CHAPTER 5

第 5 章

面向对象分析

【案例 5-1】 银行网络 ATM 系统的需求陈述、静态模型、动态模型和功能模型。

【案例 5-2】 饮料自动售货机系统的面向对象的分析实例。

5.1 面向对象分析概述

面向对象分析(Object-Oriented Analysis,OOA)就是运用面向对象的方法进行需求分析,其主要任务是分析和理解问题域,找出相应的描述问题域和系统责任所需的类及对象,分析它们的内部构成和外部关系,正确地抽象为规范的对象,定义其内部结构和外部消息传递关系,建立问题域精确模型的过程。为后续的面向对象设计和面向对象编程提供指导。

面向对象分析的目的是对客观世界的系统进行建模,是定义所有与待解决问题相关的类(包括类的操作和属性、类与类之间的关系以及它们表现出的行为),完成对所求解问题的分析,确定系统“做什么”,并建立系统的模型。

面向对象分析的基本任务是运用面向对象的方法,软件工程师和用户必须充分沟通,以了解基本的用户需求,对问题域和系统责任进行分析和理解,找出描述它们的类和对象,定义其属性和操作,及其层次结构、静态联系和动态联系,模型化对象的行为,直到模型建成。

面向对象分析的特点是有利于对问题及系统责任的理解,人员之间的交流,并对需求变化有较强的适应性,并支持软件复用。

为建立分析模型,要运用如下 5 个基本原则。

① 建立信息域模型。

② 描述功能。

③ 表达行为。

④ 划分功能、数据、行为模型,揭示更多的细节。

⑤ 用早期的模型描述问题的实质,用后期的模型给出实现的细节。

这些原则形成面向对象分析的基础。

5.1.1 面向对象分析模型

完整的面向对象分析模型分为基本模型和补充模型以及详细说明,

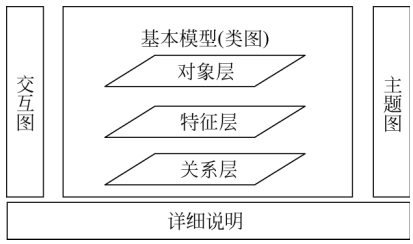

图 5.1 面向对象分析模型

如图 5.1 所示。

(1) 基本模型

基本模型是一个类图(Class Diagram),是以直观的方式表达系统最重要的信息。面向对象分析基本模型的三个层次分别描述了系统中应设哪几类对象、每类对象的内部构成和对象与外部的关系。

构成类图的元素所表达的模型信息,分为以下三个层次。

① 对象层。

对象层给出系统中所有反映问题域和系统责任的对象。

② 特征层。

特征层给出类(对象)的内部特征,即类的属性和操作。

③ 关系层。

关系层给出各类(对象)之间的关系,包括继承、封装、一般-特殊、整体-部分、属性的静态依赖关系,操作的动态依赖关系。

(2) 补充模型

补充模型有主题图和交互图。

① 主题。

主题(Subject)又称为子系统(Subsystem),是将一些联系密切的类组织在一起的类的集合。按照粒度控制原则,将系统组成几个主题,便于理解。主题图画出了系统的主题。

② 交互图。

交互图(Interaction Diagram)是 Use Case 与系统成分之间的对照图。

(3) 详细说明

按照分析方法所要求的格式,对分析模型进行说明和解释,主要以文字为主。

5.1.2 面向对象分析过程

(1) 获取客户对系统的需求

需求获取必须让客户与开发者充分地交流,采用用例来收集客户需求的技术:分析员先标识使用该系统的不同的执行者(Actor),代表使用该系统的不同的角色。每个执行者可以叙述他如何使用系统,或者说他需要系统提供什么功能。执行者提出的每一个使用场景(或功能)都是系统的一个用例的实例,一个用例描述了系统的一种用法(或一个功能),所有执行者提出的所有用例构成系统的完整的需求,其中涉及对需求的分析及查找丢失的信息。

注意,执行者与用户是不同的两个概念,一个用户可以扮演几个角色(执行者),一个执行者可以是用户,也可以是其他系统(应用程序或设备)。得到的用例必须进行复审,以使需求完整。

(2) 标识类和对象

描述如何发现类及对象。类和对象是在问题域中客观存在的,从应用领域开始识别,可以先标识候选的类和对象,然后从候选的类和对象中筛选掉不正确的或不必要的。类及对象是形成整个应用的基础,据此分析系统的责任。

(3) 定义类的结构和层次

该阶段分为以下两个步骤。

第一,识别一般-特殊结构,该结构捕获了识别出的类的层次结构。

第二，识别整体-部分结构，该结构用来表示一个对象如何成为另一个对象的一部分，以及多个对象如何组装成更大的对象。

有的面向对象方法中，把互相协作以完成一组紧密结合在一起的责任的类的集合定义为主题(Object)或子系统(Subsystem)。主题由一组类及对象组成，用于将类及对象模型划分为更大的单位，便于理解。

主题和子系统都是一种抽象，从外界观察系统时，主题或子系统可看作黑盒，它有自己的一组责任和协作者，观察者不必关心其细节。观察一个主题或子系统的内部时，观察者可以把注意力集中在系统的某一个方面。因此，主题或子系统实际上是系统更高抽象层次上的一种描述。

(4) 建造对象-关系模型

对象-关系模型描述了系统的静态结构，它指出了类间的关系。类之间的关系有关联、依赖、泛化、实现等。两个或多个对象之间的相互依赖、相互作用的关系就是关联。分析确定关联，能促使分析员考虑问题域的边缘情况，有助于发现那些尚未被发现的类和对象。

(5) 建立对象-行为模型

对象-行为模型描述了系统的动态行为，它们指明系统如何响应外部的事件或激励。建模的步骤如下。

① 评估所有的用例，完全理解系统中交互的序列。

② 标识驱动交互序列的事件，理解这些事件如何和特定的对象相关联。

③ 为每个用例创建事件轨迹(Event Trace)。

④ 为系统建造状态机图。

⑤ 复审对象-行为模型，以验证准确性和一致性。

综上所述，在概念上可以认为，面向对象分析大体上按照下列顺序进行：寻找类和对象，确定关联，确定属性，识别结构，识别主题，定义属性，定义服务，建立动态模型和建立功能模型。但是，分析不可能严格地按照预定顺序进行，大型的、复杂的问题需要反复多次进行寻找、确定、识别、定义和建立来构造模型。通常，先构造出模型的雏形或部分，再逐步扩充、修改、求精直至满意为止，最后，构造出符合问题域需求的正确的、准确的、完整的目标系统的模型，并编写出需求规格说明，如图 5.2 所示。

分析也不是一个机械的过程。大多数需求陈述都缺乏必要的信息，所缺少的信息主要从用户和领域专家那里获取，同时也需要从分析员对问题域的背景知识中提取。在分析过程中，系统分析员必须与领域专家及用户反复交流，以便弄清二义性，改正错误的概念，补足缺少的信息。面向对象建立的系统模型，尽管在最终完成之前还是不准确、不完整的，但对做到准确、无歧义的交流仍然是大有益处的。

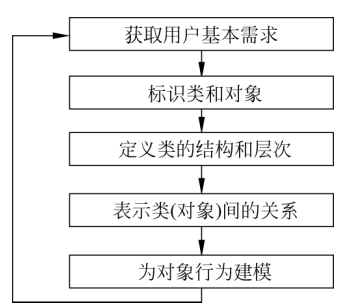

图 5.2　面向对象分析的过程

基于前面介绍的基础知识，本书主要结合案例 5-1：银行网络 ATM(Auto Trade Machine)系统的具体实例，提出需求陈述，建立静态模型及动态模型，并引用该实例讨论面向对象分析和面向对象设计。

5.2　需求陈述

需求陈述也叫问题陈述。陈述需求是开发任何一个系统的首要任务。主要陈述用户的需求，即该系统应该“做什么”，而不是“怎么做”(系统如何实现)。应该陈述系统任务是什么，而

不是解决问题的方法。应该指出哪些是系统必要的性质，哪些是任选的性质。应该避免对设计策略施加过多的约束，也不要描述系统的内部结构，因为这样做将限制实现的灵活性。对系统性能及系统与外界环境交互协议的描述，是合适的需求。此外，对采用的软件工程标准、模块构造准则、将来可能做的扩充以及可维护性要求等方面的描述，也都是适当的需求。

需求陈述必须要将解决问题的目标清楚地表达出来，如果目标模糊，将会影响系统分析、设计和实现等后续开发阶段的工作。分析人员和用户一起研究和讨论才能准确表达用户的要求，并找出遗漏的信息。

需求陈述是软件系统生存期中定义阶段的最后一个步骤，是作为整个软件开发范围的指南，是软件开发人员开发出正确的符合用户要求的软件的重点。

需求陈述过程中需要解决如下问题。

① 问题域(Problem Domain) ——被开发系统的应用领域。

② 系统责任(System Responsibilities)——所开发的系统应具备的职能。

③ 充分的交流——获得准确分析结果的关键。

④ 需求的不断变化——应变能力的强弱是衡量一种方法优劣的重要标准。

⑤ 考虑复用要求——提高开发效率，改善软件质量的重要途径。

(1) 系统分析员对需求陈述进行分析

需求陈述通常是不完整、不准确的，也可能还是非正式的。通过分析可以发现和改正需求陈述中的歧义性、不一致性，剔除冗余的内容，挖掘潜在的内容，弥补不足，从而使需求陈述更完整、更准确。

在分析需求的过程中，系统分析员不仅应该反复、多次地与用户讨论、交流信息，还应该调研、观察、了解现有的类似系统。

快速地建立一个原型系统，通过在计算机上运行原型系统，使得分析员和用户尽快交流和相互理解，从而能更正确地、更完整地提取和确定用户的需求。

(2) 需求建模

系统分析员根据提取的用户需求，深入理解用户需求，识别出问题域内的对象，并分析它们相互之间的关系，抽象出目标系统应该完成的需求任务，并用 OOA 模型准确地表示出来，即用面向对象观点建立对象模型、动态模型和功能模型。

面向对象分析模型是面向对象设计的基础，它应该准确地、简洁地表示问题。通过建立模型，可避免理解上的片面性，提高目标系统的正确性、可靠性。在此基础上，编写出面向对象的需求规格说明。

(3) 需求评审

通过用户、领域专家、系统分析员和系统设计人员的评审，并进行反复修改后，确定需求规格说明。

需求分析是复杂而又艰辛的过程。系统分析人员应该多和用户交流，认真和领域专家探讨。必须在领域专家的指导和密切配合下进行，才能有效地完成任务。

需求分析的工作将最终交给软件设计和开发人员进行具体的软件设计和开发，其针对的对象是软件设计和开发人员。

案例 5-1：银行网络 ATM 系统的需求陈述。

① 问题综述。

某银行拟开发一个自动取款机系统，它是一个由自动取款机、中央计算机、分行计算机及营业终端组成的网络系统，如图 5.3 所示。ATM 和中央计算机由总行投资购买。总行拥有

多台 ATM,分别设在全市各主要街道上。分行负责提供分行计算机和营业终端。营业终端设在分行下属的各个储蓄所内。该系统的软件开发成本由各个分行共同承担。

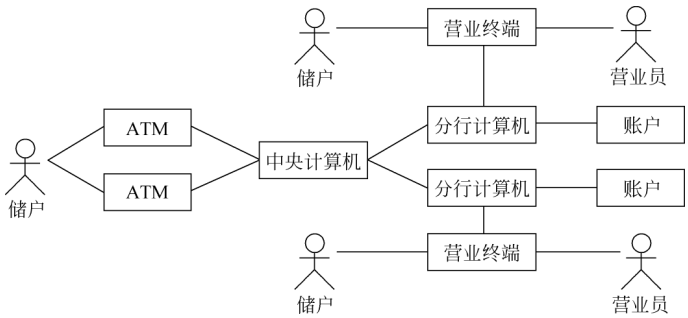

图 5.3 银行网络 ATM 系统的示意图

② 实施陈述。

银行营业员使用营业终端处理储户提交的储蓄事务。储户可以用现金或支票向自己拥有的某个账户内存款或开新账户,储户也可以从自己的账户中取款。通常,一个储户可能拥有多个账户。营业员负责把储户提交的存款或取款事务输进营业终端,接收储户交来的现金或支票,或者付给储户现金。营业终端与相应的分行计算机通信,分行计算机具体处理针对某个账户的事务并且维护账户。

拥有银行账户的储户有权申请领取现金兑换卡。使用兑换卡可以通过 ATM 访问自己的账户。目前,仅限于用现金兑换卡在 ATM 上提取现金(即取款),或查询有关自己账户的信息(例如,某个指定账户上的余额)。将来可能还要求使用 ATM 办理转账、存款等事务。

所谓现金兑换卡,就是一张特制的磁卡,上面有分行代码和卡号。分行代码唯一标识总行下属的一个分行,卡号确定了这张卡可以访问哪些账户。通常,一张卡可以访问储户的若干个账户,但是不一定能访问这个储户的全部账户。每张现金兑换卡仅属于一个储户所有,但是,同一张卡可能有多个副本,因此,必须考虑同时在若干台 ATM 上使用同样的现金兑换卡的可能性。也就是说,系统应该能够处理并发的访问。

当用户把现金兑换卡插入 ATM 之后,ATM 就与用户交互,以获取有关这次事务的信息,并与中央计算机交换关于事务的信息。首先,ATM 要求用户输入密码,接下来,ATM 把从这张卡上读到的信息以及用户输入的密码传给中央计算机,请求中央计算机核对这些信息并处理这次事务。中央计算机根据卡上的分行代码确定这次事务与分行的对应关系,并且委托相应的分行计算机验证用户密码。如果用户输入的密码是正确的,ATM 就要求用户选择事务类型(取款、查询等)。当用户选择取款时,ATM 请求用户输入取款额。最后,ATM 从现金出口输出现金,并且打印出账单交给用户。

5.3 建立静态模型

所谓静态建模是指对象之间通过属性互相联系,而这些关系不随时间而转移,即建立对象模型。

建立对象模型时,我们的目标是从问题域中提炼出对目标系统有价值的概念。对象模型描述了问题域中的类和对象以及它们之间的关系,表示了目标系统的静态数据结构。一般来说,当用户的需求变化时,静态数据结构相对来说比较稳定。静态数据结构较少地依赖应用细节,因此比较容易确定。

系统中的类和对象模型描述了系统的静态结构，在 UML 中用类图和对象图来表示。类图由系统中使用的类以及它们之间的关系组成。类之间的关系有关联、依赖、泛化和实现等。类图是一种静态模型，它是其他图的基础。一个系统可以有多张类图，一个类也可出现在几张类图中。对象图是类图的一个实例，它描述某一时刻类图中类的特定实例以及这些实例之间的特定链接。

对象模型表示了静态的、结构化的系统数据性质，描述了系统的静态结构，它是从客观世界实体的对象关系角度来描述，表现了对象的相互关系。该模型主要关心系统中对象的结构、属性和操作，它是分析阶段三个模型的核心，是其他两个模型的框架。

(1) 模板

模板是类、关联、一般化结构的逻辑组成。

(2) 对象模型

对象模型是由一个或若干个模板组成。模板将模型分为若干个便于管理的子块，在整个对象模型和类及关联的构造块之间，模板提供了一种集成的中间单元，模板中的类名及关联名是唯一的。

面向对象分析是复杂而又艰辛的过程，是系统分析员经反复迭代逐步深化的认知过程来创建模型，将初始的分析模型变为最终的分析模型。也可以研究、借鉴以前对相同的或类似的问题域进行面向对象分析后所得到的结果，这样既可以实现复用，又可以提高系统分析效率。

本节结合案例 5-1：银行网络 ATM 系统的实例将进行面向对象的分析，进而建立静态模型。

5.3.1 寻找类与对象

构造对象模型的第一步是标出来自问题域的相关的对象类，对象包括物理实体和概念。所有类在应用中都必须有意义，在问题陈述中，并非所有类都是明显给出的。有些是隐含在问题域或一般知识中的。

类-责任-协作者(Class-Responsibility-Collaborator，CRC)技术用于完成类的定义。CRC 是一组表示类的索引卡片，每张卡片分成三部分，它们分别描述类名、类的责任和类的协作者，如表 5.1 所示。责任是用来描述类的属性和操作，协作者描述为完成该责任而提供信息的其他相关的类。

表 5.1 CRC 卡

类名：	
责任：	协作者：

确定和标识类包括发现潜在对象、筛选对象、为对象分类，最后将同类型的对象抽象为类。

(1) 标识潜在的对象类

通常，在需求陈述中不会一个不漏地写出问题域中所有有关的类和对象，陈述中的名词或名词短语是可能的潜在对象，它们以不同的形式展示出来，因此，分析员应该根据领域知识或常识进一步把隐含的类和对象提取出来。

对象是人们要研究的任何事物及对问题域中有意义的事物的抽象，它们既可能是物理实体，也可能是抽象概念(规则、计划和事件)。具体地说，对象可分为以下几种类型。

① 物理实体。

指有形的实物，例如，飞机、汽车、计算机、书或机房等。

② 与系统交互的人或组织的角色。

例如，医生、教师、学生、工人、部门或公司等。

③ 与系统发生交互的人及系统必须保留其信息的人，可作为候选的类及对象。

例如：柜员、储户等。

④ 这些人所属的组织单位，可作为候选的类及对象。

例如：总行、分行等。

⑤ 事件。

指在特定时间所发生的事，例如，飞行、演出、开会、访问或事故等。

⑥ 系统必须观测、记忆的与时间有关的事件可作为候选的类及对象。

例如：建立账户的日期，打开一个账户等。

⑦ 系统的工作环境场所。

例如：车间、办公室。

⑧ 系统需了解掌握的物理位置、办公地点等可作为候选的类及对象。

例如：ATM 机器、账户等。

⑨ 性能说明。

指厂商对产品性能的说明，如产品名字、型号、规格和各种性能指标等。

⑩ 与系统有关的外部实体。

其他系统、设备、人员等生产或消费计算机系统所使用的信息，例如：打印机等。分析阶段可不把与实现有关的计算机部件作为候选的类及对象。

⑪ 系统必须记忆、且不在问题域约束中的顺序操作过程(为了指导人机交互)，可作为候选的类及对象。

例如：柜员事务、远程事务等。其中属性是操作过程名，操作特权及操作步骤的描述。

在面向对象分析时，可以参照上述几类常见事物，找出在当前问题域中潜在的类和对象。另外，还可以以自然语言书写的需求文档(陈述)为依据，这种分析方法比较简单，是一种非正式分析。文档中的名词可作为潜在(候选)的类和对象，形容词可作为线索来确定属性，动词可作为潜在的服务(操作)。这个结果可作为更详细、更精确的正式的面向对象分析的雏形，当然，也是正式的面向对象分析的一个良好的开端。

下面以案例 5-1：银行网络 ATM 系统的实例，说明非正式分析过程。从陈述中找出下列名词作为类-对象的初步的候选者：银行、自动取款机(ATM)、系统、中央计算机、分行计算机、营业终端、网络、总行、分行、软件、成本、市、街道、营业厅、储蓄所。营业员、储户、现金、支票、账户、事务、现金兑换卡、余额、磁卡、分行代码、卡号、用户、副本、信息、密码、类型、取款额、账单以及访问等。

通常，在需求陈述中不会一个不漏地写出问题域中所有有关的类-对象，因此，分析员应该根据领域知识或常识进一步把隐含的类-对象提取出来。

在案例 5-1：银行网络 ATM 系统中，系统的需求陈述虽然没写“通信链路”和“事务日志”，但是根据领域知识和常识可以知道，应该包含这两个实体。

(2) 筛选对象类，确定最终对象类

显然，仅通过一个简单、机械的过程不可能正确地完成分析工作。通过非正式分析仅帮助分析员找到一些候选的类和对象，接下来应该严格考察、筛选每个候选对象，从中去掉不正确的或不必要的，仅保留确实应该记录其信息或需要其提供服务的那些对象。

假定已经找到了一个候选对象，这时又发现了另一个可能成为对象的实体，那么，是否应该将它作为对象放到模型中去呢？在现实世界中，存在着许多对象，但仅可讨论而已，不能全部纳入系统中去。这些对象才确实是应该记录其信息或需要其提供服务的对象。

① 筛选时主要依据下列标准，删除不正确或不必要的类和对象。

• 关键性。

缺少这个对象信息，系统就不能分析并确定问题域中的对象。

• 可操作性。

潜在对象必须拥有一组可标识的操作，它们可以按某种方式修改对象属性的值。

• 信息含量。

选择信息量较大的对象确定为最终对象，只有一个属性的对象可以与其他同类对象合并为一个对象。

• 公共属性。

可以为潜在的对象定义一组属性，这些属性适用于该类的所有实例。

• 公共操作。

可以为潜在的对象定义一组操作，这些操作适用于该类的所有实例。

• 关键外部信息。

问题空间中的外部实体和系统必须生产或消费信息。

② 类和对象还可以按以下特征进行分类。

• 确切性。

类表示了确切的事物(如键盘或传感器)，还是表示了抽象的信息(如预期的输出)。

• 包含性。

类是原子的(即不包含任何其他类)，还是聚合的(至少包含一个嵌套对象)。

• 顺序性。

类是并发的(即拥有自己的控制线程)，还是顺序的(被外部的资源控制)。

• 完整性。

类是易被侵害的(即它不防卫其资源受外界的影响)，还是受保护的(该类强制控制对其资源的访问)。

• 持久性。

类是短暂的(即它在程序运行期间被创建和删除)、临时的(它在程序运行期间被创建，程序终止时被删除)，还是永久的(它存放在数据库中)。

基于上述分类，CRC 卡的内容可以扩充，以包含类的类型和特征，如表 5.2 所示。

表 5.2 扩充后的 CRC 卡

类名：	
类的类型：(如：设备，角色，场所……)	
类的特征：(如：确切的，原子的，并发的……)	
责任：	协作者：

在填好所有 CRC 卡后，应对它进行复审。

③ 复审应由客户和软件分析员参加，复审方法如下。

• 参加复审的人，每人拿 CRC 卡片的一个子集。注意，有协作关系的卡片要分开，即没有一个人持有两张有协作关系的卡片。

• 将所有用例/场景分类。

• 复审负责人仔细阅读用例，当读到一个命名的对象时，将令牌(Token)传送给持有对应类的卡片的人员。

• 收到令牌的类卡片持有者要描述卡片上记录的责任，复审小组将确定该类的一个或多个责任是否满足用例的需求。当某个责任需要协作时，将令牌传给协作者，并重复此步骤。

- 如果卡片上的责任和协作不能适应用例，则需对卡片进行修改，这可能导致定义新的类，或在现有的卡片上刻画新的或修正的责任及协作者。

这种做法持续至所有的用例都完成为止。

④ 审查和筛选候选项，根据下列标准，去掉不必要的类和不正确的类。

- 冗余类。

若两个类表达了同样的信息，则应该保留在问题域中最富于描述力的那个名称，去掉冗余的类。

上面初步分析得出了银行网络 ATM 系统的 34 个候选的类，其中，储户与用户，现金兑换卡与磁卡及副本分别描述了相同的信息，因此，应该将“用户”、“磁卡”、“副本”等冗余的类去掉，仅保留“储户”和“现金兑换卡”这两个类。

- 不相干的类。

删除那些与问题没有多少关系或根本无关的类，仅把与问题密切相关的类-对象放进目标系统中。有些类在其他问题中可能很重要，但与当前要解决的问题无关，同样也应该把它们删掉。

在案例 5-1：银行网络 ATM 系统中，应该去掉“成本”、“市”、“街道”、“营业厅”和“储蓄所”等候选类。

因为，该系统并不处理软件开发成本的问题，而且 ATM 和营业员终端放置的地点与本软件的关系也不大。

- 模糊类。

在初步分析时，列出来的作为候选的类和对象中，可能有一些模糊的、泛指的名词，其中，有的是系统无须记忆的信息；有的是在需求陈述中，它们所暗示的事务，有更明确更具体的名词来表示。因此，通常应去掉这些笼统的或模糊的类。类必须是确定的，有些暂定类边界定义模糊或范围太广，如“系统”、“安全措施”等就属于模糊类。

在案例 5-1：银行网络 ATM 系统中，“银行”实际指总行或分行，“访问”在这里实际指事务，“信息”的具体内容在需求陈述中随后就指明了。此外，还有一些笼统含糊的名词。因此，在本例中，应该去掉“银行”、“网络”、“系统”、“软件”、“信息”和“访问”等候选类。

- 属性准则。

对象是用属性来描述的，若有些名词只是其他对象的属性的描述，则应该把这些名词从候选类和对象中去掉。当然，如果某个性质具有很强的独立性，则应把它作为类而不是作为属性。如“教师信息”、“学生信息”、“课程信息”等就属于属性。

在案例 5-1：银行网络 ATM 系统中，“现金”、“支票”、“取款额”、“账单”、“余额”、“分行代码”、“卡号”、“密码”和“类型”等，都应该作为属性而不是作为类。

- 操作准则。

在需求陈述中，有时可能使用一些既可作为名词又可作为动词的词，此时，应根据它们在本问题中的含义来决定它们是作为类还是作为类中定义的操作。

例如，通常把电话“拨号”当作动词，当构造电话模型时，确实应该把它作为一个操作，而不是一个类。但是，在开发电话的自动记账系统时，把“拨号”作为重要的一个类，因为它有自己的日期、时间受话地点等属性。

总之，当一个操作具有属性需独立存在时，应该作为类-对象而不是作为类的操作。

- 实现准则。

在分析阶段，应该去掉仅和实现有关的候选的类和对象。

在案例5-1：银行网络ATM系统中，“事务日志”无非是对一系列事务的记录，表示的是面向对象设计的议题；“通信链路”在逻辑上是一种联系，在系统实现时它是关联链的物理实现。因此，应该暂时去掉“事务日志”和“通信链路”这两个类，在设计或实现时再考虑它们。

综上所述，在案例5-1：银行网络ATM系统中，经过初步筛选，剩下了下列类-对象：ATM、中央计算机、分行计算机、营业终端、总行、分行、营业员、储户、账户、事务和现金兑换卡等。

5.3.2 确定关联

一个类可以用它自己的操作去操纵它自己的属性，从而完成某一特定的责任，一个类也可和其他类协作来完成某个责任。如果一个对象为了完成某个责任需要向其他对象发送消息，则说该对象和另一对象协作，协作实际上标识了类间的关系。

两个或多个对象之间的相互依赖、相互作用的关系就是关联。一种依赖表示一种关联，可用各种方式来实现关联，但在分析模型中应删除实现的考虑，以便设计时更为灵活。关联常用描述性动词或动词词组来表示，其中有物理位置的表示、传导的动作、通信、所有者关系、条件的满足等。从问题陈述中抽取所有可能的关联表述，把它们记下来，但不要过早去细化这些表述。分析确定关联，能促使分析员考虑问题域的边缘情况，有助于发现那些尚未被发现的类和对象。

一般情况下，在初步分析问题域中的类-对象确定之后，接着就可以分析、确定类-对象之间存在的关联关系。由于在整个开发过程中，从面向对象的分析到面向对象的设计，面向对象概念和表示符号都是一致的，因此，分析员可以不按照这样的工作顺序，灵活地选取自己习惯的工作方式。

(1) 初步确定关联

通常，在需求陈述中使用的描述性动词或动词词组，通常表示关联关系。因此，在初步确定关联时，大多数关联可以通过直接提取需求陈述中的动词词组而得出。通过分析需求陈述，还能发现一些在陈述中隐含的关联。最后，根据问题域实体间的相互依赖、相互作用关系，分析员还应该与用户及领域专家讨论，根据领域知识还能挖掘一些在陈述中隐含的关联，再进一步补充一些关联。

以案例5-1：银行网络ATM系统为例，用直接提取动词短语得出关联、需求陈述中隐含关联和根据问题域知识得出关联等方法，经过分析，初步确定出如表5.3所示的关联。在表5.3中，标“☆”符号的关联表示是删掉的关联，标“★”符号的关联表示是分解后又删掉的关联。

(2) 筛选关联

经初步分析得出的关联只能作为候选的关联，还需经过进一步筛选，以去掉不正确的或不必要的关联。筛选时主要根据下述标准删除候选的关联。

① 删除已去掉的类之间的关联

在分析、确定类和对象的过程中，如果已经删掉了某个候选类，则与这个类有关的关联也应该删掉，或用其他类重新表达这个关联。

在案例5-1：银行网络ATM系统中，由于已经删掉了“系统”、“网络”、“市”、“街道”、“成本”、“软件”、“事务日志”、“现金”、“营业厅”、“储蓄所”和“账单”等候选类。因此，与这些类有关的8个(表5.3中的(1)，(3)，(5)，(6)，(16)，(17)，(22)，(23))关联也应该删掉。

表 5.3 确定银行网络 ATM 系统的关联

确定方法	关联
直接提取动词短语得出关联	(1) ATM、中央计算机、分行计算机及营业员终端组成网络（☆） (2) 总行拥有多台 ATM（★） (3) ATM 设在主要街道上（☆） (4) 分行提供分行计算机和营业终端（★） (5) 营业终端设在分行营业厅及储蓄所内(☆) (6) 分行分摊软件开发成本(☆) (7) 储户拥有账户 (8) 分行计算机处理针对账户的事务（★） (9) 分行计算机维护账户(★) (10) 营业终端与分行计算机通信 (11) 营业员输入针对账户的事务(★) (12) ATM 与中央计算机交换关于事务的信息(★) (13) 中央计算机确定事务与分行的对应关系（★） (14) ATM 读现金兑换卡（☆） (15) ATM 与用户交互（☆） (16) ATM 输出现金（☆） (17) ATM 打印账单（☆） (18) 系统处理并发的访问（☆）
需求陈述中隐含关联	(19) 总行由各个分行组成 (20) 分行保管账户 (21) 总行拥有中央计算机 (22) 系统维护事务日志（☆） (23) 系统提供必要的安全性（☆） (24) 储户拥有现金兑换卡
根据问题域知识得出关联	(25) 现金兑换卡访问账户 (26) 分行聘用营业员
进一步完善关联	(27) 中央计算机与分行通信(中央计算机确定事务与分行的对应关系改名) (28) 营业员输入事务（营业员输入针对账户的事务的分解） (29) 事务修改账户（营业员输入针对账户的事务的分解） (30) 分行计算机处理事务（分行计算机处理针对账户的事务的分解） (31) 事务处理账户（分行计算机处理针对账户的事务的分解） (32) ATM 与中央计算机通信（ATM 与中央计算机交换关于事务的信息） (33) 在 ATM 上输入事务（ATM 与中央计算机交换关于事务的信息） (34) 总行拥有中央计算机（总行拥有多台 ATM 的分解） (35) 分行保管账户（分行计算机维护账户） (36) 事务修改账户（分行计算机维护账户） (37) 分行拥有分行计算机(分行提供分行计算机和营业终端的分解） (38) 分行拥有营业终端（分行提供分行计算机和营业终端的分解） (39) 营业员输入营业事务(增补) (40) 营业事务输进营业终端(增补) (41) 在 ATM 上输入远程事务(增补) (42) 远程事务由现金兑换卡授权(增补)

② 删除不相干的关联或实现阶段的关联

在候选类中，应该把与本问题域无关的关联或与实现密切相关的关联删去。

在案例 5-1：银行网络 ATM 系统中，“系统处理并发的访问”只提醒我们在实现阶段需要使用实现并发访问的算法，以处理并发事务，并没有标明对象之间的新关联（表 1 中序号为(18)的关联），因此，应删去它。

③ 删除瞬时动作

关联应该描述问题域的静态结构性质，而不应该是一个瞬时事件，因此应删除瞬时事件的关联。

在案例 5-1：银行网络 ATM 系统中，“ATM 读现金兑换卡”描述了 ATM 与用户交互周期中的一个动作，它并不是 ATM 与现金兑换卡之间的固有关系，因此应该删去。类似地，还应该删去“ATM 与用户交互”这个候选的关联。（表 5.3 中序号为(14)和(15)的关联。）

如果用动作表述的需求隐含了问题域的某种基本结构，则应该用适当的动词词组重新表示这个关联。

在案例 5-1：银行网络 ATM 系统的需求陈述中，“中央计算机确定事务与分行的对应关系”隐含了结构上“中央计算机与分行通信”的关系（表 5.3 中序号为(13)的关联），因此，应该删去。

④ 分解多元关联

多元关联是三个或三个以上对象之间的关联，多数可以分解为二元关联或用词组描述成限定的关联。

在案例 5-1：银行网络 ATM 系统中，“营业员输入针对账户的事务”可以分解成“营业员输入事务”和“事务修改账户”两个二元关联，而“分行计算机处理针对账户的事务”也可以做类似的分解（表 5.3 中序号为(11)和(8)的关联）。“ATM 与中央计算机交换关于事务的信息”这个候选的关联，实际上隐含了“ATM 与中央计算机通信”和“在 ATM 上输入事务”这两个二元关联（表 5.3 中序号为(12)的关联）。

⑤ 派生关联

有的关联可以用已有的、必要的关联来定义时，应该去掉这些冗余的关联。

在案例 5-1：银行网络 ATM 系统中，“总行拥有多台 ATM”派生了“总行拥有中央计算机”和“ATM 与中央计算机通信”这两个关联。“分行计算机维护账户”的实际含义是，“分行保管账户”和“事务修改账户”（表 5.3 中序号为(2)和(9)的关联）。

(3) 完善关联

经过筛选后余下的关联不够精确、完善时，应该进一步分解和增补，以调整关联。通常从下述几个方面进行改进。

① 重命名

关联的命名相当重要，准确的名字有利于读者理解。因此，如有不合适的、含义不清的名字，应该重新选择含义更明确的名字作为关联名。

在案例 5-1：银行网络 ATM 系统中，将“分行提供分行计算机和营业员终端”改为“分行拥有分行计算机”和“分行拥有营业员终端”就更明确些（在表 5.3 中序号为(4)的关联）。

② 分解

为了能够适用于不同的关联，必要时应该分解以前确定的类-对象。

在案例 5-1：银行网络 ATM 系统中，应该把“事务”分解成“远程事务”和“营业事务”。

③ 补充

发现了遗漏的关联或分解类和对象之后需要新关联时，应该及时增补。

在案例 5-1：银行网络 ATM 系统中，把“事务”分解成上述两类之后，需要补充“营业员输入营业事务”、“营业员事务输进营业终端”、“在 ATM 上输入远程事务”和“远程事务由现金兑换卡授权”等关联（表 5.3 序号为(39)、(40)、(41)和(42)的关联）。

④ 标明阶数

当确定各个关联的类型之后，可以初步地确定关联的阶数。随着系统分析反复改进，阶数也会经常改动。

如图 5.4 所示是经上述分析过程之后得出的案例 5-1：银行网络 ATM 系统的原始对象图。

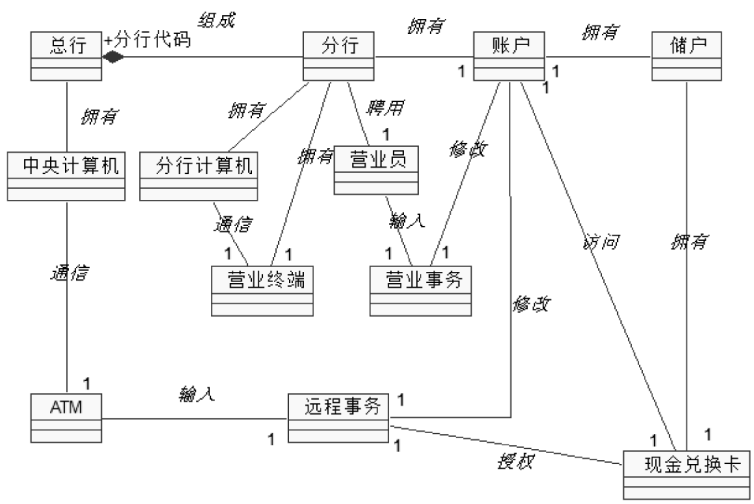

图 5.4　银行网络 ATM 系统的初始对象模型图

5.3.3　确定属性

属性是用来描述类和对象的稳定特性的，即为了完成客户规定的目标所必须保存的类的信息。一个属性是一个数据项（状态信息），类中对象都有相应的值（状态）。在面向对象分析中，“属性”用来反映问题域和系统的任务。属性能帮助人们更深入、更具体地认识类和对象和结构，能为“类和对象”以及“结构”提供更多的细节。因此，在一个系统中，确定属性是非常重要的。

一般可以从问题陈述中提取出或通过对类的理解而辨识出属性。一般说来，确定属性的过程包括分析和选择两个步骤。

（1）分析

属性是个体对象的性质，属性通常用修饰性的名词词组来表示。形容词常常表示具体的可枚举的属性值，属性不可能在问题陈述中完全表述出来，必须借助于应用域的知识及对客观世界的知识才可以找到它们。

属性的确定既与问题域有关，也和目标系统的任务有关。应该仅考虑与具体应用直接相关的属性，不要考虑那些超出所要解决的问题范围的属性。在分析过程中应该首先找出最重要的属性，以后再逐渐把其余属性增添进去。在分析阶段不要考虑那些纯粹用于实现的属性。

（2）选择

在需求陈述中找出属性或通过分析找出属性。这些属性必须是问题域中对象的基本性质，而且在目标系统中是必要的。然后，认真考察经初步分析而确定下来的那些属性，删除不

正确的和不必要的属性，选择正确的和必要的属性。最后，恰当地给属性命名。

通常有以下几种常见情况。

① 误把对象当作属性

对象是在应用领域内具有自身性质的实体。若某个实体的独立存在相当重要，而相比之下它的值不那么重要，则应把它作为一个对象而不是对象的属性。同一个实体在不同的应用领域中是作对象还是作属性，需要根据应用需求具体分析而定。

例如：在邮政目录中，“城市”是一个属性，然而在人口普查中，“城市”则被看做对象。在具体应用中，具有自身性质的实体一定是对象。

② 把链属性误作为属性

在分析过程中，不应该把链属性作为对象的属性。若某个性质依赖于某个关联链（具体上下文）的存在，则该性质是链属性而不是属性，则可考虑把该属性重新表述为一个限定词。链属性在多对多关联中很明显。在整个开发过程中，不要把它作为两个关联对象中任意一个的属性。

③ 把限定误当成属性

名称常常作为限定词而不是对象的属性，当名称不依赖于上下文关系时，名称即为一个对象属性，尤其是它不唯一时。当属性固定下来后，能减少关联的阶数时，则可将该属性重新定义成为一个限定词。

在案例 5-1：银行网络 ATM 系统中，分行代码、聘员号、账号和站号等都是限定词，而不要把它误认为属性。

④ 误把内部状态当成了属性

若属性描述了对外不透明的对象的内部状态，则应从对象模型中删除该属性。

⑤ 过于细化

一个对象的属性不能过于细化，在分析过程中，应去掉那些对大多数操作没有影响的属性。

⑥ 存在不一致的属性

在考虑对象模糊性时，引入对象标识符表示，在对象模型中不列出这些对象标识符，它是隐含在对象模型中，只列出存在于应用域的属性。

5.3.4 识别主题

在开发大型、复杂系统的过程中，为了降低复杂程度，人们习惯于把系统再进一步划分成几个不同的主题，也就是在概念上把系统包含的内容分解成若干个范畴。

主题是一种指导开发者或用户研究大型复杂模型的机制，是把一组具有较强联系的类组织在一起而得到的类的集合。主题是一种手段，有助于分解大型项目以便分组承担任务。主题还可以给出面向对象分析和设计的模型总体概貌。主题所依据的原理是整体-部分关系的扩充。一个系统模型可以包含多个主题。也就是说，主题是整个问题域和系统任务的一部分，是用来与整个问题域和系统任务（总体）进行通信的部分。

(1) 主题的特点

① 是由一组类构成的集合。

② 一个主题内部的对象类应具有某种意义上的内在联系。

③ 描述系统中相对独立的组成部分（如一个子系统）。

④ 描述系统中某一方面的事物（如人员、设备）。

⑤ 解决系统中某一方面的问题（如输入输出）。

⑥ 主题的划分有一定的灵活性和随意性。

(2) 如何划分主题

是否划分主题要看目标系统的大小。对于含有较多对象的系统,应采用选择、精炼和构造的方法来确定主题。

首先,由高级分析员粗略地识别对象和关联,初步选取结构中最上层的类作为一个主题。经进一步分析,在更深入了解系统结构的基础上,修改和精炼主题,通过实例连接互相联系的类可划分到一个主题。最后,按问题领域构造出一个主题(主题编号和主题名),应该将相互间依赖和交互较多的对象确定为同一个主题。把不属于任何结构,也没有实例连接的类作为一个主题。

按问题领域确定主题,应该将相互间依赖和交互较多的对象确定为同一个主题。以案例 5-1:银行网络 ATM 系统为例,加入限定词的系统对象模型图。我们确定"总行"、"分行"和"ATM"等为系统中的三个主题,用(1)、(2)和(3)分别表示这三个主题的编号,如图 5.5 所示。该例不是很复杂,可以不引入主题层,在这里主要是为了说明如何确定主题。为了使图简单、清晰,在下面的章节中讨论这个例子时将忽略主题层。

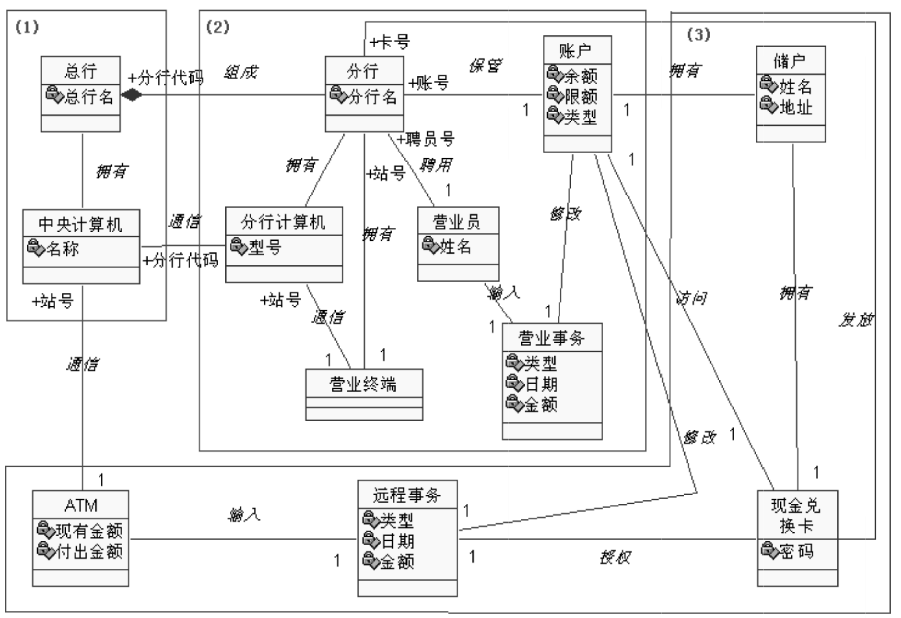

图 5.5 加入限定词的银行网络 ATM 系统的对象模型图

备注:

① "卡号":前面的分析过程中,遗漏了"分行发放现金兑换卡"这一关联,现在发现了,因而就把"卡号"这个限定词补上。

② "账号"是关联"分行保管账号"上的限定词。

③ "聘员号"是关联"分行聘用营业员"上的限定词。

"站号"是关联"分行拥有营业终端"、"营业终端与分行计算机通信"和"中央计算机与 ATM 通信"等上的限定词。

5.3.5 识别结构

确定了类的属性后,就可以利用继承来共享公共的性质,以结构的形式重新组织类。结构

是问题域复杂关系的表示，它与系统的任务直接相关。一般-特殊结构具有继承性，一般类和对象的属性和方法一旦被识别，即可在特殊类和对象中使用。

对系统中众多的类加以组织，一般说来，可以使用两种方式建立继承（即归纳）关系。

（1）自底向上

自底向上抽象出现有类的共同性质泛化出父类，通过查找出具有相似的属性、操作或关联的类来发现继承，这个过程实质上模拟了人类的归纳思维过程。

在案例 5-1：银行网络 ATM 系统中，“远程事务”和“营业事务”可以一般化为“事务”（父类）。也可以将“ATM”和“营业终端”一般化为“输入站”（父类）。

在识别中，应尽可能应用基于客观世界边界的常用分类结构，如不能直接使用现有的类，可以将属性或类稍加细化再表示出来。对称性常有助于发现某些丢失的类。

（2）自顶向下

自顶向下把现有的类细化成更具体的子类，这模拟了人类的演绎思维过程。通常，具体化的子类可以在应用领域中直接找出来。如具体化类与现有实际情况矛盾时，说明该类定义不当，需要重新考虑。

在案例 5-1：银行网络 ATM 系统中，“远程事务”和“营业事务”是“事务”（父类）的具体化类（子类）。同样，“ATM”和“营业终端”是“输入站”（父类）的具体化类（子类）。

又例如：菜单，可以有固定菜单、顶部菜单、弹出菜单、下拉菜单等，这就可以把菜单类具体细化为各种具体菜单的子类。当同一关联名出现多次且意义也相同时，应尽量具体化为相关联的类。

在类层次结构中，可以为具体的类分配属性和关联。特殊类共有的属性应放在父类中。特殊类中应定义自己独有的属性，当然它可以继承父类的属性。以案例 5-1：银行网络 ATM 系统为例，加入继承的 ATM 对象模型如图 5.6 所示。

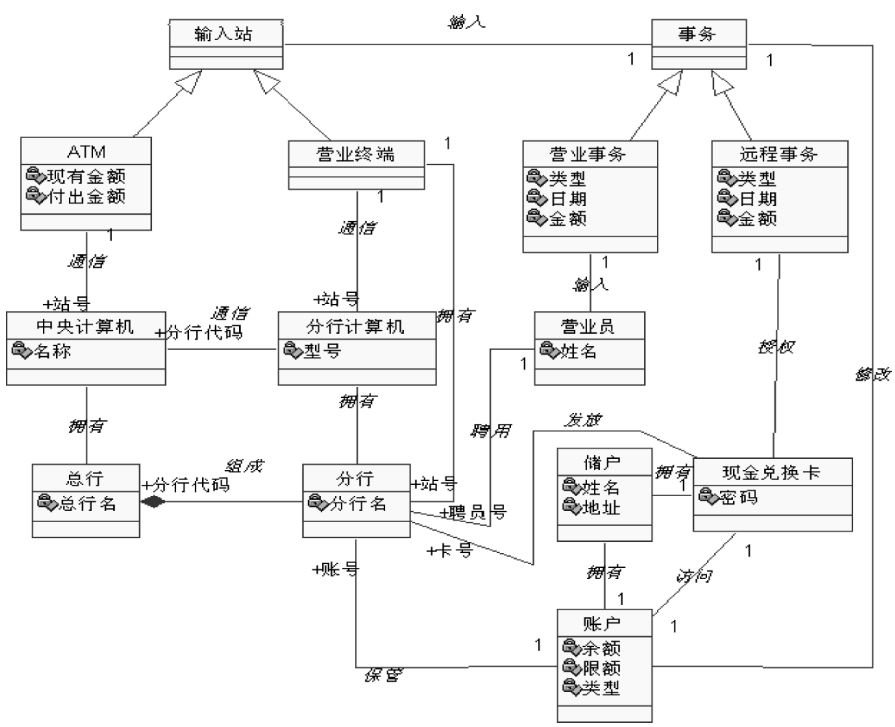

图 5.6 加入继承的银行网络 ATM 系统的对象模型

5.3.6 定义服务

(1) 访问对象属性的操作

在对象模型中,对类中定义的每个属性都是可以访问的,应该提供访问这些属性的服务。因此,需要定义访问这些属性的读、写操作。这些操作在对象模型中没有显式表示出来,但隐含在属性内。

操作定义了对象的行为并以某种方式修改对象的属性值。操作可以通过对系统的过程叙述的分析提取出来,通常叙述中的动词可作为候选的操作。类所选择的每个操作展示了类的某种行为。在确定类中应有的服务时,既要考虑该类实体的常规行为,又要考虑在本系统中特殊需要的服务。

(2) 来自事件驱动的操作

发往对象的事件驱动修改对象状态(即属性值),对象被驱动后的行为可定义成为一个操作,并通过执行该操作提供相应的服务。也就是说,当对象接收到事件后,在事件驱动下完成相应的服务。

在案例 5-1:银行网络 ATM 系统中,发往分行的事件"请分行验卡"驱动该对象的服务"验证卡号";而事件"处理分行事务"驱动分行对象的服务"更新账户"。

(3) 处理对应的操作

数据流图中的每个处理都对应于一个对象(也可能是若干个对象)上的操作。可将完成每个处理的功能定义成为相应的操作。应该仔细对照状态机图和数据流图,以便更正确地确定对象应该提供的服务。

在案例 5-1:银行网络 ATM 系统中,数据流图上的处理"验证密码"就可以定义成为一个"验证密码"操作,该分行对象通过执行这个操作提供"验证密码"服务。

(4) 消除冗余操作

应该尽量利用继承机制以减少所需定义的服务数目。只要不违背领域知识和常识,就尽量将相似类(子类)中共享的属性和操作抽取出来,以建立这些类的新父类,并在类等级的不同层次中正确地定义各个服务。可利用继承关系消除冗余的定义、共享的属性和操作,简化实现。

5.3.7 完善对象模型

在建模的任何一个阶段中,一旦发现了模型的缺陷,就必须返回到前面阶段进行修改。有些细化工作(如定义服务)要等到动态模型和功能模型建完以后才能进行。

在建模的过程中,不一定按前述的工作顺序进行,分析员完全可以以自己的独特方法进行,既可以将几个阶段并行处理,又可以随意组织前述工作顺序。

通过以上各步,对象模型就建立起来了,但这样不能确保模型是完全正确的。事实上,软件开发过程就是一个反复修改,逐步完善与完善的过程。如果是初次使用面向对象方法,建议还是按照前述顺序进行比较好。由于面向对象的概念和符号在整个开发过程中都是一致的,因此远比使用结构化分析和设计技术更容易实现反复修改及逐步完善的过程。

(1) 几种可能丢失对象的情况及解决办法

① 同一类中存在毫无关系的属性和操作,则分解这个类,使各部分相互关联。

② 一般化体系不清楚,则可能分离扮演两种角色的类。

③ 存在无目标类的操作,则找出并加上失去目标的类。

④ 存在名称及目的相同的冗余关联，则通过一般化创建丢失的父类，把关联组织在一起。

（2）删除冗余的类

如果某类中缺少属性、操作和关联，则可删除这个类。

（3）补充关联

丢失了操作的访问路径，则加入新的关联以回答查询。

在案例 5-1：银行网络 ATM 系统中，一个“事务”由若干个“更新”组成，它们构成整体-部分关系。一个“更新”是一个动作，即对账户所做的一次处理，如存款、取款、查询等。“更新”有类型、金额等属性，所以，可补充定义成为一个单独类，“事务”与它构成整体-部分关系。

（4）分解类

在案例 5-1：银行网络 ATM 系统中，“现金兑换卡”可分为“卡权限”和“现金兑换卡”两个功能，前者表示储户访问账户的权限，后者则表示含有分行代码和卡号的数据载体。

（5）合并类

如在一个应用系统中，两个类虽然名字不同，但是它们所完成的任务以及与其他类的关系也相同，这时可将这两个类合并成为一个类。

在案例 5-1：银行网络 ATM 系统中，“分行”与“分行计算机”合并为“分行”。同样，可将“总行”与“中央计算机”合并成为“总行”。

以案例 5-1：银行网络 ATM 系统的为例，通过进一步的完善，得出如图 5.7 所示的 ATM 对象模型。

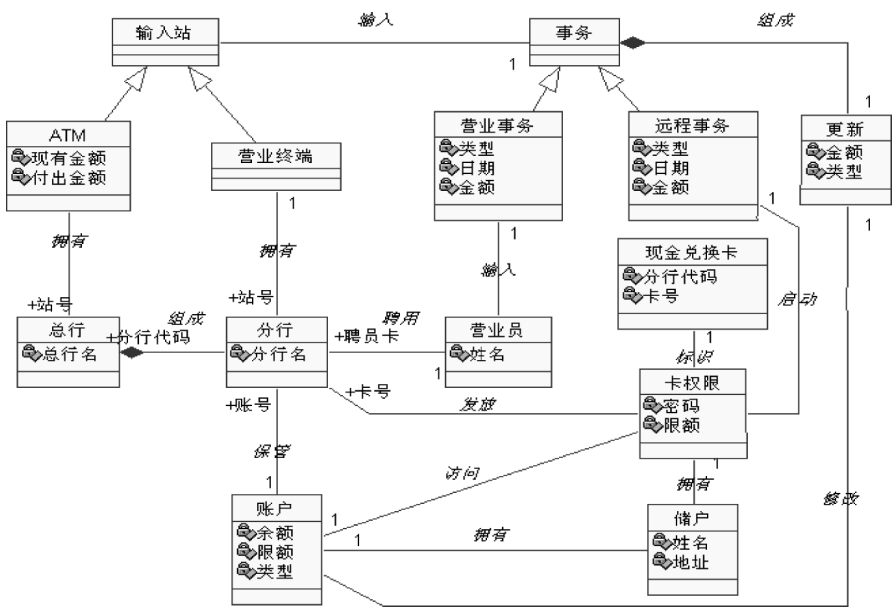

图 5.7 进一步完善的银行网络 ATM 系统的对象图

5.4 建立动态模型

当对象模型建立起来后，接着可以建立动态模型。对于一个系统来说，功能模型指明了系统应该“做什么”，而动态模型则明确规定了“什么时候做”。即在何种状态下、接受了什么事件的触发，来确定对象的可能事件的顺序。

动态模型是与时间和变化有关的系统性质。该模型描述了系统的控制结构，它表示了瞬间的、行为化的系统控制性质，它关心的是系统的控制，操作的执行顺序，它表示从对象的事件和状态的角度出发，表现了对象的相互行为。该模型描述的系统属性是触发事件、事件序列、状态、事件与状态的组织。在分析阶段不考虑算法的执行，它是实现模型的一部分。通常使用时间跟踪表或状态机图作为描述工具。对于数据库系统来说，动态模型并不重要，如果是交互式系统，建立动态模型却是非常重要的。

建立动态模型的步骤：建立动态模型的第一步，是编写典型交互行为的脚本。虽然脚本中不可能包括每个偶然事件，但是，至少必须保证不遗漏常见的交互行为。接下来从脚本中提取出事件，确定触发每个事件的动作对象以及接受事件的目标对象。第三步，排列事件发生的次序，确定每个对象可能有的状态及状态间的转换关系，并用状态机图描绘它们。最后，比较各个对象的状态机图，检查对象之间的一致性，确保事件之间的匹配。

本节结合案例 5-1：银行网络 ATM 系统的实例将进行面向对象的分析，对动态模型的建立方法给予叙述。

5.4.1 准备脚本

在建立动态模型的过程时，为了确保整个交互过程的正确性和清晰性，保证不遗漏重要的交互步骤，对目标系统的行为有更具体的认识，首先要编写脚本，用脚本表示系统的行为，为建立动态模型奠定基础。在分析阶段不考虑算法的执行，算法是实现模型的一部分。

所谓“脚本”，原意是指“表演戏曲、话剧，拍摄电影、电视剧等所依据的本子，里面记载台词、故事情节等”。在建立动态模型的过程中，脚本是指在某一执行期间内系统中的对象（或其他外部设备）与目标系统之间发生一个或多个典型的互换信息时产生的事件，所互换的信息值就是该事件的参数，对于各事件，应确定触发事件的动作对象和该事件的参数。

编写脚本的过程是分析用户对系统交互行为的需求的过程，需要用户参与，提出意见，并审查和更改。例如在案例 5-1：银行网络 ATM 系统的需求陈述中，虽然表明了应从储户那里获得有关事务的信息，但并没有准确说明获得信息的具体过程和需要什么参数，动作顺序如何等还是模糊的。

脚本包括“正常脚本”、“例外脚本”。首先编写正常情况的脚本。然后，考虑特殊情况，例如输入或输出的数据为最大值（或最小值）。最后，考虑用户出错情况，例如，输入的值为非法值或响应失败等。此外，还应该考虑在基本交互行为之上的“通用”交互行为，如帮助要求和状态查询等。

案例 5-1：银行网络 ATM 系统正常情况下的脚本，如表 5.4 所示。

表 5.4 银行网络 ATM 系统正常情况下的脚本

序号	脚　本
1	ATM 请储户插卡；储户插入一张现金兑换卡
2	ATM 接受该卡并读它上面的分行代码和卡号
3	ATM 要求储户输入密码；储户输入自己的密码，如“1234”等数字
4	ATM 请求总行验证卡号和密码；总行要求“9”号分行核对储户密码，然后通知 ATM 这张卡有效
5	ATM 要求储户选择事务类型（取款、转账、查询等）；储户选择“取款”
6	ATM 要求储户输入取款额；储户输入如“100”等数字
7	ATM 确认取款额在预先规定的限额内，然后要求总行处理这个事务；总行把请求转给分行，该分行成功地处理完这项事务并返回该账户的新余额

续表

序号	脚　　本
8	ATM 输出现金并请储户拿走这些现金；储户拿走现金
9	ATM 问储户是否继续这项事务；储户回答“不”
10	ATM 打印账单，退出现金兑换卡，请储户拿走它们；储户取走账单和卡

案例 5-1：银行网络 ATM 系统异常情况下的脚本，如表 5.5 所示。

表 5.5　银行网络 ATM 系统异常情况下的脚本

序号	脚　　本
1	ATM 请储户插卡；储户插入一张现金兑换卡
2	ATM 接受这张卡并顺序读它上面的数字
3	ATM 要求密码；储户误输入如“1111”等数字
4	ATM 请求总行验证输入的数字和密码；总行在向有关分行咨询之后拒绝这张卡
5	ATM 显示“密码错”，并请储户重新输入密码；储户输入“1234”等数字；ATM 请总行验证后知道这次输入的密码正确
6	ATM 请储户选择事务类型；储户选择“取款”
7	ATM 询问取款额；储户改变主意，不想取款了，随之按“取消”键
8	ATM 退出现金兑换卡，并请储户拿走它；储户拿走他的卡
9	ATM 请储户插卡

5.4.2　确定事件

应该认真分析脚本的各个步骤，以便从中确定所有外部事件。事件是指已发生并可能引发某种活动的一件事。事件包括系统与用户(或外部设备)交互的所有信号、发送者、接收者、输入、输出、中断、转换和动作等。从脚本中容易发现正常事件，但是，应注意不要遗漏了出错条件和异常事件。

传递信息的对象的动作也是事件。经过分析，应该区分出每类事件的发送对象和接收对象。一类事件相对它的发送对象来说是输出事件，但是相对它的接收对象来说则是输入事件。有时一个对象把事件发送给自己，在这种情况下，该事件既是输出事件又是输入事件。

5.4.3　准备事件跟踪图

完整、正确的脚本为建立动态模型奠定了必要的基础。但是，用自然语言书写的脚本往往不够简明，而且有时在阅读时会有二义性。为了有助于建立动态模型，通常在画状态机图之前先画出事件跟踪图，为此首先需要进一步明确事件及事件与对象的关系。事件跟踪图实质上是扩充的脚本。接着在脚本和事件跟踪图的基础上，再画出状态机图。最后，由状态机图组成动态模型。

案例 5-1：银行网络 ATM 系统的事件跟踪图，如图 5.8 所示。

事件跟踪图能形象、清晰地表示事件序列以及事件与对象的关系。在事件跟踪图中，一条竖线代表一个类和对象，每个事件用一条水平的箭头线表示，箭头方向从事件的发送对象指向接收对象。事件按照先后顺序排列，时间从上向下递增，也就是说，画在最上面的水平箭头线

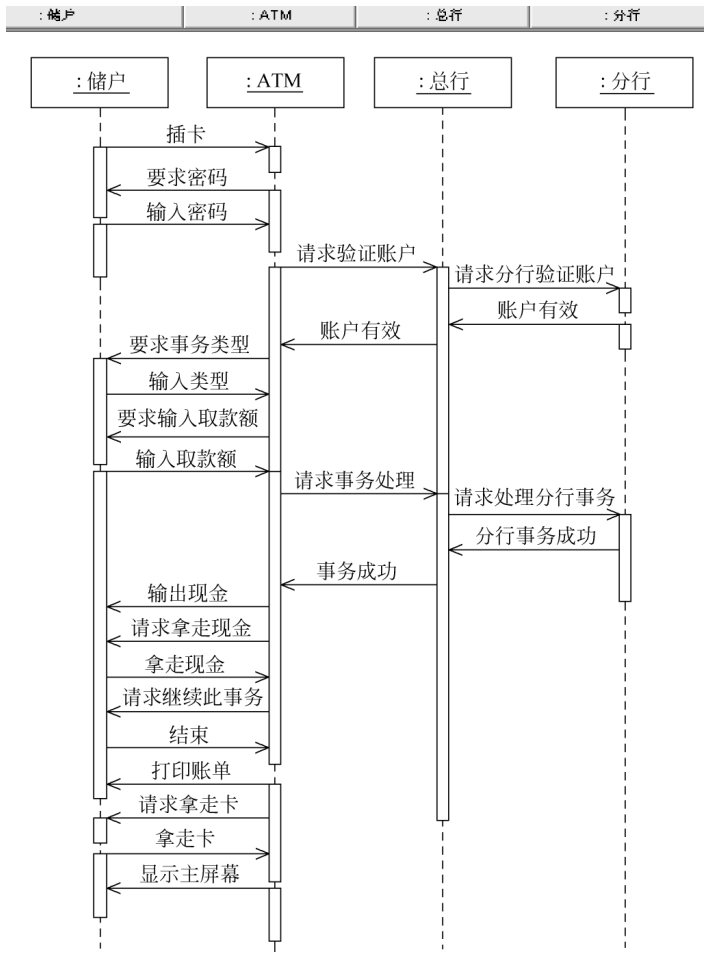

图 5.8　银行网络 ATM 系统的事件跟踪图

代表最先发生的事件，画在最下面的水平箭头线所代表的事件最晚发生。箭头线之间的间距并不表示两个事件之间的精确时间差，图中仅用箭头线在垂直方向上的相对位置表示事件发生的先后次序，并不表示两个事件之间的精确时间差。

5.4.4　构造状态机图

状态机图通常是对类描述的补充，它说明该类的对象所有可能的状态，以及哪些事件将导致状态的改变。状态机图描述了对象的动态行为，是一种对象生存周期的模型。

对各对象类建立状态机图，反映对象接收和发送的事件，每个事件跟踪都对应于状态机图中的一条路径。所有对象都具有状态，状态是对象执行了一系列活动的结果。当某个事件发生后，对象的状态将发生变化。

画出事件跟踪图后，可根据事件跟踪图再画出状态机图。从一张事件跟踪图出发画状态机图时，应该集中精力仅考虑影响一类对象的事件，也就是说，仅考虑事件跟踪图中指向某条竖线的那些箭头线。把这些事件作为状态机图中的有向边（即箭头线），边上标以事件名。两个事件之间的间隔就是一个状态。一般说来，如果同一个对象对相同事件的响应不同，则这个对象处在不同状态。应该尽量给每个状态取个有意义的名字。通常，从事件跟踪图中当前考

虑的竖线射出的箭头线，是这条竖线代表的对象达到某个状态时所做的行为（往往是引起另一类对象状态转换的事件）。

状态机图描绘事件与对象状态的关系。当某个对象接收了一个事件以后，会转换成什么样的状态，这取决于该对象的当前状态和所接收的事件。由事件引起的状态改变称为“转换”。通常，用一张状态机图描绘一类对象的行为，它确定了由事件序列引出的状态序列。但是，也不是任何一个类和对象都需要有一张状态机图描绘它的行为。很多对象仅响应与过去历史无关的那些输入事件，或者把历史作为不影响控制流的参数。对于这类对象来说，状态机图是不必要的。系统分析员应该集中精力仅考虑具有重要交互行为的那些类。

（1）确定状态机图中的事件与类-对象

一般情况下，状态机图确定了由事件序列引出的状态序列，因而，可用一张状态机图描绘一类对象的行为。

在动态模型中，并不是任何一个类-对象的行为都需要用一张状态机图描绘，只需考虑那些具有重要交互行为的类就行了。

（2）状态机图的表示

一个状态机图反映对象接收和发送的事件：每个脚本或事件跟踪图都对应状态机图中的一条路径（即箭头线），路径上应标以事件名。两个事件之间的间隔就是一个状态，应给每个状态取个有意义的名字。这就是事件和状态的一个序列初始图。

（3）画状态机图的策略

① 考虑分支点

画出了初始状态机图之后，再合并其他脚本的事件跟踪图到该初始的状态机图中。首先，在以前考虑过的脚本中找出分支点。然后，把其他脚本中的事件序列作为一条可选的路径并入已有的状态机图中。

② 考虑异常情况

状态机图不但要考虑正常事件，还需要考虑边界情况、特殊情况和异常情况（例如，用户要求取消正在处理事务）。当发生了异常事件后，系统应给出出错处理的脚本，并且并入已有的状态机图中。

③ 补充遗漏情况

状态机图的构造应考虑所有脚本，并且包含影响某类对象状态的全部事件。因而，在完成初始状态机图后，应进一步检查状态机图，发现有遗漏的情况，应该立即补充遗漏脚本，并且并入已有的状态机图中。

（4）实例分析

在案例 5-1：银行网络 ATM 系统中，“现金兑换卡”、“事务”和“账户”等是被动对象，并不发送事件；“储户”和“营业员”是系统外部的动作对象，无须在系统内实现它们；“ATM”、“营业终端”、“总行”和“分行”都是相互发送事件的主动对象，由于“营业终端”的状态机图和“ATM”的状态机图类似，因此，只需要考虑 ATM、总行、分行的状态机图。

“ATM”、“总行”和“分行”的状态机图如图 5.9～图 5.11 所示。这些状态机图都是简化的、粗略的，尤其对异常情况和出错情况等考虑不周（例如，图 5.9 并没有表示在网络通信链路不通时的系统行为，在这种情况下，ATM 停止处理储户事务）。

案例 5-1：银行网络 ATM 系统的 ATM 类的状态机图，如图 5.9 所示。

案例 5-1：银行网络 ATM 系统的总行类的状态机图，如图 5.10 所示。

案例 5-1：银行网络 ATM 系统的分行类的状态机图，如图 5.11 所示。

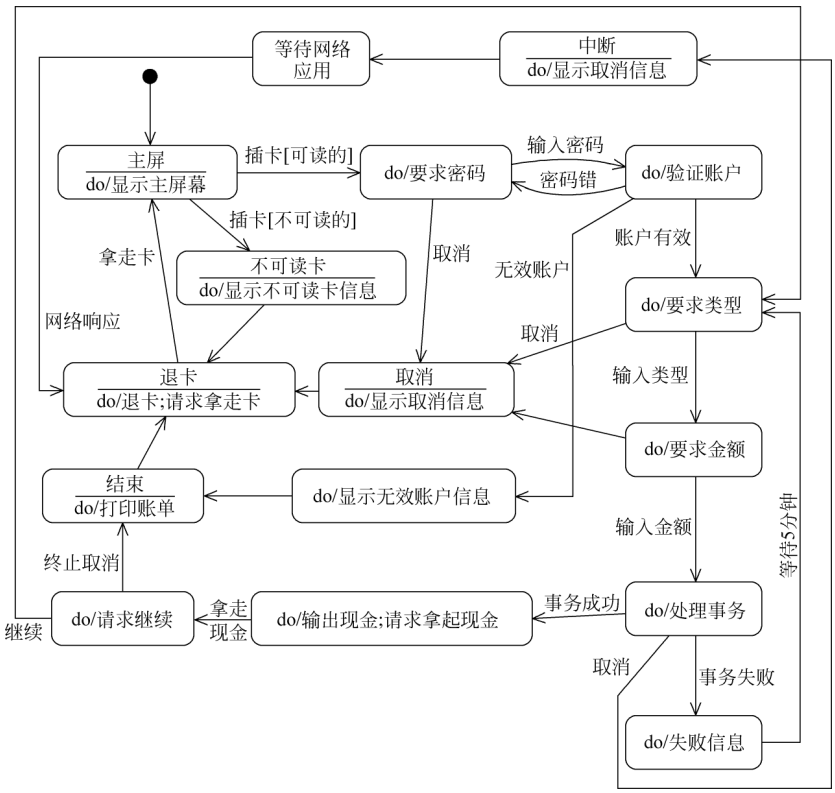

图 5.9 银行网络 ATM 系统的 ATM 类的状态机图

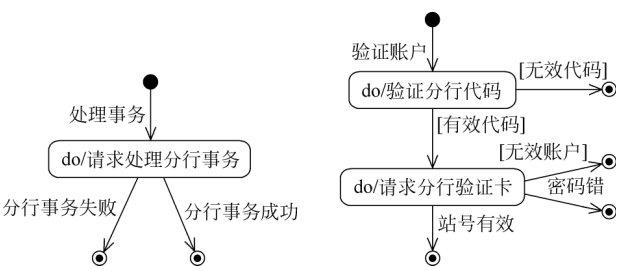

图 5.10 银行网络 ATM 系统的总行类的状态机图

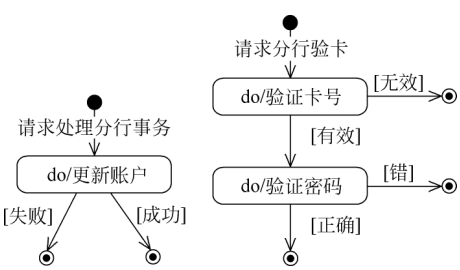

图 5.11 银行网络 ATM 系统的分行类的状态机图

5.4.5 完善动态模型

每个类的动态行为用一张状态机图来描绘，各个类的状态机图通过共享事件合并起来，从而构成系统的动态模型。多个类的状态机图完成之后，还需要检查系统级的完整性和一致性。每个事件应该有个发送者和接收者，当发送者和接收者是同一个对象时，对无前驱或后续的状态应该着重检查，如果这种状态既不是交互序列的起点，又不是终点，则一定是个错误。也就是说，动态模型是基于事件共享而互相关联的一组状态机图的集合。

以案例 5-1：银行网络 ATM 系统为例，如总行类的状态机图中，事件“分行代码错”是由总行发出的，但是在 ATM 类的状态机图中，并没有一个状态接收这个事件。因此，在 ATM 类的状态机图中应该再补充一个状态“do：显示分行代码错信息”，它接收由前驱状态“do：验证账户”发出的事件“分行代码错”，它的后续状态是“退卡”。

5.5 建立功能模型

功能模型描述的是外部执行者(Actor)所理解的系统功能。它描述了待开发系统的功能需求,被广泛应用到了面向对象的系统分析中。功能模型指出发生了什么,动态模型确定什么时候发生,而对象模型确定发生的客体。

5.5.1 确定基本系统模型图

基本系统模型是用来确定系统的边界和输入/输出数据流的,表明一个计算如何从输入值得到输出值,表明值之间的依赖关系及相关的功能,它不考虑计算的次序。基本系统模型仅包含一个加工,它代表被开发系统的加工和变换数据的整体功能。它的输入流是该系统的输入数据,输出流是系统的输出数据。建立功能模型时,先列出输入、输出值,输入输出值是系统与外部世界之间的事件等参数。然后,检测问题陈述,从中找出遗漏的所有输入输出值。

以案例 5-1:银行网络 ATM 系统为例,基本系统模型如图 5.12 所示。它包含两个外部实体,一个是储户,它既是数据源点又是数据终点,储户将事务(如密码、事务类型、金额等)由营业员通过营业终端提交给系统,或系统可给储户提供信息(如金额、账单等);另一个是现金兑换卡,它是数据源点,系统从它上面读取分行代码和卡号等信息。

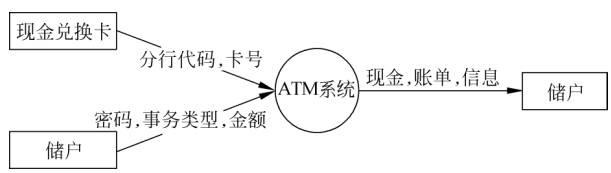

图 5.12 银行网络 ATM 系统的基本系统模型

5.5.2 细化数据流图

功能模型由多张数据流图组成。数据流图用来表示从源对象到目标对象的数据值的流向,说明输出值是怎样从输入值得来的。数据流图通常按层次组织,最顶层由单个处理组成,也可由收集输入、计算值及生成结果的一个综合处理构成。数据流图不包含控制信息,控制信息在动态模型中表示。同时,数据流图也不表示对象中值的组织,值的组织在对象模型中表示。

数据流图有助于表示功能依赖关系,其中的处理对应于状态机图的活动和动作,其中的数据流对应于对象图中的对象或属性。数据流图中包含处理、数据流、动作对象和数据存储对象。

(1) 处理

把功能模型中的处理框逐步分解,得到描述系统加工和变换数据的基本功能的若干个处理框。处理用来改变数据值。最底层处理是纯粹的函数,一张完整的数据流图是一个高层处理。

(2) 数据流

数据流图中的数据流将对象的输出与处理、处理与对象的输入、处理与处理联系起来。在一个计算机中,用数据流来表示中间数据值,数据流不能改变数据值。

(3) 动作对象

动作对象是一种主动对象,它通过生成或者使用数据值来驱动数据流图。

(4) 数据存储对象

数据流图中的数据存储是被动对象,它用来存储数据。它与动作对象不一样,数据存储本身不产生任何操作,它只响应存储和访问的要求。

首先，把基本系统模型中的处理框逐步分解，得到描述系统加工和变换数据的基本功能的若干个处理框，然后，构成功能级数据流图。

案例 5-1：银行网络 ATM 系统的功能级数据流图如图 5.13 所示。

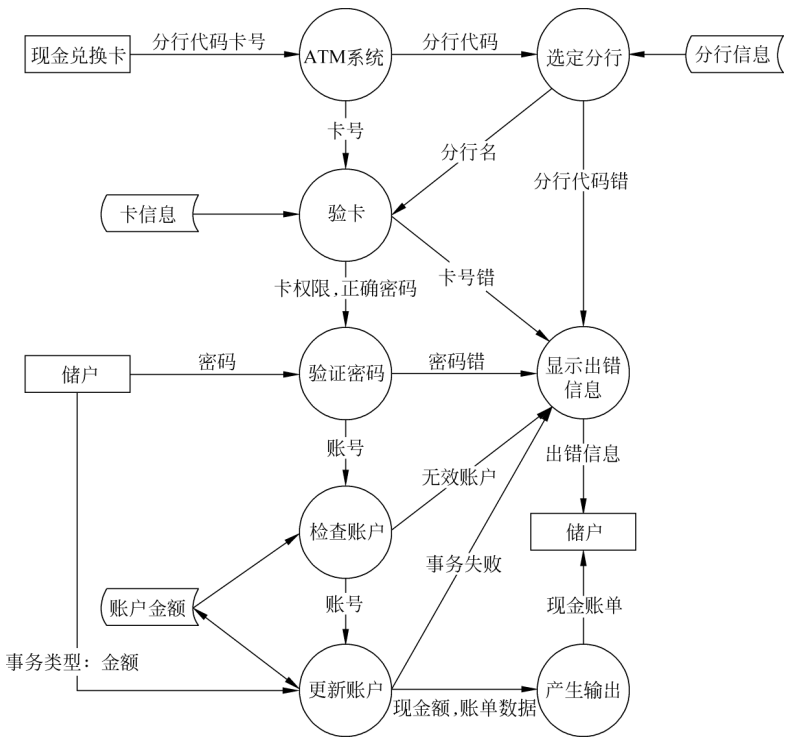

图 5.13　银行网络 ATM 系统的功能级数据流图

5.5.3　功能描述

当数据流图已分解到一定程度后，就应该对各个处理进行描述。描述功能可用自然语言、流程图、IPO 图（或表）和伪码等工具，描述可用说明性的或过程性的。说明性描述确定了输入、输出值之间的关系。说明性描述优于过程性描述，因为它隐含实现的考虑。过程性描述确定一个算法来实现处理功能，算法只是用来确定处理干什么。过程性描述着重描述实现处理功能的算法。

案例 5-1：银行网络 ATM 系统“更新账户”的功能描述，如表 5.6 所示。

表 5.6　银行网络 ATM 系统“更新账户”的功能描述

```
输入：账户，数量，事务类型
输出：现金，收据，信息
IF 取款数目超过账户当前余额
   THEN   退出事务，不付出现金
   IF 取款数目不超过账户当前余额
      THEN  记账并付出储户要求的现金
      IF 事务是存款
         THEN  建立账户并无现金付出
         IF 事务是状态请求
            THEN  无现金付出
```

在上述任何情况下，显示账单内容为：ATM 编号、日期、时间、账户编号、事务类型、事务数量(若有)以及新的余额。

5.6 面向对象分析实例

案例 5-2：饮料自动售货机系统的面向对象的分析实例。

一个饮料自动售货机可以放置 5 种不同或部分相同的饮料，可由厂商根据销售状况自动调配，并可随时重新设置售价，但售货机最多仅能放置 50 罐饮料，其按钮设计在各种饮料样本的下方，若经金额计算器累计金额足够，则“选择”键灯会亮；若某一种饮料已销售完毕，则“售完”灯会亮。

顾客将硬币投入售货机，经累加金额足额的饮料“选择”键灯亮，等顾客按键选择。顾客按键后饮料由取物口掉出，并自动结算及找钱。

顾客可在按下选择键前任何时刻，拉动退币杆取消交易收回硬币。

分析过程：

① 自动售货机可以放置 5 种不同或部分相同的饮料。

② 厂商根据销售状况自动调配。

③ 厂商设置售价。

④ 售货机最多仅能放置 50 罐饮料。

⑤ 金额计算器累计金额。

⑥ “选择”键灯会亮。

⑦ 饮料已销售完毕。

⑧ “售完”灯会亮。

⑨ 顾客将硬币投入售货机。

⑩ 顾客按键选择。

⑪ 饮料由取物口掉出。

⑫ 结算及找钱。

⑬ 顾客拉动退币杆。

⑭ 顾客取消交易。

候选对象-类：售货机，饮料，厂商，销售状况，售价，按钮，金额计算器，金额，选择键；“售完”灯，顾客，硬币，取物篓，退币杆，交易。

筛选之后：售货机，金额计算器，选择钮；顾客，退币杆。

隐含对象：存量计数器。

案例 5-2：饮料自动售货机系统的对象图，如图 5.14 所示。

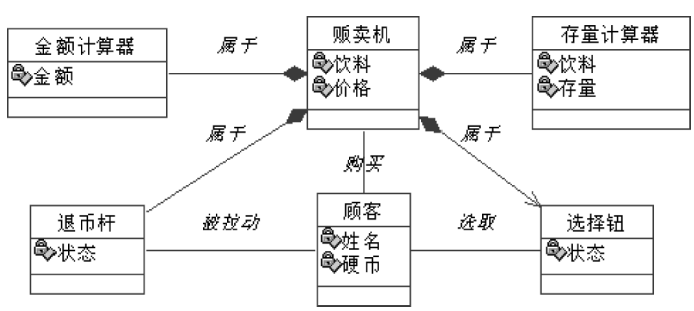

图 5.14 饮料自动售货机系统的对象图

案例 5-2：饮料自动售货机系统的事件追踪图，如图 5.15 所示。

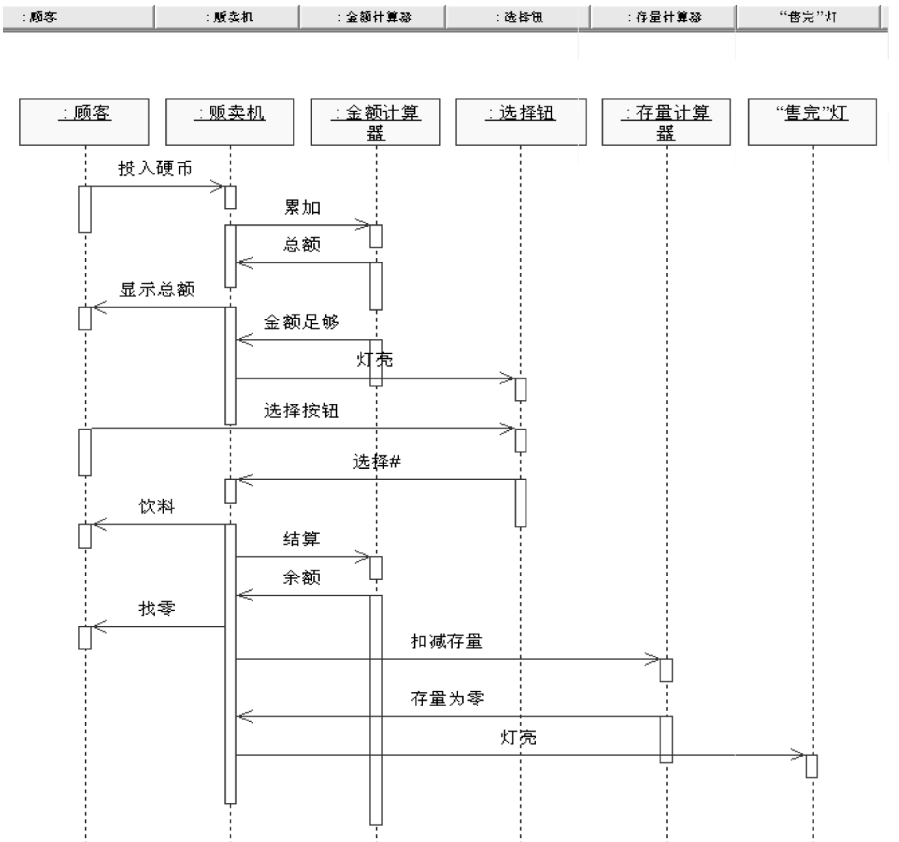

图 5.15　饮料自动售货机系统的事件追踪图

案例 5-2：饮料自动售货机系统正常情况下的脚本，如表 5.7 所示。

表 5.7　饮料自动售货机系统正常情况下的脚本

序号	脚　　本
1	售货机请顾客投币；顾客投币到售货机
2	售货机接受投币，由金额计数器累加币值
3	金额计数器将累加币值显示在自动售货机上
4	币值足额时，选择钮灯亮
5	顾客按钮选择饮料，相应的饮料由售货机取物口掉出
6	金额计数器计算余额，售货机将余额退回顾客
7	存量计数器扣减出售饮料数量，若数量为零，“售完”灯亮

案例 5-2：饮料自动售货机系统的状态机图，如图 5.16 所示。

案例 5-2：饮料自动售货机系统的初始功能图，如图 5.17 所示。

案例 5-2：饮料自动售货机系统的功能图，如图 5.18 所示。

案例 5-2：饮料自动售货机系统的对象图，如图 5.19 所示。

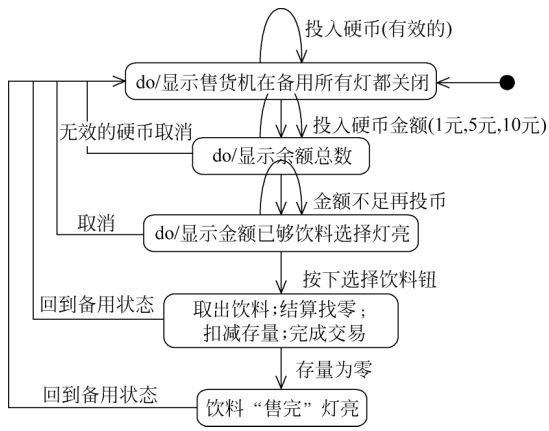

图 5.16 饮料自动售货机系统的状态机图

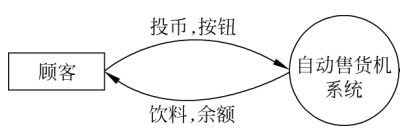

图 5.17 饮料自动售货机系统的初始功能图

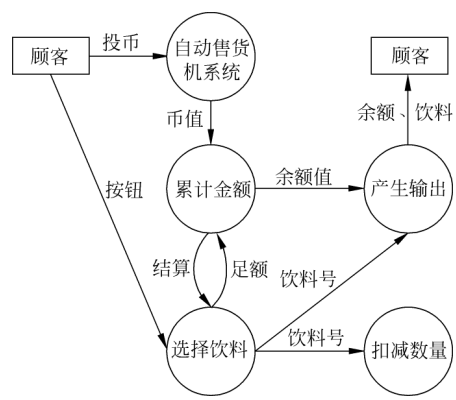

图 5.18 饮料自动售货机系统的功能图

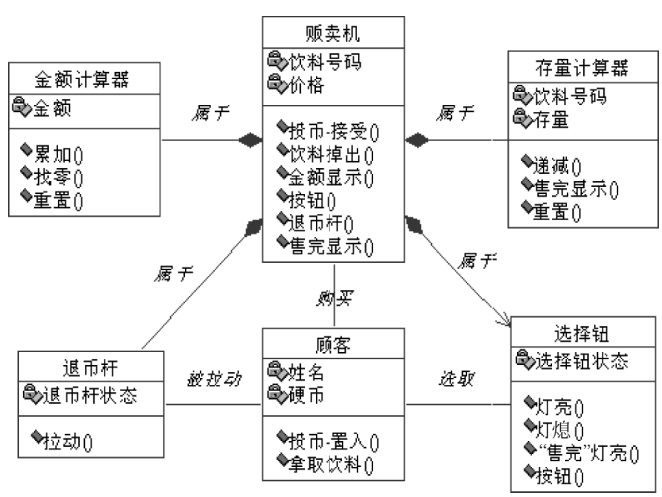

图 5.19 饮料自动售货机系统对象图

5.7 本章小结

本章全面介绍了面向对象的分析方法，包括面向对象的分析模型及分析过程。详细阐述了面向对象的分析过程的需求陈述，建立静态模型，建立动态模型及功能模型的内容。其中建立静态模型在完成寻找类与对象、确定关联、确定属性、识别主题、识别结构及定义服务之后，可得到进一步完善的对象模型；建立动态模型在完成准备脚本、确定事件、准备事件跟踪图及构造状态机图之后，可得到进一步完善的动态模型；建立功能模型包括确定基本系统模型图、细化的数据流图及功能描述。

本章主要列举银行网络 ATM 系统和饮料自动售货机系统等实例对面向对象分析的相关理论进行描述和说明。

学习本章之后，可以进一步掌握面向对象分析的过程及方法，为面向对象设计做好准备。

习题 5

一、判断题

1. 不可以用自然语言描述功能。

2. 静态建模是指对象之间通过属性互相联系，而这些关系不随时间而转移，即建立对象模型。

3. 可以从问题陈述中提取出或通过对类的理解而辨识出属性。

4. 是否划分主题要看目标系统的大小。对于含有较多对象的系统，应采用选择、精炼和构造的方法来确定主题。

5. 确定了类的属性后，就可以利用继承来共享公共的性质，以结构的形式重新组织类。

6. 结构是问题域复杂关系的表示，它与系统的任务直接相关。

7. 功能模型指明了系统应该“做什么”。

8. 动态模型明确规定了“什么时候做”。

9. 事件是指已发生并可能引发某种活动的一件事。

10. 从脚本中容易发现正常事件和异常事件。

11. 面向对象分析的特点是有利于对问题及系统责任的理解，人员之间的交流，并对需求变化有较强的适应性，并支持软件复用。

12. 状态机图描绘事件与对象状态的关系。

13. 当某个对象接收了一个事件以后，会转换成什么样的状态，这取决于该对象的当前状态和所接收的事件。

14. 功能模型描述的是外部执行者所理解的系统功能。

15. 功能模型描述了待开发系统的功能需求，被广泛应用到了面向对象的系统分析中。

二、填空题

1. 构成类图的元素所表达的模型信息，分为三个层次：（ A ）、特征层和（ B ）。

2. 补充模型有（ A ）和（ B ）。

3. 可以用自然语言、（ A ）、（ B ）（或表）和（ C ）等工具描述功能。

4. 确定和标识类包括（ A ）、（ B ）、（ C ），最后将同类型的对象抽象为类。

5. 确定关联包括（ A ）关联、（ B ）关联和（ C ）关联。

6. 一般说来，确定属性包括（ A ）和（ B ）两个步骤。

7. 对于含有较多对象的系统，应采用（ A ）、（ B ）和（ C ）的方法来确定主题。

三、简答题

1. 什么是面向对象分析？
2. 说明面向对象分析的模型的构成。
3. 简述面向对象分析的过程。
4. 简述如何确定服务。
5. 什么是动态模型？
6. 简述如何准备脚本。
7. 简述如何准备事件跟踪图。
8. 简述如何确定基本系统模型图。
9. 简述面向对象分析的目的。
10. 简述面向对象分析的基本任务。
11. 为建立分析模型，要运用的是哪些基本原则？
12. 简述建立对象-行为模型的步骤。

四、综合题

完成实例图书管理系统面向对象的分析过程，包括：静态模型、动态模型和功能模型。

系统需求：

在图书馆管理系统中，要为每个借阅者建立一个账户，并给借阅者发放借阅卡（借阅卡号，借阅者名），账户存储借阅者的个人信息、借阅信息以及预订信息。

持有借阅卡的借阅者可以借阅书刊、返还书刊、查询书刊信息、预订书刊并取消预订，但这些操作都是通过图书管理员进行的，也即借阅者不直接与系统交互，而是图书管理员充当借阅者的代理与系统交互。

在借阅书刊时，需要输入所借阅的书刊名、书刊的 ISBN/ISSN 号，然后输入借阅者的图书卡号和借阅者名，完成后提交所填表格，系统验证借阅者所借阅的书刊是否存在，若存在，则借阅者可借出书刊，建立并在系统中存储借阅记录。

借阅者还可预订该书刊，一旦借阅者预订的书刊可以获得，就将书刊直接寄给预订人。另外，不考虑书刊的最长借阅期限，假设借阅者可以无限期地保存所借阅的书刊。