目　录

引　言

从广义上讲，任何协助与软件开发相关的人员在一个给定的应用环境中建立软件系统的辅助手段都可以看作是软件工具。例如图表、模型等，常见的例子有流程图等。支持这些图表、模型的软件(CASE)也被看作是软件工具。

实际上，我们平时所说的开发工具主要是指计算机辅助软件工程(CASE)工具(后面的章节均称为 CASE 工具)，它是在具体的软件开发理论和软件方法的基础上进行研发，然后投入软件的开发过程中的。以 Rational Rose 为例，它是基于面向对象这样一种软件设计分析方法和 UML、支持 RUP 过程的 CASE 工具。

0.1　CASE 工具的发展史

CASE 工具是伴随着计算机辅助软件工程(CASE)的发展而产生和发展的。业界的一般看法是 CASE 工具起源于 20 世纪 70 年代初期并在 20 世纪 80 年代得到了蓬勃发展。20 世纪 70 年代末，自动软件开发工具的使用逐步增加，推动了当时的软件工程学科的发展。20 世纪 80 年代，CASE 工具得到了前所未有的重视，对应于软件生命周期各个阶段的各种 CASE 工具开始出现在各个软件工程工作站，此时的 CASE 技术把自动建立文档作为了提高软件开发效率的关键并初步引入了自动检查结构化图形和设计信息库。CASE 工具的蓬勃发展曾使得在 1986 年的时候，布鲁克斯教授由于不满于当时的 CASE 工具开发商所做的过多不实的宣传，提出了著名的“银弹”理论。进入 20 世纪 90 年代后，将人工智能、专家系统技术引入 CASE 工具成为了 CASE 工具的发展趋势，同时基于 PC 的 CASE 取代了原有基于工作站的 CASE 而成为了 CASE 的主流。此外，由于 Windows 技术的盛行，CASE 工具在界面友好性、交互性等方面相对过去有了极大改观，CASE 工具成为了开发人员的必备。目前 CASE 工具的发展集中于以下几方面。

1. 提高开发阶段之间衔接的流畅性

CASE 环境包括在软件工程初期使用的工具，如需求分析、系统生成、原型生成等工具以及在软件工程晚期使用的工具，如代码生成、测试生成、运行维护等工具。从系统的数据流动看，这两个阶段的工具衔接点之间无断层。但是，就软件过程而言，由于目前一些软件企业使用的还是传统的结构化开发模式，所以使用的 CASE 工具是基于结构化分析和结构化设计的技术，这一技术局限性使得这两个阶段的工具衔接点之间存在着明显的间隙。后来

支持面向对象开发的 CASE 工具不断涌现，代替了传统的 CASE 工具，填充了开发阶段的间隙。但是，就各阶段衔接的流畅度来说，尚不能达到令人满意的程度，因此提高开发阶段间的衔接流畅性是 CASE 工具的一个发展趋势。

2. 标准化

由于目前 CASE 技术仍处在发展阶段，各公司自行其是，只求功能实现，不求界面功能的基本统一和数据格式的统一。因此各种不同 CASE 工具间的文件无法交换使用，造成了不必要的人力和财力浪费。因此尽快制订有关标准是 CASE 技术发展的当务之急。随着 XML 技术在市场上的蓬勃发展，我们完全有理由相信，未来的 CASE 技术在数据文件交换上一个显著的特征就是采用 XML 作为交换格式。

3. 自动化

我们不可能赶上待开发的软件系统积压的速度，减少软件积压的方法之一就是不断地增加软件开发人员，具体的做法是：教会更多的人开发软件、使计算机系统更容易使用以及让用户参加软件的开发过程。如果这样做，我们就可能建立更多的软件系统。但我们可能还是跟不上最终用户计算任务的增长速度。因此这种方法作用不大，不值得提倡。

另一种解决问题的方法是教会计算机建立软件系统，即实现软件开发过程的自动化。现有业务领域中的大部分业务流程是相同的，通过建立标准的"业务模板"，在具体开发中稍加修正，这样的任务是比较适合于自动化的。记住某个程序设计语言的语法规则或某类计算机的操作系统的命令结构以及对系统进行查错这样单一重复的任务，也适合采用软件自动化。另外，需要用到专家的知识和经验的那些软件开发任务也同样适于实现自动化。

国外市场上已经出现了这方面的工具。在日本市场上常见的就有 Xupper、QuiqPro、Proness 等。越来越多的软件开发厂商为了谋求最大的利润空间（以降低人力成本为中心）不断苦苦寻觅出路，而软件自动化恰恰打开通往这条路的大门。

4. 业务反工程

CASE 工具的重要功能之一是反向工程，它是指 CASE 工具通过分析现有运行程序，反向生成 CASE 所使用的各种文档，如 UML 图形、E－R 图等，以维护现有的软件资源或作为下一次软件开发的依据的过程。反向工程类似于反汇编，但比反汇编更为复杂，要求更高。因为反向工程的对象既包括数据，又包括过程。它要求反向生成的有关文档既要正确无误，又要求能够根据这些文档，使用 CASE 工具生成与原功能相符的软件系统。反向工程还应通过对原软件系统的分析，得出程序复杂度以及可靠度的评估报告。目前的反向工程面临两个较大的难题，即如何在进行反向工程时尽量保证与原有系统的分析设计信息的一致性，以及如何实施业务级的反向工程。

5. 公用库

由于开发人员使用 CASE 环境的目的复杂多样，因此增加尽可能多的公用库，可使开发人员能够根据自己要开发的软件系统，选择最适合作业目的的 CASE 工具和信息包，从而高效率地利用 CASE 进行软件系统开发。由此可以看出，CASE 的发展方向之一是生成数量巨大，可供不同开发人员选择使用的公用库。

6. 人机界面

Windows 技术获得的巨大商业成功显现了人机界面在当今社会中的商业价值。未来的 CASE 工具亦必然会不断改善其人机界面。未来的 CASE 工具在人机界面设计方面除了提供用户友好的、反应迅速的、可诊断的、能提供求助的功能之外，还增加了一些新的特征如以用户为中心、可反馈、可纠错、有教学功能，并能起到指导开发人员的作用。

0.2 本书对 CASE 工具的划分

在本书中，CASE 工具主要是依据使用该 CASE 工具的开发阶段和 CASE 工具的主要功能来划分。但是一些集成化的 CASE 环境，诸如 Rational Rose、Microsoft Visio、PowerDesigner 等，由于它们的使用往往跨越开发过程的多个阶段，因此很难将它们进行归类。为此，我们根据它们使用较多的场合和目前业界大多数人的看法将它们进行归并介绍。下面是常见的 CASE 工具的分类。

1. 需求分析工具

(1) 国产系列：PlayCASE；
(2) 国外系列：BPwin、Rational Rose(Requisite Pro)。

2. 软件设计工具

Microsoft Visio、PowerDesigner、Rational Rose。

3. 数据库设计工具

数据库设计工具：Erwin、ER/Studio；
数据库开发工具：Oracle/Form，Oracle/Developer，Object Browser for Oracle 等。

4. 项目管理工具

项目管理工具：Microsoft Project；

配置管理工具：Visual Source Safe、WinCVS、Rational ClearCase。

5. 程序设计工具

Microsoft 系列：Visual Studio.NET；
Borland 系列：JBuilder、C++ Builder、Delphi；
其他：PowerBuilder、Macromedia 系列等。

6. 测试工具

自动化系列：Panorama 系列；
非自动化系列：SoftIce、Junit 等。

第1章　需求分析工具

1.1　需求工程与需求分析工具

软件产业过去40多年的经验和教训已经证明了这样一个事实:软件需求的质量高低将决定软件产品的质量。高质量的需求分析过程一直是软件开发人员梦寐以求的目标。不幸地是,在需求过程中往往会产生一些错误,许多错误并没有在需求阶段的初期被发现(实际上这些错误是能够在其产生初期被需求人员检查出来的)。此外在绝大多数情况下,需求分析也缺乏定量的验收标准来供需求分析人员对需求过程本身和需求结果进行客观、可靠的度量。由于客户在软件知识方面的匮乏,同时由于需求人员专业领域知识的贫乏,往往无法全面、正确地理解并获得用户的实际需求信息,因此所产生的需求产品——SRS(需求规格说明书)很难得到客户的认可。对于需求质量,需求人员往往只能给出"好"或是"不好"的模糊评价。此外,由于软件开发过程中软件错误的"放大效应",需求错误作为"放大效应"的源头会使最后交付的软件产品不能满足用户的实际需要而浪费大量的时间和金钱。最糟糕的结果可能会是客户与开发商之间的法律纠纷。据统计,软件中的错误大约有15%是由于软件需求分析错误,软件开发失败大约有50%是需求的不合理所导致的。可以得出这样一个结论:完善的需求分析是软件开发成功的关键。

高质量的需求分析的需求促使我们必须审视我们在软件需求分析过程中所采用的方法、技术以及过程的管理。这些概念实际上属于软件需求分析工程的范畴。

1.1.1　需求工程

需求工程是需求的供需双方采取被证明行之有效的原理、方法,通过使用适当的工具和符号体系,正确、全面地描述用户待开发系统的行为特征、约束条件的过程。需求工程的结果是对待开发系统给出清晰的、一致的、精确的并且无二义性的需求模型(model),并通常以SRS(需求规格说明书)的形式来定义待开发系统的所有外部特征。该模型实际上是对用户在不同需求层次上的模拟性说明,是用户的"业务世界(可系统化业务对象)"向由软硬件组成的"电脑世界"建立一一映射的过程。

需求工程涉及的角色(不要与人相混淆,角色是指一种职责,同一个人可以担当多种角色)包括客户方(客户、系统使用者)、系统分析师、项目开发及管理人员。其中系统分析师起到桥梁工程师的作用,负责完成用户"业务世界(可系统化业务对象)"逻辑向由软硬件组成

原书缺页

首先从需求分析工具的自动化程度来看，需求分析攻具可以分为两类。

(1) 以人工方式为主的需求分析工具。人工方式为主的工具为系统分析师们提供了一种意义明确的技术(通常附有某种图形、符号的表示方式)，该技术使得需求分析工作能够系统地进行。虽然该技术可以由一个或多个自动工具来协助实施，但是分析和规格说明却仍然要求人工实现。在这类工具中，结构化分析和设计技术(SADT——SofTech 公司的商标)是一种有代表性的工具。

(2) 以自动化方式为主的需求分析工具。过去的 10 年中，对于需求规格说明已经有了一些自动工具。在证实人工描述系统的一致性和完善性的过程中所遇到的困难促使形成了一种自动方式。该方式通过保证需求信息的一致性和完整性来实现需求分析的自动化。不少工具还内建了用户的用语信息库或是流程信息库，日本著名的 Xupper 便是这类工具中的佼佼者。

其次从工具的支持分析设计技术可将需求分析工具分为下面几类。

(1) 支持传统的结构化方法的需求分析工具。这类工具的共同特点是支持数据流程图的生成和分解，支持对数据流程图的索引，同时支持数据字典的生成和管理。不少工具还支持程序结构图的生成和分解。此外还有很多的此类软件支持美国军方的 IDEF(Integration DEFinition)系列的软件开发规范，典型代表如美国 CA 公司的 BPwin。

(2) 面向对象分析的需求分析 CASE 工具。这类工具支持 OMT、OOSE、Booch 等面向对象的方法。就目前来讲，不少市面的面向对象分析的需求工具均支持 UML 的全部或是一部分(主要针对基于用例的面向对象方法)，从内容上讲这类工具至少支持用例分解和描述、用例索引的生成等。典型的面向对象的需求分析产品代表如 Rational Rose 家族。

(3) 原型化分析的需求分析工具。该类工具支持画面的快速生成，能够较快地生成用户界面，不少工具自身内建了标准的代码模板，经过简单修改后能够生成系统的大致框架以供用户和系统分析师参考。原型化分析的需求工具特别适合于 RAD 开发。目前这类工具在日本市场比较受软件开发厂商的欢迎，典型代表有富士通的 Proness、QuiqPro for Web/VB 等。

(4) 基于其他方法的需求分析工具。这类工具往往针对特定的领域，因为在这些领域需要专有化的方法来进行需求分析。比如实时系统一般采用的 Petri 网技术就属于该类型。

最后根据需求工具和客户的业务领域的关系可将需求分析工具划分为多类，比如 ERP 领域需求分析工具、实时领域的需求分析工具和其他业务领域的需求分析工具等。

目前需求分析工具非常多，而且大多与设计、乃至代码生成工具组合在一起，从而使得开发人员使用时可以非常方便地从需求分析阶段平滑地过渡到设计阶段，然后再过渡到代码阶段。

2. 需求分析工具的功能特性和衡量标准

作为需求分析的 CASE 工具应当尽可能满足下列特性。

(1) 针对结构化方法

- 多种分析与设计方法(SA、SADT、面向数据结构等);
- 作为采用结构化方法的需求分析工具应当支持 DFD(数据流程图)的编辑功能。包括图形、文字的添加、删除、修改、块搬移、块复制等;数据字典自动生成与管理功能。即根据用户对数据及其相互关系的描述,自动生成数据字典,并最终生成数据关系图以及数据流程图;
- 一致性检查功能,即对涉及的所有数据项进行检查,防止产生数据项命名、重名、数据流向等错误。

(2) 针对面向对象方法

- 支持典型的多种面向对象方法(OOA/OOD, OMT, Booch, OOSE, UML 等);
- 支持类定义和类关系描述;
- 支持对象复用;
- 支持对象交互描述;
- 一致性检查,检查对象关系的逻辑一致性,防止产生对象重名、消息流向和关系标识误用等错误。

(3) 以下是一些共性

- 支持信息仓储(repository),信息仓储对在开发人员间共享需求分析资料是必要的。两个以上的开发人员可以通过共享来进行需求的协同分析;
- 支持业务反向工程;
- 支持版本控制。工具应允许存储各种版本,以便后续迭代开始时,以前的版本仍然可以得到,并用于重建或保持基于该版本的原有资料;
- 脚本支持,用脚本编程是需求建模工具应该支持的另一个强大特性。有了脚本功能,用户可以定制和添加其他功能。
- 支持生成需求分析规格说明书;
- 能够改进用户和分析人员以及相关开发人员之间的通信状况;
- 方便、灵活、易于掌握的图形化界面;
- 需求分析工具产生的图形应易于理解并尽量符合有关业务领域的业界标准;
- 支持扩展标记语言(XML);
- 支持多种文件格式的导出和导入;
- 有形式化的语法(或表),能够供计算机进行处理;
- 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。

衡量一个需求分析 CASE 工具功能强弱的主要依据如下。

- 所支持的需求分析方法的类型与数量的多少。优秀的需求分析工具应支持尽可能

多的分析方法和符号体系。

- 使用的方便程度。优秀的需求分析工具应支持图形用户界面(GUI),并提供详细的帮助文档和示例,使用户易学易用。
- 与设计工具衔接的程度。优秀的结构化需求分析工具所产生的数据流程图和数据字典,优秀的面向对象的需求分析工具产生的用例图、对象交互图、类图等可以无任何阻碍地为后继的设计工具所使用。
- 所占资源,即系统开销的多少以及对硬件环境的需求程度。优秀的需求分析工具应当占用尽可能少的资源,并且对硬件环境的需求很低。
- 是否提供需求错误检测机制,好的需求分析工具应当提供不一致性和冗余性方面的校验甚至纠错的功能。
- 用户领域知识提示功能。针对专门领域建模的需求分析工具应当提供该领域的知识提示,并通过相应的用语信息库和流程信息库来帮助分析人员快速掌握客户领域的知识。

3. 需求分析 CASE 工具的选择

选择适合个人或者公司的需求分析 CASE 工具,应该遵循因地制宜的原则。首先从所从事的行业入手,分析该行业信息建模方面的关键所在,目前哪些方面极其需要用工具代替人工,这些工具应该支持哪些方面的业务(活动),支持的程度应该达到多少,有针对性地进行购买,切忌盲目听信媒体宣传;其次价格也是一个比较重要的因素,如果是一次性的项目,采用高昂的需求分析 CASE 工具会得不偿失,建议从该工具的复用度入手,分析需求分析 CASE 工具在功能满足度、价格、在其他项目中的复用程度以及与现有 CASE 工具间的可集成度等方面进行权衡。

1.2　需求分析方法与需求分析工具

软件需求分析的方法与工具众多,常用的需求分析方法有 Yourdon 公司的结构化分析方法 SA(Structured Analysis)、IDEF 方法系列、面向对象的分析方法 OOA(Object-Oriented Analysis)。

常用的需求分析图形工具有:

- UML(Unified Modeling Language);
- 数据流图 DFD(Data Flow Diagram);
- 数据词典 DD(Data-Dictionary);
- 判定表(Decision Table);
- 判定树(Decision Tree);
- 结构化高级分析语言;
- 层次图 HC(Hierarchy Chart);

- 输入处理输出图 IPO(Input/Processing/Output);
- Warnier 图;
- 结构化分析与设计技术 SADT(Structure Analysis& Design Technique);
- 软件需求工程方法 SREM(Software Requirements Engineering Methodology);
- 问题描述语言与问题描述分析器 PSL/ PSA(Problem Statement and Problem Analyzer)等。

支持这些图形绘制的软件 CASE 工具常见的有 Microsoft Visio、CA 公司的 BPwin,Sybase 公司的 PowerDesigner、IBM 公司的 Rational Rose 系列、Borland 公司的 Together 和开源的 ArgoUML 等。

在软件生命周期中的分析设计阶段,主要目标是精确、简明地把用户需求转换为系统需求,并在此基础上构造稳定、健壮的系统架构,为软件的实现打下坚实的基础。

结构化方法历史悠久,比较成熟,并且已经形成了一整套规范、标准,涵盖了分析设计的各个方面,如分析设计的基本语言、分析设计的过程、分析设计的规范与分析设计工具的结合,甚至包括分析设计文档的标准。因而。结构化方法仍然具有强大的生命力。

面向对象编程技术目前已比较成熟,但面向对象技术应用于软件分析设计阶段的时间还不是很长,UML 虽然已经成为事实上的工业标准,但是仍然处于不断发展、修正的过程中。并且,UML 是建模语言,它不是方法,它不包含应用的过程,尽管这使得 UML 可以应用于任何过程之中,但这恰恰说明,从软件开发过程的角度来看,面向对象分析设计方法过程标准的形成还需要一定的时间。

应当看到,不论何种方法都是在实践的基础上提炼出来的分析设计方法,它们没有本质上的优劣之分,只有特点上差异之别。在分析设计领域内,我们不应当局限于方法和工具的选择之中,只要它能提高分析设计的效率,就可以采用,而不必过多地去考虑它是结构化方法还是面向对象方法。工具的采用也是同样的道理。

1.2.1 结构化方法与工具

结构化分析(Structured Analysis)方法,简称 SA 方法,是在 20 世纪 70 年代中期由 E. Yourdon 等人倡导的一种面向数据流的分析方法。结构化方法是建立在系统分析师已经比较全面地获取了用户需求的基础上的。适用于用户需求变更较少或是局部变更的场合。

结构化分析的特色体现在如下几个方面。

- 理念:模块化的思想,采用"自上而下,逐步求精"的技术对系统进行逐层次分解划分。
- 方法手段:分解和抽象。
- 适用范围:适用于以数据进行处理加工为出发角度的软件系统的分析。
- 特点:用图形工具来模拟数据处理过程。

所采用的工具包括:

- 数据流图(Data Flow Diagram,简称 DFD);
- 数据字典(Data Dictionary,简称 DD);

- 结构化英语或结构化语言；
- 判定表；
- 判定树。

1. 数据流图介绍

结构化分析的核心是数据流图。数据流图由 4 种基本符号组成，分别代表了不同的数据元素。

- 用命名的箭头表示数据流，箭头的起点和终点代表数据流的源和目标，它表示数据和数据流向；
- 用方块表示数据源(终点)，它是系统外的实体；
- 用圆形表示对数据的加工(处理)，它接受一定的输入数据，对其进行处理，并产生输出(注：Gane/Sarson 方式则为带圆角的方块)；
- 用两端或一端开口的长方形表示数据的存储(注：Gane/Sarson 方式为一端开口)；

图 1-1 是一个 ATM 机取款的数据流图。

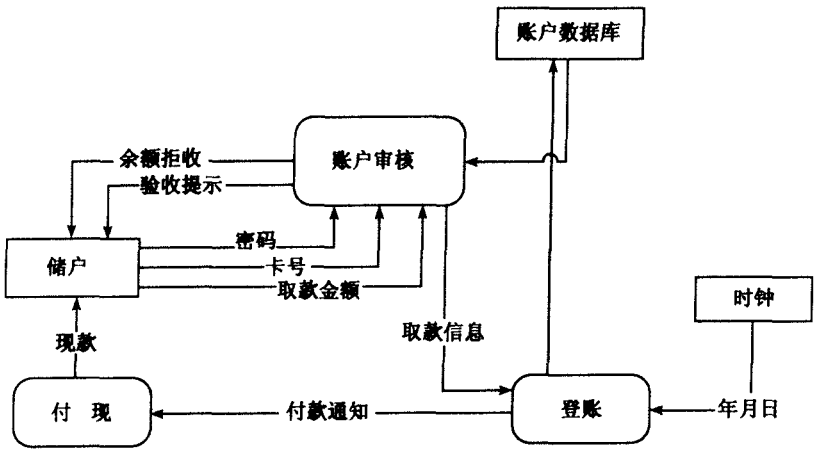

图 1-1　ATM 机取款的数据流图

在数据流图中，系统(子系统)被看作是多个相关的数据加工(处理)的集合。

数据流源是对系统提供输入数据流的外部实体(external entity)，数据流的终点是指接受系统输出数据流的外部实体，源点和终点可以是同一实体。

加工(处理)是对数据执行某种操作或变换，把输入数据变成输出数据流的一种函数变换。每个加工(处理)应有一个符合加工(处理)机理的名字来代表它的意义，此外在数据流程图中每个加工还应该有编号。

数据流是加工(处理)的对象或结果，这一点类似于函数的变量和值。而加工(处理)则是函数原型。值得注意的是，两个加工间可以有多个数据流。同一数据流可以流向不同的加工，不同的加工可以流出相同的流(数据流的合并与分解)。数据流图中描述的是数据流，

而不是控制流。因此，分清数据流和控制流是很重要的。刚刚进入需求分析阶段的开发人员特别容易把这两个概念混为一体。

此外数据存储不同于文件，它可以表示文件、文件的一部分、数据库、纸质报表等。此外，数据流程图中指向数据文件的箭头的方向可以是双向的。

2. 数据流图的建立步骤

(1) 确定系统的输入输出

首先要确定软件系统的输入和输出，然后再考虑系统的内部情况。刚开始分析时，可能一时还弄不清楚系统究竟包括哪些功能，这时可以将系统作为一个黑盒子，只需了解它需要输入什么数据，输出什么数据。这样做比较符合人的逻辑思维习惯。

(2) 由外向内，画出系统的顶层数据流图

列举各种可能的加工，将系统的输入数据和输出数据用一连串的加工连接起来。对于没有数据输入和输出的加工进行删除。如果两个加工的输入和输出完全一样，则它们很可能是同一个加工。要记住数据发生变化的地方一定有加工过程。其次需要给各个加工进行命名。加工的名字应包含该加工的全部功能。如果发现对一个加工的命名很困难，那么这意味着数据流图分解不当，这个加工中可能包含着几种不同的功能，此时应考虑重新分解这个加工。此外还应该给加工之间的数据命名。加工之间流动的数据有数据流和数据项两大类。数据流是指系统中的一组数据，把它作为一个整体来处理。数据项是指一些互不相关的数据单项，它不具有整体性，尽管它们也会一起流到某个加工，因此发生命名困难的情况时，很有可能就是由于数据流中有数据项的存在。

(3) 自上而下，逐层分解，绘出分层数据流图

在分解数据流图的过程中，我们只需要考虑加工之间的数据流动，而不需考虑数据之间的控制，即数据流图中只有数据流，没有控制流。分层数据流图由顶层、中间层和底层组成。顶层数据流图抽象地描述了系统的结构；中间层数据流图描述了某个加工的分解底层数据流图由一些功能最简单、不能再分解的加工组成，这些不能再分解的加工叫基本加工或元加工。较小的系统可以只有顶层数据流图，较大的系统中间层可达八九层之多。根据心理学的研究结果，加工的分解最多不要超过 7 ± 2 个。一般说来，当每个加工都已足够简单时，我们就可以认为绘出分层数据流程图的工作完成。在进行绘图时，要注意如下几个问题。

- 绘图与编号要同步，并且编号应该具有条理性。
- 父图中某个加工的输入输出数据流应该与这个加工的子图的输入输出数据流相同，即父图与子图必须平衡，这是分层数据流图的重要性质和分解原则。
- 是否应当画出局部文件取决于文件是否被某个加工读写。当这个文件被读写时，就应该画出来。
- 数据字典实际上是对数据流图的补充及描述。数据字典由 4 类元素的定义组成，它

们分别是数据流、加工、数据存储和数据元素(数据流和数据存储的组成成分)。

数据流图体现的是对系统的功能和数据流的分解。数据字典中对数据的定义也体现为对数据的自上而下的分解。当数据被分解到不需要进一步解释说明、每个参与需求的人员都能理解数据条目的含义时,数据的定义就完成了。与程序的流程方式相似,数据条目的组成方式由3种基本类型构成。

- 顺序方式:两个或多个成分以排列形式连接在一起。
- 选择方式:在两个或多个成分中选取一个。
- 重复方式:特定的成分需要重复零次或者多次。

数据字典常常使用特定的逻辑运算符来产生数据和数据成分间的精确而简洁的描述。这些逻辑运算符介绍如下。

=	由…组成(被定义为…)
+	和(AND 关系 顺序关系的连结)
{ }	重复关系
[|]	选择(但至少选择一个)
( )	选择(可以不选)
* *	注释

一个完整的数据流条目应该包括名称、描述、频率、数据量和数据结构。

一个完整的数据项条目应该包括名称、描述、数据类型、取值范围及默认值、精度和计量单位。

一个完整的数据存储条目应该包括名称、描述、数据存储方式、关键码、频率和数据量、安全性要求和数据结构。

例,数据流条目: 课程 = 课程名 + 教师 + 教材 + 课程表

数据项条目: 阿拉伯数字 = [0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9]

虽然我们在数据词典中定义和说明了各种处理,并用文字进行了概括描述。但是,对于某些处理功能,只用文字说明还存在许多含糊不清之处,此时,应借助一些逻辑描述工具来表达。常用的加工逻辑描述工具有结构化英语或结构化语言、判断表、判断树等。

3. 其他表述方法

(1) 结构化英语或结构化语言

形式化语言严格精确,但不易被用户理解。自然语言容易理解,但不精确且可能有二义性。结构化语言是介于自然语言和形式化语言之间的类自然语言。它虽不如形式化语言精确,但具有自然语言简单易懂的优点,又避免了自然语言的一些缺点。

结构化英语或结构化语言没有严格的语法。结构化英语或结构化语言的结构通常分为内外两层。外层语法比较严格,用于描述操作的顺序、选择和重复的控制。内层语法比较灵活,使用规定的语法和词汇,也可以使用数据词典中定义过的词汇、运算符、关系符和一些易