第 10 章 面向对象分析

不论采用哪种方法开发软件，分析的过程都是提取系统需求的过程。分析工作主要包括 3 项内容，这就是理解、表达和验证。首先，系统分析员通过与用户及领域专家的充分交流，力求完全理解用户需求和该应用领域中的关键性的背景知识，并用某种无二义性的方式把这种理解表达成文档资料。分析过程得出的最重要的文档资料是软件需求规格说明(在面向对象分析中，主要由对象模型、动态模型和功能模型组成)。

由于问题复杂，而且人与人之间的交流带有随意性和非形式化的特点，上述理解过程通常不能一次就达到理想的效果。因此，还必须进一步验证软件需求规格说明的正确性、完整性和有效性，如果发现了问题则进行修正。显然，需求分析过程是系统分析员与用户及领域专家反复交流和多次修正的过程。也就是说，理解和验证的过程通常交替进行，反复迭代，而且往往需要利用原型系统作为辅助工具。

面向对象分析(OOA)的关键是识别出问题域内的类与对象，并分析它们相互间的关系，最终建立起问题域的简洁、精确、可理解的正确模型。在用面向对象观点建立起的 3 种模型中，对象模型是最基本、最重要、最核心的。

10.1 面向对象分析的基本过程

10.1.1 概述

面向对象分析，就是抽取和整理用户需求并建立问题域精确模型的过程。

通常，面向对象分析过程从分析陈述用户需求的文件开始。可能由用户(包括出资开发该软件的业主代表及最终用户)单方面写出需求陈述，也可能由系统分析员配合用户，共同写出需求陈述。当软件项目采用招标方式确定开发单位时，“标书”往往可以作为初步的需求陈述。

需求陈述通常是不完整、不准确的，而且往往是非正式的。通过分析，可以发现和改正原始陈述中的二义性和不一致性，补充遗漏的内容，从而使需求陈述更完整、更准确。因此，不应该认为需求陈述是一成不变的，而应该把它作为细化和完善实际需求的基础。在分析需求陈述的过程中，系统分析员需要反复多次地与用户协商、讨论、交流信息，还应该通过调研了解现有的类似系统。正如以前多次讲过的，快速建立起一个可在计算机上运行的原型系统，非常有助于分析员和用户之间的交流和理解，从而能更正确地提炼出用户的需求。

接下来，系统分析员应该深入理解用户需求，抽象出目标系统的本质属性，并用模型准确地表示出来。用自然语言书写的需求陈述通常是有二义性的，

内容往往不完整、不一致。分析模型应该成为对问题的精确而又简洁的表示。后继的设计阶段将以分析模型为基础。更重要的是，通过建立分析模型能够纠正在开发早期对问题域的误解。

在面向对象建模的过程中，系统分析员必须认真向领域专家学习。尤其是建模过程中的分类工作往往有很大难度。继承关系的建立实质上是知识抽取过程，它必须反映出一定深度的领域知识，这不是系统分析员单方面努力所能做到的，必须有领域专家的密切配合才能完成。

在面向对象建模的过程中，还应该仔细研究以前针对相同的或类似的问题域进行面向对象分析所得到的结果。由于面向对象分析结果的稳定性和可重用性，这些结果在当前项目中往往有许多是可以重用的。

10.1.2 3个子模型与5个层次

正如9.3节所述，面向对象建模得到的模型包含系统的3个要素，即静态结构(对象模型)、交互次序(动态模型)和数据变换(功能模型)。解决的问题不同，这3个子模型的重要程度也不同：几乎解决任何一个问题，都需要从客观世界实体及实体间相互关系抽象出极有价值的对象模型；当问题涉及交互作用和时序时(例如，用户界面及过程控制等)，动态模型是重要的；解决运算量很大的问题(例如，高级语言编译、科学与工程计算等)，则涉及重要的功能模型。动态模型和功能模型中都包含了对象模型中的操作(即服务或方法)。

复杂问题(大型系统)的对象模型通常由下述5个层次组成：主题层、类与对象层、结构层、属性层和服务层，如图10.1所示。

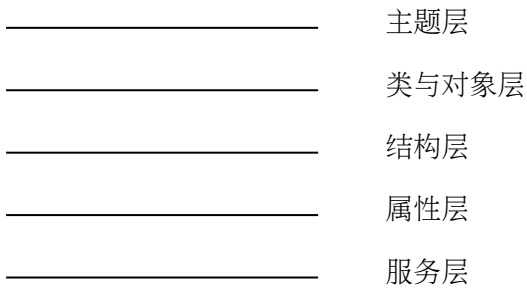


图 10.1 复杂问题的对象模型的5个层次

这5个层次很像叠在一起的5张透明塑料片，它们一层比一层显现出对象模型的更多细节。在概念上，这5个层次是整个模型的5张水平切片。

在本书第9章中已经讲述了类与对象(即UML的“类”)、结构(即类或对象之间的关系)、属性和服务的概念，现在再简要地介绍一下主题(或范畴)的概念。主题是指导读者(包括系统分析员、软件设计人员、领域专家、管理人员、用户等，总之，”读者”泛指所有需要读懂系统模型的人)理解大型、复杂模型的一种机制。也就是说，通过划分主题把一个大型、复杂的对象模型分解成几个不同的概念范畴。心理研究表明，人类的短期记忆能力一般限于一次记忆5～9个对象，这就是著名的7±2原则。面向对象分析从下述两个方面来体现这条原则：控

制可见性和指导读者的注意力。

首先，面向对象分析通过控制读者能见到的层次数目来控制可见性。其次，面向对象分析增加了一个主题层，它可以从一个相当高的层次描述总体模型，并对读者的注意力加以指导。

上述5个层次对应着在面向对象分析过程中建立对象模型的5项主要活动：找出类与对象，识别结构，识别主题，定义属性，定义服务。必须强调指出的是，我们说的是。”5项

活动”，而没有说5个步骤，事实上，这5项工作完全没有必要顺序完成，也无须彻底完成一项工作以后再开始另外一项工作。虽然这5项活动的抽象层次不同，但是在进行面向对象分析时并不需要严格遵守自顶向下的原则。人们往往喜欢先在一个较高的抽象层次上工作，如果在思考过程中突然想到一个具体事物，就会把注意力转移到深入分析发掘这个具体领域，然后又返回到原先所在的较高的抽象层次。例如，分析员找出一个类与对象，想到在这个类中应该包含的一个服务，于是把这个服务的名字写在服务层，然后又返回到类与对象层，继续寻找问题域中的另一个类与对象。

通常在完整地定义每个类中的服务之前，需要先建立起动态模型和功能模型，通过对这两种模型的研究，能够更正确更合理地确定每个类应该提供哪些服务。

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：寻找类与对象，识别结构，识别主题，定义属性，建立动态模型，建立功能模型，定义服务。但是，正如前面已经多次强调指出过的，分析不可能严格地按照预定顺序进行，大型、复杂系统的模型需要反复构造多遍才能建成。通常，先构造出模型的子集，然后再逐渐扩充，直到完全、充分地理解了整个问题，才能最终把模型建立起来。

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，以便澄清二义性，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。

10.2 需求陈述

10.2.1 书写要点

通常，需求陈述的内容包括：问题范围，功能需求，性能需求，应用环境及假设条件等。

总之，需求陈述应该阐明“做什么”而不是“怎样做”。它应该描述用户的需求而不是提出解决问题的方法。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。

书写需求陈述时，要尽力做到语法正确，而且应该慎重选用名词、动词、形容词和同义词。

不少用户书写的需求陈述，都把实际需求和设计决策混为一谈。系统分析员必须把需求与实现策略区分开，后者是一类伪需求，分析员至少应该认识到它们不是问题域的本质性质。

需求陈述可简可繁。对人们熟悉的传统问题的陈述，可能相当详细，相反，对陌生领域项目的需求，开始时可能写不出具体细节。

绝大多数需求陈述都是有二义性的、不完整的、甚至不一致的。某些需求有明显错误，还有一些需求虽然表述得很准确，但它们对系统行为存在不良影响或者实现起来造价太高。另外一些需求初看起来很合理，但却并没有真正反映用户的需要。应该看到，需求陈述仅仅是理解用户需求的出发点，它并不是一成不变的文档。不能指望没有经过全面、深入分析的需求陈述是完整、准确、有效的。随后进行的面向对象分析的目的，就是全面深入地理解问题域和用户的真实需求，建立起问题域的精确模型。

系统分析员必须与用户及领域专家密切配合协同工作，共同提炼和整理用户需求。在这个过程中，很可能需要快速建立起原型系统，以便与用户更有效地交流。

10.2.2 例子

图 10.2 所示的自动取款机(ATM)系统，是本书讲述面向对象分析和面向对象设计时使用的一个实例。

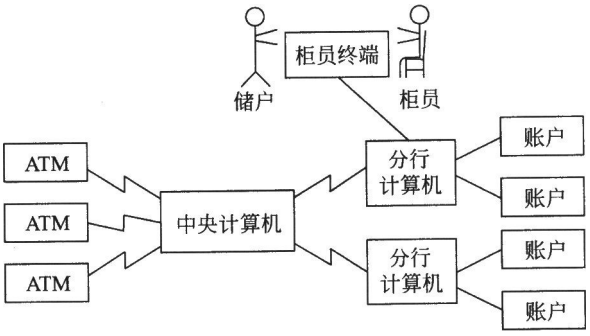


图 10.2 ATM 系统

下面陈述对 ATM 系统的需求:

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、分行计算机及柜员终端组成的网络系统。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柜员终端。柜员终端设在分行营业厅及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分摊。

银行柜员使用柜员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。柜员负责把储户提交的存款或取款事务输进柜员终端，接收储户交来的现金或支票，或付给储户现金。柜员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。使用现金兑换卡可以通过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。分行代码惟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从现金出口吐出现金，并且打印出账单交给用户。

10.3 建立对象模型

面向对象分析首要的工作，是建立问题域的对象模型。这个模型描述了现实世界中的“类与对象”以及它们之间的关系，表示了目标系统的静态数据结构。静态数据结构对应用细节依赖较少，比较容易确定；当用户的需求变化时，静态数据结构相对来说比较稳定。因此，用面向对象方法开发绝大多数软件时，都首先建立对象模型，然后再建立另外两个子模型。

需求陈述、应用领域的专业知识以及关于客观世界的常识，是建立对象模型时的主要信息来源。

如前所述，对象模型通常有5个层次。典型的工作步骤是：首先确定对象类和关联(因为它们影响系统整体结构和解决问题的方法)，对于大型复杂问题还要进一步划分出若干个主题；然后给类和关联增添属性，以进一步描述它们；接下来利用适当的继承关系进一步合并和组织类。而对类中操作的最后确定，则需等到建立了动态模型和功能模型之后，因为这两个子模型更准确地描述了对类中提供的服务的需求。

应该再一次强调指出的是，人认识客观世界的过程是一个渐进过程，是在继承前人知识的基础上，经反复迭代而不断深化的。因此，面向对象分析不可能严格按照顺序线性进行。初始的分析模型通常都是不准确不完整甚至包含错误的，必须在随后的反复分析中加以扩充和更正。此外，在面向对象分析的每一步，

都应该仔细分析研究以前针对相同的或类似的问题域进行面向对象分析所得到的结果，并尽可能在本项目中重用这些结果。以后在讲述面向对象分析的具体过程时，对上述内容将不再赘述。

10.3.1 确定类与对象

类与对象是在问题域中客观存在的，系统分析员的主要任务就是通过分析找出这些类与对象。首先找出所有候选的类与对象，然后从候选的类与对象中筛选掉不正确的或不必要的。

1.找出候选的类与对象

对象是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。具体地说，大多数客观事物可分为下述5类：

(1)可感知的物理实体，例如，飞机、汽车、书、房屋等等。

(2)人或组织的角色，例如，医生、教师、雇主、雇员、计算机系、财务处等等。

(3)应该记忆的事件，例如，飞行、演出、访问、交通事故等等。

(4)两个或多个对象的相互作用，通常具有交易或接触的性质，例如，购买、纳税、结
婚等等。

(5)需要说明的概念，例如，政策、保险政策、版权法等等。

在分析所面临的问题时，可以参照上列5类常见事物，找出在当前问题域中的候选类与对象。

另一种更简单的分析方法，是所谓的非正式分析。这种分析方法以用自然语言书写的需求陈述为依据，把陈述中的名词作为类与对象的候选者，用形容词作为确定属性的线索，把动词作为服务(操作)的候选者。当然，用这种简单方法确定的候选者是非常不准确的，其中往往包含大量不正确的或不必要的事物，还必须经过更进一步的严格筛选。通常，非正式分析是更详细、更精确的正式的面向对象分析的一个很好的开端。

下面以ATM系统为例，说明非正式分析过程。认真阅读10.2.2节给出的需求陈述，从陈述中找出下列名词，可以把它们作为类与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，柜员终端，网络，总行，分行，软件，成本，市，街道，营业厅，储蓄所，柜员，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，访问。

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与对象，因此，分析员应该根据领域知识或常识进一步把隐含的类与对象提取出来。例如，在ATM系统的需求陈述中虽然没写“通信链路”和“事务日志”，但是，根据领域知识和常识可以知道，在ATM系统中应该包含这两个实体。

2.筛选出正确的类与对象

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。非正式分析仅仅帮助我们找到一些候选的类与对象，接下来应该严格考察每个候选对象，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。

筛选时主要依据下列标准，删除不正确或不必要的类与对象：

(1)冗余

如果两个类表达了同样的信息，则应该保留在此问题域中最富于描述力的名称。

以ATM系统为例，上面用非正式分析法得出了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两类信息，因此，应该去掉“用户”、“磁卡”、“副本”等冗余的类，仅保留“储户”和“现金兑换卡”这两个类。

(2)无关

现实世界中存在许多对象，不能把它们都纳入到系统中去，仅需要把与本问题密切相关的类与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。

以ATM系统为例，这个系统并不处理分摊软件开发成本的问题，而且ATM和柜员终端放置的地点与本软件的关系也不大。因此，应该去掉候选类“成本”、“市”、“街道”、“营业厅”和“储蓄所”。

(3)笼统

在需求陈述中常常使用一些笼统的、泛指的名词，虽然在初步分析时把它们作为候选的类与对象列出来了，但是，要么系统无须记忆有关它们的信息，要么在需求陈述中有更明确更具体的名词对应它们所暗示的事务，因此，通常把这些笼统的或模糊的类去掉。

以ATM系统为例，“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。总之，在本例中应该去掉“银行”、“网络”、“系统”、“软件”、“信息”、“访问”等候选类。

(4)属性

在需求陈述中有些名词实际上描述的是其他对象的属性，应该把这些名词从候选类与对象中去掉。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。

在ATM系统的例子中，“现金”、“支票”、“取款额”、“账单”、“余额”、“分行代码”、“卡号”、“密码”、“类型”等，实际上都应该作为属性对待。

(5)操作

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，应该慎重考虑它们在本问题中的含义，以便正确地决定把它们作为类还是作为类中定义的操作。

例如，谈到电话时通常把“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，“拨号”需要有自己的属性(例
如日期、时间、受话地点等)，因此应该把它作为一个类。总之，本身具有属性需独立存在的操作，应该作为类与对象。

(6)实现

在分析阶段不应该过早地考虑怎样实现目标系统。因此，应该去掉仅和实现有关的候选的类与对象。在设计和实现阶段，这些类与对象可能是重要的，但在分析阶段过早地考虑它们反而会分散我们的注意力。

在ATM系统的例子中，“事务日志”无非是对一系列事务的记录，它的确切表示方式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类的物理实现。总之，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ATM系统的例子中，经过初步筛选，剩下下列类与对象：ATM、中央计算机、分行计算机、柜员终端、总行、分行、柜员、储户、账户、事务、现金兑换卡。

10.3.2 确定关联

多数人习惯于在初步分析确定了问题域中的类与对象之后，接下来就分析确定类与对象之间存在的关联关系。当然，这样的工作顺序并不是绝对必要的。由于在整个开发过程中面向对象概念和表示符号的一致性，分析员在选取自己习惯的工作方式时拥有相当大的灵活性。

如前所述，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与对象。

在分析确定关联的过程中，不必花过多的精力去区分关联和聚集。事实上，聚集不过是一种特殊的关联，是关联的一个特例。

1.初步确定关联

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，分析员还应该与用户及领域专家讨论问题域实体间的相互依赖、相互作用关系，根据领域知识再进一步补充一些关联。

以ATM系统为例，经过分析初步确定出下列关联：

(1)直接提取动词短语得出的关联

·ATM、中央计算机、分行计算机及柜员终端组成网络。

·总行拥有多台ATM。

·ATM设在主要街道上。

·分行提供分行计算机和柜员终端。

·柜员终端设在分行营业厅及储蓄所内。

·分行分摊软件开发成本。

·储户拥有账户。

·分行计算机处理针对账户的事务。

·分行计算机维护账户。

·柜员终端与分行计算机通信。

·柜员输入针对账户的事务。

·ATM与中央计算机交换关于事务的信息。

·中央计算机确定事务与分行的对应关系。

·ATM读现金兑换卡。

·ATM与用户交互。

·ATM吐出现金。

·ATM 打印账单。
·系统处理并发的访问。
(2)需求陈述中隐含的关联
·总行由各个分行组成。
·分行保管账户。
·总行拥有中央计算机。
·系统维护事务日志。
·系统提供必要的安全性。
·储户拥有现金兑换卡。
(3)根据问题域知识得出的关联
·现金兑换卡访问账户。
·分行雇用柜员。
2.筛选
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)已删去的类之间的关联
如果在分析确定类与对象的过程中已经删掉了某个候选类，则与这个类有关的关联也应该删去，或用其他类重新表达这个关联。
以 ATM 系统为例，由于已经删去了“系统”、“网络”、“市”、“街道”、“成本”、“软件”、“事务日志”、“现金”、“营业厅”、“储蓄所”、“账单”等候选类，因此，与这些类有关的下列 8 个关联也应该删去:
① ATM、中央计算机、分行计算机及柜员终端组成网络。
② ATM 设在主要街道上。
③分行分摊软件开发成本。
④系统提供必要的安全性。
⑤系统维护事务日志。
⑥ ATM 吐出现金。
⑦ ATM 打印账单。
⑧柜员终端设在分行营业厅及储蓄所内。
(2)与问题无关的或应在实现阶段考虑的关联
应该把处在本问题域之外的关联或与实现密切相关的关联删去。
例如，在 ATM 系统的例子中，“系统处理并发的访问”并没有标明对象之间的新关联，它只不过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。

(3)瞬时事件

关联应该描述问题域的静态结构，而不应该是一个瞬时事件。
以 ATM 系统为例，“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。
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。例如，在 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。
(4)三元关联

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

在 ATM 系统的例子中，“柜员输入针对账户的事务”可以分解成“柜员输人事务”和“事务修改账户”这样两个二元关联。而“分行计算机处理针对账户的事务”也可以做类似的分解。“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 上输人事务”这两个二元关联。

(5)派生关联

应该去掉那些可以用其他关联定义的冗余关联。

例如，在 ATM 系统的例子中，“总行拥有多台 ATM”实质上是“总行拥有中央计算机”和"ATM 与中央计算机通信”这两个关联组合的结果。而“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”。

3.进一步完善

应该进一步完善经筛选后余下的关联，通常从下述几个方面进行改进:

(1)正名

好的名字是帮助读者理解的关键因素之一。因此，应该仔细选择含义更明确的名字作为关联名。

例如，“分行提供分行计算机和柜员终端”不如改为“分行拥有分行计算机”和“分行拥有柜员终端”。

(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与对象。

例如，在 ATM 系统中，应该把“事务”分解成“远程事务”和“柜员事务”。

(3)补充

发现了遗漏的关联就应该及时补上。

例如，在 ATM 系统中把“事务”分解成上述两类之后，需要补充“柜员输入柜员事务”、“柜员事务输进柜员终端”、“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。

(4)标明重数

应该初步判定各个关联的类型，并粗略地确定关联的重数。但是，无须为此花费过多精力，因为在分析过程中随着认识的逐渐深入，重数也会经常改动。

图 10.3 是经上述分析过程之后得出的 ATM 系统原始的类图。

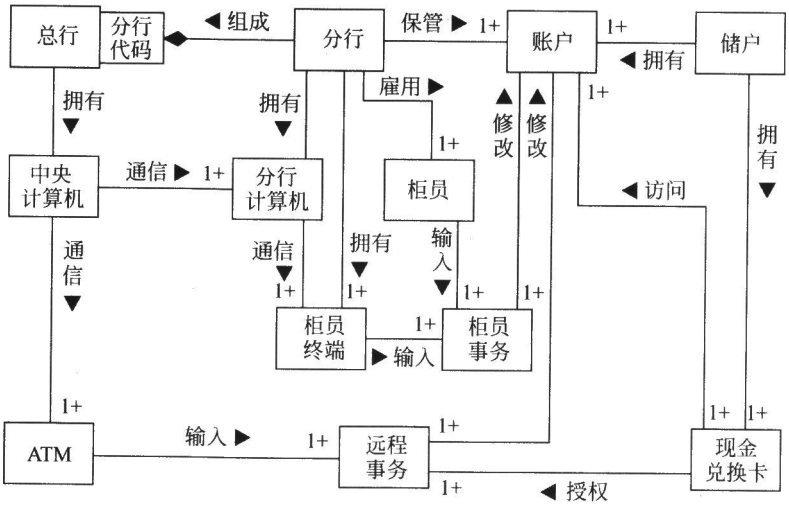


图 10.3 ATM 系统原始的类图

10.3.3 划分主题

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。

在开发很小的系统时，可能根本无须引入主题层；对于含有较多对象的系统，则往往先识别出类与对象和关联，然后划分主题，并用它作为指导开发者和用户观察整个模型的一种机制；对于规模极大的系统，则首先由高级分析员粗略地识别对象和关联，然后初步划分主题，经进一步分析，对系统结构有更深入的了解之后，再进一步修改和精炼主题。

应该按问题领域而不是用功能分解方法来确定主题。此外，应该按照使不同主题内的对象相互间依赖和交互最少的原则来确定主题。

以 ATM 系统为例，可以把它划分成总行(包含总行和中央计算机这两个类)、分行(包含分行、分行计算机、柜员终端、柜员事务、柜员和账户等类)和 ATM(包含 ATM、远程事务、现金兑换卡和储户等类)等 3 个主题。事实上，我们描述的是一个简化的 ATM 系统，为了简单起见，在下面讨论这个例子时将忽略主题层。

10.3.4 确定属性

属性是对象的性质，藉助于属性我们能对类与对象和结构有更深入更具体的认识。注意，在分析阶段不要用属性来表示对象间的关系，使用关联能够表示两个对象间的任何关系，而且把关系表示得更清晰、更醒目。

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

通常，在需求陈述中用名词词组表示属性，例如，“汽车的颜色”或“光标的位置”。往往用形容词表示可枚举的具体属性，例如，“红色的”、“打开的”。但是，不可能在需求陈述中找到所有属性，分析员还必须藉助于领

域知识和常识才能分析得出需要的属性。幸运
的是，属性对问题域的基本结构影响很小。随着时间的推移，问题域中的类始终保持稳定，属性却可能改变了，相应地，类中方法的复杂程度也将改变。

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

认真考察经初步分析而确定下来的那些属性，从中删掉不正确的或不必要的属性。通常有以下几种常见情况：

(1)误把对象当作属性

如果某个实体的独立存在比它的值更重要，则应把它作为一个对象而不是对象的属性。在具体应用领域中具有自身性质的实体，必然是对象。同一个实体在不同应用领域中，到底应该作为对象还是属性，需要具体分析才能确定。例如，在邮政目录中，“城市”是一个属性，而在人口普查中却应该把“城市”当作对象。

(2)误把关联类的属性当作一般对象的属性

如果某个性质依赖于某个关联链的存在，则该性质是关联类的属性，在分析阶段不应该把它作为一般对象的属性。特别是在多对多关联中，关联类属性很明显，即使在以后的开发阶段中，也不能把它归并成相互关联的两个对象中任一个的属性。

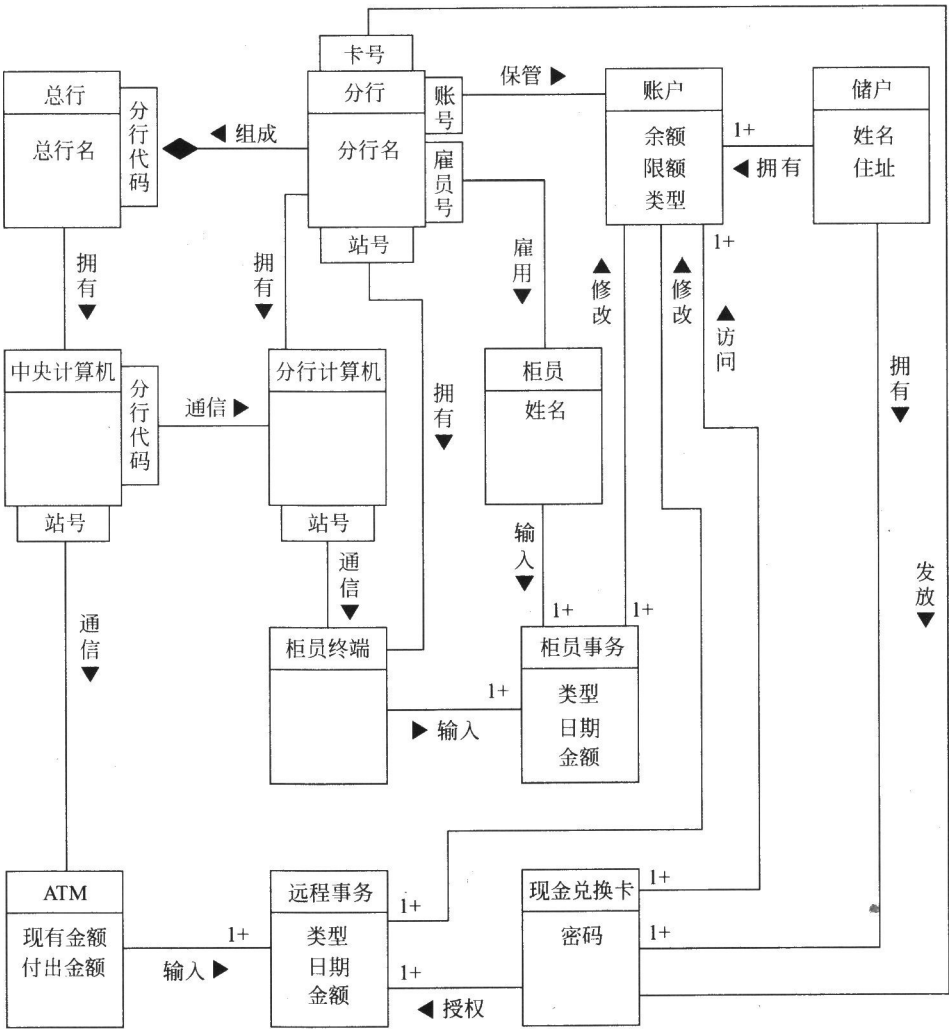


图 10.4 ATM 系统对象模型中的属

(3)把限定误当成属性

正如 9.4.2 节所述，正确使用限定词往往可以减少关联的重数。如果把某个属性值固定下来以后能减少关联的重数，则应该考虑把这个属性重新表述成一个限定词。在 ATM 系统的例子中，“分行代码”、“账号”、“雇员号”、“站号”等都是限定词。

(4)误把内部状态当成了属性

如果某个性质是对象的非公开的内部状态，则应该从对象模型中删掉这个属性。

(5)过于细化

在分析阶段应该忽略那些对大多数操作都没有影响的属性。

(6)存在不一致的属性

类应该是简单而且一致的。如果得出一些看起来与其他属性毫不相关的属性，则应该考虑把该类分解成两个不同的类。

经过筛选之后，得到 ATM 系统中各个类的属性，如图 10.4 所示。图中还标出了一些限定词：

·“卡号”实际上是一个限定词。在研究卡号含义的过程中，发现以前在分析确联的过程中遗漏了“分行发放现金兑换卡”这个关联，现在把这个关联补上，是这个关联上的限定词。

·“分行代码”是关联“分行组成总行”上的限定词。

·“账号”是关联“分行保管账户”上的限定词。

·“雇员号”是“分行雇用柜员”上的限定词。

·“站号”是“分行拥有柜员终端”、“柜员终端与分行计算机通信”及“中央计算ATM通信”等三个关联上的限定词。

应该说明的是，我们讨论的ATM系统是一个经过简化后的例子，而不是一个完实际应用系统。因此，图10.4中示出的属性远较实际应用系统中的属性少。

10.3.5 识别继承关系

确定了类中应该定义的属性之后，就可以利用继承机制共享公共性质，并对系统中众多的类加以组织。正如以前曾经强调指出过的，继承关系的建立实质上是知识抽取过程，它应该反映出一定深度的领域知识，因此必须有领域专家密切配合才能完成。通常，许多归纳关系都是根据客观世界现有的分类模式建立起来的，只要可能，就应该使用现有的概念。

一般说来，可以使用两种方式建立继承(即泛化)关系：

(1)自底向上：抽象出现有类的共同性质泛化出父类，这个过程实质上模拟了人类归纳思维过程。例如，在ATM系统中，“远程事务”和“柜员事务”是类似的，可以泛化出父类“事务”；类似地，可以从"ATM”和“柜员终端”泛化出父类“输入站”。

(2)自顶向下：把现有类细化成更具体的子类，这模拟了人类的演绎思维过程。从应用域中常常能明显看出应该做的自顶向下的具体化工作。例如，带有形容词修饰的名词词组往往暗示了一些具体类。但是，在分析阶段应该避免过度细化。

利用多重继承可以提高共享程度，但是同时也增加了概念上以及实现时的复杂程度。使用多重继承机制时，通常应该指定一个主要父类，从它继承大部分属性和行为；次要父类只补充一些属性和行为。

图10.5是增加了继承关系之后的ATM对象模型。

10.3.6 反复修改

仅仅经过一次建模过程很难得到完全正确的对象模型。事实上，软件开发过程就是一个多次反复修改、逐步完善的过程。在建模的任何一个步骤中，如果发现了模型的缺陷，都必须返回到前期阶段进行修改。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，因此远比使用结构分析、设计技术更容易实现反复修改、逐步完善的过程。

实际上，有些细化工作(例如，定义服务)是在建立了动态模型和功能模

型之后才进行的。

在实际工作中，建模的步骤并不一定严格按照前面讲述的次序进行。分析员可以合并几个步骤的工作放在一起完成，也可以按照自己的习惯交换前述各项工作的次序，还可以先初步完成几项工作，再返回来加以完善。但是，如果你是初次接触面向对象方法，则最好先按本书所述次序，尝试用面向对象方法，开发几个较小的系统，取得一些实践经验后，再总结出更适合自己的工作方式。

下面以ATM系统为例，讨论可能做的修改：

1.分解“现金兑换卡”类

实际上，“现金兑换卡”有两个相对独立的功能，它既是鉴别储户使用ATM的权限的卡，又是ATM获得分行代码和卡号等数据的数据载体。因此，把“现金兑换卡”类分解为“卡权限”和“现金兑换卡”两个类，将使每个类的功能更单一：前一个类标志储户访问账户的权限，后一个类是含有分行代码和卡号的数据载体。多张现金兑换卡可能对应着相同的访问权限。

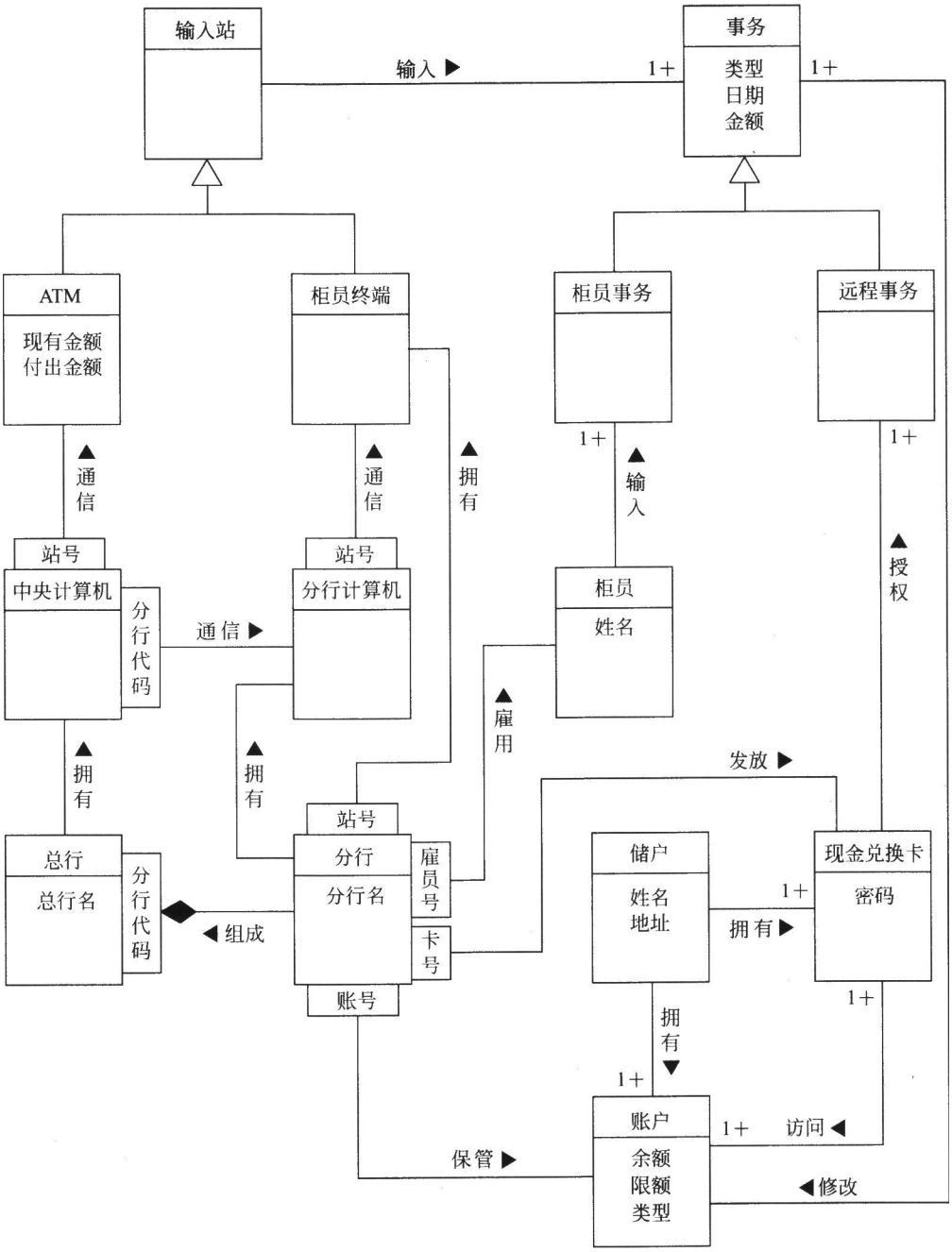


图 10.5 带有继承关系的 ATM 对象模型

2. 通常，一个事务包含对账户的若干次更新，这里所说的更新，指的是对账户所做的一个动作(取款、存款或查询)。“更新”虽然代表一个动作，但是它有自己的属性(类型、金额

等)，应该独立存在，因此应该把它作为类。

3.把“分行”与“分行计算机”合并

区分“分行”与“分行计算机”，对于分析这个系统来说，并没有多大意义，为简单起见应该把它们合并。类似地，应该合并“总行”和“中央计算机”。

图 10.6 给出了修改后的 ATM 对象模型，与修改前比较起来，它更简单、更清晰。

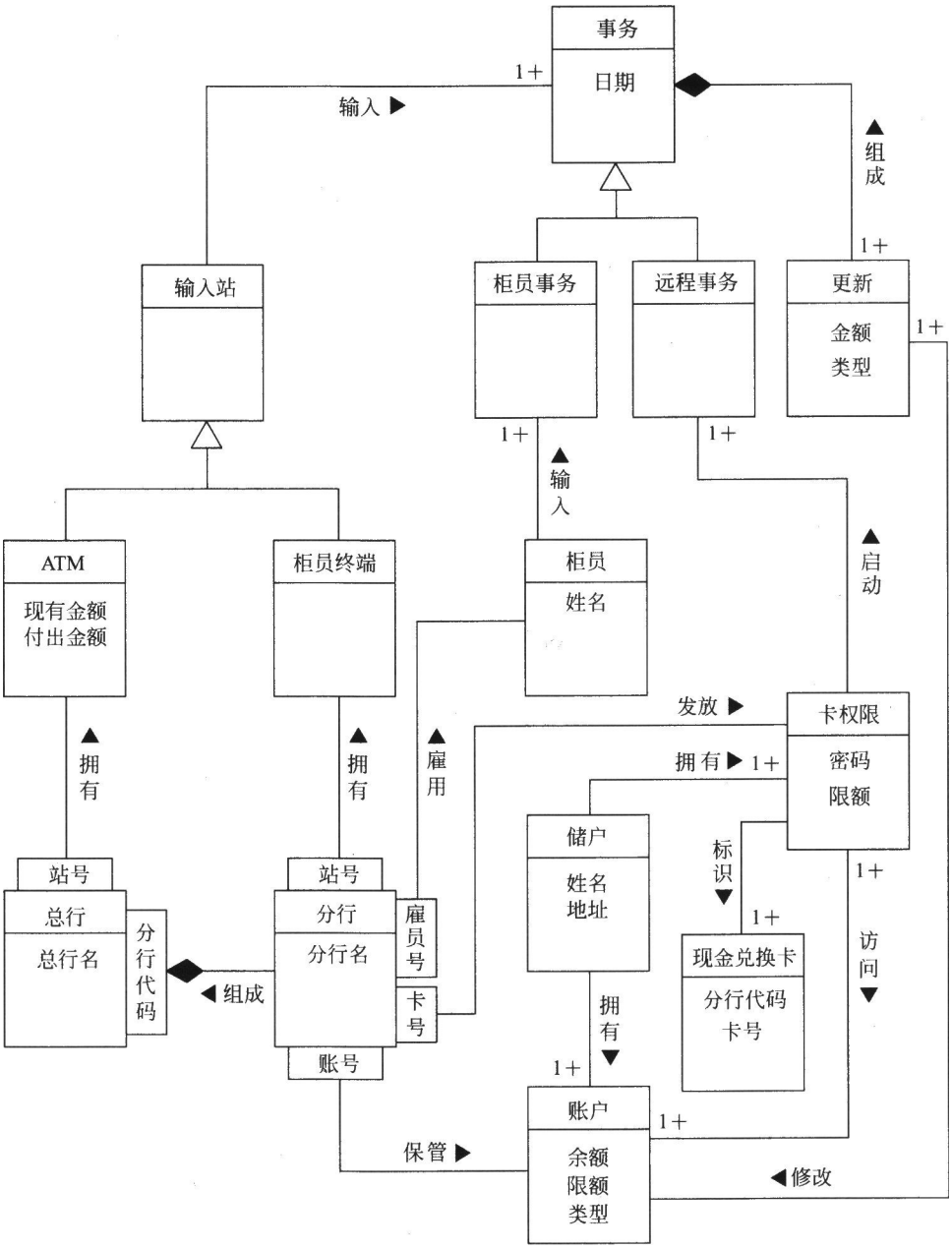


图 10.6 修改后的 ATM 对象模型

10.4 建立动态模型

本书 9.5 节和 3.6 节已经介绍了动态模型的概念和表示方法，本节结合 ATM 系统的实例，进一步讲述建立动态模型的方法。

对于仅存储静态数据的系统(例如数据库)来说，动态模型并没有什么意义。然而在开发交互式系统时，动态模型却起着很重要的作用。如果收集输入信息是目标系统的一项主要工作，则在开发这类应用系统时建立正确的动态模型是至关重要的。

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。第二步，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，并用状态图描绘它们。最后，比较各个对象的状态图，检查它们之间的一致性，确保事件之间的匹配。

10.4.1 编写脚本

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，脚本是指系统在某一执行期间内出现的一系列事件。脚本描述用户(或其他外部设备)与目标系统之间的一个或多个典型的交互过程，以便对目标系统的行为有更具体的认识。编写脚本的目的，是保证不遗漏重要的交互步骤，它有助于确保整个交互过程的正确性的和清晰性。

脚本描写的范围并不是固定的，既可以包括系统中发生的全部事件，也可以只包括由某些特定对象触发的事件。脚本描写的范围主要由编写脚本的具体目的决定。

即使在需求陈述中已经描写了完整的交互过程，也还需要花很大精力构思交互的形式。例如，ATM 系统的需求陈述，虽然表明了应从储户那里获取有关事务的信息，但并没有准确说明获取信息的具体过程，对动作次序的要求也是模糊的。因此，编写脚本的过程，实质上就是分析用户对系统交互行为的要求的过程。在编写脚本的过程中，需要与用户充分交换意见，编写后还应该经过他们审查与修改。

编写脚本时，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，考虑出错情况，例如，输入的值为非法值或响应失败。对大多数交互式系统来说，出错处理都是最难实现的部分。如果可能，应该允许用户“异常中止”一个操作或“取消”一个操作。此外，还应该提供诸如“帮助”和状态查询之类的在基本交互行为之上的“通用”交互行为。

脚本描述事件序列。每当系统中的对象与用户(或其他外部设备)交换信息时，就发生一个事件。所交换的信息值就是该事件的参数(例如，“输入密码”事件的参数是所输入的密码)。也有许多事件是无参数的，这样的事件仅传递一个信息——该事件

表 10.1 ATM 系统的正常情况脚本

对于每个事件，[illegible]，系统、用户或

·ATM 请储户插卡；储户插入一张现金兑换卡。
·ATM 接受该卡并读它上面的分行代码和卡号。
·ATM 要求储户输入密码；储户输入自己的密码“1234"等数字。
·ATM 请求总行验证卡号和密码；总行要求“39”号分行核对储户密码，然后通知 ATM 说这张卡有效。
·ATM 要求储户选择事务类型（取款、转账、查询等）；储户选择“取款”。
·ATM 要求储户输人取款额；储户输入“880”。
·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。
·ATM 吐出现金并请储户拿走这些现金；储户拿走现金。
·ATM 问储户是否继续这项事务；储户回答“不”。
·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。
·ATM 请储户插卡。

表 10.2 ATM 系统的异常情况脚本

·ATM 请储户插卡；储户插入一张现金兑换卡。 ·ATM 接受这张卡并顺序读它上面的数字。 ·ATM 要求密码；储户误输入“ 8888” 。 ·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。 ·ATM 显示“密码错”，并请储户重新输入密码；储户输入 "1234” ； ATM 请总行验证后知道这次输入的密码正确。 ·ATM 请储户选择事务类型；储户选择“取款” 。 ·ATM 询问取款额；储户改变主意不想取款了，他敲“取消”键。 ·ATM 退出现金兑换卡，并请储户拿走它；储户拿走他的卡。 ·ATM 请储户插卡。

用系统的控制逻辑。

但是，用户界面的美观程度、方便程度、易学程度以及效率等等，是用户使用系统时[illegible]来，用户界面的好坏[illegible]因此，在分析阶段也[illegible]重要，重要的是在这[illegible]必要的信息交换，而[illegible]

不[illegible]发人员往往快速地建[illegible]设想出的 ATM 界面格[illegible]

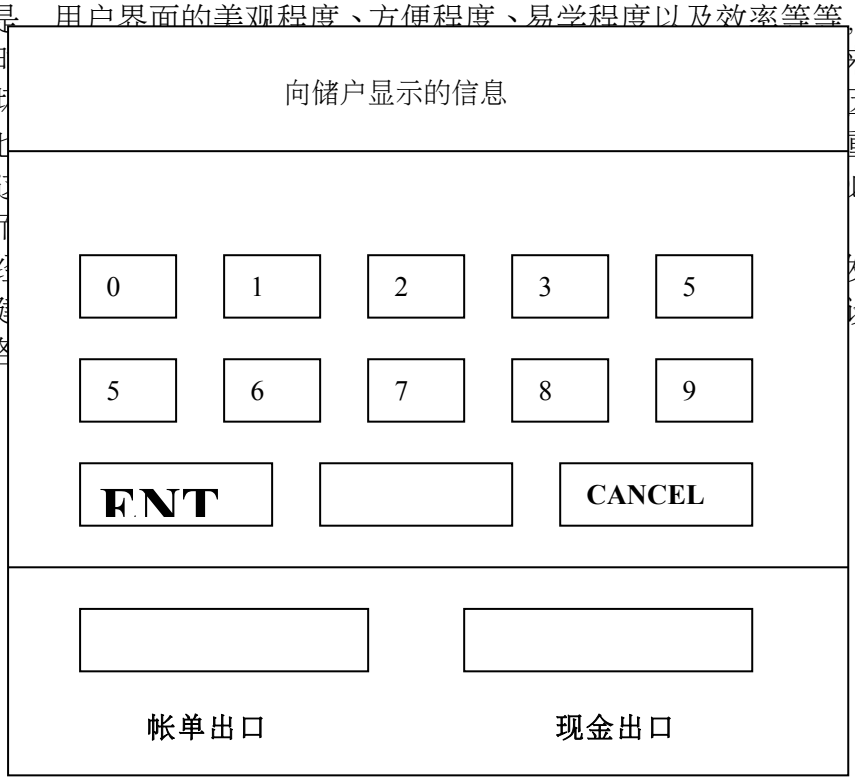


图 10.7 ATM 的界面格式

10.4.3 画事件跟踪图

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。

1.确定事件

应该仔细分析每个脚本，以便从中提取出所有外部事件。事件包括系统与用户(或外部设备)交互的所有信号、输入、输出、中断、动作等等。从脚本中容易找出正常事件，但是，应该小心仔细，不要遗漏了异常事件和出错条件。

传递信息的对象的动作也是事件。例如，储户插入现金兑换卡、储户输入密码、ATM 吐出现金等都是事件。大多数对象到对象的交互行为都对应着事件。

应该把对控制流产生相同效果的那些事件组合在一起作为一类事件，并给它们取一个惟一的名字。例如，“吐出现金”是一个事件类，尽管这类事件中的每个个别事件的参数值不同(吐出的现金数额不同)，然而这并不影响控制流。但是，应该把对控制流有不同影响的那些事件区分开来，不要误把它们组合在一起。例如“账户有效”、“账户无效”、“密码错”等都是不同的事件。一般说来，不同应用系统对相同事件的响应并不相同，因此，在最终分类所有事件之前，必须先画出状态图。如果从状态图中看出某些事件之间的差异对系统行为并没有影响，则可以忽略这些事件间的差异。

经过分析，应该区分出每类事件的发送对象和接受对象。一类事件相对它的发送对象来说是输出事件，但是相对它的接受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。

2.画出事件跟踪图

从脚本中提取出各类事件并确定了每类事件的发送对象和接受对象之后，就可以用事件跟踪图把事件序列以及事件与对象的关系，形象、清晰地表示出来。事件跟踪图实质上是扩充的脚本，可以认为事件跟踪图是简化的 UML 顺序图。

在事件跟踪图中，一条竖线代表一个对象，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受对象。时间从上向下递增，也就是说，画在最上面的水平箭头线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没有具体含义，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，并不表示两个事件之间的精确时间差。

图 10.8 是 ATM 系统正常情况下的事件跟踪图。

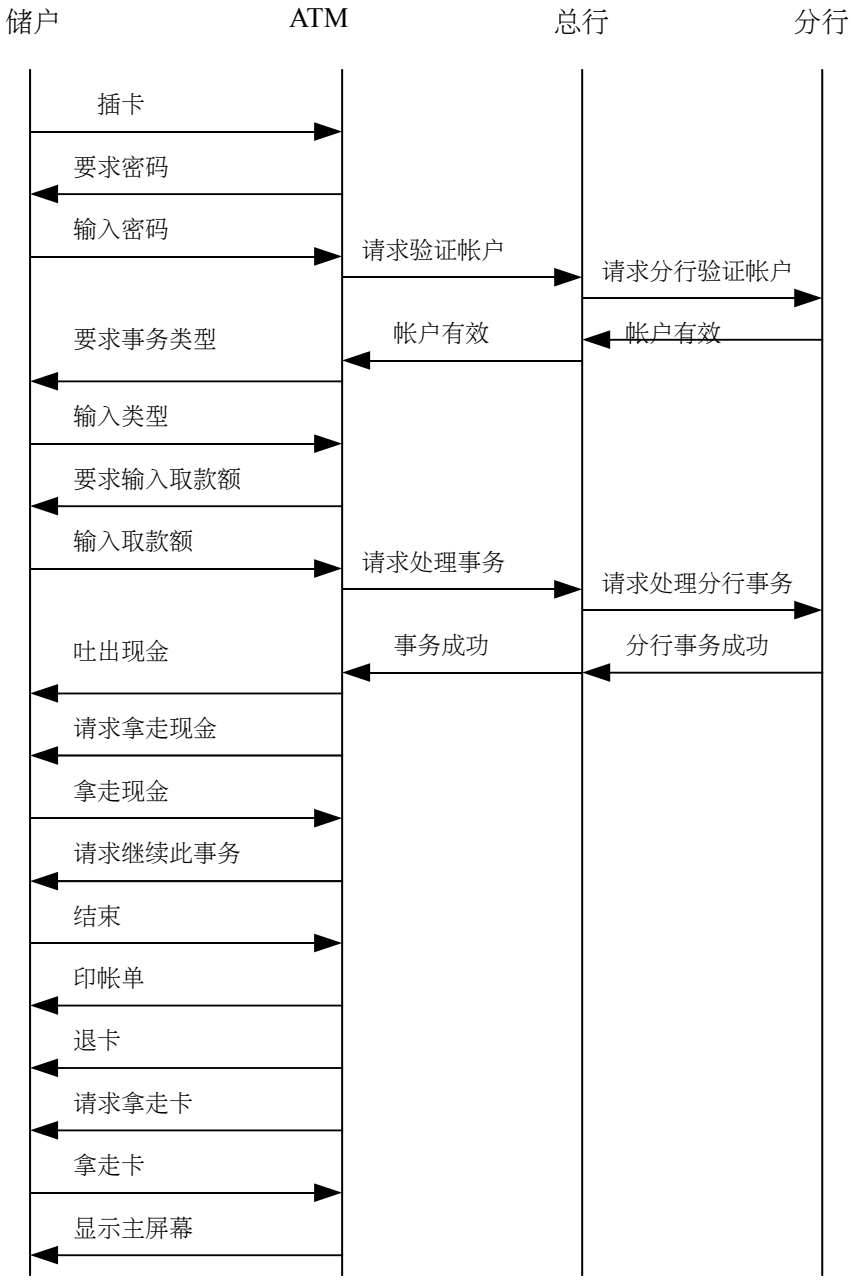


图 10.8 ATM 系统正常情况脚本的事件跟踪图

10.4.4 画状态图

状态图描绘事件与对象状态的关系。当对象接受了一个事件以后，它的下

个状态取决于当前状态及所接受的事件。由事件引起的状态改变称为“转换”。如果一个事件并不引起当前状态发生转换，则可忽略这个事件。

通常，用一张状态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，也不是任何一个类都需要有一张状态图描绘它的行为。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。

从一张事件跟踪图出发画状态图时，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)。

根据一张事件跟踪图画出状态图之后，再把其他脚本的事件跟踪图合并到已画出的状态图中。为此需在事件跟踪图中找出以前考虑过的脚本的分支点(例如“验证账户”就是一个分支点，因为验证的结果可能是“账户有效”，也可能是“无效账户”)，然后把其他脚本中的事件序列并入已有的状态图中，作为一条可选的路径。

考虑完正常事件之后再考虑边界情况和特殊情况，其中包括在不适当时候发生的事件(例如，系统正在处理某个事务时，用户要求取消该事务)。有时用户(或外部设备)不能做出快速响应，然而某些资源又必须及时收回，于是在一定间隔后就产生了“超时”事件。对用户出错情况往往需要花费很多精力处理，并且会使原来清晰、紧凑的程序结构变得复杂、繁琐，但是，出错处理是不能省略的。

当状态图覆盖了所有脚本，包含了影响某类对象状态的全部事件时，该类的状态图就构造出来了。利用这张状态图可能会发现一些遗漏的情况。测试完整性和出错处理能力的最好方法，是设想各种可能出现的情况，多问几个“如果……，则……”的问题。

以ATM系统为例。“ATM"、“柜员终端”、“总行”和“分行”都是主动对象，它们相互发送事件；而“现金兑换卡”、“事务”和“账户”是被动对象，并不发送事件。“储户”和“柜员”虽然也是动作对象，但是，它们都是系统外部的因素，无须在系统内实现它们。因此，只需要考虑"ATM”、“总行”、“柜员终端”和“分行”的状态图。

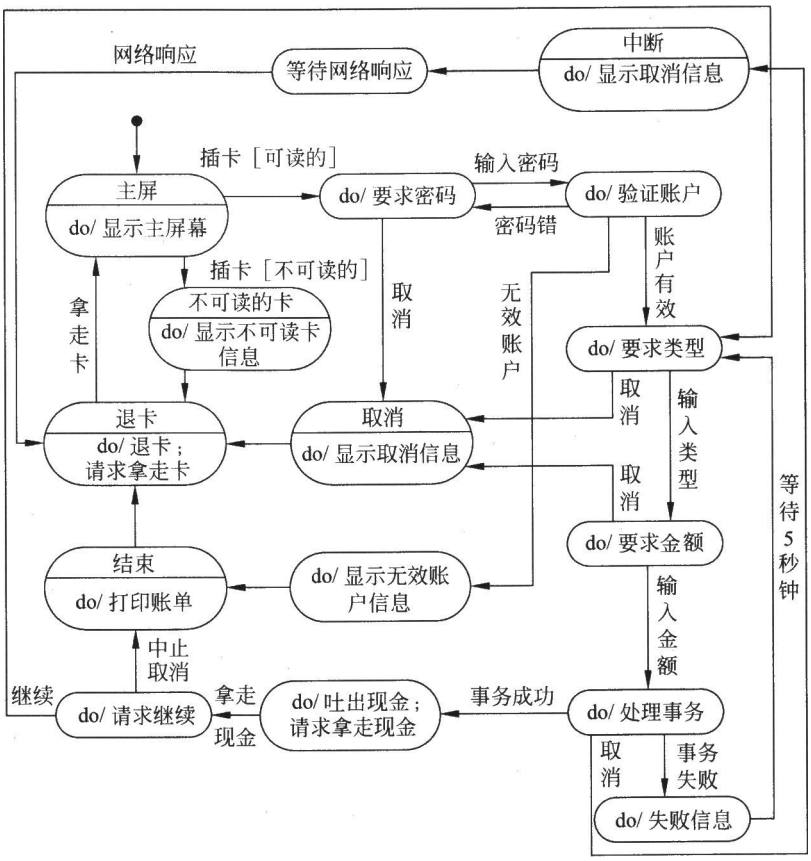
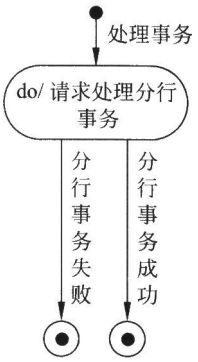


图 10.9　ATM 类的状态

图 10.9，图 10.10 和图 10.11 分别是"ATM”、“总行”和“分行”的状态图。由于“柜员终端”的状态图和"ATM"的状态图类似，为节省篇幅把它省略了。这些状态图都是简化的，尤其对异常情况和出错情况的考虑是相当粗略的(例如，图 10.9 并没有表示在网络通信链路不通时的系统行为，实际上，在这种情况下 ATM 停止处理储户事务)。



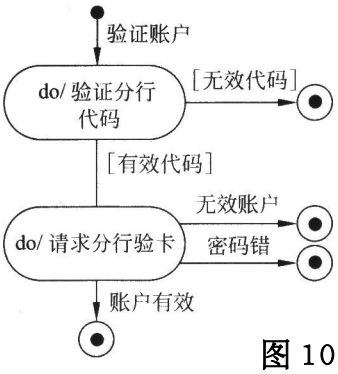


图 10.10 总行类的状态图

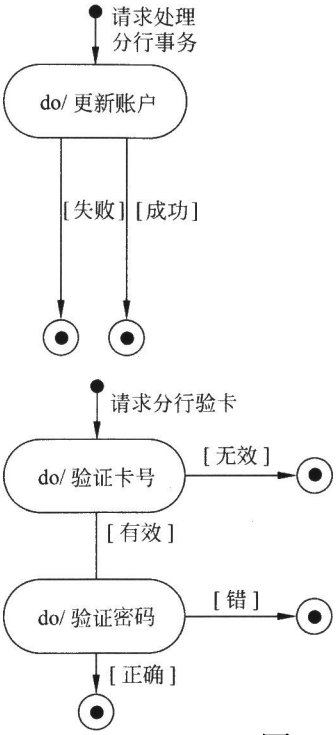


图 10.11 分行类的状态图

10.4.5 审查动态模型

各个类的状态图通过共享事件合并起来，构成了系统的动态模型。在完成了每个具有重要交互行为的类的状态图之后，应该检查系统级的完整性和一致性。一般说来，每个事件都应该既有发送对象又有接受对象，当然，有时发送者和接受者是同一个对象。对于没有前驱或没有后继的状态应该着重审查，如果这个状态既不是交互序列的起点也不是终点，则发现了一个错误。

应该审查每个事件，跟踪它对系统中各个对象所产生的效果，以保证它们与每个脚本都匹配。

以 ATM 系统为例。在总行类的状态图中，事件“分行代码错”是由总行发出的，但是在 ATM 类的状态图中并没有一个状态接受这个事件。因此，在 ATM 类的状态图中应该再补充一个状态"do／显示分行代码错信息”，它接受由前驱状态"do／验证账户”发出的事件“分行代码错”，它的后续状态是“退卡”。

10.5 建立功能模型

功能模型表明了系统中数据之间的依赖关系，以及有关的数据处理功能，它由一组数据流图组成。其中的处理功能可以用 IPO 图(或表)、伪码等多种方式进一步描述。

通常在建立了对象模型和动态模型之后再建立功能模型。

本书第 2 章已经详细讲述了画数据流图的方法，本节结合 ATM 系统的例子，再复习一遍有关数据流图的概念和画法。

10.5.1 画出基本系统模型图

基本系统模型由若干个数据源点／终点，及一个处理框组成，这个处理框代表了系统加工、变换数据的整体功能。基本系统模型指明了目标系统的边界。由数据源点输入的数据和输出到数据终点的数据，是系统与外部世界之间的交互事件的参数。

图 10.12 是 ATM 系统的基本系统模型。尽管在储蓄所内储户的事务是由柜员通过柜员终端提交给系统的，但是信息的来源和最终接受者都是储户，因此，本系统的数据源点／终点为储户。另一个数据源点是现金兑换卡，因为系统从它上面读取分行代码、卡号等信息。

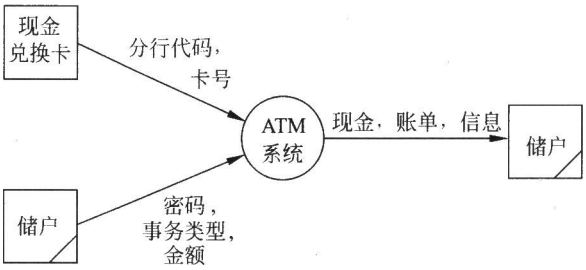


图 10.12 ATM 系统的基本系

10.5.2 画出功能级数据流图

把基本系统模型中单一的处理框分解成若干个处理框，以描述系统加工、变换数据的基本功能，就得到功能级数据流图。

ATM 系统的功能级数据流图如图 10.13 所示

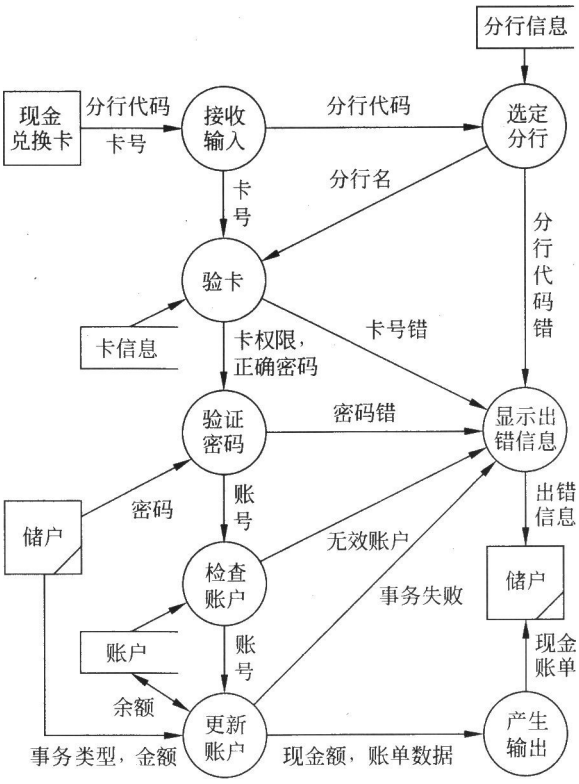


图 10.13　ATM 系统的功能级数据流图

10.5.3 描述处理框功能

把数据流图分解细化到一定程度之后，就应该描述图中各个处理框的功能。应该注意的是，要着重描述每个处理框所代表的功能，而不是实现功能的具体算法。

描述既可以是说明性的，也可以是过程性的。说明性描述规定了输入值和输出值之间的关系，以及输出值应遵循的规律。过程性描述则通过算法说明“做什么”。一般说来，说明性描述优于过程性描述，因为这类描述中通常不会隐含具体实现方面的考虑。

ATM 系统数据流图中大多数处理框的功能都比较简单。作为一个例子，表 10.3 给出了对“更新账户”这个处理功能的描述。

表　10.3　对更新帐户功能的描述

更新账户（账号，事务类型，金额）—现金额，账单数据，信息 如果取款额超过账户当前余额，拒绝该事务且不付出现金。 如果取款额不超过账户当前余额，从余额中减去取款额后作为新的余额，付出储户要取的现金。 如果事务是存款，把存款额加到余额中得到新余额，不付出现金。 如果事务是查询，不付出现金。 在上述任何一种情况下，账单内容都是：ATM 号，日期，时间，账号，事务类型，事务金额（如果有的话），新余额。

10.6 定义服务

正如10.3节指出的那样，“对象”是由描述其属性的数据，及可以对这些数据施加的
操作(即服务)，封装在一起构成的独立单元。因此，为建立完整的对象模型，既要确定类中应该定义的属性，又要确定类中应该定义的服务。然而在10.3节中已经指出，需要等到建立了动态模型和功能模型之后，才能最终确定类中应有的服务，因为这两个子模型更明确地描述了每个类中应该提供哪些服务。事实上，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。

1.常规行为

在分析阶段可以认为，类中定义的每个属性都是可以访问的，也就是说，假设在每个
类中都定义了读、写该类每个属性的操作。但是，通常无需在类图中显式表示这些常规操作。

2.从事件导出的操作

状态图中发往对象的事件也就是该对象接收到的消息，因此该对象必须有由消息选择符指定的操作，这个操作修改对象状态(即属性值)并启动相应的服务。例如，在ATM系统中，发往ATM对象的事件“中止”，启动该对象的服务“打印账单”；发往分行的事件“请分行验卡”启动该对象的服务“验证卡号”；而事件“处理分行事务”启动分行对象的服务“更新账户”。可以看出，所启动的这些服务通常就是接受事件的对象在相应状态的行为。

3.与数据流图中处理框对应的操作

数据流图中的每个处理框都与一个对象(也可能是若干个对象)上的操作相对应。应该仔细对照状态图和数据流图，以便更正确地确定对象应该提供的服务。例如，在ATM系统中，从状态图上看出分行对象应该提供“验证卡号”服务，而在数据流图上与之对应的处理框是“验卡”，根据实际应该完成的功能看，该对象提供的这个服务应该是“验卡”。

4.利用继承减少冗余操作

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，就尽量抽取出相似类的公共属性和操作，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。

10.7 小　结

分析就是提取系统需求并建立问题域精确模型的过程，它包括理解、表达和验证等3项主要工作内容。面向对象分析的关键工作，是分析、确定问题域中的对象及对象间的关系，并建立起问题域的对象模型。

大型、复杂系统的对象模型通常由下述5个层次组成：主题层、类与对象层、结构层、属性层和服务层。它们对应着在建立对象模型的过程中所应完成的5项工作。

大多数分析模型都不是一次完成的，为了理解问题域的全部含义，必须反复多次地进行分析。因此，分析工作不可能严格地按照预定顺序进行；分析工作也不是机械地把需求陈述转变为分析模型的过程。分析员必须与用户及领域专家反复交流、多次磋商，及时纠正错误认识并补充缺少的信息。

分析模型是同用户及领域专家交流时有效的通信手段。最终的模型必须得到用户和领域专家的确认。在交流和确认的过程中，原型往往能起很大的促进作用。

一个好的分析模型应该正确完整地反映问题的本质属性，且不包含与问题无关的内容。分析的目标是全面深入地理解问题域，其中不应该涉及具体实现的考虑。但是，在实际的分析过程中完全不受与实现有关的影响也是不现实的。虽然分析的目的是用分析模型取代需求陈述，并把分析模型作为设计的基础，但是事实上，在分析与设计之间并不存在绝对的界线。

习 题 10

1.用面向对象方法分析研究本书习题2第2题中描述的储蓄系统，试建立它的对象模型、动态模型和功能模型。

2.用面向对象方法分析研究本书习题2第3题中描述的机票预订系统，试建立它的对象模型、动态模型和功能模型。

3.用面向对象方法分析研究本书习题2第4题中描述的患者监护系统，试建立它的对象模型、动态模型和功能模型。

4.下面是自动售货机系统的需求陈述，请建立它的对象模型、动态模型和功能模型：

自动售货机系统是一种无人售货系统。售货时，顾客把硬币投入机器的投币口中，机器检查硬币的大小、重量、厚度及边缘类型。有效的硬币是一元币、五角币、一角币、五分币、二分币和一分币。其他货币都被认为是假币。机器拒绝接收假币，并将其从退币孔退出。当机器接收了有效的硬币之后，就把硬币送入硬币储藏器中。顾客支付的货币根据硬币的面值进行累加。

自动售货机装有货物分配器。每个货物分配器中包含零个或多个价格相同的货物。顾客通过选择货物分配器来选择货物。如果货物分配器中有货物，而且顾客支付的货币值不小于该货物的价格，货物将被分配到货物传送孔送给顾客，并将适当的零钱返回到退币孔。如果分配器是空的，则和顾客支付的货币值相等的硬币将被送回到退币孔。如果顾客支付的货币值少于所选择的分配器中货物的价格，机器将等待顾客投进更多的货币。如果顾客决定不买所选择的货物，他投放进的货币将从退币孔中退出。