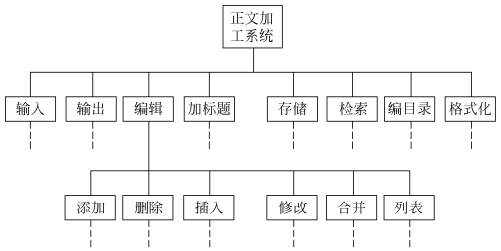
4.4 软件结构的设计工具

1.层次图

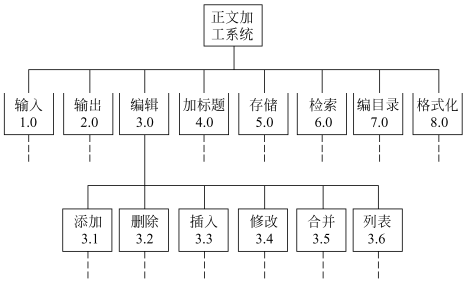
- 层次图（H图）广泛用来描绘软件的层次结构。层次图中的一个矩形框代表一个模块，方框间的连线表示调用关系。
- 注意区分：层次方框图与层次图。
- 层次图适合于在自顶向下软件设计的过程中使用。它表明各个功能的隶属关系，自顶向下逐层分解得到的。

层次图例子



正文加工系统的层次图

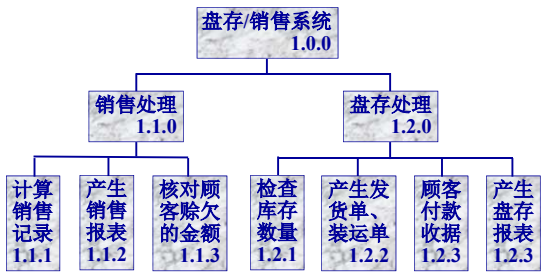
- 带编号的层次图：除了用一个名字标识每个框完成的功能，还用一个编号记录它所在的层次及在该层次的位置。



2. HIPO图

- HIPO是1976年由IBM公司提出来的，全称是“层次图加输入/处理/输出图”。
  - ✓HIPO图采用层次图（H图）给出程序的层次关系，采用IPO图来描述程序逻辑。
  - ✓和H图中每个方框相对应，应该有一张IPO图描绘这个方框代表的模块的处理过程。
  - ✓HIPO图中的每张IPO图内都应该明显地标出它所描绘的模块在H图中的编号，以便追踪了解这个模块在软件结构中的位置。

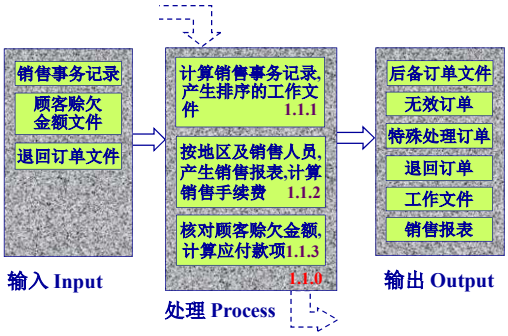
HIPO图例子



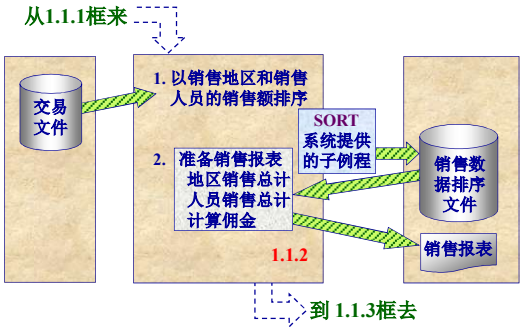
盘存/销售系统的层次图

编号	说　　明
1.0.0	销售/盘存处理框图
1.1.0	顾客订单检查, 核对顾客赊欠金额, 产生销售报表
1.1.1	用工作文件的盘存项目号, 对顾客订单进行核对和排序
1.1.2	以地区和人员为单位, 编制销售报表, 计算销售佣金
1.1.3	检验顾客赊欠金额, 计算折扣, 确定支付项目
1.2.0	处理顾客盘存管理报表, 顾客付款收帐, 处理发货、包装、托运
……	

对应H图上1.1.0框的概要IPO图

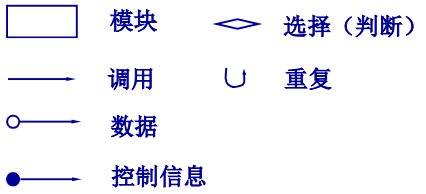


对应于H图1.1.2框的详细IPO图



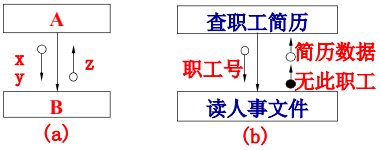
3. 结构图

- Yourdon提出的结构图是进行软件结构设计的另一个有力工具。
- 结构图中一个方框代表一个模块，框内注明模块的名字或主要功能；方框之间的箭头(或直线)表示模块的调用关系。
- 结构图并不严格表示模块的调用次序，并不指明什么时候调用下层模块。



结构图的基本符号

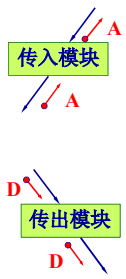
- 数据：模块之间传送的数据用带空心圆的箭头表示，并在旁边标上数据名。
- 控制信息：控制信息与数据的主要区别是前者只反映数据的某种状态。



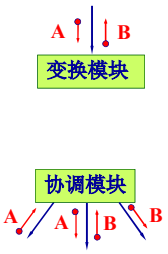
模块间的数据传递

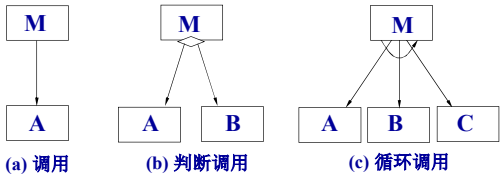
结构图中的模块

- 传入模块－从下属模块取得数据，经过某些处理，再将其传送给上级模块。它传送的数据流叫做逻辑输入数据流。
- 传出模块－从上级模块获得数据，进行某些处理，再将其传送给下属模块。它传送的数据流叫做逻辑输出数据流。



- 变换模块－它从上级模块取得数据，进行特定的处理，转换成其它形式，再传送回上级模块。
- 协调模块－对所有下属模块进行协调和管理的模块。





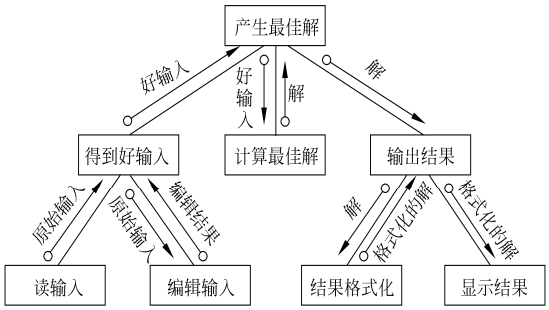
(a) 调用　(b) 判断调用　(c) 循环调用

判定为真时调用A，为假时调用B

模块M循环调用模块A、B、C

模块调用示例

结构图例子



4.5 结构化设计方法

- 结构化设计（SD）的基本思想:将系统设计成由相对独立、功能单一的模块组成的结构。
- 结构化设计是基于模块化、自顶向下细化、结构化程序设计等程序设计技术基础上发展起来的。
- 结构化设计是一种面向数据流的设计方法，可以与结构化分析方法（SA）衔接。

- 该方法实施的要点是：
  1) 首先精化数据流图。根据需求规格说明，研究与审查数据流加工过程，解决发现的问题。
  2) 然后根据数据流图确定数据处理的类型，针对不同类型（变换型或事务型）进行变换分析或事务分析处理。
  3) 由数据流图推导出系统的初始结构图。
  4) 利用启发式原则，改进系统初始结构图。
  5) 修改和补充数据字典。
  6) 制定测试计划。

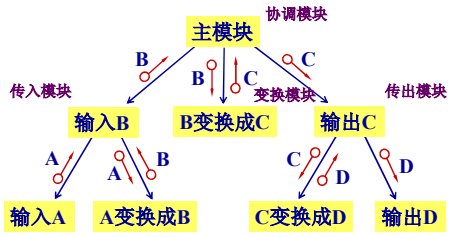
数据处理的类型

- 在需求分析阶段，SA方法产生数据流图；在软件设计阶段，SD方法将数据流图转换成程序结构图。
- 数据处理即为在DFD中从系统的输入数据流到系统的输出数据流所经历的一连串连续变换。数据处理的类型分为变换流与事务流。

变换流

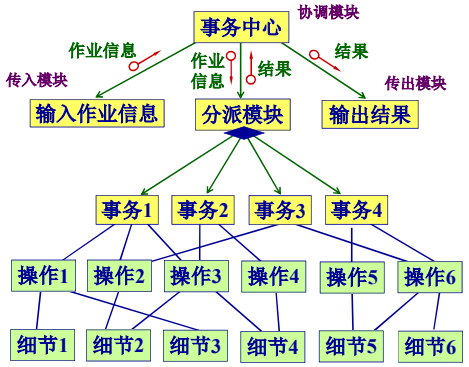
- 数据沿着输入通路进入系统，经过一系列数据变换，将数据的外部形式转换成对应的内部表示，然后通过变换中心（也称主加工）处理，再沿着输出通路转换成外部形式离开系统。具有这种特性的数据流称为变换流。
- 变换流型DFD可以分成：
  输入＋变换中心（主加工）＋输出

- 用结构图表示变换流型的软件结构。传入模块、变换模块、传出模块分别对应于输入、变换中心和输出三部分组成。



事务流

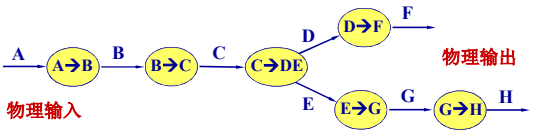
- 数据沿着输入通路到达一个事务中心，事务中心根据输入数据（即事务）的类型在若干个动作（称为活动流）中选择一个来执行，这种数据流也称为事务流。
- 事务流有明显的事务中心，各活动流以事务中心为起点呈辐射状流出。
- 在事务流型结构图中，事务中心模块按所接受的事务的类型，选择某一事务（活动流）处理模块执行。各事务处理模块并列。



变换分析

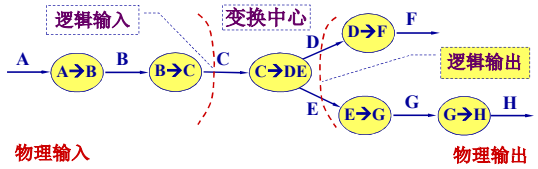
- 变换分析从变换流的数据流图导出系统结构图.
- 步骤
  1.重画数据流图；
  2.区分有效（逻辑）输入、有效（逻辑）输出和变换中心部分；
  3.进行一级分解，设计模块结构的顶层和第一层模块；
  4.进行二级分解，设计输入、输出和中心变换部分的中、下层模块。

1. 重画数据流图（平铺）



- 为了建立系统结构，将数据流图平铺开来，物理输入画在左侧，物理输出画在右侧。
- 如果一个外部实体既是物理输入又是物理输出，则两侧都要画出它。

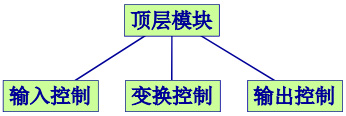
2. 确定逻辑输入、逻辑输出和变换中心部分

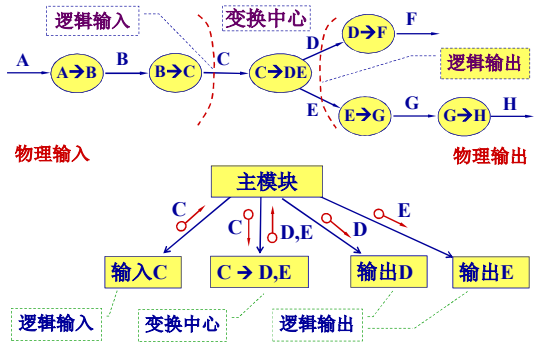


- 变换中心是程序的核心功能，它的输入是逻辑输入，它的输出为逻辑输出。

3. 第一级分解：设计模块结构的顶层和第一层

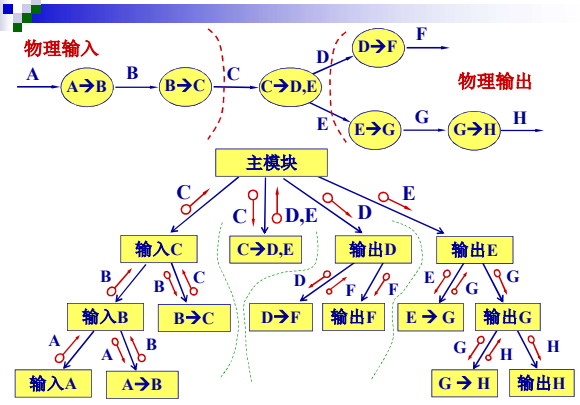
- 顶层模块：其功能就是整个系统的功能；
- 输入控制模块：接收所有的输入数据；
- 变换控制模块：实现输入到输出的变换；
- 输出控制模块：产生所有的输出数据。





4. 第二级分解：设计中、下层模块

- ✓ 输入控制模块的分解：从变换中心的边界开始，沿着各输入通路，把输入通路上的每个加工映射成输入控制模块的一个低层模块。
- ✓ 输出控制模块的分解：从变换中心的边界开始，沿着各输出通路，把输出通路上的每个加工映射成输出控制模块的一个低层模块。
- ✓ 变换控制模块的分解：变换控制模块通常没有通用的分解方法，应根据数据流图中变换部分的实际情况进行设计。

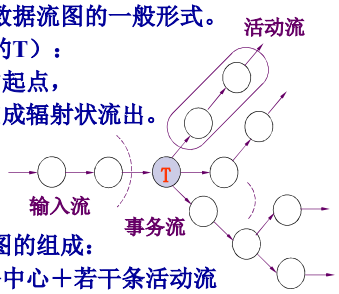


事务分析

- 事务分析是从事务流的数据流图导出系统结构图。
- 步骤：
  1. 确定事务中心和每条活动流的流特性
  2. 将事务流型数据流图映射成高层的系统结构
  3. 进一步分解

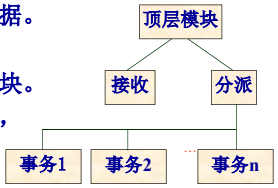
1. 确定事务中心和各活动流的流特性

- 右图为事务流的数据流图的一般形式。
- 事务中心（图中的T）：
  位于活动流的起点，
  活动流从该点成辐射状流出。
- 活动流：
  可为变换流
  可为事务流
- 事务流的数据流图的组成：
  输入流＋事务中心＋若干条活动流



2. 将事务流型DFD映射成高层系统结构

- 右图为事务流的数据流图的高层结构形式。
- 顶层模块：其功能就是整个系统的功能。
- 接收模块：接收输入数据。
- 分派模块：调度模块，
  控制下层的所有活动模块。
- 事务模块：对应活动流，
  是该活动流映射成的。



3. 进一步分解

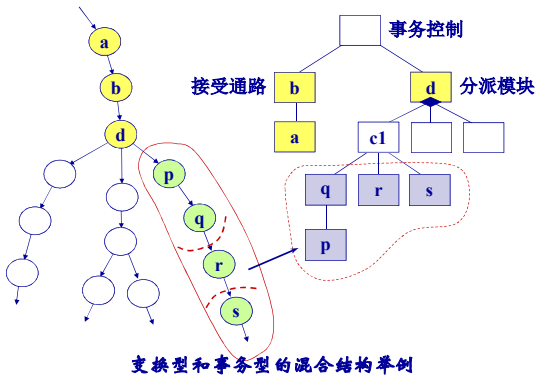
- 接收模块：类同于变换分析中输入控制模块的分解。
- 活动流模块：根据其流特性（变换流或事务流）进一步采用变换分析或事务分析进行分解。

模块设计的原则

- 在选择模块设计的次序时，必须对一个模块的全部直接下属模块都设计完成之后，才能转向另一个模块的下层模块的设计。
- 使用“黑盒”技术: 在设计当前模块时，先把该模块的所有下层模块定义成“黑盒”，在设计中利用它们时暂不考虑其内部结构和实现。在这一步定义好的“黑盒”，在下一步就可以对它们进行设计和加工。最后，全部“黑盒”的内容和结构应完全被确定。

- 在设计下层模块时，应考虑模块的耦合和内聚问题，以提高初始结构图质量。
- 模块划分时，一个模块的直接下属模块一般在 5 个左右。如果直接下属模块超过 10 个，可设立中间层次。
- 如果出现以下情况，就停止模块分解：
  - ✓ 模块不能再细分为明显的子任务；
  - ✓ 分解成用户提供的模块或库函数；
  - ✓ 模块接口是输入输出设备传送的信息；
  - ✓ 模块不宜再分解得过小。

- 变换分析是软件系统结构设计的主要方法。
- 一个大型的软件系统通常是变换型结构和事务型结构的混合结构。利用以变换分析为主，事务分析为辅的方式进行软件结构设计。



变换型和事务型的混合结构举例

4.6 软件设计的评价

- 软件设计既是过程又是模型。
- 设计过程是一系列的迭代步骤，使设计人员能够描述目标系统的各个侧面。
- 设计模型首先描述目标系统的整体架构，然后逐步细化架构得到构造每个细节的指导原则，从而得到系统的一系列不同的视图。
- 良好的设计原则可为设计过程导航。

- 衡量设计过程的技术原则：

①设计必须实现分析模型中描述的所有显式需求，必须满足用户希望的所有隐式需求。

②对于开发者和未来的维护者而言，设计必须是可读的、可理解的，使得将来易于编程、易于测试、易于维护。

②设计应该给出软件的全貌，包括从实现角度可看到的数据、功能、行为。

- 衡量设计模型的技术原则

①设计模型应该是一个分层结构。该结构：
- ✓ 使用可识别的设计模式搭建系统结构。
- ✓ 用显示良好设计特征的构件构成。
- ✓ 可以用演化的方式实现。

②设计应当模块化。

③设计应当包含数据、体系结构、接口和构件（模块）的清晰的视图。

④ 设计应当根据将要实现的对象和数据模式导出合适的数据结构。

⑤ 设计应当建立具有独立功能特征的构件。

⑥ 设计应当建立能够降低模块与外部环境之间复杂连接的接口。

⑦ 设计模型应当通过使用软件需求信息所驱动的可重复的方法导出。

小结

- 软件设计的基本目标是确定问题的解决方案，并且用较概括抽象的方式表达出来。
- 软件设计细化为总体设计与详细设计两个阶段，其好处在于可以在软件开发的早期从全局高度对软件结构进行优化。
- 总体设计阶段的主要任务包括确定系统的物理模型和确定软件的体系结构。

- 设计原理和启发规则是进行软件结构设计的主要手段。模块独立性是应该遵循的最主要的原理。
- 层次图、HIPO图、结构图是描绘软件结构的常用工具。
- 结构化设计方法是一种面向数据流的设计方法，并基于模块化、自顶向下细化、结构化程序设计等技术，并可与结构化分析方法衔接。
- 软件设计既是过程又是模型，应采用合适标准评价软件设计模型与设计过程。