软件系统的设计方法

按照面向对象的软件开发方法，软件系统的所有组成部件都被抽象为对象，所有数据信息和数据处理功能都被分解到不同的对象，所有对象通过消息传递相互配合，实现软件的功能。因此，管理各种数据信息的实体型对象和管理型对象，以及承担各种数据处理功能的加工型对象和事务型对象组成了系统的组成模块，连同对象之间的各种对象关联，形成了软件系统的静态模型。另一方面，软件系统的动态工作过程则表现为各种对象之间按照一定的计算逻辑和交互控制关系组织起来的消息传递序列。设计者根据应用需求中的人机交互过程、通信控制需求以及计算处理逻辑，来分析各种对象的交互控制过程，进而明确每个软件模块承担的任务，确认每个对象所必须提供的计算处理功能，从而完善软件系统模型。

本章将介绍面向对象的软件系统建模方法、软件模块和软件结构的设计方法。从应用系统的需求分析出发，介绍按照数据和功能来划分系统组成结构的方法，确认出表示系统组成的对象及其相互关系，形成系统的静态模型；介绍根据应用场景来分析系统工作中对象之间的交互控制过程，确认每个对象承担的任务，细化系统模型。随后，介绍基于对象接口的软件模块设计方法，以及软件结构的设计方法。

通过本章的学习，读者应该解决以下问题：

(1) 如何从应用需求中抽象出表示软件模块的对象？

(2) 如何从应用问题的结构和数据分析中确认对象之间的组成关系？

(3) 如何从系统工作过程中分析对象之间的交互关系？

(4) 如何利用 UML 语言来图形化地表示系统模型？

(5) 什么是软件模块的接口？

(6) 如何利用 C++ 语言来描述软件模块的接口？

(7) 如何处理软件需求中的复杂控制逻辑问题？

(8) 如何处理软件需求中的复杂数据集问题？

本章以下各节将分别介绍软件系统建模、软件模块设计和软件模块的实现技术。5.1 节介绍软件系统建模的基本方法，包括从应用需求中确认对象的方法、从应用场景中描述对象交互过程的方法，进而通过应用实例介绍一个应用软件系统建模过程。5.2 节讲述软件系统设计中基于对象的软件模块描述方法，软件模块接口在软件设计中的作用，及其基于 C++ 语言的接口设计与描述方法。5.3 节介绍软件结构设计中的复杂控制逻辑和复杂数据集的处理方法，以及一个软件模块的设计与实现。

5.1 软件系统的建模

面向对象方法将任何系统都抽象为一组对象及其基于消息传递的交互关系。对象及其交互关系就形成了系统模型。各种计算机软件系统本质上都是信息处理系统，需要管理和维护各种数据信息，并完成各种信息处理。在基于面向对象的程序设计中，各种数据信息和信息处理功能都被封装在对象之中，在整个系统功能中，有些功能是由特定的对象负责实现的，而其他功能则需要多种对象相互配合来实现。因此，在软件系统建模的过程中，首先需要进行基于对象的抽象，设置一组对象来封装数据信息和系统功能；其次需要根据数据信息和功能要求的组成结构，确认对象之间的关系，包括继承关系、整体部分关系和对象关联；随后，针对需要对象协作来完成的功能，分析系统的工作过程，找出相关的对象及其交互关系，分析出每个对象在协作过程中承担的任务，从而确认协作中要求每个对象提供的支持，以细化对象及其交互关系的设计，完善软件的系统模型。以下各节将分别介绍用于创建系统模型的软件功能划分与对象设计、系统交互过程分析和方法设计，最后通过一个应用案例介绍软件系统建模和程序实现的全过程。

5.1.1 功能划分和对象设计

按照面向对象方法，对象就是组成软件系统的模块。系统建模的第一个步骤就是要将分析对象和设计目标抽象为一组对象。从对象的基本概念可知，对象代表了相对独立的实体或事物，然而针对软件设计涉及的各种数据信息和处理功能，究竟设置哪些对象、将哪些功能包装为独立的对象、将哪些功能作为哪些对象所提供的功能？在有些应用系统中，这些问题看似简单，而对于比较复杂的应用系统，却很难确定应该设置哪些对象。这种对象设计的决策问题涉及整个系统设计的合理性，错误的系统设计可能影响软件开发的效率，更可能导致软件可维护性、可扩充性的丧失。

良好的系统建模必须符合应用系统的自然规律。面向对象方法本身就来源于客观世界各种事物的组成和描述方法。在对象设计中，正是针对客观世界中各种实体和事物来进行对象抽象，然而软件系统中涉及的实体和事物往往具有不同的性质，有些系统侧重于复杂的数据分析，也有些系统涉及复杂的数据处理逻辑。因此，对于不同性质的数据信息和处理逻辑，需要采用不同的抽象分析方法，将问题包装成不同的对象，形成不同的软件模块。以下介绍几种设计方法，分别用于不同应用背景的对象抽象。

1. 数据信息的封装

任何信息系统都将涉及应用数据信息的处理。这些数据信息可能来自于应用需求，也可能来自于软件功能的实现需求，也可能是数据处理的中间结果。例如，企业信息系统需要管理产品信息、销售数据、人事管理信息、财务管理数据等各种数据，网上书店系统需要维护商品信息、订单信息、客户信息等各种数据。这些应用系统大都把各种数据信息保存在数据库内。系统运行时从数据库取出数据，进行加工处理后，打印成报表或保存在数据库中。鉴于这些数据都具有持久性和独立的语义，在数据库设计中都会组织成独立的

实体，而在软件设计中也应该作为独立的对象来处理。在应用问题的分析中，人们不难辨别出系统中维护的数据信息，进而根据数据信息的种类来设置相应的对象。

对于此类数据实体，通常可以封装为实体型对象和管理型对象。然而，对象设计和数据库设计有许多不同之处。一方面，对象设计支持考虑数据之间的一般与特殊关系和整体与部分关系的描述；另一方面，对象不仅描述数据的组织结构，而且支持相关功能的实现。因此，面向对象方法能够更有效地描述不同的数据实体之间可能存在的一般与特殊关系，也可以通过类成员对象等方式来支持对象的组合关系和聚合关系，从而支持复杂数据对象的描述。再者，根据各种数据信息的管理和维护必然需要相应的查找、增删改等基本操作，这些功能理当封装在此类对象之中，并为使用者提供简单的访问方法，使得开发者不必了解对象内部的实现细节，即可访问或修改对象内部的数据信息。因此，将软件需求中的数据信息及其维护操作封装为对象，是系统建模中最直观的方法。

有些软件系统中，数据信息本身的结构和语义可能都十分复杂。例如，在计算机语言处理系统中，语言本身表现为一种复杂的数据，各种数据实例具有不同的数据结构和语义。此类数据也需要根据其结构和语义进行分类，进而抽象为各种数据对象。

2. 系统功能的封装

各种计算机软件系统的系统功能大都涉及数据信息的处理。各种系统功能往往都是围绕着特定的数据信息展开的。例如，不同的图形图像处理系统可能涉及各种图形图像数据，并可能为这些数据提供编辑、绘制、变换和转储等各种功能。但是，每个具体的处理大都仅仅针对某种特定的图形或图像数据。又如，在各种企业信息管理系统中，可能提供了针对各种数据信息的管理功能，其中，产品管理功能针对产品数据，销售管理功能针对销售信息，财务管理功能针对财会数据信息而人事管理功能针对人事数据信息。在各种软件系统中，都有不少系统功能可以看作围绕着特定数据的数据处理功能。对于这种常见的应用场景，通常应该将各种数据处理功能划分给各种数据对象，按照数据的种类来设置对象，并将每个数据处理功能作为任务分配给负责管理此类数据的对象。在对象设计中，为该类数据的组织提供数据结构和成员变量，为相应的数据处理提供操作接口。

这种按照系统功能来划分软件模块和设置对象的方法是十分有效的。原理上和上述数据信息封装方法是一致的，它们之间的差别主要在于数据信息封装中对象仅仅考虑了查找和维护的操作需求，而系统功能封装方法从系统设计的角度考虑了各种系统功能的实现，要求按照各种功能来设置对象的方法。因此，前者的实现算法比较简单，而后者涉及的数据处理功能可能比较复杂，可能采用比较复杂的处理逻辑，其算法实现中也可能需要其他对象的协作，通过对象关联来完成。

3. 人机交互界面的封装

常见的各种软件系统中，几乎每个系统功能都具有独立的窗口、对话框或者菜单作为人机交互界面。整个系统中的多数功能似乎都可以划分到某个人机交互界面所代表的子系统。窗口、对话框等每个人机交互界面都为特定功能的实现提供一组操作，经常用于某类信息的处理。这种相对独立性使得开发者可以按照人机界面来进行系统功能的划分。

按照面向对象方法，每个人机交互界面都可以抽象为独立的对象，并且以界面操作方法作为这种软件模块的接口，以界面的组成元素和交互状态信息作为界面的属性。

在实用的软件系统开发中，Web 应用系统普遍采用浏览器作为人机交互界面，桌面软件普遍采用 Windows 图形化的视窗系统。各种软件开发工具已经以基本类库的方式为人机交互模块提供了支持。因此，此类软件模块的实现中普遍利用各种基本类库提供的各种窗口基类或对话框基类，根据应用需求来设计派生类。然而，在系统功能的实现中，窗口类仅仅负责人机交互的任务，接收到的各种输入指令数据通常需要转发给后面负责业务逻辑的控制模块和数据管理模块。因此，系统建模中必须根据系统功能的实现需求来建立三类软件模块的对象关联。

4. 数据加工模块的封装

在各种软件系统中，有些软件模块的主要功能是进行数据的加工处理，此类软件模块的特点在于加工处理的计算逻辑比较复杂。算法实现需要使用输入输出数据，也经常需要将计算过程划分为多个阶段，其中需要使用中间数据来保存中间加工结果。但是，加工处理的核心计算逻辑都是过程型的算法，而且往往需要多个子例程组合成完整的算法。例如，科学计算中常见的数学分析算法、多媒体应用中的各种图形图像处理算法、计算机语言处理中的语法分析算法、各种数据挖掘算法都可以封装为加工型对象。

在加工型对象的设计中，需要为数据加工任务设置成员函数，为中间数据设置成员变量。对于实现算法的各个子例程也应设置为独立的成员函数。对于复杂的中间数据，也应封装为独立的对象，根据计算过程中中间数据的使用需求来设置相应的方法。在数据加工对象实例的初始化阶段，要求将数据加工的对象作为输入数据。数据加工的成员函数应返回数据加工的结果。在不少加工型对象的任务中，输入输出数据都可能封装为不同的对象类，也可能需要通过数据访问代理获得。此时，系统建模中则需要建立相应的对象关联。

5. 虚拟机模块的封装

在文档编辑、表格编辑等各种人机交互功能的开发中，系统的动态工作可以表现为一个虚拟机。所谓虚拟机是一个仅仅接收特定的一组命令控制的、虚拟的机器。虚拟机按照依次接收到的命令进行工作，每个命令可能改变其工作状态。具有状态是虚拟机的主要特征，而状态变化取决于之前它接收到哪些命令。具有虚拟机特征的软件模块可以抽象为事务型对象。这种对象的任务是按照应用需求，根据收到的指令完成特定的操作。因此，事务型对象需要为每种命令提供响应方法。同时，为了表示虚拟机的工作状态，事务型对象需要设置成员变量。例如，对于一个支持鼠标操作的数据编辑模块，编辑对象必须响应各种编辑命令，同时需要保存正在编辑的数据，也需要记录鼠标键、Ctrl 键是否按下等操作状态信息。

和加工型对象相比，事务型对象并不能承担完整的计算任务，而是参与到计算过程中，负责部分计算功能，与其他对象一起支持整个计算任务的完成。事务型对象工作机制的主要特征在于这种虚拟机完全是被动地工作，完全听令于接收到的命令，而每个命令的

响应可能比较简单，也可能涉及复杂的算法实现和复杂的对象关联。

上述5种对象设计方法介绍了软件的系统建模中根据应用需求来确认对象，进行对象设计的基本方法。在面向对象的软件工程技术中，针对应用问题的对象确认和对象设计属于面向对象分析技术；针对计算机系统设计的对象确认和对象设计属于面向对象设计技术。本节则从程序设计的角度，针对软件系统的设计需求，探讨了软件设计中的对象确认和对象设计问题。读者可以从软件工程相关的技术论著中，找到更多的面向对象分析和设计方法。

5.1.2 交互过程分析和方法设计

按照上述的设计方法，人们可以将软件需求中的数据信息组织和数据处理功能分解为若干软件模块，抽象表示为一组对象。根据数据和事务的分类不难确定对象之间的继承关系，根据数据内部和功能结构的组成关系，也不难确认出对象之间的整体与部分关系。然而，面向对象系统的动态工作建立在消息传递和消息响应的基础之上，而消息传递则发生在存在对象关联的对象之间。在人机交互、通信代理和复杂算法的实现需求中，人们可以从软件需求中找到各种软件模块之间的协作关系，进而确认对象之间的关联。但是，在比较复杂的应用系统中，面对多个对象之间复杂的交互控制逻辑，设计者经常难以确认整个计算任务应该如何划分，难以确认相关的每个对象应该承担其中哪些子任务。

此类复杂问题来源于软件需求中复杂的算法和交互控制逻辑，更多地反映出系统的动态工作性质。对于系统的动态工作性质，可以借助于面向对象分析中的案例建模方法来解决。所谓案例建模就是分析软件的使用过程，设计一组使用案例(Use Case)来覆盖所有使用方法，进而要求系统设计和程序设计支持每个使用案例的实现。每个使用案例由按照发生顺序排列的一组交互事件组成，表示软件的一次具体的使用过程。

【例5-1】 图形编辑的使用案例。

考虑Word编辑软件中提供的图形编辑功能，通过鼠标操作可以移动文档中的直线、矩形和椭圆的位置，也可以通过鼠标拖动图元端点来修改图元的形状。发生在用户和Word软件之间的图形编辑过程可以抽象为一个使用案例。

这个使用案例描述了Word软件的一个使用过程，说明了用户的一系列操作和Word软件响应的对应关系。该案例完整地表示了一个图形编辑的过程，涉及Word软件所提供的部分图形编辑功能。显然，利用一组不同的使用案例，就有可能覆盖Word软件的所有使用方法。面向对象方法中的使用案例建模就是要找到一组能够满足系统需求的使用案例，作为系统分析、系统设计和程序设计的基础。

为了实现Word软件的图形编辑功能，软件内部设计必须支持此类使用案例的实现。在这个使用过程中，也不难分析出软件的内部组织。首先，图形编辑的对象是由若干图元组成的图形。图形和图元都具备相对的独立性，以及整体与部分关系，应该被抽象为独立的对象，分别管理组成图形的图元数据和图元内部数据。其次，Word软件的主要功能是文档编辑，其中包括文本编辑、图形编辑、表格编辑等功能。因此，按照功能划分的原则，图形编辑应该是独立的模块，也应抽象表示为图形编辑对象。同时，图形编辑对象是一个人机交互模块，应该提供响应用户输入操作的方法，而在本案例的范围内，输入操作只有

鼠标键的按下、释放和鼠标移动。再者，由于编辑的对象是图形中的当前图元，图形编辑对象需要建立与图形对象的对象关联，也需要建立与当前图元的对象关联，以支持图形数据的修改。从使用案例描述的交互过程可以看出图形编辑对象是图形中的当前图元，而当前图元来自于图形。然而，本题最大的难点在于如何确认图形对象和图元对象应该提供哪些方法以支持上述使用案例的实现。因此，设计者需要细化使用案例描述的交互过程，分析出每个对象应承担的责任。统一建模语言 UML 为对象之间协作关系的描述提供了图形化的方法。其中，序列图支持按照事件发生的顺序，描述使用案例中各个对象的交互关系。

【例 5-2】 图形编辑软件的建模方法。

面向上述分析得到的图形和图元等图形编译对象，基于例 5-1 给出的使用案例，采用 UML 序列图可以描述出图形编辑对象、图形对象、图元对象之间的协作关系。如图 5-1 所示，表 5-1 中每个步骤都由这些对象之间的一组消息传递来完成。

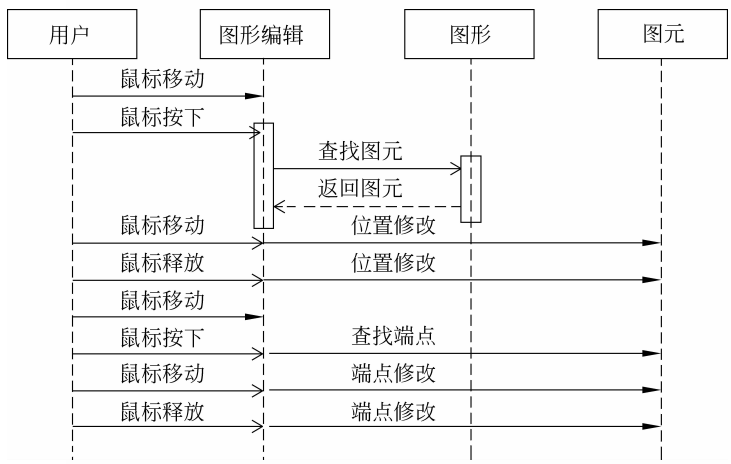

图 5-1 图形编辑使用案例的序列图

表 5-1 图元编辑的使用案例

序号	用　　户	Word
1	将光标移到某图元上	光标变为十字形
2	按下鼠标左键	显示被选中图元的各个端点
3	移动鼠标	光标指示的移动位置
4	释放鼠标左键	将被选中图元移到当前光标所在位置
5	将光标移到被选中图元的某端点上	光标变为箭头型，指示可变更方向
6	按下鼠标左键	光标变为十字形
7	移动鼠标	指示图元变更的形状
8	释放鼠标左键	按照当前光标位置，改变图元形状

如图 5-1 所示，序列图不仅描述了用户和图形编辑器的交互关系，而且描述了各种对象之间的交互关系。鉴于对象之间的交互都是通过消息传递来实现的，消息发送者和消息接收者之间必须存在对象关联。从序列图中的 8 个事件可以看出，图形编辑对象中应该保存指向图形对象和当前图元对象的指针，以支持对象之间的消息传递。同时，从每个对象接收到的消息中可以分辨出这个使用案例中每个对象必须响应的消息及其属性信息。在第 2 步，图形编辑对象需要从图形对象获得光标捕获的图元，因此要求图形对象必须提供一个方法，根据图形编辑对象提供的光标坐标信息，来查找并返回被捕获的当前图元对象。这一功能的完成也要求图元对象提供检查光标坐标是否处于图元外框之内的方法，来支持图元的捕获。在后续的几步操作中，图形编辑对象在响应鼠标操作时，都需要获取或修改图元的内部信息。在移动图元时，需要修改图元的坐标。在捕获图元的端点和修改图元形状时，都需要获得或修改图元各个端点的坐标。因此，从这个使用案例中可以确认出图形对象（Graph 对象）和图元对象（Shape 对象）分别应该具有以下方法。

```
class Graph
{
public:
    Shape * capture(CPoint pt);     //按照 pt 指定的位置查找并返回图元
};                                  //未找到时，返回 NULL 指针

class Shape
{
public:
    int isSelected(CPoint pt);      //判断该图元是否处于 pt 指定的位置
    int getControl(CPoint pts[ ]); //获得所有端点，返回端点数量
    void moveTo(int cx, int cy);    //按照指定的偏移量来移动图元
    void moveControl(int idx, int cx, int cy);
};                                  //按照给定的偏移量来修改图元的第 idx 个端点坐标
```

在上述类设计中，CPoint 是 MFC 基本类库提供的二维坐标类。在 Graph::capture 方法和 Shape::isSelected 方法中 CPoint 对象用于描述光标位置。在 Shape::getControl 方法中，CPoint 数组用于返回该图元所有端点的坐标，包括矩形的四角、直线的起点和终点、椭圆的外接矩形四角的坐标。通过 Shape::moveTo 方法可以修改图元的当前位置；通过 Shape::moveControl 方法可以修改端点坐标，达到改变图元形状的目的。

由此可见，通过细化使用案例的序列图，可以分清相关对象应该提供哪些方法来支持图形编辑软件的这个使用过程，从而保证了对象设计的完备性。鉴于一组完备的使用案例可以描述出完整的系统动态工作体系，如果每个对象都根据它所涉及的使用案例来确认收到的消息，并为这些消息设置相应的响应函数，就能够支持所有系统功能的实现，形成完整的系统模型。

这种基于使用案例的分析与设计方法，注重于系统动态工作性质的分析，为对象及其方法的设计提供了比较严谨的设计方法。在面向对象的软件工程技术中，使用案例建模主要用于问题域的系统分析，而后续的面向对象的系统设计和程序设计阶段中，需要根据

系统结构和软件结构的角度，进行使用案例模型的细化，以及类图形式的静态模型和序列图形式的动态模型的细化。本节则从程序设计的角度，针对软件系统的交互协作功能的实现需求，探讨了软件设计中的对象及其方法设计问题。面向对象的软件工程技术为系统的静态建模和动态建模提供了许多有效的分析工具和方法，包括统一建模语言 UML 提供的活动图、协作图和状态图等。读者可以从相关的软件工程技术论著中，找到更多的使用案例建模方法、静态建模方法和动态建模方法。

5.1.3 应用案例：网上书店

【问题陈述】

某网上书店系统用于实现通过网络来销售图书的功能。其主要功能包括：为网上用户提供注册登记、图书分类浏览、书籍信息浏览、购物车、订单制作和订单查询等功能；为书店的管理者提供图书信息输入、订单查询、出库、发货和结算等功能。系统应该支持多用户访问，每个注册用户拥有自己的购物车。未注册用户也可以进行图书信息的浏览，并拥有购物车；但是，用户只有在注册后才能提交订单。

为了简便起见，本系统假定仅仅支持货到付款的支付方式。本题的系统设计要求仅仅提供软件的系统模型，不考虑在 Web 应用环境下实用开发环境的约束条件。

【案例分析】

从上述应用需求来看，网上书店的使用者有三种：注册用户、未注册用户和书店管理者。采用使用案例的方式，人们可以描述不同使用者的使用过程。首先，考虑注册用户的一个使用案例，如表 5-2 所示。

表 5-2 注册用户的使用案例

	注册用户	网上书店
1	输入用户名和密码	检查用户名和密码的合法性；通过后，继续执行
2	选择不同的分类	展示指定分类的图书目录
3	选择有兴趣的图书	展示图书信息
4	将图书放入购物车	在购物车中保存该图书的信息
5	浏览购物车	展示购物车内容
6	选择结算	进入订单制作页面
7	填写数量、送货地址，提交订单	保存订单信息
8	查询订单信息	显示订单信息
9	选择撤销订单	删除订单

在注册用户的使用过程中，自然可能反复浏览和选择图书，也可能反复注册查询订单。但是，上述使用案例已经基本覆盖了网上书店为注册用户提供的所有功能。其次，考虑非注册用户的一个使用案例，如表 5-3 所示。

表 5-3　非注册用户的使用案例

	非注册用户	网 上 书 店
1	选择不同的分类	展示指定分类的图书目录
2	选择有兴趣的图书	展示图书信息
3	将图书放入购物车	在购物车中保存该图书的信息
4	浏览购物车	展示购物车内容
5	选择结算	提示需要用户注册
6	输入姓名、密码	将当前购物车合并到该用户的购物车
7	选择建立新用户	进入登记界面
8	填写姓名、密码、电话等信息	记录输入的个人信息，创建新用户

上述使用案例支持了非注册用户浏览选择图书、注册登记和创建新用户的过程，覆盖了网上书店系统为非注册用户的各种使用方法。当非注册用户成为新用户后，自然可以以注册用户的身份来使用网上书店系统。最后，考虑书店管理者的一个使用案例，如表 5-4 所示。

表 5-4　书店管理者的使用案例

	书店管理者	网 上 书 店
1	输入登录界面	检查用户名和密码的合法性；通过后，展示书店管理界面
2	选择不同的分类	展示指定分类的图书目录
3	选择图书信息输入	展示图书信息输入界面
4	输入图书信息	在当前分类内保存图书信息，返回书店管理界面
5	选择创建分类	展示分类信息输入界面
6	输入分类名称	创建图书分类，返回书店管理界面
7	选择订单查询	展示所有订单
8	选择订单	提示发货信息输入
9	填写送货人姓名	打印发货单，设置发货标记
10	确认成交	设置成交标记，返回书店管理界面

在这个使用案例中，描述了书店管理者添加图书信息、按照订单安排发货，以及在完成销售后进行结算三个使用过程。虽然，三个过程都是彼此独立的，可能反复发生，然而这个使用案例已经覆盖了网上书店系统为书店管理者提供的所有功能。

上述三个使用案例的设计覆盖了网上书店系统为三种用户提供的所有功能，也就是完整地说明了该系统的整体功能，涵盖了所有的使用方法。按照上述方法建立的这种使用案例模型为网上书店的软件开发提供了基本的设计需求。

另一方面，网上书店系统是一个面向多用户的网络应用系统，允许多用户（包括书店

管理者)同时访问图书和目录信息,制定或查询订单。图书信息为所有用户共享,而购物车数据则为具有相同用户名登录的用户所共享,系统允许不同的使用者从不同的网络客户机以相同的用户名登录到系统中,共同选购图书。

【解题思路】

从使用案例中可以分析出系统应该提供的人机交互功能。对于注册用户,系统需要提供注册页面、分类浏览页面、图书信息页面、购物车展示页面、订单制作页面、订单查询页面;对于非注册用户,还需要新用户登记页面。对于书店管理者,则需要图书信息输入页面、目录创建页面、订单审核页面和发货单填写页面。

1. 人机交互设计

从使用案例描述的交互关系出发,需要进行系统的人机交互设计,以确认每个页面提供数据输入、命令输入以及页面切换关系。考虑到用户反复浏览和操作的各种可能性,系统设计中设置了各页面的展示内容、数据输入、操作命令和页面切换转移关系,如表5-5所示。

表5-5 人机交互功能的设计

页面名称	展示内容	输入数据	操作命令	转移目标
登录页面		用户名、保密字	确认	分类浏览页面
			取消	分类浏览页面
			新用户	新用户登记
			管理	书店管理页面
分类浏览页面	一组书名、作者、出版社、价格		注册	注册页面
			指定图书	图书信息展示
			其他分类	分类浏览页面
			购物车	购物车展示
图书信息页面	书名、作者、出版社、价格、书号、目录、摘要		选定	购物车展示
			继续	分类浏览页面
购物车展示页面	一组书名、作者、出版社、价格		预订	订单制作页面或注册页面
			继续	分类浏览页面
订单制作页面	用户名、电话、购物车内容	每本书的数量、送货的时间和地址	确认	订单查询页面
			取消	分类浏览页面
订单查询页面	一组订单号码、所有书名和数量、总价、送货时间和地址、处理状态		确认	分类浏览页面
			撤单	订单查询页面
注册页面		用户名、保密字、电话	确认	分类浏览页面
			取消	分类浏览页面