前　言

我国在工业界推行计算机集成制造(CIM)已经8年了,很多企业取得了令人瞩目的成就。如北京第一机床厂在1995年获得了美国SME的工业领先奖,成都飞机公司、沈阳鼓风机厂、经纬纺织机械厂等企业都获得了国家科委的CIMS应用工程领先奖,他们都从CIM实施中取得了很好的效益。但是,并非所有实施CIM的企业都能取得很好的效果。这里原因是多种多样的,有社会大环境的影响,有决策的正确与否;而缺乏正确有效的分析方法和实施集成的工具,也是一条重要的因素。

从1990年,我们刚开始在国内工厂重点应用CIMS时,"863"CIMS专家组就规定了,所有CIMS应用工厂,都必须应用IDEF0方法建立功能模型、进行需求分析。有的企业对这一工具用得较好,现在确实尝到了它能帮助做好系统集成的甜头。但也有不少企业只是奉命画图,画了几百张IDEF0图,却只是在评审时用一下,以后就束之高阁,不再起任何作用了。从国内外实施CIM的实践经验中,我们充分意识到系统集成中设计分析工具的重要性。当然,我们不是把这里所说的工具和方法局限于IDEF方法,各个国家和地区都有一些很有效的系统分析方法,但其中IDEF系列是应用最广的。我们在1991年曾编写了"IDEF0及IDEF1x——复杂系统通用的设计分析方法"(陈禹六,周之英,裴少鹏,俞盘祥,汤荷美编,电子工业出版社出版),作为90年代头几年CIMS重点应用工厂的暂行规范,起了它应有的作用。现在国内越来越多的企业开始实施CIM,因此有必要根据这几年CIM实践的经验,将IDEF0和IDEF1x方法的说明,进行修改后再版。IDEF2方法由于其内容已被广泛应用的仿真软件SLAM Ⅱ和SIMAN所覆盖,实际上已自然淘汰。而美国KBSI(knowledge based systems Inc.)推出的IDEF3过程描述获取方法、IDEF4面向对象设计方法,已经在国际上被越来越多的人了解和应用。当前,面向对象技术在计算机技术领域中应用日益广泛,作用越来越突出,亟需系统地将它介绍给CIMS领域的工程技术人员。另外,从系统集成技术的进展来说,很多CIMS专家都已提出了要在信息集成、功能集成,进到更多(或者说更深入)考虑过程集成的问题。因此,用于过程描述的IDEF3方法,将会在今后的CIM实践中起到越来越大的作用。IDEF3解决了IDEF0不能反映时间和时序的问题,因此可以和仿真软件相结合,用来检验过程的合理性并指导过程重构,实现优化。KBSI公司对IDEF3,IDEF4这两种建模方法,在1992年就已提供了完整的手册,张锋同志和赵虹同志分别对IDEF3和IDEF4的手册进行了初译;然后,我校电机系黄益庄教授对IDEF4作了详细的校订;笔者对IDEF3初稿作了部分改写,对翻译作了仔细的修改和全面的校订。

IDEF5本体论获取方法,也已有完整的手册,但因为与实用还有一定的距离,笔者只是编写介绍了其基本概念,而由吴笑一同志初译了IDEF5手册中的第4章"本体论语

言",笔者进行了全面的修改和校订。关于其他几种方法,本书只是在"后记"中对 IDEF9、IDEF6、IDEF14 和 IDEF8 作了一些概念性的简介。

KBSI 正在不断开发完善这整个系列。我们在取得更完整的资料及有了一定的实践经验后,还会向读者作进一步介绍。

除了 IDEF0 和 IDEF1x 之外,对其他 IDEF 方法,我们的实践经验还很少,书中错误和不当之处一定不少,恳切希望读者来信批评指正。

作者

1998 年 11 月

目　　录

第一篇　IDEF0 方法

第二篇　IDEF1X 方法

第三篇 过程描述获取方法 IDEF3

第四篇 IDEF4 面向对象的设计方法

第五篇 本体论获取方法

第一篇

IDEF0方法

第1章 引 言

1.1 背 景

IDEF 的基本概念是在 70 年代提出的结构化分析方法基础上发展起来的。结构化分析方法在许多应用问题中起了很好的作用。在降低项目开发费用，减少系统中的错误，促进交流的一致性及加强管理等方面都产生了效益。1981 年美国空军公布的 ICAM (integrated computer aided manufacturing) 工程中用了名为“IDEF”的方法。IDEF 是 ICAM DEFinition method 的缩写。后来就称之为 integration definition method，简称不变。刚开始时，此方法由 3 部分组成：

(1) IDEF0 描述系统的功能活动及其联系，在 ICAM 中建立加工制造业的体系结构模型，其基本内容是 SADT(system analysis and design technology)的活动模型方法。这是由 Softech 公司发展起来的。

(2) IDEF1 描述系统信息及其联系，建立信息模型作为数据库设计的依据。这是由 Hughes 飞机公司为主发展起来的。

(3) IDEF2 用于系统模拟，建立动态模型。这是由 HOS 公司为主发展出来的。

现在 KBSI 公司正在继续将此方法发展成一个系列，列写如下：

IDEF0	功能模型(function modeling)
IDEF1X	数据模型(data modeling)
IDEF2	仿真模型设计(simulation model design)
IDEF3	过程描述获取(process description capture)
IDEF4	面向对象设计(object－oriented design)
IDEF5	本体论描述获取(ontology description capture)
IDEF6	设计原理获取(design rationale capture)
IDEF7	信息系统审定(information system auditing)
IDEF8	人与系统接口设计(human－system interface design) 用户接口建模(user interface modeling)
IDEF9	经营约束的发现(business constraint discovery) 场景驱动信息系统设计(scenario－driven IS design)
IDEF10	信息制品建模(information artifact modeling) 实施体系结构建模(implementation architecture modeling)
IDEF11	信息工具建模(information artifact modeling)
IDEF12	组织设计(organization design)

	组织建模(organization modeling)
IDEF13	三模式映射设计(three schema mapping design)
IDEF14	网络设计(network design)

1.2 IDEF0 的基本特色

IDEF0 的基本思想是结构化分析方法,来源于 SADT 方法。它具有以下基本特色,这些特色形成一种思维规则,适用于从计划阶段到设计阶段的各种工作。

1 全面地描述系统,通过建立模型来理解一个系统

一般地说,一个系统可以被认为是由对象物体(用数据表示)和活动(由人、机器和软件来执行)以及它们之间的联系组成。但如果是从数据或活动来描述,那至多只反映了一个侧面,这样的技术很难说明系统的全貌。IDEF0 能同时表达系统的活动(用盒子表示)和数据流(用箭头表示)以及它们之间的联系。所以 IDEF0 模型能使人们全面描述系统。

对于新的系统来说,IDEF0 能描述新系统的功能及需求,进而表达一个能符合需求及能完成功能的实现。对已有系统来说,IDEF0 能分析应用系统的工作目的,完成的功能及记录实现的机制。在这两种情况下都是通过建立一种 IDEF0 模型来体现。所谓模型就是系统的一种书面描述。它不一定必须用某种数学公式表示,可以是图形,甚至可以是文字叙述。因而我们可以说:"不管何种形式,只要 M 能回答有关实际对象 A 的所要研究的问题,就可以说 M 是 A 的模型"。对于有关复杂的企业对象或其他系统,由于用自然语言无法精确又无二义性地表示分析及设计结果,所以这里采用一种图形语言来表示 IDEF 模型。这种图形语言能做到:

- 有控制地逐步展开细节;
- 精确性及准确性;
- 注意模型的接口;
- 提供一套强有力的分析和设计词汇。

一个模型由图形、文字说明、词汇表及相互的交叉引用表组成,其中图形是主要成分。IDEF0 图形中同时考虑活动、信息及接口条件。它把方盒作为活动,用箭头表示数据及接口。因此在表示一种当前的操作,表示功能说明或设计时,总是由一个活动模型、一个信息模型及一个用户接口模型组成。

工程界对系统开发过程一般可安排几个阶段:分析(确定系统将做什么)、设计(定义子系统及其接口)、实现(独立地创建子系统)、集成(把子系统联接成一个整体)、测试(证明系统能工作)、安装(使系统能运行)和运行(使用系统)。我们介绍 IDEF 方法的目的,在于强调科学地进行复杂系统的分析和设计的重要性。一般来说,在分析或设计阶段造成的错误,在后续阶段可能花 2 倍时间去找到,花 5 倍时间去纠正。或者如人们所说的:分析阶段的一个错误未被纠正,在设计阶段要花 2 倍时间,测试阶段要花 10 倍时间,运行或维修阶段要花 100 倍的时间才能纠正。

2 目的与观点(purpose and viewpoint)

由于模型是一个书面说明,因此像一切技术文件一样,每一个模型都有一个目的与一

个观点。目的是指建模的意义，为什么要建立模型。观点是指从哪个角度去反映问题或者站在什么人的立场上来分析问题。功能模型是为了要进一步做好需求分析，要实现预定的技术要求（不论是对已有系统的改造还是新建系统），所以要明确是对功能活动进行分析（逐步分解），而不是对组织机构的分解。一个活动可能由某个职能部门来完成，但活动功能不等于组织，因此必须避免画成组织模型的分解过程。模型描述的内容反映各种用户的要求，从单一角度描述问题是困难的，也是不可能的。如：

物资管理人员——仓库管理员关心收、发、存；

计划人员关心什么时候物料从库存点到采购点；

厂长关心哪一个工程项目节约用料，加快进度。

因此要求所有的用户，有同样重要而且相同的需求是不可能的，也不切实际。IDEF0 要求在画出整个系统的功能模型时，具有明确的目的与观点。譬如，对一个企业的 CIM 系统，必须有明确的站在厂长（或经理）的位置上建模的观点，所有不同层次的作者都要以全局的观点来进行建模工作，或者说就是为厂长而建模。这样才能保证是从全企业的高度来揭示各部分之间的相互联系和相互制约的关系。否则有的人强调设计处的利益，有的人突出供销处的要求，甚至有的可以只为某个岗位的操作人员的要求来建立各功能模块之间的联系，那就整个乱套了。

3　区别“什么”（what）和“如何”（how）

“什么”是指一个系统必须完成的是“什么”功能，“如何”是指系统为完成指定功能而应“如何”建立。就是说，在一个模型中应能明确地区别出功能与实现间的差别。

IDEF0 首先建立功能模型。把表示“这个问题是什么”的分析阶段，与“这个问题是如何处理与实现”的设计阶段仔细地区别开来。这样，在决定解法的细节之前，保证能完整而清晰地理解问题，这是系统成功开发的关键所在。

在设计阶段，要逐渐识别各种能用来实现所需功能的机制，识别选择适当机制的依据是设计经验及对性能约束的知识。根据不同模型机制可以是很抽象的，也可以是很具体的。重要的是，机制指出了“什么”是“如何”地实现的。IDEF0 提供了一种记号，来表示在功能模型中如何提供一个机制来实现一个功能，及单个机制如何能在功能模型的几个不同地方完成有关功能。

有时机制相当复杂，以致机制本身需要进行功能分解。

4　自顶向下分解

用严格的自顶向下地逐层分解的方式来构造模型，使其主要功能在顶层说明，然后分解得到逐层有明确范围的细节表示，每个模型在内部是完全一致的。

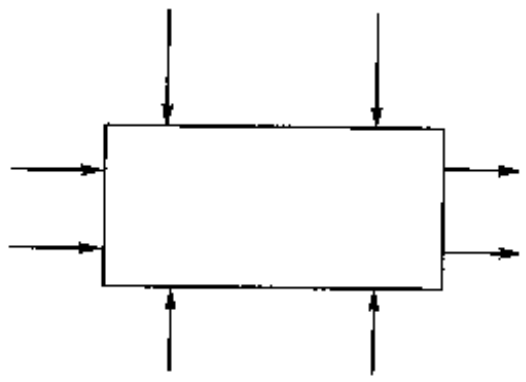


图 1.1.1　盒子及其接口箭头

IDEF0 在建模一开始，先定义系统的内外关系和来龙去脉。用一个盒子及其接口箭头来表示，确定了系统范围，如图 1.1.1。由于在顶层的单个方盒代表了整个系统，所以写在方盒中的说明性短语是比较一般的、抽象的。同样，接口箭头代表了整个系统对外界的全部接口。所以写在箭头旁边的标记也是一般的，抽象的。然后，把这个将系统当作单一模块的盒子分解成

另一张图形。这张图形上有几个盒子，盒子间用箭头连接。这就是单个父模块所相对的各个子模块。这些分解得到的子模块，也是由盒子表示，其边界由接口箭头来确定。每一个子模块可以同样地细分得到更详细的细节。如图 1.1.2。

IDEF0 提供的规则，保证了如何通过分解得到人们所需要的具体信息。一个模块在向下分解时，分解成不少于 3 个、不多于 6 个的子模块。上界 6，保证了采用递阶层次来描述复杂事物时，同一层次中的模块数不会太多，以致不适宜于人的认识规律。下界 3，保证了分解有意义的。但是，原始的 SADT 方法，规定一张图上的盒子数为 2 个～7 个，故我们也不作很硬性的限制。

模型中一个图形与其他图形间的精确关系，则用互相连接的箭头来表示。当一个模块被分解成几个子模块时，用箭头表示各子模块之间的接口。每个子模块的名字加上带标签的接口，确定了一个范围，规定了子模块细节的内容。

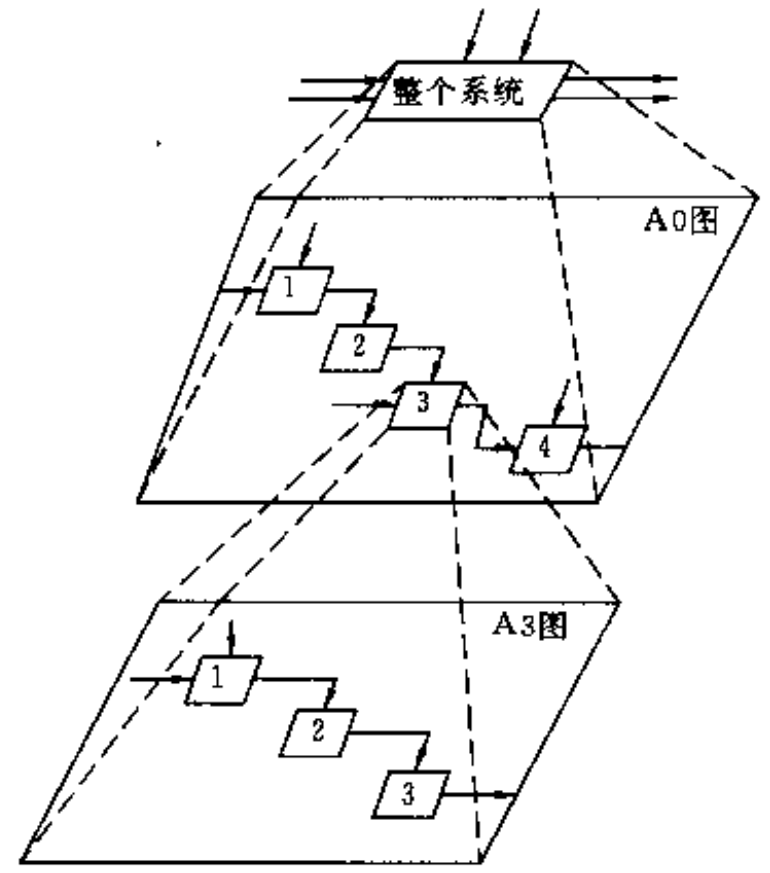


图 1.1.2　递阶分解关系

在所有情况下，子模块忠实地代表了父模块，以既不增加也不减少的方式反映着各自父模块所包含的信息。

5　严格的人员关系，评审手续及文档管理办法

（1）人员：IDEF0 适合于研究分析一个大而复杂的系统，因此要求有一个技术上熟练、而且能相互协调的集体来一起工作。这个集体应有各方面的人员组成。通常，将人员分成以下几类：

- 作者(author)：研究需求及限制条件，分析系统功能，建立 IDEF0 模型。
- 评审员(commentor)：也可以是其他图的作者。主要是进行复审，并写出对其他人所做工作的书面意见。是广义的读者。
- 读者(reade)：读 IDEF0 图，口头上提出意见。没有提书面意见的义务，读图的目的主要是为了互相了解，互相协调。
- 专家(expert)：作者对专家进行访问，了解需求、限制条件等专门信息。
- 技术委员会(technical committee)：对每个主要分解阶段进行复查，并对项目管理作技术决策，仲裁作者和读者间不能协商一致的分歧。
- 项目资料员(project librarian)：(或称“文档管理员”)维护文件，复制分配材料及记录。
- 项目负责人(project manager)：负有分析及设计系统的技术责任，也是技术委员会的主席。

（2）评审手续：建模活动每前进一步，IDEF 方法都要求这个集体成员交换见解，用以互相检查工作的结果，有名的作者/读者循环就体现了这个工作程序。如图 1.1.3。

作者访问专家，画出系统的 IDEF0。由资料员编成文件存档，分发给评审员及读者。评审员把加上意见的材料退还给作者，同时由资料员存档。作者根据意见修改图形，反复循环，直至这一层问题全部解决，再送给作者准备下一步的分解。第一个作者可以是下一层的某个作者或较低层的评审员。最后由技术委员会来解决必要的技术问题及技术分歧。

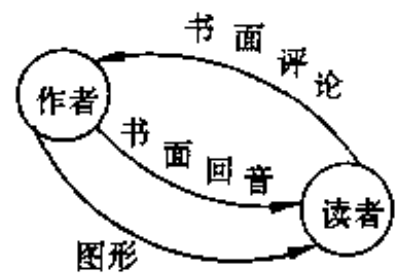


图 1.1.3 作者/读者循环

(3) 文档：无论是作者的模型，还是评审员的评论，都要以书面的形式反映出来。每次修改意见都要保存，一面工作，一面把文档建立起来。

以上几个方面构成了 IDEF0 方法的基础。它们相互补充，失去其中任一个都会降低 IDEF0 方法的效用。因此理解好这几方面的概念是成功地运用 IDEF0 方法的前提。

当然，正如引进先进技术通常都有一个“国情化”的步骤一样。由于我国生产、设计的管理体制与欧美不同，因此有些做法可以在理解其基本精神的基础上，结合具体情况做出我们自己的规定。譬如，作者/读者循环，IDEF0 方法没有规定必需有几个评审员，每个评审员的身份是否有区别。我们则可以根据中国的习惯，规定三个读者，第一读者是本设计组内同级的技术人员，主要审查模型的内容、命名、标注、及关系等，是否正确；第二读者可能是标准化或规范组对 IDEF0 方法很熟悉的人，进行规范的形式审查；第三读者则可以是设计组长或分系统主任设计师，从全面要求和不同部分间的相互关系，进行审查；然后项目负责人再作审批(在“审阅”栏签字)。再则，对一个子系统的全套图纸还应由相关业务部门或其他子系统进行会签，以保证全局范围的协调和集成。最后对整个集成系统的图纸形成组表文件(见第四章)，还需交上级领导批准。

第 2 章　IDEF0 图形的意义

IDEF0 是由结构化分析方法得到的建模方法。该模型由一系列图形组成。图形的元素主要是简单的盒子及箭头。我们把描述功能活动的图形称为活动图形(activity diagrams)。在活动图形中，盒子表示活动，而箭头表示由系统处理的事件。我们把事件看作任何可以用名词命名的数据。这些数据可以代表具体事物，也可以是抽象的信息。活动是指用动词或动宾短语命名的任何动作事项。

2.1　基 本 概 念

1　活动图形

模型由一系列图形组成，它把一个复杂事物分解成一个个部分、成分。初始图形最一般或最抽象地描述了整个系统。它把每个主要成分表示成一个盒子。每个成分经过分解再用一个图形来显示它的细节。这个图形把它的组成成分也用盒子来表示。此盒子又能分裂成更多的图形，直到系统描述得足够细致为止。

每个详细图是较抽象图中一个盒子的分解图，每一步中我们把抽象图称为详细图的父图。把详细图称为抽象图的子图。

用这种图形来描述的一个系统称为模型。每个模型必须说明一组特定的需求，如：

- 描述系统完成的是什么功能。
- 说明系统是如何设计的及如何构造的。
- 解释如何使用及维护一个系统。

在图形中盒子代表系统功能(活动)。箭头代表数据(信息或对象)。一个上层图中的盒子由下层图中一系列盒子及箭头来说明。进入与离开上层图中盒子的箭头完全与进入与离开下层图中的箭头一致。因为它们是代表系统的同一部分的。

结构化分析的原则之一，是图形中盒子不多于 6 个，因此活动图形具有抽象性。高层图形包含了相当多的信息，因此在高层图中，盒子及箭头所附带的文字标签必然是抽象的，它描写的是一般概念。而下层图中信息逐步增加。因此使用更为专门的术语。

2　盒子(box)

图形中的盒子代表活动，用主动的动词短语来描述，我们在盒子的内部写上描述盒子的短语，在盒子的右下角写上编号，从 1 编到 6。

连到盒子上的箭头，表示由活动产生的、或活动所需要的、信息或真实对象。用一个名词短语作标签，写在箭头旁边。“数据”可以是信息、对象或任意用名词或短语描述的任何东西。箭头限制了盒子间的关系，而不是表示活动的顺序。

盒子的边表示所进入或离开的箭头的作用。它们分别是输入、控制、输出与机制。如图 1.2.1 所示。

盒子左边及上方进入的箭头表示为完成此活动所需要的数据。盒子右边离开盒子的箭头表示执行活动时产生的数据。输入与输出的关系就是活动，是将输入数据转变为输出数据的一种变换。所以“输入”就是这个活动需要“消耗掉”、要“用掉”或“变换成”输出的东西，它可以是具体的事物、抽象的数据或其他用名词表示的东西。

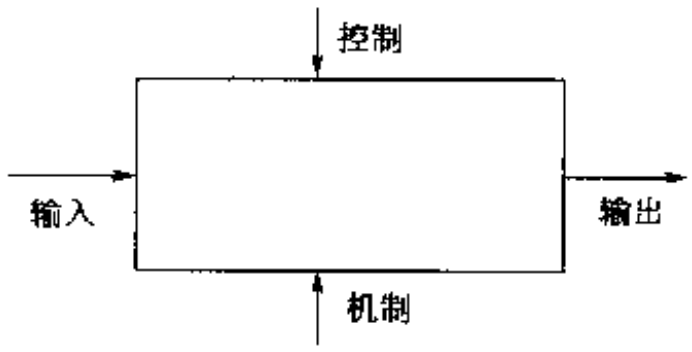


图 1.2.1　箭头名称

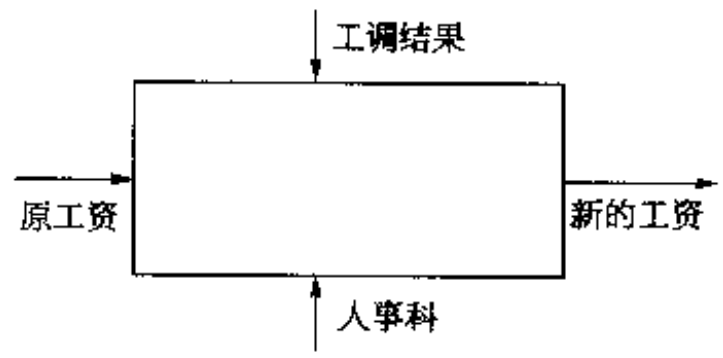


图 1.2.2　箭头名称举例

控制说明了控制变换的条件或环境，或者说是约束。输入与控制二者的作用是不同的，例如：图 1.2.2 例中，“工调结果”是指增薪 20 元的控制条件，“原工资”是 80 元，则“新的工资”是 100 元。

理解输入、控制二者不同的含义，对理解系统的工作是重要的。在输入与控制无法明确区分时，可看作控制。每个活动盒子至少有一个控制箭头。或者说，一个活动可以没有输入，但不允许既没有输入又没有控制。

盒子底部的箭头作为机制，它可以是执行活动的人或设备。图 1.2.3 给出了又一个例子。

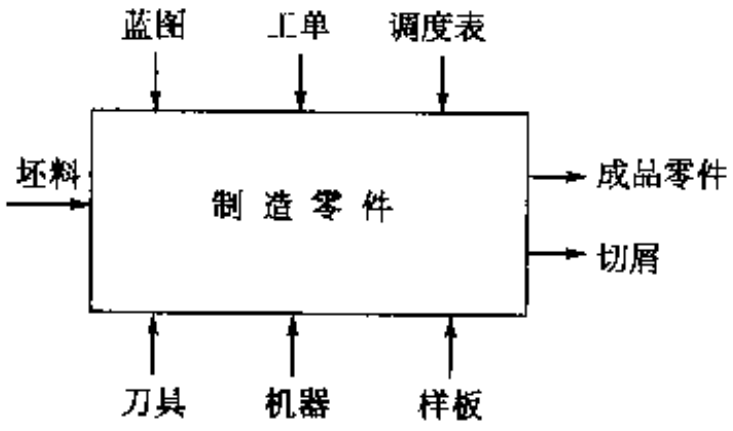


图 1.2.3　箭头名称又一例

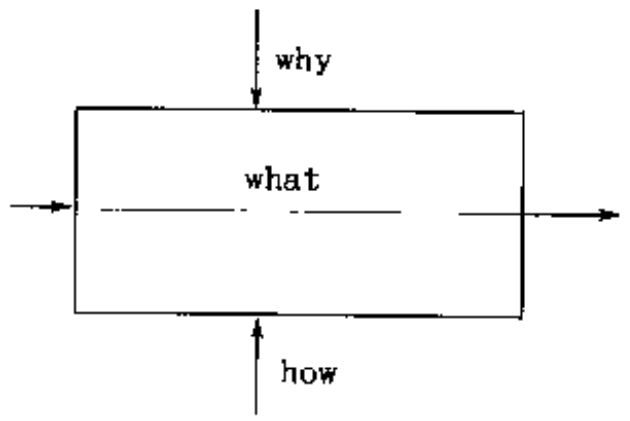


图 1.2.4　箭头作用

总之，输入输出箭头表示活动进行的是什么(what)，控制箭头表明为何这么做(why)而机制箭头表示如何做(how)。如图 1.2.4 所示。

盒子表示了一组相互有关的活动，而不一定是单一的作用。在不同条件下，不同环境下，用不同的输入或控制可以产生不同的输出，执行功能的不同部分。这称为对盒子的不同激活。这种情况发生在盒子的每一边上有多个箭头的场合。

3　箭头

在活动图形上，箭头代表数据约束，而不是代表流或顺序。一个盒子上的输出，连到另

…盒子的输入或控制，表示一种约束。表示接受数据的盒子的执行条件，可利用输出该数据盒子所产生的数据。在同一图上，若几个盒子所需的约束都满足时，几个活动可以同时执行。一个盒子的输出可以提供一个或多个盒子所需的一部分或全部数据。

活动图中没有明确的顺序及时间。但我们很容易表示如反馈、迭代、连续处理及时间上重叠等情况。

数据流可以有集合性。在活动图上箭头可以有分支，表示多个活动需要同一数据。图 1.2.5 表示 A 同时进入模块 2 及模块 3。

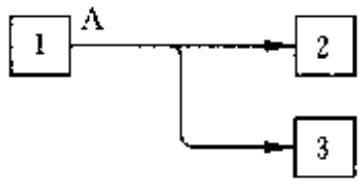


图 1.2.5　数据流的分流

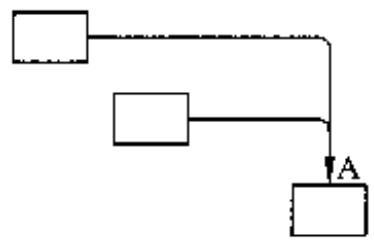


图 1.2.6　箭头的汇合

箭头可以联合，以表示多个活动产生同一类数据(或合成为一类数据)。如图 1.2.6 所示。

箭头分支可以代表一类东西或同一种类的不同东西。

图 1.2.7 表示 A 进入模块 2，B 进入模块 3。

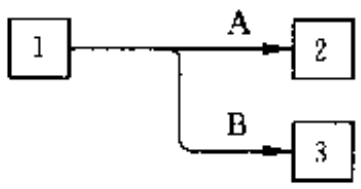


图 1.2.7　箭头的分支

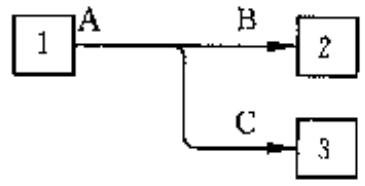


图 1.2.8　箭头的另一种分支

图 1.2.8 表示 B，C 为 A 的子集 B∪C＝A。

箭头有二类。一类称内部箭头，它的两端分别连在图形的盒子上。另一类称边界箭头，它的两端中一端是开的，表示由图形以外的活动所产生，或供图形以外的活动所使用。

4　机制箭头

在盒子底部的机制箭头，指出活动是由什么来完成的。通常先不画机制箭头，这种图形表示一个系统执行什么功能，然后，再选择适当的机制来表示功能的实现方案。

向下指的机制箭头(称调用[call]箭头)指出完整地执行盒子所代表功能的处理器。它已经在另一个模型中进行了细化，如需要了解细节，就可按调用的图号(或结点号)在另一个模型中找到有关图形。如图 1.2.9 所示。

如果一个盒子准备好一些处理器，则这盒子的输出也可以作为另一盒子的机制箭头。通常机制模型是多模型分析或设计的一个单独部分。

5　通道箭头

把一个箭头在盒子的连接端加上括号，则此箭头将不出现在子图的边界箭头中。这个括号表示该箭头将通到模型的未定义部分，与下一个子图无关。或者是众所周知，或有共同理解的可省略的内容，在子图中为简化图面而省略了。

如果箭头出现在子图中，并作为边界箭头，但不出现在父图中，则在子图箭头的开端

加上括号，表示该箭头是子图的一个必要接口，但与父图没有关系。或有共同理解，在父图中不表示亦无妨。

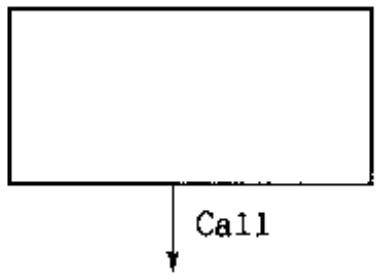


图 1.2.9　调用[call]箭头

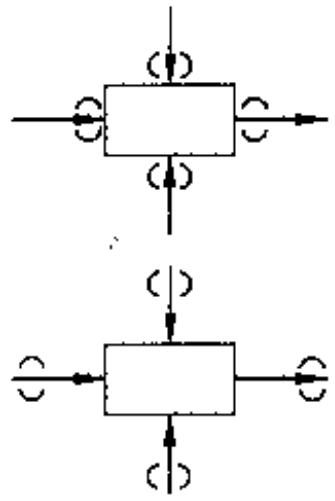


图 1.2.10　通道箭头

这些箭头，使人们在分解中能推迟表示它们所代表的信息，只是到了非表示不可的时候，才将它们表示出来。这些箭头的使用，避免了在高层次图形中出现信息过细的现象。而过细的信息往往会干扰人们对图形含义的正确理解。在某一层次信息不必出现时，可以不列入，直至需要说明时，才表现出来。这类箭头，称为“通道箭头”，如图 1.2.10 所示。

有时，带括号的箭头在一个或多个详细图中都不出现，而到某个详细图中才以带括号的形式再度出现。当然，这种通道箭头一般使用得比较少。

6　双向箭头

在图形中对互为输入，或互为控制的两个盒子可用双向箭头连接。

图 1.2.11 是互为输入。首先触发的盒子画在上面，后触发的放在后面。两个“·”强调注意。

图 1.2.12 是互为控制。

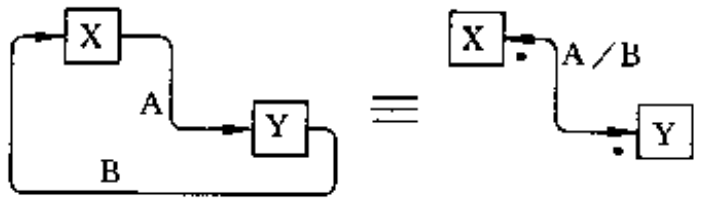


图 1.2.11　互为输入的双向箭头

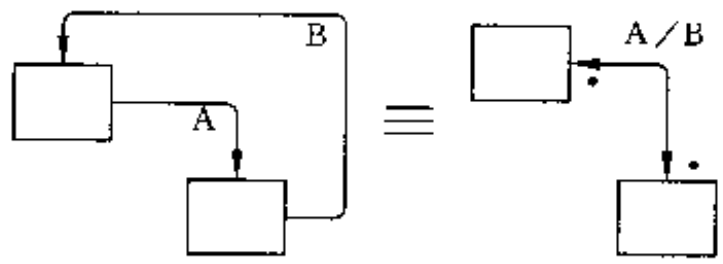


图 1.2.12　互为控制的双向箭头

7　虚箭头表示触发顺序

用虚线表示虚的输出控制。图 1.2.13 触发顺序为 A→B→C。

8　选择箭头

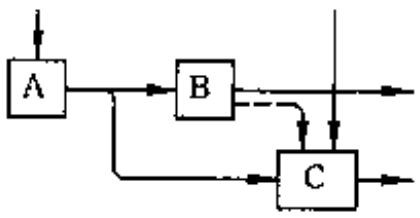


图 1.2.13　虚箭头

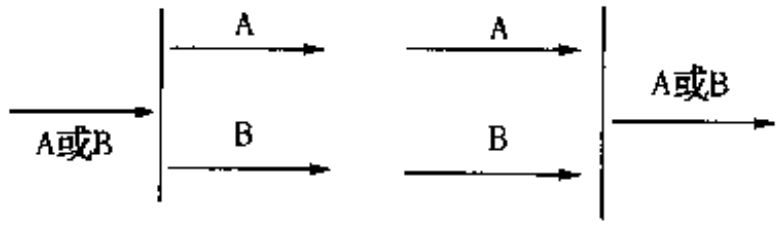


图 1.2.14　选择箭头

9 ICOM 码

边界箭头是有一个开端(即只有一端与盒子相连)的箭头。它表示父盒子的输入、控制及输出等箭头。要找到边界箭头的源及始点就要检查父图的内容。换句话说,有开端的边界箭头一定与父图中父盒子的箭头一致。我们用专门的符号称为 ICOM 码来说明父子图中的箭头关系。把子图中每个边界箭头的开端分别用字母 I,C,O,M 来标明是父盒子的输入、控制、输出及机制。再用一数字表示父盒子上箭头的相对位置。例示于图 1.2.15。编号次序是从上到下、自左到右。例如图 1.2.15 中盒子 H 的控制是 C1,表示是父盒子 A 的第一个控制 a。在子图中用到父图 ICOM 的顺序,不一定与父图一致,只是编号必须准确地对应。如子图中 C2(父图中的 b)就用于子图的第一个盒子 B,放在了 C1 的左边。

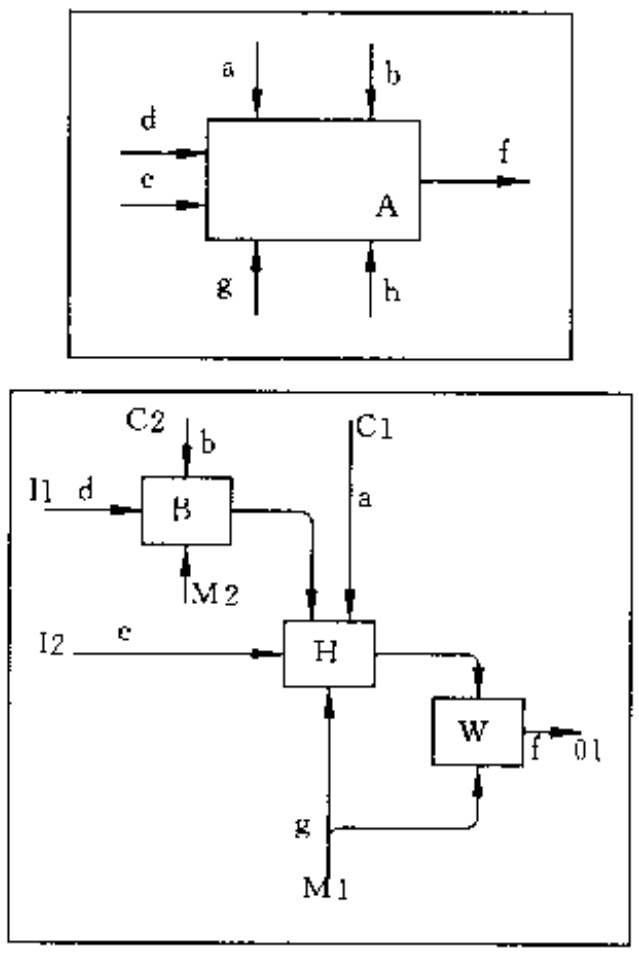


图 1.2.15 ICOM 码

10 结点号

一个模型是一组有一定层次的图形。我们用结点号来标志图形或盒子在层次中的位置。结点号是由盒子的编号推导出来的。活动图的所有结点号都用字母 A 开头。最顶层图形为 A0 图,在 A0 以上只用一个盒子来代表系统内外关系的图,编号为 A－0。(读作"A 减 0")(必要时还可有 A－1,A－2,甚至 A－4 图,而此时模型的顶层仍是 A0 图)。每个结点号是把父图的编号与父模块在父图中的编号组合起来。也就是说"父→子→孙……"每增加一"代",结点号的位数就增加一位。形成如图 1.2.16 的结点树。

此外还常用到其他结点号。如"参照图"用"FEO"(for exposition only),表示在层次图形以外的图形。FEO 图一般更为灵活,可以有多于 6 个盒子的图形。只要作者认为说明一个问题所必需的内容都可包含入内。此时结点号上要加 F(例如 A2F),当有多个 FEO 时,可在 F 后面编号(如 A2F1,A2F2,…)。另外,文字说明的文本号应加上 T(如 A2T),

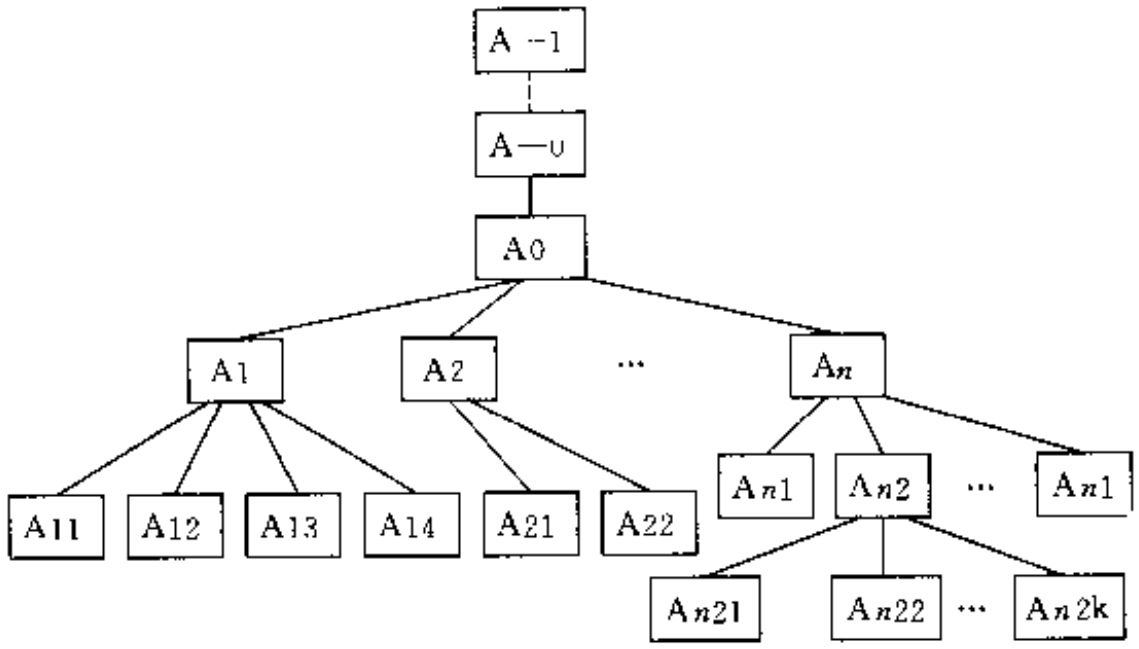


图 1.2.16　结点树

每张图的文字说明一般不要超过一页。

11　模型名

每个模型有一个名字。通常用名字代表主体，用子名字表示不同的模型。基本名字与子名字间用斜杠(/)隔开。对每个图形或图形中方盒则再加上一道斜杠联上结点号。例如

TOPIC/A3

TOPIC/REQTS/A42

其中 TOPIC 是主体，REQTS 是模型名，A3，A42 是结点号。这种形式称为引用表达式。

2.2　如何读图

建议用以下阅读图形的步骤：

1　查看当前图中的各盒子，得到描写事物的一个概貌。

2　参看父图，注意有关箭头。试着识别一个最重要的输入、控制及最重要的输出。

3　回到当前图，试着确定是否有一条主路径连接最重要的输入或控制，达到最重要的输出。

4　把主路径作为线索从左上角到右下角遍历这个图形。注意与盒子有关的其他箭头。确定是否存在第二条路径。检查图形所表明的全部内容。

5　最后读补充文本，使理解更加完整。

在读图时要注意有无 FEO 图。要仔细阅读以了解图形的目的。

2.3　实　　例

附 2 张计算机集成制造系统的 IDEF0 图，图 1.2.17，图 1.2.18，供读者参考。

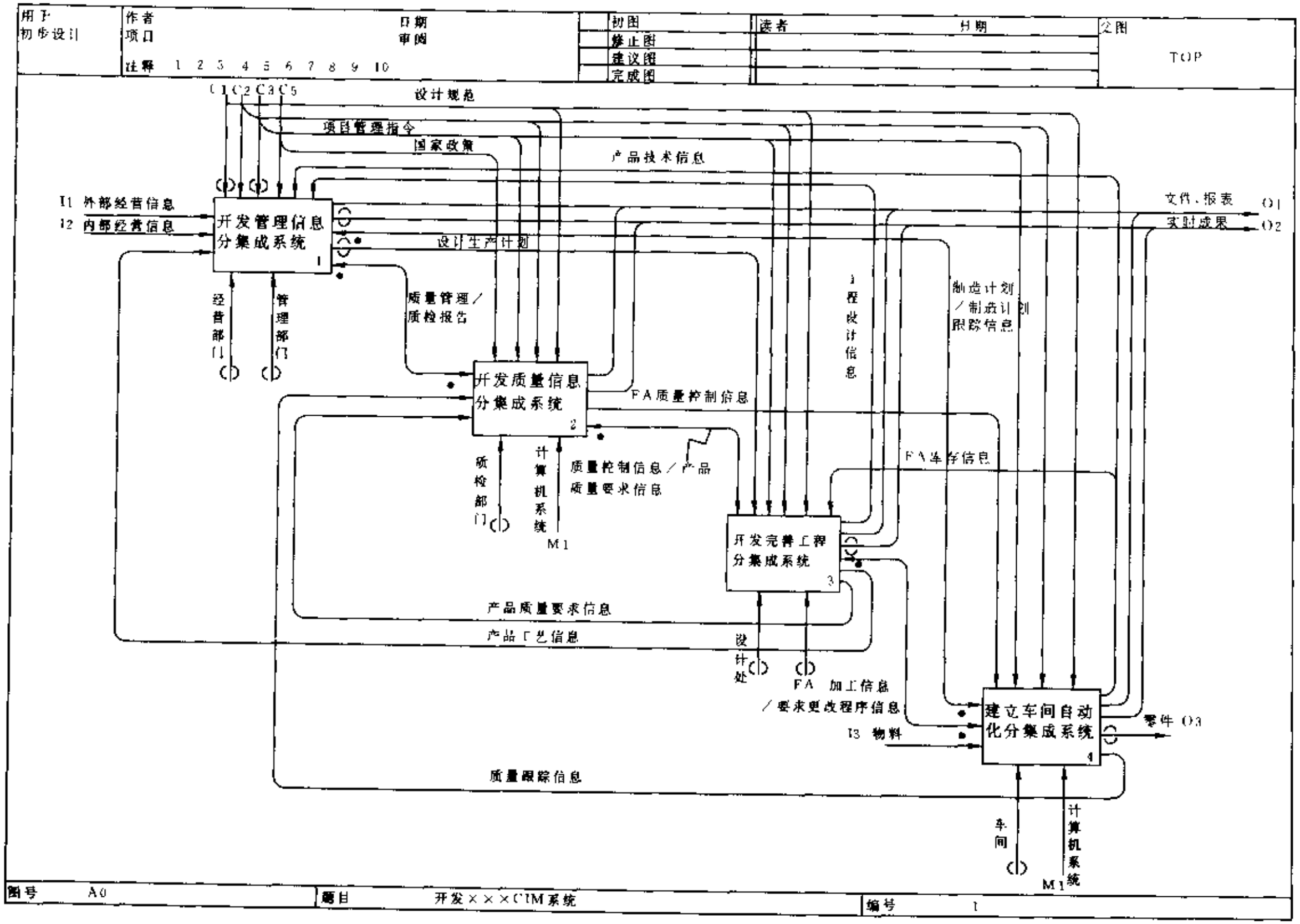

用于
初步设计
作者
项目
日期
审阅
注释 1 2 3 4 5 6 7 8 9 10
初图
修正图
建议图
完成图
读者
日期
父图
TOP
C1 C2 C3 C5
设计规范
项目管理指令
国家政策
产品技术信息
I1 外部经营信息
I2 内部经营信息
开发管理信息分集成系统
1
设计生产计划
文件、报表 O1
实时成果 O2
经营部门
管理部门
质量管理/质检报告
开发质量信息分集成系统
2
FA质量控制信息
工程设计信息
制造计划/制造计划跟踪信息
质检部门
计算机系统
M1
质量控制信息/产品质量要求信息
FA库存信息
开发完善工程分集成系统
3
产品质量要求信息
产品工艺信息
设计处
FA 加工信息/要求更改程序信息
建立车间自动化分集成系统
4
I3 物料
零件 O3
质量跟踪信息
车间
计算机系统
M1
图号 A0
题目 开发×××CIM系统
编号 1

图 1.2.17 实例之一

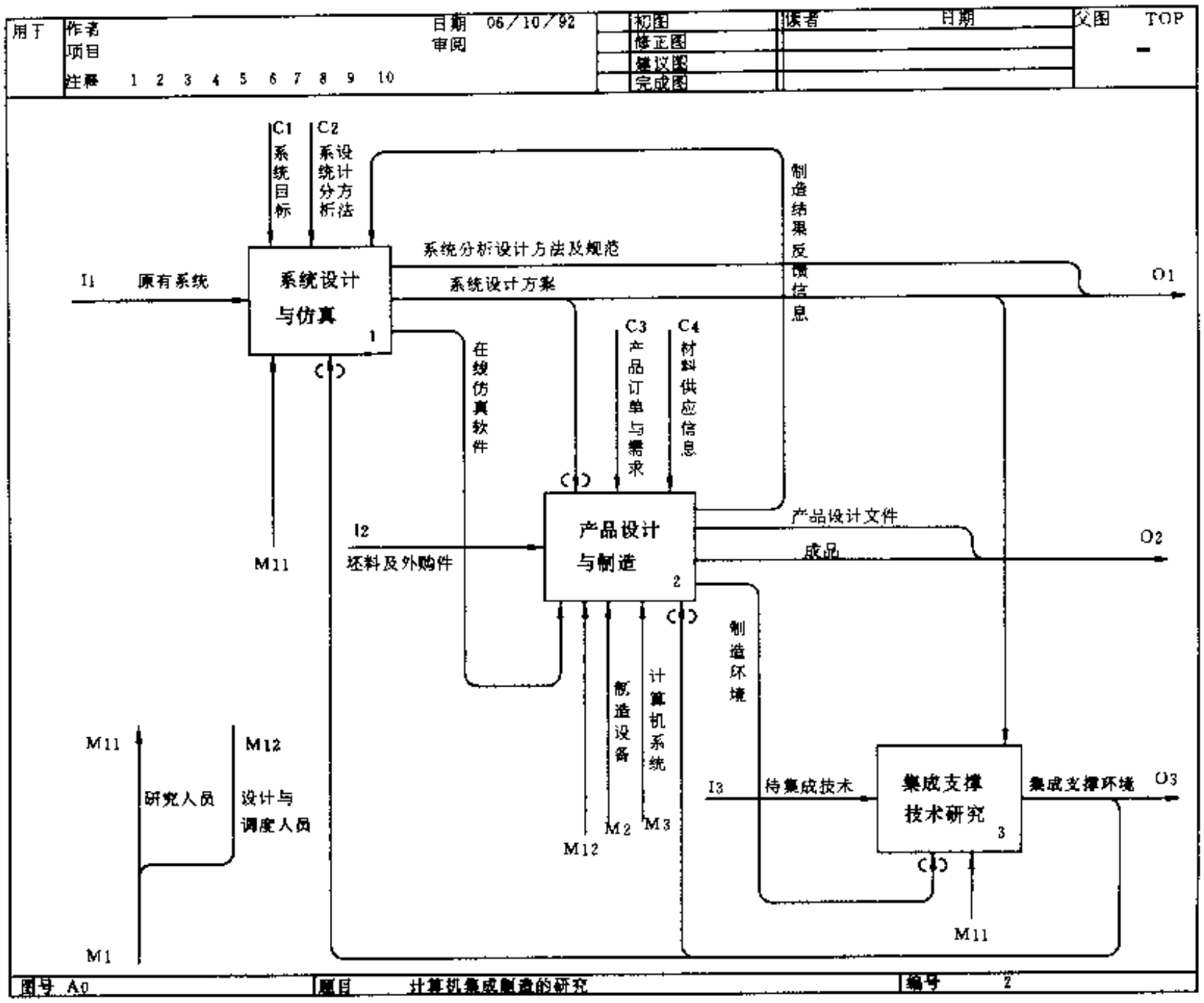

用于
作者
项目
注释 1 2 3 4 5 6 7 8 9 10
日期 06/10/92
审阅
初图
修正图
建议图
完成图
读者 日期
父图 TOP
C1 系统目标
C2 系设统计分方析法
制造结果反馈信息
系统分析设计方法及规范
I1 原有系统
系统设计与仿真 1
系统设计方案
O1
在线仿真软件
C3 产品订单与需求
C4 材料供应信息
M11
I2 坯料及外购件
产品设计与制造 2
产品设计文件
成品
O2
制造环境
制造设备
计算机系统
M12
M2
M3
M11 研究人员 M12 设计与调度人员
M1
I3 待集成技术
集成支撑技术研究 3
集成支撑环境
O3
M11
图号 A0
题目 计算机集成制造的研究
编号 2

图 1.2.18 实例之二

第 3 章　作者建模方法

负责建立模型的人称为作者。建立活动模型的基本目标是描述系统的功能。我们用 IDEF0 图来描述一个系统或系统的功能，通常就称为功能模型。本章将说明建立这一模型的一系列步骤。

3.1　建立活动模型的基本方法

1　选择范围、观点及目的

在开始建立模型之前，确定模型的立足点是很重要的。这包括要确定范围、观点及目的。“范围”把模型的主题作为更大的总体的一部分来看待，描述了外部接口，建立了与环境之间的界线，确定了什么是讨论的问题与什么不是讨论的问题。“观点”确定了从什么角度去观察问题，以及在一定范围内能看到什么。目的是确定模型的意图或明确其交流的目标。“目的”包含了为什么要建立模型的理由（功能说明，实现设计或顾客操作等），这三个概念指导并约束了整个建模过程。虽然在建模过程中这些内容也可以反复推敲，但必须是自始至终保持一致、清晰，而不受曲解。

2　建立内外关系图——A－0 图

建模的第一步通常是建立内外关系图——A－0 图。画一个单个的盒子，里面放上活动的名字。名字要概括所描述系统的全部内容。再用进入及离开盒子的箭头表示系统与环境的数据接口。这个图形确定了整个模型的内外关系，确定了系统的边界，构成进一步分解的基础。

有人首先从初始需求及约束条件中画出 A－0 图。有人喜欢先画出 A0 图，再抽取其边界箭头得到 A－0 图。事实上，可能发现有必要在 A－0 及 A0 图间反复几次，才能得到好的结果。

在很多情况下，会发现 A－0 图只是一个局部，而希望从更大范围来考虑全局性问题，则可以将 A－0 的盒子看作是上层父图的一个模块，画出更大范围内各模块的相互关系，称之为 A－1 图。还可逐级上推至 A－2，A－3，A－4。

3　画出顶层图

把 A－0 图分解成 3 个～6 个主要部分，得到 A0 图。A0 图表示了 A－0 图同样的信息范围。

A0 图是模型真正的顶层图。它第一个也是最重要的一个，从结构上反映了模型的观点。A0 图的结构，清楚地表示了 A－0 盒子的名字所要说明的含义。比 A0 图更低级的图形，说明了 A0 中各个盒子所要说明的内容。同样情况对更低级图形也成立。

4　建立一系列图形

为了形成图形结构，我们把 A0 图中每个盒子处理得像 A－0 盒子一样。即把它们分解成几个主要部分来形成一张新图。新图具有父盒子同样的内容，只是更为详细。

根据建模者的经验，或来自文件内容，或者通过对专家的访问，进行模型分解。在分解时，要列出所有的数据项及活动表，由此建立图形的初稿。要注意，这些数据项及活动表，包括了父盒子的全部内容，在分解中要根据此表画出盒子，在盒子间画出接口箭头，而不要丢失任何东西。

分解的次序可采用以下原则：

(1) 保持在同一水平上进行分解——均匀的模型深度。

在建立和分解 A0 时，人们先工作在 A1，A2，A3 等图形上，而不是在 A1，A11，A111。这样可避免较高层次的变化影响较低层次，造成可能的重复工作。同时可较早查出错误及遗漏。

本规则并非是一条严格的规则。有时为了更好地理解一个图，常常要了解较深的几层(如 A11 和 A111)。重要的是，要把这种特殊处理作为草图，直到水平层次都已确定为止。如果 A111 与 A2，A3 有矛盾，则要随时准备重画 A111 的材料。但是一旦当你有了新思想，知道该怎样深入进行时，就不必推迟画 A111 的草图，而应马上画下它来。

(2) 按困难程度进行选择

已知一个合适的模型深度，则存在一个问题，下一步应选择哪一个盒子去分解？这里可考虑以下 2 个因素：

① 从最困难部分开始。即从最不熟悉及最不清楚的部分开始。

② 选择某一盒子分解。该盒子的分解将产生最多的关于其他盒子的信息。

以上两条原则的实质，是让较简单或较容易理解的盒子较晚分解。较简单的问题，可以在较晚时候较容易地处理；进而错误及遗漏就要少一些，也就更容易地与较复杂问题的分解状况相匹配。

总之，在建立图形时要反复仔细研究，要通过较详细图形来深入考虑某些要搞清问题的重要部分。此时常常画几个(3 个～4 个)图形作为一组，而不是一次画一个图形。

5　写文字说明

最后每张图将附有 1 页(特殊情况可增加)的叙述性文字说明。文字说明分成两半，左边一列为“说明”(text)，右边一列为“词汇表”(glossary)。

A－0 说明了系统的内外关系，讨论范围。文字则补足了目的及观点。其他图(包括 A0 图)的文字说明是各不相同的，它是一种简短而精确的叙述，补充了图中不能表达的重要信息，但词汇表则要对图中所用的名字做出解释。只要不是通用的众所周知的名词，又是第一次出现，就要给以明确定义，作为继续向下分解时，共同理解的依据。

3.2　作者的工作阶段

1　数据收集阶段

(1) 阅读背景材料：作者通过阅读原始材料来收集问题有关的材料、信息。此阶段不

与其他人员往来。

(2) 访问:作者主动访问与所讨论问题有关的其他人士(专家)。

(3) 思考:作者不收集材料,也不画图,也不写文字材料。而是进行思考,把各种信息材料归纳起来。思考时间是在进行实际作图之前的一个准备阶段。

(4) 挑选盒子:这是一个重要的阶段,作者在各种设想中徘徊。此时的目的是选择下一步深入的题材。

2 构造阶段

(1) 作图:这一步包含了产生一个图形的实际创造性过程,不仅要画出盒子及箭头,还应包括在画盒子和箭头之前的工作,如列出数据元素及画草图等工作。

(2) 重画:这一步的实质是对图形进行消化整理,相当于重新编排一下,还要重写文字说明。这些活动不是创造性的,而是为使图形清晰所需的工作。

3 表达阶段

(1) 编写词汇表(glossary):对每张图中第一次出现的(即上层图或其他模型中未用过的)盒子名字或 ICOM 名词,如果含义不很确切的,需列在词汇表中作扼要的定义或解释。

(2) 写文字说明:这一阶段要写出适当的结构化的文字说明。有时,一开始分析员写出长长的说明来描述下一级方盒所说明的内容。这不是写说明而是描绘了非图形形式的图形。我们要求写文字说明,是要写出精确的结构化的文本,以作最终出版之用。

(3) 编辑:同样,这阶段是对结构化文字说明进行编排,相当于重画图形记号。

(4) 制作组表:制作图形的副本,画结点图。装订读者组表。

(5) 审查:审查及修订模型出版工作。这一步由作者之外的其他人执行。

4 交互阶段

(1) 读组表:通常,这一步是用于阅读由其他作者准备的读者组表。但有时,作者在长时间工作时,可中断一下,读读自己的模型。

(2) 反应:当作者接到读者评审意见时,读评审意见,对读者意见加以注释。

(3) 谈话:读者与作者坐在一起讨论对读者评审意见的看法时,称为谈话。与"访问"及"小组会"是不同的。

(4) 小组会:当作者发现单独工作不能得到进展时,可以用小组会来检查工作的进展及讨论下一步工作的办法。通常小组会的效率不如一两个人最多三个人进行的"访问"、"画图"及"谈话"的效率高。这种方法一般用来推动工作的进展及审查其他方面工作的结果。

3.3 怎样画一张活动图

在建模过程中,建立活动图是最主要及最有创造性的一步。不同作者有不同方法,下面建议的一系列步骤已证明对大多数作者是有效的。

(1) 建立一张尽可能好但尚未结构化的数据表,或称数据池,列出在父图范围内首先想到的一些项目。有可能时,将其归类。

(2) 对作用在数据上的活动进行命名，用盒子将活动名包围起来。

(3) 布置合适的箭头。

(4) 画出草图。尽可能使盒子及箭头排列清楚，如果结构太细，则把箭头组合在一起。保留最主要的元素。并在必要时进行修改。

(5) 必要时写出文字说明或 FEO 图以说明重要部分。检查图形目的、观点及平衡性、精确性。必要时，提出对父图的修改。

下面将着重讨论以上步骤。

1 建立活动盒子

当我们写出了数据表后，就可以设计“活动”盒子。先写出盒子的名字来代表所研究的问题。用一个盒子围住名字作为活动图形的第一步。这时还没有箭头连接各盒子。第二步是画出一些主要箭头进入或离开盒子，使盒子意义更明确。

要少使用专门的术语或缩写词。特别对 A－0 图及 A0 图，它们经常要提供给非技术人员参阅，所以所有名字更应一目了然。所定义的术语及缩写词要在词汇表中给出说明。

活动的名字要用动词短语。

建立活动盒子时要注意以下几点：

(1) 多数情况下把盒子安排成从左上角到右下角的对角线排列。当然，能使作者意图得到明确表达的其他形式也可采用。但列成水平或垂直一列时，常使箭头挤在一起，不利于表达良好的结构化风格。

(2) 通常，放在左上方的盒子通过控制箭头支配放在右下方的盒子。这样，便于读者理解其作用。但少量其他方式的安排，也是允许的。

(3) 在盒子的右下角有编码，如图 1.3.1。一般在图上从左到右，从上到下进行编码。这个号码确定了每个盒子的结点号。完整结点号的头几个数字与图形的结点号相同，最后一个数码即盒子号。例如，A2 图的第二个盒子，其结点号就是 A22。

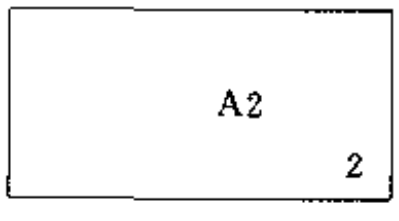


图 1.3.1 盒子的编码

(4) 每张图最多 6 个盒子。

(5) 当一张图只有 2 个盒子时，分解的作用不明显。可以试着再考虑一下是否分解合理。IDEF0 一般希望至少是 3 个盒子。

2 建立接口箭头

首先在盒子上安排互不联结的数据接口箭头。再把箭头的端点连起来表示由某个盒子提供输出给那个盒子的输入或控制。此时不考虑画面的清晰，只是把盒子连起来就行。

安排箭头的基本原则是：箭头表示约束条件而不是执行的顺序。

输出箭头表示活动可能出现的结果。输出箭头的语义，并不表明在那种条件下一定出现哪种结果，而只是指出一种可能性。如果对顺序有特殊重要意义，则画一个 FEO 图解释这个问题。

尽可能把一些箭头组合在一起，成为较少较概括的箭头。在建立箭头时，把箭头结构搞得过细或把箭头的标签写得太详细，是一种最常见的错误。要认识到箭头的详细程度必须与盒子的详细程度相吻合。在高层图形中，盒子名与箭头的标签都必然是较抽象的。

最后把箭头与数据表对照一下，保证正确的元素都在箭头中表现出来了。而不正确的元素确实是不正确的，而不是被忽视了。

3　尽量做好，但不追求完美

在画图时不必追求一个人能做到的最好表达形式。一开始的目标是努力做好，完成图形，满足语法规则。当图形完成后，由其他人提出评注意见来改进初稿。任何图形的第二版本一定能修改得比初稿更好。

训练有素的小组工作方式，总是能发现缺点、提出改进以提高质量的。

我们要求作者做好工作，但不追求最好的，而是通过评审周期，使进展在纸面上呈现出来，通过一系列改进，所有重要问题都可在纸面上把握住了。

4　重画

(1) 修改盒子、分裂与合并

在画一张图时，首先要注意把活动分成 3 个～6 个复杂度、详细度相似的盒子。如果不满意，则把这些盒子分裂，再合并，直至盒子间具有相似的详细程度。盒子间箭头尽量少，关系最简单，边界更为容易理解。

① 合并：把 2 个或多个盒子并成单一盒子，其目的是把有关活动集中成一个单一的更一般的活动。合并消除了应出现在子图上的过早细节，但也模糊了本级原先传递的信息。合并时用一个圈圈起应合并的 2 个或多个盒子，在圈的直径处写上新组合的盒子的名字。

② 分裂：把单一盒子分成 2 个或多个部分，是合并的反面。其目的是提供细节。一般来说，当目前的图形没有足够的信息来说明被分解的问题，或一个子活动不正确地包含在一个盒子内部时，可采用分裂的方法将其分解成 2 个或多个盒子。分裂的表示方法是将老的盒子用“×”叉掉，把新的盒子写在其旁边。

通常是分裂与合并在一起发生。盒子分裂后把产生的几部分重新组合成另几个部分，形成新的盒子，更贴切地表达所要传递的意图。虽然反映的问题是同一的，但因为是重新组合，所以应该更容易理解。

(2) 对父图进行修改

在画一张图时，了解了一些详细情况后，经常会发现要对父图进行修改，来改正父图中的错误及遗漏。我们有这样一条有关箭头结构的规则：如果在作图时，对是否需要某数据箭头有怀疑，则不画该箭头——等以后详细图决定是否真正需要该箭头。因此当细节明确后，修改父图是经常的。

父图修改的困难程度是各不相同的。最简单的情况是只调整修改直接的父图。有时要上溯好几张图，则情况要复杂多了。一般情况下，提出修改父图，往往不只是修改一张，而是有关系的好几张图。修改后要仔细检查 ICOM 码是否正确。如果有另一位作者也在分解同一张图的其他部分，则应把改变通知他。

(3) 连接图形的 ICOM 码的语法

理解图形的重要一步是弄清楚从父图向子图分解的细节。首先需要用结点号来反映盒子在结构中的位置。其次就是用 ICOM 码标注边界箭头，来提供严格的接口连接。在画子图中每个边界箭头时，用 ICOM 码来说明其在父盒子上箭头的位置。子图中写 ICOM

码时，要靠近边界部分，而不是靠近盒子部分。

在画一张新图时要满足以下要求：

① 每张图的目的观点要与 A－0 图的目的观点前后一致。

② 它的边界箭头与父图箭头一致。

③ 它的内容必须是父盒子所包含的一切。

对①及③要运用你的判断力，而对②可用 ICOM 码来跟踪箭头关系。在分裂与合并时更要用 ICOM 码来管理其变化，使之在模型其他部分不发生错误。

5　图形设计的注意事项

当数据池及活动盒子都已画好，基本箭头也画上了，则原来的图形可能显得混乱。就要重新整理图形并写上注释。首先安排盒子呈对角线排列，按其约束关系从左到右、从上到下。此时应对盒子进行编码。

整理图形首先从最经常使用的约束箭头安排起，把不太使用的箭头留待以后处理，这样来确定盒子的位置，然后画出父图上全部边界箭头，最后画其余的箭头。

本节要总结一下使用盒子和箭头的一些注意事项，这在起初画出粗略的图形，解决了图形应当包含什么内容后，再整理一个图形时，是很有用的。

（1）控制

① 当一个箭头可同时用作控制及输入功能时，把它表示为控制。当怀疑是输入还是控制时，使之成为控制。一个箭头在父图是控制箭头时，在子图中可能是控制，也可能是输入，也可能是二者，这取决于在子图一级的功能是什么。

② 活动盒子至少要有控制箭头，但可能没有输入。

③ 一个盒子既没有输出，也没有双向输入及控制的，则解释为此盒子是“只写”。不应该出现这类盒子。

④ 通常不要把一个箭头分裂成对同一盒子的输入及控制箭头，如图 1.3.2 所示。最好把这种分裂细节表示在下一层详细图中，以表明分裂成这 2 种箭头的原因。若实在必须这样做时，要分别写出所分出 2 个箭头的标签，以表示出这种区别的重要原因。

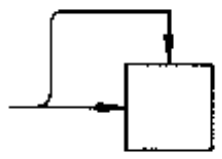


图 1.3.2　不正确的箭头分裂

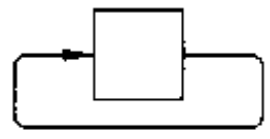


图 1.3.3　活动图形的循环处理

⑤ 可以用图 1.3.3 表示对一个活动图形的循环处理。

用图 1.3.4 表示在活动图上的数据存储。此时强调认为是在本级进行存储，且不再作进一步分解，否则就应将反馈线放在下一级详细图上，表示在盒子内部进行作用。

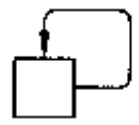


图 1.3.4　活动图上的数据存储

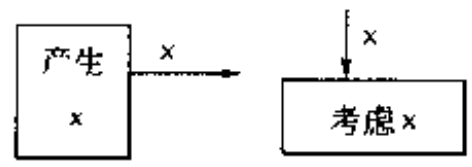


图 1.3.5　避免冗余性

⑥ 要避免冗余性，如图 1.3.5 所示。

这 2 种情况下，盒子名只反映了箭头所传递的信息，因此是重复的。只要多考虑一下就可以找出更有意义的盒子名字了。

(2) 箭头

① 画箭头时要作水平或垂直线，不要画对角线，也不要画曲线(除了拐角外)。

② 离盒子一段距离处放箭头的交叉处、拐角或标记。不要太拥挤。

③ 非绝对必要时，不要用“数据”“活动”“输入”“输出”“控制”及“机制”来命名盒子及标记箭头。

④ 如果箭头太长，标记它 2 次，如图 1.3.6 所示。

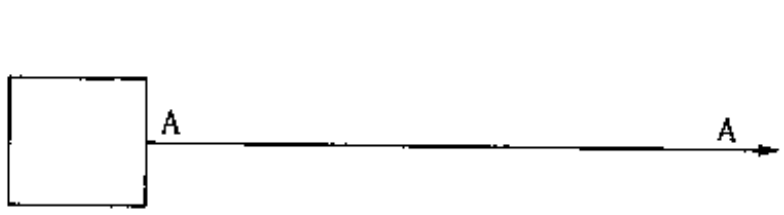


图 1.3.6　长箭头的标记

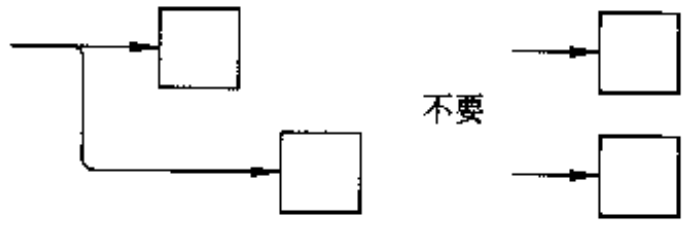


图 1.3.7　开的边界箭头画法

⑤ 把 ICOM 码放在箭头的非连接端。

⑥ 若箭头是开的边界箭头，把所有进入有影响盒子的箭头都连在一起，如图 1.3.7 所示，否则读者可能会发生遗漏。

⑦ 不要把箭头一直画到纸的边框处。如图 1.3.8 所示。

图 1.3.8　箭头与边框保持一定距离

⑧ 平行的箭头要安排得疏松些。又长、靠得又近的直线很难看得清楚。见图 1.3.9 所示。

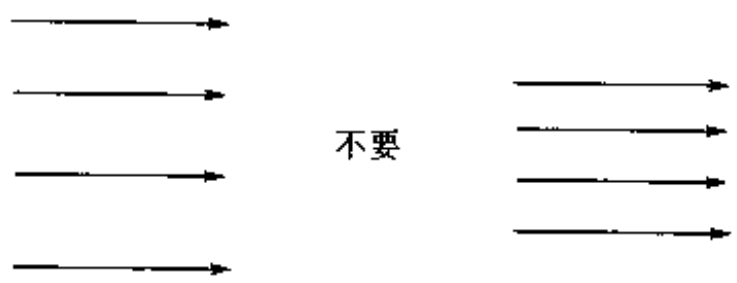


图 1.3.9　平行箭头的安排

(3) 安排

① 除特殊原因外，一般把有同样源及同样目的地的一组箭头，用 1 个箭头来代表，如图 1.3.10 所示。

② 盒子的一边一般不要超过 4 个箭头。如果有多于 4 个箭头则用图 1.3.11 方式表示。

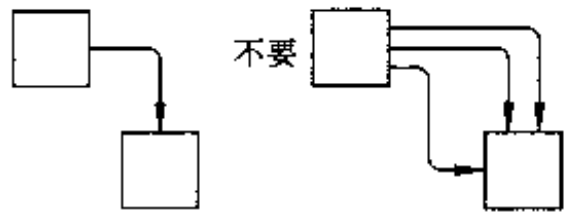


图 1.3.10　同样源及同样目的地的箭头应合并

图 1.3.11　一边多于 4 个箭头的画法

③ 控制反馈先上后下较为清楚，如图 1.3.12 所示。输入反馈先下后上较为清楚，如图 1.3.13 所示。

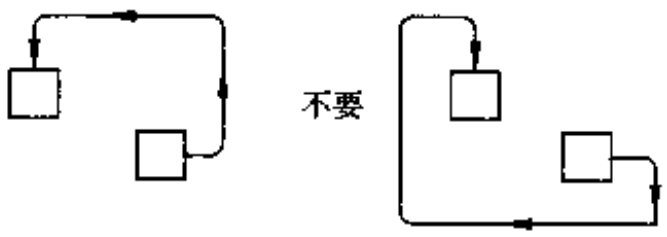


图 1.3.12　控制反馈先上后下

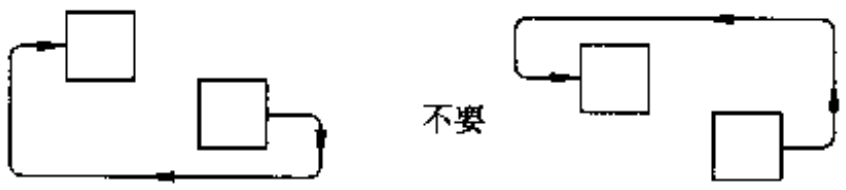


图 1.3.13　输入反馈先下后上

④ 如果箭头分支及送入几个盒子，只要可能，按相同的 ICOM 码关系来做图。如图 1.3.14 所示。

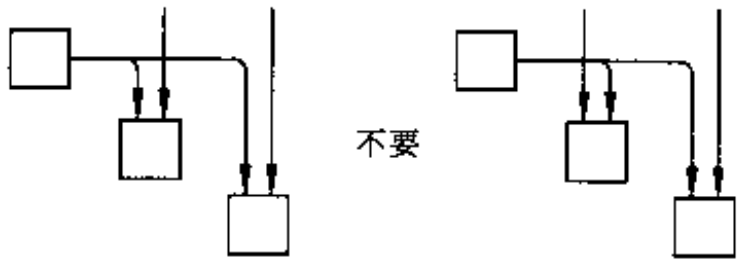


图 1.3.14　ICOM 码尽可能相同

⑤ 使箭头安排得尽可能减少交叉，如图 1.3.15 所示。

⑥ 尽可能减少曲线及拐角，如图 1.3.16 所示。

图 1.3.15　箭头安排尽可能减少交叉

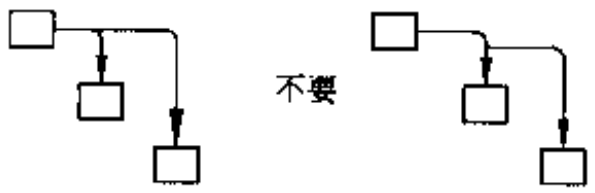


图 1.3.16　尽可能减少曲线及拐角

⑦ 为了表达清楚要用以下方式，如图 1.3.17 所示。

⑧ 分流及汇集的简化表示，如图 1.3.18 所示。

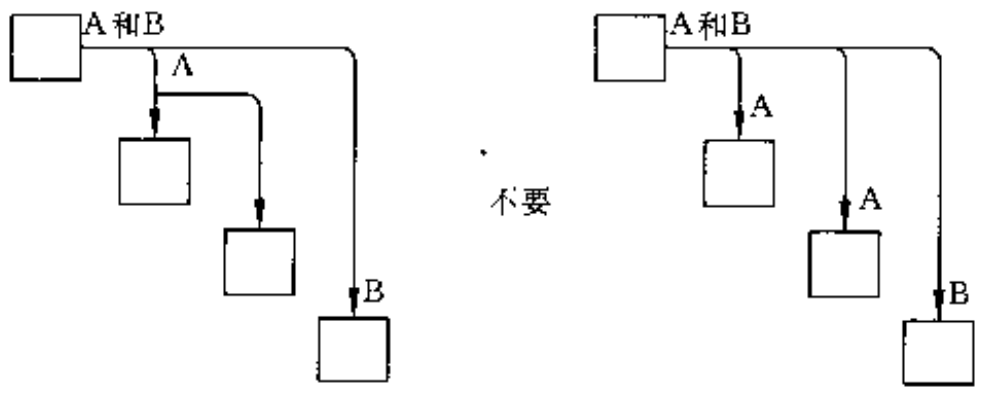


图 1.3.17　一种表达清楚的方法

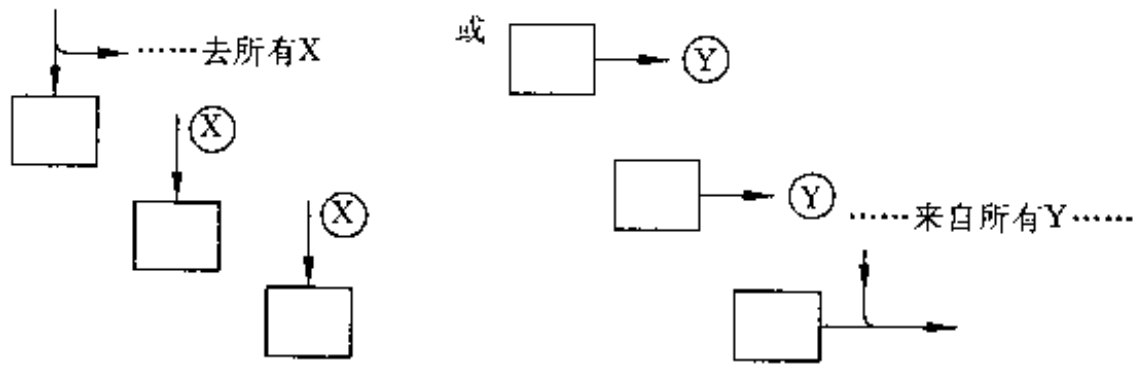


图 1.3.18　分流及汇集的简化表示

3.4　怎样写文字说明

1　图形的文字说明

每个图形附有一简短的文字说明，来补充说明对图形的总的看法，通常长度不到一页。文字说明突出了作者感到特别有意义的方面，而不是重复图形本身已经表达的细节。文字说明帮助读者理解图形的主要思想，有时图形已表达得足够清楚时，可以不要文字说明。文字说明是图形的补充，但 IDEF0 建议主要是用图形语言，所以不一定在一开始就在说明上花很多功夫。

由于早期不依赖文字说明，所以要求图形本身能说明要传递的思想。而文字说明用简短的记叙文式书写，所以也是结构化的。

写 A--0 图的文字说明时，要充分考虑它是一切分解的源头，所以应说明最显著最本质的东西。A—0 图的文字说明要精确陈述模型的目的与观点。而且与其他图形的文字说明不同，A—0 的文字说明在 A—0 图一画出时，就写出，而且随时与 A—0 图附在一起送给读者评注。

2　引用与注释

在写文字说明及注释时可以使用以下简明的引用语言。

例如：

3	意味着	盒子 3
O2	意味着	具有 ICOM 码 O2 的边界箭头
2I1	意味着	盒子 2 的输入 1

2O2to3C1　　意味着　　从 2O2 到 3C1 的箭头

n　　意味着　　注释 n

以上引用表示在当前图上的符号，否则前面应放上结点号及必要时的模型名，“·”表示看指定图上某一东西，例如：

MODEL/A21·3C2　　意味着　　在模型 MODEL 的图 A21 上看盒子 3 的控制 2

D42·③　　意味着　　在本模型图 D42 上看注释 3

在处理读者资料袋时，由资料员放入的活页字母可以代替模型名及结点号，例如：

E·3I1　　意味着　　在本资料页 E 页上看盒子 3 输入 1

每个引用应足以不引起二义性，完整形式如下：

ACCT/USER/A21(T56)·1O2to4C3

意味着在模型 ACCT，USER 版本，图形 A21，版本 T56，看从盒子 1 输出 2 到盒子 4 控制 3 的箭头。

注意：“·”的位置指出要看那个图形。例如：表达式 A2·3O2 意味着在图形 A2 看 3O2(盒子 3 的第二个输出)，表达式 A23·O2 意味着在图 A23 上看 O2(具有 ICOM 码 O2 的边界箭头)。

第 4 章　项目参加人员的任务和有关图表的说明

对于一个大项目而言，系统分析和设计绝不会是一个人来单独完成的；相反，一定是由众多参加人员协同工作来完成的。在这种协作中，每一个参加人员都在项目中承担一项或多项职责（任务）。相当于 IDEF 方法中，我们将它定义为单一“角色”或多个“角色”。IDEF 方法允许任何参加人员，不分职称高低地担任适合自己能力的单一“角色”或多个“角色”。对于作者新创作的所有组表而言，作者本身不能充当评审员或读者。当然，此作者充当其他作者的组表评审员或读者是无可非议的。按 IDEF 方法，一个项目中，为确保系统分析和设计的基本要求，一定有作者、评审员、读者、文档管理员、技术委员会和项目负责人 6 类“角色”，每一类都可以是一个或多或少的参加人员的组合，只是项目负责人组合中一定有一个全面负责，其余各类角色中，任何参加人员具有相同的职能和权力。当然，工作的对象是完全可以不同的。这 6 类角色在一个项目中，如何协调一致地工作？如何交流设计思想？如何决策设计方案？各角色的详细定义是什么？介绍如下。

4.1　项目中各“角色”的定义

1　作者

作者是在 IDEF0 方法上训练有素的系统分析和设计人员，作者除对 IDEF0 方法颇有了解外，他还是所要分析和设计的对象系统某一方面的专家。作者的主要任务，就是用所掌握的 IDEF0 方法，对其所要分析的系统，进行系统功能分解和设计。在分解中，不断创作出层次结构化的 IDEF0 图表。这些图表完全体现了作者的系统分析和设计思想，通过这些图表，作者可以与读者、评审员及其他参加人员沟通思想、交换意见，最终确定系统设计方案。有关 IDEF0 的标准图表定义，后面还有详述。

2　评审员

评审员通常是由不同的作者担任的，也就是由本图表创作者之外的作者担任的。他们也可以是职位稍高一点的专家、学者、项目负责人或技术委员会成员，来充当这一角色。承担这一角色的人员，同样应具备作者所具备的 IDEF0 方法知识和被分解系统的专业知识。他们的主要任务是，对作者的设计思想和分析结论进行书面评审。最主要的任务包括：确定模型结构和内容上的一致性和完备性；确保模型的质量要求；确保所建模型满足项目的目标；恰当地完成 IDEF0 图的项目审批职责；试图发现建模中的问题并明确提出解决的方法。

3　读者

读者和评审员的素质要求基本相同，只是两者读图的目的不太一样，读者读图是为了

扩展视野，获取本项目中其他有联系的系统的分解信息，以期能对本职工作有某种借鉴意义。当然 IDEF 方法不压制读者的积极性，读者也可以提出一些口头意见，但不能象评审员那样进行书面评阅。

4　文档管理员

他是一般技术人员就能承担的角色，他不要求很多 IDEF 方法的知识，更不要求某一专业知识，只要求有一定的技术管理能力。他的主要任务就是：确保作者－读者（评审员）循环的顺利进行；确保项目中其他各类角色的事务工作开销最省；定期更新文档资料；编写工作日记；保管除作者的 IDEF0 初稿外，管理项目中的其他所有文件；发现并解决作者－读者（评审员）循环中的“拥塞现象”；及时准备各种评审会和技术交流会等。

5　技术委员会

技术委员会是一个由有关专家、学者、项目高层决策人员（包括项目负责人）和一些有经验的作者、读者、评审员所组成的一个实体。其主要任务是评判作者与评审员之间的不同意见；制定项目总体规范和策略；召集各类技术问题的评议会；是项目负责人的技术“参谋部”。

6　项目负责人

他是项目的总负责，他对全局问题负有决策、指挥权，是所有高层技术问题的最上层决策人。

4.2　作者－读者（评审员）循环

作者－读者/评审员循环，包括两大分支。一是作者－读者循环，是一个开环循环。若读者有兴趣对作者的工作提出意见时，这又是一个闭环循环。一般不要求读者以书面形式提出自己的意见，这一循环的目的是，帮助与作者有关联的读者了解作者的工作、进度，并且从作者的工作中，及时获取对读者必要的或是有益的系统信息，从而保证系统分析设计工作的协调一致。二是，作者－评审员循环。这一循环的目的是得出一个正确而可靠的系统分析设计结论。这一循环包含了 4 类角色：作者、评审员、文档管理员、技术委员会，有时还要项目负责人参与。作者－评审员循环的大体内容如图 1.4.1 所示。

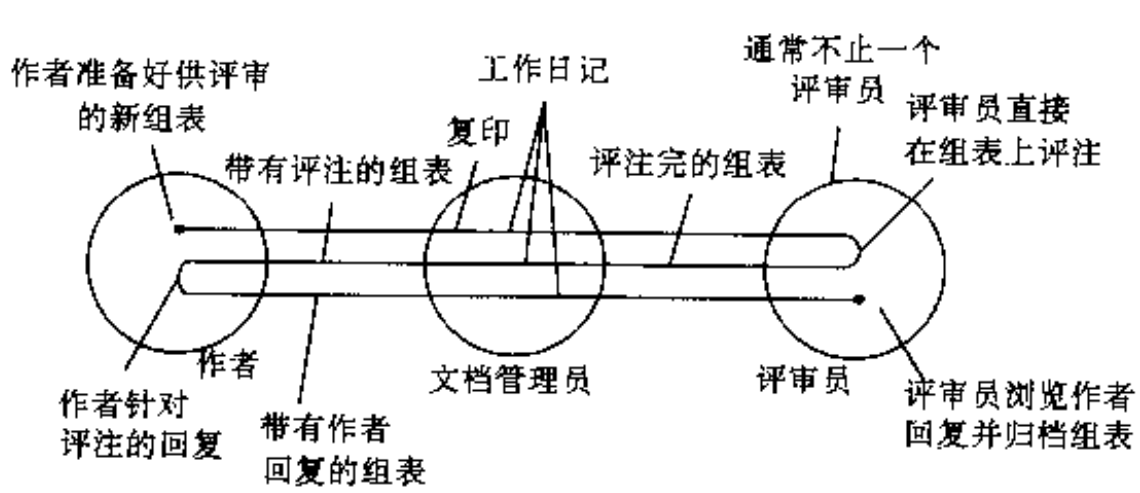


图 1.4.1　作者－评审员循环的示意图

不难看出循环的步骤大体分为如下 4 步：

1　形成评审组表：作者将要送审的一组 IDEF0 图表归并在一起，附加上封面页，形成一套评审组表。作者在封面页上写上评审员清单（这个清单中可以包含一个集合的人员统称，如技术委员会成员，这就是说，技术委员会的每一个成员都将是一个评审员）以及本套组表要求收回评审意见的时间要求。然后将这套组表送交文档管理员复印多份，分别分发给每个评审员和作者一份复印件，并在每一份复印件的封面页上说明准确的交回时间，原件存档备查。

2　评注意见：在规定的回收时间内，评审员读完这套组表并且直接在此份复印件上写上评注意见，每评注一条意见都要写上序号，并将图头的“注释”栏内相应的序号用圆圈圈上，然后将组表通过文档管理员回送给作者。

3　作者回复评注：收回带有评注意见的组表后，作者直接在这些组表的评注意见附近给出回复。作者若同意某个评审员的某条评注意见，则将图头注释栏内相应地被圈上的序号打“√”，然后收编此评注到这组图表的下一版本。若作者不同意某一评注意见，则将相应的被圈上的注释号打“×”并写明理由，然后将组表再次通过文档管理员送给评审员。

4　归档或开评议会：评审员认真阅读作者的回复后，若作者的回复能使评审员满意，他就归档这套组表，（评审员必须保存所有他评审过的任何材料）。若评审员坚持与作者的不同意见，他就安排一次与作者的面谈以解决分歧。若面谈仍没有统一意见，那只好由下一次技术委员会的评审会来最终解决。

此循环的结果是使作者的初图升级为修正图级，通过文档管理员将这些组表及评注过程和结论归档。然后此套组表就准备提交技术委员会的评审会议，讨论是否将此组表升级到建议图级，关于本节的一些文档图表的定义请见下节详述。

4.3　IDEF0 图表的定义

1　IDEF0 标准（规范）图表的定义（图示见附录 1.1）

标准图表主要由三大部分组成：图表头部分（图表体上部）、图表信息部分（图表体中部）和标识部分（图表体底部）。上部是记录有关工作过程信息的表。在最终出版时，可以裁去此部分内容。中部是 IDEF0 画图规范所定义的内容，这里不再重复，请见前几节。

下部标识和上部信息的填写规范定义如下：

“用于”栏：填写此图用在什么地方，或其用途。

“作者/日期”栏：填写最初创作此图表的作者姓名和第一稿的时间。

“审阅/日期”栏：填写最终负责审阅人姓名和交付出版的日期。

“项目”栏：填写此图表所属的项目名称。

“注释”栏：是提供评审员评注本图表内容时用的，评审员每评注一条意见，就将“注释”栏的 10 个整数中数值最小且没有被划过的序数画上圆圈，并且将此整数作为此条评注的序号，放在此条评注的开头，且用一个小圆圈上。作者对本图表内容上的注释不允许使用带圆圈的序号标注。

“图状态”栏：IDEF0 规定图的状态分为：初图、修正图、建议图和完成图（出版图）。

初图：就是作者准备送审的初稿，其他三个等级的图表都是由此等级修改演变而得

到的。

修正图：若一张(或一组)图表的所有评审员都评审通过了，这张图表就被升级到修正图级，修正图级图表供送交分系统主任设计师或技术委员会审议用。

建议图：修正图等级的图经由主任设计师(或技术委员会)评审通过后，升级为建议图，此图表送交项目负责人最后审阅后就成为完成图。

完成图：最后出版的，不再变动的图表。完成图是图表的最完善状态。

“图状态”栏前的小方格就是用做图的等级标志的，用“×”代表此图已达到相应的等级，用“/”代表此图表正在相应等级的荐升之中。

“读者”、“日期”、“附页”栏：此栏供不同等级的评审员签名、注日期和所附意见页数之用。按我国实施 CIMS 实践中的做法，提出下面三个签名，供参考：第一位是同组技术人员作为评审员，第二位是 CIMS 总体设计组中负责 IDEF0 建模规范的技术人员，第三位是分系统主任设计师。

“父图”栏：此栏用来示意此图与上一层父图的关系。

填写方式由图 1.4.2 所示。此栏填写的是一个父图的简化原型，父图中，被本图所分解的盒子保留用方框表示(以志区别)，其余盒子全部用随手画的圆或椭圆来表示。(也可以都画小方框，但把所分解的盒子涂黑。)图号和序号请见下文。

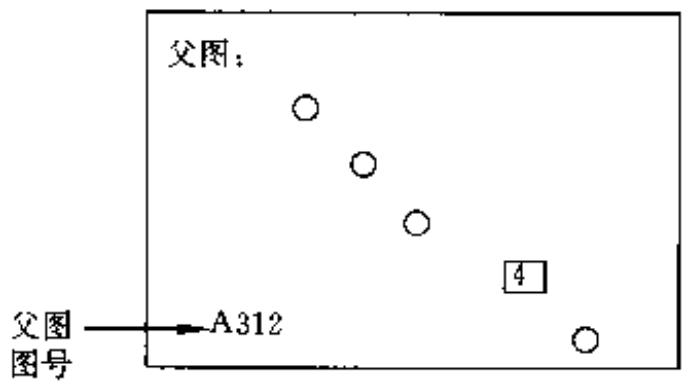


图 1.4.2 “父图”栏的填写

“图表信息部分”：此部分的规范同前几章节。

“图号”栏：按 IDEF0 规范分配的本图的节点号。关于如何分配节点号请见前文所述。

“编号”栏：此栏主要有三方面的编号，见图 1.4.3 所示。

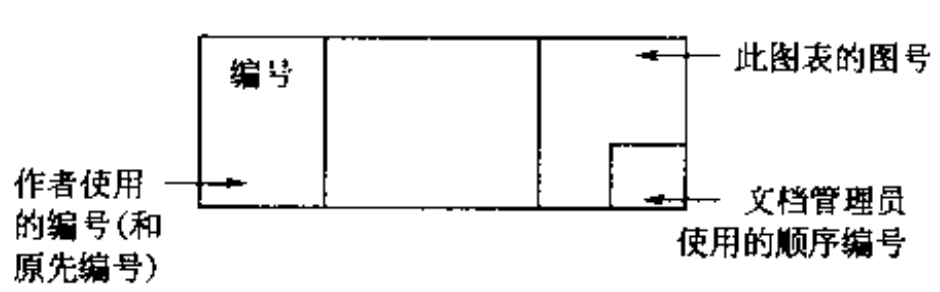


图 1.4.3 “编号”栏的填写

作者使用的编号可以是作者的姓名头一个字母缩写加上序号，若一张图修改过版本，可以在当前编号的后面加圆括号填入先前版本的编号，以表现编号的历史进程。图 1.4.4 是一个例子。这里就表示编号为 PSP66 的图表是由旧版本编号为 PSP62 的图演变而来(其中 PSP 是作者姓名的缩写)。此处的图号与前面所提图号定义一样，重复出现是为了

查阅方便。另外，结合我国情况，一般不强调作者名字，而强调所属的分系统，则这个编号就可由三部分组成：先是分系统的代号，然后是图纸顺序流水号，第三部分则表示在什么阶段的第几张图纸（如初图 W，修正图 D，建议图 R，完成图 P），例如写成 MIS004-W1。文档管理员编号是由代表此张图表所在文件的代号加序号组成，如此例，此图（PSP66）是 K 文件中的第五张图表。

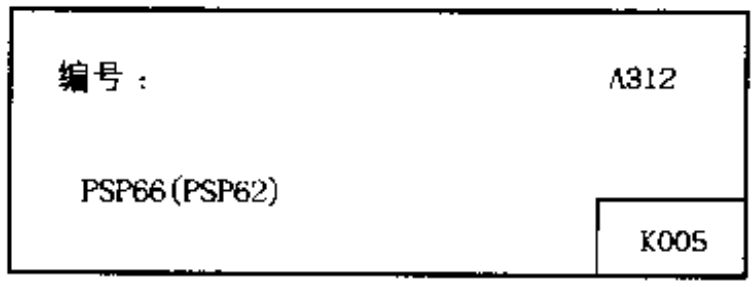


图 1.4.4 “编号”栏举例

2 评审组表封面页的说明（图见附录 1.2）

此封面页的头部和底部栏目与前节的标准图表填写方法一样，不再重复。中间部分的栏目填写方法如下：

“评审员”栏：列出所有评审此组表的评审员姓名，这里也可填“技术委员会成员”等集体名，文档管理员会自动给技术委员会每一个成员复印一份副件并发送出去。

“新组表送交评审员日期”栏：填写新组表要求送交评审员的日期。

“应收回日期”栏：告诉评审员应在什么时间之前评注完此组表，并及时送回给文档管理员。

“评注送交作者日期”栏：指的是某份评注过的组表应在此时间（日期）前送交回作者。

“返回日期”栏：指的是作者给评注意见回复的最后期限。

“对评注回复送交评审员日期”栏：填写作者对评注做出答复后，应将此套组表送给评审员的最后期限。作者、评审员任一方延误时间时，文档管理员应及时提醒他们。

“回复要求”栏：有四种选择，分别是：①尽快②一般③慢④随意，若作者要求此组表尽快返回，他在交文档管理员前就在此栏的“尽快”前的方格内填入“√”。其他情况类推。当然同时在两个以上的小方格内填入“√”标志，是会让文档管理员糊涂的。若一个“√”标志也没有填，那么文档管理员可认为是第 4 种情况，即“随意”。

“组表内容”栏：要求填入组表中的所有图表内容，以供查对。此栏中的“页号”、“图号”、“题目”、“编号”四个子栏的填法与前述相同，只是状态子栏有些不一样，此栏是用来标志某张图表所属关系和修订或替换的状态标志栏，此栏的定义如下：

☐ 空格表示此张图表无需放在模型文件① 中。换句话说就是，该张图表的变化与否并不影响该模型文件。

☑ 表示此页图表是某个模型文件中的一页。

☒ 此页图表是该模型文件中的一页，并且此页的变更已经收编到此模型文件的相应

① 模型文件——是以图号为序排列的全套项目模型图表的总和，组表的修改和替换应及时反映到模型文件中来。

页。原稿亦已返回到评审组表文件① 中。

“评议”栏:填入该组表要影响的模型文件名。如图 1.4.5,意即,此组表中状态为“☒”的所有图表,一旦在评审中有变更,一定要反应到总系统模型文件中来,这是一个保证信息统一性的必要手段。

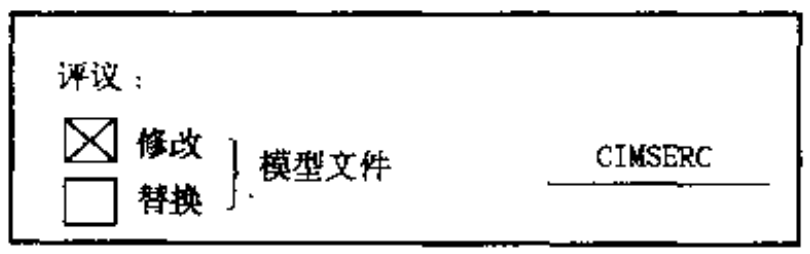


图 1.4.5 “评议”栏的填写

此栏下部空白,可以填写评审员的一般性或共性的评注意见,也可以填写评审员对作者的任何要求信息。此空白还可以用来填写“内容”栏的内容。当然是在“内容”栏填写不下所有组表内容时才这样使用。

“特殊指示”栏:若作者需要复印件,就在此行的小方格内填入“√”。若作者认为有必要多复印几份此组表备用,就在此栏第二行填上所要额外备份的份数。此栏余下空白可以非正式地填写作者的任何要求。

以上介绍了 IDEF0 方法中的主要图表填写与规范,不同的项目也许还要求不同的文件和图表,读者可在此思想的基础上加以发展。

① 组表文件——就是将为评审目的而合成的图表,在评审完后,按组表编号按倒序方式归档形成的全体组表的总和文件。

第 5 章　一些工作方法

IDEF0 的建模过程还有一些完整的工作方法和步骤，以确保不致因工作人员本身的经验不足而造成疏漏。这里主要谈一下最初的数据收集过程，以及画出 IDEF0 图后的遍历检查过程。

5.1　数 据 收 集

分析设计一个系统，当然首先要掌握有关该系统各个方面的事实和数据。这些信息来源可以是：①阅读已有文件、各种表格和所需信息的索引。②如为已有系统，则可观察系统的运行操作。③通过提问或其他方式，向较多的人进行广泛调查。④与具有这方面专门知识的一个或几个“专家”深谈。⑤运用作者本人的已有知识。⑥猜测或提出一些假设，请读者协助使之逐渐接近现实。而其中最重要的是与专家的面对面的谈话，因为一般说来已有的信息不可能都是书面材料，只有在这种调查座谈中做好记录，甚至画出草图，才能逐步深入地掌握建模所需的各种信息。

1　谈话类型

专家谈话又可以根据目的和对象的不同分为四类：①寻找事实，以便理解当前的操作和现有的环境。②明确问题，有助于建立未来的需求。③求解讨论，涉及到未来系统的能力。④IDEF 作者与评审员对话，用于解决建模中的问题。

2　谈话组文件内容

(1) 封面页。

(2) 谈话登记表。

① 谈话人姓名(IDEF 作者名)，

② 谈话日期(IDEF 绘图日期)，

③ 谈话时间(开始时间与结束时间)，

④ 答话人姓名，

⑤ 答话人职称和职务，

⑥ 答话人地址与电话，

⑦ 发现的其他信息源(文件名及其出处，其他可进行谈话的人名、职务、地址及电话)，

⑧ 信息的基本元素 —谈话中关键内容的小结，

⑨ 进一步的问题以及谈话中未涉及或被推迟了的一些方面，

⑩ 在词汇表中应新增加的术语。

(3)“活动”和数据表。

(4) 谈话日程。

(5) 谈话笔记和草图。

3 谈话指南

为了成功地进行谈话，获取尽可能多的信息，把谈话分成准备、开场白、对话、结束和总结五个阶段，每个阶段都要有很好的气氛和对答话人造成心理上的信任感。

(1) 准备

事先想得越细越深，谈话效果必然越好。大体上有下面几步准备工作：

① 选好对话人(根据负责的领域，别人的推荐，或者不同层次的人，上层人物谈大局概貌，下层的人谈信息的细节，中层的人则弥合了这两者的间隙)；

② 约会(时间不要太长，一般为半小时到一小时，不要在临近午饭或下午很晚的时间，要讲清谈话的目的和要求)；

③ 写出初步议程(谈话议题范围，避免泛泛而谈，提出专门问题)；

④ 回顾一下可用的背景信息；

⑤ 检查一下相应的术语；

⑥ 保证与别的谈话人协调(答话人是否已与别人谈过，如果是一次补充谈话则要检查一下原谈话结果)；

⑦ 填写谈话登记表；

⑧ 正式做出谈话议程。

(2) 开场白

在简短的互相问候以后，可以做下列事情：

① 简单自我介绍，譬如给一张名片；

② 说明谈话的目的；

③ 征求对方同意你作谈话记录，可能答话人会要求你保密；

④ 确定专家/作者关系，解除答话人的顾虑；

⑤ 从一般性问题(当然不能离开议题)开始，让答话人便于提起话头；

⑥ 估计一下答话人能否提供中肯的信息，如果信息太一般或者太细，不适宜于IDEF0 建模阶段的要求，趁早结束免得浪费时间；

⑦ 开始形成专门的问题；

⑧ 记录，而不要自己说很多。

(3) 引导谈话

谈话过程中，要把握两个方面，一是信息的合格性(qualification)，另一是诱导信息流(通俗的说就是打开话匣子)。

人脑的综合速度比说话速度要快一倍，谈话人要不断在想对方应该说什么，而不单是听对方正在说什么，下面这些问题可作为参考：

- 所提出的事实是否支持了所讨论的主要观点？
- 这些信息是新近的吗？
- 这些信息完整吗？

- 我真的理解了所说的内容吗？
- 所提供的细节是否适合我的目的了？
- 有没有漏掉什么方面？
- 这些信息跟别人讨论过吗？
- 这些信息有多重要？
- 是否讨论枝节问题了？
- 谈话的观点变掉了吗？

另一方面还要诱导信息流，使答话人能提供最多最可靠的信息。可有下列做法：

- 尽量少发表评论和交谈，因为这次谈话是为了获取信息，而不是交朋友，或推销你的思想；
- 要意识到如果答话人弄不清楚要讨论的问题存在于哪些方面的话，可能是他与谈话人还比较拘束；
- 给答话者以思考时间，不要去建议这样那样的答复或提出另外的问题；谈话过程中的暂停有利于答话人回忆一些最关键的信息；
- 避免可能打断他思路的外界干扰，如果可能，谈话地点尽量避开答话人通常呆的地方；
- 要意识到内部分心的表现，答话人是否感到不舒服或者拘束；
- 尽可能确定一下所获得的信息是事实还是意见；
- 请求重述一遍或小结一下以便表达得更精确些；
- 弄清答话人的背景和与所讨论事物的联系，因为只有知道了他与组织或已有系统的关系才能理解他的评论的价值；
- 不要用讽刺或幽默；
- 不要提到或讨论与其他人的任何谈话；
- 记下答话人提出的所有问题，除了涉及用户组织的管理、计划或人身评论的事情，谈话人应回答所有问题；
- 对答话人所谈的内容要表示出感兴趣；
- 集中精力于所讨论问题的不熟悉的和困难的方面，避开那些显而易见的事；
- 警惕不确切或不正确的用词，对任何不熟悉或有疑问的名词要问清定义，为词汇表记下这一定义；
- 不要跟答话人当面顶牛，即使他说的不符合事实，也要在组文件循环中去解决这矛盾；
- 要谦虚，答话人是专家，而不是你谈话人；
- 推迟那些在商定的时间框框内来不及谈论的题目，与其拖长谈话时间不如另约一次；
- 要喜欢对同一问题的不同意见，然后用 IDEF 来表明这些意见并解决矛盾；
- 用恰当的不急于做出结论的问题来启发答话人的思路。

另外，为了更确切地记录答话人对 IDEF0 建模的意见，谈话记录不必局限于文字的记述，还可以是针对一张一张 IDEF0 草图，很具体地对活动名、活动间关系、各种 ICOM

的标注等，听取答话人的意见，并直接标注在图上，回来后对这些意见进行分析，决定取舍后，重画 IDEF0 图。

（4）结束谈话

在下面 4 种情况下，应结束谈话：①谈话中所获得的信息不合适，②时间已到，③谈话人觉得信息已经饱和了，④谈话人与答话人之间个性不合。

随着不同的结束的原因，还需做下面一些事：

- 不要突然中止，还应有几分钟非正式讨论后结束；
- 小结一下谈话的主要内容；
- 明确一下被推迟的或未谈到的共同关心的问题；
- 如果需要，安排一次补充谈话；
- 请答话人推荐一些可以谈话的其他人；
- 假如谈话笔记在散发前还要请答话人检查一下的话，应在结束前讲清楚；
- 要对答话人的帮助表示感谢。

（5）总结

这个阶段在于要保证谈话中所获得的信息被正确地记录下来，并通过谈话组文件散发给项目组成员。如果答话人不同意散发记录，则谈话人应在结束谈话前立即写下所讨论的要点。总结过程包括下列步骤：

① 确定附加的信息源；
② 小结信息的基本内容；
③ 确定词汇表中的新名词；
④ 列出补充问题和谈话中被推迟的或未及谈到的问题；
⑤ 完成“活动”和数据表；
⑥ 把回忆起来的任何信息补充到笔记中去；
⑦ 准备出反映所得信息的粗略 IDEF0 图；
⑧ 明确任何正在做出的假设或还不太清楚的名词；
⑨ 公布并散发谈话组文件；
⑩ 写下在谈话中提到的人的名字、专业领域、电话号码和地址。

5.2 IDEF0 模型的遍历步骤

在评审检查过程中，除了组文件循环以外，IDEF0 还有一套对模型的遍历（walk through）过程，帮助评审人一步一步有序地去找出图中潜在的问题，这个结构化的遍历步骤可以归纳为下面 6 步：

第 1 步：扫描全图

使读者得到一个此图上下文的总印象，读者会先检查父图，本图只是作为父图中的一个盒子，而读者现在就要检查作者是如何分解这个盒子的。

通过准则：

（1）分解对模型的目的是有用的，在其父图上下文内是完整的，所有再低一层的功能

或数据都可以清楚地归属到它的每个盒子中去。

(2) 从评审人的意见看,本图反映了基于建模目的的一种相应的观点。

(3) 从评审人的意见看,有足够的新信息被提供来理解父盒子,而又不是那么细化以致使本图显得复杂而难读。

除非问题太明显,在第 3 步前暂时不要作评论。然而,不要丢失第一印象。把它们记在黑板上或记事本上,留待以后解决。

第 2 步:查看父图

一旦读者理解了本图的分解,就应检查父图以保证兼容性。

通过准则:

(1) 读父图时看到分解过程包含了评审人预期的全部内容。

(2) 父图中这部分的分解都显示出来后,评审人认为这些细节也都是正确的,否则就标注上遗漏的细节。

在这一步也许有必要简略地回到父图去,把在这一分解中得到的认识加上一个新的ⓝ标注,或者补充某一个已有的标注。

第 3 步:连接父图和子图

本步要检查父图到子图的箭头交互连接。

通过准则:

(1) 没有遗漏或额外的接口箭头。

(2) 接口箭头都标上了正确的 ICOM 码。

(3) 子图箭头的标签与父图上相匹配的箭头相同或更详细,标签表达了正确而完整的箭头内容。

(4) 连接箭头的检查显示了父图没问题(多加一个接口箭头就可能造成对父图表达的信息的误解)。

对父盒子四边顺时针的检查,是一种有条理地检查 ICOM 码匹配关系的方法。

第 4 步:检查内部箭头模式

按结点号的顺序检查每个盒子,又按 ICOM 的次序检查每个箭头。在完成这一过程后,评审人就要根据自己熟悉的工作情况,探讨图上所表示的前因后果,以及检查其相互存在的关联。

通过准则:

(1) 本图看上去不是混乱的,箭头的交叉和弯曲减到了最少。

(2) 每个盒子的细化程度大体上是平衡的。然而,有时为了清楚起见,或者为避免出现只有 2 个盒子的一层(IDEF0 规定最少每张图 3 个盒子),这条准则可以有所妥协。

(3) 本图应与评审人的经验以及对该事物的理解相符合,反馈和出错条件应该要有说明。

第 5 步:阅读支持文档

这步检查了作者在内容说明、词汇表和 FEO 中要强调说明的一些看法。

通过准则:

(1) 内容说明肯定了从检查图形本身所获得的解释。

(2) 正常通路、反馈、出错处理和由内容说明提到的其他特征都能在图面上找到，或者在一张 FEO 图(参照图)中找到。

(3) 在第 1 步至第 4 步未包括的有意义的图形特征，可以在内容说明、词汇表或 FEO 中找到。

(4) 本图的参考资料已足够细致，以便将内容说明、词汇表或 FEO 与一张图的各专门部分相连接。

第 6 步：设置图形状态

(1) 按该图所处状态推荐。

(2) 修改后推荐。

(3) 重新画图：已作太多改变，需要重画，还要进一步评阅。

(4) 不接受：需要完全重新分析。

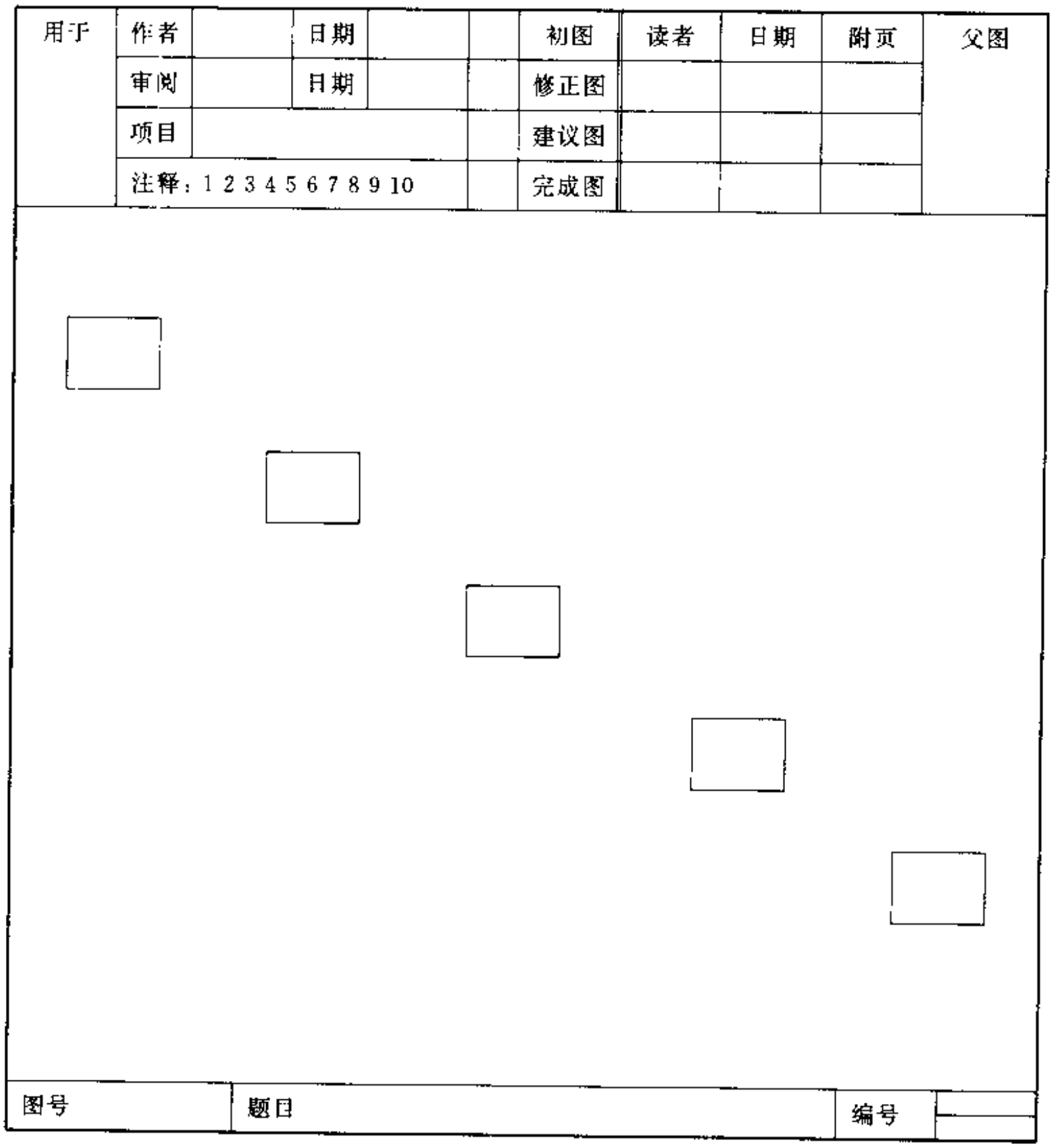
5.3 结 束 语

由于国内应用 IDEF0 方法的时间不长，经验不足，本规范主要是依据美国 ICAM 的 IDEF0 手册再结合我们的看法编写的，有些内容只能算作是“指导性”或者“参考性”的，欢迎使用本规范的单位或个人在实用中提出自己的看法，不断与我们交流，逐步形成一套适合中国特色的系统设计方法和规范。

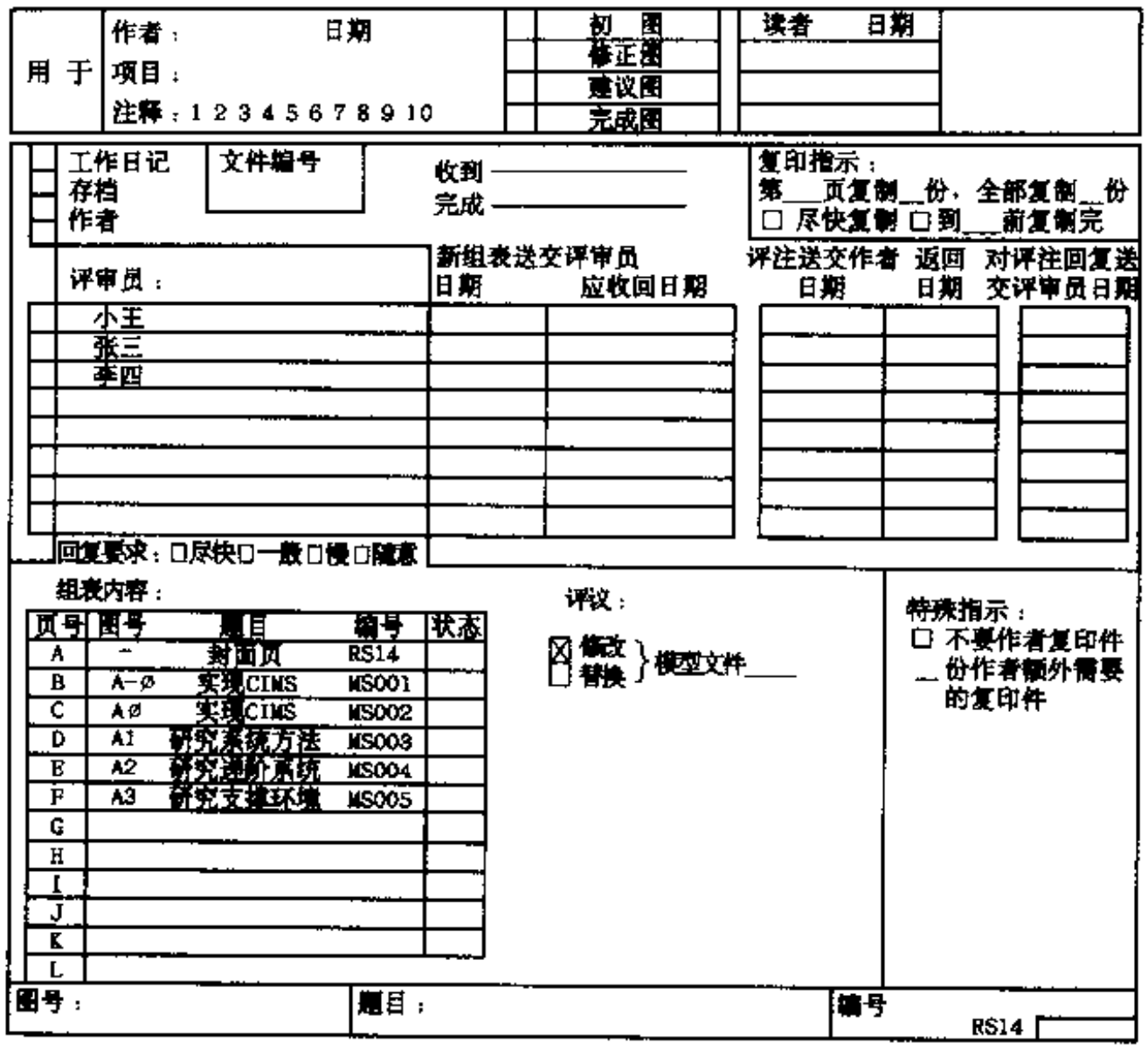
因为任何方法本身都要在实践中不断丰富和修正，IDEF0 当然也不是完善得不必再更改的。譬如 IDEF0 的层次体系是清楚的，但简单的结点树终究还不能表示复杂系统的全貌。我们在工作中把结点树扩展成了“家谱图”，把同一“代”(generation)的模块号和模块名放在同一水平(或纵列)内，这样可看到整个系统分解的全貌，可以分析各个分支分解的不同深度，作为全局检查的一种参考图。又如 IDEF0 与 IDEF1 原来是独立开发的，如能建立其联系，将能互相补充互相促进，对系统的分析和设计会有好处。总之，有很多方面都还有待大家共同来修正和发展。

附 录 1

1.1 IDEF0 图纸格式标准



1.2 IDEF0 组文件封面格式



第 二 篇

IDEF1X 方法

第1章 引 言

本篇旨在详细说明开发数据语义特征逻辑模型新概念的技术和过程。在企业中，这些语义数据模型可以作为一个数据源支持数据管理、信息系统集成和计算机数据库的构造。

70年代中期，美国空军的ICAM计划首次意识到语义数据模型的必要性。这一计划的宗旨是通过系统地应用计算机技术来提高机械制造业的生产率。ICAM认为：对于想要提高生产率的人员来说，需要一个好的分析和通讯技术(交流技术)。所以ICAM计划开发了大家了解的一系列IDEF(ICAM DEFinition)方法。最初阶段，这一系列方法只包括3种不同的建模方法，这3种方法用图形方式来刻画整个制造企业环境系统。

(1) IDEF0被用来产生“功能模型”，这个模型结构化地描述了所研究系统的活动和处理进程。

(2) IDEF1是用来开发“信息模型”的，这个信息模型表述了制造系统环境的信息结构和语义。

(3) IDEF2被用来开发“动态模型”。它表述环境或系统时变行为的特征。

1983年美国空军ICAM计划着手IISS(integrated information support system)项目。此项目的目的是提供一套能逻辑地和物理地集成异种计算机硬件和软件的技术。IISS把集成的途径放在数据源单一语义定义的获取、管理和使用上，这个“概念模型”(conceptual model)是用IDEF1建模技术来定义的。

本篇详细说明一个IDEF1的扩展版本(被称为IDEF1X)。这个扩展版本是基于IISS项目以及一些工业应用的需要和经验提炼出来的。IDEF1X对IDEF1做了一些改进、IDEF1X增强了图形表述能力，丰富了语义和简化了开发过程。1980年来，这一扩展版本(IDEF1X)已在应用中得到发展和验证。

本篇可用作IDEF1X建模指南和参考手册。在本篇的第1章引言之后，在第2章介绍一般的数据建模概念。IDEF1X建模的具体语法和语义将在第3章给出。建模所采用的方式是多种多样的，假设不采用自动支持建模，那么建模的基本过程如第4章所述。第5章介绍了编制文档和评审过程。最后还以附录形式给出了名词术语的总结以及IDEF1X与IDEF1的差别。

第 2 章　数据模型化的概念

本篇着眼于介绍 IDEF1X 数据建模的语法和过程。在详细介绍 IDEF1X 技术细节之前，这一章将先介绍一下数据模型化的重要性何在，什么是 IDEF1X 方法的总体目标。

2.1　把数据作为资源管理

过去的 20 年里，一些第一流的公司都逐步意识到：把数据作为一种资源管理起来是很必要的。其原因主要是：频繁变化的企业环境中，竞争对柔性的要求，许多公司必须不断改变他们的组织和工艺过程，以期能适应技术上的进步和市场的变化，从而使公司能迅速且平稳地改组，各公司必须辨别和管理任务的基础结构，也就是要了解执行任务所必需的数据以及它们之间的关联信息。为了对付好数据管理问题，许多公司都已组成了专门小组（如数据管理处或信息资源管理处）。但是，数据的迅速增加并且千变万化，使得这些小组的工作越来越复杂。现有的数据与现行任务的需要之间存在着很大的不一致性，不适时性和非柔性等等，不难看出，为了管理数据，我们必须了解它的基本特征。数据可以被认为是：有含义事件的符号表示，而一个单一的含义能被用于许多不同的事件。例如："邮政编码"的含义可以被用于许多 6 位数字的数；一个没有含义的事件，是无用的；一个带有错误含义的事件将带来"麻烦"和"灾难"。由此，数据管理的着眼点应该放在与数据有关的含义上。

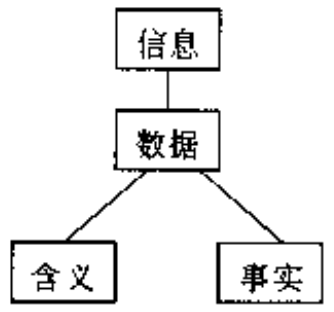


图 2.2.1　信息的组成

"信息"是为某个特定目的或在一定范围内聚合起来的数据集（如图 2.2.1 所示）。这就意味着许多不同类型的信息可能产生于同一数据。据统计，400 个数据可以组合成 10 的 869 次方个不同形式的信息，因此，管理信息资源的策略应该放在管理数据被应用于事件的含义上，这要比着眼于试图控制或限制信息的产生要好得多。

2.2　三模式概念

这几年来，建立信息系统的技术和技巧已有了惊人的发展。但是，就大多数而言，建立系统的传统方法仅仅是着眼于从以下两个不同的观点来定义数据：用户和计算机视图。对于用户视图（也被称为"外部模式"），数据定义从记录和屏幕角度来设计，以帮助各个单位处理他们自己的具体任务。而从使用观点来说，对数据结构的要求，是随事务环境和各用户的偏好而变化的；对于计算机视图（也被称为"内部模式"）来看，数据以文件结构方式来

存储和检索，计算机存储数据的结构是根据所用的具体计算机技术和高效处理数据的要求而定的。

在过去一些年里，分析家们通过处理一个一个的具体事务要求，逐步定义出了这两种数据视图，如图 2.2.2 所示。通常，为一个最初应用所定义的内部模式，并不能方便地用于后续的应用，结果导致了数据冗余和不一致性。在早期的信息系统中，用物理记录形式定义数据和按顺序方式处理数据，此后，由于柔性的要求，导致了数据库管理系统(DBMS)的引入。DBMS 允许随机存取逻辑上相关的数据。在 DBMS 中，逻辑数据结构通常被定义为层次的、网状的、关系的或者面向对象的。尽管 DBMS 大大地改进了数据的共享性，但是单单使用一个 DBMS 并不能保证数据的一致性。因此，大部分公司已不得不开发了多个数据库。这些数据库常常是由不同的 DBMS 来控制的，即使如此，也依然存在冗余和不一致性问题。

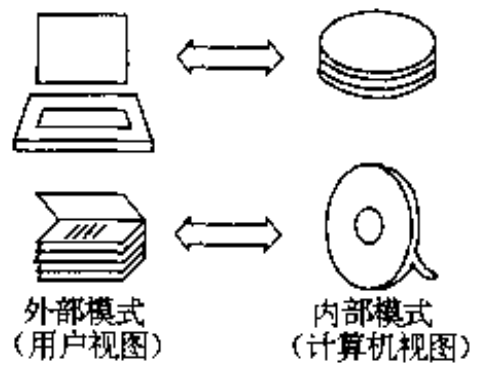


图 2.2.2　传统的数据视图

由于意识到以上问题，美国关于计算机和信息处理的国家标准委员会的下属标准规划和需求委员会 ANSI/X3/SPARC (American National Standards Committee on Computer and Information Processing/ Standards Planning & Requirements Committee) 的 DBMS 研究小组认为：在一个理想的数据管理环境中，需要数据的第 3 个视图。这个视图被称为“概念模式”，它是数据的一个单一集成定义，在一个企业中，它不偏向于任何专门的数据应用，同时还独立于数据的物理存储和存取方式，如图 2.2.3。这个概念模式的主要目标是提供一个数据的含义和相互关系的一致定义，从而用来集成、共享和管理数据的完整性。概念模式必须有以下 3 个重要的特性：

1　它必须和企业基础结构相一致，并且真正覆盖所有的应用范围。

2　它必须是可扩展的，如：可以在原先定义的数据不变更的情况下，再定义新的数据。

3　概念模式必须可转化成用户所需的视图和多种数据存储及存取结构。

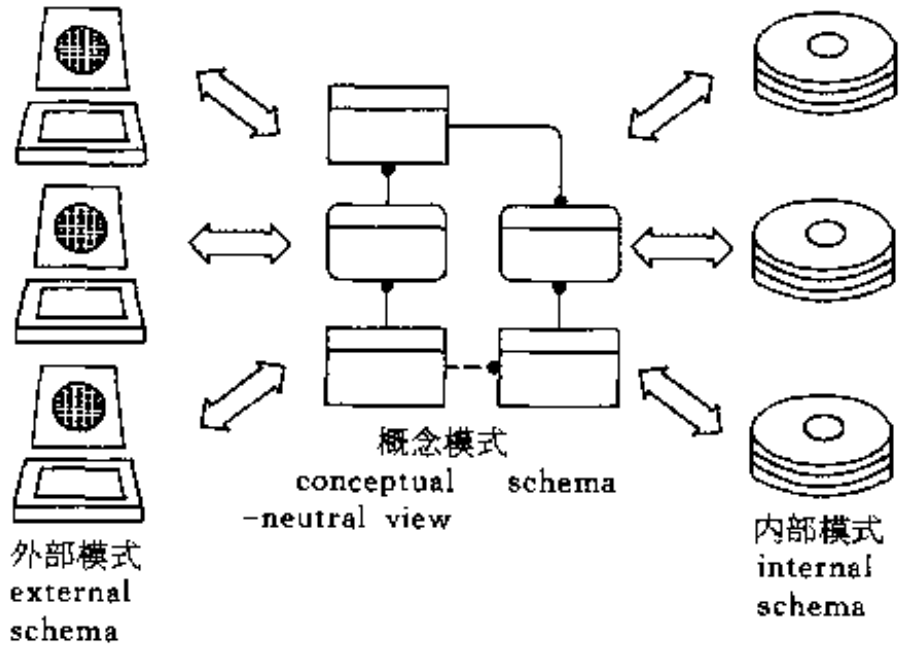


图 2.2.3　三模式方法

2.3 数据模型化的目的

一个 DBMS 的逻辑数据结构，无论是层次的、网状的、还是关系的，都不能完全满足数据概念定义的需要，这是由于它在范围上受到限制，且偏向于被 DBMS 用来实现对策。因此，从概念视图定义数据的要求导致了语义数据模型化技术的发展。那就是在数据相互关系中定义数据的含义，如图 2.2.4 所示。现实世界[就资源（resources）、概念（ideas）、事件（events）等而论]被符号化地定义在物理数据存储器内。语义数据模型是一个抽象化的定义，它定义了所存储的符号如何与现实世界相关联。因此，语义数据模型必须真实地反映现实世界。语义数据模型有许多用途，一些主要用途列举如下：

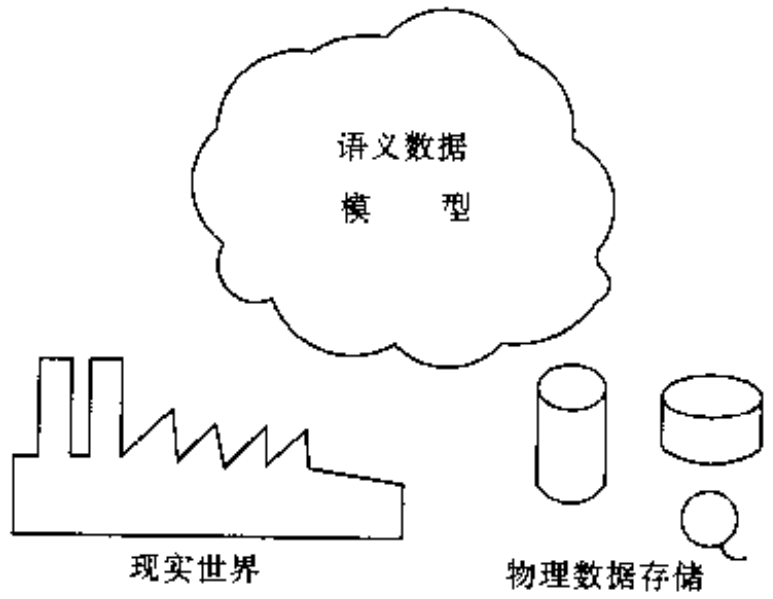


图 2.2.4　语义数据模型

（1）计划数据资源。

（2）用来建立可共享的数据库。

（3）评估一些厂家的软件。

（4）用来集成现有数据库。

2.4 IDEF1X 方法

IDEF1X 是语义数据模型化技术，它主要用来满足下列需要和应具有的特性：

（1）支持概念模式的开发。

IDEF1X 语法支持概念模式开发所必需的语义结构，完善的 IDEF1X 模型具有所期望的一致性、可扩展性和可变换性。

（2）IDEF1X 是一种相关语言。

IDEF1X 对于不同的语义概念都具有简明的一致结构。IDEF1X 语法和语义不但比较易于为用户掌握，而且还是强健而有效的。

（3）IDEF1X 是便于讲授的。

语义数据模型对许多 IDEF1X 用户都是一个新概念。因此，语言的易教性是一个重要

的考虑因素，设计 IDEF1X 语言是为了教给事务专业人员和系统分析人员使用，同样也是教给数据管理员和数据库设计者使用的。因此，它能用作不同学科研究小组的有效交流。

(4) IDEF1X 已在应用中得到很好地检验和证明。

IDEF1X 是基于前人多年的经验发展而来的，它在美国空军的一些工程和私营工业中充分地得到了检验和证明。

(5) IDEF1X 是可自动化的。

IDEF1X 图能由一组图形软件包来生成。并且，空军已开发出来了一个现行三模式词典，在分布异构环境中，它利用所得的概念模式来进行应用开发和具体事务的处理。商品化的软件还能支持 IDEF1X 模型的更改、分析和结构管理。

IDEF1X 把实体 -联系方法应用到语义数据模型化中，IDEF1 的最初形式是在 P. P. S (Peter) Chen的实体联系模型化概念与 P. P. (Ted) Codd 的关系理论的基础上发展起来的。此次介绍的 IDEF1X 是 IDEF1 的扩展版本。IDEF1X 除在图形表达和模型化过程方面的改进外，还对语义进行了增强和丰富。例如：分类联系的引入[也称概括联系]。IDEF1X 语言还编入了 DApplelon 公司和数据库设计组商品化的开发工作。

IDEF1X 模型的基本结构是：

(1) 包含数据的有关事物。例如：人、概念、地方和事物等用盒子来表示。

(2) 事物之间的联系用连接盒子的连线来表示。

(3) 事物的特征用盒子中的属性名来表示。

基本结构如图 2.2.5 所示，详细说明请见以下章节。

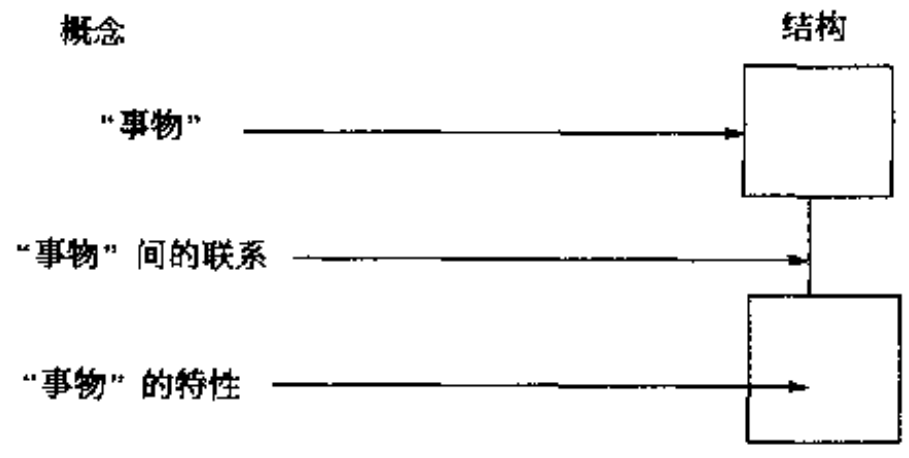


图 2.2.5 基本模型化概念

第 3 章　IDEF1X 的语法和语义

本章将介绍 IDEF1X 模型中每一成分的语义(或含义),表示各成分的图形语法和一些使用规则,尽管各成分是紧密相关的,但在以下详述时,并不考虑结构的实际次序而是分别地详细叙述每一成分,至于建立一个 IDEF1X 模型的过程,那将在第 4 章中详述。每个 IDEF1X 模型都必须遵守被定义的语法和语义。

IDEF1X 模型的成分有:

(1) 实体 (entities)

- 独立标识实体 (identifier-independent entities)
- 从属标识实体 (identifier-dependent entities)

(2) 联系 (relationships)

- 可标定连接联系 (identifying connection relationships)
- 非标定连接联系 (non-identifying connection relationships)
- 分类联系 (categorization relationships)
- 非确定联系 (non-specific relationships)

(3) 属性/关键字 (attributes/keys)

- 属性 (attributes)
- 主关键字 (primary keys),亦称"主键"
- 次关键字 (alternate keys),亦称"次键"
- 外来关键字 (foreign keys),亦称"外来键"。

3.1　实　　体

1　实体的语义

一个"实体"表示一个现实和抽象事物的集合,这些事物必须具有相同的属性或特征。这个集合的一个元素就是这一实体的一个"实例"(instance)。一个现实世界的事物可以由数据模型中的多个实体来表示。例如:张三可以同时是"雇员"实体和"顾客"实体的一个实例。而一个实体的实例又可以描述现实世界事物的一个组合。例如:约翰和玛丽就可以是"婚配双方"实体的一个实例。

如果一个实体的每个实例都能被唯一地标识,而不决定于它与其他实体的联系,那么该实体就是"独立标识符实体"或简称为"独立实体";如果实体的一个实例的唯一标识,依赖于该实体与其他实体的联系,那么它就是"从属标识符实体"或简称为"从属实体"。

2 实体的语法

IDEF1X 用盒子来表示实体(见图 2.3.1 所示),圆角盒子用来表示从属实体,方角盒子用来表示独立实体。每一个实体分配一个唯一的名字和号码,名字和号码之间用斜杠(/)分开,放在盒子的上方,这个号码必须是正整数。实体名必须是一个名词短语,这个名词短语描述了实体所表示事物的一个集合。这个名词短语应该是单数而不是复数的,且允许用缩写和字母缩写词,但实体名必须是有意义的,且在整个模型中保持一致。实体的形式定义和同义词(或别名)的清单,必须在模型的词汇表中给出。一个实体可以出现在多张 IDEF1X 图上,但在一张图中,只出现一次。

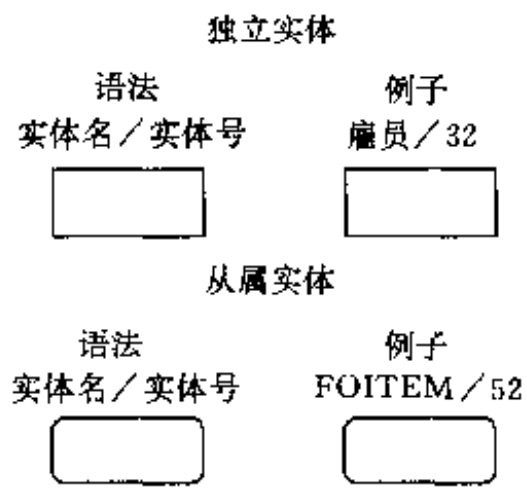


图 2.3.1 实体的语法

3 实体规则

(1) 每一个实体必须使用唯一的实体名,相同的含义必须总是用于同一实体名。而且相同的含义决不能用于不同的实体名,别名除外。

(2) 一个实体可以有一个或多个属性,这些属性可以是它自身所具有的,也可以是通过一个联系而继承得到的(见 3.7 节中的外来关键字部分)。

(3) 一个实体应有一个或多个能唯一标识实体每一个实例的属性(见 3.6 节中的主关键字和次关键字部分)。

(4) 任意实体都可与模型中任意其他的实体有任何联系。

(5) 如果一个完全外来关键字是一个实体主关键字的全部或部分,那么该实体就是从属实体。相反地,如果根本没有外来关键字属性用做一个实体的主关键字,那么这个实体就是独立实体。

3.2 连 接 联 系

1 连接联系语义

一个“确定连接联系”(specific connection relationship)或简称“连接联系”(connection relationship)[还可称为“父子联系”或“依存联系”(parent-child or existence-dependency relationship)],是实体之间的一种连接或联结,这种连接联系中,被称为父实体的每一个实例与子实体的 0 个、1 个或多个实例相连接,子实体的每个实例精确地同父实体的 1 个实例相连接。也就是,只有与之相连接的父实体的实例存在时,子实体的这个实例才能存在。例如:假设“买主”与“订单”两实体之间存在确定的连接联系,则若一个买主(实例)发出 0 张、1 张或多张订单(实例),并且每张订单必须只由一个买主发出。IDEF1X 模型描述的是两实体间的联系类型或类型集合。而一个具体的联系实例,只是使两个实体的具体实例发生联系。例如:“买主约翰发出 123 号订单”就是一个联系的实例。

连接联系可以由确定联系基数(cardinality)来进一步定义。那就是,详细说明对子每个父实体的实例,可以对应存在几个子实体的实例。在 IDEF1X 中,能够表示下列不同类型的联系基数:

(1) 每个父实体的实例可以与 0 个、1 个或多个子实体的实例相连接。

(2) 每个父实体的实例至少与 1 个或多个子实体的实例相连接。

(3) 每个父实体的实例能够没有或至多与一个子实体的实例相连接。

(4) 每一个父实体的实例与确定个数的子实体的实例相连接。

连接联系又因父子关系上的差别而分为 2 种：

(1) 如果子实体的每个实例都是由它与父实体的联系而确定的，那么这个关系就被称为一个“可标定联系”(identifying relationship)。例如：“任务”是指完成一项具体的工作；“项目”则是指为了某一特定的目标而做的一系列工作，直至最后达到该目标的整个过程。“项目”中当然有一系列“任务”，但一个任务也可以同时为几个项目所用。如一个或多个任务与每一个项目相关，且各任务在一个项目中能唯一地确定，那么“项目”和“任务”两实体之间存在确定关系。也就是，为了从所有其他任务中唯一地确认某个任务，就必须知道该任务所相关的项目(可参见 3.7 节的外来关键字部分)。

(2) 如果子实体的每一个实例都能被唯一地确认而无须了解与之相联系的父实体的实例，则这个关系就被称为“非标定联系”(non-identifying relationship)。例如：尽管买主实体和订单实体之间可以存在依存联系(existence-dependency relationship)但是订单实例可以由订单编号唯一地确定而无需确定它与买主的关系。

还有更复杂的情况，就是要对多重联系做出一个阐明其作用的断言。断言(assertion)的一种可以表示为两个或多个联系之间的布尔约束(boolean constraint)。例如：一个“异或”(exclusive OR)约束规定：对于一个给定的父实体的实例，如果一种类型的子实体的实例存在，那么另一种类型的子实体的实例将不能存在，也就是，如果父实体和子实体都涉及同一现实世界的事物，那么就存在一个潜在的分类联系(categorization relationship)(关于这一联系参见 3.3 节介绍)。

另一种约束是一个“路径断言”(path assertion)，在两实体通过不同联系次序直接或间接地相联系时，路径断言用来约束父实体和子实体的具体实例，例如：一个“部门”实体具有两个子实体“雇员”和“项目”。如果“雇员”和“项目”有一个公共的子实体，“项目分配”实体，那么“项目分配”实体通过两条不同的联系路径(relationship path)间接地与“部门”实体相联系，一个路径断言可以陈述为：“雇员只能分配到他所在部门的项目中”。

2 连接联系的语法

一个确定连接联系用父子实体之间的一条连线来表示，连线的子实体端带有一个小圆点。在此小圆点旁边用字母“P”(positive)表示 1 或多个的基数，用“Z”表示基数为 0 或 1，用正整数表示基数的确定值；缺省情况表示基数可以是 0，1 或任意大于 1 的正整数，如图 2.3.2 所示。

(缺省)	0，1 或大于 1 的正整数
P	1 或大于 1 的正整数
Z	0 或 1
n	确定基数 n

图 2.3.2 关系基数的语法

在 IDEF1X 图中，用一条实线来表示父子实体之间存在标定联系，那么子实体总是从属标识符实体，用圆角的盒子来表示，此时，父实体的主关键字属性还为子实体的主关键

字属性所继承。如图 2.3.3 所示。(可以参见 3.7 节的外来关键字部分)。

在可标定联系中,父实体一般是独立实体,但如果该父实体又在其他某个联系中充当可标定联系的子实体,那么该实体和子实体都是从属实体。一个实体可以与其他实体有任意多个联系,但是,只要它在任一个可标定联系中充当了子实体,就用圆角盒子来描述,而与它在其他联系中充当什么角色无关。

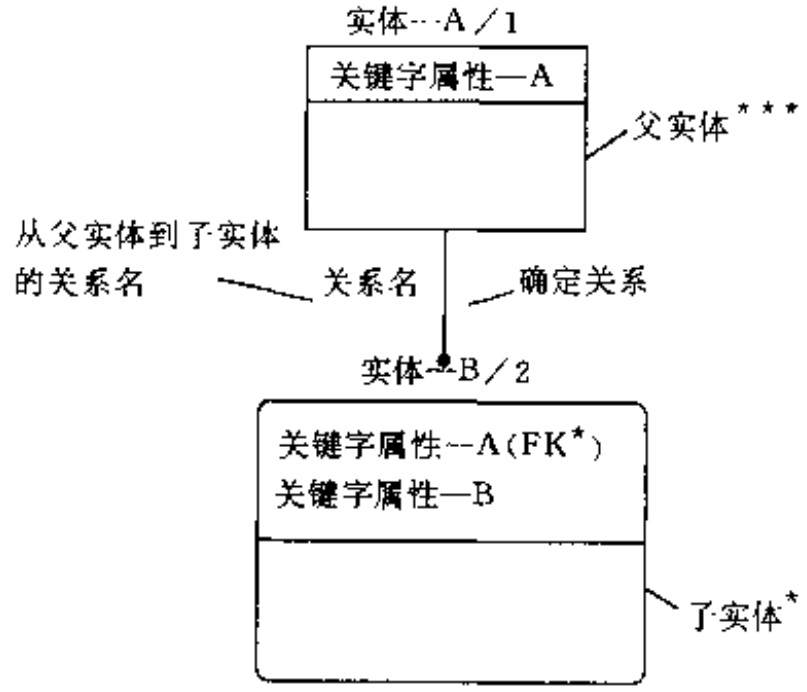


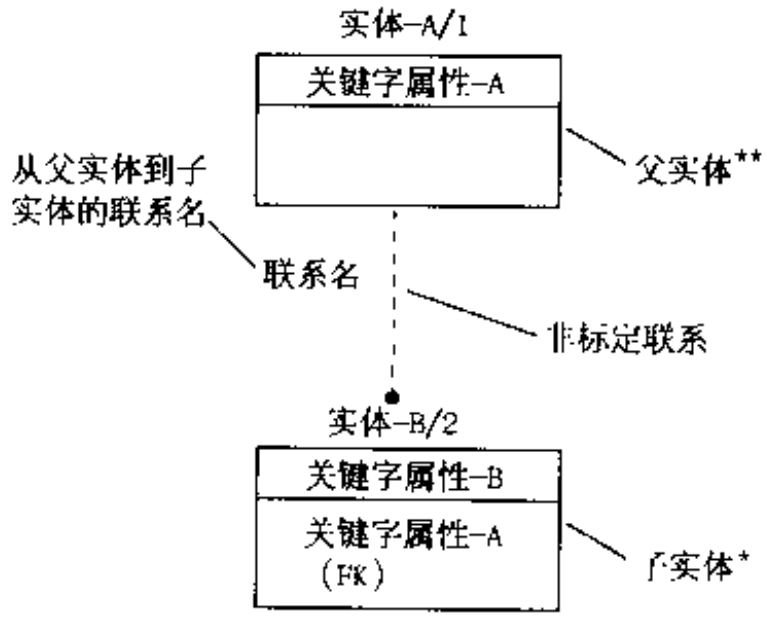
图 2.3.3 可标定联系实例

IDEF1X 用虚线来描述父子实体之间的非标定联系。如图 2.3.4 所示,一般情况下,在非标定联系中的父实体和子实体将都可以是独立标识符实体,但如果该非标定联系中的父实体或子实体在某个其他标定联系中充当了子实体的角色,那么它就是一个从属标识符实体。

一个联系应用动词短语(verb phrase)来命名,联系名被放在关系连线的旁边。相同的两个实体之间的联系名必须是唯一的,但对整个模型而言,并不要求所有联系名是唯一的。联系名总是按父亲到儿子的方向来表示。因此,一个语句可形式化为下列 4 部分的组合:"父实体名"+"联系名"+"基数表达式"+"子实体名"。例如:如果"包含"表示联系名,"项目"用作父实体名,"任务"用作子实体名,"P"表示基数标志,那么从这个父子实体联系我们可以得到这样的描述:"一个项目包含一个或多个任务"。请注意一点:尽管从子实体到父实体方向的反方向描述并没有明确给出,但当反方向(子→父)描述时,以上正向(父→子)联系描述也应该保持是真的(true)。反向(子→父)描述时,上例可改为:"一个任务必须且仅属于一个项目"。

3 连接联系的规则

(1) 一个确定性联系总是存在子两个实体之间,一个作为父实体,另一个作为子实体。



注：* 非标定联系中，子实体可以是独立实体。除非它又是另外某个标定联系的子实体，那它是从属实体；

** 非标定联系中的父实体可以是独立实体（如本图所示），也可以是从属实体，这依据其他关系而定。

图 2.3.4 非标定联系的语法

(2) 子实体的一个实例必须且总是恰好地与父实体的一个实例相联系。

(3) 父实体的一个实例可与子实体的 0 个、1 个或多个实例相联系，具体情况由特定的基数而定。

(4) 在标定联系中的子实体总是一个从属标识符实体。

(5) 一个实体可以与任意多个其他实体相联系，可以在不同的联系中充当不同的角色，如在一些联系中当父实体，而在另一些联系中当子实体。

3.3 分 类 联 系

1 分类联系的语义

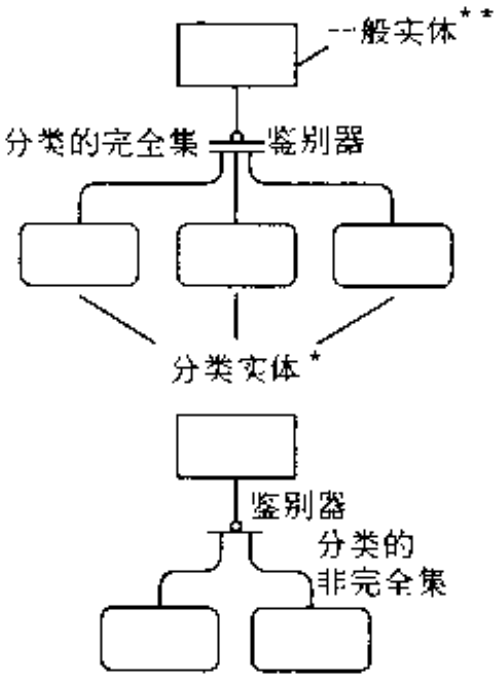
为了从一些现实事物中获取信息，我们用实体来表示有关事物的概念。由于一些现实世界的事物是其他事物的类，因此，有些实体也必须在某种意义上是其他实体的类。例如，假设“雇员”是一个要被获取信息的事物。尽管这个“雇员”实体已有了一些关于所有雇员的所需的信息，但对于拿月薪雇员，也许还需要有不同于计时雇员的附加信息。因此，月薪雇员和计时雇员就是“雇员”实体的分类，在 IDEF1X 模型中，他们通过一个分类联系互相联系。

“完全分类联系”(complete categorization relationship)是两个或多个实体之间的联系，且这些实体中，存在一个一般实体(generic entity)，它的每一个实例都恰好与一个且仅一个分类实体(category entity)的一个实例相联系。一般实体的每一个实例和与之相关的一个分类实体实例，描述的是现实世界的同一事物，因此它们具有相同的唯一标识符。如上例，“雇员”实体是一般实体，而“月薪雇员”和“计时雇员”则是分类实体。对于同一个

一般实体的分类实体总是互相不相容的，也就是，一般实体的一个实例只能与一个分类实体的一个实例相对应。在上例中，这就意味着一个雇员不可能既是月薪雇员又是计时雇员。此外，IDEF1X 语法还允许类的不完全集的情况出现，如存在一般实体的一个实例不与任何分类实体的任一实例相关联，那么，这个联系就被定义为“不完全分类联系”(incomplete categorization relationship)。在一般实体实例中的一个属性值，确定哪一个可能有的分类实体实例将与该一般实体实例相关联。这个属性被称为分类联系的“鉴别器”(discriminator)。以上的例子中，鉴别器可以命名为雇员类型。

2 分类联系的语法

分类联系用如图 2.3.5 所示的线、圆圈加下划线和几条分支线组合起来表示。对分类实体而言，基数不必说明，因为总是 0 或 1，分类实体全部是从属实体。而一般实体通常是独立实体，当然，也不排除从属实体的可能性，因为，它的标识符可由一些其他联系而继承得到。



注： * 分类实体总是从属实体。

** 一般实体可以是独立实体(如图上所示)或从属实体。

图 2.3.5 类别关系的语法

分类联系表示中，圆圈的下划线是双线时，这表示分类实体集(set)是完全的，而单划线时，则表示分类的非完全集。一般实体中，用作鉴别器的属性名字写在圆圈旁边。尽管分类联系本身并没有明确地命名，但一般实体到分类实体的联系可读作：“可以是”(“can be”)。例如，雇员“可以是”一个月薪雇员。对于分类的完全集而言，判别联系可读作“必须是”(“must be”)。例如，雇员“必须是”一个月薪雇员或一个计时雇员。相反方向的联系读作：“是一个”(“is a/an”)。例如：一个计时雇员“是一个”雇员。在分类联系中，一般实体和每一个分类实体必须具有相同的关键字属性。作用名(role name)可以被用于分类实体(可参见 3.7 节的外来关键字部分)。

3 分类联系的规则

(1) 一个分类实体只能有一个对应的一般实体。那就是，对于一个分类联系而言，它只能是一个分类集的一个成员。

(2) 一个分类联系中的一个分类实体可以是一个其他分类联系中的一般实体。

(3) 一个实体可具有任意个分类联系，在这些分类联系中，这个实体作为一般实体。例如："女雇员"和"男雇员"可以是一般实体"雇员"的第二个分类集合。

(4) 一个分类实体不能是可标定连接联系中的一个子实体。

(5) 分类实体的主关键字属性必须和一般实体的主关键字属性相同。

(6) 一个分类实体的全部实例都具有相同的"鉴别器值"(discriminator value)，并且不同分类实体的所有实例都具有不同的鉴别器值。

3.4 非确定联系

1 非确定联系的语义

父子连接和分类联系都被认为是"确定联系"(specific relationship)，它们都确切地定义了一个实体的实例如何与其他实体的实例相关联。在完善的 IDEF1X 模型(fully refined IDEF1X models)中，实体间的所有联系都必须用确定联系来描述。但是，在建模的初始阶段，标识一些两实体间的"非确定联系"(non-specific relationship)常常对开发模型是有帮助的。这些非确定联系，在进一步的建模阶段中不断地得到细化。处理非确定联系的过程将在 4.4 节中给出。

一个非确定联系又称为"多对多关系"(many to many relationships)，这种联系，就是关联两实体之间的任一实体的一个实例，都将对应另一实体的 0 个、1 个或多个实例。例如，一个雇员可以分配到多个项目中，而一个项目可分配多个雇员，那么"雇员"实体和"项目"实体之间的连接关系就是一个非确定联系。在建模中，通过引入第三个实体的方法，这个非确定联系能够用确定联系来替换。例如，在上例中我们引入第三个实体"项目分配"，该实体在与"雇员"实体和"项目"实体的确定联系中，充当"雇员"和"项目"实体的公共子实体(common child entity)。这样新的联系被确定为：一个雇员具有 0,1 或多个"项目分配"，且一个项目也具有 0,1 或多个"项目分配"。每个项目分配都是确切地针对一个雇员和一个项目。这个为处理非确定联系而引入的实体，通常被称为"相交实体"(intersection entity)或称为"相关实体"(associative entity)。

通过从联系的双向来确定联系基数，一个非确定联系可进一步被定义，几种基数的任意组合都可以用来指定一个非确定关系，那就是，对于第一实体的每一个实例存在

- 0,1 或多个；
- 1 或多个；
- 0 或 1 个；
- 或一个确切的数目的

第二个实体的实例与之相关联，同样对于第二个实体的每个实例也存在：

- 0,1 或多个；
- 1 或多个；
- 0 或 1 个；
- 或一个确切数目的

第一实体的实例与之相关联。请注意：在关联的任一端存在一个确切数“1”时，则这个联系就是一个确定联系，而不再是非确定联系了。

2　非确定联系的语法

一个非确定联系用一个两端都带圆点的连线来描述。请见图 2.3.6。联系基数可以在联系的两端表示，如图 2.3.2 所示。标在圆点旁的“P”，表示联系另一端实体的每一个实例，都对应本端实体的 1 或多个实例。标在圆点旁的“Z”，表示联系另一端实体的每一个实例，都对应本端实体的 0 或 1 个实例。同样，连线上圆点旁的一个正整数或正整数的最大最小范围，说明了确切的基数值。缺省情况下，基数为 0、1 或更多。

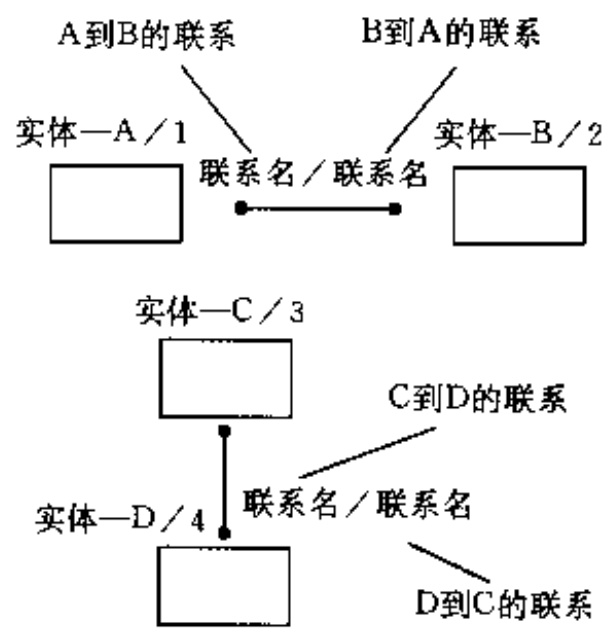


图 2.3.6　非确定联系的语法

一个非确定联系是被双向命名的。联系名用动词短语来描述。联系名称用斜杠“/”分开且放在连线旁边。联系名的次序由实体之间的相对位置而确定。如果两个实体是水平排列的，第一个联系名按从左到右方向来描述；如果两实体是垂直排列的，那就按从上到下的方向来描述。第二个联系名按第一联系名相反的方向来描述。

3　非确定联系的规则

（1）一个非确定联系总是存在于两个实体之间，而不是三个或更多个实体之间。

（2）两个实体中，任意一个实体的实例可以与另一实体的 0，1 或多个实例相关联，具体情况要视基数而定。

（3）为了完全地设计出一个模型，非确定联系必须由确定联系来替代。

3.5　属　　性

1　属性的语义

一个“属性”表示一类现实或抽象的事物的一种特征或性质（如：人、物、地点、事件、概念等），一个“属性实例”（attribute instance）是这类事物一个个体成员的具体特性。属性实例由特征型和值来定义，这个值也称为“属性值”。在一个 IDEF1X 模型中，属性是与具体的实体相联系的。实体的每一个相关属性都必须具有一个单一且确定的值。例如：雇员名和生日是与实体“雇员”相关的属性，“雇员”实体的一个实例可有这样的值：“珍妮”和“2/27 1953”。

一个实体必须具有一个属性或属性组，其值唯一地确定该实体的每一个实例。这个属性或属性组就构成该实体的“主关键字”（primary key），请参见 3.6 节。例如，属性“雇员号”就可以作为“雇员”实体的主关键字；而属性“雇员名”和“生日”只是其他属性，因为它不能唯一地确定该实体的每一个实例。

在 IDEF1X 模型中，每一个属性仅仅属于一个实体，并且，对于每一个与实体相关的属性而言，该实体的每一个实例都必须具有一个值。也就是，实体的属性必须对所有该实

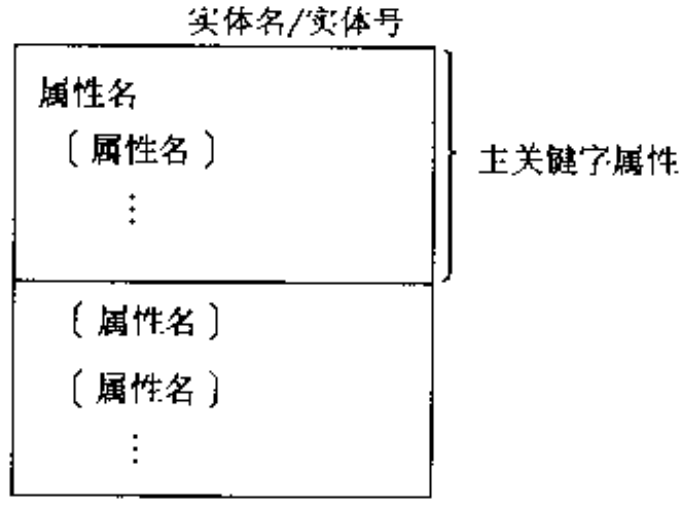
体所代表事物集合中的每个组元都适用，例如，因为有计时雇员存在，属性“月薪”只能适用于一部分雇员，而不能适用于全体雇员。因此，为了为属性“月薪”建立所属关系，我们可以定义一个分离且相关的“月薪雇员”实体，由于一个月薪雇员将同时由两个实体(“雇员”实体和“月薪雇员”实体)的一个实例来描述，对于所有雇员的公共属性(例如：雇员名和生日等)就不必在“月薪雇员”实体的属性中出现。

“属于”某个实体的一些属性是该实体所描述事物的一些基本特征，除这些属性外，还有一种“被继承”的属性。这种属性可以通过一个确定的连接联系或分类联系“继承”而得到。并且该实体在这些联系中充当儿于或分类实体。(参见 3.7 节)。例如，如果每一个雇员都必须分配到一个部门，那么属性“部门号”将是“雇员”实体的一个属性。而这个属性就是通过“雇员”实体与“部门”实体的联系继承而得到的。“部门”实体是属性“部门号”的所有者。请注意：只有主关键字属性才可以通过联系被遗传。例如：若“部门名”属性不是“部门”实体主关键字的一部分，那么“部门名”属性就不是“雇员”实体的被继承属性。

2 属性的语法

每个属性都由一个唯一的名字来标记，这个名字用一个名词短语来表示，它描述了属性所表示的实体特征。名词短语采用单数，而不使用复数形式，允许使用缩写和英文首字母缩写词，但是属性名必须有意义并且对整个模型是一致的。属性的形式定义和同义词(或别名)的清单必须在模型的词汇表中列出。

在实体的盒子内，属性被列出且每一行只列一个属性，主关键字属性被放在列表的最上面，且用水平线把它与其他属性分开。如图 2.3.7 所示。



例子：

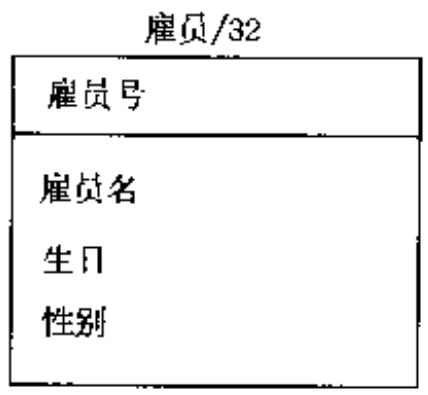


图 2.3.7 属性和主关键字语法

3　属性规则

(1) 每个属性都必须具有一个唯一的名称，且相同的名字必须总是描述相同的含义。因此相同的含义不可能对应于不同的名字(别名除外)。

(2) 每个实体可以具有任意个属性，一个属性只能归属于一个实体，这一规则称为“单主规则”(single-owner rule)。

(3) 一个实体可以有任意个继承属性，而每个继承属性都必须是某个相关的父实体或一般实体主关键字的一部分。

(4) 实体的每一个实例，对每一个属性都必须具有一个值。这个规则称为“非空规则”(no-null rule)。

(5) 对于同某实体相关的属性而言，该实体没有一个实例可能具有一个以上的值。这个规则称为“非重复规则”(no-repeat rule)。

3.6　主关键字和次关键字

1　主关键字和次关键字语义

一个实体的一个候选关键字(candidate key)，可以由一个或多个属性组成，它唯一确定实体的每一个实例。例如：属性“订单号”可以唯一地确定“订单”实体的实例。又如：“账号”和“支票号”的组合可以唯一地确定“支票”实体的实例。

每一个实体至少有一个候选关键字。有些情况下，一个实体可以有多个候选关键字，例如：属性“雇员号”和“身份证号”都唯一地确定“雇员”实体的实例。如果对一个实体存在多个候选关键字，那么必须指定其中一个为“主关键字”，而其他候选关键字为“次关键字”(alternate key)。如果只有一个候选关键字，那么它当然就是主关键字。

2　主关键字和次关键字的语法

主关键字属性，放在实体盒子内的属性表顶部，用水平线与其他属性分开。每一个次关键字，分配一个唯一的整数号，这个整数号放在“AK”标注之后，然后用圆括号括起来，放在次关键字属性之后，如图 2.3.8 所示。个别属性也可以被确定为多个次关键字的一部分，一个主关键字属性也可以用作次关键字的一部分。

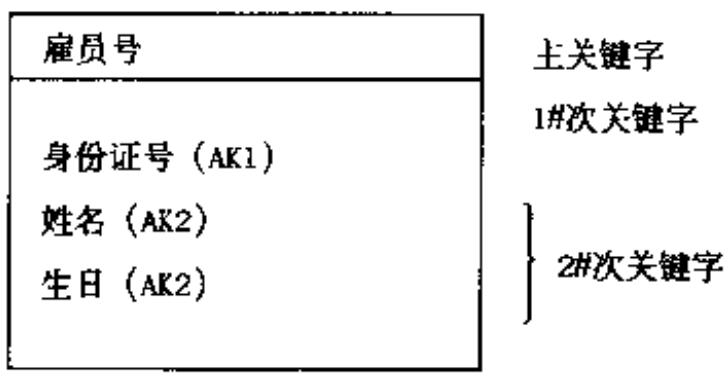


图 2.3.8　次关键字的语法

属性名(AK n [, AK m]…)

这儿 n，m…用来唯一地确定每个次关键字所包含的有关属性，也就是用一个同样的

标记来注出某个次关键字的所有属性。

3　主关键字和次关键字的规则

(1) 每个实体必须有一个主关键字。

(2) 每个实体可以有任意个次关键字。

(3) 主关键字或次关键字可以由单个或多个属性组成。

(4) 个别属性可以是多个关键字(主关键字或次关键字)的一部分。

(5) 构成主关键字或次关键字的属性可以是实体自身所具有的,或由某些联系继承得到的属性。(参见本第 3.7 节的外来关键字部分)。

(6) 主关键字和次关键字必须仅仅包含有助于唯一标识实体的那些属性。也就是,如果主关键字或次关键字中去掉任一部分属性,那么就无法唯一地确定实体的实例。这一规则被称为"最小关键字规则"(smallest-key rule)。

(7) 如果主关键字是由多个属性所组成的,那么每个非键属性的值必须完全函数依赖于主关键字,也就是,如果主键已知,则就知道了每个非键属性的值,如果只有主关键字的一部分属性被确定了,那么非键属性的值无法唯一确定。这一规则称为"完全函数依赖规则"(full functional-dependency rule)。

(8) 每个非键属性,必须是仅仅函数依赖于主关键字和次关键字,也就是,没有一个非键属性的值能够由其他非键属性值所确定。这个规则被称为"非传递依赖规则"(no-transitive-dependency)。

3.7　外来关键字

1　外来关键字的语义

如果在两个实体之间存在确定连接和分类联系,那么构成父实体或一般实体主关键字的属性将被继承为子实体或分类实体的属性。这些继承属性被称为"外来关键字"。如果在"项目"和"任务"两实体之间存在一个连接联系,"项目"充当父实体,"任务"充当子实体,那么"项目"实体的主关键字属性将被继承为"任务"实体的属性。例如:如果"项目标识号"属性是"项目"实体的主关键字,那么"项目标识号"将是"任务"实体的继承属性或"外来关键字"(foreign key)。

一个继承属性可以在一个实体中作为部分或全部主关键字、次关键字或非键属性。如果父实体的主关键字的所有属性继承为子实体的主关键字的一部分,那么,用来继承这些属性的联系就是"标定型连接联系"。如果任一继承属性都不是主关键字的一部分,那么这个联系为"非标定型连接联系"。请参见 3.2 节。例如,如果任务仅在一个项目内唯一地编号,那么继承属性"项目标识号"将与"任务"实体本身的属性"任务号"组成"任务"实体的主关键字。这两个实体"项目"和"任务"之间的关系就是标定型连接联系。在另一种情况下,如果属性"任务号"是唯一的,即使在不同项目之中也是如此,那么继承属性"项目标识号"将是"任务"实体的非键属性。此时,"项目"实体与"任务"实体之间是非标定型连接联系。

在一个分类联系中,一般实体和分类实体表示现实世界同一事物。因此,所有分类实

体的主关键字都是由一般实体的主关键字通过分类联系继承而得到的。例如：如果“月薪雇员”和“计时雇员”是分类实体，“雇员”是一般实体，那么若“雇员号”属性是“雇员”实体的主关键字，那么它也将是“月薪雇员”和“计时雇员”实体的主关键字。

在某些情况下，一个子实体对同一父实体可以有多种联系。对每个联系而言，父实体的主关键字都将在子实体中作为继承属性。对于子实体的一个给定实例，继承属性的值对每个联系都可以是不相同的，也就是，该实例可以涉及父实体的两个不同实例。例如：一张材料组织清单可用“部件”和“组件－结构”两实体来描述，“部件”实体作为“组件－结构”实体的父实体，这存在两类联系。对于同一部件，它有时是装配某组件的一个构件，那就是，一个部件可以是一个或多个组件中的一个构件；它有时又作为由一些构件所装配成的一个组件，那就是，一个部件可以是一个或多个构件装配成的一个组件。如果，实体“部件”的主关键字是“部件号”，那么，“部件号”将在“组件－结构”实体中作为继承属性而出现两次。当单一属性被继承多次时，每一次继承都应指定一个“作用名”(role name)，如上例中，“构件号”和“组件号”两作用名可以被指定，从而把两次继承的“部件号”属性区别开来。对于只有一次继承联系，并不一定要指定作用名，但也可用作用名来更明确地表达在子实体中的继承属性的含义。

2　外来关键字的语法

外来关键字是通过把继承属性名加“(FK)”标注放到实体盒子中的方法来描述的，如图 2.3.9 所示。如果继承属性属于子实体的主关键字，那么，该继承属性被放在水平线上面并且这个实体应画成圆角盒子，这表示该实体的标识符(主关键字)是根据一个联系而继承得到的。如果继承属性不属于子实体的主关键字，那么，该继承属性应放在水平线下面。继承属性也可以是次关键字的一部分。

作用名与属性名一样，是一个名词短语，一个作用名后加上这个继承属性的名字，作用名和继承属性之间用小圆点分开，请见图 2.3.10 所示。

3　外来关键字的规则

(1) 在确定连接联系或分类联系中的子实体或分类实体时，必须包含一个外来关键字。

(2) 一般实体的主关键字必须被每一个分类实体继承为其主关键字。

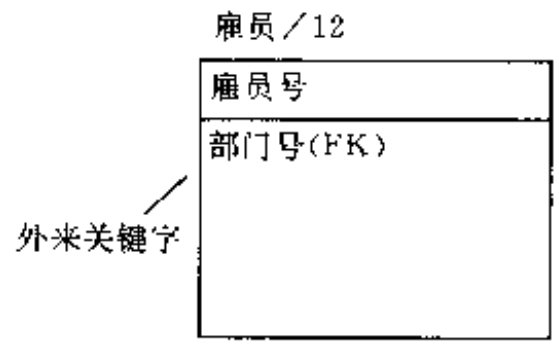
(3) 对于子实体或分类实体的每一个实例而言，子实体或分类实体都不应含有两个完整的外来关键字，这两个外来关键字被用来确定同一父实体或一般实体的同一实例。换句话说，也就是，存在一个联系，只能有一个外来关键字。

(4) 每一个子实体或分类实体的继承属性，都必须是这样一个属性，它是相关的父实体或一般实体的主关键字的一个属性。相反，父实体或一般实体的每一个主关键字属性，都必须是相关的子实体或分类实体中的继承属性。

(5) 分配给继承属性的每一个作用名都必须是唯一的，同时，同一含义必须应用于同一作用名。当然别名可以除外。

(6) 如果在某实体的任一给定实例中，对于两个外来关键字而言，单一继承属性总是具有相同值，那么，该属性可以是多个外来关键字的部分。

继承非键属性的例子：



继承主关键字属性的例子：

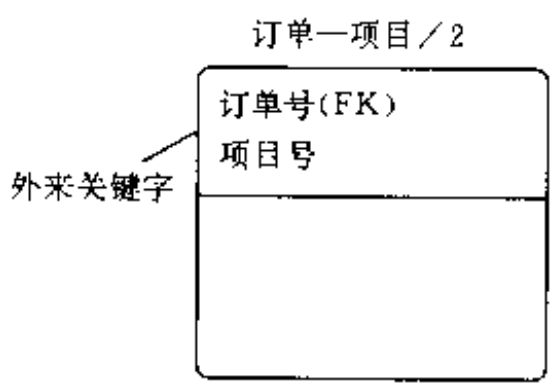
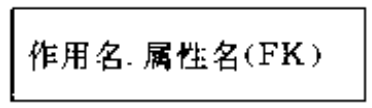


图 2.3.9　外来关键字语法的例子

作用名语法：



例子：

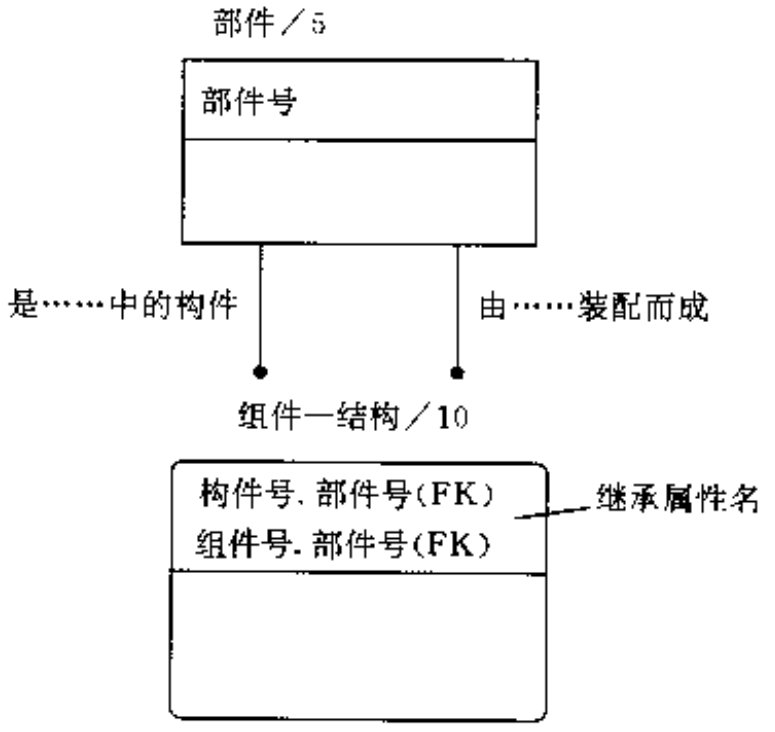


图 2.3.10　作用名语法

第 4 章　建立模型过程

4.1　0 阶段——设计的开始

IDEF1X 数据模型必须描述和定义系统的边界和要求达到的目标。建模者对系统的开发范围有主要的影响，建模者要和项目负责人一起，阐明 0 阶段所要达到的目标，这些目标包括：

(1) 项目说明：一般地叙述曾做了什么，为什么做及将来如何做。

(2) 材料源：获得源材料的计划，包括在编的索引及归档计划。

(3) 作者惯例：惯例(选择方法)的基本说明，作者根据该惯例建立和管理模型。

这些成果以及其他的叙述和解释信息作为 0 阶段的产品。

1　建立模型的目标

工程的目标包括两方面的叙述：

(1) 目标说明：定义模型事务的说明，即：其上下文的限制。

(2) 范围说明：所表示的功能模型的边界说明。

建模目标所建立的结果，主要回答在模型引用期间的主要事务，是一个当前活动的模型(即：当前模型 AS-IS model)还是将来改变后的模型(即：待建模型 TO-BE model)。对于一个 IDEF1X 建模工作的问题范围，其形式描述包括评审、构造、改进或(和)对一个或多个 IDEF0(活动)模型的完善工作。由于这些原因，建模者和项目负责人在某种程度上必须熟知 IDEF0 模型的来源和用途。一个 IDEF0 模型可以作为问题范围的基础。

虽然数据模型的目的是建立一个无偏见的、以底层结构为基础的视图支持整个企业，但重要的是每个模型要有一个给定的范围，该范围帮助标识所关心的数据。这个范围可能和用户的类型、业务功能以及数据类型有关，要把范围说明和目标说明两者结合起来，定义建模目标。

2　开发建模计划

建模计划概述了要完成的任务和这些任务开发的顺序。任务的布局要和建模项目的总任务相一致：

(1) 项目计划

(2) 收集数据

(3) 定义实体

(4) 定义联系

(5) 定义键属性

(6) 定义非键属性组

(7) 确认模型

(8) 评审验收

建模计划是分配、调度任务和预算建模项目开销的基础。

3 组织队伍

衡量模型的价值，不是完全依照某些绝对的规范，而是依赖于该范围专家和设计者的接受能力。该工作通过两种机制完成：(1) 根据开发模型的专家们，进行常规评审提出的，在特定环境下模型确认的度量。(2) 由专家和设计者委员会，周期地评审模型，提出一致的意见。在模型开发期间，各部门所处理的业务方面，可能会发现一些不一致的问题，为了产生一个代表全企业的、并且是一个集中式的数据模型，必须解决这些不一致问题。

建模者应该尽可能对模型所表示的内容负责。不要留下任何假设给模型的读者去想象，也不可能在模型说明的范围之外，再要求读者能够得出什么结论。这就要求建模者仔细地考虑加到模型中的每个数据块，以避免在模型解释过程中存在想象。

为了支持这些基本原则和项目管理的需要，要组成开发机构，IDEF1X 的开发机构有 5 种主要的角色。

(1) 项目负责人

(2) 建模者

(3) 信息源

(4) 课题专家

(5) 评审委员会

分配角色的目的是明确责任，下面要说明每个角色的职能。一个人可能担任几种角色，但要懂得在建立模型时，如果考虑不周，则模型可能代表一个很狭窄的看法，只能局部地达到建模目标。至于项目负责人，必须是起领导作用或作为主要负责人履行这个职责。建模者向项目负责人写报告，由评审委员会批准模型。用此方法可以摆脱建模者、评审委员会和项目负责人之间的冲突。项目负责人位于领导地位，但自动地委托给有资格的参与者进行各种技术讨论和批准手续。

图 2.4.1 说明，项目负责人处于所有活动核心的职能地位。

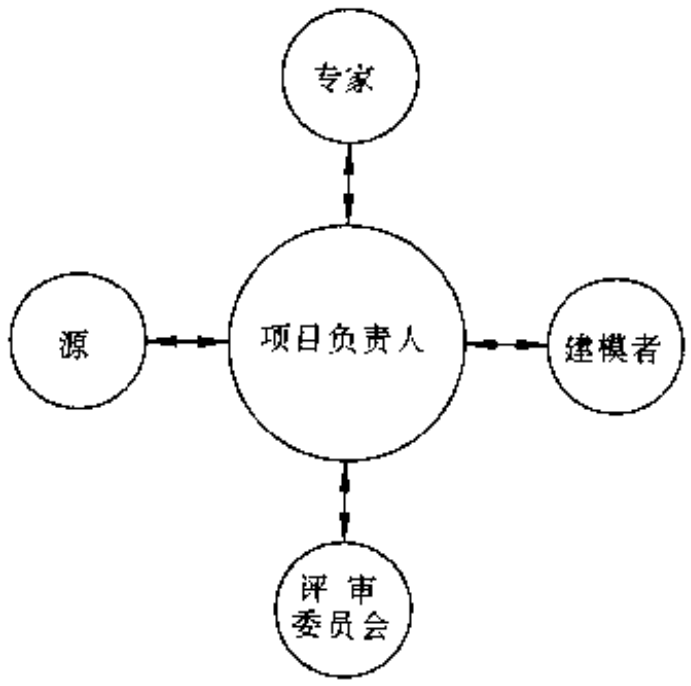


图 2.4.1 队伍组织

(1) 项目负责人的作用

项目负责人是一个能管理指挥整个建模设计的人，他必须完成 4 种基本职能。首先由项目负责人选择建模者，该职能最主要的是项目负责人和建模者对在建模中的基本原则要有一致的意见，其中包括所采用的本方法论，项目负责人对建模者行使的控制范围，和要开发模型的方向和范围。

由项目负责人完成的第 2 种职能是确定信息源，根据这些信息，建模者构造模型。信息来源首先可能是某专业领域内人的知识，谈论或有关的报告。从建模的观点，吸收能够说明和解释信息的人员参加更为理想。其次，因为有记载的信息文件比较容易得到，项目负责人要负责为建模者提供信息源。在 0 阶段，初步地确定信息源，但由于在模型的开发过程中需要的信息可能会变化，因而必须要核对和修改信息清单。

第 3 种职能，项目负责人要选择专家，根据专家的经验和智慧，来确认建模者开发的模型。确认意味着一致，即模型令人满意地反映被模拟的对象，给专家们提供部分模型并请他们审查和评论。当然，专家在所建模型的工作中所花的时间，要比别人在采集信息上的时间多。专家名单要在 0 阶段提出，在整个建模过程中需要时可做必要的审查和调整。

第 4 种职能，项目负责人要组织并召集评审委员会。项目负责人担任委员会主席。委员会定期地开会考虑需要仲裁的问题和评审部分模型。项目负责人不参加投票，而在建模者和委员会之间提供联络。虽然建模者不是委员会的成员，但项目负责人要经常地邀请建模者参加评委会的会议，以便提供背景信息和解释技术难点。委员会的首次会议在 0 阶段举行。此后，由项目负责人斟酌。

(2) 建模者的作用

建模者以可能采集到的原始材料为基础建造模型。建模者的职能是把建模技术应用到项目负责人提出的课题中去。建模者执行 4 种主要职能：即收集原始数据、培训、表示模型和控制模型。建模者是模型构造技术信息和有关模型自身信息两者的交换中心。

建模者开始履行其职能前，要和项目负责人共同研究并制定建模工作的范围。由建模者起草设计计划，即达到规定目标所需的任务。项目负责人把信息源清单和建模者可依靠的专家名单提供给建模者。建模者必须保证与所有的参与者建立必要的通信联络。

首先，建模者根据项目负责人所确定的各种信息源采集数据，这些数据的特性，主要依赖于正要实施的建模阶段。在整个建模工作中，人和文件起到信息源的作用。每个数据的生产者和使用者，对数据有不同的观点，建模者必须特别清楚每一数据源在企业中所代表的特殊观点。建模者要尽可能从源的角度理解数据的语义和结构。每个源提供一种观点，一种获取数据的视图，通过组合这些视图，并比较和对照各种观点，提出以现实为基础的一个映象。每个文件可看作系统的一个微观实现，满足数据模型的规则。建模者要说明所获得的全部规则，用这种方式表示，可读，易理解，并与专家和非专业人员达到一致。

职能 2，建模者对需要建模技术的人员给以帮助。一般可提供 3 种帮助：给评审委员会成员、信息源和某些专家提供一般性的介绍；给信息源和某些专家介绍作为模型读者所需的技能；需要的话，为某些专家和建模者介绍建模技术。

职能 3，表示模型，建模者用文字和图形描述方法表示数据模型。5.3 节介绍其标准形式。

职能 4，建模者也要控制模型的开发，维护由源得到的信息文件，以便为项目负责人作出决策提供适当的后备，并允许记录参与状况。这些记载为建模者指出了预期要达到的范围和程度。通过了解在什么领域中，谁曾提供了信息和信息之间相互活动的特点，建模者可以估计现行的建模工作是否已有效地满足了原有目标。

建模者要负责定期地把模型的内容组织到一些读者组表中去。一个读者组表是关于模型的一组信息，便于组织评审和从信息专家收集评论。组表的讨论见 5.2 节。

(3) 信息源的作用

一个 IDEF1X 模型信息源来自企业各部门。这些源常常是指具有特别管理知识或某种业务处理能力的人。其与模型的接触限于很少几分钟的访问时间，然而这些源形成建模过程的核心。他们的贡献被用于建模中，他们的理解力为建模者构造一个有效的而又有用的模型，提供了必要的见解，无论哪里，必须设法找到通常认为是最好的信息源。

项目负责人根据建模者的需求，确定有效的信息源。随着建模工作的进展，必然要改变和修订源清单。建模者必须仔细地考虑每个源所提供的信息，建模者和源都应该意识到，任何特定的事件都存在着倾向性。每个源对世界的认识总是有些不同的，建模者的责任是要从中理出那些不同的观点，使源文件真实地反映现实。

文件记载着企业某部分某时刻的状态。然而文件上的信息总是按使用者方便整理的，并很少直接反映其基础的数据结构，最常见的例子是数据的冗余，而在文件上发现数据的不一致性，也是常见的混乱来源。对模型来说，文件是有效的信息源，但他们需要大量的解释、理解和实际使用的证实。

如果正在开发的数据库综合或代替了现有的数据库，那么现有数据库的设计，应该参照为一个源文件。和其他的文件一样，现有数据库的设计一般不反映基础的数据结构而需要解释。

另一方面，作为源的人可在他们直接使用的信息基础上，告诉建模者那些信息是如何得到、解释和使用的。通过适当的提问，可以协助建模者理解源和源之间相关的概念。

(4) 专家的作用

专家由项目负责人任命。他具有要建模的制造领域中某方面的专门知识，他的专长能对开发模型作出关键性的评论。但专家对建模工作的影响不能过分强调。建模者和项目负责人应该严格的考虑选择每一个专家。要求专家对开发的模型做关键性的评审，这通过相当数目反复验证的实践和采用读者组表完成，这些组表将有关情节的信息集提供给专家，以这种方式，专家得到了融会贯通形式的信息，同时也接到了填写空白或是将整个内容补充完整的要求。虽然组表很大程度基于建模者对信息源的信息解释，专家的注释也是为优化模型提供高质量的源材料。专家们独特的经验使他们有唯一的资格协助建模者构造和优化模型。建模者必须抓住每一个机会取得这样的帮助，这也是信息组表必须简要地介绍给专家的原因。显然这也和建模工作要解决的问题有关系。

专家的主要工作是确认模型，专家确认是专家们达到一致意见的主要手段，这就是说，一个确认的模型是一个得到专家一致承认的模型。要注意的是，对一个确认模型没有必要验证它是“正确”的。如果在这个领域，绝大多数专家一致认为模型确切地并完全地代表了所涉及的范围，那么就认为模型被确认。不同的意见总是被记下，直至别的方法证明

模型是无效的。这就是专家参与建模工作的重要原因。当建模者首次构造一部分模型的时候，他会说“我已经审查并得出了下面的……结论”，当这部分提供专家评审的时候，他问“我是对的吗”？在修正专家们有异议的那部分模型时，要认真考虑专家们的评论，以寻求一致。

与任何非建模参与者相比，专家更需要有效的培训。事实上，建模者的责任之一是保证专家对建模方法学和过程有适当的了解。专家们尤其需要好的模型阅读技巧。对专家进行阅读模型的入门训练是有益的，通过向专家提供建模的基本理解，就可以从专家那里得到更有用的信息。随着建模过程不断深化，逐步少量地把建模方法学介绍给专家。这样可以增加专家对建模工作的理解能力和贡献。

(5) 验收评审委员会的作用

评审委员会由建模工作所涉及领域中的专家和博学多闻的人组成。项目负责人组织该委员会并担任主席，评审委员会的职能是对建模工作提出指导和仲裁，工作的最终成果是对一个 IDEF1X 数据模型作最后鉴定。由于该模型是一系列决定和实现系统化改善企业生产率问题这种复杂事件的一部分，因而评审委员会要广泛地代表数据的供应者，加工者和最终用户。通常这意味着方针的制定者和数据处理专家要参与委员会。这些关系到数据模型的最终使用。进一步吸收事务范围外但和研究领域有关的专家参加，可能是有益的。这些专家常能为数据模型提出宝贵的见解，这样数据模型会受到其他领域从事的工作的影响。

经常可以看到，有些人既作为专家又作为评委会的成员。一般说来，这不会有利益的冲突。一个专家容易受到模型各阶段的局限，而评委会则要作出对整个模型的评价。作为源的个人，参加委员会的情况是很少见的，通常，由于他们知识的限制很难对委员会有实际的帮助。最后一点，建模者参与评委会，一般说，不是一种理想的建议。因为严重的利益冲突非常明显。再则，建模者的责任是无偏见的表示模型，而评委会的责任是要确保模型真正代表他们特定的企业。

项目定义阶段的最后成果，是项目负责人所作的具体的任命报告，以满足建模技术所需的每一种职能角色。

4 收集源材料

建模者面临的首要问题是决定要采集什么种类的源材料和应该从哪些源采集信息。通常以 IDEF0 功能模型的分析为基础，确定 IDEF1X 模型的范围和上下文。一旦完成了功能分析及功能之间的传送途径，由功能模型所表示的企业内的目标功能就可以确定。一个目标功能节点，是使用中信息的一种集中表示，是问题范围的表示。

一旦已确定了目标的功能范围和主要感兴趣的信息类，就可以选择功能中的每一个体参与数据的采集过程。数据的采集可以用几种方法完成，包括同有知识的个人交谈，观察活动，评价文件、策略和过程，应用指定信息模型等。这需要把目标功能结点翻译成为它们的等价的或起作用的参与建模的成分，一旦确定了参与到目标功能中的这些成分，项目负责人可以着手确定能够作模型源材料的个体或指定的可观察的范围。

源材料可以呈各种形式，并广泛地分布在整个机构中，源材料可以包括：

- 调研的结果

- 观察的结果
- 策略和生产过程
- 原系统的输出(报表和屏幕)
- 原系统的输入(数据的输入形式和屏幕)
- 原系统数据库和文件的说明

这里不考虑使用的方法,建模者的任务是为收集反映有关模型的目标和观点的信息,做一个采集信息文件计划。一旦收集了,该文件的每一块都要有能够追溯它的源的标记。这个文件,和建模的过程中发现的追加文件一起,为模型开发的各阶段所参照。建模者要学习和研究那些源材料,这些材料为模型的基本机构特征及数据所表达的意义,提供了可信度。

在前面第 2 章,讨论过数据建模的目标,是要定义一个数据资源单一的一致的企业视图,在 ANSI/SPARC 结构中称作概念模式。绝大部分源文件不论代表外部模式还是内部模式,都必须被映射到概念模式,但它们总是偏重于特殊的用途。例如,用户报告是数据的一个外部模式,可以作为源文件,文件描述和数据库设计表示内部数据视图,也可用作源文件。通过建模过程可以大大地简化数据结构,而数据模型又可以映射回外部和内部模式。

为成功地实现目标,一个正确的数据采集计划是最重要的。计划要说明,数据收集中哪类数据是重要的,在哪里有用和谁能提供。

5 采用作者约定(惯例)

作者约定是授权建模者(作者)帮助开发模型、评审组表以及其他表示的自由范围。共同的目的是为了加强材料的表示。它们也用于促进对模型的任何部分有较好的理解和了解。例如,一个标准命名的约定,可为实体名和属性名所采用。

作者约定可以采用各种形式并出现在各种地方,但最重要的问题是要弄清作者约定不该是什么:

- 作者约定不是技术的形式化延伸。
- 作者约定不该违反技术。

作者约定是为特殊服务的需要所开发的,每个约定必须提供报告,并包括在 0 阶段文件中,在评审时分发。

4.2 1 阶段——定义实体

1 阶段的目标是标识和定义在建模的问题范围中的实体,这个过程的第一步是标识实体。

1 标识实体

在 IDEF1X 模型上下文中,一个"实体"(ENTITY)表示一组事物,它们与数据相联系。每一事物可能是一个物体,一种物理的物质,一个事件,一种状态,一种行为,一种思想,一种概念,一个点或一个地方等。由实体所表示的集合的成员,有共同的属性集或特征集,例如所有职工集的全部成员,有一个职工号、姓名和其他一些共同的属性。一个实体集

的个体成员，称作为实体集的一个实例。如职工号为 789、名为张三的职工是实体“雇员”(EMPLOYEE)的一个实例。实体总是用单数的通用名词命名，并必须有一个属性(键)唯一地标识每一个实例。

可以从 0 阶段收集的源材料中，直接地或间接地标识绝大部分实体。如果建模工作是扩充和改进以前的数据模型，则应该从以前的模型中选择合适的实体。对于事先没有定义实体的情况，建模者必须首先在源材料“名字表”中标识事物，他潜在地表示了可行的实体。一种简化的方法是标识表中所有名词，例如名词：零件，运输工具，机器，图等，在这一阶段中，考虑为潜在的可行的实体。另一种方法是标识具有以“号”或“号码”结尾的名词。例如：零件号，订购单号，路线号等。在该阶段，具有“号”和“号码”之前的短语或词可作为实体处理。对于表中剩余的项，建模者必须问清楚哪些字是表示一个对象或事物，或是关于实物和事件的信息，这些属于事物类的项也是可行的实体。

实体由基本实体实例综合而成，也就是若干实体实例，其所有的特征是同类型的，通常表示为一个实体。此概念的实例在图 2.4.2 中说明。实体每一个实例是实体的一个成员，每个成员具有同一类型的标识信息。

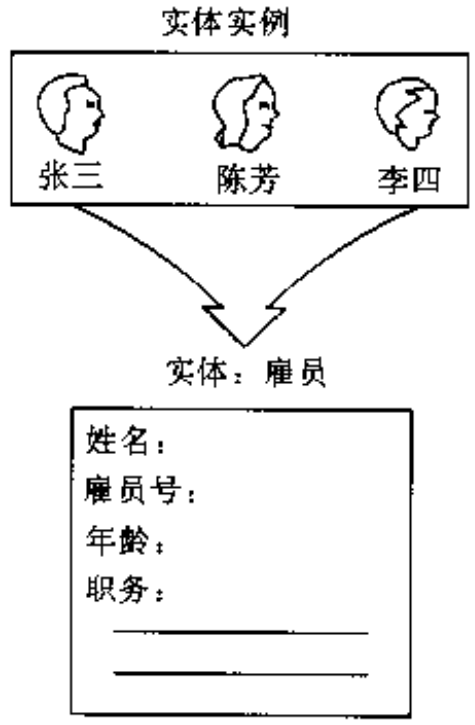


图 2.4.2 综合成一个实体

为帮助区分非实体和实体，建模者对于每个候选实体应提出下列问题：

- 它能被描述吗？（它有性质吗？）
- 有 n 个这类的实例吗？
- 每个实例可被区分或标识吗？
- 它属于或描述某物吗？（回答“是”则隐含是一个属性而不是实体）

该分析结束时，建模者已定义了最初的实体，即包含了此时模型上下文中的所有实体名。

建模者在建实体的时候，把一些不连续的标识分配给每个实体并记下它所参照的源。这样可保持信息的追溯及其完整性，维护和管理也相对容易。实体的实例在图 2.4.3 中说明(给出英文，表示存入数据库时需要)。

直至 4 阶段结束时，表中的大部分名字会作为实体保留。除此，在建模过程中逐步增进对信息理解，从而增加一些新的实体成为信息模型的一部分。

在以后阶段中发现的实体名，也必须加到实体中并分配一个唯一的标识号。阶段 1 工作的成果之一是实体，它必须保持可行的数据。

2 定义实体

阶段 1 工作要形成的第二个成果是早期实体词汇表。阶段 1 中的词汇表只是实体定义集。实体定义的内容包括：

(1) 实体名

实体名是一个在 IDEF1X 模型中用作识别实体的唯一名字，是描述性的。实体名允许缩写和简写，但必须有意义。

number	entity name	source material log number 1
序　号	实体名	源材料存档号
E　1	backorder(欠料单)	2
E—2	bill of loading(提货单)	2
E　3	carrier(承运人)	2
E—4	clock card(考勤卡)	3
E—5	commodity(货物)	2
E　6	contractor(承包商)	4
E—7	delivery(交货)	2
E　8	department(部门)	2
E　9	deviation waiver(偏差弃权说明)	6
E　10	deviation waiver request(偏差弃权请求)	6
E—11	division(分部)	4
E—12	employee(雇员)	10
E—13	employee assignment(雇员分配)	10
E—14	employee skill(雇员技能)	10
E—15	end item requirement(最终产品需求)	6
E—16	group(组)	6
E—17	inspection tag(检验标签)	12
E—18	inventory adjustment(库存调整)	6
E—19	invoice(发票)	11
E—20	issue from stock(从库存发料)	12
E—21	job card(调度卡)	12
E—22	labor report(工时报告)	12
E　23	machine queue(加工排队)	14
E　24	master schedule(主调度)	14
E　25	material(物料)	14
E—26	material availabilty(物料可用性)	15
E　27	material handling equipment(物料搬运设备)	15
E—28	material inventory(物料库存)	15
E　29	material move authorization(物料移动权限)	15
E　30	material requirement(物料需求)	15
E—31	material requisition(请料单)	15
E　32	material requisition item(物料申请单项)	15

图 2.4.3　实体的实例

(2) 定义实体

这是一种在企业中常用的定义法，不是用于字典的。因为在模型中信息的含义是针对阶段 0 中模型的一种具体的观点和所定义的上下文的。处于 0 阶段范围之外的定义是无意义的(如果不完全混乱的话)。即使在定义实体的方法上稍有不同的含义，也主要是要以模型内容范围为基础。一旦出现一些定义(这些定义，就反映模型最普遍的观点来说，不是必不可少的)应先记下来，由评审员鉴别什么样的定义应该和这些标识实体的项相联系。

阶段 1 的定义过程，是一种迫使普遍地接受定义的进化机制。

(3) 实体同义词

这是一张关于实体的另一名字表，该表所遵循的规则是，与实体名有关系的定义也必须完全精确地适用于同义词表中的每个同义词。

经初步分析工作后，很容易组织和定义实体。在短期内词汇量可能会骤增，此后建模者可以进行研究，实体池中其余名字的全部定义。

4.3 2 阶段——定义联系

2 阶段的目标是标识和定义实体之间的基本联系，在该阶段中，有些联系可能是非确定的并需要在以后的阶段中进一步改善。2阶段的主要结果是：

- 联系矩阵
- 联系定义
- 实体级图

1 标识相关的实体

一个“联系”可以被简单地定义为两个实体之间的一种关联或连接。确切地说，这叫做二元联系。IDEF1X 被限制到二元联系，是因为它们比 n 元联系更容易定义和理解。二元联系也有一种简明的图形表示法。其缺点是必然难于表达 n 元联系，但不会丧失这种能力。因为任何 n 元联系都可以用 n 个二元联系来表示。

一个联系实例是两个实例之间有意义的关联或连接。例如，实体“操作员”的一个实例，其名字是张军，号码是 862。被分配于实体“机器”的实例，其类型是钻床，机器号码是 12678。一个 IDEF1X 联系，表示两个确定的实体之间具有同类联系实例的集合。尽管相同的两个实体也可以有多种联系类型。

IDEF1X 模型的目标，不是要描述所有可能的联系，而是通过存在的依赖性(父-子)联系定义实体间相互的联系，也就是定义父实体和子实体间的一种联系。在这种联系中，父实体的每个实例与 0 个、1 个或 n 个子实体的实例相联系，而子实体的每个实例精确地与一个父实体的实例相联系，也就是说子实体是依赖于父实体而存在的。例如一个买主发出了0，1 或 n 张购货单，而一张购货单是由一个买主发出的。

如果父实体和子实体表示同一客观世界的实体，那么父实体是一个一般实体。子实体是个分类实体，每一个分类实体实例，总有一个一般实例。一般实体的每一个实例，可以有 0 个或 1 个分类实体实例。例如一个领薪水的雇员是一个雇员。而一个雇员可能是、也可能不是领薪水的雇员，几个分类实体可在一个分类联系中与一般实体相联系。但对于一个给定的一般实体实例，只能取一种分类。例如，一个分类联系可用来表示一个雇用人员的事实；一个雇员，可以是一个月工资的雇员，或是一个计时工资的雇员，但不可能是两者都是。

在模型的开发初期，不可能将全部联系表示成父-子联系或分类联系。因而在阶段2中，允许定义非确定的联系，非确定联系的一般形式是(n:m)，即 0，1 或 n 对 0，1 或 m。两者不相互依赖。

2 阶段的第一步，是标识各种实体成员之间的联系，这一任务需要开发如图 2.4.4 所示的联系矩阵。一个联系矩阵是一个有水平轴和垂直轴的二维数组，沿一个轴记录一组预定因子，沿另一轴记录另一组预定因子(在两种情况下，都为所有实体)，"X"被放在两轴的交叉点，用来表示两个实体之间可能的联系。在这儿，联系的性质并不重要，联系存在的事实是充分的。

实体-联系矩阵	buyer 采购员	requester 申请者	approver 审批人	purchase requisition 采购申请	purchase req item 采购申请项
buyer 采购员		x		x	
requester 申请者	x			x	
approver 审批人				x	
purchase requisition 采购申请	x	x	x		x
purchase req item 采购申请项				x	

图 2.4.4　实体 - 联系矩阵

新建模者的一般意图是说明实体之间的联系，但要记住，最终的目的是按父-子联系定义模型，避免标识间接的联系。例如，一个部门负责一个或多个项目而每个项目引入一个或多个项目任务，就没有必要定义部门和项目任务之间的联系。因为所有的项目任务和一个项目有关。许多有经验的建模者，可能是宁愿画实体级图，而不愿构造联系矩阵；然而，在标识它们的时候，定义联系是很重要的。

2　定义联系

2 阶段的第 2 步是定义已标识的联系，这些定义包括：

- 表示依赖
- 联系名
- 关于联系的说明

作为联系定义的结果，可能会删去一些联系及增加新的联系。为了建立依赖，实体间的联系必须在两个方向检验，通过在联系的每端决定完成这一工作的基数。为了决定基数，假设实体的一个实例存在，然后决定，相对于第一个实体来说，第二个实体存在多少确

定的实例，然后调换实行，重复这一分析工作。

例如，考虑实体“班级”和“学生”之间的联系。一个学生个体，可以说明在0，1或多个班级中（注：指选取不同的课程）；从另一个方向分析，一个班级个体（课堂），可以有 0，1 或多个学生。因此在班级和学生之间存在多对多的联系，在联系的每一端都有基数 0，1 或 n（注：这种关系是非确定的，因为没有“精确的1”的基数存在于联系的任何一端）。这种非确定的联系，必须在以后的建模过程中消除。

另一实例，实体“买主”和“购货单”之间的联系。一个买主可以发出 0，1 或多张购货单，但一张购货单总是由一个买主发出而不可能由多个买主发出。因此在买主和购货单之间存在1对多的联系，在联系的买主端有基数1，而在联系的购货单端有基数 0，1 或 n（注：这是确定性的联系，因为在联系买主端存在一个真正“1”的基数。即，在这个确定型联系中，买主是购货单的父实体）。

一旦建立了联系依赖性，建模者可着手对联系命名和定义。联系名用简单的动词短语表示。这个短语反映了联系所表示的意思，通常联系名是单个动词，然而副词和介词也经常作为联系名，联系名选好后，建模者可以读联系，为两个实体的联系产生一个有意义的定义和描述语句。

在确定型联系的情况下，总存在一个父实体和一个子实体；联系名先从父实体端说明，然后说明子实体到父实体。如果实体间存在分类联系，则说明两个实体参照现实世界的同一客体，并且子实体端的基数总是0或1，这种情况，由于名字可以是隐含的，联系名可被忽略。如，“雇员”(employee)可以是一个“月薪雇员”(a salaried employee)。

在非确定型联系的情况下，有两个联系名，每个实体有一个联系名用“/”符号分隔，联系名从顶向下或从左向右说明，取决于实体在图中的相对位置，然后反之。

联系名必须有意义，必然要存在它们所关联的实体的全部含义。事实上，建模者选择一个具体的联系名的原理，可能是联系定义的一个文字说明，应用于定义实体的定义规则，也适用于联系定义。

- 联系定义必须是具体的
- 联系定义必须是简明的
- 联系定义必须是有意义的

例如，在操作者和工作站这两个实体之间，如果定义联系是 1 对 0 或 1，则该联系名可能读作“当前分配到”，该联系可由下列的定义支持：“在任意工作中，每个操作员可能被分配到某些工作站。但这个联系在此刻，反映了操作员分配到某一个工作站。”

3 构造实体级图

在定义联系的时候，为了图形化描述联系，建模者可以着手构造实体级图，实体级图如图 2.4.5 所例示。在该阶段，所有的实体用方角盒子表示，并允许非确定的联系，实体级图的范围和数目可以变化，决定于模型大小和个别评审的着眼点。如果可行，用单个图描绘全部实体和它们之间的联系，有助于建立上下文和确保一致性。若产生多个图，建模者一定要注意图形之间的一致性以及实体和联系定义。一个综合实体级图将描述所有定义的关系。

当实体级图集中于某一个实体（可称为“主题实体”）的特殊情况时，简称为“实体图”，

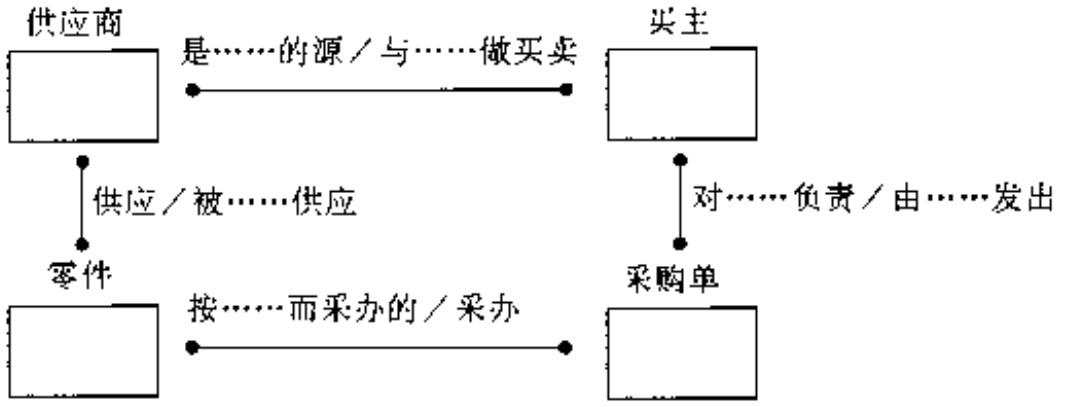


图 2.4.5　实体级图

如图 2.4.6 所示。对每一个实体来说，实体图的生成是可选的，若要使用，必须服从下列准则：

（1）主题实体总是放在页的中心位置。

（2）父实体或一般实体应放在主题实体上方。

（3）子实体或分类实体应放在主题实体的下方。

（4）非确定的联系常放在主题实体盒子的侧面。

（5）联系线从主题实体盒子射向有关的实体，图中所示的联系仅仅是主题实体和有关的实体间的联系。

（6）每个联系有一个标签；对于非确定的联系，该线有2个标签用“/“分隔。

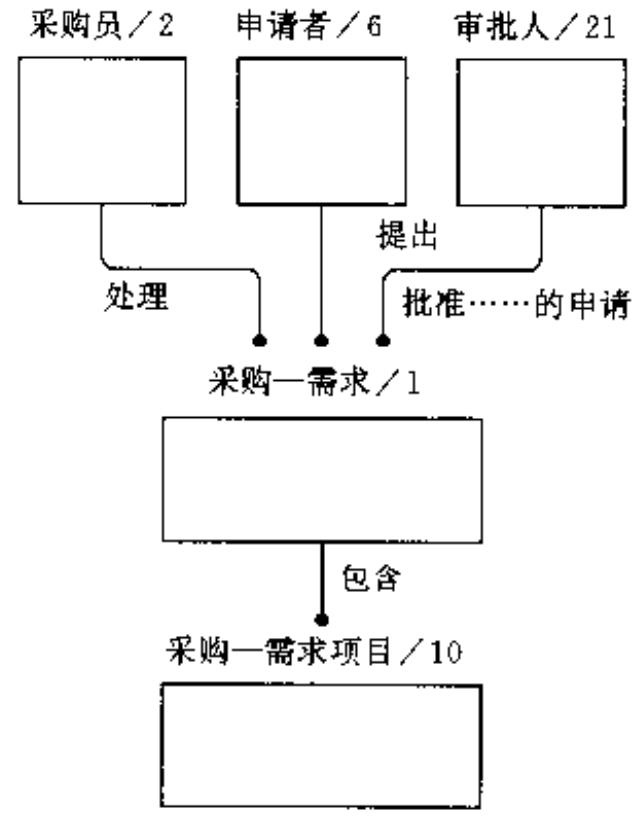


图 2.4.6　实体图

此时，对每个实体现有的信息包括如下：

（1）实体定义；

（2）联系名和任选的定义(对父-子联系)；

（3）在一个或多个实体级图中描述。

建模者的深入分析，可通过追加参照图来扩充有关的实体信息。参照图(用来说明的图，有时叫做 FEO)是一个可选项，建模者在图上可采用自己的建模习惯。这些图是建模者和评审员之间讨论问题的场地。它们为建模者在说明原理、讨论问题、分析比较方案和观察模型进展的各方面，提供了一种极好的工具，图 2.4.7 例示了一张参照图。该图描述了联系选择的方案比较，并记载了建模者的选择。

另一种参照图如图 2.4.8 所示。该图描绘了建模者面临的问题，在此例中，建模者为了引起评审员的注意标出了问题及其复杂性。到此阶段，建模者已收集了充足的信息，着手通过组文件和遍历过程开始正式确认(参见 5.2 节和 5.4 节)。

用于说明不同方案的“参照图”

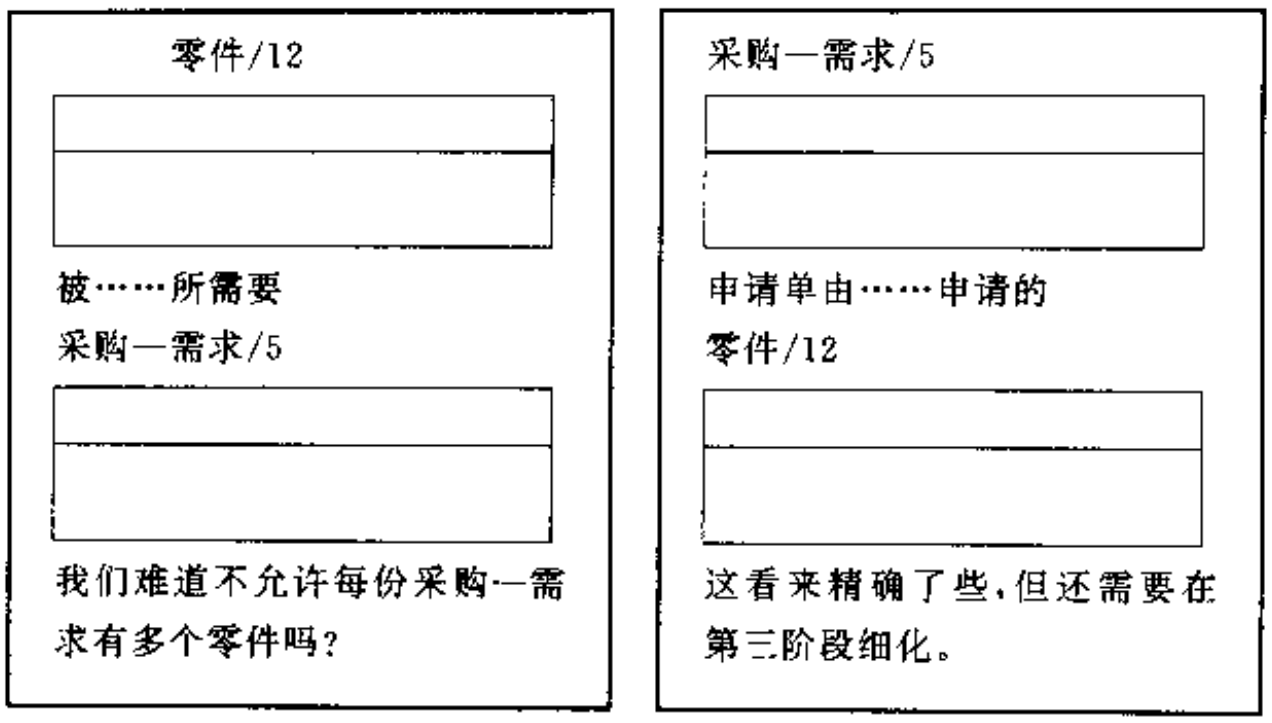


图 2.4.7　参照图(FEO)

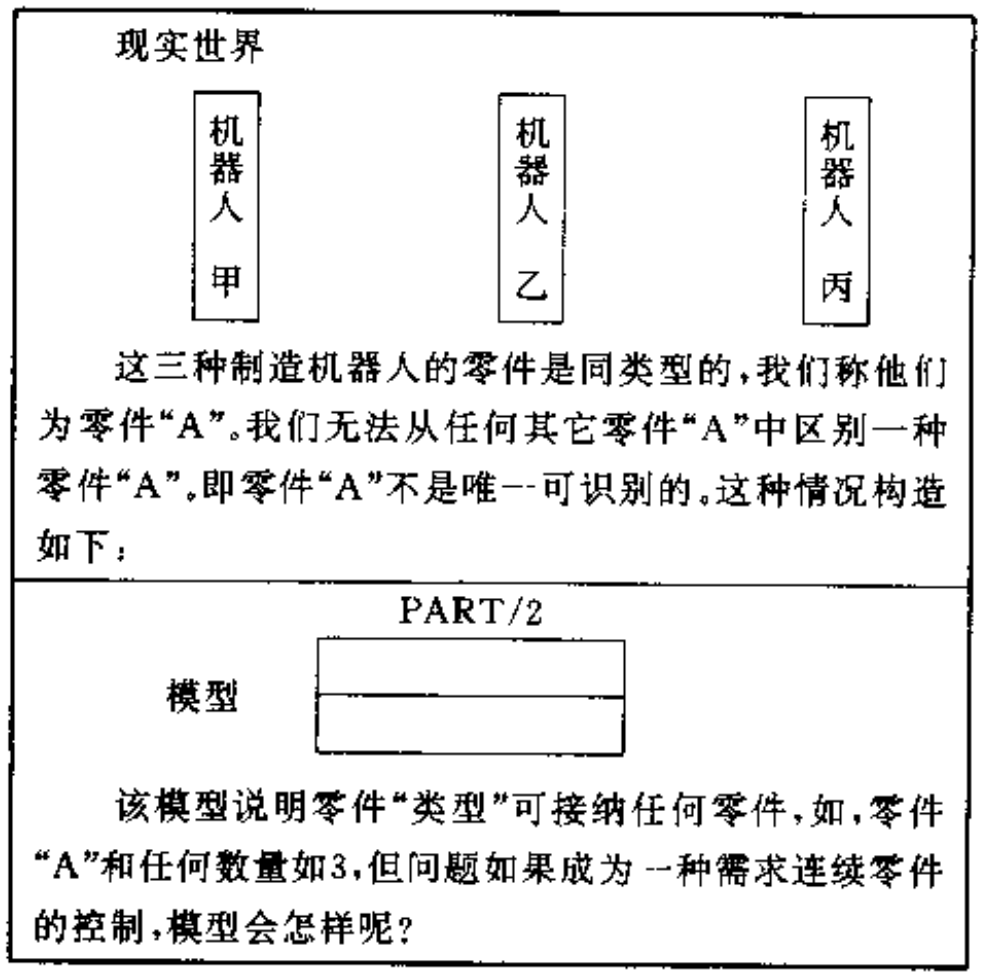


图 2.4.8　例子参照图

4.4　3 阶段——定义键

3 阶段的目标是:

- 改善阶段2产生的非确定联系;
- 为每个实体定义键属性;
- 迁移主键以建立外来键;
- 确认联系和键。

3 阶段的结果在3 阶段图(键级图)中描述，另外增加键属性的定义。3 阶段将扩充和改善实体和联系的定义。

1 分解不确定联系

此阶段的第一步是保证2阶段产生的所有不确定关系全部得到改善。3 阶段要求只能采用确定联系形式；或是确定的连接联系(父-子)或分类联系。为满足这一要求，建模者使用改善替换技术，改善替换图分为两部分：一部分是主题(要改善的不确定关系)，一部分是改善替换部分。消除多对多联系的改善替换图例，见图 2.4.9。

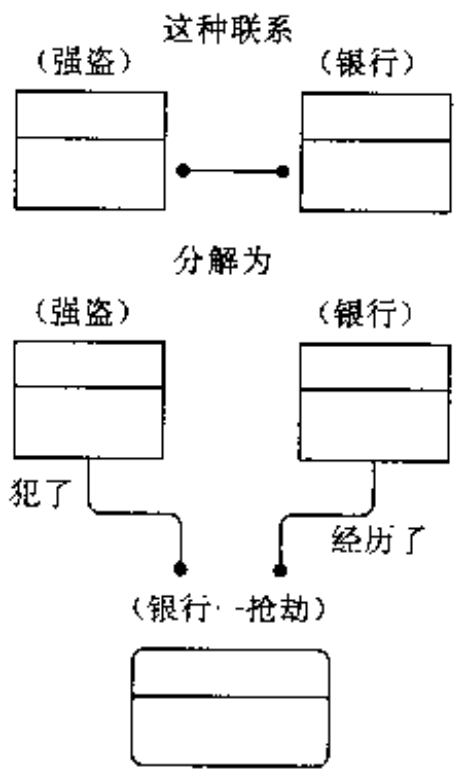


图 2.4.9 不确定联系改进的例子

改善联系的过程，是转化或变换每一个不确定的联系为两个确定联系的过程。通过该过程演变出新的实体，图 2.4.9 中所示的不确定的关系表示一个强盗可以抢劫多个银行，而一个银行可以被多个强盗抢劫。然而，在我们为解决不确定联系而引入第 3 个实体“银行-抢劫”前，我们不能标识哪个抢劫者抢劫哪个银行这一事实，实体“银行-抢劫”的每个实例，与一个银行和一个抢劫者有完全确切的联系。

前面我们一直在处理自然实体，即每一个实体，我们都可以从原始数据清单或源材料文件中找到根据。一个自然实体会包含类似于下列的名字：①购货单，②雇员，③买主。

而第三阶段我们要着手了解联结实体或交叉实体。交叉实体用来消除不确定的联系。通常交叉实体代表事物的有序对，这些事物有相同的基本特征(唯一的标识符，属性等)，像自然实体一样。在前面的例子中，尽管“银行-抢劫”可以认为是自然实体，而它实际上代表抢劫者与银行的对，自然实体和交叉实体之间之微小的差别在于实体的名字，自然实体典型的实体名是单数普通名词，而交叉实体名可能是一个复合名词。

实质上，交叉实体更抽象。交叉实体，通常是首次用于第3阶段的那些实体确认规则应用的产物，这些规则中最重要的规则是，改善所有不确定的联系，这一改善过程在确定综合的数据结构方面，是很重要的步骤。

这一改善过程包含以下基本步骤：

(1) 对每个不确定的联系提出一种和多种改进替换。

(2) 由建模者择优选择，该选择要反映在3阶段的模型中。

(3) 修改 1 阶段的信息，以便能包含由改善过程所产生的新实体。

(4) 修改 2 阶段的信息，以便定义与新实体有关的联系。

2 描绘功能视图

此时数据模型的规模和复杂程度多半是可估计的。在 1 阶段，独立于其他实体评估每一实体是很自然的，因为那时实体是简单的词的定义。在 2 阶段中，由于实体的总量和联系一般不太大，在一个图中画所有的联系也可能曾尝试过。而在 3 阶段，反映在模型中的大量实体和联系的复杂性，在正常情况下，一个个人不再能为数据模型的含义构造一个全面的映象，因此模型可从多种观点评审和确认，这些观点使模型评估更直接与要模拟企

业的功能方面相联系，这些观点可由功能视图表示(但含义与 IDEF0不同)，每个功能视图在一张图上表示，其目的是建立有限制的上下文，使模型的这些部分可在一次会上评估。

功能视图在模型的评估和确认中可作为一种工具，建模者要注意决定或选择一个功能视图所说明的题目，下面是常用的两种方法。

(1) 选择样本源材料作为功能视图的题目，如购货单。

(2) 将功能视图与任务类别或指定的处理联系，由组织部门或在 0 阶段作为源标识的功能域表示。

如图 2.4.10，样本功能视图中的数据，用来重建采购单或重构一些采购单报表。在构造功能视图的时候，作者要考虑题目，使其确切地表达其内容。

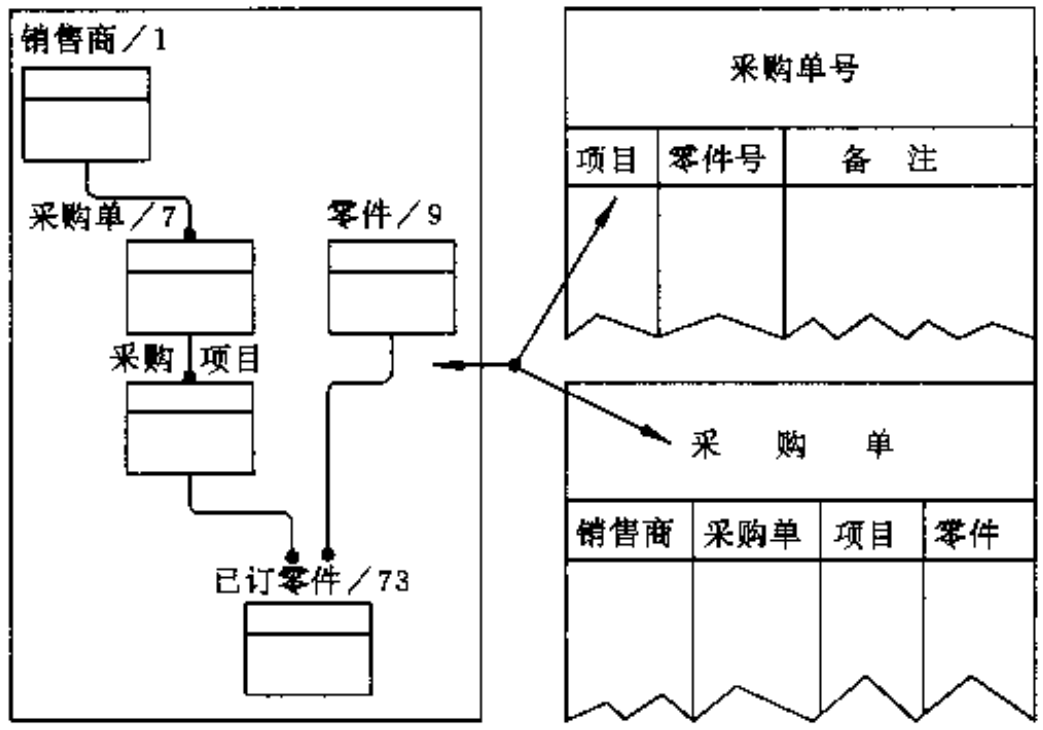


图 2.4.10　功能视图的范围

3　标识键属性

IDEF1X 方法学第 3 阶段，涉及到关于实体实例的数据元素的标识和定义，这些数据元素称为候选键、主键、次键和外来键。该步的目的是标识属性的值，以能够唯一识别每个实体实例。

在此，强调术语属性实例和属性定义的含义是很重要的。一个属性实例是一个实体实例的一种特性或性质，由一个名字和一个值组成，也就是说，一个属性实例是一个特定实例的信息元素。属性实例都是些描述词，其性质类似于形容词。

图 2.4.11 中的例子说明了某些属性实例以及相应的实体实例。注意由工号“1”标识第一个实体实例或个体。与该实体实例相联系的姓名是“王军”。其职务是“操作员”。这些属性合在一起，就唯一地描述了那个实体实例，并区别于其他类似的实体实例。每个属性实例有一个类型和一个值，属性实例的唯一组合描述一个特定的实体实例。

属性表示一个同类型的属性实例的集合，它们适用于同一实体的全部实体实例。通常属性名用单数描述性的名词来表示，在雇员实体实例中，有下列几个属性：

· 雇员号

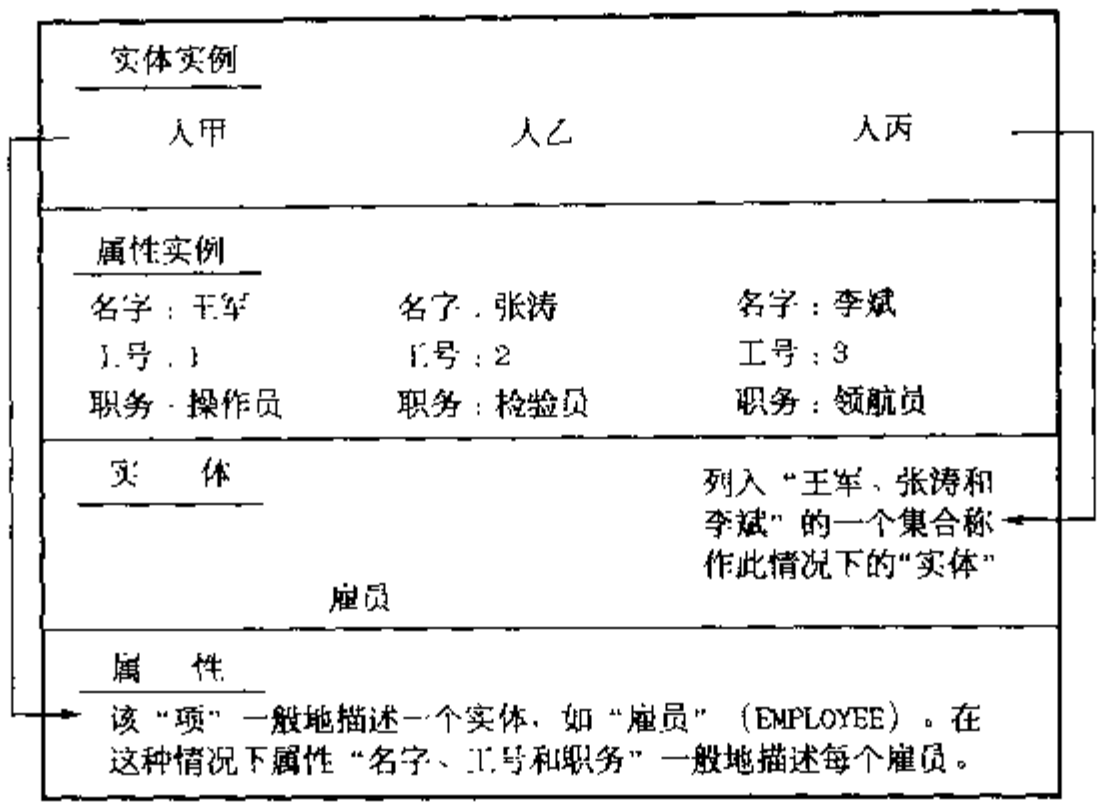


图 2.4.11 属性例子

· 雇员名

· 雇员的工作/职务

图 2.4.11 也说明了属性实例如何表示为属性，属性实例属于实体实例，而属性本身属于实体，因而实体和属性之间建立了所有权的联系。

一个属性只有一个所有者，就是实体，属性起始于实体。在实例中，雇员号属性的所有者是雇员实体。虽然属性只有一个所有者，但所有者可以和其他的实体共享属性。这一事情在以后的阶段中进一步讨论。

一个属性，代表了我们使用一个属性实例，来描述一个确定的实体实例的具体特性。某些属性用来帮助唯一的标识一个具体的实体实例，通常这些属性称为键属性。3阶段集中在模型的上下文中标识键属性。在4阶段中，将标识和定义非键属性。

一个或多个键属性组成一个实体的候选键，一个候选键定义为能唯一识别实体每个实例的一个或多个键属性。属性“雇员号”，就是作为一个实体候选键的例子，每个雇员通过雇员号区别于其他所有的雇员；因而属性“雇员号”被作为一个实体候选键的例子，该键唯一地识别雇员实体中的每个成员。

某些实体有多组属性可被用来区别不同的实体实例，如雇员实体，有“雇员号”、“社会安全号”两个属性，两者都是候选键。被选在键迁移时使用的候选键，就称为主键。另一些键叫做次键，若一个实体只有一个候选键，它自动地作为主键。因此每个实体有一个主键，还可有一些次键。这两种类型的键都可以用来唯一地识别实体实例，但只有主键能用于键的迁移。

在模型图中，通过在主题实体盒子内画一条水平线，主键表示在该线上面的框中。如果主键中存在多个属性（如：项目编号和任务编号，需要用这两者共同标识项目任务），则它们都放在线的上面。若一个实体有一个次键，该键分配了唯一的次键编号，在图中该编号写在每个属性之后的括弧中。这些属性是次键的一部分。若一个属性属于多个次键，则

每个编号都写在括弧中。若一个属性同时属于一个次键和主键，则该属性写在水平线的上面，后跟次键号。若它不属于主键，则写在水平线的下面，各种形式的键在图 2.4.12 中加以说明。

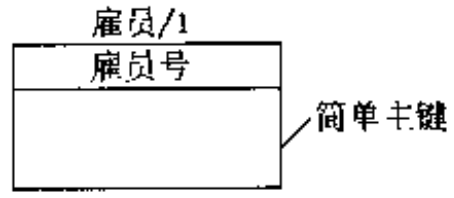


图 2.4.12　键的形式

标识键的过程包括：

（1）为一个实体标识候选键。

（2）为一个实体选一个键作为主键。

由于一些候选键可能是迁移的结果，因而键标识是一个迭代的过程，由在任何联系中不是子实体和分类实体的实体开始，通常这些实体的候选键比较明显。键迁移也从这些实体开始，因为它们不包括任何外来键。

4　迁移键

键的迁移是把一个实体的主键复制到其他有关实体中的过程，此时复制键称为外来键。第2个实体每个实例外来键的值，完全等于第1个实体相关实例中主键的值，这说明了其他实体如何共享属于一个实体的属性。下面3条规则控制着键迁移：

（1）在一个联系中，迁移总是从父实体或一般实体移向子实体或分类实体。

（2）由实体对所共享的每个联系来说，整个主键（即：全部属性是主键的成员）必须一次迁移。

（3）决不迁移次键和非键属性。

外来键中的每个属性与父实体或一般实体主键中的一个属性相匹配。在一个分类联系中，分类实体的主键必须与一般实体的主键相同。而在其他联系中，外来键属性可以是于实体主键的一部分，也可能不是。由于外来键属性是父实体中属性的反映，它不属于出现外来键的那些实体，因而一个实体中的每个属性，或属于该实体，或是属于该实体的一个外来键。

在模型图中，外来键像次键一样标识，其表示法是在外来键的属性后面加上(FK)的标记。如果该属性也是主键，则写在水平线上而；否则写在水平线的下面。

如果一个于实体的主键包括了全部外来键属性，则该子实体称为“标识从属于”父实体，该联系叫做一个“标定联系”。如果外来键的任何属性不是子实体的主键，则子实体不依赖于父实体，也叫做“非标定联系”。在3阶段和4阶段的图中，可标定联系用实线表示，非标定联系用虚线表示。

在一个或多个可标定联系中的子实体叫做“标识从属实体”，仅在非标定联系中的子实体（或不是任何联系中的子实体）称为“标识独立实体”。在3阶段和4阶段的图中，标识独立实体用方角矩形盒子表示，而从属实体用圆角矩形盒子表示。一个从父实体到子实体属性迁移键的例子如图 2.4.13 所示。

该例中，“顾客号”属性（“顾客”实体的主键）移至“支票”实体（作为外来键），在“支票”实体中，该键被当作主键的一个成员使用，并和支票的另一个属性“支票号”一起形成支票实体的主键。

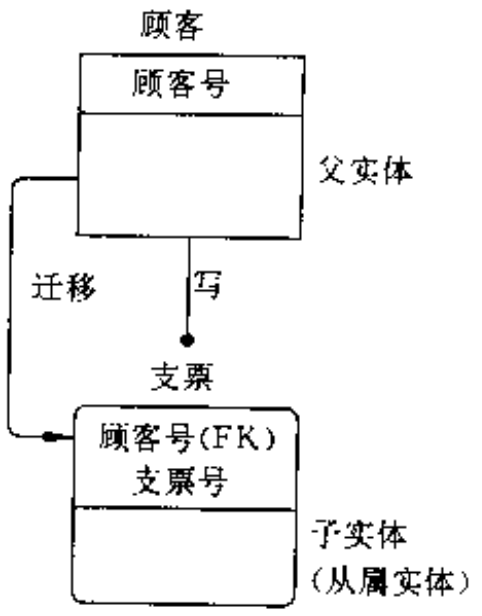


图 2.4.13　键迁移到从属实体

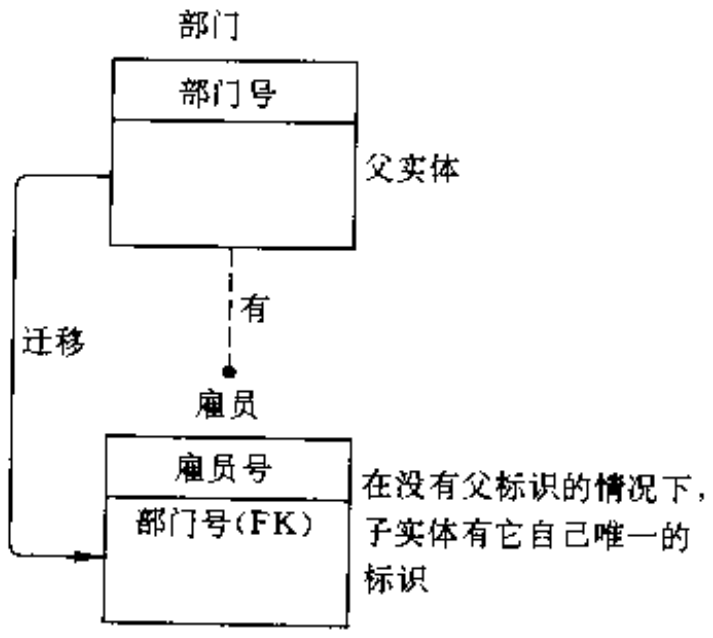
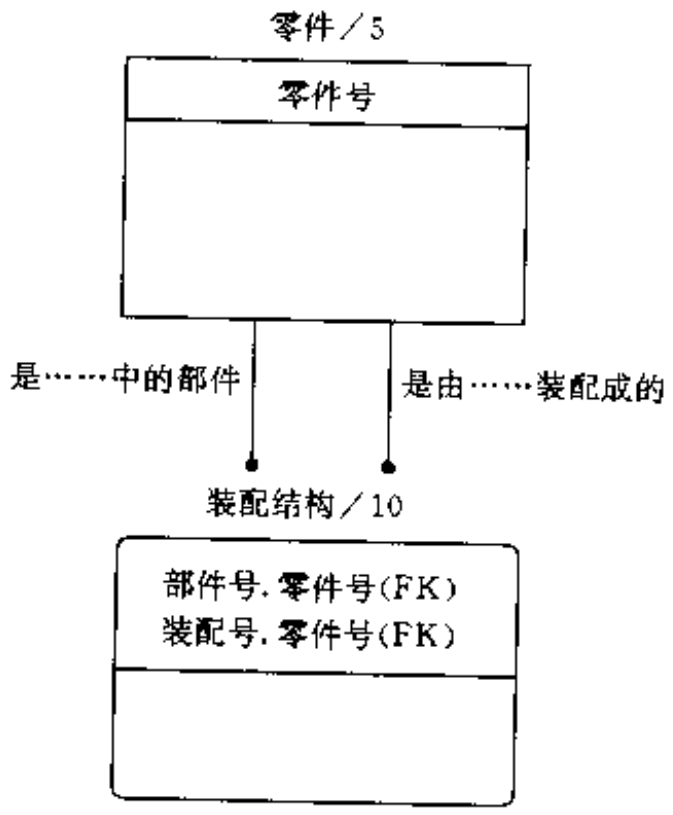


图 2.4.14　键迁移到标识独立实体

从一个标识独立实体移到另一个标识独立实体，键属性的迁移例子见图 2.4.14。在这个例中，部门号属性移到了雇员实体，而雇员实体的主键是雇员标识号。因而部门号作为一个外来键放在键属性线的下面。由于该联系是非标定联系，所以该联系线以虚线表示。

同一属性在同一子实体中可以生成多个外来键，当属性迁移由两个或多个联系到达子实体时，一般出现这种情况。在某些情况，对于每个外来键中的属性，每个子实体实例必须有相同的值，也就是，该属性在实体中只出现一次，并标识为一个外来键。在另一些情况，一个子实体实例对每个外来键，可能(或必须)有不同的键值。在这种情况下，属性在实体中出现多次，就有必要区别它们的出现，为了做到这一点，通常给每个键值分配一个作用名。通过该作用名区分它们，实例见图 2.4.15。



每个迁移键“零件号”可以用另外的“作用名”定义它在子实体中的功能，作用名用一个句点和外来键相区别。

图 2.4.15　属性作用名

5　确认键和联系

控制键的迁移和标识的基本规则如下：

(1) 禁止使用不确定型联系语法。

(2) 键从父(或一般)实体移向子(或分类)实体是强制的。

(3) 对于一个给定的实体实例，禁止使用多值属性(不重复规则)。

(4) 在一个实体实例中，禁止使用具有空值(即没有值)的属性(非空规则)。

(5) 有复合键的实体，不能划分成具有更简单键的多个实体(最小键规则)。

(6) 两个实体间双联系路径需要断言。

前 2 个规则我们已在前节中讨论过了，在此我们讨论后面几条规则。

图 2.4.16 说明了“不重复规则”的应用，要注意图的主题说明，“采购单号”和“采购单项目号”是“采购单”实体主键的成员，而使用采购单项目号是说明一个采购单(实体实例)可以有多个采购单项目号，为了在数据模型中描述这一情况，我们建立了一个新的实体，称为“采购单项目”。此时，采购单和采购单-项目间联系的真实特征则开始出现。

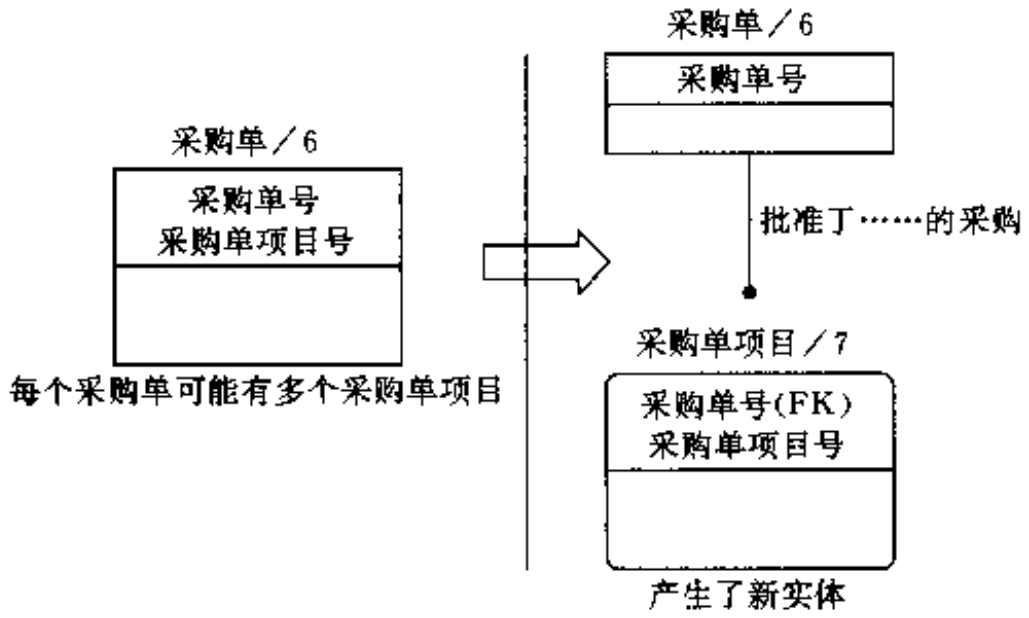


图 2.4.16　改善不重复规则

图 2.4.17 说明“非空规则”的应用，注意“零件号”已经移到了“采购单项目”中，从而建立了采购单项目和零件的联系。所示的图断言：每个采购单项目真正同一个零件号相联。但通过调查(或评审员评论)我们发现，不是所有的采购单项目都与零件有联系，有些可能与服务或其他一些没有零件号的物品(如行政办公用品)相联系，这就禁止零件号直接迁移到采购单项目实体，而需要建立一个新的实体，在我们的例子中称为“订购的零件”。

一旦建立了新实体，其键迁移也要符合迁移键的规则，建模者要再次用非空和不重复规则确认实体-联系结构。

也应该考查每个复合键，以确保它们符合最小键规则，该规则要求在不丢失某些信息的情况下，不存在带复合键的实体，划分成两个或多个具有更简单的键的实体。该规则是关系理论的第 4 和第 5 范式的组合和扩充，其他规范化规则，如完全函数依赖和非传递依赖，要到 4 阶段非键属性引入到模型后才可使用。

在 2 阶段曾提到指定冗余联系的倾向。而 2 阶段的分析，主要靠建模者的判断；随着

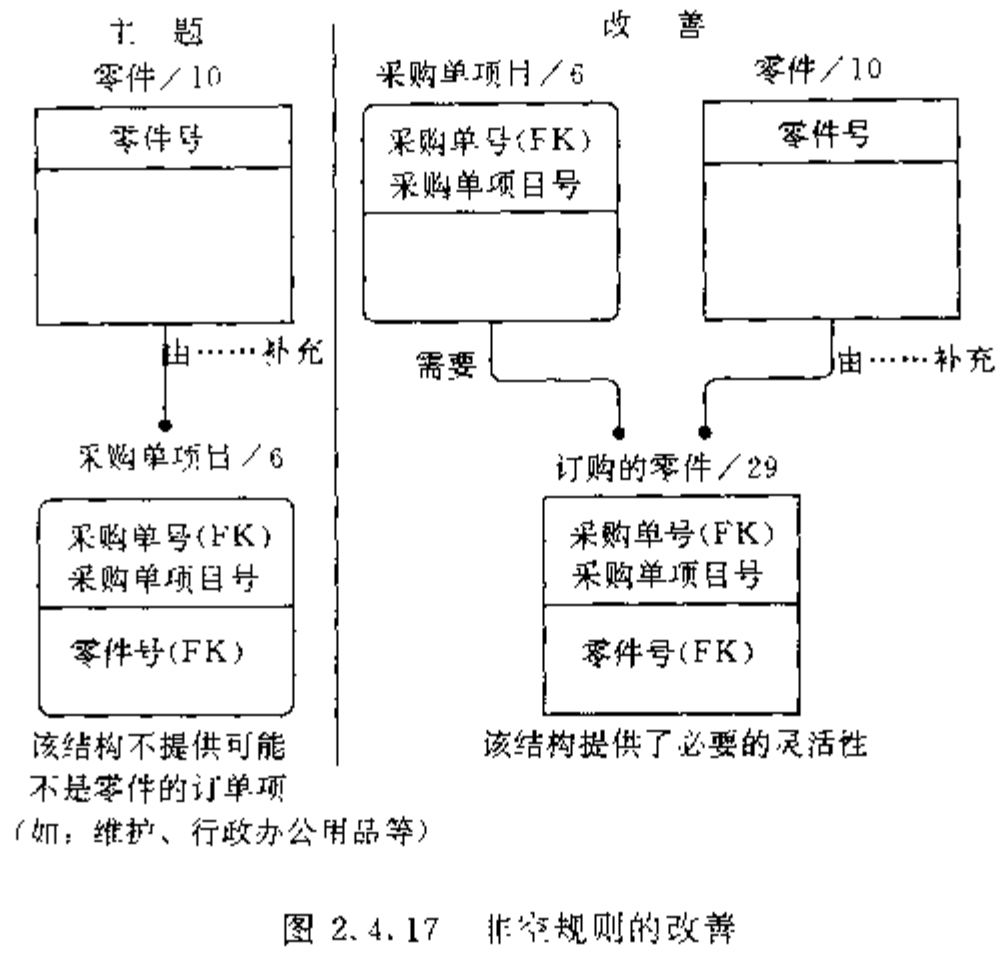


图 2.4.17　非空规则的改善

键的建立，建模者现在可更严格地分析。无论什么时候存在双路径联系，也就是存在一个子实体具有两个联系，其最终通过一个或多个联系回到公共的“根”父实体，存在这样的双路径时，需要定义该路径是相等、不相等还是不确定的路径断言。如果对于子实体的每个实例，两个联系路径始终指向同一个根父实体实例，那么这两条路径是相等的；如果对于子实体的每个实例，两个联系路径始终指向不同的根父实体实例，则两条路径不相等；如果它们对一些子实体实例是相等的，而对另一些不相等，则该路径是不确定的；如果路径之一仅由一个单联系组成，并且两条路径相等，那么那条单联系路径是冗余的，应删除。

双路径联系最简单的情况，是每条路径由单一联系组成，见图 2.4.15 例；由于每个零件使用实例，可能和两个不同的零件实例相联系，因而不存在冗余。在这种情况，路径断言要求路径是不等的。因为一个零件不会组合成它自身。

如果路径之一由多个联系组成，而另一路径由单一联系组成，这种结构称为三元组。三元组例示于图 2.4.18。在这种情况下，雇员直接地或间接地通过部门与单位相联系。如果断言是，一个雇员所属单位和该雇员的部门所在单位相同（即相等），那么单位和雇员间的联系是冗余的，应该移走。要注意的是，事实上，如果我们曾断言，有些、而不是全部雇员可能属于每个不同的单位，为了满足对单位号应用非空规则，则要增加另一实体譬如叫做“借出的雇员”实体，“单位号”成了“雇员”的外来码。

断言也适用于双路径联系，两路径包含多个联系，如图 2.4.19，图中的部门和“任务分配”之间，存在两种联系路径。如果一个雇员只被分配到一个由他的部门管理的项目，则路径是相等的。如果一个雇员被分配一个不是他所在的部门管理的项目，则该路径是不相等的。若一个雇员所分配的项目不考虑其部门，则路径是不确定的。如果没有路径定义说明，一般认为是不确定的路径。断言应该作为注解附于 3 阶段的图中，同时包含在子实体定义中。

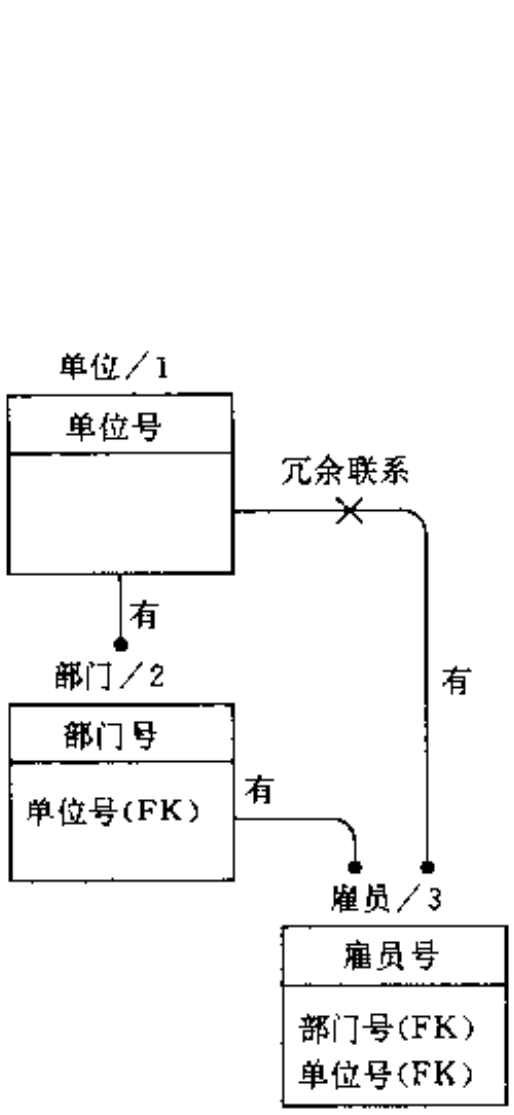


图 2.4.18　三元组例子

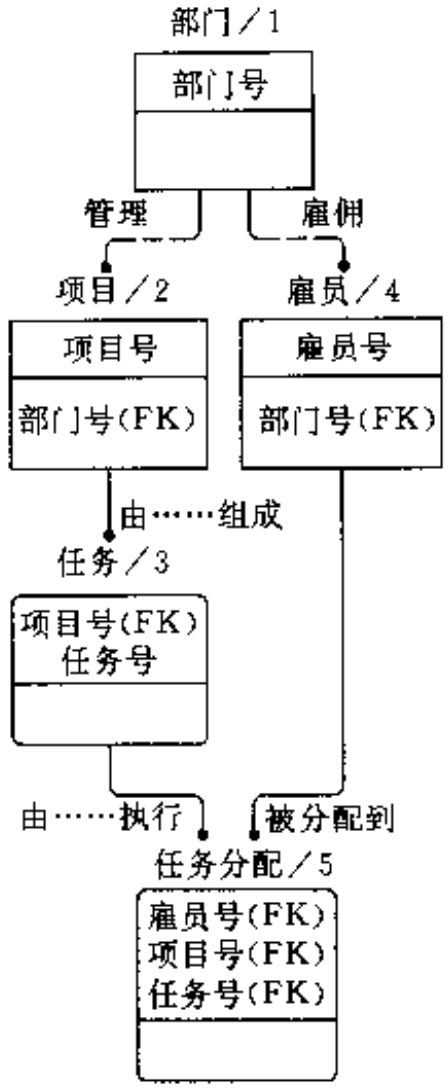


图 2.4.19　路径断言

当标识主键成员时，把实体项放到一个属性池中，可用一个实体/属性矩阵来标识整个模型中属性的分布及其用途，该矩阵有如下特征：

(1) 所有的实体名写在侧面。

(2) 所有的属性名写在顶部。

(3) 实体所用的属性在相邻的矢量区中描述，用下列的代码表示：

“O”＝所有者

“K”＝主键

“I”＝继承

图 2.4.20 说明了实体/属性矩阵的一个实例，这种矩阵是维护模型的连续性的一种主要工具。

6　定义键属性

一旦模型标识了键，就该定义作为键使用的属性，3 阶段讨论的确定键属性准则，也适用于这些定义，它们必须是精确的、具体的、完全的和全面可理解的。

属性定义总是和拥有属性的实体相联系，即，它们总是所有者实体文件集中的成员。因此，标识那些被每个实体所占有、并用作那个实体的主键或次键的属性，是件简单的事。图 2.4.20 所示的例子中，那些属性在实体属性矩阵中用代码“OK”表示。属性定义由下列各项组成：

· 属性名；

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
		Entity 实 体	Payment Terms 支付项	Pattern Number 式样号	Resale Code 再销售码	Taxable Code 应征税码	Q.C.Att Code 质控辅助设备码	Purchase Req Issue 采购申请提出日期	Purchase Order No 采购单号	Buyer Name 采购员名	Ship Via 运送经由	Department Code 部门码	Requester Name 申请者名	Extra Copy Code 额外副本码	Confirmation Name 批准名	Confirmation Code 批准码	Vender Address 销售商地址	Vender Name 销售商名	Ship to Location 运往地点	Change Number 更改号	Order Code 订单码	Vendor Code 销售商码	Buyer Code 采购员码	Purchase Req No 采购申请号
1	Purchase Requisition 采购申请	OK	I																				
2	Buyer 采购员		OK																				
3	Vendor 销售商			OK																			
4	Purchase Order 采购单		I	I													OK						
6	Requester 申请人												OK										
9	Part 零件																						
10	Purchase Req Item 采购申请项	IK																					
12	Purchase Req Line 采购申请线	IK																					
21	App over 审批人		IK																				
22	Part Source 零件源																						

图 2.4.20 实体·属性矩阵

· 属性定义；

· 属性同义词。

7　3 阶段结果的描述

作为键标识和键迁移的结果，功能视图现在可以被修改以反映和改进联系。3 阶段的功能视图也应该描绘：

· 主键、次键和外来键属性；

· 标识独立实体(方角)，标识从属实体(圆角)；

· 标定联系(实线)，非标定联系(虚线)。

3 阶段功能视图的实例如图 2.4.21 所示。许多在 3 阶段的分析中产生的信息，可以在实体中作出报告(说明)。每个实体文件集包括：

· 实体定义；

· 主键、次键和外来键属性清单；

· 拥有键属性的定义；

· 由一般实体组成的联系的清单；

· 由分类实体组成的联系的清单；

· 由父实体组成的标定联系清单；

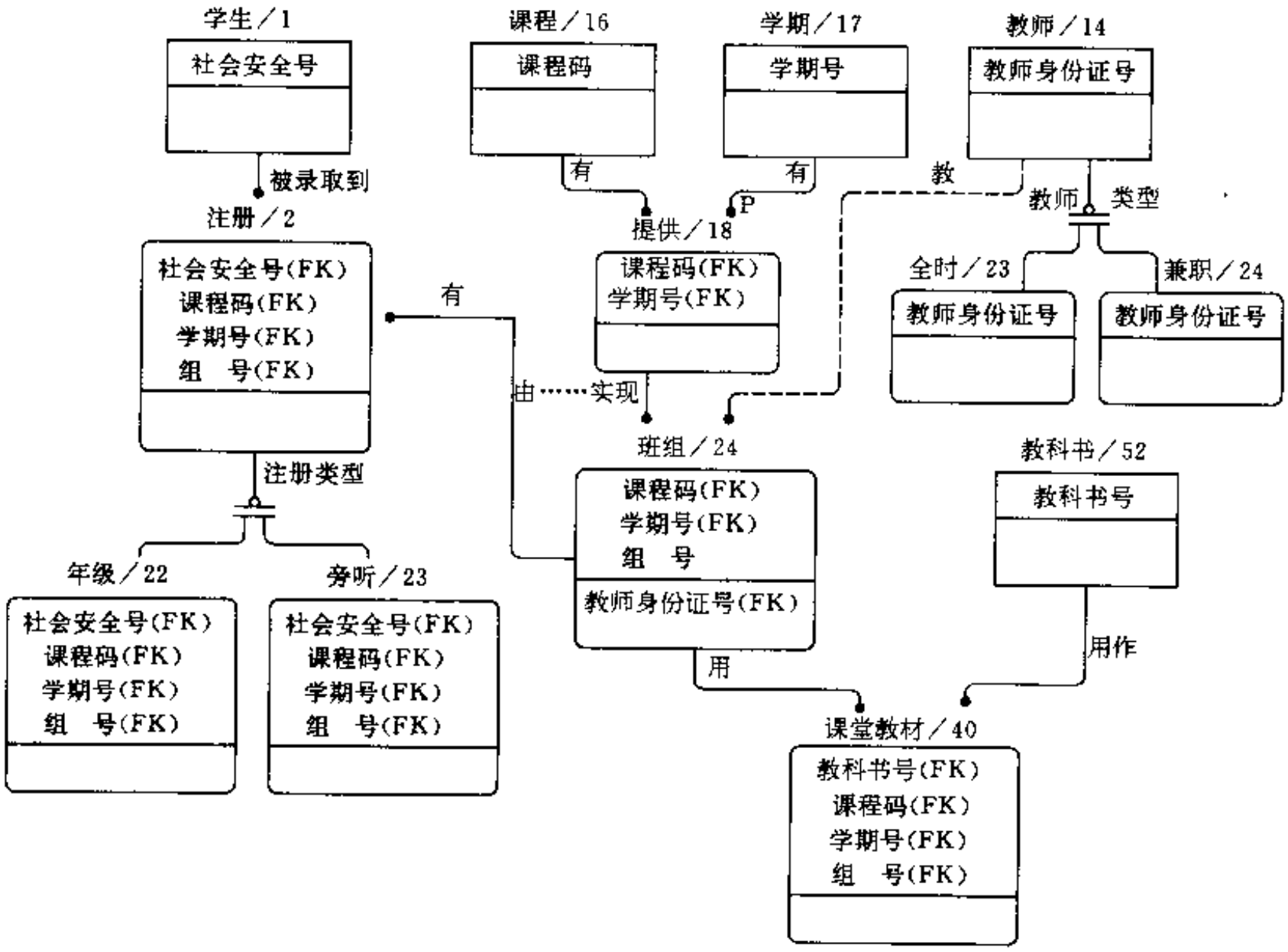


图 2.4.21　3阶段功能视图的这例

· 由子实体组成的标定联系清单；
· 由父实体组成的非标定联系清单；
· 由子实体组成的非标定联系清单；
· 双路径断言定义(如果合适)。

对于实体，建模者也可以有选择地按 2 阶段中任选实体图的方法构造单个实体图。

除了联系定义的列表清单外，补充一份联系实体的交叉参照表是有用的，在 3 阶段的报告中也有交叉参照拥有和共享属性。

4.5 4 阶段——定义属性

4 阶段是模型开发的最后阶段，这阶段工作的主要目标是：

· 开发属性池；
· 建立属性的所有者关系；
· 定义非键属性；
· 确认并改进数据结构。

4 阶段的结果可在一个或多个"属性级"图中描述，在4阶段结束时，我们得到完全改进的数据模型(该模型符合关系理论的第 5 范式)。模型由一个完整的对实体、属性(键和非键属性)和联系的定义集和交叉参照集支持。

1 标识非键属性

随着键的标识，属性池的构造在3阶段就开始了。4 阶段的第一步是扩展属性池以包含非键属性。属性池用来收集那些有生命力的属性名，在属性池中，每个名字只能出现一次，并为其分配一个唯一的标识号。

构造属性池的过程，实质上类似于构造实体池的过程。在 1 阶段中，我们曾从 0 阶段的源数据清单中，选取似乎是实体的名词形成了实体池，现在我们还要回到源数据清单，并抽取那些说明性的名词，这类名词(用来描述物件)一般代表属性。属性池的实例在图 2.4.22 中所示。

在 0 阶段源数据表上的许多名字，作为潜在的实体进入了实体池，而在 3 阶段所确认的没有资格作为实体而被剔除的那些名词，很可能是属性；此外许多首次在表中没有被选的名词，可能也是属性。这份表以及在 1 阶段、2 阶段中得到的知识一起，作为建立属性池的基础。属性池在模型的上下文中是一个潜在可行的属性清单，该清单多半比实体池大得多。

属性池是模型中使用的属性名源，即使在建模工作的后期所发现的属性名，也要追加到属性池中，并分配一个唯一的标识号，然后在模型中使用它们。

2 建立属性的所有权

下一步需要为每个非键属性分配一个所有者实体，绝大部分属性的所有者较为明显，例如建模者很容易把"销售商名"与"销售商"实体相联系，但有些属性在确定它们的拥有实体时，建模者可能会遇到困难。

number 序号	attribute name 属性名	source data number 源数据号
1	purchase requisition number(采购申请号)	1
2	buyer code(采购员码)	2
3	vendor name(销售商名)	3
4	order code(订单码)	4
5	change number(更改号)	5
6	ship to location(运往地点)	6
7	vendor name(销售商名)	8
8	vendor address(销售商地址)	8
9	configuration code(配置码)	9
10	configurer's name(配置员名)	9
11	extra copy code (额外副本码)	10
12	requester name(申请者名)	11,42
13	department code(部门码)	12
14	ship via(运送经由)	13
15	buyer name(采购员名)	14
16	purchase order number(采购单号)	15
17	purchase requisition issue date(采购申请提出日期)	16
18	quality control approval code(质量控制检验码)	17
19	taxable code(应征税码)	19
20	resale code(再销售码)	20
21	pattern number(式样号)	21
22	payment terms(支付项)	22
23	freight on board delivery location(船上货物交货地点)	18
24	purchase requisition item number(采购申请项号)	23
25	quantity ordered(所订数量)	24
26	quantity unit measure(数量单位度量)	25
27	part number(零件号)	26
28	part des cription(零件描述)	27
29	unit price(单价)	28
30	price unit of measure(价格计量单位)	29
31	purchase requisition line code(采购申请路线码)	31
32	requested delivery date(要求的发货日期)	32
33	requested delivery quantity(要求的发货量)	33
34	commodity code(货物码)	34

图 2.4.22　属性池举例

在建模者不能确定一个属性的所有者实体时，他可参照那些抽取该属性的源材料，这对决定属性的所有者有帮助。0 阶段建的原始数据总是属性池的基础，源数据表为建模者指出了所要的属性值在源材料表中的位置，通过对属性在源材料中用途的分析，建模者在数据模型中更容易确定其所有者实体。建模者也要注意，决定属性所有者的控制因素，是反映在源材料中的属性值所代表的属性实例。当把每个属性分配到它的所有者实体时，应

记录分配情况。

3　定义属性

在4阶段中所标识的每个属性，必须形成定义。在数据模型中使用的其他定义原则，特别是3阶段使用的原则在这里也适用。开发的定义必须是精确的、具体的、完整的和完全可理解的。这些属性所产生的定义和3阶段中属性定义的格式相同。属性定义包括：

- 属性名
- 属性定义
- 属性同义词或属性别名

在 IDEF1X 模型中“同一名—— 同一意思规则”适用于实体和属性，因而每个属性必须有唯 一的名字。建模者也可采用标准的属性命名方法，一般采用用户可认识的或自然语言名，属性名也必须严格满足编程语言规则。

在属性定义中，建模者也希望标识属性的格式。如字母-数字码、文本、钱、日期等，可接受值的范围也可由表定义，如星期 一、星期二、星期三、星期四、星期五，或一个大于零而小于10 的范围定义，包含多属性的断言，也可在定义中说明。例如：当雇员-职务码少于 20 时，雇员-工资必须大于20 000元。

4　改善模型

建模者现在要着手 4 阶段联系的改善工作，3 阶段所使用的基本规则也适用于此。在 3 阶段介绍的非空和不重复规则的应用，现在可用于键和非键属性，由此建模者可能期望寻求某些新的实体。一旦标识了这些实体，必须应用和 3 阶段完全相同的键迁移规则。

在 4 阶段中，非空和不重复规则的应用，与以前唯一不同的是它们主要用于非键属性。图 2.4.23 的实例说明了对非键属性的非空规则应用。图 2.4.24 的实例说明对非键属性不重复规则的应用。对于违反改善规则的属性，直接建立新实体的一种选择方法，是在发现时对违反者作标记，以后再建立新实体。在属性图中，在属性名后面的括号中，对非空规则的违反者填上“N”(非空规则)，对不重复规则的违反者填上“R”(不重复规则)。

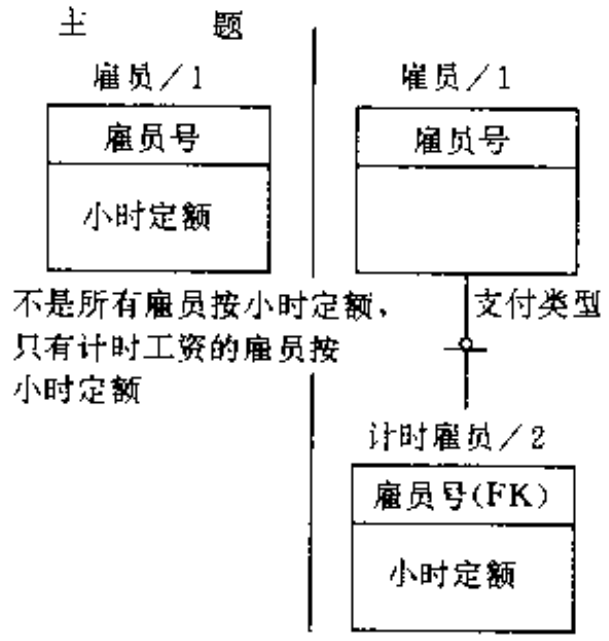


图 2.4.23　4 阶段应用非空规则

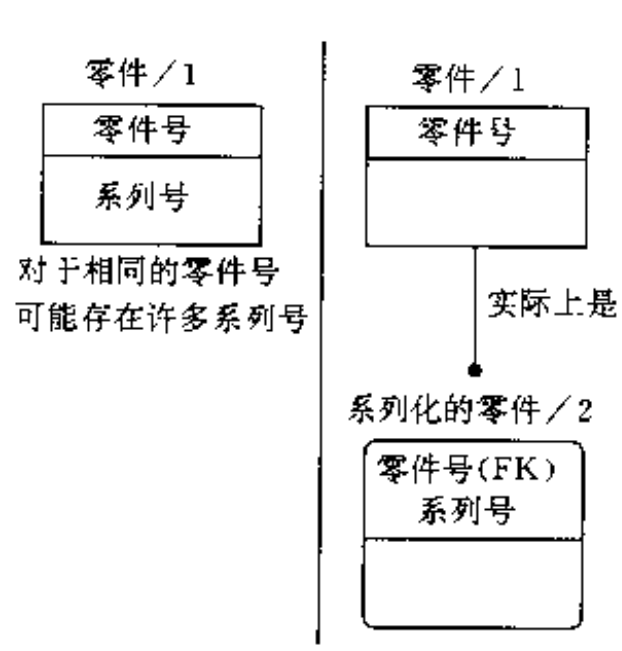


图 2.3.24　4 阶段应用不重复规则

当产生新实体的时候，必须要把它们登记到实体池中，并在关系矩阵中反映等。它们

必须满足以前阶段所有的文件要求，以便符合4阶段的材料。

要对每个属性的所有权评估，满足完全函数依赖规则，该规则说明，一个实体实例的非键属性值，必须由全部的键值来标识。该规则仅适用于有复合键的实体，并且等价于关系理论的第二范式。其例在图 2.4.19 中所示，如果“项目名”是实体“任务”中的一个非键属性，该属性已满足非空和不重复规则，但由于“项目名”可由“任务”的键属性的一部分“项目号”唯一识别，因而“项目名”不满足完全函数依赖规则，显然项目名应该是实体“项目”的一个属性。

第 4 阶段模型中的全部属性，也必须满足非传递依赖的规则，该规则要求，不存在一个实体实例的非键属性值，可由其他拥有或继承的非键属性来识别，该规则等价于关系理论的第 3 范式。

例如，以图 2.4.19 的“雇员”实体来说，显然部门名是雇员实体的一个非键属性，该属性满足非空和不重复规则，但由于部门名可以由部门号决定，部门号是一个继承的非键属性，显然部门名是“部门”实体的一个非键属性。

记住完全函数依赖和非传递依赖规则的简单办法是，“一个非键属性必须依赖于键，整个键，仅仅是键”。

5　4 阶段的结果描述

弄清了属性总体，就可以进一步改善模型，可以修改功能视图，并扩充说明非键属性。非键属性被列在每个实体盒子内横线的下面，实体盒子的大小可根据需要扩充，以提供写属性的地方。4阶段功能视图如图 2.4.25 所示。

对模型支持的定义和信息也应该修改，以反映非键属性的定义和所属的分配。这些附加信息，可由实体和先前定义的信息来报告。每个实体文件集现在包括：

- 每个实体的定义。
- 主键、次键、外来键属性表。
- 所属的非键属性表。
- 每个所属属性(键和非键)的定义。
- 父实体的联系表，该父实体是：
  - * 一个分类联系的一般实体；
  - * 可标识的父联系；
  - * 不可标识的父联系。
- 子实体的联系表，其子实体为：
  - * 一个分类联系的分类实体；
  - * 可标识的子联系；
  - * 不可标识的子联系。
- 任何双路径断言的定义。

也可以扩充任选的单一实体图来说明非键属性。

在文件集中可以为每个实体重复联系定义，或对实体以相互参照方式列出联系定义，同时也应该列出键和非键属性以及对实体的相互参照。

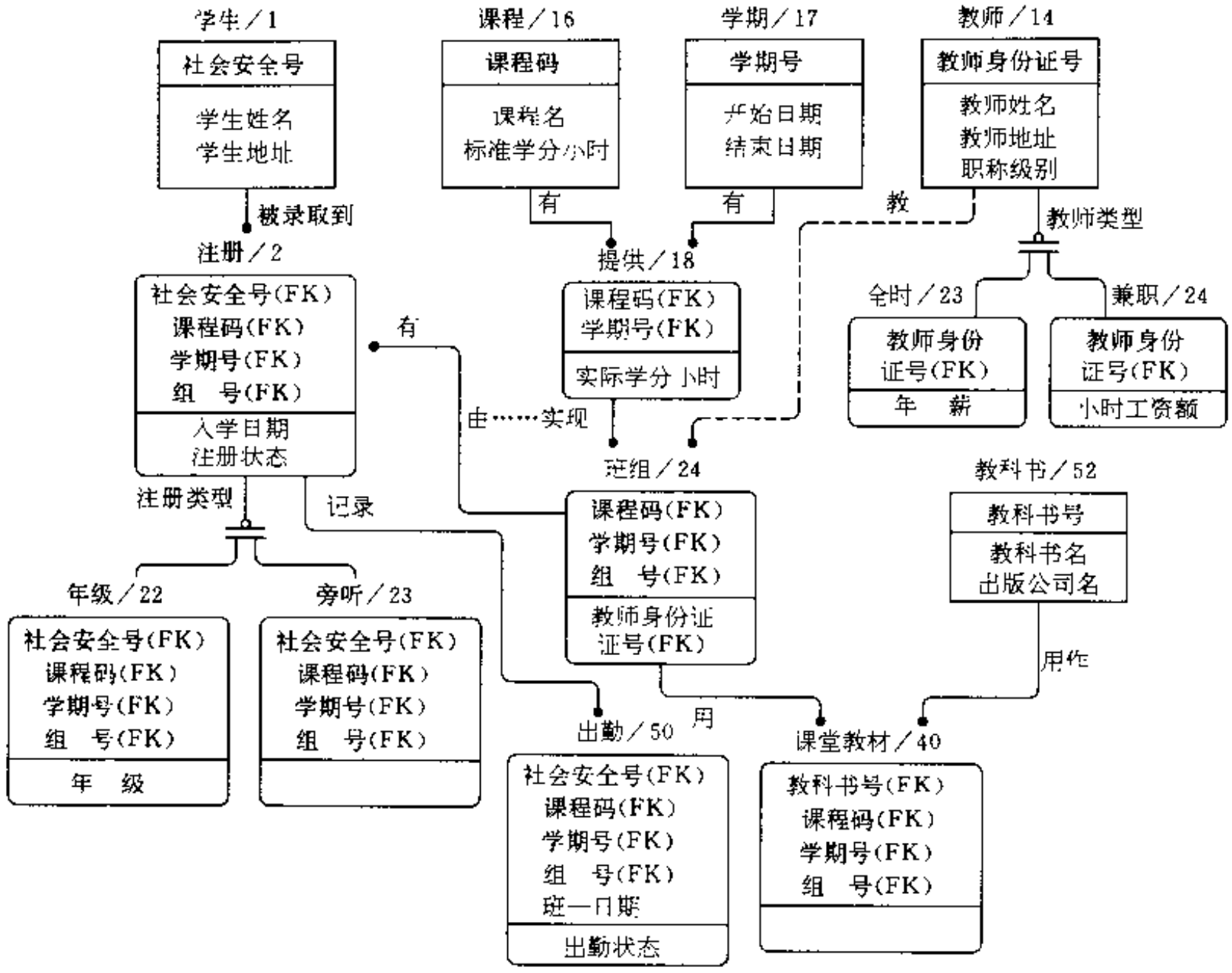


图 2.4.25 4 阶段功能视图举例

第 5 章　文件编制和确认

5.1　引　　言

IDEF1X 的目的是要提供一个一致的数据语义特征的集成定义，这些数据可用来为共享数据库的设计和信息系统的集成提供数据管理和控制。这就意味着模型必须编好文件，并由经营专业人员和系统专业人员两方面一起来确认，一旦一个初始模型建立并被批准，数据模型的配置管理就成了要慎重考虑的问题，因为新模型是与现有模型一起开发和集成的。

模型的文件编制和配置管理的很多工作可以通过利用软件工具来减轻负担。如在最简单的支撑水平上，一个文字处理系统就可用来保存实体、联系和属性的定义；标准的人机图形软件包，就可用来绘图，但这些工具并没有考虑模型内容，只是局限于它们本身的好处，大多数商用的数据字典系统不支持语义数据的定义；然面有些数据字典系统有一个用户可定义的章节，这一章节可设置来存放定义和提供各种报告。另一个方案是构造一个简单的数据库，来容纳模型描述，并利用数据库管理系统咨询工具来产生各种报告。现行的美国空军集成信息支持系统(IISS)的三模式字典，就是由一个关系数据库管理系统实现的，现在市场上已可买到专门的建模软件。一个建模软件工具的主要性能应包括：

- 模型图形的自动产生和布置
- 数据模型的组合
- 对照建模规则进行一致性检查和建模的自动精炼
- 作出报告的能力
- 配置管理支持。

虽然很希望有某种水平的自动化支持，但对 IDEF1X 建模先不要求，下一节将讨论假设在最大的自动化支持水平上，模型文件编制和确认的问题。

5.2　IDEF1X 组文件

组文件是一个技术文件，它可以包含图形、文本词汇表、决策概要、背景信息，或者任何为评审和提意见而放入程序包的东西。IDEF1X 建模项目的每个阶段，都需要创立一个以上的组文件，供主题专家和模型批准人评审。图 2.5.1 概括了组文件的评审循环。如一个组文件被送出去征求书面意见，作者通常必须回答评审人的意见。作为散发组文件征求意见的另一种方案，可以用模型遍历来取得评审人的一致意见，遍历将在 5.4 节讨论。

每个参加到一个项目中的人，可能都希望保存文件编制中收到的文件。然面，必须建

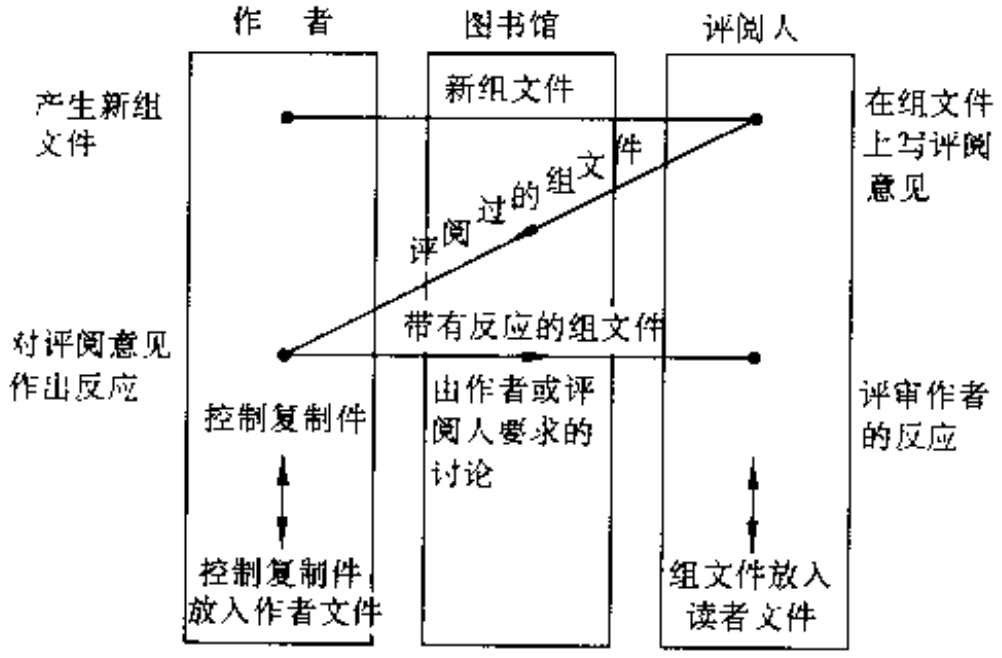


图 2.5.1　组文件的评审循环

立一种图书馆功能，来为每个组文件保存其主文件和参考文件。图书馆功能还要为组文件评审，起到散发机制的作用。图书馆文件的完整的解释见“ICAM 计划图书馆保存过程”。

在建模的每个阶段，都会用上多于一个组文件。下面给出了应该产生的全体组文件内容的概要：

（1）0 阶段组文件

组文件封面；目的和观点的阐述；模型开发和评审安排；工作组成员及其作用；源材料；作者约定（可选项）。

（2）1 阶段组文件

组文件封面；实体池；实体定义。

（3）2 阶段组文件

组文件封面；联系矩阵（可选项）；二阶段（实体级）图；实体报告（定义和联系）；联系定义；联系/实体交叉参照。

（4）3 阶段组文件

组文件封面；3阶段（键级）图；实体报告（定义、联系、断言、键）；联系定义；键属性表和定义；联系（实体交叉参照）；键属性/实体交叉参照。

（5）4 阶段组文件

组文件封面；4阶段（属性级）图；实体报告（定义、联系、断言、键和属性）；联系定义；属性表和定义（键和非键）；联系/实体交叉对照；属性/实体交叉对照（键和非键）。

5.3　标 准 格 式

1　封面

一个合适的封面标志着这份材料是一个组文件。封面有作者、日期、项目、文件号、标题、状态以及标注等栏目，对每个提交的组文件要完成一个封面，并在封面上填好见图 2.5.2 所示的栏目。

（1）工作信息（图中Ⓐ）

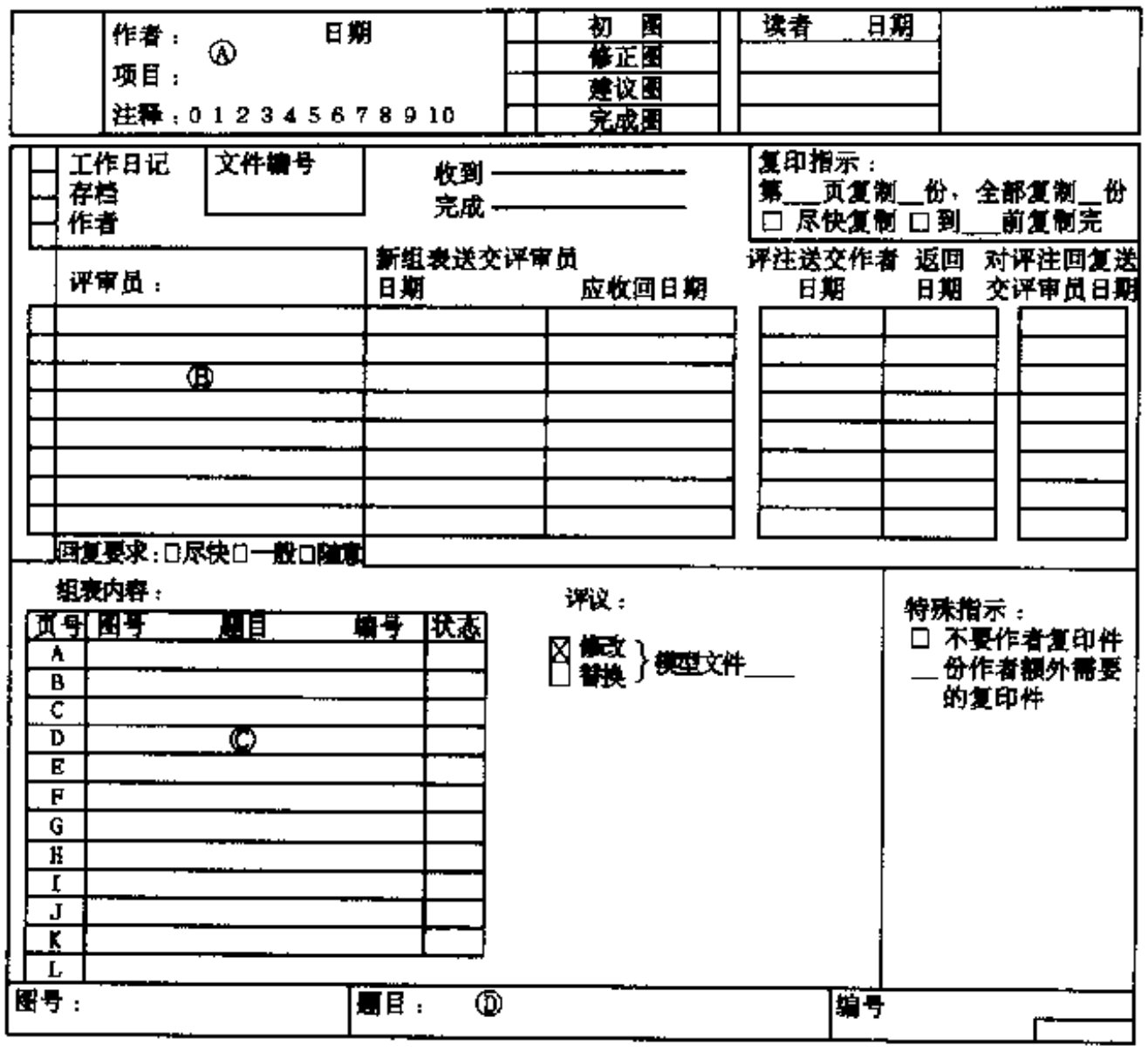


图 2.5.2 封面的标准格式

· 建立此模型的作者或工作组

· 项目名和任务号

· 初次提交图书馆的日期

· 所有已公布的修订版的日期

· 模型状态，即指初图(工作阶段)、修正图(校对阶段)、建议图(评阅后推荐为可接受的)、或完成图(作为最后模型公布)

· 评审以后读者签名及日期

(2) 评审人信息(图中Ⓑ)

· 归档和复印信息

· 组文件评审人表

· 组文件循环的各个阶段的安排日期

(3) 内容信息(图中Ⓒ)

· 组文件内容表格

· 每个组文件章节的状态

· 评阅意见或对图书管理员的特殊指令

(4) 识别信息(图中Ⓓ)

· 模型名(结点号)例如 MFG－1

· 模型标题

· 页号

2 标准图形格式

标准图形格式(图 2.5.3)具有最小结构与约束。此页只支持结构化分析学科所关注的功能：

· 文本的建立

· 图形和支持页之间的交叉参照

· 关于每一页内容的标注

本图形格式为便于归档和复制，取单一的标准尺寸，此格式分成3个主要部分：

· 工作信息(图中Ⓐ)

· 内容栏(图中Ⓑ)

· 识别栏(图中Ⓒ)

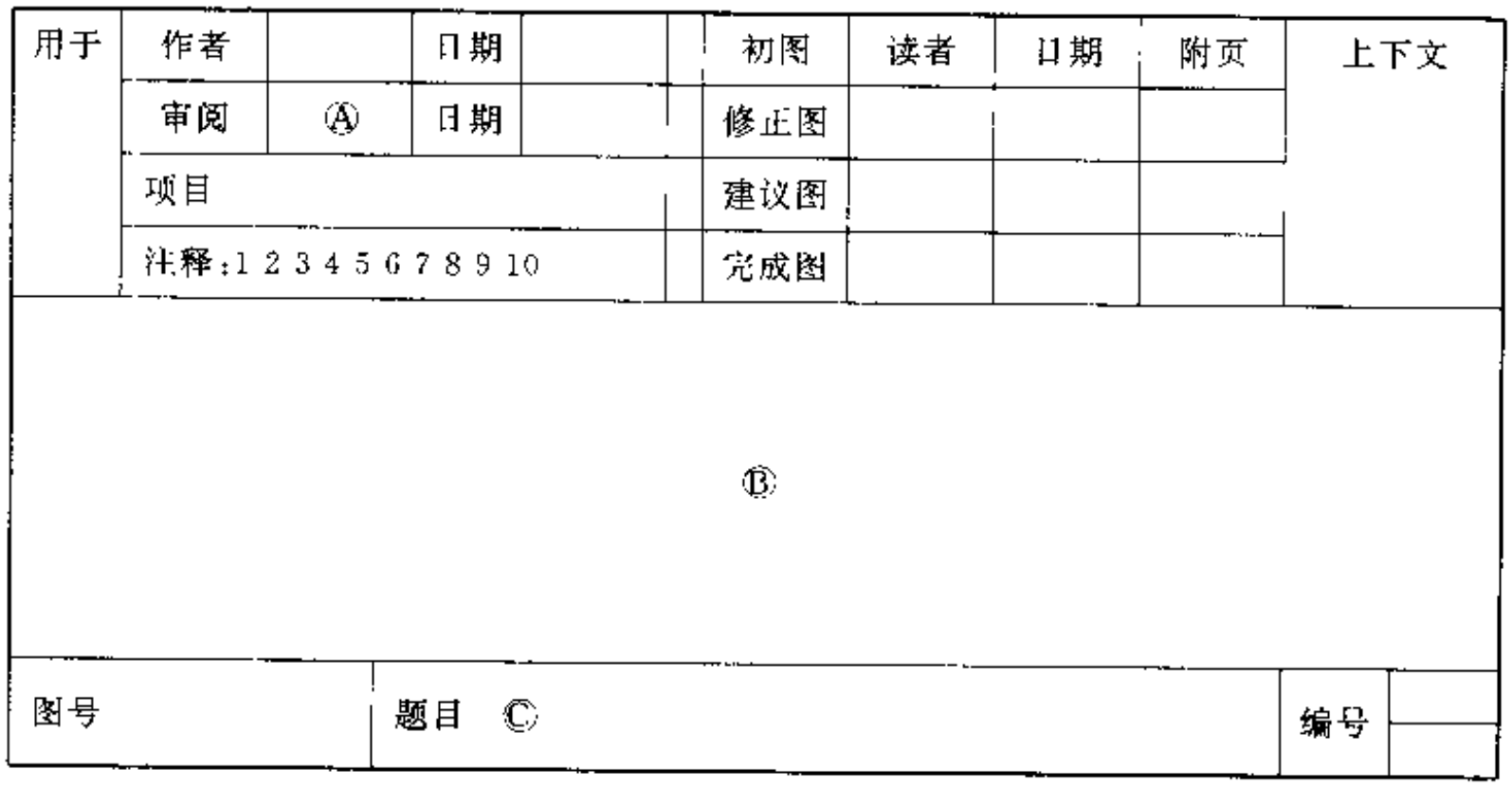


图 2.5.3 标准图形格式

设计此格式是为了，当完成了最后的批准公布的版本时，在格式顶端的工作信息可被切掉。这一标准格式应该用于建模过程中创立的每样东西，包括原始标准。

(1) 作者、日期、项目栏

此栏说明谁最初创作了这张图，第一次画的日期，赖以创作的项目的标题。日期栏可以包含多个附加日期，写在初始日期下面。这些日期代表着对最初那一张的修订，如果一张图未作任何修改地推出了，则就不必加修订日期。

(2) 标注栏

此栏提供了对写在本张图上的标注的一种检验。当你在此页作出评阅意见时，就顺序地把这里的标注数字勾掉，这样就可很快知道已有意见的数目。

(3) 状态栏

状态分类提供了批准过程中的等级。

初图(工作阶段)：此图形是一种不管以前状态的大改变的产物，新画的图形都属于工

状态分类提供了批准过程中的等级。

初图(工作阶段):此图形是一种不管以前状态的大改变的产物,新画的图形都属于工作阶段。

修正图:此图是对原有图形作些小改变,并已达到被一组读者在某种程度上一致接受的程度。修正图图形是指那些由一位任务领导人提出,但尚未被技术委员会或联合小组的评审会所接受的。

建议图:此图及其支持文本都已经过评审,并被技术委员会或联合小组批准,此图不希望再被改变。

完成图:此图可以送去作最后打印和公布。

(4) 读者、日期栏

在这里,一个评阅人开始了对每张表格评审,并注上日期。

(5) 上下文栏

此栏在开发 IDEF1X 模型时不用。

(6) 用于栏

这里列出了将本页以某种方式应用的那些图的表格。

(7) 内容栏

内容栏包含了要被传递的原始消息,在 IDEF1X 中,此栏可以包含图形、功能视图、定义、矩阵、索引等,作者不应使用与图形格式不一样的纸,可以将如图 2.5.4 所示的标准矩阵图形用于各种目的。

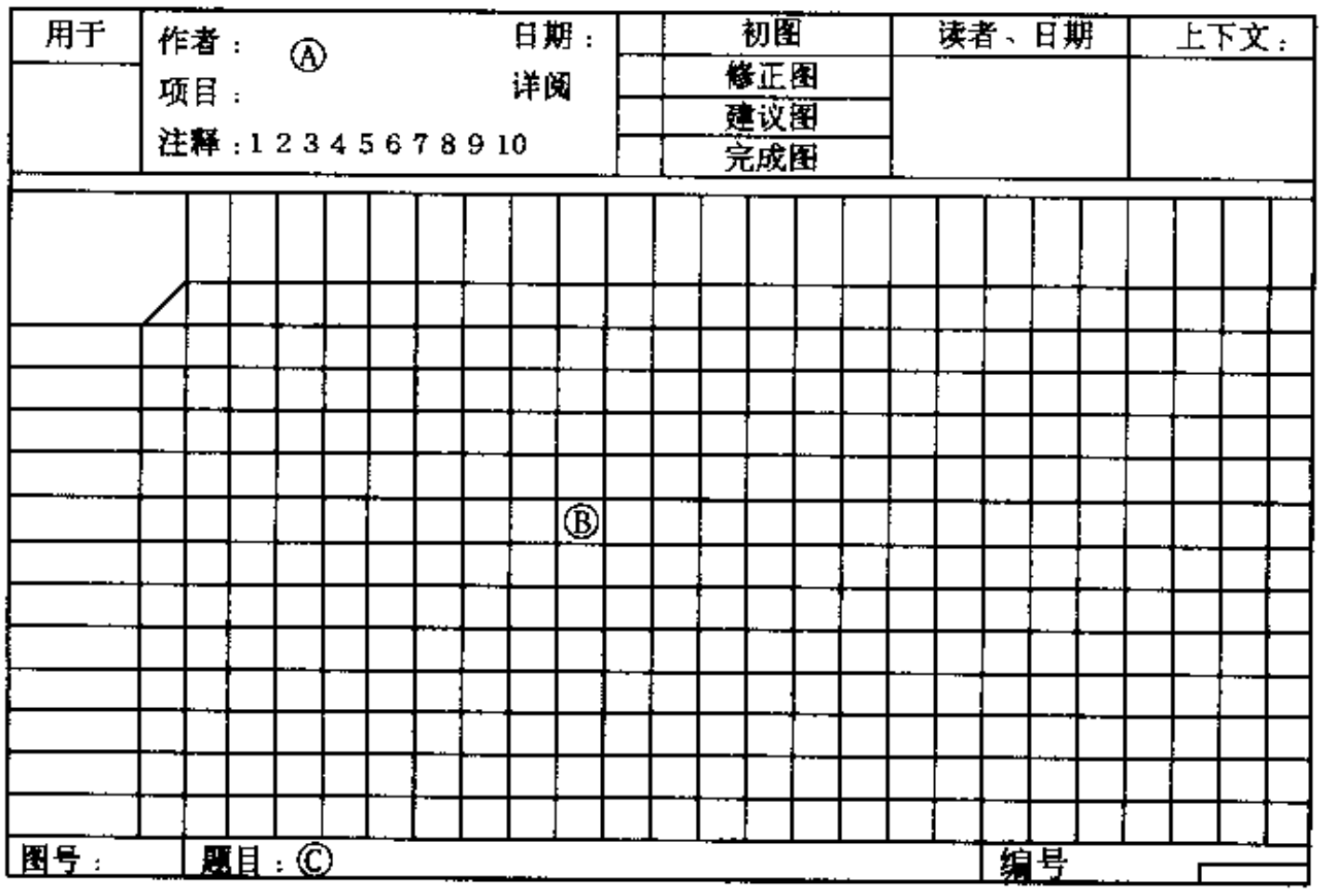


图 2.5.4 标准矩阵图形

(8) 标题栏

标题栏包含了表示在标准图形格式上的材料的名字。假若内容栏包含一张实体图

此栏包含了所有可以用来参考到这张图的数字。它包括下列内容：

① C 编号

C 编号由作者名字的缩写跟着作者按顺序指定的一个数字组成。C 编号放在数字栏的左下角，是参照一张图的原始工具。作者使用的每张图形都会有一个唯一的 C 编号，当模型被公布时 C 编号可以被一个标准的顺序页号所取代。

② 页编号

组文件的页编号由图书管理员写在数字栏的右边，它由文件号跟着一个用以在文件内识别这张图的一个数字组成。

5.4 IDEF 模型遍历步骤

除了组文件循环外，还已开发了一套遍历步骤，当建模中的参加者可以集合在一起作评阅时，可以应用这种步骤：

（1）演示将要用其实体池来分析的模型。这就是模型的内容表，并使评审人能很快地统观一下什么事将出现。

（2）演示术语的词汇表。这将允许每个评审人用演示小组选择的字义来代替那些个人认为的字义。此时字义就不应被怀疑了。因为意思上的一点改变将要求图形上很多改变。

（3）演示为评审用的功能视图图形。

功能视图遍历过程是一个有序的、一步接一步的过程，其中可以提些问题以便识别在模型中的潜在弱点，下面是结构化遍历的6个步骤。

在任何一步都可提出模型修改，这些修改可以标注上，以备后来某个日期的执行或者立即采纳之用。

第 1 步：扫描实体池。

这步使读者能获得有关模型内容的一般印象，因为实体池把删除了的实体也列上了，这使得读者更好地了解模型是如何演变到目前状态的。在这里，作者应检查一下实体定义。

通过的准则：

① 所选实体代表了为支持当前正在建模的环境所必需的信息类型。

② 所选实体从评审人的意见来看，对于模型的目的和范围是合适的。

除非问题非常明显，一般说这一步不做实质性改动。然而，不要丢失第一印象，在问题解决以前可把它们记在黑板上或活页本上。

第 2 步：阅读功能视图图面。

一旦读者理解了实体，就要读图检查一下联系的表达形式是否符合规范。

通过的准则：

① 联系基数符合 IDEF1X 手册所定义的细化规则。

② 所有需要的联系都直接或间接地表示了。

③ 图纸的结构安排使它很容易读（即交叉线最少，相关的实体安排得比较靠近）。

第 3 步:检查联系。

通过形式审查后,就要确定联系的内容是否被正确地表达了。

通过的准则:

① 不允许有非确定联系,即 $m:n$ 的联系存在。

② 在标定联系中的子实体和分类联系中的分类实体,都已明确为从属实体,用带圆角的盒子表示。

③ 一个分类实体只能对应一个一般实体,它只能是一个分类集的成员。

④ 一个分类实体不能是标定连接联系中的一个子实体。

⑤ 一个分类实体的全部实例都具有相同的“鉴别器”。

第 4 步:检查键。

通过的准则:

① 每一个实体必须有一个主关键字,满足非空规则。

② 在每一个连接联系或分类联系中,将父实体或一般实体的主关键字向子实体或分类实体迁移。子实体、分类实体至少包含一个外来关键字。

③ 主关键字和次关键字必须仅仅包含唯一标识实体的那些属性,符合“最小关键字规则”。

④ 键属性要满足“非重复规则”。

第 5 步:检查属性。

属性的填写是最后完善 IDEF1X 模型的步骤,要全面检查各种规则是否已正确贯彻。

通过的准则:

① 每个实体可以有任意个属性,一个属性只能归属于一个实体。每个实体可以有任意个继承属性,每个继承属性都必须是相关联的父实体或一般实体主关键字的一部分。

② 没有一个实例的属性值,可能具有一个以上的值,即符合“非重复规则”。

③ 实体的每一个实例,对每一个属性都必须具有一个值,即符合“非空规则”。

④ 模型中的每个属性名是唯一的。

第 6 步:设置图的状态。

① 按现实情况推荐。

② 修改后推荐。

③ 绘图:已作了太多改变,有必要重绘,且需要将来评审。

④ 未被接受:需要完全地重新分析。

附 录 2

2.1 IDEF1X 词汇表

评审委员会(acceptance review committee)

功能组织的一种,其职责是对建模工作提供引导和仲裁,并对完成了的产品通过最后的裁决(也就是模型接受了)。

断言(assertion)

一条语句,用来说明一种必须为真的条件。

属性(attribute)

用于描述关于一个实体的某些事情的数据的特征或元素,对属性要给予一个专门名字来指明其意义(例如,头发颜色)和值(例如,棕色)。

继承属性(attribute, inherited)

一种属性,它是另一个实体的主键(或主键的一部分),是由于实体之间的联系而从那个实体迁移来的,亦被称为迁移属性。

迁移属性(attribute, migrated)

与继承属性相同。

占有属性(attribute, owned)

此属性不是继承来的,占有权是相对于一个属性而言,此属性只能被一个实体所占有。

属性总体(attribute population)

用以确定属性类的"占有权"的那种工作。

属性作用(角色)(attribute role)

描述了一个属性在描述实体[包括继承(=迁移)、占有、主键、次键(又译作"主关键字""次关键字")等属性中所起的作用。

属性值(attribute value)

赋予一个属性的确切的数据值(例如,属性:头发颜色;属性值:棕色)。

作者约定(author convention)

为加强模型的表达和利用,由作者开发的一些特有的实际做法和标准。作者约定不允许违反任何方法论中的规则。

约束(constraint)

一种断言,其目的是为了明确地说明数据的含义。

布尔约束(constraint, boolean)

在对同一个父实体的多重联系中,限制子实体实例的一种条件。算子"与"(AND)表示

在两个联系中父实体必须都有子实体实例。算子“或”(OR)表示父实体可以在两个联系中的一个或两个联系中有子实体实例。算子“异或”(XOR)表示父实体只可以在最多一个联系中有子实体实例。

基数约束(constraint, cardinality)

对父实体联系中可能存在的子实体出现次数的限制。

存在约束(constraint, existence)

一个实例,除非另一相关实体的一个实例也存在,否则其一个实例就不能存在的条件。

数据搜集计划(data collection plan)

用来标识诸如功能、部门或工作人员等目标的计划,这些目标是用于开发模型的资料的源。

域(domain)

一组可允许的值。域可以用一个数据类型来说明(例如整数、日期、钱),并可包含对值的范围的约束(例如,大于零;在 2 与 12 之间;17 个字符;在所列数字 2,5,10,16 中取)。域可以被分配给一个或更多属性。

实体(entity)

相似实例(人、地方、东西或事件)的集合,可以用一个通用名词来命名,它有一个键(或称关键字,它将唯一地标识每个实例),并有一个或多个属性(它们将描述每个实例)。

实体图(entity diagram)

表述“主题”实体和所有直接与主题实体相关的主题的一张图。

实体实例(entity instance)

一个命名了的实体的一次出现,它可以专门用其键的值来标识,一旦这实例被确定,该实例的所有其他属性的值也就都知道了。

分类实体(entity, category)

一种实体,其实例是代表着同一种现实世界中物体的另一实体的实例的子类。例如,“拿薪水的雇员”就是一般实体“雇员”的一个分类实体。

依存(existence dependency)

两个实体间的一种约束,它表明,从属实体的实例,如果没有与另一实体的实例相关,就不能存在。依存关系,就是参照的整体性,加上外来键不许有零值的约束条件。

专家评审员(评阅人)(expert reviewer(commentor))

建模队伍的成员之一,其专业知识集中于制造企业内某项专门的活动,他的职责是对在建立过程中的模型提出一些关键性的意见。

FEO(参照图)(for exhibition only)

这个缩写意思是“仅供说明用”,这是一种载体,它通过把图、文本等的某种组合,为模型提供支持性的和解释性的信息。

函数依赖(functional dependency)

两种属性之间的约束,它表明一个属性的值是由另一属性的值所确定的。

IDEF 组文件循环(IDEF kit cycle)

在开发过程中,模型的一部分在建模者和读者/专家评审员之间正规的交换,其目的是

为了隔离和检查错误、遗漏和误表示。

IDEF1X 模型(IDEF1X model)

在一种环境下数据含义的一种图形表示。它显示了数据的基本结构和联系。应用扩展的ICAM 定义语言作信息/数据建模(IDEF1X)的产物。

标识符依赖(identifier dependency)

两个实体之间的约束,它要求在从属实体中的外来键成为其主键(或部分),标识符依赖是一种更强的依存关系形式。

键(key)

实体的一个属性或属性的组合,其值唯一地标识每个实体实例。又称关键字。

次键(key, alternate)

不同于实体主键的键。

合成键(key, composite)

由两个或更多属性组合起来的键。

复合键(key, compound)

与合成键相同。

外来键(key, foreign)

在从属实体中出现的属性,也在另一个实体中作为主键。

成员键(key, member)

作为合成键的一部分的一个属性。

迁移键(key, migrated)

与外来键相同。

键迁移(key migration)

在联系中将父(或一般)实体的主键放入子(或分类)实体中的过程。

主键(原始键)(key, primary)

被选作对所有联系作迁移的键,而该实体在这些联系中是作为父实体或一般实体的。

建模者(作者)(modeler(author))

建模小组成员之一,其职责包括在模型开发过程中的数据收集、教育和训练、模型记录及模型控制,建模者是在 IDEF1X 方法方面的专家。

范式(normal forms)

反映了在识别实体和将属性置入数据模型中的实体内时细化的程度的条件,每个范式反映了对实体的属性间的联系越来越严格的控制。

- 第 1 范式(1NF):对实体实例中的任一属性没有多于一个的值。
- 第 2 范式(2 NF):1 NF 加上非键属性的值是由实体实例的整个键,而不是仅由其一部分所确定(即非键属性对键的完全函数依赖)。具有一个非复合的键的1NF 的实体,就自动地属于 2NF。
- 第 3 范式(3 NF):2 NF 加上没有非键属性的值是由另一个非键属性的值所确定的(即非键属性不存在对键的传递依赖)。只有一个非键属性的 2 NF 的实体,就自动地属于 3 NF。

- 第 4 范式(4 NF):3NF 加上三个以上属性的复合键中,没有一个属性,它对键中另外两个属性之一的关系比其他属性更密切些。3 NF 中的实体,如果其键少于三个属性,就自动地属于 4 NF。
- 第 5 范式(5 NF):4 NF 加上如不引入新的含义,不可能把属性分裂出来加到其他实体中去。4 NF 中的实体,如果其键少于三个属性,就自动地属于 5 NF。

规范化(normalization)

根据范式的实体进行细化和重组属性的过程,它会使数据的相互关系更加简明。

0 阶段(phase zero)

建模活动的初始工作,在这里建立了前后关系的定义,也就是项目定义、数据收集计划、作者约定、标准等等、

1 阶段(phase one)

建模工作循序渐进的第2步,这时要识别和定义实体。

2 阶段(phase two)

建模工作循序渐进集当中的第 3 步,这时要识别和定义联系。

3 阶段(phase three)

模型开发循序渐进集当中的第 4 步,这时要识别和定义键。

4 阶段(phase four)

模型开发循序渐进中的第 5 步,这时要识别和定义"非键"属性。

项目经理(项目负责人)(project manager)

建模小组成员之一,其职责是对建模工作的行政上的控制。其责任有:配备小组成员,设定范围和目的,主持验收评审委员会等。

联系(relationship)

实体之间的一种逻辑关联。

联系基数(relationship cardinality)

在一个联系中可以被互相关联的实体实例的数目。参见基数约束。

联系名(relationship name)

一个短语样的定义,它反映了图上所示的两个实体之间所表达的联系的意思,这名字就出现在那张图上。

非标定联系(relationship, nonspecific)

一种联系,这里没有一个实体能说成是独立于或存在依赖于另一个实体。例如多对多联系和 0 或 1 对多联系。

标定联系(relationship, specific)

一种联系,这里一个实体是存在依赖于另一个实体的。

作用名(role name)

一个分配给外来键的名字,它在一个实体中出现多于一次。

模式(schema)

数据结构的定义:

- 概念模式:在一企业内部集成的共享数据的一种中间定义。它由语义数据模型来代

表，该模型要与数据精炼所用的规则相一致，并以第五范式出现。

- 外部模式：描述了共享数据的一种应用的或使用人的观点。
- 内部模式：描述了共享数据的 DBMS 的物理表示法。

语义(semantics)

在语言中字和句子，或者模型中结构的意义，与句法相对。

源(source(s))

建模小组的成员之一，其职责是提供在开始和继续模型开发中所用的信息成分(文件、表格、步骤、知识等等)。

句法(syntax)

文法。从词汇表中的字来形成有意义的短语和句子的一组规则，与语义相对。

批准(确认)(validation)

一种工作，它导致对模型有丰富知识的专家们取得一致意见；假如大多数专家同意该模型能够合适地和完全地代表了所涉及的领域，则就认为此模型"生效"。

2.2 IDEF1X 与 IDEF 1 的比较

1 术语

所推荐的 IDEF1X 的术语与 IDEF1的略有不同。新术语更适合于在大范围内的数据建模行业所用的术语，它也更适合于 IDEF1用户实际上用于模型构造的方法。

下面的表列出了在 IDEF1X 和 IDEF1中一些概念的对应项。

IDEF1X	IDEF1
实体	实体类
属性	属性类
联系	关系类
候选键	次键类
主键	键类
带(多个)次键的主键	(多个)次键类
外来键	迁移键类
实体实例	实体
属性值	属性
联系实例	联系

本附录下面就应用被推荐的 IDEF1X 术语，即使当涉及一个 IDEF1模型的构成时也这样。

IDEF1X 术语更适宜于开发数据源管理领域的工业标准词汇表。(由 IBM 和别人一起开发的)关系模型和(由 UCLA 和别人一起开发的)实体-联系(E-R)模型用了 IDEF1X 的名词。

IDEF1X 对一些更经常用到的概念，给出了更简单的名词。譬如，建模者一般都涉及

实体而很少涉及实体实例。

2 实体句法

下面几节将先阐明 IDEF1如何支持这些构成，然后讨论在 IDEF1X 方法中的相似点和不同点。

IDEF1X 支持的有关实体的语义构成如下。

(1) 实体

IDEF1用矩形盒子代表实体，IDEF1X 也这样。

(2) 独立标识符对从属标识符实体

IDEF1X 区分了在识别时不依赖于其他实体的独立标识符实体和识别(或存在)时必须依赖于另外的实体的从属标识符实体。

相反，IDEF1则对两种构成用同样的符号(一个带方角的矩形盒子)。IDEF1X 的句法允许独立标识符实体在一张数据模型图中更加突出。

在 IDEF1X 中，独立标识符实体被画成带方角的矩形盒子，从属标识符实体则被画成带圆角的矩形盒子。

注意，IDEF1X 中，实体标识符的独立或从属是相对于整体模型的，而不只是相对于一个特殊的联系。

(3) 实体名

IDEF1的实体标号被放在实体盒子里面，实体名是作为定义的一部分给出的。

相反，IDEF1X 的实体名被放在实体盒于上面，因为它要用于盒子中每样东西，这个惯例使得建模者能很快草拟出模型图。在 IDEF1X 中，因为实体名可以用缩写字，就没用实体标号。

(4) 实体号

IDEF1的实体编码模式，勾画出了实体盒子中很重要的一部分。实体号被放在盒子的左上角用一根斜角线隔出来的区域中。如附图 2.2.1。

语义构成	IDEF1X	ICAM的IDEF1
实体 在识别时不取决于其他实体的实体 在识别时要依赖于某个另外实体的实体	实体名　实体号 雇员／32 实体盒子 雇员／32	实体类号 32 实体类盒子 雇员 实体类名

附图 2.2.1　实体画法的差别

相对来说，IDEF1X 的实体号不占有实体盒子中的空间。它跟着实体名而用一个“/”与实体名隔开。

3 属性句法

IDEF1X 支持的有关属性的语义构成如附图 2.2.2 所示。

语义构成	IDEF1X	ICAM的IDEF1
属　　性	单位号 部门号 —— 主键属性 部门位置 部门大小 —— 非键属性	56 单位号 部门号 键类 部门位置 部门大小 非键属性类 部　门
主键属性	XXX YYY	XXX YYY
次键属性	YYY —— 主键 XXX(AK#) —— 次键 例：　雇员号(AK1)	(YYY) (XXX)　次键类（未规定七键）
外来键属性	XXX(FK) 例：　单位号(FK)	未规定
带作用名的属性	作用名.主键名 例：部件号.零件号	未规定，无作用名

附图 2.2.2　属性表示的差别

(1) 属性

IDEF1把属性放在实体盒内，IDEF1X 也这样。在 IDEF1X 中把属性名用在 IDEF1中属性标号的地方。

(2) 候选键属性

候选键是一个或几个属性。其值唯一地标定实体实例。IDEF1对候选键属性下面划了线，若有多于一个候选键，则每个都要用括弧括起来。若候选键是复合的或重叠的，则在多个键中出现的属性会在实体中出现多次。

与 IDEF1相对照，IDEF1X 需要把候选键之一规定为主键，这是一个通过联系可被迁移到其他实体去的键，另一些候选键则称为次键，规定主键对于自动规范化和一致性检查是很必要的。

IDEF1X 对次键的每个属性都在属性名后面标上一个次键号：(AK n)。一个属性可以是多个次键的一个成分，因此就有多个次键号，例如，考虑下面三个属性：

绘图号(AK1)

修改号(AK1,AK2)

零件号(AK2)

次键之一是绘图号,修改号,此键被规定为 AK1;另一个次键是零件号、修改号,此键被规定为 AK2;修改号是两个次键的成分,作为属性,修改号在实体中只出现一次,相反,IDEF1则要在实体中出现两次修改号,对每个次键出现一次。

(3) 主键属性

IDEF1并不把主键与任何其他候选键相区分,所有候选键属性都是下面划线的。

相反,IDEF1X 将主键属性放在分割实体盒子的一条线上面,这使最重要的属性很容易被区分出来,因为它们是明显地与非键属性分开的。假如在图形句法中不用下划线,那么自动绘图支持也就变得对设备更少依赖了。

(4) 外来键属性

外来键属性是这样一种属性,它是另一实体主键的一部分。

IDEF1根据一个属性出现在一个候选键中时具有相同的名字,来规定外来键属性(称为迁移键类)。IDEF1属性必须有同样的名字才会被检测为"外来键-主键"对。

相反,IDEF1X 则对外来键属性在其名字后面标上(FK),这就使问题很清楚,表明了哪些属性是从其他实体迁移进来的。一个外来键属性可以是主键属性、次键属性或非键属性,外来键属性通常是它迁移由来的那个实体的主键的一部分。

(5) 作用名

IDEF1不支持作用名,假如一个属性通过两个联系迁移进来的话,则它就要在每个实体中以同样名字出现两次。

IDEF1X 支持对属性的作用名,这里"第二名字"更好地阐明了它们的含义。这些属性通常都是外来键属性,它们一般是通过多个联系迁移进来的,在实体中出现两次且需要进行区分。自动规范化要求给它们以不同的名字以阐明它们有不同的含义。

例如,考虑在实体"零件"和"部件"之间的物料单结构,"零件"的主键属性迁移到"部件"中两次,一次是通过联系"是在……中"(IS－IN),另一次是通过联系"有"(HAS)。在 IDEF1模型中"零件号"会在"部件"中出现两次,相反,IDEF1X 支持对这些"零件号"的出现赋予作用名:一次出现可被命名为"部件号·零件号",另一次出现则可被命名为"装配号·零件号"。然后软件支持将能为外来键检测出这两种作用而不是把它们解释为冗余的。

4 联系句法

IDEF1X 支持的有关联系的语义构成如下。

(1) 连接联系

连接联系是两个不同实体间的一种交往。例如"部门"和"雇员"用一个名为"雇用"的连接联系关联起来,"绘图"和"零件"用一个名为"规定"的连接联系关联起来。

IDEF1用一条线把参与一个连接联系的两个实体联系起来,这条线两端的符号表明了每个实体的多少个实例可以与另一个实体的多少个实例相关联,这些符号称之为基数

符号。IDEF1X 基本上用了同样的方法。

然而，IDEF1用了两种根本不同的符号来代表联系基数，零或多个的基数用一个开放的菱形表示(零或1的基数用线上的半个菱形表示)。因为它看来非常像一个箭头，半菱形可能被误解为表明了数据流。

相反，IDEF1X 经常用一个“大点”符号来代表联系的基数。这个大点被注解来表明确切的基数。不标什么的，这个大点就是指“零、1或许多”，一个“P”表示正数(亦即1或更多)，一个“Z”表示零或1，一个“n”表示一个特定的数(亦即$=n$)。IDEF1X 用了大点符号因为它是最大的单字符号，且不会在用照相缩小时扭曲，它也容易用徒手画。

(2) 分类联系

IDEF1 没有一个满意的方法来代表分类联系，它把用于具有 0 或 1 基数的连接联系的同样的句法，用到分类联系上，IDEF1依靠建模者去指定一个“可能是”的联系名，来将分类与连接相区分；一般说分类实体是通过应用 IDEF1的“非空规则”来发现的。

此外，IDEF1 不能把分类实体集合到一起，来形成一个一般实体，IDEF1 模型可以阐明“雇员”“可能是”0 或 1 个“计时雇员”，“雇员”可能是 0 或 1 个“工资雇员”，“雇员”也“可能是”0 或 1 个“未分类雇员”。但它无法表明，一个“雇员”实例只能是这些分类中的一种。

相反，IDEF1X 用一条线带一个开放的圆来代表分类联系，分类联系的图显然与连接联系的不一样。

一般实体的鉴别器属性穿过这个圆，该鉴别器值，确定了对这一特定的一般实体实例，存在着哪些类别的分类实体。

假如给出了该类别的完全集，则分类圆下面划双线，这意味着鉴别器的每个可能的值都由一个分类实体代表了。例如，若只有三个可能的“雇员”子类型，而所有三者都已作为分类实体建立在模型中了，则就要用下划双线(附图 2.2.3)。

然而，若要代表类别的不完全集，则分类圆下面就划单线，这意味着鉴别器可以有一个值而没有用分类实体。例如，若三个可能的“雇员”类别中只有两个建在模型中了，则就要下划单线。这是很普通的。即当一个分类实体没有它自己的而只有一般实体的属性时就是这样。

如果分类联系没有在图上标出名字，当要读这个联系时，就用名字“可能是”。

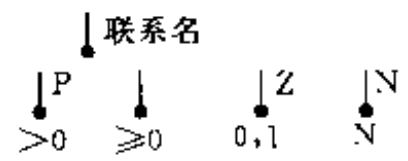
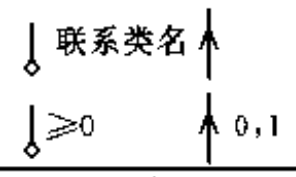
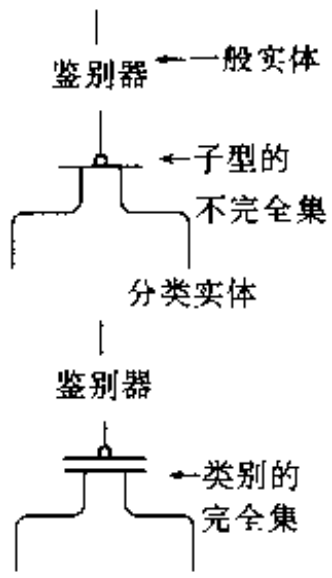
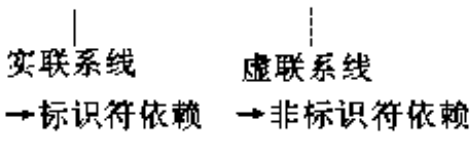
(3) 标识符依赖

当迁移键属性变成子实体中的(部分)主键时，就出现标识符依赖。子实体的识别取决于父实体的键。这样，子实体就是标识符依赖于父实体。这是完全地依赖于父实体，没有父实体就不能存在。

当外来键属性(也就是迁移键属性)被下划线时，IDEF1就意味着标识符依赖。

相反，IDEF1X 在模型图中使标识符依赖很明显，它用实联系线代表标识符依赖的联系，而虚联系线则为非标识符依赖的联系，标识符依赖是作为两实体间联系的一个特征，因此是用联系的图形表示的。

分类联系总是有标识符依赖的，连接联系则可以有也可以没有标识符依赖。

语义构成	IDEF1X	ICAM的IDEF1
联系 连接联系 基数		
分类联系		除了用 外没有表明
标识符依赖		在迁移键类用下划线／无下划线来隐含

附图 2.2.3　分类联系表示的差别

5　例子

下面是 IDEF1和 IDEF1X 图形表示法之间的相似点和不同点的例子。序号依次对应着下面几页上的图。(附图 2.2.4)

(1) 两个实体 ALPHA 和 BETA，具有非标定的1对多联系 REL，ALPHA 的主键 A 成了 BETA 中的非键外来键属性。

(2) 两个实体 ALPHA 和 BETA，具有标定的1对多联系 REL，ALPHA 的主键 A 成为 BETA 主键的一部分。

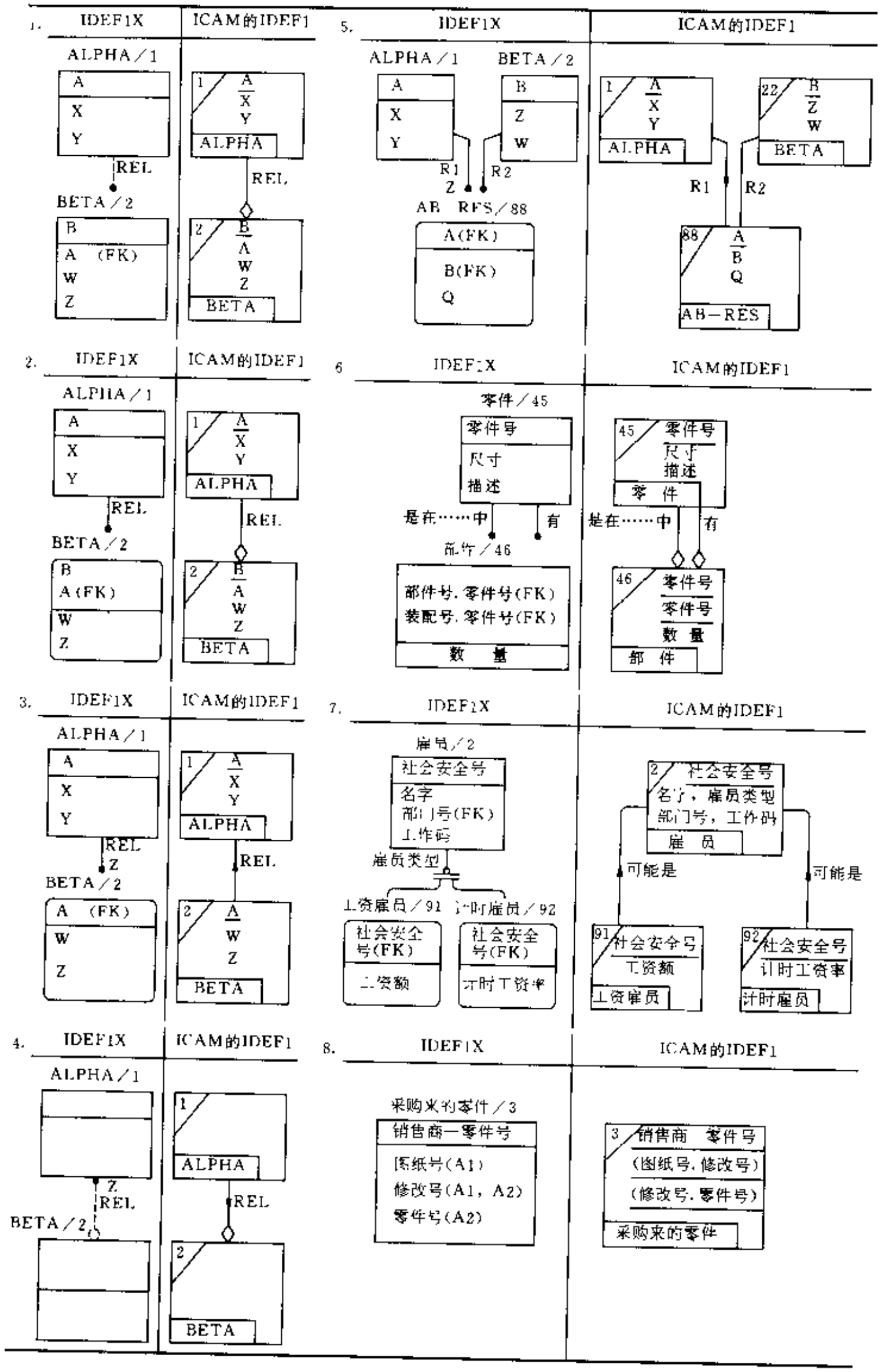
(3) 两个实体 ALPHA 和 BETA，具有标定的1对零或1联系 REL，ALPHA 的主键成为 BETA 的主键。

(4) 两个实体 ALPHA 和 BETA，具有非确定联系 REL，一个 ALPHA 实例可以与多个 BETA 实例关联，一个 BETA 实例可以与零或1个 ALPHA 实例相关联。

(5) 两个独立实体 ALPHA 和 BETA，具有它们的依赖交叉实体 AB-RES，ALPHA 的主键 A 成为 AB-RES 中的主键外来键，因为联系 R1是标定的；BETA 的主键 B 成为 AB-RES 中的非键外来键，因为联系 R2是非标定的。

(6) 独立实体“零件”与从属实体“部件”之间的物料单结构。“零件”的主键“零件号”有作用名“部件号”，这里它是通过联系“有”迁移进“部件”中去的，两者都是标定联系。

(7) “雇员”是一个一般实体，具有类别“工资雇员”和“计时雇员”。每个“雇员”实例，必须有或者是相应于“工资雇员”或者是“计时雇员”的实例。而具有同样的主键“社会安全



附图 2.2.4 IDEF1和 IDEF1X 图形表示法之间的相似点和不同点的例子

号”值。(注意,IDEF1模型不代表这两个联系的互斥性,也不代表必须有一个存在的需求,在“雇员”中“部门号”和“工作码”是两个非键外来键属性,通过另外的联系迁移进来的。

(8)“采购来的零件”有三个候选键:“销售商—零件号”是主键;图纸号,修改号是一个次键;零件号,修改号是另一个次键。

2.3 初步设计阶段 IDEF1X 方法实施规则

1 本实施规则的目的

为了简化 IDEF1X 方法的实施,方便大家掌握和运用 IDEF1X 方法表示初步设计的“信息模型”,我们在“成飞 CIMS 工程”初步设计中采用了简化 IDEF1X 方法。这样可使初步设计的信息模型作为详细设计阶段的基础,又力求使简化 IDEF1X 方法符合 IDEF1X 的规定。

2 初步设计阶段简化 IDEF1X 方法的设计步骤

第 0 步:收集源数据。建立源数据资料档案,包括 IDEF0图、现行系统输入和输出报表、现行文件报表,将收集的源数据进行归档,供以后设计使用。

归档文件格式:

序　　号	数 据 名	说　　明	所引用的源文件

第 1 步:确定实体。

确定数据对象集,以实体为单位表示数据集。例如:定单、BOM、主生产大纲……,它表示一定概念的数据项的集合。

在可能的条件下,确定实体的属性,以属性名形式表示。建立实体属性表,表示形式:

序号 实体名 (属性名1,属性名2,……)

第 2 步:确定实体间的联系。

实体间的联系,表示实体的实例之间对应关系。在本方案中只允许二元联系,即只能表示两实体间联系。

允许 1:n 联系,m:n 联系和分类联系。

为了方便检查,建立功能视图和实体级图。

功能视图:即表示某功能(处理)所涉及的全部视图及联系,仅在一张图上表示出来。

实体级图:仅应用于关键实体,即它与其他实体联系较多的实体,它表示一个主题实体与其他实体的联系。

关于建立实体/联系矩阵,作为选择项,由各设计组自行决定。

3 符号表示

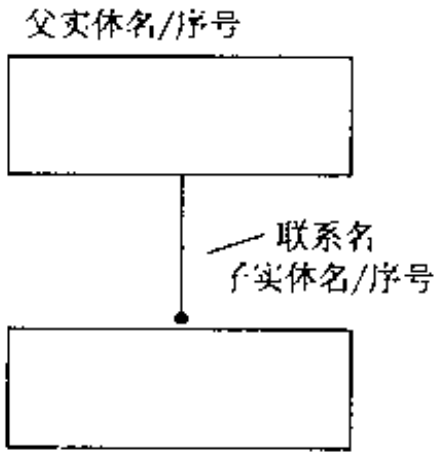
为了便于阅读,IDEF1X 的图形表示采用下列统一表示:

(1) 实体表示

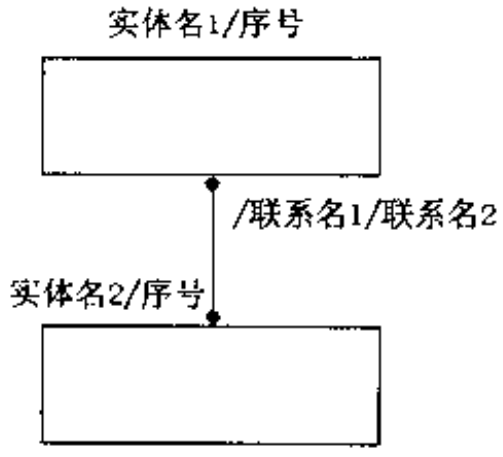
(2) 联系表示

1:n 联系类型图形表示,如附图 2.3.1。

m:n 联系类型图形表示,如附图 2.3.2。



附图 2.3.1 1:n 联系

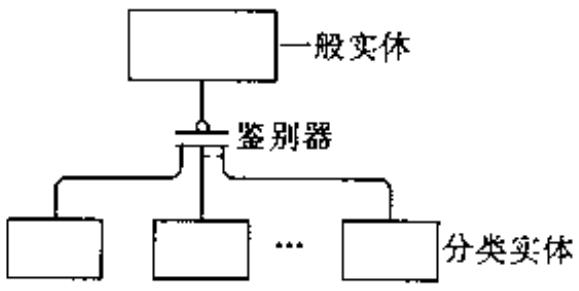


附图 2.3.2 m:n 联系

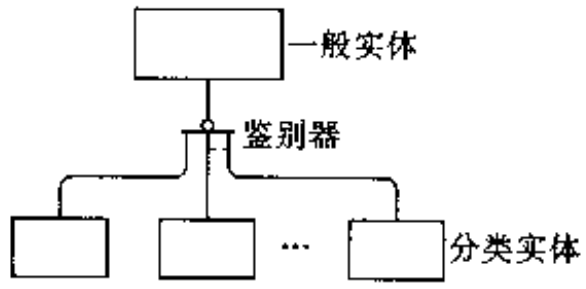
分类联系(一般实体与分类实体联系)分为完全分类联系和不完全分类联系。

完全分类联系的表示方法如附图 2.3.3所示。

不完全分类联系的图形表示如附图 2.3.4所示。

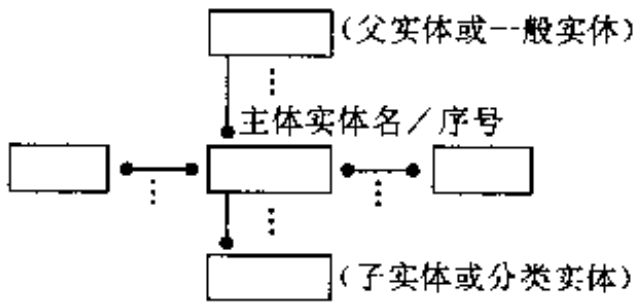


附图 2.3.3 完全分类联系



附图 2.3.4 不完全分类联系

(3) 实体级图的表示如附图 2.3.5所示。



附图 2.3.5 实体级图的表示

(4) 功能视图

使用上述实体、联系图形表示符号将一功能(处理)所涉及的实体、联系表示在一张

图上。

(5) 实体/联系矩阵

如果实体 i 与实体 j 之间有联系，则在表的(i,j)和(j,i)位置用×号表示。(附图2.3.6)

实体名 \ 实体名	实体1	实体2	⋮	实体 n
实体1				
实体2				
⋮				
实体 n				

附图2.3.6　实体/联系矩阵

4　IDEF1X 初步设计阶段资料清单

(1) 原始数据资料；

(2) 归档文件；

(3) 实体属性表；

(4) 实体级图；

(5) 功能视图；

(6) 实体/联系矩阵。

第 三 篇

过程描述获取方法 IDEF3

第1章 引 言

当前,“经营过程重构”已经被认为是企业界的一场革命。而要进行过程的根本变革,必须有一套正确的方法和工具来分析过程 。正像在系统集成中人们已熟悉的 IDEF0 和 IDEF1X 方法那样,IDEF 家族不断开发着新的成员。IDEF3 就是一种为获取对过程的准确描述所用的方法,准确的描述是分析问题、进而改造过程的基础。

首先要说的是,“方法”可以定义为一种有组织的,单一目的的规程或实践(Coleman,1989)。正如我们以前对“建模”作的定义一样,它用于回答对象的某一方面的问题,也只能回答一个方面的问题,而不可能包罗万象。IDEF3 则是为了进行过程描述的。

通常可以认为,一种方法将包含 3 个基本组成成分:(1)定义;(2)规程;(3)应用。定义指在方法背后的基本的直觉和动机,所涉及的概念以及运作的理论。有的方法有正式的理论基础,但还有很多是没有的。规程包含应用该方法的步骤以及方法的语言(或称语法)。这一由方法的规程所规定的步骤为实践者提供了一种可靠地取得较好结果的过程。方法的语法则是用来消除复杂工程产品开发中的含混不清。许多系统分析和工程方法,都用图形语言来明确显示所收集的数据,以达到突出主要矛盾,去粗取精的作用。图 3.1.1 就是这一思想的骨架表示,骨架中的第三个要素——应用,则侧重于阐明应用此方法的来龙去脉。

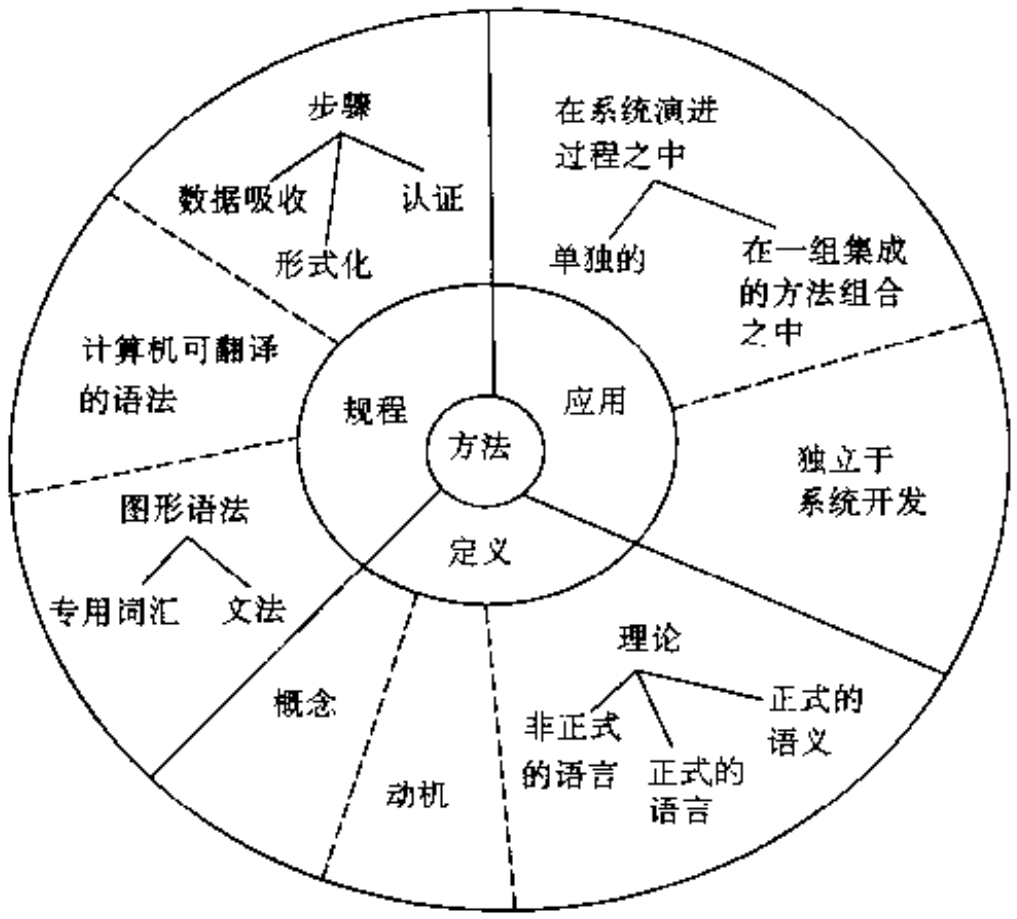


图 3.1.1 方法的骨架

具体到 IDEF3 方法，就是要有一套结构化的规程，把一个领域专家对其熟悉的过程或系统的运行知识确切地描述清楚。IDEF3 是 IDEF 家族中的一员。正如 IDEF0 是用于建立功能活动及其相互关系的功能模型，IDEF1 和 IDEF1X 是用于建立信息和数据实体的相互关系及属性描述的信息模型和数据模型，IDEF2 是用于建立系统动态行为的仿真模型，IDEF3 也是一种单一目的的建模方法，如前所述。

第 2 章　IDEF3 总貌

我们从一个简单例子开始，来了解一下 IDEF3 的主要内容。例如，有一个零件需要油漆，然后送到下一工序进行装配，漆层需要覆盖得足够厚，以便在长期运行中起保护作用。其过程可描述如下：

零件来到车间，可以进行喷漆；液态的油漆用压缩空气喷向零件，覆盖了整个表面；然后进行覆盖情况的检侧，如果喷漆厚度足够，就转到下一工序，如果发现不满足要求，则返回起始点重新喷漆。这一过程可以用 IDEF3 过程流描述图表示，见图 3.2.1。

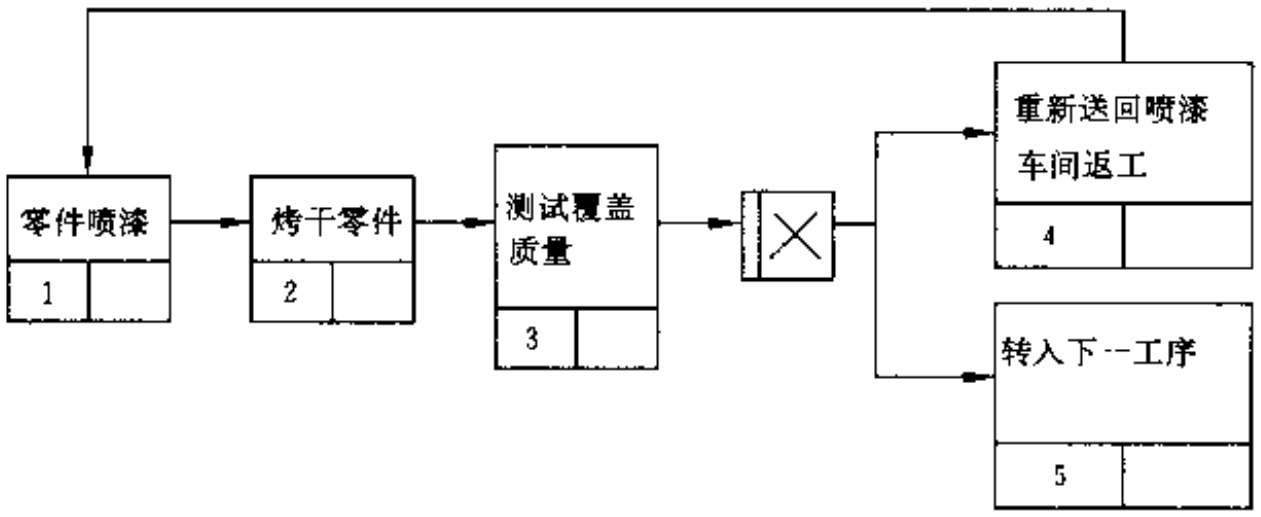


图 3.2.1　IDEF3 一例

每一个有编号的方盒子代表着可区分的一个信息小包，它可以是关于一个事件、一个决定、一个动作、或一个过程。这样的方盒子代表的是事件的类型，该类事件可用一个中性名词称之为“行为单元”(unit of behavior)来描述，简写为 UOB。联接盒子的箭头称为“联接”(link)用以表明所述过程之间的先后关系，或更一般地称之为“约束”。那个带有“X”记号的小盒子表示一个“交汇点”(junction)。交汇点表示在过程中的这样一个点，在这里，一个过程流通路，可分叉成几个通路，或者多个过程流通路，合并成一个。交汇点描述了过程的“流逻辑”(flow logic)。图 3.2.1 中的过程流图，代表了“零件喷漆过程”的场景描述，简称场景，在 IDEF3 中，场景限定了过程描述的上下文。

图 3.2.1 中的 IDEF3 图，代表了对油漆车间的以过程为中心的视图，它着眼于过程的出现及其次序。有时，为了便于组织场景的描述，还可用以对象为中心的视图，它更集中注意于参与活动的对象。例如，可以把油漆考虑成一个在过程中改变其状态的对象，然后用“对象状态转换网图”OSTN(object state transition network diagram)，来实现以对象为中心的视图，如图 3.2.2 所示。

应用这样一些基本图形和一些描述性的表格，就可以把领域专家对这一过程的了解，比较确切地描述清楚。下面先引入一些 IDEF3 中应用的基本概念。

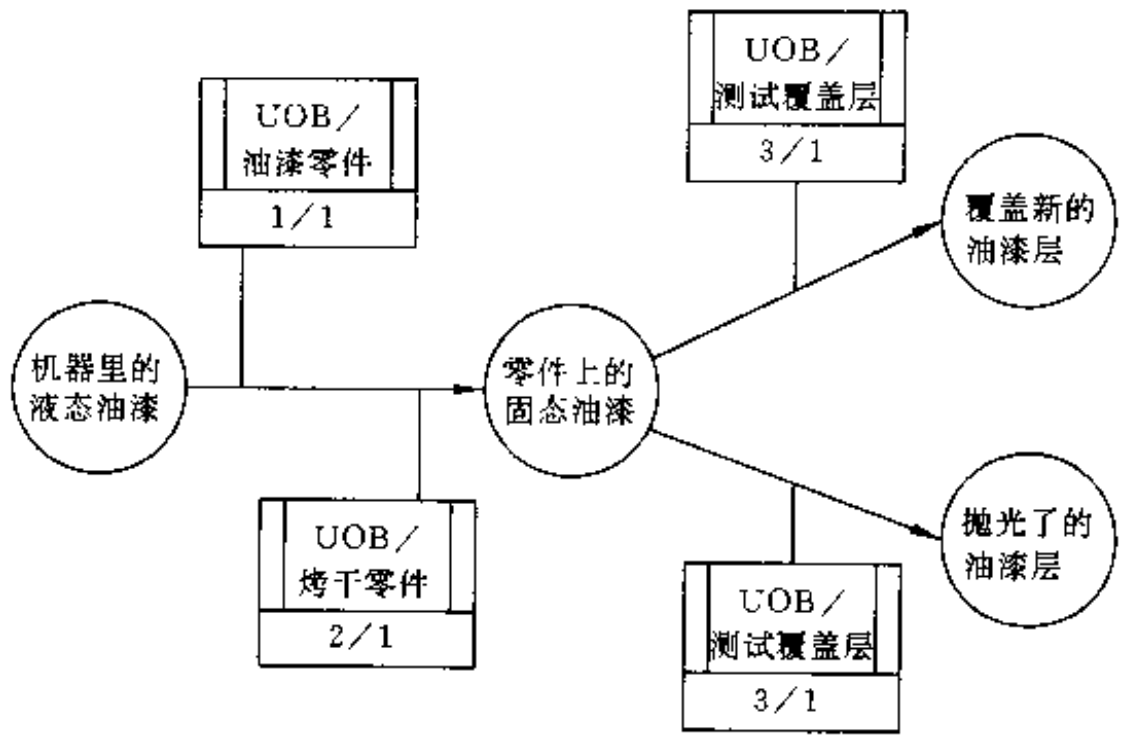


图 3.2.2　油漆过程的 OSTN 图

2.1　场景描述和对象

IDEF3 用两个基本组织结构——场景描述和对象来获取对过程的描述。“场景描述”可以被看作：(1)在一个组织中，需要用文件记录下来的一种特殊的重复出现的情景；(2)描述了由一个组织或系统阐明的一类典型问题的一组情况；(3)过程赖以发生的背景。

场景的主要作用，就是要把过程描述的前后关系确定下来，故必须重视给场景以恰当的命名。场景的名字通常为动词、动名词或动词短语。识别、抽取特征、再对场景描述命名是建立 IDEF3 描述中的重要步骤。命名合适与否直接关系到能否正确反映现实世界的情况。这里，可举几个典型过程流场景描述的名字，如：开发键盘孔板的铸模设计；处理用户意见；实施工程修改等。

“对象”则是任何物理的或概念的事物。这些事物，是领域中的参加者认识到的，或谈到的，那些发生在该领域中的过程描述的一部分。对象的识别、抽取特征和恰当命名，也是 IDEF3 的重要步骤。接下去，就是进行过程流描述和对象状态转换描述。

每个 IDEF3 描述可以有一个或多个场景，有一个或多个对象，它们组成了描述结构的各个部分。但是一个场景，可以有、也可以没有与之相联系的对象流图，这就是它可能尚未被细化。一个对象，也可能有、可能没有一张 OSTN 图与之相联系，但仍然是整个描述的一部分。

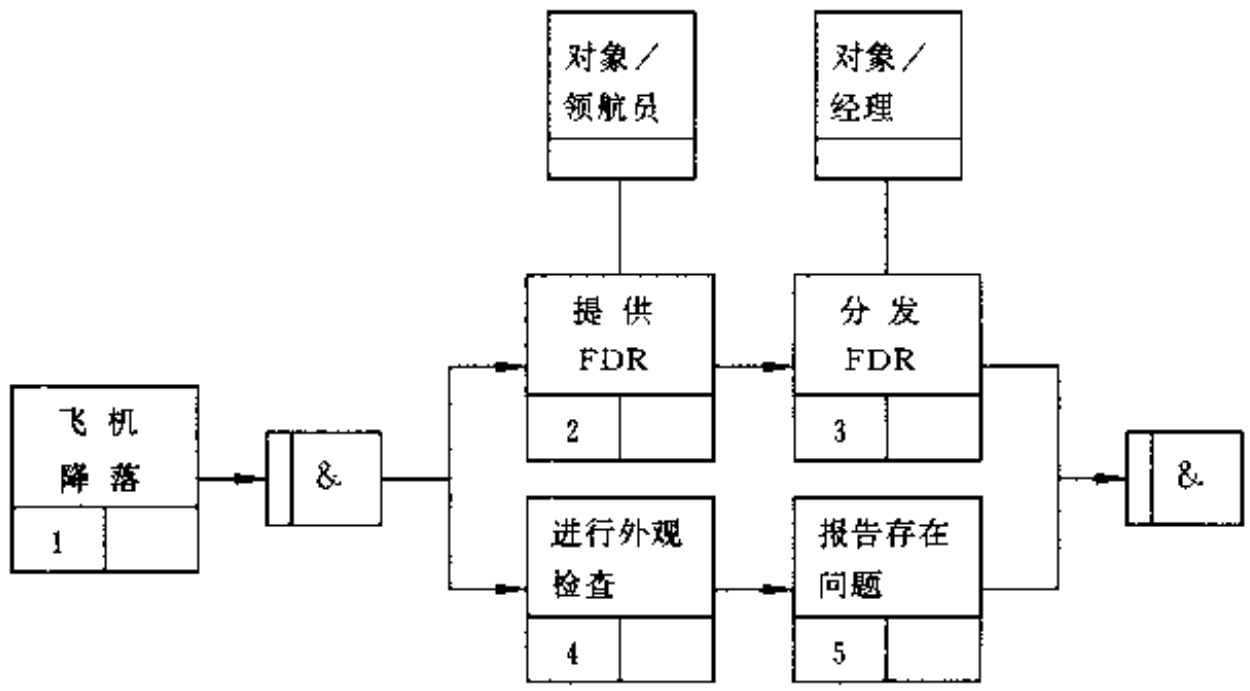
2.2　以过程为中心的视图：过程流图

IDEF3 中用过程流网 PFN(process flow network)作为获取、管理和显示以过程为中心的知识的主要工具，显示手段就是过程流图。这些图，包含了专家和分析员，对事件与

活动、参与这些事件的对象、以及驾驭这事件的行为的约束关系等的知识。这种描述，需要一些基本建造块，建造块必须有明确的语义。下面以一架飞机发现飞行中有问题，需要维修的过程为例，来说明过程流的描述方法。

图 3.2.3 是这一例子的过程流图，其中带编号的盒子为与场景相联系的“行为单元”(UOB)，每个 UOB 代表着一个现实世界的过程，它应包括：(1)表明这个 UOB 所指内容的一个名字(通常为动词或动词短语)；(2)参与这过程的对象的名字及其特性；(3)存在于对象之间的关系。UOB 之间的箭头称为“顺序联接”，它表示了过程之间时间上的前后顺序，这就是说，一个联接箭头源头上的 UOB 必须在其尾端 UOB 开始之前完成。譬如图 3.2.4 上半部的两个 UOB，领航员“提供 FDR(飞行异常记录)”的活动，必须在经理“分发 FDR”之前已经完成。另外，图 3.2.3 中左边带有一个小条的小盒子，称为“交汇点”，交汇点表示两个或两个以上的过程流在此处分开或者汇合，用以表示具有多道流的过程中的流逻辑，也就是“与”、“或”、“异或”等情况。有名字而没有标号的盒子称为“参照物”，它指向其他 IDEF3 的元素，如 UOB、场景描述、或对象，而表明在 IDEF3 过程流描述的其他地方，还有更详细的信息。至此，再回头来解释图 3.2.3 的例子。该例子是说，有一架飞机在飞行中发现了不正常的问题，一旦它已着陆了就有两个活动同时开始：带 & 符号的小盒表示逻辑“与”，一方面是要提供和处理飞行异常报告(FDR)，并行地开始的另一活动是进行外观检查及提出报告，两者都完成后才能开始下一步的具体维修措施，故右边有一个汇合的 & 结点。图中还有两个参照物，第一个叫做“领航员”，这里主要突出这一“对象”在 IDEF3 描述中的重要性，至于他是如何参与“提供 FDR”的情况，将在细化该 UOB 时说明。有时，还可能要再用 OSTN 来描述一些特殊对象。

场景 1：飞机飞行问题报告过程



FDR(flight discrepancy record)：飞行异常记录

图 3.2.3 过程流图举例

对于需要细化描述的 UOB，可作进一步“分解”(decomposition)，一个 UOB 的分解，实际就是形成另一张细节描述的过程流图。同一个 UOB 可以有不同观点的多种分解结果。如图 3.2.3 中的“飞机降落”从地面指挥塔的观点可以分解成图 3.2.4 的过程流图。

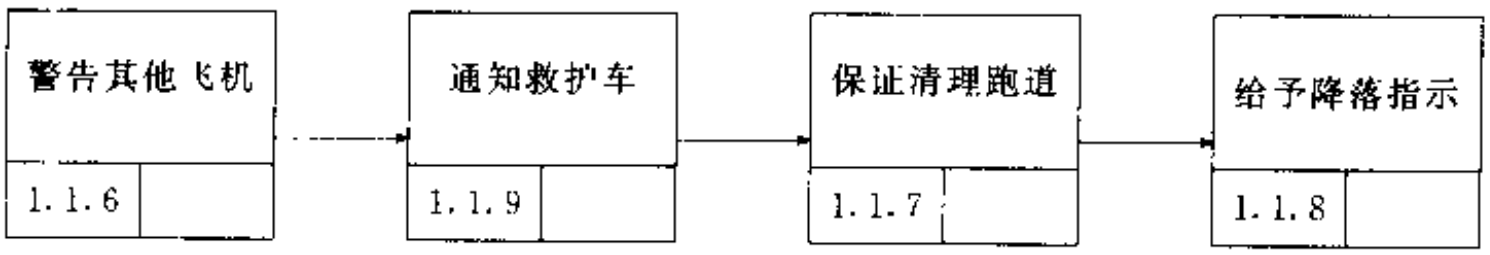


图 3.2.4 UOB 分解的例子

还可以画出从领航员的观点对 UOB“飞机降落”的分解图。其中节点号的前两位表示对 1 号 UOB 的第 1 种分解，最后一位则表示这个 UOB 本身的序号，这个序号与活动时序没有必然的联系。

2.3 以对象为中心的视图：对象状态转移网图

对象状态转移网(OSTN)是 IDEF3 中用以获取、管理和显示以对象为中心的知识的基本工具，OSTN 的显示就称为 OSTN 图。它用以表示一个对象在多种状态间的演进过程。图 3.2.5 为其基本概念。写了名字的大圆代表对象状态，外面的弧(实际形式可以是直线箭头)代表状态之间允许的转换，入口条件、状态描述和出口条件则记录在一种专用的表格中。

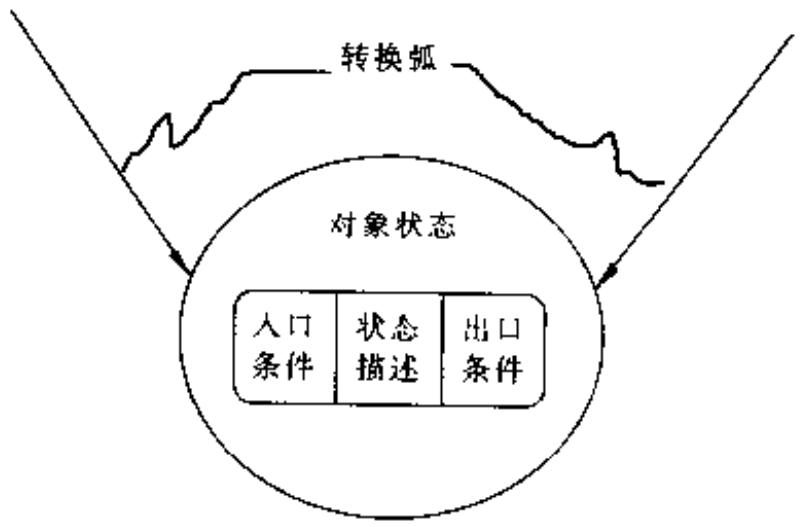


图 3.2.5 OSTN 图基本概念

每一张 OSTN 图集中于一个对象，首先要识别这一对象可能有的所有状态，实际生活中有的对象状态常常是连续演进的，但 OSTN 图则着眼于领域专家们感兴趣的可区分的状态。对每一个状态，OSTN 图要提供几项说明：①表征该状态特点的条件；②允许转入这个状态的条件(入口条件)；③为该对象转换出这个状态所要具备的条件(出口条件)。

下面以某公司采购单生成过程的 IDEF3 过程流图为例来说明这些条件，如图 3.2.6 所示。从生产计划处提出的“申请物料”活动，开始了这一物料订购过程。如果所需物料是现有库存中的一个项目，则只要向原有的供应商去订购就可以了，如果这是一种新的物料，需要去找新的供应源，那么就需要登广告招标、评标，最后发订单给选中的供应商。这里，交汇点小盒子中有一个 X 表示逻辑“异或”，表示在几个可能的通路中只有一个过程流通路会被选中。

在图 3.2.6 所示的过程流描述图中，采购单生成过程中的关键文件是“采购申请

表”。这个表要通过采购单 PO(purchase order)生成过程，转换成采购单(PO)，这一转换的 OSTN 图示于图 3.2.7。其中每一个圆表示一种可能的状态，每个状态还要有一张详细的表格，称为“对象状态描述”OSD(object state description)表，它用于获取附加信息，诸如对象从什么状态转移来、又要转移到什么状态去(入口、出口条件)，以及状态所规定的特征。这样，图 3.2.7 中状态“PO 草案”的 OSD 表就要规定由“采购申请表”状态转移到“PO 草案”状态的条件，图中的箭头就表示了由一个状态向另一状态的状态转移。边上有带子的名为“审批 PO”的盒子，则是一个“参照物”的例子，它涉及状态转移的条件，本例中就表示，“PO 草案”必须经过一个审批程序才能转移成为“批准的 PO”。

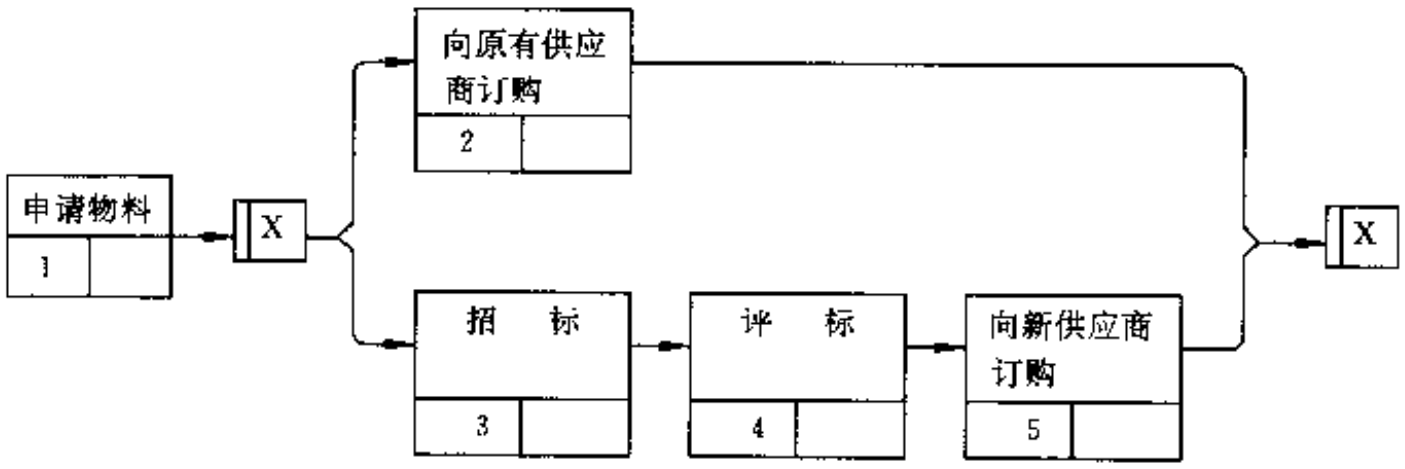


图 3.2.6 过程流描述图的例子

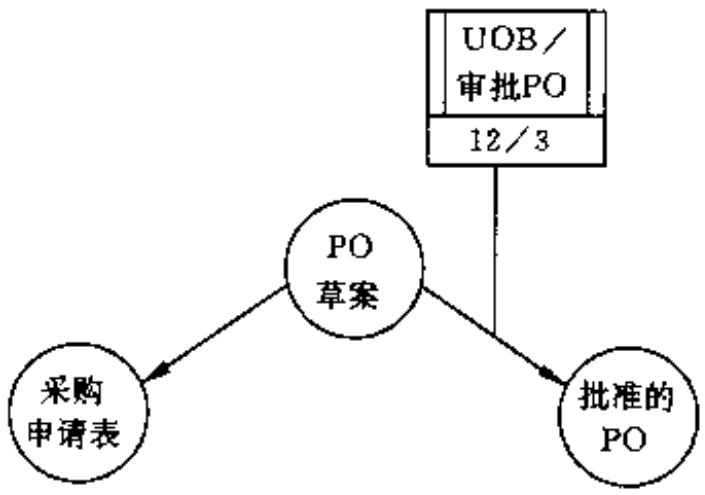


图 3.2.7 采购申请表的 OSTN 图

第 3 章　IDEF3 过程描述的基本元素

IDEF3 通过一些基本元素(或称建造块)的各种各样的组合,来描述丰富多彩的现实世界。它所用的两种方法以过程为中心和以对象为中心,并不是互相排斥的。它们所作的图,能够交叉参考互为补充。其中用作 IDEF3 过程流描述语言的基本语法元素有下列几种(如图 3.3.1):

① 行为单元 UOB(unit of behavior)

② 交汇点(junction)

③ 联接(link)

① 参照物(referent)

⑤ 细化说明(elaboration)

⑥ 分解(de composition)

一份 IDEF3 过程流描述是由一组过程流图和相应的细化说明文件所组成。过程流图包含了代表着基本建造块的符号所构成的图形语句。一张图显示了一组 UOB 盒子,它们代表了现实世界中的活动、动作、过程和操作,又用联接箭头把这些内容组合在一起,以反应其前后顺序(实线箭头)、用户定义的关系(虚线箭头)、或对象流(双头箭头)。过程出现的逻辑,则通过另一种符号——交汇点来描述;交汇点盒子可以表示多股过程流的汇总(扇入)或分发(扇出)。另外还有一些支持 IDEF3 图的语法元素,包括:①用以表示内容上前后相关的信息的盒子(参照物),②对 UOB 和联接的详细说明表格(细化说明表),③对其他图的参考(UOB 分解)。下面逐个介绍这些建造块。

3.1　UOB

行为单元 UOB 用以描述一个组织或一个复杂系统中“事情进行得怎样”的情况,它要用许多自然语言的概念,或者用下面这些概念以日常用语来描述“实际生活中发生了什么事”:

(1) 功能(function)

(2) 过程(process)

(3) 场景(scenario)

(4) 活动(activity)

(5) 操作(operation)

(6) 决策(decision)

(7) 动作(action)

(8) 事件(event)

(9) 步骤(procedure)

每一个概念都有其特定的行为含义,阐明了在一定的时空范围内,事情是如何进行的。

在图 3.3.1 中的 UOB 就是用一个有唯一标签的盒子来表示的,每个 UOB 要阐明与其相关的对象(客体)及其相互关系,即所谓 UOB 的"细化说明",要说明其与其他 UOB 的关系,即所谓 UOB 的"分解"。

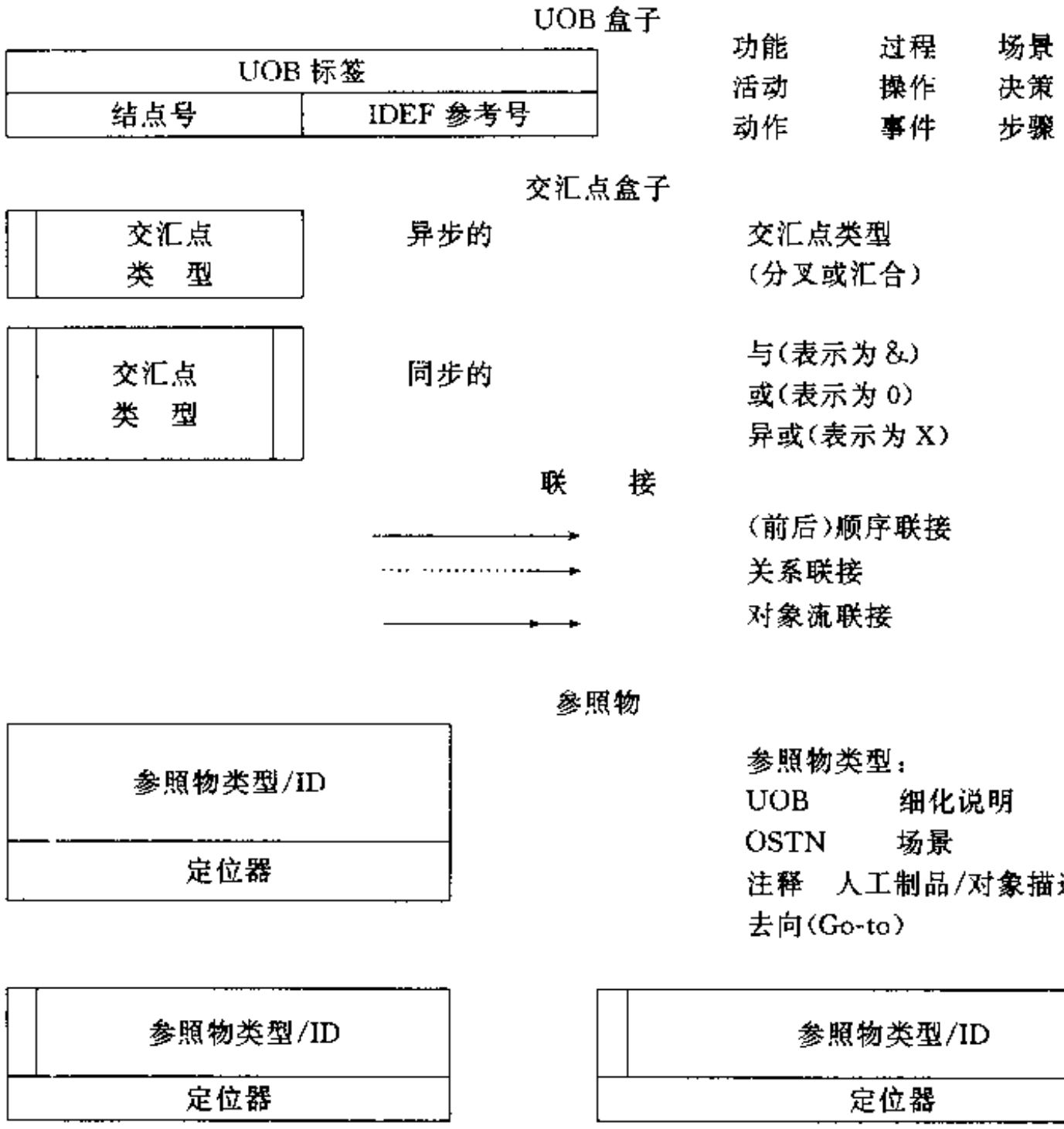


图 3.3.1 用于 IDEF3 过程描述图的符号

3.1.1 UOB 的细化说明

IDEF3 过程流图描述了用盒子表示的各种活动所组成的一个过程。但只有给出了 UOB 的细化说明,才能完全理解一个过程流。细化说明用图 3.3.2 所示的"细化说明文档",提供了实际生活中 UOB 的确定的特征。它认定了构成和控制一个 UOB 的对象、事

实和约束，并提供了 UOB 上下文的描述。然而，如果加上更多信息，就可用以理解复杂过程的 UOB。

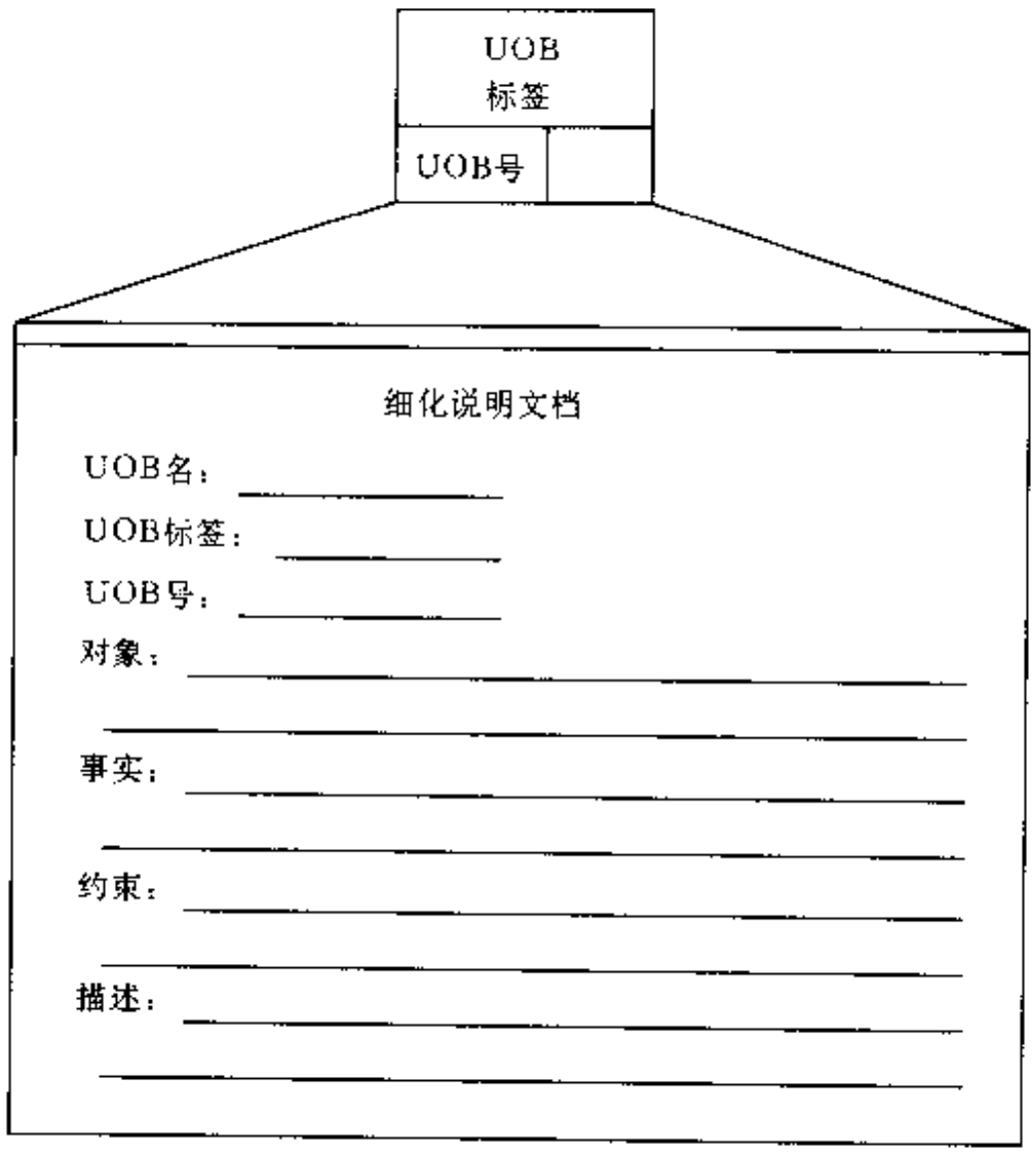


图 3.3.2　UOB 细化说明文档

这一文档各个区段的内容解释如下：

(1) 文档标识。这一段由所要描述的 UOB 的名字、标签和编号组成。这些内容唯一地标识了由本细化说明文档所要说明的 UOB。

(2) 对象(客体)。这一段列出了参与由这个 UOB 所描述过程的所有对象的名字。这些对象可以是物理的，也可以是概念的。在过程进行中，对象可以被创造、修改或清除。可以将对象分类成：代理、受影响的、参与者、或被创造的或被清除的对象。

① 代理：如果对象是 UOB 的“执行者”；

② 受影响的：如果在 UOB 活动的进行中对象被改变；

③ 参与者：这个对象是 UOB 描述的一部分，但必须没有任何因果关系或变换与之相联系；

① 被创造或被清除的：如果这对象是在 UOB 活动进行中被创造或被清除的。

(3) 事实：本段列出了对这个 UOB 或参与该 UOB 的一些对象的各种说法。细化说明中的“事实”，包括对象的特征(特性)和在该过程进行中对象之间所必须具有的关系；“事实表”也包括 UOB 的特性，如它的时间区段、出现频率或成本等。

(4) 约束：本段包括了一系列关于该 UOB 运作必须受到的限制的说法。约束表达了

一个 UOB 开始、继续或终止所必须满足的条件；约束是一组限制着或控制着一个 UOB 实例出现的事实或说法。约束和事实是密切关联着的。约束与事实的区分在于它包含了那些指出时间的或活动之间因果关系的字眼，例如："以前"、"正当"、"以后"、"从不"、"经常"……。

(5) 描述：本段为 UOB 保留了一个词汇表的条目(原文的描述)。典型的是，词汇表条目提供了对那些在对象、事实和约束表中已有信息的文字上的列举。

3.1.2 UOB 分解

"细化说明" 使我们对所掌握的过程知识结构化了。但是，如果过程太复杂，则可将它分成一些子过程，层次越低，抽象程度越低、越具体，这个过程称为"分解"，也就是用"分而治之"的原则来处理复杂问题。

"分解"提供了从不同知识来源和不同观点对同一过程进行建模的可能性，因此 IDEF3 允许同一个 UOB 有不同的分解，或"视图"。这一点在一个给定过程包含多功能的组织时特别有用。

从语法上来说，一个分解就是另一张 IDEF3 过程流图，所有 IDEF3 的建造块都可以用来构造一个分解。图 3.3.3 就是从处理合同的领域中取来的一个例子，进行分解。

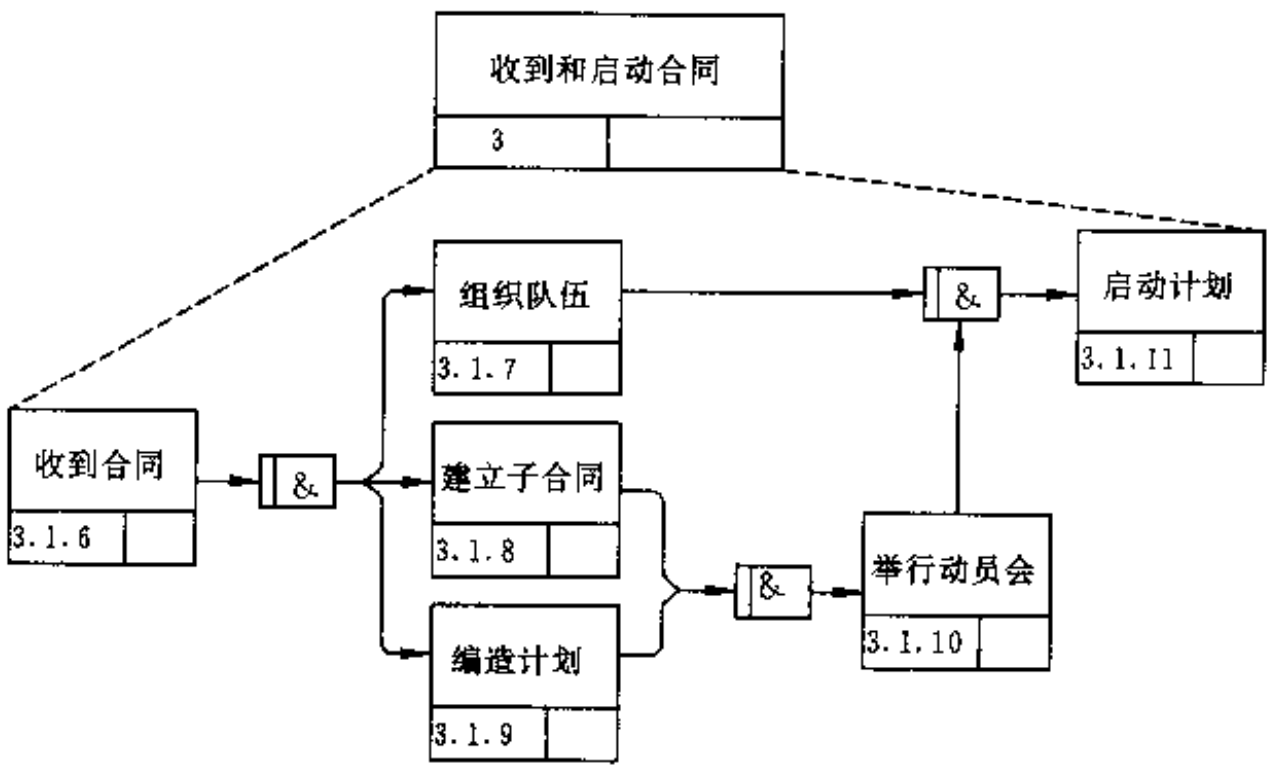


图 3.3.3 "收到和启动合同"的分解 3.1

被分解的 UOB"收到和启动合同"，称为父 UOB。它的每个分解就称为"子"分解。进而，就要给每个子分解一个标签和一个编号。一个分解中的 UOB 还可以有进一步的分解。

多视图分解还可以合称为"目的视图"。图 3.3.4 是 UOB"举行动员会"的目的视图的例子。这是从一个动员会的旁观者眼里的视图；然而，合同的项目经理会对这一过程有另一种看法，图 3.3.5 就是按项目经理的认识来进行的分解。

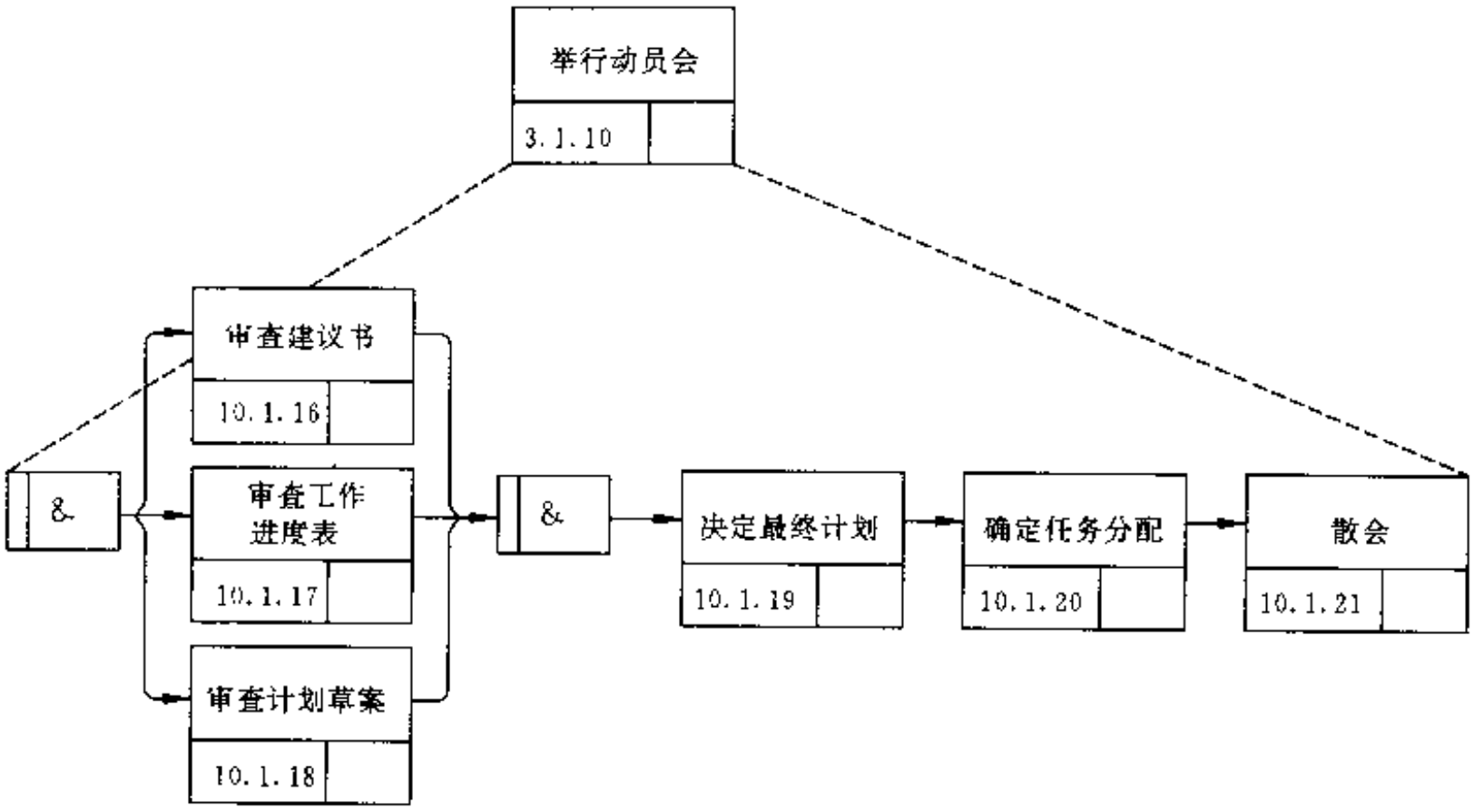


图 3.3.4 “举行动员会”UOB 的分解 10.1

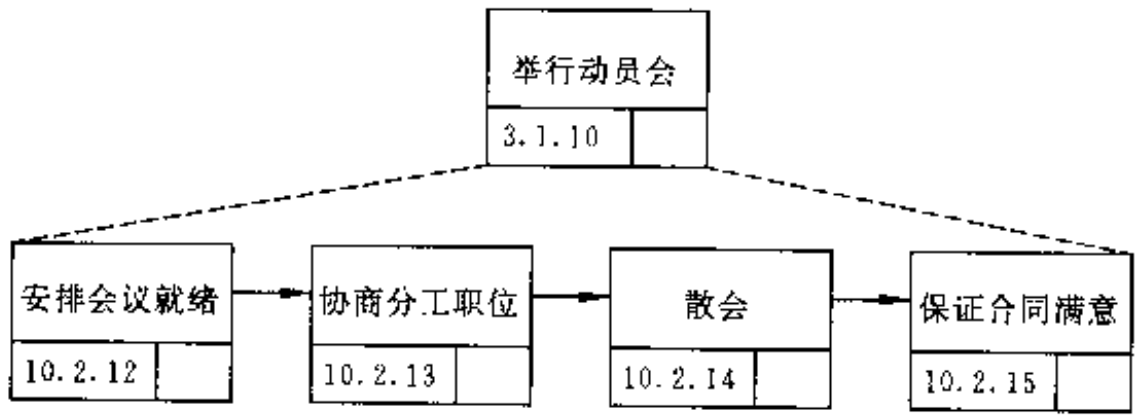


图 3.3.5 项目经理的视图分解

3.1.3 UOB 的编号方案

为了能明确地参照和跟踪 IDEF3 过程流描述，每个 UOB 必须给一个唯一的编号。UOB 的编号基本上是按创建和发现的顺序编排，而与进程描述的号无关。

从图 3.3.6 所示的方案中可以看出，每个 UOB 只能有一个号，即整个复杂系统所有的 UOB(不论层次)顺序编号下来的那个号；当进行分解时，其第一位一定要写成其父 UOB 的编号，跟着一个句点“.”；第二位则表明它是第几种分解，再跟着一个句点；第三位就是在整个系统中的序号。这样，从“整个系统的序号”表明其唯一性，从头两位则可追踪其分解过程的父 UOB。由图亦可看出顺序号没有从左到右排列的要求，只是根据其创建的先后赋予编号。

当整个模型由多人同时分头建模时，则要分配编号的区段，如张三用 1～100，李四用 101～200 等；顺序号可以空缺，但绝不许重复。最后组长要汇总作全面检查，保证编号的唯一性。

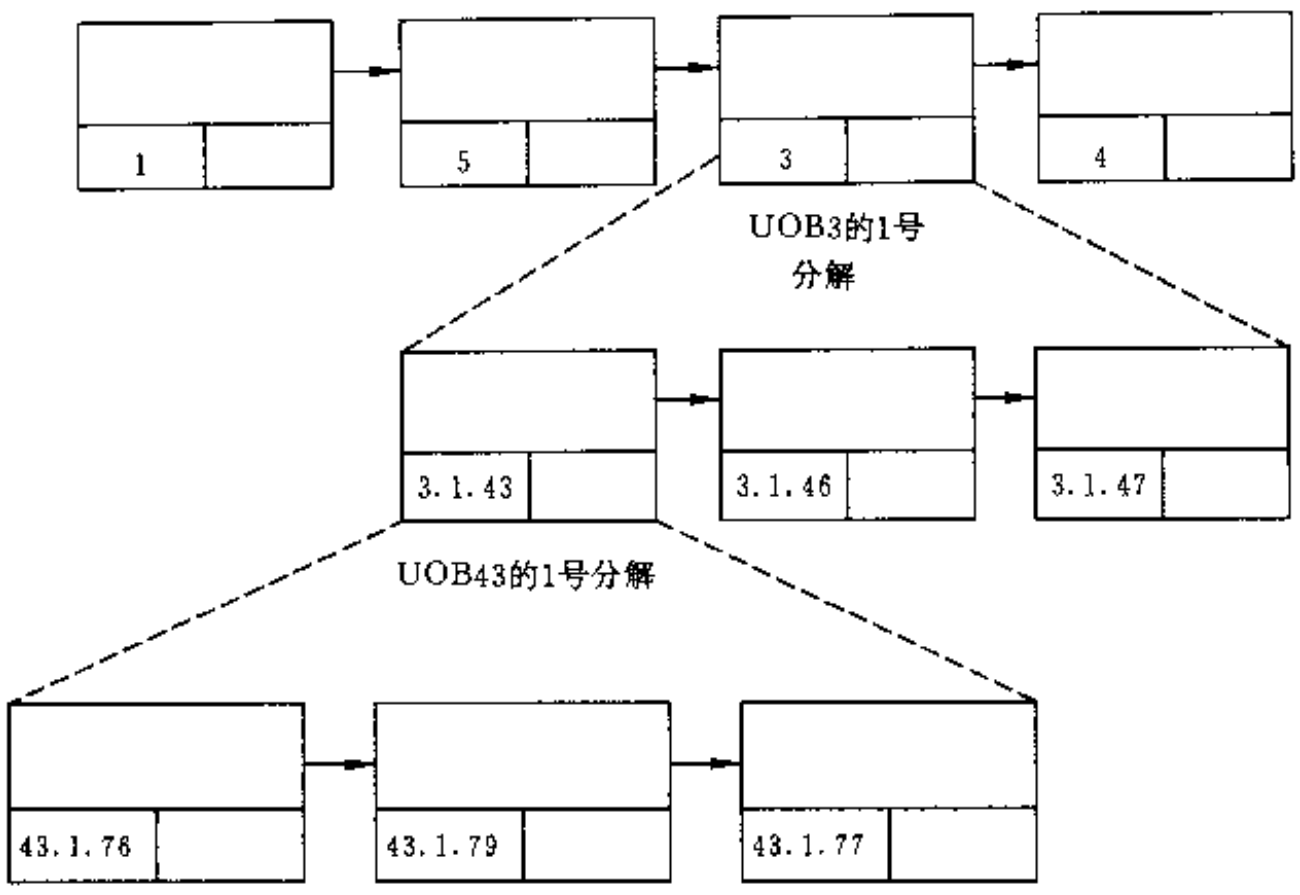


图 3.3.6　UOB 编号方案

3.1.4　部分描述

在 IDEF3 的过程流图中，UOB 盒子都是用箭头“联接”在一起的，但有时会发现有的 UOB 无法与其他 UOB 联接，如图 3.3.7 所示例子。其中 UOB“项目经理对照原计划检查进度”与其他 UOB 没有联系。这种现象可能有两种原因，一是这确是实际情况，另一是知识还不完整，对联接关系未搞清。本例属于前者，UOB“项目经理对照原计划检查进度”所以会成为这张图的一部分，原因在于它和其他 UOB 共享了一个对象——“原(项目)计划”，其他 UOB 是按照“原(项目)计划”在进行活动，而这一 UOB 则超脱于那些活动之上，在一定的时间去进行检查，这种现象称为“部分描述”。IDEF3 允许存在部分描述，便于建模者不要盲目地去追求完整性而加一些牵强附会的联接。但是，如果本身是由于了解过程不够，知之不深而造成的缺乏联接，还是应该多追问一些问题，多向专家了解过程而改进联接关系的。

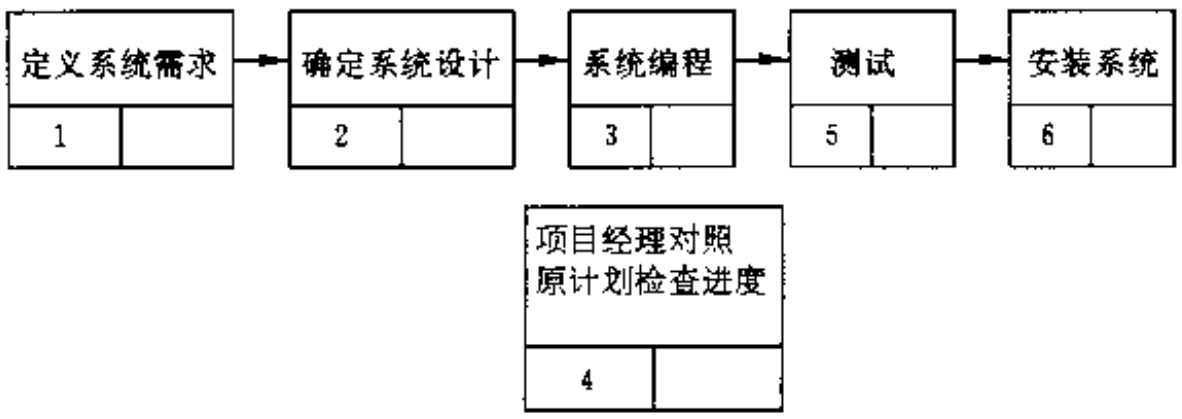


图 3.3.7　不联接 UOB 的例子

3.2 联　　接

“联接”是把 IDEF3 的一些建造块组合在一起的粘接剂，它可以进一步阐明一些约束条件和各成份之间的关系。联接关系的类型可以有：时间的、逻辑的、因果的、自然的和传统的等。联接说明文档可以描述清楚每个联接的细节。

联接箭头的起始和终止，可以画在 UOB 盒子或交汇点符号的任何部位。但为了增加过程图的可读性，最好是从左到右、从上到下地表示对象流（物理的或信息的）的方向或时间的顺序。

3.2.1 联接类型

IDEF3 有三种联接类型如图 3.3.8 所示。

“顺序联接”用来表示 UOB 之间时间上的前后关系，是 IDEF3 过程流图中用得最多的联接，它用实线箭头表示。其含义为：联接起始端（箭尾）的 UOB 实例必须在联接终止端（箭头）的 UOB 实例开始运作前完成，或者说，第一个 UOB 的 完成才使得第二个 UOB 可以开始工作。

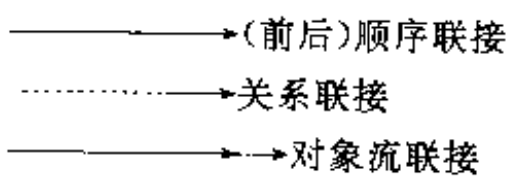


图 3.3.8　IDEF3 联接类型

“关系联接”用虚线箭头表示，它没有预先定义的语义，为此它们常被看作是“用户定义的联接”。它只是为了强调在两个或多个 UOB 之间存在着某种关系，这种关系或约束将在下一节的“联接说明文档”中阐述。图 3.3.9 中在 UOB“商谈修改”和 UOB“接受建议书”之间有一个虚线箭头，就表示这种关系联接。形式上它们是并行出现的，但这两个密切相关的活动之间显然有着明显的关联。

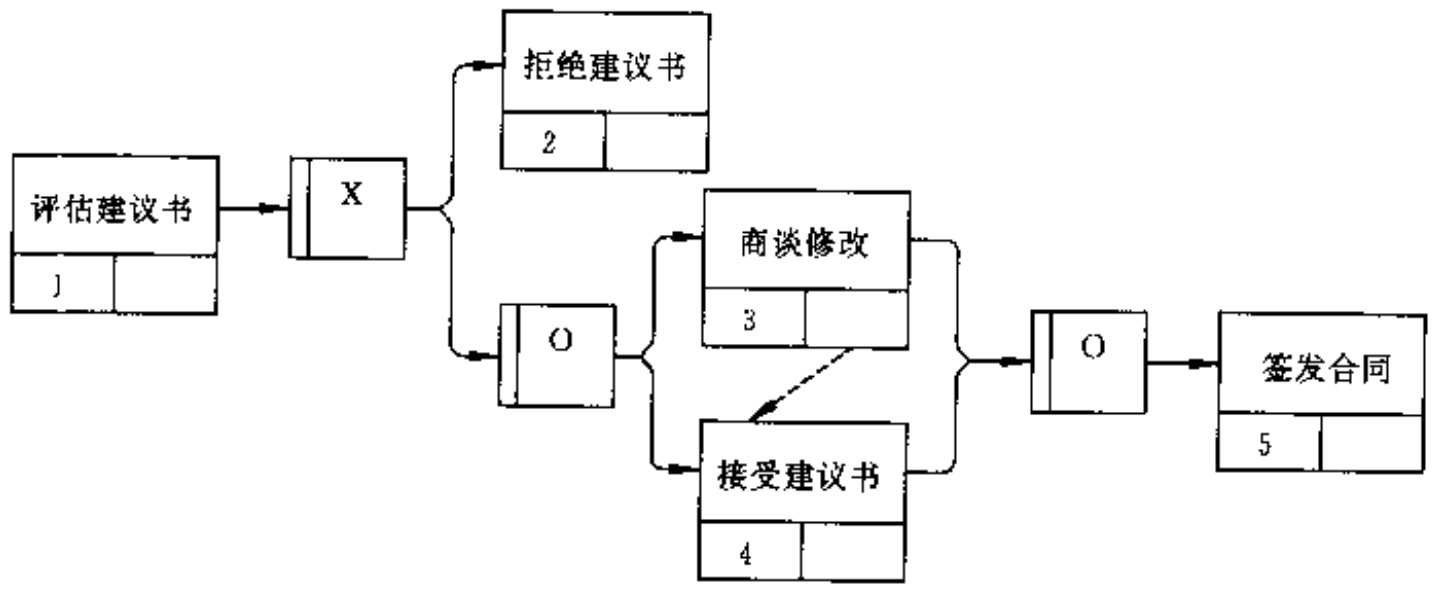


图 3.3.9　关系联接的例子

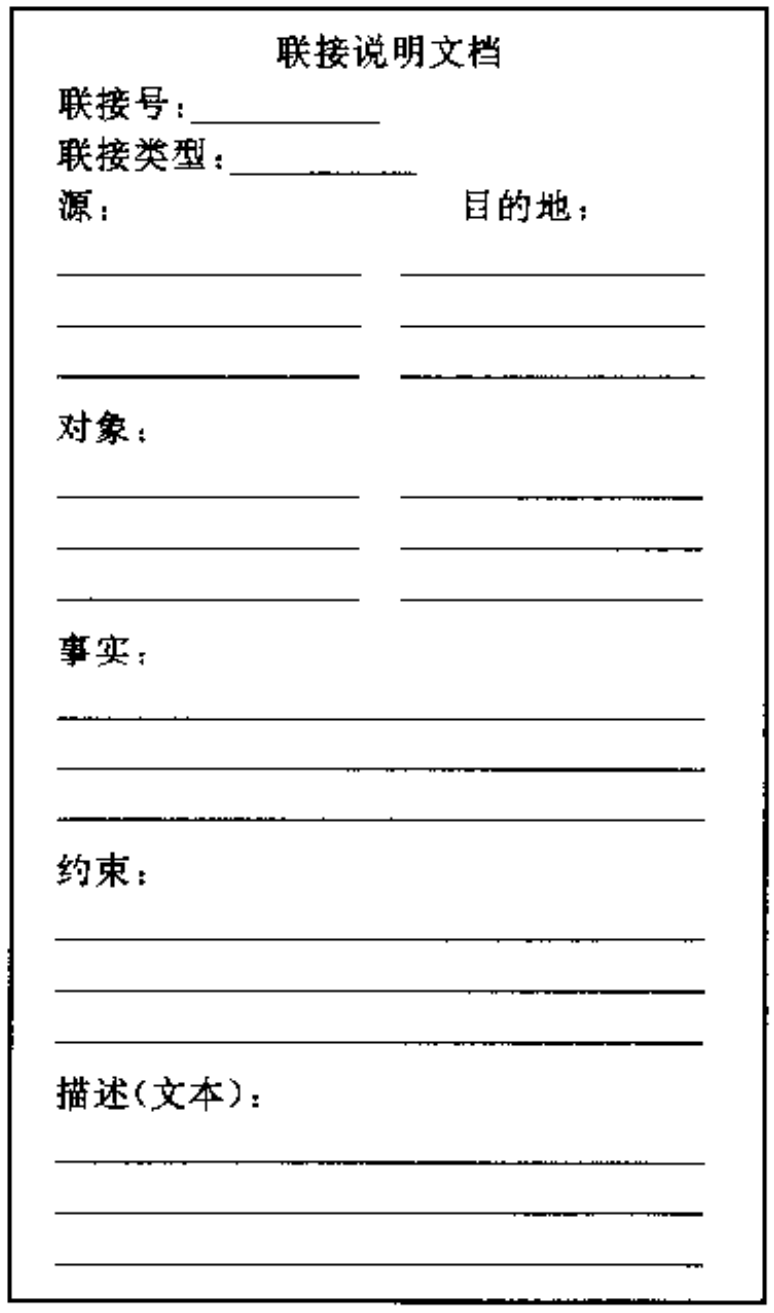
“对象流联接”提供了一种强调一个对象参与到两个 UOB 中的机制。它用一个有两个箭头的实线箭头表示。首先，它像顺序联接一样表示了时间上的前后关系，其次，它强调了有一个对象从第一个 UOB 流向了第二个 UOB。这里并不意味着如果不用对象流联接，

两个 UOB 就不可能共享一个对象，而只是一种必要情况下的强调。譬如由第一个 UOB 所创造的对象，对第三个 UOB 过程的完成起关键作用，就要用这种表示。

3.2.2 联接说明文档

关系和对象流联接被用来传达比简单的 UOB 之间时间顺序更多的信息，把关系和对象流联接上的特殊约束记录在一张表上，叫做“联接说明文档”（见图 3.3.10）。这一文档在形式和目的上都与 UOB 细化说明文档很类似。下面是其各区段的说明：

（1）文档标识：联接号唯一地标识了所涉及的那个联接及其类型。此外，文档标识还包含了标识联接类型的区段。


联接说明文档

联接号：_____

联接类型：_____

源：　　目的地：

对象：

事实：

约束：

描述（文本）：

图 3.3.10　联接说明文档

（2）源：联接的源的名字，它是指联接开始的那个 IDEF3 盒子；多个源的情况通常出现在联接终止于扇入交汇点（见 3.3 节）。

（3）目的地：联接的目的地的名字。它是指联接终止的那个 IDEF3 盒子；多个目的地通常出现于联接由扇出交汇点起始的情况。

（4）对象：所有参与该联接表达的关系中有意义的对象。这些对象可以包括在联系的源和目的地中的对象。

(5) 事实:参与该联接表达的关系的对象所具有的有意义的特征。这里既包括与该联接相关对象的特性,也包括已知存在于这些对象间的关系。

(6) 约束:联接运行所受限制的特征化。

(7) 描述:与该联接相关联的描述性词汇表。文档中任何在逻辑上不适合于其他区段的描述性信息都可放在这里。

3.2.3 联接编号

联接也要像 UOB 编号那样给出其唯一的顺序编号,记作 L1,L2…(L 表示 link)。与 UOB 的做法一样,可以将号码的不同区段分给不同的建模人员。编号的做法对于描述模型的分叉特别有用,因为很容易根据联接编号来描述分叉逻辑。联接编号在过程流图上是否表示是任选的。

3.3 交 汇 点

IDEF3 中引入交汇点这一机制说明各过程分支间的逻辑关系。它借助于类型多样的交汇点来获取现实世界过程中各分支的语义。交汇点完成对以下过程的描述:(1)一个过程可分叉或分为两个以上的过程路径;(2) 两个或两个以上的分叉汇合为一个过程路径。交汇点简化了对各多路径过程间顺序或时间关系的描述获取,但它并不是唯一的手段。如果描述利用具有预定义的交汇点符号的语义,进行清晰或准确描述的话,分析员就得使用专门定义的关系联接。

3.3.1 交汇点类型

交汇点可以从不同角度来分类。首先,依照逻辑语义含义可分为:"与(&)","或(O)"及"异或(X)"。基于过程描述中逻辑表述上的分叉或汇合,又可进一步划分为"扇入"或"扇出"型交汇点。同时,也可根据所关联的行为单元时间上的同步或异步关系进行分类。图 3.3.11 对上面各种分类方法之间的关系,以图形方式进行了概括。

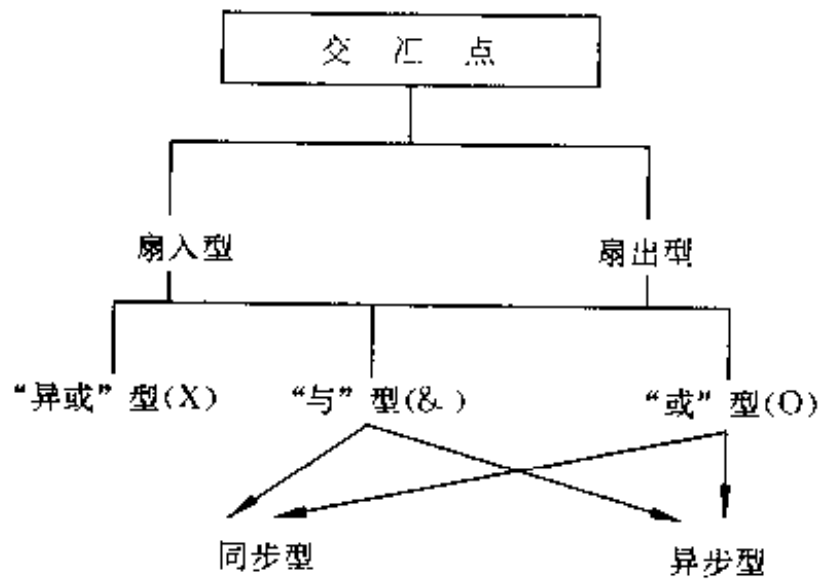


图 3.3.11 交汇点类型结构

3.3.1.1 “与”、“或”及“异或”型交汇点

“与”、“或”及“异或”这种分类方式，对通过交汇点的多分支过程，提供了一种标准的逻辑解释。进入或离开一个“与”型交汇点的所有行为单元，必须已启动或完成。“异或”型交汇点则意味着通过的行为单元集合中只有一条路径可以启动或完成。“或”型交汇点提供了对行为单元集合中某些路径的选择自由，也就是说，集合中的一个或多个行为单元要在通过交汇点时启动(扇出时)或完成(扇入时)。

3.3.1.2 “扇入”型与“扇出”型交汇点

IDEF3 图中所表述的过程往往很复杂，具有多条分支，这些分支路径产生于(扇入情况下)或终止于(扇出情况下)一个交汇点。“扇入”型交汇点是指将一组不同过程分支汇合起来的交汇点，图中两条以上的联接将它们与“扇入”型交汇点连接起来。“扇入”型交汇点依据结尾过程的逻辑或时序关系进行划分，见图 3.3.12。“扇出”型交汇点在流图中代表着过程分离或分叉成为一组过程路径集合，从它引出几条顺序联接。“扇出”型交汇点所包含的通常语义是指这些扇出过程分支要在满足交汇点约束条件下进行初始化，这种初始化约束条件的确切本质取决于“扇出”型交汇点的类型。“扇出”型交汇点类型及其相关语义说明见图 3.3.13。

扇入型交汇点类型	语义说明
& — “异步的与”型	交汇点前的所有过程分支必须已完成
& —“同步的与”型	交汇点前的所有过程分支必须同时完成
O — “异步的或”型	交汇点前的过程分支中的一条或多条已完成
O — “同步的或”型	交汇点前的过程分支中的一条或多条同时完成
X — “异或”型	交汇点前的过程分支中只能有一条完成

图 3.3.12　各种类型的“扇入型”交汇点及其语义

扇入型交汇点类型	语义说明
& — “异步的与”型	交汇点后的所有过程分支都触发执行已完成
& —“同步的与”型	交汇点后的所有过程分支必须同时触发执行
O — “异步的或”型	交汇点后的过程分支中的一条或多条可触发执行
O — “同步的或”型	交汇点后的过程分支中的一条或多条同时触发执行
X — “异或”型	交汇点后的过程分支中只能有一条触发执行

图 3.3.13　各种类型的“扇出型”交汇点及其语义

3.3.1.3 “同步”与“异步”型交汇点

交汇点的同步性指明了汇合于它或从它分离出来的各过程分支路径间的相对时序关系。这种解释是与 IDEF3 图的激活(或例示说明)的看法相联接的,也就是遍历一下 IDEF3 图描述其行为的那个过程。比如,当一个行为单元激活,它“启动”,“执行”和“完成”,在一个行为单元完毕后,过程的激活继续沿 IDEF3 流图中这个 UOB 后面的那个元素进行。“同步扇入”型交汇点,要求汇入的各过程路径,必须要在交汇点后的行为单元激活前同时完成。“异步扇入”型交汇点,则没有这些路径完成时序上的约束。

总之,一个交汇点的语义取决于它是否为:(1)“与”,“或”或“异或”;(2)“扇入”型或“扇出”型;(3)“同步”型或“异步”型。在 3.3.1.4 节,将对这些不同类型交汇点组合使用的语义进行详细说明。

3.3.1.4 交汇点语义

正确使用与理解交汇点的关键,是能否认识到,过程流图是对已收集到的“过程中发生了什么事情”的一组图形语言的说明。图 3.3.12 与 3.3.13 已对 IDEF3 各交汇点符号的语义进行了总结。这些语义包括了采用逻辑操作符,例示说明与时序控制的约束元素。下面进一步解释一下。

“扇出的与”型交汇点所关联的语义,说明从一个扇出的交汇点引出的所有过程分支都将发生。同理,对于一个“扇入的与”型交汇点,所有汇入的过程,也要全部终止或完成。运用这两种类型是为了在通过交汇点的各过程激活的相关时序上施加额外的约束。从一个“同步的扇出的与”型交汇点引出的各过程,要同时发生;若是“异步的扇出的与”型交汇点,各过程分支则可以以任意顺序启动。对“扇入的与”型交汇点,上述条件只是略有不同。汇入一个“异步的扇入的与”型交汇点的各过程,要全部完成,但不必有某种特定次序或时序限制。“同步的扇入的与”型交汇点则要求所有汇入过程同时完成。

“异步的扇出的或”型交汇点的相关语义表明,交汇点引出的一个或多个过程可以任何次序启动。对于同步性的,尽管可有任意数目的过程分支在交汇点上启动,但这些分支必须同时发生。“扇入的或”型交汇点的一般语义是指交汇点前的各过程分支中,至少有一条要在通过它之前完成。对一个“异步的扇入的或”型交汇点,各过程的终止的相关时序,并不重要,而“同步的”交汇点则要同时完成。

“异或”型的交汇点要求,从其引出的过程分支中,只能有一个会启动。从图 3.3.12 和 3.3.13 可以看出,没有提供对“同步的异或”型交汇点定义。这是因为“异或”型交汇点所涉及的,只是其前后的一个行为单元,避免对任何同步类型的需求。

3.3.2 交汇点组合

交汇点组合使用后,其语义解释依然建立在其中涉及到的各交汇点的类型及基本语义上。这一节讲述一下实践中使用的几个例子。

图 3.3.14 所显示的是最常用的一种交汇点类型:“异步的与”型交汇点。“接受提议”行为单元完成后,紧跟着成本与技术评估。在图中的过程描述中,成本与技术评估都必须

在“审批合同”前完成，但两者之间并未规定时序关系。它们都必须在“接受提议”和“审批合同”行为单元之间执行，但启动与完成上并没有时间约束。

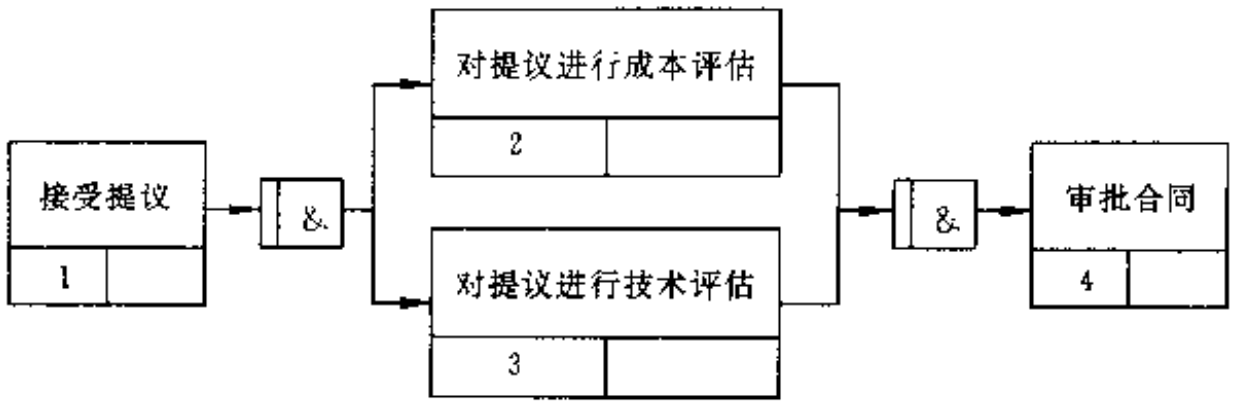


图 3.3.14 “异步的与”型交汇点示例

与图 3.3.15 相比，后者利用“同步的与”型交汇点来进行过程描述，其中的成本与技术评估必须同时发生，但未必同时结束。然而，如果已有结构化规则要求两者也同时结束的话，图 3.3.15 就得采用“同步的扇入的与”型交汇点。

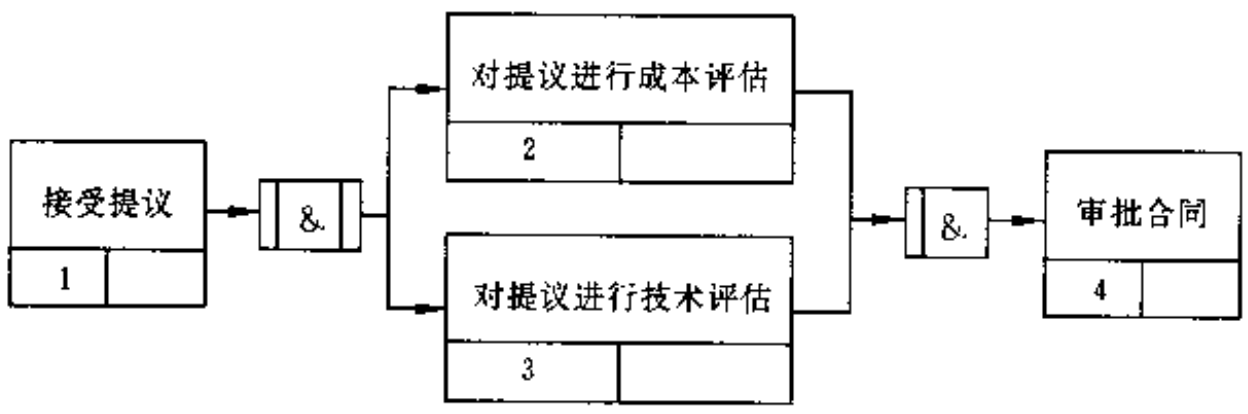


图 3.3.15 “同步的与”型交汇点示例

图 3.3.16 所表述的是建议书评估过程。流图描述说明，在评估之后，可以拒绝建议，协商修改，完全接受，或采取折衷方案。

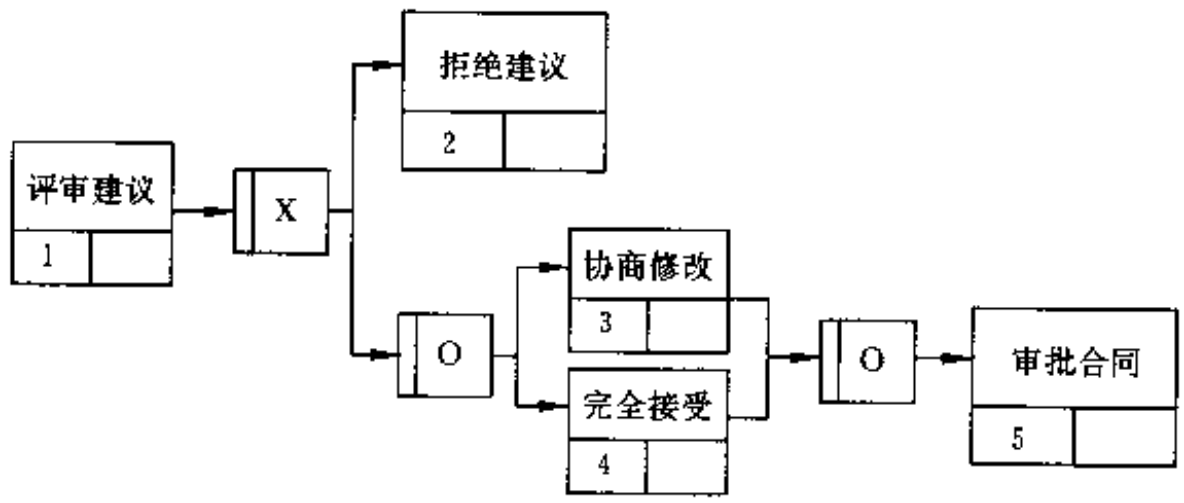


图 3.3.16 “异步的或”型交汇点示例

图 3.3.16 的进程描述中，“拒绝建议”是一个终止性活动，执行剩下两者中的一个或全部则会出现“审批合同”。其中的关系联接表明，“协商修改”与“完全接受”之间存在着关

联。另外，上述描述仍是不完全的，因为它并未说明当协商不成功时的事件或行为。举个例子，大多数情况下，合同的审批取决于协议双方对其中关键条款的接受程度，这可能要求协议双方重新提交建议，作为协商进程中的一部分。类似这样的附加信息，在 IDEF3 中，既可以作为对当前流图的附加部分，或者很容易在图中行为单元 UOB3 的分解描述中进行表述。

3.3.3 交汇点的编号方案

为明确地参照 IDEF3 流图中的各交汇点，就要提供 IDEF3 交汇点的识别方案。回顾前文，各联接都赋以“L”开头的唯一编号。交汇点也遵循相同的编号方案，只是各交汇点参考编号以字母“J”开头。这样，在一个 IDEF3 过程流描述中，就会出现 J1，J2，…Jn.

3.4 参 照 物

参照物可使 IDEF3 分析员进行如下工作：

(1) 在流图布置中可以展开成多页，或者成环状返回；

(2) 借助以前定义过的行为单元，不必重复，来说明过程中的特定时刻发生的以前定义过的行为单元的另一个实例(不用闭环返回)；

(3) 强调在一个行为单元中个别对象或关系的参与；

(4) 将参考数据或对象的特定例子联系起来(如屏幕布置)；

(5) 将专门的约束集合与交汇点关联起来，也就是，给交汇点以细化说明，其中包括额外的事实、约束、或描述交汇点如何工作的决策逻辑；

(6) 在过程流图与 OSTN 图间建立参考或联接。

参照物的图形符号如图 3.3.17 所示。IDEF3 的初学者会发现使用参照物可以代替交汇点类型、破折线箭头、或约束性语言说明，很容易地表述思想或概念。

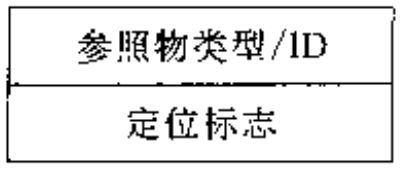
参照物符号语法允许有三种基本类型，见图 3.3.18。最常用的无条件型参照物，它可以指一个行为单元、细化说明、交汇点或对象。每种类型的参照物都可用于过程流图或 OSTN 图中。经验告诉我们，无条件型参照物常用于过程流图，而异步型、同步型参照物则在 OSTN 图中使用比较广泛。

同步型与异步型参照物之间的区别，主要在于它们所参照的元素的相对时间关系。使用异步型参照物说明参考元素只需在中心元素(即产生该参照物的 IDEF3 元素)前启动。而同步型的情况则是要在那之前启动与执行完毕。下一节就对参照物符号的不同格式的语义进行概述。

1 无条件型参照物

若参照物类型为“GO-TO”，并且识别号(ID)为行为单元编号，过程流中下面所进行的就是所参考的行为单元。这种情况通常用于过程流中编写循环。

若参照物类型为“GO-TO”，并且 ID 为交汇点参考编号，过程流接下来执行所参考交汇点之后的行为单元。



参照物类型：
- 行为单元：图表页中或之外的一个行为单元
- 交汇点：一个特定的交汇点
- 对象：参照物所关联的行为单元中的一个感兴趣的对象
- 细化说明：一份细化说明(通常用于对参照物与交汇点的关联进行说明)
- 场景：OSTN 图中某对象在状态转换前必须完成的场景
- 注释：对参照物所关联的 IDEF3 元素，由使用者定义的额外的信息
- OSTN：OSTN 图中某对象在状态转换前所必须完成的状态转换网络
- GO-TO：过程将要转换而成的 IDEF3 元素(如，转至某行为单元，然后从此点继续进行)

ID：
行为单元标签
交汇点类型(&,O,X)
空白(如果参照物指向细化说明)
对象名称
OSTN 标签
场景名称

定位标志：行为单元号，场景号，交汇点号，OSTN 号，或空白。对于行为单元或交汇点类型的定位标志，它应包括该 ID 出现的场景号或分解图号。

图 3.3.17　参照物图符结构

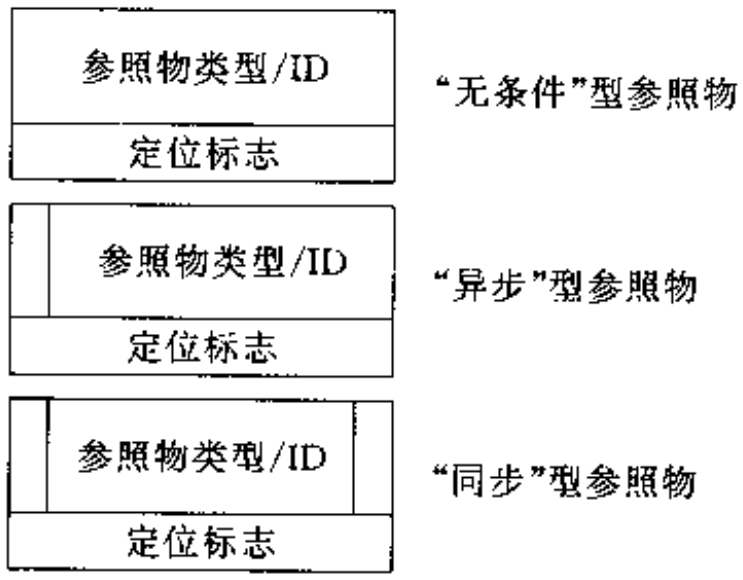


图 3.3.18　参照物图符句法

若参照物类型为"行为单元"(UOB)，并且 ID 为行为单元标签，这就意味着以前定义的行为单元的另一个实例在过程流的特定点上发生(没有闭环返回)。这种情况用来获取发生在相关内容之外的断言过程。

若参照物类型为"细化说明"(elab.)，并且 ID 为交汇点参考编号，这表明参考交汇点附带有一份细化说明。

若参照物类型为"对象"(object)，并且 ID 为对象名称，这表明命名的对象参与到参照物所指向的行为单元、联接或交汇点。这种参照物类型通常用来表示被传送对象的接受

者，好像一份设计文件被分发到许多不同的部门去的情况。

若参照物类型为“OSTN”，并且 ID 为 OSTN 标签，如果这用在过程流图中，表示一个行为单元的完成取决于经历该 OSTN 图中某些状态的对象。如果用在 OSTN 图中，这意味着状态转换取决于经历被参考的 OSTN 图中某些状态的对象（见 3.3.6 节）。

若参照物类型为“场景”（scenario），ID 必须是场景名称。如果它用在过程流图中，表示过程流下一步要做的是所参考场景的激活的情况。如果是用于 OSTN 图中，状态转换取决于所参考场景的一次激活（见 3.3.6 节）。

若参照物类型为“注释”（note），ID 应是使用者自己构造的，指向一组额外信息的参考号。他希望这些信息与注释所属的特定 IDEF3 建模元素有关联。这种参照物类型可将说明、文本、屏幕布置、评论等与描述联系起来。

2 异步型参照物

若参照物类型为“行为单元”，ID 必须是行为单元标签。这意味着一个以前定义过的行为单元的另一个实例在过程的特定点上发生（没有循环节点）。如果用于 OSTN 图中，参考行为单元的激活必须在状态转换前启动（见 3.6 节）。

若参照物类型为“OSTN”，ID 必须是 OSTN 标签。如果用在过程流图中，表示一个行为单元的完成取决于经过该张 OSTN 图开始状态转换的对象。如果用于 OSTN 图中，表示对象在状态转换前必须通过所参考的 OSTN 图中那个状态开始转换（见 3.6 节）。

若参照物类型为“场景”，ID 必须是场景名称。如果用在过程流图中，表示过程流中下一步骤，就是所参考场景的激活的一次实例。如果用于 OSTN 图中，表示参考场景的激活必须在状态转换前开始（见 3.6 节）。

3 同步型参照物

若参照物类型为“行为单元”，ID 必须是行为单元标签。这意味着一个以前定义过的行为单元的另外一个实例发生在这个过程中的某一特定时刻（无循环）。如果用于 OSTN 图中，所参考行为单元的激活必须在状态转换前启动和完成（见 3.6 节）。

若参照物类型为“OSTN”，ID 必须是 OSTN 标签。如果用于过程流图中，表示一个行为单元的完成取决于经过该 OSTN 图中状态转换的对象。如果用于 OSTN 图中，表示在状态转换开始前，对象必须经过所参考的 OSTN 图中的那些状态才转换（见 3.6 节）。

若参照物类型为“场景”，ID 必须是场景名称。如果用在过程流图中，表示过程流中下一步骤就是所参考场景的激活的一次实例。如果用于 OSTN 图中，表示参考场景的激活必须在状态转换前开始（见 3.6 节）。

值得注意的是，使用参照物可以避免流图中的混乱，提供数据或信息，或作为帮助读者理解的注释。图 3.3.19 说明一个参照物如何将一些专门约束集合与交汇点联系起来。这种描述说明对某些条件，可能需要返回到行为单元“PMAA”（perform mission area analysis）进行使命范围分析”（perform mission area analysis，PMAA）。这种情况下，交汇点 J1 的参照物作为指向一份细化说明表的指针，细化说明描述了参照物 UOB/PMAA 激活的条件。

参照物可以用来转换控制或在过程中表明循环节点。参照物 GO-TO/PMAA 指明过程循环返回行为单元“进行使命范围分析”。最后，参照物还可用来指向执行一个以前定义

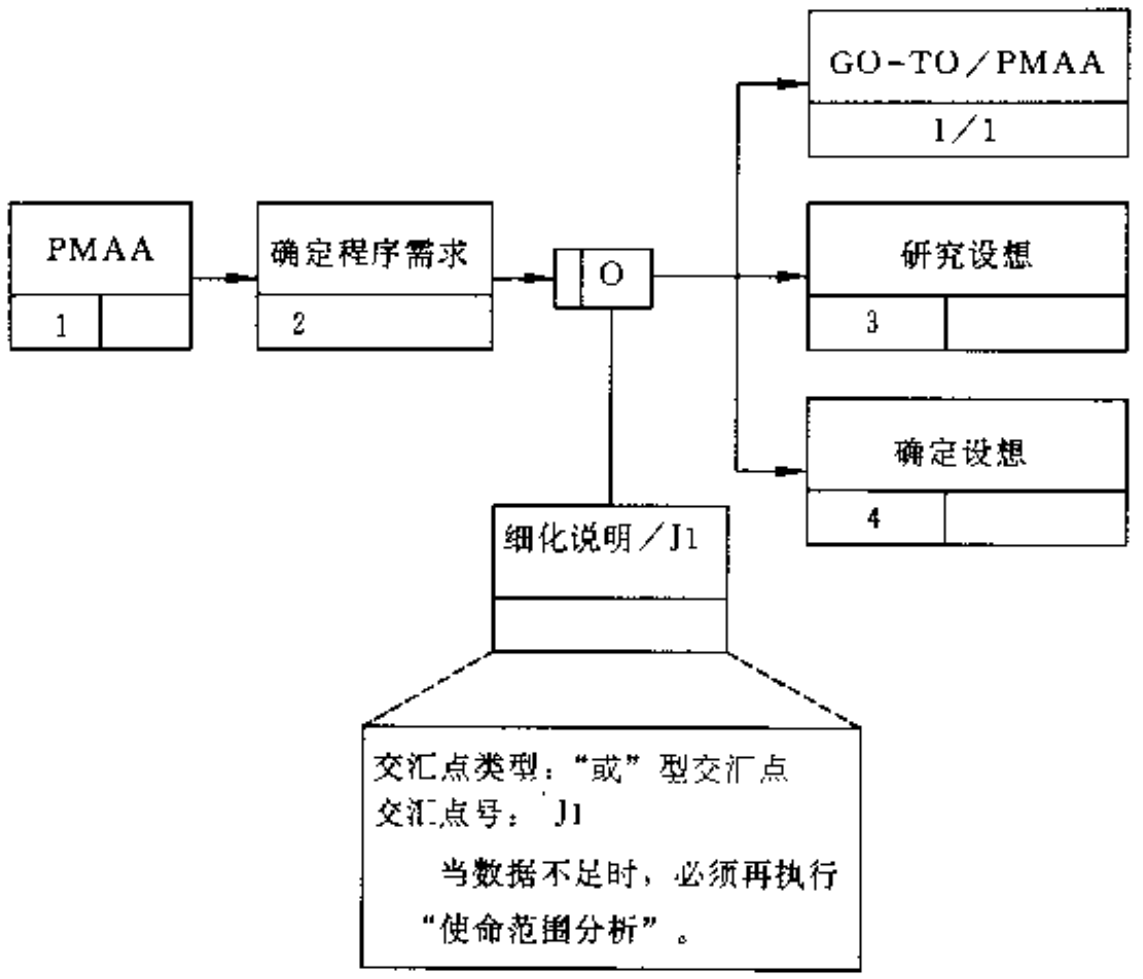


图 3.3.19　带有约束条件的交汇点类型参照物

的行为单元，而无需复制行为单元的定义（如行为单元类型参照物）。

3.5　基本建造块的组合使用

IDEF3 图形语言的各句法元素，可以用各种方式进行组合，以形成一个完整的过程流描述图。为帮助读者理解，本节以例子的形式说明 IDEF3 流图中常用的行为单元、联接及交汇点的正确组合使用，即 IDEF3 建造块的可能组合。

下面几条定义对完全理解本节中的示例大有裨益：

（1）激活（也称为例示）是 IDEF3 过程流图中过程流的事件。要注意，同一过程流在重叠时间段内会有多重激活。

（2）实例是一个过程的一种事件。这样，行为单元的一个实例就意味着该行为单元所代表的过程会发生一次。

（3）实现是一个过程事件的完成或过程流激活前的初始化。

为更进一步理解这些术语，可考虑过程流"处理顾客投诉"这一情况。激活是"处理顾客投诉"这一过程流的初始化，并贯穿整个 IDEF3 过程流描述的始终。实例"记录顾客投诉"便是整个过程流中这一过程的一次事件。当记录完成时，也就实现了"记录顾客投诉"。若对一位顾客而言，规定的过程都已完成，那么"处理顾客投诉"也就实现了。

3.5.1　行为单元与联接组合

在行为单元与联接的组合中，我们可以看到，如何利用联接来形象化地表述两个行为单元间的关系（如：时间的，顺序的等）。

图 3.3.20 告诉我们，其中的行为单元 1、2 之间具有定义准确的顺序关系，图中的联接是一种顺序联接，也就是说，活动 B 只有在活动 A 结束后才能开始。它并不妨碍过程中活动 A，B 作用于同一对象。

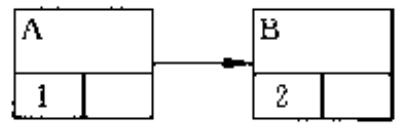


图 3.3.20　顺序联接

行为单元与联接的组合可用于：

(1) 体现一种类型行为单元的实例与另外一种行为单元实例间的时序关系；

(2) 图中左侧的行为单元的每一个实例要在相应的右侧行为单元的实例开始前完成；

(3) 说明在一个过程的激活中，若前面的行为单元有一个实例，那么后面的行为单元必有一个相关联的实例。

图 3.3.21 中虚线连接是一种关系联接。由于这种类型的联接并未带有任何预定义语义，使用者必须根据其中关联的 IDEF3 元素间的关系本质进行相应定义。

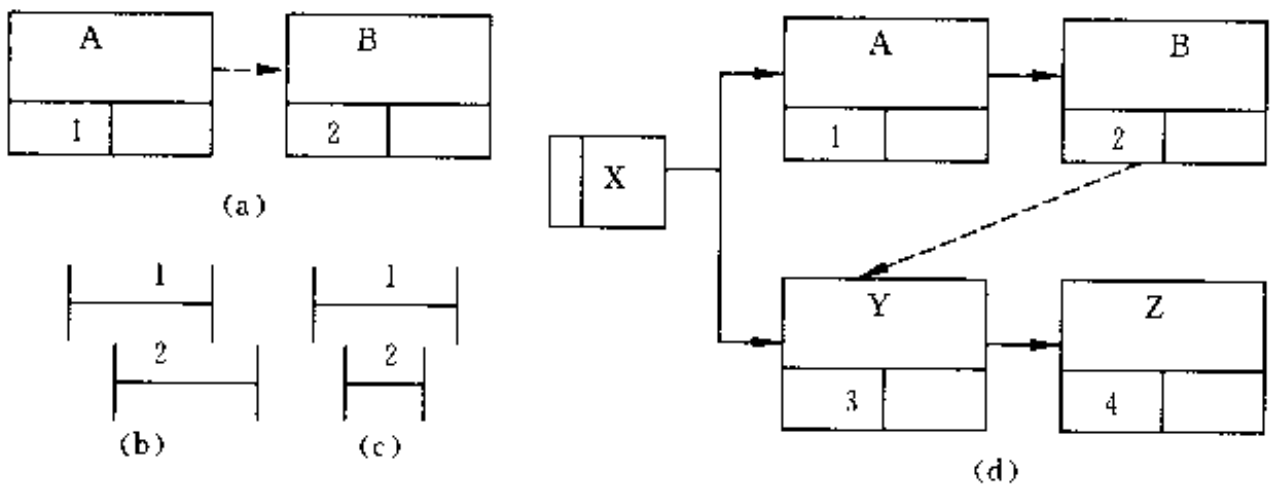


图 3.3.21　虚线联接示例

要注意，虽然使用者可针对关系联接指定专门的时序关系，但它本身并没有预定义的时序语义。其确切语义要由使用者在联接定义文档中加以定义。举例来说，图 3.3.21(a)中，两个行为单元的初始化和结束可以有(b)，(c)两种不同情况。图 3.3.21(d)中则说明两个行为单元间的关系并非是同一个过程路径。联接说明中应包括对这些专门关系的描述。

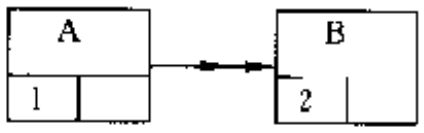


图 3.3.22　对象流联接

图 3.3.22 中采用了对象流联接，它与顺序联接具有相同的时序语义，然而，它的主要目的是要强调同一对象在前后不同行为单元中的参与。这种联接隐含地说明了，此类对象的存在对两个行为单元间关系的关键作用。比如，图 3.3.22 中的对象流联接表明，行为单元 A 创建的对象对于行为单元 B 所代表活动的执行，相当重要。联接说明就要来确切地阐明这个对象参与到不同行为单元中的本质。

3.5.2　行为单元、联接和交汇点的组合使用

这种组合结构是最常见的行为单元、联接和交汇点的代表。如图 3.3.23 中的联接 L1 表示了行为单元 1 和交汇点 J1 的顺序关系。例中，行为单元 1 必须在交汇点 J1 的决策逻

辑前完成(在交汇点做出决策前)。交汇点 J1 和 J2 是“异步的与”型交汇点。J1 是扇出,J2 是扇入。

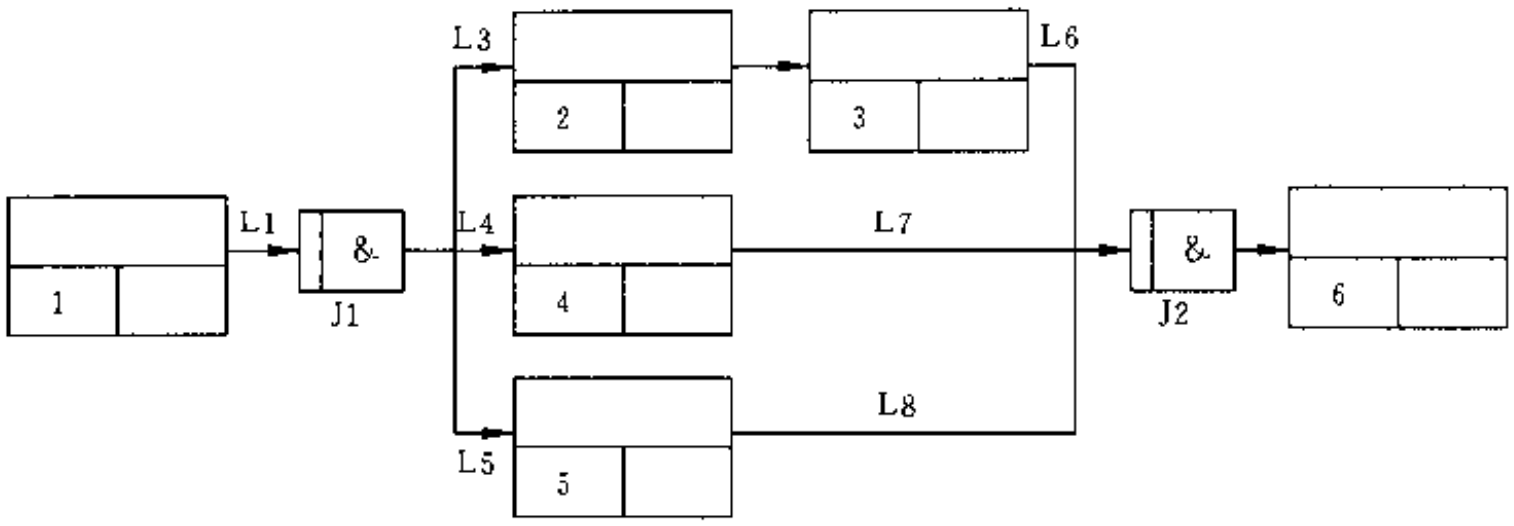


图 3.3.23 “异步的与”型交汇点

(1) 扇出(J1)异步:虽然不必同时,但跟着 J1 的 3 个 UOB(2,4 和 5)都要开始。

(2)扇入(J2)异步:每个前置的过程(3,4和5)都必须在J2后面的过程激活前完成。

图 3.3.23 中的激活过程可以是这样进行的。在到达交汇点 J1 后,三个 UOB(2,4 和 5)必须激活(不管次序);然后,只有在 UOB3,4 和 5 都完成后才能到达“异步的与”型交汇点 J2。完成的时间次序没有要求,然而三个 UOB 都必须在 J2 实现前完成。最后,在 J2 实现后,只有 UOB 6 一个实现作为图形的激活。

图 3.3.24 中的顺序联接 L1 要求 UOB 1 必须在执行交汇点 J1 前完成。如要激活,“同步的与”型交汇点 J1 就意味着 UOB 2,4,5 所代表的过程要同时启动。同样,“同步的与”型交汇点 J2,则要求在执行之前行为单元 3,4,5 同时完成。

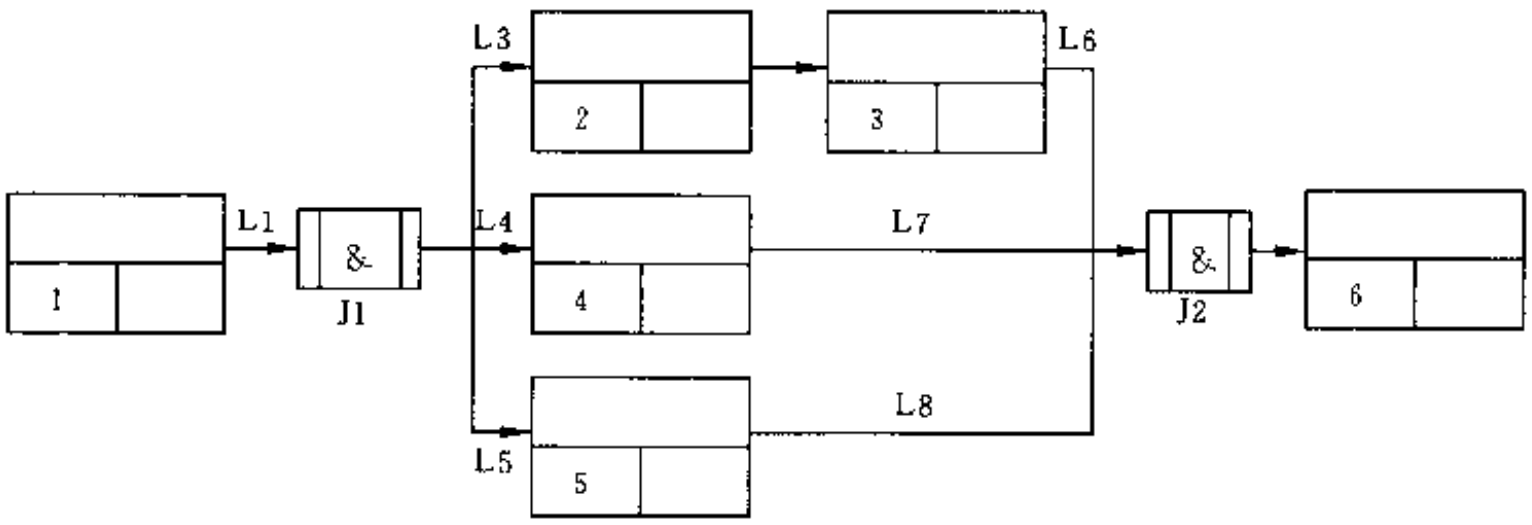


图 3.3.24 “同步的与”型交汇点

图 3.3.25 与图 3.3.24 类似,只是其中的两个交汇点都换做了“异步的或”型交汇点。在图中的过程流激活中,交汇点 J1 要求执行 UOB 2,4,5 中的任意一个或几个,这就会启动一至三个过程通路。激活中接下去要考虑的是交汇点 J2。由于 J2 是一种“异步的或”型交汇点,只要这些路径中有一条完成,J2 后面的行为单元就会触发执行。但这并不是说,

在某些激活中会有一条以上的路径在行为单元 6 实现之前最终完成。然而，UOB 6 只有一次实现，在它之后，结构中任何未完成的过程路径都可忽略。

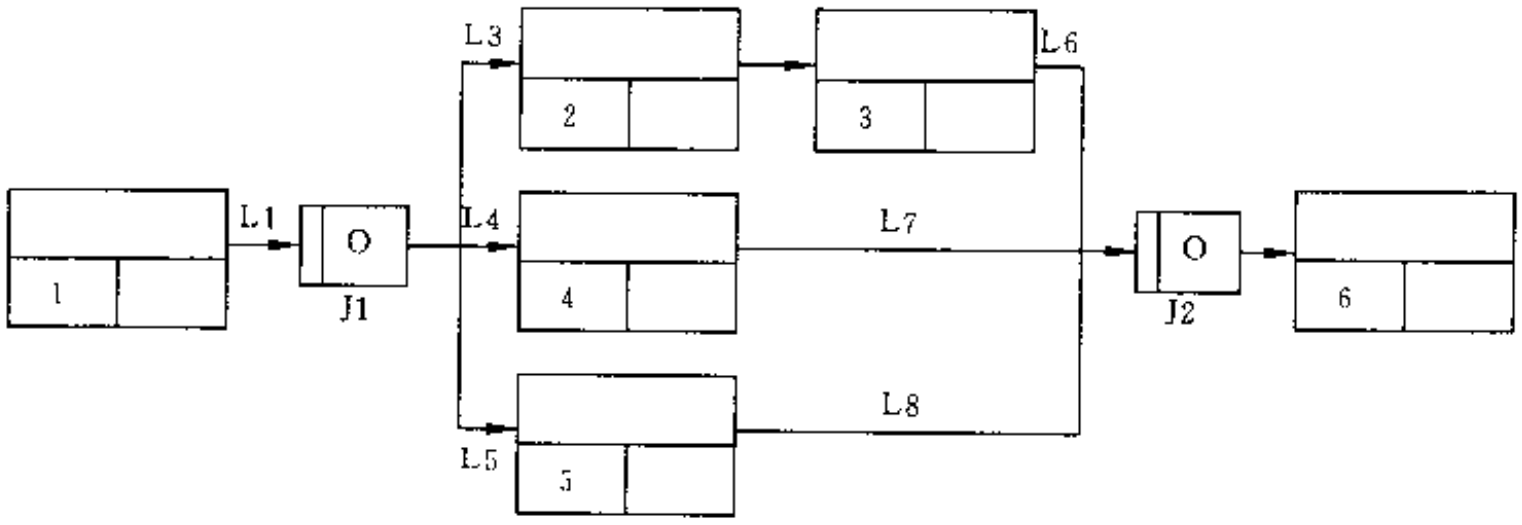


图 3.3.25 “异步的或”型交汇点

在图 3.3.26 中，将要说明两个“同步的或”型交汇点在组合结构中的使用。扇出型“或”交汇点 J1 意味着 UOB 2,4,5 中的一个或多个可以开始。由于这是一个同步型的交汇点，当有多于一个的行为单元启动，其启动必须同时开始。同理，当 UOB 3,4,5 中的一个或多个结束时，它们也要在“同步的扇入”型交汇点 J2 前同时完成。

注意，要实现交汇点 J2 并使过程通过行为单元 6 而进行，前面行为单元中至少要有一个已完成。相应地，首先完成的行为单元会成功触发交汇点 J2，至于其他几个(如果有的话)稍后完成的，则会被忽略掉。

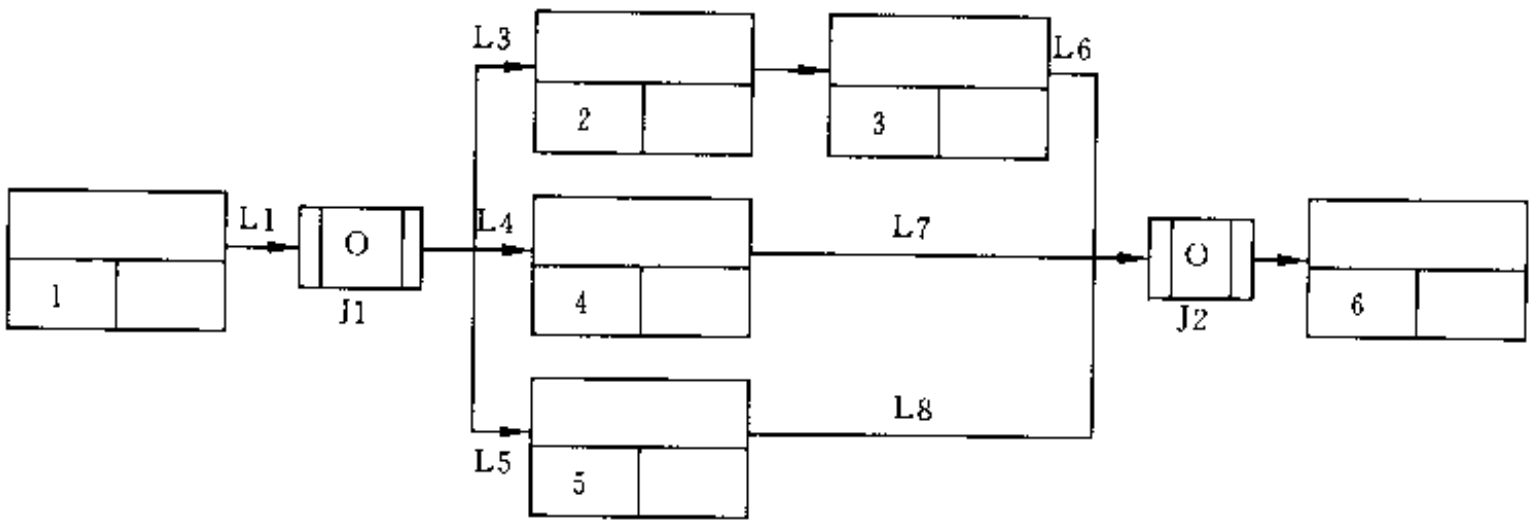


图 3.3.26 “同步的或”型交汇点

下面就是有关事件的执行顺序：

(1) 在时刻 t，UOB 2,4,5 开始启动；

(2) 在时刻 $t+5$，UOB 3 与 4 在交汇点 J2 完成；

(3) 在时刻 $t+10$，UOB 5 在交汇点 J2 完成。

交汇点 J2 将在时刻 $t+5$ 实现，同时 UOB 3,4 所代表的过程也恰好结束。由于 UOB 5 结束稍后于 3 与 4，这个行为单元的激活就会“丢失”。

图 3.3.27 中出现的两个交汇点类型不同，例中一些有效的行为单元完成顺序如下所示。

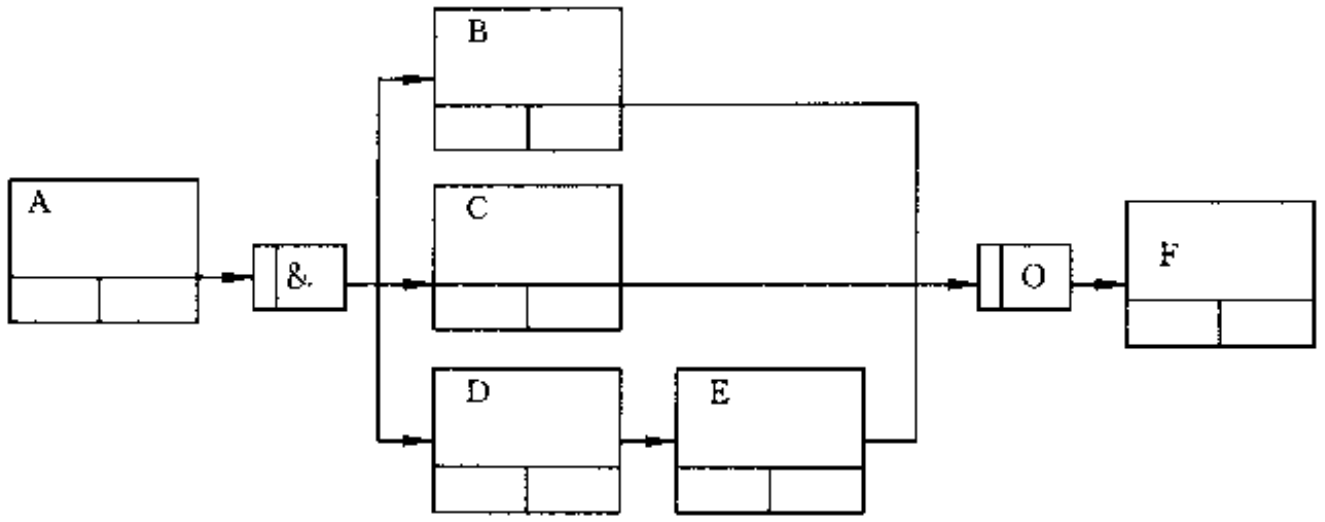


图 3.3.27 “扇出的与”型加上“扇入的或”型交汇点

(小写字母代表图中相应行为单元的实例)

(1) (a,b,f)

(2) (a,c,f)

(3) (a,d,e,f)

(4) (a,b,c,f)

(5) (a,b,c,d,f)

(6) (a,b,c,d,e,f)

尽管行为单元 B,C 与 D 在“扇出的与”型交汇点之后执行,但在过程通过扇入型交汇点前可能这些行为单元中会有两个以上并未完成,从而存在不同的过程顺序。这是因为扇入的交汇点是异步的,要使过程成功地通过这一交汇点,只需联向它的几条路径中有一条在交汇点之前完成就足够了。这样的话,就可以理解次序(a,d,e,f)意味着行为单元 B 与 C 虽已启动,但在 E 完成之前尚未完成。

3.5.3 IDEF3 流图中参照物的使用

IDEF3 图中的参照物可以是无条件的,同步的或异步的。它能够加深理解,提供额外的信息,以及简化过程流图与 OSTN 图的构造。本节讨论参照物在过程流图的使用,至于在 OSTN 图中的情况,要在下一节讨论。图 3.3.28 是利用参照物来强调行为单元中一个对象的参与。例中,对象“里程碑”在行为单元“确定项目需求”中加以强调。这就使流图以形象化的方式突出了对象“里程碑”在完成行为单元“确定项目需求”所代表的活动中的重要性。

图 3.3.29(a)中在扇出交汇点后,采用参照物来代表一个过程逻辑,这是参照物的常用方法。“或”型交汇点完成之后,参照物“GO-TO”说明可能存在一种循环,即在完成行为单元“确定项目需求”之后再返回到行为单元“进行使命范围分析”。这样就可以利用参照物这种手段在过程流图中设置循环节点。图 3.3.29(b)中的参照物则表示行为单元“研究概念”之后的过程转向分解 2.1 中的交汇点 J4 处。

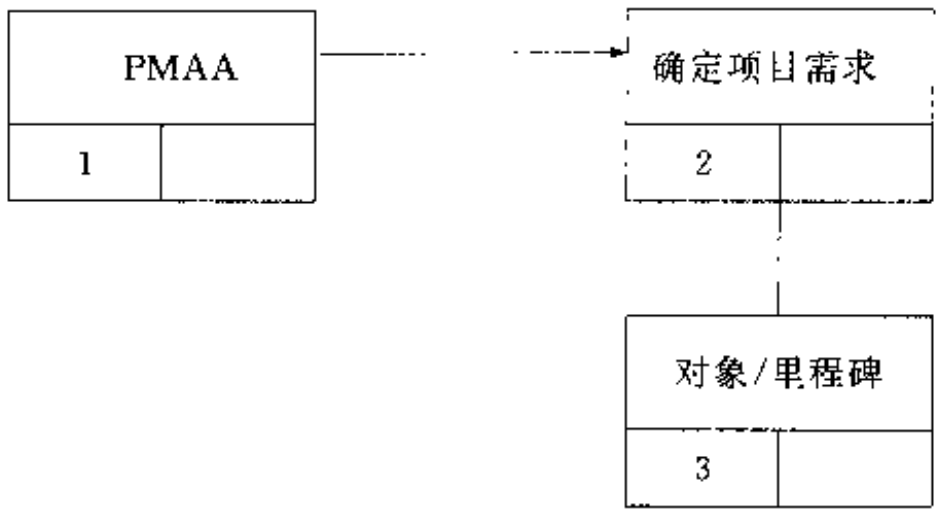


图 3.3.28　对象参照物实例

有时参照物也可将某个行为单元所代表的过程在流图的其他地方进行复制，这种用法见图 3.3.29(c)。例中，执行完行为单元“确定概念”后的过程路径实例沿着行为单元“PATO”(编号 15)所复制的过程进行，后者出现在分解图 9.1 中。这种复制是通过图中参照物行为单元来实现的。

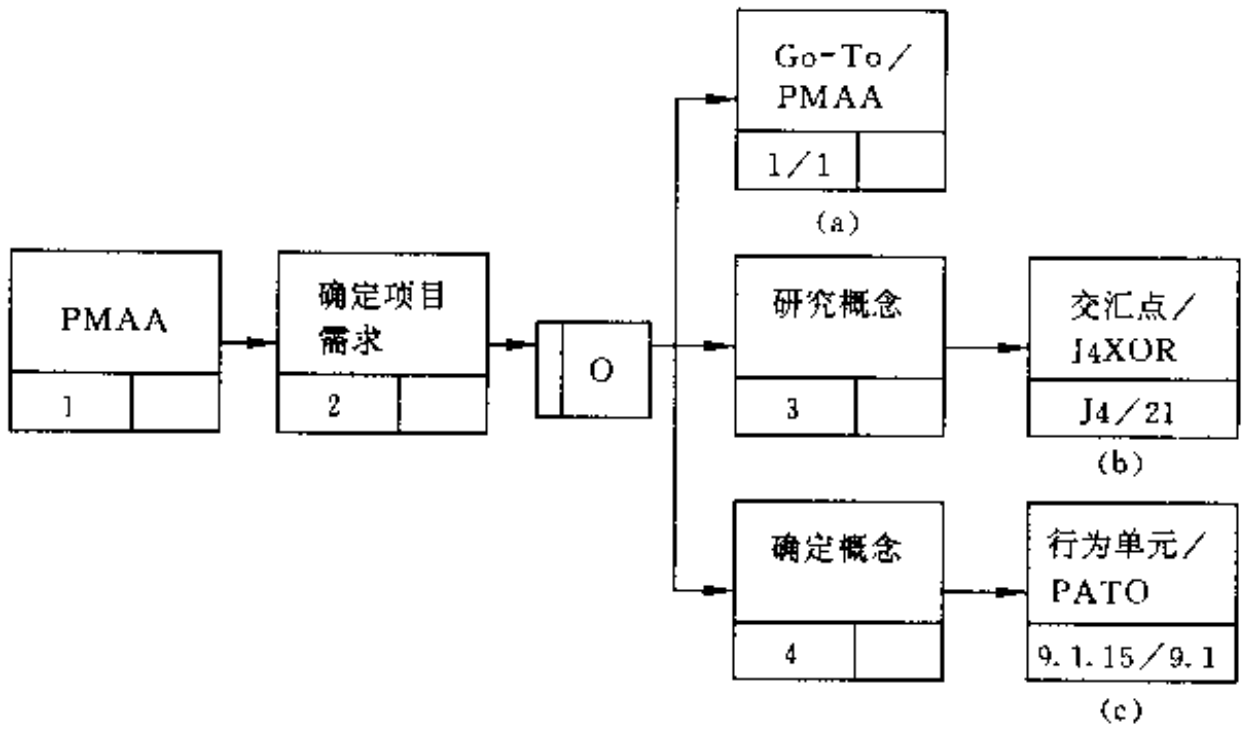


图 3.3.29　过程流图中的参照物

参照物通常用来强调一个过程中特定对象的参与，如图 3.3.30(c)，(d)。而(a)，(b)表明参照物另外一个重要应用，是详细说明交汇点的逻辑关系。在(a)，(b)中，交汇点 J1，J2 的逻辑关系由 Elab(细化说明)型的参照物进行修改，这样，我们就要对交汇点加上细化说明来表述交汇点的额外约束。

图 3.3.30(a)中参照物的更细致的描述见图 3.3.31。其中的细化说明反映了在扇出交汇点 J1 发生过程流分离的逻辑关系。

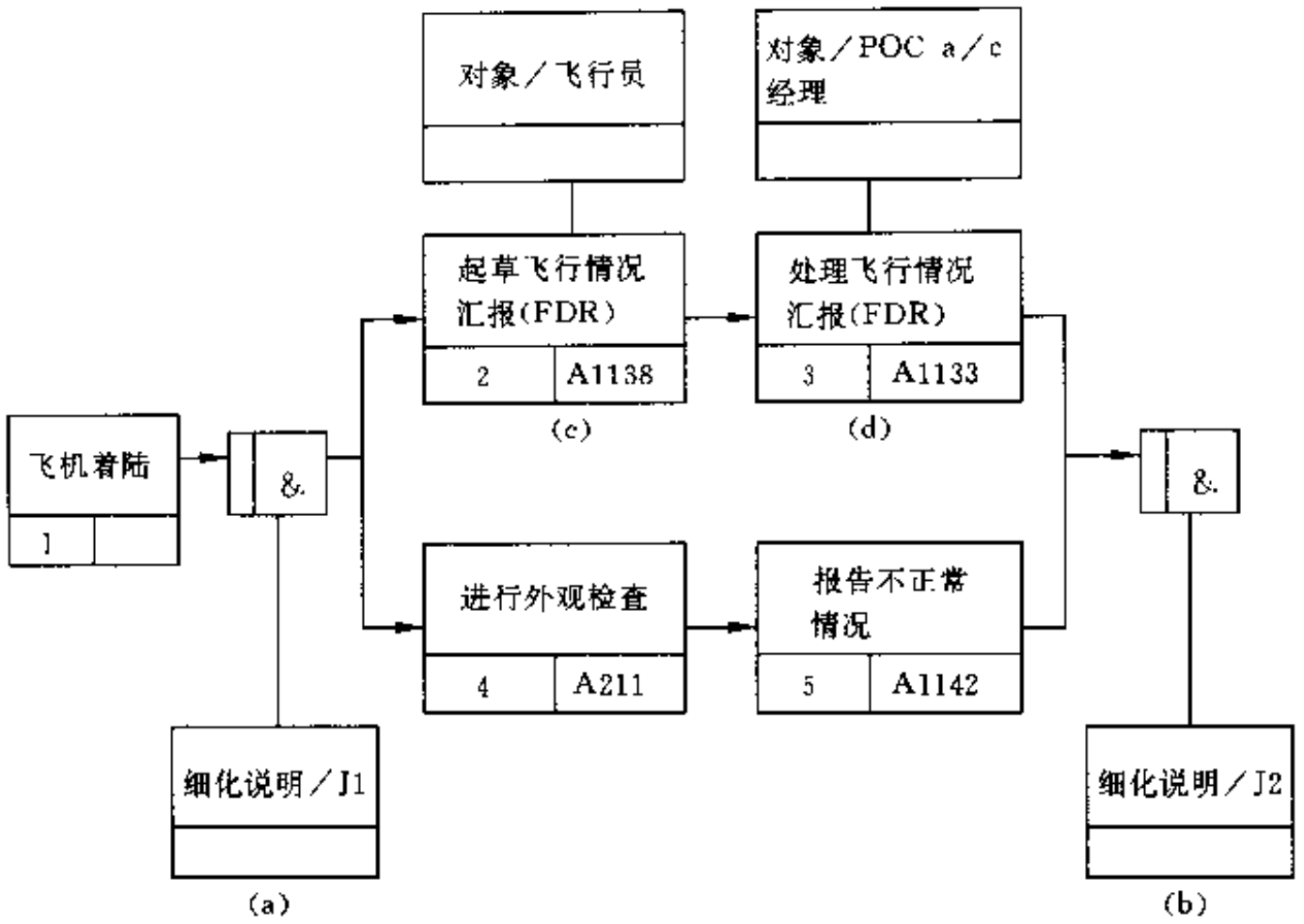


图 3.3.30　交汇点及细化说明

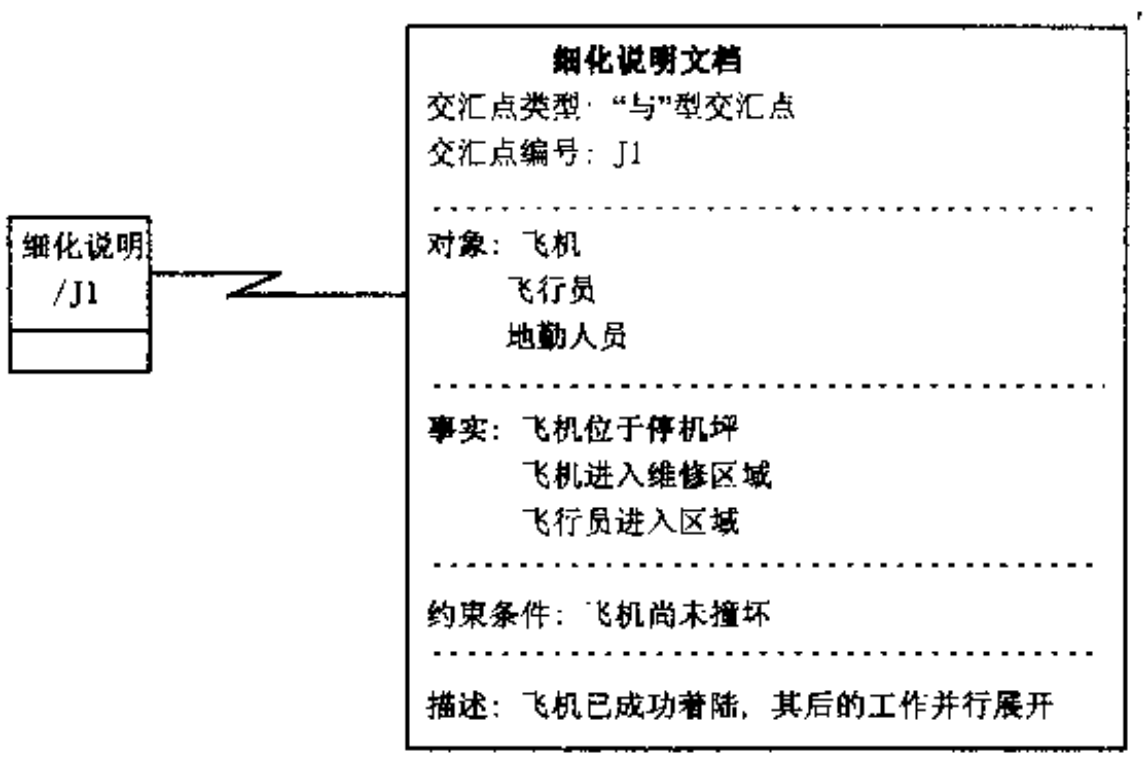


图 3.3.31　参照物的细化说明表

3.6　对象状态转换网络描述

IDEF3 中的 OSTN 图可以获取对领域内有关对象的叙述，以及有对象参与的状态和状态改变的约束条件。引入这种机制可以建立以过程中对象为中心的视图。这种视图贯穿了过程流图并总结了领域中对象可能的转换，已证实，它们在数据生命周期的文档说明中尤为有用。

图 3.3.32 所示的即为 OSTN 的基本句法元素，图 3.3.33 则是 OSTN 的一般形式。

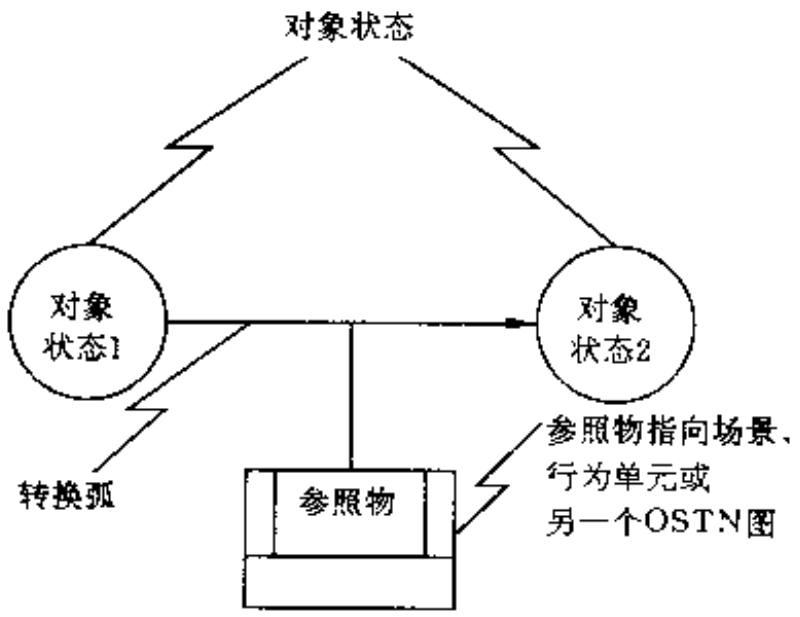


图 3.3.32　对象状态转换图中的符号表示

节点(圆圈)代表对象状态。节点间的圆弧表示状态间的对象可能进行的转换。这些转换弧可以有指向场景描述、行为单元或其他 OSTN 图的参照物，参照物可能是无条件、同步或异步类型。IDEF3 流图行为单元中所涉及的对象都可以通过 OSTN 图进行特征抽取。原则上讲，OSTN 只应用于最重要的对象。本节讨论其中句法元素的语义及 OSTN 语法。

3.6.1　对象与对象状态

OSTN 图重点在于对象及对象状态。对象可以是：(1)物理的，如报告、零件或机器；(2)概念的，如决议、计划或设计思想等；或者(3)是两个或更多个物理的或概念的对象的结合。某组织(如企业)可能会对特定对象进行研究，就研究对象的生命周期而言，对象的性质或特征会发生改变。这样，一个对象就会以各种状态存在，其中每一个都可看作对象状态。对象的状态通常由对象所在环境的工作人员加以标识。

3.6.2　OSTN 描述元素

OSTN 描述的图形表达方式是 OSTN 图(见图 3.3.33)。与每一张 OSTN 图相关联，都有一张 OSTN 描述表所定义的细化说明。如图 3.3.34 所示，这张表中概括了整个 OSTN 图中有用的信息。

下述清单包括了 OSTN 描述表中所有字段的说明：

(1) 对象名称：OSTN 中心对象的名称；

(2) OSTN 号：OSTN 唯一标识的编号；

(3) OSTN 名称：对象状态转换网络图的名称；

(4) 场景名称：OSTN 所存在的场景名称；

(5) OSTN 标签：IDEF3 图形元素中用以指向 OSTN 的字符串；

(6) OSTN 词汇表：对 OSTN 的文字描述，对应于行为单元细化说明文档中的描述字段；

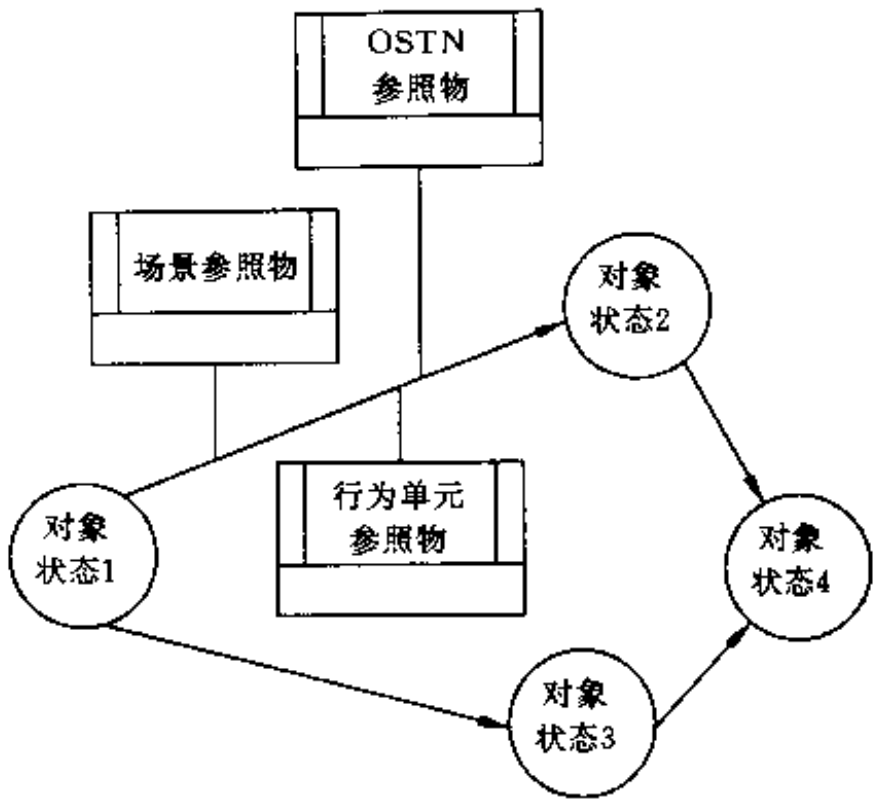


图 3.3.33　对象状态转换图

IDEF3 对象状态转换网络描述表

对象名称：_____　　OSTN 号：_____

OSTN 名称：_____　　场景名称：_____

OSTN 标签：_____

OSTN 词汇表 ：

_____

_____

对象状态集合：　　OSTN 集合：

_____　　_____

_____　　_____

_____　　_____

场景集合：　　行为单元集合：

_____　　_____

_____　　_____

_____　　_____

_____　　_____

_____　　_____

图 3.3.34　对象状态转换网络描述表

(7) 对象状态集合:组成 OSTN 的对象状态组;

(8) 场景集合:OSTN 中所提到的场景名称;

(9) 行为单元集合:OSTN 中所提到的行为单元名称;

(10) OSTN 集合:与本 OSTN 有关的其他 OSTN 名称。

另外还有一种表格,即对象状态描述表(OSD),可使对参与 OSTN 图中对象状态的

特征化详细描述变得方便。OSTN 图中的每一个对象状态都有一张 OSD 表。除去能实现进行对象状态的特征抽取外，OSD 表还能对当前状态相关的转换条件，即入口、出口条件进行定义。另外也包括对象维持当前状态的条件。从三个方面来定义一个状态：(1) 入口条件，即保证对象转换为当前状态的条件；(2) 状态描述条件，对象用以保证当前状态所需的条件；(3) 出口条件，对象由当前状态转换为其他状态所必须的条件。上述这些条件以“属性-值”对或约束条件的形式表达，如图 3.3.35。

IDEF3 对象状态描述表

对象状态名称：_____ 对象名称：_____

对象状态标签：_____ OSTN 名称：_____

对象状态集合：　　词汇表：

入口条件集合：

上一级状态名称：_____

事实：_____ 约束条件：_____

上一级状态名称：_____

事实：_____ 约束条件：_____

当前状态描述条件：

事实：　　约束条件：

出口条件集合：

下一级状态名称：_____

事实：_____ 约束条件：_____

下一级状态名称：_____

事实：_____ 约束条件：_____

图 3.3.35　对象状态描述表

对象状态描述表中会包含以下字段：

(1) 对象状态名称：对象状态的名称；

(2) 对象名称：对象的名称；

(3) 对象状态标签：用以代表 OSTN 图中的对象状态符号的字符串；

(4) OSTN 名称：对象所在的 OSTN 名称；

(5) 对象状态集合：能够转换为当前状态或由当前状态转换而来的对象状态；

(6) 词汇表:对当前对象状态的文本描述;

(7) 入口条件集合:能够保证对象通过转换进入当前状态的所有条件,即所有引入当前状态的转换弧;

(8) 状态描述条件:确保对象维持当前状态所必须的条件;

(9) 出口条件集合:能够保证对象由当前状态转换为其他状态的条件,即所有从当前状态引出的转换弧。

3.6.3 对象状态转换网络图的语义及使用

如图 3.3.32 所示,OSTN 图有圆圈标识的对象状态,以圆圈间圆弧(带箭头)所代表的对象状态转换,以及附属于各圆弧的已标记过的参照物所组成。所谓网络图,是对 IDEF3 描述中有重要作用的对象的特征抽取。

OSD 表对对象可能存在的所有状态进行了详细的定义,这些细化说明同时确定了各对象状态上所能发生转换的条件。至于转换弧上的参照物,则作为一种直观、图形化方式来说明状态转换条件。相应于一个转换弧的参照物,共有三种类型:OSTN 参照物,场景参照物和行为单元参照物。各参照物既可以是同步性的也可以是异步性的。转换弧上的参照物意味着在状态转换进行前它所指代的 IDEF3 元素必须产生。其中,若参照物是同步性的,则要参考过程必须在转换进行前启动和完成;对于异步性的而言,只要求在此之前启动,至于完成与否,并没有要求。转换弧上参照物发生的次序同时也确定了它们所指代的场景、行为单元或 OSTN 将要发生的次序。为了表达清楚,以图 3.3.36 中的 OSTN 图为例。

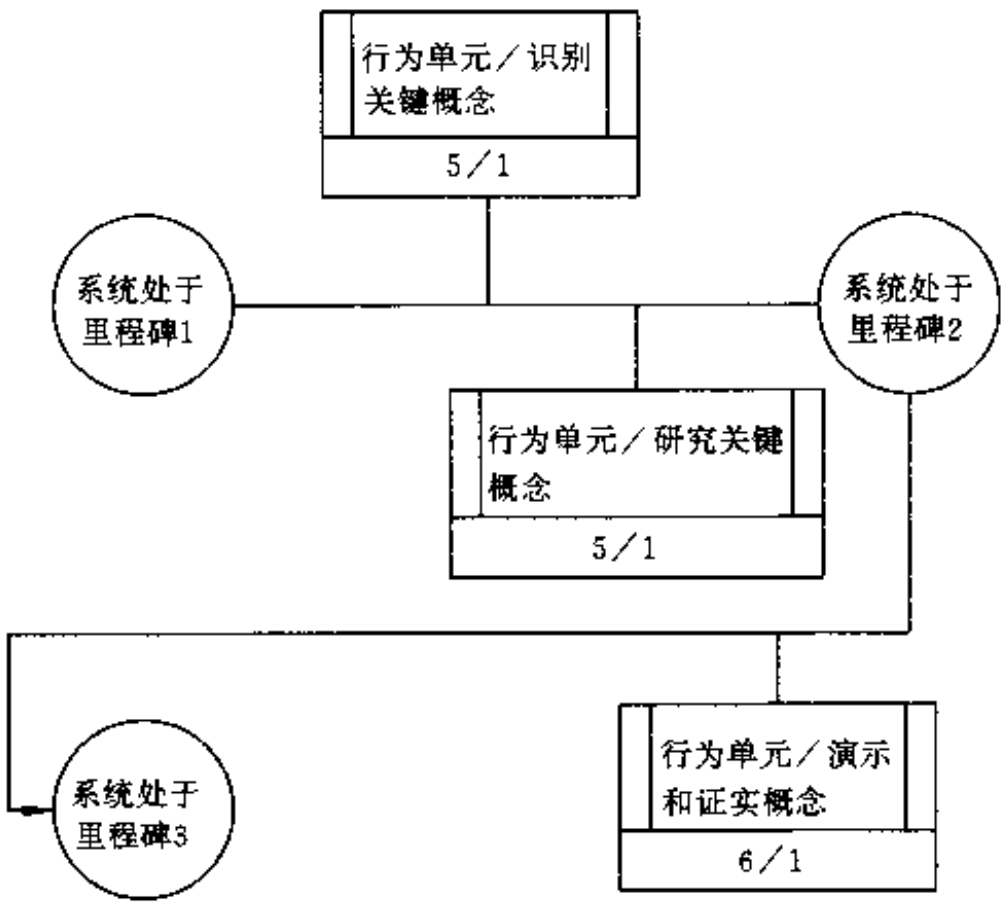


图 3.3.36 实例:说明 OSTN 中概念

图 3.3.36 中,OSTN 感兴趣的对象作为一个系统,可以三种状态形式存在,分别是:

"系统处于里程碑 1","系统处于里程碑 2"和"系统处于里程碑 3"。如果要由里程碑 1 转换为里程碑 2,行为单元"识别关键概念",作为同步性参照物所代表的过程必须要启动和完成,然后启动行为单元"研究关键概念"。同样,"演示和证实概念"过程也要在系统由里程碑 2 向里程碑 3 转换前启动和完成。

第 4 章　应用 IDEF3 描述方法的开发过程

本章是应用 IDEF3 获取统一、有效的过程描述方法的开发过程。它着眼于实现一般以协作开发方式进行的大系统描述获取的需求。范围较小的系统，则无需以下所谈及的全部活动。IDEF3 开发采用功能描述形式。

IDEF3 开发经验告诉我们，描述获取与知识获取、设计工作存在相通之处，即具有反复叠代、结论驱动、体现开发者个人风格等特点。开发过程要看作一种“思维模式”，而非有严格时间顺序的操作步骤。用户不必以一种十分严格的时序方式而拘泥于这些活动。这对于初次使用 IDEF3 方法的用户来说还是很重要的。

IDEF3 过程流描述获取中，通常涉及的人员角色有：

（1）分析员：IDEF3 专家，IDEF3 过程流描述的主要开发人员。

（2）客户：提出描述开发需求的个人或组织。

（3）领域专家：应用领域中的知识来源。

（4）主要联络员：保持分析员与领域专家之间接触的中间人。

（5）项目负责人：对整个过程描述工作最后负责的个人。

（6）评审人：承担这一角色的人员对该领域或 IDEF3 方法有所了解，他们负责对描述草案及文档说明进行评审和审批。其中又分为两种，有权做出书面评注的称为评注者，余者称为读者。开发组成员与领域专家都可作为评审人。

（7）开发组成员：参与 IDEF3 过程流描述应用项目的所有参与者。

IDEF3 应用已涉及到如下广泛的领域：

（1）装备生产；

（2）产品设计和工程；

（3）数据生命周期管理；

（4）国防获取过程；

（5）国防命令、控制、通信与智能系统；

（6）实时控制逻辑；

（7）软件中人机交互的场景描述。

开发中分析员经常把重点放在过程描述或对象状态描述上，特别是“粘着”在一些领域中。但过程与对象并非具有严格的界限，在类似实时控制领域中，领域专家也会将某一过程看作一个对象。IDEF3 提供一组集中的语言手段对来自领域专家的事实论述加以组织。因此分析员在项目开始前，要先对这一事实集合加以检查。另外 IDEF3 也可以与 IDEF0，IDEF1 及 IDEF5 协同开发。IDEF0 与 IDEF1 有助于降低事务复杂性，IDEF5 则具有描述本体论信息的能力（例如事实分类，如将“碳化钙插头”划入易报废工具中）。在描

述过程中，若分析员感到有些描述费解以致有可能发生误解，他就要回溯到相关阶段，并对使用的开发机制重新评估。

4.1 描述获取项目的界定

项目中的开发人员要尽早确定描述获取工作的目的和内容，包括对项目所涉及的范围进行限定。描述内容是依据描述获取项目的范围和初始场景而做出的。项目目的则提供了项目验收标准。它包括以下内容：(1) 对开发目的的论述；(2) 描述所要满足的需求；(3) 客户方要解决的问题。

一般情况下项目开发目的与内容难以预先完整地和精确地定义。客户方经常在程序设计和系统实施阶段又对其需求提出修改要求，所以分析员思考的问题领域常要超出项目范围。项目目的与内容在项目初始阶段逐渐形成，它们以 IDEF3 描述摘要表的形式给出，如图 3.4.1 所示。

IDEF3 描述摘要表

项目名称：_____	项目负责人：_____
目的：_____	条件：_____
场景描述清单：	对象清单：

图 3.4.1 IDEF3 描述摘要表

4.1.1 定义目的

定义目的是开发过程中一项很重要的初始化工作。很多情况下，开发人员在没有明确开发目的的条件下所做出的成果，对客户方基本没用。此外，在目的不明确的情况下，唯一的完成标准就是所给的预算及时间。如果目的明确，项目完成时间往往要比预算时间少得多。

定义目的可分为两部分：(1)定义需求说明 SON(statement of need)；(2) 就如何使用描述信息定义其信息目标。

需求说明(SON)应标识出需求来源，并对客户意图加以描述。可以通过对以下问题的回答来确定信息目标：

(1) 一旦可用时，谁将使用这种描述？

(2) 客户方的问题是什么？

(3) 隐藏在过程描述需求之后的问题是什么？

(4) 过程描述需求后的决策是什么？

(5) 为在描述过程中来解决问题，做出决策，或者回答问题，描述的细化程度应如何？

4.1.2 确定初始范围和细化层次

完成对项目目的特征抽取后，才有可能在以下两方面确定项目内容：(1) 项目覆盖范围；(2)细化层次的划分。项目范围确定了讨论问题的边界，系统中哪些部分应包括、哪些部分不应包括。理想情况下，范围应只包括那些与客户需求相关的领域。与选定范围密切相关的一项活动，是决定描述获取工作的细化层次，一般以一组例子的形式规定文档要求。但要说明的是，这一阶段选定的范围与细化层次，只是一种尝试，在描述数据收集好之后它应被修正。一位敏锐的项目负责人，要定期对依据客户特殊需求和信息目标所获取的描述数据进行评审。

4.1.3 识别主要组织场景

在 IDEF3 的应用中，另外一种建立内容的机制就是在项目范围内识别重要操作场景。对那些熟悉 IDEF0 的人来说，场景识别过程与 IDEF0 模型中 A－0 图的开发很相似。识别涉及到开发人员在对系统(组织)引出的共性问题认识上的一致性问题。有时遇到一些标识不同的场景，它们不过是对同一过程从不同角度的表述。在可能情况下，场景中的初始和终止的行为单元(UOB)，应能确定下来。另外，那些在描述内容外，对场景有影响的活动也要加以标识，以进一步确定描述获取的边界。项目目的和范围提供的指南有助于我们顺利完成对场景描述的标识，但主要还是依赖领域专家对过程的知识和理解能力。项目负责人要明白，这些已标识的场景只是中间开发成果，在数据收集和分析阶段之后，仍可能改变。

4.2 收集数据

建立描述开发工作的内容后，便进入实际数据获取阶段。领域专家与组织(系统)中的源文档是主要的信息来源，分析员要与领域专家密切合作以有效地获取相关数据。数据收集过程是一种重复和交互的过程。最初的数据为获得其他数据提供了指南。

分析员与领域专家接触中，获得对所研究过程的书面或记录形式的描述，活动的名称和参与对象要从中提取出来。通常，分析员要与研究领域中不同专业方向的专家进行会面交谈，从中得到的数据要仔细地加以记录，以便得到最后的描述。

还有其他几种可行的数据来源，包括 IDEF0 功能模型，IDEF1 信息模型，数据流图，操作规则描述等。功能模型对行为单元(UOB)和对象提供线索。IDEF1 模型为场景描述中的对象和对象状态提供相关信息。如果已经有了相应的 IDEF0 模型，则其创建过程收集的或本身包括的信息，可用来减少 IDEF3 的开发时间。IDEF0 模型中的活动与 IDEF3 描述中的行为单元密切相关，但对 IDEF3 开发而言，也并非缺之不可。

因为数据是在项目进程中收集的，它应在 IDEF3 源资料记录表中加以说明，如图 3.4.2所示。

用于：	分析员：建模者某甲　　日期：1993.2.28 项目：IDEF3 描述举例 注释：1 2 3 4 5 6 7 8 9 10 审核：	X	初图	读者： 日期：	内容：
			修正图		
			建议图		
			完成图		

原始资料号#	原始资料名称/描述	收自	注释
SM#1	采购申请/表格 PI-R6 4-72	采购员	
SM#2	程序#079-003　/申请准备	采购员	
SM#3	程序#079-001　/采购单准备	规则与程序手册	
SM#4	程序#101-506　/采购码	规则与程序手册	
SM#5	商品码清单	采购员	
SM#6	产品码清单	采购员	
SM			
SM			
SM			
SM			
SM			

题目： 原材料记录表	编号：

图 3.4.2　原材料记录表

4.2.1　确定专家及数据收集准备工作

分析员与主要联络员一起确定要访问的专家名单，以便收集数据。对数据收集格式，IDEF3 未作限制，不过在访问前，分析员还是要准备一份包括几个专门性问题在内的会面日程。一般从下面几个方面考虑：

（1）从主要联络员处得到每位专家的背景材料，包括他们的职务，目前工作以及专业领域。当然也包括他的姓名、地址、电话号码。

（2）准备一份摘要：包括①访问目的；②涉及的主题；③寻求的信息类型；④要求访问的（权限）依据；⑤利于讨论深入的探索性问题。

（3）确定访问的时间表。

4.2.2　访问领域专家

对专家的访问是很关键的。分析员（访问者）要在这过程中努力创造积极友好的交流气氛，使专家意识到这种交流其实已是对系统的描述开发，并能为该组织解决某些问题。对于新手而言，分析员应不断的告诫自己，要依赖访问对象获取最初的数据，毕竟他对过程的操作相当了解，因而要尊重谈话对象。

通常，若访问前作好充分准备的话，交流过程中专家会乐于提供一些超出访问者考虑范围之外的信息，这也算作是对准备工作的额外收获吧。另外，在访问中专家会提供一些

有关描述过程的文档与表格附件。这些文档实际上已大致勾勒出过程流，或是“应该如此”的过程流。有这些文档后，分析员还是应将注意力放在实际执行着的过程上，而不是形式上有文件规定，但实际没有遵循的过程。

由于 IDEF3 是一种过程流描述获取方法，在访问中应着重于以下信息类型：

(1) 过程初始化的约束条件；

(2) 过程中必须成立的条件；

(3) 过程结束标志；

(4) 该过程初始化或结束时所触发的其他过程；

(5) 过程中出现的事件的特性(如时间区段，可中断性)；

(6) 以代理、信息、源或产品形式参与过程的对象；

(7) 对象特性(尤其是像出现率、破损率等与过程相关的特性)；

(8) 单一过程对象间的联系或关系；

(9) 处于各过程间的对象上的约束或关系(如共享资源)；

(10) 过程中正常与异常情况的区别。

总的说来，上述信息集合可看作“事实”集合。

4.2.2.1 收集对象名称

一般情况下，专家提供的第一类信息是该领域内对象的名称，采访者要仔细加以注释。然后，分析员和(或)采访者依此列出对象清单。这份清单(或称“对象池”)要作进一步地分析，以便把领域内有关的行为单元与领域内的对象相联系。

4.2.2.2 收集活动与因果关系名称

对专家提供的已命名的活动，要有详细注释，因为它们往往会成为 IDEF3 过程流描述图中的行为单元名称。另外，在收集活动名称的同时，也要确定其顺序和结构。访问后的分析中，分析员和(或)访问者要准备一份所有活动的清单，这样做出的活动清单可看作是 IDEF3 过程流图中潜在行为单元的“池”。而且，分析员也要准备一份这些活动之间因果关系的清单，它是构成联接过程流图中各“行为单元”的约束关系“池”。

4.2.2.3 收集状况描述

IDEF3 中，我们将状况描述作为活动出现的特征。这种特征既包括在活动出现中存在特定关系的活动与对象之间的联系，也包括某活动与在其前后发生的活动间的联系。

可以通过观察正在工作的过程，得到状况描述(如访问生产一种特殊零件的工厂)。但类似这样的直接考察所提供的信息通常只是短周期状况的进展。因此，分析员要依靠专家的专业知识，分析长周期情况及某些非正常进程中的信息。在对以上状况描述的分析中，分析员要辅以先前做出的对象和活动清单。状况描述有助于我们加深对活动的前后关系、事实清单、以及被描述过程约束条件的认识。

4.2.2.4 描述结论池的维护

在数据收集和分析阶段，原数据登记在源材料登记表中。最初的研究结果以目录形式收录，称为“池”。IDEF3 中共使用 4 种池：①对象池；②场景池；③行为单元池；④对象状态池。图 3.4.3 是一个对象池的例子。其他几种基本设计与其类似。

用于：	作者：建模者某甲　　日期：1993.2.28 项目：IDEF3 描述举例 注释：1 2 3 4 5 6 7 8 9 收到日期：	X	初图	读者： 日期：	内容：
			修正图		
			建议图		
			完成图		

对象		原材料 ID	对象		原材料 ID
ID 号	名称		ID 号	名称	
01	订购要求	SM1	018	材料清单	SM38
02	采购员	SM2	019	路径表	SM40
03	销售商	SM3	020	目的地	SM41
04	订单	SM15	021	审批人	SM43
05	运抵地点	SM6			
06	需求人	SM11			
07	部门	SM12			
08	样式	SM21			
09	零件	SM26			
010	订购要求项目	SM23			
E10	商品	SM30			
012	订购要求路线	SM31			
013	工作	SM34			
014	计价	SM36			
015	产品	SM37			
016	经营管理员	SM38			
017	经营管理路线	SM39			

题目： 对象池	编号：

图 3.4.3　IDEF3“对象池”举例

4.3 过程流描述的表示法

至此，分析员应有相应的对象、活动、事实和约束的清单。利用数据和状况描述，分析员就可确定行为单元，并着手表达 IDEF3 图的基本结构。初始过程流图表现了分析员对来自专家信息的理解。然后根据初始图，他与专家一起评审在此之前所作的描述以保证其正确性，这些初始图也有助于专家回忆与过程有关的其他知识。初始数据收集过程是由领域专家对其个人知识的回忆能力所决定的。

从专家处获得的描述，其准确性和完整性要不断地进行完善，直至分析员的流图与专家知识一致。有时分析员与专家一起进行描述开发工作更合适一些。这种协作开发方法可以缩短开发周期，而且做出的描述更为完整。但由于专家的时间很宝贵，他很难自始至终参与开发工作。

构图(有或无专家参与)过程分为以下 6 步：

(1) 确定行为单元 UOB；

(2) 将行为单元与相应的场景相联系；

(3) 将场景中的行为单元按基本因果顺序进行组织；

(4) 逻辑描述中加入交汇点结构；

(5) 对行为单元及联接做出细化说明；

(6) 对选定行为单元做分解。

构造过程流图中重要的是要明白我们所作的工作，只是记录过程中发生的事实。因而最初的流图中，一般不表示逻辑流，而只是一些彼此间几乎无联系的行为单元盒子。有时在项目完成时，很可能我们发现图中仍存在孤立的行为单元，因为项目开发并没有要求用掉所有的信息去填补这些空白；应用 IDEF3 获取描述，分析员只是对已知的关于系统如何工作的事实加以组织，而非设计系统；似乎难以置信的是，许多在工作的系统，有些单元没人懂，甚至没人知道。

4.3.1 行为单元细化说明及联接规范

行为单元细化说明和联接规范从对专家的访问中得到，还要经过专家的确认。它们开始看上去是一些专业词汇条目，但随着分析阶段的进行，它们与生产过程计划的操作安排表越来越类似。行为单元抽取特征所关心的是以下事实：①参与对象及其角色，②关系，③作用在行为单元上的约束；这些事实都要在细化说明中阐述。这种用自然语言完成的细化说明与联接规范文档格式，提供了对专家论述最详尽的信息描述。流图不过是对其中部分信息的一种图形表述方式。

4.3.2 特定行为单元的分解

一个行为单元的分解，其实是用一些子行为单元对父单元进行详细的说明。基础的场景过程流图中的行为单元一般都要分解，分解得到的子单元也可以再分解下去。对同一张流图的分解结果不同，源于领域专家对活动进行过程的认识角度不同，也可能是对过程参

与对象的抽象认识角度不同。例如，一张分解图，可能是信息系统为了支持组织活动而建立来表示进展步骤的，但最后分析员为使流图简化，却画了对特定行为单元的分解。这样，分解提供给我们的是，从各种不同观点或更细致角度出发描述的过程，或一种简化过程描述的手段。要着重指出一点，描述开发过程是一个细化过程。

分解描述开发与初始描述开发遵循同样的开发规则，也是一种不断细化完善的过程，其完善周期包括：①分析活动，②收集其他数据，③就相关 UOB 进行状况描述，④评审，⑤有必要的话回溯到前面一步。

4.3.3 与 IDEF0 模型的交互参考

IDEF0 模型中的活动与 IDEF3 过程流图中的行为单元间存在着确定的联系。但 IDEF3 不能看作是 IDEF0 的替代。在对一些较大系统的分析中(如制造航天产品)，行为单元间的因果关系可能不是很明显。这时候，最好先从 IDEF0 做起，然后将得到的 IDEF0 模型分解到一种活动间顺序关系较明显的层次。另一方面，若收集到的事实确实具有某种内聚性组织特点，也不妨先作 IDEF3 过程流描述，然后抽象化得到 IDEF0 模型。设计 IDEF3 方法时，就已考虑了这种交互性。在提供参考手段的情况下，IDEF3 语法，可以识别出与行为单元相对应的 IDEF0 中的活动。3.3.1 节已提到，所有的行为单元盒子中有一个区域标明 IDEF0 模型中相应的活动参照物号(图 3.4.4 盒子的右下角)。

图 3.4.4 行为单元盒区域划分

IDEF3 参考性规则，允许 IDEF0 中的零个、一个或多个活动映射到一个行为单元。有些情况下，行为单元事实上可能只与 IDEF0 活动的某一部分存在映射关系，这时的活动参照物，就要变为 IDEF0 中所包含的子活动集合，如果 IDEF0 活动没有定义到更低的细节层，就要在行为单元细化说明中进行解释。一旦行为单元确定下来，相应的 IDEF0 参考活动也会跟着得到。

4.4 对象状态转换简要介绍

对象状态转换图(OSTN)是 IDEF3 中对过程流图的一种补充说明。它基于以系统描述中对象为中心的观点，对对象状态和状态转换进行细致的特征描述。OSTN 图，可以在过程流图开发前、后或期间完成。本节提供开发 OSTN 图的指南。

4.4.1 选择感兴趣的对象

构造 OSTN 图的首要工作是确定要描述的对象。分析员确定的对象应具有状态信息，并在专家角度来看是起着重要作用的。过程中涉及的对象清单范围是广泛的，相比之

下，特别感兴趣的对象则可能少得多，它们常常是该过程中状态发生改变的对象。

由于 OSTN 的创建，经常是在过程流图完成后，因而感兴趣对象的确定可依据：①行为单元细化说明，②场景描述，③场景描述中用到的 IDEF1 或 IDEF1X 信息模型，以及④初期访问数据。尽管来源不同，它们一般有两点共性：①在过程中状态发生重大改变，②存在于过程的不同阶段中的几个状态。

由于对象可以是任何一个物理或概念的事物，也就不会有一根魔杖或科学的方法来决定某对象是否一定在领域范围内。一般来说，IDEF3 中，我们首先关心那些在系统运行中发挥重要角色的对象，它们具有规范的命名。也就是说，分析员在访问中频繁遇到的一些词或词组，它们所指的可能就是要考虑的对象。其次考虑那些状态令我们感兴趣的对象（显然，没有状态的 OSTN 图没有意义）。这要用到一些直观推断法：①各对象状态应显示其在领域中公认的特征；②对象状态要有一定保持期；③存在着触发、导致和抑制状态改变的约束或过程。对每一个选定的对象，至少可做出一张 OSTN 图。

4.4.2 对象状态转换的特征抽取与对象状态转换网络图的安排

对每一张 OSTN 图，都要配有状态转换描述表，它可以是文字描述，或是词汇条目。但要说明作 OSTN 图的目的。状态转换表中，也包括一些暂时无法填充到其他领域中去的 OSTN 信息（例如，以后要用于 IDEF5 模型中的本体论信息）。除了文字描述外，作为图的参考，分析员还要记录 OSTN 图中涉及到的对象状态及其他 IDEF3 元素（行为单元、过程流、OSTN）等。初步列出这种表，是分析员构造 OSTN 图活动的一部分，尽管不能一次完成，它有助于分析员从源数据中开发出 OSTN 图。

OSTN 图开发的下一步便是描述各对象状态，并对状态转换进行特征抽取。实现这一点，要求分析员完成如下工作：

（1）确定每一个对象状态的特征；

（2）确定进入每一个状态的准则；

（3）确定离开每一个状态的条件；

（4）确定状态间可能的转换；

（5）确定引起每一个状态转换的特定条件；

（6）确定导致、允许转换或由状态转换所导致的活动。

前面的三种工作的结果，以对象活动描述（OSD）表的形式记录下来。其余三个的研究结果则决定了网络图的安排。一旦分析工作完成后，OSTN 图也随之构造出来，OSD 表和 OSTN 描述也就做出了相应的修改。

4.4.3 与 IDEF1 模型的交互参考

在许多大型 IDEF3 开发项目中，可在项目开始前先建立 IDEF1 或 IDEF1X 模型，如果有的话，将有助于 OSTN 图中对象的确定。与各对象和对象状态相关的 IDEF1 模型、实体类别号和属性类都应在 OSTN 或相应的 OSD 表的词汇条目中列为其参考信息。

4.5 IDEF3 过程描述的有效性

4.5.1 动机

IDEF3 过程描述获取的充分发挥作用，要到其有效性验证阶段才体现出来。传统的过程建模技术只是试图利用严格的语法和语义结构来消除过程描述中一些不一致或不完全的地方。进一步来说，它们掩盖描述中的一些缺陷或简单地将实际情况进行理想化处理。IDEF3 方法则不然，它采用一种灵活而又规范的手段，对已知系统运行中的事实加以记录。首先，流图设计阶段存在的缺陷或不一致的地方，会很明显的引起分析员与专家的注意，并着手进行解决；另外，对过程描述获取、表述的认识角度不同，虽会引发专家之间的争论，但最终也取得一致，消除分歧。总之，从以上两种情况来看，分析员和领域专家总会修补流图设计中的缺陷，取得一致性的结果。在这种解决矛盾的过程中，会进一步加深对研究过程的理解，并且发现，专家们对同一过程认识角度不同是产生问题的原因。反之，传统的过程建模，只是反映了分析员对过程中一些假定的看法，再用他对专家描述的理解点缀一下。这种模型就有效性验证来讲，也是冗长难懂的。有时，专家出于方便考虑，或者迫于保证形式上一致的压力，在对系统没有完全深入理解时，就匆匆结束了一个系统过程模型。而利用 IDEF3，就可能采用系统描述图这种讨论工具，来解决分析员与用户由于角度不同所产生的对系统如何工作的看法上的分歧。

4.5.2 构造和分发组表

证实 IDEF3 过程描述有效性的一种主要手段，是对组表文档的评议和审批。这些文档本身就是具有某种完善程度的总体描述。

建立组表文档，要么作为一种获取额外事实的手段，可以在描述开发过程的任意阶段进行；要么在主要开发工作完成后进行(例如，在 UOB 和对象初始清单完成，若干 OSTN 图完成，过程流图完成后)。下面讨论的组表产生及相关的评审循环，提供了一种规范化方法，它可产生准确的过程描述，进而产生满足项目目标的最终产品。

4.5.2.1 组表评审过程中的角色

在前面曾提到的角色，在组表评审过程中会有进一步的定义：

(1) 分析员：IDEF3 描述中的主要开发人员。评审过程以他作为起点和终点。与描述获取一样，组表评审循环中分析员同样要依靠领域专家的技术知识。

(2) 评审人：所有参与 IDEF3 组表评审的员工。

(3) 评注者：既对应用领域有所了解又对 IDEF3 很精通，同时能提供书面结构化注释的评审人。评注者阅读分析员的材料，并核实其中的技术精确度。作为领域专家，他们还负责发现 IDEF3 过程描述中的错误，并提出改进意见。他对能否产生高质量的结果至关重要。评注者要决定是否没有脱离描述的目的，是否存在错误和疏漏。评注者有权在评审阶段提出书面意见。

(4) 读者:仅为了信息目的而发给 IDEF3 组表的评审人。通常是给分析员提供信息的个人。

(5) 文档管理员:他的任务是保存文档资料,进行复制,分配 IDEF3 组表并定期记录。

角色与各人的工作职称无关,因此一个人可能具有几种角色。事实上,每个人的参与是唯一的,是依赖于 IDEF3 组表的。

4.5.2.2 IDEF3 组表评审循环

组表代表了已达到某种完成状态的 IDEF3 过程流描述,这些描述草案,以标准的 IDEF3 组表形式分发进行评审。图 3.4.5 中的 IDEF3 组表评审循环,建立在其他 IDEF 方法评审过程基础上。为了清楚起见,以下的讨论并不主要涉及文档管理员,而将重点放在分析员与评注者之间的交流上。当然在大系统中,文档管理员的作用还是很重要的。小系统中这个角色可以由分析员担任。

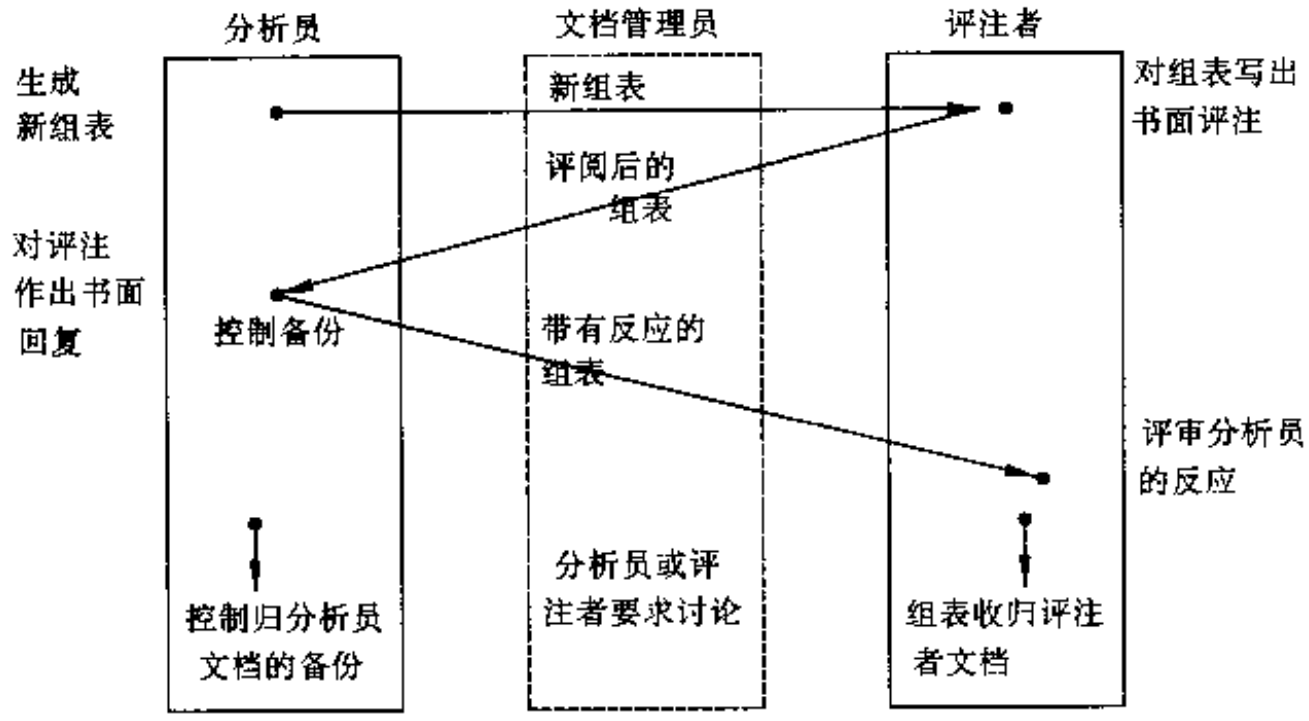


图 3.4.5 IDEF3 组表评审循环

下面是 IDEF3 组表文档评审循环的主要步骤:

(1) 分析员建立组表文档(如池组表文档、场景组表文档、对象文档或附有过程流图、OSTN 图的描述文档),分析员本人存有备份,并将一份提交评注者评审。

(2) 评注者在规定时间内,阅读和研究分析员文档。评审的主要目的是确定过程描述是否与开发的整体目标和前后关系一致。评注者可直接在图表、组表文档或封面上评注。评注后的组表,要在封面规定的时限内返回给分析员。

(3) 分析员直接在评注者组表文档上进行回复。分析员若同意评注意见,可将它记录在工作日志中,并在下一版本的 IDEF3 过程描述中加以改进。若有异议,他将其标注在组表中,并返回评注者。

(4) 若分析员的回复令人满意,评注者阅读后会将其存档。否则就要安排两人之间的会面以消除分歧。

(5) 这种循环进行下去，直至得到双方都可接受的成熟的 IDEF3 描述。

这种评审循环的结果是一份分析员和评注者都做出了贡献的 IDEF3 描述，如果需要，要把问题清单管理起来。评审循环的一个副产品，是评审过程的历史记录。

在以上循环中，文档管理员从事备份、分发、存档、以及在分析员和评注者间传递组表。

4.5.2.3 IDEF3 组表类型

IDEF3 组表与其他 IDEF 方法中的组表结构类似，包括图、文本、描述、细化说明以及与评审有关的任何原材料。

IDEF3 组表的 3 种类型：

(1) 针对某一场景描述及其全部或部分从属文档的场景描述组表，它可包括以下部分：

① 过程流图与所有从属的行为单元分解图。开发初期完成的组表可能忽略某些分解。

② 所有现有的 UOB 细化说明与联接规定。开发初期的场景组表可能省略其中部分或全部。

(2) 针对若干对象提出的对象组表，包括所有从属的 OSTN 图，他们的描述及其从属的对象状态描述。

(3) 开发后期会得到描述组表，它是对特定项目的完成了的场景和对象组表的一种编纂，包括 IDEF3 描述中所有的场景描述及从属的文档。被批准的描述组表是开发工作的最后提供物。

场景组表可提供任何级别的细节信息，从一张单场景过程流图，到包含所有细化说明和 UOB 分解图的全套过程描述。过程描述组表，也能提供任何级别的成果，不过与前者着眼于单一场景描述不同的是，过程描述组表反映的是整个系统的当前状况。更详细的场景和描述组表的叙述见 4.5.5 节。

4.5.2.4 分析员与评注者指南

1 评注者指南

由于问题的性质、类型与技术手段千差万别，很难制定适当的评审问题和规则的模式。然而，提高质量的指南还是有的。评价文档质量的主要准则是：文档能否与其原定的读者很好地交流？是否达到了目的？在给定范围内是否符合事实而且准确？下面是对评审工作的全局指南：

(1) 注释要做到简明扼要、完整。分析员要懂得，精确的表达会使交流易于进行，减少不必要的争执。

(2) 对评注部分用@标记。首先在注释清单中找到下一个尚未用到的序号，在相应的地方记下序号值，用圆圈圈起，旁边用“～”联接。

(3) 提供建设性的改进意见，尽量是可行的解决方法，而不是否定的评注。

(4) 收集整理所有评注，把它们放在封面上或一张单独的纸上。(不要收集属于个别

页的特殊意见到这里来。)归纳一下分析员、评注者会面的议事事项,使之尽量明确。

评审时间的长短取决于几个因素:对所描述系统是否熟悉,评审次数,分析员与评审人的经验等。IDEF3 组表返回分析员时未带任何评注,意味着两者意见一致,不过评注者要明白,他与分析员一样,也要对工作质量负责。

2 分析员和评注者交换意见

收到评注后的组表,分析员要在每个评注@号旁打"√"或"×"给予回复。"√"表示分析员同意该意见,并在下一版本的 IDEF3 组表中根据评注进行改进。"×"则表示不同意,并要做出相应的解释。分析员回答所有评注后,组表将又返回给评注者。

阅读分析员的回复后,评注者确定不能解决的分歧,要求面谈,然后依此形成会议事项。

3 会谈规则

没有书面的评注和回答,最好劝阻分析员和评注者的交谈。在提出会面要求后,要遵循以下约定:

(1) 每次会谈要有时限;

(2) 会谈要按议事事项进行,不能离题;

(3) 若双方各自坚持自己的观点,不肯让步,成效很少时,会谈可以中止;

(4) 双方可以约定有明确议程的下几次会面日程等活动事项表,然后结束本次会谈;

(5) 会谈中应该指定"书写员"写纪要,及对注释、决定和主题等的记录;

(6) 未能消除的严重分歧可进行专业处理(如写成两种观点的文档,或提交技术委员会裁决)。

会谈的最终结果,应是关于问题解决或提交管理层决策的书面意见,亦可用要任一位参加者作进一步研究的方式解决。

4.5.2.5 IDEF3 组表内容

IDEF3 组表是一种技术性文档,包括图表、文字说明、词汇表、决策概述、背景知识以及关于评审、评注的其他资料等。

1 关于组表准备的通用指南

为避免遗漏,可将 IDEF3 组表看作手头唯一的信息资料来评审。注意其中的印刷错误,并在 IDEF3 组表上加上简单注释的说明。至于出现在组表中的一些词汇条目定义,可作为附属材料。

收集便于评注者阅读的有用的材料,但不能用这些补充材料来传达必须由图本身传达的信息。尽可能地使用最自然的交流手段来表示细节,这对使读者理解其中的概念非常重要。再把所有的材料合并起来,配以封面页,递交文档管理员。

2 封面页

封面页标识 IDEF3 组表文件,它包括分析员、日期、项目、文档号、题目、状态和注释等栏。见图 3.4.6。

(1) IDEF3 过程描述和文档描述

题目:应能说明 IDEF3 组表

图 3.4.6 IDEF3 组表评审封面页

生命周期阶段：AS-IS 或 TO-BE(取决于组表中的描述是当前事实，还是未来事实)

系统名称：系统或子系统的缩写

(2) 项目信息

分析员：提交 IDEF3 组表者的名字

日期：送至文档库的日期

公司：提交 IDEF3 组表的公司名称

(3) IDEF3 组表信息

检查描述组表,场景组表或对象组表。标出由文档管理员提供的文件序号。

(4) 评审循环

由评注者和分析员在评审结束后签名并签上日期。

(5) 索引和内容

列出每页中场景、分解、对象和对象状态的名称,同时要标出 C-序号(下面讨论)。若有必要,与组表封面页一起,还可提交 IDEF3 组表内容页(见图 3.4.7)。

文档号	分析员: 日期: 公司: 项目号: 任务号:		场景组表	评审人 日期
			描述组表	
			替代或评审后的文档号	分析员 日期

页号	对象/场景/分解 题目	C 序号	状态	页号	对象/场景/分解 题目	C 序号	状态

描述名称:	C-序号: 页号

图 3.4.7 IDEF3 组表内容页

(6) 评注和专门说明

与评审人员有关的其他信息,也可用作对文档管理员管理文档的专门说明。文档库亦可用此栏作对 IDEF3 组表接受人的专门说明。

3 图的格式

图 3.4.8 为图的格式,它的结构和约束很简单。这一页所涉及到的只是对结构化分析领域重要的功能。包括:①建立观点,②各页间的交互参考,③每页内容的注释。图表采用标准格式方便存档和备份。

图表分为 3 个主要部分:

(1) 工作信息(图表体头部)

(2) 信息栏(图表体中部)

用于：	分析员： 日期：		初图	评审人：	视图
	项目：		修止图		
			建议图	日期	
	注释：1 2 3 4 5 6 7 8 9 10 收到日期：		完成图		
场景： 对象： 分解：		描述名称：		编号：	

图 3.4.8 IDEF3 组表图的格式

(3) 标识栏(图表体底部)

如此设计的目的是当最终的"完成图"生成后可将头部分切掉。图表格式用于所有书面内容。

工作信息：

下面几栏记录有关工作信息

(1) 用于

此图所用于的方面，而不是图的内容。

(2) 分析员、日期、项目

填写此图作者姓名、第一稿的时间以及所属的项目名称。日期栏也可包括其他时间，填写在初稿时间下面，表示对初稿的修改，若没有改动，就不必填写。

(3) 注释

这一栏是对图表页进行评注所用。对每页的评注用@号标注，如评注是写在另一页上，序号要顺次圈掉。由于圈上的序号唯一地对应一条评注，就可以方便地查看各条评注。

(4) 状态

规定的状态有 4 种：初图，修正图，建议图和完成图。

初图：无论以前状态如何，改动最大的图。刚刚完成的图当然是初图。

修正图：与前一状态相比，改动最小，并在一定程度上为一组读者接受。

有的修正图由项目负责人提出，但尚未被项目组全体通过。

建议图：图表及文字说明评审后，被项目组接受，不会再有改动。

完成图：作为最后出版或发布的图表。

(5) 评审、日期

这一栏是供评注者在各图表上签名和注日期的。

信息栏：

信息栏包含要传达的基本信息，一般用于作图，也可作他用（如词汇表，校核记录，注释，摘要等）。除了这种图的格式外，分析员不能随便用其他纸。

标识栏：

（1）题目栏

本栏包含图表中所表达的材料的名字。若信息栏是一张图，则题目栏的内容应精确地与父盒子中名称一致。

（2）编号

该栏涉及所有与本页有关的号码。

C 序号：由两部分组成。前面是分析员姓名中的 2 或 3 个字母，后面是分析员给出的一个顺序号，C 序号位于序号栏的右下角，是该页或该图表的主要参考标志。每张图表有唯一的序号。在 IDEF3 描述组表发表时，可用标准的页序号来取代 C 序号。

页号：IDEF3 组表页号位于序号栏的右端，由文档管理员填写。也由两部分组成，首先是文档号，然后是标识文档中该页的一个号码。

第 5 章　IDEF3 应用开发实例

本章以对一个理发店的运作过程的描述为例，来说明 IDEF3 方法的开发，其中也包括开发人员使用 IDEF3 经常会犯的错误。对初学者而言，这个实例可看作今后开发的指南。

5.1　确定目标和范围

理发店主希望通过 IDEF3 过程描述得到店里日常工作的细节。为此，他聘用一位分析员来开发 IDEF3 过程流描述。店主的意图是要通过该项目的开发，使新雇员熟悉本店工作，而无需花费专门时间进行培训。本例中，问题讨论的范围可只限于理发店本身，讨论细节也只包括有助于向新员工解释工作的信息。项目目标和范围在 IDEF3 描述摘要表中进行说明。

这一阶段中分析员可先确定几个待选场景描述，组成“场景池”。然后随项目开发深入而不断完善。

这里，只有 3 种建模角色：(1)分析员(IDEF3 专家)，(2)领域专家(店主)，(3)客户(也是店主)(一般后两者不会是挑一个人)。下面用角色名称称呼他们。

5.2　收集数据

5.2.1　访问领域专家获取最初描述

在初始开发阶段，分析员要从事典型的知识获取活动。他通过向领域专家提问，得到对理发店运作情况的描述：

本店有两位理发师，身边各有一把理发椅。此外，还有四把椅子供顾客等候就座。为便于顾客等候时消遣时间，书架上还备有图书。出纳员平时在前台负责收钱。若顾客进门时，理发师空闲，他会走到理发师前的空位上理发；否则，就要一直等到理发师有空。在等候时间内，他可以从图书架上取杂志阅读。然而，若理发师都忙着，而且没有空位可坐，他只好失望地离开。顾客理发后，在离店前要先付钱。

通常，访问中所获取的描述的完整性，取决于下列因素：

(1) 领域专家乐意或能够在访问中付出的时间。

(2) 访问者的经验及专业知识。

(3) 领域专家对被描述过程的了解。

访问中，分析员会得到包括书面记录在内的原始描述。这种获取描述的目的，只是要

表述系统实际操作过程，而不是领域专家想象中的系统工作情况。因此分析员要将访问记录与对过程的亲自观察结合起来，来避免个人主观观点的影响。否则，这种描述会有失偏颇，且不利于完整性的实现。

5.2.2 分析数据识别的描述

访问结束，分析员要对他的记录与观察进行仔细研究，以确定对象、活动、事实及描述约束条件，分别列出相应清单。

在过程描述中，我们经常把注意力放在该过程中的关键对象及作用在其上的活动。下面是该例中要确定的一系列对象：

顾客	理发师	理发需求
理发店	等候椅	杂志
理发椅	出纳员	杂志架

分析员就以下两点而言，要建立清晰的“对象池”。

1 他可能在描述获取过程的后阶段忽略了某些对象；

2 由最初分析阶段得到的对象表中，常包含对保证 OSTN 图的设计至关重要的基本对象。对象标识之后，便要研究访问记录来决定理发店日常运行中的活动与过程。重要活动，就是在描述中要被表示为行为单元(UOB)(活动、动作或过程)的候选者。这时并不急于找出活动间的前后关系。我们只是列出候选的 UOB 表，形成“UOB 池”。

(1) 顾客到达；

(2) 顾客坐在理发椅中；

(3) 顾客就座等待；

(4) 理发师为顾客理发；

(5) 顾客失望地离开；

(6) 顾客付钱；

(7) 顾客离店；

(8) 顾客阅读杂志。

访问分析的最后一步，是根据领域专家的描述，对过程中的事实与约束加以标识。事实是有关对象的论述；约束是过程中对象之间或过程间已知的控制条件。在访问记录中，类似“不”，“从来不”，“没有”等否定性词组描述，有助于我们确定约束条件。在开发初期，事实和约束清单往往是不完全的，进一步的访问和与领域专家交谈，有助于使清单更完全。

两个理发师	两把理发椅
四把等候椅	理发师或忙或闲
没有一个理发师或	一把椅子空闲

5.3 进行过程流描述

完成对象、活动、事实和约束的标识后，就可进入正式表达过程流描述。数据收集阶段

的工作成果，对过程流描绘的顺利进行至关重要，因为我们要利用候选 UOB 构造 UOB（行为单元）。在分析访谈结果中识别的事件与约束，是建立 UOB 细化说明必不可少的。过程流描述开发有两个主要步骤，即：(1) 构造具有前后顺序的 UOB，(2) 建立 UOB 细化说明。

5.3.1 确定图中行为单元的顺序

对行为单元及其前后关系的确定，要按一定的步骤进行。

第 1 步：首先定下过程流中最左侧的行为单元，本例中为“顾客进店”。

第 2 步：确定下一个 UOB。本例中有三种可能：“顾客理发”，“顾客就座等待”，“顾客失望地离开”。这一步意味着流图中开始出现分叉，要用扇出交汇点来表示这种分流。分析员根据过程描述，必须明确这种交汇点的类型。实际上，一位顾客在某一时刻只能从事以上三种活动中的一种。因此要用异或(XOR)类型。这样就得到图 3.5.1 所示流图。

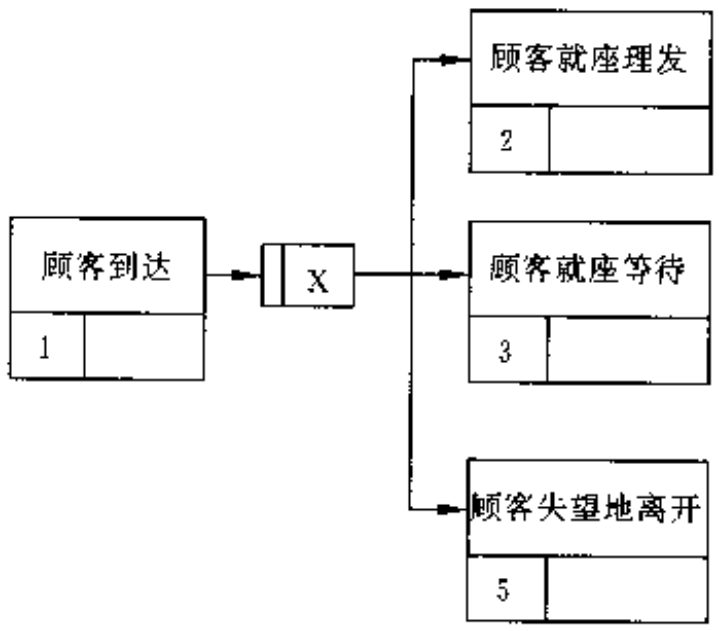


图 3.5.1 场景描述的第 1 步

若过程中没有分叉，开发工作根据 UOB 前后关系继续进行，直至遇到分叉。分叉后各条路径相互独立，它们在给定的描述条件下可能汇合在一点。

第 3 步：选择由行为单元 2 开始的路径进行讨论，其后顺序发生的是“理发师为顾客理发”，“顾客付钱”，“顾客离店”。如图 3.5.2 所示。

第 4 步：同第 3 步类似，完成图 3.5.2 其余两条路径。注意，图中各行为单元的序号与数据收集阶段的活动列表中的编号相一致。

第 5 步：作出图 3.5.3 后，可能 UOB 表中的活动已全部使用。但流图并未最后完成。因为，从领域专家的描述中，我们知道一旁等候的顾客最终也要理发。也就是说，在“顾客就座等待”的活动之后，同样顺序发生“顾客理发”、“顾客付钱”及“顾客离店”。与此相应地，在图 3.5.4 中的行为单元“顾客阅读杂志”后，使用一个参照物“转入顾客就座理发椅”来指代后面的三种活动。这就形成了图 3.5.4。

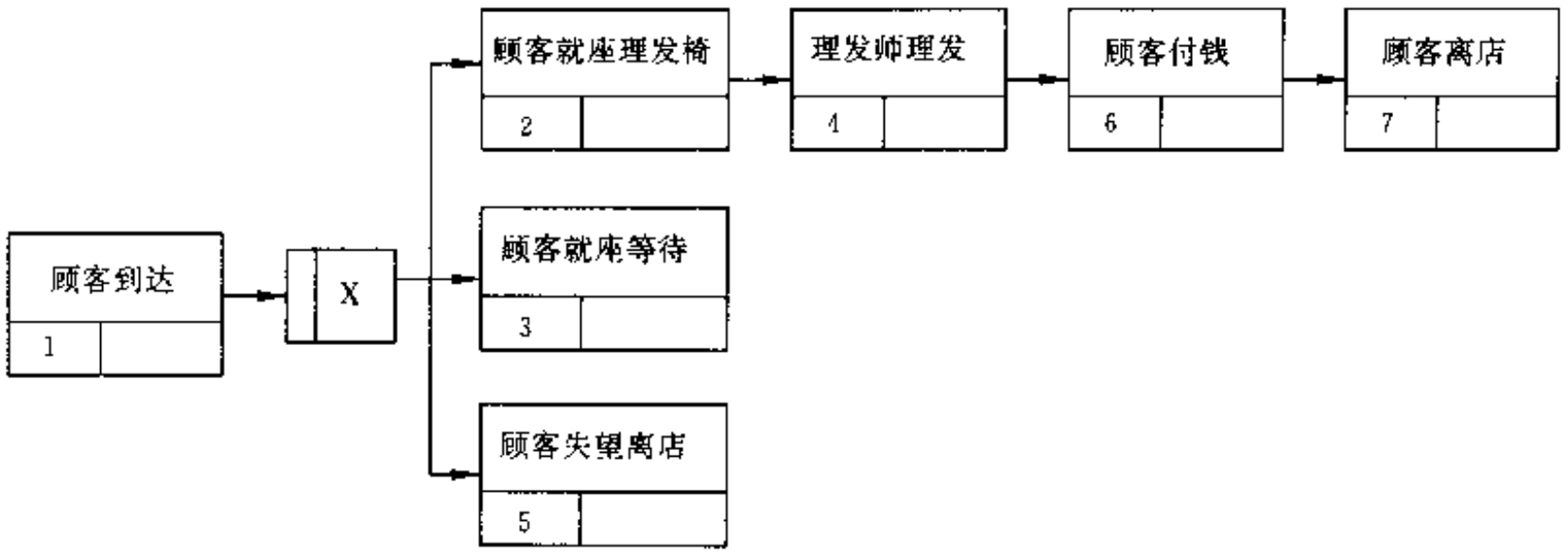


图 3.5.2　完成一条分叉路径后的流图

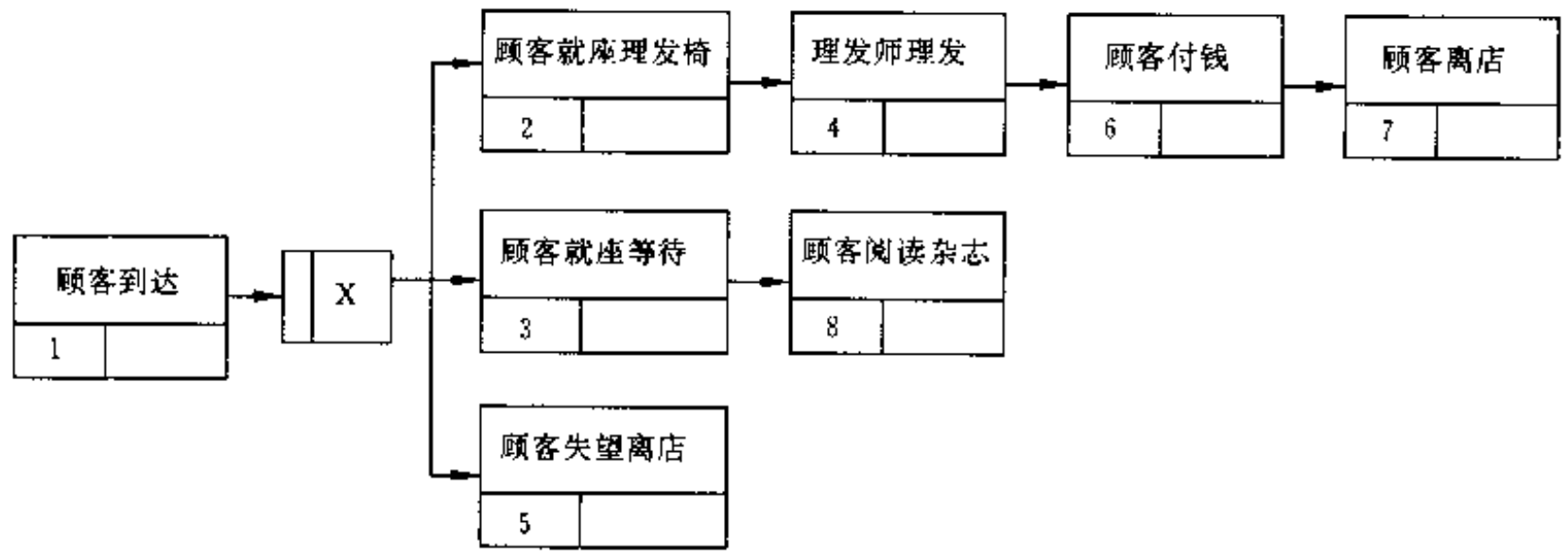


图 3.5.3　接近完成的流图

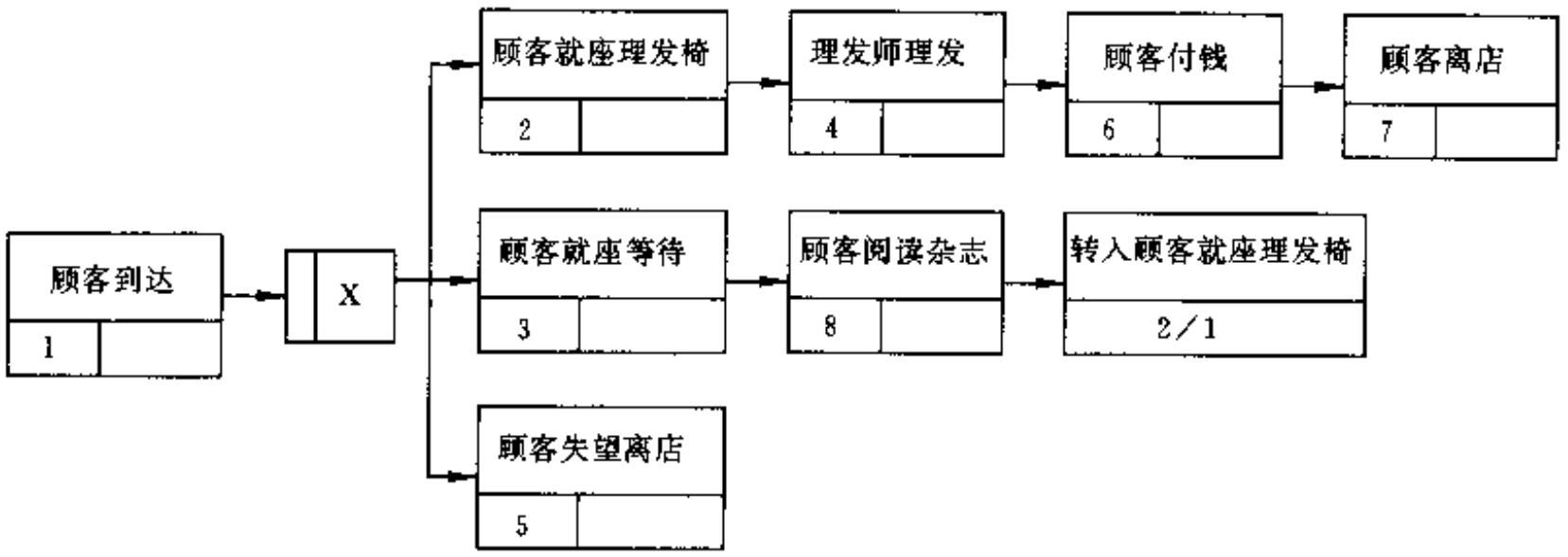


图 3.5.4　第一次评审前的完成图

5.3.2　分析行为单元细化说明中的数据

过程流图完成后，图 3.5.5 和图 3.5.6 中的各行为单元，必须配以细化说明（如图

3.5.5，图 3.5.6)。细化说明开始可能不完整，原因之一在于分析员首次访问的主要精力放在对象和活动的标识上。若分析员对过程不大熟悉，或描述过程过于复杂，这种情况尤其突出。

反过来，分析员对过程比较熟悉的话(如本例中的理发店)，他会从首次访问中获得相当多的、对建立细化说明有帮助的信息。这种熟悉程度，反映在他为访问准备的问题中。细化说明中，分析员同样要避免个人经验对细化说明中信息的影响。

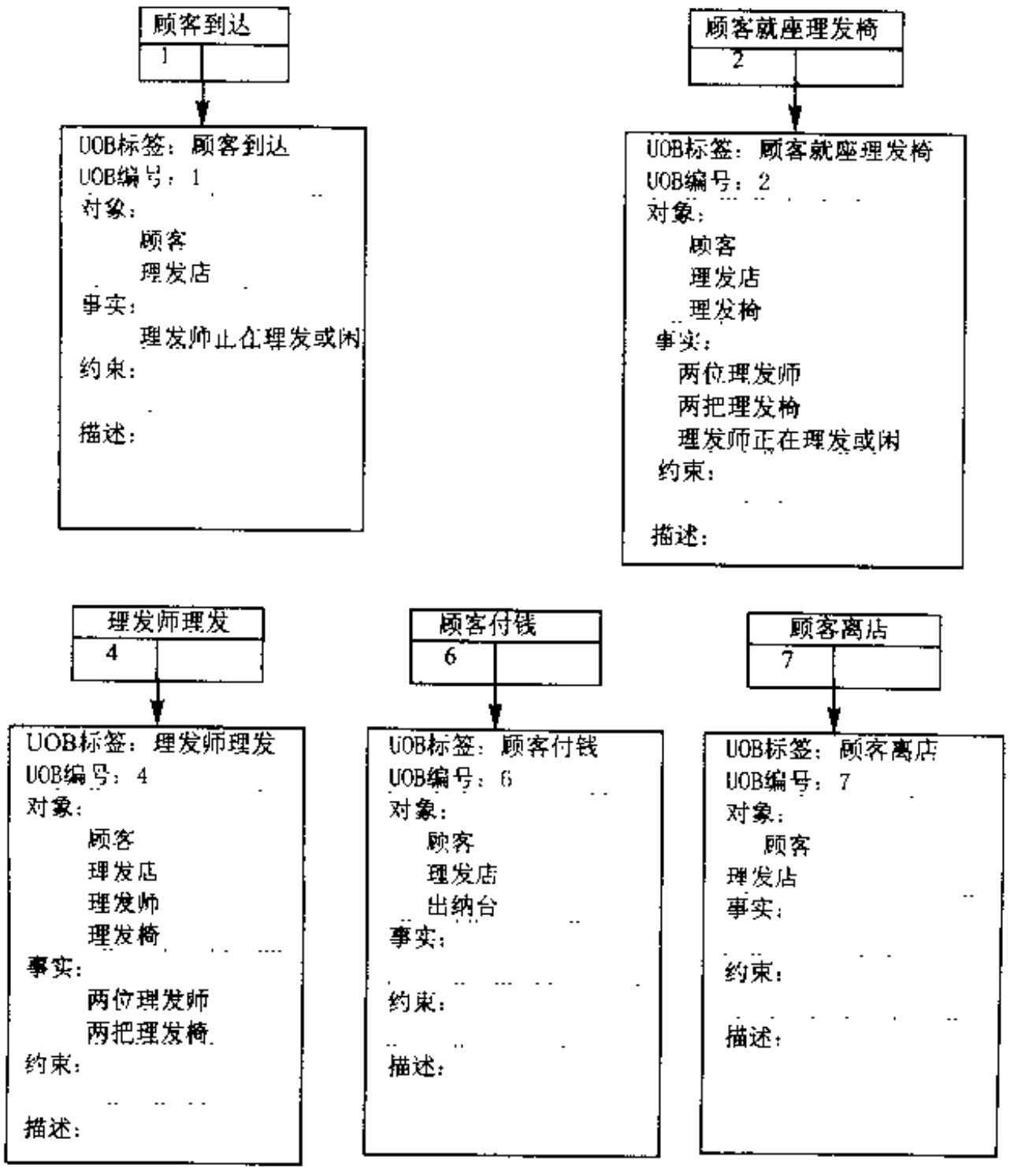


图 3.5.5　细化说明表

细化说明与过程流图谁先完成并不重要。许多情况下，我们在完善过程流图的同时，对前面的 UOB 建立细化说明，这种做法有助于分析员建图。本例中的细化说明，则是在过程流图完成之后，所得细化说明示于图 3.5.5 和图 3.5.6 中。为简洁起见，这些细化说明中并未包括约束条件。事实上，过程流图中各联系都会产生与其前后连接的 UOB 相应的约束条件(入口和出口条件)。

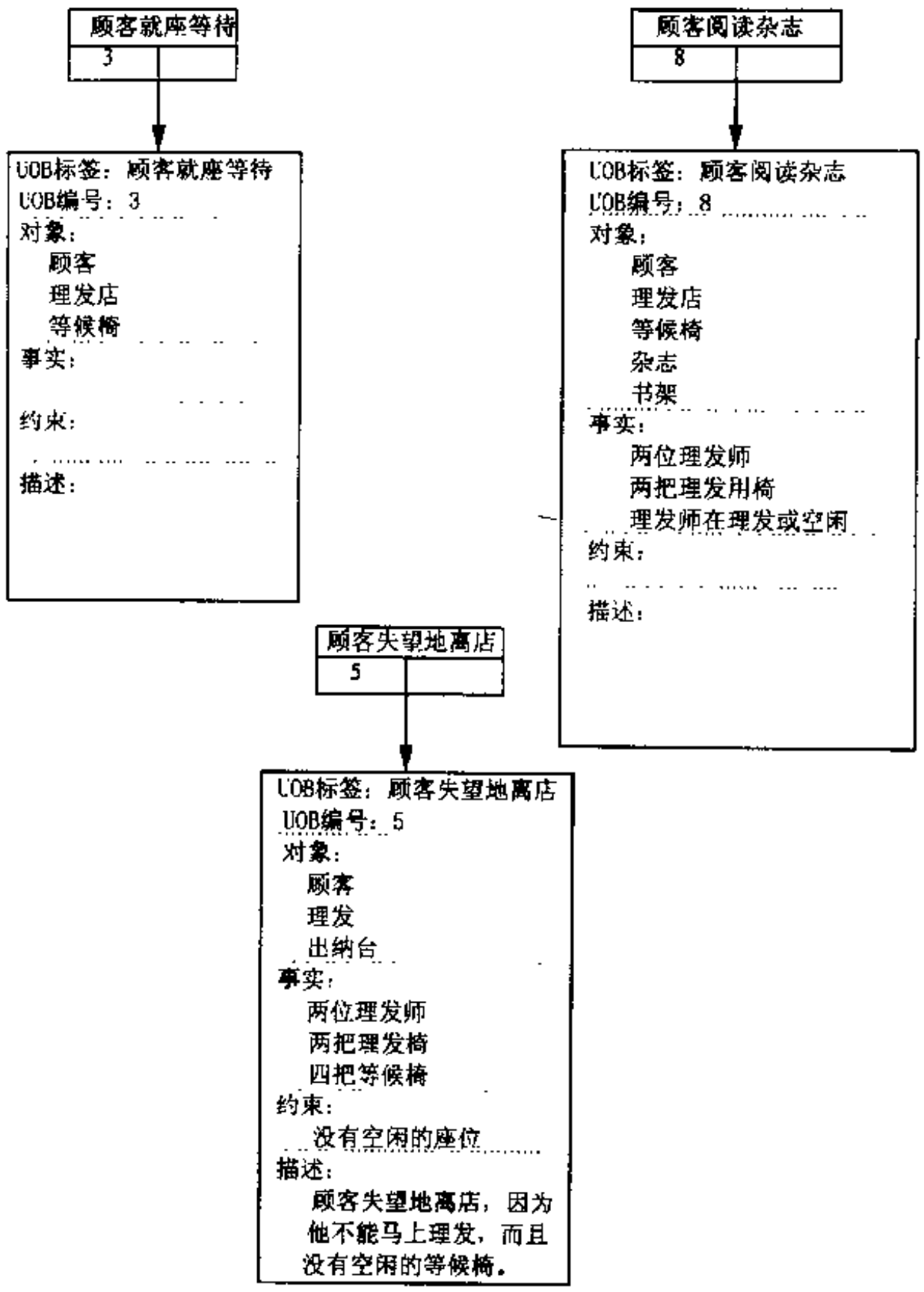


图 3.5.6　细化说明表

5.3.3　与领域专家一起评审过程流描述

细化说明的完整性，主要取决于首次访问的深度和获取的信息量。分析员应力图使过程描述和细化说明，准确反映在访问中得到的领域专家的观点。

本例中，分析员在尽可能完整地完成描述后，还要将其送交领域专家评审。在这次访问中，要检查过程流图的结构，以保证与专家对场景的认识相一致。另外，还要纠正流图及细化说明中的错误。评审意味着可能要对已获取的过程描述进行改动，即对最初的对象、活动、事件和约束条件列表，进行增加，删除或修改。

与店主(即本例中的领域专家)完成评审后，分析员明确要改动的部分：

(1) 部分顾客对理发师有偏好。

(2) 某顾客若有其喜爱的理发师,他会一直等到该理发师为自己理发。

(3) 顾客理发遵循先来先服务的原则。

(4) 理完发,顾客检查发型、付钱。若发型不满意,顾客带着不满的表情离店。

(5) 评审交流中,分析员想进一步知道:顾客如何在图 3.5.4 中行为单元 2、3 和 5 之间选择?

领域专家回答如下:

顾客进店后,首先要做的是找一个有空的理发师,如果有,而且恰好是他所喜爱的理发师,或者他对理发师无挑剔,顾客就会坐在理发椅上,由该理发师为他理发。否则,他会在旁边就座等待。如果顾客进店后,发现没有空位,他只能失望地离开。

评审交流之后,会得到一些新的数据,依此对列表进行改动。下面就是本例中得到的对象列表中的新数据:

喜爱的理发师,

失望的顾客,

满意的顾客(理发后顾客对发型效果会满意或不满意)。

下面是附加的 UOB 池(附加的活动或过程):

检查理发效果(UOB9),

寻找空闲理发师(UOB10),

对理发效果不满意地离开(UOB11)。

这里 UOB 按前面的序号排下来。

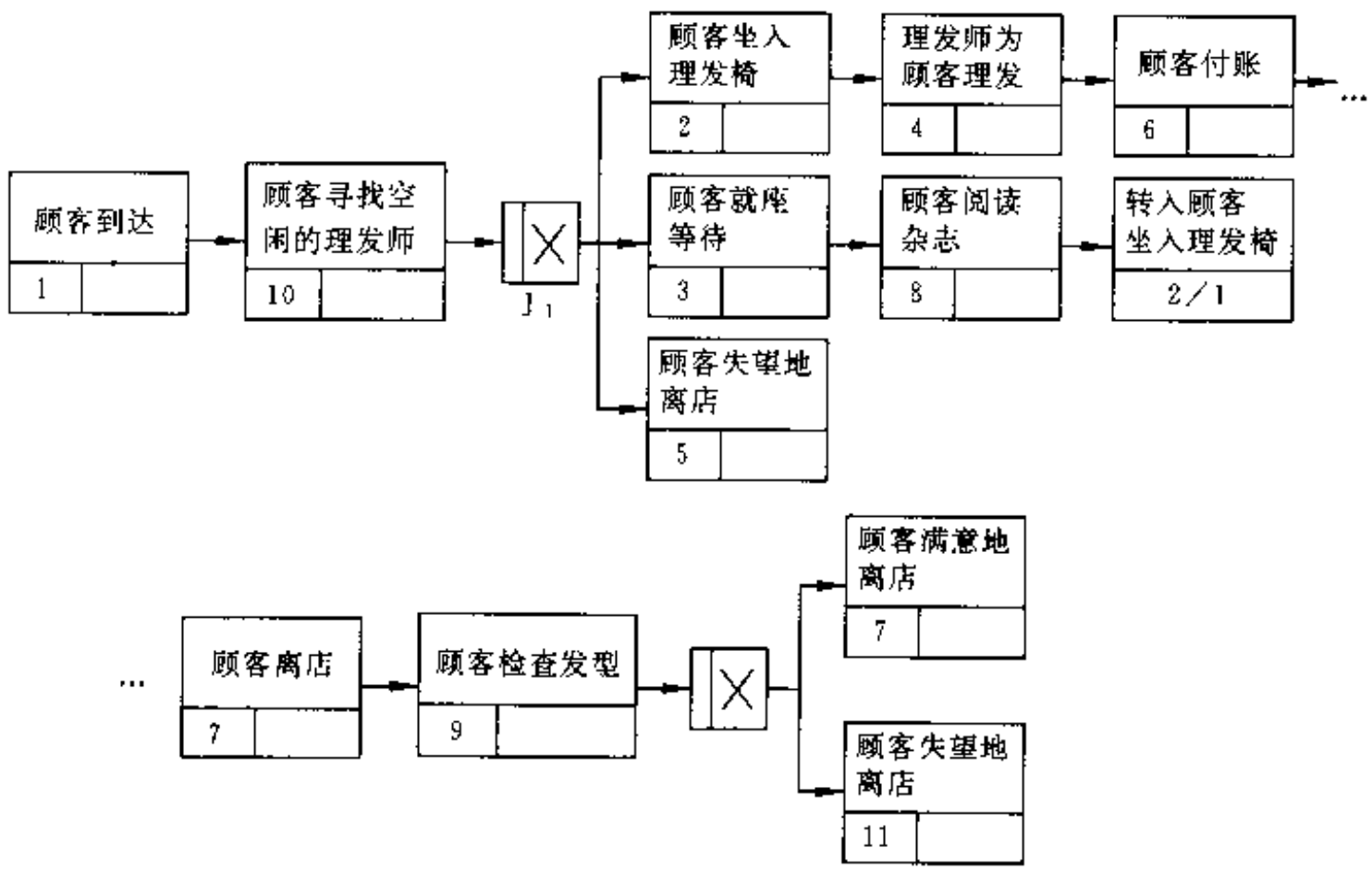


图 3.5.7 最后的完成图

下面是附加的事实与约束：

排队规则是先入先出(FIFO)，

部分顾客对理发师有偏爱，

顾客等待他所喜爱的理发师为其理发，

顾客对其理发效果满意或不满意。

领域专家提出的新数据和改动，要体现在过程流描述中，据此做出的流图如图3.5.7。其中 UOB11“顾客对发型效果不满意地离开”与 UOB5 乍看上去很相似。

如果 UOB11 与 UOB5 所表示的行为相同，UOB11 必须用指向 UOB5 的参照物代替。然而，在对 UOB11，UOB5 相应的对象、活动和约束条件仔细研究之前，还无法确定这种做法的正确性。

对图 3.5.8 中 UOB11 的分析发现，其对象、事实以及约束与图 3.5.6 中 UOB5 的并不相同。由于存在这些区别，因此必须要采用一个新的行为单元(本例中的主要区别在于 UOB11 中顾客理过发而效果不好，UOB5 中顾客根本没有理发)。可见有些情况下，标签相同的 UOB 并非完全一样。

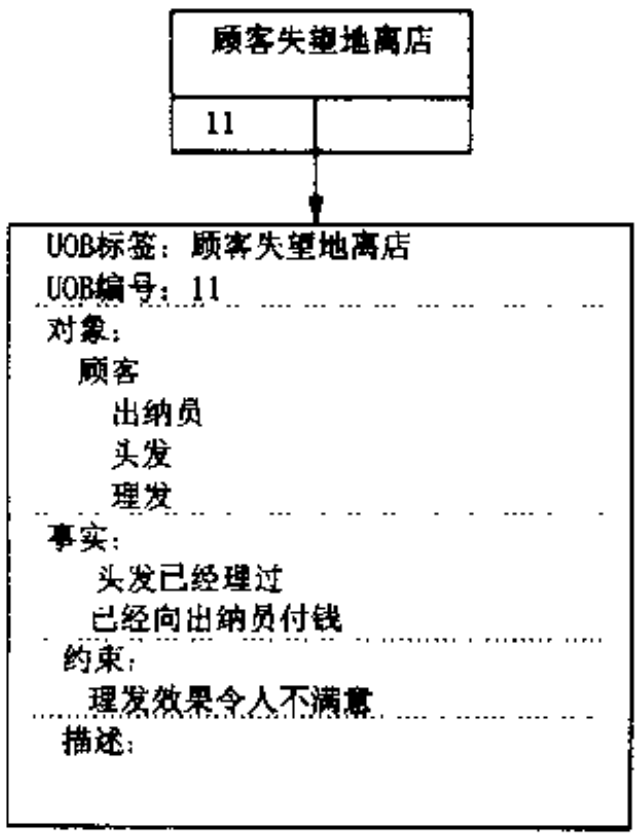


图 3.5.8 行为单元“顾客失望地离店”的细化说明

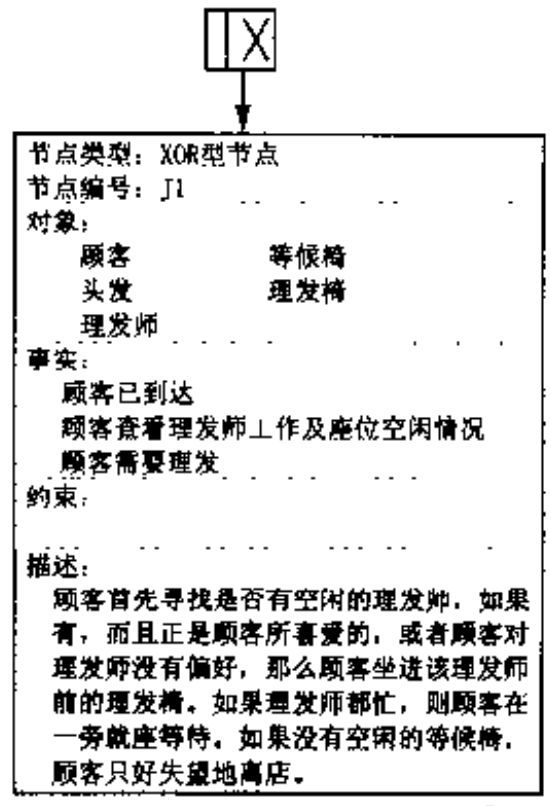


图 3.5.9 交汇点“J1”细化说明表

在完成图(图 3.5.7)中，交汇点 J1 的逻辑关系，可以通过一种借助参照物的细化说明进行较详细地解释(见图 3.5.9)。在参照物细化说明表中，标签栏简要指明其交汇点类型，序号栏内容就是 J1 的序号。本例中的异或(XOR)型交汇点细化说明表，使分析员可对各特定扇出路径的选择规则，进行详细地描述。

另外，对过程流的补充，还包括对联系的定义(图 3.5.10)。在类似本例的简单情况下，这种联系定义可能不必要。联系定义中要标明联系的序号，这样可帮助读者将联系定义与其对应。定义的内容是与联系相关的信息。

联接说明文档

联接编号：1.1
联接类型：顺序
源：　　目的地：
　顾客就座等待(UOB 3)　　顾客阅读杂志(UOB 8)

对象：
　需要理发的顾客
　书架
　杂志
　等候椅

事实：
　书架上放有杂志
　顾客坐在等候椅上

约束：
　或是没有一个理发师空闲，或是顾客所喜爱的理发师没有空。

描述(文本)：
　顾客要么等待一位有空的理发师，要么等待他所喜爱的理发师空闲。

图 3.5.10　联系定义表

5.4　设计对象状态转换网络图(OSTN)

OSTN图的设计，主要是实现参与过程的各对象详细的特征抽取。一般只适用于过

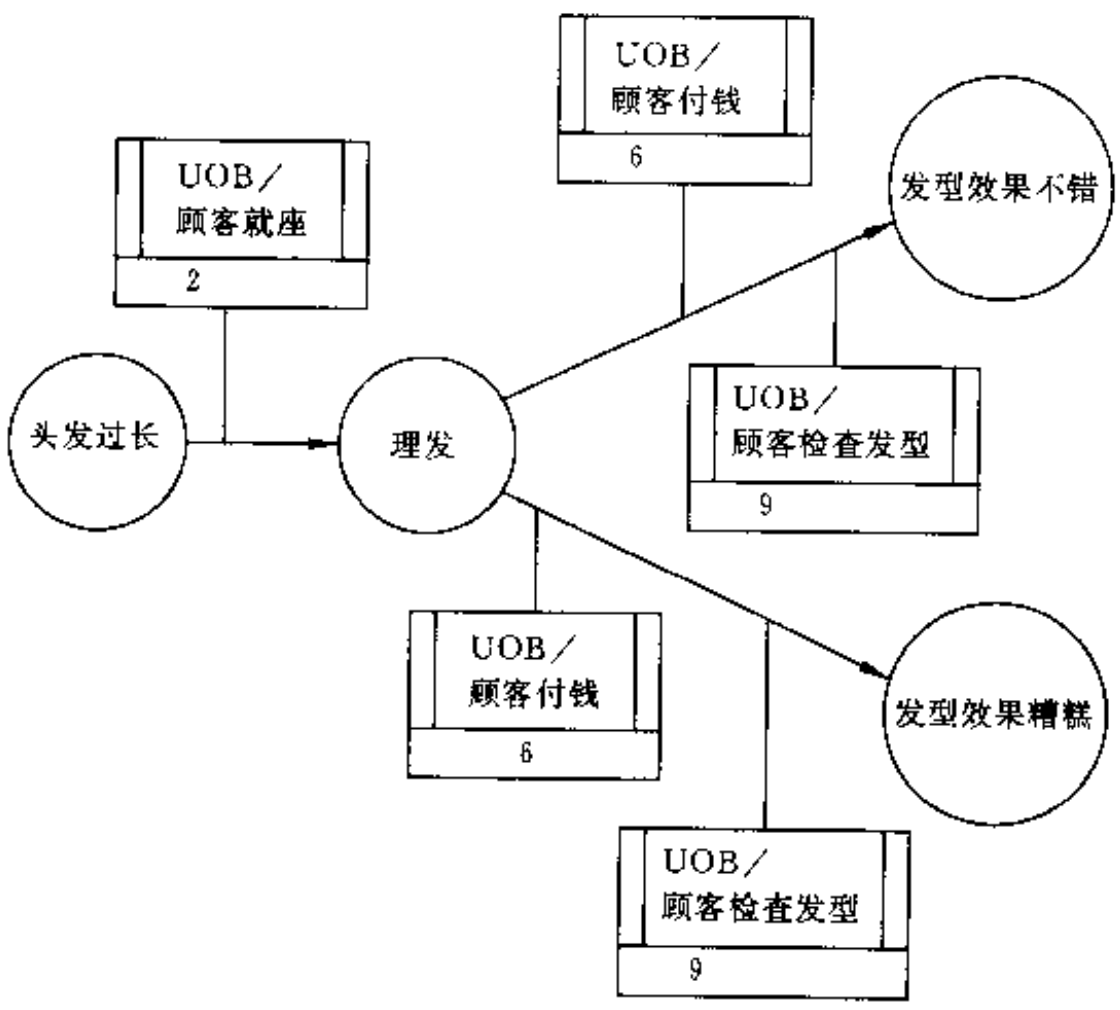


图 3.5.11　例：对象状态转换网络图

程流描述中重要的对象。OSTN 图的设计角度与过程流描述不同，它是以对象为中心进行分析。

假定本例中顾客的头发是重要对象，将其在当前描述过程中的各状态转换，进行概念抽象，将有助于以下的分析。图 3.5.11 就是针对头发的 OSTN 图。其中初始状态为“头发过长”，从这一状态出发，进入“理发”状态，进而转换为两种：“发型效果不错”和“发型效果糟糕”。图 3.5.11 中，OSTN 还包括对各对象状态相对独立的描述。

第 6 章　对 IDEF3 过程描述的理解

IDEF3 过程流描述的主要目的，在于为特定系统或组织的运行方式提供准确的表述，从中可以得到对过程流及与特定场景相联接的对象状态转换的实际描述。IDEF3 描述的评审人，可能并非分析员，但与分析员一样，要保证其中事实的有效性。IDEF3 描述的读者，则可能要从别人创建的描述中获取知识。对于评审人与读者而言，IDEF3 过程流描述的阅读和理解过程如下文所述。

对 IDEF3 过程流描述的阅读，要从场景中最左侧的 UOB 开始。一般来讲，阅读顺序是由左向右。为了解系统概貌，可先将其 UOB 匆匆浏览一下。其中，读者要注意描述中行为单元的顺序关系及逻辑层次。要想分析细节，就要进一步阅读对各 UOB 及其联系的细化说明或描述。系统地研究描述中所隐含的逻辑关系，才能加深我们对 IDEF3 过程流描述的综合理解程度。

6.1　阅 读 步 骤

IDEF3 过程流描述，是将收集到的系统中各事实结构化地组织起来。阅读方法随读者本身及他期望获取的信息量而变。

由于阅读过程的因人而异，很难对这一过程进行严格规范。有些人乐于通读全图后，为便于理解再将其进行逻辑分组，随后在所选部分描述中，为理解各 UOB 和联接之间的关系，对分组进行分析。描述中各部分的意义弄明白后，再考虑其间逻辑关系及交汇点，整个系统的轮廓就会显现在面前。大致的阅读过程如下：

1　仔细阅读目的、范围、所描述场景的目的、以及 IDEF3 过程流描述的观点；

2　对描述中各行为单元、联接和交汇点，从左至右浏览一遍，对其具有大致印象，并基本理解场景中的流逻辑关系；

3　从左至右将流图划分为概念或逻辑上相对完整的逻辑结构，它们是对行为单元、联系、细化说明及交汇点的组合。这种逻辑组合体，可以是过程的通路，其本身还会包含结构或子结构。为了理解的目的，进行这种结构或子结构的划分，像对全图的划分那样；

4　由最左侧的结构入手，根据以下阅读指南，从左向右地读图：

(1) 阅读行为单元及细化说明；

(2) 分析联接，注意在联接说明中发现的信息；

(3) 在给定结构范围内研究各参照物；

(4) 以基本的逻辑结构为单元，一次一个地对整个描述思考一遍；

(5) 遇到交汇点，根据不同条件循着相应路径读图；

(6) 检查路径的设置是否与描述中的逻辑关系一致。

某些读者可能并不需要仔细研究，他们常采用下文介绍的更简便的方法。

6.2 举例说明 IDEF3 过程描述的一种快速阅读方法

许多 IDEF3 过程流描述的读者，在阅读中会依照 6.1 节介绍的过程读图。不过，他们希望能从个人情况出发，进行快速阅读，并在过程中获得经验。我们下面举例讨论一种阅读场景图的方法。其中，对任何行为单元分解的阅读要点，是略加修改地重复应用的。且一般来说，UOB 分解是在读完并理解场景图以后再读的。

6.2.1 基本轮廓

阅读过程中的关键一步，是要理解对应于所描述的现实生活情况的基本轮廓。这要通过对目的、范围、所描述场景的目的、及 IDEF3 过程流描述的观点等的阅读与分析。这些部分确定了流图的界限，并告诉读者一些顶层图信息（尤其是对过程比较熟悉的读者）。另外，也指明了期望的细化分解的层次。

6.2.2 浏览

从左至右地浏览使读者逐渐熟悉进程描述，包括图中的对象、联接及交汇点。它并非是对流图的深入研究。确切地讲，它使读者对被描述过程有一个大概的印象，并能对场景中的逻辑流有一般的理解。

6.2.3 理解场景描述

在这一步，读者对一个场景（或行为单元分解）相关的过程流，能有更细致的理解，这是最为个性化的通信过程的一部分，所用时间也最多。为便于理解，可将流图进行划分。划分的原则主要根据流图的结构。对图 3.6.1 所示的 IDEF3 流图可作像图 3.6.2 所示那样的划分。

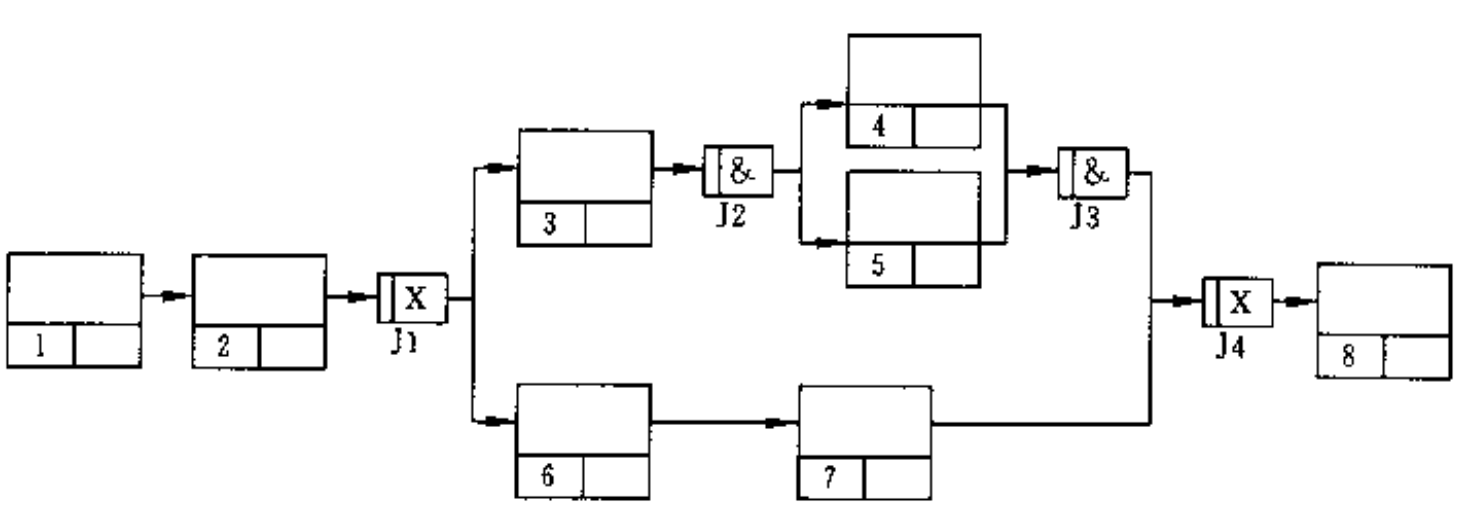


图 3.6.1 例：IDEF3 流图

图 3.6.2 中，流图首先划分为 4 个主要结构：A，B，C 和 D，代表 4 种过程路径。进一步研究发现 B 又可划分为子结构 b1，b2，b3 和 b4。若再向下划分已经没有意义，因此只需

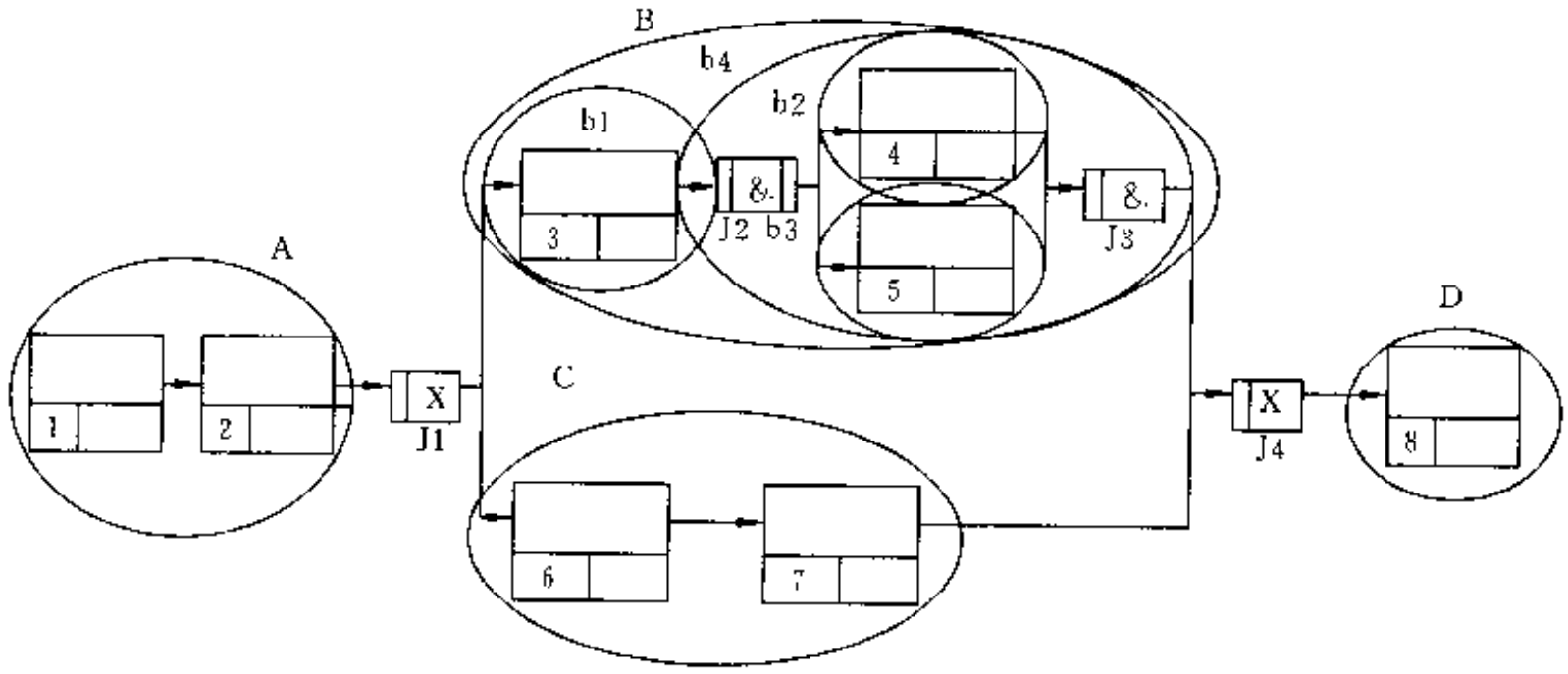


图 3.6.2　对流图的划分

对现有的这些结构进行分析。

将各结构从最左侧用数字 1 开始，标明序号 1～8(见图 3.6.3)。读者会从左向右进展，并做以下的工作：

(1) 阅读 UOB 及细化说明；

(2) 分析联接，注意从联接说明中发现的信息；

(3) 考虑与所选结构相关的全部参照物。

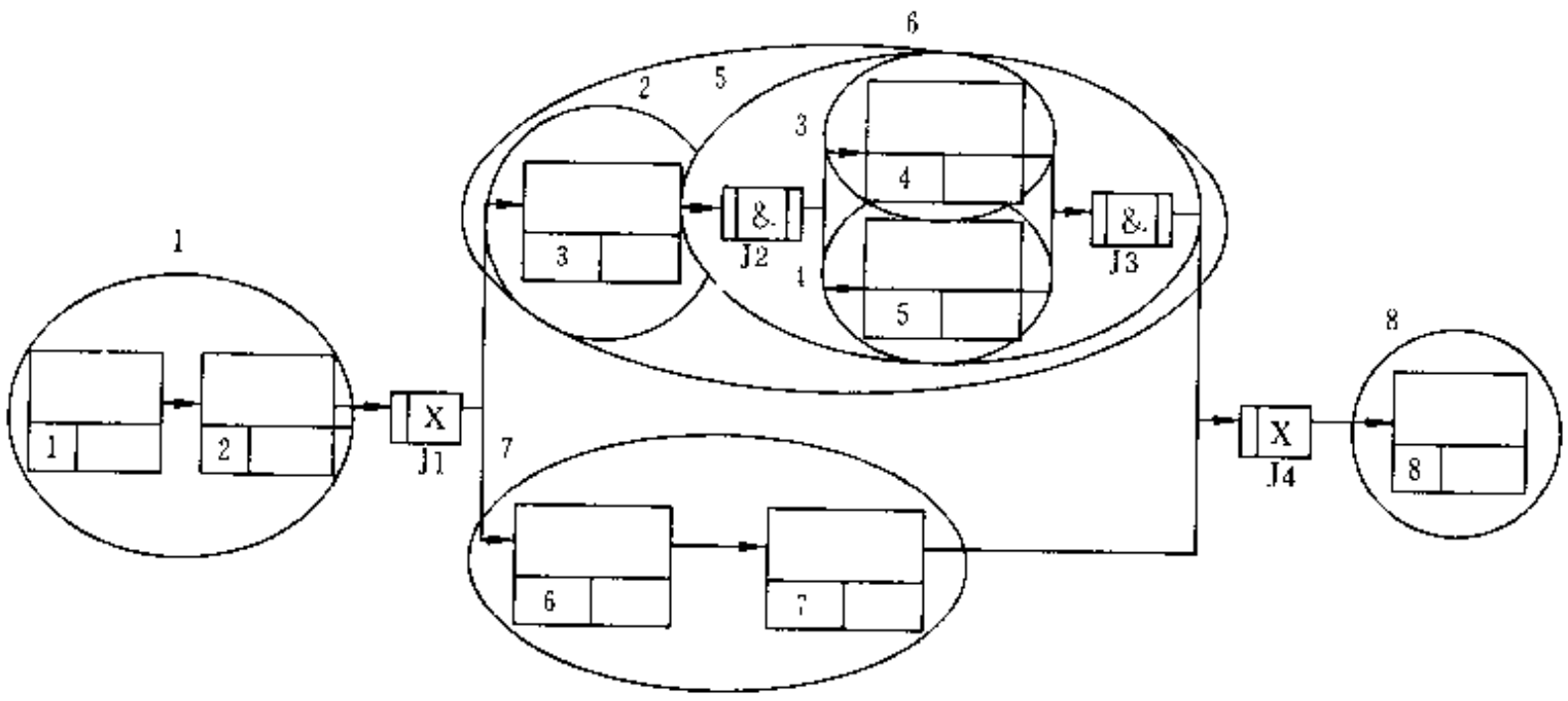


图 3.6.3　分析流图中各结构

对图中的结构 1 理解之后，读者会在结构 6 或结构 7 之间选一，这时并不需要立即考虑交汇点 J1。若先考虑结构 6，就要分析其中的子结构 2 或 5。对结构 2 的分析，还应包括行为单元及其细化说明。结构 5 稍微复杂一些，因为它可以分解为子结构 3，4。完成结构 3 和 4 的研究后，就从整体上分析结构 6。这过程包括对从结构 2 到结构 3，4 的扇出交汇点 J2 的逻辑关系的理解。为理解交汇点 J2，读者就要对扇出路径和约束条件加以研究。一

般说来，交汇点逻辑分析，都是从其扇入、扇出路径、及选择路径的条件入手分析的。结构5则是通过分析扇入交汇点J3的逻辑来研究的。

同样从左至右对结构7进行分析：

(1) 阅读行为单元6,7及其细化说明；

(2) 由左向右地研究各联接及蕴含其中的信息；

(3) 考虑当前结构范围内的参照物。

在对结构6和7的分析完成之后，就可以考虑扇出交汇点J1。这儿，读者在从结构1开始浏览时，要注意出现分叉的原因，以及各条扇出路径的约束条件。

下一个要分析的元素是扇入交汇点J4，它是由结构6和结构7产生的过程通道汇合的地方，读者还要做一次包含分析交汇点J4逻辑的浏览，注意两个过程流通道汇合的条件。

最后，我们通过阅读行为单元8及其细化说明，和相关的参照物来完成对结构8的分析。至此，读者就可以对全图进行一次通读了。通读仍然要从左端开始，直至结构8，但要包括其中所有的交汇点。

第 7 章　使用 IDEF3 方法的建议

本章概略讲一下实际使用 IDEF3 方法中的一些建议。

7.1　如何设计一套有效的 IDEF3 流图

IDEF3 作为一种方法，它以结构化、直观化的方式来获取对真实世界的描述。IDEF3 语言，支持系统操作过程中局部知识的获取。在如何组织已有描述问题上，这种方法给出相当大的自由度。其语言的语法，对于涉及有效的图形安排方面的限制也并不多。这些限制或规则，确保过程描述中的语法或句法，最大程度上实现使用者的意图。而且，这些有效性检查，力图使语言的使用者之间，通过这种不会含糊的通讯手段，来加强过程流描述中的标准化。

“验证”(Validation)，是检查和保证所构成的 IDEF3 进程描述的有效性的一个过程。其中分为三种验证形式：句法的、语义的及模型理论的。句法验证活动，针对 IDEF3 流图是否遵照 IDEF3 语言的句法规则。这种验证有时称为“核实”(Verification)。语义验证，则保证 IDEF3 流图说明准确，与领域专家的看法一致。模型理论验证，通过规范的理论框架对描述的一致性、完整性进行检查。

IDEF3 使用者要认识到，现有系统(AS-IS)与未来系统(TO-BE)描述的不同之处。前者是对运行中的过程或组织的主张的集合；后者则着眼于系统未来发展情况。现有系统描述中，如出现建模理论性错误，只是说明我们对现有系统当前工作情况的知识，尚有局限性。对未来系统描述而言，则指出了系统设计中可能存在的错误。也就是说，当发现现有系统描述中存在不一致性时，系统设计人员要保证未来系统的描述中，避免出现此类问题。

7.1.1　IDEF3 过程描述的模型理论验证规则

IDEF3 的句法验证规则，我们已在第 3 章讲过，在第 4 章组表评审过程，对语义验证有专门论述。下面的模型验证规则，以附加的句法规则的形式进行表述。因此，它在前面的句法验证规则基础上，多了对流图的逻辑分析的约束。下面是作 IDEF3 描述模型理论验证的基本规则：

(1) 每个描述必须有场景名称；

(2) 每个描述和场景必须有需求、目的和范围说明；

(3) 每个场景(不同于分解子图)和分解子图中，只能有一个左端点，它或是行为单元(UOB)或是交汇点；

（4）每个分解子图只能有一个右端点，不是分解子图的场景则可有多个右端点；

（5）任何分解中都不允许出现两个以上不连接的图形；

（6）任何 IDEF3 元素（UOB 或交汇点）不许直接反过来连接其本身。

7.1.2 行为单元分解规则

下面是行为单元及其分解的规则：

（1）每个行为单元必须要有标签（名称）；

（2）每个分解必须要有名称；

（3）分解中同层的 UOB，要依照其产生的顺序给出序号，它们只须在本层唯一，在整个描述、场景或流图中可以不唯一；

（4）从一个行为单元不能引出多个先后顺序的联接，若有这种需求，说明应在此处设置一个扇出交汇点；

（5）多个先后顺序的联接，进入同一行为单元是允许的，不过，它们相应的时序和逻辑的语义解释，要在该行为单元的细化说明中阐明。

许多交汇点与相关联接的检验，与对结构的识别有关。结构是行为单元、联接及交汇点的任一种逻辑句法的组合。图 3.7.1 中，A，B，C 是结构，它们可以复杂亦可以简单。复杂的结构，其实是 IDEF3 流图中扇出交汇点与相应扇入交汇点间的部分（如图中 A）。简单结构，则指不包括交汇点的 IDEF3 流图的片段（如 B 和 C），它可能是复杂结构中的一部分。许多交汇点和联接规则有助于确保复杂系统的句法有效性。针对 IDEF3 描述中交汇点和联接组合的模型理论验证，提出下面的系列句法规则：

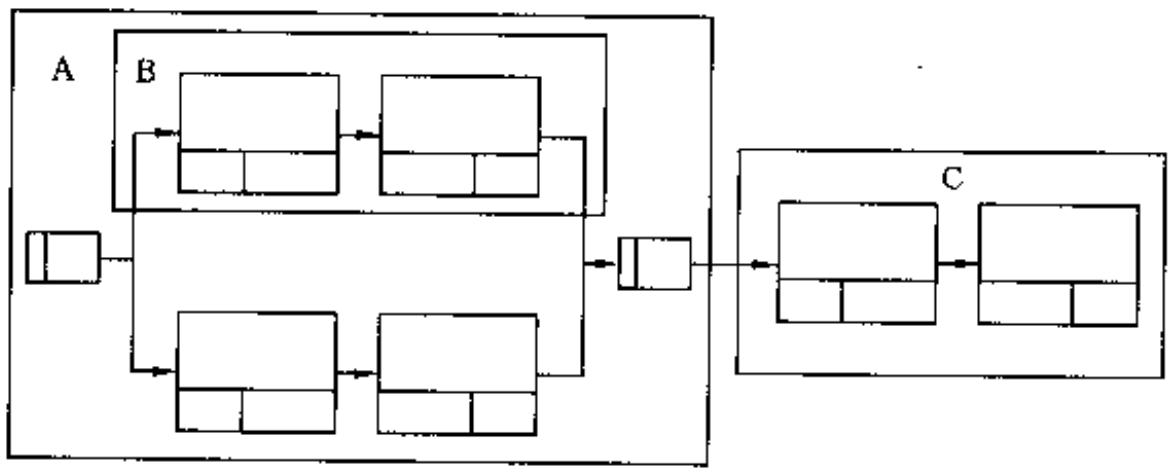


图 3.7.1 流图结构

（1）每个扇入交汇点要有相应的扇出交汇点；

（2）不允许从复杂结构内部引出指向结构外的任何点的环路。如图 3.7.1 中，从结构 A 到结构 C 之间不能出现环路；

（3）“扇入的与”型交汇点与“扇出的或”型交汇点不能匹配；

（4）“扇入的与”型交汇点与“扇出的异或”型交汇点不能匹配；

（5）“扇入的异或”型交汇点与“扇出的与”型交汇点不能匹配；

（6）每个扇入交汇点必须要有两条以上先后顺序的联接进入；

（7）每个扇出交汇点必须要有两条以上先后顺序的联接离开。

7.2 构造 IDEF3 流图中的通病及指南

7.2.1 “扇出的异或”型交汇点后带有“扇入的与”型交汇点

“异或”型扇出交汇点后面，不允许有“扇入的与”型交汇点来结尾；否则只能是一种模型理论上不一致的情况。换句话说，真实世界中类似情况的例证肯定失败。为便于说明，考虑图 3.7.2 中 IDEF3 流图。

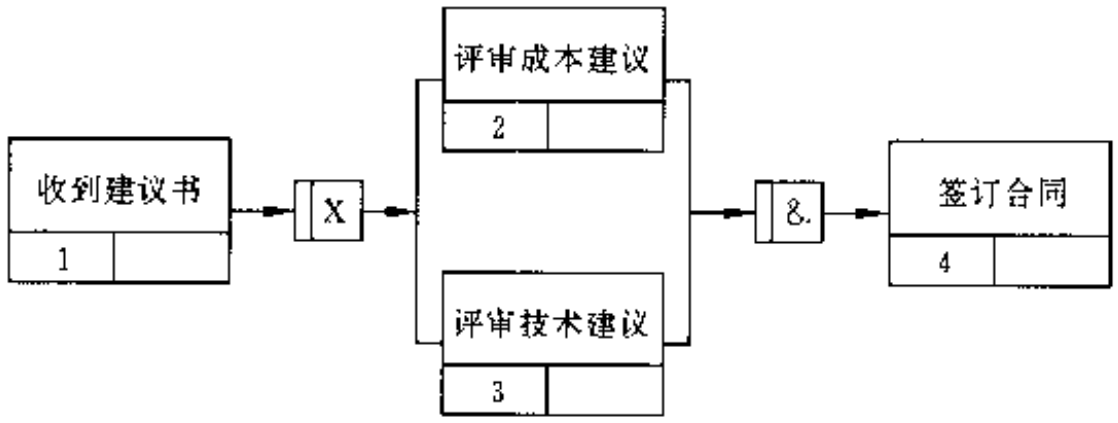


图 3.7.2 例：流图中无效的异或(XOR)和与(AND)结构组合

图 3.7.2 中，在“收到建议书”行为单元之后，“异或”型交汇点引向两个行为单元，在给定的例子中，只有一个行为单元可以实现，要么是“评审成本建议”，要么是“评审技术建议”。这样的话，行为单元“签订合同”就无法实现，因为“扇入的与”型交汇点前的行为单元必须同时实现。然而根据上文，无法满足这一要求。为什么有人会犯这样的错误呢？现实描述中，经常会存在未被发现的不一致性，这就需要我们加强有关的管理工作。如前文的情况，合同就永远签不成，这样 IDEF3 图就揭示了一个组织上的问题，并要解决这一矛盾。对现有系统过程描述中，作者可能会发现这类组织图上的问题。即使它并非模型理论上有效，但可能会是一张语义有效的图。在对一些系统、组织结构或过程的未来系统描述图中，出现此类的结构也都是错误的。在以上两种情况下，描述验证过程应能识别出这种作为 IDEF3 流图错误的结构。

7.2.2 一个行为单元盒上出现多个先后顺序联接

考虑图 3.7.3 中的喷漆工艺过程，零件在喷漆、烘干之后应接受质量测试。若结果指示喷漆量不足，零件必须重新喷涂；若检验合格，则将零件运离喷漆车间。图 3.7.3 中的流图存在模型理论性错误。因为“检测表层”行为单元引出的分支，其语义不明确。该行为单元两个分支中只能有一个分支可选择，而这一事实在流图中并未说明。解决这一问题的方

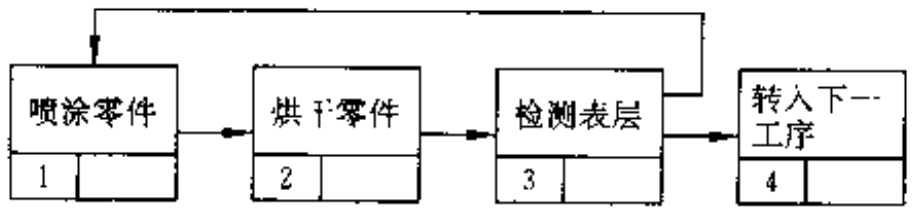


图 3.7.3 例：流图中存在语义多义性的分支

法是，采用另外的事实以消除语义的歧义性。这样就引入了额外的“异或”型交汇点，修改后的流图见图 3.6.4。

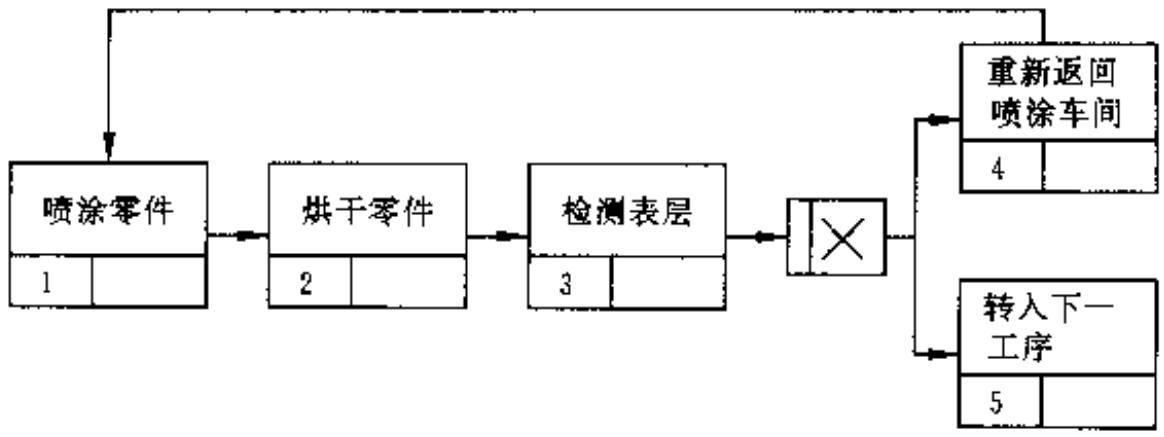


图 3.7.4　修改后的喷涂车间过程流图

7.2.3　场景或分解图中存在多个左端点

该类错误如图 3.7.5。

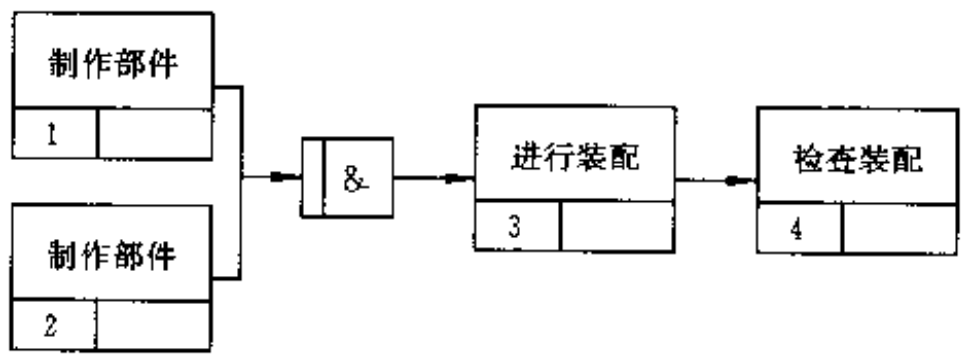


图 3.7.5　具有多起点的 IDEF3 场景

对此类常犯的模型理论性错误的纠正方法，是在流图的左侧增加一个交汇点盒子，如图 3.7.6。本例中，正确的交汇点类型应是“扇出的与”型。

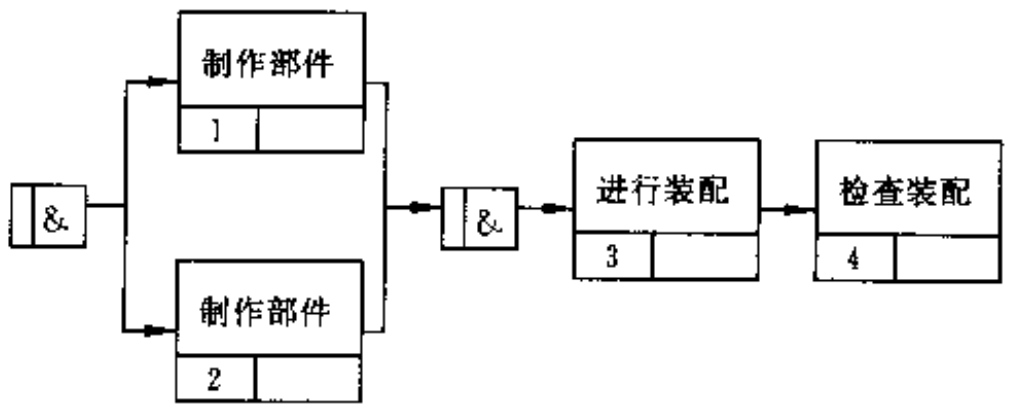


图 3.7.6　修改后的装配车间场景 IDEF3 流图

第 四 篇

IDEF4 面向对象的设计方法

第1章 概 述

本篇的目的是：提供 IDEF4 面向对象设计方法的全面描述，以指导人们熟练地运用 IDEF4 产生高质量的面向对象的设计。

本篇包括下述内容：(1)IDEF4 句法的描述及采纳每个句法元素的动机，(2)用 IDEF4 进行面向对象设计的开发过程的描述，和(3)应用 IDEF4 进行设计的例子。在上述范围内，这篇 IDEF4 方法报告是为下列读者所设计的：(1)正在寻找一种设计语言的面向对象的编程员，和(2)学过一种面向对象的编程语言(OOPL)，想知道怎样设计出好的面向对象的程序的编程员。

开发 IDEF4 方法的动机，来自潜在的误用功能强大的面向对象方法的可能性，这种错误的使用，会导致低质量的设计。面向对象的哲理和开发实践，显示了产生具有所需要的生命周期特性(如模块化，易维护，可再用)的代码的能力。另外，面向对象的编程方法，也显示了其最主要的优点，在于可以很容易地生成软件代码，使人们能更有效地编制和维护代码。但是与此同时，这种编制代码的容易性，使得产生设计糟糕的软件也变得更容易了，这将导致非模块化、难维护和难以再使用的系统。IDEF4 的目标，就是帮助人们正确使用面向对象的编程技术，以保证更有效地使用这项技术。

第2章 简 介

越来越多的人认识到，面向对象的编程方法产生的模块化、易维护及代码的可再用性，在传统的数据处理中也可以实现。这个方法支持大的、复杂的、分布式的系统，在数据水平上进行集成的能力，已被证实。这也是它引起传统数据处理领域越来越广泛的兴趣的一个主要原因。面向对象的方法为开发者提供一个抽象的程序的视图，这个视图由一系列状态保持对象组成，这些对象通过它们之间关联的协议来定义程序的行为。

面向对象的编程语言包括CLOS (Common Lisp Object System)，C++，Smalltalk Object Pascal 等。IDEF4 正是为使用这些面向对象语言的软件设计者提供的设计方法。由于有效地使用面向对象的方法，要求一种完全不同于传统的过程化或数据库语言所使用的思考过程，因此标准的方法，像结构图，数据流图和传统的数据设计模型(层次的，关系的和网状的)都不再适用。IDEF4 面向对象设计方法(OOD)力图提供必要的工具，来支持面向对象设计的决策过程。IDEF4 主要的设计目标为：(1)为产生面向对象的设计提供支持，该设计的实现将具有所需生命周期特性，并缩短总的实施开发时间；(2)使得评价面向对象的代码是否符合设计和是否具有所需的生命周期特性，变得容易进行。

IDEF4 用一种方法，保存了有关面向对象设计的信息，使得在以前方法开发的努力中获得的许多概念的和记号标志的进展情况，得以保留。这种和以往工作的一致性，可以帮助应用软件工程师，学习和有效地使用 IDEF4 建模技术。更进一步地，IDEF4 囊括了设计面向对象的数据库和编程环境的最好的实践经验。因此，IDEF4 致力于对下述内容的标识、操作、显示和分析：

(1) 对象定义：包括属性和它们的类型；

(2) 对象结构：包括继承层次或网格，和个别对象相对于其他对象的组成；

(3) 个别对象的行为：包括方法说明和约束(事前，事后和过程中的条件)，控制实例化，删除和其他为对象所要求的行为的限制；

(4) 系统例程的协议：包括对象例程在对象继承网格中的位置。协议还包括：

- 参数的类型、数量和顺序，
- 每个协议项行为的抽象描述，
- 各个对象类型的抽象行为的说明。

作为一种设计方法，IDEF4 是专门用来说明面向对象系统设计的组成部分的，以利于在系统开发过程的设计阶段，进行管理。它运用一种独特的组织机制，来保证设计的模型，不会随着项目的日益增大而变得麻烦和难以使用。

从概念上说，一个 IDEF4 设计模型由两个子模型组成：类子模型和方法子模型(见图4.2.1)。两个子模型通过分配映射联系起来。这两个结构，包括了设计模型中体现的全部

信息。子模型是特定类型信息的一个非常概括性的分组，而数据单则显示了非常局部化的，通常是文本形式的，与特定的数据类型或方法集相关的信息。除了协议图和客户图通常比较小外，图 4.2.1 中的其他图的大小，可以从单个盒子到代表整个子模型中所能得到的所有特定类型的信息。这些图是子模型的视图。分配映射将两个子模型联系起来，并用继承图和方法分类图中的某些可选的注释来表示。在图 4.2.1 中，方角盒子代表子模型类型和分配映射，圆角盒子代表图的类型和 IDEF4 模型的数据单。

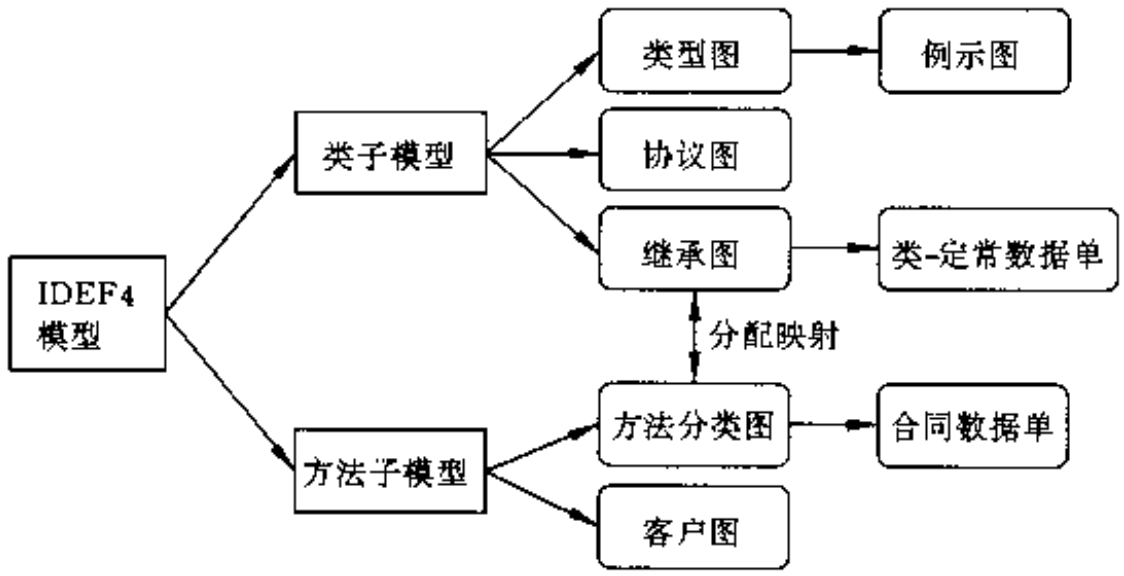
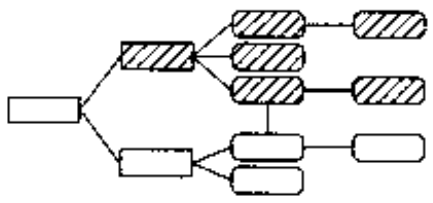


图 4.2.1　IDEF4 模型的组织

换句话说，IDEF4 将面向对象的设计(OOD)活动分成离散的、可管理的块。每个子活动，由阐明必须做出的设计决策及它们对设计其他方面影响的图形句法来支持。没有单个的图能显示 IDEF4 设计模型所包含的全部信息，这样就减少了混淆，使得快速检查所需的信息成为可能。仔细设计的图类型间的重叠，用来保证不同子模型间的兼容性。

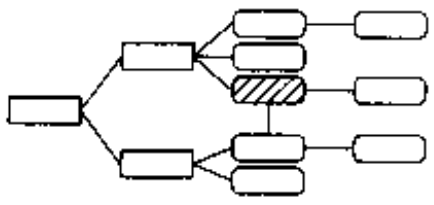
由于类和方法子模型的尺寸太大，设计者从来看不到这些结构的全貌，而是用较小的图的集合来有效地获取类和方法子模型所表达的信息。组成类子模型的图类型有：(1)继承图，(2)协议图，(3)类型图，(4)例示图。组成方法子模型的图类型有：(1)方法分类图，(2)客户图。这六个图类型，在 IDEF4 设计中分别被用来表示：(1)用类型图表示数据对象类的组成结构，(2)用继承图表示继承关系，(3)用方法分类图表示方法约定，(4)用协议图表示面向对象设计的协议，(5)用客户图表示功能分解。下面各节将给出各个图的解释，并对它们的目的作一个简要的描述。

2.1　类 子 模 型



类子模型由协议图，类型图，继承图，例示图和类-定常数据单 CIDS(class—invariant data sheet)组成。类子模型显示了类继承关系和类组成结构。

2.1.1　继承图



继承图详细说明了类间的继承关系。例如，图 4.2.2 显示了类“填满的矩形”继承了直

接来自类“矩形”和类“填满的对象”，以及间接来自类“对象”的结构和行为。

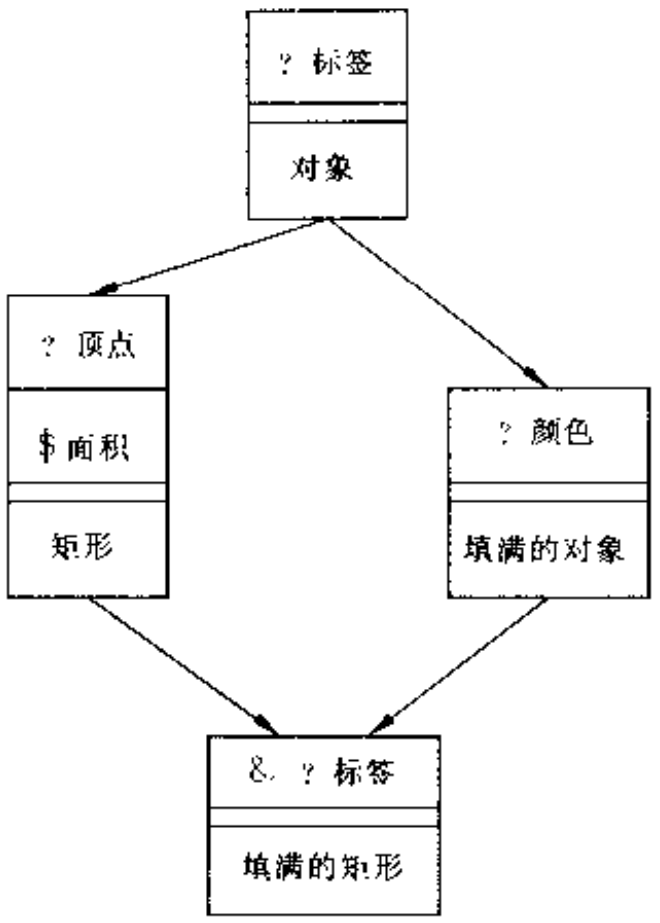
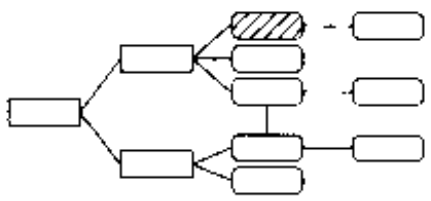


图 4.2.2 继承图

2.1.2 类型图



类型图详细说明了通过某个类的属性定义的类之间的关系；这些类，或者具有作为另一类的实例的值，或者由那一类的实例组成。图 4.2.3 显示了一个类型图。图中，类“双轮的”具有特征“轮-1”和“轮-2”，这些特征返回来是类“轮子”的实例；而类“轮子”，就其特征“直径”来说，又返回来是类“实型”的一个值。

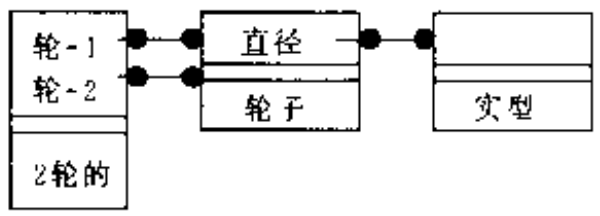
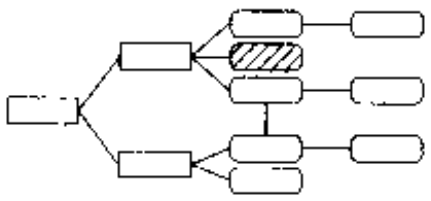


图 4.2.3 类型图

2.1.3 协议图



协议图详细说明供方法援引的类参数类型。图 4.2.4 显示了“填充的封闭对象”：“多边形”例程-类对的协议图。从图上可以很明显地看出，“填充的封闭对象”将接受类“多边形”的一个实例作为它的主要的(自身)参数，接受类“颜色”的一个实例作为次要参数，并

将返回一个“多边形”的实例。

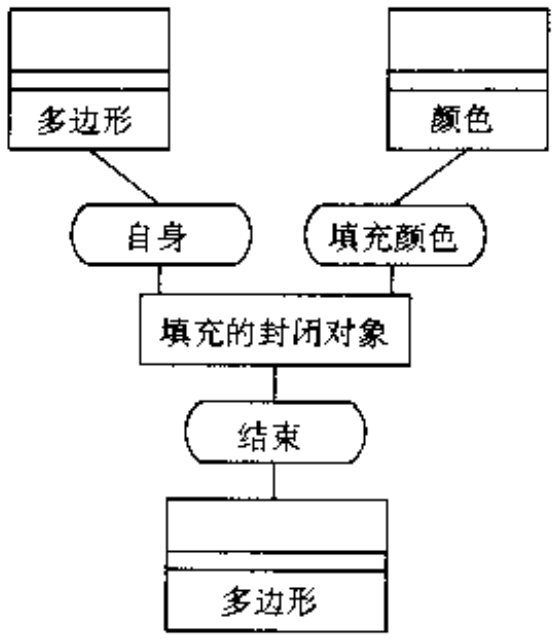
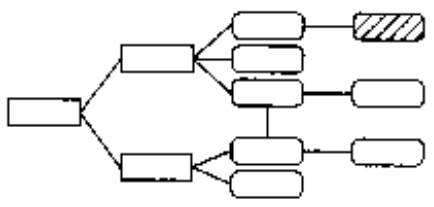


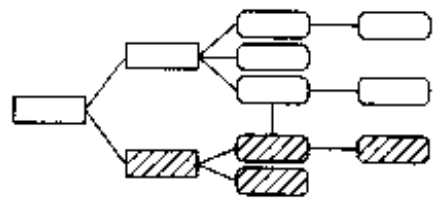
图 4.2.4　协议图

2.1.4　例示图



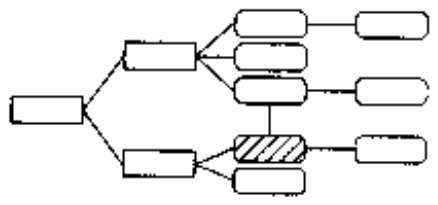
例示图和类子模型中的类型图相联系。例示图描述了用来验证设计的实例化对象间，合成联接的可能的情况。

2.2　方法子模型



方法子模型由客户图、方法分类图和合同数据单(CDS)组成。下面各节将给出这些组成部分的概述。

2.2.1　方法分类图



方法分类图，通过行为的相似性对方法进行分类。方法分类图根据分类中表示的加在方法集上的限制，来划分特定的系统行为类型。箭头指出加在方法集上的附加的限制。图 4.2.5 显示了“打印”的方法分类图，分类中的方法集，是根据加在每一集里方法上的附加约定来分组的。在这个例子中，第一个方法集“打印”，有如下约定，即“对象必须是可打印的”；“打印文本”方法集约定则一定有这样的限制，“要打印的对象必须是文本”。

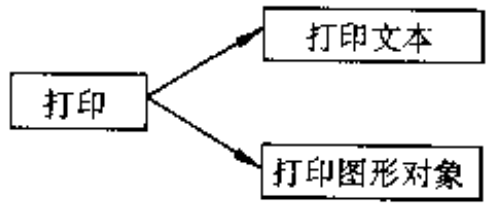
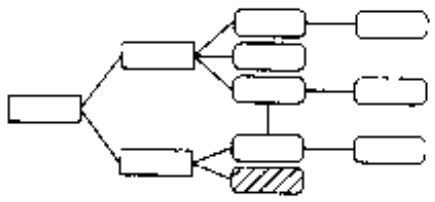


图 4.2.5　方法分类图

2.2.2 客户图



客户图说明了客户和提供者这一例程-类对。双箭头从被调用的例程指向调用它的例程。图 4.2.6 显示了一张客户图，其中附着于类“可重新显示的对象”的“重新显示”例程，调用了类“可擦除的对象”的“擦除”例程和类“可绘制的对象”的“绘图”例程。

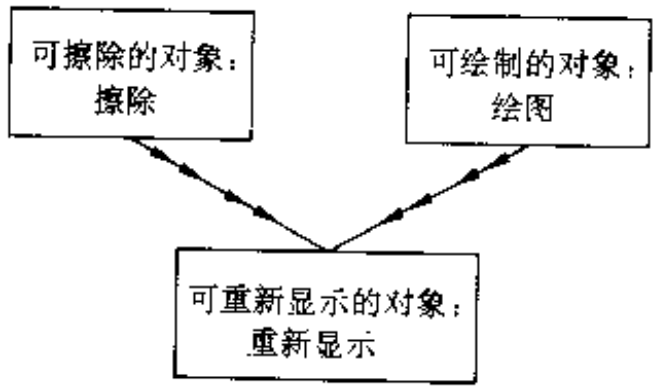
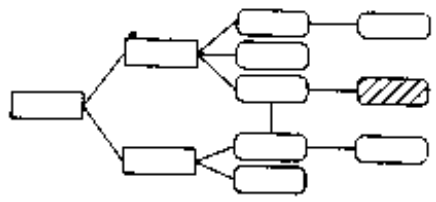


图 4.2.6 客户图

2.3 数 据 单

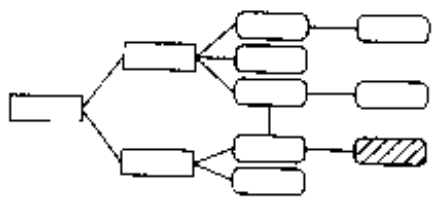
和图一起的，还有两种专门的数据单：类-定常数据单(CIDS)和合同数据单(CDS)。要理解这些图表达的信息，需要下一节所提供的 IDEF4 的基本概念的描述。

2.3.1 类-定常数据单(CIDS)



类-定常数据单和继承图相联系，详细说明应用于一个特定对象类的每一个实例的限制。每一个类都有一个 CIDS。

2.3.2 合同数据单(CDS)



合同数据单和方法分类图中的方法集相联系，详细说明，方法集中实现了的方法必须满足的合同。每一个方法集都有一个 CDS。

第 3 章　IDEF4 面向对象的概念

IDEF4 是用于面向对象设计的方法，而不是面向对象编程的语言。它被开发来供面向对象系统的设计者使用，而不考虑用于系统实施的面向对象的语言。但是，在讨论 IDEF4 设计的组成部分前，必须先理解面向对象编程语言的基本概念。本节将描述基本的面向对象编程元素和它们在 IDEF4 设计中的表示方法。另外，面向对象编程的基本元素（类，特征和方法），也将和它们如何出现在 IDEF4 模型中的基本描述一起，介绍给读者。

以对象为基础的传统，通常是与消息传递相结合的（即包含了操作名称和参数的消息，被送往接收器）。而在数据驱动的传统中，则把接收器和操作名称放在一起来选择调用的方法。

面向对象的系统有 4 个焦点（见图 4.3.1）：以对象为中心，以类为中心，以操作为中心和以消息为中心。以对象为中心的系统，提供最少量的数据抽象，即“状态”和“操作”，而没有专门提供类。以类为中心的系统，把类放在第一位。类要：(1)描述对象的结构，(2)包含定义行为的方法，(3)提供继承拓扑结构，(4)详细说明对象间的数据共享。以操作为中心的系统，把操作放在第一位，操作本身包含对象的所有行为，但可以用类或对象来描述继承和共享。以消息为中心的系统，把消息放在第一位：消息是首要的类，它把操作和数据从一个对象传递给另一个对象。在以类为中心和以对象为中心的系统中，被强调的是类或对象，而不是操作。

方面	以消息为中心	以操作为中心
以类和类型为中心	Smalltalk C++ virtuals	CLOS FORTRAN Overloads C++ Overloads
以对象为中心	Self Actors	Prolog Data-driven

图 4.3.1　面向对象语言的方面

3.1　类

面向对象编程，通过对象的建立和操作来生成程序，是一种非常直观的编程风格。这种编程风格，由于它比传统的编程风格更自然，使得对更多的人来说，编写代码变得较容易了。遗憾的是，对更多的人来说，这种由面向对象产生的容易性也容易产生低效、难维护

和不可再用的代码。IDEF4 的主要的目的之一，就是要帮助人们充分利用这项技术而避开潜在的陷阱。

用面向对象语言书写的程序，定义类并为类定义方法。初学面向对象编程的人，常常会混淆类型、类和对象这些术语。

“类型”至少有五个不同的含义，包括：声明类型，表示类型，签名类型，方法类型和值类型。声明类型用于说明一个常量，其值可以被存放在一个特殊的变量或结构里。值类型指可以被存放在一个特殊的变量或结构中的对象集。表示类型定义对象（如 C++中的 double 类型）的存储方法。签名类型由可以对对象执行的操作类型来定义。比如说，可以对一个数和一个顶点执行的操作类型就很不相同。方法类型是能为其写出方法的类型。类在大多数面向对象的语言中，是方法类型的例子。

类是一种类型（方法类型），而对象是类型的实例，因此对象也是类的实例。类的描述，是一组局部的状态定义的属性的定义，是定义了该类实例的行为及其与组成系统的其他类实例关系的方法集的定义。换句话说，类是一种数据结构，它包含了状态定义的属性集和用于该类对象的例程集。IDEF4 设计将描述类和与那些类相联系的方法。

3.1.1 类与对象

面向对象和结构化设计和编程，有一个共同的重要概念，即封装或信息隐藏。封装试图将一个程序分为几个模块，以使模块间的交连或耦合最小化。因此，在一个变量或过程可以从一个模块输出前，必须有一个明确的声明。

术语“类”，在面向对象的方法中有多重含义，它指：

- 现实世界中对象的范畴或类型（现实世界的观点），
- 表示对象范畴的数据类型（数据项的观点），和
- 定义数据类型的相关操作的模块（模块的观点）。

术语“对象”，也有多重含义，它指：

- 现实世界中的对象（现实世界的观点），和
- 属于一类或另一类的数据项（数据项的观点）。

对象是类的实例。通常很便于把类的现实世界的和数据项的观点合并，以及把对象的现实世界的和数据项的观点合并。这使得我们可以把对象看作是，附加了需要在计算机内跟踪的一些额外特性的现实世界的对象，而把类看作这些对象的范畴。这还使得我们可以把类看作是，附加了操作类的计算机表示所需信息的对象，换句话说，计算机把对象看作现实世界的对象。可以很容易地表示这些概念的能力，正是面向对象语言优于结构化编程语言（如 PASCAL 语言和 C 语言）的地方。

面向对象的系统中的类，定义了存在于系统中的对象的类型。每个类都包括一个特征定义集，表示该类实例的状态和行为的特征。特征定义集由属性和方法组成。属性定义，被该类的实例用来存储它们的状态，而方法刻画了该类实例行为的特征。

和所有面向对象的方法一样，在 IDEF4 中，类是主要的句法成分。在如用英文 IDEF4 中，类用一个方角的盒子来表示（见图 4.3.2）。类的名字写在盒子的底部双线的下面。IDEF 4要求类名的第一个字母大写。类的特征也显示在类盒子中，私人特征（即那些只

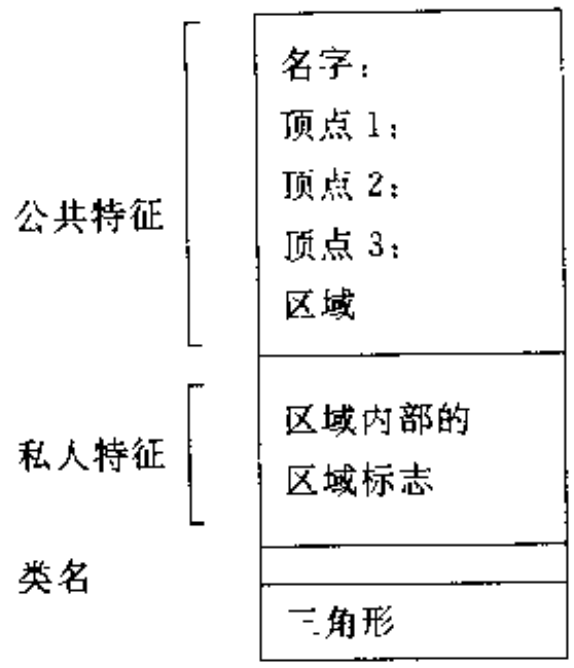


图 4.3.2　IDEF4 中的类盒子

在该类实例内部使用的特征)写在输出线下面，而公共特征(即那些被写入文档、广告，并支持外部使用的特征)写在输出线上面。面向对象的语言像 C++，Eiffel 和 Flavors 都明确支持定义公共和私人特征的能力。附在特征名前而的各种特征符号，可用来提供有关这个特征所扮演的角色的附加信息。对每一个定义了的类，IDEF4 允许其用 2.1 节中提到的类-定常数据单附上类-定常约束。这些类-定常约束代表了有关类的附加信息，这些信息在任何时候对该类中的所有对象都成立。在设计中描述的类-定常量，将提供对设计实现的限制，并作为对类所做说明的一部分。这节介绍的所有概念都将在以后的章节中进行更详细的阐述。

3.1.2　类-继承

也许 OOPL(面向对象编程语言)最突出的特点就是继承，特别是多重继承。多重继承发生在一个类(一个较特殊的类)从多于一个超类(更通用的类)继承特征(属性和方法)的时候。OOPL 中的继承概念，提供了把类的实例组织成相互关联的集合的工具，并允许在继承层次内部，重复使用方法和类特征。从现实世界的观点来看，继承现象相当于一种特殊化的关系。也就是说，继承类(子类)是它所继承的类(超类)的特殊化。

图 4.3.3 显示了 IDEF4 中为类-继承层次建模的一个例子(例示时，为了清楚起见，和讨论无关的细节被忽略)。图中的箭头，由超类指向子类。类“三边形”是类“多边形”的子类。由此可知，“三边形”的任何对象都是“多边形”的特殊化。进一步来说，除非在类“三边形”的定义中专门说明的行为，“多边形”展示的任何一种行为也将被“三边形”展示。在这个例子中，类“矩形”的对象，从类“平行边图”和“四边形”继承特征和方法。例中，类“四边形”是类“多边形”的直接子类，而类“矩形”是类“多边形”的间接子类。每一个子类，继承了与它们的直接或间接超类相联系的特征和方法。

这里有一个重要的问题，就是解决继承网格中特征名冲突的问题。当一个特征在一个线性继承(单一继承)图中被激活时，就可能会有几个具有相同特征名的“特征-类对”。按照惯例，被选中执行的特征-类对，一定来自最特殊的类，因此我们有对特征-类对“最特殊者优先”排序的说法。这在 IDEF4 中被假定是缺省情况。

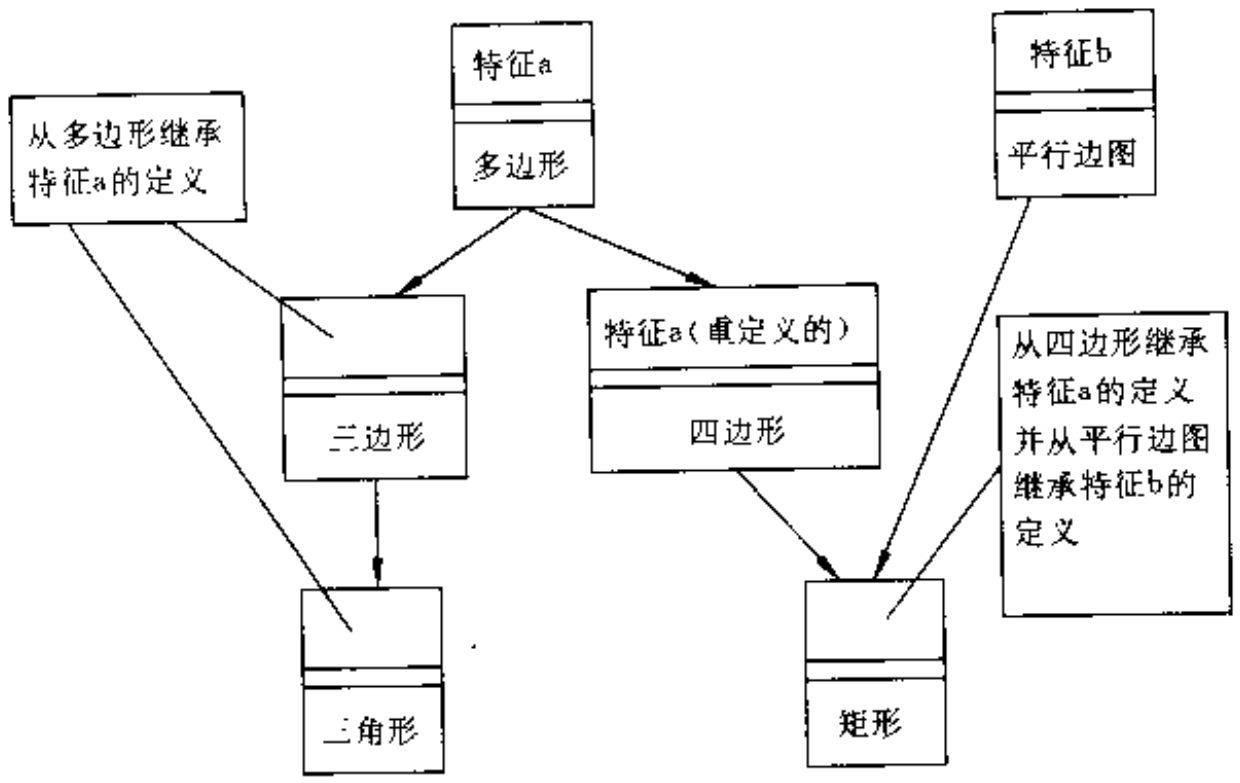


图 4.3.3 类继承

在多重继承的情况下,从一系列冲突的特征-类对中,选择一个特征-类对来执行,是件更加麻烦的事。在 IDEF4 中,缺省情况是,通过在继承网格中冲突的特征-类对的最特殊者优先,拓扑结构排序中,选择最特殊的特征-类对来解决冲突。这种解决冲突的策略,被大部分 OOPL 所采用,像 CLOS,Smalltalk,Flavors 和 C++。

设计者可能并不总是想使用 IDEF4 采用的缺省的策略(即,根据不同的优先权方案来完成调配)。其他可以考虑的优先权方案包括:(1)最不特殊者优先排序,(2)KEE 中的深度优先、最特殊者优先排序,(3)广度优先、最特殊者排序,(4)执行所有可使用的特征-例程对等。在最后一种情况下,设计者也许想指定将返回结果以一种特定的方式结合,这在 OOPL 中被称为方法-组合规范,而在 IDEF4 中被称为方法集组合合同。如果设计者想改变 IDEF4 用于某个特征的缺省的方法集组合合同,就必须在建立在那个特征基础上的方法分类的方法集合同中说明。值得注意的是,重新定义缺省的方法集组合合同,在实践中很少发生,而且应该尽可能避免,因为许多 OOPL 不允许这样做。

从模块的观点来看,继承在宏观上类似于"虚拟拷贝"操作。除了那些在子类中重新定义了的特征和操作,所有的和超类相联系的操作和相似的特征,都被它的子类自动地继承。比如说,在图 4.3.3 中,"多边形"类定义了一个特征叫"特征 a",这个特征将在它的所有子类:"三边形"、"三角形"、"四边形"和"矩形"中被继承。在"三边形"和"三角形"中用于"特征 a"的例程和在"多边形"中的一样。由于在"四边形"中对"特征 a"有附加的合同,所以我们说在该类和它的子类中,这个特征被重新定义了。既然"矩形"是"四边形"的子类,在"矩形"中使用的定义,也将是被重新定义过的。

子类可以看见并使用它的超类的任何特征,但是反过来不行。继承概念和传统的信息隐藏概念相冲突。这种冲突,由于只允许在有控制和有限制的方式中(而且仅仅是单方向的),所以是面向对象的方法功能强大的关键之一。一个正确的结构化的 OOD(面向对象设计),使用继承工具使模块的重复最小化。而 IDEF4 致力于把方法构造到逻辑模块或类中去,以帮助保证所得到的 OOD 达到这个目标。

IDEF4 中，用继承图来显示类之间的继承关系，正如图 4.3.4 所示。继承图提供描述类的、其特征的和任何特征的重新定义的信息。比如说，熟悉 IDEF4 句法的读者可以看出，“填满的矩形”继承了类“多边形”、“矩形”和“填满的图形”的所有特征。进一步，还可以看到，特征“周长”是在“矩形”中被重新定义了的属性。有关继承图的细节将在 4.1 节中讨论。

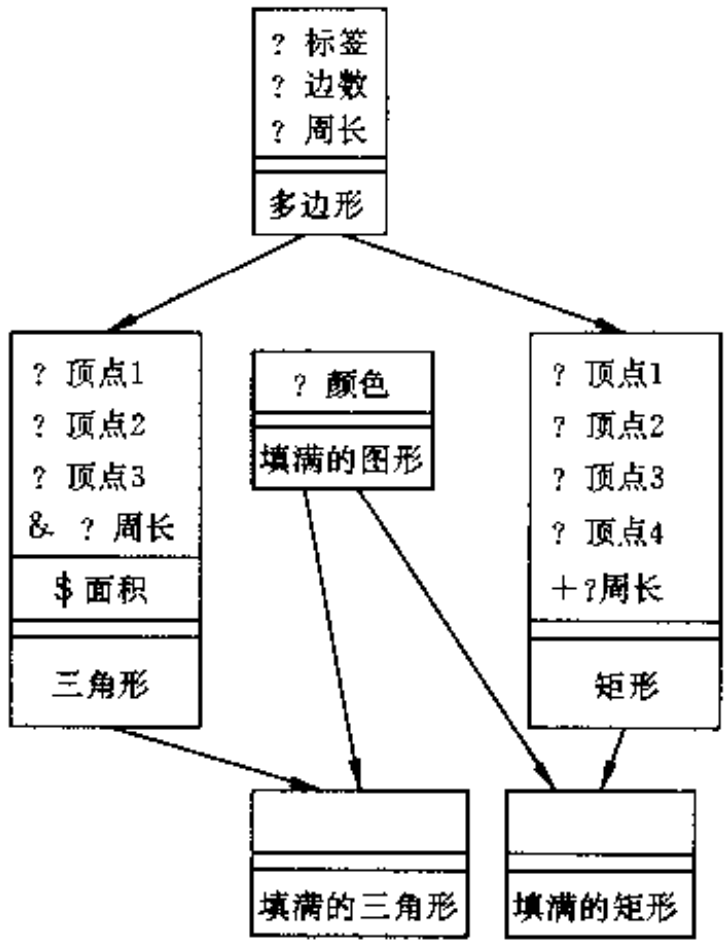


图 4.3.4 一个图形系统的部分继承图

3.2 特 征

特征是类的特殊特性的命名了的表示。特征用来获取一个特殊类的实例的状态和行为。特征可以是有返回值的和(或)副作用的。比如说，类“三角形”可能有一个特征叫“面积”，它返回该类的一个实例的面积(返回值的，value－returning)，可能有另外一个特征叫“移动”，它改变类“三角形”的实例在屏幕上的位置(副作用的，side－effecting)。一个给定的返回值的特征，是通过存储来完成的，还是通过计算来完成的，这从功能上来说是无关紧要的。我们所关心的，只是叫“面积”的这个特征提供的行为(或状态)的类型，至于面积是作为一个槽口(slot)实现，还是由三角形的其他特征计算得到的结果，在最初的设计阶段不必考虑。

这种推迟作决策的能力，在 IDEF4 中由下述事实支持：(1)把特征范畴分为属性(返回值的)和例程(计算引入的)，(2)把属性范畴分为槽口(从存储中返回值的)和函数(通过计算返回值的)，(3)把例程范畴分为函数和过程(副作用的)。类特征继承网格用 IDEF4 继承图的类盒子句法示于图 4.3.5 中。

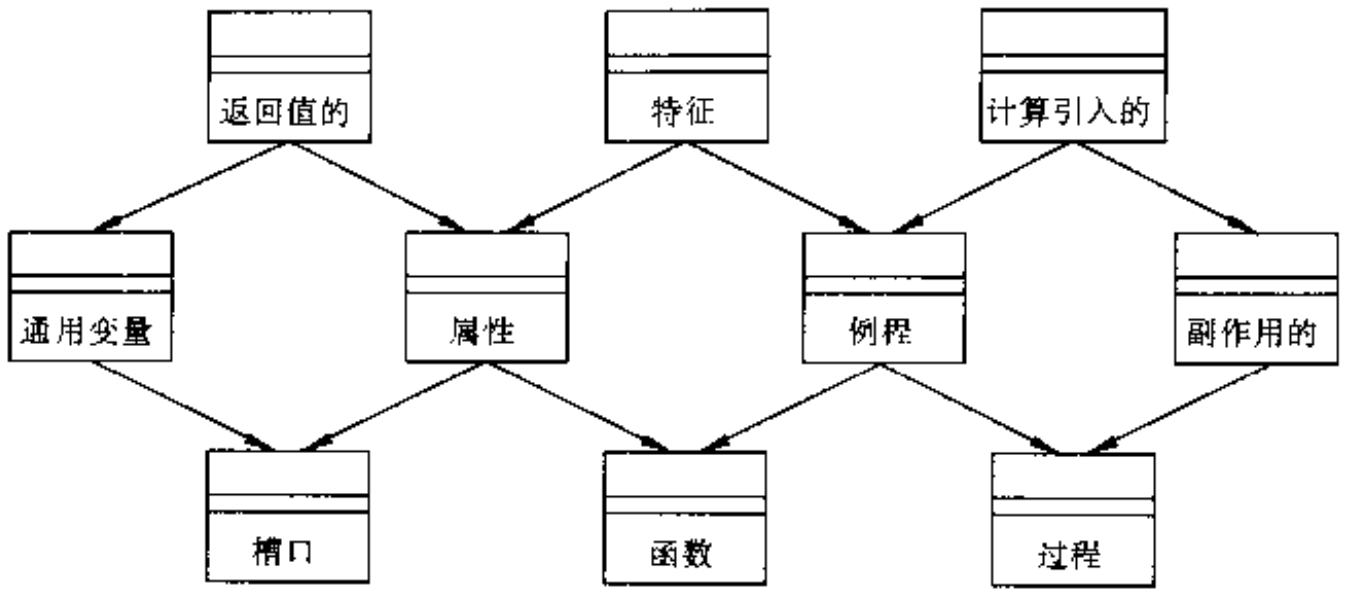


图 4.3.5　类特征继承网格

3.2.1　特征分类

特征分类使特征在一开始可以用一般的术语来刻画，然后，随着设计的进展，逐步被定义得更特殊。比如说，设计者可能在一开始，定义了类的一个特性为特征，然后，随着设计的进展，设计者可以将那个特征的定义特殊化为属性、例程、槽口、函数或过程。

图 4.3.5 显示了被继承的类特征的主要类型和它们的特性的继承网络。箭头代表特殊化关系，从最不特殊的类指向最特殊的类。在最不特殊层(顶层)，所有类特征归并在一起，简单地称为特征。属性的主要定义的特性，是在查询时返回一个值，而例程则是在正确触发时，将启动计算活动的东西。这两者并不是互相排斥的范畴。三个最特殊的特征集(即槽口、函数和过程)是互相排斥的。函数这一特征，同时具有属性和例程的特性，当查询时它通过计算返回一个值。过程这一特征，具有例程的特性，但它不返回值，它是仅有副作用的计算活动。但是严格说来，一个空操作(不操作)过程，在某些场合可能会有用的。槽口是属性而不是例程。它和 Common LISP (list processing)中的通用变量相似。也就是说，当查询时能返回值。因为它们可以访问某种存储工具，最终是为重新获得存放值的计算机内存。对于一个通用变量来说，分配操作或改变为重新获得而存放的值是有意义的。这对 IDEF4 特征的槽口类型也成立。三种最特殊的特征类型范畴(槽口、函数和过程)是互相排斥的，并形成了 OOD 中所有类特征集的划分。

3.2.2　特征继承

通过类继承机制，所有和超类相联系的特征都自动地和子类相联系。子类可以看到并使用它的超类的任何特征，但是反过来不行。比如说，图 4.3.6 显示了类“多边形”具有一个被称为“面积”的特征。“多边形”的子类，像“正方形”和“三角形”就从“多边形”继承了“面积”特征。

如果“面积”特征是通过计算实现的，那么用多边形的一般面积计算方法计算正方形面积将是低效的，因此，需要为“正方形”定义一个“面积”特征，对正方形的实例采用更高效的、更特殊的计算方法。这个特殊化的计算方法，就会被用来代替较一般的方法。这样，更特殊化的方法“遮蔽”了或重新定义了较一般的方法。图 4.3.7 就显示了这样一个情况，

类“正方形”的“面积”特征，遮蔽了或重新定义了类“多边形”的这个特征。“三角形”类仍从“多边形”继承了“面积”。

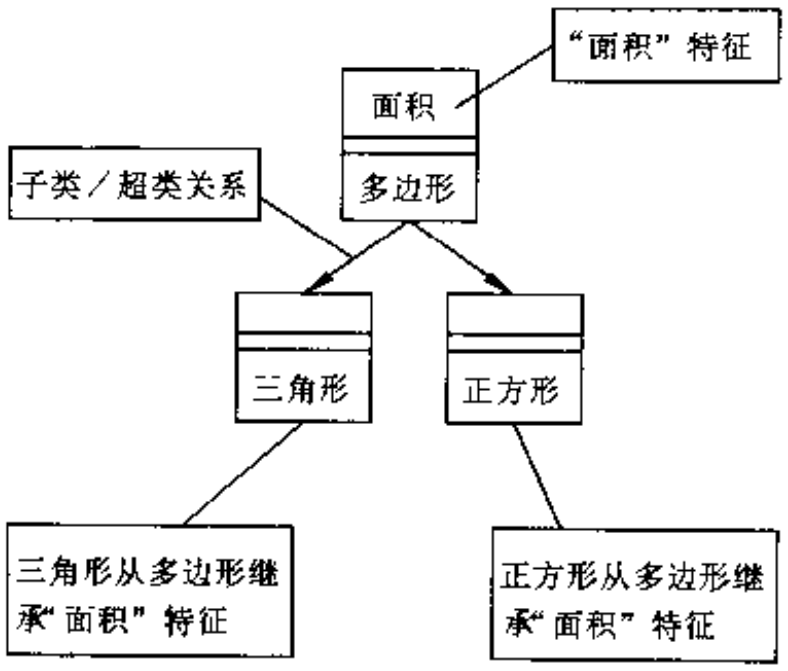


图 4.3.6　三角形和正方形从多边形对面积的继承

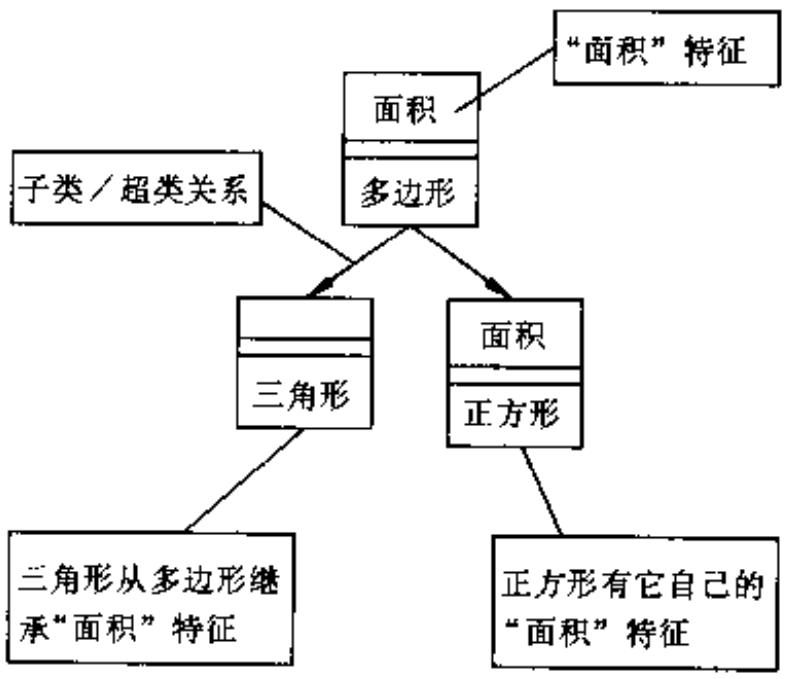


图 4.3.7　正方形的“面积”特征遮蔽了多边形的“面积”特征

3.2.3　特征/类分类法

考虑出现在每个对象类中的特征，来对特征进行分类是很重要的。这第二种分类方

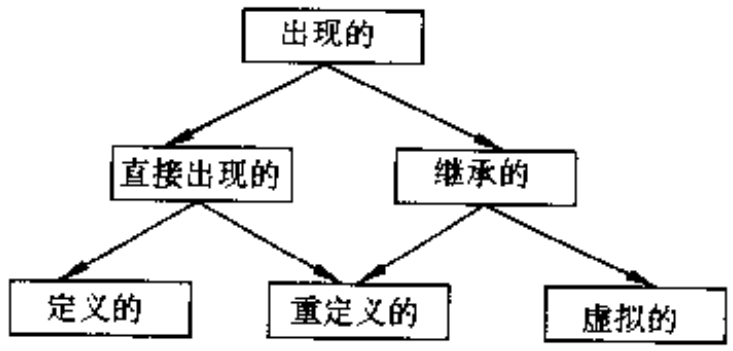


图 4.3.8　特征/类分类法

案，使设计者可以指出他打算以何种方式将特征和类相联系（即在类中定义、在类中重定义、或是类中的继承特征）。这二次分类方案的分类法示于图 4.3.8 中。

采用这种分类方案，以任何方式与类相联系的特征，都在类中出现。那些名字显示在类 A 的类盒子中的特征，被称为在 A 中直接出现（与图有关）。那些在 A 的超类 B 中出现的特征，被认为也在 A 中出现，并且是 A 的继承特征。A 中既直接出现又是继承的 A 的特征，是在 A 中被重新定义的。它们直接出现，是因为类盒子给出了与它们有关，而没有出现在 A 的超类中的，附加的或修正的信息。A 中直接出现，但不是继承的特征，被称为在 A 中定义的，而那些继承的而没有直接出现的特征，被称为在 A 中虚拟的。

图 4.3.9 中的矩阵，对图 4.3.7 描述的类层次的“面积”特征进行了分类。图 4.3.9 显示了“面积”特征的二次分类，考虑层次中的每个类，(1)在类“多边形”中，该特征是出现的、直接出现的和定义的，(2)在类“三角形”中，该特征是出现的、继承的和虚拟的，(3)在

类“正方形”中，该特征是出现的、直接出现的、继承的和重新定义的。

类	出现的	直接出现的	继承的	定义的	重定义的	虚拟的
多边形	X	X		X		
三角形	X		X			X
正方形	X	X	X		X	

图 4.3.9　多边形类层次中的“面积”特征

3.2.4　特征类型

返回值的特征，有一个与之相联系的说明返回值类型的类型。这个类型可以是典型的类型，像整型或实型，也可以是类或结构化的类的集合。

在 IDEF4 的规则中，所有属性都有返回类型，定义为它们返回值的类型。从设计管理的角度看，特征类型提供了类间关系的一个暗示。这个信息在继承网格上是不可见的。但是，经验表明，这些关联的管理，对开发大型的面向对象的系统来说是关键性的。通过特征类型的研究，编程小组可以交流想要的关系。

图 4.3.10 显示了特征-类型关系。图中有四个类：“可移动的”、“2 轮的”、“实型”和“轮”。类“可移动的”定义了两个特征(X 和 Y)，它们返回实型数，指出了可移动对象实例的位置。类“2 轮的”通过子类/超类关系继承了类“可移动的”的特征。类“2 轮的”定义了特征(“轮-1”和“轮-2”)，它们返回类“轮”的实例。类“轮”有特征“直径”，它返回类型“实型”的值。

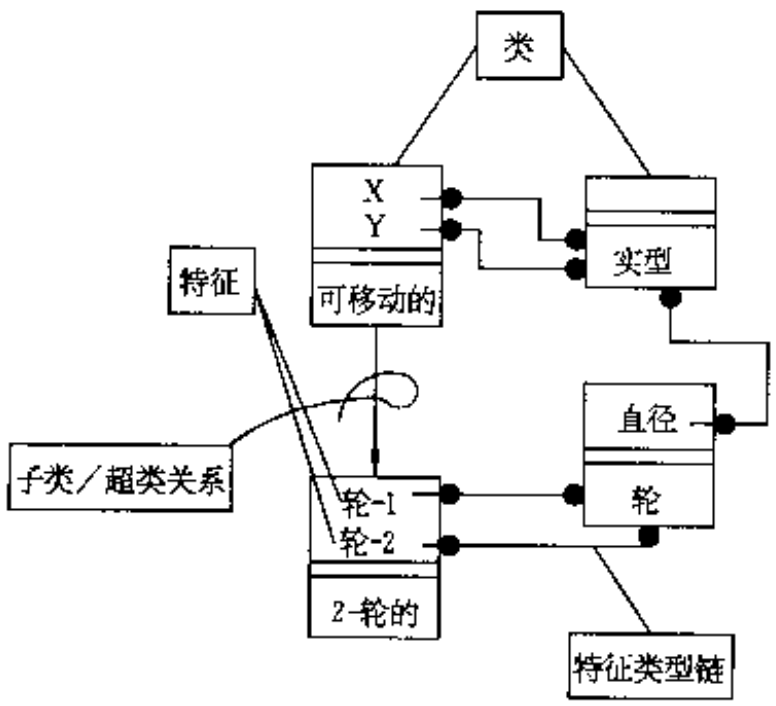


图 4.3.10　特征类型

3.2.5　类特征

图 4.3.11 显示了“形状”类-继承网格的类型联系和继承联系。这个继承网格的基础类是类“多边形”，它被类“凸多边形”所继承，而类“凸多边形”又被类“正方形”和“三角形”

所继承。在“多边形”和“凸多边形”的实例中，特征“边数”返回整型值，而在“正方形”和“三角形”中，特征“边数”受到限制，对该类所有实例只能返回一个固定值，有这种限制的特征被称为类特征。这种限制可以作为类-定常约束(即类中所有实例都遵守的约束)，或作为特征满足的合同的约束表示出来。

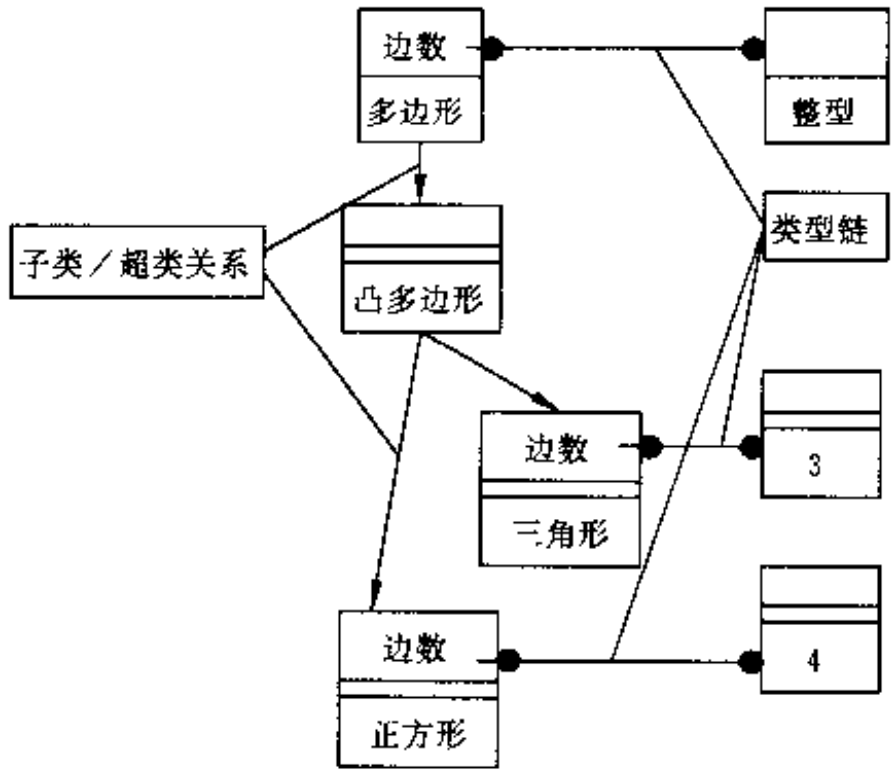


图 4.3.11　类特征

满足这种限制的类-定常约束的一个例子是：

对所有 (x)(三角形 x)→(边数 x) = 3

对所有 (x)(正方形 x)→ (边数 x) = 4

某些面向对象的语言像 CLOS 可以直接把类特征表示为类属性。

3.3　方　　法

正像在对类特征的讨论中所指出的，类可以具有定义它的实例的行为的特征。对象的行为，就是它怎样对消息或类属函数(generic function)做出响应：改变它的状态、改变它的环境、对其他对象起作用和返回值。定义行为的特征是计算引入的，因此，根据定义，它们都是例程。函数和过程特征类，是例程特征类的子类。更具体地说，它们从例程特征类继承了计算引入的特性。将行为定义的特征说成例程是很方便的。每一个例程，都必须在一个类中定义。这样，例程可以用“类属例程-类对”(generic routine－class pair)唯一的标识。在本文中，当显然是指定义在一个特定类中的一个具体的行为时，术语“例程-类”对将被用来代替“类属例程-类”对，而当显然是指行为的类时，术语“例程”将被用来代替“类属例程”(generic routine)。在有可能发生混淆的情况下，我们将使用不省略的形式。

3.3.1　设计中的方法与编程中的方法

作为特征，例程可以和多于一个的类相联系，并且可以根据在其上执行计算活动的对象类的不同，而执行得有所不同。在 OOPL 中，这些例程-类对由提供所需行为的一个方

法来实现。IDEF4 设计中，方法的概念并不和从面向对象语言角度看的方法的通常的概念完全一样。在面向对象编程时，方法是一段可执行的代码，它从算法上详细说明了，可用一系列指令来完成的计算活动，而在 IDEF4 中，方法由它必须满足的合同来定义。实际上，在 IDEF4 中，我们不说单个的方法，而说方法集，其中任何一个都能满足特定的合同。换句话说，我们指的是编程语言方法满足的合同，而不是方法实现的代码。

通常来说，有不止一个方法可以和一个例程相联系，甚至在编程语言里也是这样的。同一个例程，在不同的类中可以有不同的方法，但是在 OOPL 中，只有一个方法将和每一

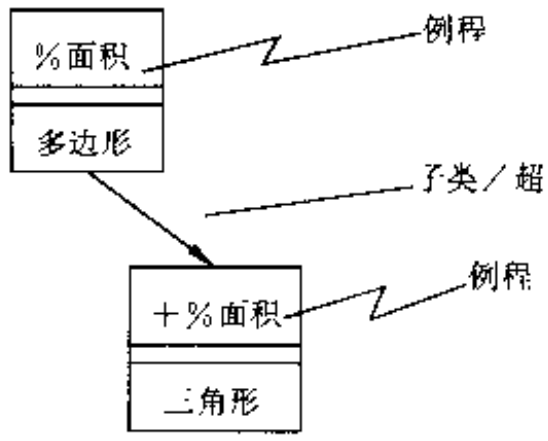


图 1.3.12　不同类中的同样的例程

对例程和类相联系。在图 4.3.12 中，例程“面积”是两个例程-类对：“面积：多边形”和“面积：三角形”的成员。在编程语言中，将为它们中的每一个定义不同的方法，像“得到多边形的面积”和“得到三角形的面积”。而在 IDEF4 中，不主张单个的方法。这样，在 IDEF4 中，“得到多边形的面积”和“得到三角形的面积”将参照方法集和它们在“而积”方法分类中阐述的有关合同。例程-类对详细说明了用与之联系的合同定义的方法集。这个集合中的任何成员都满足，并满意地执行该类的例程。IDEF4 的思想是描述或设计行为，而不是给行为编程。

3.3.2　方法集

方法集可以被认为是由约束集定义的特征函数，它挑选出一系列可能正确的实施。约束集——为方法集中定义的特性，被称为方法集合同。IDEF4 把方法集放在概念层，因此，在设计中，更关心的是合同的定义，而不是集合中单个方法的定义。“方法集”常被用来看作表示一系列方法特征的约束。这些特征化的约束，是任何实现都需要的。在 IDEF4 中，实现例程-类对行为的任何方法，都应该满足这些约束。这些被定义的约束集，就是 4.2.1节中将讨论的方法集合同。

举例来说，在类“士兵”中，类型为整型的特征“身份号”的类-定常约束，可以表示为：

“类‘士兵’中的特征‘身份号’，在类‘士兵’的所有实例中，必须是一个唯一的整数。”

更正式地，可以写成下面的约束，说明没有两个士兵会有相同的身份号：

对所有（x y）（士兵 x）^（士兵 y）^（不等的 x y）^（不等的（身份号 x）（身份号 y））

类-定常约束的另一个例子用于双链表。

用于类“表”中的方法“弹出”(pop)的合同的约束，可以定义为：

“当正在用的表是空表时，‘弹出’操作不适用。当这个操作试图在空表上进行时，操作应返回‘空’，指出异常情况。”

更正式地，如果“弹出”操作被用于表上，并且表是空的，则结果是“空”：

对所有 (x y)(引用 弹出 x) ^ (表 y) ^ (空 y)→(返回 x y 空)

图 4.3.13 中的方法分类图，例示了某系统中“打印”例程的方法集的设计。图中的每个节点代表一个方法集，该方法集需要一个约束该集合中方法的实现的合同。和节点相联系的方法的集合，可能由一个或多个方法实现所组成，但是，该集合中所有的方法，都将遵守相同的合同。

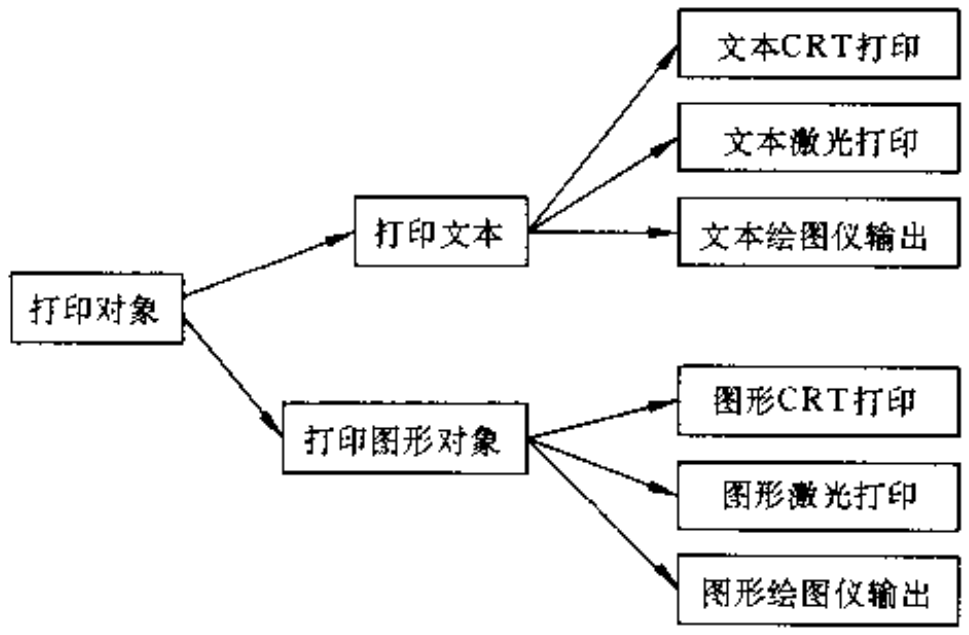


图 4.3.13　方法集关系

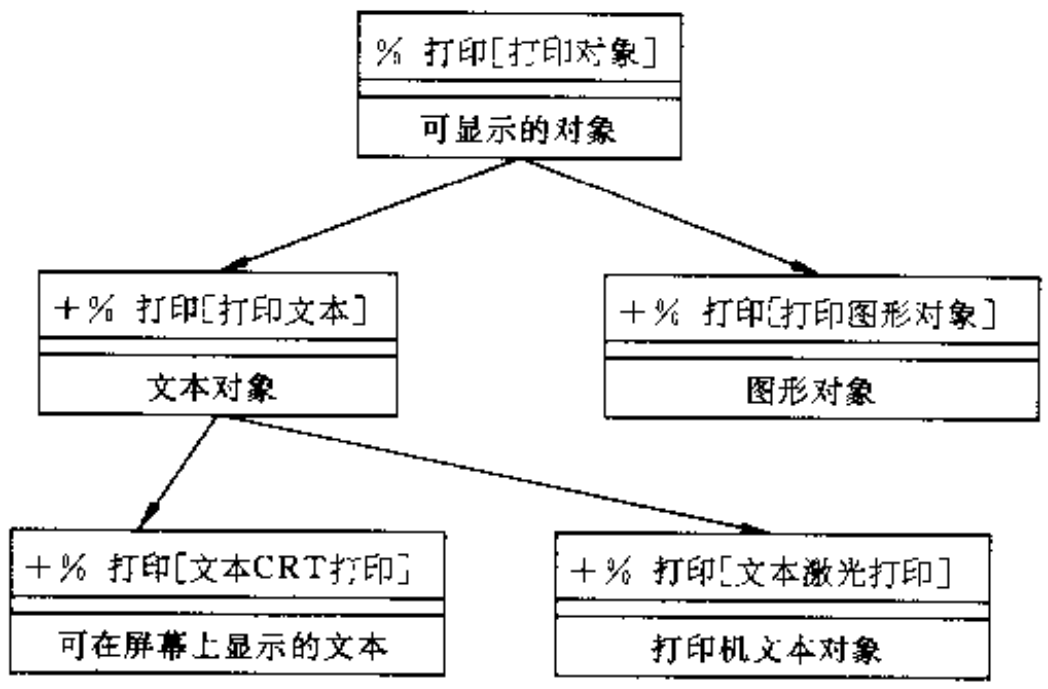


图 4.3.14　具有方法重定义的类结构

正如前文所指出的，在 IDEF4 中，一个例程-类对，仅决定一个可以为那个类实现例程的方法的集合。这是非常合理的，因为 IDEF4 方法产生的是一个系统设计，而不是一个实现。举例来说，在图 4.3.13 中显示了“打印”方法集。从左到右，图中的每个盒子代表一个“打印”的更具体的方法集(即，由方法集中的方法履行的附加的合同)。

图 4.3.13 中的每个节点，代表置于实现“打印”例程的方法之上的重定义或附加约

束。在方法集中，最不具体的方法是“打印对象”，但是，对某些对象类，像“可屏幕显示的文本”来说，需要附加的约束，并且要将例程重定义为图 4.3.14 中所示的“CRT 文本打印”方法集。图 4.3.14 中“打印”例程左边的“＋”指出例程整个都被重定义了。“打印”方法分类图中那个特征-类对的合同，将不得不整个说明，因为它可能覆盖前面的“打印”方法集合同，或和前面的“打印”方法集合同冲突。

3.4 约　　束

在 IDEF4 中，约束被用来说明方法上的合同和类-定常结构。约束通常在设计的开始阶段用自然语言说明。随着设计的进展，它们将被细化并且被更正式地说明(即用形式语言，像一阶谓词逻辑)。

第4章　图

有五种图和两种数据表格被用来建立一个 IDEF4 设计模型。五种图分组为下述两个子模型：

(1) 类子模型：

- 类型图
- 协议图
- 继承图

(2) 方法子模型：

- 方法分类图
- 客户图

这些子模型通过可以在继承图和方法分类图中表示的分配映射联系起来。对每个图类型的讨论，将包括所有会在图中用到的符号的简要描述。

4.1　类 - 继 承 图

类-继承图是 IDEF4 类子模型的一部分。它们被用来图形化地显示系统的类层次。

4.1.1　类-继承图的符号集

本节我们讨论用来建立类-继承图的符号，包括类盒子，描述类特征用的符号，和显示超类到子类关系的箭头。

如图 4.4.1 所示，在类盒子中，类名出现在盒子底部两条水平线下面。那些直接出现的特征名字，出现在两条水平线上面。在两条水平线上面，特征又可以被一条水平线(输出线)分成两组。输出线把类的直接出现的公共特征和直接出现的私人特征区分开来。公共特征是那些出现在线上面并被说成是从类输出的，而私人特征是不输出的，也就是说，它们只对该类和它的子类是可见的。虚拟特征是公共的还是私人的，要根据它们在其源类中的状态是公共的还是私人的面定。事实上，虚拟特征有着和它们在源类中所具有的完全相同的特性。如果有多于一个的源类，这些源类必须就虚拟特征的特性达成一致。

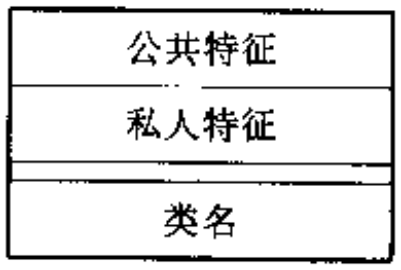


图 4.4.1　类盒子

如果某一类中的一个特征是公共的，那么它在该类中的存在，对 IDEF4 模型中所有的类来说都是可见的和可用的。如果类 A 的一个特征是私人的，那么它在 A 中的存在，只对 A 的所有子类来说是可见的，面对其他任何类来说都是不可见的(除非用其他方法使

它对它们可见)。如果不出现输出线,则类的所有直接出现的特征都是公共的。如果出现输出线,但没有特征在它的上面,则类的所有直接出现的特征都是私人的(如果一个类打算单独作为"图形",通过多重继承和其他类结合,这种情况就会发生)。也允许一个类没有任何直接出现的特征,在这种情况下,输出线不能出现。在图 4.4.2 关于"三角形"的 IDEF4 类盒子中,"名字"、"顶点 1"、"顶点 2"、"顶点 3"和"面积"是公共特征,"面积内含内容"和"面积标志"是私人特征。

IDEF4 的特征可以是例程、属性、函数、过程或槽口。一开始,设计者不能决定如何实现特征,只能决定哪些特定的特性应存在(见图 4.4.2)。这种设计实践是完全可以接受的,并且很可能这是类特征早期定义时的情况。随着设计开发的继续,设计者可以选择把特征分类具体化(设计也可能以这样一种方式进展,就是使某些特征从较特殊的类变成较一般的类)。当设计者希望确定某个特征的类型时,可以把类型符号加在特征名的左边(见图 4.4.4)。特征说明符号显示在图 4.4.3 中的特征类-继承图中。

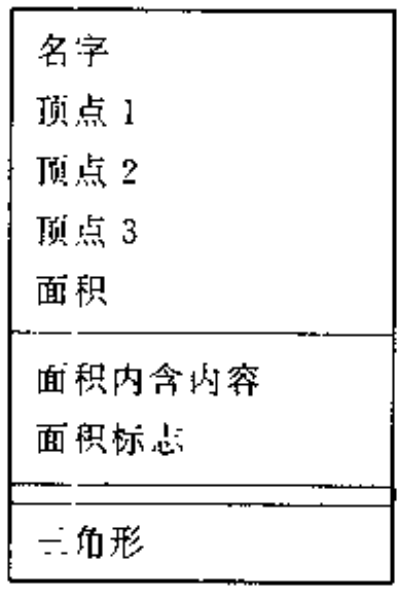


图 4.1.2 具有特征的三角形类盒子

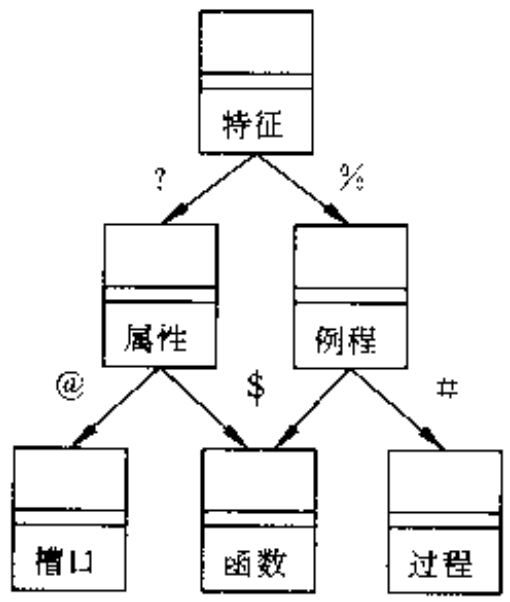


图 4.4.3 显示特殊化符号的特征分类

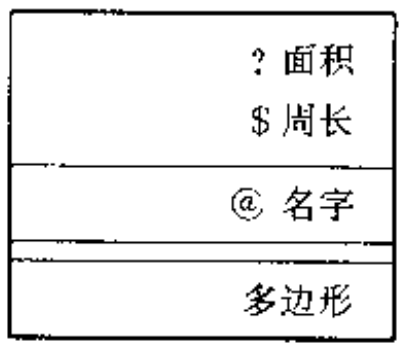


图 4.4.4 具有特征符号的多边形类

特征符号是:

特征:特征名的左边没有符号,表示特征是出现的,但是还没有决定如何实现它。

%例程:符号%表示在正确触发时,特征将启动某些计算活动,不知道或者没有说明是否将返回值。这些细节可以留给设计的实现者。

?属性:符号?表示该特征在查询时将返回值,这个值可以是存储的值,也可以是某种计算的返回值。

$函数:符号$表示该函数在查询时将通过计算返回值。作为函数实现的特征,同时具有属性和例程的特性。

#过程:符号#表示该特征是不返回值的例程,它具有副作用。

@ 槽口:符号@表示该特征是通用变量。槽口在查询时,能返回值,因为它对存有槽口值的存储单元具有读写权。用来检索值的操作,和用于检索的值的分配操作,将和这种类型的特征相联系。

特征符号指出设计者希望某个特定的特征如何被实现。比如说，图 4.4.4 中，“多边形”类有三个特征：“名字”、“面积”和“周长”。“名字”特征将作为槽口来实现，“面积”特征将作为属性来实现，而“周长”特征将作为函数来实现。

在图 4.4.5 中，阐述了“多边形”和“三角形”类之间的超类/子类关系。这个关系用一个从超类“多边形”指向子类“三角形”的箭头来定义。“多边形”的子类将继承“多边形”的所有特征和实现。这个继承特性并不总是设计者所希望的。举例来说，“三角形”类(图 4.4.5)是“多边形”类的子类，但是计算一般多边形面积的算法，对三角形来说就太通用了。由于这个原因，设计者必须指出，将“面积”特征的重定义用于“三角形”类的对象。

如果继承的特性是重定义的，IDEF4 提供了用在子类中的符号。下述辅助特征符号中的一个或多个将出现在重定义的特征名(和特征符号，如果有的话)的左边来表示这个特征是如何被重定义的。

& 附加合同——这个符号表示在与该特征相联系的方法合同上，加一些附加的约束(比如说，前提条件或后续条件)。

＋ 新方法——这个符号表示该特征代表另一个例程，它的名字和从超类中继承来的相同(如图 4.4.5 中的“三角形”类)。加在例程上的附加约束要求完整地说明一个新的合同。

！新的分类说明——这个符号表示一个特征在子类中被更具体地重新定义为函数或槽口，而且可能在不同的子类中被定义得不同。

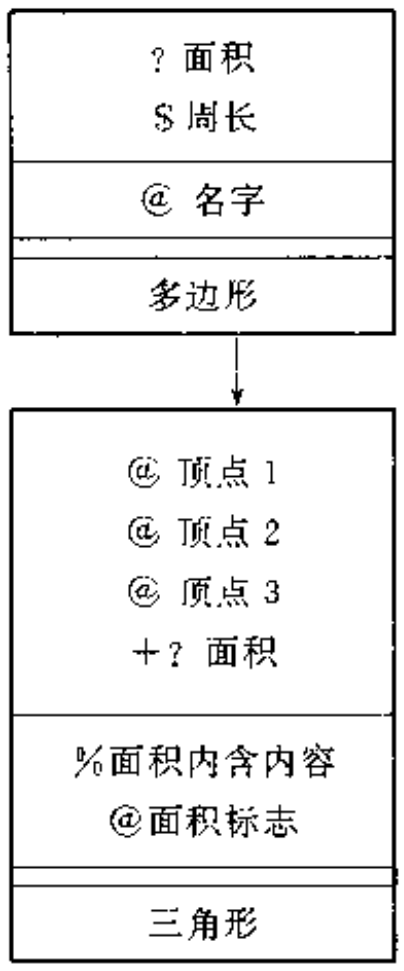


图 4.4.5　三角形类——多边形的子类

^ 现在公共的——这个符号表示原来的私人特征变成公共的，它在该类中(因此在所有后续的子类中)的存在，对 IDEF4 模型中的所有类来说，都是可见的和可用的。被重新定义为“公共的”的特征名字应放在输出线上面。

* 现在私人的——这个符号表示原来的公共特征变成私人的，它在类中的存在，对该类的所有子类来说是可见的，但对 IDEF4 模型中的任何其他类来说，就不是可见的了。重新定义为“私人的”的特征名字应放在输出线下面。

这些表示类盒子、特征识别/重定义的符号和箭头结合起来，形成了类-继承图。

4.1.2 理解类-继承图

既然我们已经讨论了继承图的基本符号集，下面我们将讨论显示这些信息的图形。用于这个目的的基本的 IDEF4 语言成分是继承图。某个特定 IDEF4 模型的继承图形，仅仅是一张巨大的显示了所有的类和直接继承关系的继承图。这个尺寸的继承图如果真的画出来的话，几乎没有实用性，但是保持在头脑中，作为想象继承图全貌的一种方式还是非常有用的。实用的继承图仅由于省略而不同于完全的继承图。完全的继承图也可以存储在计算机内存中，以自动生成可显示的图。

在 IDEF4 的继承图中，类用类盒子的概念来显示，继承用从一个类盒子（超类）指向另一个类盒子（子类）的箭头来表示。在图 4.4.6 中，从“多边形”指向“三角形”的箭头表示“三角形”是“多边形”的子类。继承是可传递的，如果“填满的三角形”是“三角形”的子类，而“三角形”是“多边形”的子类，则“填满的三角形”是“多边形”的子类。对一给定的 IDEF4 图来说，直接用一个从类盒子“多边形”指向类盒子“三角形”的箭头表示的继承，称为直接继承（之所以说“对一给定的 IDEF4 图来说”，是因为并非所有的 IDEF4 图都必须显示两个类间的所有连接关系）。“多边形”被称为“三角形”的直接超类，而“三角形”被称为“多边形”的直接子类。

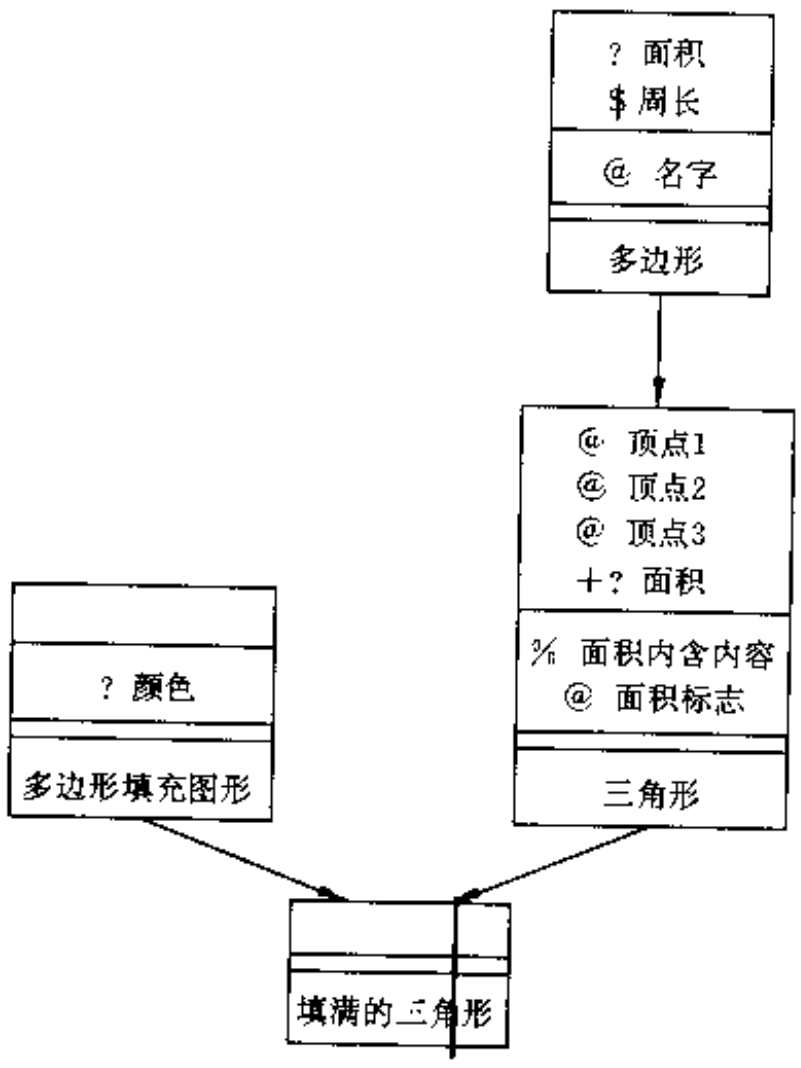


图 4.4.6 多边形类的部分继承图

如果继承不是用一个箭头来表示的，而是只能利用传递关系从箭头链的存在中推理得到——像图 4.4.7 中“标准对象”和“简单的锁”之间——则使用这些术语：间接继承、间接超类和间接子类。

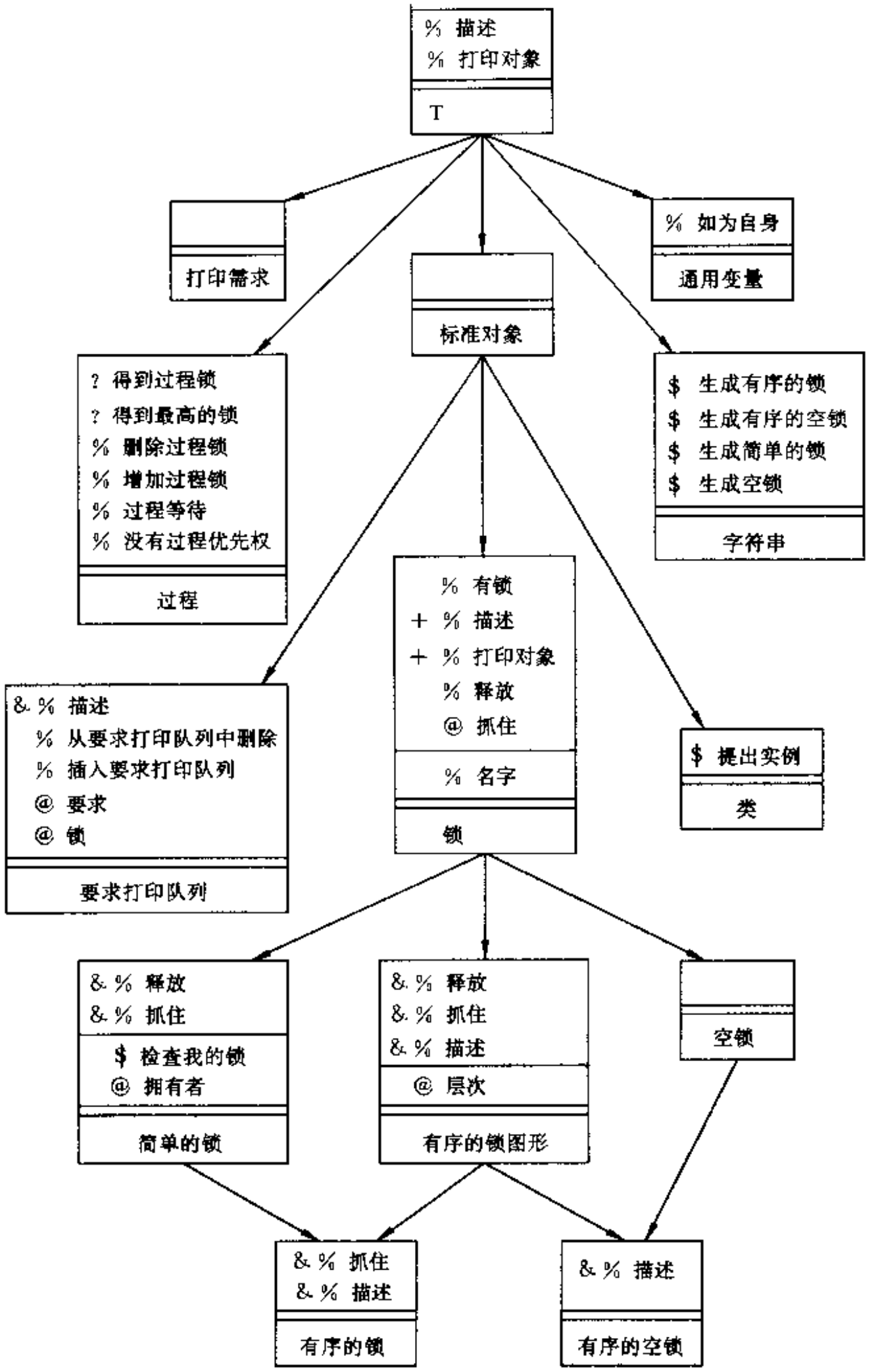


图 4.4.7　继承图——网络管理程序系统

设计这个用于描述类层次结构的图形化的方法是为了在最少量的空间中显示最大量的关键信息。类-继承图确定了显示在盒子里的类和子类的特征，还揭示了有关特征实现的细节和它们在系统中的可见性。在图 4.4.7 中，较具体的类“标准对象”只有从类 T 继承的特征。因为这些继承的特征没有被重新定义，所以它们不出现在“标准对象”的类盒子中。“锁”是“标准对象”的直接子类，是 T 的间接子类，也继承了特征“描述”和“打印对象”。在“锁”中，“描述”和“打印对象”前的加号(+)，表示这些特征用遮蔽超类定义中某些约束的约束重新定义过了。

如果想表示，重新定义是把前提条件或后续条件附加到超类例程的约束集中去，可以用符号 &。直接出现的特征“有锁”、“释放”和“抓住”，在类“锁”中都被定义为例程，并且是公共的。直接出现的特征“名字”前的符号@，定义该特征作为槽口实现，它出现在输出线下面，表示它是私人特征。

4.1.3 类-定常数据单

在 IDEF4 中，每个类都有相关的该类定义的详细说明。这个定义载入类-定常数据单(CIDS)，是因为它包括了(以说明语言的形式)设计者对有关该类单个实例必须具有的特性值和对象关系的打算。除了列出各种各样的 IDEF4 图形语言中显示的类的特征和关系外，CIDS 允许类定义的精确化。CIDS 可以被认为是附在继承图中的各个类盒子上的(见图 4.4.8)。CIDS 给出了有关类中对象的进一步的信息，也许以文本的形式涉及各种各样独特的特征。这些信息作为一个整体，构成了一个类-定常量，它必须对类中所有对象一直保持是正确的。“链表”的 CIDS 可能包括如下约束：由表的第一个元素和表的其余部分构造的表和原来的表是一样的。

相等（构造（第一个（x），其余的（x）），x）

类-定常约束可像特征一样被继承。结合的原理是逻辑与。类-定常量是出现在它自己的 CIDS 上的和出现在它的所有超类的 CIDS 上的信息的逻辑与(术语“类-定常”也可以不严格地用于表示一个 CIDS 的内容，甚至用于表示出现在这种单上的一个约束，像上面提到的“链表”的约束)。

CIDS 并不仅仅是用于列出类上的约束的工具，CIDS 的目的之一是为那些实现设计的人和那些维护安装好的系统的人提供文档。IDEF4 设计文档，以为系统定义的类为中心，因此，CIDS 包含数字标识符，以便引用该类出现在其中的继承图和类型图。提供类的直接出现的特征名，使得可以访问例程-类对，其他设计成分都是围绕例程-类对组织的。每个特征的定义和特征名包括在一起。虚拟特征通过该类的直接超类的列表访问。类的 CIDS 可能大于 8.5 英寸× 11 英寸的一页。因为它要包含那么多的信息，通过把页码包括进来，可以解决这个问题。甚至在类的描述已经完成时，也不要指望 CIDS 已经完备，CIDS 中的某些信息可能直到系统设计接近完成时才能得到。举例来说，并不是所有的继承图，也并不是所有的类型图(类出现在这些图中)，是在决定约束和特征时都可以知道的。因此，正像设计的其他成分随着设计过程演进一样，这些数据单也将不断演进。

类-定常数据单：

（类名）

类继承图：

（标识和标题）

类型图：

（标识和标题）

拥有者：

姓：　　名字：

批准日期：

描述和目的：

约束：

直接超类：　　直接子类：

特征(名字,类型,公共的/私人的,定义的/重定义的)

（类名——第1页）

图 4.4.8　类-定常数据单格式

图 4.4.9 和图 4.4.10 阐明了 CIDS 的应用。图 4.4.10 中的 CIDS 可以被认为是和出现在图 4.4.7 所示的类-继承图中的“锁”类(图4.4.9)相联系的。表顶部的信息是类名和其他有关的数据的簿记类型。用标识符和标题列出该类所在的继承图，这提供了和 CIDS 的连接，使得实施者可以得到在系统类层次中类的位置的可视化的表示。描述的成分，还包括由类描述的对象的类型标识和与它存在的理由有关的其他任何信息。

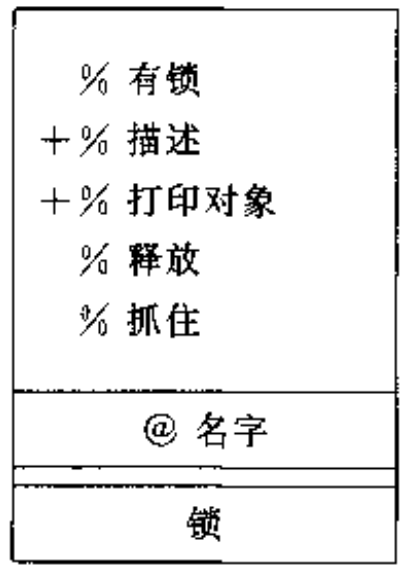


图 4.4.9　网络管理系统“锁”类

CIDS 最基本的成分是类实现的约束集。这些约束可以用普通的语言(见图 4.4.10)、一阶逻辑、或其他一些适于表达约束的语言来表述。在类层次中，每个类都继承了它的超类的类-定常约束。通过该类的直接超类表，实施者可以确定继承的约束位置。直接子类列表则提供了与那些将继承该类约束的类的联接。超类和子类列表提供的不仅是对约束的跟踪能力。譬如类“锁”被删除了或被修改了，这两张表可以为那些修改系统设计的人提供工具，快速跟踪系统中因这个改变

类-定常数据单： 锁

（类名）

类继承图： T1— -网络管理系统

（标识符和标题）

类型图： T1——网络管理锁

（标识符和标题）

拥有者： 张 三

姓： 名字：

批准日期： 3/15/91

描述和目的：

"锁"是基本的锁类型。它用作所有锁类型的超类。

约束：

一个锁只能被一个过程拥有。
每个共享的资源只有一个锁。
一个锁不能被它的拥有者外的任何过程释放。

直接超类：

标准对象

直接子类：

简单的锁
有序的锁图形
空锁

特征(名字,类型,公共的/私人的,定义的/重定义的)

有锁:(例程,公共的,定义的)这个例程是对上"锁"协议的句法扩展。

描述:(例程,公共的,重定义的)这个例程专说明锁的对象打印例程并覆盖系统定义的描述例程。

打印对象:(例程,公共的,重定义的)这个例程专说明锁的对象打印例程并覆盖系统定义的打印对象例程。

释放:(例程,公共的,定义的)被调用时,这个例程将释放要进行释放的过程所拥有的一个锁。

抓住:(例程,公共的,定义的)所有锁的类属"抓住"例程。

名字:(槽口,私人的,定义的)这个特征是一个锁的名字属性。

(锁——第1页)

图 4.4.10 "锁"的类-定常数据单

将受影响的类。

CIDS 的最后一部分是类的直接出现特征的列表。继承的特征可以通过查阅类“锁”的超类的 CIDS 来找到。对于“锁”的每个直接出现特征，CIDS 都将包含在特征继承图中可找到的名字，类型等；此外，特征实施要满足的合同的方法集，和特征的定义也包含在 CIDS 中。CIDS 中的特征定义，是设计中唯一提供特征的文本形式定义的地方。

4.2 方法分类图

本节我们将讨论方法分类图和与图中方法集相联系的 CDS。通过合同对方法进行分组，得到的方法集形成一种分类法（成组），该分类至少部分独立于例程通过类-继承图分组的方式。作为 IDEF4 模型的方法子模型的一部分，IDEF4 引入方法分类图来表示设计中各组相关的方法集合同。因为对方法集合同的理解是对方法分类图理解的关键，所以我们将首先介绍方法集合同。

4.2.1 合同数据单（CDS）

严格地说，OOPL 中的方法用其代码来描述，而在 IDEF4 中，方法是满足方法集中合同的任何实现。在 IDEF4 的术语中，方法集完全由它的合同决定，并且逻辑上等价的合同挑选出相同的方法集。虽然这样听起来似乎方法合同仅仅是一个规范，但实际上它远远不止这些。方法合同是定义一类例程或一个方法集的抽象的描述。它是方法集中方法所期望产生的效果的阐述。对于一个无副作用的函数来说，合同将说明函数参数表和相应的返回值之间的关系。对于一个过程或一个有副作用的函数来说，合同还将定义方法集在给定参数表和原先的状态时如何改变现实的整个状态。在某些情况下方法合同可能包含算法的限制，如“移动方法将先执行擦除方法然后执行画方法”。合同还可能给出关于方法集期望的性质的其他信息（比如说，它的时间复杂性）。另外，人们还可能希望找到在方法被引用前，应该发生的行为的说明，和在方法完成了它的任务后，应该发生哪些动作。

方法集合同为所有的实现方法（该方法集中的成员）提供必须满足的约束。正像类-定常量对类的所有成员都成立一样，合同对方法集来说也是一个常量（即，它对所有的实现（该集合中的成员）都成立）。举例来说，考虑图 4.4.11 所示的类层次，类“多边形”中的例程“面积”，说明了计算任意多边形面积的方法。但是，当多边形是三角形或矩形时，有更有效的计算面积的方法。类层次显示了在每一个这种子类中，“面积”特征被重新定义了。这些重定义的方法集合同，要说明设计者希望方法集满足的附加约束。在图 4.4.11 中，辅助符号“+”表示附加约束是这样的，要为“三角形”和“矩形”类的“面积”例程写一个全新的合同。每一次例程为一个类定义或重新定义时，设计者必须说明实现该例程的方法应满足的约束。这一系列约束形成方法集合同。

交流和合作不好是软件开发时生产率不高的根本原因。经常，软件演示未能按希望的那样工作的原因，是另一个编程员未经记录或解释就修改了它。与方法集相联系的 CDS（合同数据单），就是保证所有编程员产生的代码，都符合相同的预期系统行为的一种手段。

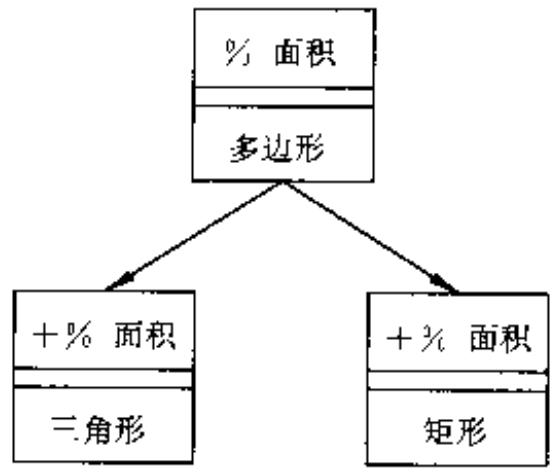


图 4.4.11　具有面积特征的部分继承图

在 IDEF4 中，如图 4.4.12 所示，合同被记录在和方法集相联系的 CDS 中。使用合同的目的之一，是便于大型软件项目中设计者和实现者之间的交流与合作。在最终的文档中，CDS 是按例程名后接类名，再按拼音字母顺序组织的。给定一张 CDS，通过使用例程名和(或)类名，可以很容易地找到所有其他的 IDEF4 图。例程名用于查找方法分类图，类名找例程在其中被定义的类的 CIDS(类-定常数据单)，而 CIDS 包含了所有该类出现在其中的类型图和继承图的列表。例程名和类名结合起来，可以找到用于该方法集的协议图和客户图。输入类、输出描述和过程信息，可以作为约束被包括进来。合同中的描述或定义成分，是方法合同的文本形式的描述，还可能包括方法集的理论基础。

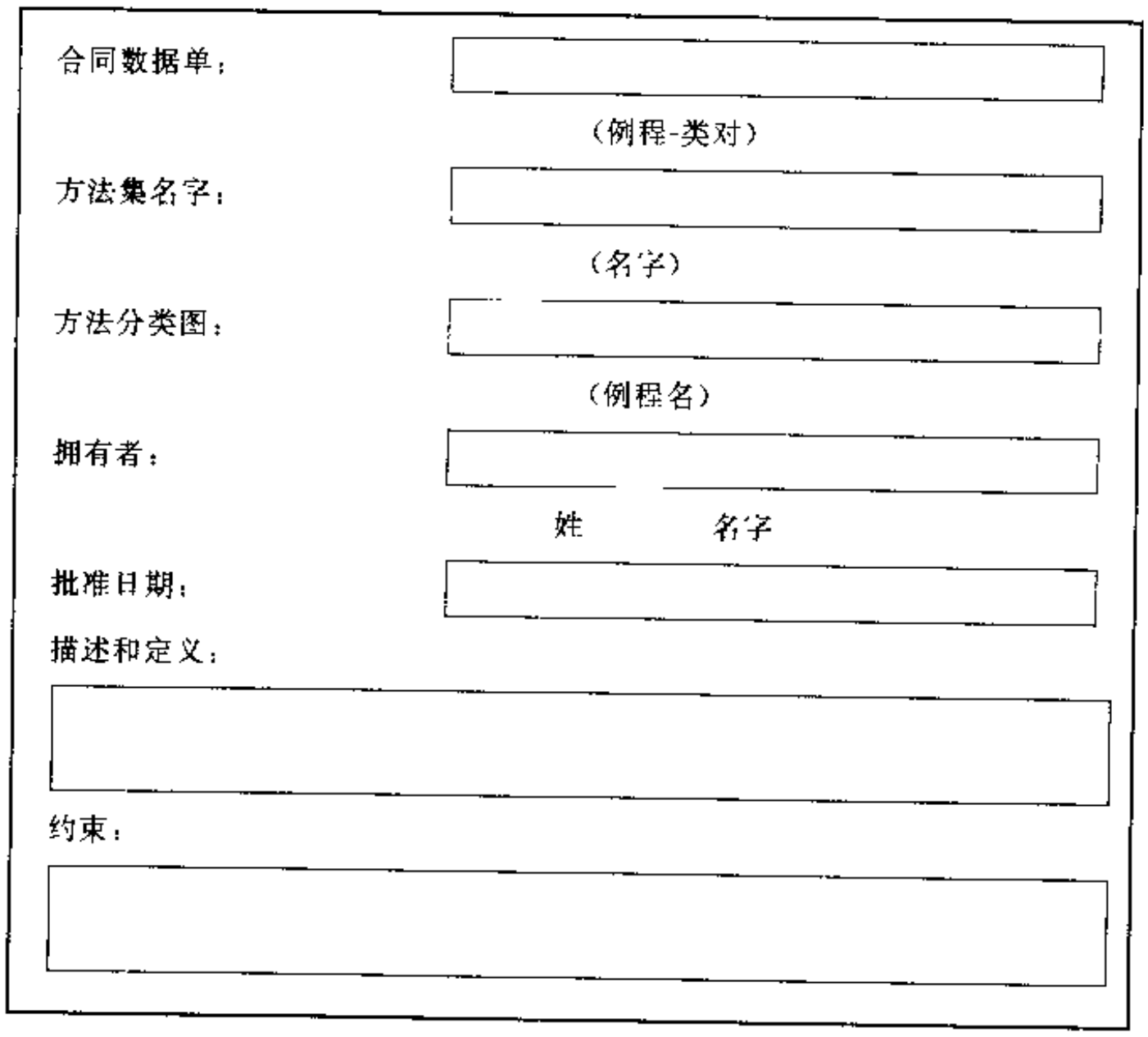


图 4.4.12　方法集合同

在设计中，方法集通过相关的合同组合起来，形成用于某个特定类型的系统行为或例程的方法分类。这些组代表了从最不具体的到最具体的用于某个特定类型的合同的框架。方法分类图，为展示设计中方法组如何互相关联提供了图形化的工具。

4.2.2 方法分类图符号集

方法分类图中用到的两个符号是盒子和箭头。盒子代表方法集，而箭头表示附加约束关系或重定义关系。图中的箭头从较不具体的方法集指向较具体的方法集。方法分类图在一页上可以从左向右或从上向下安排，表示从最一般的方法集到最特殊的方法集。

在图 4.4.13 中，左边的方法集代表满足较不具体的合同的方法的集合，这些用箭头表示的附加合同，和直接联系的较一般方法中的合同，可能冲突，也可能不冲突。这意味着在箭头末尾的方法集的合同，将有加在前面方法集上的所有约束，再加上一些附加约束，这些附加约束可能会以某种方式覆盖或遮蔽那些继承来的约束。不管在哪种情况下，附加合同使方法集更具体。合同关系的种类：重新定义或附加合同，它们将分别用辅助符号“+”或“&”，在类特征在其中定义的继承图中说明。如果合同被遮蔽或重新定义，它在类-继承图中的属性，将包括“+”这个前缀，并且如果无法规定较一般合同中哪些合同项将被遮蔽，方法集合同必须整个重新说明（即，增添到较一般的方法集的约束中这个概念不适用）。

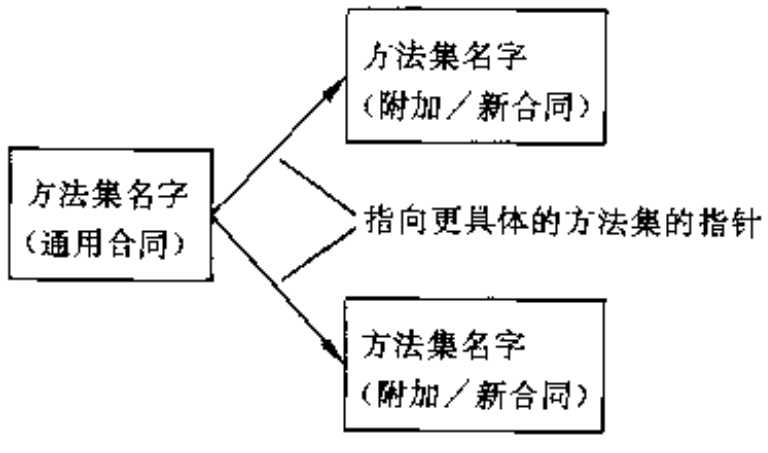


图 4.4.13 方法分类图符号

4.2.3 理解方法分类图

IDEF4 的方法分类图是为某类行为组织合同结构的方法。它为用图形说明给例程编写附加的或新的合同提供了手段。按照惯例，方法分类图的名字，就是要描述的例程。

图 4.4.14 给出了一个部分类层次。在这个层次中，“排序”例程在根类“序列”中定义，并被子类“阵列”和“表”所继承。在这些类的每一个类中，该例程有新的合同（比如说，附加的和（或）新的约束），这一点用图 4.4.15 中的“排序”方法分类图来展示。

注意，每一个方法分类图都和一个类属活动相联系，这是很重要的。在图 4.4.15 中，类属活动是“排序”。图中的每一个方法集都和排序操作或某些例程-类对的排序合同相联系。为了增加可读性，需要将方法集命名为一系列互相联系的事物的组合，包括（1）活动标志符，（2）集合中的方法要完成的操作、过程或函数，（3）类名。包括活动标志符的原因是使人们能够区分具有相同例程-类对的方法集。例如，设计者在类“阵列”中定义了多于一种

"排序"例程(如冒泡排序,插入排序和堆排序),并且都命名为"排序",就是这种情况。在这种情况下,排序方法集应被命名为"堆排序阵列"、"插入排序阵列"和"冒泡排序阵列"。类继承图中的特征定义就需要把包含方法集全名的内容明确地分配到它该要分配的地方。

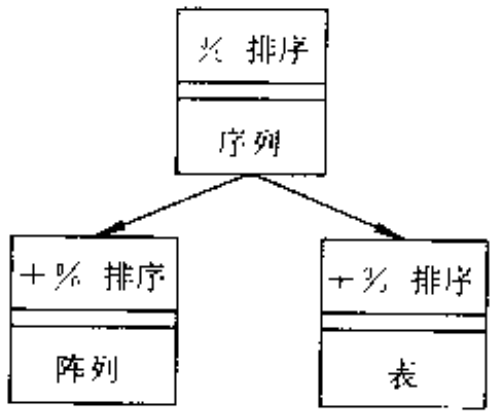


图 4.4.14 具有附加合同的类结构

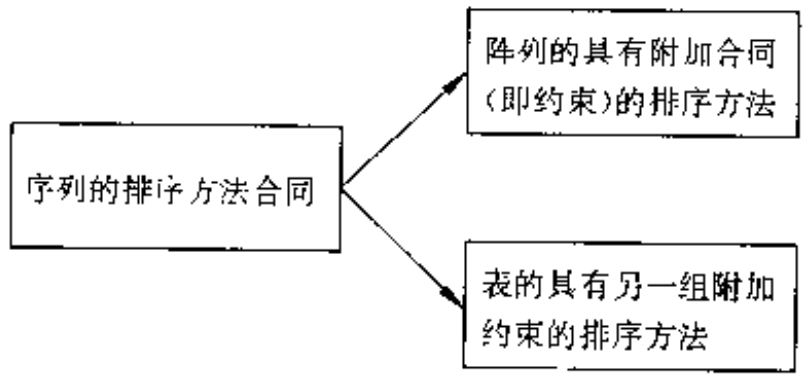


图 4.4.15 序列结构中排序方法集的安排

理解方法分类图的关键是理解与图中方法集相联系的合同。方法分类图中的盒子代表方法集,方法集通过从较通用的方法集指向较特殊的方法集的箭头相连。这样,我们就可以说方法分类图中的箭头表示附加合同。图中表示的附加合同可能提供冲突或不冲突的约束(见图 4.4.16)。

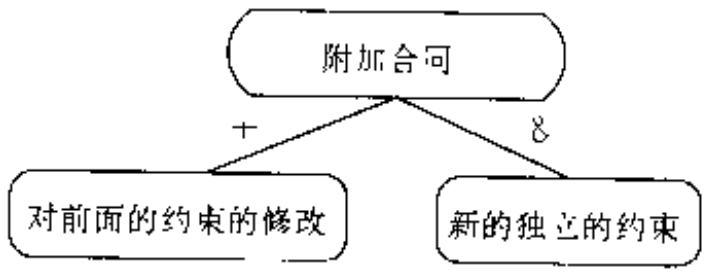


图 4.4.16 附加合同

冲突的约束就是那些以某种方式详细说明或重新定义了前面方法集中合同的约束。在 CLOS 程序中,这类似于新的遮蔽了任何较不具体的方法的基本方法定义。不冲突的约束仅仅表示约束的增加,这些约束不遮蔽,或不受任何所谓继承约束的影响。不冲突的约束提供了"纯粹"意义下"附加合同"的概念,并且是一系列希望用于基本方法集中方法的前提或后续条件。前提或后续条件的概念,意味着新方法集中的方法将执行基本的方法,并且或者在前,或者在后执行一些附加的方法。在 CLOS 模型中,这种行为通过用合适的方法结合策略,结合前、后、周围和原始方法来完成。而在 C++中,人们可以明确地调用前置和后置方法来完成这些操作,或把前置、后置方法的能力加到 C++中去。这两项技术在 C++参考书中都有记载。

当设计或阅读方法分类图时,值得注意的重要的一点是,首先要识别用来命名的图的行为类。这将给读者提供一个总体印象,理解要说明的是什么和要实现的是什么。但是,要完全理解图中方法集间的关系,还需要研究每个 CDS。正是由于这个原因,如果打算遮蔽在较通用方法集中的一些或全部约束,则 CDS 必须包含新方法集约束的全部而完整的列表。更进一步,CDS 应说明是该合同取代以前的所有合同,还是加入前提和(或)后续(不冲突)条件。图的布局本身,仅提供一个从较不具体的方法集到较具体的方法集这一顺序的总体概念,而 CDS 提供了方法集间实际上的关系。

图 4.4.17 例示了具有附加冲突约束的“面积”计算的方法分类图。在这张图中，加在方法集“计算多边形面积”上的约束，指出了在这个方法集中的方法，可以计算任何归类为“多边形”对象的面积。但是，图中另外两个方法集，表示加在第一个方法集上的约束的专门化。“计算三角形的面积”，仅计算是三角形的多边形的面积，另一个方法集，代表那些仅计算具有平行边的图形的面积的方法。在这两种情况下，新的或附加的约束，取代或专门化了方法集“计算多边形面积”的合同，要求更严格的或更专门类型的行为。

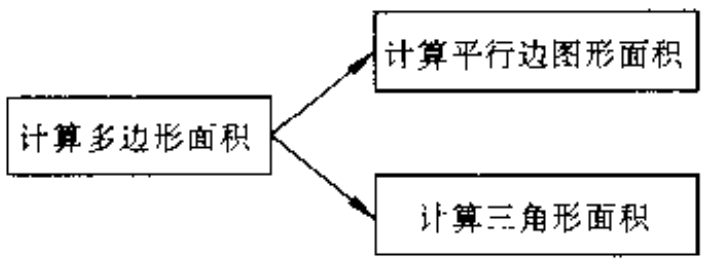


图 4.4.17 “面积”方法分类图

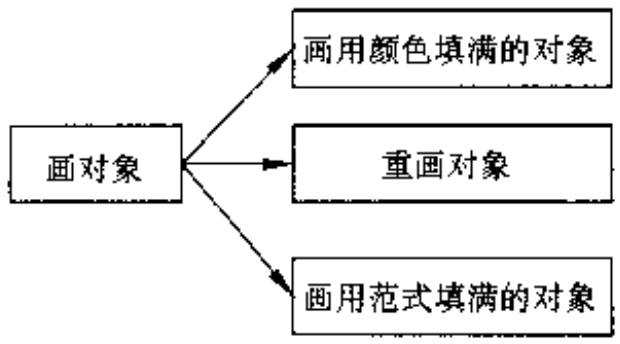


图 4.4.18 纯粹的附加合同

图 4.4.18 例示了“画”方法分类图，对于图中箭头代表附加合同这个概念，这张图提供了更深的理解。不查阅 CDS 本身，不能确定图意，下面的内容可以在与该方法集相联系的 CDS 中找到：

“画对象”是画一个对象的基本方法。

“画用颜色填满的对象”是“画对象”合同上的后续条件，“在画完一个对象后给它染色”。

“重画对象”具有约束(前提条件)，“在画新对象前，旧的对象必须被擦除”。

“画用范式填满的对象”具有约束(后续条件)，“在画完对象后用范式填充”。

CDS 揭示了最具体的方法集需要在执行它们之前或之后，执行较通用的方法集。

方法分类图有另一个有些脱离单个系统设计的用途，这个用途，就是作为一种对用于多种系统的共同操作的方法集进行分类和组织的手段。它可以提供以前编码的方法的目录。如果某个特定的合同被广泛地使用和研究(像排序)，那么相应的方法集和它的子集，就会形成相当复杂的分类(见图 4.4.19)。这样的分类可以作为设计者的资源，这种复杂程度的分类，也说明了方法分类对例程-类继承的相对独立性。

图 4.4.19 和 4.4.20 例示了如何使用明显的分配映射从设计的其他成分找到方法分类图。在例程-类对挑选出多于一个方法集时，分配映射必须明确地定义。

图 4.4.20 (a)例示了在类“表”中，例程“排序”被重新定义并将被分配给方法集“表归并排序”。术语“分配”指的是，指出哪一个方法集合同和例程-类对相联系的方法(见 4.6 节)。比较图 4.4.20 (b) 和图 4.4.19 可以知道，排序方法分类图不必按照和类-继承图对例程分组的层次一样的层次，来对方法集分组。和重新定义的排序例程一起，在“表”中定义了另一个例程“表插入排序”，这个例程被分配给“插入排序”。在图 4.4.20 (b) 中的方法集的顺序，和图 4.4.19 中的排序方法分类图中的不一样。“排序”和“表插入排序”都是合法例程，并且都被分配给在同一方法分类图中的方法集。

方法分类图，也是帮助设计者找出设计中函数有可重复使用机会的重要工具。函数重

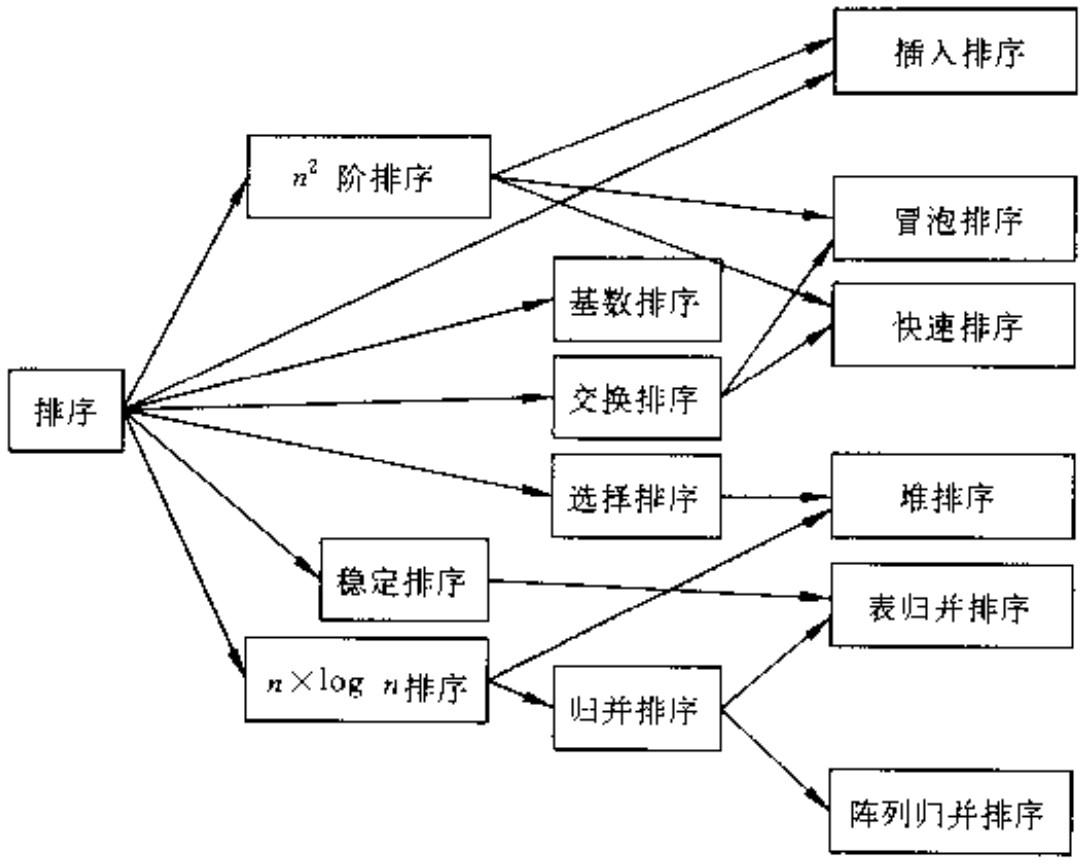


图 4.4.19 "排序"方法分类图

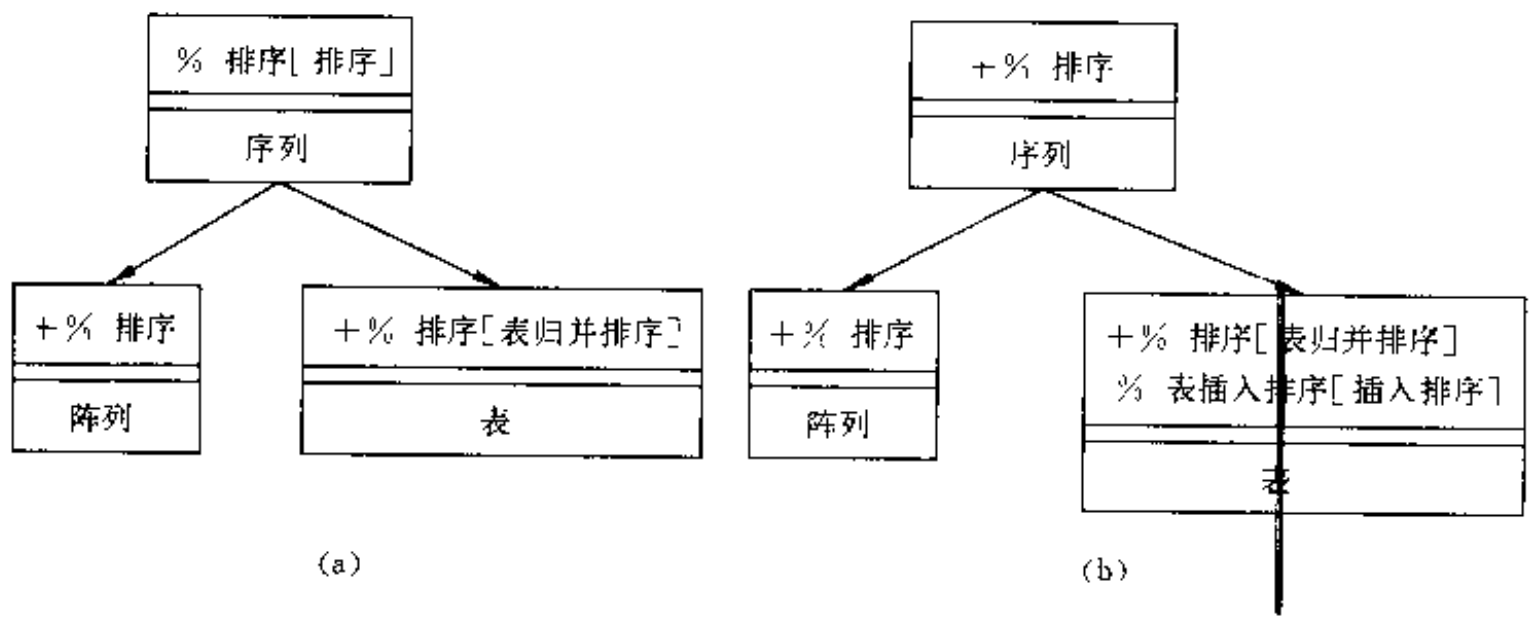


图 4.4.20 对方法分类图的引用

复使用的有力标志是:(1)方法分类图中的扇出度,(2)方法分类图中的层次数,(3)方法的CDS 中的相似程度。具有高扇出度的方法分类图,是对引入更抽象的类,以提高设计中重复使用度的有力标志。这一点可用图 4.4.21 来例示。

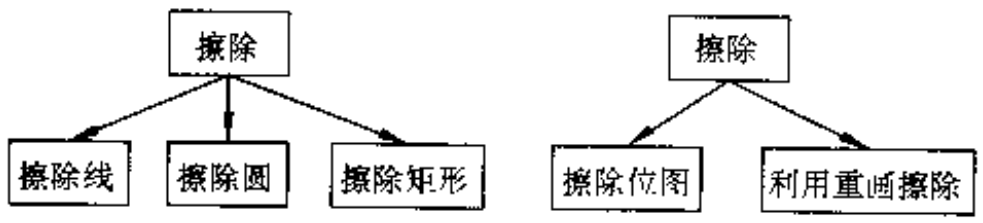


图 4.4.21 方法分类图的多形性评价

图 4.4.21 左边的方法分类图没有右边的好。这是因为左边的图对一个简单图形对象的擦除,要用三个不同的方法集来实现,而右边的设计,相比之下,更通用(即,它建立在较

抽象的类的基础上)。“擦除”例程的解答,传送给“画”方法分类图,它仅仅使用相反颜色映射将“画”函数用于对象。当引入新的图形类时,只需要设计少得多的方法集。举例来说,在图 4.4.21 所描述的系统中增加一个新的类,最坏情况下,只需要为新类定义一个“画”例程。

4.3 类 型 图

最初接触面向对象编程,会使人相信对象类间最基本的关系是那些在继承网格中定义的关系。但是,从应用程序执行的观点看,许多有兴趣的关系,是通过对象类的属性特征的值隐含地建立起来的。那些从函数和关系方法,转向面向对象方法的应用程序和数据库设计者,注意到的最基本的一个区别是,对键和引用完整性问题的定义和管理强调不够。由于 OOPL 中的对象是由其系统定义的处理机制唯一标识的,存取权一般围绕该对象相对于其他对象所担当的角色面构成的,这些角色通常由对象类属性建立。设计这些角色,通常是为了确定某个特定行为的边界。这样,相关的对象类型常常是决定对象对某个特定角色的适用性的关键约束。这个角色定义过程的管理,通过 IDEF4 类型图完成。在 IDEF4 的规范中,所有属性有一个返回类型,定义为它们的返回值的类型,IDEF4 中的类型图,作为类子模型的一部分,提供图形的和文本的标记来显示 IDEF4 模型中类属性的返回类型。

4.3.1 类型图符号集

类型图的目的是对返回类型展示其返回一个值的类特征。由于这个原因,类型图使用了图 4.4.22 所示的继承图中的类盒子。但是,在类型图中,只显示被说明为属性、函数或槽口的特征,不用辅助特征符号(&,+,!,^ 和 *)和输出线。

类型图包含类盒子和类型链,类型链有两个作用:

① 阐明返回类型,

② 显示其逆或部分逆。

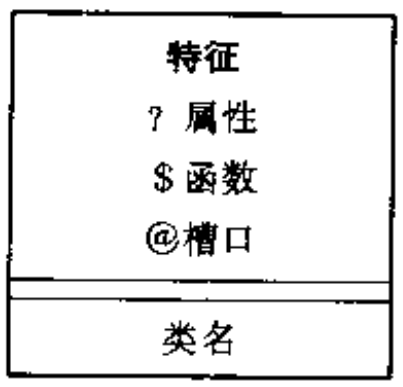


图 4.4.22 类型图中的类盒子

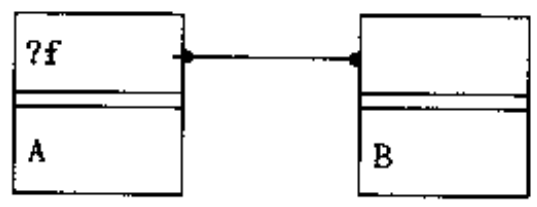


图 4.4.23 返回类型是类的实例

类盒子间的类型链通常至少有一端伸入类盒子中,指向类盒子中的某个特征。这表示靠近穿透类盒子的类型链的特征,返回了这样类型元素的值,其类盒子是放在类型链另一端的。

没有逆或部分逆的链接:

(1) 如图 4.4.23 所示，一个类中特征的返回类型可能是另外一个类的实例。这用一条线来表示，这条线从类盒子里的特征名伸出，穿过盒子外的点，到达返回类型的类盒子外的点。举例来说，图 4.4.23 中的图表示类 A 实例中特征 f 的返回类型是类 B 的一个实例。

(2) 特征的返回类型可能由类型的结构化的集合组成。用来表示结构化集合类型的链和图 4.4.23 中的简单类型链相似，除了一个“半菱形”或称“乌鸦脚”(>)被用在类型链的一端。图 4.4.24 表示类 A 的特征 f 返回由类 B 构造的某些类型的值。比如说，这个符号可以用在 f 代表类型 B 的一系列对象时，CDS 或 CIDS 将用来支持 f 代表的结构的确切的类型。

在支持持久性的面向对象系统的设计中(即那些用于 ODBMS 的实现的)，与逆和部分逆有关的决定变得很重要。“逆”和“部分逆”的概念例示如下。在图 4.4.23 中，类型 A 的实例有一特征 f，它将返回一个值，这个值是类型 B 的一个实例，但是，类型 B 的实例不携带任何与 A 相关的实例(如果有的话)的信息。这种和类型 B 相联系的可逆函数的缺乏，通过关系链在 B 的类盒于边缘中止表示。同样的理由可以用于说明图 4.4.24 中的类型，表示没有可逆函数。

有逆和部分逆的链：

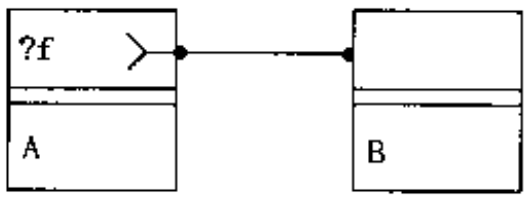


图 4.4.24 由某类构成的返回类型

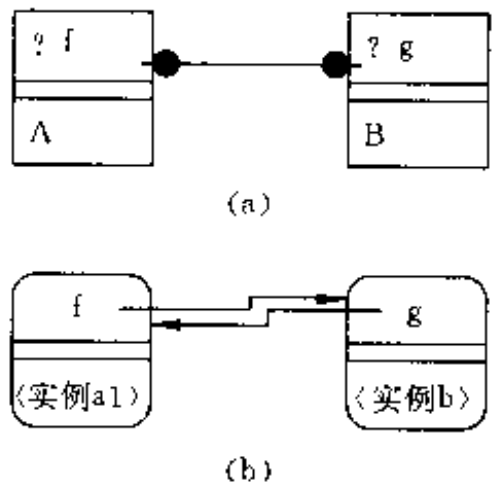


图 4.4.25 可逆关系

(1) 图 4.4.25 (a)例示了如何在一张类型图中显示两个类型实例间的逆关系。图显示了类型 B 有一个特征 g 和 A，B 之间的链相连接。这表示类型 B 的某一实例的特征 g 包含一个值，这个值是类型 A 的一个实例，并且实际上恰恰是 A 的这个实例具有类型 B 的那个实例作为它的 f 特征。因而，我们可以认为类型 B 的实例具有一个“在哪里使用”的指针指向类型 A 的实例。如图 4.4.25 (b)。因为这个链的头上没有乌鸦脚，我们知道这个关系是双向起作用的，因此我们称 A 的 f 具有 B 的逆 g。相似地，我们称 B 的 g 把 A 的 f 作为它的逆。

(2) 当考虑类型间非一对一的关系时，要讨论部分可逆的问题(见图 4.4.26)。图 4.4.26(a) 表示对于 A 的一个实例，特征 f 返回一些由类型 B 的实例构造的某种类型的值；并且，B 的那些实例中，任何一个的特征 g 将返回 A 的那个实例。如果我们有类型 A 的一个实例 a，然后 f (a) 是某种由类型 B 的实例构造的对象的结构，并且对于类型 B 的任何 b，它作为构造 f (a) 的元素，都有 g (b) = a。

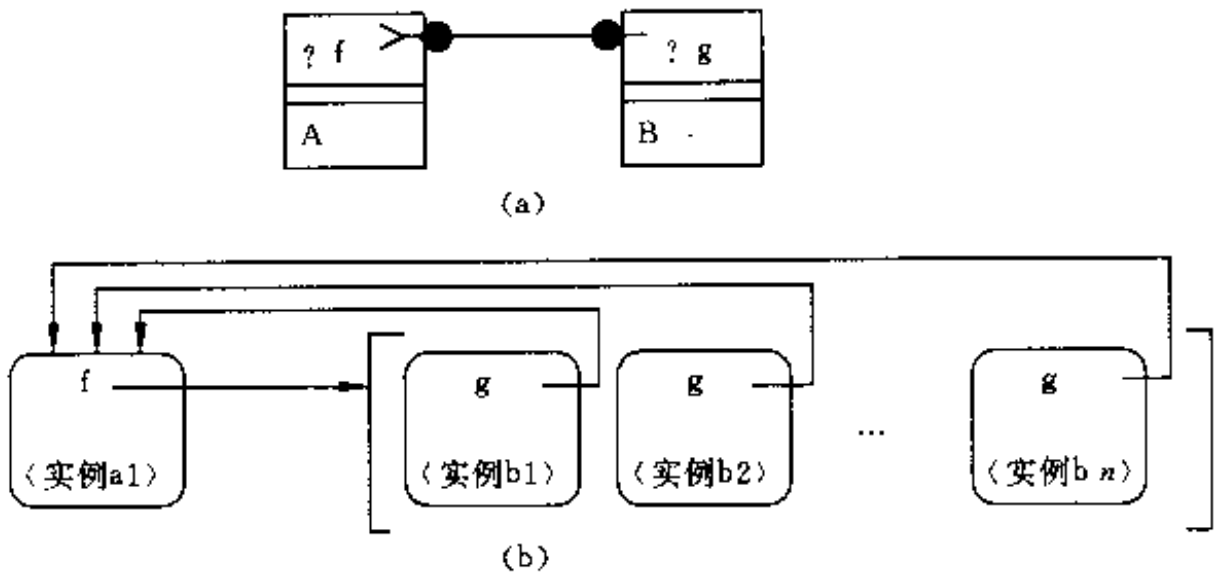


图 4.4.26　部分逆

在类型图中，每个特征的返回类型都必须表示出来，但是设计者可以自由选择是使用文本的注释，还是图形的链，还是都用。

属性名后跟特征返回类型，减少了图的复杂性(见图 4.4.27)。在较大的图中，通过取消许多链接，用这种句法减少图的不必要的繁杂。它还可以被设计者用来不强调某些关系，而把设计评阅人的注意力，集中在特殊的关系上(即那些用链显示的关系)。最后，这种方法常被用于较普通的数据类型，如整型和布尔型。对这些数据类型来说，类盒子可以和所有图形化的类型链一起从图中省去，这样简化了图，避免了不必要的繁杂。

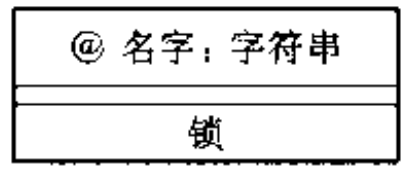


图 4.4.27　用文本显示的返回类型

4.3.2　理解类型图

我们已经讨论了用于构成类型图的符号，再看一看这些符号是如何组合起来形成完整的图的。从设计管理的观点，类型图提供了隐含在每个特征类型定义中的类间关系的可视化表示。这个信息在继承网格中是看不到的，但是，经验表明，这些相互关系的管理，对开发大型面向对象的系统来说是很关键的。通过类型图的使用，编程小组能够交流某些关系，他们还能够使用类型图，很快看到一个已提出的改变会引起的效果。

在 4.1.2 节中，图 4.4.7 用网络管理系统例示了一张类继承图。在这一节中我们将继续用这个主题看一看类型图。从那张类继承图中，我们选取类“字符串”(见图4.4.28)。

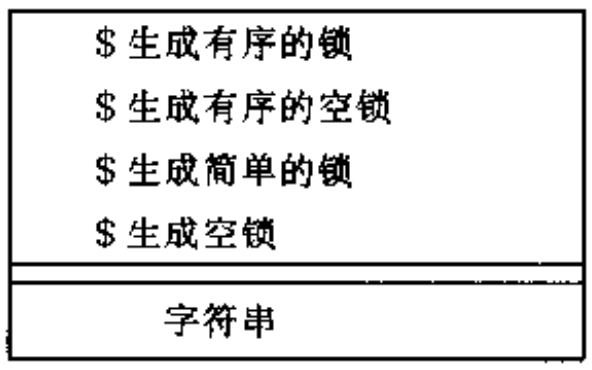


图 4.4.28　“字符串”的类盒子

这个类中的每个特征都是具有返回类型的函数，该返回类型是另外某个类的实例。

图 4.4.29 表示函数“生成有序的锁”返回一个类型为“有序的锁”的对象，函数“生成有序的空锁”返回一个类型为“有序的空锁”的对象。

为了表示槽口“要求”中，要包含类型“打印要求”的一个对象集，要在类盒子中，特征“要求”旁，画上带半菱形的连接线（见图 4.4.30）。或者是 CIDS 或者是 CDS，必须提供“打印要求队列”所含有的对结构类型作具体说明的信息。图 4.4.31 显示了完整的类型图。

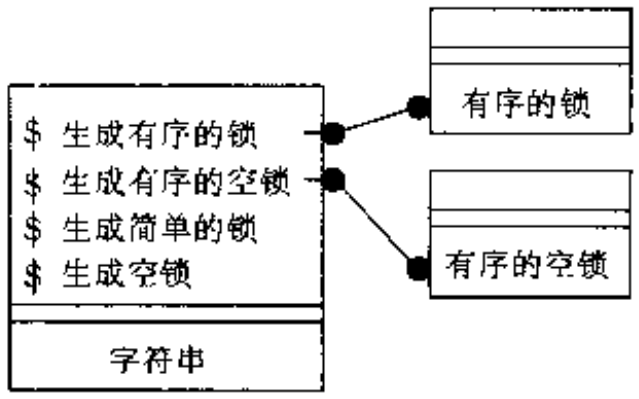


图 4.4.29 简单的返回类型

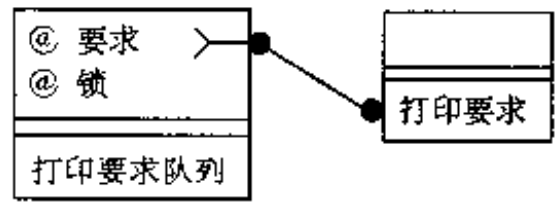


图 4.4.30 由其他类型构造的返回类型

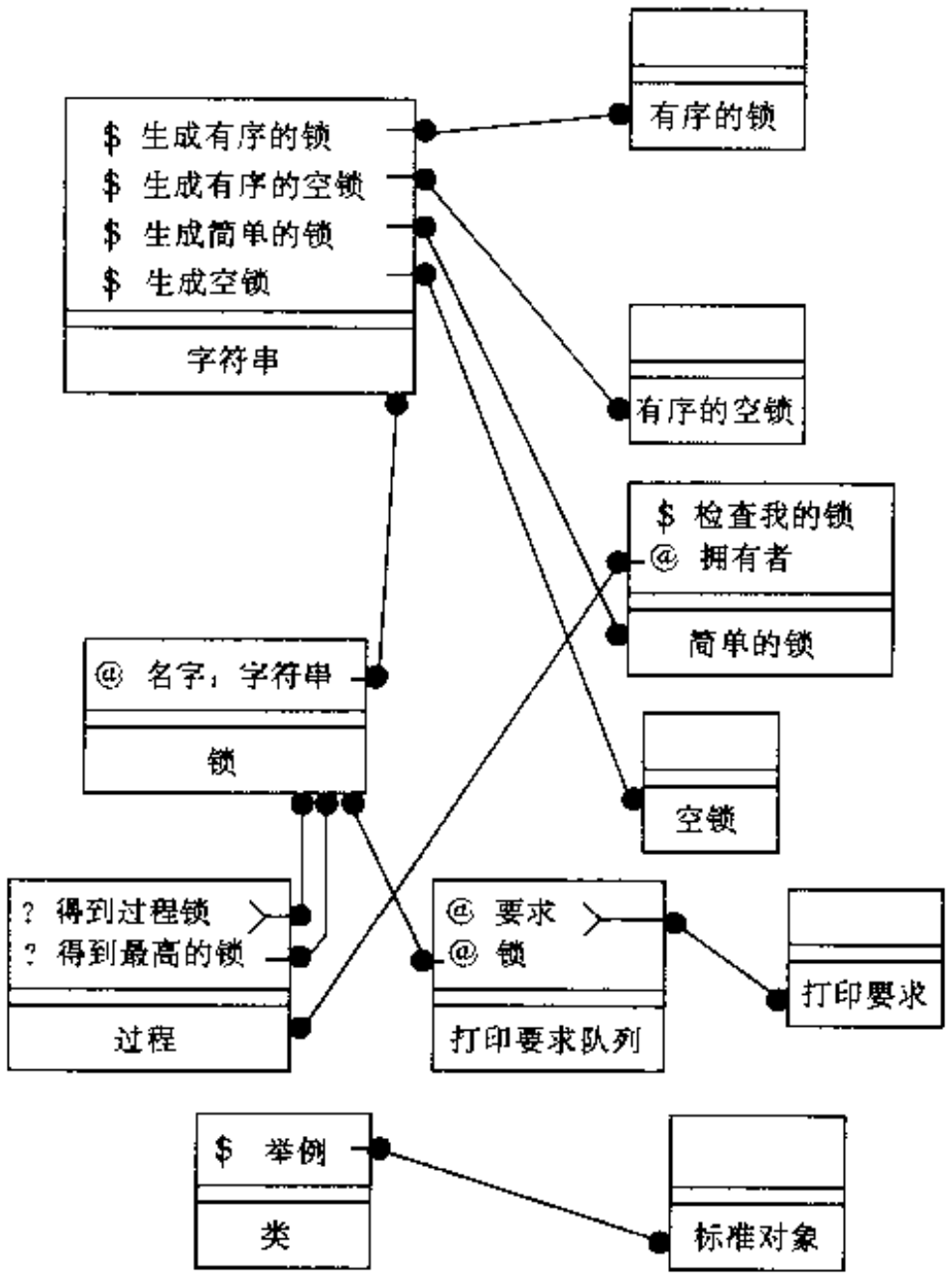
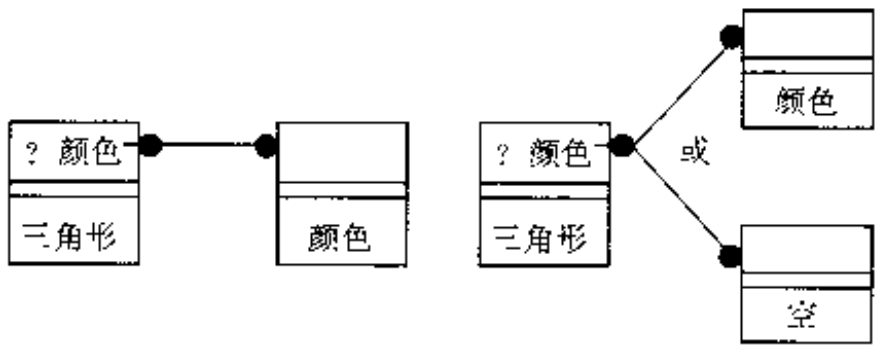


图 4.4.31 类型图(网络管理系统)

有可能出现这种情况，一个例程可以返回两个类型中的一个。比如说，在图 4.4.32 中，类型图能够显示特征“颜色”，返回类型“颜色”的一个实例。但不可能显示，特征“颜色”将要：或者返回三角形的颜色（如果它有颜色），或者返回“空”（如果没有颜色）。换句话说，IDEF4 类型图（和协议图）的图形句法，不允许表示一个类特征的返回值有多个返回类型。如果这种情况发生，就有必要引入一个基于不同返回类型的更抽象的类。如果不可能定义一个更抽象的类，或者如果额外的类型仅在例外情况下需要，那么这种情况就应该在 CDS 或 CIDS 中作为约束说明。



（a）允许的　　（b）不允许的

图 4.4.32　允许的/不允许的类型图

4.4　客　户　图

在这一节中，我们将讨论 IDEF4 模型的客户图的成分。客户图，作为方法子模型的一部分，被用于算法分解。尽管是抽象的，它们是唯一用于说明方法内部结构的 IDEF4 图。

4.4.1　客户图符号集

客户图使用两个符号——盒子和链。客户图盒子代表特征或特征-类对(feature-class pairs)。首先是类名，后面跟着一个冒号，特征名写在下面。连接这些特征类盒子的链称为客户链。

这些客户链用箭头表示，箭头的两端都有指向相同方向的双钩（见图 4.4.33），箭头从“提供者”（被调用或引用的特征）指向“客户”（进行调用或“进行控制引用”的特征）。盒子本身代表特征或特征-类对。客户图一般都很小，并且集中于某一个中心例程（严格地说是例程-类对）。

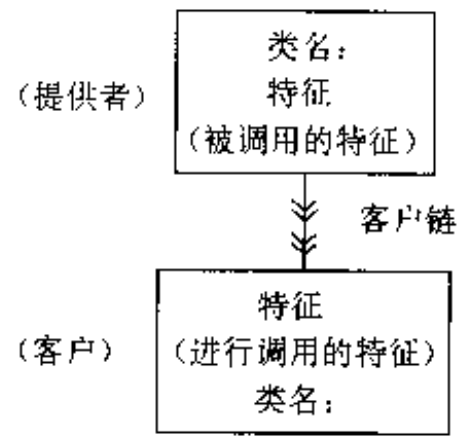


图 4.4.33　客户图符号

4.4.2　理解客户图

在客户图中，盒子间的链代表控制引用，或编程语言中从一个例程到另一个例程的“子程序调用”的存在。客户图具体说明了方法将有的算法结构。

我们将用从“过程：过程等待”到“简单的锁：抓住”(simple－lock：seize)的链作为一个例子来说明客户链要表示什么（见图 4.4.34）。因为例程-类对“简单的锁：抓住”在客

户图中作为客户出现(即在客户链的前面)。“简单的锁”必须是“抓住”在其中直接出现的一个类(即“抓住”在“简单的锁”中不可以是虚拟的)。回顾 3.2.3 节,直接出现特征,是那些名字显示在类的类盒子中的特征。直接出现的特征,在它们的名字出现在其中的类中,或者定义或者重定义,因此“简单的锁:抓住”作为客户出现,表示设计者打算为“简单的锁”中的“抓住”提供一个新的或重定义的方法(没有在任何“简单的锁”的超类中提供的方法),它也说明了“抓住”方法集的实现,打算调用在类“过程”中定义的特征“过程等待”。

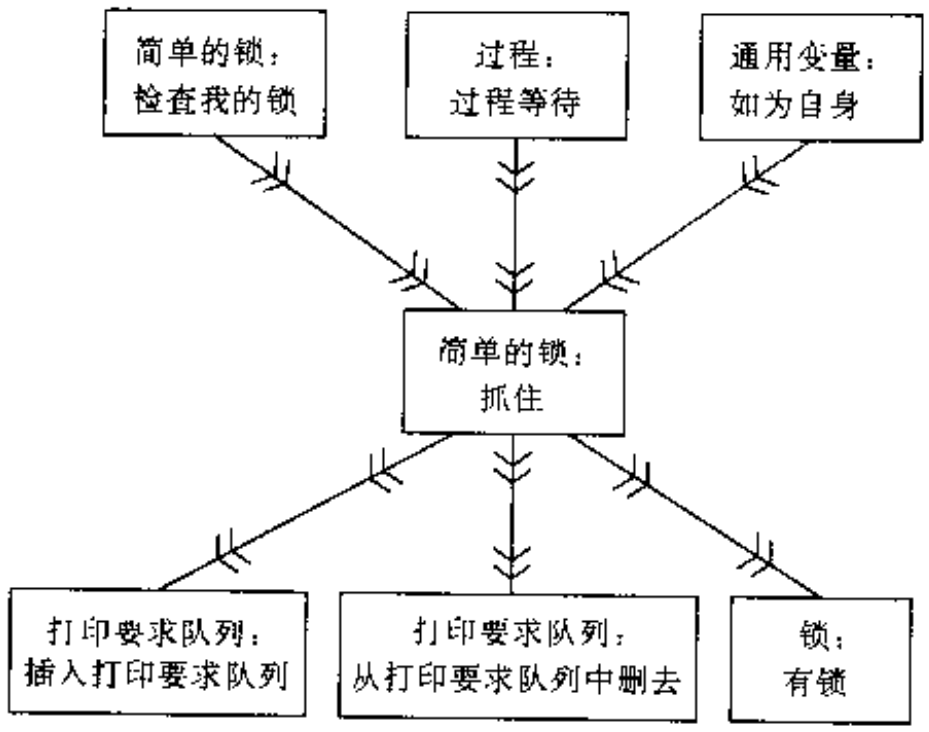


图 4.4.34 “简单的锁:抓住”的客户图

注意,该图定义了,类“简单的锁”中“抓住”的实现,将直接调用实现了的例程“过程等待”,而不是“过程等待”的某个类属例程(该图表示这个例子的分配是静态完成的)。如果和例程联系的类,在图中没有具体说明,这个分配将在任何实现运行时发生。对 C++ 中的实现来说,这表示需要虚拟函数。

4.5 协 议 图

当一个特征被激活,它总是相对于某个特定的对象而被激活的,这个对象是特征所在的某个类的实例。根据某些 OOPL(像 Smalltalk 和 Old Flavors)的术语,特征是“送”给对象的“消息”。在其他语言中(像 New Flavors 和 CLOS),特征是以对象为参数被“调用”的类属函数。IDEF4 采用这个以对象为参数被调用的函数。IDEF4 采用这后一种观点,但用了术语“特征”,而没有用“类属函数”。在 IDEF4 中,特征被认为是被调用时至少有一个参数(自身参数)的类属操作符。这个概念自然导致了考虑其他参数的存在。这些其他参数被称为第二参数位置或第二参数。IDEF4 不要求执行于一个对象上的特征返回一个值(并不是所有的特征都是属性)。

IDEF4 类型图提供了属性的“返回类型”,但没有提到特征的参数或它们的类型。术语“协议”指特征和它的输入、输出的界面的具体说明。IDEF4 引入“协议图”(类子模型的一部分)来表示这些信息。这些小图用于表示特定特征的输入输出。

4.5.1 协议图符号集

用于建立协议图的符号示于图 4.4.35 中。一个简单的盒子用来表示特征——某个特定的图的中心。圆角盒子表示中心特征的参数位置，修改后的类盒子用于表示参数类型和类型链（见图 4.4.35）。

中心特征的名字出现在图中心的一个普通的方盒子里。每一张图的注意中心，都是一个例程-类对。在完整的 IDEF4 设计模型中，每一个例程-类对都应该有一张协议图。例程-类对是图的标识，并被用于组织文档中的图。圆角盒子代表输入和输出。输入参数放在中心特征盒子的上方，输出参数放在下方。一般的惯例是将自身参数作为第一个参数放在左边。为了避免类型链的不必要的重复，只显示特征的根类（在整个继承子模型中）。如果有多于一个的根类，就用连到自身参数位置的多重类型链全部显示。参数用图 4.4.35 所示的方式和中心特征相连。参数类型通过类型链和参数（输入和输出）相连。如前所述，IDEF4 不允许多于一个的自身参数。参数类型用只给出类名的类盒子表示。

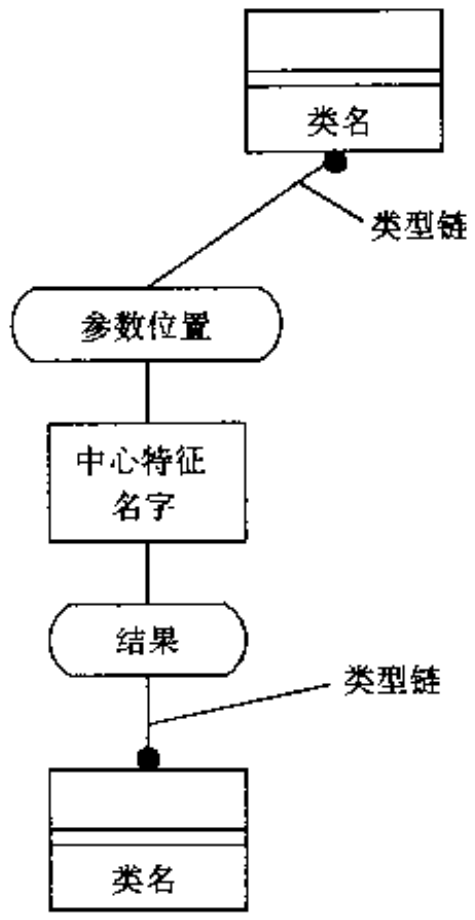


图 4.4.35　协议图符号集

4.5.2 理解协议图

既然我们已经了解了协议图的符号，我们将看一看它们怎样被用来建立一张图。在网络管理系统的例子中，我们有如图 4.4.36 所示的具有特征“检查我的锁”的类“简单的锁”。

函数“检查我的锁”的协议图，如图 4.4.37 所示，它把这个特征名放在图中央的中心特征盒子内。该图显示了例程-类对“检查我的锁：简单的锁”的协议。在图中，圆角盒子具体说明了参数名。在最左边参数盒子中的参数名左边的词“自身”，是一个修饰词，表示这个参数是基本参数。“检查我的锁：简单的锁”的剩下的参数，出现在这个基本参数的右边。图中的基本参数是“可能拥有的”，它属于类型“简单的锁”。那个第二参数“可能的拥有者”，属于类型“过程”。

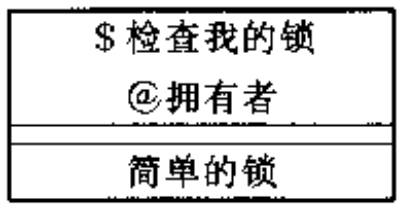


图 4.4.36　具有“检查我的锁”特征的类盒子

协议图本身提供了一个特征输入、输出特性的总结。协议图也用于建立显示其他有关参数位置和它们的类型信息的视图。如果没有协议图作为成分，这样的视图是不可能建立的。为图 4.4.37 中的协议图建立一般类型的视图，可以通过为“简单的锁”的所有子类，增加继承链来形成，这些子类中“检查我的锁”是被重定义的（碰巧，这里一个没有）。

通常，只有例程被认为有多于一个的参数；但是，我们对协议的讨论适用于所有的特

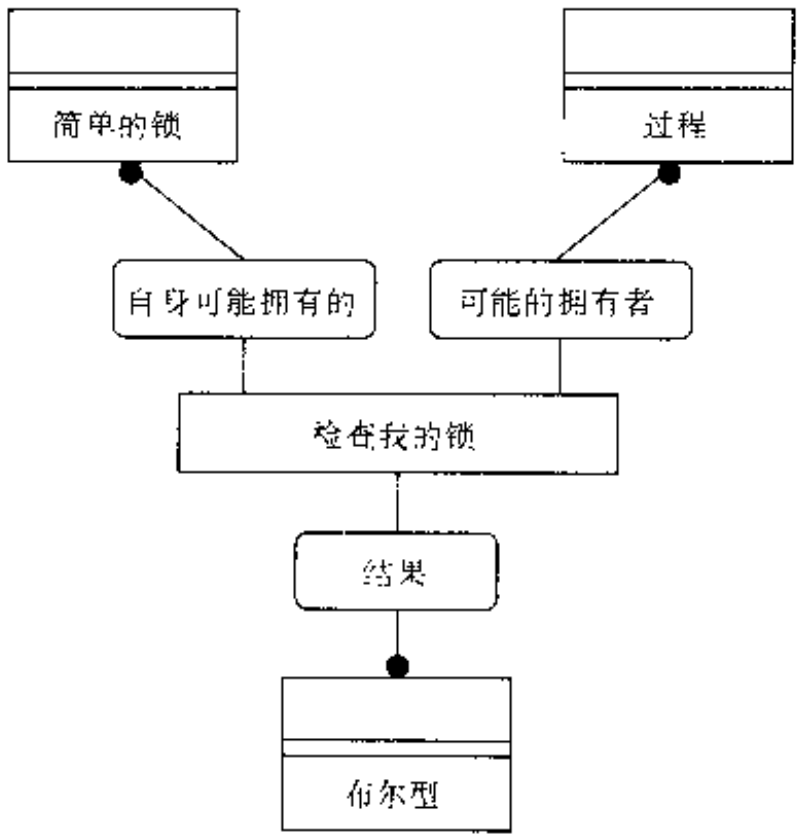


图 4.4.37 "检查我的锁：简单的锁"协议图

征，包括槽口。实际上，仅有一个参数和一个返回值的属性也有协议，即使还不知道它是函数还是槽口。辅助特征符号"!"(新的分类说明)，可以用来使特征在子类中更具体地重定义为函数或槽口，也许在不同的子类中不同。

通常，槽口的协议仅有自身参数，并且返回一个在使用它的实例中存储于该槽口的不管什么样的值。但是，也可以想象，槽口具有附加的参数。比如说，一个槽口存储的是类似于数据库的值表，而不是一个单个的值。这样的一个槽口 f，可以作为类 A 中的一个多参数属性，给予一个协议，而不要求设计者认为 f 的状态是一个槽口（或许，除了在 A 的某个子类 B 中）。附加的参数是找出表中正确的那一行的关键，从那一行可抽取合适的返回值。属性 f 可以在 A 的另一个子类 C 中被定义为函数，不进行查表而是计算它的返回值。这种槽口和函数间的区别完全不需要在 A 中表示出来。

4.6 分配映射

IDEF4 中的分配映射和 OOPL 中的分配类似。一般来说，可以有多于一个的方法和类属例程相联系。在 IDEF4 中，在同一类中，一个类属例程可能有多于一种的方法。在 OOPL 的实现中，每个例程-类对只能有一种方法。在 OOPL 中，分配是指在编译或运行时，为类属例程的某个特定调用，寻找合适的方法的过程。在 IDEF4 中，方法集就是服务于实现的一系列方法(或方法集)，IDEF4 把分配映射和分配看作是一系列方法的例程、类、和合同之间的联系。

回忆一下，一个 IDEF4 模型由类子模型和方法子模型组成（见图 4.4.38)。这两个子模型通过分配映射相连。IDEF4 模型中的分配映射，或清晰或隐含地在继承图和方法分类图中记录下来。当一张图（继承或方法分类）包含分配，它指出哪个方法集合同将用来实现某个特定的例程-类对。

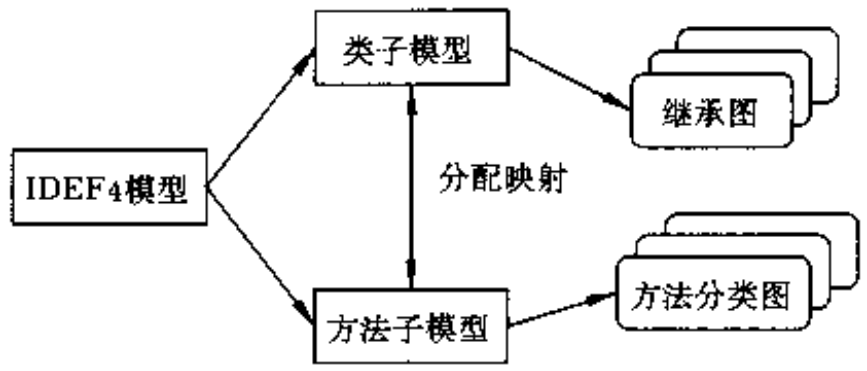
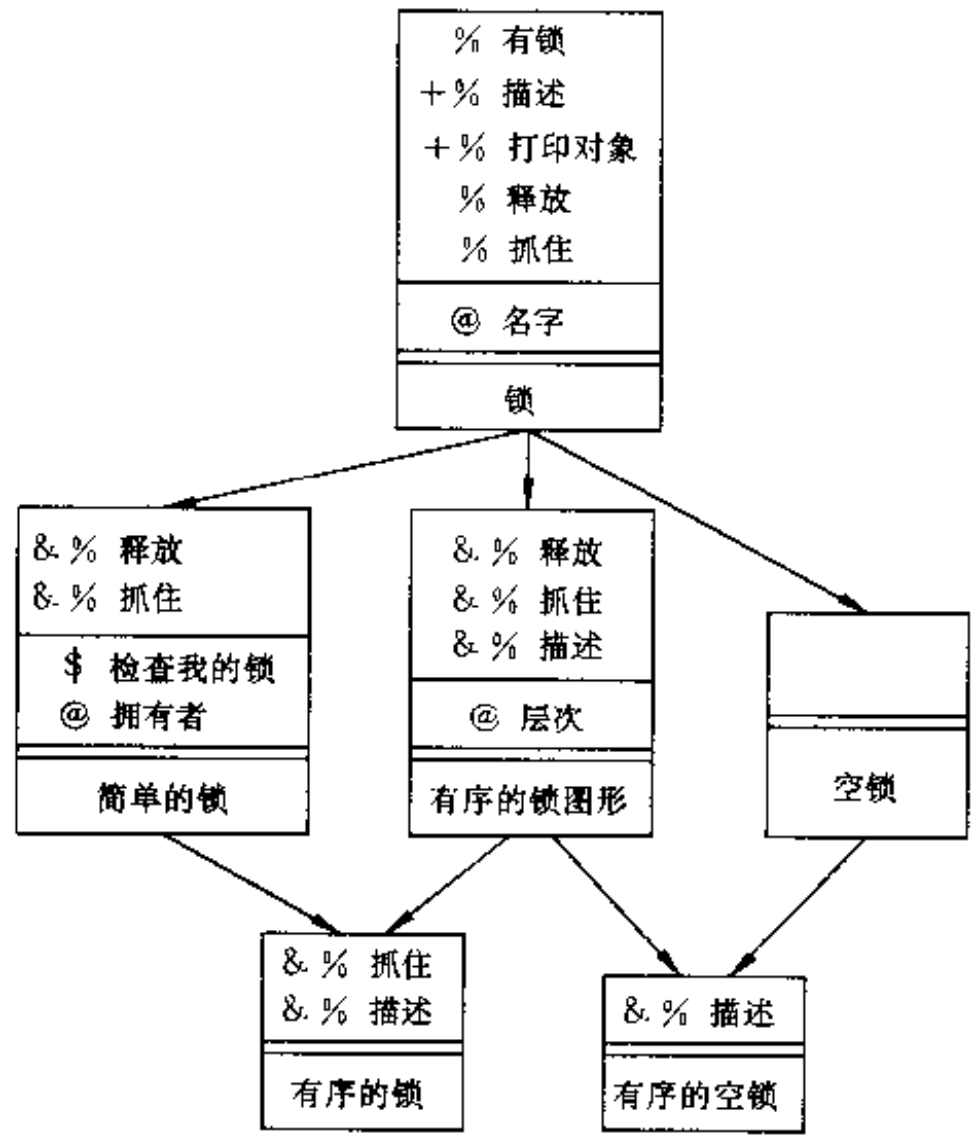


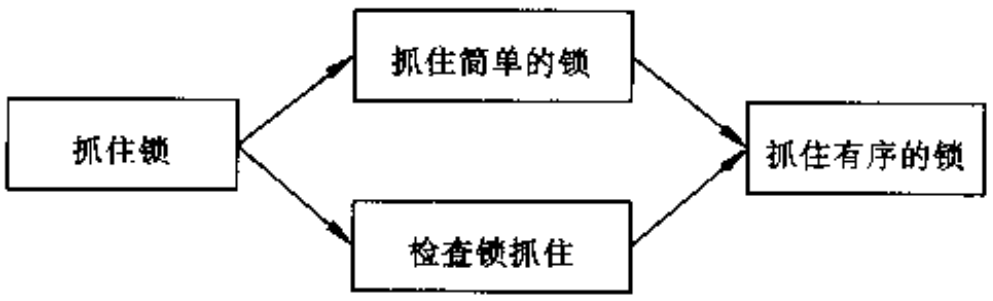
图 4.4.38　IDEF4 模型成分

用这两个图类型来说明分配，不需要改变任何一个图的一般句法。每个图画出附加的分配说明。在类继承图中，“分配”通过把方法集名放在括号里，置于例程名的右边来表示。在方法分类图中，“分配”通过把例程-类对放在括号里，置于方法集名的下面来表示。

举例来说，考虑在我们整个讨论图的过程中，一直使用的网络管理系统。图4.4.39



(a) 类-继承



(b) “抓住”方法分类

图 4.4.39　部分图

(a) 和 (b) 分别显示了“锁”的类继承图和“抓住”的方法分类图。这些图可以用来显示类-继承图中的例程，和方法分类图中的方法集之间的映射，图 4.4.40 在“抓住”例程的继承图中，显示了类、例程、方法集的映射。在图 4.4.41 中，“抓住”例程的同样的分配映射，显示在“抓住”的方法分类图中。

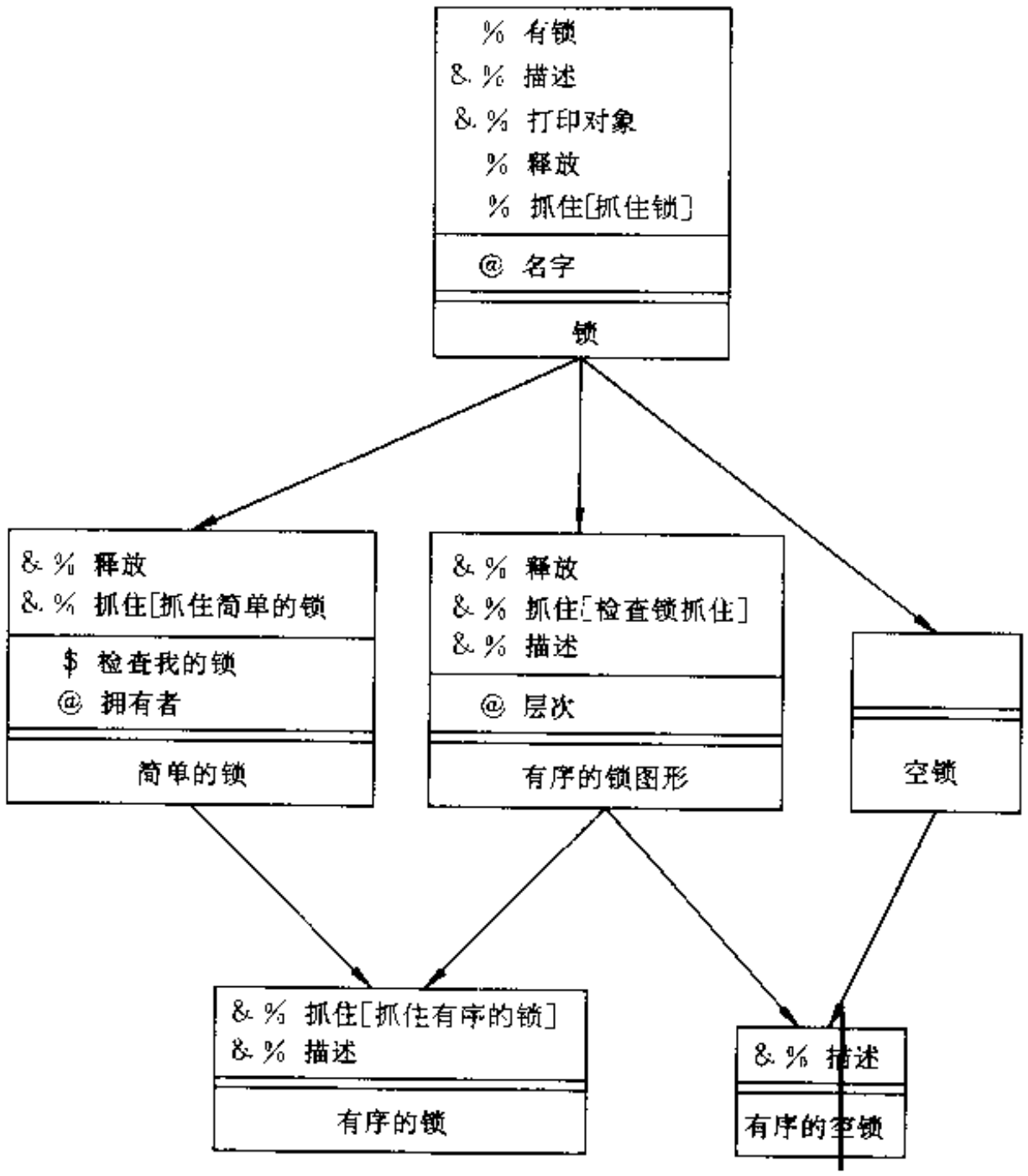


图 4.4.40　具有“抓住”分配映射的继承图

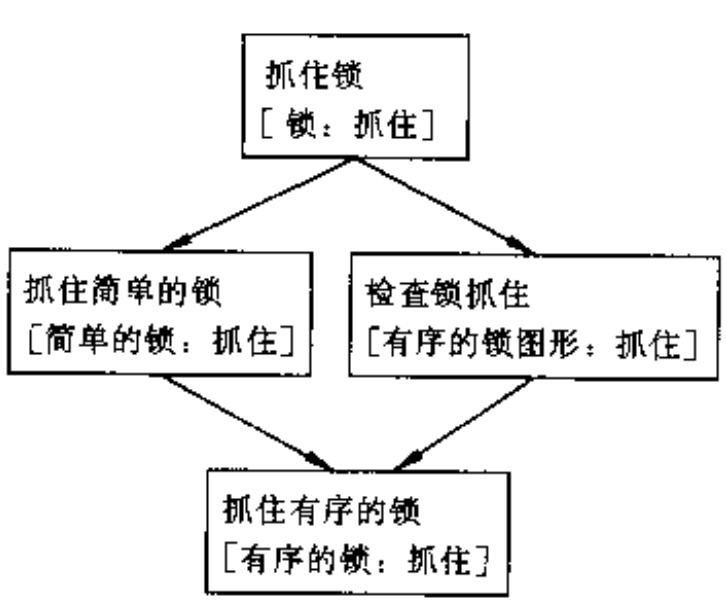


图 4.4.41　方法分类图

分配在方法分类图中的表示，和在继承图中的很类似。但是，由于这些图的注意中心是方法集，例程-类对的名字列在方法集下方的括号里。只有例程直接出现于其中的类，才被表示出来。

4.7 IDEF4 实例化语言

实例化语言的目的，是使测试用案例的场景开发成为可能。测试用案例场景，反过来，被用来验证所做的设计布局例子的设计和文档的有效性。最终，这个验证过程帮助编程员实现设计。实例化语言的图形映像，看上去很像 IDEF4 类型图。实例化语言，用类似于 IDEF4 类盒子所用的圆角盒子，代表类的实例。其中使用的值链，则类似于 IDEF4 的类型链，但是两端都没有实心圆。

4.7.1 IDEF4 实例化语言简介

在 IDEF4 中，类的实例图形化地表示为分成两个区域的圆角盒子（见图 4.4.42）。最底部的区域用来唯一地标识实例（一个对象有状态、行为和一个唯一的标识）。这个唯一的标识符，通过把实例的类名和实例号连在一起，并把它们用尖括号括起来形成。比如说，类“顶点”的第 13 个实例，把＜顶点 13＞作为它的唯一标识符。

实例属性（返回值的，也许是计算引入的），在 IDEF4 实例盒子上面的区域中被列成一列，正像它们在 IDEF4 类盒子中一样。IDEF4 实例化语言提供两种方法，表示分配给属性或由属性返回的值，也就是，使用值链或把值实例的唯一标识符打在属性的右边来完成。在数值类型实例的情况下，最好打出值。允许使用值 1 代替＜整数 1＞，作为整数 1 的唯一标识。IDEF4 的值链类似于 IDEF4 的类型链，也是从 IDEF4 实例盒子内部紧挨着其值被注释的属性出发，并以指向另一实例盒子的边界的箭头结束，如图 4.4.44 所示。值链可以和几个指向同一个实例的值链汇合在一起，如图 4.4.44 中由“顶点”实例的双亲属性引出的链。

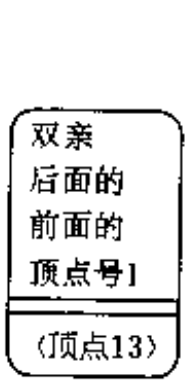


图 4.4.42 顶点类的 IDEF4 实例

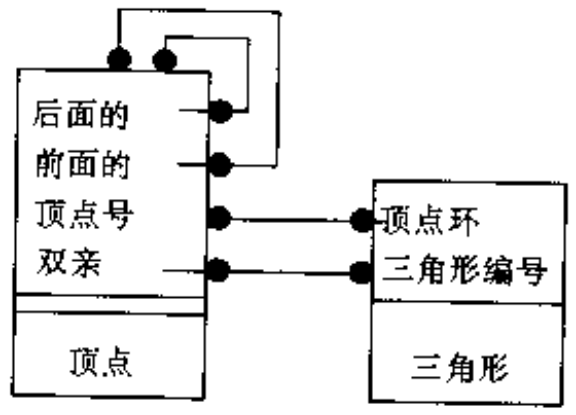


图 4.4.43 三角形例示化场景的类型图

图 4.4.43 和 4.4.44 分别显示了类型图和例示化图，表示具有顶点的三角形。顶点具有成为顶点的双向链表的一部分所必需的属性，和指向其双亲（在本例中是“三角形”的实例）的属性。“三角形”类具有“顶点环”属性，它指向顶点的双向链表上的任何实例。类“顶点”和“三角形”都有在类型图上没有具体说明其类型的编号属性，如图 4.4.43 所示。

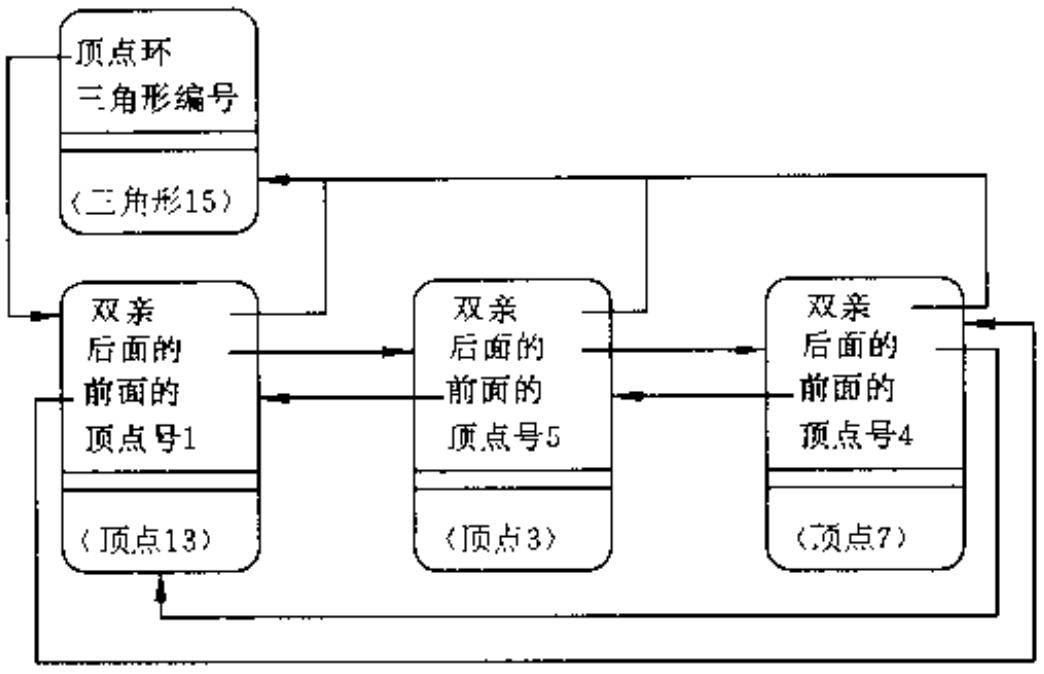


图 4.4.44　具有顶点的三角形的例示图

4.7.2　IDEF4 设计的证实

如果我们想为一辆汽车进行面向对象的设计，我们一定有类“引擎”和“车身”。“引擎”类有类型为“车身”的属性“我的车身”，并且“车身”有类型为“引擎”的属性“我的引擎”，如图 4.4.45 所示。这个设计在 IDEF4 中能很容易地并且无二义地用图 4.4.45 所示的类型图表示出来。

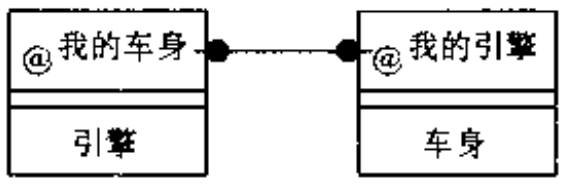


图 4.4.45　引擎和车身的类型图

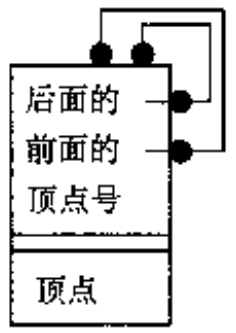


图 4.4.46　三角形顶点的类型图

如果我们想详细说明为一个三角形的三个顶点作的对象设计，这个三角形以双向循环链表的形式链在一起，此事可能就不那么容易了。这个设计的类型图示于图 4.4.46 中。问题是要记住定义约束，该约束具体说明：第 1 个顶点的前面的顶点是第 3 个顶点，而第 3 个顶点的后面的顶点是第 1 个顶点。这类说明的问题，可以通过为顶点设计原始的实例化场景来解决（见图 4.4.47）。这些场景可以用来测试演进的设计，说明是否表达了我们想表达的内容。在图 4.4.47 中，状态（即＜顶点 1＞和＜顶点 4＞的槽口值）是一样的，这样，如果双向循环链表配置的约束只涉及属性值（即，“顶点”实例的状态），并且对“顶点号”属性的唯一性没有约束，那么系统将无法区别＜顶点 1＞和＜顶点 4＞，另外，还可能出现这种情况，顶点的“后面的”和“前面的”槽口所指的值和说明一致，但却以与预期设计说明不一致的方式使用（见图 4.4.47）。

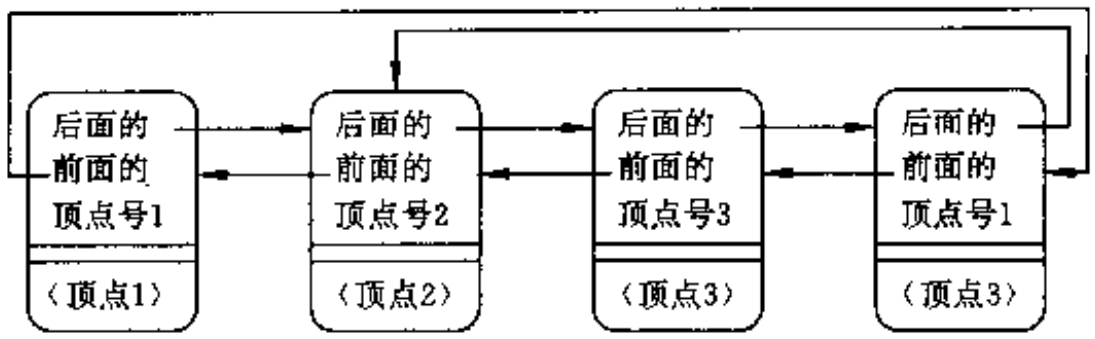


图 4.4.47　和设计一致,但错误的例示

第 5 章　IDEF4 设计开发过程

设计过程是软件系统开发生命周期早期极为重要的活动。设计活动产生的设计成果对实施和后面的持续活动都是很重要的。设计生命周期的概念，常作为一个方便的工具，用来帮助产生对基本设计过程的理解，特别是为了管理的目的。从这样一个观点来看，设计过程被认为从某一点开始，持续发展，趋向成熟，并最终停止。这种设计观点（看作一系列渐增的和顺序相关的步骤），试图安排过程的步骤，使得每一步可以被看作一个独立的状态，只不过是它的发生和它周围的状态有关。系统的使用，常常揭示一些，或者是系统没有考虑到的，或者是系统环境变化产生的问题。

为处理设计中时常出现的情况的复杂性，设计战略常被考虑为“元-计划”。它们可以被看作是，对上面所说的原始设计活动的系统化和组织。有三种类型的设计战略可以考虑：

1　外部约束驱动设计：软件目标、意图和需求还没有明确特征化、没有定义好的情况下的设计。当设计者过早被投入产品开发过程，这种情况经常出现。

2　特性驱动设计：软件设计是在严格要求、严格的责任和充分的验证，这种严密控制的情况下进行的。这些设计通常是面临潜在的生命威胁的情况。

3　转手(carry—over)驱动设计（例程设计）：改变已经存在的、实现了的设计或被很好理解了的设计（如排序）。

从 OOD(面向对象设计)的观点来看，外部约束驱动和转手驱动策略，是最普通的设计情况。后面各节所涉及的设计开发过程，是在为不同类型的案例，建立了几种 IDEF4 设计后，得到的经验和领悟的精华。

5.1　面向对象的分解

作为一个 OOD 方法，IDEF4 致力于系统要求的对象和行为类型的描述。着眼于由类所展示的行为（方法集）的类型，提供了一种合适的工具，把系统划分成小的、容易理解的片。这些行为类型，与数据类型一起（正如 OOD 中所做的），提供了一种更加模块化的设计；这导致了具有所希望的生命周期特征的实施，为此面向对象的实施成了众所周知的。

图 4.5.1 显示了根据类和它们的行为划分系统，通过允许功能和数据相联系，提供了模块的可组合性。这样，设计就致力于定义数据类型（类）必须显示的行为的类型。

图 4.5.2 显示了 IDEF4 设计和可能的实施之间的映射。每个 IDEF4 例程-类对，选择一个IDEF4方法集。IDEF4类，映射成一个实现了的类，而IDEF4例程，映射成类属方法

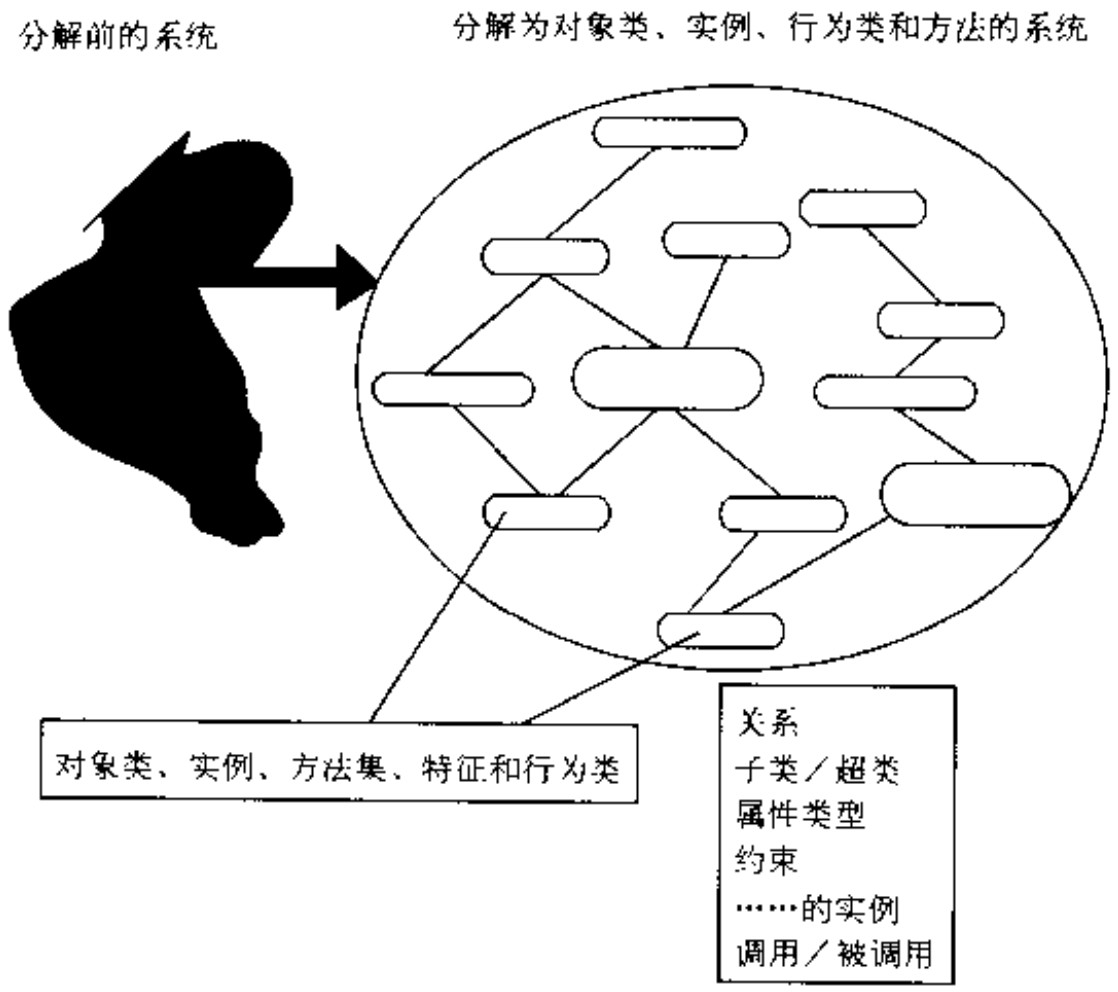


图 4.5.1　面向对象的分解

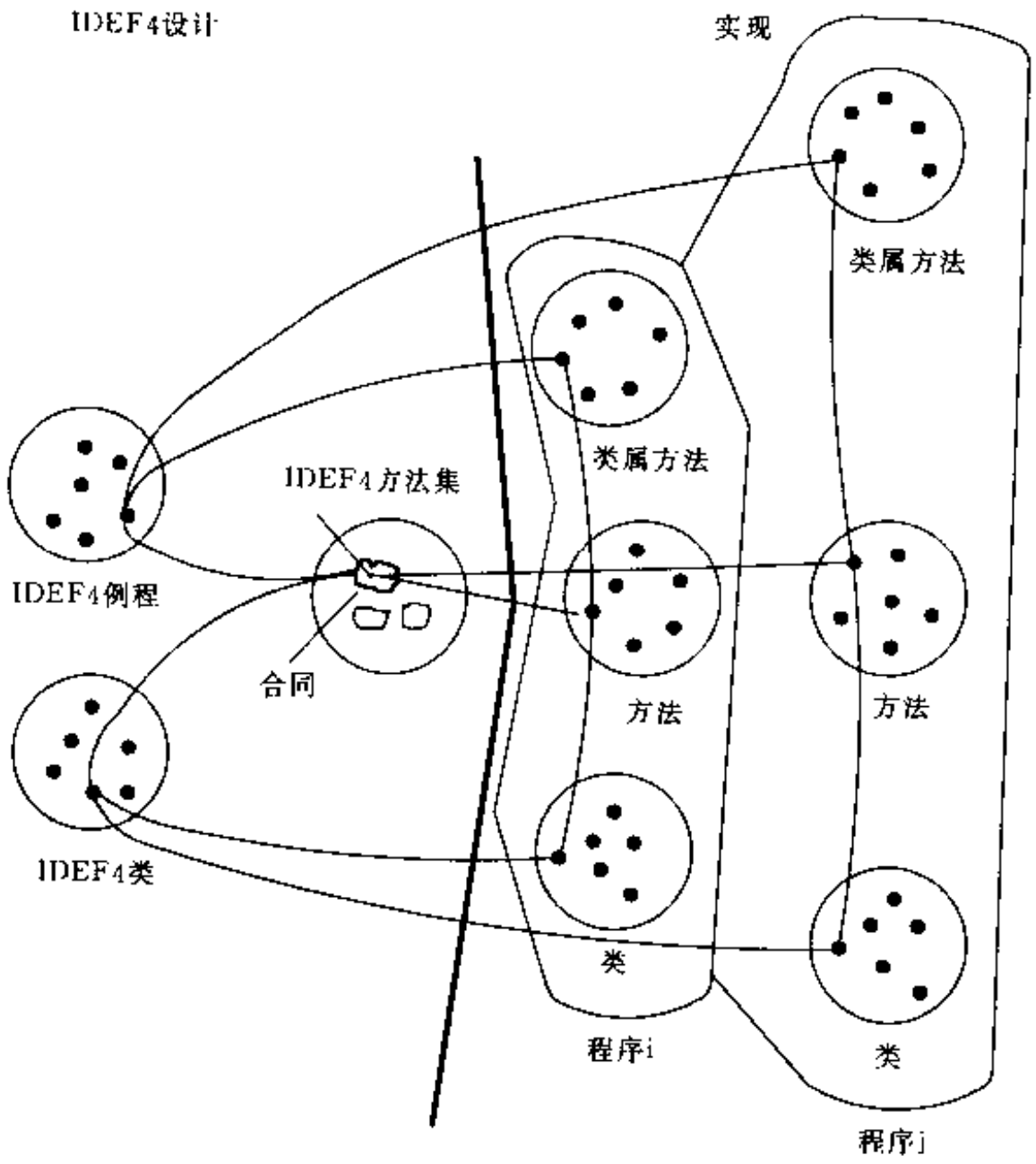


图 4.5.2　IDEF4 设计的实现

或实施中的消息。实施中的每个类/类属-方法对，选择实施中的一个方法。这个方法必须满足，分别和实现了的类和类属方法相联系的 IDEF4 例程-类对选择的方法集的合同。这样，方法集合同和 IDEF4 类的类-定常约束，对实施就提出了要求。这些要求限制得越严，实现者必须作的编程决定就越少，并导致可能的实现也越少。

5.2 IDEF4 设计开发活动

在 IDEF4 方法实现后，下述活动将在整个软件设计过程中被采用：

(1) 分析不断发展的系统需求；

(2) 开发类层次（继承图）；

(3) 开发方法分类（方法分类图）；

(4) 开发类的组成结构（类型图）；

(5) 开发协议（协议图）；

(6) 开发算法分解（客户图）。

通过 IDEF4 实践发现的这些类属活动之间的依赖关系，显示在图 4.5.3 中。方法集和类通常直接来自需求，而客户图来自方法集和方法合同。下面各节将详细解释每一个活动。

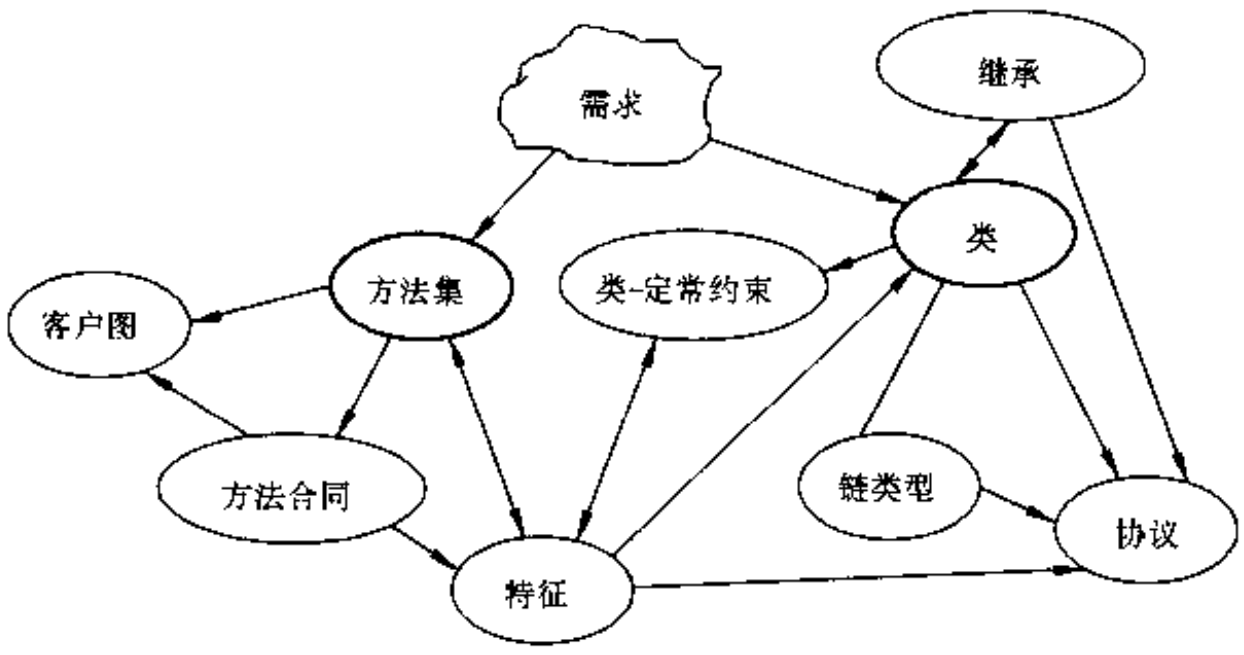


图 4.5.3 IDEF4 元素间的因果关系

5.2.1 分析不断发展的系统需求

系统设计开发的第一步，是系统需求的分析。系统将要做什么？涉及到哪些类型的对象？这种分析不仅仅包括系统需求。如果可以得到功能、信息或过程模型的话，它们也可以用来提供这个阶段的输入。这个早期分析的目的，是识别直观的类和系统行为的类型。换句话说，设计者把系统划分成粗略的对象类型集、对象组成集和行为集。

5.2.1.1 识别初始的类和方法集

在 OOD 中，类分解、对象组成、算法分解和多形分解之间存在某种紧张关系。现在还

没有一种对象设计方法支持对象、类、方法和算法结构之间的设计折衷。IDEF4 不仅为对象设计的这四个方面中每一个的“最少-承诺建模”(least-commitment modeling),提供支持,还允许人们把一方面的设计决定的效果,和其他方面的比较对照。IDEF4 所用的最少承诺哲理,允许人们逐步细化设计,从一个粗略地满足需求的初始设计开始,然后通过把附加的需求作为下一个设计周期的需求,对现有的设计进行迭代,如此循环直至设计趋于成熟。每次设计迭代,减少了和设计(设计正确)一致的、可能实施方案的数目。设计迭代逻辑上的结论是,设计将只有一个可能的实施方案(设计将是实施方案)。

需求分析应产生一张初始的类和方法集的表。对于那些不熟悉目标系统的领域的人来说,系统需求将是初始类和方法集的主要来源。这些表无疑是不完整的,只不过比缺少类特征和继承结构的潜在的类列表稍好一点。这些初始方法集,将进一步说明系统,并将主要建立在系统需求之上。如果可以得到未来系统的功能模型(IDEF0),那么该模型可以提供对所需系统行为和初始方法集的看法。

要易于产生这些初始类和方法集的表,取决于两个因素:

- 设计者对目标系统领域的熟悉程度,和
- 系统需求的明确程度和详细程度。

在开发的这个阶段,设计者主要是识别用户/顾客对系统能力和操作的期望。下一步,设计者将决定类结构和提供所希望的系统行为的设计方法集。

5.2.1.2 在分析中用其他 IDEF 方法

在理解顾客对系统的期望时,帮助分析的主要来源是系统需求。但是,由于不能总是希望顾客对涉及到的问题的实质有全面的了解,设计者不应仅仅把自己限制在需求文件上。快速、准确地识别初始类和方法集,对设计的成功实现,和对问题性质的充分理解,都是十分重要的。因此,关注所提出的系统及其环境的其他信息源,是很有好处的。这些信息源包括功能、过程和信息模型,和现有的 OOD。IDEF 家族包括了构造这些类型的模型的各种方法。

IDEF0 可以用来帮助设计者识别系统的一些概念和活动,建立功能模型。IDEF1 可以被设计者用来开发对组织使用的,并且因此必须由系统维护的信息的理解。IDEF3 提供了对过程流和对象状态转换的描述,这些将帮助设计者组织:(1)现有系统,(2)所建议的系统、(3)用户和系统的相互交连,(4)将由未来系统操作的对象经历的状态变化的内部工作的概念。虽然这些方法提供了许多有用的信息,但是它们很昂贵并且很费时。

首先,我们看一看功能建模(特指 IDEF0)对 IDEF4 设计过程的帮助。IDEF0 是用于建立功能和与之相关概念的模型的方法。但是,因为 IDEF0 建模方法主要强调的是功能建模,可能会对 IDEF4 的 OOD 有严重的影响。虽然目标系统必须完成的功能很重要,但是 IDEF4 是以对象为中心的设计,如果太强调系统的功能,设计出的系统将更多的面向功能,而不是面向对象。围绕着功能分解组织的系统,容易由难于、且很花钱去维护的耦合过紧的模块组成,这样就去掉了面向对象方法的一个主要优点。很小的变化,可能意味着很大的需时几个月的系统重设计和重编程。即使有这些缺点,我们也不是不鼓励将 IDEF0 用作帮助面向对象系统设计者的有用的工具,而是鼓励谨慎地使用。用以对象为

中心的观点开发的 IDEF0 模型，为系统需要的初始的类和例程提供了有价值的看法。最后，IDEF0 模型是沿着功能分解的线索开发的，只分解成两层或三层的 IDEF0 模型最有用，具有更多细节的模型，容易强调功能分解，多于面向对象分解。

用于信息建模的 IDEF1 方法和用于数据建模的 IDEF1X 方法，为初始类识别和类间关系方面提供了输入。但是，我们再一次强调，使用它们必须谨慎。这两个方法性质上都是关系型的，因此，不会全面提供系统的对象分解。对这两种方法来说，完全的非常详细的模型是不需要的。实际上，它们对面向对象系统的开发是有害的。

也许对面向对象的设计者来说，最有用的 IDEF 方法是 IDEF3。IDEF3 为设计者提供了开发系统过程流描述，和对象状态转换描述的方法。从以对象为中心的观点开发和使用这些描述，对类定义和初始行为识别很有帮助。

通常，如果一个实现了的系统要被使用，用户和系统之间的关联必须是可接受的。面向对象的系统设计者必须考虑人机界面。从软件开发的角度讲，IDFE3 在设计这些使用场景时非常有用。如果选择了以对象为中心的观点，系统的行为可以划分成对象必须展示的共同功能。系统中的所有对象，将有某些作用在它们上面的共同操作。比如，考虑一个系统对象的删除。以对象为中心的 IDEF3，将允许对所有系统对象共同的删除功能所作的描述。最后，我们看一看 IDEF3 对象状态转换网络，这些将帮助设计者决定方法、事务设计和系统对象中，状态变化的前提和后续条件。

这些方法对面向对象设计的有用程度，可以从大到小排序如下：IDEF3，IDEF0，最后，远远落在后面的是 IDEF1 和 IDEF1X。下述几点是那些使用这些方法的人应该注意的：

(1) 模型或描述应从以对象为中心的观点来开发。

(2) 对中小型系统来说，如果这些模型可以得到，就使用它们，否则用不着开发所有类型的模型，除非这是大型系统。

(3) 完全的或全面开发了的模型和描述，通常是不需要的，甚至会产生误导。

5.2.2 开发类层次

类层次的开发，包括详细说明类、特征、继承网格和类-定常约束。设计者首先把需要的系统行为划分为直观的类，然后再系统地细化它们。

开发类层次，包括识别类、特征和类间继承关系。这通常由下述步骤循环完成：

(1) 根据相似、行为和状态把类划分成更专门的集合。

(2) 在现有继承图的基础上，把类分类或用特征和类-定常约束具体说明它们。

(3) 把这些类组装成现有的继承图（和类型图）。

(4) 重新安排继承网格以简化它们的结构，可能导致重新划分。

5.2.3 开发方法分类

设计者一开始就把要求的系统行为划分成直观的方法集。对于每一个初始方法集来说，有必要开发客户图。方法分类图的开发，要求识别对方法和过程的约束，这些是在完成

的系统中发生不同的动作时将出现的。这由下述步骤循环完成：

（1）把方法集划分成更专门的集合。

（2）在现有分类法的基础上，把方法集分类，或通过新的合同详细说明它们。

（3）把这些方法集和合同组装成现有的方法分类图。

（4）重新安排方法分类图中的方法集和合同，以细化它们的结构，可能导致重新划分。

对于没有参加过这个过程的人来说，这些动作好像以很不受控制的方式发生的。在设计小型系统时，对清晰定义的过程步骤的认定，可以被忽略。在选择和细化初始方法时，设计者依赖于系统的初始约束（即需求），随着设计过程的继续，不断发展的方法集的说明，变成了对设计的附加约束。

5.2.4 开发类分解结构（类型图）

类型图用来说明，如何把类通过属性（返回值的）和类间的类型链组成群集。比如说，类型图可以用来描述一辆汽车有一个引擎和四个轮子，而引擎有八个活塞，分别连着八个曲轴。类型图可以使用来自信息和数据模型，像 IDEF1 和 IDEF1X 的关于联接类型的信息，作为输入。有关类型链的大部分信息，将出现在方法分类的合同中，和与继承图相联系的 CIDS 的约束中。

5.2.5 开发协议（协议图）

协议图详细说明，一个例程-类对（方法集）必须有有效输入参数类型和返回值类型。这些信息可以来自参照方法集，而附属于一个类的类-定常约束，和附属于通过例程-类对选择的方法集的 CDS。比如说，对于“画”这个例程-类对（方法集），输入参数必须是一个窗口对象和一个图形对象，并且该窗口对象必须为将要建立的图像打开。这个约束可以正式定义如下：

图形（X）^ 窗口（Y）^ 打开（Y）^ 画（X Y）→建立图象（Z X）

由此可以很容易地推断，“画”例程-类对（方法集），应该接受一个图形对象和一个窗口作为参数。

5.2.6 开发算法分解（客户图）

客户图说明了方法集的算法或函数分解。比如说，“重画”例程-类对（方法集），调用“擦除”例程，然后再调用“画”例程。注意有必要详细说明客户方的例程-类对，这样，间接地说明了其方法集，但是，在提供者方，只有例程需要详细说明。如何抓住机会使用算法分解，来自对方法集的观察，内部控制逻辑，在另外的方法分类的其他方法集说明中被重复。在“重画”这个例子中，我们可以注意到，“擦除”方法集的控制流和“画”方法集的控制流是相似的。我们实际上能用调用“画”来说明“擦除”。这意味着“重画”和“擦除”是类属方法集，即，对每一个需要“重画”函数的新类，只要说明一个“画”方法即可。

5.3 IDEF4 设计进展过程

任何 IDEF4 设计的开发，需要建立：(1)方法分类图，(2)继承图，(3)类型图，(4)协议图，(5)客户图，(6)CIDS，(7)CDS，以及方法分类图中的一个方法集相联系的 CDS，实际上是那个方法集的特征函数。换句话说，方法在被说明时必须被定义。CIDS 详细说明对每一个类的所有实例共同的行为；这样，在类设计早期，说明这些约束是很重要的。每个成分都对系统的最终设计作出贡献。对于设计过程，正如任何递归(循环)的过程那样，过程终止的准则是很重要的。不可能给出设计活动完成的精确的准则，只是当设计接近完成时，设计改变的比率下降。这是因为 CDS 和类-定常约束中说明的每一个新约束，对设计提出了越来越严格的要求。

IDEF4 设计的开发，是详细说明一个面向对象的程序或系统的结构和行为的过程。在 IDEF4 设计的开发中，下述 4 个一般步骤被循环地使用：

(1) 划分：把不断发展的需求划分成较小的、更容易管理的片。

(2) 分类/详细说明：在现有定义的基础上分类或详细说明新的设计对象。

(3) 组装：将设计对象合并入现有的结构。

(4) 重新安排：重新安排现有的结构来利用好新引入的设计对象，这可能导致对设计划分的调整，从而引起设计循环中的另一次迭代。

术语“循环使用”，意味着同样的说明过程，通常用于五个图类型的划分的每个成分，也用于整个设计开发活动。这个循环一直持续到可以清晰地建立所得到的元素的原型分类。

5.3.1 划分

在系统需求分析中，设计者把系统划分成直观的类和方法集。在早期阶段，检查类属算法分解是否可能是非常有用的，以便发现重复使用设计结构的机会。必须强调，面向对象或数据分解，在时间稳定性上比面向功能的分解好，因此，相对于算法实现中可能牵涉到的数据类型，所做的任何函数分解都是类属的，这一点很重要。“重画”的类属函数分解如图 4.5.4 所示。

设计的类属函数（或算法）划分，为减少系统实施时代码的重复量，提供了工具。另外，它也为识别可以被重复使用的设计部分，或用于识别可能的前一个系统的代码可以用来实施现在的设计的区段，提供了方法。比如说，在图 4.5.4 中，“重画”是最初为系统识别的行为类型，在该例示中，“重画”方法集和另外两个方法集之间的链，显示了方法集“擦除”、“画”和“重画”之间的成员关系。基本上，设计者是声明，必须书写“擦除”和“画”方法集，这样它们就可以被“重画”方法集使用。其目的是减少“重画”方法所要求的编码（即，使“重画”通用化），并提供“擦除”和“画”方法集重复使用的可能性。如果“重画”的类属能力没有被认识到，就必须为几乎每一个使用它的类，设计一个“重画”方法集。在这个例示中，实际上是，设计者声明了“重画”具有类属函数，并且它将使用“擦除”和“画”方法集。

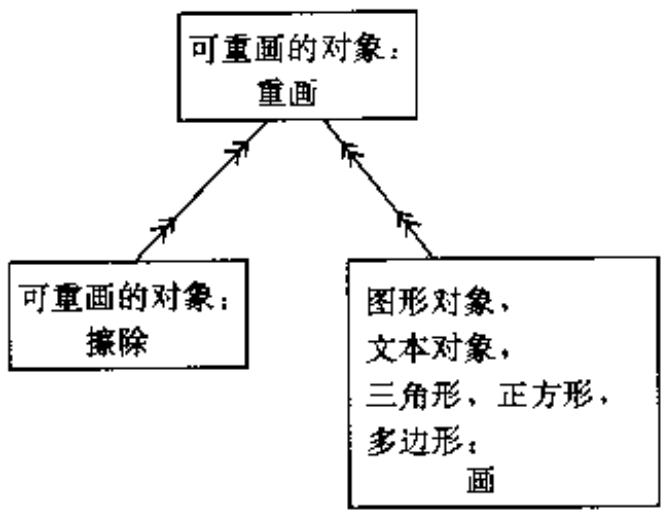


图 4.5.4 “重画”的客户图

在混合图的开发和当前识别的方法集的划分中，扩展了附属于每个方法集的 CDS 的说明。比如说，在对图 4.5.4 的讨论中，对“重画”方法集的约束，就是它将通过“擦除”和“画”方法集，提供部分功能。进一步地，“擦除”和“画”的 CDS 也应加上约束，它们可以被“重画”方法集访问和使用。

5.3.2 分类/详细说明

在分类/详细说明阶段，要将划分阶段识别了的方法集和类，与现有的方法集和类相匹配，以便对它们分类。如果一个方法集或类不能被分类，那就必须设计一个详细说明。类的详细说明，包括定义类-定常约束、协议和特征，方法的详细说明，包括说明 CDS 和客户图。

因为这个生成活动被循环进行，所以类和方法集的说明，将变得越来越完全。在每个周期中，类和方法的说明，可以被看作下一个设计周期中的附加需求。

5.3.3 组装

被分类或详细说明的类和方法集，必须被组装成 IDEF4 使用的有组织的结构。新分类或说明的类，必须被组装到新的或现有的类-继承网格（继承图）和类-组成网格（类型图）中去。新分类或说明的方法集，必须被组装到新的或现有的方法分类图和客户图中去。可以使用混合的客户/方法分类图，识别共同的行为或类属函数。在大型系统的设计中，组装活动很可能要在负责设计不同方面的人的联合会议上进行。

5.3.4 重新安排

在重新安排设计阶段，设计者将寻找结构、相关的方法、或类中的相似性，以识别重复使用设计结构的机会。重新安排，可能迫使类的、方法的和与它们相关的组织结构的说明发生改变。比如说，设计者可以把几个方法集结合起来，并检查和每一个方法集相联系的 CDS，以找出可能的改进。这些改进，以在图中上下移动约束的形式进行，既不改变超结构或子结构的行为，也不引起和任何不断发展的系统需求间的冲突。

5.4 IDEF4 设计文档的组织

一个完成了的 IDEF4 模型，将包括一个类子模型和一个方法子模型，这些子模型将被分成五种类型的图，两种类型的数据单，和两种类型的分配映射说明。依据所要开发系统的大小，可能会有几百页的文档。如果没有对图的某种系统化的分类，根本不可能驾驭这些文档。在这一节中，我们将描述为 IDEF4 设计文档所推荐的组织安排。

面向对象的范例，本身并不提供，无论是设计的、还是相关的程序源代码的顺序巡视。这样，对设计文档的组织，就必须能对不同的设计成分，提供非顺序访问。巡视过程应具有某些超文本文档的特性，即，从文档的任何一个成分，都应可能直接到达任何一个其他成分。

面向对象的范例的基本组织结构是类，因此，IDEF4 文档以类和例程名为中心。文档被分为两部分，类子模型和方法子模型。简而言之，文档用下述框架来组织：

1 类子模型

- CIDS：CIDS 是所提供的第一个成分，它们按类名的拼音字母顺序排列。
- 继承图：类继承图按建立的顺序编号（如，I1，I2，…，In），每个继承图应有一个唯一的标题。
- 继承图中的分配映射：每张继承图，有零或一张相关的显示分配映射的继承图。
- 类型图：在类子模型中，类型图放在继承图的后面，并按建立的顺序编号（如，T1，T2，…，Tn），每张类型图都有一个标题。
- 例示图：在每张类型图后面将有零、一或多张相关的例示图。
- 协议图：协议图放在类子模型中类型图之后，按例程名（中心例程）后跟与之相联系的类名（例程-类对）的拼音字母顺序排列。

2 方法子模型

- CDS：CDS 以例程名，后跟与之相联系的类名（例程-类对）的拼音字母顺序排列。方法集的名字将出现在数据单上例程-类对名的后面。
- 方法分类图：方法分类图按以之命名的系统例程的拼音字母顺序排列。
- 方法分类图中的分配映射：显示分配映射的相关的方法分类图，将跟在每张方法分类图后面。
- 客户图：客户图按例程名（中心特征），后跟与之相联系的类名（例程-类对）的拼音字母顺序排列。

用类名、例程名或例程和类名结合作关键词，可以直接找到任何设计成分，除了继承图和类型图，它们必须通过 CIDS 查找。如图 4.5.5 所示，IDEF4 设计文档，围绕着按类名拼音字母顺序排列的 CIDS 组织。每个 CIDS，包含类出现在其中的继承图和类型图的数字标识（ID）。继承图和类型图，以建立的顺序，用数字安排。在文档中，每张分配映射继承图，放在它扩充的图的后面。设计中的其他成分，按与之相联系的例程或例程-类对的拼音字母顺序组织。因为 CIDS 包含类中所有直接出现特征的列表，所以设计中任何一个和特定的类相关的成分，都可以很容易地直接（在继承图和类型图的情况下）或间接（通过

类例程）找到。

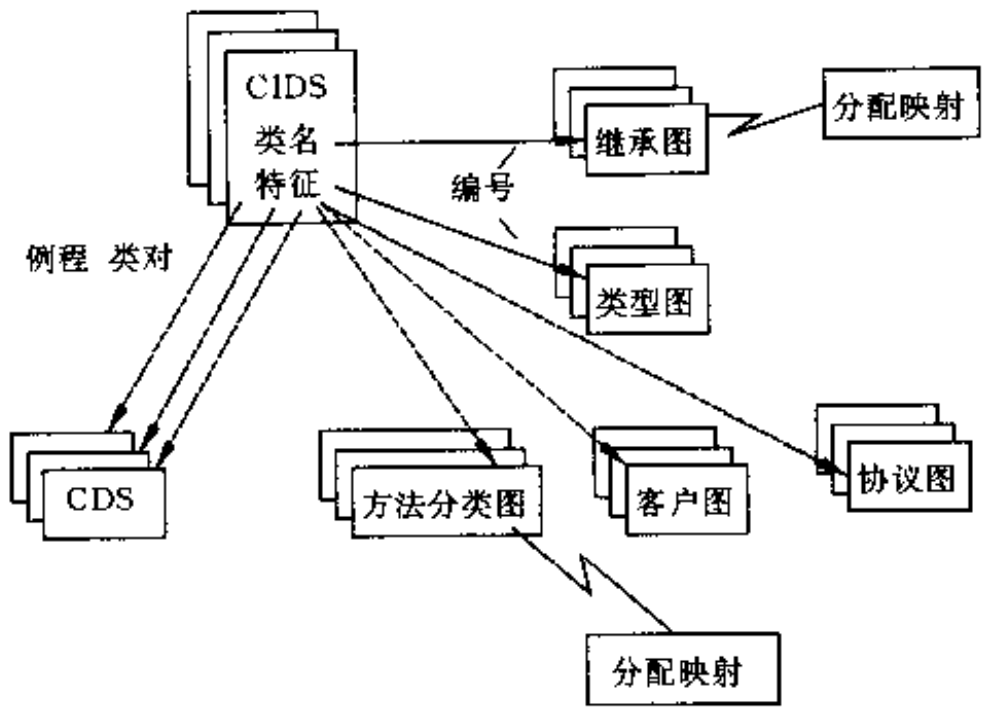


图 4.5.5 IDEF4 文档组织

在实践中，任何设计检查的可能开始点，将是继承图。出现在这些继承图中的类和特征名，提供了除类型图外，和所有其他设计成分的直接联系，类型图则可以通过 CIDS 找到。

这种组织方法允许随机阅读设计，而不用考虑开始点。比如说，在图 4.5.6 中，开始点是方法分类图。方法分类图按相关的例程名的拼音字母顺序排列。给定文档中一个特定

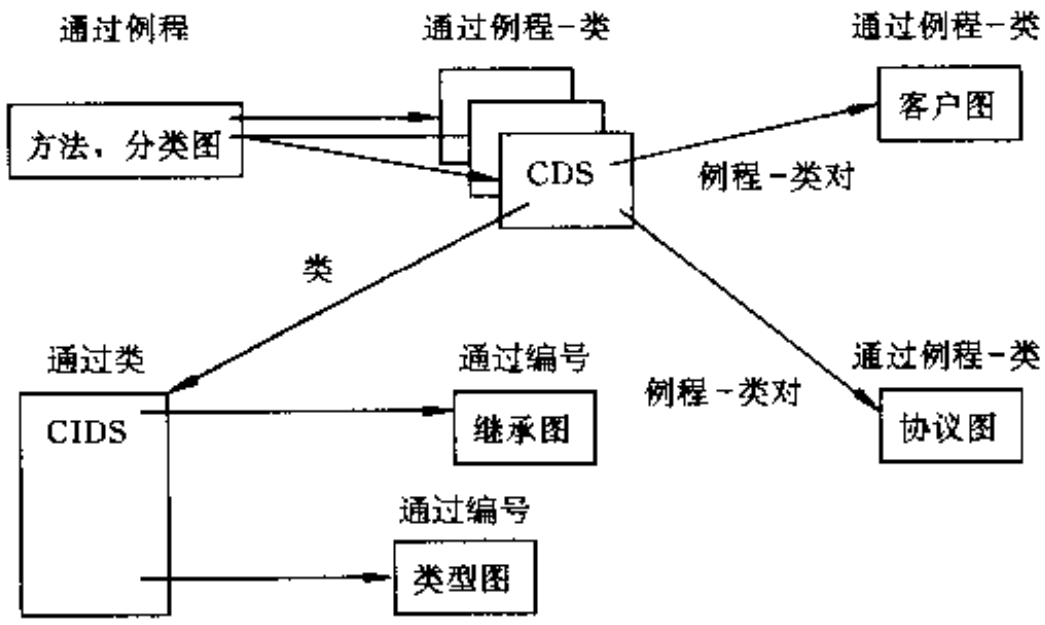


图 4.5.6 从方法分类图巡视文档

的方法分类图，例程名就知道了（方法分类图的名字）。和那个例程相联系的 CDS、客户图和协议图，在文档中按先是例程名后是类名的拼音字母顺序安排。用例程名，人们可以很容易地集中所有和特定例程有关的设计成分。这些文件中的任何一个，将为感兴趣的类提供指向 CIDS 的指针（类名），而 CIDS 将提供对类出现在其中的继承图和类型图的访问。这样，就可能在设计中很容易地找到有关成分。

第 6 章　IDEF4 设计开发中的若干问题

用这个方法建立的第一个设计将是最困难的。设计者和实现该设计的人，将会遇到许多困难，这些困难会随着经验的增加而减少。本章希望能回答一些用户在建立和理解早期面向对象设计中会遇到的问题。在这最后一章中，我们将重申 IDEF4 面向对象过程的一些基本特征，并重述前面已说过的有关建立面向对象系统的 IDEF4 设计的一些提法。

6.1　细致的和粗略的方法和类

在设计一个相对小型的系统时，初始类可能出现在较低的抽象水平上，并且方法集可能显得非常细致。初始方法集中的细致的细节，也可能是由下述因素造成的：(1)设计者在这一领域中进行系统开发的技能，(2)系统需求的清晰性和简单性。

另一个导致在设计的早期阶段产生细致的方法和类的因素，在于对问题的分析。有要实现的系统的功能模型 (IDEF0)和信息模型 (IDEF1)的设计者，会发现他们的初始类和方法集的列表，将比那些没有这些模型的人完善得多，并包含更多的细节。

6.2　方法分类图和例程

方法分类图，展示了设计中方法集合同间的关系。分类中的每个方法集，都和一个例程-类对相配对。作为一种规则，和分类中的每个方法集配对的例程是相同的，因此，惯例是用其描述的系统例程的名字为方法分类图命名的（如“排序”方法分类图，或“抓住”方法分类图）。

方法分类图不必遵循类层次。比如说，当系统中任何类的一个特征执行一个例程时，特征的名字通常就是例程的名字。但是，情况并非必须如此。完全可能，设计者说明一个类中的两个特征，去进行同一个例程，这是完全可以接受的。这种情形的一个例子，就是前面描述过的一个系统，该系统中同一个类定义了两个不同的“排序”例程。当这种情况发生时，“排序”的基本例程具有特征名“排序”，而次要的例程，就给一个其他的名字。分配继承图，显示了把特征分配给不同的方法集，但在同一个方法分类“排序”中。

6.3　例程和例程-类对

IDEF4 的读者和使用者，一开始对术语例程、类属例程和例程-类对会有些混淆。例程指启动计算的 IDEF4 特征；类属例程指希望系统或类层次提供的类属操作或函数。比如

说，声明“希望系统提供文件按名字和编号排序，数字列表排序和雇员名列表排序”，这就指出了，系统必须提供不同类型对象的排序。因此，我们说系统将有名为“排序”的类属例程。当从上下文看意思很清楚时，例程可以用来代替类属例程。例程-类对是由类属例程名和类名组成的顺序对。例程-类对可用作索引。如果“排序”在类“表”中定义或重定义，这个例程-类对就被称为“排序-表”例程-类对。

在 IDEF4 中，类属例程名被用来命名方法分类图，它是特定类型的系统行为的图。例程-类对涉及通常是以类属例程名命名的方法分类图的一部分的方法集（方法集合同）。例如，“排序-表”例程-类对，和一个叫同样名字的方法集合同相联系。该方法集合同将用“排序”方法分类图的一个节点来表示。

6.4 一个返回值的多个返回类型

通常，设计者希望显示，成功执行一个函数，返回一个特定的类型；而不成功执行一个函数，将返回另一个类型。比如说，设计者可能希望指出，函数“用颜色填充对象”返回对象的颜色。但是，如果函数无法执行时，将返回字符串，像“没有可能的颜色”。不论是 IDEF4 的协议图还是类型图，都不允许显示这类信息。这类信息必须，或者在和“用颜色填充对象”相联系的方法集的 CDS 中说明，或者作为定义特征的类-定常量的需求部分。

6.5 面向对象过程的特征

在设计开发中，设计者从一个粗略的系统设计开始，然后扩展、细化这个设计，直到设计完成。IDEF4 设计过程，揭示了几个值得一提的面向对象过程的特征：

1 需求说明：清晰地说明系统需求，使得初始类和方法更具体，从而缩短开发时间。这些系统需求将随着设计过程的继续而扩展、细化。

2 需求的理解：在功能、信息、过程流和 IDEF4 模型间，似乎存在一种确定的关系，如图 4.6.1 所示，比如说：

- 功能模型中的活动和过程描述中的行为单元（UOB）为方法集提供了支持的根据。
- 过程流描述的细化说明为 CDS 和 CIDS 提供了约束。
- 过程描述的对象状态转换网络（OSTN）提供了特征、类和类-定常约束的根据。
- 功能模型中的概念（输入、输出、控制和机制）提供了特征的根据。
- 信息模型通过实体类为对象类提供根据，通过连接类为合成提供根据，并通过属性类为特征提供根据。

3 类和方法的列表：初始设计阶段，应产生显然来自需求说明的类和方法的列表。在初始设计阶段，不应有意识地试图确定这类事情，像已识别的类的属性，或方法可执行的算法。

4 对系统类型的熟悉程度：如果设计过程中，一个或更多的参与者熟悉正在设计的系统的类型，在初始设计阶段，设计将有可能产生更多类子结构和更细致的方法集，而有

可能用较少的开发时间。如果系统的设计者，不熟悉目标系统领域中的系统的开发，最显著的差别将是较长的设计开发时间。

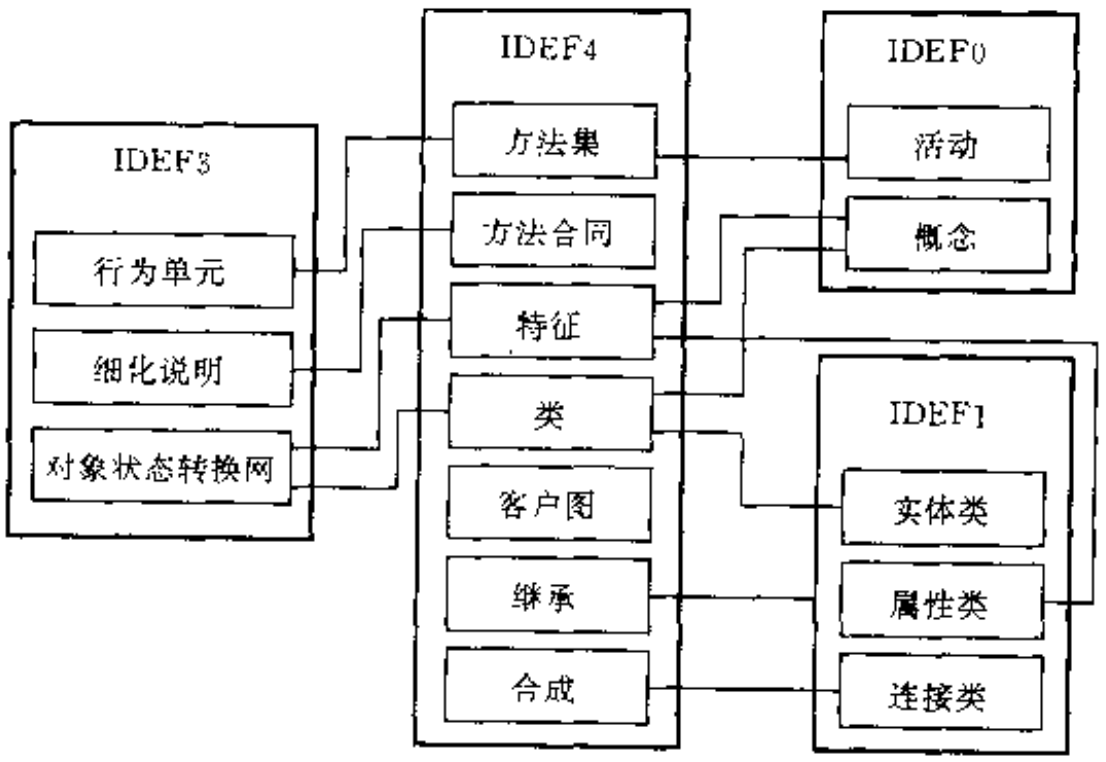


图 4.6.1　IDEF4 与 IDEF0，IDEF1，IDEF3 的关系

附录 3　IDEF4 词汇表

类(class)：	在面向对象的开发中，属性和例程被认为是和提供基本封装机制的类一致的。在面向对象的世界中，术语“类”，在概念上是指某些数据类型和它们的相关的操作，这些操作，描述了一类处于运行时间的计算机程序对象的特征和行为。
类子模型(class submodel)：	IDEF4 类子模型包括继承图、协议图和类型图。
客户图(client diagram)：	阐述方法集中“客户”与“供应商”的方法集的算法分解。
合同(contract)：	方法集用它履行的合同来说明。合同是对方法集中的方法所希望具有的效果的声明。
合同数据单(contract data sheet)：	说明方法集中的方法必须满足的合同。
设计方法(design method)：	运用某种设计哲理(设计方法论)的一种启发式思考指南。
设计方法论(design methodology)：	设计哲理(参见“设计方法”)。
分配映射(dispatch mapping)：	在 IDEF4 中，分配映射连接类子模型和方法子模型。它用继承图和方法分类图中的注释表达。
函数方法(functional method)：	返回一个值，并且没有副作用。
函数方法集(functional method set)：	说明那些返回一个值并且没有副作用的方法。
继承图(inheritance diagram)：	阐述类之间的子类和超类关系。
方法(method)：	满足方法集中说明的约束的实现，用类-类属函数对唯一标识。
方法集(method set)：	用合同数据单中定义的，称为方法合同的特征函数定义的集合。IDEF4 方法集用 IDEF4 类-例程对唯一标识。
方法子模型(method submodel)：	IDEF4 方法子模型着眼于方法的结构，包括方法分类图和客户图。

方法分类图(method taxonomy diagram)：　通过行为相似性和类特征，与方法类型之间的连接，对方法类型进行分类。

过程方法(procedural method)：　不返回值，并且有副作用。

过程方法集(procedural method set)：　说明不返回值并且有副作用的方法。

协议图(protocol diagram)：　说明方法要求的协议。

例程(routine)：　计算引出的特征，可能返回值，并有副作用。

类型图(type diagram)：　阐述类之间的复合关系。

附录4 缩略语

CDS (contract data sheets) 合同数据单

CIDS (class-invariant data sheets) 类-定常数据单

CLOS (common lisp object system) 公共表处理对象系统

FORTRAN (formula translator) (一种计算机语言)

ID (identifier) 标识符

IICE (information integration for concurrent engineering) 并行工程的信息集成

KBSI (Knowledge Based System, Ync.) (公司名)

LISP (list processing) 表处理(一种计算机语言)

ODBMS (object data base management system) 对象数据库管理系统

OOD (object-oriented design) 面向对象的设计

OOPL (object-oriented programming language) 面向对象的编程语言

OSTN (object-state transition network) 对象状态转换网络

UOB (unit of behavior) 行为单元

第五篇

本体论获取方法

第1章　基本概念介绍

1.1　什么是“本体论”

“本体论”(ontology)本来是一个哲学名词。在词典中的定义是“指哲学中研究世界的本原或本性的问题的部分”,或者是指“玄学”(metaphysics)中涉及存在的性质和关系的一个分支。它与“认识论”(epistemology)不一样,认识论是关于知识和理解的问题,而本体论则是对概念的本质和关系的描述。

在工程研究中,从知识共享的角度来说,本体论这个名词是作为一种概念化的说明,是对客观存在的概念和关系的描述。它是通用意义上的“概念定义集”,是关于“种类”(kind)和“关系”的词汇表。这种词汇表,是在各种事务代理人之间交换意见时所用到的共同语言。表面看来,一个“本体论”与一个数据字典没有太大差别,也是许多名词的集合。然而,数据字典只不过是一批用自然语言描述的名词,将其定义组合在一起的一个纲要;而本体论的文法和公理,则是要用精确的形式语言、非常精确的句法和明确定义的语义来阐述的。这样,本体论就要比一个典型的数据字典在内容上要严格得多和精确得多。本体论还试图做得更完整,它要使得领域中概念和对象之间的关系,以及在领域对象上所加的约束更明显而不是隐含的,从而大大减小了对领域中逻辑关系可能造成的误解。相对来说,数据字典通常则只是对所讨论的术语的直观的理解,或者对其所代表的概念或对象之间的逻辑连接的描述。这在一些小的有限领域中还是可以的;但当所研究的信息系统扩张到组织上、地理上、企业范围都非常大的领域时,问题就产生了。这不仅是因为在传统方而,不同领域中不同的人,可能对同一个术语有很不一样的理解,更重要的是,有些重要的问题不能用自然语言定义来覆盖。而利用这些领域中的本体论,用一种规范语言的组织化就可解决这个问题。另外,将本体论信息用形式语言表达的训练,又加强了抽取信息所必需的技巧。特别是从特殊的对象抽象到它们所属的种类,从特殊的联接抽象到这些实例所代表的关系,从特殊的行为抽象到把不同种类的实例在该领域内逻辑上规约在一起的约束。这些都是它比数据字典更严格的方而。

1.2　为什么要提出本体论

本体论的提出,主要对大规模的制造或工程项目中,实行并行工程的做法非常重要。通常一个大项目,都有来自不同组织不同专业或背景的各种工作小组,他们之间需要一种规范化的共同语言,作为保证项目成功的基本条件。例如,制造本体论,它包括了一个给定的零件是如何制造的约束条件,就能帮助设计师在进行复杂产品设计时,对可制造性有较

深的了解；又如工程本体论，它包括了特定形状和装卡时给定零件的应力特点等约束条件，就能帮助工艺师制定合适的制造工艺过程。一个公共可读取的有关的本体论集合，就促使整个企业内来自各种资源的信息能更有效地被大家共享。所以，获取本体论的动机的第一条，就是术语的“标准化”。使各种人员之间交换信息时，有更多规范化的术语作基础，而不会引起不必要的误解。

获取本体论的另一个强烈的动机是“可再用性”(reusability)。工程和制造领域中，往往会对已有的信息，花费冗余的劳动去重复创造。譬如软件编程中的很多子程序，往往由不同的程序员，一再重复地编写相同或类似的子程序。如果把共享的程序库做得好些，让大家及早了解库的内容，可以直接调用，就能节省很多不必要的重复开发。类似的问题在制造领域同样存在；有很多公共的特征(feature)是可以共享的，领域越接近，共享的特征越多。如能把这种公共信息收集到“本体论库”中去，使这些信息可以被再用，也可被修改以适合当前的需要，就一定能大量节省重复劳动的精力。

本体论模型的开发带来的好处，可以有下列两个方面：

1　本体论分析的过程，是一个揭示对象内在关系、加深对领域理解的过程，所以本体论分析可用于(1)识别问题(诊断)；(2)识别问题原因(因果分析)；(3)识别其他方案(设计)；(4)统一意见和团队建立；(5)知识共享和再用。

2　本体论开发得到的结果可用于：(1)信息系统开发：本体论为开发更加智能化的和集成的信息系统提供了一张蓝图；(2)系统开发：本体论可用作规划、协调和控制复杂产品和过程开发活动的参考模型；(3)经营过程重构：本体论提供了识别组织重构的焦点的线索，并建议了为进行重构可能最有潜力的转换途径。

总之，在信息系统、接口和面向对象设计和编程等方面，本体论都是一个很好的工具。所以，IDEF 家族就开发了一个本体论获取方法 IDEF5。

1.3　IDEF5 与其他 IDEF 方法之间的关系

复杂系统体系结构是一种多种特征的综合表示法，所以 IDEF 家族发展了各种不同的方法来描述不同的特征。有的专家认为，可以用一种统一的描述方法来建立一个复杂系统(如 CIMS)的体系结构(国内也已有这一类的文章)。实际上，这是一种不切实际的空想。有一位学者(Zachman)说，假如有人提出这种所谓“体系结构”的话，那么“一个是错的，另一个是不正确的”。由于问题(或称系统)本身的复杂性，一种表达形式只可能用于一种特征。譬如，IDEF0 是一种很有用的工具，用以描述经营活动之间的相互关系，互相之间传递的事物和信息，互相之间实现的控制和支持等关系。但如要用它来设计一个关系数据库，那就很不方便了。关系数据库设计，需要建立 IDEF1X 模型。又如，IDEF0 的好处是能够以递阶结构来描述规模很大、关系很复杂的系统；为此就不可能加上时间特性。因为一个工业企业，其底层的生产活动可能以分、秒作为计量单位，而中层的生产计划以月为单位，上层的战略规划则以年为计量单位，很难在一个模型中用一种统一的尺度来表示；所以 IDEF0 只描述活动间的关系，不包含时间因素。而要表达明确的时间间隔、相对的时间关系的过程模型，就要用 IDEF3 方法。然而要用此建模方法建立一个整个企业的

IDEF3 模型，几乎就不可能了，只能对分成不同领域、有限范围的过程，进行建模分析。而对于 IDEF0 很容易表达的 ICOM(输入、控制、输出、机制)关系，如果都要在 IDEF3 中表示，那又成了一种难事。

IDEF5 与 IDEF1/IDEF1X 的关系，有点类似于 IDEF3 与 IDEF0 关系。IDEF5 的“种类”(kind)(与 IDEF4 中的“类”(class)不同，但我们找不出更确切的译法，暂且如此区分。)与 IDEF1/IDEF1X 的“实体”(entity)很相似，然而 IDEF1/IDEF1X 只能表示比较肤浅的逻辑关系的信息，本体论则获得了深层次的事物特性。

1.4 本体论的中心概念

1.4.1 种类

一般来说，“种类”(kind)是指对那些具有共同性质的物体给出的一种范畴的划分，或者说所有这个种类中的成员都具有(也只有这个种类中的成员才能具有)这一组性质。更确切地说，对每一个种类 K，就会有一组性质 N，其中每一条性质都是必需的，而所有这些性质组合起来，就成了可以成为 K 的成员的充分条件；即，“X 是 K 的成员，当且仅当 X 具有 N 中的每一条性质”。IDEF5 是用以获取本体论的，要抓事物的本质，当然就要以种类的划分及其基本性质作为研究的出发点。

事物的性质，首先可以区分为“本质的”(essential)和“附属的”(accidental)(或称非本质的)两种。例如“金子”，它具有一种特殊的原子结构，故凡具有这种性质的东西都是金子，这是本质的；另一方面，这种东西一定会有一定的“形状”，然而，不管是方的、圆的、奇形怪状的，都可以是金子，所以，“形状”这个性质是附属的、非本质的。

然而，IDEF5 要用来获取企业本体论，问题会复杂得多，因此要把“种类”的定义做些灵活性的修正。把那一组用以确定种类 K 中成员的性质，称之为“限定性的性质”(defining properties)。譬如，四边形“必须有四条边”。这种性质的阐述，实际上不是用种类本身来说明，而是用种类的实例来例证的。另外，“限定性的性质”不一定是“本质的”，也可以是“附属的”。下表给出几个不同的种类所具有的不同性质。

	限定性的	非限定性的
本质的	种类：四边形 性质：具有四条边	种类：圆 性质：没有内角
附属的	种类：刀具 性质：有金刚石的嵌入刀头	种类：需求文件 性质：长度为 10 页

至此，IDEF5 将“种类”的定义从前述“每一条性质都是必需的，而所有这些性质组合起来就成了可以成为 K 的充分条件”，改成了仅仅要求“对种类 K 中的每个实例 X，X 要至少有 K 的多项限定性的性质中的一个”，就可以了。

“种类”和其他数据模型中提到的“类型”和“类”，都是对个体集合的分类，都是可以有

多个示例的。但是，种类和类型的实例是可以随时间改变的，但种类本身则不变。例如，“雇员”种类，并不因为一个企业雇员数目的增减、具体人员的变动而改变这个种类本身。而类则有时依赖于一些可记录的个体的集合，略有差别。

1.4.2 性质和属性

本体论中要明确区分性质和属性。属性最好被看作一种函数，它一定要被赋予一个值。例如，属性“……的颜色”（简称“颜色”），就把每一个对象映射到它的颜色；属性“……的年龄”（简称“年龄”），就要把每个雇员映射到他（或她）的年龄。而性质则是直观的，事物的特征，所有个体所共同具有的抽象的特征。

事物总要求显示某种属性值。事物的“颜色”（属性）是红的（属性值），则其有性质“是红色的”。雇员的“年龄”（属性）是 40（属性值），他就有一条性质“年龄是 40 岁”。

经常有这种情况，一个种类的限定性性质，相对于其特定的属性值来说，会是无限多的。例如，“有可识别的序列号”，可以是“数控机床”种类的一个限定性性质，这时不管实际序列号是多少，但作为“……的序列号”这一属性，建模者可以赋予无限多个属性值。

在建立本体论的实践中有时分不清性质和属性，所以 IDEF5 中有时用一个中性名词“特征”，包容了这两个词。

1.4.3 关系

除了各个个体的性质和属性，本体论中当然要考虑个体之间的连接或关联，称之为“关系”。例如，“工作在……”关系，就是一个雇员与其所工作的部门之间的关系。正如性质一样，“关系”也是可以多重示例的，并且是强制性的。

在本体论中，一般说“关系”是存在于二者之间的，但并不排斥存在于三个以上个体之间的关系。

1.4.4 二阶性质和关系

上面说到了“性质”和“个体”。显然，这是不同逻辑类型的东西。性质是不同个体间共有的抽象的通用的特征。同样，关系也是不同的成对的（或三个以上）个体之间共有的通用的关联。因此，性质和关系是从个体的特征中抽象出来的，就被认为是一种更高的（更抽象的）逻辑类型。

如果把个体看作是“一阶对象”，一阶对象的性质和关系就叫做一阶性质和关系。然而存在于个体之间的性质和关系，本身也是一种可识别的（虽然是抽象的）对象，因为它们比普通一阶对象在抽象程度上高了一级，被看作是更高的逻辑类型，就称之为“二阶对象”。而一阶性质和关系作为一种对象，它们也有自己的性质（就不是用于个体的），例如，性质“具有至少一个实例”。这种性质因是用于二阶对象的故称为“二阶性质”。另外，二阶对象相互之间存在着关系，例如，在两个种类之间存在的“具有比……更多的实例”关系。又如，存在于一个给定种类和包含着它的一个更通用的种类之间的“子种类”（subkind）关系；人类是哺乳类的子种类，数控机床是机器的子种类。有些二阶关系把个体作为其变元，如“是……的实例”关系，就存在于一个个体 a 与一个种类 K 之间（当 a 是 K 的一个实例时）。这

种不同逻辑类型对象之间存在的混合类型关系，也称为是二阶的。因此，二阶关系就是至少包含一个一阶性质或关系作为其变元的一种关系。

这里对“子种类”关系还要说明一下。这个关系是这样的：如果 K 是 K′的一个子种类，则 K 的每个实例都是K′的实例。要注意的是，一般说，反过来是不成立的；不能说：如果种类 K 的每一个实例都是另一个种类 K′，则 K 就是 K′的子种类。因为子种类关系必须是：单说每个 K 是 K′的实例是不够的，还有不可能有一个 K 不在 K′之中。譬如每个“美国总统”都是“男人”，但是“美国总统”不是“男人”的子种类，因为可能有一天会出现一位女总统。然而“美国总统”是“美国公民”的子种类，则是成立的。

1.4.5 将种类引入本体论的方法

要将种类引入本体论，可有两种方法，即定义或者规定一些条款。前一种方法是当在所研究的领域中，根据已存在的对象、性质和关系，可以对一件事物构成种类 K 的一个实例，提出充分必要条件，就可以进行定义。这种情况下，整个 K 的逻辑特性，是根据本体论中以前已经给出的元素而给定的。譬如，种类“质数”(素数)，就可以从种类“数”中进行定义，即数字 1 以及根据关系“不可以被……除尽”，就定义了整个“质数”种类。而后一种情况，则是假设已经存在一个种类K，而只提供了其特性的部分定义，则就要作一些规定。譬如图书馆中的“书籍”种类，就不能根据其性质、关系等做出完全的定义，它不能是一堆书页的集合；它可以根据这个领域中其他的种类来进行部分定义或公理化，譬如根据种类“作者”、“出版商”等，规定某些条件下的这类事物，才算作“书籍”种类的实例。至于性质和关系，一般来说也可以用定义和规定引入本体论。

1.4.6 部分、整体和复杂种类

从前面所考虑的个体的例子中，似乎 IDEF5 中的个体都是逻辑上比较简单的。然而实际上有很多种类的个体，其本身就是由许多各种“种类”的其他对象所组成的复杂种类。一般来说，这些个体所以被看作是简单的，只是因为在那个研究场合下，不必要考虑其合成特性。而同一个事物，在另一些场合下，某个对象种类的合成特性要突出地考虑时，这个对象种类，就要把其他种类的对象看作其“部分”(或零件)(parts)。所以在 IDEF5 中有一个基本的“是……的部分”关系，存在于一个个体与将此个体作为其一部分的那个更复杂的个体之间。譬如，火花塞与引擎，就存在这种关系，读作“火花塞是引擎的一部分”。而这种具有其他“种类”作为其“部分”的种类，被称为“复杂种类”。

IDEF5 中的“是……的部分”关系，也完全具有两个(高阶的)性质：

反身性(reflexivity)：每个对象都是其本身的“部分”；

传递性(transitivity)：对象 a 的部分的部分，也是 a 的部分。

譬如火花塞是引擎的“部分”(零件)，引擎是汽车的“部分”，故火花塞也是汽车的“部分”。

1.4.7 过程、状态和过程种类

分析研究对象的种类，离不开实例所涉及的“过程”。过程涉及两类变化：“种类”的改

变和“状态”的改变。例如，一个燃烧过程，一定数量的木头变换成了灰和煤气，木头本身被完全毁掉了，对象发生了“种类”的改变。另一种情况，如冰溶化为水，汽车喷上了另一种颜色的漆。对象本身性质没有变，只是“状态”改变了。

正像对象可以有“种类”，过程也有其通用的“种类”，不同的个别事件就是其实例。然而，过程是指“发生的事物”，所以不仅要包含其他事物作为其“部分”（如前述“复杂种类”的实例），还要指出“发生”在一个时间段上，并表明事物在这个时间段内，至少某一部分时间内是“真”（true）的。由此特点，过程之间也是可以关联的，也可以存在“子种类”关系。

1.4.8 本体论的层次

企业本体论一般来说可分成三个层次，如图 5.1.1 所示。通用性最高的层次为领域本体论，表征该领域中最通用的信息。例如为半导体生产的一个领域本体论，就要包括产品、制造技术与工具等为整个半导体生产领域所需的通用信息。

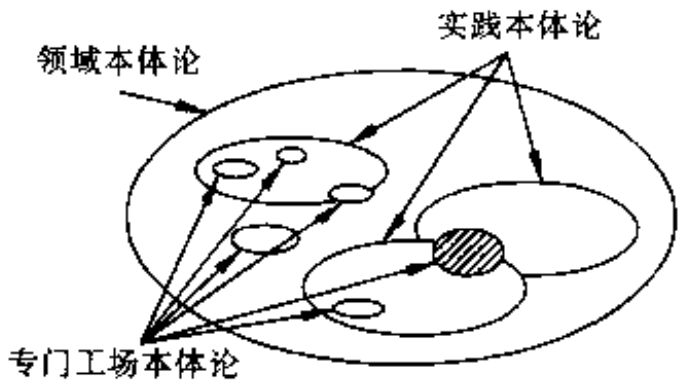


图 5.1.1 本体论的三个层次

下一层为实践本体论，它是领域本体论的推广，包含了领域中相似工场的共同特点。例如，为开发一条类似的生产线所涉及的几个半导体公司，可能要开发一个表征这条生产线特征的本体论。

最低层则是专门工场本体论，它包含了一个专门工场中，所有有关的对象种类、性质和关系等的信息。例如北京的半导体 x 厂，可以在领域本体论和实践本体论的基础上创建一个详细描述自己的设施的本体论。

1.5 本篇的构成

本体论模型在今后复杂系统的系统集成中，可能会起到很重要的作用。但是到目前为止，不论从国内 CIMS 实践所提出的需求，以及这一建模方法本身的成熟程度，都与实际应用 IDEF5 模型还有相当距离。所以本书中，我们不准备像前面几种方法那样作详细介绍，只是在本章简述了一些最必要的概念后，将“IDEF5 手册”的第 4 章“IDEF5 本体论语言”进行了翻译，作为本篇的第 2 章介绍给读者。从本体论语言的介绍中，读者可以更具体地了解本体论的内容及 IDEF5 提出的建模方法，然后可能在自己系统集成的实践中，找到应用的机会。

至于 IDEF5 的建模过程，也与其他 IDEF 方法类似。可以有下列 5 个步骤：

(1) 定义课题、组织队伍。要明确这个本体论开发课题的目的、观点、背景(与其他模型的关联),然后组成开发队伍,分配各种角色。

(2) 收集数据。就是建立所需原始数据的“池”。

(3) 分析数据。分析已有原始数据,以便本体论进行抽取。

(4) 开发本体论初稿。

(5) 细化和验证本体论。这里包括细化、评审,最后完成。

至于每一步的具体内容,这里就从略了。

第 2 章　IDEF5 本体论语言

一个领域的“本体论”，是对一个给定的应用领域中“存在什么”的一种详细的特征化描述。这种特征表述，当然应以一定的语言来表达，因此语言在本体论获取过程中起了很重要的作用，特别是出于以下两个方面的原因：

(1) 它们提供了知识获取和储存的媒介。

(2) 它们为所获取的知识提供表现形式。

表达能力强的语言结构对于本体论很重要，因为以前获取的知识，通常是作为新知识发现过程的指导。由于一般来说，不能预先确定，为获取给定领域的本体论，需要哪一种表现结构类型，故本体论的语言应在表达形式上十分丰富。然而，如果要使本体论获取方法有用，它的表现结构，对本体论开发人员就应易于理解。一种语言使用上的难易程度，是由它的“看和感觉”，以及对本体论开发过程中认知活动的支持的好坏所决定的。本体论语言，必须同本体论开发过程具有协同合作的关系，也就是说，语言必须支持过程的使用，同时过程也要有助于语言的使用。

本章重点介绍 IDEF5 本体论获取语言的两种形式：

(1) IDEF5 图表语言：它是 IDEF5 语言的图形部分，在本体论获取过程及促进通信中，提供可视化支持。

(2) IDEF5 细化说明语言：它作为一种完整的结构化文本语言，具有完整地描述一阶逻辑的能力。

这两种 IDEF5 语言相互补充，相互支持。图表语言在表达能力方面有些不足，然而其图形化的结构，使其很直观并易于使用。相对而言，IDEF5 细化说明语言则是表达能力丰富的文本语言。特别是可以表达“经典的一阶逻辑”中的一切事物。但是，要有效使用这种结构化的文本语言，需要较高的熟练性和精通程度。

2.1　IDEF5 图表语言

基本上讲，本体论是对专门应用领域中的各相关种类、个体、它们的性质、以及它们之间的关系网络加以标识与组织。IDEF5 图表语言提供各类图形化构造块，以帮助本体论的构造。在 2.1.1 节中给出 IDEF5 图表语言专用词汇的概况，其中表示的各种用以开发图表的“建造块”的用法，并结合例子加以说明。

2.1.1　图表语言专用词汇

IDEF5 图表语言中的基本符号见图 5.2.1。

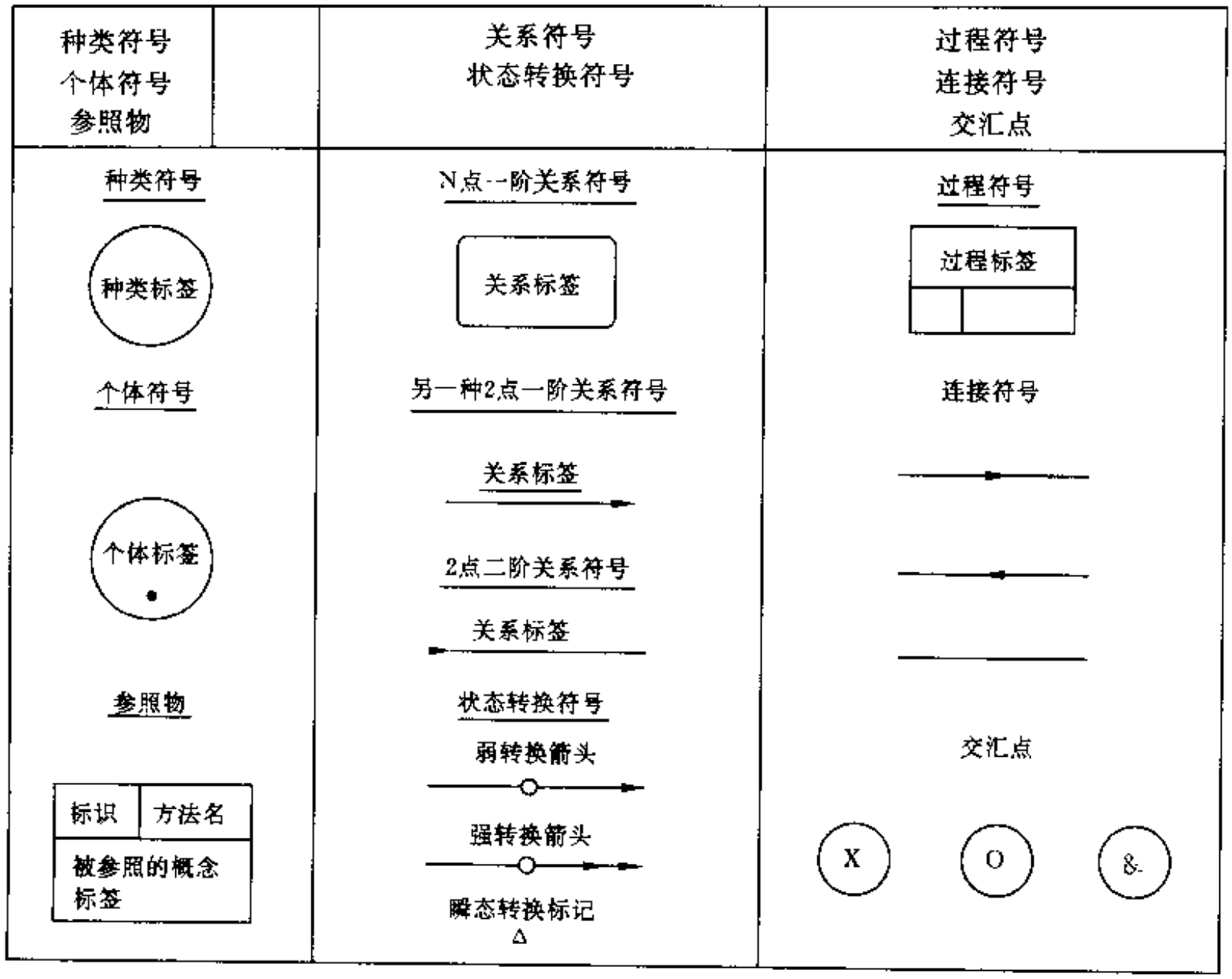


图 5.2.1 IDEF5 图表语言的基本符号

IDEF5 图表语言基本的专用词汇(见图 5.2.1)包括以下内容:

(1) 种类符号:一个种类以一个中间带有标签的圆圈表示。种类符号中的标签,应与为种类建立的相关定义表(见 2.1.4 节)中的名字相同或是其缩写。

(2) 个体符号:一个带有标签的圆圈,其中还包括一个小实心圆点(见图 5.2.1),以此标识本体论中已经标识过的特定个体。标签在本体论中应是唯一的。

(3) 参照物:IDEF5 参照物是用以参考任何 IDEF 方法中的某一概念的人工工具。参照物矩形框包括以下信息项。

① 参照物的概念标签:这是指被参照的(某个 IDEF 模型)概念标签,例如,一个活动种类,可能参考 IDEF0 模型中一个活动标签。

② ID:参照物的标识标签。

③ 方法名称:用以参照的 IDEF 方法名称。

(4) 关系符号:用一个带有圆角的矩形框标明一阶 N-点关系(即存在于一阶个体之间的关系)。在 2-点关系的情况下,带有标签的箭头符号可以代替矩形框。箭头在后的箭头符号,表示二阶关系(即存在于种类之间及种类与个体之间而非个体之间的关系)。在 IDEF5 图表语言中,并没有符号来表示 N-点高阶关系,因为经验表明它们在本体中很少

见。然而，如果有必要的话，这种关系可以通过 IDEF5 细化说明语言加入到本体中(见 2.2 节)。由于本体论中 -阶“是……的部分”关系的重要性，IDEF5 图表语言中还包括突出的“是……的部分”的标签。

各关系符号包含一个标志其所表述关系的标签。与种类符号一样，标签应或者是相关关系说明表中的关系名称、或者是其缩写。类似“是……的部分”的突出标签，还有“是……的实例”(instance of)和“是……的子种类”(subkind of)，它们用来标识二阶关系“实例”和“子种类”。

(5) 状态转换符号：IDEF5 图表语言提供两类状态转换连接符：

① 单箭头中间带有空心圆圈的符号表示弱转换。

② 双箭头中间带有空心圆圈的符号表示强转换。

IDEF5 图表语言提供瞬态转换符号标记“△”，以表示一类转换时间小于建模时规定的最小时间单位的状态转换。

(6) 过程符号：用下部带有一横线的方角矩形框来表示过程类型。2.1.5 节讨论了其句法及非正式的语义。

(7) 连接符号：连接符号是一种箭头状符号，用来连接一阶关系的种类。2.1.5 节将讨论其句法。

(8) 交汇点：交汇点符号是一类表示布尔操作符的简单符号。2.1.5 节将讨论其句法及语义。

2.1.2 IDEF 图表及其解释

本节包括各种以 IDEF5 图表语言建立的图表类型。这些图表，同任何表述一样，是信息的载体。这样，就要规定用来解释可能出现的图表的语义规则。同通常的方法一样，这些规则首先从提供对语言中的基本构造块的解释入手，然后递推应用到复杂构造块中。不过，图表语言的语义特征，同其他的图形语言不同，特别是各基本图表只能提供缺省语义，这些语义可以用细化说明语言加以覆盖(见 2.2 节)。原因在于图表的主要目的，是作为构造本体论的辅助工具，它们并非是存储本体论的基本表达媒介。上述任务是用细化说明语言完成的。这种图表语言，主要用在构造第一轮本体论的过程中，这时，主要考虑的是，以粗略方式记录下领域中存在的事物的基本种类、它们特别的性质、以及这些种类的对象之间和种类本身之间的显著关系。由此，就能专门设计图表语言中的基本构造块，以获取此类信息。不过，缺省语义可能使对所需信息的获取不很准确。在这种情况下，使用者就可以在细化说明语言中，明确地否定掉这个缺省语义。

然而，各类对象的性质及它们之间关系的精确特征，往往是因情况而千差万别的。这种关系，是否存在于种类的所有两两实例之间？还是部分实例之间？如果一种关系存在于所有实例之间，是否必须存在于两个实例之间？因此，针对某种给定结构，指定一种可以增强某种语义可能性的语义，也许就会排除其他可能的合法语义，同时也限制了在本体论构造中语言的灵活性和有用性。因为任何一种可能性，都可能出现在所考虑的领域中。对每种构造来说，缺省语义的意义，在于将被经验视为最有意义的语义列为缺省的，但如果有需要的话，也同时允许用细化说明语言，对其意义进行修改和进一步的说明。

这里就有一个问题：为什么不赋予图表语言以直接表达这些微妙语义差别的能力呢？为什么不直接采用像语义网络 SN（semantic nets，外观更为复杂）或概念图 CG（conceptual graphs）这样的图形表达语言，它们可以充分表达一阶逻辑？为什么不采用一种只求自身完善，而无需用一种非图形语言来进行补充的图形语言，以取代这种远远缺乏一阶语言能力的图形语言？以上问题的核心答案又是基于这样一个事实：即图表语言只被设计用来构造第一轮本体论，及对现存本体论进行浏览和迅速添加新的信息。它是为了理顺本体论构造及演进过程而设计的，而不是为了成为本体论的基本表现媒体。这样，在其设计中所考虑的主要问题是，要给予那些对于一阶语言以及模态和量化命题极不熟悉的领域专家，以进入基本本体论信息的直观的接口。为此目的，一种图形化语言是理想的。不过，除了相对简单的有关种类、及其特征性质和关系的信息外，图形语言无论是在可理解性和使用性上，都比标准线性语言强不了多少。事实上，就许多专家看来，在对复杂信息的描述上，前者比后者更繁琐。这样，就没有理由不直接使用细化说明语言，对本体论进行精炼，不管怎么说，这对于自动化的本体论支持工具的开发而言，总是有效得多；图形化语言本身就不是计算机可处理的。这样的话，如果要使该语言表达的信息能够操作、查询、修改等等，它就需要被编译为更标准的计算机可以处理的形式。然而，细化说明语言是计算机可以直接处理的，从而再次说明，在本体论提炼中，使用相等表达功能的图形化语言没什么优点可言。

2.1.2.1 基本一阶图表

本节讨论语言的基本一阶图表的句法和缺省语义，其中最简单的是 2-点一阶关系 2-place（first-order relation）。

1 2-点关系图表

IDEF5 图表，是通过以不同方式综合各种基本的 IDEF5 图形符号建立起来的。基本的 IDEF5 图表就是这样建立起来的最小的图。它们的句法很简单：通过关系和原始关系符号把圆圈连接起来就得到了。

这类图表中最普遍的一种结构，是用一阶关系符号将两个种类符号连接起来，见图 5.2.2。

图 5.2.2 一种基本的一阶图表的一般形式

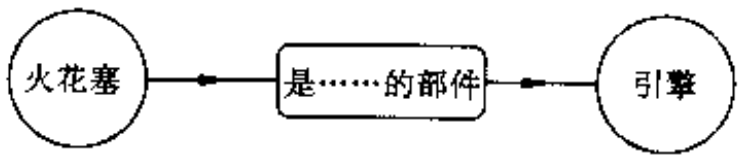


图 5.2.3 一种基本的一阶图表的例子

此类图表需要缺省语义（即公认的意义，从而无需在细化说明语言中加以进一步说明），为此，我们考虑图 5.2.3 中的实际例子。

确切地说，这个结构的缺省语义应是什么？需要注意的是，这并不意味着种类“火花塞”是种类“引擎”的一部分，因为 IDEF5 中的“是……的部分”的关系是存在于一阶物理对象之间的一阶关系，而不是像种类这样的抽象的、更高阶的对象之间的。这样，图 5.2.3 就应涉及所考虑的种类的实例——可例示说明那些种类的个体。下面给出图 5.2.3 的各

种可能的含义：

(1) 每个火花塞(所考虑的领域内)是每台引擎(所考虑的领域内)的一部分。

很明显，这个意思在物理上一般来说也是不可能的：火花塞只能是一台引擎的某一部分。

下面是一个相对较弱的说法。

(2) 每个火花塞是某个引擎的一部分。

它相对而言更为可取，但作为缺省语义，仍然太强，也许在仓库或车间有松开未安装的火花塞。

(3) 每个引擎在其零件中包括一些火花塞。

这种说法考虑到了未安装的火花塞的可能性，可是问题在于仍然可能存在某些放在一边的引擎，其中的火花塞已被移去，因而它的语义仍然太强。

(4) 某些火花塞是某些引擎的一部分。

这种说法现在考虑了两种反例，不过作为缺省语义，它仍然说的太多。也就是说，图 5.2.3 只是记录了所考虑的领域内的通常情况，但并非所有时间内的任何情况。例如：当前讨论的领域可能是一家汽车工厂的本体论，这个工厂暂时关闭了它的引擎生产部，而只做车身。因此，目前它根本没有引擎。在这种情况下，需要比(4)更弱的缺省语义：特别是像图 5.2.3 的图表，其缺省语义只能是说明，在某一个领域中，火花塞和引擎之间对于“是……的部分”这个关系的可能性或许可性。

(5) 火花塞可以是引擎的一部分。

这就是说，一件火花塞的一部分(即“火花塞”种类中的一个实例)作为一台引擎(即“引擎”种类中的一个实例)是可能的或许可的。因此，类似图 5.2.3 的图表的缺省语义，应被视为类似于程序中的类型说明，它赋予某些函数可行的参数，而无需明确指出在程序运行时，这些函数是如何被实例化的。与此类似，图 5.2.3 也主要要说明在“是……的部分”的关系中许可的参数。

需要注意的是，现在一般讨论的可能性的意思，要比单纯的逻辑可能性强。在纯粹抽象的意义上讲，几乎任何事物逻辑上都可能与任何其他事物存在某种关系。例如：逻辑上讲，一个火花塞作为一台设计奇特的计算机的一部分也是有可能的。所以，虽然这里讨论的可能性的意思，是为了反映该领域的事实本质，但它实际上是为了获取事物在该领域内可能存在的方式，(其他事物也是相等的，)而非事物在任意想象环境下的存在方式。事实上一般来说，在本体论的演进过程中，基本主张能更概括地反映领域内的实际的观察印象(即对领域中实际观察到的内容的一般化)：特定的火花塞被观察为特定引擎的一部分。这些观察，再以图 5.2.3 的形式加以概括，并出于上述原因，用(5)的语义加以解释。需要再次强调的是，这只是一个缺省的语义；类似(3)这样的更强的语义，可以通过细化说明语言中加以利用。

作为另一种 2-点一阶图表的句法，允许(通常是更喜欢)用带有同样标签标注的单个箭头来代替两个连接的符号及其关系符号。如图 5.2.4 所示。

图中的箭头，正如连接符号所表示的方向，对应于包含种类标签和关系标签的自然语言读法：火花塞可以是一台引擎的一部分。

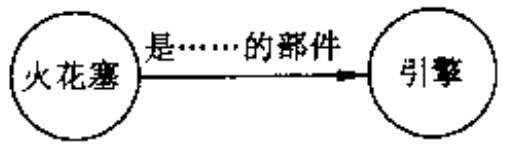


图 5.2.4　说明基本一阶图表另一种句法的例子

2　存在性图表

图 5.2.3 的基本图表语义，提出了对一个单独存在的种类符号的自然理解，如图 5.2.5。

上图应该简单理解为这样一种论述，即火花塞是领域内可以实例化的一个种类，或者更通俗地讲，火花塞是领域内可以找得到的事物的一种。这是一种非常弱的说法，它并不意味着事实上领域内有任何火花塞，而只是说可能会有，也就是说，现在所考虑的领域对此类事物是合适的。更强的说法可以用细化说明语言实现。

图 5.2.6 给出了一个单独的一阶关系符号。

图 5.2.5　存在性图表

图 5.2.6　关系存在性图表

与图 5.2.5 类似，图 5.2.6 指出，“是……的可选项”关系，是当前领域内存在的关系，或更通俗地说：一事物成为另一事物的可选项，是人们在该领域中可以观察到的一种关系。

因为它们指出了一个领域内可能存在的某一类对象，如图 5.2.5 中单独的种类符号那样一类东西，就被看作是存在性图表。由于它们能记录给定领域内观察到的专门种类，而无需任何有关这些对象与其他对象如何相处的关系的进一步信息，这样的图表是很有用的。

3　N-点一阶图表

从包含 2-点(一阶)关系符号的一阶图表语义，可以推广到包含 N-点关系符号的图表。例如，图 5.2.7 只表明，“传送到……”关系的一个实例，可以包括一台传送机，一个车身，以及一个底漆缸。

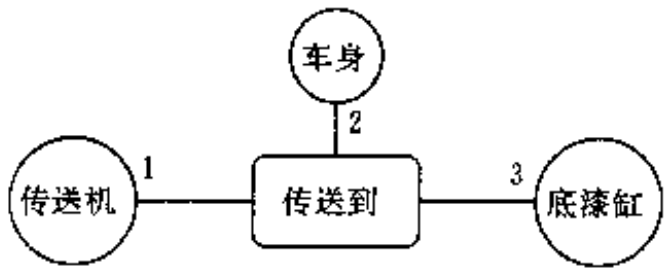


图 5.2.7　基本的 3-点--阶图表的例子

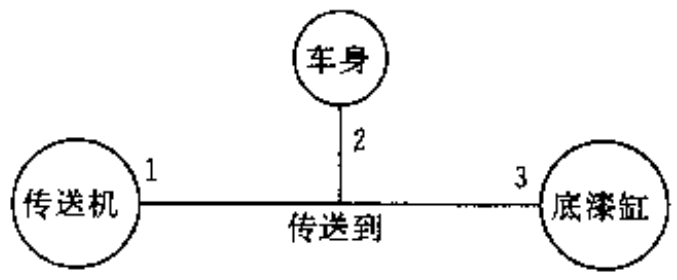


图 5.2.8　图 5.2.7 的另一种句法

图中辐条线上的数字(有选择地设置)比 2-点情况中连接符号的箭头更通用。特别是它们分别指出“传送机”，“车身”，“底漆缸”与“传送到……”关系中的第 1，2，3 个参数点相

联系，正如它们可以用自然语言读作：传送机把车身传送到底漆缸去。

同 2 点的情况一样，关系符号可以忽略，而简单地用带有标签的连接代替，如图 5.2.8。

本文推荐这种表达方式。

尽管有些不常用，4-点以上的关系也可以用这种方式来表达。为便于说明，用图5.2.9来表示 4 或 5-点关系图表的一般格式(其中关系符号的位置和种类符号的序号可以改变)；当然，如果有必要的话，含有关系标签的矩形关系符，也可以用另一种方式放在图表的接近中间的位置。

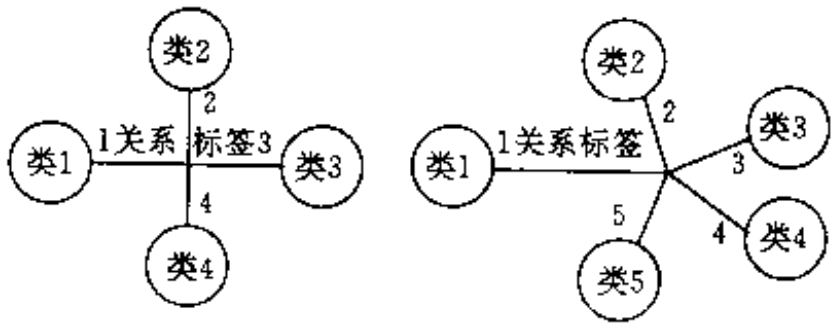


图 5.2.9　4 或 5-点一阶图表的 一般形式

4　使用个体符号

个体符号的使用，消除了图 5.2.8 中的一些图表的不确定性。例如，图 5.2.8 所表述的情况，允许多个底漆喷涂缸。不过有些情况下，需要专门指某个特定的油漆缸，因此要用图 5.2.10 的个体符号将其明显地表示出来。

现在该图表就可以表达为：一台传送机可以将一个车身传送到特定的底漆缸 PPV-1(通过个体符号表示)，这是一种比图 5.2.8 更为确定的陈述。

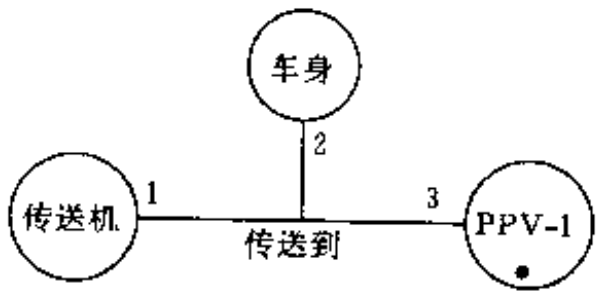


图 5.2.10　说明个体符号使用的例子

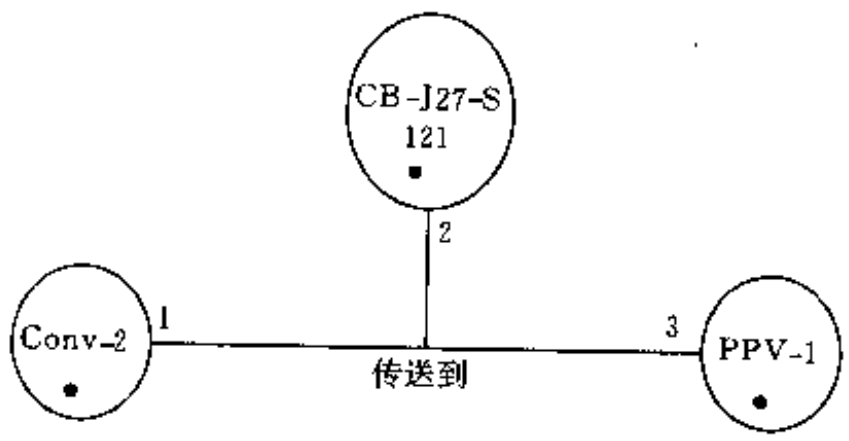


图 5.2.11　完全特例化的实例

全部使用个体符号可以完全消除不确定性。这样，图 5.2.11 就可以表示：特定的车身 CB-J27-S121，在某一时刻，被(相对于只是“可能被”)传送机 Conv-2 传送到底漆缸 PPV-1。

2.1.2.2　复杂的一阶图表

多个圆圈可以用不同的箭头连接到同一个圆圈上，从而产生了复杂图表。通常，不包括过程符号的复杂图表，还是比较方便的；它只是使你能重复使用图形元素，以及使你能

通过一张复杂图表来说明几条论述。举例来说，如果有人希望表达以下的情况：火花塞是引擎的部件，引擎又是汽车的部件，那这两个事实就可以用图 5.2.12 的简洁的形式来表示。

图 5.2.12　一个小的复杂图表

同样，如果他还想加入给定领域内的其他信息：如汽车在底特律制造，并从那里发送给销售商，这就可以方便地用图 5.2.13 表示。

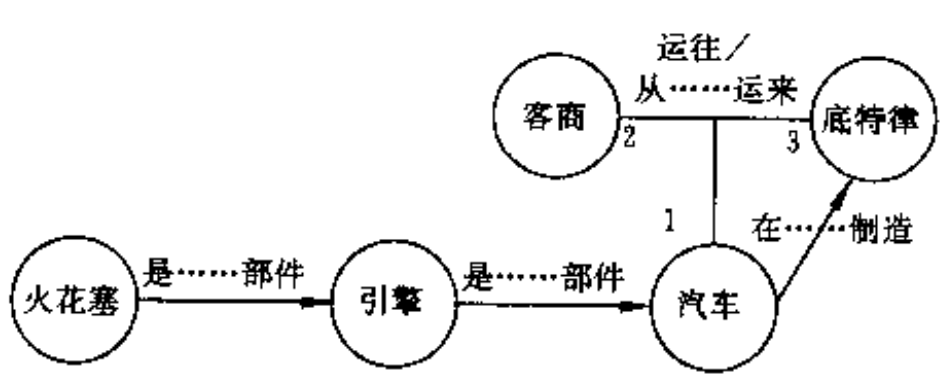


图 5.2.13　包含多种关系的复杂图表

具有过程符号的复杂图表将在 2.1.5 节讨论。

下面介绍单一关系复杂图表的约定。

比较常见的 IDEF5 图表中，通常只有一种类型的关系。在这种情况下，为了避免无谓的混乱，建模者可以省略关系标签，只在图的底部标注上关系符号的含义。如图 5.2.14 所示：

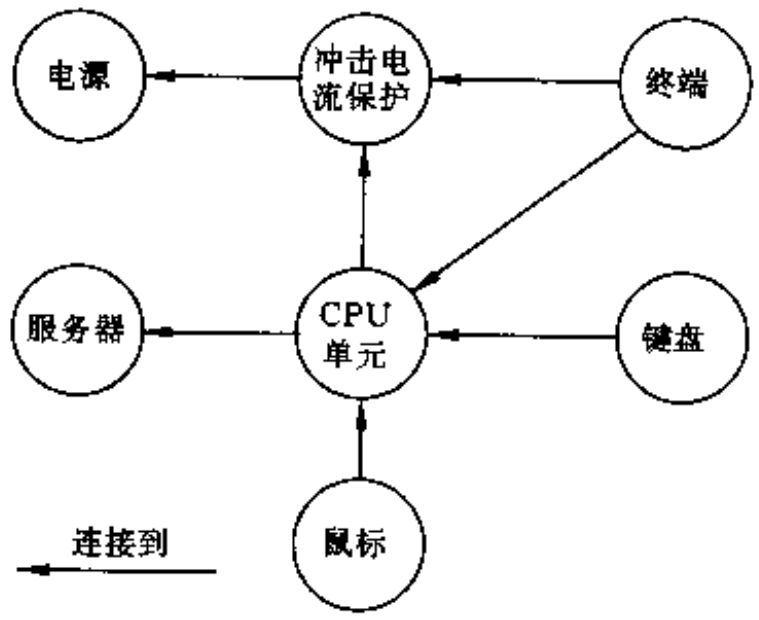


图 5.2.14　个人电脑的外设连接

注意，“连接到”这个关系已包括在 IDEF5 关系库中。

2.1.2.3 二阶图表

如前所述，二阶关系是存在于二阶对象(即种类和一阶关系)之间的、或一阶与二阶对象之间的关系。前者的范例是种类之间的“是……的子种类”关系，而后者的范例则是“是……的实例”的关系。要表示二阶关系就得引入一种新的箭头符号因为两种关系都是由连接于表示种类的圆圈间的箭头符号来表示的。由于它们的语义完全不同，为了避免产生混淆，因而采用完全不同的结构。

二阶关系图表的基本形式与一阶关系很相似，只是采用二阶关系箭头来代替一阶关系箭头。

二阶图表的语义也比一阶图表要明确，特别是这些图表是直接涉及到种类，而不是其实例的：图 5.2.15 表示左边的圆圈所代表的种类，与右边的圆圈所代表的种类之间，存在由箭头所表示的二阶关系。另外还要注意，缺省语义并不是已被鉴定为合适的；不像一阶图表，其语义不仅是考虑事物如何能存在于领域内的问题，而且还要考虑事实上两类事物之间是如何关联的。其原因在于，这种二阶论述通常关心的是所考虑的种类的本质，而常常不依赖于领域内的偶然事件(当然，也非永远如此)。图 5.2.16 说明一种包含典型二阶关系“是……的子种类”的图表(它是作为 IDEF5 提供的原语)。

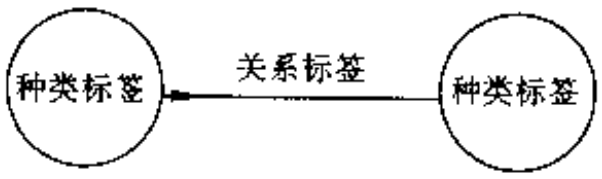


图 5.2.15 基本的二阶图表

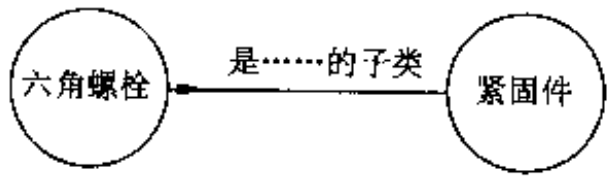


图 5.2.16 “是……的子种类”关系的二阶图表的实例

由给定语义可知，种类“六角螺栓”是种类“紧固件”的子种类。同理，图 5.2.17 的语义表示，美国公民比加拿大公民多(即更书面地讲，就是种类“美国公民”比种类“加拿大公民”有更多的实例)。

由于“是……子种类”关系在本体论中很普遍，不附标签的二阶关系箭头的缺省语义，就表示这种子种类关系。这样就使用户不必在一张图表上重复使用“是……子种类”的标签。

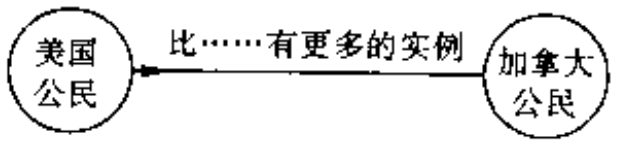


图 5.2.17 通用二阶图表的一个例子

2.1.2.4 关系图表

这一节主要介绍如何用 IDEF5 图表来获取和显示一阶关系之间的关系，这样做的主要目的来自于二阶关系，是从人们通常根据当前概念描述和发现新概念的观察中推演出来的。这与 Ausubel 的知识学习理论相一致，在这个理论中，学习通常是发生在“将新信息包容在更为通用更为兼容的概念中”这一过程。利用这一假设，一种描述新的(或不太理解

的)关系的自然方法,是将它与一个“已被很好理解的”旧关系相联系,更广泛地讲,就是在其他关系的概念空间中,分出它的位置。IDEF5 的关系库,提供了一个基本的参考以帮助用户用关系图表发现关系并将其特征化。

关系图表的基本句法如图 5.2.18 所示。

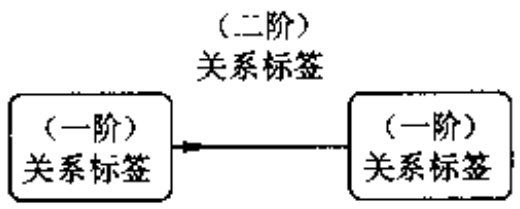


图 5.2.18　基本关系图表的通用形式

应该注意到,除了将种类标签换成一阶关系标签之外,图 5.2.18 的句法,与二阶图表的句法是相同的。在上述图表中,用二阶关系符号,并不仅仅是为了使表达符号简练,因为种类与一阶关系是同样的逻辑类型。如前所述,种类是个体的特性,一阶关系是这些个体之间的联系,即 n 元组特性。因此,一阶关系之间的关系,与种类之间的关系有同样的逻辑类型;两者都是把相对于个体而存在(或否)的实体关联起来,因此这些关系都为二阶关系(如图 5.2.18 明确指出的那样)。

下面将要说明,如何利用关系图表使包含关系(包括种类之间的关系以及关系之间的关系)的概念分析易于进行。考虑一个汽车制造公司的工程物料清单(BOM)的本体论,“是……的部分”关系在 BOM 中起了一个很重要的结构角色。在数据收集过程中获取的数据可能包含以下有关 BOM 的叙述:

- 自动变速装置是变速装置的一种变形。
- 手工变速装置是变速装置的一种变形。
- 收音机是汽车的一个可选项。

一件产品的一个“变形”,通常是由客户选定的产品的本质特征。客户挑选出几种可能变形中的一种。在本例中,汽车用户可以在自动变速及手动变速装置中选择一个。请注意,“有一个变速装置”是汽车的一个本质特性。从多种变速装置中选择一种变速装置,是客户要做的。

可选项是由客户挑选的产品的一个特征。“选项”之所以不同于“变形”在于:选项可以从一个产品中完全去掉,而“变形”一定需要描述。在本例中,收音机对于汽车功能并非必需的,因而可以有选择地去掉。

在源叙述分析的基础上,IDEF5 开发者假定了两种关系的存在,称之为“是……的变形”,“是……的选项”。这些关系将以图 5.2.19 所示方式被映射到一个关系图表中。

在这一阶段,IDEF 开发者应浏览一下 IDEF5 关系库。他或她应该注意到关系“是……的选项”和“是……的变形”在概念上与一些库中关系很相似。尤其他或她会发现“是……的选项”这一关系是关系“是……的部分”的特例。

假定上述看法现在用一个二阶关系“是……的特例”来代表,且在图 5.2.20 中用关系图表的方式来表示。

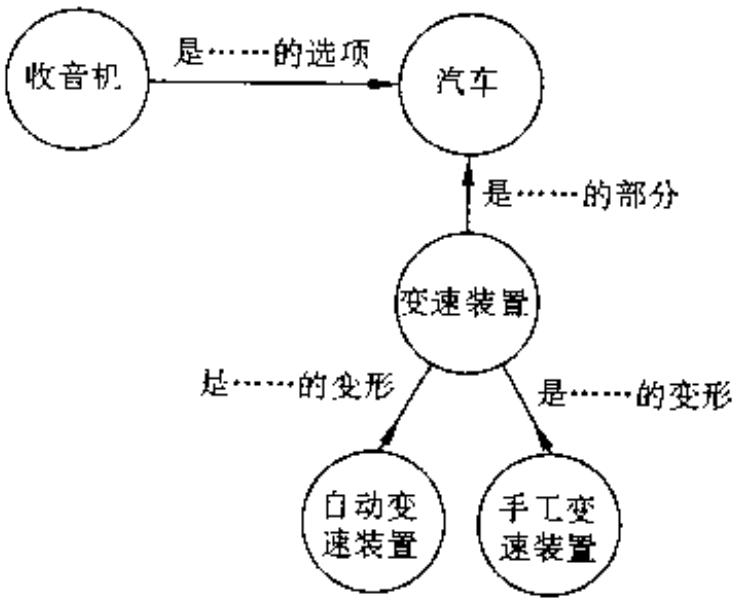


图 5.2.19　物料清单的关系图表

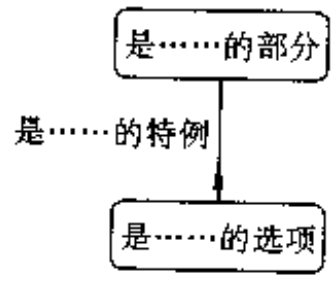


图 5.2.20　有关“是…的特例”关系的关系图表

高阶的“是……的特例”这一关系，用来确认一阶关系之间的通用化-特例化关系。更进一步的反复研究，可以使 IDEF5 开发者画出一个更为详细的关系图表，这个图表将关系“是……的选项”置入一个关系分类图中(即一种通过通用化-特例化关系相联系的，显示关系等级的特殊类型的关系图表)。

需要指出的是，单一关系的复杂图表惯例，对二阶关系也同样成立(如图 5.2.21)。复杂的二阶关系图表，也同样可以按一阶图表的方式构成。这就是说，人们在一张单一图表中，可以“重新运用”种类和(一阶)关系符号。例如：图 5.2.19 和图 5.2.20 中包含的信息，可以在一张图表中表示(如图 5.2.22)。

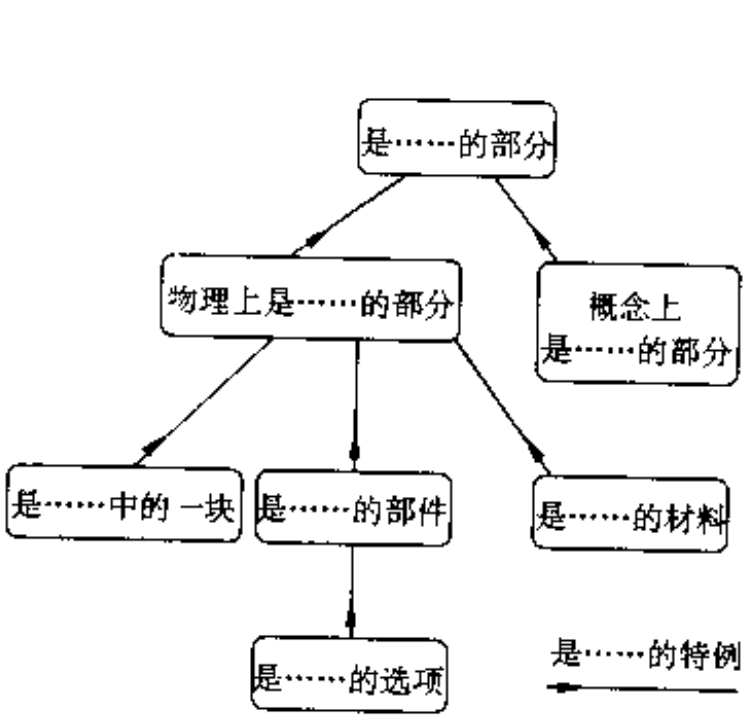


图 5.2.21　关系“是……的部分”的部分关系分类学

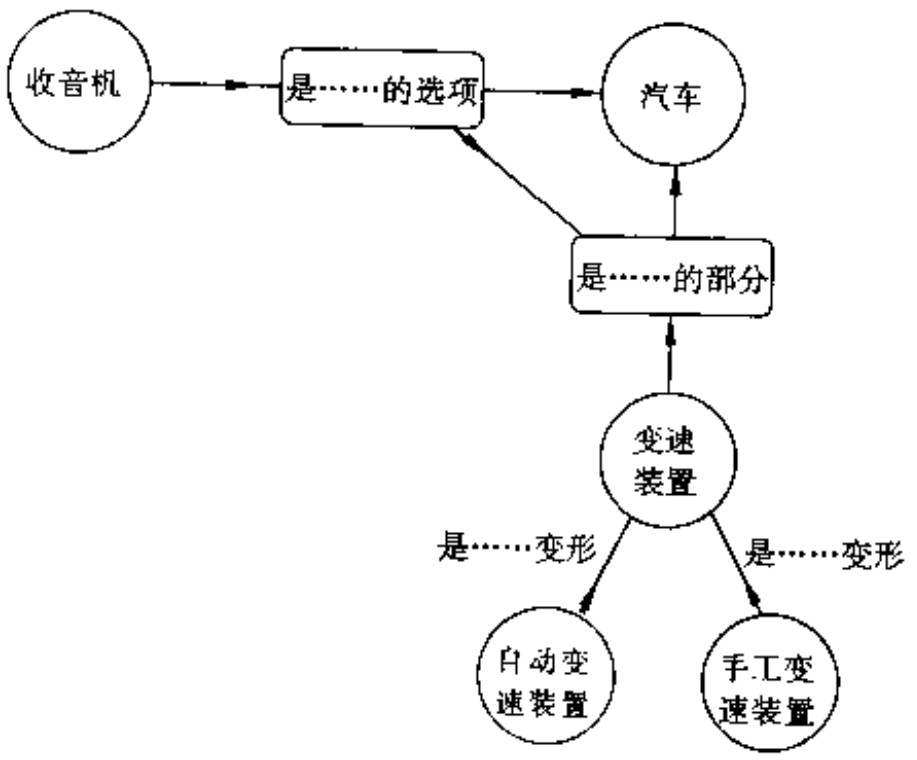


图 5.2.22　复杂二阶关系图表

2.1.2.5　参照物

一个 IDEF5 参照物，是用来指示一个其他 IDEF 方法中概念的人工建模工具。参照

物矩形包含以下的信息项。

(1) 参照的概念标签:这是被参考的(相应 IDEF 模型中的)概念标签。例如,一个活动种类可以参考一个 IDEF0 模型中的活动标签。

(2) ID(标识):指示参照物的标识标签。

(3) 方法名:这是所参考的 IDEF 方法名。

IDEF5 的参照物,提供一种将 IDEF5 与其他 IDEF 方法相连接的机制。例如,IDEF5 中一个种类,可以用 IDEF1 中的实体来表示,IDEF5 开发者可以在 IDEF5 模型中明确指出这一参考。

2.1.3 合成图表

因为关系"是……的部分",在设计、工程、制造本体论中非常通用。"是……的部分"这一标签,以及它所对应的公理,都包括在 IDEF5 语言中。特别是,这种能力可以使用户能表述组成一个给定对象种类的事实。通常,这可以通过图 5.2.23 的图表方式来实现。

如 2.1.2 节所述,图 5.2.23 只是一种方便地描述图 5.2.4 所述的基本一阶关系图表的多重实例的手段,因此图 5.2.23 的缺省语义是:A_1 们(即 A_1 的实例,多数)可以是 B 们的一部分,A_2 们可以是 B 们的一部分,…A_n 们可以是 B 们的一部分。然而,从"是……的部分"的角度说,通常需要更强的读法。例如,在一张物料清单中,有人可能想说,不单 A_1 们可以是 B 们的一部分等等,而且事实上,每一个 B 都由一个 A_1,一个 A_2,…组成。举一个例子,某人可能想要表示某一特定种类圆珠笔的组成结构,如图 5.2.24 所示。

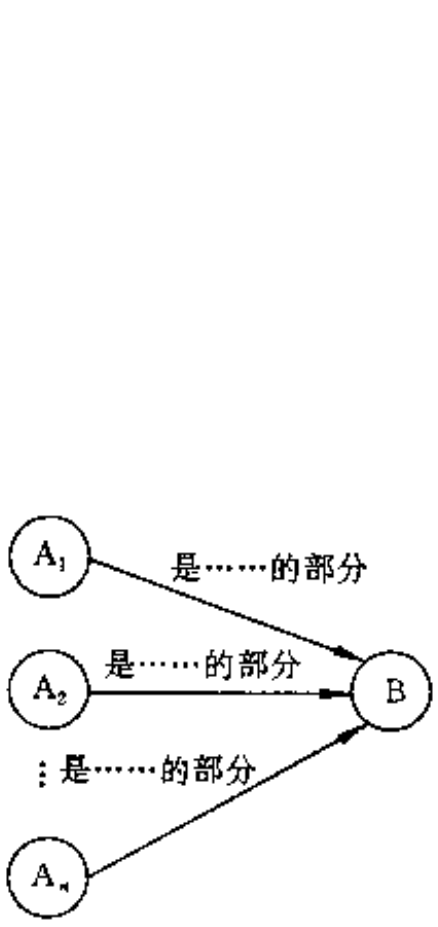


图 5.2.23 合成图表

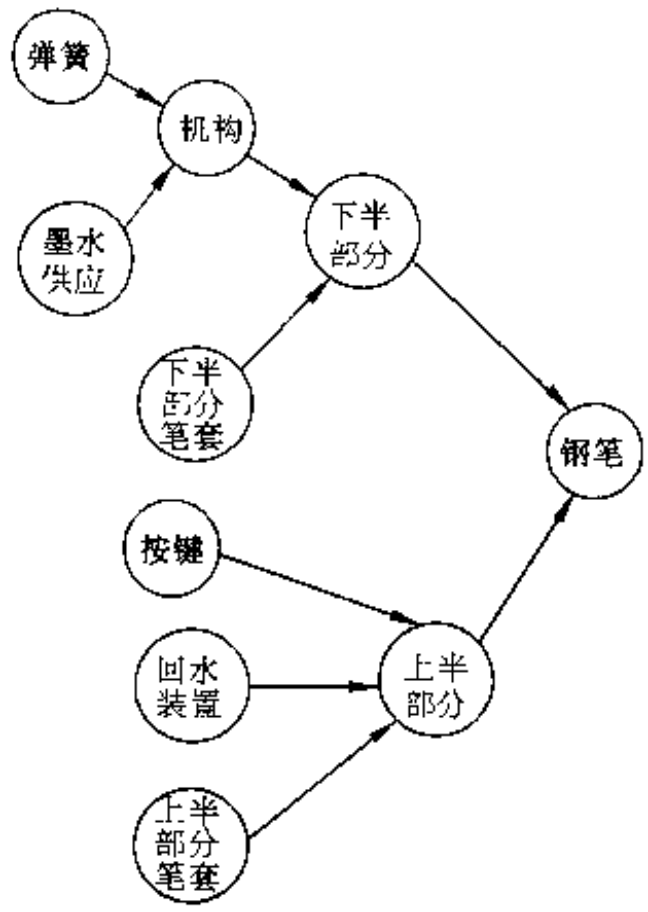


图 5.2.24 种类"圆珠笔"的合成图表

为了获取这种更强一些的意思,我们必须借助于详细描述语言,对于图 5.2.23 的每一个实例,加入"每个 B,事实上拥有一个 A_1, 一个 A_2,…和一个 A_n 作为其部分"的陈述。

有了这种更强的语义，图 5.2.24 的图表可表达这样一种意思，即在这个所考虑的领域中，一支圆珠笔有一个上半部分和一个下半部分；前者包含一个按键，一个回水装置，和一个笔套；后者则由下半部分笔套和一个机构组成；而机构又由一个弹簧和一个墨水供应装置组成。

隐匿合成信息

如图 5.2.24 所示，合成图表可以进一步细化，但这种细化可能造成 IDEF5 图的混乱。比如，除了描述种类“圆珠笔”的组成结构外，有人可能还想谈论许多与它和它的实例有关的其他关系，例如，笔可以在华盛顿塞古姆生产的；通常钢笔比圆珠笔贵；而圆珠笔是笔的一个子种类等。因此，从很多方面来说，一个种类的组成结构，往往是不相关的，在这种情况下，有必要隐匿有关信息。被隐匿的信息在图中以一个双圆来表示一个种类（而非一个标准的单圆），同时在圆上部有一个大写字母“P”（表示“是……的部分”关系）来指出被隐匿的信息的种类，如图 5.2.25 所示：

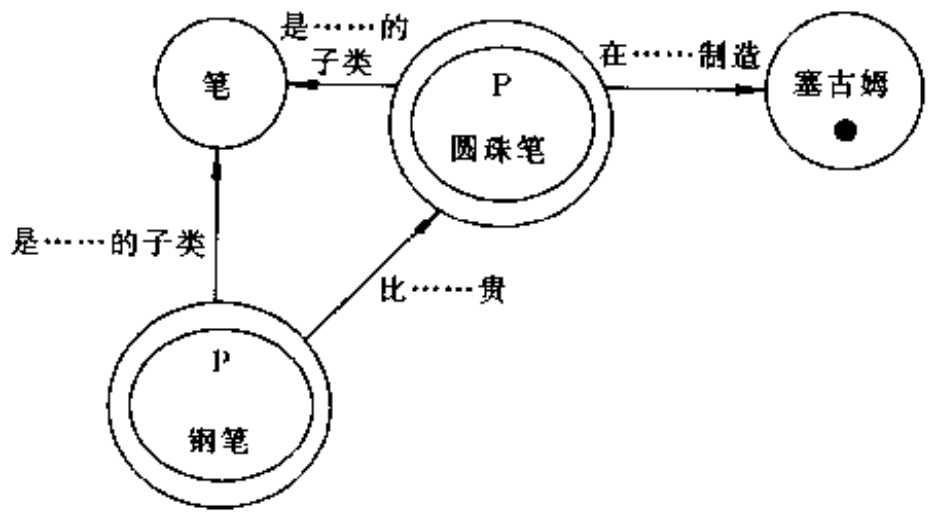


图 5.2.25　隐匿合成信息

请注意，本例在一张图表中同时使用一阶和二阶关系符号。

2.1.4　分类图表

人类用来组织知识的最常用的结构机制是分类图表。参与信息收集的领域专家们，通常说这样的话：A 是一个 B，A 是 B 的一种类型，A 是 B 的一个种类。以这种方式组织知识的认知活动，称为“分类”。有几种分类方式，其中两种尤其著名，它们是描述归类法和自然种类分类法。在描述归类法中：(1) 分类中顶层种类 K 的定义特性，同它所有子种类的定义特性，组成了归属为该种类成员的严格的必要和充分条件。同时(2)所有子种类的定义特性，都归入 K 的定义特性，即每一子种类的定义特性都继承了 K 的定义特性，而 K 的定义特性则构成了一个更通用的概念。

相反地，在自然种类分级中，它并不假定，要归属为顶层种类 K 的成员，需要严格的必要和充分条件；不过，它的实例中都有一些基本的结构特性，当它们以不同方式特例化时，就产生了 K 的子种类。这种分类方案的最好例子，就是真正的自然种类，如金属，猫科动物等，但同样的意思也可扩展到人工种类，如：汽车及数控机床。上述两种分类方

法如图 5.2.26 所述。

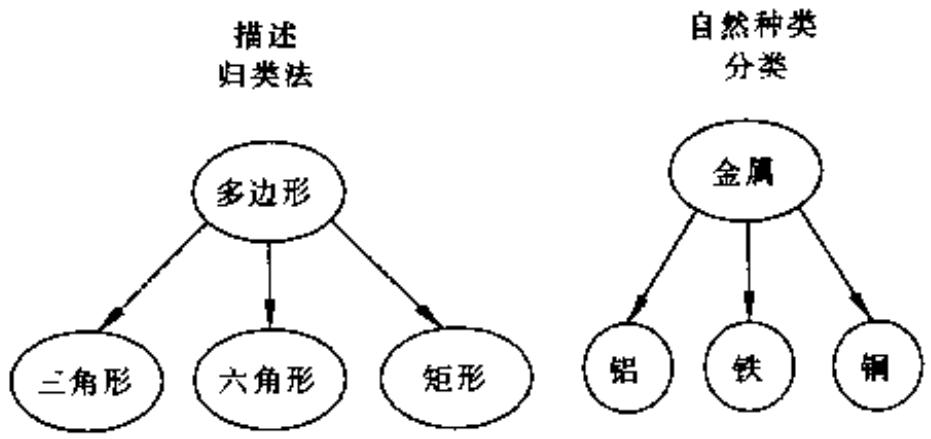


图 5.2.26　分类的不同类型

显然，以种类的中心意图来说，IDEF5 图表语言的自然应用就是开发分类图，或者我们称之为“分类图表”。

事实上，分类比所举的例子中提出的要细致得多。许多分类方法在方案中一些通用种类的“下面”，都包含几级更特殊的子种类（这些方案通常称为“是一个递阶层次”，但是前面曾说过，“是一个”的用法在 IDEF5 中是极不鼓励的，因此关系“是……的子种类”或者关系“是……的实例”就被用来根据具体意思来代替它）。比如，在一个项目的计划过程中，将项目完成所需的资源加以分类是必需的。一个资源可以被非正式地定义为被消费掉、被用来或需要用来执行某种活动或任务的对象。因此，资源在过程中起着一个“使能”的角色。分类图提供一种将必要的资源进行分类的自然方式，如图 5.2.27（无任何标签的二阶关系符号隐含地代表子种类关系）。

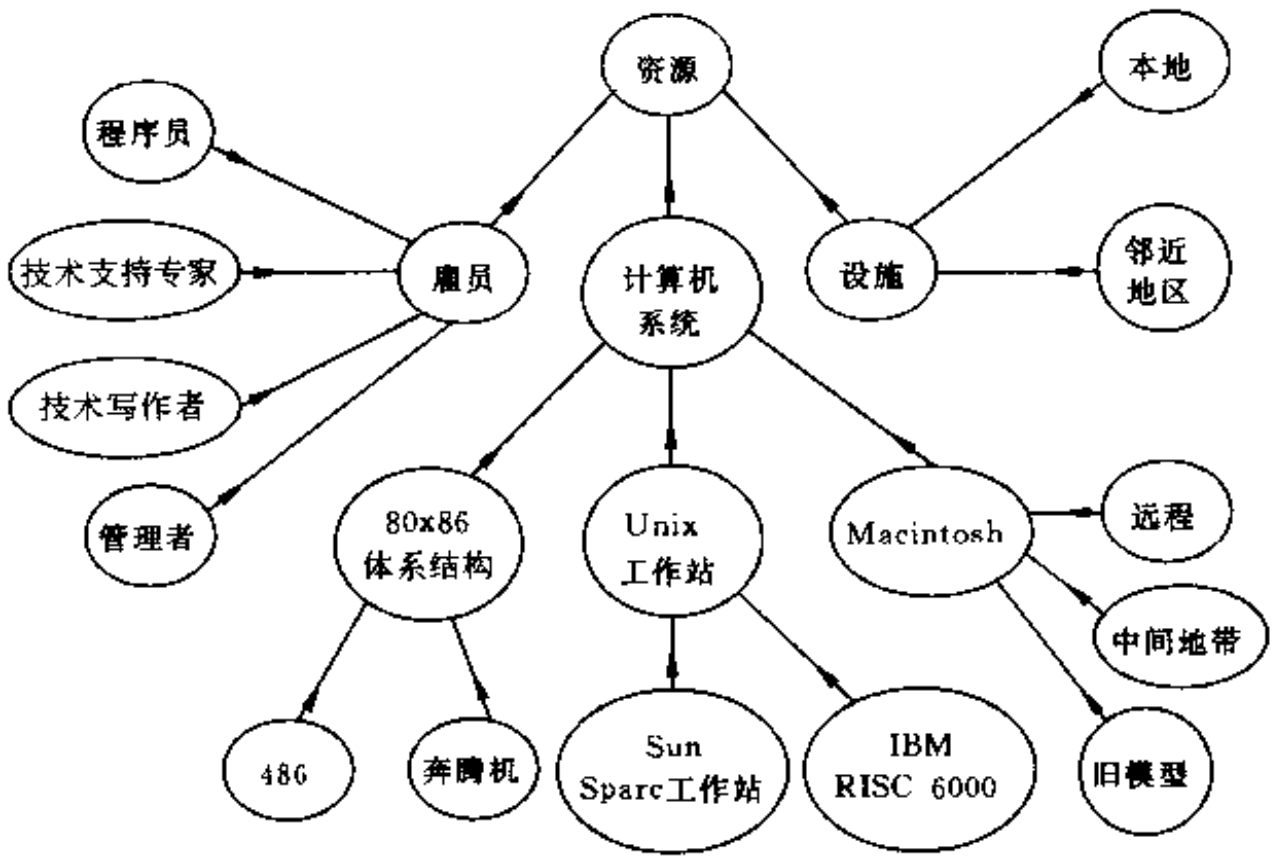


图 5.2.27　资源分类

隐匿分类信息

事实证明，隐匿分类图表的细节，对复杂的合成图表，通常是一种很有用的方法。因此，在某些情况下(例如：那些需要强调人和设施的地方)，有关计算机系统的信息，就无需明确给出。在合成图表中，隐匿信息可以由一个双圆来指明，在这个双圆中加上一个大写“C”(指“分类”)，如图 5.2.28。因而，我们可以隐匿该信息，并加入有关设施的信息，而得到了该图。

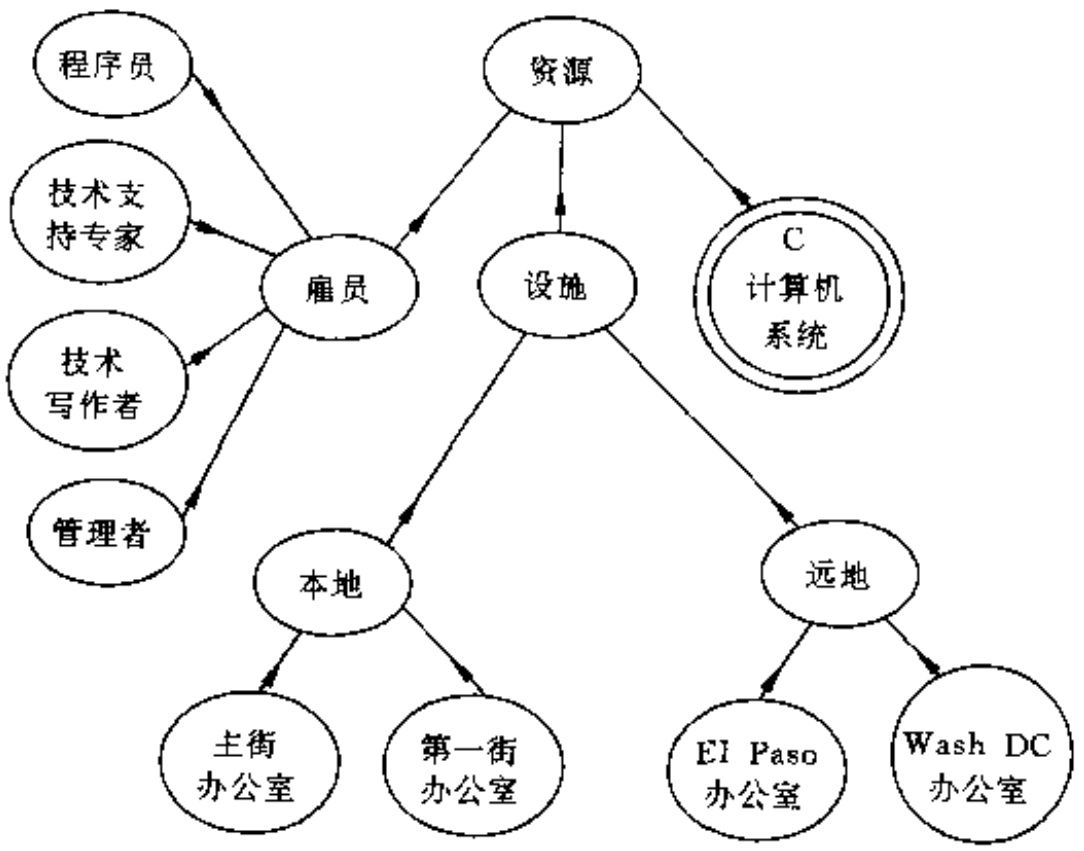


图 5.2.28　有隐匿信息的资源分类

2.1.5　对象状态图表

对有关种类和状态的信息及有关过程的信息，我们并没有作清楚的区分。这一小节将介绍 IDEF5 图表语言如何能很仔细地表达“以对象为中心”的过程信息，即与某些过程关联的对象的种类，及其各种状态的信息，由这样构成的图称为对象-状态图表(OS)；将这些 OS 与前面介绍的各种种类的图表集成的问题，将在下面讨论。

经历一个过程的对象，可以有两种变化类型：种类的变化和状态的变化。实际上，两种变化之间并没有形式的区别：在某种特定状态下的给定种类 K 的对象，可以简单地被看作构成了 K 的一个子种类。基于这个目的，例如，热水可以被作为水的一个子种类。然而，在图表语言中将这二者区别开来，以明确指出这一类东西处于某种状态，是非常有用的。这是通过一个冒号注解完成的，如“种类：状态”。比如，热水可以用标签“水：热”来标记、冰水则用“水：结冰”标记等。如图 5.2.29 所示：

在这里，“是……的子种类”关系，最好被认为是“是……的状态”关系，如图 5.2.30 的分类图所示。

为了指示过程中对象是如何改变了种类或状态的，需要一种全新的结构。本节下面

的部分将引入这种结构的句法及信息语义。

图 5.2.29　种类及状态

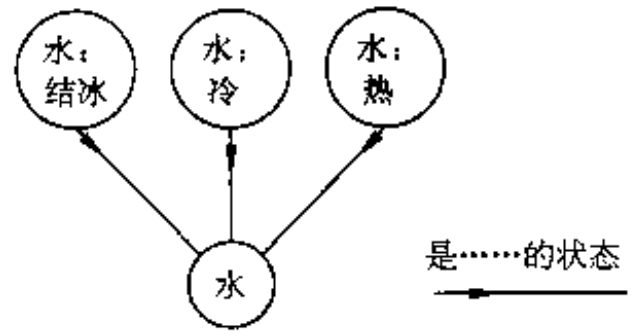


图 5.2.30　图表描述水的状态

2.1.5.1　基本对象状态转变图表

首先，也是最基本的结构，是简单“转变”连接，如图 5.2.31 所示。其中引入一个空心圆圈是为了区别对象状态转变连接和通常的关系箭头。

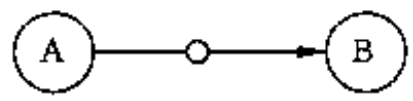


图 5.2.31　基本状态转变图表

在连接中以 A，B 标记的圆，仍然表示两个类，前面提到的有关种类及种类符号的内容仍然起作用。然而为了以一种更通用方式表征对象在一个过程中的转变方式，很自然的，可以根据在过程中不同阶段它们所处的状态，来将这些对象分类。因此，状态转变图表中的种类符号，往往是指处于某一特定状态的种类，例如：干木头，热水，未经返工的零件等。为了强调种类符号的这种作用，它们的含义通常就被认为是“对象状态”，或简称“状态”。

直观地，一个对象状态转变连接，指出了一旦有这样一个可允许的转变，就会有一个给定状态 A 的对象，可能被修改、转变、或消耗掉，而产生了不同状态的对象（很可能是同样的对象）。图 5.2.31 描绘了观察到的由 A 到 B 的某种类型转变的情境，但这里既没有知识，也没有愿望去说明转变中包含的过程。

区别对有关某个在给定状态的对象的特征描述，和控制该对象如何转变到那个状态，和从那个状态转变出来的条件和规则，是很重要的。在 IDEF3 过程描述获取方法中，有关“进入”或“离开”一个状态的条件，分别称为“入口”，“出口”条件。IDEF5 细化描述语言，可以用来说明对一个给定状态的相关入口及出口条件。

有关包含在一个状态转换过程中的附加信息，可以在 IDEF5 中显示，如图 5.2.32，图 5.2.33 所示：

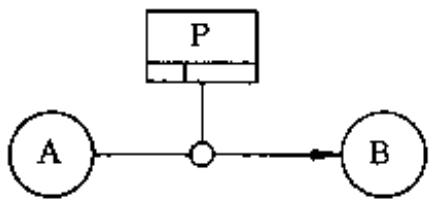


图 5.2.32　发生在一个过程中的对象状态转变图表

更准确地讲，图 5.2.32 所示的状态转变图表的语义是这样的：在过程 P 的任一事件 p 的某个初始时间段 t_{init}，有一个处于状态 A 的对象 a（在图 5.2.34 中用间隔图的原子表达式“A a”表示），经过一段时间到了 p 的某个最后时间段 t_{fin}（可能是最后一刻），可能会

出现一个处于状态 B 的对象 b。

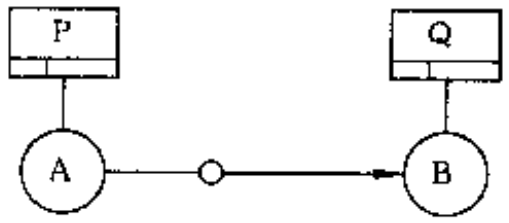


图 5.2.33　在两个过程之间的对象状态转变图表

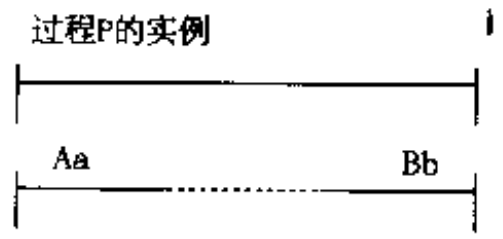


图 5.2.34　图 5.2.32 的通用语义

然而，有可能一个已被识别的转变不是在一个过程中出现，而在一个给定过程 P 的最后和另一个过程 Q 的开始之间发生的(见图 5.2.33)。这种特殊情况在 2.1.5.3 节中予以讨论，在那里，两个相邻的过程 P 和 Q 相遇了。这样一个图表中的转变箭头，并不代表一个单独的未说明过程，而是标记着一个在过程 P 的末端，处于状态 A 的对象的存在，及转换成处于另一个过程 Q 开始时的处于状态 B 的对象的存在，显示成时间间隔图，如图 5.2.35 所示(P,Q 实例间的虚线表示它们不必是瞬间邻接的)。

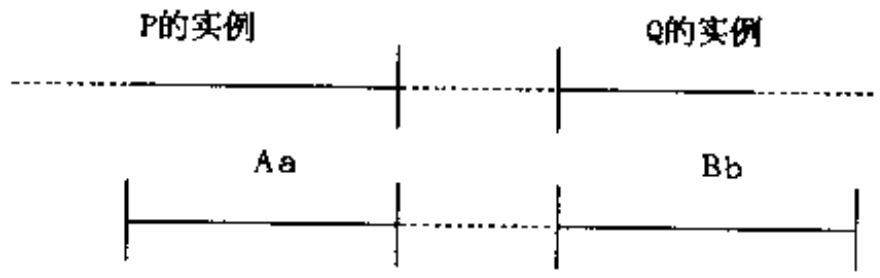


图 5.2.35　图 5.2.33 的通用语义

2.1.5.2　强状态转变图表

一个对象状态转变连接的语义，关系到转换初处于状态 A 的对象，是否同转换后处于状态 B 的对象是相同的对象，正如一个未被上漆的对象变为一个上漆的对象；还是变为不同的对象，如一片木头经过炉子变成一堆灰。基本状态转变图表，可以被用于以下三种方式中任何一种：(1)过程开始与结束时的对象是不同的；(2)我们并不知道它们是否不相同；(3)它们是否相同没有什么关系。另一方面，如果一个建模者想明确表示，在一个状态转换的实例中，开始时的对象与结束时的对象是相同的，就需要一个强状态转变连接。这就需要在一个状态转变箭头上再加上一个箭头，如图 5.2.36 所示。

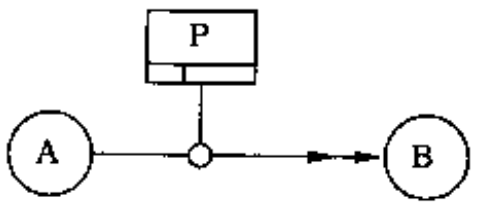


图 5.2.36　强状态转变图表

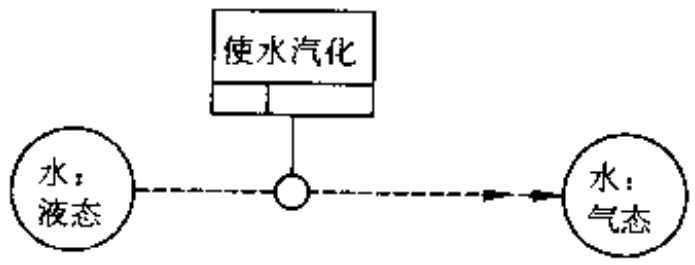


图 5.2.37　强状态转变图表的一个例子

为了能在过程的一个实例 p 中表达非瞬间的状态改变，一般不要求 p 的开始时刻 t_{init}(此时对象处于状态 A)，与最后时刻 t_{fin}(此时对象处于状态 B)，在时间上是相邻或覆

盖的。在这种情况下，中间的时间可以被看作由状态 A 导致向状态 B 转换的过程，但在这段时间里，对象并不处于两种状态中的任何一种，如一定量的水从 50℃(A 状态)加热到 100℃(B 状态)的过程。这种瞬间及非瞬间状态转变之间的语义不确定性，在图中以连接两对象状态之间的虚线来反映，如图 5.2.37 所示。

2.1.5.3 瞬间状态转变图表

为了表示对象在转变中的同一性，通常需要明显表示一个过程的瞬间状态转变(相对于某个时间量级)。相应地，本体论句法，允许建模者在对象状态转变连接的圆圈中，画一个 Δ(如图 5.2.38)，表示转变的时间段比建模内容中可识别的最小时间单位 Δ 还短。

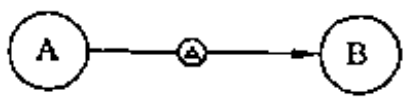


图 5.2.38 瞬间状态转变图表

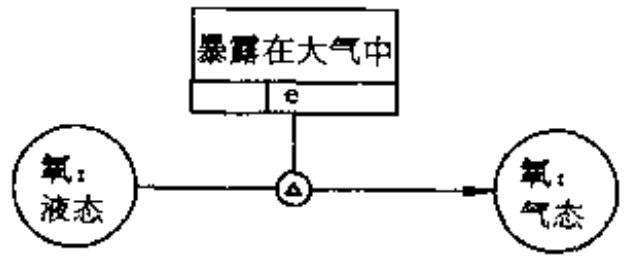


图 5.2.39 瞬间状态转变的一个例子

当然，如要表示强状态转变，可加一个双箭头。比如，当液态氧暴露在空气中时，它就立刻转换为气态氧(如图 5.2.39)。

有人可能对瞬间转变之前或之后的过程，比瞬间转变本身更感兴趣。这种情况可以用图 5.2.39 的间隔图来说明，如图 5.2.40 所示。

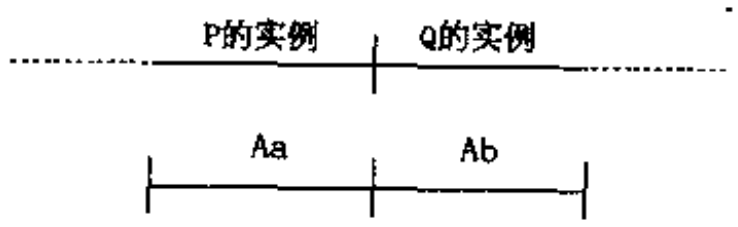


图 5.2.40 图 5.2.39 的间隔图

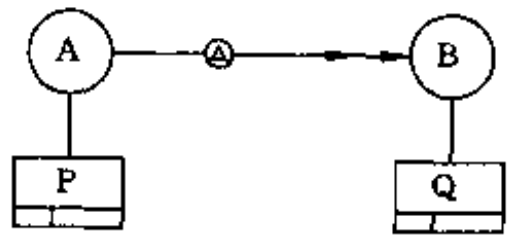


图 5.2.41 图 5.2.40 的准确表述

为了更准确地表达图 5.2.40 的内容，可使用图 5.2.41 的结构。这张图表中的转变箭头，标识着一个对象，从处于过程 P 的最后时刻的状态 A，转变到一个对象处于另一个过程 Q 的开始时刻的状态 B 的时间间隔。

举一个例子，可以设想，一把刀具被驱动着穿过一件工件，直到碰到一个限位开关，切割机才关闭，刀具返回其初始位置(图 5.2.42)。

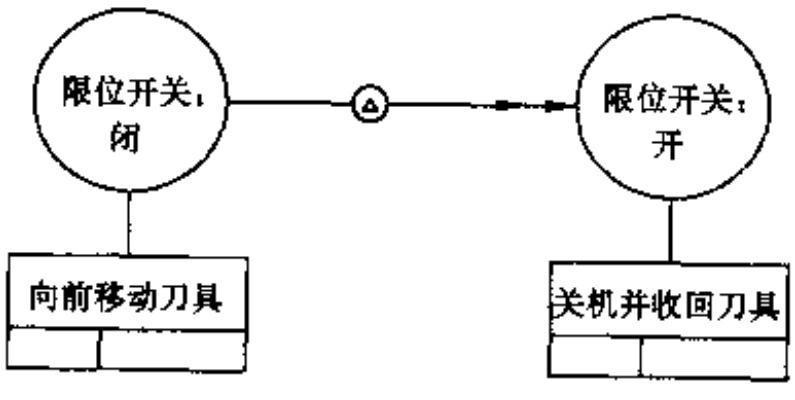


图 5.2.42 图 5.2.41 的切割开关的实例

2.1.5.4 复杂对象状态图表

到此为止，前面介绍的简单对象图表，可以组合起来形成更复杂的对象状态图表。然而，与以前介绍过的复杂图表不同的是，复杂状态图表不单为了表达上的方便，它们还带有更多其他的信息。比如，图 5.2.43 就比图 5.2.41 描述的信息要多，后者并没有指明对象 a 在它成为状态 A 之前的状态(特别是没涉及状态 D)，以及对象 b 在它成为状态 B 之后的状态(在本例中就是指状态 C)。

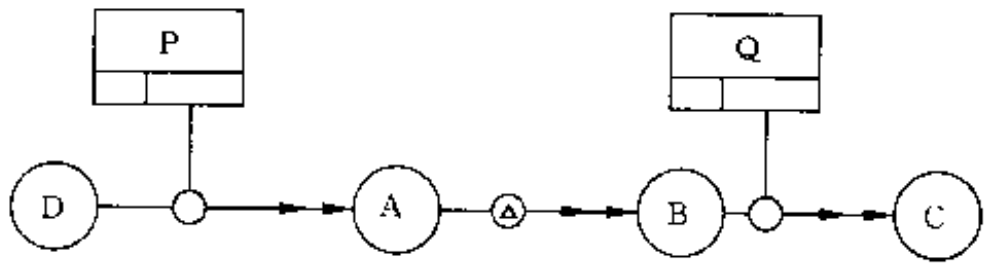


图 5.2.43 一个有更多信息的对象状态转移图表

在某些时候，有人可能只想记录转变过程本身，而不想提供有关转变"两头"的更多的信息，这样图 5.2.31 就足够了。反之，则需要使用图 5.2.43 这样的图表。图 5.2.44 显示了图 5.2.43 的间隔图。因此图 5.2.43 在这样的情况下是适用的：即某人知道或希望表达一个过程中一个对象变成状态 A 之前的、以及一个对象处于状态 B 之后的信息。

另一个有关复杂图表如何提供更多信息的例子如下：

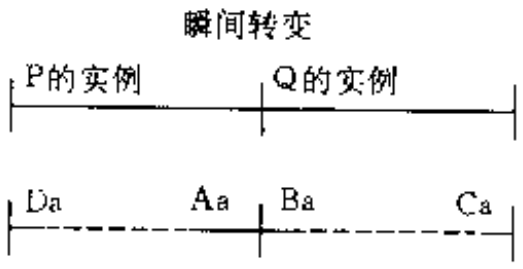


图 5.2.44 图 5.2.43 的间隔图

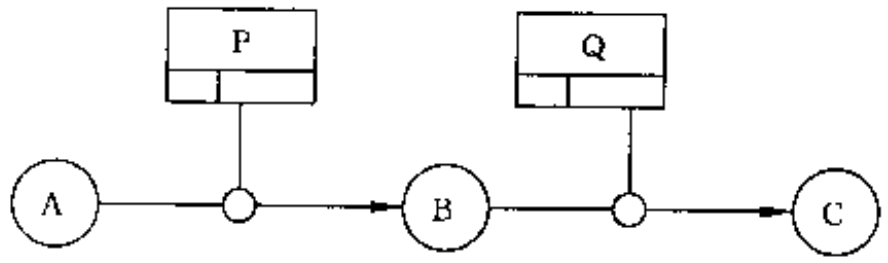


图 5.2.45 一个复杂的对象状态图表

图 5.2.45 的意思是：首先，有某物在过程 P 的实例中，从状态 A 到状态 B 的转换过程；接下来又有一个同样事物，在过程 Q 的实例中，从状态 B 到状态 C 的转换。这就是说，图 5.2.45 中对象状态 A 和 C 的符号都与 B 符号相连。这一事实表明：首先，P 和 Q 可以被看作是一个更复杂的包含了这两者的过程 R 的一部分；其次，在一个贯穿 R 的实例中的 P 和 Q 的实例，共享一个处于 B 状态的对象。这种通用语义可用图 5.2.46 的间隔图加以描述。

当然，这样的语义可以推广至更复杂的对象-状态图表。图 5.2.45 可用于与图5.2.47 相对照，在后者中，状态 A 与状态 C 的符号，被分别连到分开的状态 B 的符号上。

由于两个图表并不像图 5.2.45 那样连起来，就隐含着没有任何一个能把 P，Q 两个过程都包含进去的过程 R，同时也不能隐含，有任何两个转变的实例共享处于同一状态 B 的对象。而图 5.2.45 则表达了图 5.2.47 中两个分开的图所表达的内容，同时还有更多的信息。

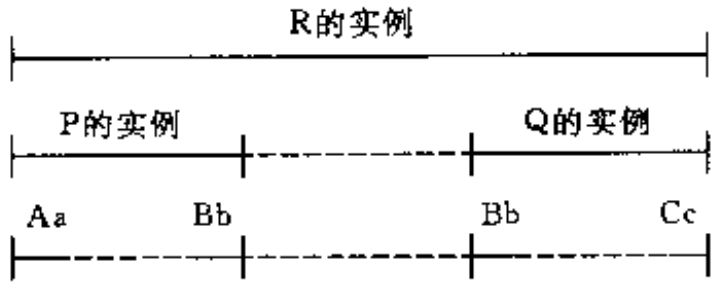


图 5.2.46　图 5.2.45 的缺省语义

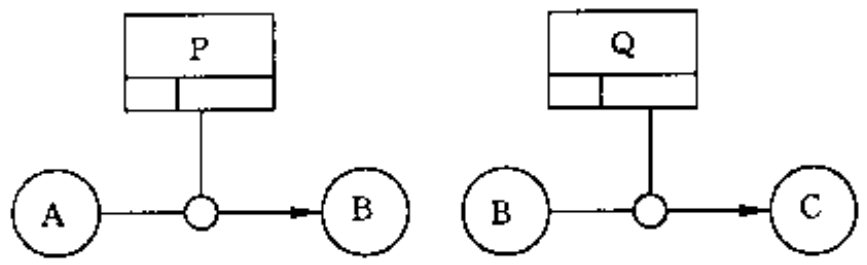


图 5.2.47　由图 5.2.45 归类的图表

2.1.5.5　状态分类图表

从“在转变图表中，把包含的对象状态看作是种类符号的可能的含义”这一点，可以得到一条重要的推论，即从这个角度，可把“是……的子种类”关系看作是“是……的状态”关系，如图 5.2.48 所示。

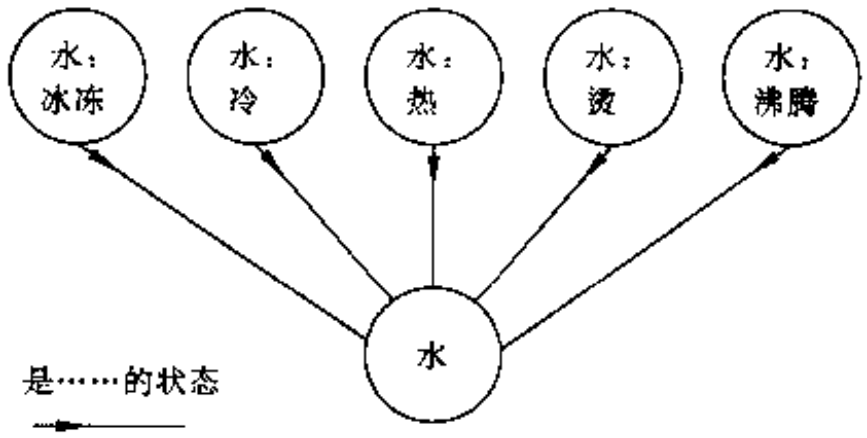


图 5.2.48　描述水状态的图表

在这里，我们用一个给定领域中水的各种可能状态表示水，而不是用水的各种子种类表示。这种图表，可以同标准的对象-状态图表结合起来，如图 5.2.49。

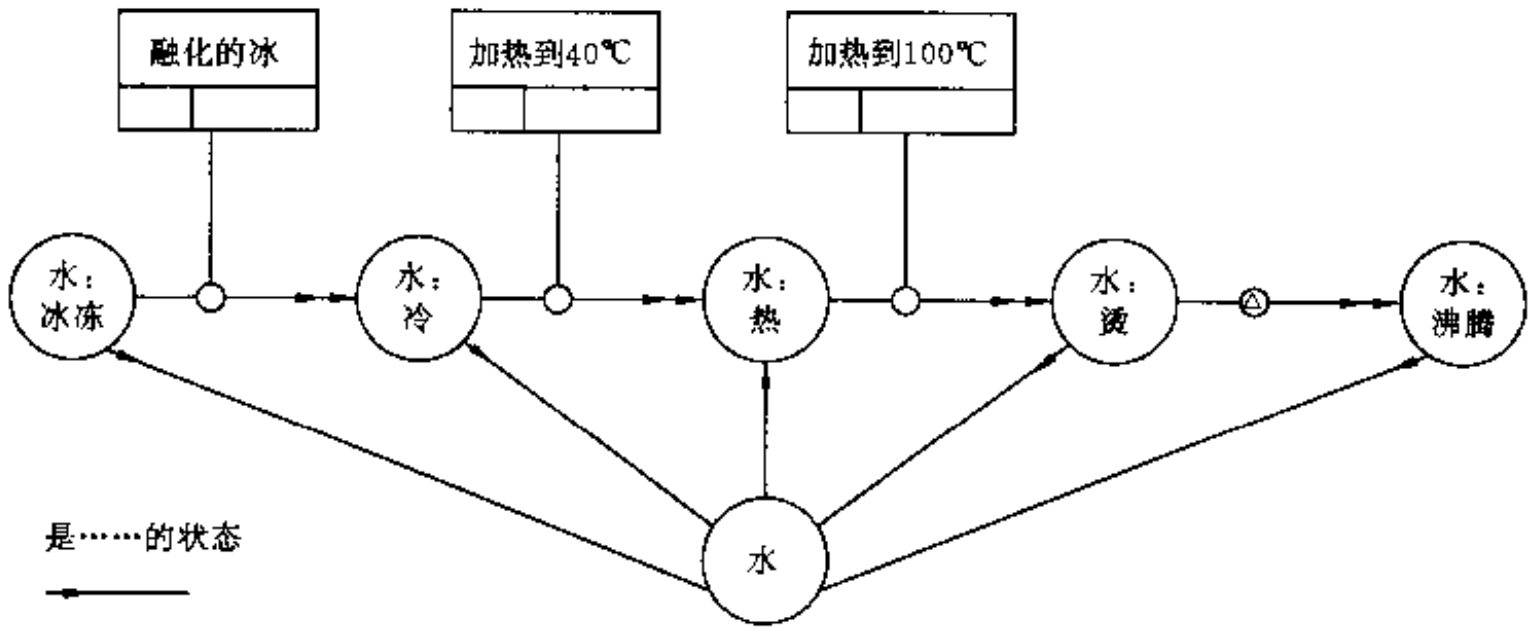


图 5.2.49　显示状态及状态转换的图表

2.1.5.6 状态合成图表

对象-状态图表，与基本 IDEF5 图表之间结合的最重要的一点，在于合成。通常的对象-状态合成图表，如图 5.2.50 所示。

状态合成图表的语义是状态转变图表语义的通用化。很自然地，图 5.2.49 中的对象-状态图表，表达了一个过程类型 P，在这个过程中，处于状态 $A_1 \cdots A_n$ 的对象 $a_1 \cdots a_n$，分别参与某个过程，而产生了处于状态 B 的对象 b。在状态转变图表中，每一个 a_i，都应在 p 的某开始阶段处于状态 A_i。而在状态合成图表中，并不一定要求每一个 a_i，在 p 的整个这个初始阶段处于状态 A_i，他们可以在 p 整个过程中，晃动着向这些状态(A_i)靠拢。只要每个对象 a_i，在 b 们变为 B 状态之前处于状态 A_i(从另一方面说，与图表的直观含义相反，处于相应状态的对象 a_i 们，不会组合到对象 b 中去)。图 5.2.50 的通用语义，在图5.2.51予以表示。

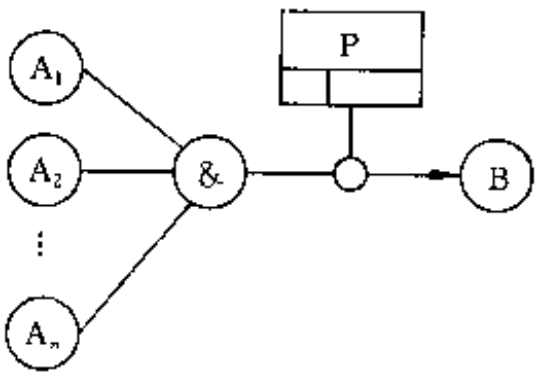


图 5.2.50 状态合成图表

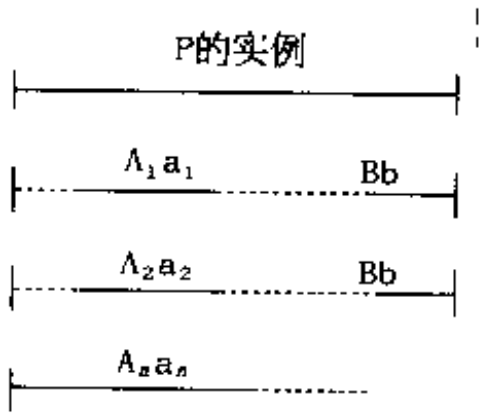


图 5.2.51 一个状态合成图表的通用语义

图 5.2.52，表述的过程开始时，有处于干状态的木头和含氧丰富的空气。处于这些状态的对象，经过一个燃烧过程，产生了灰(注意：这个例子中，与对象状态无关的种类标签，也可以用在对象-状态图表中；标签“灰”就属于这种情况，这样的用法，出现在那些给定过程中对象的状态与建模目的无关的情况下)。

将这个语义，与图 5.2.23 合成图表相结合，就产生一个“严格”状态合成图表的意图，这种图表的形成与一个合成图表相同，但在箭头上加上一个“是……的部分”的标签，如图 5.2.53 所示。

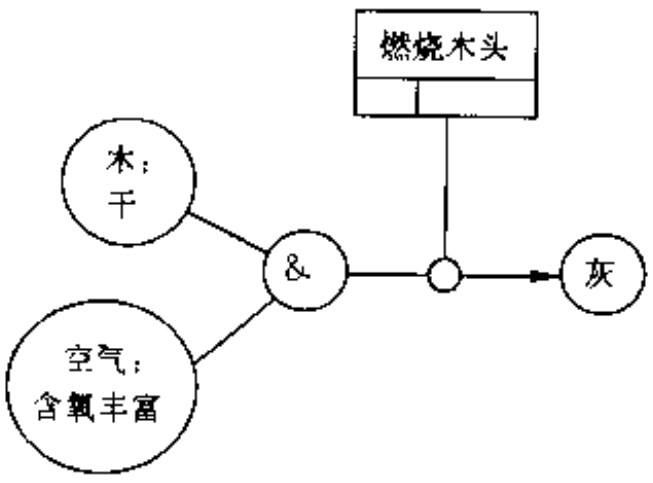


图 5.2.52 状态合成图表的一个例子

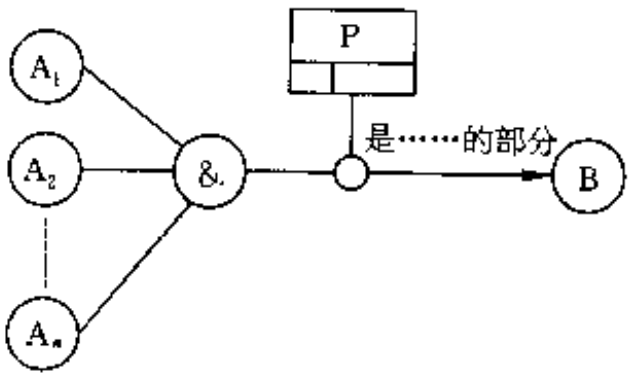


图 5.2.53 “严格”状态合成图表

一个严格合成图表的语义是:不仅是 $A_1 \cdots A_n$ 的实例 $a_1 \cdots a_n$ 分别参与了一个过程，从而生成了 B 的一个实例 b，而且这些对象都是 b 的一部分。这种意思可以通过图 5.2.54 来叙述，图 5.2.54 通过把过程信息，明显加入到图 5.2.23 的复杂合成图表中，来描述“圆珠笔”种类的成分结构。(需要强调的是，这种图表并不试图表达所有可能的圆珠笔的结构(例如某些圆珠笔没有缩回机构)，而是在假想领域中找到的某种笔的结构)。

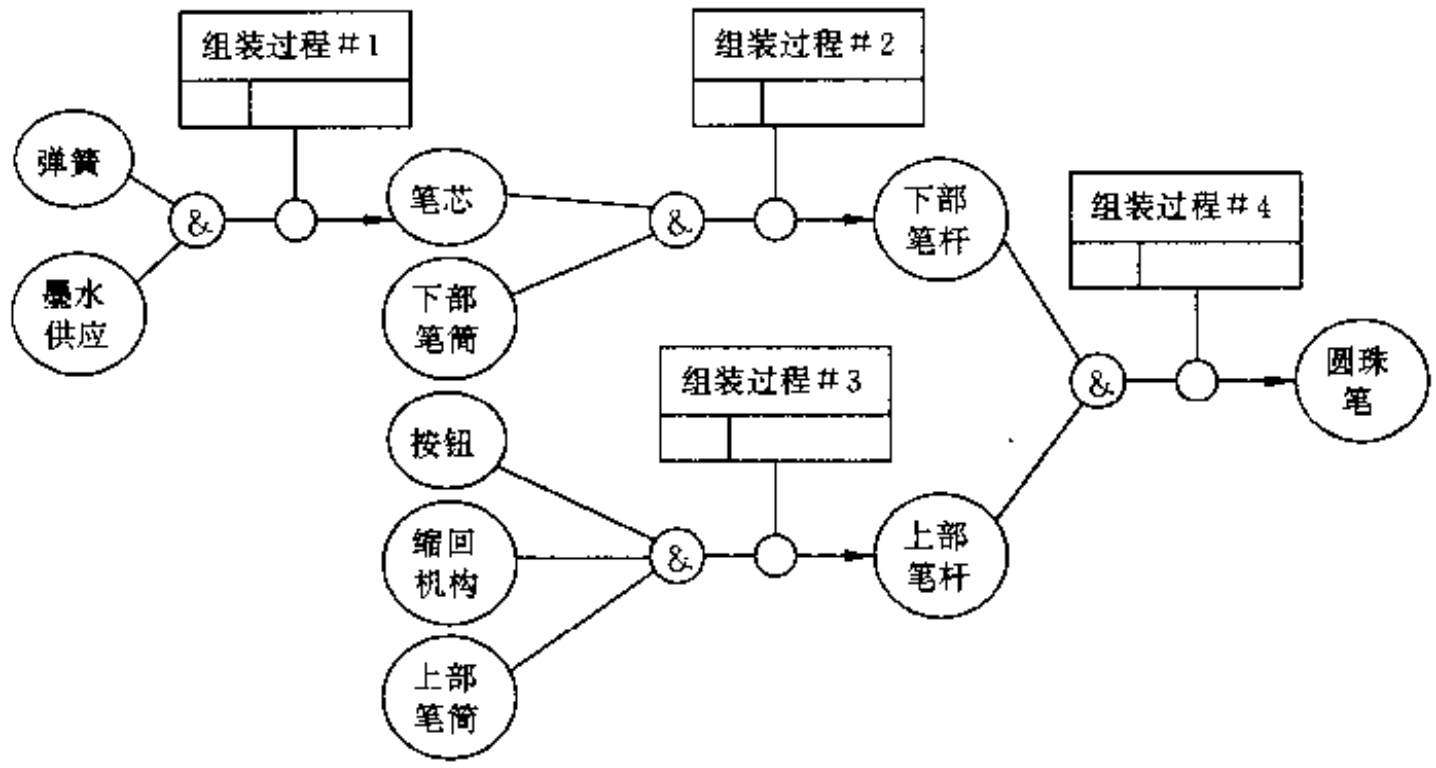


图 5.2.54　有关种类“圆珠笔”的复杂严格状态合成图表

在通常情况下，由一个合成过程产生的对象 b，同组成它的 $a_1 \cdots a_n$ 是有区别的。然而，“对象”的概念，在这里是非常灵活的，因而使得上述情况并不一定总是成立；一个没有侧视镜的车身与有它的车身是一样的。因此，合成图表中，同样允许强转变的表示，还是采用双箭头示于图 5.2.55。请注意，瞬间转变同样也可以用△来表示。

从对象-状态图表的角度来说，存在着一个明显的时间成分，这种对象状态“分解”的提法就有意义了。进一步地说，它就是状态合成的逆过程。它的表示方法如图 5.2.56 所示。

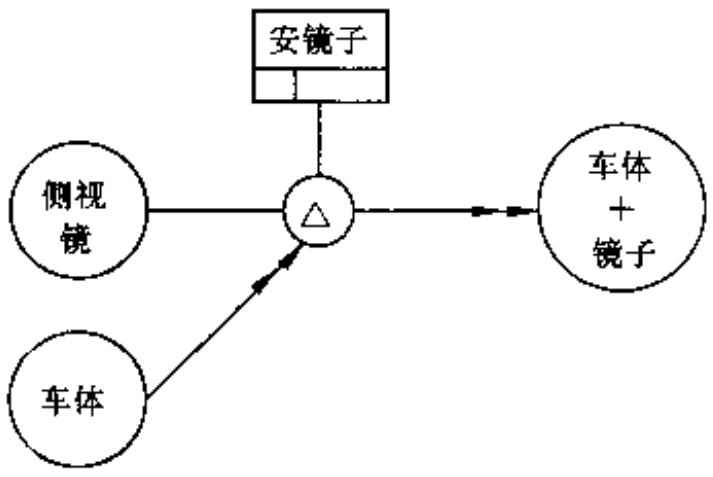


图 5.2.55　合成图表中强状态转变的例子

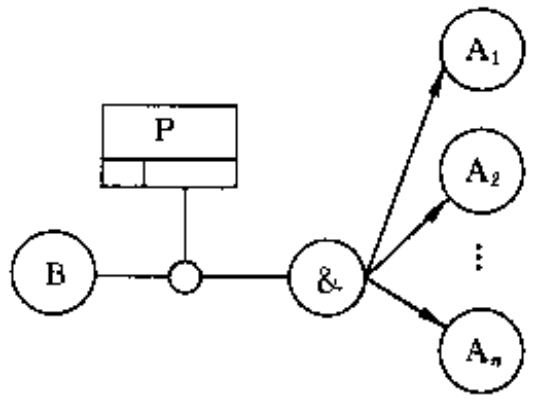


图 5.2.56　对象状态分解图表

状态分解图表的语义，正好是合成图表语义的逆。同样，这里允许使用双箭头以表示强转变。状态转变图表，可以认为是状态合成及分解图表在 $n=1$ 时的特例。下一个要介绍的对象-状态图表结构：分离的状态转变图表，它是一种逻辑图，表示一个状态可以转变为其他许多状态中的一个，如图 5.2.57 所示：

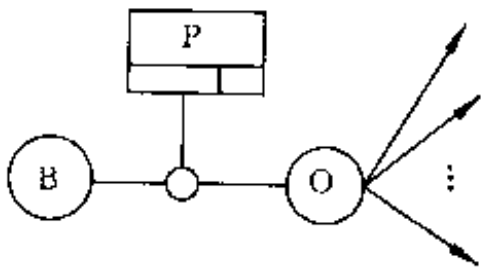


图 5.2.57　分离的状态转变图表

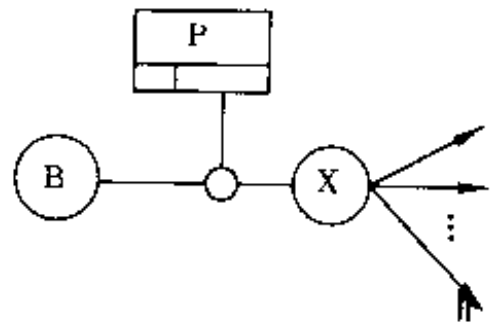


图 5.2.58　异或的分离状态转变图表

这里分离的类型是“或”的，它允许一个从 B 到任何后续状态的转变，可能超过一个状态。图 5.2.58 表示“异或”的分离，即只能转换到后续状态中的一个。

用同样的标记，引入“合取”(“与”)图表来表示一个从给定状态到所有的后续状态的转变，如图 5.2.59 所示。

在每种情况下，建模者可以把过程符号附在交汇点，也可以附在由交汇点伸出的每个箭头处(但不能两种都用)。在分离的情况下，将一个过程符号附在交汇点处，表示所示过程可以引出任何一种后续状态。在合取的情况下，将过程符号附在交汇点处，意味着所示过程带来了向所有的后续状态的转变(分解，因而只是一个合取状态转变图表的特例，在这里交汇点处引出的线，代表“是……的部分”关系)。如果在(“或”或“异或”)分离或合取图表中，有的转变是瞬间的，可妥善地应用 Δ 算符来表明。对于合成而言，这些逻辑图表有“逆”。具体地，当符号“ * ”是表示 O，X，或 & 时，图 5.2.60 的图也是一张 IDEF5 图表。

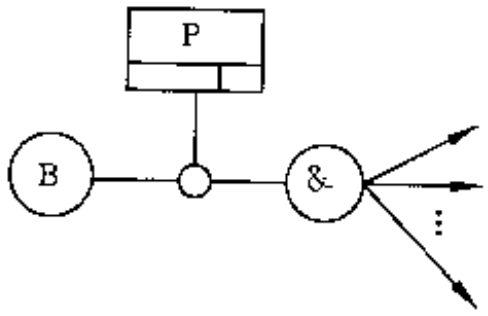


图 5.2.59　合取状态转变图表

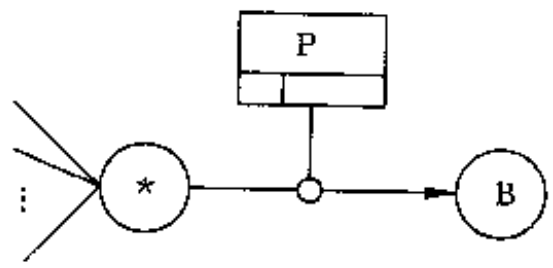


图 5.2.60　逆逻辑状态转变图表

上图中每种情况下的语义是上面相应图表语义的准确的逆(因此，特别是，合成图表将是合取图表逆的特例)。可以认为，一个对象状态能分别以不同方式转变，如图 5.2.61。

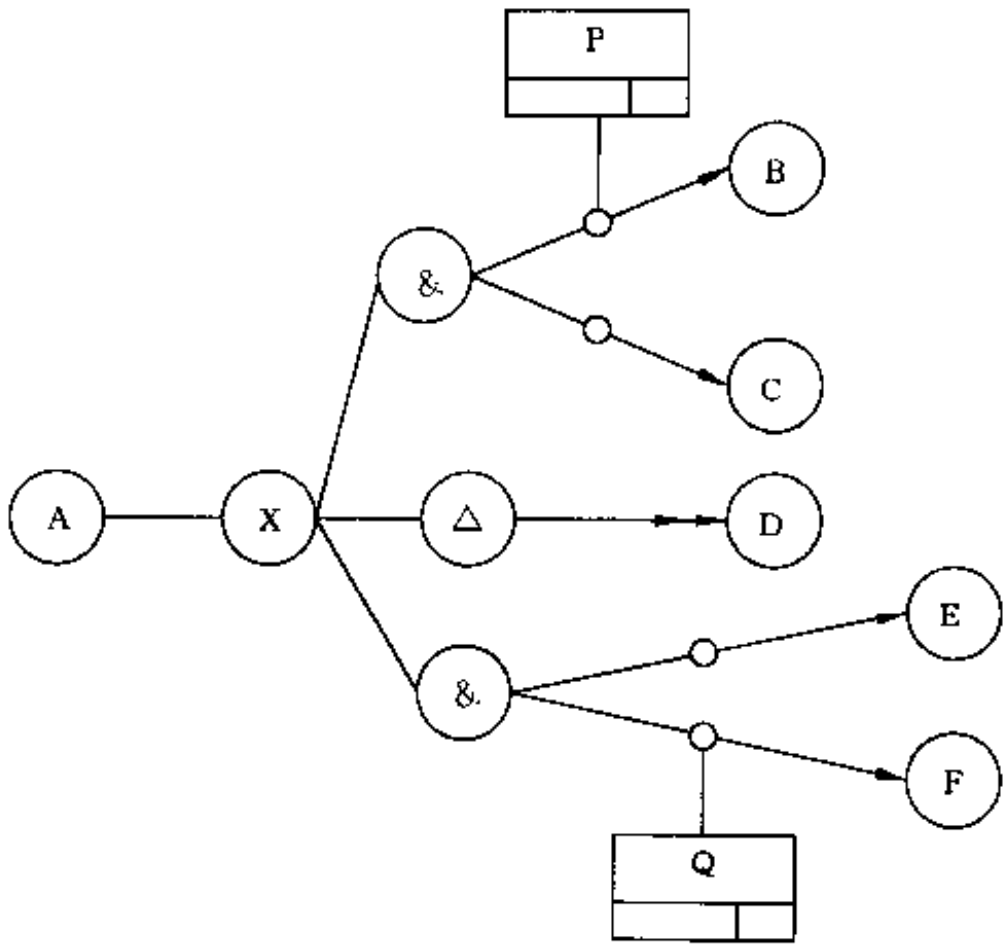


图 5.2.61　描述可能的复杂状态转变逻辑的本体论图表

2.1.5.7　隐匿对象状态信息

对于合成和分类图表而言，可以在对象状态图中隐匿信息。这就是说，出于某种目的，将一个有关给定对象的复杂状态转变信息，压缩到一个简单对象状态中去，是非常有用的。例如：一系列有关将水从冰冻加热到沸腾，这一过程中牵涉到的状态转变，如图5.2.62所示。

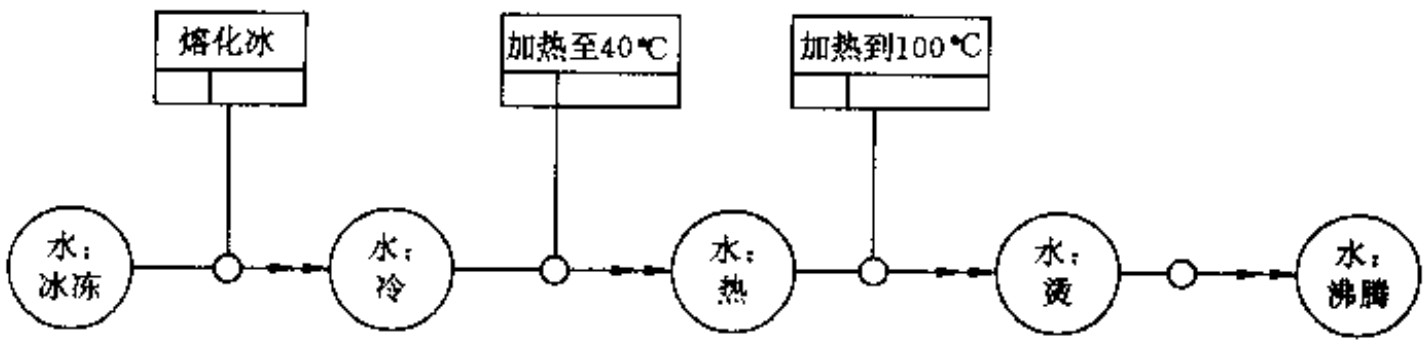


图 5.2.62　加热过程的状态转变

如果出于某种看法，认为从冰到沸水的中间转变是无关紧要的，这些转变可以被隐匿 在如图5.2.63描述的，称为“被加热的水”这样一个粗略的单状态中；在这里再次使用

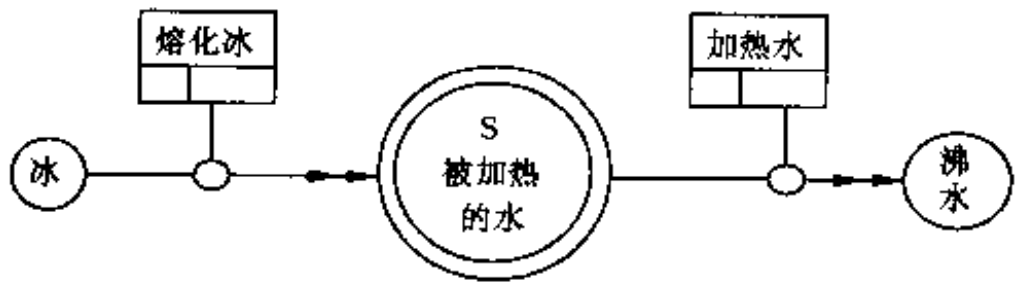


图 5.2.63　隐匿状态转变信息

双圆，并在其中加一个大写 S 以表示被隐匿的信息是状态转变信息。

从一个较细致的图表中，产生一个粗略的图表，这一过程并没有什么算法 。在本例中，种类符号“被加热的水”，可以被认为直接替代了图 5.2.62 中中间三个种类符号，以及它们之间的连接的“图表”。然而，图 5.2.62 中，瞬间转变图表就不得不被一般状态转变图所代替，就需要给附上的过程盒，找到一个合适的标签。这种变换的本质，是由所表示过程的本质所决定的，因此是一个非算法性的建模决定。

2.1.5.8 集成转换图和关系图

对象-状态图表，可以很自然地集成到普通的 IDEF5 的图中去。因为对象-状态图表结构上 是前述图表语言的扩展，人们可以用对象-状态图表结构，补充任何用原来结构构成的图表。具体例子见图 5.2.63；其中，合成信息和过程信息被集成到一个简单的通用状态合成图中。另一个把初始结构和对象-状态图表结构相集成的简单例子，如图 5.2.64 所示。

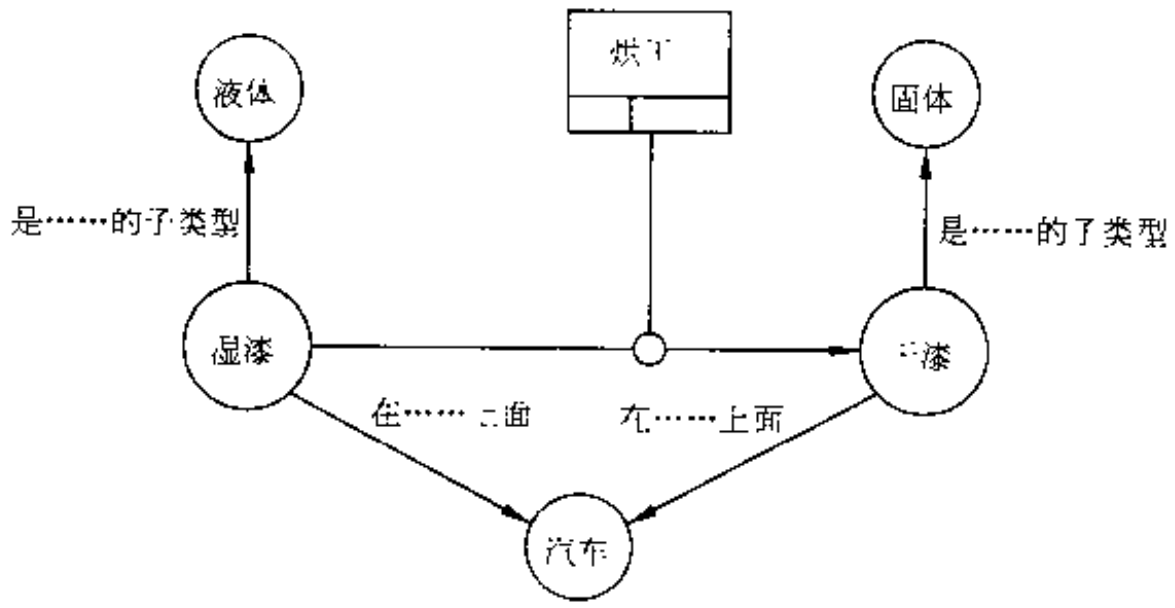


图 5.2.64 与对象-状态图表结构有关的 IDEF5 图表

在这个例子中，除了表示一个通过烘干过程，将湿漆对象到干漆对象的转换，关系符号被用来表示状态“湿漆”以及“干漆”同其他种类还有关，如箭头上的标签所示。这里，尽管关系符号与状态转变图表和转变点图表中的箭头相似，但这里却不会混淆，因为转变图表中的箭头总有一个圆圈。

2.2 IDEF5 细化描述语言

2.2.1 概貌

IDEF5 细化描述语言是一种用于直接获取本体论的结构化文本语言。语言的魅力，来自于其良好的理论基础以及丰富的表达结构。设计 IDEF5 细化描述语言的动机，来自在一个大的应用领域内获取及表达复杂本体论知识的需要。尽管图形媒介可帮助可视化分析相对简单的本体论知识，但表达能力的限制，却使它无法很方便地表达复杂信息。因此，显然需要一种有更强表达能力的媒介。

IDEF5 细化描述语言的用户，是与领域专家合作的知识工程师和系统分析员。它试图与 IDEF5 图表语言一同使用，两者将在各种不同的应用环境中互相补充。

- 系统分析员和领域专家用 IDEF5 图表语言记录一个初始本体论，对其加以分析，然后记入(更结构化的)细化描述语言的格式中去。
- 同时利用细化描述语言和图表语言来获取本体论，在整个获取过程中两种语言互为补充。
- 利用细化描述语言完成知识获取。图表语言主要用来通信以及获得后的可视化分析。

如前所述，细化描述语言可被用来获取整个本体论。因此，它的一些结构重复了图表语言的功能，以及与种类、关系、性质、原材料、源陈述、和本体论术语等相联系的说明形式。通常，利用细化描述语言进行本体论信息的获取，比用图表语言要难得多，它需要对语言有较好的理解。

2.2.2 节描述了语言中语句的正确表达，给出了每个结构用法的例子。如果要语言的详细介绍，请参考附录 A，它包括了语言的语法(本书从略)。

2.2.2 语言的描述

这种语言的句法，是用一个前缀注释和圆括号来界定表达式。语言的字母包括标准字母、数字及标点字符。IDEF5 细化描述语言的核心，是基于知识交换格式 KIF(knowledge interchange format)的，它可写出公理的表达式。在本小节中，术语“字符串”指任一有限字符序列和包含在双引号内的空格。在下面的小节里，语言中专用字(语言中有特别含义的)将保持英文写法。

2.2.2.1 常量，变量及操作符

“词”的概念即是 IDEF5 细化描述语言中的原语。语言中的一句话是由操作符、常量、和可能的变量组成。

1 常量

常量是一个既非变量又非操作符的词。常量可以看作是表示世界中对象的词。在 IDEF5 细化描述语言中，常量被分为如下几类：

(1) 本体论常量：一个本体论常量指一个描述本体论的词。

(2) 逻辑常量：一个逻辑常量指描述一个真值的词。

(3) 个体常量：一个个体常量指一个描述个体的词。

(4) 种类常量：一个种类常量指一个描述种类的词。

(5) 一阶关系常量：一个一阶关系常量是指一个描述一阶关系谓词(即存在与个体之间的关系)的词。

(6) 二阶关系常量：一个二阶关系常量是指描述二阶关系谓词(即存在于一阶关系之间的关系)的词。

(7) 关系常量：一个关系常量既可是一个一阶谓词常量，也可以是一个二阶谓词常量。

(8) 函数常量：一个函数常量指描述一个函数的词。

(9) 属性常量：一个描述属性的词。

(10) 特性常量：一个描述特性的词。

(11) 对象-状态常量：一个对象-状态常量是这样一种“种类:特性”形式的结构，其中种类为一个种类常量，特性为一个特性常量。

(12) 源-陈述常量：即描述一个源-陈述的词。

(13) 本体论术语常量：即描述一个本体论术语的词。

(14) 源常量：即描述一个源的词。

(15) 注释常量：即描述一个注释的词。

常量的例子有玛丽(一个个体常量)，汽车(一个种类常量)，传递的(一个特性常量)。在术语、定义、语句的描述中(见 2.2.2.2～2.2.2.4 小节)，术语“IDEF5 常量”被用来指除本体论常量及逻辑常量之外的任意常量。为了不混淆术语，可以在常量前加上一个本体论名(如“制造::技术”，在这里“制造”是一个本体论的名字，而“技术”则是作为制造本体论一部分的一个种类的名称)。

2 变量

一个变量可以看作一个常量的一位占有者。在 IDEF5 细化描述语言中，变量是以?或#开头的一个词。以? 开头的词代表 一个“个体变量”(即一个个体常量的一位占有者)。以#开头的词代表一个“谓词变量”(即一个谓词常量的一位占有者)。个体变量用来确定个体常量的数量，而谓词变量则用来确定谓词常量的数量。变量的例子有?x，? 个体，#p，和#谓词。

3 操作符

操作符是用来构成复杂表达式的专用词或字符。IDEF5 细化描述语言的操作符将在本节中列出。为了更好地理解这些操作符的用法，请参考 2.2.2.2 节和 2.2.2.3 节。

(1) 定义操作符：这些操作符用来形成定义。在语言中有五种定义操作符：“definition-relation”操作符用来形成二阶或三阶谓词；“define-function”操作符用来形成函数的定义；“define-individual”操作符用来形成个体的定义；最后，操作符：＝和:arg-type是用在定义中的操作符(详见 2.2.2.2 节及 2.2.2.3 节)。

(2) 项操作符：这些操作符用来构成复杂项。它们有六个操作符：listof，setof，if，cond，the 和 setofall。

(3) 语句操作符：这些操作符用来构成复杂语句。IDEF5 细化描述语言包括了通用逻辑操作符(＝，/＝，not，and，or，implies，equiv，forall 和 exists)，两个模态操作符(用于必要性的 nec 及用于可能性的 pos)，和许多 IDEF5 专用操作符。

2.2.2.2 词

IDEF5 细化描述语言支持三种类型表达式：词(terms)，定义(definitions)及语句(sentences)。词用来叙述对象。图 5.2.65 给出了一些词的例子。

词被分为以下几类

(1) IDEF5 常量及本体论常量：除了逻辑常量之外的所有常量均为词，因为它们描

述的是所讨论领域内的对象。

(2) 变量:变量是一个词,因为它们是常量的一位占有者。

(3) 属性词:属性词是一个"(属性常量 个体常量)"形式的表达式。它描述了应用于个体常量的属性常量所指出的属性的值。属性词的一个例子为如图 5.2.65 所示表达式(玛丽的年龄),其中"的年龄"是一个属性常量而"玛丽"是一个个体常量。如玛丽年龄为 25 岁,那么属性词值就为 25。

(4) 函数词:函数词是包含一个函数常量的表达式,这个函数常量还跟有一个或多个词。它描述了对应于用于这个参数"词"时功能常量所描述的函数值的对象。例如,函数词(2 的平方) 就指对象 4。

(5) 列表词:列表词包括一个 listof 操作符及一个或多个词。列表词指的对象就是由给定词所指对象的列表。图 5.2.65 的列表词的例子指出了包含对象蓝,红,白的列表词。

(6) 集合词:集合词包含一个 setof 操作符以及跟随其后的一个或多个词。一个集合词所指的对象,是相应词所指对象的集合。例如,集合词(setof 雇员,经理)就指包括雇员和经理的集合。

```
属性词(玛丽的年龄)
函数词(2 的平方)
列表词(listof 蓝,红,白)
集合词(setof 雇员,经理)
逻辑词 1.(if(<(重量? x)120)中号,大号)
       2.(cond ((<重量? x)80)小号)
               ((<重量? x)120)中号)
               ((>=重量? x)120)大号)
定量词 1.(the(? x)(结婚 保尔? x))
       2.(setofall(? x)(与(已婚? x)(=(年龄? x)30))
```

图 5.2.65 IDEF5 细化描述语言中词的例子

(7) 逻辑词:逻辑词指出了基于某种给定条件的几种对象中的一种。它可以由 if 操作符后跟一个句子及一或两个词组成,或由 cond 操作符及一系列有限的"语句-词"对组成。在这两种情况下,描述对象 O 的词前面的句子,代表了描述对象 O 的逻辑词要满足的条件。逻辑词的两个例子在图 5.2.65 中给出。那个词可能用来指对特定衣服尺码的小孩大小。第一个词指出,如果小孩的属性"重量"严格地小于 120 磅,那么词值就为常量"中号",否则应为常量"大号"。在这种形式的逻辑词中,第二词可被省略。在这种情况下,并没有为不满足条件的情况指定缺省词。第二个逻辑词可以分得更细。它指出如果小孩属性"重量"小于 80,那么词值为常量"小号",如果"重量"大于 80 小于 120,则词值为常量"中号",反之为常量"大号"。

(8) 定量词:定量词用于通过规定识别对象的条件来指定对象(或一组对象)。它可以用 the 或 setofall 操作符。一个用 the 操作符的定量词,指定一个满足给定条件的对象,如

图 5.2.65 的第一个例子所示，它指出了与保尔结婚的对象。一个用 setofall 操作符的定量词，指出了满足给定条件的对象集合。图 5.2.65 的第二个例子就是用 setofall 操作符的定量词，它指出了满足条件“30 岁且已婚”的对象的集合。

2.2.2.3 定义

定义是指用来定义一个个体、关系或函数常量的表达式类型。一个定义可以是完整的也可以是部分的。当一个常量可以被完整地定义时就使用完整定义，而一个部分定义则是在只有部分关于常量的信息时使用。例如，表达式(define-individual 原点 : =(listof 0 0))定义了常量“原点”是列表(0,0)，而表达式(define-function F(? x? y))则只规定了函数 F 有两个参数。一个常量只能有一个完整定义，但可有多个部分定义。

图 5.2.66 列出了三种定义类型。个体定义用来定义一个个体，在一个完整的个体定义中，个体名字及定义个体的词都必须被指定。在部分定义中，只要指定个体名字，还可以有选择地把一个句子包括在一个部分定义中来限制个体定义。

常量/定义	完整的	部分的
个体	(define-individual 原点 : =(listof 00))	(define-individual 太阳 (围绕它转动 地球 太阳))
函数	(define-function 整数部分(? x) : 变元类型((实整数)) : ==(the (? y)(and (整数? y) (=<? y ? x) (>? y(-? x 1))))	(define-function 持续时间(? x) : 变元类型((过程时间间隔)) (数(持续时间 ? x)))
关系	(define-relation 在……下面(? x? y) : =(在…… 上面 ? y ? x))	define-relation 是……的女儿(? x ? y) : 变元类型((女性 双亲)) (—>(是……的女儿 ? x ? y) (是……的孩子 ? x ? y)))

图 5.2.66 IDEF5 细化描述语言中的定义

一个函数定义用来定义一个函数。在一个完整的函数定义中，必须给出一个函数名，变量列表(表中的变量数规定了函数的元数)，还要给定一个定义函数的词。函数所需参数类型以及函数返回值类型也可以被指定，它们通过一个种类或对象-状态的列表来指定。每一个表格代表一个函数参数类型的合理集合。表格中最后一个元素指定了函数返回的参数类型。在一个部分函数定义中，只要指定函数名及变量列表，有时也需要加上函数所要的参数类型和一个限制函数定义的句子。

关系定义用来定义一个关系。一个完整的关系定义包括关系名、变量列表(表格中的变量数指定了关系的元数)、以及完整地定义关系的语句。有时还可专门说明处于一定关系中的实例的种类(以一个种类或对象状态列表作其元素的表格形式给出)。在一个部分关系定义中，只要指定关系名和变量列表。同时也可以有选择地加上处子一定关系中的实例的种类(以种类或对象列表的形式给出)以及一个限制关系定义的语句。

2.2.2.4 语句

IDEF5 细化描述语言中的语句，描述了本体论中常量的有关事实。在语言中有 7 种类型的语句，它们中的一部分如图 5.2.67 所示。

(1) 逻辑常量：逻辑常量是一个指定真值的词。

(2) 等式：等于是由运算符号＝以及两个词组成的表达式。

(3) 不等式：不等于是由符号/＝以及两个词组成的表达式。

(4) 关系语句：关系语句是由一个关系常量或带一个或几个词的函数常量组成的表达式。

(5) 逻辑语句：逻辑语句是由一个逻辑操作符及相应数量的句子组成。有 not，pos 或 nec 操作符的逻辑语句只包括一条参数语句，而其他逻辑语句则包括两个参数语句。

(6) 定量语句：有两种类型定量语句：一般定量语句及存在定量语句。一个一般定量语句由操作符 forall 后跟一个或几个封在括号中的变量及一条语句组成。这样一条语句，用来对某一类对象作一般性叙述。举个例子，图 5.2.67 中的一般定量语句指出所有 21 岁以上的人都是成年人。一条存在定量语句，有操作符 exists，后跟一个或几个封在括号中的变量以及一条语句组成。一条存在定量语句，可以宣称具有某种特性的对象的存在。

等式	(＝(保尔的年龄)30)
不等式	(/＝(卡车每哩耗油量)(轿车每哩耗油量))
关系语句	(与……结婚　苏珊)
逻辑语句	(and(＝(年龄 保尔)30)(not(与……结婚　苏珊)))
定量语句	(forall(? x)(＝>(and(人?)(>(年龄? x)21))(成人? x)))

图 5.2.67　IDEF5 细化描述语言的语句的例子

(7) IDEF5 特定语句：IDEF5 特定语句，是一条根据 IDEF5 方法的概念结构而引入常量的语句。其类型在 2.2.2.5 小节有详细介绍。

2.2.2.5 IDEF5 特定语句

有 7 种类型 IDEF5 特定语句，每一类对应于 IDEF5 方法中的一个概念。

1 本体论结构块

有 8 种本体论结构块，每一种都提供表达一个本体论某种信息的结构。这 8 种结构块将在本小节中加以解释，部分实例请见图 5.2.68。

(1) 本体论语句：这种语句用来发布一个本体论，本体论的发布由一个操作符 I5-ontology，在其后跟一个本体论常量或个体变量组成。

(2) 上下文语句：这种语句用来发布获取本体论的上下文。一个上下文的发布，由一个操作符 I5-ontology-context，在其后跟一个本体论常量或个体变量、以及一条字符串组成。

(3) 观点语句：这种语句用来发布描述本体论时采用的观点，观点的发布由一个操

作符 I5-ontology-viewpoint，在其后跟一个本体论常量或个体变量、以及一条字符串组成。

（4）目的语句：这种语句用来发布获取本体论的目的，目的发布由一个操作符 I5-ontology-purpose，在其后跟一个本体论常量或个体变量及一条字符串组成。

(I5-ontology 车间层控制系统)
(I 5-ontology-context 车间层控制系统，这个主体用来理解这样一个问题，即什么样的知识可以用来管理作业车间环境里的作业和制造资源。这个本体论还要描述车间层控制系统需要考虑的关键资源约束。
(I 5 ontology-purpose 车间层控制系统，这个本体论的目的是要指出哪些信息是在作业车间环境里车间层控制系统真正需要的。它可以帮助以后的分析员来决定，什么样的信息和知识对于成功地控制车间里的资源和作业是关键的。
(I 5-ontology-viewpoint 车间层控制系统，这个本体论是从整个精密齿轮制造设备公司的不同成本中心的成本中心调度员的角度来描述的。

图 5.2.68 IDEF5 细化描述语言本体论结构的例子

（5）项目语句：这种语句用来发布一个项目，获取本体论的工作属于该项目的一部分，项目发布由一个操作符 I5-ontology-project，在其后跟一个本体论常量或个体变量以及一条字符串组成。

（6）分析员语句：这种语句用来发布负责本体论获取工作的分析员，分析员发布，由一个操作符 I5-ontology-analyst，在其后跟一个本体论常量或个体变量以及一条字符串组成。

（7）评审员语句：这种类型用来发布负责评审本体论的人．评审员发布，由一个操作符 I5-ontology-reviewer，在其后跟一个本体论常量和一个个体变量以及一条字符串组成。

（8）“在本体论中”语句：这种语句用来发布某一常量是某一本体论的一部分，它由一个操作符 I5-in-ontology，在其后跟一个 IDEF5 常量或变量及一个本体论常量或个体变量组成。

2 种类结构块

有 9 种种类结构块，每一种都提供一种结构，以表达有关种类的信息。本小节将讨论其中 7 种结构块，并在图 5.2.69 中给出部分例子。

(I5-kind 钻孔)
(I5-kind-description 钻孔：钻孔是一种切割操作)
(I5-kind-property 钻孔：用钻头定义)
(I5-kind-attribute 钻孔：钻头类型)

图 5.2.69 IDEF5 细化描述语言中种类结构的例子

（1）种类语句：这种语句用来发布一个种类，它由一个操作符 I5-kind，在其后跟一个种类常量或谓词变量组成。图 5.2.69 的例子给了一条宣称“钻孔”是某一特定本体论中的种类语句。

（2）种类-特性语句：这种语句用来将特性与种类相联系。一条特性语句包括操作符 I5-kind-property，其后跟有一个种类常量或谓词变量，一个特性常量或谓词变量，以及有选择地包括专用词 defining 和“本质的”。图 5.2.69 的例子中给出特性“用钻头”（大概前面发布过）是种类“钻孔”的一个定义特性。

（3）种类-属性语句：这种语句用来将属性与一个种类相联系，它包括一个操作符 I5-kind-attribute，在其后跟有一个种类常量或谓词变量及一个属性常量或谓词变量。图 5.2.69 的种类-属性语句，发布了“钻头类型”是种类“钻孔”的一个属性。这里假定“钻头类型”在前面已经发布过是一个属性。

（4）种类-描述语句：这种语句用来将一个描述字符串与一个种类相联系，它包括一个操作符 I5-kind-description，后跟一个种类常量或谓词变量及一个字符串。

（5）种类-同义词语句：这种语句用来发布与种类常量同义的本体论-词，它包括一个操作符 I5-kind-synonyms，其后跟一个种类常量或谓词变量、及一个或多个本体论-词常量或被括号括起来的个体变量。

（6）引用-关系语句：这种语句用来发布一个与种类有关的关系，它包括一个操作符 I5-referenced-relations，其后跟有一个种类常量或谓词变量、及一个或多个关系常量或被括号括起来的谓词变量。

（7）子种类语句：这种语句用来发布某一种类为另一种类的子种类，它包括操作符 I5-subkind-of，其后跟一个种类常量或谓词变量、和另一个种类常量或谓词变量。

（8）对象-状态语句：这种语句用来发布对象状态，它包括一个操作符 I5-object-state，其后跟一个对象-状态常量或一个谓词变量。

（9）过程语句：这种语句用来发布一个过程，它包括一个操作符 I5-process，其后跟一个过程常量或谓词变量。

3 特性结构块

特性结构块用来提供有关特性的信息，共有 3 种特性结构块，图 5.2.70 给出一些例子。

```
(I5-individual CB-J27-S121)
(IS-property 底特律制造)
(IS-has-property CB-J27-S121 底特律制造)
(I5-is-of-kind CB-J27-S121 车身)
```

图 5.2.70 个体及特性结构的例子

（1）特性语句：特性语句用来发布一个特性，它包括操作符 I5-property，其后跟一个特性常量或谓词变量组成。图 5.2.70 中的特性语句发布了特性“底特律制造”。

（2）特性-描述语句：这语句可以被用来描述特性，它包括操作符 I5-property-

description，其后跟有一个特性常量或谓词变量及一个字符串。

（3）“有特性”语句：这条语句用来发布某个体具有某特性，它包括操作符 I5-has-property，其后跟有一个个体常量或变量、以及一个特性常量或谓词变量。图 5.2.70 所示“有特性”语句发布了特性“底特律制造”是个体 CB-J27-S121 的一个特性。

4 个体结构块

个体结构块用来发布有关个体的信息，在 IDEF5 细化描述语言中，共有 3 种个体结构块，图 5.2.70 给出了它们的例子。

（1）个体语句：这条语句用来发布个体，它由操作符 I5-individual，其后跟一个个体常量或变量组成，具体例子见图 5.2.70 发布 CB-J27-S121 为一个个体。

（2）个体-描述语句：这条语句用来将一个描述字符串与个体相联系，它由一个操作符 I5-individual-description，其后跟一个个体常量或变量及一条字符串组成。

（3）“是……种类”语句：这条语句用来发布某个体属于某一种类，它由操作符 I5-is-of-kind，其后跟一个个体常量或变量、及一个种类常量或谓词变量组成。图 5.2.70 中的例子宣称个体 CB-J27-S121 是属于种类“车身”的。

5 属性结构块

属性结构块用来提供有关属性的信息，共有 3 种类型属性结构块：

（1）属性语句：这条语句用来发布属性，它由操作符 I5-attribute，其后跟一个属性常量或谓词变量、及一个属性类型组成。一个属性类型可能是以下几种情况：一个列表词，一个集合词，或一个种类常量。属性语句的一个例子是，语句(I5-attribute 尺寸 (listof 实数))，它指出“尺寸”是类型“实数表格”的一个属性。

（2）属性-描述语句：这条语句用来描述一个属性，它由操作符 I5-attribute-description，其后跟一个属性常量或谓词变量以及一个字符串组成。

（3）属性-“应用于”语句：这条语句用来发布某条属性应用于某一个体，它由操作符 I5-attribute-applies-to，其后跟一个属性常量或谓词变量、以及一个个体常量或变量组成。这条语句的一个例子是，语句(I5-attribute-applies-to 尺寸 CB-J27-S121)，它表示“尺寸”是个体 CB-J27-S121 的一个属性。

6 关系结构块

关系结构块用来提供有关关系的信息，共有 4 种关系结构块，图 5.2.71 给出部分例子。

```
(I5-relation 是……部分)
(I5-relation-arity 是……部分函数为 2)
(I5-rel-arg-type 是……部分((火花塞 引擎)(引擎 汽车))
```

图 5.2.71 关系结构块的例子

（1）关系语句：这语句用来发布关系，它由一个操作符 I5-relation，其后跟一个关系常量或谓词变量组成。图 5.2.71 所示关系语句宣称“是……的部分”是一个关系。

（2）关系元语句：这语句用来发布关系的元数，它由操作符 I5-realtion-arity，其后跟

一个关系常量或谓词变量、及一个正整数或个体变量组成。图 5.2.71 中关系语句宣称“是……的部分”是一个二元关系。

(3) 关系-参数-类型语句：这条语句用来发布处于某一关系中各个体的种类，它由操作符 I5-rel-arg-type，其后跟一个关系常量或谓词变量、以及一个由种类列表和(或)谓词变量组成的表格组成。每一个种类列表代表关系的合理参数种类的集合。例如：图 5.2.71 给出的一条关系-参数-类型语句发布了种类“火花塞”的实例，可以与种类“引擎”的实例之间有“是……的部分”关系，同样的关系也存在于种类“引擎”的实例与种类“汽车”的实例之间。

(4) 关系描述语句：这条语句用来描述一个关系，它由操作符 I5-relation-description，其后跟一个关系常量或谓词变量和一条字符串组成。

7 函数结构块

函数结构块用来提供有关函数的信息，共有 4 种功能结构块，图 5.2.72 给出部分例子。

(1) 函数语句：用来发布函数，它包括一个操作符 I5-function，其后跟一个函数常量或谓词变量。图 5.2.72 的例子宣称“保修过期时间”为一条函数。

```
(I5-function 过期时间)
(I5-function-arity 过期时间元数为 1)
(I5-function-description 保修过期时间:这个函数的作用是,给出
一个汽车蓄电池上标的时间,就可以返回这个电池的保修过期
时间)
(I5-function-ary-type 保修过期时间(日期  日期))
```

图 5.2.72 函数结构块的例子

(2) 函数-元语句：用来发布函数的元(即函数所取的参数数)，它包括操作符 I5-function-arity，其后跟一个函数常量或谓词变量和一个正整数或个体变量。图 5.2.72 中的例子宣称，函数“保修过期时间”有 1 个参数(即元数为 1)。

(3) 函数-参数-类型语句：用来发布函数的参数类型，它由操作符 I5-fct-arg-type，其后跟一个函数常量或谓词变量、和一个种类列表和(或)谓词变量组成。图 5.2.72 的例于宣称，函数“保修过期时间”以日期的一个实例为参数，并返回一个日期的实例。

(4) 函数-描述语句：用来描述一个函数，它由操作符 I5-function-description，其后跟一个函数常量或谓词变量及一个字符串组成。

8 源结构块

源结构块用来提供有关原材料的信息，它共有 8 种结构块。图 5.2.73 给出部分例子。

(1) 源语句：用来发布原材料，它由操作符 I5-source，其后跟一个源常量或个体变量组成。图 5.2.73 中给出一个源语句的例子，语句宣称“张三”是一个原材料。

(2) 源描述语句：用来描述原材料，它由操作符 I5-source-description，其后跟一个源常量或一个个体变量和一条字符串组成。

(3)“从……收集”语句：用来发布原材料是从谁那里收集来的，它由操作符 I5-collected-from，其后跟一个源常量或个体变量和一条字符串组成。

```
(I5-source 张三)
(I5-source-purpose 张三:制造领域中的一些通用词的描述)
(I5-source-abstract 张三:计算机在制造过程的计划,控制,调度中所起的作用;强调硬件和软件系统设计和集成的计算机控制制造系统)
(I5-support-statement 张三:(s1 s2 s3))
```

图 5.2.73　源结构块的例子

(4)“由……收集”语句：用来发布原材料是由谁收集的，它由操作字符 I5-collected-by，其后跟一个源常量或个体变量以及一条字符串组成。

(5) 源-摘要语句：用来给出原材料的摘要，它由操作符 I5-source-abstract，其后跟一个源常量或个体变量和一条字符串组成。图 5.2.73 给出了一条源摘要语句的例子，它指定了原材料“张三”的源摘要。

(6) 源目的语句：用来发布一个原材料的目的，它由操作符 I5-source-purpose，其后跟一个源常量或个体变量和一条字符串组成。图 5.2.73 给出的例子指出了用原材料“张三”的目的。

(7) 支持-本体论-词语句：用来发布被某一原材料支持的本体论-词，它包括操作符 I5-support-ontology- term，其后跟一个源常量或个体变量、和 一个或多个本体论-词或括在括号中的个体变量。

(8) 有-支持-源语句：用来发布 一个 IDEF5 常量是有一个或多个原材料支持的，它包括操作符 I5-has-supporting-sources，其后跟 一个 IDEF5 常量或变量和一个或多个源常量或括在括号中的个体变量。

9　源-陈述结构块

源-陈述结构块用来提供有关源-陈述的信息，有 4 种这样的结构块，图 5.2.74 给出了部分例子。

```
(I5-source-statement 设计-问题)
(I5-source-statement-description 设计-问题:设计问题应当考虑最适当的物料去除过程,它由体积、尺寸、精度、完成情况、材料和成本等约束条件决定。)
(I5-status 设计-问题　活动的或初始的)
```

图 5.2.74　源陈述结构块的例子

(1) 源-陈述语句：用来发布一个源陈述，它包括了操作符 I5-source-statement，其后跟一个源-陈述常量或个体变量。图 5.2.74 的例子宣称“设计-问题”是一个源-陈述。

(2) 源-陈述-描述：用来提供与一个源-陈述相关联的文本和(或)提供源-陈述的描述。它由操作符 I5-source-statement-description，其后跟一个源-陈述常量或个体变量和一条字符串组成。图 5.2.74 中给出的相应例子提供了与源-陈述“设计-问题”相关联的文本。

(3) 状态语句：用来发布源-陈述的状态，它由操作符 I5-status，其后跟一个源-陈述常量或一个个体变量以及以下操作符之一：活动的-初始的(active-original)，活动的-派生的(active-derived)，退下的-派生的(retired-derived)。图 5.2.74 给的相应例子表示源-陈述“设计-问题”的状态是活动的和初始的。

(4)“有-初始的-陈述”语句：用来发布一个派生的源-陈述的初始源-陈述。它包括操作符 I5-has-original-statement，其后跟一个源-陈述常量或个体变量以及一个源-陈述常量或个体变量。

10 本体论-词结构块

本体论-词结构块，用来提供有关本体论-词的信息，共有 3 种结构块，图 5.2.75 给出了部分描述。

(1) 本体论-词语句：用来定义一个本体论-词，它包括操作符 I5-ontology-term，其后跟一个本体论-词常量或一个个体变量。图 5.2.75 给出的相应例子给出“易损坏工具”是一个本体论-词。

(2) 本体论-词描述语句：用来描述一个本体论-词，它由操作符 I5-ontology-term-description，其后跟一个本体论-词常量或个体变量及一个字符串组成。图 5.2.75 给出的例子，描述了本体论-词“易损坏工具”。

(3)“用在……陈述中”语句：用来发布本体论-词用到的陈述句，它由操作符 I5-used-in-statements，其后跟一个本体论-词常量或个体变量及一个或多个源-陈述常量或括在括号中的个体变量组成。图 5.2.75 给出的相应例子表示本体论-词“易损坏工具”用在源陈述 SS39 中。

```
(I5-ontology-term 易损坏工具)
(I5-ontology-term-description 易损坏工具:由于磨损因而只具有有限使用生命的工具)
(I5-used-in-statements 易损坏工具(SS39))
```

图 5.2.75 本体论-词结构块的例子

11 标注结构块

标注结构块用来提供有关标注的信息，下面介绍它的 3 种结构块：

(1) 标注语句：用来发布一个标注，它由一个操作符 I5-note，其后跟一个标注常量或个体变量组成。

(2) 标注-描述语句：用来描述一条标注，它由操作符 I5-note-description，其后跟一个标注常量或个体变量以及一个字符串组成。

(3)“有标注”语句：用来将一标注“附在”一个 IDEF5 常量上。它包括操作符 I5-has-

note，其后跟一个 IDEF5 常量或变量以及一个标注常量或个体变量。

2.2.2.6 预定义关系

这一小节所列举的关系常量，是在语言中预定义好的，因此可在语句中直接使用。

（1）是……的部分(part-of)：这个关系常量指“是……的部分”关系，它被部分定义为一个 2 元的一阶关系，并以两个种类的实例作为参数。

（2）转变为(transitions-to)：这个关系常量描述了“转变为”的关系，它在基本状态转变图表中也出现过。在那里，它被表示为一个无过程符号的对象-状态转变连接。这里，它被部分定义为一个 2 元的一阶关系，以两个种类的实例为参数。

（3）当……时转变(transitions-during)：这个关系常量指出了“当……时转变”关系，它同样在基本状态转变图中出现过。在那里，它被定义为一个与过程符号相连的状态转变连接。在这里，它被部分定义为一个 3 元的一阶关系，并以两个种类实例和一个过程实例为参数。

（4）瞬间转变为(inst-transitions-to)：这个关系常量指出了“瞬间转变为”这个关系，它在基本状态转变图中，被表达成一个不与过程符号相连的瞬间状态转变连接。这里，它被定义为一个 2 元一阶关系。

（5）当……时瞬间转变(inst-transitions-during)：这个关系常量指出了“当……时瞬间转变”的关系，在基本状态转变图中，它被表达成一个与过程符号相连的瞬间状态转变连接。这里，它被部分定义为一个 3 元的一阶关系，以两个种类实例和一个过程实例为参数。

（6）强转变为(s-transitions-to)：这个关系常量指出了“强转变为”关系，它在强转变图表中，被表示为一个没有与过程符号相连的强转变连接。这里，它被部分定义为一个 2 元的一阶关系，并以两个种类实例为参数。

（7）当……时强转变(s-transitions-during)：这个关系常量指出了“当……时强转变”关系，在基本状态转变图中，它被表达成一个与过程符号相连的强状态转变连接。这里，它被部分定义为一个 3 元的一阶关系，并以两个种类实例及一个过程实例为参数。

（8）瞬间强转变为(inst-s-transitions-to)：这个关系常量指出了“瞬间强转变为”关系。它被部分定义为一个 2 元的一阶关系，并以两个种类实例为参数。

（9）当……时瞬间强转变(inst-s-transitions-during)：这个关系常量指出了“当……时瞬间强转变”关系，它被部分定义为一个 3 元的一阶关系，并以两个种类实例和一个过程实例为参数。

（10）处于-状态-贯穿始终(in-state-throughout)：这个关系常量用来表明，一个个体处于某个给定状态的情况，在某过程中贯穿始终。它被部分定义为一个 2 元的一阶关系，并以一个种类实例和一个过程实例为参数。

后　记

本书第一篇第1章，曾经列出了IDEF家族的全部名称，从IDEF0到IDEF14。但是IDEF6-14还没有完整的手册，我们在本书的末尾，分别将IDEF9，IDEF6，IDEF14和IDEF8的基本概念向读者作一简介；供读者了解这一研究方向上最新的动向和研究热点。

1　IDEF9事务约束揭示方法

政策、规则、约定、程序、合同、协议、规定、以及社会的和物理的法则，都是给企业下定义的构造成分。他们是为构成一个系统而建立的人、信息、物料和机器之间关系的各种机制。我们把这一切就称之为“约束”。如果你把一个企业看作一台机器，那么约束就形成了机器的体系结构和定义其行为的编程语言。如果你把企业看作一个有机体，它们就形成了该有机体的控制结构，从遗传因子层通过自治激励响应层到认识行为层。

IDEF9事务约束揭示方法，是用来帮助揭示和分析一个事务系统中的约束的。开发IDEF9的原始动机，是因为一般来说，构成企业的约束总是没有很好定义的，对于“到底有什么约束”、“它们是如何关联的”等问题的了解，往往都是一知半解、互不关联、甚至是完全不知道的。然而，这些也不必太担心，正如一个活的有机体并不需要知道操纵着其行为的遗传因子和自治约束，不必清楚地知道构成这系统的结合剂，照样可以活得很好。然而，假如期望要以一种可预见的方式来改造这一事务的话，就需要掌握这些约束的知识，就像遗传学知识对遗传工程师那样关键。

约束是人和自然赖以组成系统的机制。约束启动、使能、控制和限制着对象和媒体的行为去完成系统的目标。假如我们要改变系统的行为(例如，改进性能、提高效率)，我们就需要知道有关的约束。IDEF9使得这种对约束的揭示，和把相关约束项组织起来，形成系统的映射变得更容易。一旦，这些约束被整理出来，它们就能被系统地检验、调节或替换，以改进系统的性能。约束起着两种作用，既是结合剂又是系统形成的原理。也就是，一组相关约束就组成了系统为何产生这样行为的说明。从这个角度来说，IDEF9为事务工程师提供了一种反向工程的工具。它可以帮助他揭示现有系统设计背后的“逻辑”，也提供了说明未来系统逻辑的机制。

IDEF9事务约束揭示方法的主要概念有：

约束(constraint)	背景 (context)
证据(evidence)	约束的效用(effect(s) of a constraint)
征兆(symptom)	系统(system)
约束理论(rationale for the constraint)	目的(goal)

这里，主要讲一下约束。在IDEF9方法中，约束被定义为在特定背景下存在或发生作用的联系。约束采用约束陈述(constraint statement)来表示。约束陈述的例子有：

(1) 防止债务；

(2) 过程中投入量最小化；

(3) 成本回收最大化；

(4) 头等旅客、带小孩的家长、残疾人先登机；

(5) 座位在后排的旅客先登机；

(6) 所有项目作出一个最后报告；

(7) 合同超过10 000 000美元的项目需要附加一个成本报告；

(8) 任何人驾车必须有有效驾照；

(9) 只有得到允许的入员方可进入安全区；

(10) 一个小组的领导必须是管理层的一员；

等。

IDEF9模型的开发，首先就要建立起约束开发工作的目标和范围。约束开发工作是一个发展过程。通过它，候选约束被识别、证实和细化。通常，运用IDEF9来发现和整理约束，将反复运用以下6个步骤：

(1) 收集：观察和找到约束的证据；

(2) 分类：分为背景、对象、对象类型、属性和联系；

(3) 假设：从获得的数据和证据中选定候选约束；

(4) 证明：归纳或收集例子以决定哪个候选约束被证明确实是约束；

(5) 检查：包括专家在内去检查分析结论的正确性；

(6) 细化：过滤、改进、校正，对约束的特征增加细节。

整个数据收集和分析过程中应注意的事项，与其他IDEF方法类似。

2 IDEF6设计原理获取方法

技术的进展，一方面使产品的生命周期不断缩短，另一方面又导致了产生那种其寿命能延长到几十年的产品。信息系统，也要从孤立的、寿命相对说来较短的面向应用的系统，演变为大规模的，必须在很长时期内为其用户服务的分布式系统。这种很长寿命的信息系统的维护，就需要明确地获取并存储其设计原理。

设计原理，一般是以非结构化的文本形式存在的。此外，难于找到所需要的有关信息，缺乏一种结构化方法来组织和提供设计原理获取的完整性判据，都使得有些重要信息没有用文件记载下来。

不像一般设计方法，那只是为了记载一项设计“是什么”；新提出的设计原理获取方法要掌握“为什么一项设计是这么做而不是那么做的”，还要掌握“如何”达到其最终的设计配置。为此目的，设计说明(design specification)只是要掌握一项设计“是什么”，而设计原理(design rationale)则是指“为什么”、“为什么不”和“如何”使一项设计达到最终配置。设计历史(design history)则是指实现设计所用到的按实践顺序的步骤。

IDEF6企图成为一种具有获取信息系统设计原理的表达能力的方法，以及与此原理

相应的设计模型和最终系统文档。这样，IDEF6 就企图掌握那些导致和包含在最终设计中的决策的逻辑；明确地获取设计原理是为了要避免过去的错误，提供一种直接工具去确定将要提出的设计更改的影响，加强对目标和假设的明确阐述，并有助于对最终系统说明的交流。明确地获取一个设计者如何会选择或采纳一种特有的设计策略、或者企业级信息系统的特征的动机，对于该系统在全寿命周期内的维护是至关重要的。

在 IDEF6 模型开发中，基本原理的获取过程包括分解、分类/说明、装配、仿真/运行和再分解等活动；通常会反复迭代进行。在开发设计的仿真/运行过程中，常可分为 2 个阶段：阶段 1 描述和标识问题，阶段 2 开发求解的策略。IDEF6 还开发了其本身的模型语言。

3 IDEF14 网络设计方法

IDEF14 是一种支持计算机网络设计的方法，它帮助网络设计人员获取需求，说明网络成份，获取已有网络的配置以及对设计进行分析，它也支持管理决策和设计，来优化工程经济。此外，IDEF14 表述了并存储了网络设计的设计原理。IDEF14 以一种可改编的程序支持这些活动，这一程序可建立网络配置、排队、可靠性和成本等模型。网络配置模型以图表形式，显示了网络成分的拓扑结构、配置、说明和属性。利用这一模型，可产生排队、可靠性和成本模型，对这 3 个模型的分析，可作为网络选择决策过程的输入。

IDEF14 提供了一种可靠的程序，这种程序支持数据收集、多重设计创建和评价，以及最后的设计选定。网络设计过程，包含着对当前(AS-IS)网络的建模，也支持获取当前和未来的负荷，以便形成未来(TO-BE)网络设计。对每一个设计方案进行多方面分析，并记录其设计原理。该方法也为开发和扩展网络技术资料库，提供表示最新的和未来的计算机和通讯网络技术的描述。为此，IDEF14 是用来与网络仿真工具一起，以帮助网络体系结构的设计的；这个体系结构可以用可变的技术配置来测试和分析。

IDEF14 有 14 个基本概念：

节点(node)	子网(subnetwork)
联接(link)	拓扑结构(topology)
工作量(workload)	协议(protocol)
排队论(queuing theory)	可靠性分析(reliability analysis)
服务质量(quality of service)	成本分析(cost analysis)
决策支持(decision support)	设计原理(design rationale)
网络约束(network constraints)	元件库(component libraries)

其文本说明中，对这些概念有详细的定义和解释。

网络设计工作是一个发展过程，从为现有的网络建模、收集需求，到新网络的设计、制作、评估、验证、建档和改进，可以运用 IDEF14 来完成整个过程。一般说，需要反复运用以下工作步骤：

(1) 收集：获得关于现存网络的内容、需求、及现有技术的知识；

(2) 分类：分清网络概念(元类型)，如节点类型，联接类型，拓扑结构类型，以及类型的元素；

(3) 假定：从获得的数据和其他依据中选定候选网络设计；

(4) 验证：确认对当前网络的建模是正确的，设计中的网络约束已被满足；

(5) 分析：作出网络设计的排队模型、可靠性模型、和成本模型，验证模型，并分析模型；

(6) 审查：请领域专家对设计的结论进行测试和审查，以验证他们的结论；

(7) 改进：对网络设计删节、改进、修正和增加细节。

IDEF14 的各步工作是一种思考的模式，而不是单纯的顺序步骤。设计者不应该按照严格的顺序去工作，或是在这些工作开始或结束时，必须按照项目的术语去组织工作。可以说，术语反映的是一段特定时间内哪一个工作是主要的，但整个工作过程是要迭代进行的。

4 IDEF8 人-系统交连设计方法

IDEF8 人-系统交连设计方法用来产生高质量的交连设计，这种交连是确实应该出现在用户及其所操作的系统之间的，使用户能更方便和舒适地应用他(她)所面对的系统。IDEF8 不是一种图形用户接口设计方法，它不是用来描述屏幕布置或者按钮或窗口的尺寸，它是用来帮助系统开发人员掌握存在于系统及其用户间的交连的。系统是那些执行一个以上功能以完成一个特定目的的对象的集合。系统不一定是一台计算机或一个计算机程序。

在 IDEF8 方法中，对人-系统交连设计了 3 个层次的说明。第 1 层定义了系统运行的原则，产生了一组全系统过程的模型和文本说明；第 2 层说明了系统应用中的角色为中心的场景；第 3 层 IDEF8 设计是为细化人-系统设计用的，在这一层，IDEF8 提供了一个比喻的资料库，用来帮助用户和设计人员用一些其行为更为熟悉一些的对象来描述自己期望的行为。比喻根据熟悉的、具体的对象和经验提供一个抽象概念的模型。例如一个电灯开关，可用来说明包含两种选择的用户交连关系。在这一设计层次的产品中，人-系统交连模型就是其中之一，它可用于测试用户需求、制定用户接口策略(例如选取偏爱的输入和反馈装置)等。一旦验证了，IDEF8 应用的产品就被系统开发人员用来进行实施。

许多 IDEF8 的语言构造块都直接取自 IDEF3 过程描述获取方法，因为 IDEF8 需要一种机制，在多种抽象和细节的层次上获取和组织过程信息。特殊的语言扩展部分，把性质上是指令性的 IDEF8 设计模型，与 IDEF3 的描述式表达方式相区别。

IDEF8 方法的根本目的，是要对把人处于闭环中的系统，作出一些更好的设计，来实现以较少的时间和合理的成本完成更高质量的实施。IDEF8 系统能帮助用户产生很好的人-系统交连设计，进而设计出更高质量的系统，主要依靠：(1)使集中于用户的数据收集更容易；(2)使直接用户能参与设计活动；(3)利用模型和原型把力量集中于设计的早期验证；(4)通过设计过程促进更高效率的交连。

参 考 文 献

1. 陈禹六等编著. IDEF0 及 IDEF1X—— 复杂系统通用的设计分析方法. 北京:电子工业出版社,1991
2. Mayer R J ed. "IDEF0 Function Modeling", KBSI, 1990
3. Mayer R J ed. "IDEF1X Data Modeling", KBSI, 1990
4. Mayer R J et al, "IDEF3 process description capture method report", AL-TR-1992-0057, KBSI, 1992
5. Mayer R J et al, "IDEF4 object-oriented design method report", AL-TR-1992-0056, KBSI, 1992
6. Benjamin P C et al, "IDEF5 report", KBSI, 1994
7. Mayer R J et al, "Information Integration for Concurrent Engineering (IICE) Compendium of methods report", KBSI, 1995
8. 陈禹六, 谢斌, 董亚男 编著. 计算机集成制造(CIM)系统设计和实施方法论. 北京:清华大学出版社,1996
9. 李伯虎主编. 计算机集成制造系统(CIMS)约定、标准与实施指南. 兵器工业出版社,1994.
10. 《航空制造工程手册》总编委会主编. 航空制造工程手册 —— 计算机辅助制造工程(分册). 航空工业出版社,1995

[General Information]
书名=IDEF建模分析和设计方法
作者=陈禹六编
页数=297
SS号=10848131
出版日期=1999年05月第1版