入拨号音状态，到达这个状态后，电话的行为是响起拨号音并计时；
  - 这时如果拿起听筒的人改变主意不想打了，他把听筒放下(挂断)，电话重又回到闲置状态；
  - 如果拿起听筒很长时间不拨号(超时)，则进入超时状态；
  - ……

其他图形工具

- 层次方框图
- Warnier图
- IPO图

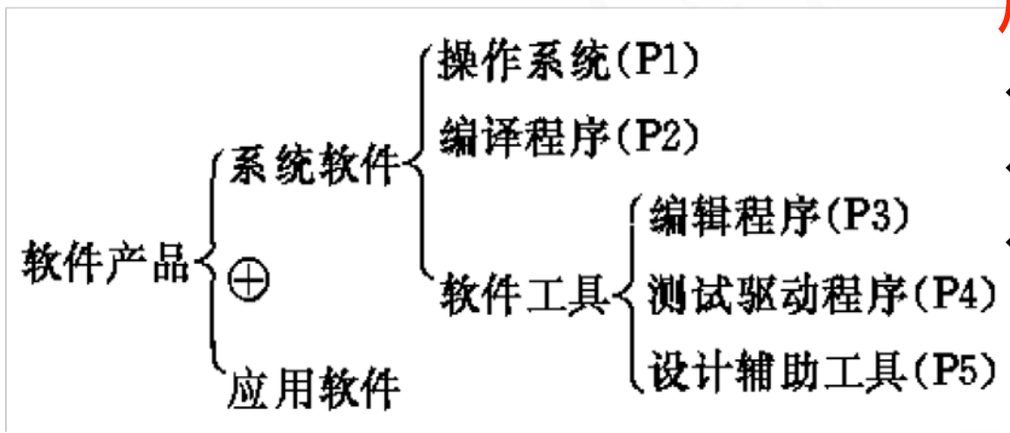
层次方框图

- 用树形结构的一系列多层次的矩形框描绘数据的层次结构
  - 树形结构的顶层是一个单独的矩形框，代表完整的数据结构
  - 下面的各层矩形框代表这个数据的子集
  - 最底层的各个框代表组成这个数据的实际数据元素(不能再分割的元素)
- 随着结构的精细化，层次方框图对数据结构也描绘得越来越详细，这种模式非常适合于需求分析阶段的需要
  - 系统分析员从对顶层信息的分类开始，沿图中每条路径反复细化，直到确定了数据结构的全部细节时为止
- 例如，描绘一家计算机公司全部产品的数据结构



WARNIER图

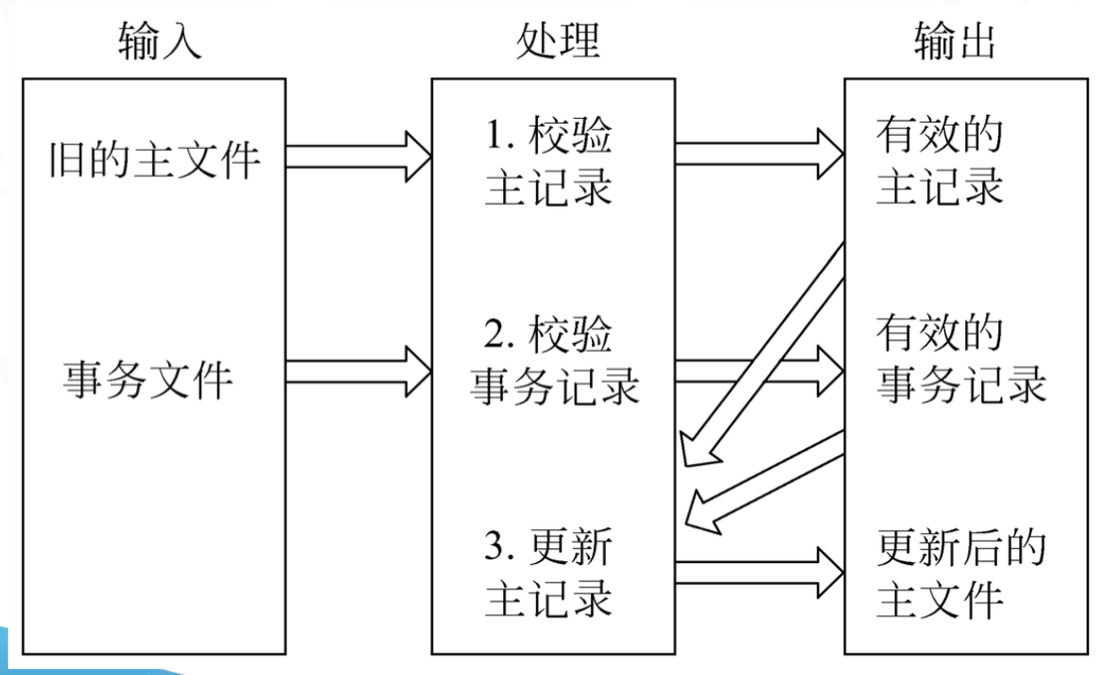
- 和层次方框图类似，Warnier图也用树形结构描绘信息，但是这种图形工具比层次方框图提供了更丰富的描绘手段
- 用Warnier图可以表明信息的逻辑组织
  - 可以指出一类信息或一个信息元素是重复出现的
  - 也可以表示特定信息在某一类信息中是有条件地出现的
- 因为重复和条件约束是说明软件处理过程的基础，所以很容易把Warnier图转变成软件设计的工具



- 示例表示：一种软件产品要么是系统软件要么是应用软件
  - 系统软件中有P1种操作系统
  - P2种编译程序，此外还有软件工具
  - 软件工具是系统软件的一种
    - 它又可以进一步细分为
      - 编辑程序
      - 测试驱动程序
      - 设计辅助工具

IPO图

- IPO图是输入、处理、输出图的简称
  - 它是美国IBM公司发展完善起来的一种图形工具
  - 能够方便地描绘输入数据、对数据的处理和输出数据之间的关系
- IPO图使用的基本符号既少又简单，易于学习和使用
- 基本形式
  - 在左边的框中列出有关的输入数据
  - 在中间的框内列出主要的处理
    - 处理框中列出处理的次序暗示了执行的顺序
    - 但是用这些基本符号还不足以精确描述执行处理的详细情况
  - 在右边的框内列出产生的输出数据
  - 有时还用类似向量符号的粗大箭头清楚地指出数据通信的情况



改进的IPO图（IPO表）

- 在软件设计过程中比原始的IPO图更有用
- 在需求分析阶段可以使用IPO图简略地描述系统的主要算法(即数据流图中各个处理的基本算法)
  - 在需求分析阶段，IPO图中的许多附加信息暂时还不具备
  - 但是在软件设计阶段可以进一步补充修正这些图，作为设计阶段的文档
- 即，在需求分析阶段用IPO图作为描述算法的工具的重要优点

IPO 表

系统：_____ 作者：_____

模块：_____ 日期：_____

编号：_____

被调用：	调用：
输入：	输出：
处理：	
局部数据元素：	注释：

验证软件需求

- 需求分析阶段的工作结果是开发软件系统的重要基础
  - 大量统计数字表明，软件系统中15%的错误起源于错误的需求
- 需求分析必须严格验证其正确性
  - 目的在于提高软件质量，确保软件开发成功，降低软件开发成本
- 从下述4个方面进行验证
  - 一致性
    - 所有需求必须是一致的，任何一条需求不能和其他需求互相矛盾
  - 完整性
    - 需求必须是完整的，规格说明书应该包括用户需要的每一个功能或性能
  - 现实性
    - 指定的需求应该是用现有的硬件技术和软件技术基本上可以实现的
      - 对硬件技术的进步可以做些预测，对软件技术的进步则很难做出预测，只能从现有技术水平出发判断需求的现实性
  - 有效性
    - 必须证明需求是正确有效的，确实能解决用户面对的问题

验证软件需求

- 验证需求的一致性
  - 当需求分析的结果是用自然语言书写的时候，除了靠人工技术审查验证软件系统规格说明书的正确性之外，目前还没有其他更好的“测试”方法
  - 非形式化的规格说明书是难于验证的
    - 特别在目标系统规模庞大、规格说明书篇幅很长的时候，人工审查的效果是没有保证的
    - 冗余、遗漏和不一致等问题可能没被发现而继续保留下来
    - 以致软件开发工作不能在正确的基础上顺利进行
- 为了克服上述困难，提出了形式化的描述软件需求的方法
  - 当软件需求规格说明书是用形式化的需求陈述语言书写的时候，可以用软件工具验证需求的一致性，从而能有效地保证软件需求的一致性

验证软件需求

- 验证需求的现实性
  - 为了验证需求的现实性，分析员应该参照以往开发类似系统的经验，分析用现有的软、硬件技术实现目标系统的可能性
  - 必要的时候应该采用仿真或性能模拟技术，辅助分析软件需求规格说明书的现实性

验证软件需求

- 验证需求的完整性和有效性
  - 只有目标系统的用户才真正知道软件需求规格说明书是否完整、准确地描述了他们的需求
  - 因此，检验需求的完整性，特别是证明系统确实满足用户的实际需要(即，需求的有效性)，只有在用户的密切合作下才能完成
  - 然而许多用户并不能清楚地认识到他们的需要
    - 特别在要开发的系统是全新的，以前没有使用类似系统的经验时，不能有效地比较陈述需求的语句和实际需要的功能，只有通过对现有正在使用的软件系统有实际使用经验和评价时，才能完整确切地提出用户需求
  - 理想的做法是先根据需求分析的结果开发出一个软件系统，请用户试用一段时间以便能认识到他们的实际需要是什么，在此基础上再写出正式的“正确的”规格说明书
    - 但是，会使软件成本增加一倍，因此实际上几乎不可能采用这种方法
    - 使用原型系统是一个比较现实的替代方法，开发原型系统所需要的成本和时间可以大大少于开发实际系统所需要的
      - 用户通过试用原型系统，也能获得许多宝贵的经验，从而可以提出更符合实际的要求
    - 使用原型系统的目的，通常是显示目标系统的主要功能而不是性能

用于需求分析的软件工具

- 该类软件工具应该满足的要求
  - 必须有形式化的语法(或表)
    - 可以用计算机自动处理使用这种语法说明的内容
- 使用该类软件工具能够导出详细的文档
- 必须提供分析(测试)规格说明书的不一致性和冗余性的手段，并且应该能够产生一组报告指明对完整性分析的结果
- 使用该类软件工具后，应该能够改进通信状况

习题

- 复印机的工作过程大致如下
  - 未接到复印命令时处于闲置状态
  - 一旦接到复印命令则进入复印状态，完成一个复印命令规定的工作后又回到闲置状态，等待下一个复印命令
  - 如果执行复印命令时发现没纸，则进入缺纸状态，发出警告，等待装纸，装满纸后进入闲置状态，准备接收复印命令
  - 如果复印时发生卡纸故障，则进入卡纸状态，发出警告等待维修人员来排除故障，故障排除后回到闲置状态
- 请用状态转换图描绘复印机的行为