返回总目录

目　　录

第 12 章　UML 在商业建模和商业工程再工程中的应用

目前，商业系统大量涌现的面向对象分析与设计（OOA&D）工具称为 CASE 工具，它们要解决的首要问题是为商业系统开发提供一个集成的“从概念到代码”的方案。然而还没有哪个工具已经达到这个目标，主要有以下原因：

- 在系统设计之前，没有足够地强调商业的需求；
- 很难把商业需求转换成系统组件；
- 没有提供商业需求和系统组件之间的可跟踪性；
- 缺少对整个企业范围的理解，不能洞察需要建模的事物。

一些新兴的技术、分析和设计方法以及与此相关的商业过程再工程（Business Process Reengineering，BPR）正在逐渐使这一目标变得清晰。

本章以 UML 在商业工程再工程（BPR）中的应用为中心，以解决和避免面向对象开发工具常见的缺陷和不足为目标，提出一个把系统分析集成在企业范围的商业需求建模的方法，给出十六种最好的商业模型以及相关的模型图，分析它们如何从不同的抽象层次、不同的侧面，描述商业的架构。最后，以商业工作流程为重点，具体讨论了如何用 UML 为商业建模。

12.1　商业、商业过程和商业过程再工程

商业，一个企业（enterprise），是一个复杂的系统，具有特定的意图或目标，商业的所有功能以及它们之间的交互都是为了完成这一目标。一个商业系统会受到其它系统的决策或事件的影响，还可能会与其它系统互相有联系，因此一个商业系统是一个开放的系统，我们无法完全界定一个商业系统的界限。

商业过程是商业的活动部分。它是一个抽象，显示资源之间的合作及转换。它强调整个工作是如何完成的，它的更正式的定义如下：

商业过程是一组有结构的活动，是用来为特定用户或市场生产某个特定的产品。它实际上强调在一个机构内是如何完成这些工作的，而不是做什么。因此过程是跨越时间和空间的一组有特定次序的活动，有开始和结束，并且明确定义了输入和输出：它是行动的结构。

商业过程（再）工程（BPR）的一条最基本原则是，通过从外部和内部对商业过程的再思考，使企业能够得到更强的竞争实力，从而获取更多的利益。外部的再思考检验客户对企业、它的产品和服务的看法，它的目标是使企业具有更快的响应和更高的效率，抛弃过去的机构障碍，寻找更有效的方法使工作流程以及企业更有效。为了达到商业过程（再）工程（BPR）的目标，必须首先为商业和它的过程建模，才能得到更好的商业系统设计。

12.2　商业工程再工程和商业建模

在进行代码设计之前，开发企业范围的商业和系统模型将提供一个概念“蓝图”，确保商业用户和软件工程师对要开发的系统有一个共同的理解，使最终的软件符合企业的需要，这是实现“从概念到代码”的关键一步。企业建模的好处是：

使最终系统的目标符合商业目标；

- 通过快速配置符合商业需求的应用系统，使企业得到更强的竞争实力，从而获取更大的利润；
- 一致的模型可以对新的开发、软件选择以及软件定制提供指导；
- 可以通过对模型而不是代码的评估来管理对已有应用程序的评估、升级和维护；
- 它提供架构框架，从而能在多个不同环境上的多计算机系统上进行重用、精化，并配置常用的商业对象。

在构建任何类型的结构时，架构的角色都是明确的。一个设计良好的架构不仅能够确保对所建结构的完整理解，规划实际的构造，估算成本，它还是结构的蓝图。一旦构造完成，好的架构设计就是整个过程和结果的文档，为以后理解、维护以及扩建系统打下了基础。在信息系统的构造中，架构也具有同样的意义。架构以一种有组织的方式体现出信息系统结构的重要部分，成为后期系统管理的重要工具。在商业中，架构定义商业结构，对架构的建模将是理解商业以及理解商业如何运作的关键。

尽管很难给出完整的商业架构的定义，但我们可以这样定义商业架构：

是一个有组织的元素的集合，各元素之间有清晰的关系，形成一个整体定义了它的功能……元素代表了商业系统的组织和行为结构，显示了商业过程中的关键过程和结构的抽象。

好的架构使建模人员能够从不同的侧面或视角给出商业的抽象，并且一次只在一个侧面或视角进行具体的描述。有一定程度的抽象，压缩细节和无关的信息是理解复杂系统和关系的基础。好的架构具有以下特点：

- 能够更真实和正确地描述实际的商业；
- 在恰当的抽象层次上，侧重于商业的关键过程和结构；
- 在商业中运作的不同的人的观点；
- 易于扩展和改变；
- 易于不同的商业参与人的理解和沟通。

采用 UML 来为架构建模，将有利于最终的架构符合以上特点。

12.2.1　商业建模所描述的对象

不同的商业有不同的目标和内部结构，但它们都是用以下概念来描述的：系统中的对象，它们之间的关系和结构，以及它们在不同的场合的交互。而商业系统的模型将描述以下这些概念，它们是：

- 资源。指商业内的对象，如人、物资、信息和产品，这些是商业使用和生产的产品。资源的分布是有结构的，并且相互之间还有关系。通过过程对资源进行操作

（作用、更新、生产）。可以把资源分成物理的、抽象的和信息的等几类。

- 过程。商业内完成的活动，其间商业资源的状态将发生变化。过程描述了商业中是如何完成工作的，它们是由规则管理的。
- 目标。商业的意图，或者说是商业作为一个整体将达到的结果。可以把目标划分子目标，分配到商业中的个体上。目标表达了想要的资源状态，并且由过程达到。目标可以用一条或多条规则表达。

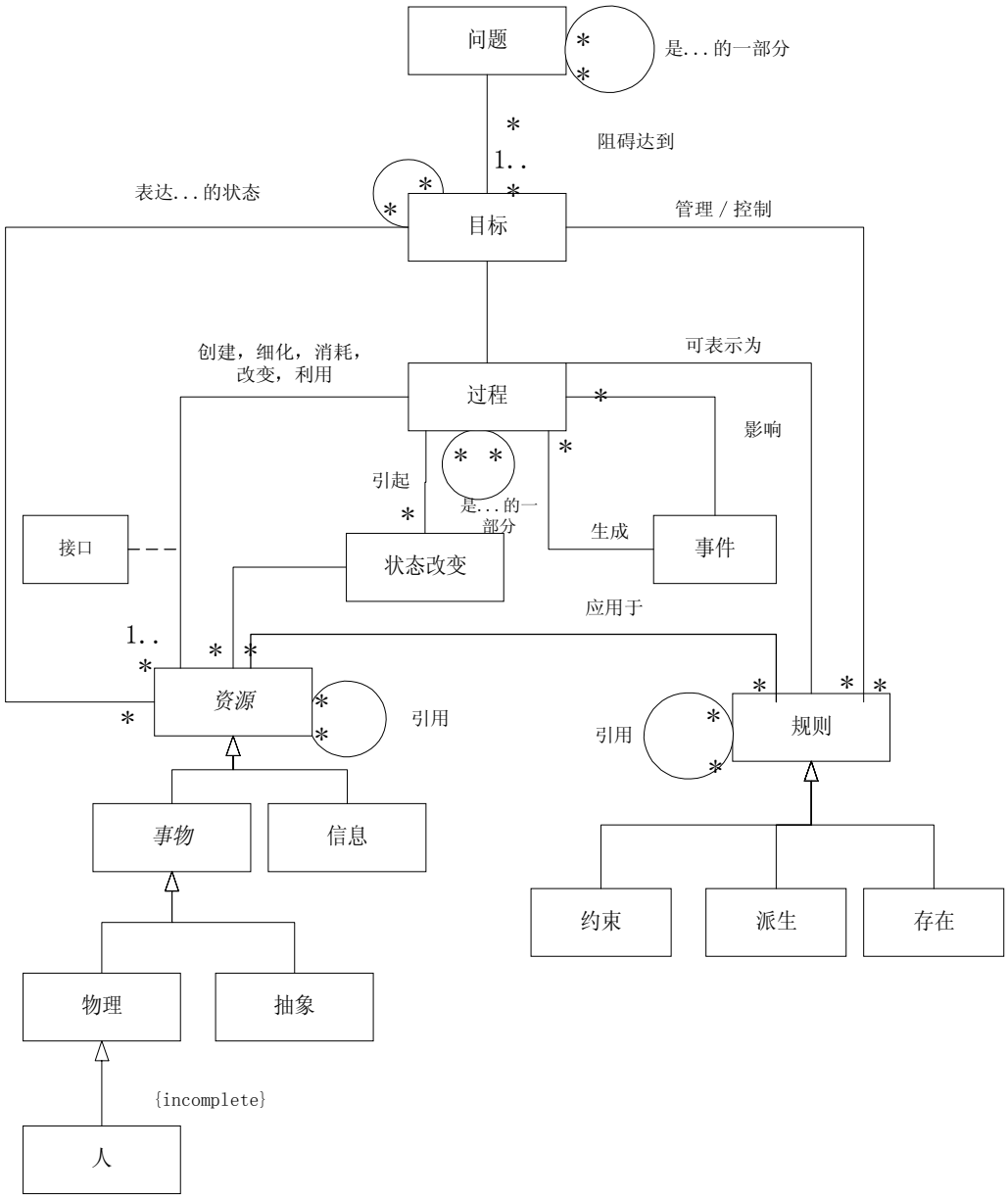

图 12-1　商业建模概念的一个基本元模型

- 规则。一些定义或约束商业的陈述，表达了商业知识。规则明确了如何运作商业（也就是说过程如何执行）以及资源的结构和相互的关系。规则可以从外界通过

法规或法律加在商业之上，也可以在商业内部定义。规则可分为功能的、行为的和结构的等几类。

这些概念相互都有关系：一条规则可以影响一些资源的结构；一个资源可以分配给某个特定过程；一个目标与某个过程的执行有关。商业建模的目的就是要定义这些概念，并且显示它们之间的关系和交互。

我们用图 12-1 总结了商业模型概念，这是一个元模型（meta-model），体现了基本的商业概念以及它们之间的关系，元模型所描述的概念可以用来创建其它模型，用于其它商业模型的开发。这个元模型是一个 UML 类图，其中每个概念都用一个类描述，而概念之间的关系则用一个关联或一个专有化表示，元模型也表明了哪些因素将影响或阻碍商业达到它的目标。

12.2.2 商业对象

仅仅为商业过程建模还不能够描述企业的所有侧面，因为每个商业过程都管理和操作一组商业对象，一个商业对象可以是与商业过程交互或它的一部分的任何个人或事物。每个商业对象都可以对商业对象进行识别和管理具有以下优点：

- 共同的商业对象体现过程之间共同的信息和操作；
- 识别出共同性有助于控制过程之间的覆盖；
- 商业过程可以有效地重用这一共同性；
- 每个使用同一个商业对象的商业过程，都要求具有管理该商业对象的商业规则。

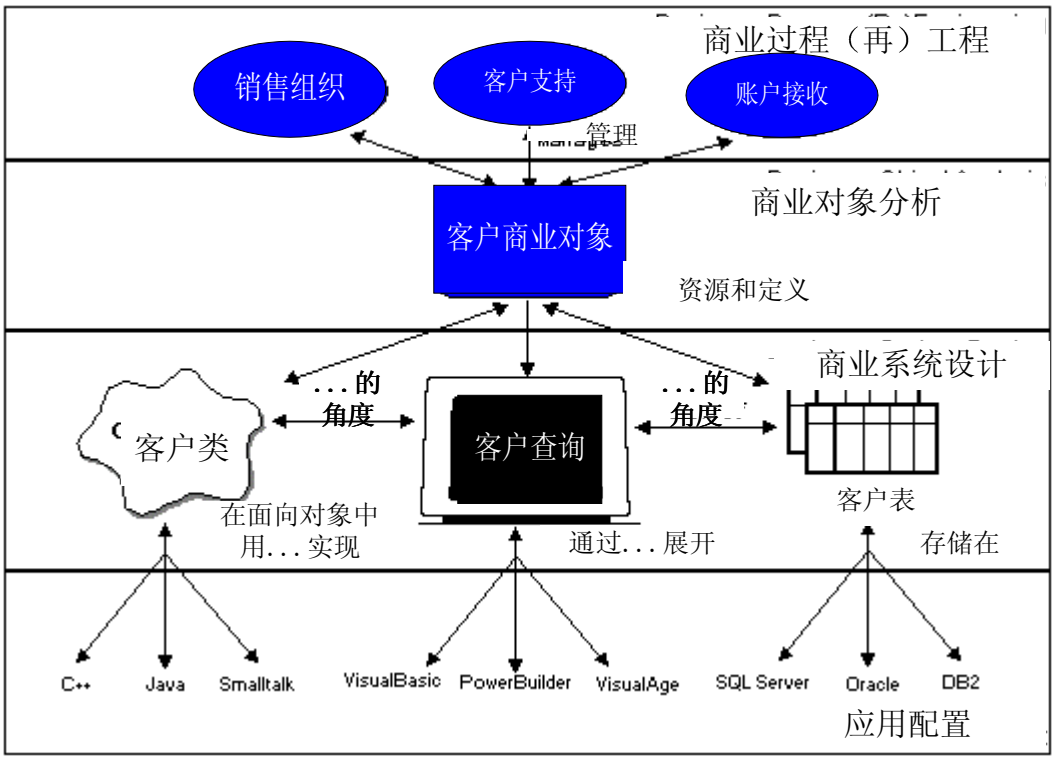

图 12-2 商业对象为在企业范围内配置（再）工程的商业过程提供概念基础

商业对象明确地把商业概念映射到系统，例如，在销售过程中有客户、客户支持系统和账目接收过程等。一旦定义了商业对象客户，就可以在企业内使用它，显示在商业和系统中，客户到底是谁，它为企业做什么。对每个现实世界的客户，在应用程序中都有一个自动的客户类、用户接口的客户表示和（或）数据库内的一个用户表（如图 12-2）。

可以用商业对象模型来配置软件，使之符合企业的定义。如果系统是用面向对象技术开发的，则商业对象和它们的性质将保留。

12.3　企业的建模和视图

要实现商业系统的“从概念到代码”，最开始是所研究对象的商业概念、过程和对象的模型，然后才是设计和实现软件系统来支持这些商业模型。随着商业需求的改变，模型也改变，最后代码也改变。

商业建模所解决的关键问题也就是以下经典问题：谁？干什么？什么时候？什么专访？为什么？怎么做？对这些问题的回答将有助于改进过程和实现系统（如图 12-3 所示）。这就是商业建模关注的最基本问题。

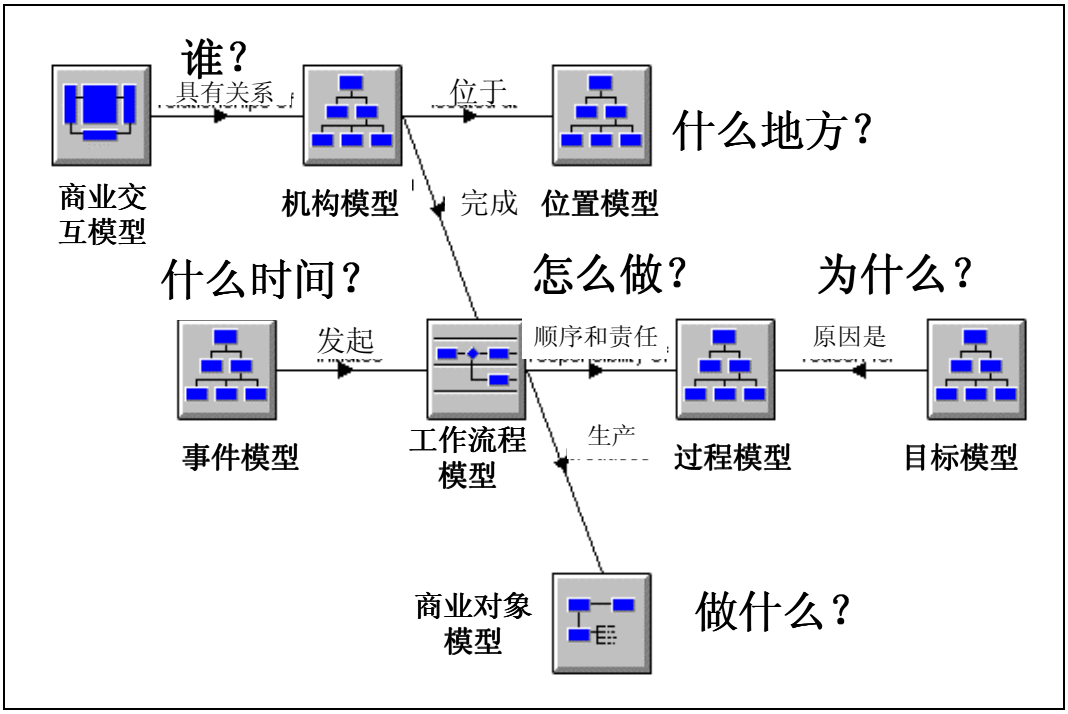

图 12-3　商业模型所回答的重要问题

为一个复杂的商业系统建模要求使用多种不同的视图，每种视图分别注重商业的一个侧面，通过一组图有时再加上一些文字文档对系统加以描述。通过视图之间信息的链接，可以在视图之间浏览，从而提供对整个商业系统更全面的描述。因此，每个视图都不是独立的，而是随着信息的不断增多，对系统理解的程度不断提高，不断递增的。视图不是单独的模型，它们是分别从某个特殊角度描述商业，它们组合在一起，成为整个企业的架构。

本书认为，描述商业系统的视图必须体现以下基本内容，它们是：

- 商业前景。该商业整体的前景，它为公司描述了一个目标结构，并揭示了为达到这些目标所必须解决的问题；
- 商业过程。该视图代表商业中的活动及其所创造的价值，并揭示了过程和资源之间为达到各个目标所必须的交互，该视图还显示了不同进程之间的交互；
- 商业结构。商业中资源的结构，例如商业的机构或生产的产品的结构；
- 商业行为。每个重要资源的个体行为以及商业模型的过程。

我们根据不同视图模型的作用和抽象级别的不同，把它们分成以下几类：商业模型、系统设计模型和实现模型。其中，商业模型分为商业概念、商业动态、商业对象。商业概念由目标模型、机构模型、位置模型、过程模型表达，商业动态由商业交互模型、工作流程模型、商业用例模型共同表达，商业对象由商业对象模型、子类型模型、状态模型共同表达。系统设计模型分为组件结构、组件动态、用户角度。其中组件结构由类模型、类层次图、关系表模型表达，组件动态由交互模型、对象消息模型、方法模型表达。用户角度由系统用户例图、用户交互模型、系统/子系统模型表达。实现模型包括组件模型和平台模型。

12.4 商 业 模 型

为商业建模，分从商业动态、商业概念和商业对象三方面建立模型。

12.4.1 商业动态

商业动态模型反映了企业所完成的商业过程，它们显示了机构之间的关系，显示了为响应商业事件所进行活动的细节，有助于用户参与分析，并把分析的结果反馈回管理层。包括以下几种模型：

- 商业交互模型（Business Interaction Model）。商业交互模型描绘了商业的全部或部分，显示了商业内部的机构界线，以及这些机构内外的交互。它定义了商业领域的范围，提供快速开发商业战略视图的方法。可为与客户、提供商、对手的交互和关系，以及主要的内部交互机构进行建模和评估。商业交互模型的重点不在于定义机构中的单元，而在于不同单位之间货物和信息的交流和传递（如图 12-4 所示）。
- 商业用例模型（Business Use Case Model）。商业用例模型把商业交易显示为商业环境（市场、机构和任务）中的“角色”与商业领域的一部分进行交互。Ivar Jacobson 在他的书《对象的长处》（The Object Advantage）中是这样描述商业用例的：是系统中的一个交互序列，目的是为商业系统的角色个体产生一些可测量的值。用例模型把商业领域划分成一组事件区分的交易，这样的划分对与商业领域进行交互的角色非常有用（如图 12-5 所示）。

工作流程模型（Workflow Model）。工作流程模型用商业过程的组件活动、以及活动之间的工作流程来描述商业过程。通过可视化地把活动分配给相应负责的机构单元，便于

分析机构（这是过程效率低下的主要原因）之间事务交互。工作流程模型也非常有效地把单独的用例可视化地集成在一起，成为一个连续的过程流（如图 12-6 所示）。

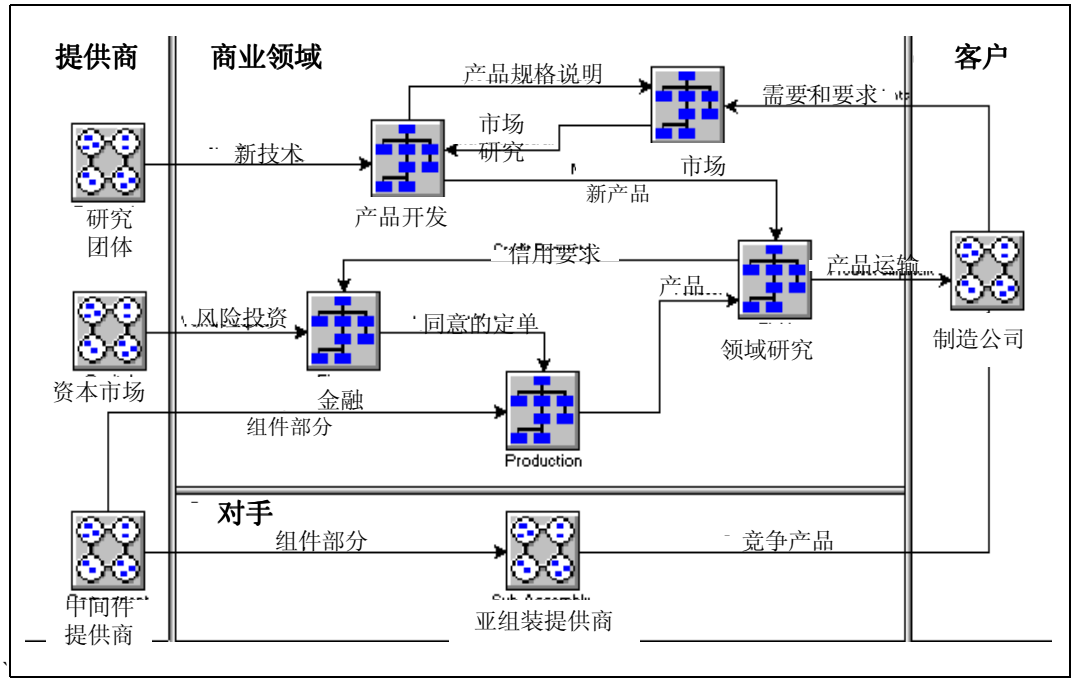

图 12-4　商业交互模型显示了机构单元之间主要的关系。商业交互模型显示的关系可以作为指示图，对主要的商业过程进行修改和定义

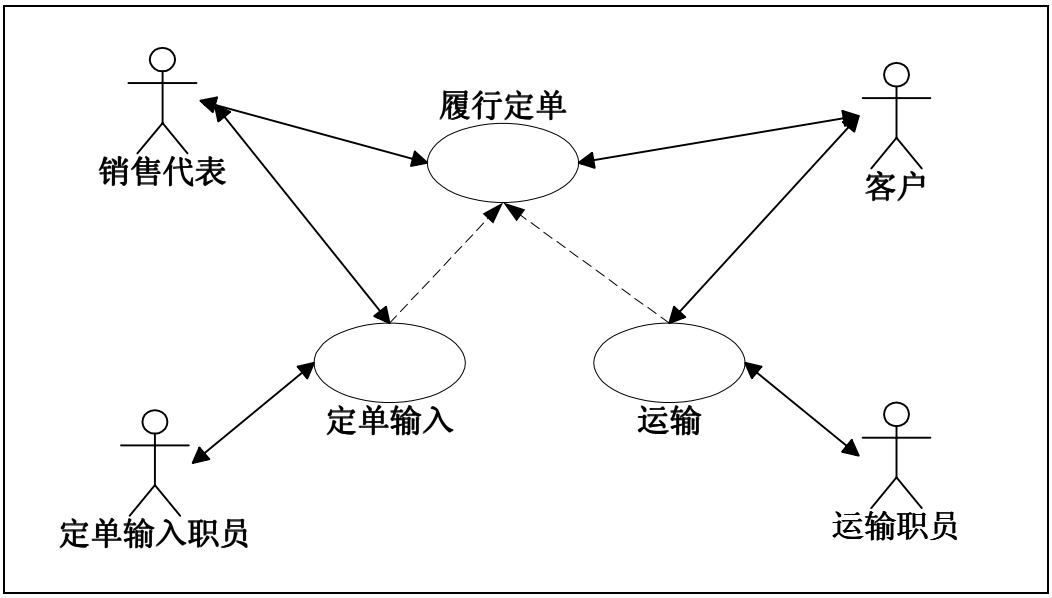

图 12-5　“履行定单”的用例模型显示了与它交流的角色以及对它进行扩展的用例

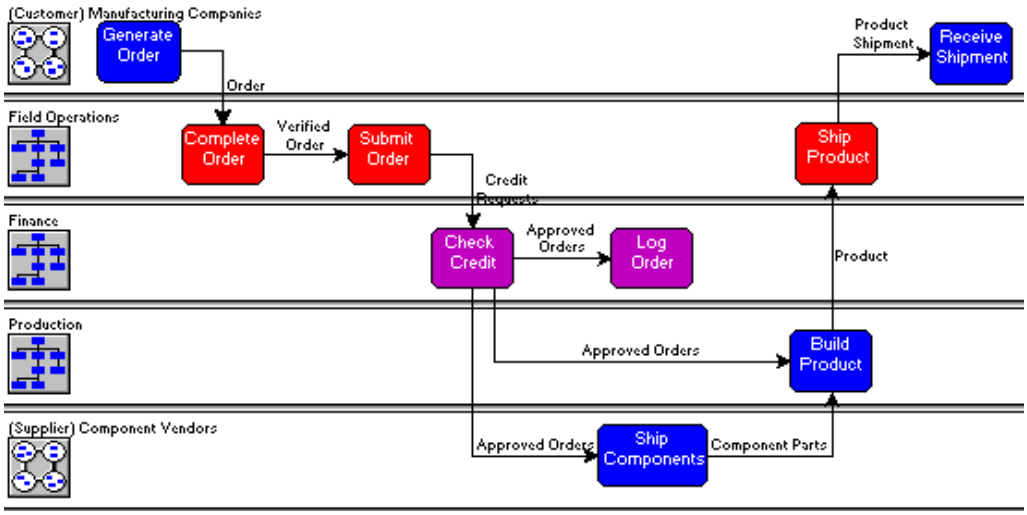

图 12-6　过程级工作流程模型。在大多数商业过程（再）工程或过程建模中，流程是最主要的用于分析的模型

工作流程模型是商业过程再工程和过程改进中应用最广泛的模型。一旦完成了工作流程模型，将对每个活动和工作流程进行检查，确定它在过程中的时间或成本，以确定性价比最好、最有竞争力的商业过程。如果需要，还可以把过程继续分解，以便注重于细节。工作流程模型是对商业过程或机构结构进行重新设计的基础。

12.4.2　商业概念

商业概念模型把“谁？干什么？什么时候？什么专访？为什么？怎么做？”的商业核心概念组织成一个层次结构，还是组织完整性和一致性检测的手段。

目标模型（Goal Model）。目标模型分层次显示了商业所要达到的目标，对有助于达到机构目标的商业过程和活动的定义是有效而增值地进行商业（再）工程的基础。所写出的目标被映射到其它商业概念来帮助评估优先级别以及设计正确的方案（如图 12-7 所示）。

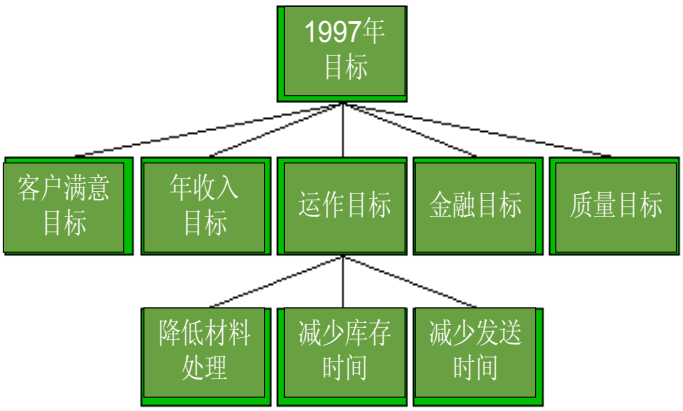

图 12-7　目标模型显示了目标的分类，除了可以把目标分配给目标商业过程和活动之外，还可以分配给机构和售货员

- 机构模型（Organization Model）。用机构模型可以用一种分层的风格表示企业当前的组织和任务。但是，对组织结构的重新是商业（再）工程的主要任务，出现在这个模型中的代理必须映射对企业的目标和活动上。谁做什么，为什么做，都是商业再工程要考虑的重要问题（如图 12-8 所示）。
- 位置模型（Location Model）。位置模型显示了企业所关心的不同地理位置。交差指向机构和任务所位于的位置，可以看出过程到底是在哪个位置上完成的。这种映射有助于从地理上对分布式系统的组件进行划分（如图 12-9 所示）。
- 过程模型（Process Model）。过程模型显示，把商业领域分解成商业过程和活动。这为商业活动给出了一个从一般到特殊的图像，有助于在不同的抽象级别上分析功能（如图 12-10 所示）。

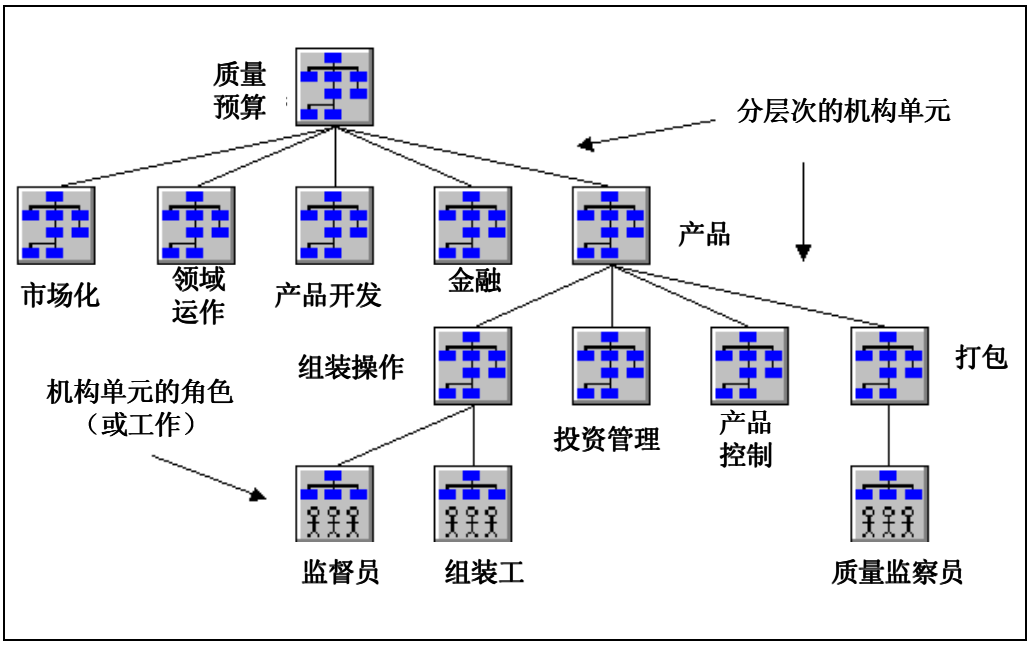

图 12-8　“质量新产品”组织模型。一直到任务（工作）层的组织都可以定义

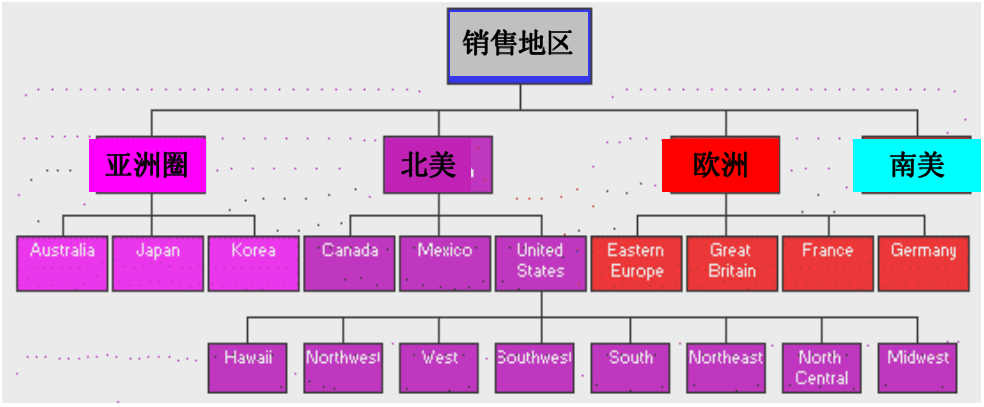

图 12-9　位置模型反映了“销售地区（Sales Regions）”的组织。可以把组织映射到位置进行配送分析

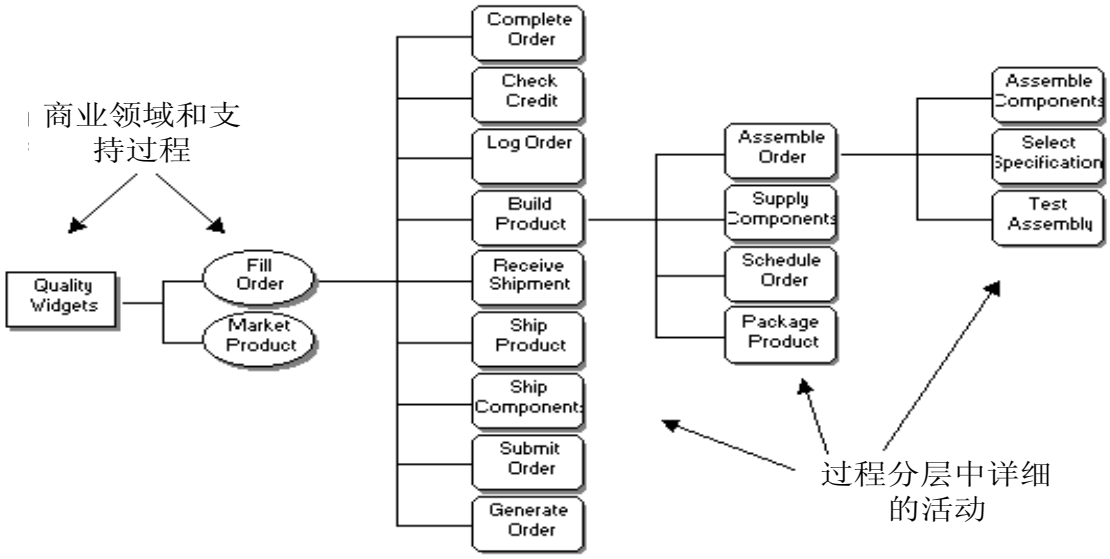

图 12-10　商业领域可以被分解成商业过程以及在商业过程中所完成的活动

12.4.3　商业对象

商业对象是人、位置、事物或其它具有商业过程的信息或提供功能的概念。所有的商业过程都使用、生产商业对象（如客户、定单、产品），或为其增值。为商业领域内的商业对象结构建模必须显示以下信息：

- 定义，说明某个类型的商业对象；
- 关系，说明商业对象之间的静态语义关联；
- 聚合，说明作为某个聚合的商业对象的组件的商业对象；
- 性质，显示了特征数据、行为和商业对象的状态。

- 商业对象模型（Business Object Model）。商业对象（或简称对象）模型显示了商业用户所关心对象的主要特征，以及这些对象之间的相互关系。商业对象模型独立于任何系统设计或实现，而体现了商业领域的语义概念和规则（如图 12-11 所示）。

对上面商业对象模型的语义进行分析，得到以下商业规则：

- 一个定单（Order）由且只能由一个客户（Customer）提交；
- 客户可以有一个到多个定单；
- 定单中有至少一个，可以有多个定单项（Order Item）；
- 一个定单项能且只能属于一个定单；
- 一个定单项可以定购至少一个，可能多个产品（Product）；
- 一个产品可以被多个定单项定购；
- 一个定单可以从至少一个或多个仓库（Warehouse）中提出；
- 仓库可以运输多个定单。

对每个定单，我们知道有以下事实：

- 定单日期（Order Date）和要求交货日期（Requested Ship Date）；

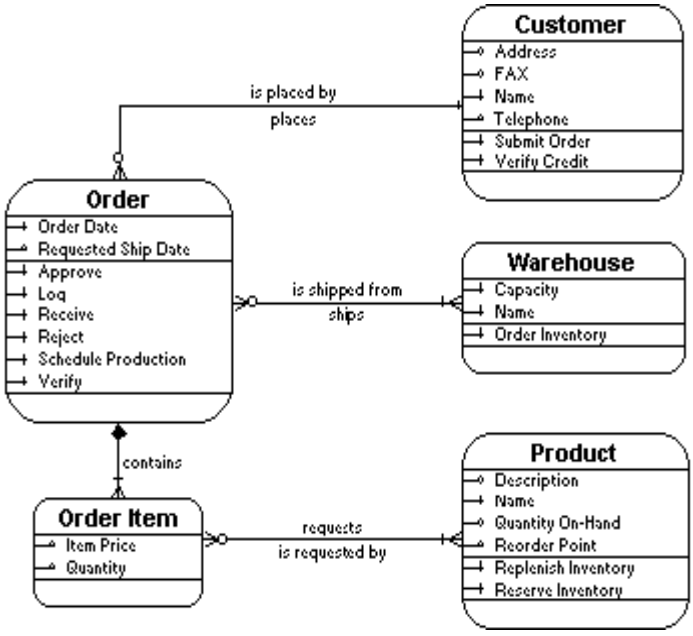

图 12-11　以 Order 对象为中心的商业对象模型。这是用于商业对象分析的最基本模型

- 提货的仓库；
- 下定单的客户；
- 定单上的定单项和要求的货品；
- 可以对定单实施的方法：同意（Approve）、记录（Log）、接收（Receive）、拒绝（Reject）、调度（Schedule）确认（Verify）。

其它有关客户（Customer）、仓库（Warehouse）和产品（Product）的陈述可以通过对模型的分析类似地得到。

- 子类型模型（Subtype Model）。子类型模型描述了商业对象定义的层次（超类型和子类型）。在分析中，可以把大量相似的商业对象划分成具有相似性质的子集，从而对商业概念进行更严格的分析。对超类型性质的继承是一种非常方便而且表达能力很强的分类方法（如图 12-12 所示）。

状态模型（State Model）。状态模型显示商业事件是如何影响商业对象进而整个系统（或组件）的状态。状态模型显示了一个商业对象的生命周期，它们用对象可能的状态以及状态之间如何转换来描述对象的动态性质。这个方法自动把对象状态的转换反映到对象模型中（如图 12-13 所示）。

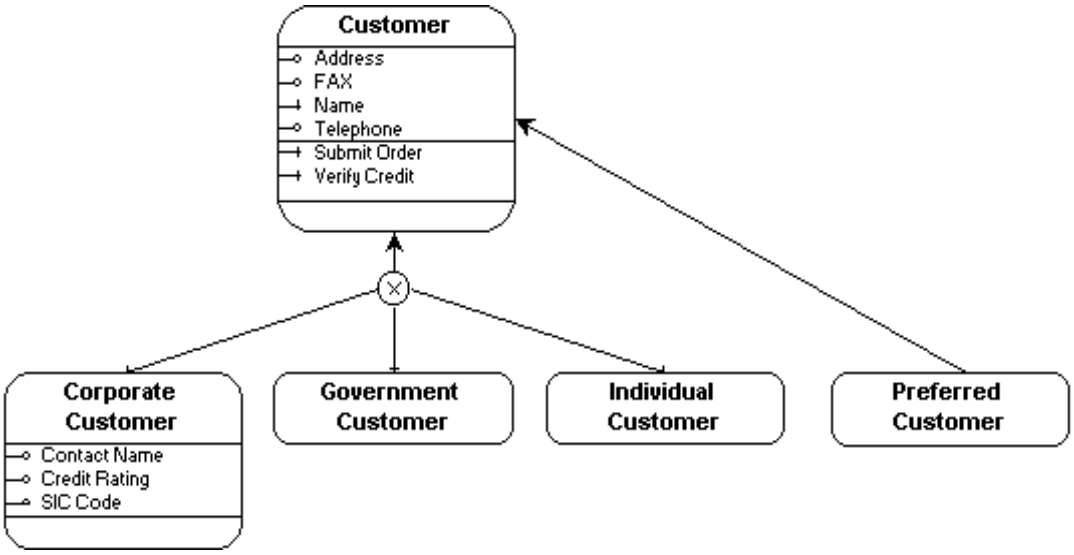

图 12-12　以商业对象 Customer 为重点的子类型模型。这个模型显示了从商业的角度对 Customer 的分类模式

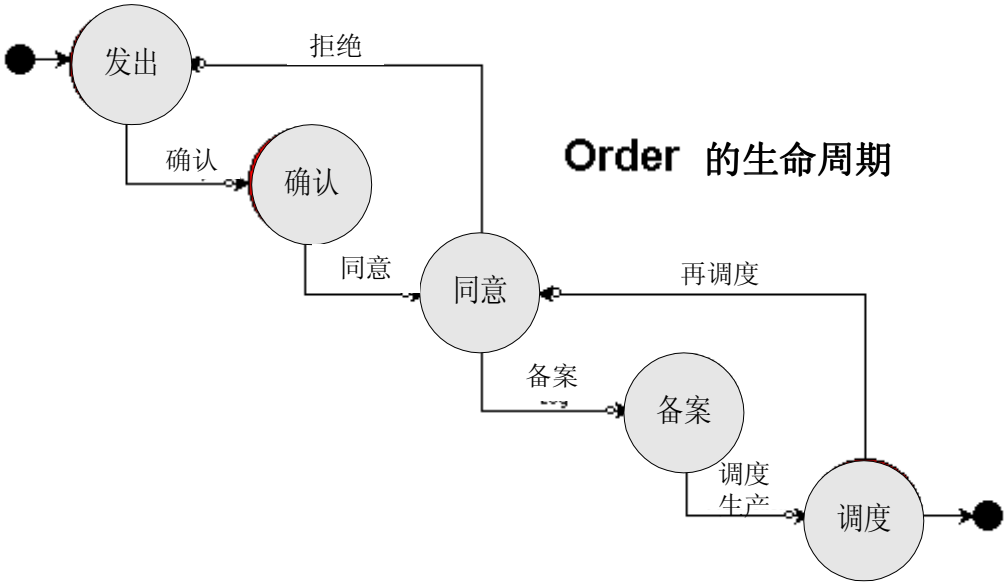

图 12-13　显示了商业对象 Order 的生命周期的状态模型

12.4.4　验证和集成

除了商业模型外，商业建模还需要其它一些能力，本节将作一简单讨论。

- 想定分析（Scenario Analysis）。商业分析人员应该能够分析出商业过程的各个活动的成本和所需要的时间，通过指定各种想定选项，考查想定对商业过程成本和效率的影响，进而对活动进行评估，最终实现对目标的量化分析。
- 商业规则规格说明（Business Rule Specifications）。精确的商业建模要求任何建模

概念或对象都有明确的规格说明以及商业规则。多个对象对商业规则的重用使这些规格说明尤其有用。

- 管理报告（Management Reporting）。在商业建模中，适于向管理层报告的高层报告和图是非常必要的，这在 BPR 的开始阶段尤其有用，因为项目团队正在准备结果，取得一致，希望获得管理层的同意。
- 符合战略方向（Alignment with Strategic Direction）。商业模型应该为建模决策（和最终系统）提供可跟踪的修正，符合管理目标和战略。要达到这一要求，对那些符合企业战略和战术目标的商业过程、活动以及可交付使用产品的可跟踪性是重要的方法。
- 打包的软件评估（Packaged Software Evaluation）。在软件包中保留商业模型完整的文档，使之成为对商业数据和规则进行评估的依据，也是确保打包软件符合需求的重要方法。在得到软件包之前，就能够知道定制的程度（以及最终的成本）。如果软件包中还有参考模型，那么把商业模型映射到参考模型上，是一种量化的对软件包的评估和确定成本的方法。
- 与设计集成（Integration with Design）。如果设计人员重视商业模型，并把它们用作系统设计的蓝图，那么把概念商业对象映射到设计层组件是非常关键的。这需要开发一个逻辑的系统设计。

12.5 系统设计模型

“从概念到代码”战略一个最关键的目标是把现实世界的商业与自动化计算机世界的应用程序系统同步起来，面向对象设计原则允许我们设计出能够模拟现实商业对象的系统组件（如图 12-14 所示）。

商业过程和对象模型是由商业管理人员和拥有过程的人员定义的，它们成为需求以及系统开发初期的基本模块。一旦把商业对象作为一个类实现，就可以把复杂（动态）的商业规则及功能封闭在对象类中。

系统设计（或逻辑设计）显示了计算机系统作为互相交流组件的集合是如何工作的，系统的设计应该是对商业模型中开发的商业过程、对象和规则的直接扩展，并且应该具有一定的抽象层次，以便把设计组件配置在不同的应用开发环境中（client/server 模式和关系，面向对象和 C++，Java 或 Smalltalk）。另外，设计组件应该能够具有可跟踪性，能够回溯到它在商业模型中从商业的角度定义的概念层。

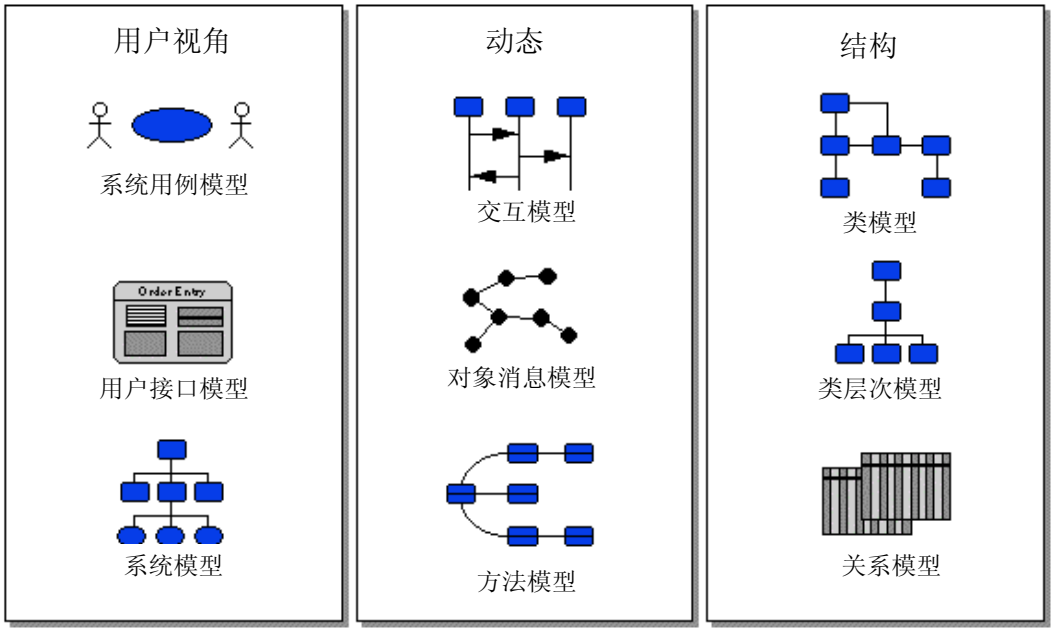

图 12-14　面向对象设计为定义支持商业对象的数据、行为和规则的组件提供支持

12.5.1　组件结构

系统设计的基础模块是用类指定的。类扩展了商业模型中定义的概念和对象，并增加了设计级的性质。在系统设计中，成为设计类的草图的概念化商业对象中心任务商业类。商业类定义了一些特定性质，比如得到 / 设置方法，创建 / 注销方法，收集方法，注册方法等。此外，关系模型是定义系统需要的持久数据的一种常用而易于理解的方法。

- 类模型（Class Model）。 类模型描述了系统设计中类的静态关系（如图 12-15 所示）。类模型包括以下内容：
  - 类的属性和方法；
  - 组件类的聚合；
  - 类之间的关联；
  - 把专有的类一般化为更一般的类。

类的其它类型（或版类）如用户接口、系统接口（代理）、持久性等，也都是是类模型在系统设计中指定的。

类层次模型。类层次模型显示了系统中类的继承。它的意图与分析阶段中的子类型模型类似，但是从不同的角度来进行的。在设计阶段，详细的类层次图显示了通过继承进行的明确的重用（如图 12-16 所示）。

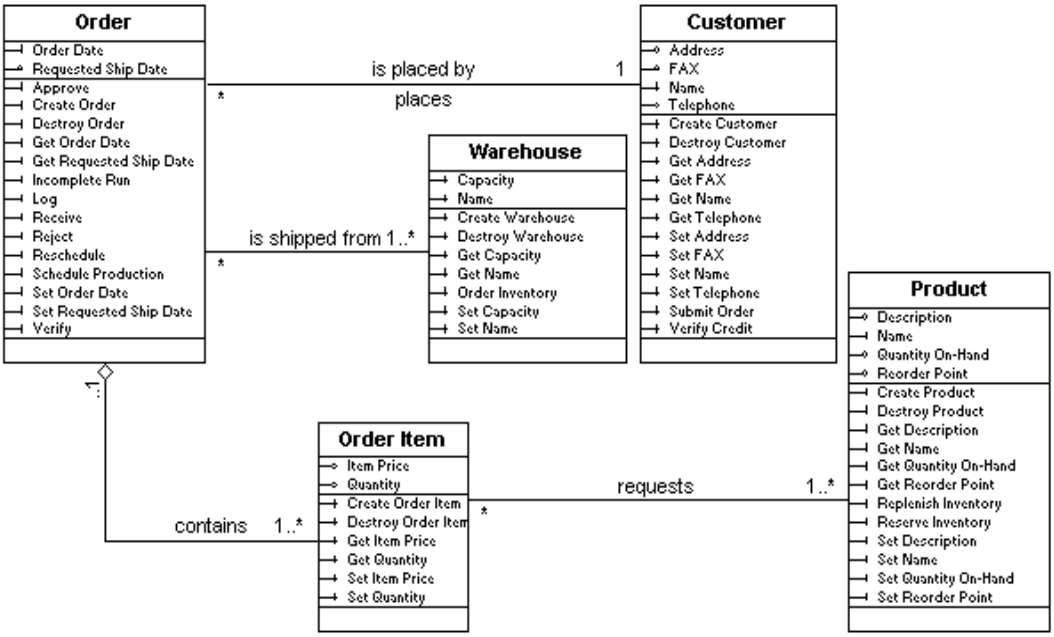

图 12-15　以定单（Order）的商业对象模型为基础，创建了它的商业类模型。这个类模型用设计专用的需求扩展的商业概念

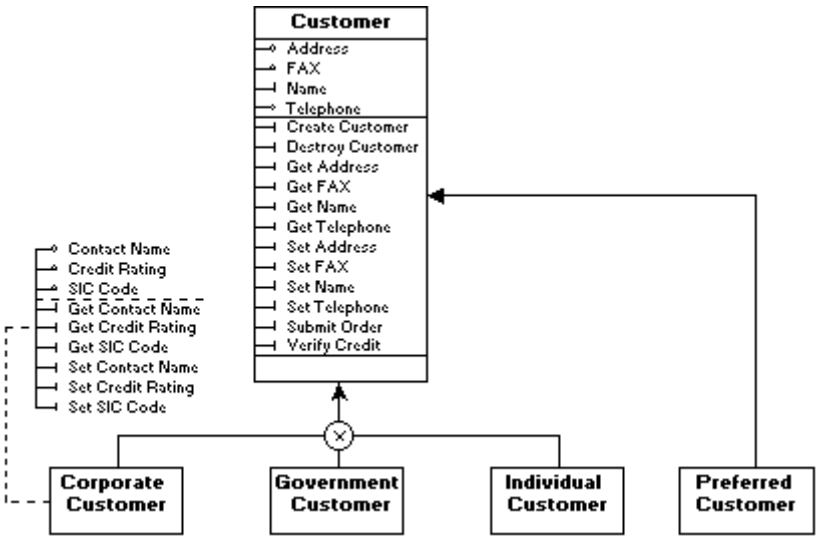

图 12-16　客户（Customer）类的类层次模型。它显示了在通用和专有的类之间的继承

- 关系模型（Relational Model）。 尽管许多机构都已经转向使用到数据库，但仍然有一些还在使用传统的数据库。关系模型以一种关系表的格式表示商业对象。这些模型支持标准化规则和关系理论（如图 12-17 所示）。

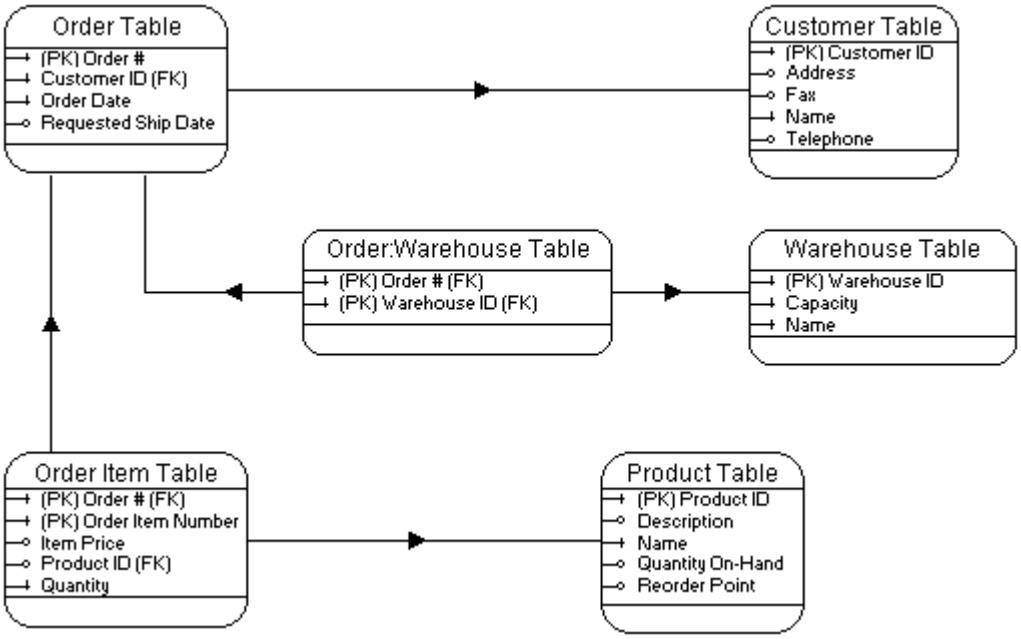

图 12-17　与定单（Order）的类模型相应的关系模型。关系模型是用来为系统数据定义持久的保存

12.5.2　用户角度

最终的用户是通过软件系统所设计的用户接口来观察系统功能的，这些用户接口为端提供了一个访问系统内部信息、启动系统交易及系统支持的“用例”的机制。用户接口需求商业类的功能来完成用例的意图。

- 系统用例模型（System Use Case Model）。系统用例模型显示了应用程序系统和它的用户（这些用户可能是商业领域的角色或其它外部系统）之间进行交互的点。系统用例模型提供了一种有效的方法，从用户的观点，找出系统完成的不同交易，从而把整个应用程序系统划分成一个个可管理的小块（如图 12-18 所示）。

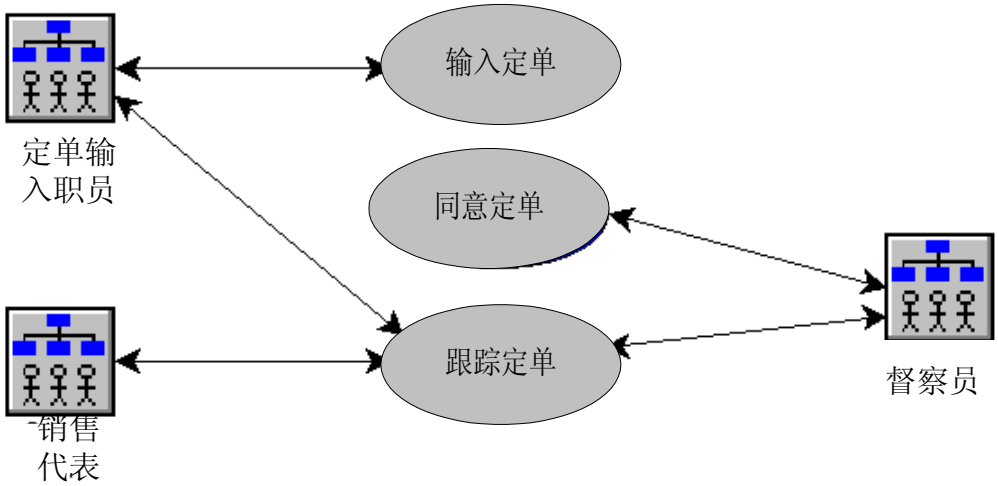

图 12-18　属于定单输入子系统的用例的用例模型。用例显示了用户与系统交互的点

通过设计“对话”（消息交换）来研究各个用例，用例内详细的消息传递是用交互模型表示的。

- 系统模型（System Model）。系统模型用一种分层次的结构把应用系统划分成系统和（用户所理解的）子系统。（子）系统可以指定它内部特定的用例（如图 12-19 所示）。

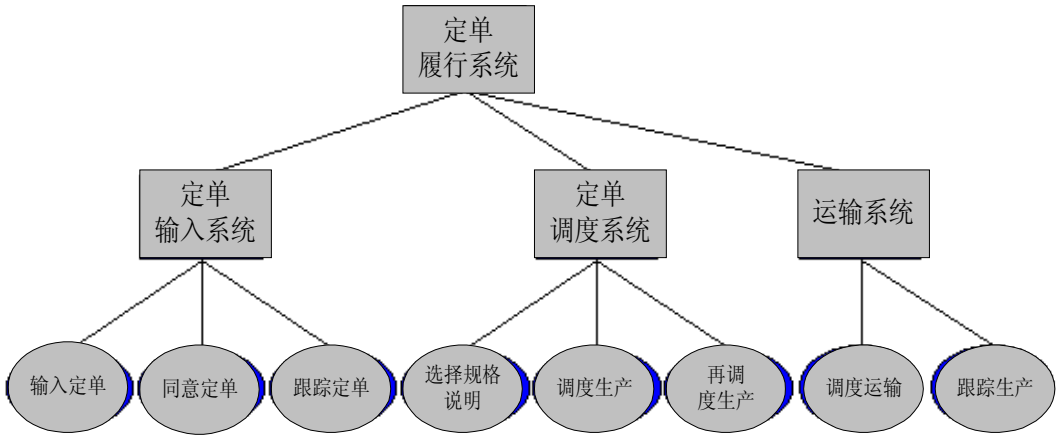

图 12-19　Order Fulfillment 系统的系统模型。这个层次结构是一个组织工具，帮助找到系统用例并对它们分类

- 用户接口模型（User Interface Model）。用户接口模型指定了端所能看到的系统的视图（组件和结构）。这个模型给出了指定类的性质的用户接口视图（如图 12-20 所示）。

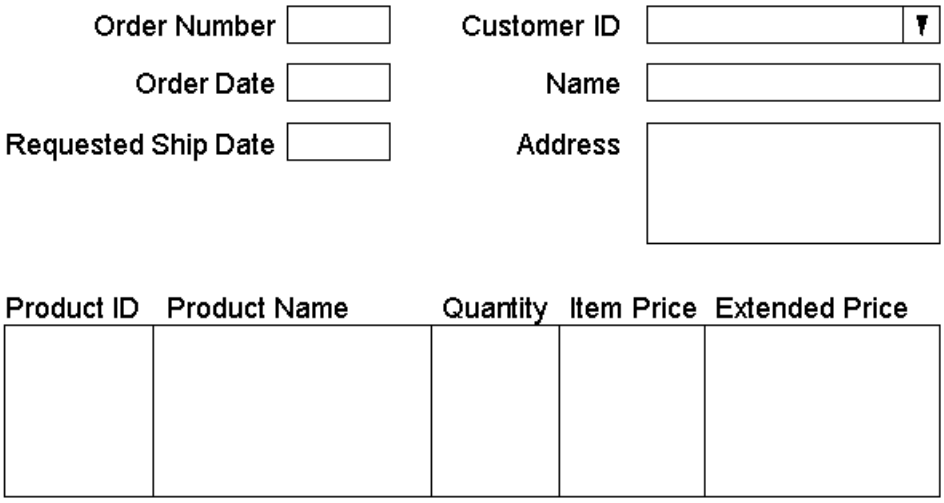

图 12-20　输入定单（Enter Order）用例的一个简单用户接口面板的例子

12.5.3　组件动态

组成系统设计的类指定了接口以及行为，而这些类的对象也通过传递消息进行通信。

类的这些动态和通信方面将通过以下几种组件动态模型（Component Dynamics Model）定义。

- 交互模型（Interaction Model）。交互模型显示了系统中的对象是如何通过消息的传递来进行协作，完成特定用例的。这种模型在显示不同类型的类如何通信时特别有用。例如，一个用户接口类可能会需要某个商业类的功能，因而会向外部系统一个代表该功能的代理类发出一个消息（如图 12-21 所示）。

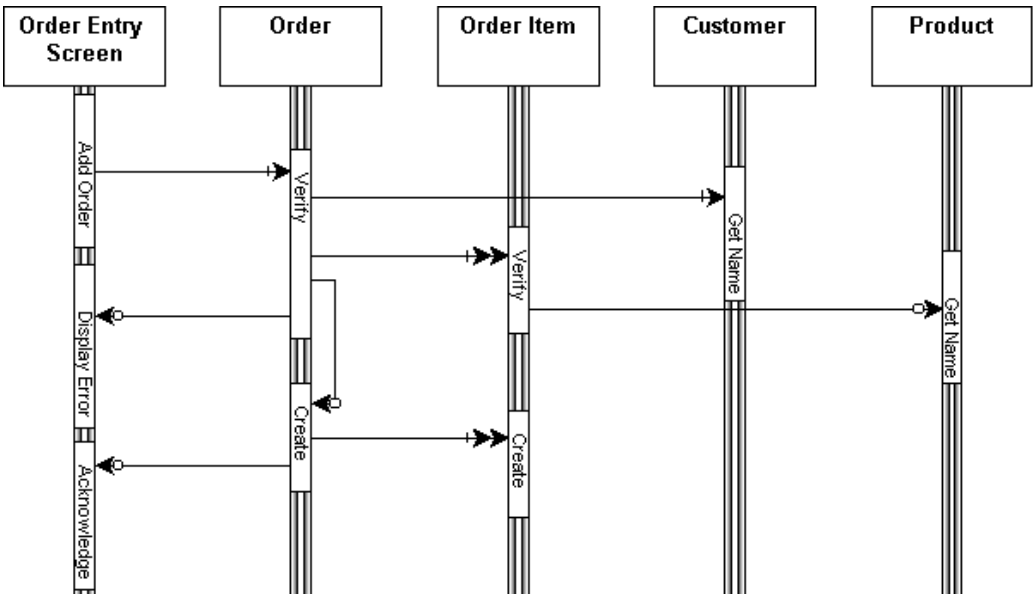

图 12-21　Order Entry 的一个交互模型。交互模型会收集面向交易（transaction-oriented）的细节信息，并图示出为完成一个交易（或用例）而进行的消息和事件的序列

- 对象消息模型（Object Message Model）。对象消息模型是类模型视图的另一种选择，但在这里显示的不是类的语义关联而是它们之间传递的消息。根据设计人员的需要，可以为同一类同时创建这两种模型，也可以只选择其中任意一种（如图 12-22 所示）。

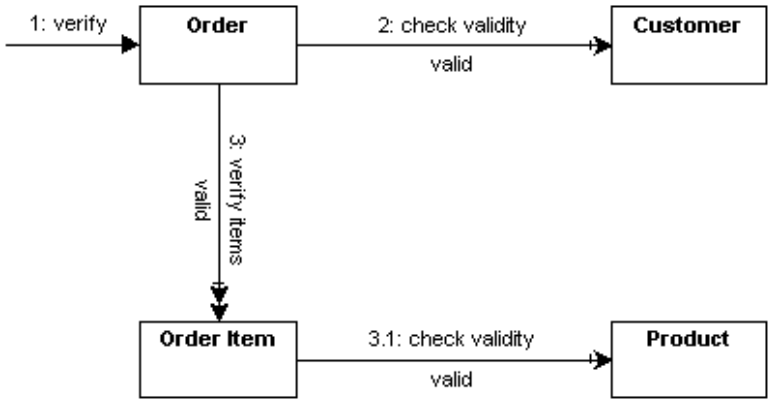

图 12-22　Verify Order 想定的对象消息模型。对象消息模型显示了为完成一个任务，不同类之间的消息序列

- 方法模型（Method Model）。方法模型显示对等对象之间的协作。方法模型的结构通过控制的方法揭示了针对一个事件或交易的一种消息模式，此外，它还说明方法之间为表示协议而进行的消息传递。逻辑约束和控制可以帮助转换成其它可用于实现的脚本语言（如图 12-23 所示）。

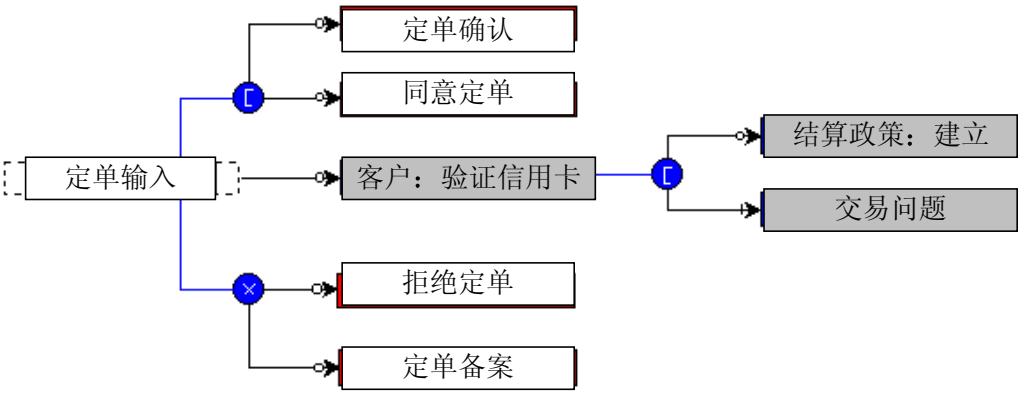

图 12-23　接收定单方法的方法模型。逻辑约束（与、或等）可以产生方法脚本或代码

12.5.4　验证和集成

除了系统设计模型外，在设计系统解决方案时，还需要其它一些功能，本节就做一个简单讨论。

- 商业映射和可跟踪性。基于商业对象的系统组件应该能够跟踪回它们的商业需求。没有需求可跟踪性，那么当商业需求发生变化时，系统的设计和实现就毫无意义。对需求的可跟踪性是一种分析商业内变化、快速识别其对系统影响的重要手段。
- 端通过原型进行确认。原型法是用户对商业过程和商业对象进行确认的非常有效的方法。在对商业需求进行确认后，原型向系统实现进行了转换。它们是端用户分析过程的一部分，并且是系统最终实现的基础。已经有一些算法把对象、关系、属性和方法映射到用户接口和 GUI 控件，可以把商业对象（以及类和关系表）翻译成应用程序开发环境可识别的一种格式。根据优先级的高低，所产生的原型应该包括：
  - 表格或面板，代表从用户观点看的商业对象；
  - 提供基于商业对象间关系的导航的菜单项；
  - 代表商业事件的菜单项和对话框；
  - 反映对象状态的恰当的用户接口；
  - 实验数据。

12.6 实 现 模 型

商业和系统设计模型是非常必要和关键的，但是应用程序代码和所实现的系统却是最终目标。

12.6.1 实现模型

- 组件模型（Component Model）。组件模型显示了如何把设计类、表和用户接口映射到可配置的组件中（例如头文件，源文件），以及这些组件之间的相互关系。
- 平台模型（Platform Model）。平台模型显示了可用来配置系统的物理设备（工作站、网络、服务器）。可以把运行时组件分配给物理平台，说明系统的实现。平台模型也可以独立于其上运行的组件，显示底层的软件和硬件。

12.6.2 配置

为概念商业和系统模型建立计算机系统得到了 Client/Server 开发环境的强大支持，如 Visual Basic 和 PowerBuilder，以及复杂的关系 DBMS。实现和配置系统组件的策略对这些环境应该是通用的，而不要为某个特定的建模工具开发专门的开发环境。

C++ 生成。C++生成应该支持特定开发环境和类框架的特色。Visual C++和 MFC 是目前的主流，因而也是最重要的。

Visual Basic 生成。Visual Basic 是当今 Client/Server 主流开发环境。 生成 Visual Basic 模块和类模型是必要的。

DDL 生成。关系数据库是目前主导的软件世界。如果不能生成关系 DDL，那么对大多数系统专家来说，建模的用途就值得怀疑了。

Java 生成。Java 很可能成为基于 Web 开发的主流，并且已经开始成熟，而企业内部网也开始成为商业的首选。

其它。包括 PowerBuilder，Delphi，以及 Smalltalk 环境等。

12.6.3 双向工程

不久的将来，迭代和反复的开发方法将主导系统（和面向对象）开发，而对双向工程（round trip engineering，RTE）的支持将非常关键；否则开发人员的代码和实现将不使用设计工具，而设计也将很快偏离商业和系统模型所指定的实现。

C++；Visual Basic；DDL。C++，Visual Basic 和 DDL 的双向工程是最关键的。

Java。Java 所具有的应用潜力不容忽视，因此对它的双向工程应该具有一定的意义。

OLE/COM 和 CORBA。 设计工具应用支持 OLE/COM 和 CORBA 接口定义生成和反向工程。

其它。包括 PowerBuilder，Delphi 以及 Smalltalk 环境。

12.6.4 设计映射以及设计的可跟踪性

仅对当前系统中的组件进行逆向工程是不够的，逆向工程后的系统组件应该能够被映射对最高层的设计抽象，而这些设计抽象应该能够被映射回概念商业模型中（或“从代码

到概念”），以此来提供最有用的可跟踪性。

12.7　企业建模工具的一般架构

这一节将讨论一个企业建模工具应该具有的架构基础和特色，以及它们演化的整体方向。

12.7.1　多用户

企业级工具要求对多用户的支持，所要求的主要的多用户和管理特色包括（如图 12-24 所示）：

- 支持连接到多个用户的任意数量的知识库；
- 在对象级强制实施登记互锁协议；
- 版本控制；
- 帐户管理，配置用户权限和安全。

随着企业级建模工具的使用越来越广泛，它对项目团队越来越重要，将需要它具有在对象级进行动态实时更新的能力。

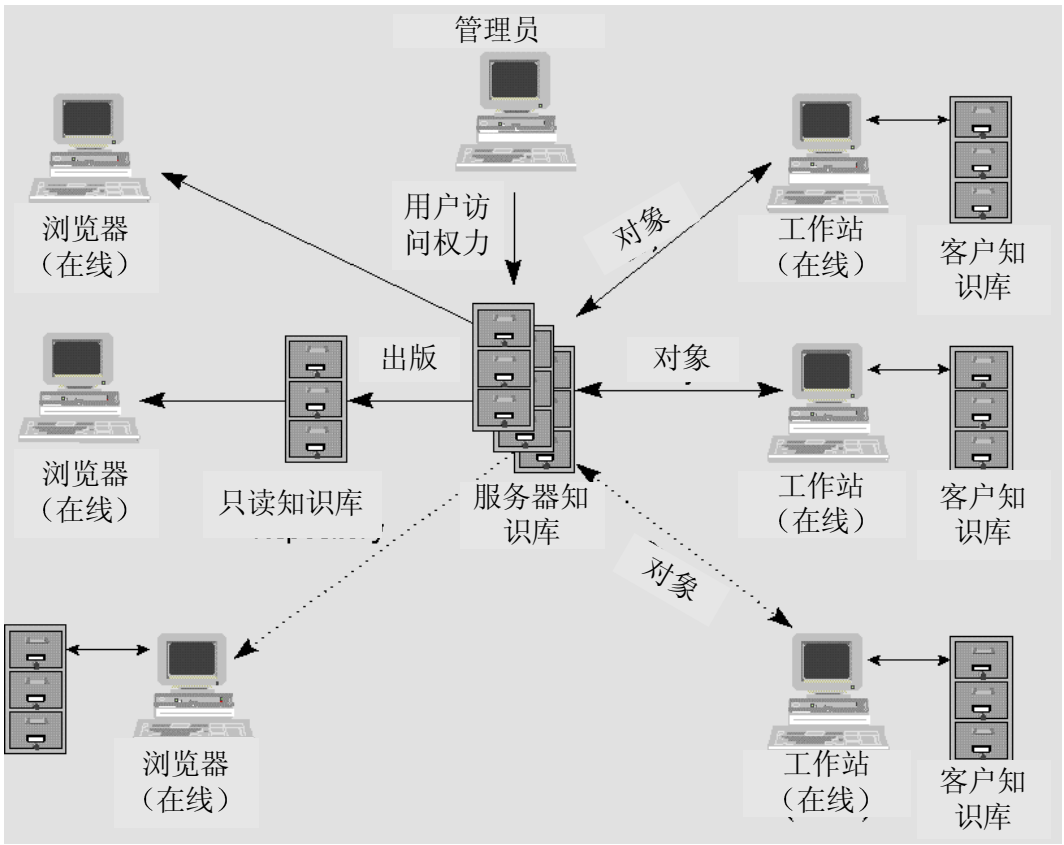

图 12-24　用于企业建模的、功能丰富的多用户环境

12.7.2 用户接口

标准的用户接口是关键的。

12.7.3 互操作性

当需要对工具进行集成时，要遵循 OMG 有关数据交换的面向对象工具互操作性标准。工具具有广泛而通用的可应用性是非常重要的；标准 API 允许习惯应用和客户报告。基本的互操作性包括：

- OLE 自动访问到模型和用户接口。这个接口特性将允许读和写，允许外部程序通过工具的用户接口，对系统做任何改变或访问任何信息。
- COM 访问模型。这个接口必须利用开放的 API 来检索和更新储存在工具模型中的知识库。

12.7.4 组件的架构

理想的架构应该允许工具把它分成一些组件，从而为不同类型的用户提供不同的功能子集。而且，组件架构应该可以作为 Java 的小应用程序，通过 Web 浏览器在远端执行。

具有三层功能层的划分是恰当的：

- 商业建模，允许商业用户定义自己的战略的过程需求。进一步的划分可以提供纯 BPR 组件，只允许过程建模；
- 系统设计的目标是开发原型；
- 为所支持的多种开发环境开发组件。

另外两层划分将支持多用户：

- 服务器（或管理员）；
- 客户（或工件站）。

12.7.5 方法支持

主要包括以下两类方法：

- 工业标准。企业建模工具应该提供对常见软件方法的支持，UML 是目前主流的方法和标准。
- 定制。为了适应工业标准的变化，允许客户的喜好，建立公司的标准，工具所支持的方法应该能够由端用户进行定制。这种定制允许开发公司标准，并无缝地集成到基线方法中。

12.7.6 可扩展性

模型的图表示具有很高的可扩展性。包括：

- 改变建模选项的能力，如对象形状、颜色、大小和几何形状；
- 取代对象的位图表示，为商业概念创建更直观的 BPR 模型表示；
- 在元模型中增加对象性质的能力，这就具有收集客户环境专有信息的灵活性；
- 用图和注释对模型进行修饰的能力，有助于表达模型的意图。

12.7.7 报告

- 客户报告生成器。支持生成客户报告，或者是内部报告生成器，或者是集成良好的外部报告生成器；
- 质量保证（Quality Assurance，QA）。在企业建模工具中有一套 QA 工具是非常重要的，一些最重要的工具包括：
  - 根据底层语义模型的规则进行一致性检测；
  - 拼写检查保证拼写正确；
  - 完整性检查确保所有要求的信息都已经保存。

12.8 企业建模工具概览

企业建模工具所需要具备的条件很多，但在不同的应用领域是不同的。然而，它们都应该具有技术架构和集成前景。这一节对各种为企业建模提供某种层次支持的工具进行分类，其中新的面向对象分析与设计（OO A&D）工具以及传统的数据库和实体——关系（ER）建模工具是当今的主流，对每个工具组都分析了它的主要或例子工龄，以及一个简单的评估。

12.8.1 作图工具

作图工具：Visio，ABC Flowcharter，Corel Flow。目前市场上有大量支持商业作图的工具。可把这些作图工具与建模工具做对比，两者之间主要的不同之处在于，在建模工具中，“模型”具有语义解释，并且是一致的，系统内部理解的，而作图工具中的“图”只是图形，需要设计人员解释它们的意思。

建模工具的基础是一个知识库，允许在不同的环境下而不是在特定的图中重用对象。知识库可以帮助确认模型的结果，把模型转换成其它格式（如系统设计），重用一些模型的对象和组件来建造其它的模型。与之相反，商业作图工具对所作的图没有语义的解释。所以不存在重用、扩展或把一种图转换成其它格式的概念，只能由分析人员手工进行。

12.8.2 过程建模和仿真工具

过程建模和仿真工具：BPwin, Optima, Promodel。对于许多“拥有”商业过程又想对他们的商业过程进行分析的人来说，具有复杂的过程仿真器的工具通常太复杂，使用的成本太高。要想正确使用它们，必须具有排队论、统计和运筹学等方面的知识。相反，稍微简单一点、容易使用的过程建模工具通常是基础一种专有化受限的方法，比如 IDEF。这一类工具中很少有集成的技术，如进行系统实现的对象模型。

12.8.3 传统 CASE 工具

CASE 工具：Key（以前 Sterling 软件公司的 ADW）Software，TI 的 IEF。传统的大 CASE 工具的主要目的是为了从模型中生成面向大型机的 COBOL 代码和数据库。Client/Server 技术使这些工具基本上都过时了，但它们的建模概念却被其它工具加以采纳

和使用。

12.8.4 数据库建模工具

数据库建模工具：ERwin, S-designor 等。这些工具一般都是很小的 CASE 市场的幸存者。它们开发有限的语义数据建模方案，强调数据库的生成。但它们原来的目的并不是用来进行企业建模的，因此很难适应过程建模以及商业和系统建模广泛的需求。

12.8.5 OO A&D 工具

OO A&D 工具：Rational, Select, Paradigm Plus, Intellicorp 等。这些工具随着面向对象方法的推广而开始流行起来。它们主要是为 OO 分析、开发人员以及 C++程序员使用的。它们使用的一些主要技术，如 CRC，过程角色、面向系统的用例等都使它们是非常适合用于企业建模的工具。

12.8.6 三层 Client/Server 工具

三层 Client/Server 工具：Forte, Dynasty。这些工具是用来解决在异构环境下配置 Client/Server 系统时出现的复杂而多样的技术问题。在其中引入复杂的新的商业和过程建模概念非常困难，因为它们需要集成到 client/server 技术中。

12.8.7 功能总结

图 12-25 对前面所介绍工具的功能覆盖作了总结。当横条覆盖垂直线时，说明该工具对两个阶段做了集成。

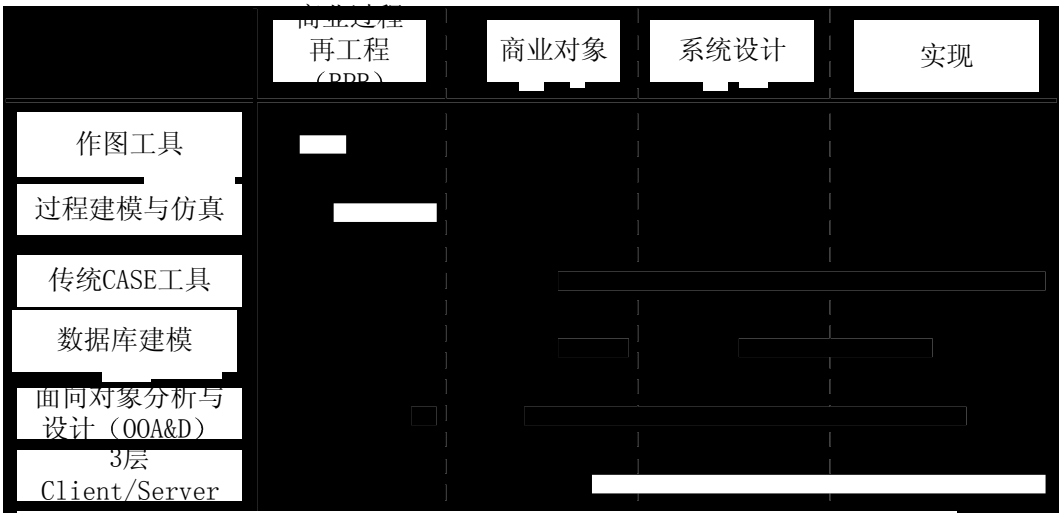

图 12-25 各种工具的功能覆盖

12.9 方法的基础

本章提出的商业建模是吸取了大量目前较为成功的建模技术，如：

许多商业过程再工程（BPR）概念和模型是以 Geary Rummler 和 Alan Brache 的工作为基础。他们的关系图（Relationship Map）是商业交互模型（Business Interaction Model）的前身，这一模型是企业的战略层视图。工作流程模型（Workflow Model）来自他们的过程图（Process Map）以及其它目前流行的许多方法。

把商业模型直接和系统设计以及最终的代码生成联系起来的概念来自信息工程（Information Engineering，IE）和 CASE 工具。IE/CASE 框架主要是 James Martin 提出的，集成了众多 20 世纪 80 年代出现的结构化方法的良好原则、实践经验、模型和技术，并形成一个整体的完全的生命周期方法。信息工程的来源可追溯到 IBM 在商业系统规划（Business Systems Planning）中的工作。主要的思想和模型来自 Yourdon（结构化分析，Structured Analysis），Chen（实体关系方法，Entity Relationship），Shlaer-Mellor，Myers 和其它参加 IE 工作的人。目前许多流行的商业和系统建模原则和表达方法都可以回溯到他们的工作。

商业对象模型（Business Object model）是语义实体——关系模型（Entity-Relationship）的一个自然演化，因为传统的结构化方法和基于信息工程的 CASE 工具而普及。通过面向对象（OO）到 ER 方法的扩展（通过方法进行封装和通过通用化进行分类）使商业对象模型成为一个语义更丰富的分析工具。目前，UML 把从用户角度理解的商业对象概念与 OO 系统设计和实现所要求的视图混和在一起。

本章的许多概念和模型(尤其是系统设计模型)是由 Booch，Rumbaugh 以及做 OO A&D 的人们提出的。目前 OMG 和 UML 正在对名词、图表示以及语义进行标准化，表 12-1，12-2，12-3 说明了 UML 对本章提出的商业建模的支持，同时还列出了其它支持的方法。

表 12-1　商业模型

模型	UML 规格说明	其它方法学
商业概念		
目标模型		一般商业原则
机构模型		一般商业原则
位置模型		一般商业原则
过程模型		一般商业原则；过程分解成结构分析
商业动态		
商业交互模型		Rummler-Brache 中的关系映射（Relationship Map）
工作流程模型	活动图	在 Rummler-Brache 中的过程映射（Process Map）
商业用例模型	用例图（当为商业事件建模时）	在结构化分析中的事件划分的 DFD
商业对象		
商业对象模型	类图（当从商业视角建模时） 对象图	语义实体-关系（ER）图
子类型模型	类图显示了继承（当从商业的角度进行建模时）	在语义 ER 图中的子类型
状态模型	状态图（当从商业的角度进行建模时）	David Harel 的状态转换图

表 12-2　系统设计模型

模型	UML 规格说明	其它方法
组件结构		
类模型	类图	
类层次图	类层次图	
关系表模型		数据库模型
组件动态		
交互模型	顺序图	
对象消息模型	协作图	
方法模型		模块结构图；伪代码；脚本语言
用户角度		
系统用户例图	用例图（当为用户－系统交互建模）	事件划分 DFD
用户交互模型		GUI 设计工具
系统/子系统模型		导航图表

表 12-3　实现模型

模型	UML 规格说明	其它方法
组件模型	组件图	
平台模型	配置图	网络图

12.10　为什么用 UML 进行商业建模

UML 定义了面向对象系统的标准表示，有利于有着不同知识背景的领域专家、软件设计人员、客户和用户之间的交流。用面向对象的概念和技术为商业建模有几大优点：

- 相似的概念。可以这样描述商业过程：不同类型的资源对象共同合作，达到某些目标的过程，而有关过程和资源相互之间的关系，则可以用规则来定义条件和约束。这些描述都可以映射到对象、对象之间的关系、对象之间的交互上来，例如可以通过创建静态和动态的面向对象模型来描述。
- 已证明性能优良的技术。面向对象建模和程序设计已经被证明非常适合开发大型复杂系统。一些新的技术，如模式等也已经被引入，有力地支持面向对象建模。
- 标准的表示方法。商业建模方法和技术需要一种标准的表示，这就是 UML。
- 很短的学习曲线。正如面向对象模型缩小了系统分析和设计人员之间的隔阂，面向对象技术和表示也将缩小商业建模人员和信息系统建模人员之间的隔阂。
- 提供一种观察机构或商业的新的易于理解的视角。传统的描述和观察一个机构的方法不能显示出商业是如何运作的。面向对象技术不仅能够体现出传统的机构体系结构，更能体现出商业过程，这是传统方法无法达到的。

这一节，我们讨论如何应用 UML 来表示商业系统的架构。根据工作流程管理在商业过程中重要性的特点，我们侧重讨论如何用 UML 描述商业系统的工作流程管理。

工作流程模型是商业过程的非常直接的表示，并且通常是领域专家在工作环境中相互交流得到的。工作流程模型是领域专家进行交流的工具，是他们验证相互之间正确理解的

工具，也是记录商业过程的文档。在很多项目中，人们都利用工作流程模型来完成以下工作：

- 商业过程的再设计，提高质量、降低成本；
- 形式化（共同化）商业过程，使整个机构内都使用一致的过程；
- 机构的再结构化，提高商业过程的效率；
- 对任务的再定义，拓宽责任范围或标识培训需求；
- 应用系统的再开发，支持成功地进行商业过程的再设计和机构的结构化；
- 把质量管理集成到当前的商业过程中。

因此，我们主要讨论如果用 UML 来表达普通的工作流程概念。

12.11　把工作流程中的概念映射到 UML

商业系统可以简单由过程和静态结构来描述。过程最直接的模型是一个活动或任务的序列，按照顺序完成它们以达到一个目标。UML 的序列图和活动图非常适用于描述商业过程的规格说明。静态结构如企业的结构图等，可以用 UML 的静态结构图如类图、对象图等描述。这一节一般性地介绍如何把工作流程概念映射到 UML，每个结构都可以用一个有恰当版类的 UML 符号描绘。图 12-26 表示了在 UML 中的商业对象、商业过程和团队角色的表示。

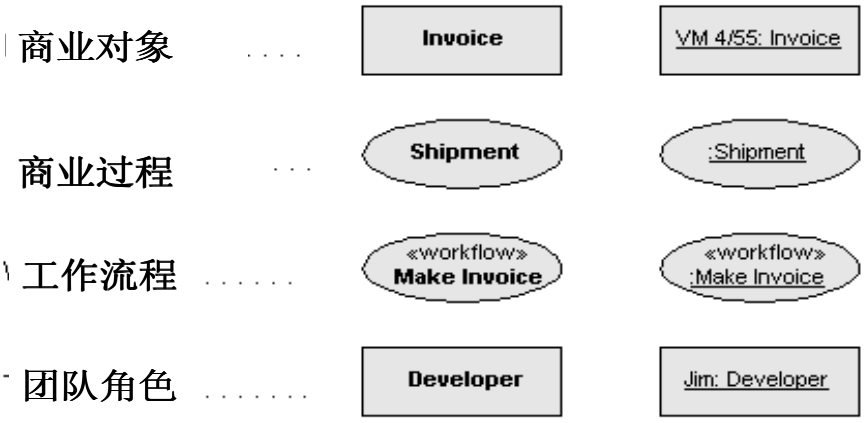

图 12-26　在 UML 中商业对象、商业过程和团队角色的表示

商业对象是用 UML 的类和对象表示的，类表示商业对象时，无需标识，如发票（invoice），对象代表商业对象时，必须有标识，如号为 VM 4/55 的 invoice。.商业过程是用用例和用例接口表示的，商业过程是用用例表示的，用目标、责任、前置条件和后置条件来描述。用例的实例是具体的事件序列，工作流程是自动化的商业过程，是用用例或用例的实例表示的，具有版类«workflow»。团队角色是用 UML 的类和对象表示的。类代表团队的角色，对象代表承担该角色的具体工人，所有的符号都可以有版类，如«business object»，«business process»和«team role»。每种版类可以用文字或一个特定图标表示。

图 12-27 是一个团队结构。团队的角色是用对象实例表示的，这样就可以在每个角色

中指定工人的编号。“客户满足团队（customer satisfaction team）”有三个开发员、两个测试员和一个产品管理经理，一个人承担用户教育角色。这些角色组合起来叫做客户满足团队，用包（package）表示。角色的组合也可以用一个对象表示——作为角色的复合。如果团队是用对象表示的，则团队和角色之间的关系是复合（composition）关系。根据 UML 的元模型概念，一个特定角色的实例不能同时属于多于一个团队。而如果用包表示，特定角色实例就可以同时属于几个团队。

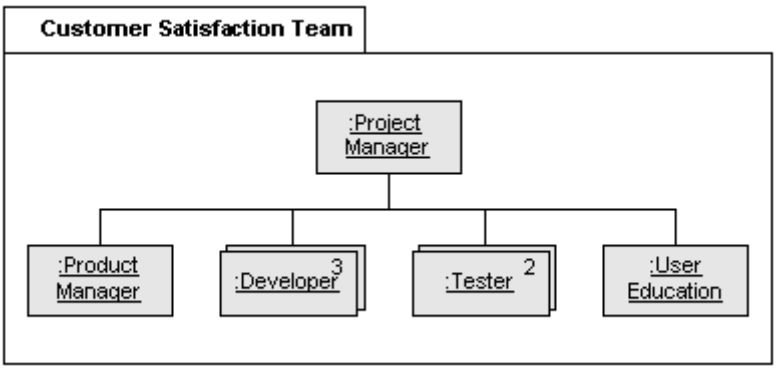

图 12-27　UML 静态结构图表示一个团队的结构

图 12-28 是商业过程的实例。角色“客户（customer）”下了一个定单，然后在“销售（sales）”部的某个工人对该定单进行确认，如果它是有效的，那么“销售（sales）”部的这名工人就启动另一个商业过程“公司运输某物品（company ships an item）”。这个类型的图在“UML 表示指南”中没有明确地提到，但符合 UML 的元模型。对象生命线顶上的符号代表分类器角色，如图 12-27 是角色，对象角色和用例角色。

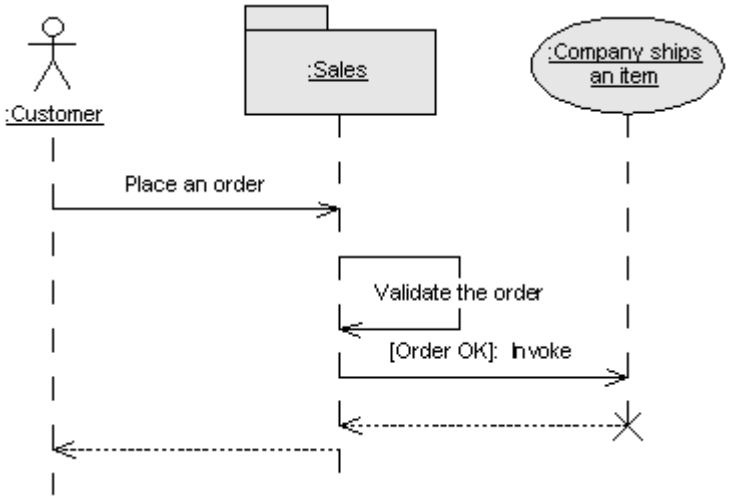

图 12-28　UML 的序列图代表了商业过程的实例

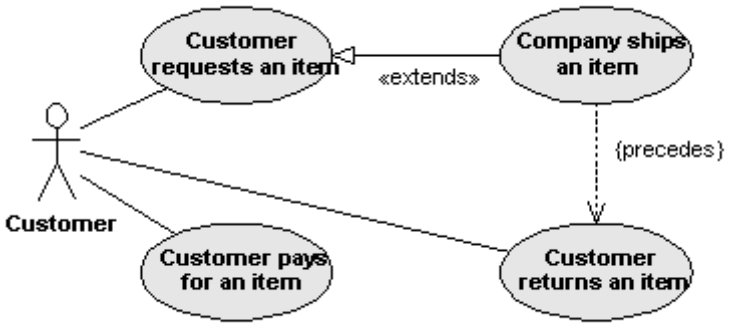

图 12-29　表示商业过程之间静态关系的 UML 用例图

图 12-29 的用例图，代表商业过程之间的静态关系。这些商业过程描述了机构（organization）和角色（customer）之间的协作。

用例实例的序列——可以用用例序列或协作图表示，见图 12-30 和 12-31 所示。与对象交互图中用消息序列表示一个想定相反，在用例交互图中是用用例序列来描述的。这种图是 UML 中唯一能够描述一个由一组其它想定的实例组成的想定。图 12-30 中发起的消息代表用例构造子，它们映射到从角色到用例的信号。可以按照每个用例的第一个操作来命名，如“发出请求（invoke request）”“调用运输（invoke shipment）”和“调用付款（invoke payment）”。

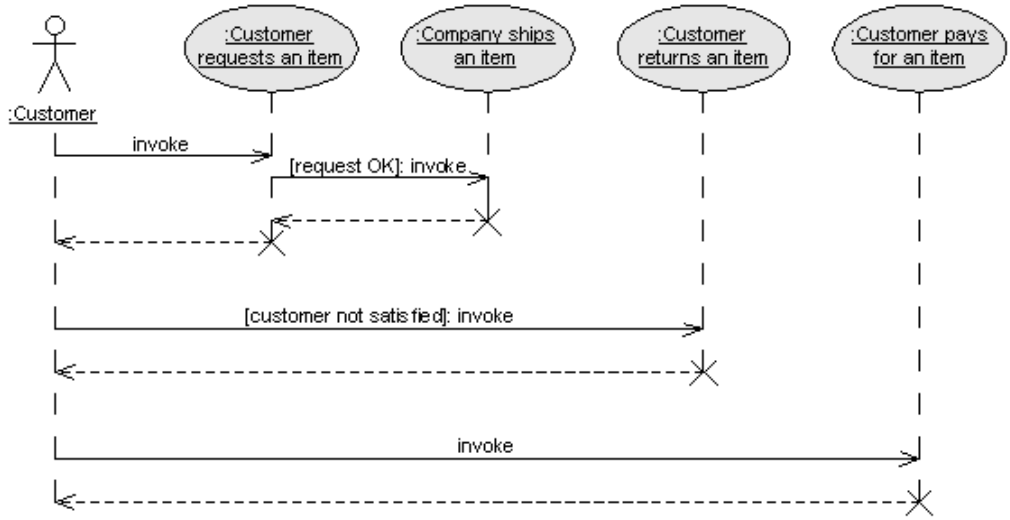

图 12-30　表示商业过程和角色之间交互的 UML 序列图

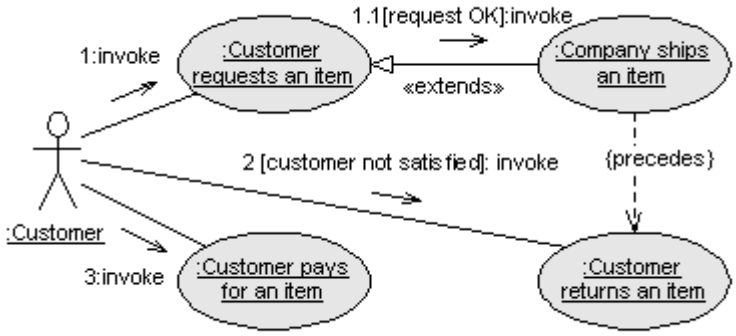

图 12-31　表达商业过程和角色之间的交互和关系的 UML 协作图

用例协作图也表示一个由一组商业过程的实例组成的想定。与用例序列图不一样，用例协作图显示了用例关系以及用例实例和角色实例之间的交换的消息。用例协作图如图 12-31 所示。

用例交互图只描述了由一组用例实例描述的典型想定，因此，它们不能表示所有允许的用例实例顺序。可以在用例包的生命周期中指定所允许的包中用例实例的顺序。用例包的生命周期是用状态图和一个活动图的 Backus 范式（BNF）表示的。在其中，状态、行动状态或 BNF 语句都映射到用例包中的用例。用例包生命周期精确地描述了用例包行为，但对于复杂情况，很难开发。用例交互图容易一些，但它只描述包中用例典型的想定。

图 12-32 中的 UML 活动图代表了用例包“订单管理（order management）”，这个活动图中的活动与图 12-29、12-30、12-31 中的用例对应。

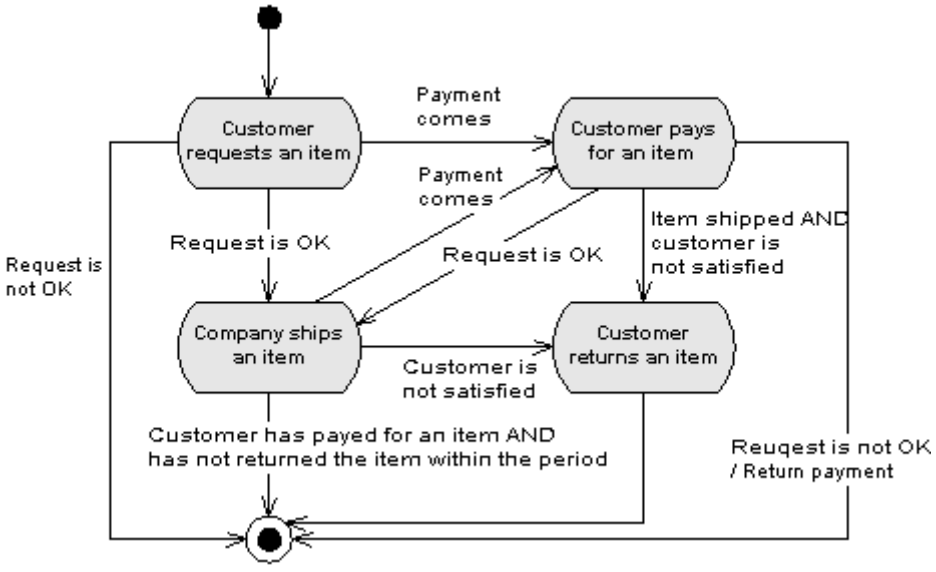

图 12-32　UML 活动图可以表达商业过程的顺序

注意 UML 元模型没有定义任何从状态或行动状态到用例实例的映射，这种映射可以在开发过程进行，与 Martin 和 Odell 方法相似，其中，子系统中的每个状态都指明子系统中的一个候选类。但其它开发过程可能以其它方式定义用例包生命周期。

用例交互模型和用例生命周期之间还有一个显著的区别——它们在项目知识库中的位置不同，并且它们与其他设计产品的关系不同。产品用例交互模型与产品用例模型有关，产品用例包生命周期与产品用例包有关，后者的抽象级别比对应的用例模型和用例交互模型高。请见图 12-33 和下一节的讨论。

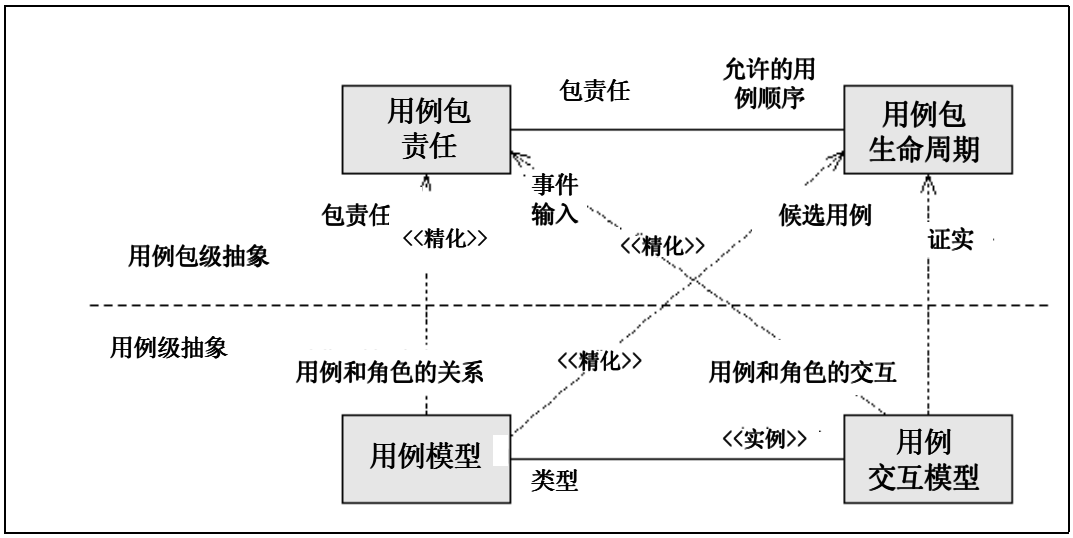

图 12-33　在项目知识库里的用例视图中产品之间的关系

12.12　设计产品的模式

在开发过程中，软件架构的设计和开发人员会有一些有关软件产品（artifact）的信息，比如用例、软件的架构、对象的协作和类的描述。这些信息可能十分抽象，例如产品的前景，或者非常具体，例如源代码。

我们必须认识到，在设计的产品和它的表达之间的区别：设计产品决定了商业系统的信息，而表达则决定了如何表示这些信息。有些设计产品是用 UML 表示的，而有些是用文字或表格表示的。例如，一个类的生命周期可以用 UML 状态图、活动图表示，也可以用状态转换表或 Backus 范式表示。对象的交互可以用 UML 的序列图也可以用它的协作图表示。

商业系统的规格说明是定义在精确定义的设计产品基础上的，而不是在图上。下面我们就来讨论设计产品的结构，这个结构可以明确商业过程、商业规则、软件架构和商业系统设计之间的关系，并且很容易扩展到覆盖商业系统的其它方面，如团队结构和项目规划。

可以在不同的抽象级别上在不同的视图中描述商业系统，如机构层、系统层、架构层和类层，有逻辑视图、用例视图、分析视图、测试视图或使用文档视图。在每一个抽象级别和每个视图中，软件产品都可以用四种设计产品来描述：分类器之间的静态关系，分类器之间的动态交互，分类器的责任和分类器的生命周期。每个产品都可用 UM 的图或文字表示。如图 12-34 和 12-35。

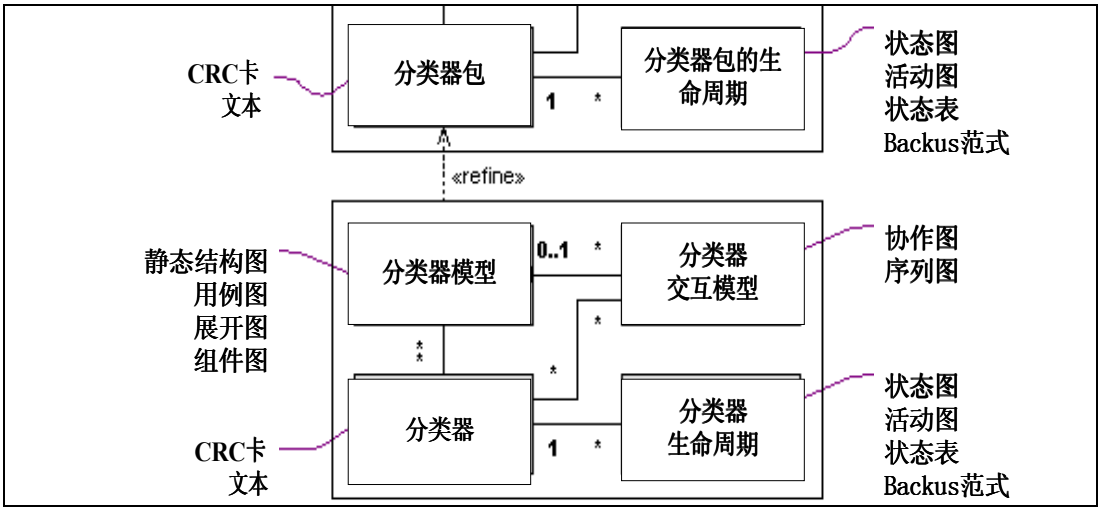

图 12-34　结构化项目知识库的设计产品模式

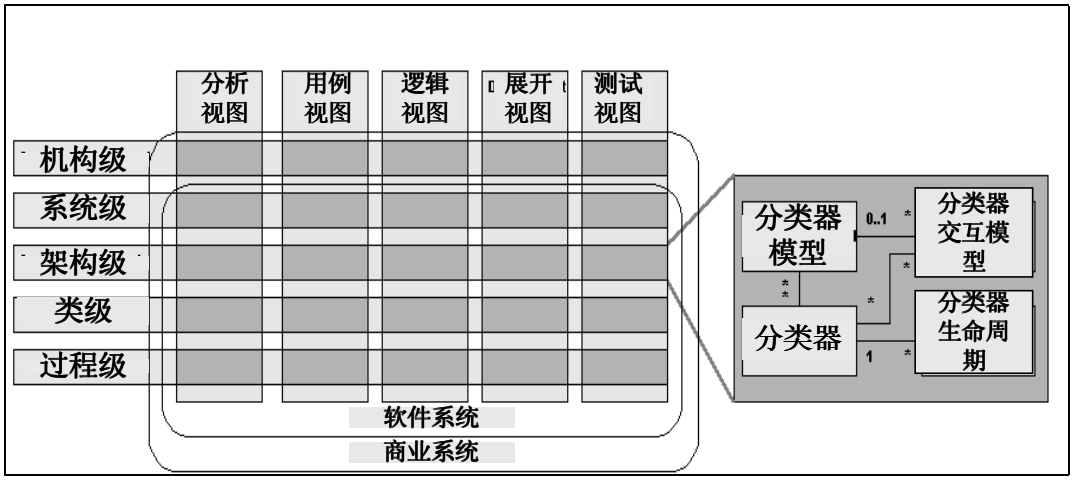

图 12-35　在每一抽象级中的每个视图中，都可以用四种设计产品来描述软件产品。
它们是 UML 分类器：类、接口、用例、节点、子系统和组件

分类器模型（classifier model）指定了分类器之间的静态关系。分类器模型可以是一组静态结构图（如果分类器是子系统、类或接口），可以是一组用例图（如果分类器是用例和角色），可以是一组展开图（如果分类器是结点），还可以是一组组件图（如果分类器是组件）。

分类器交互模型（classifier interaction model）指定了分类器之间的交互。分类器交互模型可以用交互图表示：序列图或协作图。

设计产品叫做分类器（classifier），指定了分类器的责任、角色和分类器接口的静态性质（例如，分类器的操作表，有前置条件和后置条件），分类器还可以用结构化的文字表示，如 CRC 卡。

分类器生命周期（classifier lifecycle）指定了分类器状态机和分类器接口的动态性质（例如所允许的有顺序的操作和事件）。分类器生命周期可以用状态图、活动图或一个状

态转换表表示。

可以把分类器模型的一个实例链接到分类器交互模型的几个实例上，所有这些实例都链接到分类器的实例上。分类器实例是链接到分类器生命周期的实例上。

12.13 项目知识库的结构化

在结构良好的设计中，有关商业过程和软件产品的信息都被很好地组织起来，并且相关信息链接在一起，可以很容易定位；同时结构还体现了不同设计产品的完整性和一致性；它是商业专家、顾问和软件开发人员沟通的基础。这一节讨论商业过程和软件设计产品之间的关系，用它把项目知识库结构化，然后讨论如何把商业过程的信息以及它们之间的关系结构化成软件设计产品。

图 12-36 指定了商业过程和软件系统逻辑设计之间关系的设计产品，它们是按照不同的抽象级别组织起来的：机构级，系统级，架构级和对象级。

机构级（organizational level）指定了一个机构（如公司、学校或政府实体）的责任，以及该机构所处的情景。产品“机构（organization）”指定了机构的责任以及相关的静态性质。产品“机构西式模型（organization model）”指定了机构与其它机构之间的关系。产品“机构用例（organization use case）”用过程目标、前置条件、后置条件、商业规则等过程必须符合的静态发生来指定机构范围的商业过程。这个商业过程是机构与其它机构之间的协作。这种协作是在“机构用例模型（organization use case model）”中指定的，见图 12-36 中的“协作”相关。机构商业过程的实例是由“机构交互模型（organization interaction model）”用机构与其它机构的交互来指定的。 机构商业过程可以精化到更具体的“系统商业过程（system business processes）”，见图 12-36 中的“精化”相关。“机构用例交互生命周期（organization use case life cycle）”指定了所允许的系统商业过程。 “机构用例交互模型（organization use case interaction model）”指定了典型的商业过程实例序列，见图 12-36 中的“实例”相关。机构商业过程的实现是用软件系统和它的用户（团队角色）之间的交互指定的，见图 12-36 和 12-37 中的“实现”相关。

系统级（system level）指定了软件系统的情景以及它与它的角色之间的关系。产品“系统（system）”用责任、前置条件、后置条件、参数和返回值来指定系统的接口和操作。产品“角色（actor）”指定了角色的责任和接口。“系统生命周期（system lifecycle）”指定了所允许的系统操作和事件。 “系统模型（system model）”指定了软件系统和角色（其它系统或用户）之间的关系，“系统交互模型（system interaction model）”指定了软件系统和角色之间的关系。这些交互是系统商业过程的实例，见图 12-36 中的“实例”相关。产品“系统用例（system use case）”用过程目标、前置条件、后置条件、过程非功能性需求、商业规则等相关静态性质指定了在系统范围内的商业过程。这个商业过程是系统与其它用户或系统的协作。系统与它的角色之间的协作是在产品“系统用例模型（system use case model）”中描述的，见图 12-36 中“协作”相关。商业过程接口的动态性质，如在商业过程范围内允许的系统操作顺序，是在“系统用例生命周期（system use case life cycle）”中指定的。“系统用例交互模型（system use case interaction model ）”指定了典型的商业过程

实例的序列。系统商业过程可以精化到子系统商业过程中，见图 12-36，“精化”相关系统商业过程的实现是用架构级的子系统间的交互指定的，见图 12-36 中的“实现”相关。

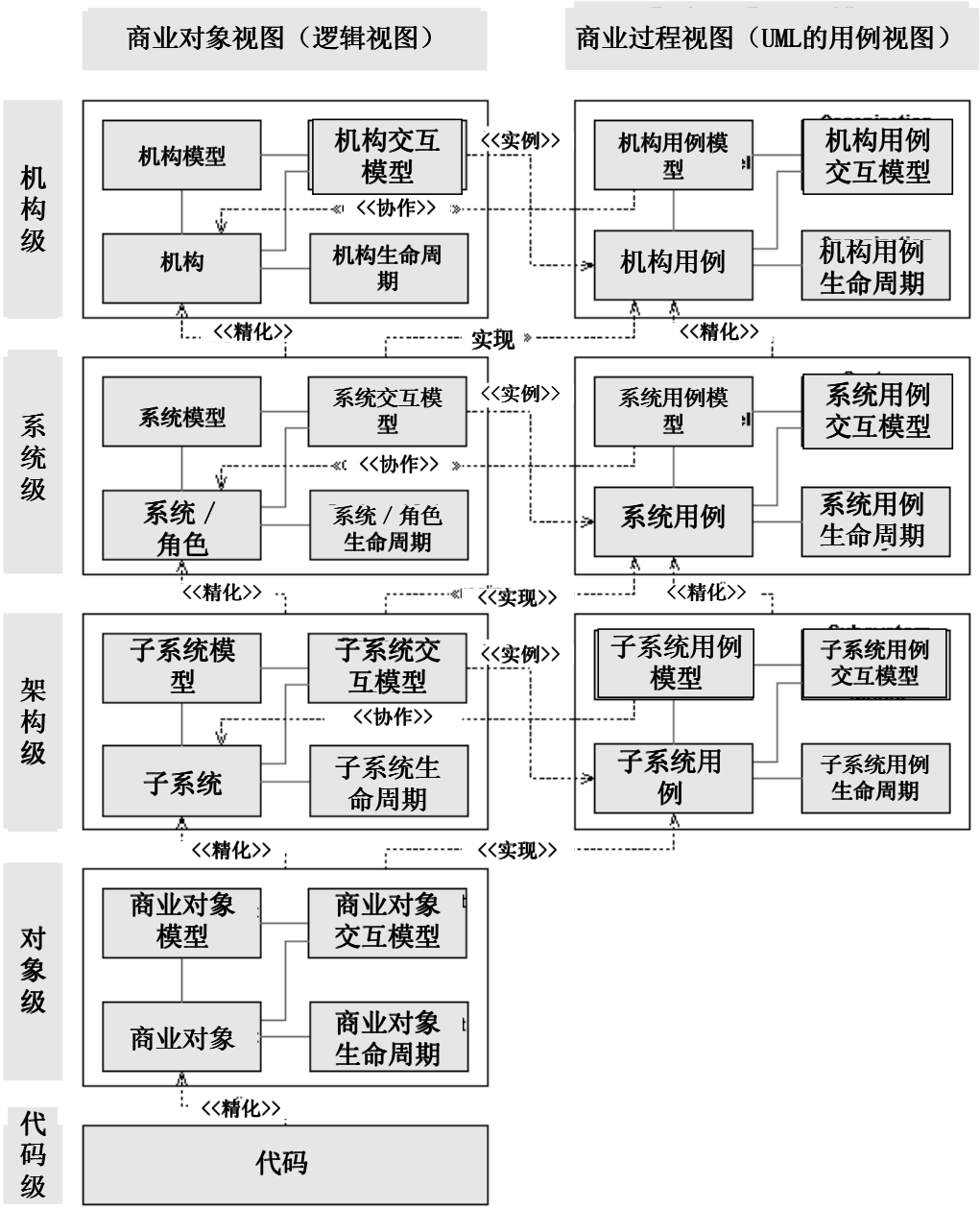

图 12-36　设计产品在逻辑和商业过程视图中描述了软件系统

架构级（architectural level）定义了子系统（组件）、子系统的责任、接口、关系和交互。产品子系统（subsystem）用责任、前置条件、后置条件、参数和返回值指定了子系统接口。子系统生命周期（subsystem lifecycle）指定了所允许的子系统的操作和事件。子系统模型（subsystem model）指定了子系统和其它子系统之间的关系，子系统交互模型（subsystem interaction model），指定了子系统之间的交互。这些交互是子系统商业过程的

实例，见图 12-36 的“实例”相关。产品子系统用例（subsystem use case）指定了在子系统内的商业过程。这个商业过程是子系统与其它子系统、系统和用户之间的协作。子系统和它的角色之间的所有协作是在子系统用例模型（subsystem use case model）中描述的，见图 12-36 中的“协作”相关。子系统商业过程接口的动态特性，如所允许的子系统操作顺序等，是在子系统用例生命周期（subsystem use case life cycle）中指定的子系统用例交互模型（subsystem use case interaction model）指定了商业过程实例的典型序列。子系统商业过程的实现是用类级上对象之间的交互描述的，见图 12-36 中的“实现”相关。

对象级（object level）用商业对象、商业对象的责任、关系和交互来描述子系统的设计。商业对象模型（business object model） 指定了商业对象之间的静态关系，商业对象交互模型（business object interaction model）用商业对象之间的交互指定了子系统操作的设计。商业对象（business object）指定了商业的责任和商业对象接口的静态性质，如用前置条件和后置条件描述的接口操作等。商业对象生命周期（business object lifecycle）指定了所允许的接口操作的顺序。

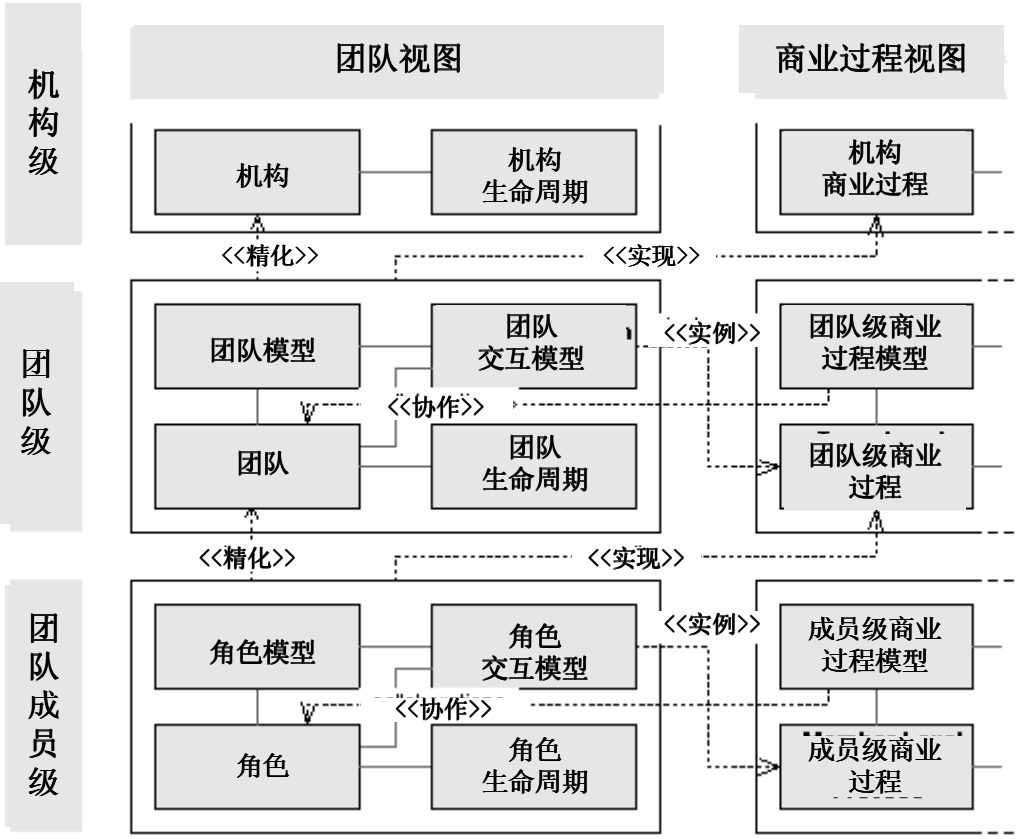

图 12-37　在团队和商业过程视图中，设计产品描述了软件系统

图 12-37 是用来描述机构团队结构的产品。事实上，它与描述软件子系统结构的产品没有什么太大区别。团队成员的角色可以表示成 UML 版类化的类。角色（role）指定了工人的角色及其相关的静态性质。 角色生命周期（role lifecycle）指定了角色的动态性质，它们的状态以及它们所响应的事件。角色模型（role model）指定了角色之间的静态关系。

成员级商业过程（member level business process）指定了团队成员角色与其它团队成员角色之间的协作，见图 12-37 中的“协作”相关。这些商业过程的实例是在角色交互模型（role interaction model）用角色实例之间的交互指定的，见图 12-37 中的“实例”相关。

叫做“团队（team）”的设计产品是一个角色包，团队了团队的责任以及相关的静态性质。团队的动态性质是在“团队生命周期（team life cycle）”中指定的，产品“团队模型（team model）”指定了团队之间的静态关系。 “团队级商业过程（team level business process）”指定了团队之间的协作，见图 12-37 中的相关“协作”。团队级商业过程是在团队交互模型（team interaction model）中指定的，见图 12-37 中“实例”相关。团队级商业过程的实现是用团队成员之间的交互以及运输队成员与软件系统之间的交互来指定的，如图 12-37 中的“实现”相关。设计产品的模式可以用相似的方法应用在更高的抽象级上。

设计产品的结构可以通过两种方式简化：

- 只使用其中的子集；
- 把紧密相关的一些设计产品结合起来，构成更大的设计产品单元。

12.14 小　　结

这一章讨论 UML 在商业建模中的应用，分析了在“从概念到代码”的整个过程中所使用的各种不同模型。在这个过程中，需要有以下几种模型：商业模型、系统设计模型和实现模型。其中，商业模型分为商业概念、商业动态、商业对象。其中，商业概念由目标模型、机构模型、位置模型、过程模型表达，商业动态由商业交互模型、工作流程模型、商业用例模型共同表达，商业对象由商业对象模型、子类型模型、状态模型共同表达。系统设计模型分为组件结构、组件动态、用户角度。其中组件结构由类模型、类层次图、关系表模型表达，组件动态由交互模型、对象消息模型、方法模型表达。用户角度由系统用户例图、用户交互模型、系统/子系统模型表达。实现模型包括组件模型和平台模型。

最后具体讨论了如何用 UML 表示商业系统，强调工作流程管理系统的 UML 表示。给出了如何把典型的商业概念映射到 UML 概念的方法，并给出了一种如何把设计产品结构化的方法，从而能够跟踪从商业过程到面向对象软件设计之间的关系。这种结构化方法的基础是一个四种相互有关的设计产品的模式，它们代表了分类器的关系户交互责任和生命周期。该模型可以在一个商业系统中的不同抽象级的不同视图中加以应用。