第 3 章

CHAPTER

# 结构化分析

结构化分析是面向数据流进行需求分析的方法。需求分析的主要任务是将用户的需求变为软件的功能和性能描述。为了将软件的功能和性能描述清楚，系统分析人员需要用文字描述、图形符号来详细说明软件必须要做什么。本章介绍结构化分析方法常用的工具并结合实例介绍结构化分析方法的具体应用。

**本章学习目标：**

- 理解结构化分析的思想和过程。
- 掌握结构化分析方法：数据流图 DFD，数据字典 DD，加工说明，实体关系图。
- 理解需求分析过程是遵循分层、抽象、分解的思想原则的。

## 3.1 概　　述

结构化分析方法是在 20 世纪 70 年代末被提出的，一直被广泛应用。结构化分析方法用抽象模型的概念，按照软件内部数据传递和变换的关系，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元素为止。

结构化分析方法使用的主要工具有数据流图、数据字典、IPO 处理描述、判定表与判定树等。

结构化开发方法（Structured Developing Method）是现有的软件开发方法中最成熟、应用十分广泛的方法，主要特点是快速、自然和方便。结构化开发方法由结构化分析（SA）方法、结构化设计（SD）方法及结构化程序设计（SP）方法构成。

### 3.1.1 结构化分析思想

结构化分析（Structured Analysis，SA）是面向数据流的需求分析方法，是 20 世纪 70 年代由 Yourdon、Constaintine 及 DeMarco 等人提出和发展，并得到了广泛的应用。

结构化分析方法的基本思想是“分解”和“抽象”。

对于一个复杂的问题，由于人的理解力、记忆力均有限，所以不可能触及问题的所有方面以及所有细节。为了将复杂性降低到可以掌握的程度，可以把大问题分解成若干小问题，然后分别解决，这就是“分解”。分解可以分层进行，即先考虑问题最本质的属性，暂把细节略去，以后再逐层添加细节，直至涉及最详细的内容。这种用最本质的属性表示一个子系统的方法就是“抽象”。

结构化分析方法的基本思路如图 3-1 所示，自顶向下的过程是分解的过程，自底向上的过程是抽象的过程。结构化方法就是采用这种自顶向下逐层分解的思想进行分析建模的，自顶向下逐层分解充分体现了分解和抽象的原则。随着分解层次的增加，抽象的级别越来越低，也越来越接近问题的解(算法和数据结构)。顶层抽象地描述了整个系统，底层具体地画出了系统的每一个细节，而中间层是从抽象到具体的逐层过渡。

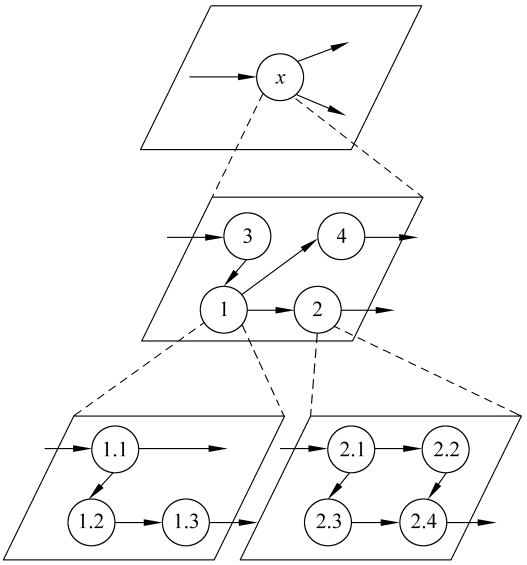


图 3-1　分解与抽象

## 3.1.2　结构化分析过程

要对一个系统进行结构化分析，首先要明确这一阶段的任务是要搞清楚“做什么”。因此，要对现行系统有一个了解，在此基础上修改要变化的部分而形成新系统。结构化分析的过程可以分为如下 4 个步骤。

**1. 建立当前系统的物理模型**

当前系统是指目前正在运行的系统，也是需要改进的系统。当前系统可能是正在计算机上运行的软件系统，也可能是人工的处理系统。了解当前系统的工作过程，对当前系统进行详细调查并收集资料，将看到的、听到的、收集到的信息和情况用图形或文字描述出来，也就是用一个模型来反映自己对当前系统的理解。

系统的物理模型就是现实环境的真实写照，即将当前系统用系统流程图描述出来，这样的表达与当前系统完全对应，因此用户容易理解。

**2. 抽象出当前系统的逻辑模型**

在系统分析中需要建立功能模型时，可以通过上述方法建立物理模型，它反映了系统当前“怎么做”的具体实现。要构造新的逻辑模型就要去掉物理模型中非本质的因素，抽取出本质的因素。所谓本质的因素是指系统固有的、不依赖运行环境变化而变化的因素，任何实现均这样做。非本质的因素不是固有的、随环境不同而不同，随实现不同而不同。运用抽象原则对物理模型进行分析，区别本质因素和非本质因素，去掉非本质因素，形成当前系统的逻辑模型。这种逻辑模型反映了当前系统“做什么”的功能。

分析系统的“具体模型”，抽象出其本质的因素，排除次要因素，获得用数据流图描述的当前系统的“逻辑模型”。

**3. 建立目标系统的逻辑模型**

目标系统是指待开发的新系统。有了当前系统的逻辑模型后，就要将目标系统与当前系统的逻辑进行分析，比较其差别，即在当前系统的基础上决定变化的范围，把那些要改变的部分找出来，将变化的部分抽象为一个加工，这个加工的外部环境及输入输出就确定了；然后对变化的部分重新分解，分析人员根据自己的经验，采用自顶向下逐步求精的分析策略，逐步确定变化部分的内部结构，从而建立目标系统的逻辑模型。

分析目标系统与当前系统逻辑上的差别，从而进一步明确目标系统“做什么”，建立目标系统的“逻辑模型”，得到修改后的数据流图。

**4. 进一步补充和优化**

目标系统的逻辑模型只是一个主体，为了完整地描述目标系统，还要做一些补充。补充的内容包括它所处的应用环境及它与外界环境的相互联系，说明目标系统的人机界面，说明至今尚未详细考虑的环节，如出错处理、输入输出格式、存储容量和响应时间等性能要求。

### 3.1.3　结构化模型的描述形式

结构化分析实质上是一种创建模型的活动。通过需求分析而建立的模型必须达到下述的 3 个基本目标。

（1）描述用户的需求。

（2）为软件设计工作奠定基础。

（3）定义一组需求，一旦开发出软件产品之后，就可以用这组需求作为标准来验收该产品。

为了达到上述这些目标，在结构化分析过程中导出的分析模型的形式，如图 3-2 所示。分析模型的核心是“数据字典”，它描述软件使用或产生的所有数据对象。围绕着这个核心有

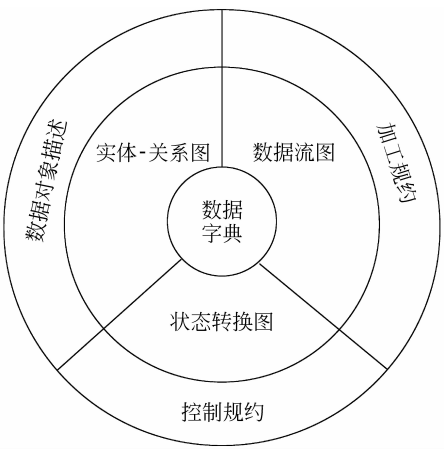


图 3-2　结构化分析模型的结构

3 种不同的图以及相应的规约或描述。

实体-关系图描述数据字典中数据对象之间的关系，它是用来进行数据建模活动的图形，图中出现的每个数据对象的属性都可以用“数据对象描述”来描述，通常存放在数据字典中。

数据流图用于软件系统的功能建模，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，这些对数据流的加工实际上反映了系统的某种功能或子功能。数据流图中的数据流、文件、数据项、加工都应在数据字典中描述。加工规约是对数据流图中的加工的说明，在结构化方法中用加工的“小说明”作为加工规约。

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。控制规约用来描述软件控制方面的附加信息。

早期的结构化分析方法的分析结果包括一套分层的数据流图、一本数据字典（包括实体-关系图）、一组加工规约以及其他补充材料（如非功能性需求等）。

## 3.2 数 据 流 图

数据流图也被称为数据流程图（Data Flow Diagram，DFD），是从数据传递和加工角度，以图形方式来表达系统的逻辑功能、数据在系统内部的逻辑流向和逻辑变换过程，是结构化系统分析方法的主要表达工具，是用于表示软件模型的一种图示方法。

### 3.2.1 数据流图的基本成分

数据流图的基本图形元素有 4 种，即数据流、加工、文件和数据的源点/终点。数据流、加工和文件用于构建软件系统内部的数据处理模型，数据的源点/终点表示存在于系统之外的对象，有助于理解系统数据的来源和去向。数据流图的基本图形元素如图 3-3 所示。

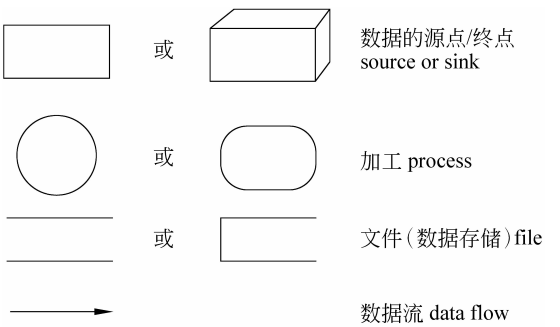


图 3-3 数据流图的基本图形元素

**1. 数据流**

数据流是数据在系统内传播的路径，因此由一组成分固定的数据组成。如某供销系统中，订货单由配件号、配件名、规格、数量、顾客名、地址等数据项组成。

由于数据流是流动中的数据，所以必须有流向，它的流动方向一般有以下几种情况：从一个加工流向另一个加工；从加工流向文件（写文件）；从文件流向加工（读文件）；从数据的源点流向加工；从加工流向数据的终点。

在数据流图中的每个数据流都要用一个定义明确的名字标识。一般从数据流组成成分或实际具体含义的角度给各个数据流命名，流向文件或从文件流出的数据流不必命名，因为这时有文件名就可以说明问题了。

但是需要特别注意的是，在数据流图中描述的是数据流，而不是控制流，可以通过查看流中包含的信息加以区分。

**2. 数据的源点/终点**

任何一个系统的边界被定义后，就有系统内外之分，一个系统总会与系统外部的实体产生联系，这种联系的重要形式就是数据。数据的源点和终点是软件系统外部环境中的实体，包括人员、组织或其他软件系统，统称为外部实体。

外部实体是为了帮助理解系统界面而引入的，一般只出现在数据流图的顶层图中，表示系统中数据的来源和去处。有时为了增加数据流图的清晰性，防止数据流的箭头线太长，在一张图上可以重复画出同名的数据的源点/终点。

**3. 加工**

加工也被称为数据处理，是对数据执行的某种操作或变换，把输入数据流加工成输出数据流。每个加工都应有一个定义明确的名字来标识，其命名采用用户习惯的且反映加工含义的名字，并加以编号来说明这个加工在层次分解中的位置。

**4. 文件**

文件用于存放数据。通常一个流入加工的数据流经过加工处理后就消失了，而它的某些数据可能被加工成输出数据流，流向其他的加工或数据的终点。除此之外，在软件系统中还常常要把某些信息保存下来供以后使用，此时可使用文件。

每个文件都要用一个定义明确的名字标识。可以有数据流流入文件，表示写文件；也可以有数据流从文件流出，表示读文件；也可以用双向箭头的数据流指向文件，表示对文件的修改。数据流图中的文件在具体实现时可以用文件系统实现，也可以用数据库系统来实现。文件的存储介质也可以是磁盘、磁带或其他存储介质。

在数据流图中，一个加工可以有多个输入数据流，也可以有多个输出数据流，此时可以加上一些扩充符号来描述多个数据流之间的关系，数据流图中的扩充符号如图 3-4 所示。图 3-4 中的 A、B、C 分别表示三个数据流，它们之间的关系可以通过星号、加号、异或号来表示。

1）星号（*）

星号（*）表示数据流之间存在“与”关系。如果是输入流则表示所有输入数据流全部到达后才能进行加工处理；如果是输出流则表示加工结束后将同时产生所有的输出数据流。

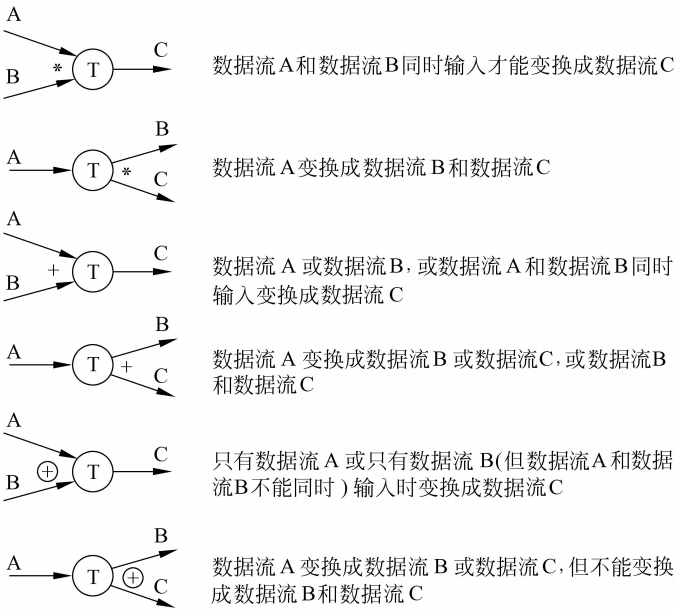


图 3-4 数据流图中的扩充符号

2) 加号(+)

加号(+)表示数据流之间存在“或”关系。如果是输入流则表示其中任何一个输入数据流到达后就能进行加工处理;如果是输出流则表示加工处理的结果是至少产生其中一个输出数据流。

3) 异或号(⊕)

异或号(⊕)表示数据流之间存在“异或”关系。

### 3.2.2 数据流图绘制方法

数据流图的基本要点是描述“做什么”,而不考虑“怎么做”。通常数据流图要忽略出错处理,也不包括诸如打开文件和关闭文件之类的内部处理。绘制数据流图的原则是由外向里,自顶向下去模拟问题的处理过程,通过一系列的分解步骤,逐步求精地表达出整个系统功能的内部关系。

**1. 绘制步骤**

(1) 找出系统的输入和输出

在分析刚开始时,先不用考虑系统究竟包括哪些功能,首先要了解“系统从外界接收什么数据”、“系统向外界送出什么数据”,以确定系统的范围和边界。系统从外界源点接收的数据流为输入数据流,系统送到外界终点的数据流为输出数据流,这样就可以画出软件系统顶层的数据流图。

(2) 画系统的内部

这一步骤将系统的输入和输出数据流用一连串加工连接起来,一般可以从输入端逐

步画到输出端，也可以反过来从输出端回溯到输入端。在数据流的组成或值发生变化的地方画上一个加工，它的作用就是实现这一变化，在需要暂时存储静态数据的地方画出文件。很明显，这与分析用户的业务流程、信息流程和功能需求是相适应的。

(3) 画加工的内部

同样用由外向内的方式继续分析每个加工的内部。如果加工的内部还有一些数据流，则可将这个加工用几个子加工代替，并在子加工之间画出这些数据流。

为了表达数据处理过程的数据加工情况，用一个数据流图是不够的。一个复杂的软件系统可能涉及上百个加工和数据流，甚至更多。如果将它们画在一张图上，则会十分复杂，不易阅读，也不易理解。为了表达较为复杂问题的数据处理过程，要按照该问题的层次结构进行逐步分解，并以一套分层的数据流图来反映这种结构关系。

**2. 注意事项**

(1) 命名

在绘制数据流图时，无论数据流、数据存储还是加工的命名都要合适，易于理解其含义。数据流的名字代表整个数据流的内容，不仅仅是它的某些数据成分。命名时不要使用抽象含义的名字，比如“数据”、“信息”等。加工的名字要反映其处理的功能，不能使用“操作”、“处理”这些笼统的名字。此外，需要注意，一个加工的输出数据流，不应与输入数据流同名，即使他们的组成完全相同。

(2) 层次结构

分层数据流图的顶层只有一张图，其中只有一个加工，代表整个软件系统，该加工描述了软件系统与外部实体之间的数据流，称为顶层图。顶层图中的加工经分解后得到的图称为 0 层图，也只有一张。处于分层数据流最底层的图称为底层图，在底层图中，所有的加工都不再进行分解。分层数据流图中的其他图称为中间层图，其中至少有一个加工被分解成一张子图。在整套分层数据流图中，凡是不再分解成子图的加工都称为基本加工。

George Miller 指出，人们在一段时间内的短期记忆似乎会限制在“7 加减 2”件事情之内。根据自顶向下逐层分解的思想将数据流图画成层次结构，每个层次画在独立的数据流图中，加工个数可大致控制在 5～9 的范围中。

(3) 图和加工的编号

如果一张数据流图中的某个加工点要分解成另一张数据流图，则上层图为父图，直接下层图称为子图，父图和子图上的所有加工都应编号。子图的编号是父图中相应加工编号的扩充，子图上加工的编号是由父图号、小数点和子图的局部号组成的。

① 顶层图只有一张，图中的加工也只有一个，所以不必为其编号。

② 0 层图只有一张，图中的加工号分别是 1、2、3、……

③ 若父图中的加工号为 $x$ 的加工被分解为某一子图，则该子图号记为图 $x$。

④ 子图中的加工号是由图号、小数点和序号组成，若父图中加工号为 $x$ 的加工分解成某一子图，则该子图中的加工编号分别为 $x.1$、$x.2$、$x.3$、……

(4) 在系统分析中要注意区别物流和数据流

数据流反映的是能用计算机处理的数据，并不是实物。因此在目标系统的数据流图

上不要绘制物流。

(5) 每个加工至少有一个输入数据流和一个输出数据流

每个加工至少有一个输入数据流和一个输出数据流,反映该加工数据的来源和加工的结果。允许一个加工有多条数据流流向另一个加工,也允许一个加工有两条相同的输出数据流流向不同的加工。

(6) 数据存储的读取

在整套数据流图中,每个数据存储都必须既有读的数据流,又有写的数据流。但是在某张子图中,可能只有读没有写,或者只有写没有读。

(7) 数据守恒

保持数据守恒,也就是说,一个加工的所有输出数据流中的数据必须是能从该加工的输出流中直接获得,或者通过加工能产生的数据。

(8) 父图和子图的平衡

在用数据流图表示系统的数据流向时,一般都要用到父图和子图来描述不同的层次。这时要注意父图和子图的平衡。子图的输入数据流、输出数据流同父图相应加工的输入数据流、输出数据流必须一致,父图和子图要平衡。

值得注意的是,如果父图中的一个输入(输出)数据流对应于子图中的几个输入(输出)数据流,而子图中组成这些数据流的数据项的全体正好是父图中的这一个数据流,那么他们仍然算是平衡的。

(9) 局部数据存储

在分层处理的过程中,当某些数据流图中的数据存储不是父图中相应加工的外部接口,而只是本图中某些加工之间的数据接口时,则称这些数据存储为局部数据存储。对于一个局部数据存储,只有当它作为某些加工的数据接口或某个加工特定的输入或输出时,才把它绘制出来,这样有助于实现信息隐蔽。

(10) 合理分解

数据流图作为以后设计和与用户交流的基础,其易理解性极为重要。因此在外层中要注意合理分解,要把一个加工分解成几个功能相对独立的子加工,这样可以减少加工之间输入数据流和输出数据流的数目,提高数据流图的可理解性。分解时要注意子加工的独立性、均匀性,特别是画上层数据流时,要注意将一个问题按照系统的相关性划分成几个大小接近的组成部分,不要在一张数据流图中出现某些加工已是基本加工,某些加工还要分解好几层的情况。

### 3.2.3 数据流图绘制实例

本小节以某供销系统为例,介绍分层数据流图的画法。该供销系统可以接受顾客的订货单,当库存中某配件的数量小于订购量或库存量低于一定数量时,向供应商发出采货单;当某配件的库存量大于或等于订购量时,或者收到供应商的送货单并更新了库存后,向顾客发出提货单。该系统还可随时向总经理提供销售和库存情况的统计表以备审查。该供销系统的部分数据流组成如下所示。

订货单 = 配件号 + 配件名 + 规格 + 数量 + 顾客名 + 地址

提货单 = 订货单 + 金额

采货单 = 配件号 + 配件名 + 规格 + 数量 + 供货商名 + 地址

送货单 = 配件号 + 配件名 + 规格 + 数量 + 金额

下面以此为例，介绍分层数据流图的画法。

**1. 画系统的输入输出（顶层图）**

把整个系统视为一个大的加工，然后根据数据系统从哪些外部实体接收数据流，以及系统发送数据流到哪些外部实体，就可以画出输入输出图，这张图也就是顶层图。

分析供销系统功能可知，该系统描述顾客、供应商、总经理与供销系统之间的数据的输入与输出的变换过程。可以确定有以下的输入数据流和输出数据流。

输入数据流：订货单（来自顾客）、送货单（来自供应商）、查询销售及库存情况（来自总经理）。

输出数据流：提货单（送往顾客）、采购单（送往供应商）、销售及库存情况（送往总经理）。

除了这些基本的数据流，软件系统应该对输入数据进行合法性验证，避免错误数据对系统造成不良后果。例如，订货单内容填写不完整等。因此，系统还应增加两个输出数据流，即不合格订货单（送往顾客）、不合格送货单（送往供应商）。根据以上分析，可画出该系统数据流图的顶层图，如图 3-5 所示。在顶层图中通常没有文件。

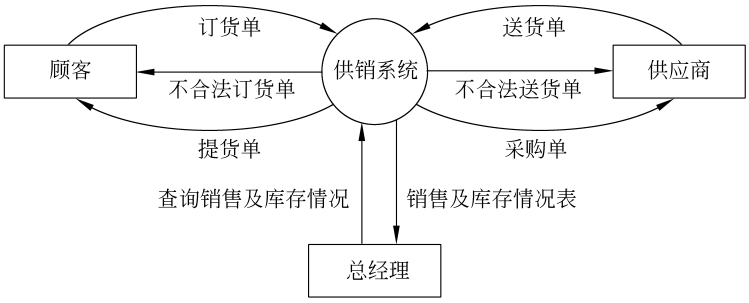


图 3-5　供销系统顶层图

**2. 画系统的内部（0 层图）**

把顶层图的加工分解成若干个加工，并用数据流将这些加工连接起来，使得顶层图的输入数据经过若干加工处理后，变成顶层图的输出数据流。这张图被称为 0 层图。从一个加工画出一张数据流图的过程就是对加工的分解。

（1）确定加工

在数据流的组成或值发生变化的地方应该画出一个加工，这个加工的功能就是实现这一变化，也可以根据系统的功能决定加工。根据系统功能的分解来确定加工的方法常见于高层数据流图中加工的分解，而根据数据流的变化来确定加工的方法多应用于底层数据流图中加工的分解。

例如，供销系统中，顾客提交的订货单首先需要检验合法性，合格的订货单还需要比较订货的数量与库存数量，以此确定是否发送采购请求。此时，在合格订货单和采购请求之间就存在一个加工，以实现该功能。

在供销系统的0层图中，采用功能分解法来确定加工。分析系统的需求说明可知系统的功能只要分为销售和采购两大部分。为此，定义两个加工，即“销售”和“采购”。

(2) 确定数据流

当用户把若干数据当作一个单位来处理(这些数据一起到达、一起处理)时，可以把这些数据看成一个数据流。实际工作环境中的表单就是一种数据流，如报名单、订货单等。

在父图中某加工分解而成的子图中，父图中的相应加工的输入输出数据流就是子图边界上的输入输出数据流。另外在分解后的子加工之间应增添一些新的数据流，这些数据流是加工过程中的中间数据，它们与所有子加工一起完成父图中相应加工的输入数据流到输出流的变换。如果某些中间数据需要保存，那么可以表示为流向文件的数据流。

同一个源点或加工可以有多个数据流流向另一个加工，如果它们不是一起到达并一起加工的，那么可以将它们分成多个数据流。同样，同一个加工也可以有多个数据流流向另一个加工或终点。

供销系统的0层图中的数据流，除了继承了顶层图中的输入数据流和输出数据流外，还应定义这两个加工之间的数据流。当“销售”加工接收到顾客的订货单后，需要查看库存的配件量是否能够满足订单的要求，如果库存量不足，就要向“采购”加工发出采购请求；经过“采购”加工处理，在供货商送来采购请求的配件后，需要向“销售”加工反馈一个到货通知，以便“销售”加工向顾客发出提货单。因此，在这两个加工中需要增加两个数据流，即“采购请求”(从“销售”加工到“采购”加工)和“到货通知”(从“采购”加工到“销售”加工)。

(3) 确定文件

对于一些以后某个时间要使用的数据，可以组织为一个数据存储来表示。在父图中某加工分解而成的子图中，如果父图中该加工存在流向文件的数据流(写文件)，或者存在从文件流向该加工的数据流(读文件)，则这种文件和与其相关的数据流都应画在子图中。

在分解的子图中，如果需要保存某些中间数据，那么可以将这些数据组成一个新文件。在自顶向下画分层数据流图时，新文件至少应有一个加工为其写入记录，同时至少存在另一个加工读取该文件的记录。注意，对于从父图中继承下来的文件，在子图中可能只会对其读记录，或只对其写记录。

在供销系统中，当“销售”加工接收输入数据流即“订货单”后，需要读取配件库存信息，若配件库存数量足够，则向顾客发出提货单，并更新配件库存信息；若配件库存不足，发出采购请求的同时，向缺货单写入数据。而“采购”加工成功进货后也需要及时更新配件库存信息和缺货单信息。所以，在0层图中可以确定有“配件库存”和“缺货单”两个文件。

通过以上三点分析，可以确定供销系统0层图中的加工、数据流及文件的详细信息，它们之间的关系如图3-6所示。