第10章　需求的图形化分析

“化学制品跟踪系统”的项目开发组正在进行第一次软件需求规格说明的评审。参加者有Dave（项目经理），Lori （需求分析者），Helen(高级程序员)，Ramesh (测试专家)，Tim(化学制品的产品代表者)，还有Roxanne(化学制品仓库的产品代表者)。Tim开始说：“我阅读过整个软件需求规格说明。大部分都符合我的需求，但是有几个部分我很难同意。我不能确信在化学制品请求过程中，我们是否确定了这些步骤。”

Ramesh又补充说：“当一个请求通过系统时，我很难想象用于覆盖该请求状态变化的所有测试用例。我发现许多关于状态变化的需求散布在整个软件需求规格说明中，但我无法确定是否有一些需求遗漏了或存在不一致性。”

Roxanne有一个类似的问题。“当我阅读了如何真正请求一种化学药品时，我感到困惑”，她说，“单个需求是能感觉到的，但我难以想像我所要完成的步骤顺序。”

在各评审员提出其它相关的问题后，Lori做出了总结：“看来软件需求规格说明似乎没有完全告诉我们对于理解系统所需的各个方面，也不能确保我们没有错过一个需求或不犯任何错误。我将画一些图来帮助我们想像这些需求，并看一下能否澄清这些问题域。谢谢你们的反馈意见。”

根据在需求方面的权威Alan Davis的见解，仅仅单一来看需求并不能提供对需求的完全理解（Davis 1995），你需要把用文本表示的需求和用图形表示的需求结合起来，绘制出对预期系统的完整描述，并可帮助你检测不一致性、模糊性、错误和遗漏。这些图形化的表示或者分析模型可以增强你对系统需求的理解。在项目的参与者之间，对于某些类型的信息，图形化交互比文本交互更高效，并且可以在不同的开发组成员之间扫清语言和词汇上的障碍。本章将提供对需求建模技术的简要概述，在我看来，这些技术有助于理解用户的业务问题和软件需求。

10.1　需求建模

许多年前，当我开始绘制分析模型时，我希望找到一种技术，可以把所有的内容都包容进一个完整的需求描述中。最终我得出一个结论：不存在一个包罗万象的图。早期的结构化系统分析的目标是用比叙述文本更正式的图形表示来替换整个分类功能规格说明(DeMarco 1979)。然而，经验告诉我们：分析模型应该增强自然语言的需求规格说明，而不是替换之(Davis 1995)。

需求的图形化表示的模型包括数据流图（ DFD）、实体关系图（ ERD）、状态转化图（STD）、对话图和类图。还有一些非常规的建模方法也是有价值的。一个项目开发组利用项目规划工具为嵌入式软件产品成功地画出时间需求，其工作在毫秒级，而不是以天或星期计算。这些模型有助于解决设计软件的问题，而且对详述和探索需求也是有益的。作为需求分析工具，你可以用这些图对问题域进行建模，或者创建新系统的概念表示法。图形有助于分析者和客户在需求方面形成一致的、综合的理解，并且还可以发现需求的错误。

在需求分析方面或设计方面是否使用模型取决于建模的定时和目的（ timing and intent of the modeling）。在需求开发中通过建立模型来确信你理解了需求。模型描述了问题域的逻辑方面，如数据组成、事务和转换、现实世界对象和允许的状态。或者可以从文本需求出发来

画模型，从不同的角度来表示这些需求，或者可以从所画的基于用户输入的模型来获得功能需求。在设计阶段，要从物理上而不是从逻辑上画出模型来明确说明将如何实现该系统：规划建立的数据库，将举例说明的对象类，还有你将开发的编码模块。

本章所叙述的分析建模技术是由各种商业计算机辅助软件工程或 CASE工具支持的。CASE工具提供了普通画图工具所没有的许多性能。首先，这些工具通过交互画图使你易于对模型进行改进。决不可能第一次就画出一个正确的模型，因此，在系统建模中提供交互功能是成功的一个关键（Wiegers 1996a）。第二，CASE工具知道它们所支持的每一种建模方法的规则。它们可以验证模型，并且识别人们在评审图形时没有发现的语法或逻辑错误。该工具还可以把多系统图形一起连接到数据字典中以共享数据定义。 CASE工具有助于你保持模型之间的一致性并使模型与软件需求规格说明中的功能需求保持一致。

分析模型方便了项目参与者在系统的某些方面的交流。可能不需要整个系统的模型集，只需关注建模中系统最复杂和最关键的部分，因为这部分最容易产生模糊性和不确定性。这里所表示的符号为项目的参与者提供了统一的语言，但是也可以使用非正式的图来增强口头和书面的方案交流。

10.2 从客户需求到分析模型

通过认真听取客户如何陈述它们的需求，分析者可以挑选出关键字，这些关键字可以翻译成特定的分析模型元素。表 10-1建议了一些可能的映射，根据客户输入，把重要的名词和动词映射成特定的模型组件，这将在本章的后面部分介绍。当把客户输入转变为书面的需求或模型时，还可以根据模型的每个组件回溯到需求部分。

表10-1 把客户的需求关联到分析模型的组件

单词类型	例 子	分析模型组件
名词	• 人、组织、软件系统、数据项或者存在对象	• 端点或数据存储（DFD） • 实体或它们的属性（ERD） • 类或它们的属性（类图）
动词	• 行为、用户可做的事或可发生的事件	• 过程（DFD） • 关系（ERD） • 转化（STD） • 类操作（类图）

在整本书中，我已经使用“化学制品跟踪系统”作为一个学习的例子。基于此例子，考虑如下用户需求部分，这些需求是由代表化学制品用户类的产品代表者提供的。由于用图示例的缘故，一些模型所示的信息可能会超出这部分所包含的信息，而另一些模型可能只描述部分信息。

> “一位化学家或化学制品仓库人员可以提出对一种或多种化学制品的请求。对该请求的执行可以有两种途径：一是传送一个存在于化学制品仓库清单上的化学制品容器，二是向外界供应商提交一份订购新的化学制品的订单。提出请求的人在准备他/她的请求时应该可以通过在线查找供应商目录表找到特定的化学制品。系统需要从请求准备直到请求执行或取消这一阶段跟踪每一个化学制品的状态。系统还必须保持跟踪每个化学制品容器的历史记录，从化学制品容器到达公司到它完全被消耗或废弃为止。”

10.3 数据流图

数据流图（data flow diagram，DFD）是结构化系统分析的基本工具（ DeMarco 1979；Robertson and Robertson 1994）。一个数据流图确定了系统的转化过程、系统所操纵的数据或物质的收集（存储），还有过程、存储、外部世界之间的数据流或物质流。数据流模型把层次分解方法运用到系统分析上，这种方法很适用于事务处理系统和其它功能密集型应用程序。通过加入控制流元素后，数据流图技术就可以扩充到允许实时系统的建模。

数据流图是当前业务过程或新系统操作步骤的一种表示方法。数据流图可以在一个抽象的广泛范围内表示系统。在一个多步骤的活动中，高层数据流图对数据和处理部分提供一个整体的统览，这是对包含在软件需求规格说明中的精确、详细叙述的补充。数据流图描述了软件需求规格说明中的功能需求怎样结合在一起使用户可以执行指定的任务，例如请求一种化学制品。在与用户一起讨论业务过程时我经常绘制数据流图。从图中迅速反馈的信息有助于对所探讨的任务流的理解进行提炼加工。

图6-2所示的关联图是数据流图最高层的抽象。关联图把整个系统表示成一个简单的黑匣子的过程，并用一个圆圈表示。关联图还表示出外部实体或与系统有关的端点，以及在系统与端点之间的数据和物质流。在关联图中，元素之间的流往往代表了复杂的数据结构，这些数据结构在数据字典中定义。

你可以把关联图详述成0层数据流图，这时将系统划分为主要部分或过程。图 10-1展示了“化学制品跟踪系统”的0层数据流图（略作简化）。在关联图中代表整个“化学制品跟踪系统”的单一过程圆圈被细分成 7个主要过程（圆圈）。在关联图中，端点用矩形框表示。关联图中所有的数据流也出现在0层数据流图上。此外，0层数据流图包含了许多数据存储（data store），它是用一对水平的平行线表示，由于数据存储在系统内部，因此它们并不出现在关联图中。从圆圈到数据存储的流表示数据放入数据存储器，从数据存储器出来的流表示一个读操作，而数据存储器和圆圈之间的双向箭头则表示一个更新操作。

在0层图中，每个独立的圆圈所代表的过程可以进一步扩展成一个独立的数据流图，以揭示系统中程序的细节部分。这种循序渐进的细化过程可以继续进行，直到最低层的图仅描述原子过程操作为止，这些原子操作可以清楚地表示所叙述文本、伪码、流程图或程序代码。软件需求规格说明中的功能需求将精确地定义每个原子过程的行为。每一层数据流图必须与它上一层数据流图保持平衡和一致，因此，子图的所有输入输出流要与其父图相匹配。高层图中复杂的数据流可以分解到低层数据流图中，并把这些数据结构写入数据字典中。

初看起来，图10-1可能有点混乱。然而，如果你仔细观察每一个过程的周围环境，你就会看到该过程的输入和输出数据项，还有它们的源和目的地。与数据存储相连的流可以引起建立或消耗数据存储内容的过程。为了看清一个过程如何使用数据项，你需要画出更详细的DFD子图或者参考系统有关部分的功能需求。

以下是绘制数据流图的一些规则。并不是每个人都要遵循这些规则，但是我发现这些规则很有用。利用模型以增进项目参与者之间的交流比生搬硬套这些规则更为重要。

- 把数据存储放在0层数据流图或更低层子图上，不要放在关联图上。
- 过程是通过数据存储进行通讯，而不是从一个过程直接流到另一过程。类似地，数据不能直接由一个数据存储直接流到另一个数据存储，它必须通过一个过程圆圈。
- 使用数据流图时，不要试图让数据流图反映处理的顺序。

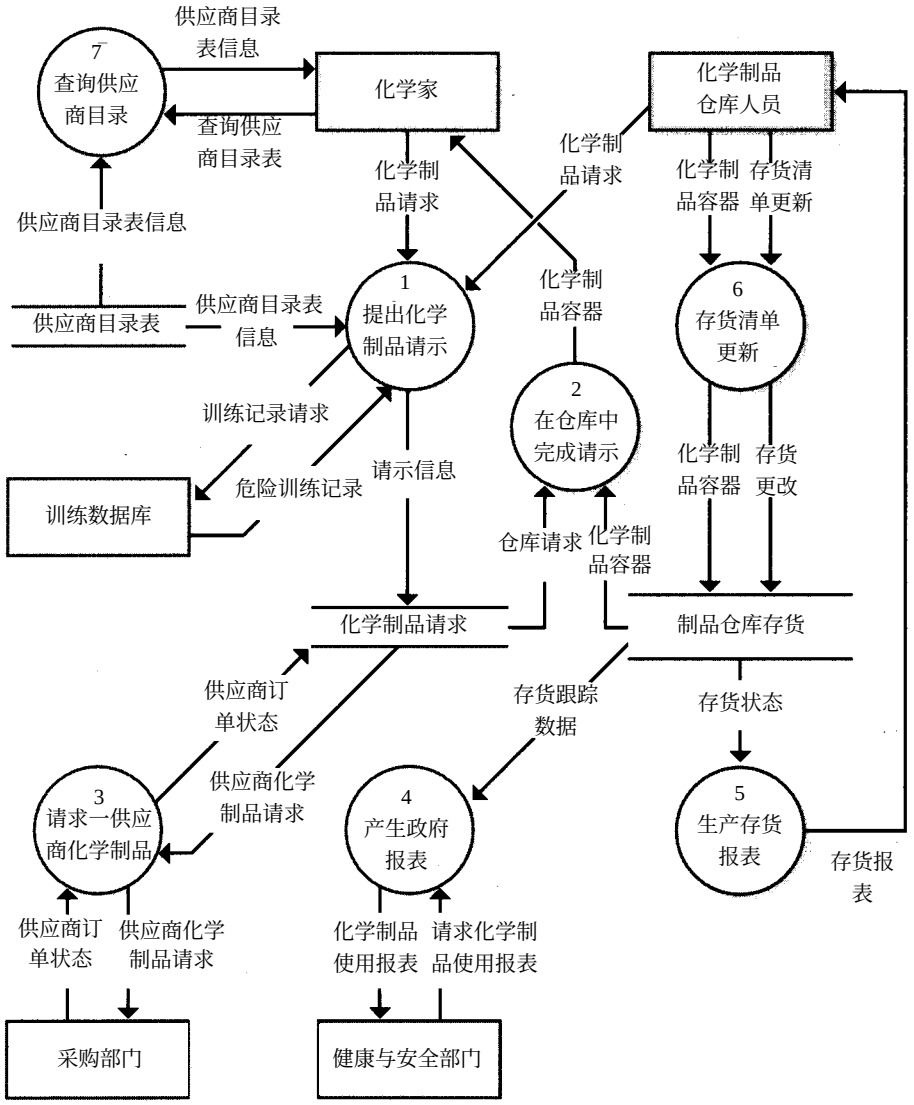

图10-1 “化学制品跟踪系统”的0层数据流图

- 用一个简明的动作命名过程：动词 +对象。数据流图中所用的名字应对客户有意义，并且与业务或问题域相关。
- 对过程的编号要唯一且具有层次性。在 0层图上，每个过程的编号用整数表示。如果你为过程3创建子图，则子图中的过程编号应表示为 3.1，3.2等等。
- 不要在一个图中绘制多达7~10个以上的过程，否则就很难绘制、更改和理解。
- 不要使某些圆圈只有输入或只有输出。数据流图中圆圈所代表的处理过程通常要求既有输入又有输出。

10.4 实体联系图

与数据流图描绘了系统中发生的过程一样，实体联系图（ entity-relationship diagram，ERD）描绘了系统的数据关系（Wieringa 1996）。如果你的实体联系图表示来自于问题域及其

联系的逻辑信息组，那么你正在利用实体联系图作为需求分析的工具。分析实体联系图有助于对业务或系统数据组成的理解和交互，并暗示产品将有必要包含一个数据库。相反，当你在系统设计阶段建立实体联系图时，通常要定义系统数据库的物理结构。

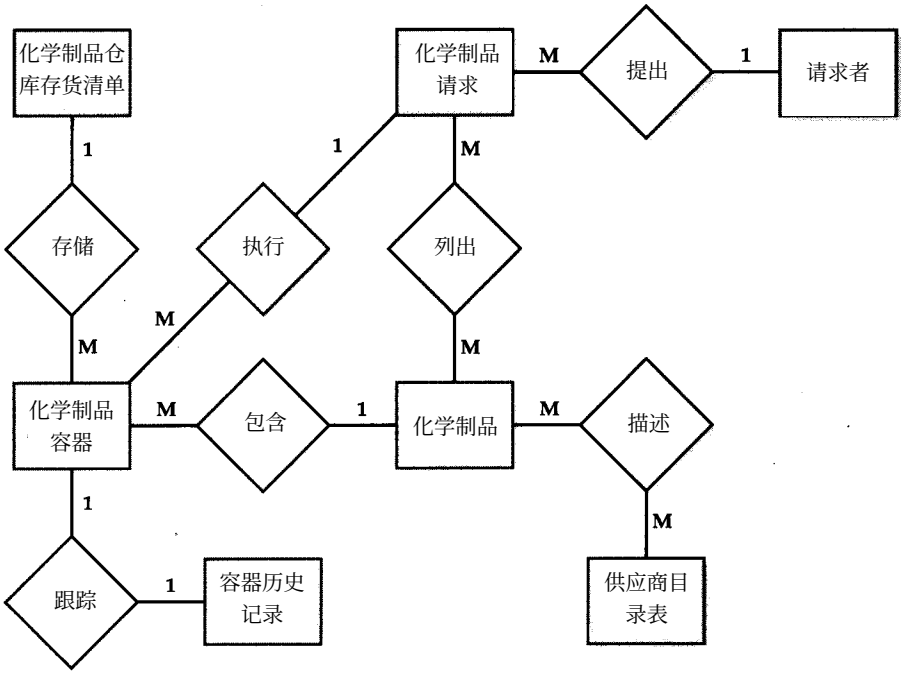

图10-2 “化学制品跟踪系统”的实体联系图

实体（entity）是物理数据项（包括人）或者数据项的集合，这对所分析的业务或所要构造的系统是很重要的（Robertson and Robertson 1994）。实体用单数名词命名并在矩形框中表示。图10-2描绘了“化学制品跟踪系统”实体联系图的一部分。注意：被命名为化学制品请求、供应商目录表和化学制品仓库存货清单的实体在图 10-1的数据流图中是作为数据存储出现的。其它一些实体代表与系统交互的操作员（请求者），业务运转中一部分物理项（化学制品容器）、数据块等，并不出现在0层数据流图中，但将出现在一个更低层的数据流图中（容器的历史记录，化学制品）。

每一实体用几个属性（attribute）来描述；每一实体的单个实例将具有不同的属性值。例如每一个化学制品的属性包括一个唯一的化学制品标识号、它的正式化学制品名称和它的化学结构的图形表示。数据字典中包含这些属性的详细定义，保证了实体联系图中的实体和在数据流图中相应的数据存储定义一致。

实体联系图中的菱形框代表关系（relationship），它确定了实体对之间逻辑上和数量上的联系。关系按照关联的属性来命名。例如，请求者和化学制品请求之间的关系被称为“提出”（placing），因为一个请求者提出一个化学制品的请求，或者，一个化学制品请求被一个请求者提出。

在实体和关系的连线上用一个数字或字母表示实体的单联系和多联系。不同的实体联系图符号用不同的规则表示联系；本例子描述了一个普遍的方法。因为每一个请求者可以提出多个请求，所以在请求者和化学制品请求之间存在一对多的联系。单联系在请求者和“提出”

关系的连线上标明“ 1 ”，多联系在化学制品请求和“ 提出 ”关系的连线上标明“ M ”(代表多)。其它可能的联系如下所示：

- 一对一（每一个容器历史记录跟踪一个化学制品容器)。
- 多对多（每一个供应商目录中描述了许多种化学制品，并且每种化学制品可以被多个供应商目录表所描述)。

如果你知道存在精确的关系，而不仅仅是“ 很多 ”关系，就可以用特定的数字或一个数字范围来表示，而不是用一般的“ M “ 表示。

在需求分析时绘制实体联系图是一种组织你所知道的业务或新系统的重要数据元素的好方法。当我在一个业务过程再建工程小组担任信息技术代表时，我就使用过这种技术。当开发组提出一个新的过程流时，我总是要问负责每一个新过程步骤的开发组成员：执行这些步骤需要什么数据项。我还问他们由这些步骤所创建的哪些数据项要存储起来以备后用。

在与这些过程步骤的拥有者商讨之后，我们定位调整所有步骤所需要的数据和所生成的数据。利用这种关联关系可以发现不能在系统中产生的所需要的数据项，还可以发现只被存储而未使用过的数据。最终，我用一个实体联系图和一个数据字典来记录这些数据关系，这可以为新的业务过程提供一个数据组成的概念性框架。这些分析工具可以增强我们对问题的理解，即使我们从未构造过软件系统来支持新的业务过程，这也是值得的。

10.5 状态转换图

实时系统和过程控制应用程序可以在任何给定的时间内以有限的状态存在。当满足所定义的标准时，状态就会发生改变，例如在特定条件下，接收到一个特定的输入激励。这样的系统是有限状态机的例子。此外，许多业务对象（如销售订单、发票，或存货清单项）的信息系统处理是贯穿着复杂生存周期的；此生存周期也可以看成有限状态机。大多数软件系统需要一些状态建模或分析，就像大多数系统涉及到转换过程、数据实体和业务对象。

用自然语言描述一个复杂的有限状态机可能会忽略一个允许的状态改变或者引起一个不允许的改变。与状态机的行为有关的需求可能将多次出现在软件需求规格说明中，它取决于软件需求规格说明是如何组织的。这对综合理解系统行为造成困难。

状态转换图(state-transition diagram，STD)为有限状态机提供了一个简洁、完整、无二义性的表示（Davis 1993)。

状态转换图包括如下面三种元素：

- 用矩形框表示可能的系统状态。
- 用箭头连接一对矩形框表示允许的状态改变或转换。
- 用每个转换箭头上的文本标签表示引起每个转换的条件。标签经常既指示条件也指示相应的系统输出。

状态转换图没有表示出系统所执行的处理，只表示了处理结果可能的状态转换。对于软件系统中只能存在于特定状态的那一部分，你可以使用状态转换图来建模。特定状态或者指诸如汽车巡航控制系统等实时世界实体的行为，或者是这系统操纵的个体项状态。

状态转换图有助于开发者理解系统的预期行为，它对于检查所要求的状态和转换是否已全部正确地写入功能需求中也是一种好方法。测试者可以从覆盖所有转换路径的状态转换图中获得测试用例。用户只要稍微学一些符号就可以读懂状态转换图。

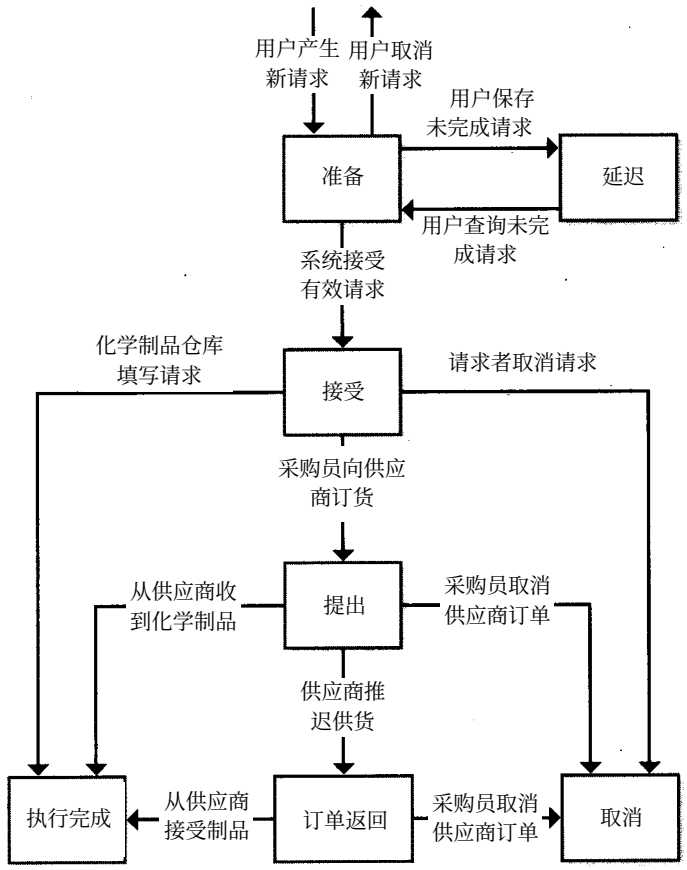

图10-3 “化学制品跟踪系统”中化学制品请求的状态转换图

回想一下，在化学制品跟踪系统中有一个主要功能是允许称为请求者的操作员提出对化学制品的请求，这一请求可以由化学制品仓库中的存货清单来执行完成，也可以通过向外界供应商发出订单来执行完成。每一个请求从创建到完成或取消这一时间段内将经历一系列可能的状态。于是，我们就可以把化学制品请求的生存周期看成一个有限状态机，其建模如图10-3所示。这个状态转换图说明了一个请求可取下列七种可能状态中的一种：

- 准备(in preparation) —— 请求者正在创建一个新的请求，已经从系统的其它部分进入那个功能。
- 延迟(postponed) —— 请求者存储他/她的工作以备后用，而并不向系统提交请求或者取消请求。
- 接受(accepted) —— 用户提交一个完整的化学制品请求，系统判断该请求顺序是否有效，如果有效就接受该请求进行处理。
- 提出(placed) —— 采购员向供应商订货，并且外部供应商必须满足请求。
- 执行完成(fulfilled) —— 通过从化学制品仓库交付一个化学制品容器给请求者或者收到供应商的化学制品收据时才能使请求得以满足。
- 订单返回 (back-ordered) —— 供应商没有可用的化学制品，他通知采购员货物已订购以后再交付。

• 取消(canceled)—— 在请求执行完成之前，请求者取消一个已被接受的请求，或者采购员在订单执行完成或返回订单之前，取消供应商订单。

当“化学制品跟踪系统”的用户代表评审最初的化学制品请求的状态转换图时，他们发现有一个不必要的状态，另外有一个必不可少的状态但分析者却没有记录，还有两个不正确的转换。这些错误在他们评审功能需求时，却没有一个人发现。

状态转换图并没有提供使开发者如何构造系统的详细细节。因此，软件需求规格说明必须包括与处理化学制品请求和它的可能状态变化相关的功能需求。然而，状态转换图对可能的请求状态和它们是如何允许变化提供了一个简洁的可视化表示。

实时系统的状态转换图除了包括一个空闲状态外，与图 10-3所示的相类似，在这种系统中，当系统不再执行其它处理时就返回空闲状态。相反，对于一个贯穿整个定义的生存期的对象，例如一个化学制品请求，其状态转换图将有一个或多个终结状态；在图 10-3中则表现为执行完成和取消状态。

10.6 对话图

在许多应用程序中，用户界面可以看作是一个有限状态机。在任何情况下仅有一个对话元素（例如一个菜单，工作区，行提示符或对话框）对用户输入是可用的。在激活的输入区中，用户根据他所采取的活动，可以导航到有限个其它对话元素。在一个复杂的图形用户界面中，可能的导航路径可以有许多种，但其数目是有限的，并且其选择通常是可知的。因此，许多用户界面可以用状态转换图中的一种称为对话图 (dialog map)来建模。

对话图代表了一个高层抽象的用户界面体系结构。对话图描绘了系统中的对话元素和它们之间的导航连接，但它没有揭示具体的屏幕设计。对话图可以使你在对需求的理解上探索假设的用户界面概念。用户和开发者可以通过对话图在用户如何利用系统执行任务上达成共同的视觉界面。可视化Web站点的构建对话图也很有用，在Web站点上，它们有时被称作“站点图”。你在Web站点中所建立的导航连接在对话图中则表现为转换。对话图与系统情节叙述相关联，这些叙述还包括对每一个屏幕意图的简短说明。

对话图抓住了用户—系统交互作用和任务流的本质，而不会使你太快陷入到屏幕布局和数据元素的特定细节中。用户可以通过跟踪对话图寻找遗漏、错误或多余的转换，和因此而有遗漏、错误或多余的需求。你可以把在需求分析过程中形成的对话图用作详细用户界面设计时的指南，最终形成一个执行的对话图，该对话图记录了产品的真正用户界面的体系结构。

就像在普通的状态转换图中一样，在对话图中，对话元素作为一个状态（矩形框），每一个允许的导航选择作为转换（箭头）。触发用户界面导航的条件用文本标签写在转换箭头上。下面列出几种触发条件的类型：

• 一个用户动作，例如按下一个功能键或点击一个超链接或对话框的按钮。
• 一个数据值，例如一个无效的用户输入触发显示一个错误信息。
• 一个系统条件，例如检测到打印机无纸。
• 这些情况的一些组合，例如输入一个菜单项数字并按下回车键。

为了简化对话图，可以省略全局功能，例如按下 F1键显示帮助。软件需求规格说明中用户界面部分必须确定这个功能可用，但把它写入对话图中作为交互工具的模型，其意义不大。类似地，在为Web站点建模时，你不必包括站点中每一页都出现的标准导航链接。你还可以

省略那些Web页导航顺序的反向流转换，因为Web浏览器上的退格键(back button)可以处理这个导航。

对话图是表示在使用实例中所描述的操作员和系统交互的好方法。对话图可以把可选过程叙述成普通过程流的分支。我发现在使用实例获取讨论会期间，在白板上绘制对话图是有益的，在这一阶段开发组成员正在探索导向任务完成的操作员动作顺序和系统响应的顺序。

第8章提出了在“化学制品跟踪系统”中称为请求一种化学制品的使用实例。这个使用实例的正常过程包括了请求一种化学制品，并由向外部供应商订货来满足该请求。可选过程将供给来自化学制品仓库存货清单的化学制品容器。提出请求的用户在进行选择之前，需要浏览仓库中可用的化学制品的历史。图10-4显示了这个使用实例的对话图。

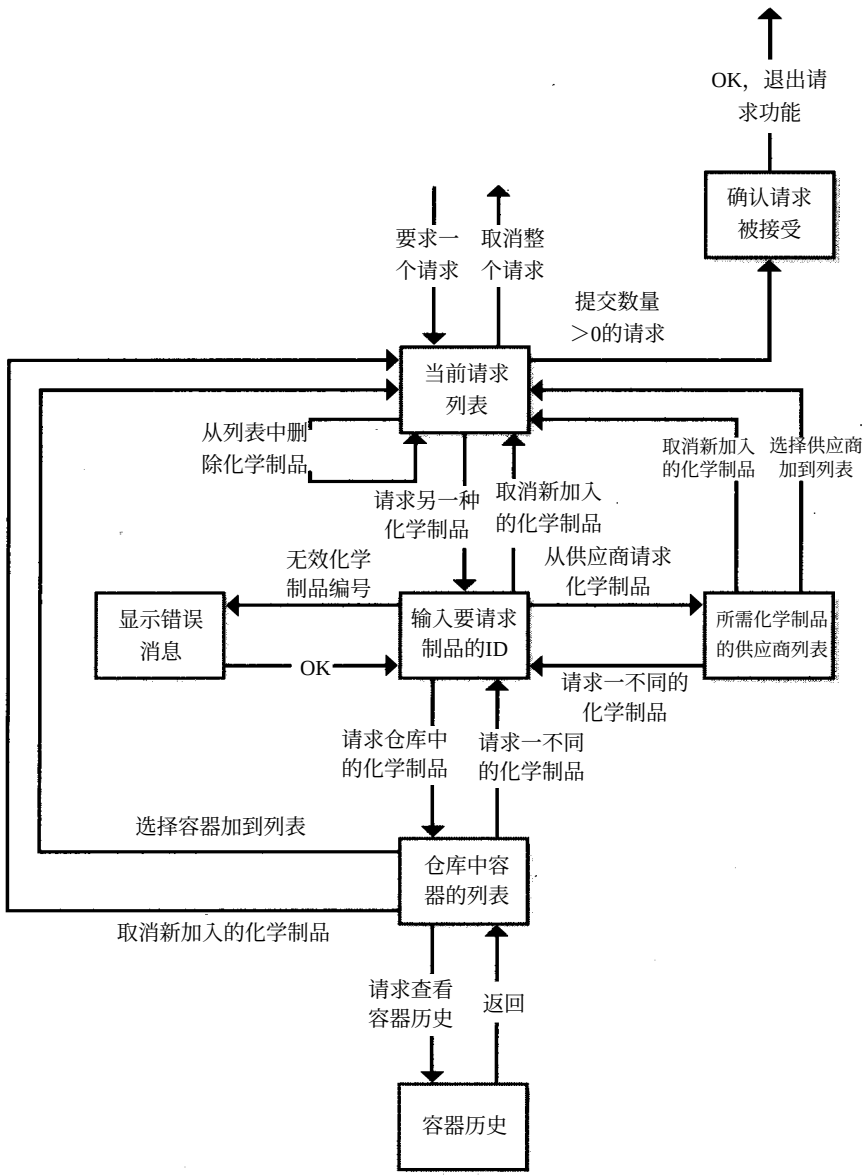

图10-4 “化学制品跟踪系统”中一种化学制品使用实例的对话图

这种对话图最初可能看起来比较复杂，但如果通过一次一条线和一个框来跟踪，也是不难理解的。请记住，在需求分析阶段，对话图代表了用户和系统在概念级上可能的交互通信作用；但真正的实现可能是有所不同的。在“化学制品跟踪系统”中，用户从菜单中选择“请求一种化学制品”启动使用实例。这个使用实例的主工作区间是在这一请求中的一个化学制品列表，并在一个称为当前请求列表中显示出来。从矩形框出来的箭头显示了所有导航选择，于是用户可从那些联系中获得可用的功能：

- 取消整个请求。
- 如果请求包含了至少一种化学制品，则提交该请求。
- 在请求列表中加入一个新的化学制品。
- 从列表中删除一种化学制品。

注意：最后一个操作，删除一种化学制品，并不涉及其它的对话元素；而仅仅是在用户做出更改之后，刷新当前请求列表。

当你遍历这张对话图时，你将会看到反映其余的“请求一种化学制品”使用实例的元素：

- 向供应商请购化学制品的一条流路径。
- 来自化学制品仓库的执行请求的另一条路径。
- 查看特定化学仓库容器的历史记录的一条可选路径。
- 对处理无效化学制品标识号输入的一条错误信息显示。

对话图中的一些转换允许用户退出操作。对于查看是否忽略任何可用性需求对话图是一种很好的方法，比如向导中遗漏了一个退格键，此时就会强迫用户完成一个不需要的动作。

当用户检查这个对话图时，他可能发现一个遗漏需求。例如，一个谨慎的客户可能认为确认取消整个请求的操作是个好主意，因它可以预防不小心丢失数据。在分析阶段，增加这一新功能只需要极少的代价，但是在已发布的产品中增加这一功能，则代价甚大。由于对话图仅表示了在用户和系统交互中包含的可能元素的概念视图，所以不要试图在需求阶段去攻克所有用户界面的设计细节。而是利用这些模型使项目的风险承担者在系统预期的功能上达成共识。

10.7 类图

面向对象的软件开发优于结构化分析和设计，并且它运用于许多项目的设计中，从而产生了面向对象分析、设计和编程的域。在业务或问题域中，对象 (object)通常与现实世界中的项相类似。对象代表了从称为类的普通模板获得的单个实例。类描述包含了属性（数据）和在属性上执行的操作。类图 (class diagram)是用图形方式叙述面向对象分析所确定的类以及它们之间的关系。

利用面向对象方法开发的产品并不需要特殊的需求开发方法。这是因为需求开发强调用户需要系统做什么以及系统所应包含的功能，而并不关心系统如何做。用户并不关心你如何构造系统，也不关心对象和类。然而，如果你要用面向对象的技术来构造系统，这将有助于你在需求分析阶段确定类和它们的属性及行为。当你考虑如何将问题域对象映射到系统对象，并进一步细化每个类的属性和操作时，面向对象技术可以方便需求开发到设计阶段的转换。

许多不同的面向对象的方法和符号几年来已取得很大的进展。最近，这些方法及符号中的许多部分都将它纳入“统一建模语言”（UML）中(Booth，Rumbaugh and Jacobson 1999)。

在适合于需求分析的抽象层上，你可以像图 10-5所示那样，用统一建模语言的符号为化学制品跟踪系统的一部分（你所假设的）绘制类图。你可以容易地把这些不包含实现细节的概念性类图详叙成在面向对象设计和实现中所需的更详细的类图。使用顺序图 (sequence diagram)和联系图(collaboration diagram)可以表示类之间的交互以及它们所交换的信息，本书未对它作深入的研究（见Booch, Rumbaugh and Jacobson 1999）。

图10-5显示了四个类，每个类都放在一个大的矩形框中：请求者、供应商目录表、化学制品请求和请求行项目。这个类图和其它分析模型所表示的信息有相似之处。出现在图 10-2实体联系图中的请求者，代表了可以由化学家或化学制品仓库用户类扮演的操作员角色。图10-2数据流图也表示这两个用户类可以提出对化学制品的请求。不要把用户类和对象类相混淆，虽然它们名字相似，但并不存在特定的联系。

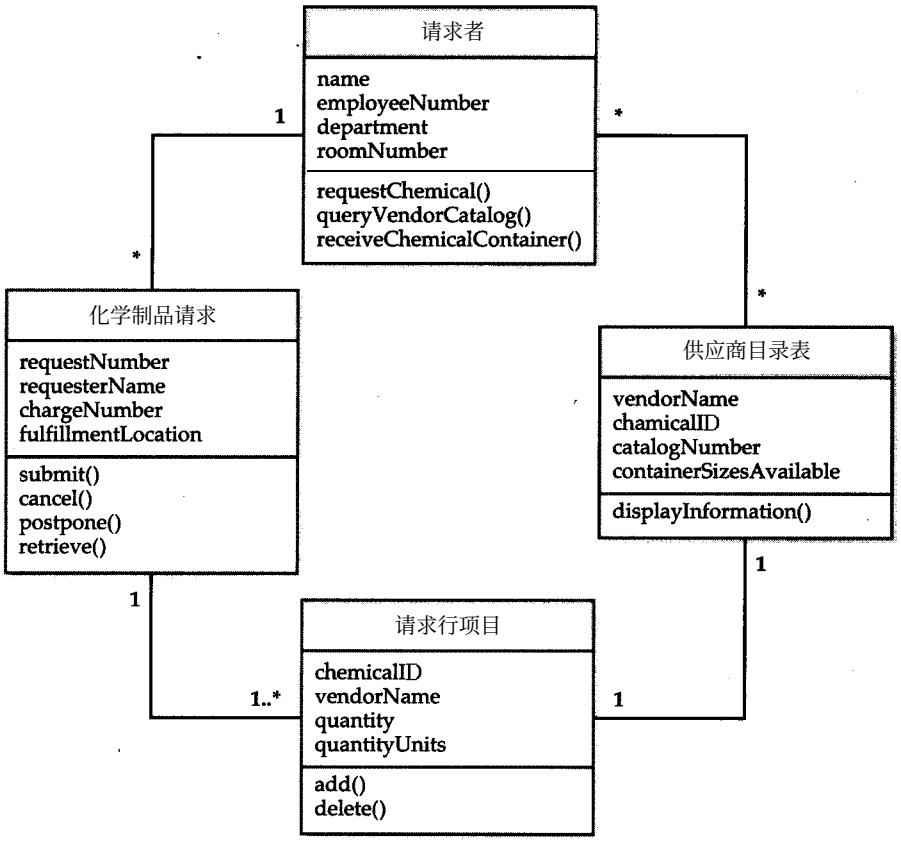

图10-5 化学制品跟踪系统中的部分类图

与请求者类相关的属性列在类方框的中部：姓名，雇员号，部门和房间号（大写是统一建模语言的一个规则）。这些数据项是与请求者类的每一对象相关的。在数据流图中，类似的属性出现在存储定义中。

请求者类的对象可以执行的操作列在类方框的底部，并且后面都跟着空括号。在表示设计的类图中，这些操作将和类的函数或方法相对应，分析类模型仅能看出：请求者可以请求化学制品，查询供应商目录和得到化学制品容器。在类图中所列出的操作与出现在底层数据流图的圆圈中的过程是相对应的。

在图10-5中连接类方框的连线代表了类之间的联系。在连线上所示的数字表示了联系的复杂度，就像在数据流图中，连线上的数字代表了实体之间联系的复杂度一样。在图 10-5中，星号代表了在请求者和一个化学制品请求之间一对多的关系：一个请求者可以提出多个请求，但每个请求只属于一个请求者。

10.8 最后的提醒

本章所述的每一种建模都有其优点和局限性。牢记，你创建的分析模型可以提供对需求理解和通信的一个等级，而这却是文本方式下的软件需求规格说明和其它单一的表示法所不能提供的。应该避免陷入在软件开发方法和模型中发生的教条的思维模式和派系斗争。

下一步：

- 使用一个编写完整的软件设计文档来实践本章所描述的建模技术。
- 找出软件需求规格说明中读者难于理解的部分或者发现缺陷的部分。选择一个本章所描述的适于表示部分需求的分析模型。绘制模型并评估如果你在需求开发的早期阶段创建它，是否对你有帮助。
- 接着，需要把一些需求编写成文档，选择一种可以补充文本模型的建模技术。在纸上勾画模型一两次，以确信你的方法是正确的，然后使用支持你所使用的建模符号的商业CASE工具。逐渐地，把对这部分的不断建模融合进你的标准需求开发过程。