计算机软件工程

质量保证与产品可靠性评审测试技术新标准实务全书

◎ 主编 王志锦 罗乔欣

本手册不单独定价出售，随光盘附赠

计算机软件工程质量保证与产品可靠性评审测试技术新标准实务全书

主编　王志锦　罗乔欣

第二卷

三、数据项条目

数据项条目主要说明数据项类型、长度、取值范围等。条目格式如下：

数据项名： 类型： 长度： 取值范围：

例如：数据项“凭证号”条目和数据项“经办人”条目如下：

数据项名：凭证号 类型：数值 长度：6位（含小数1位） 取值范围：1000.0～4999.9

数据项名：经办人 类型：汉字 长度：8个字符 取值范围：二级国标

四、加工条目

加工条目主要说明加工的输入数据、输出数据及其加工逻辑等。条目格式如下：

加工名：
输入数据：
输出数据：
加工逻辑：

例如，加工“科目汇总”条目：

加工名：科目汇总
输入数据：记账凭证（批）
输出数据：科目汇总表
加工逻辑：将整批记账凭证的数据，按总账科目分借、贷方对金额进行汇总，并作借、贷方平衡检查，最后输出科目汇总表

又如，加工“工资分配”条目：

加工名：工资分配
输入数据：工资结算单（汇总表）
输出数据：工资费用分配表
加工逻辑：各车间根据工资结算单，按产品种类或批别，分别分配管理人员工资和生产工人工资，并按比例提取福利基金。

和数据流程图的层次概念相类似，一个数据字典的定义式不宜包含过多的项，这可以采取逐级定义的定义式，使得一些复杂的数据元素自顶向下多层定义，直到最后给出无需定义的基本数据元素。例如：

日期 = 年 + 月 + 日

数据字典就是这样构造起来的一组定义式。必要时，定义式之间还可能有一些特定的注释行出现，以利于理解。

和常用的词典相似，数据字典所收集的数据定义，都按词典的编辑方法顺序排列，以方便使用。当然，不允许出现一个数据元素有多个定义的现

象。

数据字典的建立和维护是件细致而又复杂的工作，大的数据处理系统在数据字典上投入的工作量也是相当可观的。人工建立和维护数据字典常采用卡片记载数据定义，也可以采用计算机进行数据字典的自动管理（机读数据词典），其管理功能包括对数据定义的修改、补充、查询、自身的一致性检查（发现冲突的定义式）以及与数据流程图的一致性检查等。

第六节　计算机软件需求分析的结构化技术及其设计方法

一、计算机软件需求分析的结构化技术

结构化需求分析大多使用自顶向下、逐层分解的系统分析方法来定义系统的需求。在结构化分析的基础上，可以做出系统的规格说明，由此建立系统的一个自顶向下的任务分析模型。规格说明描述了系统的需求，是联系软件需求分析与软件设计之间的重要桥梁。有许多方法可以用来进行需求分析，许多图形工具可以被用来辅助需求分析，下面讨论一些基于图形的结构化的分析方法。这些方法主要包括：

（1）结构化系统分析方法（structured system analysis）

（2）结构化分析和设计技术（structured analysis and design technique）

在结构化分析领域中，或在基于图形的需求分析方法方面我们还有其他一些有关的方法，如以用户为中心的需求分析（user - centered requirements analysis），软件工程需求分析（software engineering requirements analysis），层次方框图，PSA/PSL和面向ADA的需求分析方法等，本章最后一节将对利用自动工具为核心的一些方法给出简要说明。

（一）结构化分析方法

1. 概述

结构化分析（Structured Analysis，SA）是由美国 YOURDON 公司提出的，适用于分析大型的数据处理系统的，以结构化的方式进行系统定义的分析方法。方法的特点是利用数据流图来帮助人们理解问题，对问题进行分析。也就是说，很自然地用图形工具来模拟数据处理过程。由于软件总是在对数据进行加工，因此从原则上来说，可以用数据流方法来分析任意一种应用问题。目前，SA 方法又进一步并入了一些其他技术，如加入了状态转移图以便能适用于实时控制系统等等。结构化分析一般包括下列工具：

·数据流图（Data Flow Diagram，DFD）

·数据字典（Data Dictionary，DD）

·结构化英语或结构化语言

·判定表

·判定树

结构化分析的核心部分是数据流图，数据流图有时称为 bubble charts。Yourdon 的数据流图如图 2－4－6 所示，这是最常使用的一种形式。被许多作者（尤其是 Tom DeMarco）用各种文字推广开来。

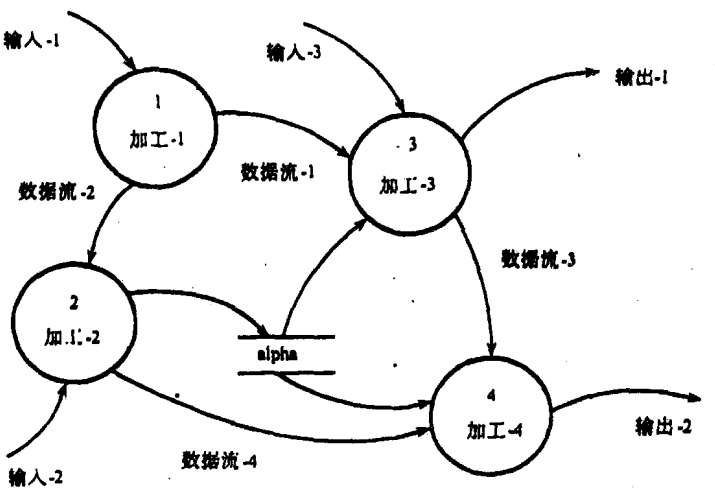

图 2－4－16　一个典型的 DFD

另一种类似的技术由 Gane 和 Sarson 所创建，其中使用了不同的图元。例如 Gane 和 Sarson 的技术中使用方框和带圆角的方框图，如图 2－4－17 所示。它们还应用了一种类似的数据字典，只是同上述的数据字典相比不够精

确。读者在不同场合可能看到不同形式的符号，只要认清其意义即可。

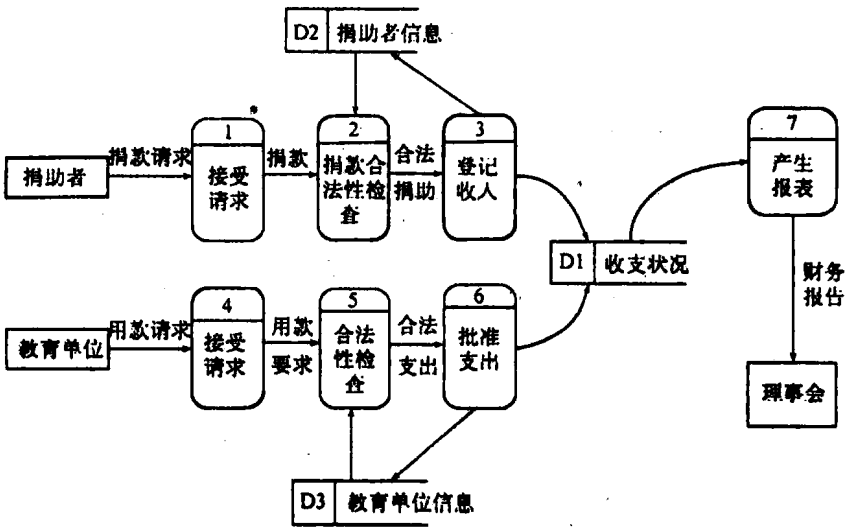

图 2－4－17　Gane 和 Sarson 的 DFD 例子

数据流图描述系统的数据流，包括组织系统、人工过程系统、软件系统、机械系统、硬件系统或这些系统的组合。数据流图是一种用来表示信息流程和信息变换过程的图解方法，把系统看成是由数据流联系的各种功能的组合。数据流图可以方便地描述用数据流的流动联系的各种功能（这些功能可以是线性联系，也可以是网状联系）。通过每种功能的输入数据/输出结果，进而表示现有系统或待开发系统的功能。此外，用数据字典定义数据流图中的各项数据；结构化英语、判定表和判定树用于具体描述数据流图中的基本功能。通过将系统分解成多层处理后，在较低的层次上，我们可以看到由数据流图的高层次加工的细节和相关的数据流。结构化分析方法的实质就是采用一组分层数据流图及相应的数据字典来作为系统的模型。

结构化系统分析方法从总体上看是一种强烈依赖数据流图的自顶向下的建模方法，是需求分析技术，也是完成规格说明文档的技术手段。

方法的第一步是研究“物理环境”。在此过程中，画出当前非自动系统（或半自动系统）的 DFD，指明系统的输入输出数据流，数据是如何流经整个系统的以及那些对数据操作的处理过程。数据流图可以包括自然环境中对数据流和处理的命名，例如部门的名称、人的姓名、本地过程、文件的组织

名称可在对自然环境进行描述的 DFD 中体现出来。在画 DFD 的时候，必须从数据的角度同用户进行交流，来判断处理的整体过程。这一步骤应当针对于自然环境的 DFD，完成时必须考虑完整性，并且让用户将它作为对现在系统的描述接受并认可。

对当前系统进行分析的基本目的是为了得到系统的逻辑模型，即逻辑上的 DFD，其中每一个数据流或处理过程都不再是真实的名称，而是系统的逻辑实体或逻辑操作。画出对自然系统的 DFD 只是为画出逻辑 DFD 提供了必要的起点。

因此，第二步分析工作就是画出相对于真实系统的等价的逻辑 DFD。在这一步骤中，在针对自然系统的逻辑 DFD 中，所有的自然数据流都由与之等价的特定的逻辑流所表示（例如，某现实的存档文件可以由“员工薪水文件”所代替）。类似的，自然处理过程的圆圈也被逻辑处理过程所代替，例如，自然系统中的加工“送往李经理的办公室”可以被替换为逻辑上等价的“报表清单”。没有对数据进行转换的圆圈应从 DFD 中删除。当用户检验通过后这一步骤也就完成了。

在最初的两个步骤中构造了当前系统的模型。接下来的步骤是建立考虑了改变后的新系统的模型，并且画出表明数据是如何流经新系统的数据流图。在此过程中同样也是分析人员在逻辑方式下进行工作，标志出哪些需要完成并如何完成，但并未区分出自动化的或非自动化的过程。

构造新系统的 DFD 并没有通用的规则。新系统还不存在，需要被开发出来。同时，新系统中什么是数据流、什么是要处理的过程都需要分析员基于他的经验和对系统的看法而判断出来。进行这种判断没有什么现成的规则。但是，在这些都完成以前，必须在现存系统的逻辑 DFD 中标出改进的分界线。新的 DFD 描述了整个新系统的逻辑。在新系统中，往往只有一部分可以被改变。这种改变是基于客户的目标以及对客户所要求的改变的清晰认识，从而必须在逻辑 DFD 中反映出改变的范围。新系统的 DFD 只对现存系统 DFD 中属于改变范围以内的部分进行改进，对于这个界线而言，新系统的 DFD 同旧系统的 DFD 应该是相同的。

接下来的步骤是完成人－机界线，指出在新系统的 DFD 中哪些被自动化完成，哪些仍保留由手工完成。注意，尽管一些处理过程并未自动化，但它们和原始系统中的那些处理过程可能是大为不同的，就如同原始系统中的

手工操作可能同新系统中的实现完全不同。通常没有固定的要求来实现人－机界线。根据哪些功能要求自动实现、实现自动化的程度存在不同的可能性。分析人员应该遍历这些可能并提供出来。

结构化系统分析能够清楚地提供组织和描述系统信息的方法，同时也提供了检验信息精确性的指标。为理解和分析一个现存系统提供了有效的工具。

2. 控制系统复杂性的基本思想

对于大型、复杂系统来说，最困难的事是如何处理复杂性。在软件工程中，控制复杂性的主要手段是“分解”与“抽象”。SA 方法采用这种“分解”与“抽象”的手段来对付一个复杂系统。如何来表示系统的功能呢？用“自顶向下逐层分解”的方式：图 2－4－18 中的系统 X 很复杂；为了理解它，可以把它分解成 1，2，3，4，……几个子系统。如果子系统 1 仍然很复杂，可以将它们再分解成 1.1，1.2 等子系统。如此等等，直到子系统足够简单，能够清楚地被理解和表达为止。

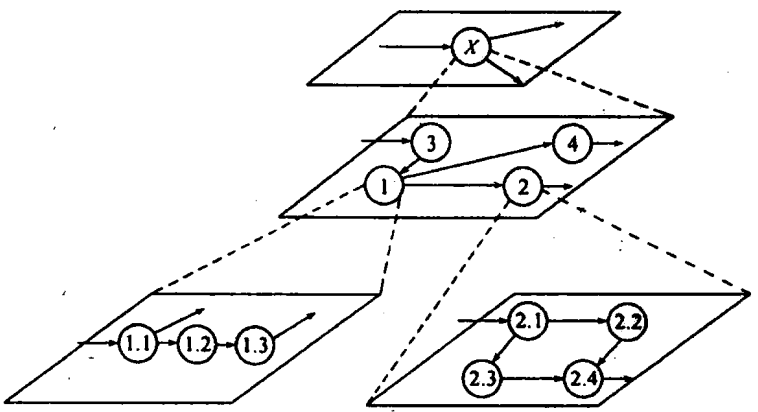

图 2－4－18　SA 中的分层子系统

对系统作了合理的逐层分解后，在最下层上我们就可以分别理解系统的每一个细节（图中的 1.1，1.2，...，2.1，2.2，...），并为每个细节写下说明（称为小说明），再将所有这些小说明组织起来，就获得了整个系统 X 的系统说明书。这种逐层分解也体现了抽象的原则，使人们不至于一下子就被许多细节所淹没，而是有控制地逐步了解更多的细节，以利理解问题。图 2－4－18 的顶层抽象地表达了整个系统，底层具体地画出系统的每一个细

节，而中间层则是从抽象到具体的过渡。按照这种办法，无论系统是多么复杂，分析工作总是可以有条不紊地进行。系统的规模无论有多大，分析工作的复杂程度不会随之而增大，而只是多分解几层而已。所以SA方法有效地控制了分析工作的复杂性，可以很好地处理大型复杂系统的需求分析工作和表达系统的需求说明书。下面我们简要地说明数据流图及数据字典的构成、建模步骤等，并举例说明。

3. 数据流图的常用元素

数据流图具有四种基本符号，分别代表了不同的数据元素：

→　一个命名的向量表示数据流，箭头的始点和终点分别代表数据流的源和目标

□　用方块表示数据源（终点）

○　用圆形表示对数据的加工（处理）

[（或=）用一端开口的长方形表示数据的存储

（1）数据流

数据流描绘DFD中各成份的接口。数据流的方向可以从加工流向加工，从加工流向数据存储，从数据存储流向加工，从源点流向加工，从加工流向终点。我们从以下几点来理解数据流的含义：

①数据流是一组成份已知的信息包。这信息包中可以有一个或多个已知的信息。

②两个加工间可以有好几个数据流。当数据流之间毫无关系，也不是同时流出（或同时到达）时，如果强制合成一个数据流，则会使问题更为含糊不清。

③数据流应有良好的命名，它不仅是作为数据的标识，而且有利于深化对系统的认识，更进一步地了解系统。

④同一数据流可以流向不同加工，不同加工可以流出相同的流（合并与分解）。

⑤流入/流出简单存储的数据流不需要命名。因为数据存储名已有足够的信息来表达数据流的意义。

⑥数据流不代表控制流。数据流反映了处理的对象，控制流是一种选择或用来影响加工的性质，而不是加工的对象。

（2）加工

加工是对数据执行某种操作或变换，是把输入数据变成输出数据流的一种变换。每个加工应有一个名字来代表它的意义。在 DFD 中每个加工还进行了编号。

(3) 数据存储

数据存储并不等同于一个文件，它可以表示文件、文件的一部分、数据库的元素或记录的一部分等等。数据可以存储在磁盘、磁带、存储器和其他任何介质上。应该认真对数据存储进行命名。在 DFD 中要注意指向数据存储的箭头的方向，它们可以是双向的。

(4) 源点和终点

源点和终点是代表系统之外的人、物或组织。它们发出或接受系统的数据，使用起来不严格，其作用是提供系统和外界环境之间关系的注释性说明。一般说来，表示一个系统只要有数据流、加工和文件就够了。但是为了有助于理解，有时可以加上数据流的源点和终点，以此说明数据的来龙去脉，使数据流图更加清晰。而源与源之间的关系则不予考虑。

(5) 系统

代表手动或自动的一组过程，属于我们问题的考虑范围之内的过程。例如图 2－4－19 中（a）表示系统中含有“手动加工 1”、“手动加工 2”、“手动加工 3”、“手动加工 4”和“自动加工”共五个加工；存在源或终点为 S1，S2 和 S3。如果只考虑自动系统部分，则图 2－4－19 中的（b）所示。这时候已经把“手动加工 2”和 S1 看作源或终点 Sa，“手动加工 1”和“手动加工 3”看作 Sb，“手动加工 4”和 S3 看作 Sc，S2 为 Sd。而 Sa，Sb，Sc 和 Sd 之间的关系（或数据流）则不必表示了。

我们看一个数据流图代表系统的例子。

真实世界或软件系统的模型都可以表示成数据流图。一个图书订购系统的简单数据流图如图 2－4－20 所示。从图 2－4－20 中可以看出，系统中处理的信息（即数据流）用带有标注的箭头来表示，有书名、数量、货物代码、书目、账单、费用等；加工（数据的变换）表示成为带有标记的圆圈，有预约系统、货物准备、记账系统等；信息的源点与终点表示为带有标记的方框，有代理商和顾客；信息的存储则采用双水平线（或三边的矩形）来表示，有图书目录和财会数据库。信息的源点是指数据的产生地点，而信息的终点是指数据流经系统后的汇集点，即数据的最终目的地。

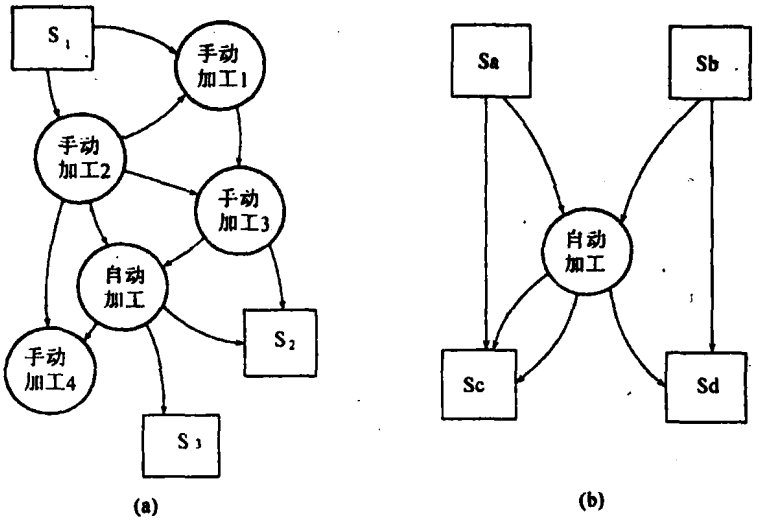

图 2－4－19　系统的例子

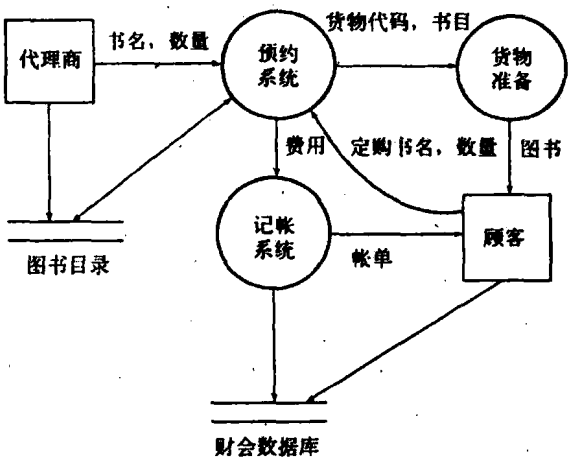

图 2－4－20　一个图书订购系统的简单数据流图

应该指出，需求分析阶段中的数据的处理（变换）并不一定是一个程序或程序模块，而是反映了系统的功能。如何实现，就要考虑在一定的约束条件下的系统设计方案的选择和编程问题。另外，它除了代表由计算机处理的过程，也可以代表人工处理过程，如由专业人员对输入数据的正确性进行检查等。与此类似，一个数据存储也不一定代表实际的一个文件。实际的数据

可以存储在包括人脑在内的任何介质上。

(6) 数据流图的扩展符号

除了四种基本符号，数据流图还具有若干扩展符号（但这些符号不经常使用）。

①星号（*）。说明数据之间的“与（and）”关系。输入数据之间的“与”关系表示输入的数据必须同时存在（满足所有的输入条件）才有可能产生输出；输出数据之间的“与”关系表示数据处理将同时产生相关的多个输出。

②加号（+）。说明数据之间的“或（or）”关系。输入数据之间的“或”关系表示只要指定的输入数据中至少一项存在，数据处理就可以产生输出；输出数据之间的“或”关系表示数据处理可以产生相关的多个输出中的至少一项，或是全部输出。

③互斥符号（⊕）。说明数据之间的互斥关系，即在待选的多个项目中，在同一时刻必须存在且只能存在一个。数据处理如果存在多个互斥的输入，则同一时刻只能接受一个特定的输入；多个互斥的输出表示数据处理将产生相关的多个输出中的一项。

4. 如何画数据流图

对于给定的问题，没有细节性的过程用来帮助完成 DFD，这里只能提供一些指导性的意见。针对软件的不同用途、实现方案和规模需要根据不同的思路来绘制数据流图。随着经验的增加，分析员会发现对于不同的问题，画数据流图的方法不一样。一种方法是始于识别出主要输入和主要输出，次要的输入和输出（如错误消息等）暂时忽略不计，然后从输入向输出推进，找出通道上的主要变换。也可以采用从输出问题找到输入（记住 DFD 中不要有循环判断等过程处理信息，设计人员不应该在画 DFD 的时候考虑这些问题，这是非常重要的）。在大多数情况下，原则上是由外向里、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。如下顺序是常用的步骤：

①在图的边缘标出系统的输入、输出数据流。

②产生数据流图的内部，将系统的输入、输出用一系列的处理连接起来，可以从输入数据流画向输出数据流，也可以从中间画出去。

③仔细为数据流命名。

④根据加工的输入输出内容，为加工命名。

⑤不考虑初始化及终点，暂不考虑出错路径等细节，不画控制流和控制信息。

⑥反复与检查。

总之，原则是“先外后内”。下面对每一步做较为详细的说明。

(1) 画系统的输入和输出

这一步实际是决定研究的内容和系统的范围，向用户了解“系统从外界接受什么信息和数据”，“系统向外界送出什么数据”。画在数据流图的外圈。最开始并不十分清楚系统包括那些功能，所以可以将系统的范围画得大些，即把可能的输入、输出都画进去（因为此时如果有些内容未包含进去，则不会分析到，以后容易永远遗漏）。然后仔细分析，删除多余的部分，增添遗漏的部分。不必担心完整性问题。

(2) 画数据流图的内部

一开始不要考虑事物应当如何出现，只要把实际情况反映出来就行了。应集中精力首先找出数据流。如果有一组数据一起到达，并一起处理，则应将这些数据画成一个数据流；反之，对不相关的数据，则应分成不同的数据流。找出数据流后，设法将它们与边界上的系统的输入、输出数据流连接起来，在需要对数据进行处理或加工的地方画上圆圈或圆角矩形作为加工。始终保持由输入到输出的方向，或者反过来，一旦在这个方向上遇到困难，就反过来处理。从开始的只代表整个系统从输入到输出的数据流中少量的加工开始构造 DFD。接下来，再想象一下每个加工中是否存在内部的数据流，是否需要用 2 到 3 个加工及数据流来替换它。每个数据流应检查它的组成，来自何处，能否从输入项得到输出项。如果有数据存储，应画出相应的图示，并了解其组成及输入输出，从而可以对每一个加工进行改进，描述其加工中的细节。为了清晰可见，如果同一个数据存储多次出现，可以把它在图中多处画出来。最后，反复修改边界，消除多余，补上遗漏。

(3) 为每一个数据流命名

数据流命名的好坏与数据流图的可理解性密切相关。命名时应避免使用空洞的名字，例如“数据”、“信息”、“输出”等等，因为这些名字并没有反映出任何实质性的内容。如果发现难以命名，不妨考虑重新分解数据流或加工，很可能原来的组成不合适。名字要反映整个数据流的含义，而不是其中

某一部分。

(4) 为加工命名

先命名数据流，再命名加工，这个次序反映了SA方法的自上而下的特性。例如图2-4-21（a）中，当数据流已经命名后，加工P的命名可以自然地给予“检查取款单的合理性”；而图2-4-21（b）中加工已命名，但无法为几个数据流命名。

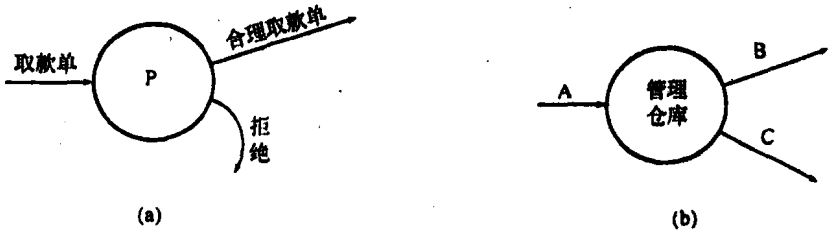

图2-4-21　(a) (b) 加工命名的例子

为加工命名时应注意：名字要反映整个加工，而不是它的一部分；名字应当是一种“动词+宾语”的形式；遇到不能适当命名的加工，要考虑重新分解。另外，名字中用一个动词就足够了，如果一定要用两个以上的动词，则应该将它分成几个加工。

最后要强调的是：

①只画出稳定状态，先不考虑如何开始、如何终止。

②忽略琐碎细节问题。对于一些更进一步的问题，如异常处理等，此时也不考虑。

③数据流图中强调的是数据流，而不是控制流。此时如果将控制流也画上去，则会使数据流图十分复杂，难于理解。此时的重点是突出数据流，即反映整个业务活动的情况，绝对不要描述控制逻辑。如果发现自己正在考虑与循环或判断有关的事项，就该停下来从头开始了。

④准备反复和重画。人类思考的过程是一种迭代的过程，不可能一次成功，需要逐步完善，直到满意为止。因此要随时准备放弃原有的DFD，用改进的DFD来替代。在确定某一DFD之前先找出可选的数据流图。重画DFD只需少许代价，减少了错误，就可得到极大效果。通常DFD总有好几次反复。

5. 分层数据流图

(1) 概念

对于一个大型的软件系统，如果只用一张数据流图表示所有的数据流、处理和数据存储，那么这张图必然会十分复杂、庞大，而且难于理解。层次结构的数据流图可以很好地解决这个问题。为了有效地控制复杂度，可以在产生数据流图时采用分层技术，提供一系列的分层的 DFD 图，来逐级地降低数据流图的复杂性。绘制分层数据流图的过程也就是逐步求精的过程。按照系统的层次结构逐步分解，并以分层的数据流图反映这种结构关系，从而可以清楚地表达和容易地理解整个系统。在这里，分层的结构起到了对信息进行抽象和隐蔽的作用。由于高层次的数据流图不体现低层次的数据流图的细节，可以暂时掩盖低层次数据处理的功能和它们之间的关系。高层次的数据流图是其相应的低层次图的抽象表示，而低层次的数据流图表现了相应高层次图中有关数据处理的细节。

(2) 组成

我们可以把一个系统的分层数据流图划分为顶层 DFD、中间层 DFD 和底层 DFD 三种。

顶层数据流图的结构简单。它描述了整个系统的作用范围，对系统的总体功能、输入和输出进行了抽象，反映了系统和环境的关系。在软件系统中，只存在一张顶层数据流图，有时称为内外关系图（context diagram）。

中间层数据流图是通过分解高层数据流和数据加工得到的。层次较高的数据流图经过进一步分解得到层次较低的数据流图。一张中间层的数据流图具有几个可分解的加工，就存在几张对应的低层次数据流图。通常有很多中间层，甚至 8～9 层。对特别简单的系统，可以没有中间层。这种分解可以不断重复，直到新的数据流图中每个数据加工的功能明确、相关的数据流被严格定义为止。

实际上，在对一个加工的进一步分解涉及到如何具体实现一个功能时，或分析人员不愿再分解时，分解就可以停止，这就得到了系统的底层数据流图。

(3) 分层原则

建立分层的数据流图时应当注意以下几个问题：父图和子图关系、编号、平衡规则、文件的局部性及分解程度（即底层数据流图的确定）。