CHAPTER 3

# 第 3 章

# 面向对象软件设计工具Rational Rose

## 3.1 概述

Rational Rose 是 IBM 公司出品的一种面向对象的统一建模语言的可视化建模工具。利用这个工具，用户可以建立用 UML（Unified Modeling Language，统一建模语言）描述的软件系统模型。

通过 Rational Rose，软件分析与设计人员、系统工程师和开发人员可以在软件项目的分析与设计阶段，快速地创建面向对象分析与设计的各种模型，以实现软件系统架构与组织结构的可视化，便于将需求分析转化为系统设计。

Rational Rose 是基于 UML 的可视化建模工具，是面向对象的设计工具。UML 是一种通过简明易懂的图形、符号和辅助文字来建模的方法，可以方便、高效地表示设计人员的设计思想，同时利于软件团队成员之间的交流。

UML 最初始于 1997 年的一个 OMG 标准，它是一个支持模型化和软件系统开发的图形化语言，为软件开发的所有阶段提供模型化和可视化支持。面向对象的分析与设计方法的发展在 20 世纪 80 年代末至 20 世纪 90 年代中出现了一个高潮，UML 就是这个高潮的产物。它不仅统一了 Booch、Rumbaugh 和 Jacobson 的表示方法，而且对其做了进一步的发展，并最终统一为大众所接受的标准建模语言。

UML 定义了下列 5 类（用例图、静态图、行为图、交互图和实现图）共 9 种模型图（用例图、类图、对象图、状态图、活动图、顺序图、协作图、构件图和部署图）。

表 3-1 对 UML 的 9 种图进行了简要描述。

表 3-1　UML 的 9 种模型图

| 类别 | 名称 | 描　　述 |
|---|---|---|
| 用例图 | 用例图 | 用系统、操作者、用例和用例之间的关系来描述系统的一个功能 |
| 静态图 | 类图 | 用于定义系统的类，包括描述类之间的关系（如关联、泛化、聚合等）以及类的内部结构，包括类的属性和操作 |
| | 对象图 | 可看作是类图的实例，辅助理解较为复杂的类图 |
| 行为图 | 状态图 | 描述一类对象的所有可能状态以及事件发生时状态的转移条件 |
| | 活动图 | 描述为满足用例要求所要进行的活动以及活动间的约束关系 |
| 交互图 | 顺序图 | 描述对象之间的动态交互关系，着重表现对象间消息传递的时间顺序 |
| | 协作图 | 用于描述相互协作的对象间的交互关系和链接关系 |
| 实现图 | 构件图 | 描述部件的物理结构及各部件之间的依赖关系，显示代码的静态结构。一个部件可能是一个资源代码部件、一个二进制部件或一个可执行部件 |
| | 部署图 | 描述处理器、硬件设备和软件构件在运行时的架构，显示系统硬件的物理拓扑结构及在此结构上执行的软件 |

基于上述 UML 绘图，Rational Rose 支持的主要功能如下。

（1）对业务进行建模（工作流）；

（2）建立对象模型（表达信息系统内有哪些对象，它们之间是如何协作完成系统功能的）；

（3）对数据库进行建模，并可以在对象模型和数据模型之间进行正、逆向工程，相互同步；

（4）建立构件模型（表达信息系统的物理组成，如有什么文件、进程、线程、分布如何等）；

（5）生成目标语言的框架代码，如 VB、Java、Delphl 等。

## 3.2　基本使用

Rational Rose 是一个可视化的面向对象建模工具，本章使用的 Rational Rose 版本为 2007，下面介绍其基本使用方法。

### 3.2.1　操作面板介绍

启动 Rose 后，进入如图 3-1 所示的主界面。Rose 的界面主要分为 5 个部分：菜单和工具栏、浏览器窗口、模型视图窗口、文档窗口和日志窗口。各个部分有不同的功能。

**1. 菜单和工具栏**

菜单和工具栏位于主界面的上方，由一系列菜单项和常用工具选项组成。

**2. 浏览器窗口**

浏览器窗口位于主界面的左侧，用于可视化地显示模型中所有元素的层次结构。浏览器窗口与模型视图窗口具有同步性，任何对模型元素的更新会同时反映在两个窗口中。

浏览器窗口分为 4 个视图：用例视图、逻辑视图、构件视图和部署视图，如图 3-2 所示。

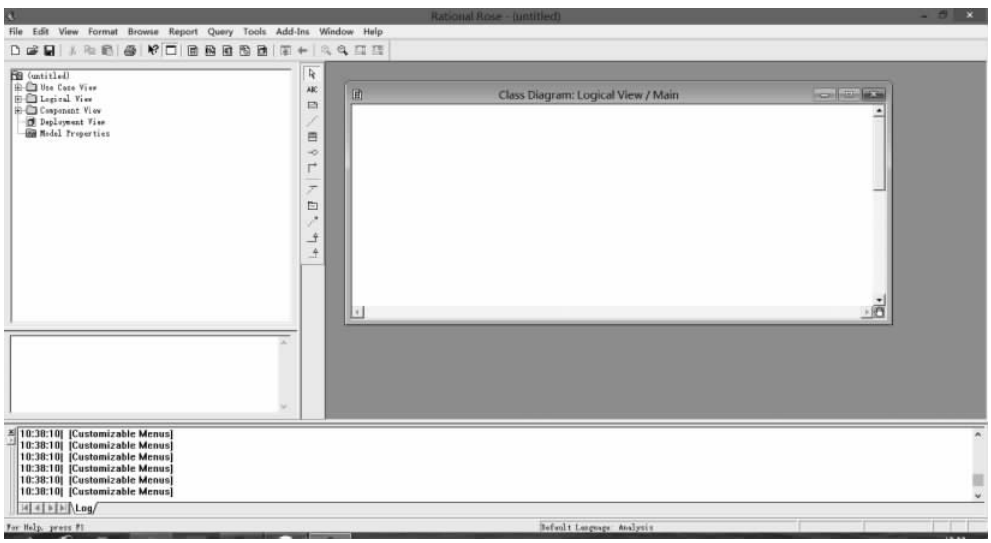


图 3-1　主界面

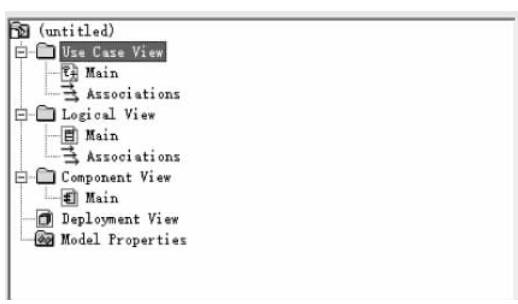


图 3-2　浏览器窗口

每个视图针对不同的模型，可以进行不同的操作。表 3-2 列举了每类视图包含的模型或元素。

**表 3-2　4 类视图**

| 视　　图 | 主要模型/元素 |
|---|---|
| 用例视图(Use Case View) | 用例图、类图、协作图、顺序图、状态图、活动图 |
| 逻辑视图(Logical View) | 用例图、类图、协作图、顺序图、状态图、活动图 |
| 构件视图(Component View) | 构件图 |
| 部署视图(Deployment View) | 设备/配置 |

在浏览器窗口中可以对模型或元素进行操作。创建一个新的模型时，右键单击对应的视图，在快捷菜单中选择 New，选择所要创建的模型图即可。双击出现的模型图，可以在右侧模型视图窗口中打开，拖曳一些浏览器窗口的模型元素到其中，即可开始绘制模型图。同理，创建一个新的模型元素时，右键单击新模型元素所属的父元素(可以是视图、模型视图或包)，在快捷菜单中选择 New，选择所要创建的模型元素即可。可以右键单击创建的模型元素进行重命名。

删除一个模型元素时，在浏览器窗口中选中该元素，右键单击选择 Delete 操作即可。

**3. 模型视图窗口**

模型视图窗口即绘图区域。可以在此窗口下创建和修改模型，模型视图中的每一个图标表示模型中的一个元素，如图 3-3 所示。

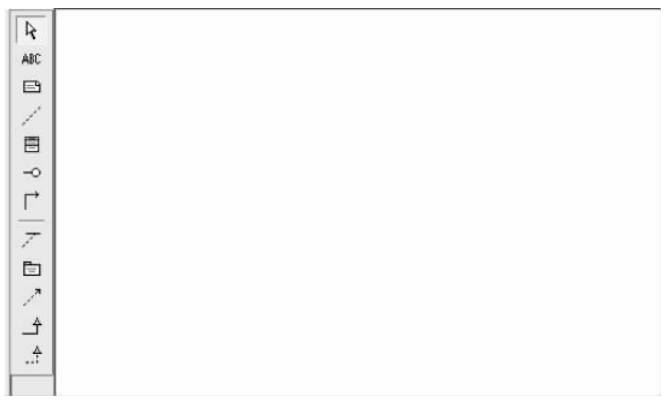


图 3-3　模型视图窗口

创建一个模型元素时，在模型视图窗口中，单击工具箱中的相应图标，并在绘图区域中合适的位置单击即可完成创建。右键单击该模型元素，选择 Open Specification 可以编辑模型元素的属性，如图 3-4 所示。

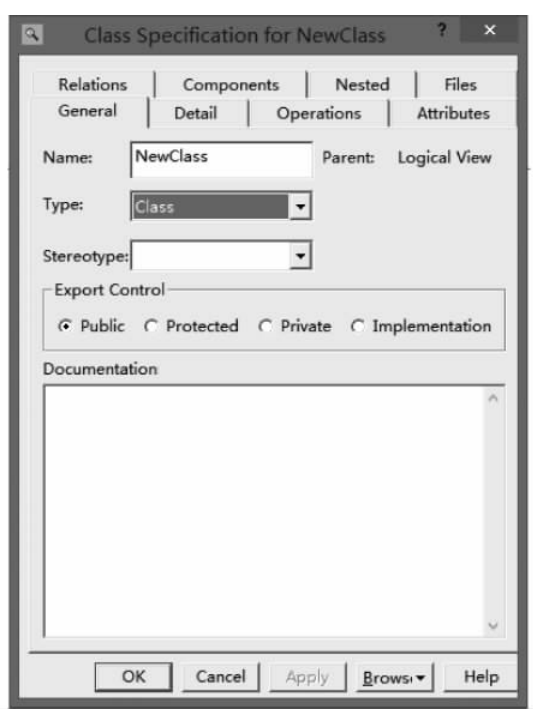


图 3-4　编辑元素属性

删除该模型元素时，右键单击该模型元素，选择 Edit 选项下的 Delete，可以将该元素图标从模型视图中删去（但图标对应的模型元素仍然存在于模型中）；选择 Delete from Model，如图 3-5 所示，可以将该模型元素彻底从模型中删去，此时浏览器窗口中对应视图下已不再显示该元素。

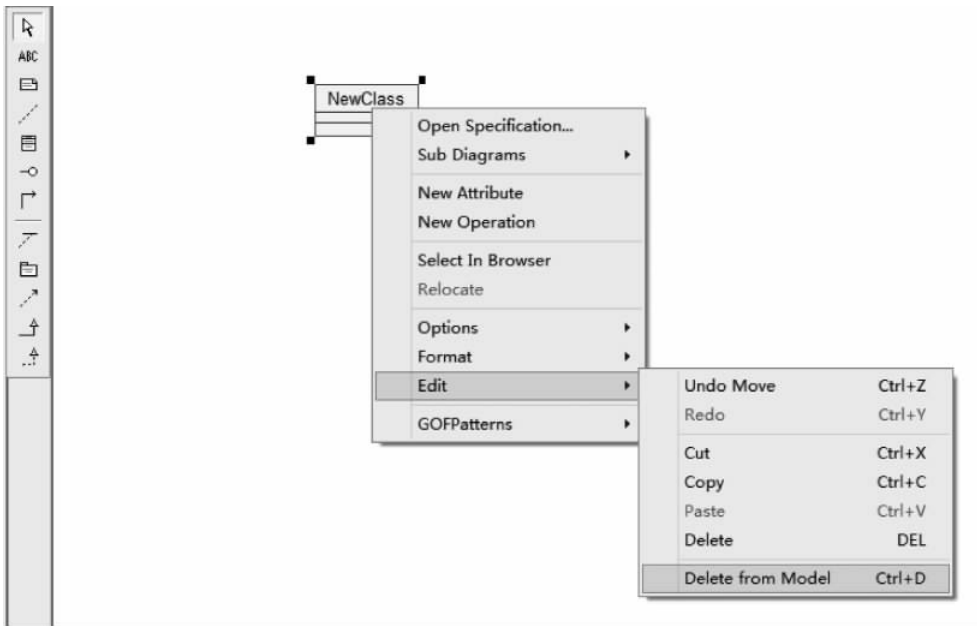


图 3-5　删除元素

**4. 文档窗口**

文档窗口用来显示和书写各个模型元素的文档注释，如图 3-6 所示。

**5. 日志窗口**

日志窗口用来显示系统操作的日志记录，如图 3-7 所示。

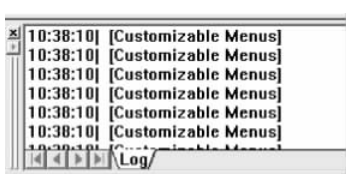


图 3-6　文档窗口

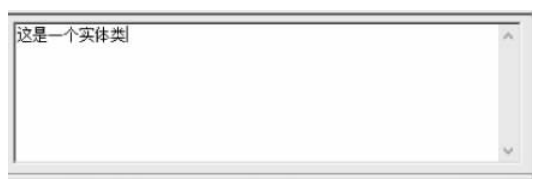


图 3-7　日志窗口

## 3.2.2　基本使用介绍

下面介绍一些 Rose 的基本使用方法，详细的建模过程和方法在后续章节会以案例的方式加以介绍。

**1. 创建模型**

启动 Rational Rose，在浏览器窗口中找到所要创建的模型所属的视图，右键单击选

择 New 后，单击选择所要创建的模型。操作完成后，对应的视图下出现了新创建的模型。右键单击 Rename 可以重命名该模型。如图 3-8 所示为在用例视图下创建用例模型。

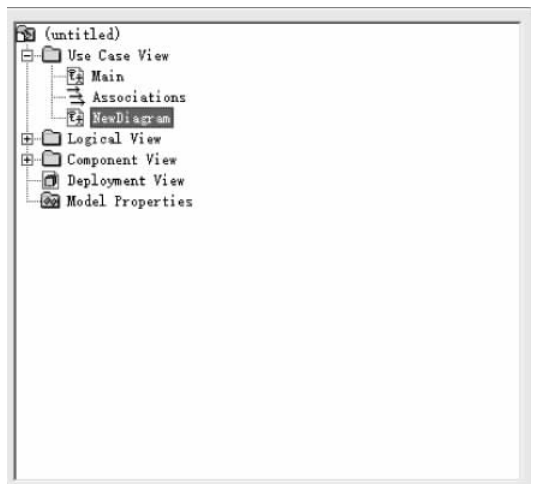


图 3-8　创建用例模型

**2. 编辑模型**

选中位于浏览器窗口中某个视图下的模型，双击打开后，则可以在模型视图窗口中显示和编辑该模型。在模型视图窗口的左侧工具箱中单击选中一个图标，并在模型图中合适位置单击，可以创建一个模型元素并在模型中显示，如图 3-9 所示。


图 3-9　编辑模型

在模型图中，选中一个模型元素，右键单击它可以进行一些操作。选择 Open Specification 可以编辑该元素的属性。选择 Edit→Delete，可以将元素图标从模型图中删去；选择 Edit→Delete from Model，可以将该元素从模型中彻底删除，如图 3-10 所示。

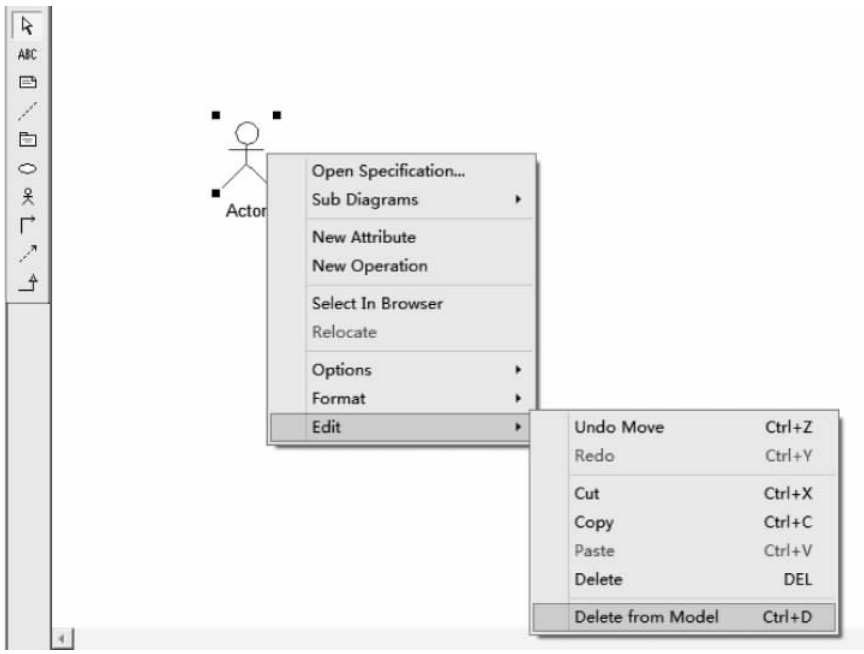


图 3-10　删除模型元素

**3. 保存模型**

在完成一个模型的创建和编辑后，可以单击菜单和工具栏中的 File，在菜单项中选择 Save 或 Save As，保存为.mdl 格式的文件，如图 3-11 所示。


图 3-11　保存模型

**4. 导入模型**

单击菜单和工具栏中的 File，在菜单项中选择 Open，可以从本地导入先前保存好的.mdl 文件。导入到 Rose 中后，可以在模型视图窗口和浏览器窗口中看到之前保存过的模型，并继续进行编辑，如图 3-12 所示。

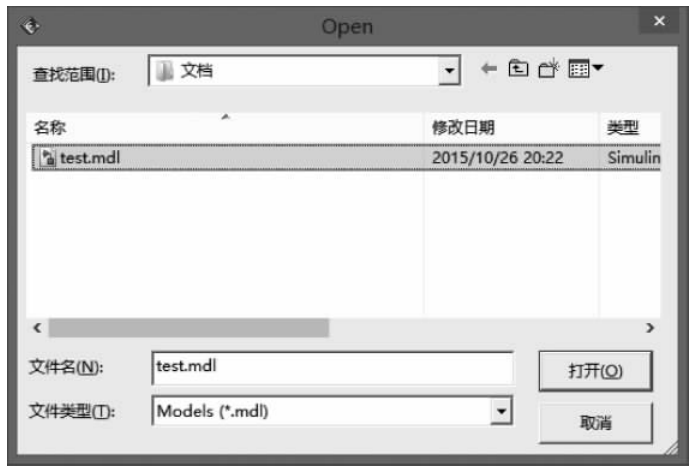


图 3-12　导入模型

## 3.3　利用 Rational Rose 进行“小型二手货交易平台”面向对象设计

本节以“小型二手货交易平台”案例为示例，利用 Rational Rose 进行面向对象设计和模型绘制。

### 3.3.1　构建用例模型

通常用用例图描述用例模型，下面以“小型二手货交易平台”的“交易子系统”为例说明如何利用 Rational Rose 进行用例模型设计。

下面为绘制用例图的实验步骤。

(1) 打开 Rational Rose，在浏览器窗口的 Use Case View 中右键选择 New→Use Case Diagram，新建一个用例图。双击在模型视图窗口中打开用例图，如图 3-13 所示。

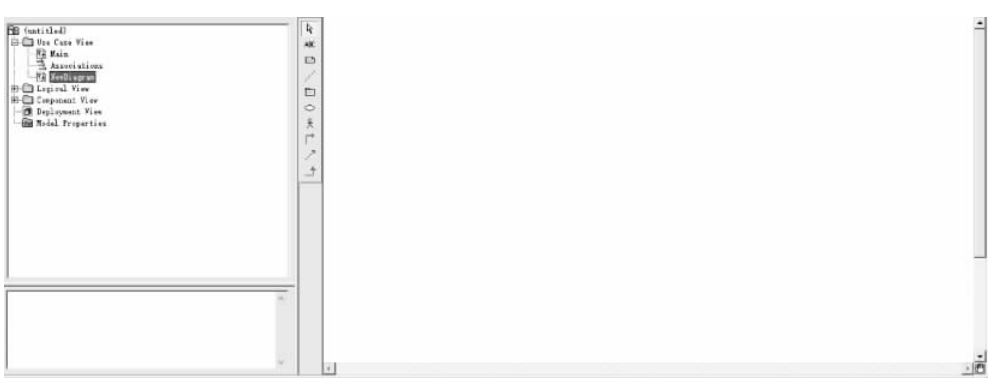

图 3-13　新建用例图

（2）单击选中工具箱中的“参与者”图标，在绘图区域的合适位置单击放入，调整其位置与大小，右键选择 Open Specification 可以编辑其属性，如图 3-14 所示。

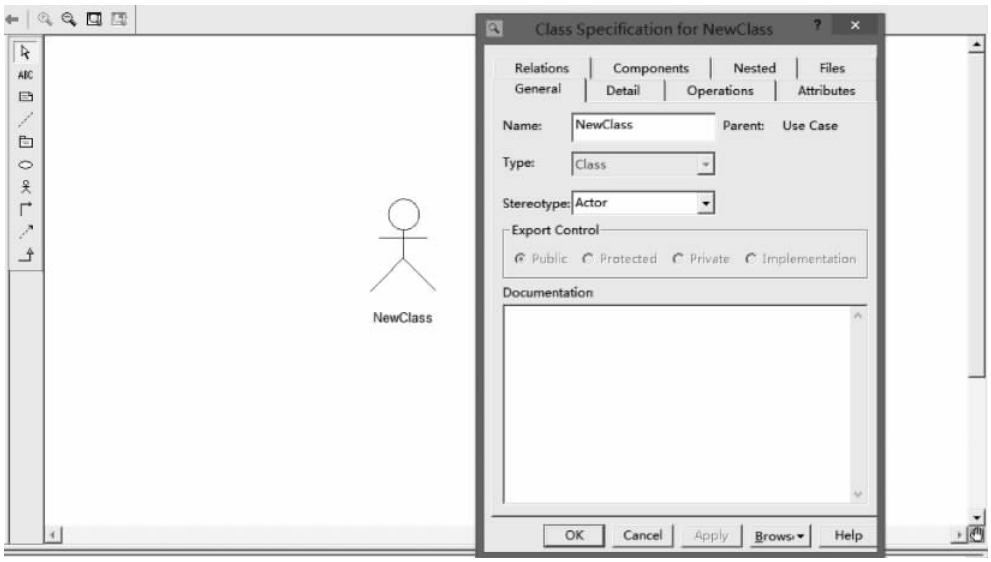


图 3-14　编辑参与者属性

（3）单击选中工具箱中的“用例”图标，在绘图区域的合适位置单击放入“用例对象”，调整其位置与大小，右键选择 Open Specification 可以编辑其属性，如图 3-15 所示。

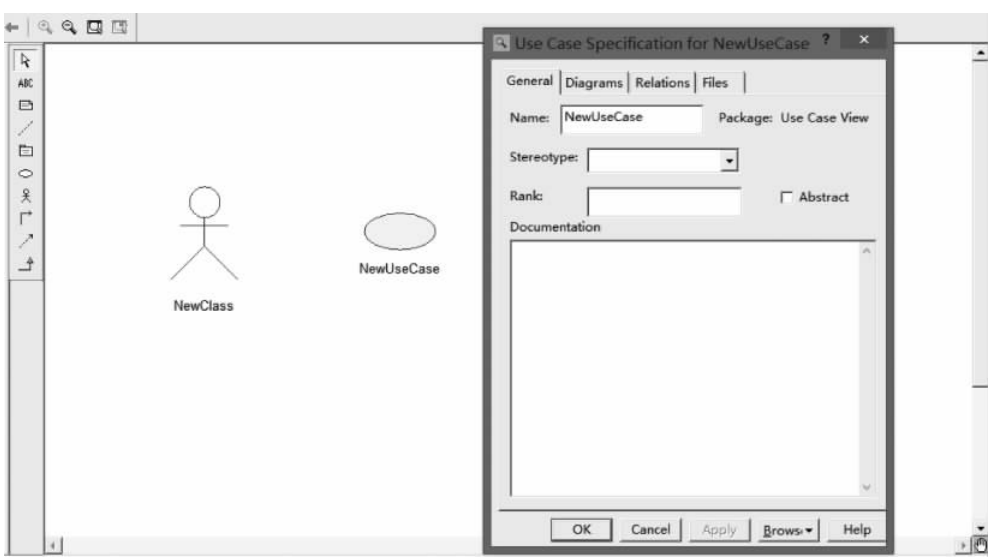


图 3-15　编辑用例属性

（4）用例之间的关系包括 include 和 extend，可以用工具箱中的 Dependency or Instantiates 图标来表示。单击选中该图标，在绘图区域的某一用例上单击并指向另一用例，出现的带箭头的虚线表示用例之间的关系。右键单击 Open Specification 编辑该虚线，在 Stereotype 中选择 include 或 extend 以代表关系的种类，如图 3-16 所示。

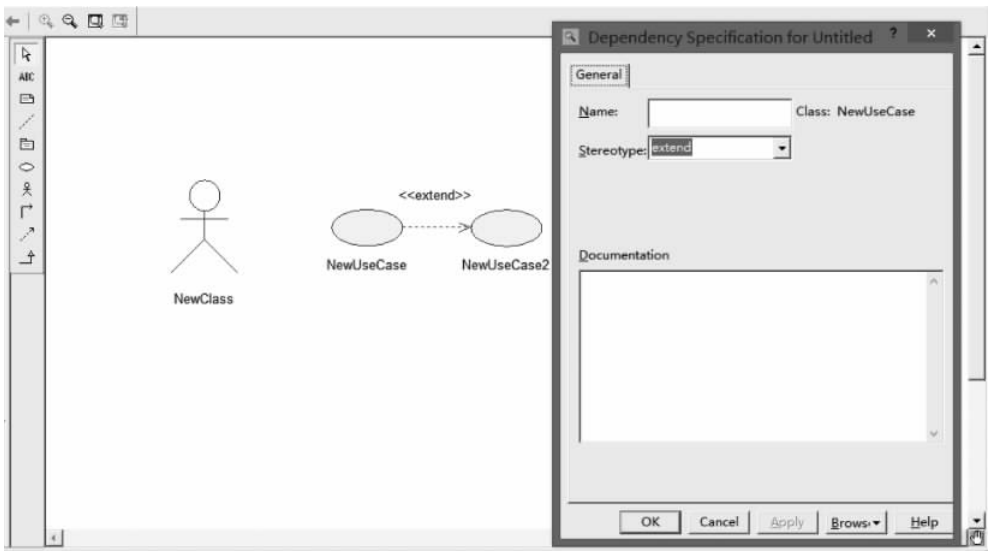


图 3-16　编辑用例间的关系

(5) 参与者之间的泛化关系可由工具箱中的 Generalization 图标来表示。单击选中该图标,在绘图区域的某一参与者上单击并指向另一参与者,出现的带三角箭头的实线表示参与者之间的泛化关系,如图 3-17 所示。

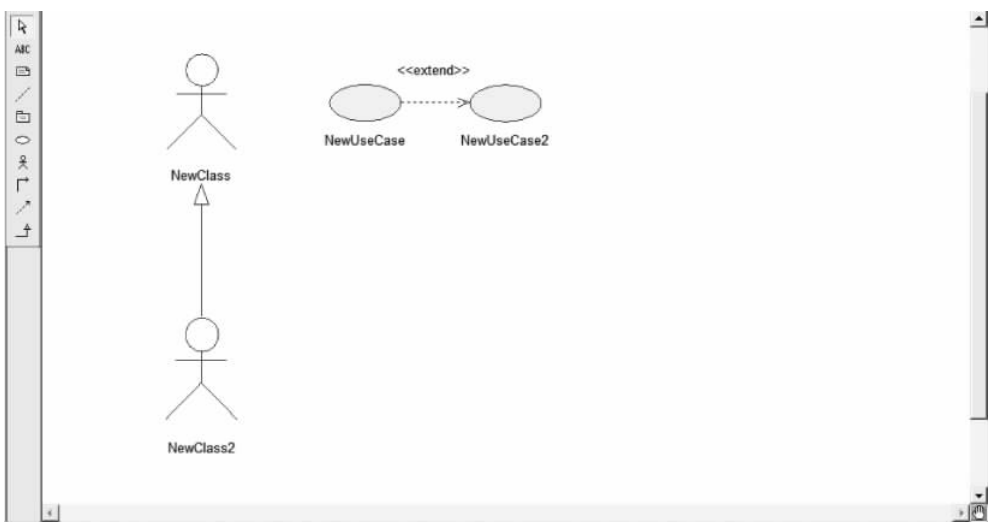


图 3-17　参与者的泛化

(6) 单击选中工具箱中的 Unidirectional Association 图标,在绘图区域的某一参与者上单击并指向某一用例,出现的带箭头的实线表示参与者使用该用例,如图 3-18 所示。

(7) 结合前文中需求分析,重复步骤(2)～(6),完成交易子系统的用例图绘制。参考绘图如图 3-19 所示。

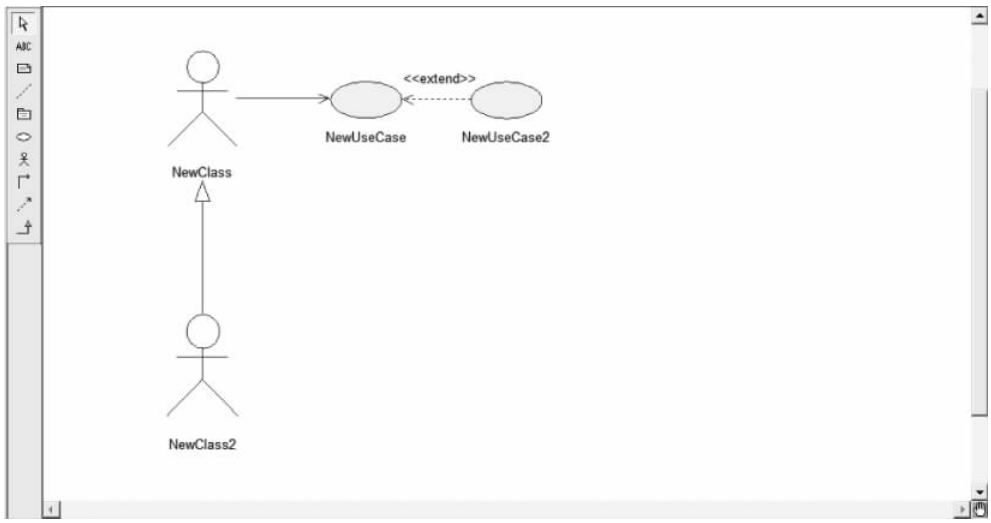


图 3-18　参与者使用用例

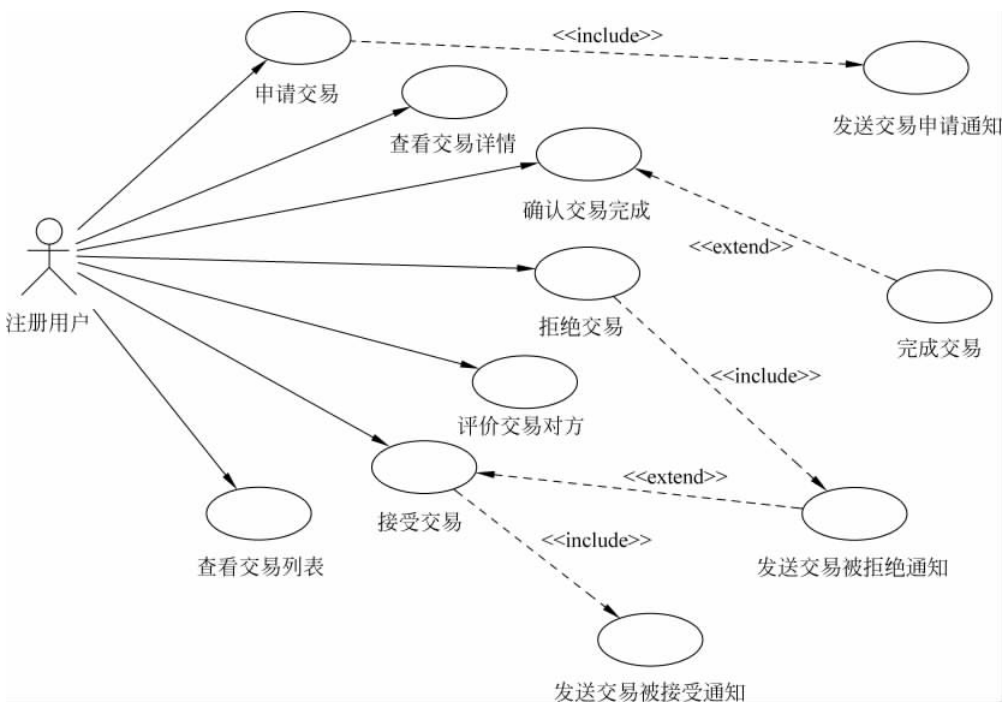


图 3-19　交易子系统的用例图

## 3.3.2　构建逻辑模型

逻辑模型主要建立了系统的类模型。2.4.2 节对类图已做了介绍，这里不再赘述。下面以“小型二手货交易平台”为例，说明如何利用 Rational Rose 绘制“用户子系统”类图。

下面为绘制类图的实验步骤。

（1）打开 Rational Rose，在浏览器窗口中的 Logical View 右键选择 New→Class Diagram，新建一个类图。双击在模型视图窗口中打开类图，如图 3-20 所示。

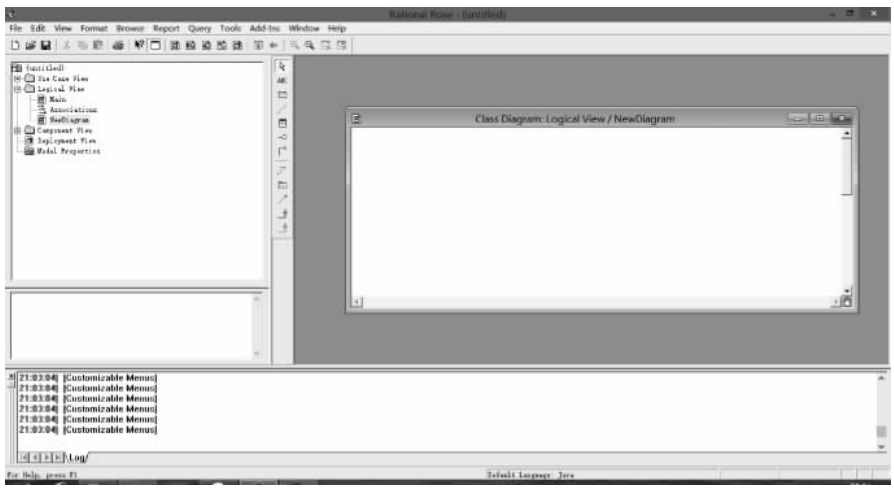


图 3-20　新建类图

（2）单击选中工具箱中的 Class 图标，在绘图区域的合适位置单击放入“类对象”，调整其位置与大小，右键选择 Open Specification 可以编辑该类的属性，如图 3-21 所示。

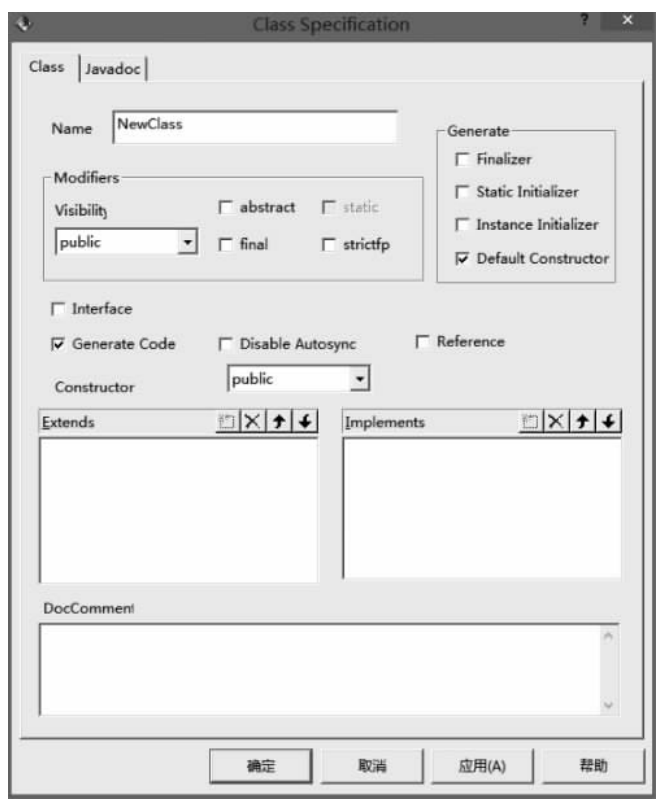


图 3-21　编辑类的属性

(3) 编辑类的属性时,在 General 选项卡下可以更改类的名称和 Stereotype。类可分为实体类、边界类和控制类,对应 Stereotype 中的 entity, boundary, control。右键单击类,选择 Options→Stereotype Display,可以选择类的 Stereotype 的显示方式。建议选择 Decoration 方式,如图 3-22 所示。

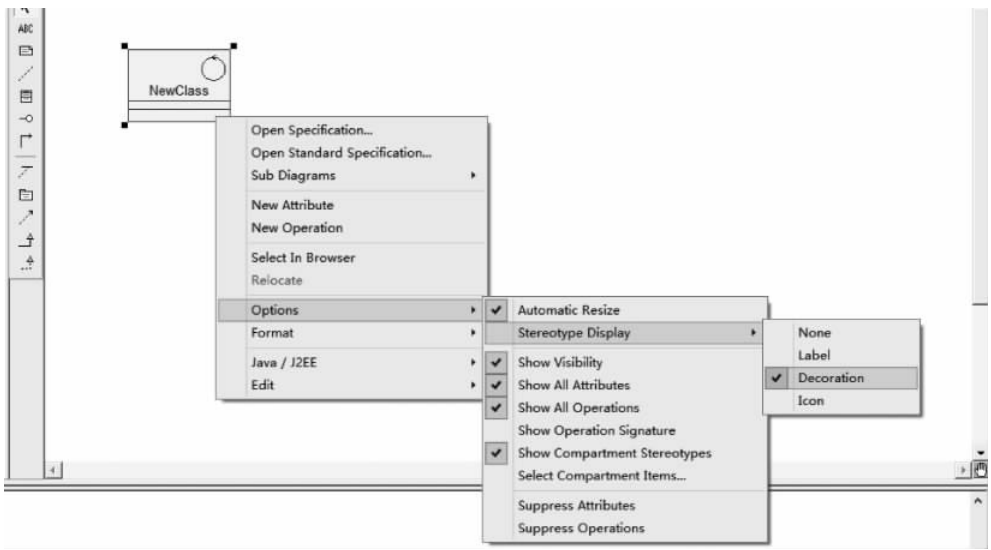


图 3-22 类的 Stereotype 及显示

(4) 编辑类的属性时,在 Attributes 选项卡下可以设置类的字段(数据成员)。右键单击 Insert,可以添加新字段。双击字段,可以编辑字段的类型、名称等,如图 3-23 所示。

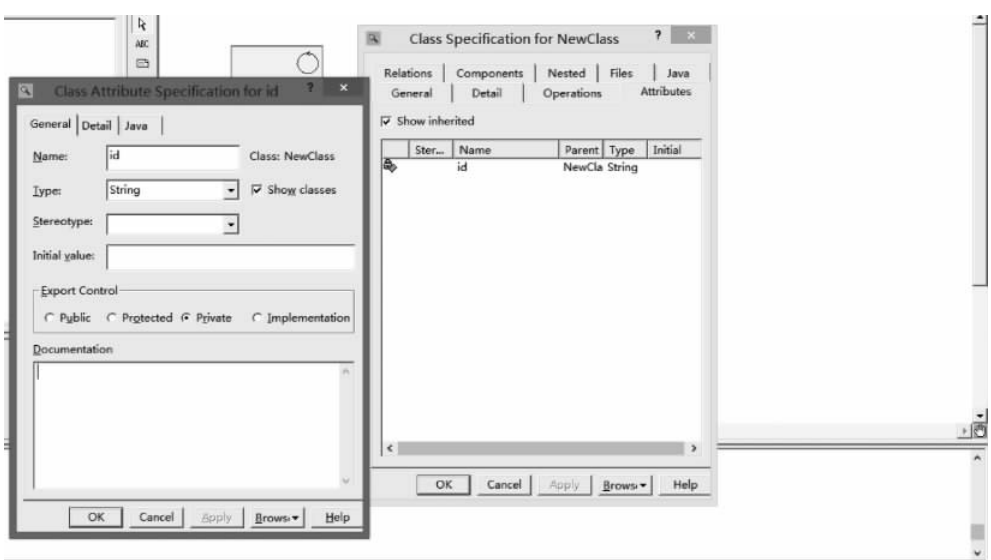


图 3-23 编辑字段

(5) 编辑类的属性时,在 Operations 选项卡下可以设置类的方法(成员函数)。右键单击 Insert,可以添加新方法。双击方法,可以编辑方法的名称、形参、返回值类型、访问权限

等，如图 3-24 所示。

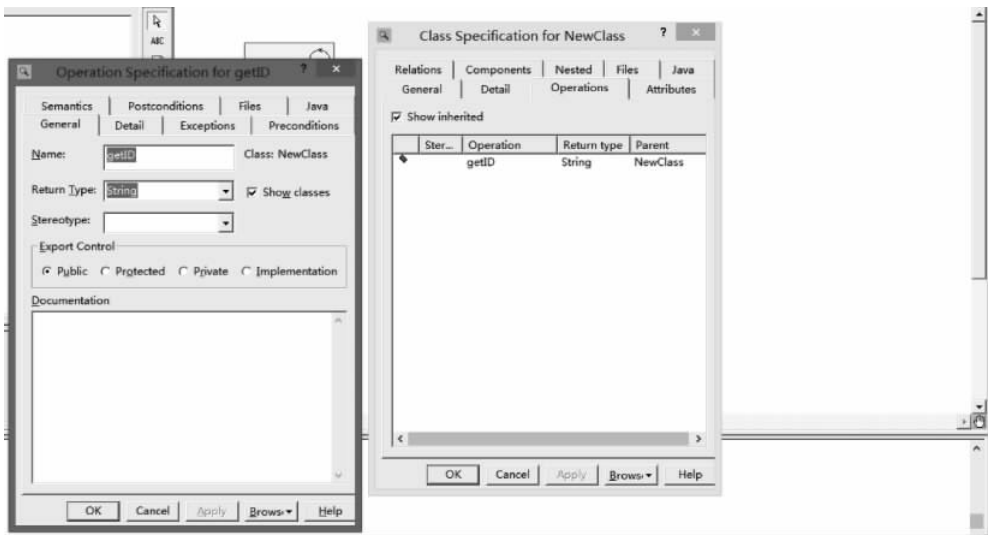


图 3-24　编辑方法

(6) 重复步骤(2)，在绘图区域中添加第二个类。单击选中工具箱中的 Unidirectional Association 图标，在绘图区域中的第一个类单击并指向第二个类，出现的带箭头的实线表示两个类之间的关系。右键单击该实线，选择 Open Specification 可以编辑关系的属性，如图 3-25 所示。

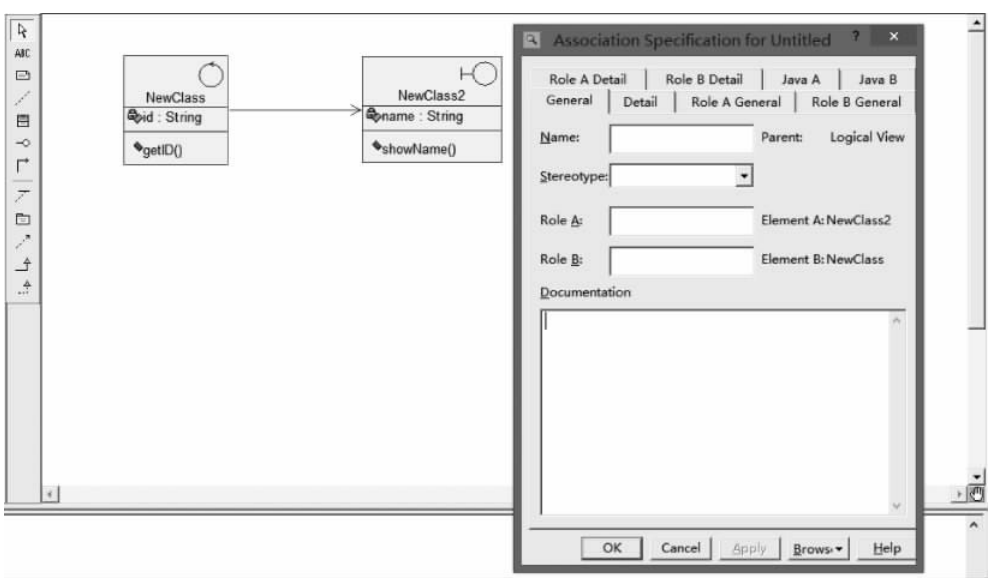


图 3-25　编辑关系的属性

(7) 在 Role A Detail 和 Role B Detail 两个选项卡下可以编辑类和关系的属性。Multiplicity 可以设置关系的多重性，Navigable 可以决定关系是否有方向性。若要表示组合或聚合关系，则在子类对应的选项卡下勾选 Aggregate，组合关系则需要在 Containment

of NewClass 下选择 By Value，如图 3-26 所示。

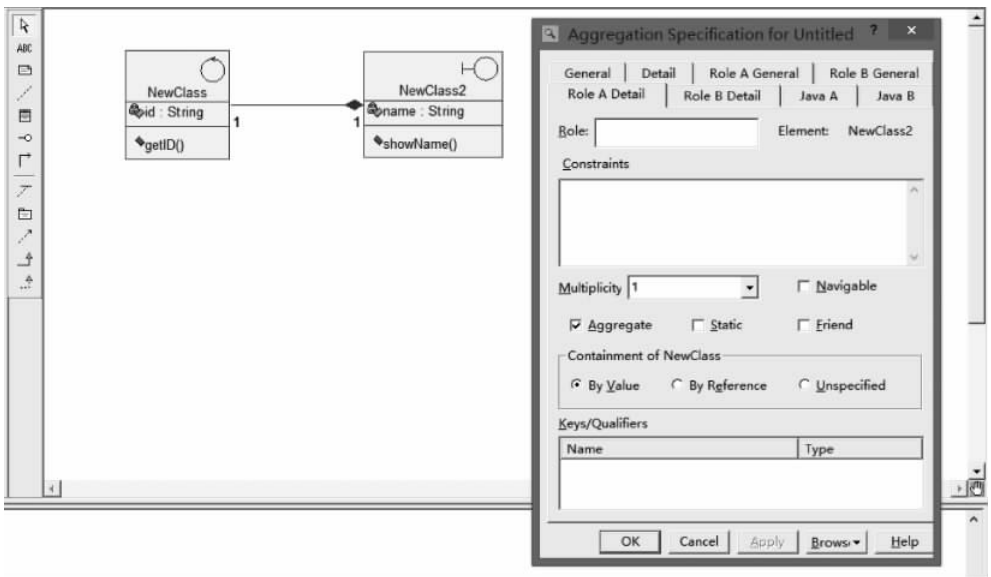


图 3-26　聚合与组合关系

(8) 结合前文中的需求分析，重复步骤(2)～(7)，完成用户子系统类图的绘制。参考绘图如图 3-27 所示。

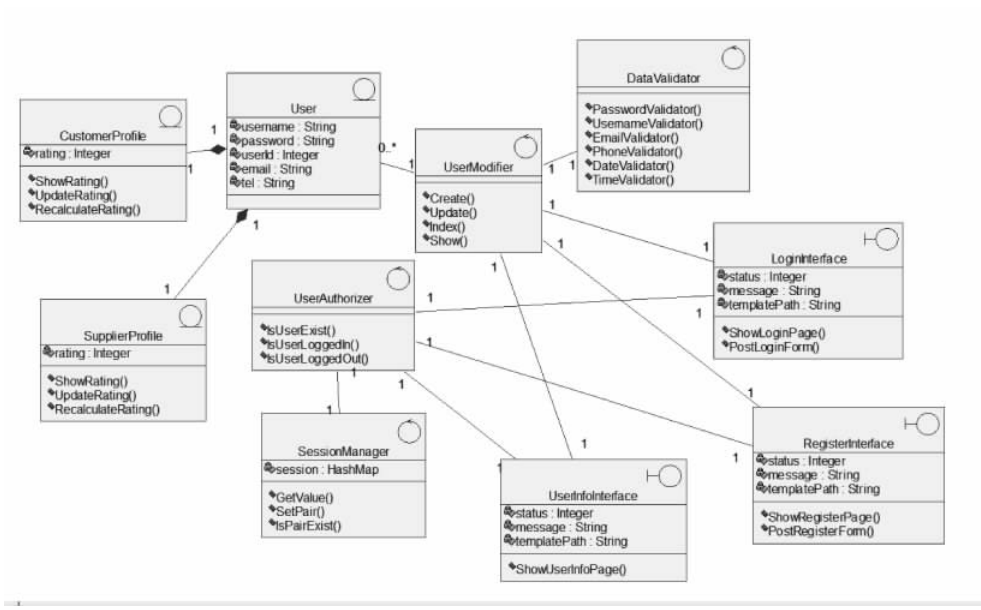


图 3-27　用户子系统的类图

## 3.3.3　根据用例模型绘制顺序图

顺序图是 UML 交互图的一种，描述对象之间的动态交互关系，着重表现对象间消息传

递的时间顺序。顺序图的横向表示不同的对象，纵向表示时间。

顺序图的基本元素包括：对象、生命线、消息、终止符。对象由矩形表示，矩形内标有对象名。从表示对象的矩形向下延伸的垂直虚线是对象的生命线，表示对象存在的一段时间。对象之间的通信由一条带箭头的水平实线表示，从发送消息的对象指向接收消息的对象。接收到消息的对象被激活，用生命线上的细长矩形表示。消息可以带名称和参数，接收方对象可以发送反馈消息到发送方对象，由一条带箭头的反向水平虚线表示。当对象终止时，在生命线底端用一个终止符“×”表示终止。

根据用例模型，可以设计并绘制出对应的顺序图。下面以“小型二手货交易平台”的“用户子系统”的“用户登录”用例为例，说明如何利用 Rational Rose 绘制顺序图。

下面为绘制顺序图的实验步骤。

(1) 打开 Rational Rose，在浏览器窗口中的 Logical View 右键选择 New→Sequence Diagram，新建一个顺序图。双击在模型视图窗口中打开顺序图，如图 3-28 所示。

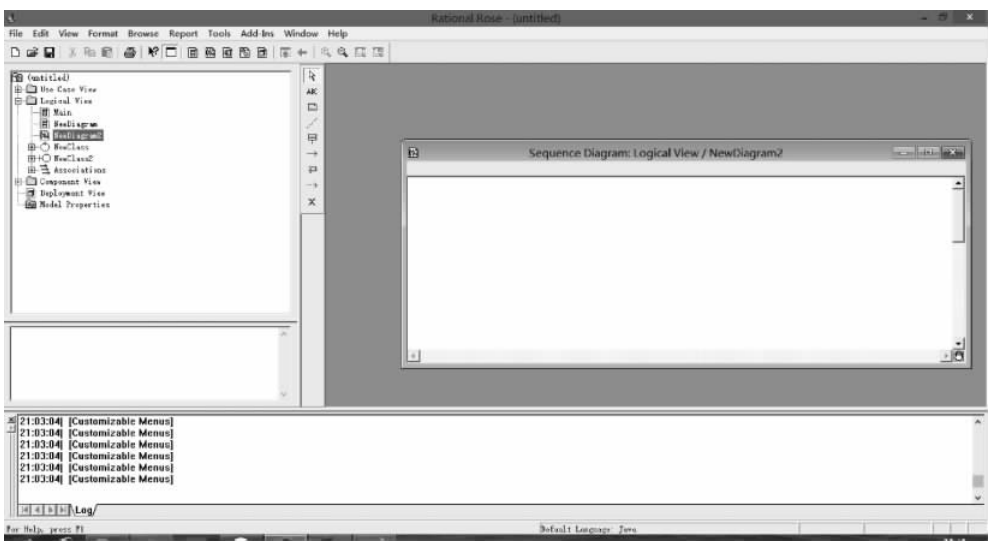


图 3-28　新建顺序图

(2) 单击选中 Use Case View 下的参与者，拖动至顺序图的绘图区域并放入，调整其位置与大小，右键选择 Open Specification 可以编辑该参与者的属性，如图 3-29 所示。

(3) 单击选中 Logical View 下的类，拖动至顺序图的绘图区域并放入，调整其位置与大小，右键选择 Open Specification 可以编辑该对象的属性，如图 3-30 所示。

(4) 重复步骤(3)，在绘图区域中添加第二个对象。单击选中工具箱中的 Object Message 图标，在绘图区域中的第一个对象下的生命线单击并指向第二个对象，出现的带箭头的水平实线代表消息。调整其位置，右键选择 Open Specification 可以编辑其属性，如图 3-31 所示。

(5) 单击选中工具箱中的 Return Message 图标，在绘图区域中的第二个对象下的生命线单击并指向第一个对象，出现的带箭头的水平虚线代表反馈消息。调整其位置，右键选择 Open Specification 可以编辑其属性，如图 3-32 所示。

(6) 单击选中工具箱中的 Destruction Marker 图标，在绘图区域中的对象下的生命线

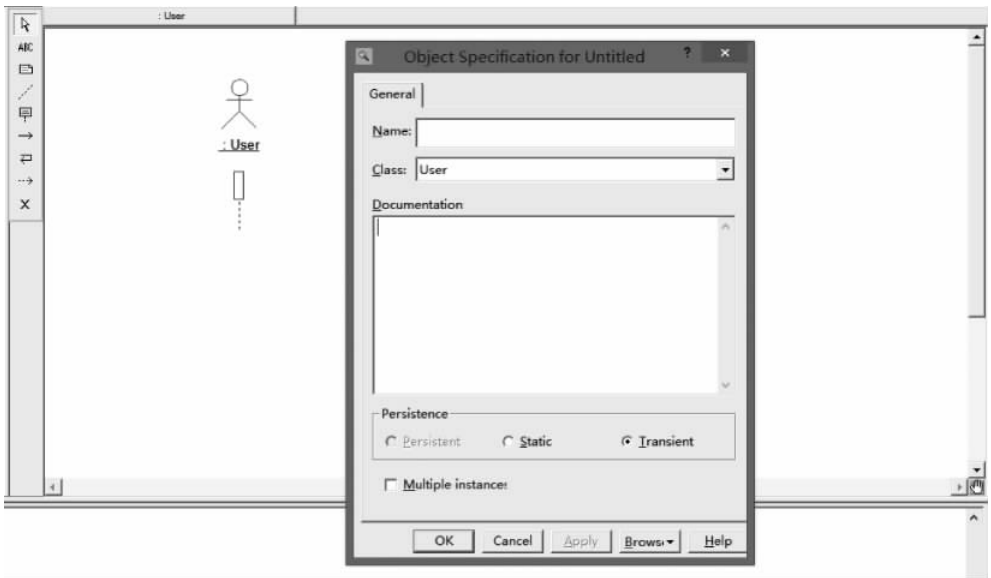


图 3-29　编辑参与者的属性

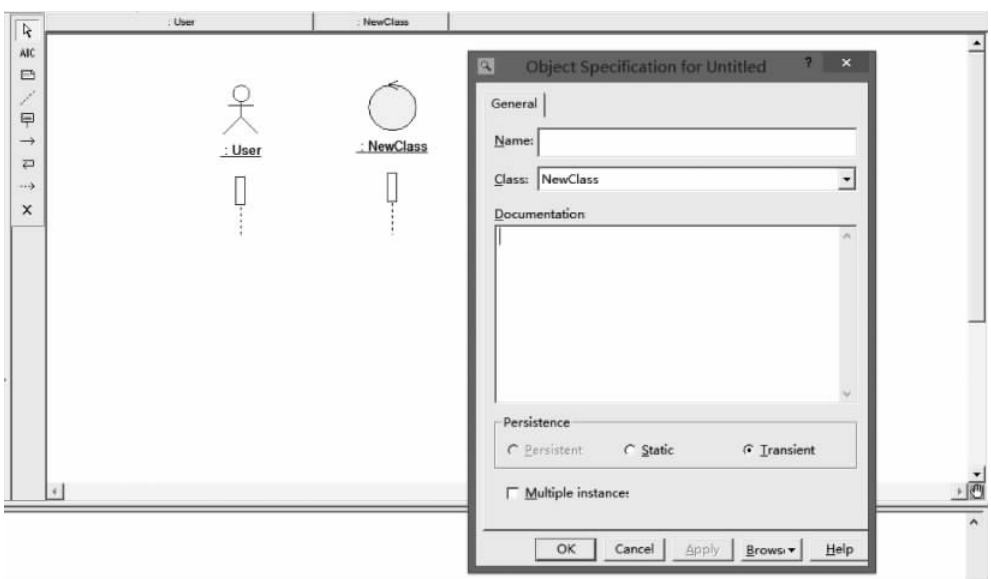


图 3-30　编辑对象属性

末端单击，出现的“╳”表示生命线终止，如图 3-33 所示。

(7) 结合前文中需求分析，重复步骤(2)～(6)，完成“用户登录”用例的顺序图绘制。参考绘图如图 3-34 所示。

### 3.3.4　构建活动图模型

活动图是 UML 行为图的一种，主要用于描述动作及动作之间的关系。它与状态图类似，但是略有区别。活动图的主要目的在于描述活动和活动的结果，以及动作状态的转换。

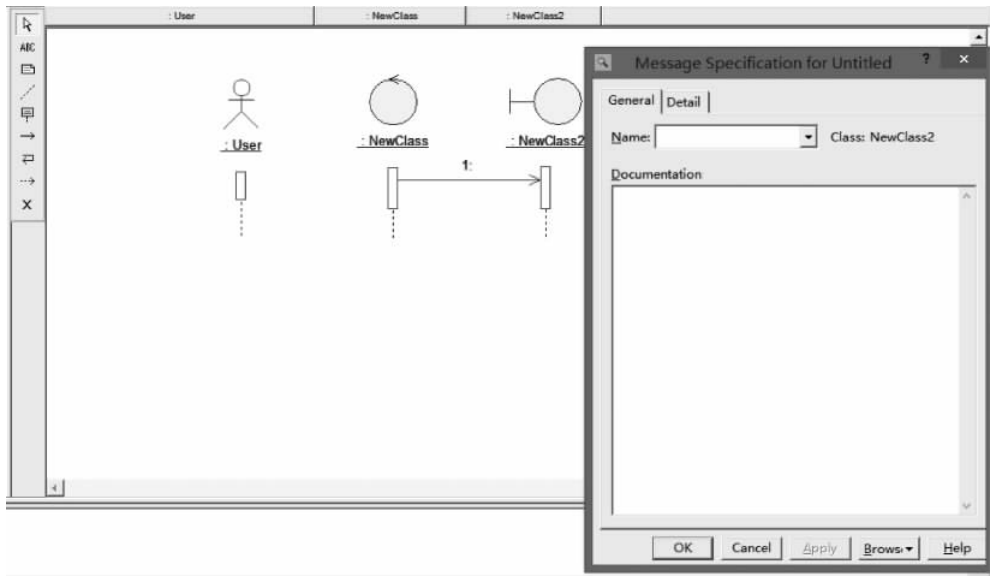


图 3-31　对象消息

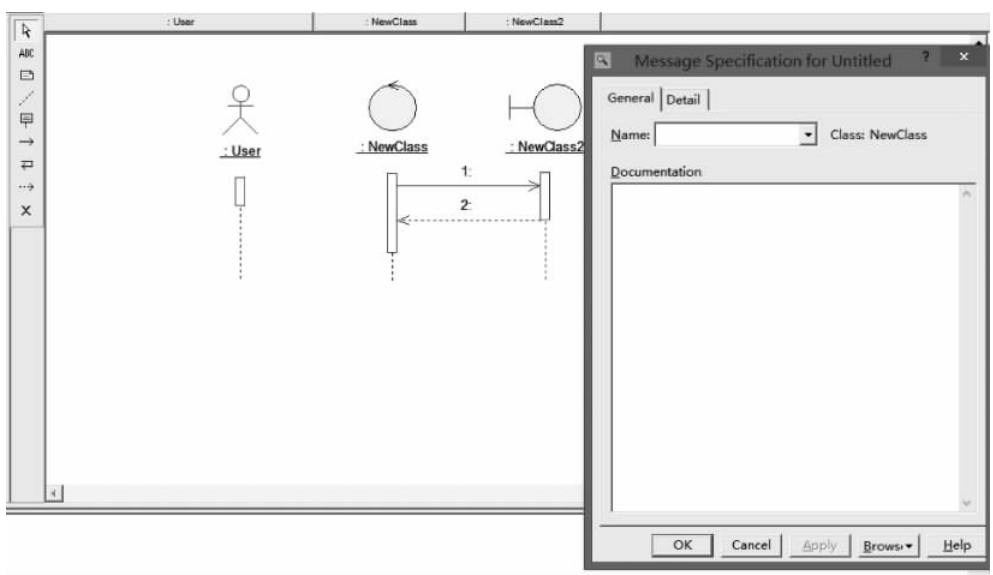


图 3-32　反馈消息

活动图中的动作可以放入“泳道”，表示一组活动，“泳道”用矩形表示和分隔。

活动图的主要元素包括：开始状态、活动、分支、同步、泳道和终止状态。开始状态用实心圆表示，活动用圆角矩形表示，分支用菱形表示，同步用直线段表示，泳道用矩形表示，终止状态用同心圆表示。

下面以“小型二手货交易平台”的“交易子系统”为例，说明如何利用 Rational Rose 绘制活动图。

下面为绘制活动图的实验步骤。

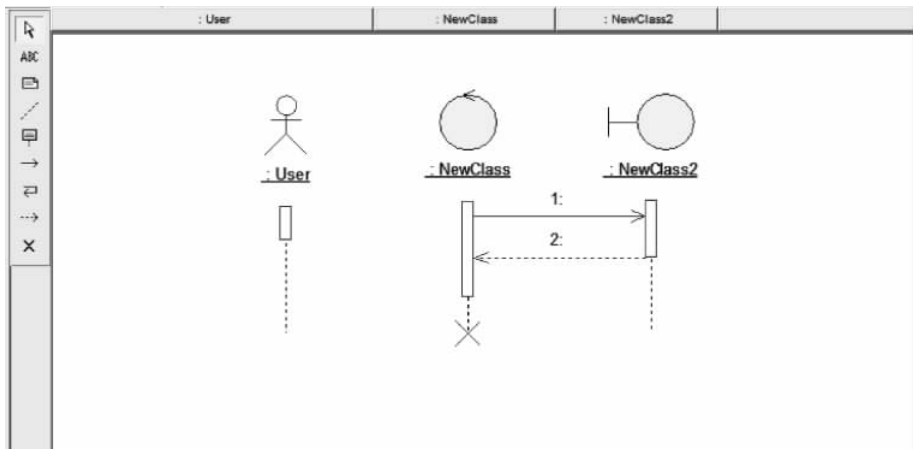


图 3-33　生命终止符

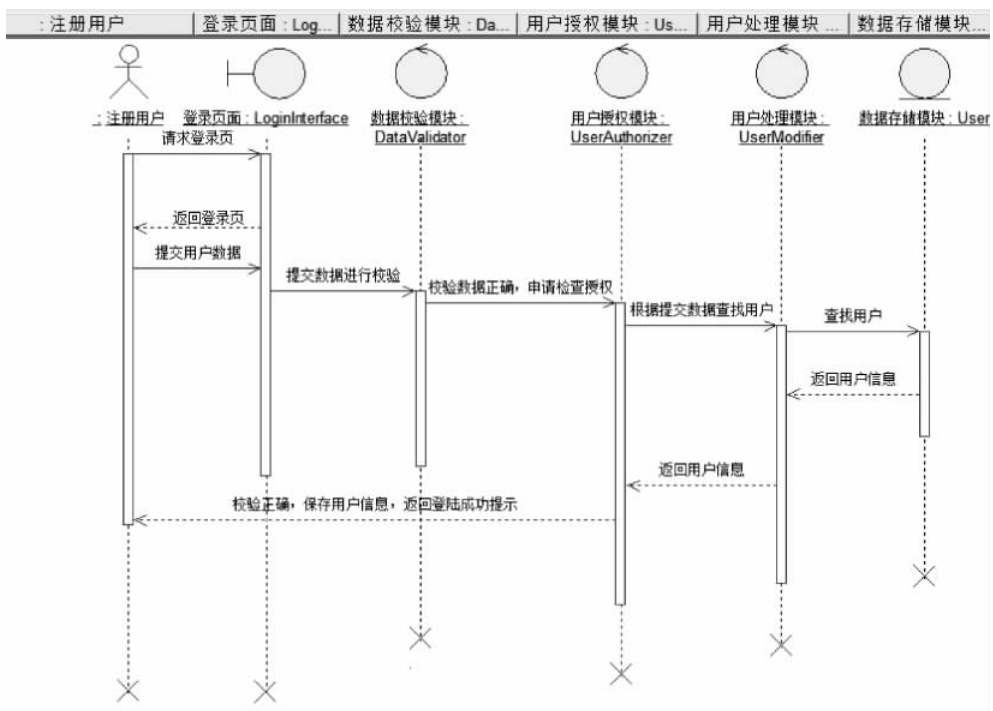


图 3-34　顺序图

(1) 打开 Rational Rose，在浏览器窗口中的 Logical View 右键选择 New→Activity Diagram，新建一个活动图。双击在模型视图窗口中打开活动图，如图 3-35 所示。

(2) 单击选中工具箱中的 Swimlane 图标，拖动至活动图的绘图区域放入，建立起第一个泳道，单击泳道上方的文字可以编辑泳道名称。重复此操作，建立第二个泳道，如图 3-36 所示。

(3) 单击选中工具箱中的开始状态图标，拖动至第一泳道中，可调整其位置与大小，表示活动的开始。再单击选中工具箱中的 Activity 图标，拖动至泳道中，可调整此活动的位置

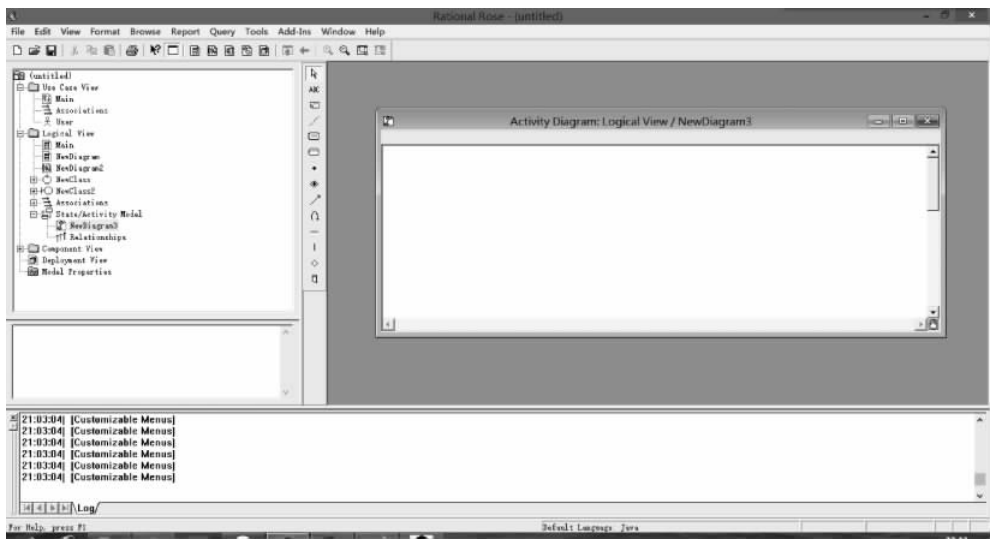


图 3-35　新建活动图

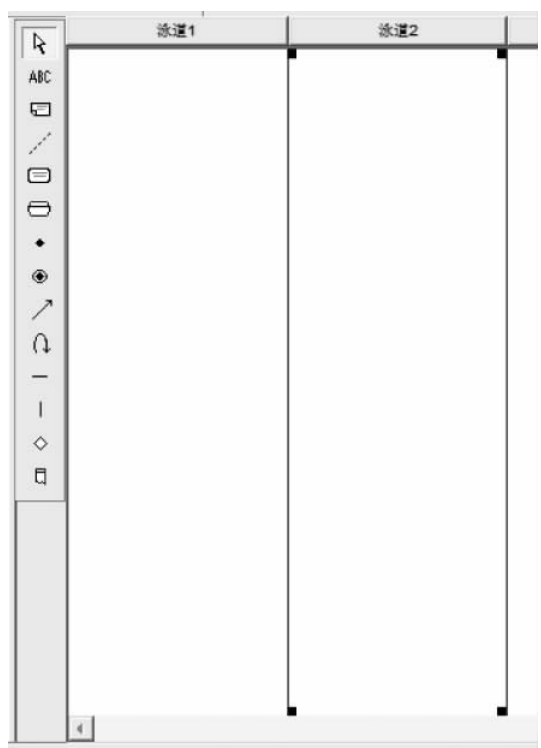


图 3-36　泳道

与大小。右键选择 Open Specification 可以编辑该活动的属性。重复此操作可添加多个活动，如图 3-37 所示。

(4) 单击选中工具箱中的 State Transition 图标，拖动至泳道中的两个活动间或开始状态与活动间，表示转换关系。再单击选中工具箱中的 Decision 图标，拖动至泳道中，可调整

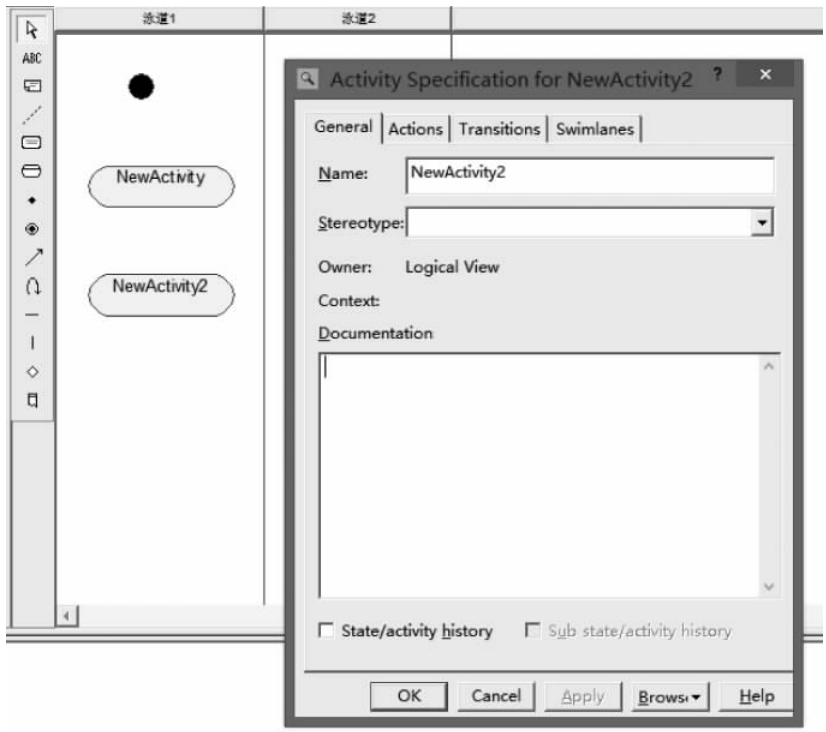


图 3-37　编辑活动属性

该分支的位置与大小，并用转换线连接活动以表示分支决策关系。单击选中工具箱中的 Horizontal（Vertical）Synchronization 图标，拖动至泳道中，可调整水平或垂直同步的位置与大小，并用转换线连接活动以表示活动之间的同步关系，如图 3-38 所示。

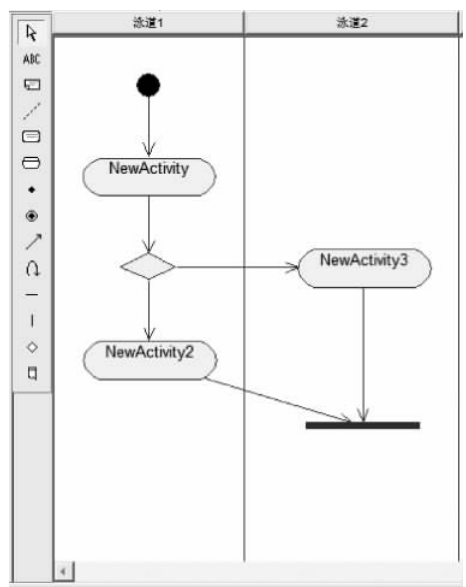


图 3-38　活动转换、分支与同步

(5) 单击选中工具箱中的终止状态图标，拖动至泳道中，可调整其位置与大小，表示活动的终止，如图 3-39 所示。

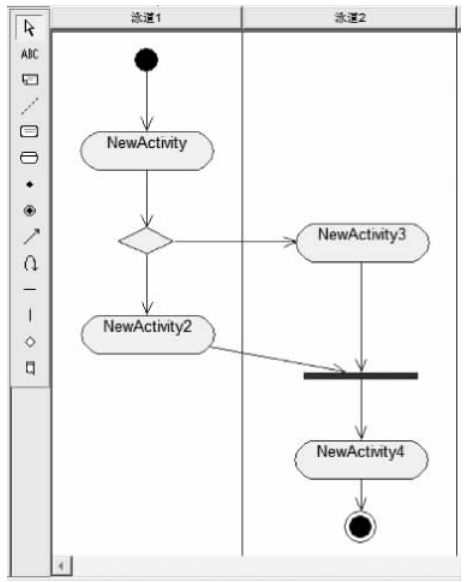


图 3-39　终止状态

(6) 结合前文中需求分析，重复步骤(2)～(5)，完成“交易子系统”的活动图绘制。参考绘图如图 3-40 所示。

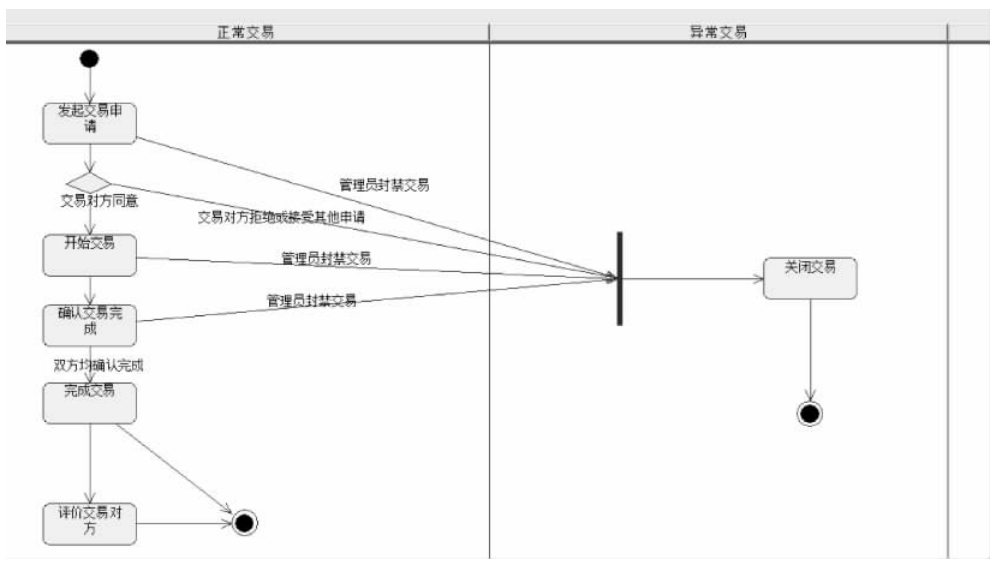


图 3-40　活动图

## 3.3.5　构建包模型

一个规模较大的软件系统往往包含复杂的组织结构。以面向对象的观点，系统拥有很

多类。为了清晰地描述系统的层级结构和逻辑架构，可以建立系统的包模型。一个包是由一组互相关联的类组成，它们可以共同完成系统的一些用例。

下面以“小型二手货交易平台”为例，说明如何利用 Rational Rose 构建系统的包模型。具体的实验步骤如下。

（1）打开 Rational Rose，在浏览器窗口中的 Logical View 右键选择 New→Class Diagram，新建一个类图。双击在模型视图窗口中打开类图，如图 3-41 所示。

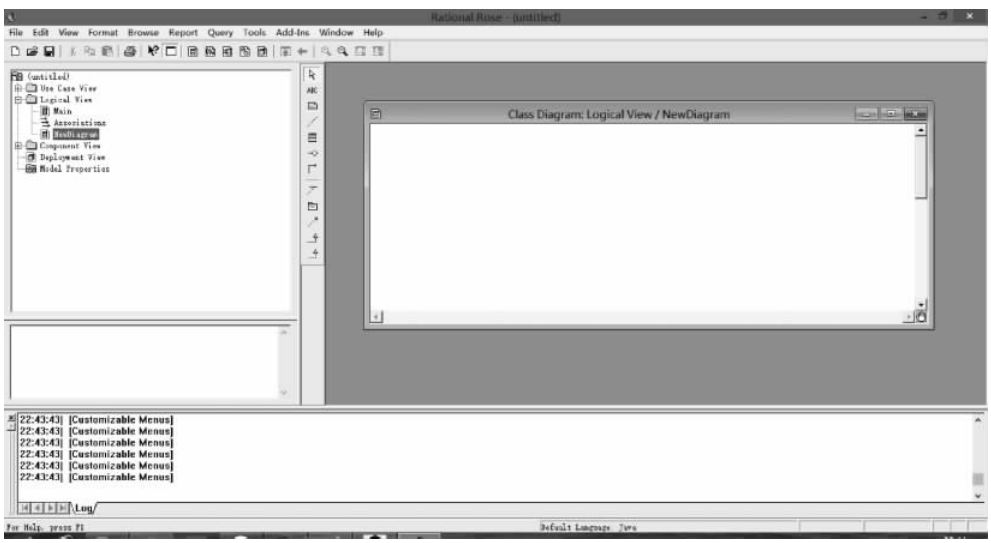


图 3-41　新建类图

（2）单击选中工具箱中的 Package 图标，拖动一个包至绘图区域，右键选择 Open Specification 可以编辑其属性，如名称和 Stereotype 等。Stereotype 包括子系统（subsystem）和层次（layer）等，如图 3-42 所示。

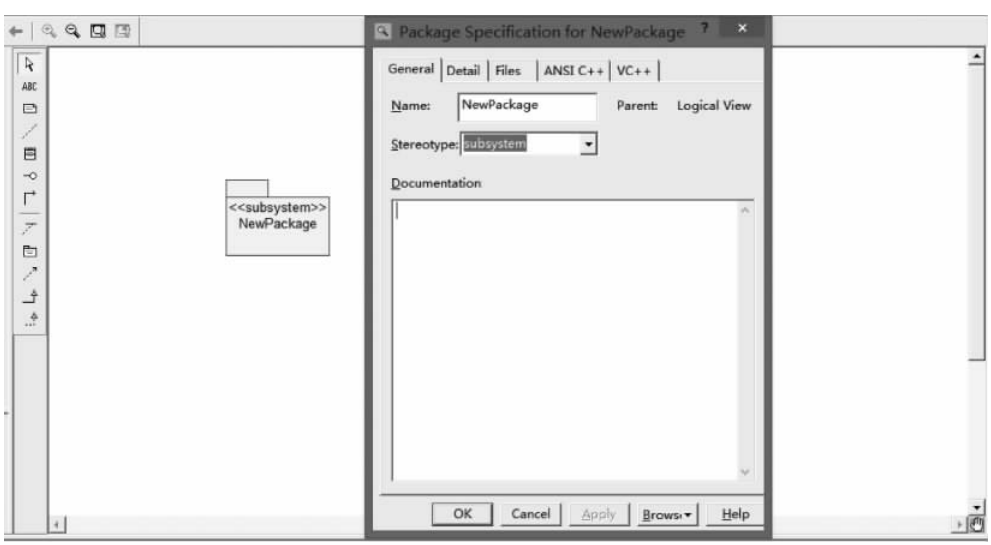


图 3-42　编辑包的属性

(3) 重复步骤(2),可以添加多个包到模型中。单击选中工具箱中的 Interface 图标,拖动一个接口至绘图区域,右键选择 Open Specification 可以编辑其属性。接口可作为包之间的数据交换标准和关联,在 Operations 选项卡下右键选择 Insert,创建接口的操作,如图 3-43 所示。

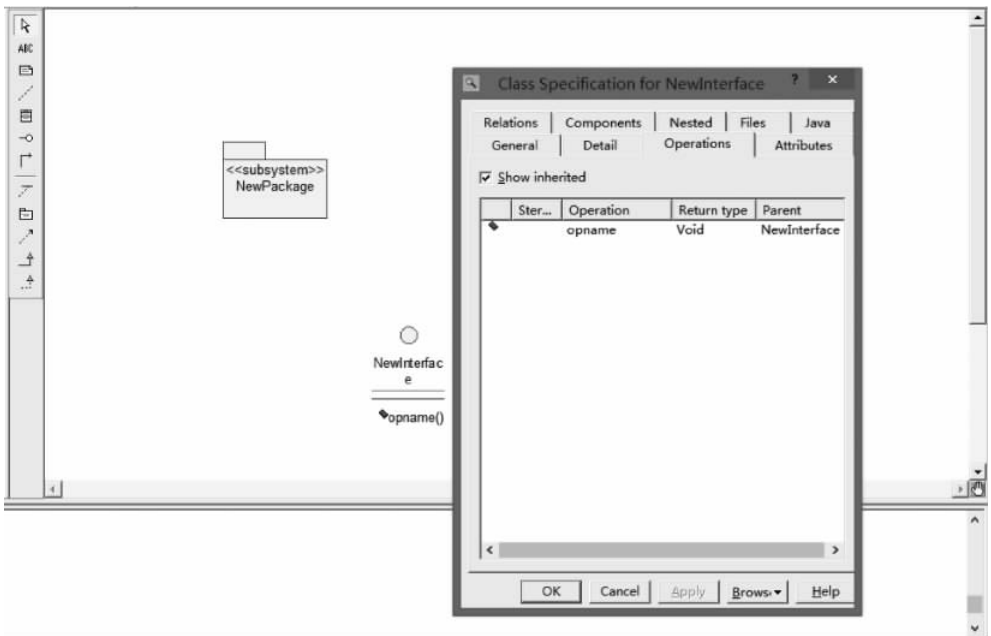


图 3-43　编辑接口属性

(4) 单击选中工具箱中的 Realize 图标,拖动至模型中的一个包和接口之间,表示包实现了此接口。再单击选中工具箱中的 Dependency or Instantiates 图标,拖动至模型中的一个包和接口之间,表示包依赖或使用了此接口,如图 3-44 所示。

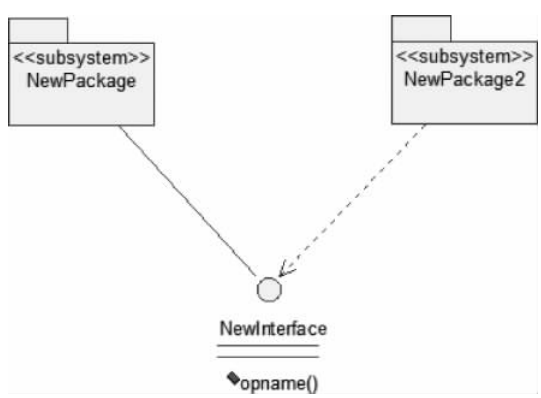


图 3-44　包和接口

(5) 结合前文中需求分析,重复步骤(2)～(4),完成“小型二手货交易平台”的包模型。参考绘图如图 3-45 所示。

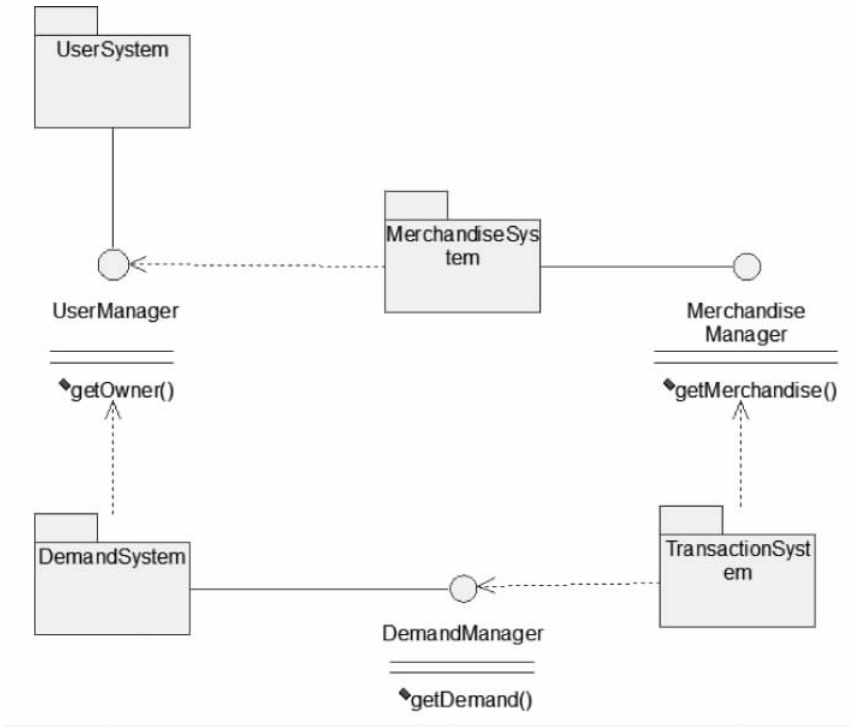


图 3-45　包图

### 3.3.6　构建构件模型

构件模型一般表示为系统的构件图，它描述了软件构件及构件之间的依赖关系，显示代码的静态结构。构件是逻辑架构中定义的概念和功能（类、对象和之间的关系）在物理架构中的实现。软件构件一般包括：源代码构件、二进制构件和可执行构件。

构件图中的构件是类型而不是实例，为显示构件的实例必须使用部署图（在 3.4 节中介绍）。在 UML 中，构件图的构件用带有椭圆和小矩形的大矩形表示，构件间的依赖关系用带箭头的虚线表示。

下面以“小型二手货交易平台”为例，说明如何利用 Rational Rose 构建系统的构件图。具体的实验步骤如下。

（1）打开 Rational Rose，在浏览器窗口中的 Component View 右键选择 New→Component Diagram，新建一个构件图。双击在模型视图窗口中打开构件图，如图 3-46 所示。

（2）单击选中工具箱中的 Component 图标，拖动一个构件至绘图区域，可调整其位置与大小，右键选择 Open Specification 可以编辑其属性，如名称和 Stereotype 等。Stereotype 决定了构件的类型，如主程序（Main Program）、子程序（Subprogram Body）、包（Package）、任务（Task Body）等，如图 3-47 所示。

（3）重复步骤（2），可以添加多个构件到模型中。单击选中工具箱中的 Dependency 图标，拖动至绘图区域的两个构件之间，表示构件之间的依赖关系，如图 3-48 所示。

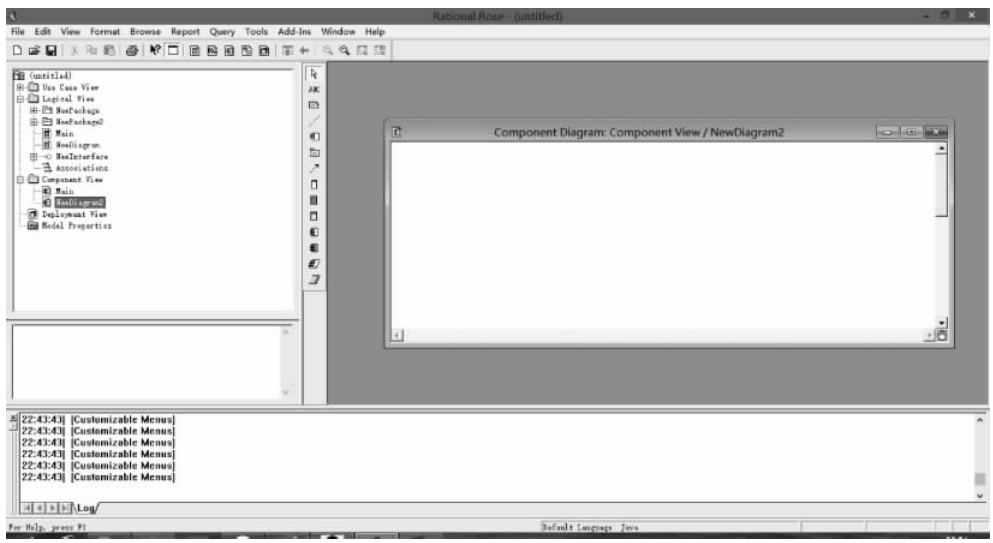


图 3-46　新建构件图

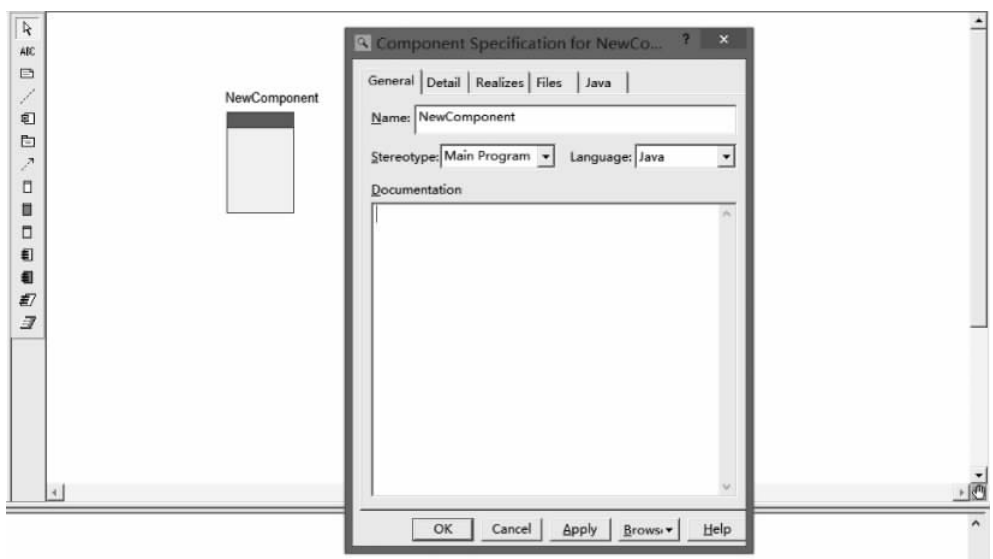


图 3-47　编辑构件属性

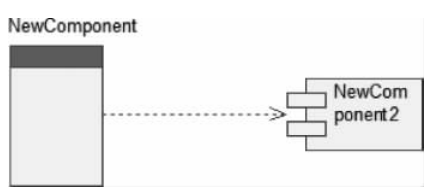


图 3-48　构件之间的关系

(4) 结合前文中的需求分析,重复步骤(2)、(3),可以完成“小型二手货交易平台”的构件图。参考绘图如图 3-49 所示。

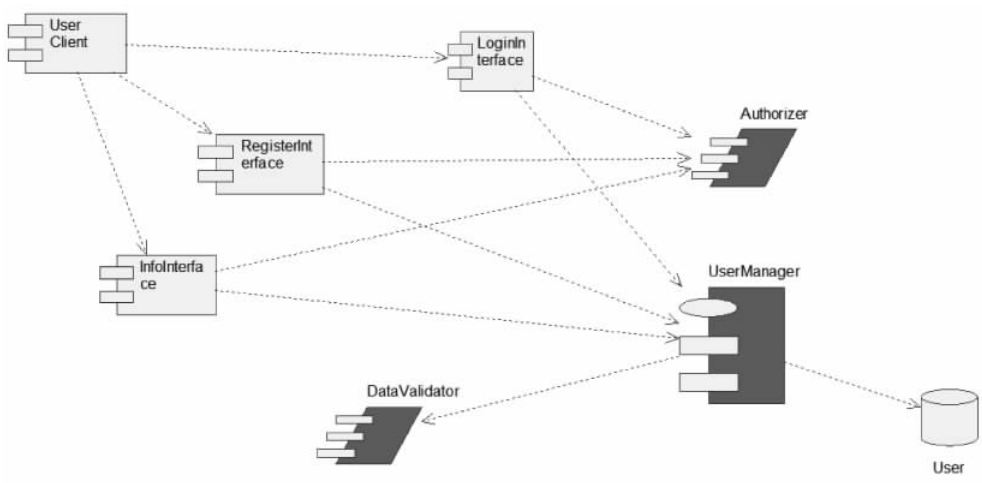


图 3-49　构件图

## 3.4　利用 Rational Rose 进行“小型二手货交易平台”系统部署架构设计

系统的部署架构设计一般用部署图来建模。在 UML 中，部署图是一种实现图，描述处理器、硬件设备和软件构件在运行时的架构，它显示系统硬件的物理拓扑结构及在此结构上执行的软件。2.5 节已对部署图做过简要介绍，这里不再赘述。

下面以“小型二手货交易平台”为例，说明如何利用 Rational Rose 构建系统的部署图。具体的实验步骤如下。

(1) 打开 Rational Rose，在浏览器窗口中的 Deployment View 双击打开部署图，如图 3-50 所示。

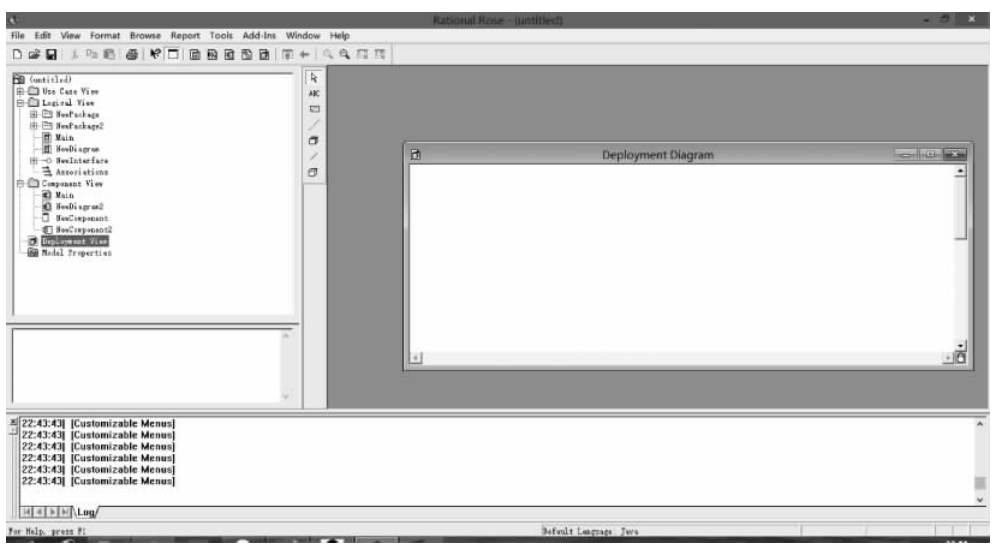


图 3-50　新建部署图

（2）单击选中工具箱中的 Processor 图标，拖动一个处理器至绘图区域，可调整其位置与大小，右键选择 Open Specification 可以编辑其属性，如名称和 Stereotype 等，如图 3-51 所示。

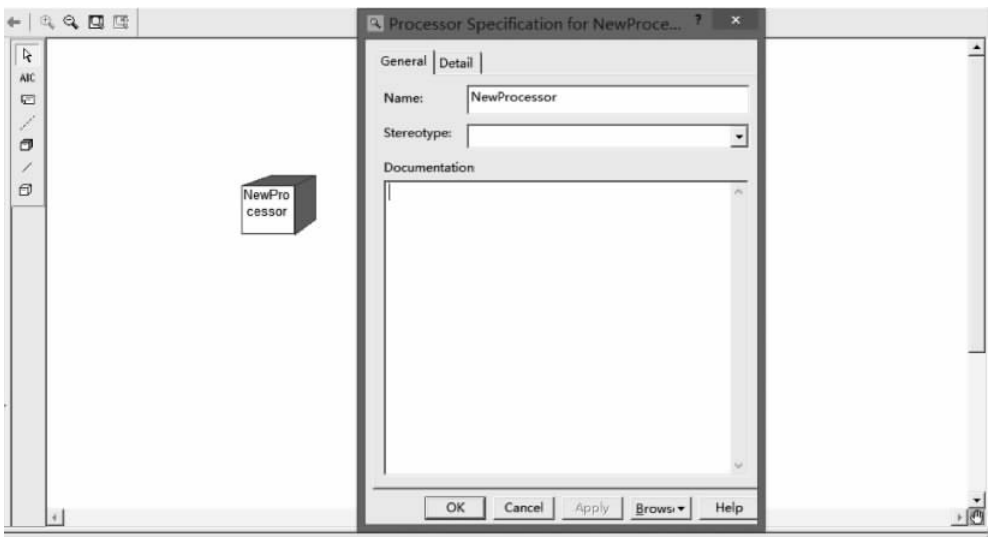


图 3-51　编辑处理器属性

（3）单击选中工具箱中的 Device 图标，拖动一个设备至绘图区域，可调整其位置与大小，右键选择 Open Specification 可以编辑其属性，如名称和 Stereotype 等，如图 3-52 所示。

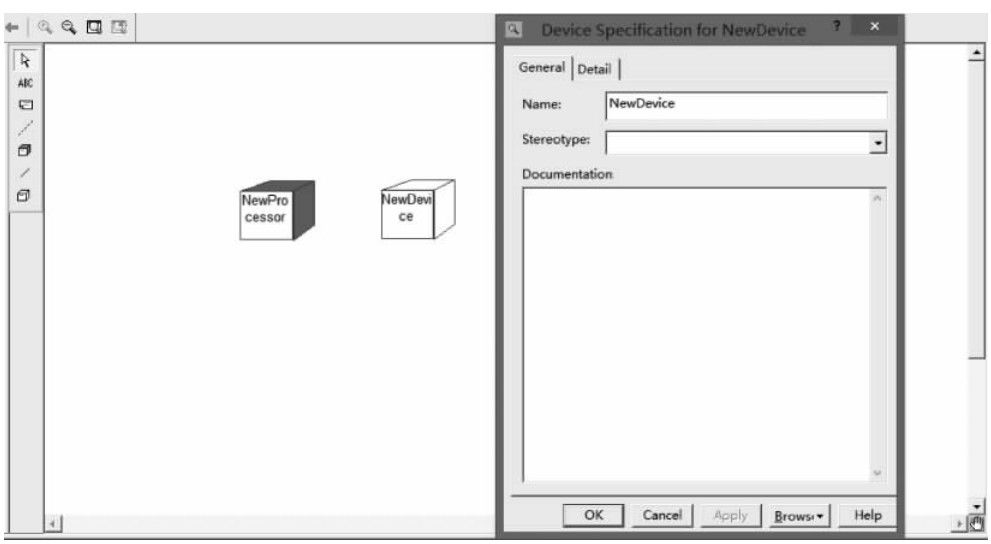


图 3-52　编辑装置的属性

（4）单击选中工具箱中的 Connection 图标，在绘图区域的两个处理器节点之间创建连接，表示节点之间的关系。右键选择 Open Specification 可以编辑其属性，如名称和 Stereotype 等，如图 3-53 所示。

（5）单击选中工具箱中的 Note 图标，在绘图区域的合适位置放入，可以为节点添加注

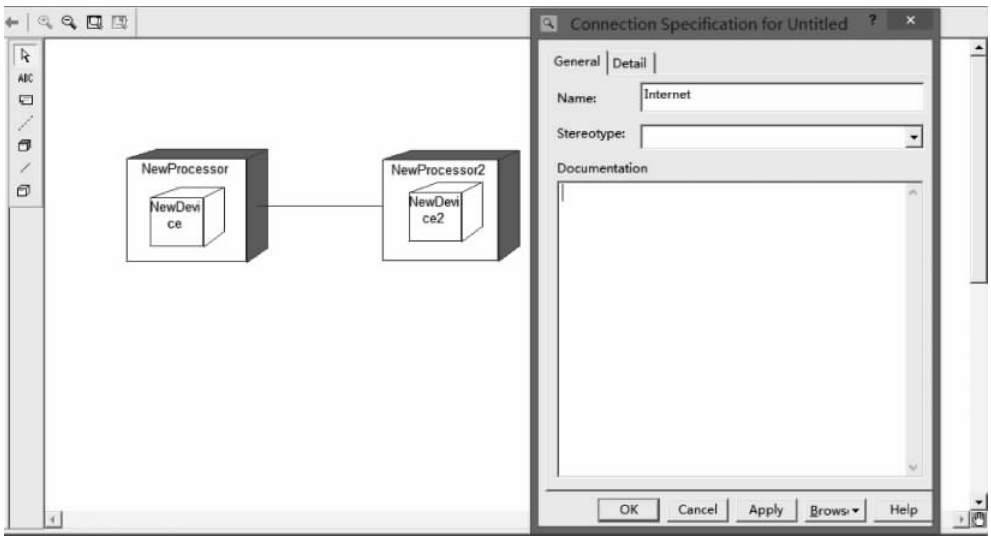


图 3-53　编辑节点间的关系

释和说明。单击此 Note 可以编辑注释文字，并用工具箱中的 Anchor Note To Item 连接 Note 和节点，如图 3-54 所示。

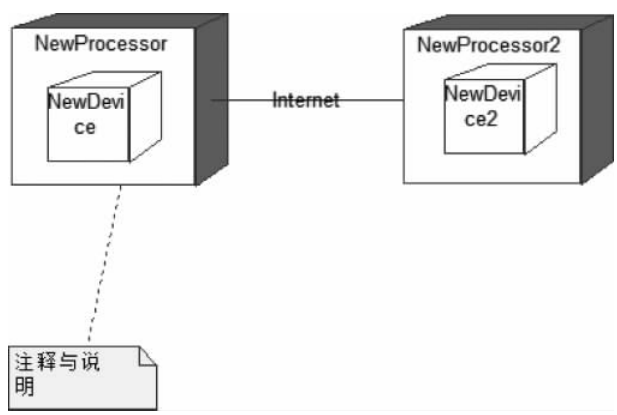


图 3-54　添加注释

(6) 结合前文中的需求分析，重复步骤(2)～(5)，可以完成“小型二手货交易平台”的部署图。参考绘图如图 3-55 所示。

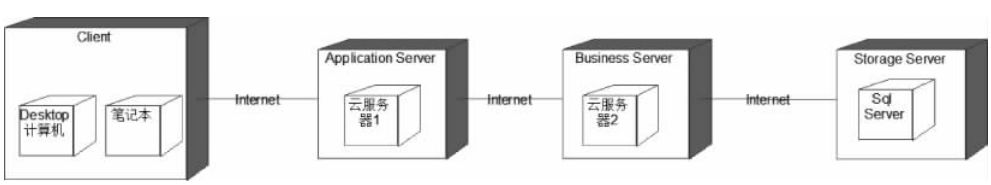


图 3-55　部署图

## 3.5　双向工程

Rational Rose的双向工程包括正向工程和逆向工程。正向工程是从模型生成代码的过程，而逆向工程是从代码到模型的过程。Rational Rose支持的语言包括C++、Visual C++、Java、Smalltalk、Ada等，因此可以用模型生成上述语言的代码。Rational Rose还能够为CORBA应用产生接口定义语言(Interface Description Language，IDL)并为数据库应用产生数据库描述语言(Data Description Language，DDL)。本节以Java语言为例，对Rational Rose的双向工程方法进行实践。

### 3.5.1　正向工程

正向工程(Forward Engineering)是通过从设计层面到实现语言的映射而把模型转换为代码的过程。在Rational Rose中，这是通过将利用UML描述的设计模型转换为编码(诸如Java、C++等)实现的。由于这种正向工程为自动化过程，因此可以节省大量编写类、属性、方法代码等的时间。在Rational Rose中，正向工程生成的代码生成元素主要有类(Class)、属性(Attribute)、操作(Function)、关系(Relation)、组件(Component)、文档(Document)等。正向工程在Rational Rose中的基本实现步骤为：建立模型→语法检查→设置代码生成属性→代码生成。

下面以“小型二手货交易平台”为例，说明如何利用Rational Rose进行正向工程，并从用户子系统的类图生成代码框架。具体的实验步骤如下。

(1) 打开Rational Rose，导入保存有用户子系统类图的.mdl文件，在浏览器窗口中的Logical View双击打开类图，如图3-56所示。

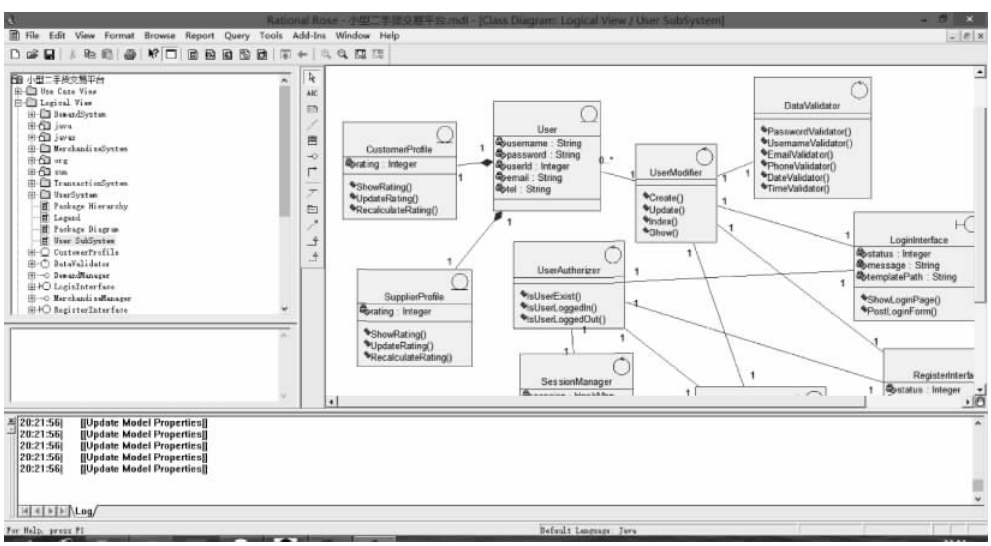


图3-56　导入类图

(2) 在工具和菜单栏中选择Tools→Options，选择Notation选项卡并设置默认语言为Java，如图3-57所示。


图 3-57　选择语言

(3) 在工具和菜单栏中选择 Tools→Java/J2EE→Project Specification，在 Code Generation 选项卡下可以编辑代码生成属性。在 Automatic Synchronization 标签下勾选 Automatic Synchronization Mode。在 Classpath 选项卡下可以添加路径作为代码框架保存路径，如图 3-58 所示。

(4) 选中类图模型的所有元素，在工具和菜单栏中选择 Tools→Java/J2EE→Syntax Check 进行语法检查。若正确应弹出提示“Syntax checking completed successfully”的对话框，如图 3-59 所示。

(5) 在工具和菜单栏中选择 Tools→Java/J2EE→Generate Code，在弹出的对话框的右侧选择模型元素，左侧选择代码保存的路径。单击 Assign 按钮可以将模型元素转换为代码框架，完成后单击 OK 按钮进行确认，如图 3-60 所示。

(6) 完成上述步骤后，可以对生成的代码进行查看和编辑。第一种方法是选中类图模型的所有元素，在工具和菜单栏中选择 Tools→Java/J2EE→Edit Code，在下方弹出的代码窗口中可以查看并编辑生成的代码框架。第二种方法是在指定的保存路径下用外部编辑器或 IDE 查看和编辑生成的代码文件(Java 文件)，如图 3-61 所示。

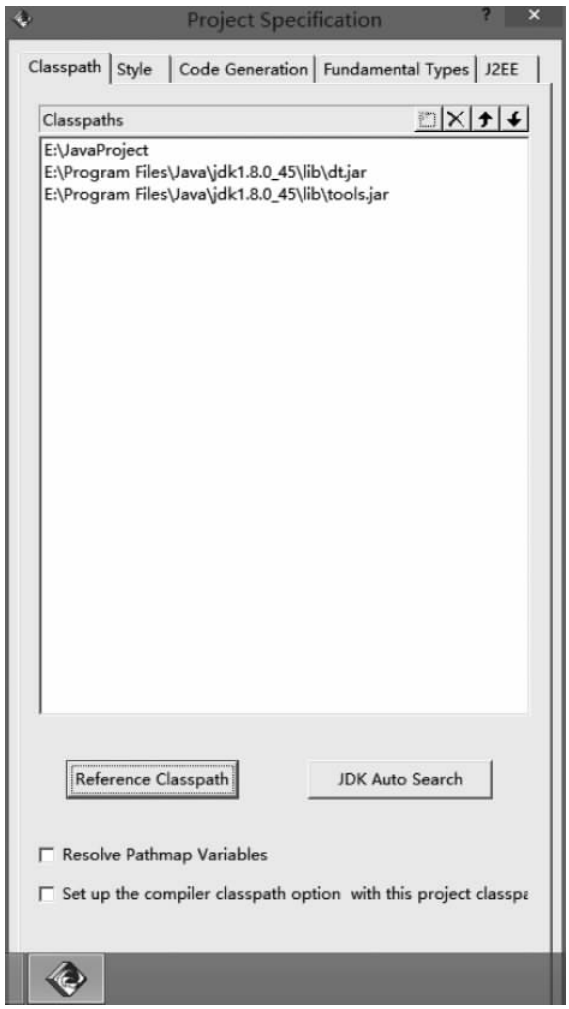


图 3-58　设置 Classpath

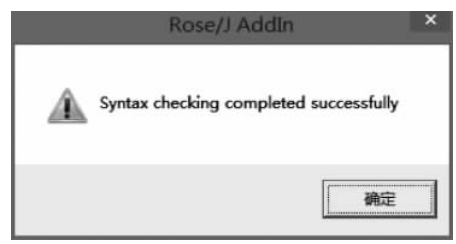


图 3-59　语法检查

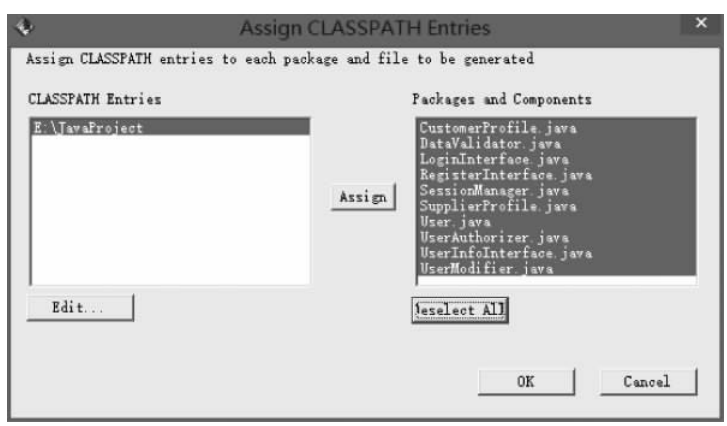


图 3-60　正向工程

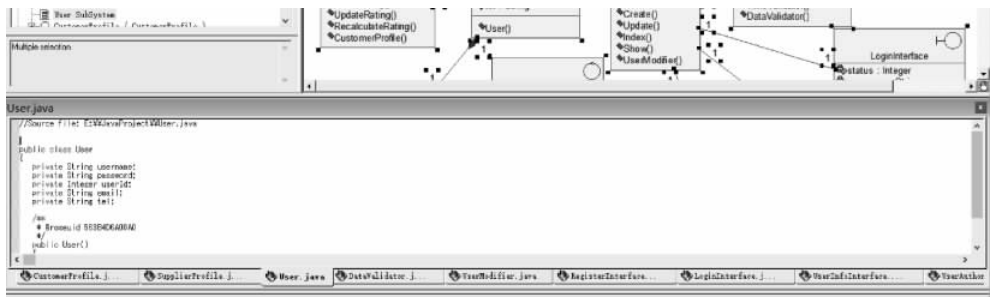


图 3-61　查看与编辑代码

## 3.5.2　逆向工程

与正向工程相对，逆向工程(Reverse Engineering)是通过从特定实现语言向设计层面的映射而把代码转换为模型的过程。在 Rational Rose 中，通常可以将无语法错误且符合内置语言标准的代码(诸如 C++、Java 等)转化为模型设计，从而便于诸如在缺少前一轮迭代时的设计模型等情况下的后续软件设计与开发工作。Rational Rose 通过收集类、属性、操作、关系、组件、文档等的信息来实现逆向工程。在 Rational Rose 中，逆向工程的一般步骤为：环境变量配置→由代码生成模型。

下面以“小型二手货交易平台”为例，说明如何利用 Rational Rose 进行逆向工程，并从代码转换为模型。具体的实验步骤如下。

(1) 打开 Rational Rose，在浏览器窗口中的 Component View 右键新建一个构件。右键单击构件选择 Open Standard Specification，可以编辑其属性。在 General 选项卡下选择语言为 Java，如图 3-62 所示。


图 3-62　编辑构件属性

（2）右键单击构件，选择 Java/J2EE→Reverse Engineer，在弹出的逆向工程窗口中可以选择待转换目录下的代码文件，如图 3-63 所示。

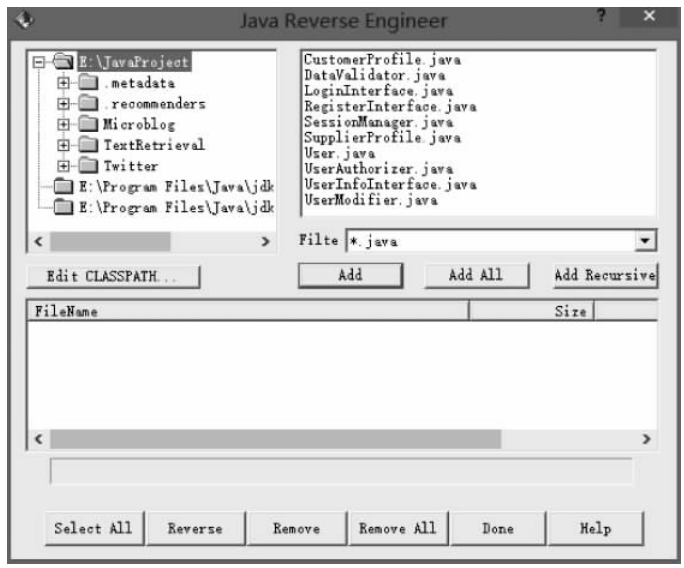


图 3-63　选择待转换的代码文件

（3）在该窗口的左侧是 Classpath，选择保存有代码文件的路径。右侧是对应路径下的代码文件。选择待转换的代码文件，单击 Add 按钮，添加到下方列表中。在列表中选中所要转换的代码文件，单击 Reverse 按钮进行逆向工程转换，完成后单击 Done 按钮进行确认，如图 3-64 所示。

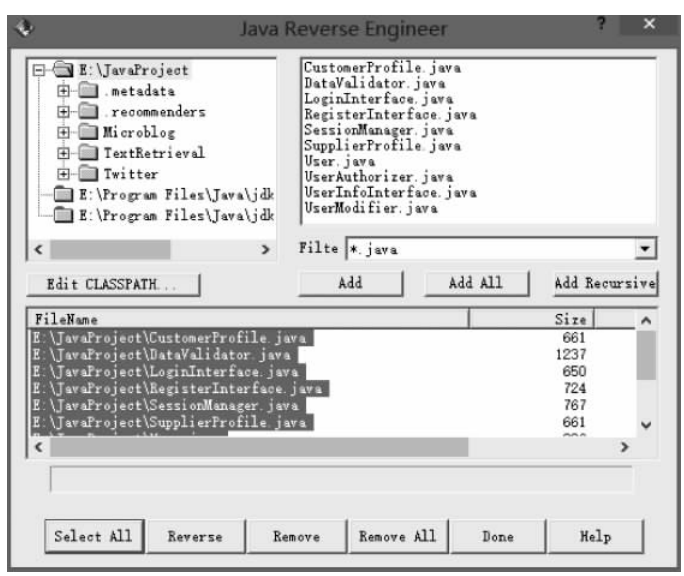


图 3-64　逆向工程

(4) 步骤(3)结束后,在 Logical View 和 Component View 下出现了转换生成的模型元素。在各自视图下新建类图或构件图,单击工具和菜单栏中的 Query→Add Classes(或 Add Components)。在弹出的对话框左侧选择对应视图下的模型元素,单击“<<>>”按钮添加到右侧的列表中。单击 OK 按钮后即可将模型元素添加并显示在对应的类图或构件图中,如图 3-65 所示。

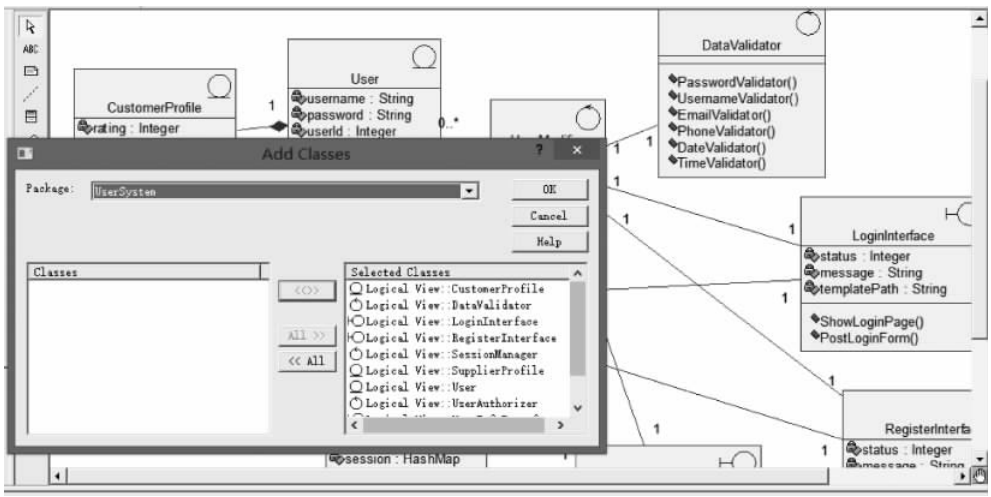


图 3-65 添加转换生成的模型元素到图中

# 小结

Rational Rose 是一款功能强大的自动化面向对象软件设计工具。通过该工具,用户可以使用 UML 便捷地建立诸多模型视图,诸如用例模型、类模型、活动模型、状态模型、顺序图、协作图、组件图与部署图等,并将这些模型设计结果通过正向工程转化为目标开发代码。用户还可以利用 Rational Rose 的逆向工程模块将规范化的代码转化为模型,以便进行后续的设计与开发。

本章介绍了利用 Rational Rose 进行面向对象软件设计的基本方法,包括用例模型、逻辑模型、活动模型、包模型、组件模型与部署模型的设计与相关图形的绘制,并简要介绍了利用双向工程实现代码与设计的双向转换的方法。为了便于展现实践过程,本章仅使用了“小型二手货交易平台”的部分设计进行实践,读者可以参考需求,利用 Rational Rose 设计其余模块,以加深对实践过程的印象。

# 思考题

1. 通常需求规格说明书中使用用例说明表格对一个用例进行描述,分析每一部分的作用何在,如果使用顺序图替代文字性的用例流程说明、用例替换流程说明,会带来什么好处与不足?(用例说明范例详见《“小型二手货交易平台”需求规格说明书》)。

2. 试分析顺序图与结构化软件设计中的流程图有何异同。

3. 面向对象软件设计中的逻辑模型(UML 静态结构)与数据模型的关系如何？如何将逻辑模型转换为数据模型？

4. 在实际软件开发中，一部分情况下即便不使用包模型，也不会对软件开发造成很大影响，试分析出现这种情况的原因，并探讨包模型对软件开发的意义。

5. UML 静态模型与动态模型的区别在哪里？两者各作用于软件设计的什么方面？哪些 UML 模型属于静态模型？哪些属于动态模型？

6. 是否能够假设：只要拥有足够完善的设计方案，便可直接利用正向工程自动化生成软件代码，从而避免人为编码工作？试分析该假设成立或不成立的原因。

## 实验练习题

1. 使用 Rational Rose 绘制“小型二手货交易平台”需求子系统的用例图，并分析用例流程。

2. 基于问题(1)中绘制的用例图与归纳的用例流程，利用 Rational Rose 绘制需求子系统的类图，并组织包图。

3. 基于问题(1)与问题(2)的绘制结果，利用 Rational Rose 绘制需求子系统各用例的顺序图。

4. 基于前三题的绘制结果，综合考虑多方面因素，利用 Rational Rose 设计“小型二手货交易平台”需求子系统的数据模型。

5. 基于以上所有绘图与分析结果，利用 Rational Rose 绘制“小型二手货交易平台”的活动图。

6. 基于所有前五题的绘图结果，利用 Rational Rose 的正向工程功能创建软件项目。

## 参考文献

Rational Rose 是一种面向对象的统一建模语言的可视化建模工具，主要用于可视化建模和公司级水平软件应用的组件构造。Rose 的主要版本为 2003 和 2007，本章出于对兼容性的考虑，使用的版本为 2007。IBM 的官方网站对 Rational Rose 有简单的产品介绍[1]。尽管目前 Rose 已经没有更新版本，但是其作为功能强大的 UML 建模工具，仍被一些企业使用。在 Rational Rose 的基础上，IBM 推出了新的产品 Rational Software Architect。IBM 官方网站对 Rational Software Architect 的介绍见参考文献[2]，用户可在参考文献[3]中了解或购买最新产品。

本章对 UML 建模进行了介绍，IBM 对 UML 也提供了一些资源和技术的支持[4]。本章基于面向对象的软件设计理念进行实验设计，面向对象软件设计理论的相关书籍参考文献[5～7]。

[1] IBM-Rational Rose Enterprise. http://www-03. ibm. com/software/products/zh/enterprise, 2015-11-01.

[2] IBM-Rational Software Architect. http://www-03. ibm. com/software/products/zh/ratisoftarch,

2015-11-01.

[3] IBM-Rational Software Architect Product. http://www.ibm.com/developerworks/cn/downloads/r/architect/, 2015-11-01.

[4] IBM-developer Works. http://www.ibm.com/developerworks/cn/rational/uml/, 2015-11-01.

[5] 面向对象分析与设计. http://book.douban.com/subject/3892590/, 2015-11-02.

[6] 设计模式. http://book.douban.com/subject/1099305/, 2015-11-02.

[7] UML Distilled. http://book.douban.com/subject/1460848/, 2015-11-02.