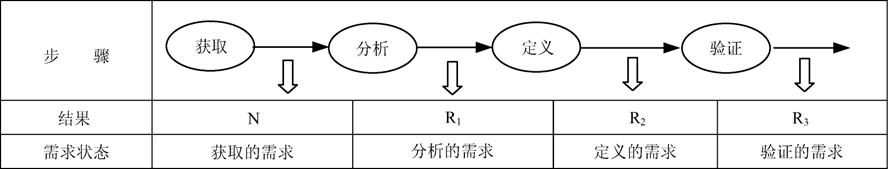
# 软件工程概论3-4

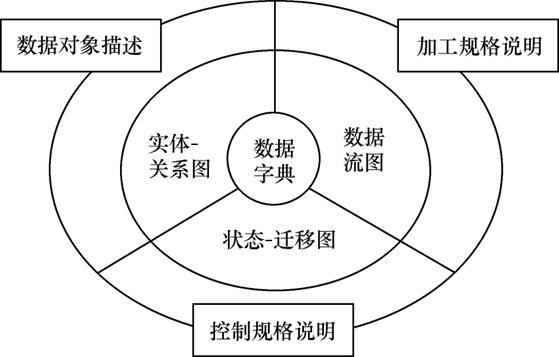
1. 三、软件需求获取与结构化分析方法
   1. 1.需求获取的任务和原则
      1. 任务：
         1. (1) 发现和分析问题，并分析问题的原因/结果关系。
         2. (2) 与用户进行各种方式的交流，并使用调查研究方法收集信息。
         3. (3) 按照三个成分观察问题的不同侧面：即数据、过程和接口。
         4. (4) 将获取的需求文档化，形式有用例、决策表、需求表等。
      2. 原则：
         1. (1) 深入浅出的原则。就是说，需求获取要尽可能全面、细致。获取的需求是个全集，目标系统真正实现的是个子集。
         2. (2) 以流程为主线的原则。在与用户交流的过程中，应该用流程将所有的内容串起来。如信息、组织结构、处理规则等。这样便于交流沟通。流程的描述既有宏观描述，也有微观描述。
   2. 2.需求获取的过程
      1. 1. 开发高层的业务模型
      2. 2. 定义项目范围和高层需求
      3. 3. 识别用户类和用户代表
         1. 系统的不同用户之间在很多方面存在差异，例如：
            1. (1) 使用产品的频率；
            2. (2) 用户在应用领域的经验和使用计算机系统的技能；
            3. (3) 所用到的产品功能；
            4. (4) 为支持业务过程所进行的工作；
            5. (5) 访问权限和安全级别
      4. 4. 获取具体的需求
         1. 确定了项目范围和高层需求，并确定了用户类及用户代表后，就需要获取更具体、完整和详细的需求。具体需求的来源可以来自以下几种典型的途径。
            1. (1) 与用户进行交流。
            2. (2) 现有产品或竞争产品的描述文档。
            3. (3) 系统需求规格说明。
            4. (4) 当前系统的问题报告和改进要求。
            5. (5) 市场调查和用户问卷调查。
            6. (6) 观察用户如何工作。
      5. 5. 确定目标系统的业务工作流
         1. 具体到当前待开发的应用系统，确定系统的业务工作流和主要的业务规则，采取需求调研的方法获取所需的信息。例如，针对信息系统的需求调研方法如下：
            1. (1) 调研用户的组织结构、岗位设置、职责定义，从功能上区分有多少个子系统，划分系统的大致范围，明确系统的目标。
            2. (2) 调研每个子系统的工作流程、功能与处理规则，收集原始信息资料，用数据流来表示物流、资金流、信息流三者的关系。
            3. (3) 对调研内容事先准备，针对不同管理层次的用户询问不同的问题，列出问题清单。将操作层、管理层、决策层的需求既联系又区分开来，形成一个需求的层次。
      6. 6. 需求整理与总结
         1. 必须对上面步骤取得的需求资料进行整理和总结，确定对软件系统的综合要求，即软件的需求。
         2. 并提出这些需求实现条件，以及需求应达到的标准。
         3. 这些需求包括功能需求、性能需求、环境需求、可靠性需求、安全保密要求、用户界面需求、资源使用需求、软件成本消耗与开发进度需求等。
   3. 3.软件需求分析阶段的任务
      1. 可以把软件需求分析阶段的工作分为4个步骤，即获取需求、分析需求、定义需求和验证需求，如图所示。



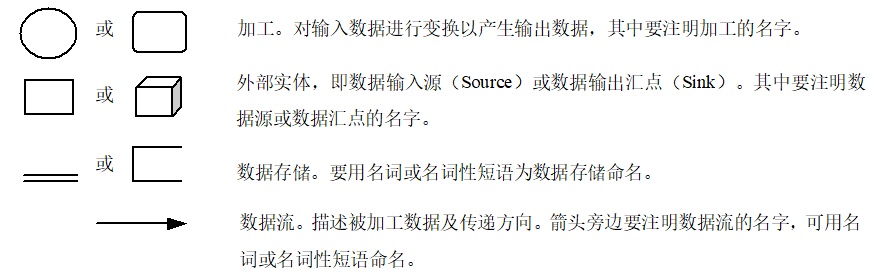
* + 1. 1. 需求获取
       1. 通过启发、引导从客户（或用户）那里得到的原始需求是他们的业务要求（needs），简称为N。
       2. 这是分析之前获取的需求，其中可能存在一些实际问题，这些问题只有通过分析才能得到解决，直接把获取的需求作为软件设计阶段的依据将会导致严重的后果。
    2. 2. 需求分析
       1. 认真研究获取的需求，必须考虑以下几方面：



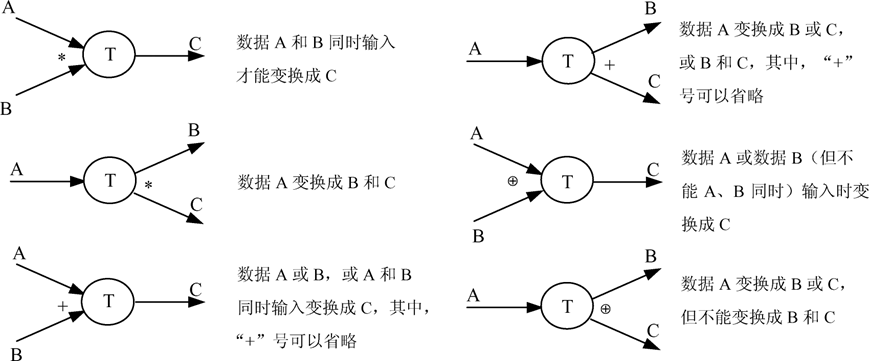
* + - 1. (1) 完整性：每项获取的需求都应给出清楚的描述，使得软件开发工作能够取得设计和实现该功能所需要的全部必要信息。
      2. (2) 正确性：获取的每项需求必须是准确无误的，并且需求描述无歧义性。
      3. (3) 合理性：各项需求之间、软件需求与系统需求之间应是协调一致的，不应存在矛盾和冲突。
      4. (4) 可行性：包括技术可行性 、经济可行性 、社会可行性 。
      5. (5) 充分性：获取的需求是否全面、周到。
    1. 3. 需求定义
       1. 将已经过分析的需求清晰、全面、系统、准确地描述成为正式的文档，这一步定义需求的工作就是编写需求规格说明。
    2. 4. 需求验证
       1. 为了确保已定义的需求（需求规格说明）准确无误，并能为客户（或用户）理解和接受，需要对其进行严格的评审。
  1. 4.结构化分析方法
     1. 传统的分析建模方法称为结构化分析（structured analysis，SA）方法。结构化分析方法是一种建模技术，它建立的分析模型如图所示。



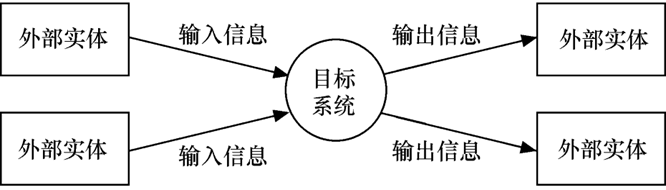
* + 1. 1.功能建模
       1. 概念： 功能建模的思想就是用抽象模型的概念，按照软件内部数据传递、变换的关系，自顶向下逐层分解，直到找到满足功能要求的所有可实现的软件为止。功能模型用数据流图来描述。
       2. 基本图形符号：



* + - 1. 多个数据流之间的关系 ：



* + - 1. 环境图：、
         1. 环境图（context diagram）也称为顶层数据流图（或0层数据流图），它仅包括一个数据处理过程，也就是要开发的目标系统。环境图的作用是确定系统在其环境中的位置，通过确定系统的输入和输出与外部实体的关系确定其边界。
         2. 典型的环境图



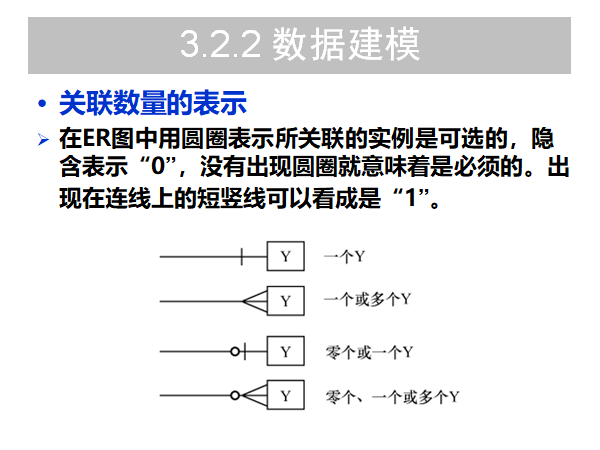
* + 1. 2.数据建模
       1. 概念
          1. 在结构化分析方法中，使用实体—关系建模技术来建立数据模型。这种技术是在较高的抽象层次（概念层）上对数据库结构进行建模的流行技术。实体—关系模型表示为可视化的实体—关系图（entity-relationship diagram，ERD），也称为ER图。
          2. ER图中仅包含3种相互关联的元素：数据对象（实体）、描述数据对象的属性及数据对象彼此间相互连接的关系。
       2. 数据对象：
          1. 数据对象是目标系统所需要的复合信息的表示，所谓复合信息是具有若干不同属性的信息。在ER图中用矩形表示数据对象。
          2. 在实际问题中，数据对象（实体）可以是外部实体、事物、角色、行为或事件、组织单位、地点或结构等。
       3. 属性：
          1. 属性定义数据对象的特征，如数据对象学生的学号、姓名、性别、专业等，课程的课程编号、课程名称、学分等。在ER图中用椭圆或圆角矩形表示属性，并用无向边将属性与相关的数据对象连接在一起。
       4. 关系：
          1. 不同数据对象的实例之间是有关联关系的，在ER图上用无向边表示。
          2. 在无向边的两端应标识出关联实例的数量，也称为关联的重数。
          3. 从关联重数的角度可以将关联分为３种。

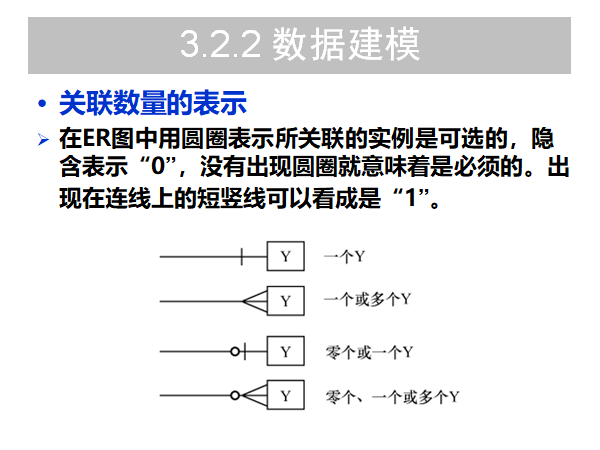
(1) 一对一（1:1）关联

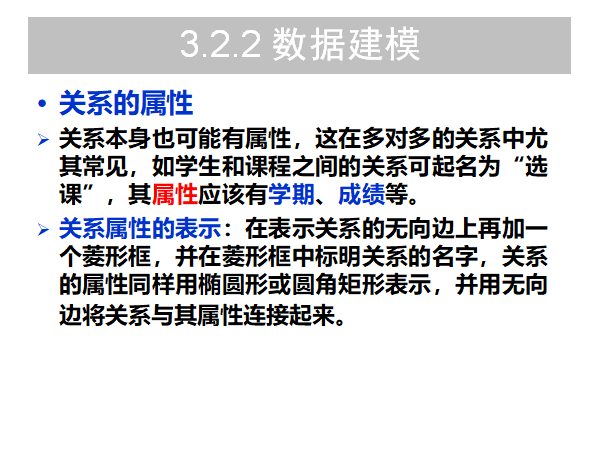
(2) 一对多（1:m）关联

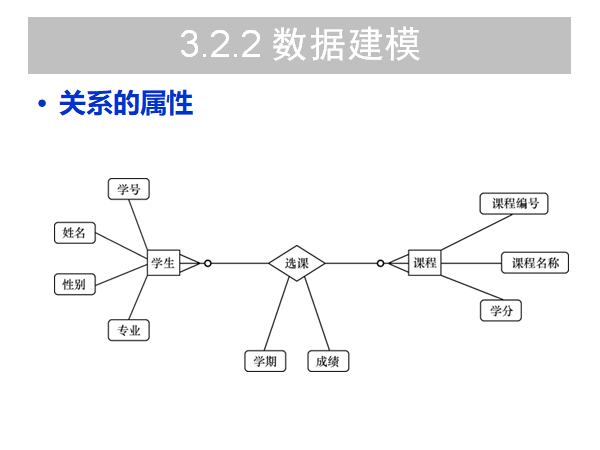
(3) 多对多（m:n）关联

* + - * 1. 实例关联还有“必须”和“可选”之分。

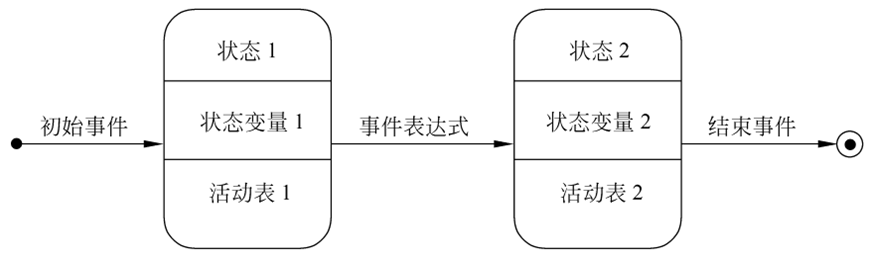




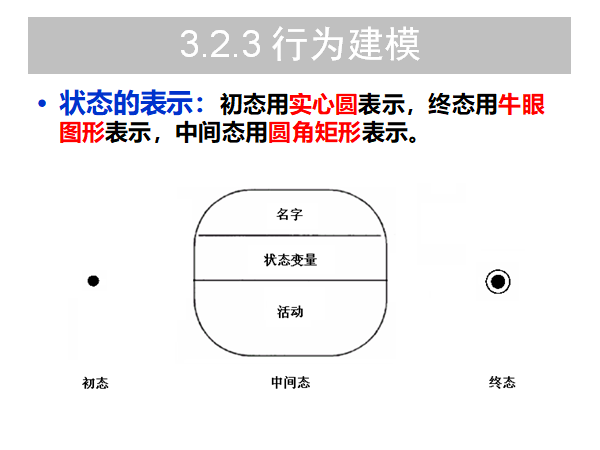




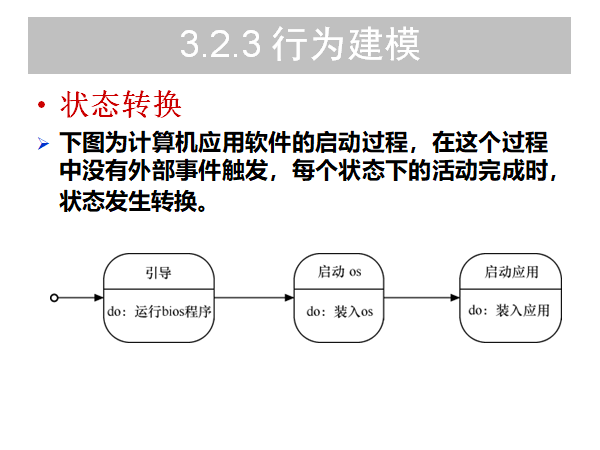
* + 1. 3.行为建模
       1. 概念：
          1. 状态转换图（简称状态图）通过描绘系统的状态及引起系统状态转换的事件，来表示系统的行为。状态图中使用的主要符号如图所示。



* + - 1. 状态：
         1. 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。
         2. 状态可能有：初态（初始状态）、终态（最终状态）和中间态。
         3. 在一张状态图中只能有一个初态，而终态则可以有多个，也可以没有。



* + - 1. 状态转换：
         1. 状态图中两个状态之间带箭头的连线称为状态转换。
         2. 状态的变迁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。
         3. 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。



* + - 1. 事件：
         1. 事件是在某个特定时刻发生的事情，它是对引起系统做动作或从一个状态转换到另一个状态的外部事件的抽象。事件表达式的语法如下：
         2. 事件说明（守卫条件）/动作表达式
         3. (1) 事件说明的语法如下：事件名（参数表）
         4. (2) 守卫条件是一个布尔表达式。如果同时使用守卫条件和事件说明，则当且仅当事件发生且布尔表达式成立时，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。
         5. (3) 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。
    1. 4.数据字典
       1. 词条描述:
          1. 含义：

对于在数据流图中每一个被命名的图形元素均加以定义；

其内容包括图形元素的名字，图形元素的别名或编号，图形元素类别（如加工、数据流、数据文件、数据元素、数据源点或数据汇点等）、描述、定义、位置等。

* + - * 1. （1）数据流词条：数据流是数据结构在系统内传播的路径，数据流词条应包括以下几项内容。

①数据流名：要求与数据流图中该图形元素的名字一致。

②简述：简要介绍它产生的原因和结果。

③组成：数据流的数据结构。

④来源：数据流来自哪个加工或作为哪个数据源的外部实体。

⑤去向：数据流流向哪个加工或作为哪个数据汇点的外部实体。

⑥流通量：单位时间数据的流通量。

⑦峰值：流通量的极限值。

* + - * 1. （2）数据元素词条：数据流图中的每个数据结构都是由数据元素构成的，数据元素是数据处理中最小的、不可再分的单位，它直接反映事物的某一特征。

① 类型：数据元素分为数字型与文字型。数字型又分为离散值和连续值，文字的类型用编码类型和长度区分。

② 取值范围：离散值的取值或是枚举的（如3，17，21），或是介于上下界的一组数（如2..100）；连续值一般是有取值范围的实数集（如0.0..100.0）。对于文字型，文字的取值需加以定义。

③ 相关的数据元素及数据结构。

* + - * 1. （3）数据存储文件词条：数据存储文件是数据保存的地方。一个数据存储文件词条应有以下几项内容。

① 文件名：要求与数据流图中该图形元素的名字一致。

② 简述：简要介绍存放的是什么数据。

③ 组成：文件的数据结构。

④ 输入：从哪些加工获取数据。

⑤ 输出：由哪些加工使用数据。

⑥ 存取方式：分为顺序、直接、关键码等不同存取方式。

⑦ 存取频率：单位时间的存取次数。

* + - * 1. （4）加工词条：加工可以使用诸如判定表、判定树、结构化语言等形式表达，主要描述如下。

① 加工名：要求与数据流图中该图形元素的名字一致。

② 编号：用以反映该加工的层次和父子关系。

③ 简述：加工逻辑及功能简述。

④ 输入：加工的输入数据流。

⑤ 输出：加工的输出数据流。

⑥ 加工逻辑：简述加工程序和加工顺序。

* + - * 1. （5）数据源点及数据汇点词条：对于一个数据处理系统来说，数据源点和数据汇点应比较少。

① 名称：要求与数据流图中该外部实体的名字一致。

② 简述：简要描述是什么外部实体。

③ 有关数据流：该实体与系统交互时涉及哪些数据流。

④ 数目：该实体与系统交互的次数。

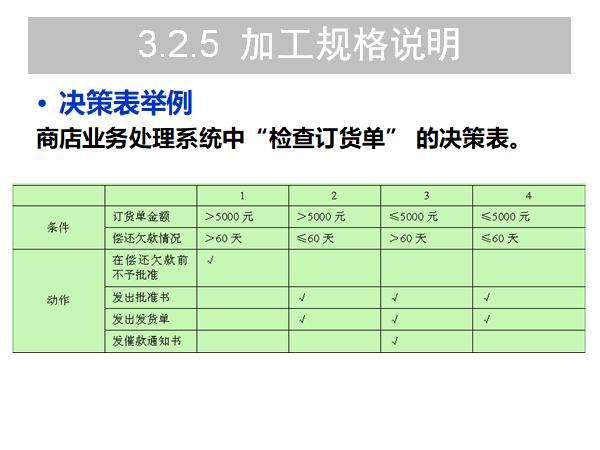
* + - 1. 数据结构描述：
         1. 在数据字典的编制中，分析员最常用的描述数据结构的方式有定义式、Warnier图等。
         2. （1）定义式。在数据流图中，数据流和数据文件都具有一定的数据结构，因此，必须以一种清晰、准确、无二义性的方式来描述数据结构。
         3. （2）Warnier图。Warnier图是表示数据结构的另一种图形工具，它用树形结构来描绘数据结构
      2. 决策表：
         1. 决策表由4个部分组成：

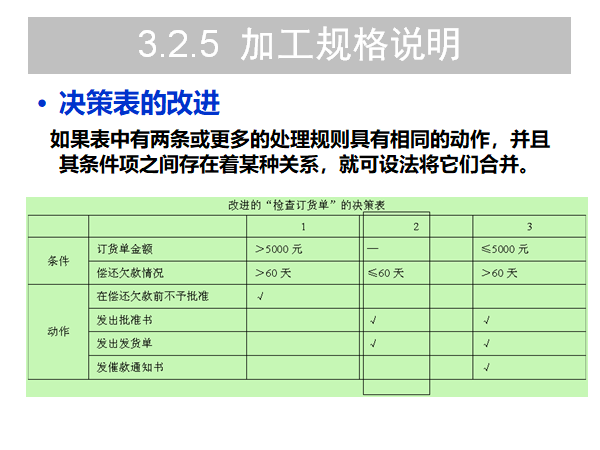
左上部分是条件茬，在此区域列出了各种可能的单个条件；

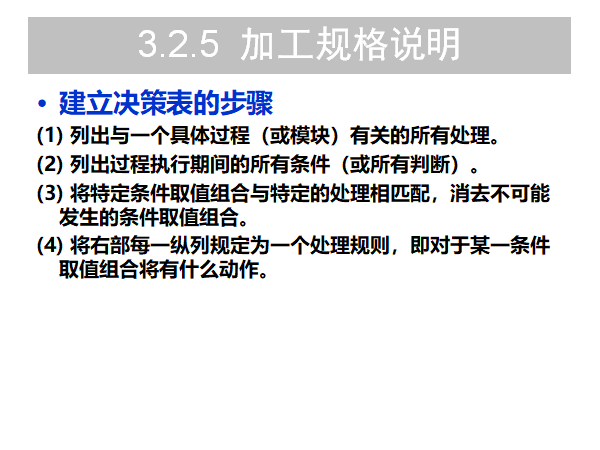
左下部分是动作茬，在此区域列出了可能采取的单个动作；

右上部分是条件项，在此区域列出了针对各种条件的每一组条件取值的组合；

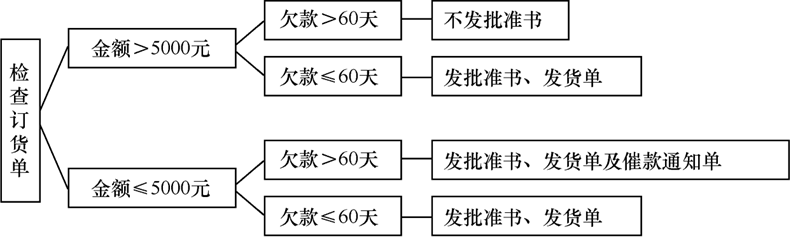
右下部分是动作项，这些动作项与条件项紧密相关，它指出了在条件项的各组取值的组合情况下应采取的动作。



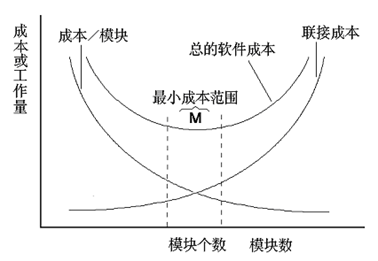




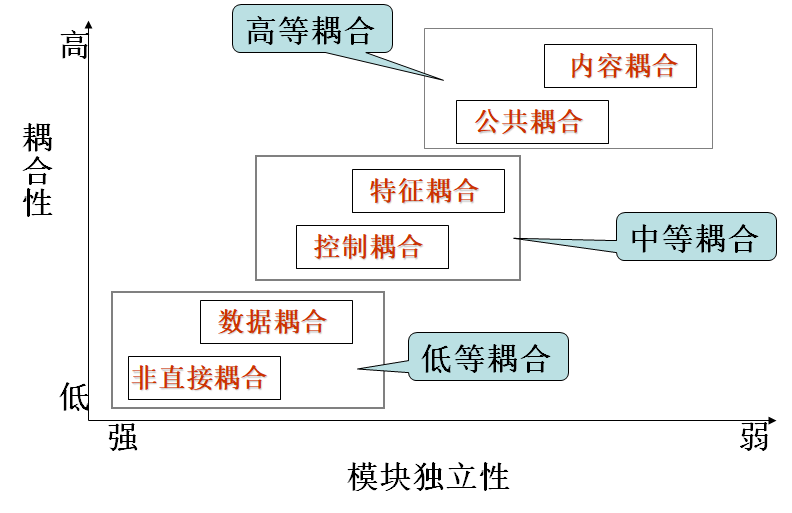
* + - 1. 决策树：
         1. 决策树（decision tree）也是用来表达加工逻辑的一种工具，有时侯它比决策表更直观。
         2. 检查订货单的决策树



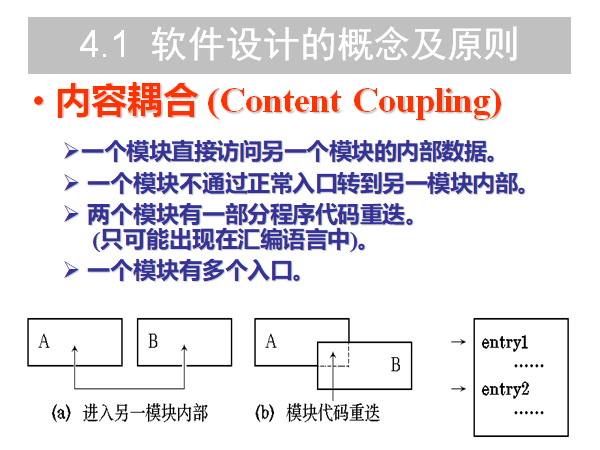
1. 四、结构化设计方法
   1. 1.软件设计的原则
      1. (1) 分而治之
         1. 分而治之是人们解决大型复杂问题时通常采用的策略。将大型复杂的问题分解为许多容易解决的小问题，原来的问题也就容易解决了。软件的体系结构设计、模块化设计都是分而治之策略的具体表现。
         2. 尽管模块分解可以简化要解决的问题，但模块分解并不是越小越好。当模块数目增加时，每个模块的规模将减小，开发单个模块的成本确实减少了；但是，随着模块数目增加，模块之间关系的复杂程度也会增加，设计模块间接口所需要的工作量也将增加，如图所示。



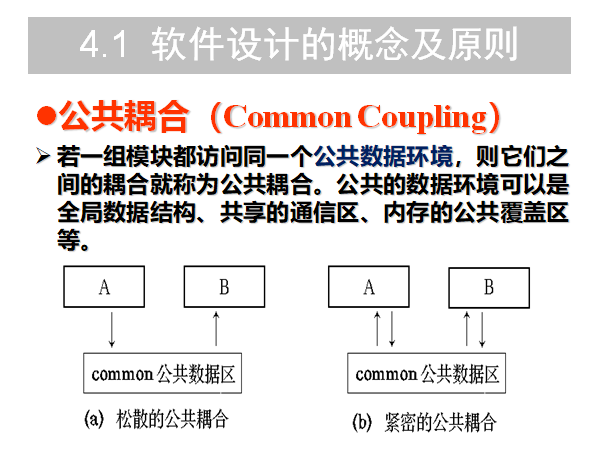
* + 1. (2) 模块独立性
       1. 定义: 是指软件系统中每个模块只涉及软件要求的具体的子功能, 而和软件系统中其它模块的接口是简单的。
       2. 有效的模块化使软件便于分工协作开发。独立的模块比较容易测试和维护。
    2. (3) 提高抽象层次
       1. 抽象是指忽视一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。
       2. 当我们进行软件设计时，设计开始时应尽量提高软件的抽象层次，按抽象级别从高到低进行软件设计。
    3. (4) 复用性设计
       1. 复用是指同一事物不做修改或稍加修改就可以多次重复使用。将复用的思想用于软件开发，称为软件复用。
       2. 我们将软件的重用部分称为软构件。
       3. 也就是说，在构造新的软件系统时不必从零做起，可以直接使用已有的软构件即可组装（或加以合理修改）成新的系统。
    4. (5) 灵活性设计
       1. 保证软件灵活性设计的关键是抽象。
       2. 面向对象系统中的类结构类似一座金字塔，越接近金字塔的顶端，抽象程度就越高。
       3. “抽象”的反义词是“具体”。理想情况下，一个系统的任何代码、逻缉、概念在这个系统中都应该是唯一的，也就是说不存在重复的代码。
       4. 在设计中引入灵活性的方法有：
          1. 降低耦合并提高内聚（易于提高替换能力）；
          2. 建立抽象（创建有多态操作的接口和父类）；
          3. 不要将代码写死（消除代码中的常数）；
          4. 抛出异常（由操作的调用者处理异常）；
          5. 使用并创建可复用的代码。
  1. 2.模块间的耦合与内聚
     1. 模块独立性的度量准则:
        1. 耦合:是模块之间的互相连接的紧密程度的度量。
        2. 内聚:是模块功能强度(一个模块内部各个元素彼此结合的紧密程度)的度量。
        3. 模块独立性比较强的模块应是高内聚低耦合的模块。



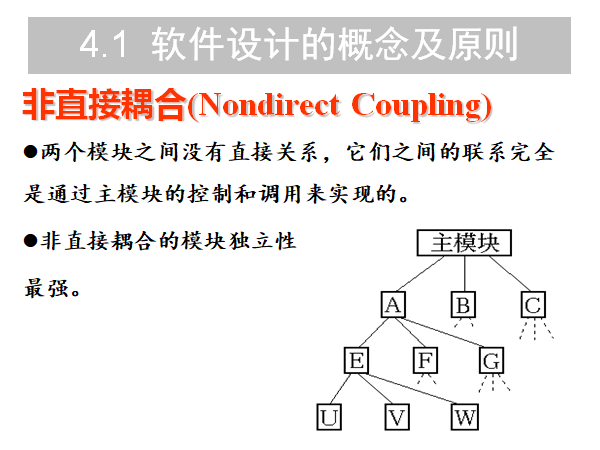
* + 1. 耦合
       1. (1)内容耦合



* + - 1. (2)公共耦合



* + - 1. (3)外部耦合
         1. 若一组模块都访问同一全局简单变量而不是同一全局数据结构，而且不是通过参数表传递全局变量的信息，则称之为外部耦合
      2. (4)控制耦合
         1. 如果一个模块通过传送开关、标志、名字等控制信息，明显地控制选择另一模块的功能，就是控制耦合。
      3. (5)标记耦合
         1. 一组模块通过参数表传递记录信息，就是特征耦合。这个记录是某一数据结构的子结构，而不是简单变量。
      4. (6)数据耦合
         1. 一个模块访问另一个模块时，彼此之间是通过简单数据参数 (不是控制参数、公共数据结构或外部变量) 来交换输入、输出信息的。也是较理想的耦合。
      5. (7)非直接耦合



* + 1. 内聚
       1. (1)巧合内聚（偶然内聚）
          1. 当模块内各部分之间没有联系，或者即使有联系，这种联系也很松散，则称这种模块为偶然内聚模块，内聚程度最低。
          2. 缺点：

1）内容不易理解，很难描述其功能。

2）把完整的程序分割到多个模块中， 在程序运行时会频繁地互相调用

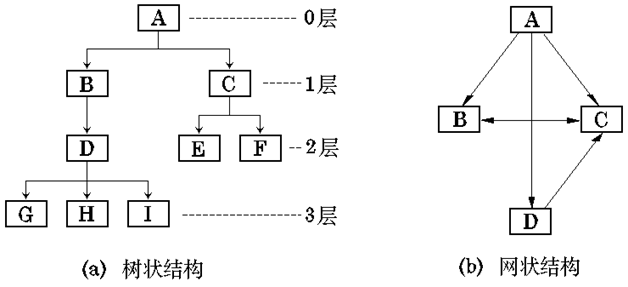
* + - 1. (2)逻辑内聚
         1. 把几种相关的功能组合在一起，每次被调用时，由传送给模块的判定参数来确定该模块应执行哪个功能。
         2. 缺点

1）不易修改，因包含多个功能

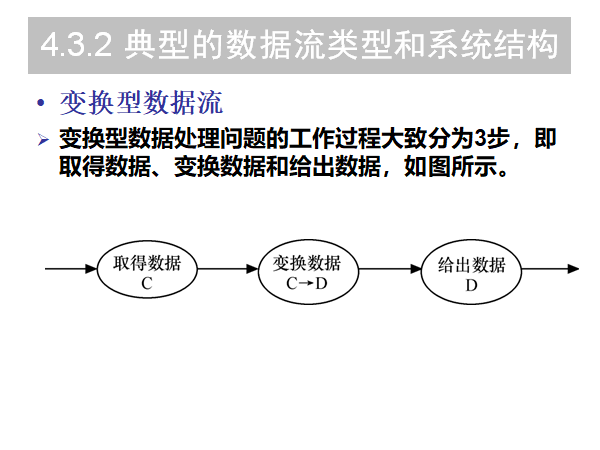
2）需传递控制参数——控制耦合

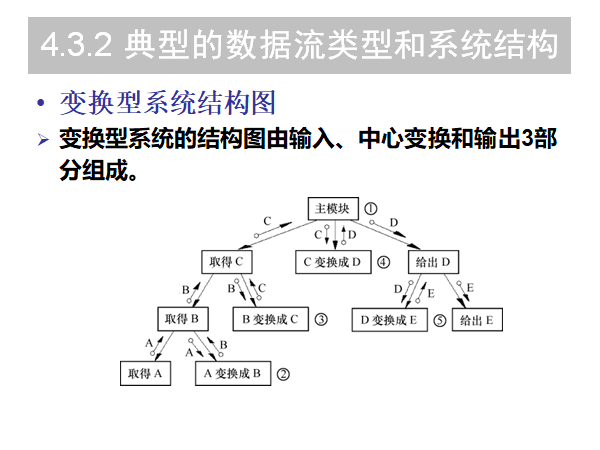
3）未用部分调入内存，影响效率

* + - 1. (3)时间内聚
         1. 时间内聚模块大多为多功能模块，但模块的各个功能的执行与时间有关，通常要求所有功能必须在同一时间段内执行。
      2. (4)过程内聚
         1. 如果一个模块内的处理是相关的，而且必须以特定次序执行，则是过程内聚。
      3. (5)通信内聚
         1. 如果一个模块内各功能部分都使用了相同的输入数据，或者产生了相同的输出数据，则称之为通信内聚模块。
      4. (6)信息内聚
         1. 这种模块完成多个功能，各个功能都在同一数据结构上操作，每一项功能都有唯一的入口点。
      5. (7)功能内聚
         1. 一个模块中各个部分都是完成某一具体功能必不可少的组成部分，或者说该模块中所有部分都是为了完成一项具体功能而协同工作，紧密联系，不可分割的。则称该模块为功能内聚模块。
         2. 优点：容易修改和维护
      6. (8)顺序内聚
         1. 一个模块中的处理元素和同一功能密切相关，而且这些处理必须顺序执行，则称为顺序内聚。
  1. 3.软件设计的阶段与任务
     1. 从工程管理的角度，可以将软件设计分为概要设计阶段和详细设计阶段。
     2. 从技术的角度，传统的结构化方法将软件设计划分为体系结构设计、数据设计、接口设计和过程设计4部分；
     3. 面向对象方法则将软件设计划分为体系结构设计、类设计/数据设计、接口设计和构件级设计4部分。
  2. 4.模块结构及表示
     1. 模块的结构
        1. 模块结构最普通的形式就是树状结构和网状结构，如图所示。

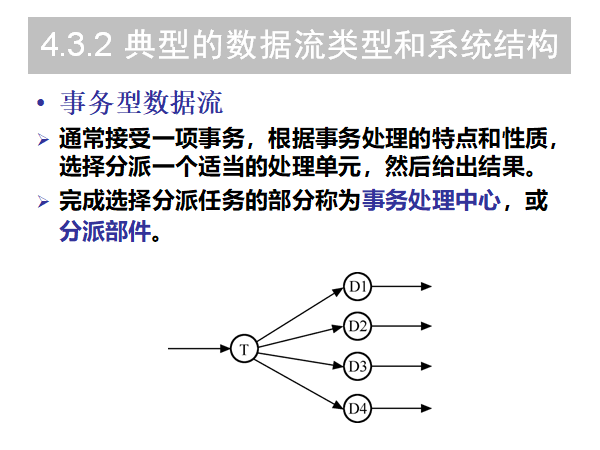


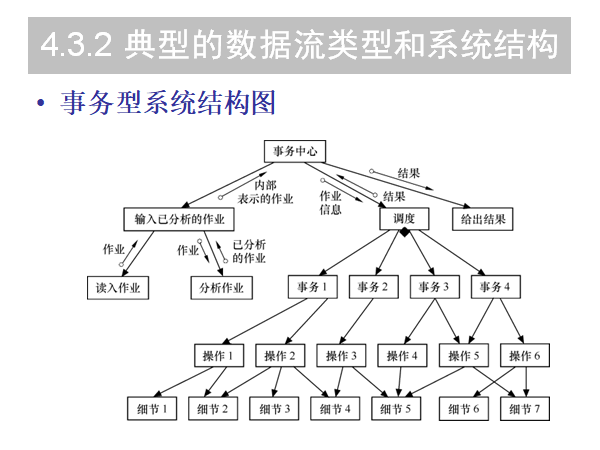
* + 1. 结构图（structure chart，SC）是精确表达模块结构的图形表示工具。
       1. (1) 模块的调用关系和接口：在结构图中，两个模块之间用单向箭头连接。
       2. (2) 模块间的信息传递：当一个模块调用另一个模块时，调用模块把数据或控制信息传送给被调用模块，以使被调用模块能够运行。
       3. (3) 条件调用和循环调用 ：当模块A有条件地调用另一个模块B时，在模块A的箭头尾部标以一个菱形符号；当一个模块A反复地调用模块C和模块D时，在调用箭头尾部则标以一个弧形符号。
       4. (4) 结构图的形态特征。在图中，上级模块调用下级模块，它们之间存在主从关系。
  1. 5.典型的数据流类型和系统结构
     1. (1)变换型数据流

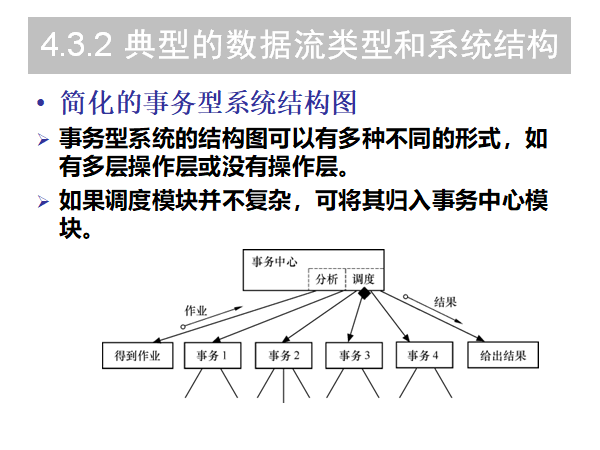




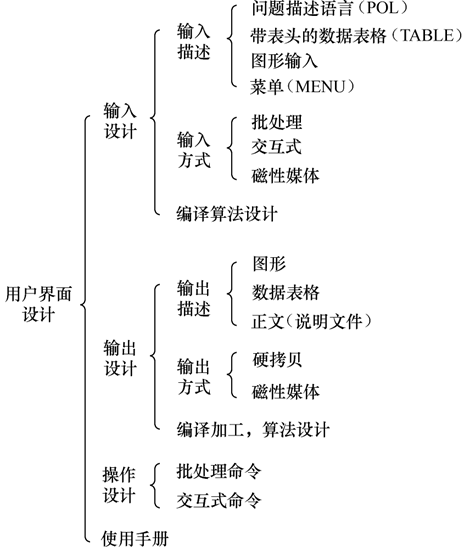
* + - 1. 变换分析方法的步骤
         1. (1) 重画数据流图。在需求分析阶段得到的数据流图侧重于描述系统如何加工数据，而重画数据流图的出发点是描述系统中的数据是如何流动的。
         2. (2)在数据流图上区分系统的逻辑输入、逻辑输出和中心变换部分。
         3. (3) 进行一级分解，设计系统模块结构的顶层和第一层。自顶向下设计的关键是找出系统树形结构图的根或顶层模块。
         4. (4) 进行二级分解，设计中、下层模块。
    1. (2)事务型数据流



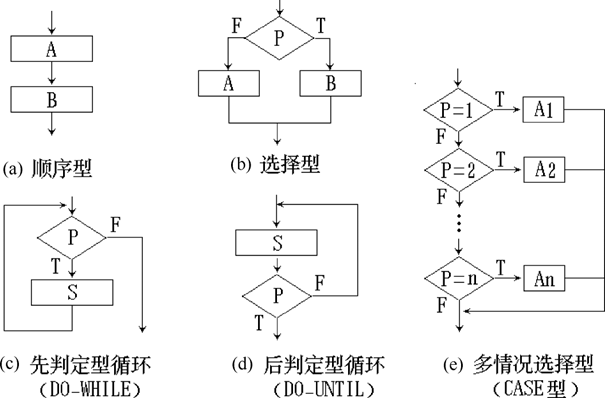




* + - 1. 事务分析方法的步骤
         1. (1) 识别事务源。利用数据流图和数据词典，从问题定义和需求分析的结果中，找出各种需要处理的事务。
         2. (2) 规定适当的事务型结构。在确定了该数据流图具有事务型特征之后，根据模块划分理论，建立适当的事务型结构。
         3. (3) 识别各种事务和它们定义的操作。
         4. (4) 注意利用公用模块。
         5. (5) 建立事务处理模块。对每一事务，或对联系密切的一组事务，建立一个事务处理模块。
         6. (6) 对事务处理模块规定它们全部的下层操作模块。
         7. (7) 对操作模块规定它们的全部细节模块。
    1. (3)软件模块结构的改进方法
       1. (1) 模块功能的完善化。(2) 消除重复功能，改善软件结构。(3) 模块的作用范围应在控制范围之内。(4) 尽可能减少高扇出结构，随着深度增大扇入(5) 避免或减少使用病态连接。(6) 模块的大小要适中。
  1. 6.接口设计
     1. (1)概述：
        1. 接口设计的依据是数据流图中的自动化系统边界。
        2. 接口设计主要包括3个方面：模块或软件构件间的接口设计；软件与其他软硬件系统之间的接口设计；软件与人（用户）之间的交互设计。
        3. 人机交互（用户）界面是人机交互的主要方式
     2. (2)人机交互界面
        1. 用户界面应具备的特性
           1. 可使用性：包括使用简单、界面一致、拥有HELP帮助功能、快速的系统响应和低的系统成本、具有容错能力等。
           2. 灵活性：考虑到用户的特点、能力和知识水平，应当使用户接口满足不同用户的要求。
           3. 可靠性：用户界面的可靠性是指无故障使用的间隔时间。用户界面应能保证用户正确、可靠地使用系统，保证有关程序和数据的安全性。
        2. 用户类型
           1. 外行型：以前从未使用过计算机系统的用户。
           2. 初学型：尽管对新的系统不熟悉，但对计算机还有一些使用经验的用户。
           3. 熟练型：对一个系统有相当多的经验，能够熟练操作的用户。
           4. 专家型：这一类用户了解系统内部的构造，有关于系统工作机制的专业知识，具有维护和修改基本系统的能力。专家型需要为他们提供能够修改和扩充系统能力的复杂界面。
        3. 界面设计类型



* + - * 1. (1) 使用的难易程度：对于没有经验的用户，该界面使用的难度有多大。
        2. (2) 学习的难易程度：学习该界面的命令和功能的难度有多大。
        3. (3) 操作速度：在完成一个指定操作时，该界面在操作步骤、击键和反应时间等方面效率有多高。
        4. (4) 复杂程度：该界面提供了什么功能、能否用新的方式组合这些功能以增强界面的功能。
        5. (5) 控制：人机交互时，是由计算机还是由人发起和控制对话。
        6. (6) 开发的难易程度：该界面设计是否有难度、开发工作量有多大。
      1. 设计详细的交互 :人机交互的设计有若干准则，包括以下内容：
         1. (1) 一致性。采用一致的术语、一致的步骤和一致的活动。
         2. (2) 操作步骤少。使击键或点击鼠标的次数减到最少，甚至要减少做某些事所需的下拉菜单的距离。
         3. (3) 不要“哑播放”。
         4. (4) 提供Undo功能。
         5. (5) 减少人脑的记忆负担。不应该要求人从一个窗口中记住某些信息，然后在另一个窗口中使用。
         6. (6) 提高学习效率。为高级特性提供联机帮助，以便用户在需要时容易找到。
    1. (3)过程设计
       1. 结构程序设计的主要原则
          1. (1)使用语言中的顺序、选择、重复等有限的基本控制结构表示程序逻辑。(2)选用的控制结构只准许有一个入口和一个出口。(3)程序语句组成容易识别的块（Block），每块只有一个入口和一个出口。(4)复杂结构应该用基本控制结构进行组合嵌套来实现。(5)语言中没有的控制结构，可用一段等价的程序段模拟， 但要求该程序段在整个系统中应前后一致。(6) 严格控制GOTO语句(7) 在程序设计过程中，尽量采用自顶向下(Top－Down)、逐步细化(Stepwise Refinement)的原则，由粗到细，一步步展开。
       2. 程序流程图的基本控制结构



* + - * 1. (1) 顺序型：几个连续的加工步骤依次排列构成。
        2. (2) 选择型：由某个逻辑判断式的取值决定选择两个加工中的一个。
        3. (3) 先判定（while）型循环：在循环控制条件成立时，重复执行特定的加工。
        4. (4) 后判定（until）型循环：重复执行某些特定的加工，直至控制条件成立。
        5. (5) 多情况（case）型选择：列举多种加工情况，根据控制变量的取值，选择执行其一。