

第1章 Rose入门



《Rational Rose 2003基础教程》

配套电子教案



内 容

- Rose与可视化建模
- Rational Rose工具简介
- Rational Rose 2003企业版的安装
- Rose应用程序界面
- Rose模型
- Rose视图



1 Rose与可视化建模

■ 1.1 理解可视化建模

- 是开发人员及其团队获得应用程序完整设计蓝图的理想方法，是理解复杂问题和相互交流的有效手段
- 可视化建模有助于软件开发人员：
 - 1) 可视化应用程序
 - 2) 说明应用程序的完整结构和行为
 - 3) 创建模板，引导应用程序的构建工作
 - 4) 将质量保证贯穿于整个开发生命周期
 - 5) 将开发过程中的所有决策信息整理归档



1 Rose与可视化建模

- 1.2 可视化建模工具的发展与UML
 - 常见的可视化建模方法：OMT、Booch、OOSE
 - UML：统一建模语言
 - 是OMG批准的标准建模方法
 - 集OMT、Booch、OOSE的优点于一身



1 Rose与可视化建模

■ 1.3 Rose: 优秀的可视化建模工具

- Rose占据了市场上可视化建模工具的主导
- 衡量可视化建模工具的标准:
 - 易于使用
 - 灵活性
 - 整合到应用程序生命周期的容易程度
 - 可伸缩性
 - 基于的标准
- Rose的优势
 - GUI
 - 允许在同一模型中使用多种构件、语言
 - 逆向工程
 - 团队管理
 - 数据建模和Web建模

1 Rose与可视化建模

■ 1.4 软件开发过程与Rose可视化建模

软件开发阶段	Rose使用情况	可能用到的Rose模型图及元素
开始阶段	建立业务模型（Business Use Case）	业务用例、业务参与者、业务工人
	确定用例模型（Use Case）	参与者、用例、关系
细化阶段	细化用例	参与者、用例、关系
	事件流程建模	顺序图、协作图、状态图
	对系统静态结构和动态行为建模	类图、交互图、状态机图
	确定系统构件	构件图、关系
构造阶段	正向工程产生框架代码	类图、交互图、状态机图、构件图
	逆向工程更新模型	构件图
	创建部署图	部署图
交付阶段	更新模型	构件图、部署图



2 Rational Rose工具简介

- 2.1 Rational Rose 2003特性
 - Rose 2003分企业版、专业版和Rose Modeler三种版本
 - Rose 2003支持的特性
 - 表1.2



2 Rational Rose工具简介

■ 2.2 Rose的基本功能

- 面向对象建模
- 用例分析
- 支持UML、COM、OMT和Booch'93
- 语义检查
- 支持可控的迭代开发
- 双向工程
- 支持多用户并行开发
- 可以与数据建模工具集成
- OLE链接、自动化
- 多平台可用性



2 Rational Rose工具简介

■ 2.3 Rose的插件及插件程序管理器

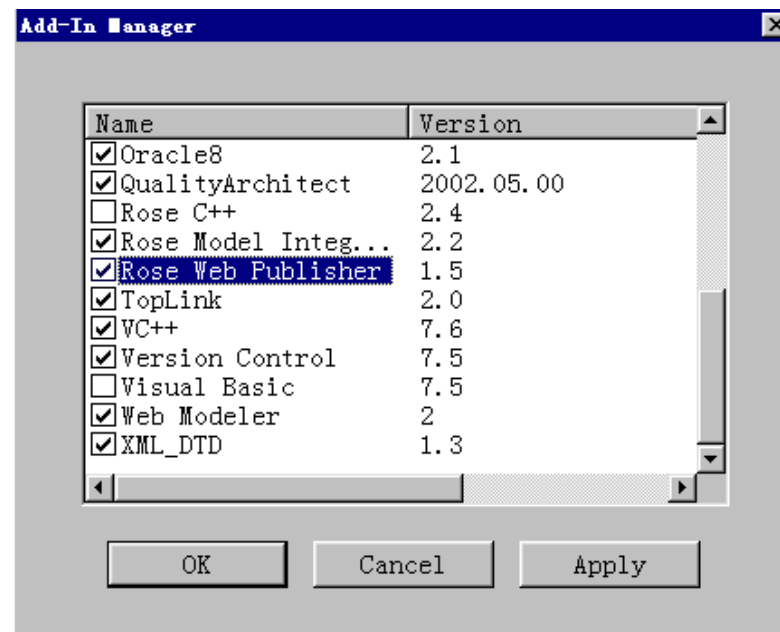
– 插件包括：

- Framework Wizard
- Rose Ada、ANSI C++、C++、VC ++、VB、Java、CORBA、Java、Oracle8、XML DTD
- Rose Data Modeler
- Rose model Integrator
- Rose Web Publisher
- Rose TOPLink Wizard
- Rose Web Modeler
- Quality Architect、Version Control

2 Rational Rose工具简介

■ 2.3 Rose的插件及插件程序管理器

- 插件程序管理器（Add-In Manager）
 - 控制插件程序的状态：激活状态或非激活状态





3 Rational Rose 2003企业版的安装

■ 3.1 系统要求

项目	需求
处理器	150MHz以上奔腾兼容机
内存	128MB（推荐128MB）
硬盘空间	400MB（建议留出最小200MB的交换空间）
其他	SVGA兼容显卡（建议256色以上），分辨率800×600，鼠标（建议两轮以上）



3 Rational Rose 2003企业版的安装

■ 3.2 安装过程

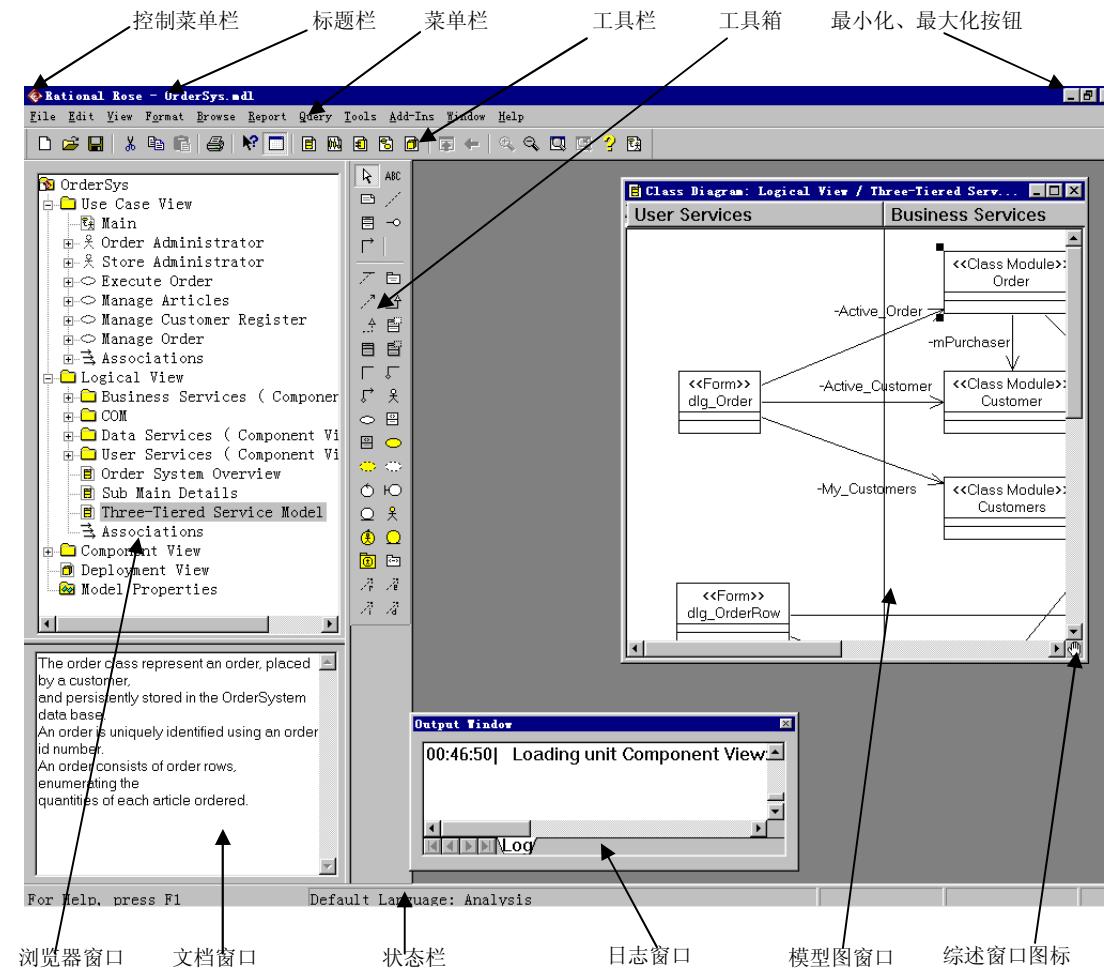
- 安装向导（引导安装）
- 安装主程序
- 配置许可协议



4 Rose应用程序界面

- Rose应用程序界面组成
 - 应用程序窗口
 - 显示载体
 - 浏览器窗口
 - 模型图超出屏幕显示范围时用于选择观察区域
 - 文档窗口
 - 记录用户操作和模型元素信息的辅助提示窗口
 - 模型图窗口
 - 用于建立和修改当前模型的图形化视图
 - 规范窗口

Rose应用程序界面组成





4 Rose应用程序界面

■ 4.1 Rose应用程序窗口

- 控制菜单栏
- 标题栏
- 最小化和最大化按钮
- 菜单栏
- 工具栏
- 日志窗口
- 工具箱
- 状态栏



4 Rose应用程序界面

■ 4.2 工具栏和工具箱

- 标准工具栏（**Standard Toolbar**）
 - 本书中简称为工具栏
 - 与打开的模型图窗口无关，包含一系列可以简化常用操作的图标，如创建新模型、保存模型等
- 模型图工具栏（**Diagram Toolbar**）
 - 本书中简称为工具箱
 - 包含适用于当前模型图的工具，每种模型图都有自己的工具箱
- 自定义工具栏/工具箱



4 Rose应用程序界面

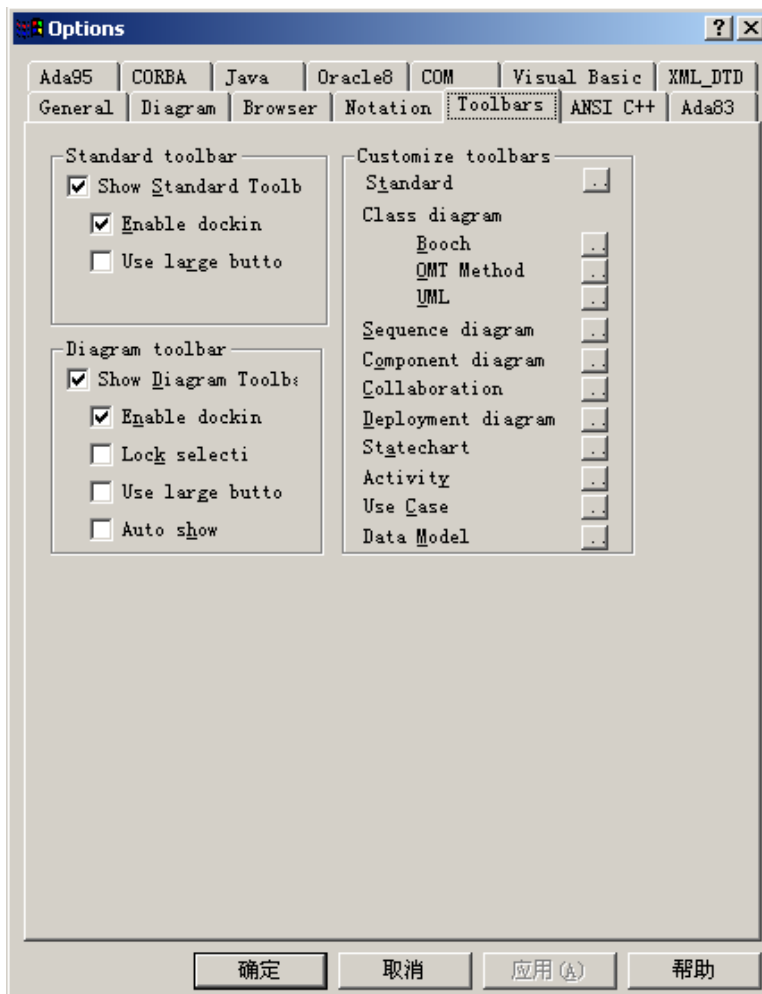
■ 4.2 工具栏和工具箱

— 自定义工具栏/工具箱

- 单击菜单栏中的**Tools>Options**，在弹出的**Options**窗口中单击**Toolbars**标签，可以在**Toolbars**标签中自定义工具栏/工具箱
- 右单击工具栏/工具箱，单击快捷菜单中的**Customize**

4 Rose应用程序界面

— 自定义工具栏/工具箱





4 Rose应用程序界面

■ 4.3 文档窗口

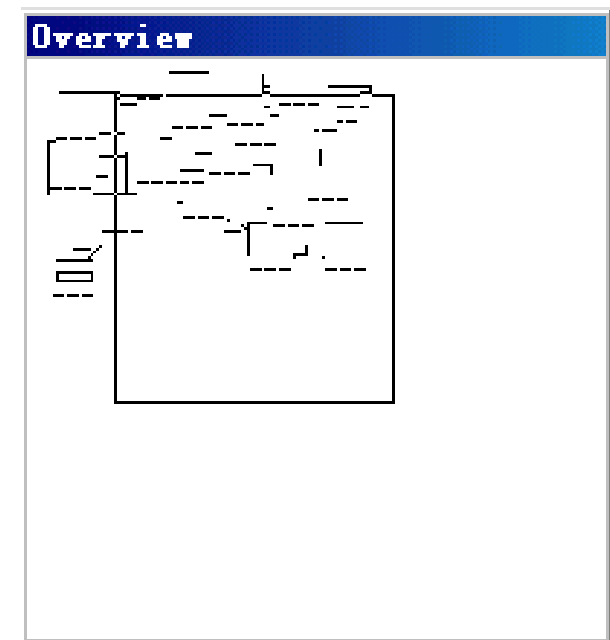
- 包含与模型元素规范窗口中完全相同的信息，描述模型元素或者关系，描述角色、约束、目的以及模型元素基本行为等信息
- 文档窗口中输入的一切都将显示为生成的代码中的说明语句，以后不必输入系统代码的说明语句

4 Rose应用程序界面

■ 4.4 模型图窗口

– 综览窗口

- 提供当前模型图的一个小比例视图，以便浏览整个模型图



4 Rose应用程序界面

■ 4.4 模型图窗口

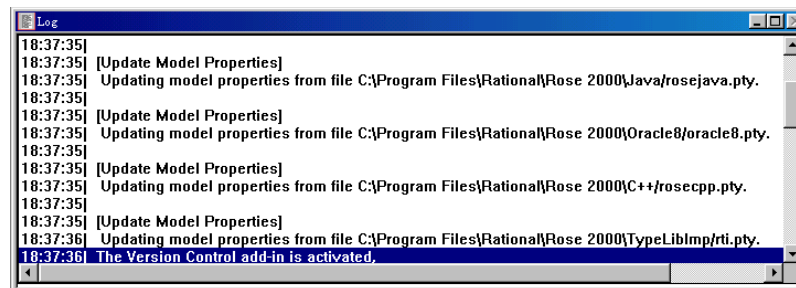
- 模型图打印对话框
 - 4个标签
- 用于设置打印信息
- 打印预览
- 应用过滤



4 Rose应用程序界面

■ 4.5 日志窗口

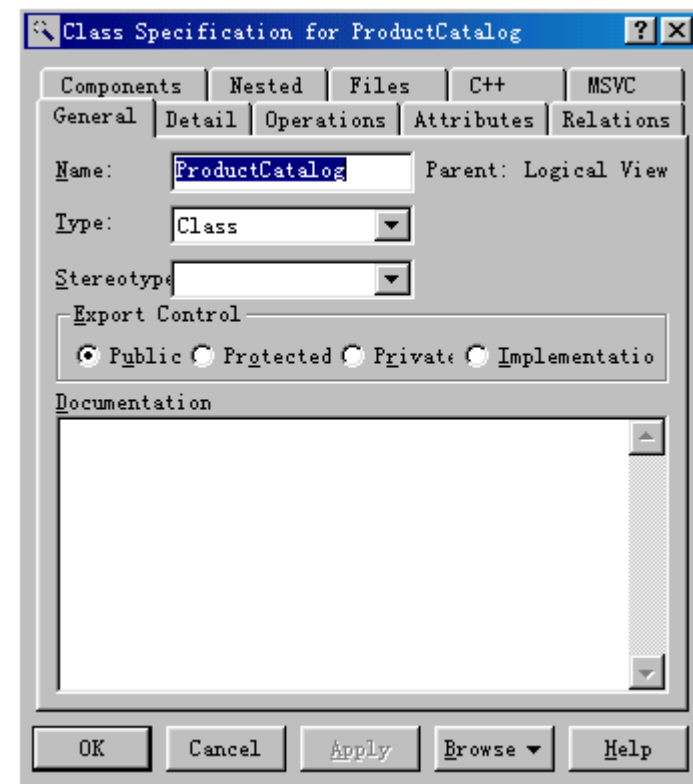
- 显示按时间顺序执行某些命令和操作后，应用程序的进展情况、结果和错误
- 可以隐藏消息前面的时间前缀
- 日志可以保存



4 Rose应用程序界面

■ 4.6 规范窗口

- 用于显示和修改模型元素的属性和关系
- 信息以文本方式呈现，在模型元素的图标内可能会显示部分规范窗口中的信息
- 信息按标签进行显示





5 Rose模型

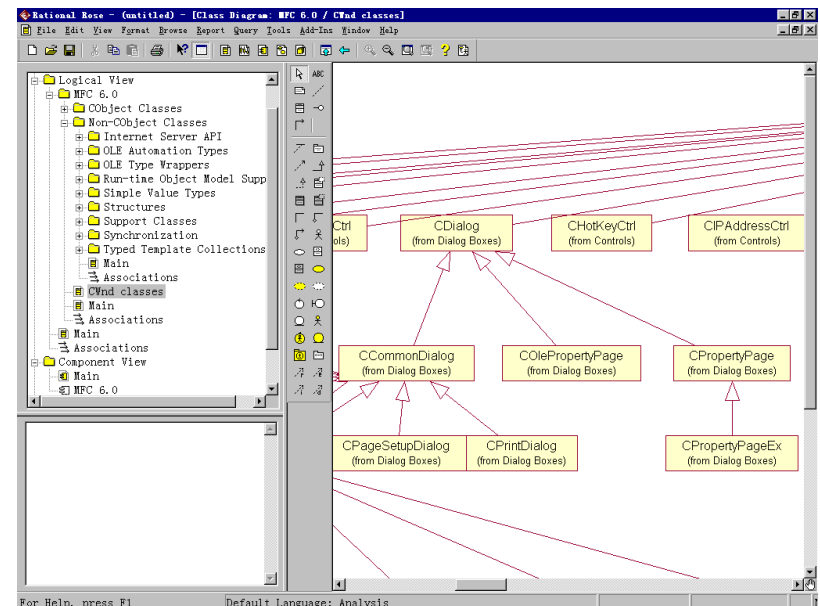
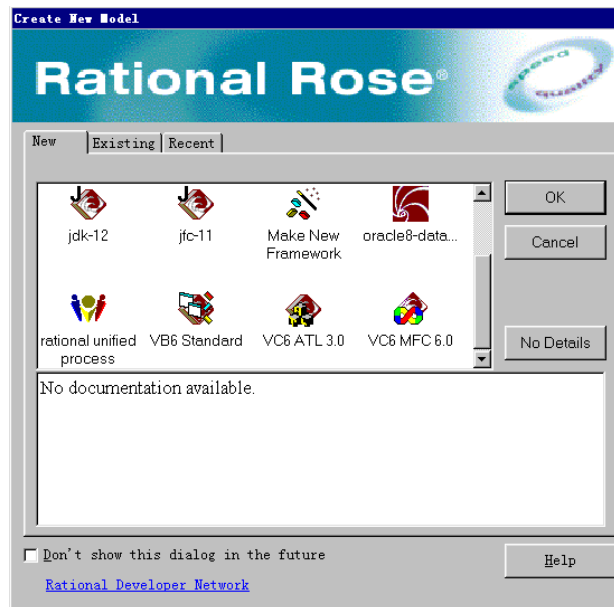
Rose模型是问题域和软件系统的表示。模型中包含的元素有类、逻辑包、对象、操作、构件包、构件、处理器、设备以及它们之间的关系。这些模型元素中的每一个元素都拥有能唯一标识它们自身的模型属性。**Rose**模型还包含模型图和规范，它们提供了对模型元素及其属性进行可视化和操作的手段。

5 Rose模型

■ 5.1 创建Rose模型

– 框架向导（Framework Wizard插件）

- 框架是一系列预定义的模型元素，可以定义某种系统的体系结构，也可以提供一系列可重用构件





5 Rose模型

■ 5.2 保存Rose模型

- 在默认的情况下，**Rose**模型都以扩展名为.mdl的文件进行保存
- *.ptl格式文件类似于模型文件 (*.mdl)，但是只是模型文件的一部分。模型文件*.mdl则保存完整的模型
- 以**Rose**的旧版本保存模型，可能会丢失某些模型元素和特性



5 Rose模型

■ 5.3 Rose模型的导入与导出

– 导出模型及模型元素

- 导出模型或者模型元素到**Petal**文件的时机：
 - 将元素从一个模型导到另一个模型
 - 在不同的平台之间传送模型或模型元素
 - 将一个模型或它的元素添加到一个新的软件版次
 - 导入模型、包或类
- 导到**Petal**文件中的内容包括：整个模型、类、逻辑包以及构件包



5 Rose模型

■ 5.3 Rose模型的导入与导出

– 导入模型及模型元素

- 导入时可选择的文件类型有：
 - 模型（.mdl）
 - petal(.ptl)
 - 类别（.cat）
 - 子系统（.sub）
- Rose会将导入的元素和当前模型中的相关元素进行比较，提示是否要用导入的元素取代当前模型中的元素。导入元素之后，Rose会更新当前模型中的所有模型图。



5 Rose模型

■ 5.4 将Rose模型发布到Web上

– Web发布者（Web Publisher）

- 创建基于Web（HTML）的模型版本，将模型发布到Web上，通过浏览器顺序或非顺序地进行查看
- Web发布者会重新创建Rose模型元素，包括图、类、包、关系、属性以及操作等
- Web发布者所发布的内容可以通过选项控制



5 Rose模型

■ 5.4 将Rose模型发布到Web上

– Web发布器生成的文件

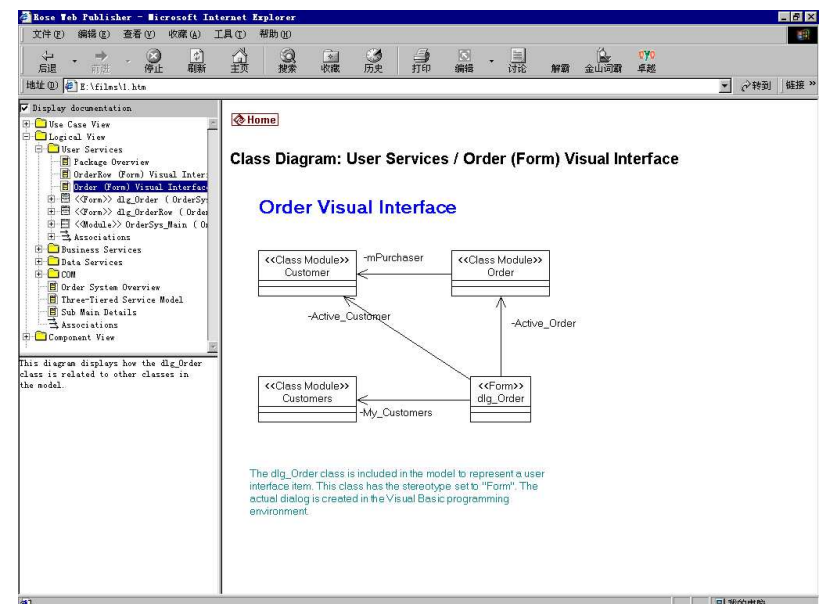
- 在发布模型之前，应当创建一个新的文件夹
- 发布一个模型时，需要提供一个HTML根文件的
名字。通过打开该文件来显示模型。

5 Rose模型

■ 5.4 将Rose模型发布到Web上

— 发布Rose模型

- 使用Web Publisher命令
- 使用Ros Web 发布器批处理器





5 Rose模型

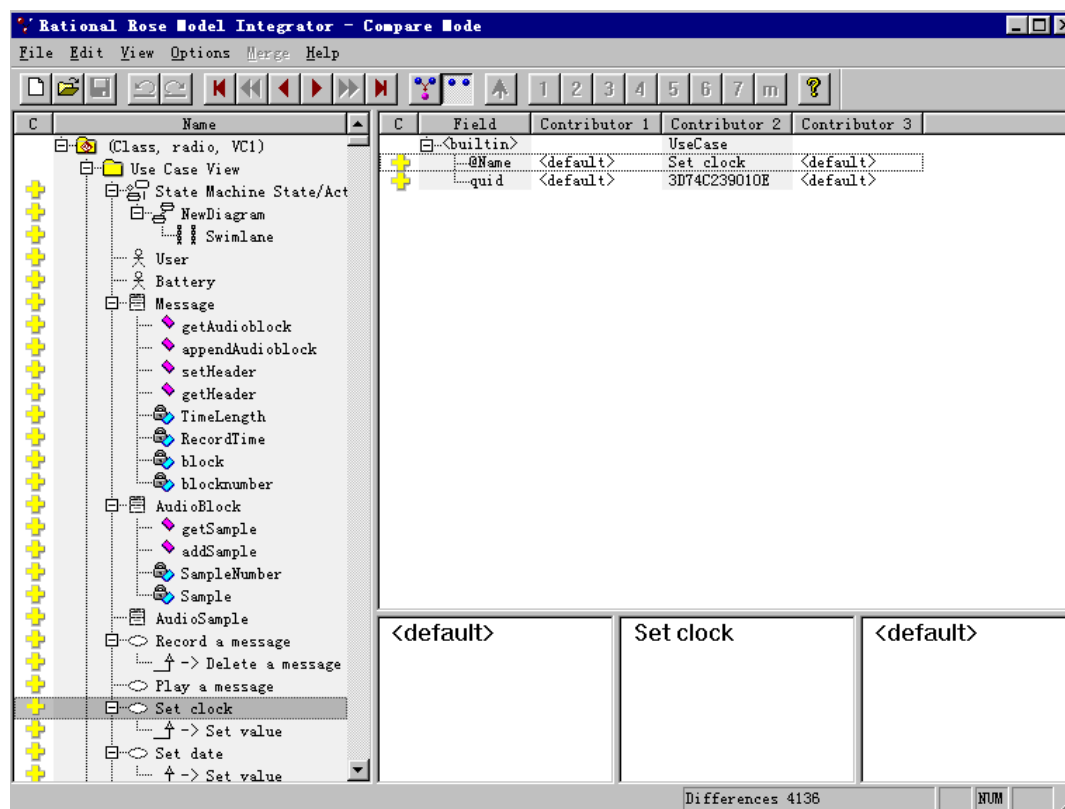
■ 5.5模型集成

— 模型集成器（Model Integrator）

- 用于对模型进行比较和合并，一次最多可以处理7个模型。个人可以独立地工作，然后通过模型集成器将模型集成起来。在对模型进行比较时，模型集成器能够显示出模型之间的差别
- 模型的比较与合并操作都在模型集成器中进行

5 Rose模型

— 模型比较





6 Rose视图

■ 6.1 用例视图

— 基本概念

- 系统中与实现无关的视图，只关心系统的高级功能，而不关心系统的具体实现细节
- 通常在项目开始时要先确定，之后不轻易修改



6 Rose视图

■ 6.1 用例视图

— 包含的内容

- 业务参与者、业务工作者
- 业务用例、业务用例图、业务用例实现
- 参与者
- 用例、用例图、用例文档
- 类图
- 状态图、活动图
- 顺序图、协作图
- 包、文件、URL



6 Rose视图

■ 6.2 逻辑视图

— 基本概念

- 关注系统如何实现使用用例中提到的功能，涵盖系统实现的具体细节
- 从中可以看到系统的逻辑结构



6 Rose视图

■ 6.2 逻辑视图

– 包含的内容

- 用例、用例图
- 类、类实体、类图
- 接口
- 活动图、状态图
- 协作图、顺序图
- 包、文件、URL



6 Rose视图

■ 6.3 构件视图

— 基本概念

- 包含模型代码库、执行库和其它构件的信息
- 从中可以看出系统实现的物理结构



6 Rose视图

■ 6.3 构件视图

– 包含的内容

- 构件
- 接口
- 构件图
- 包
- 文件
- URL



6 Rose视图

■ 6.4 部署视图

— 基本概念

- 关心系统的实际部署情况
- 一个项目只有一个部署视图



6 Rose视图

■ 6.4 部署视图

— 包含的内容

- 进程
- 处理器
- 连接器
- 设备
- 部署图
- 文件
- URL

第2章 Rose操作基础



《Rational Rose 2003基础教程》

配套电子教案



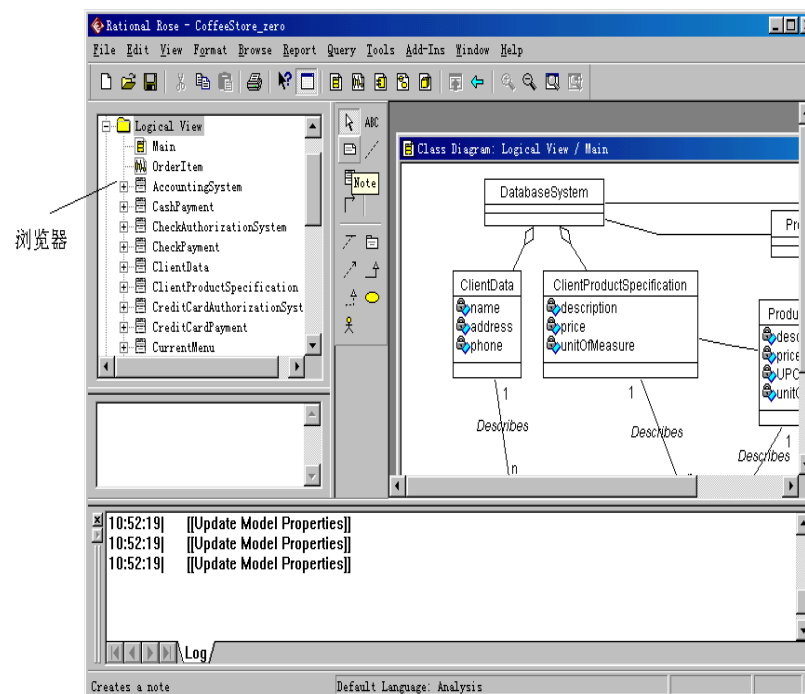
内 容

- 浏览器
- 模型图
- 模型元素的规范

1 浏览器

浏览器功能：

- 可视化显示模型中所有元素的层次结构；
- 拖放（Drag-and-drop）功能
- 同步更新模型，即，浏览器中的模型元素发生变化时，可以自动更新模型中的相应元素，反之亦然





1 浏览器

■ 1.1 浏览器的停靠模式

- 以固定的大小停靠在边框上
- 以可变的大小浮动显示在窗口的任何位置

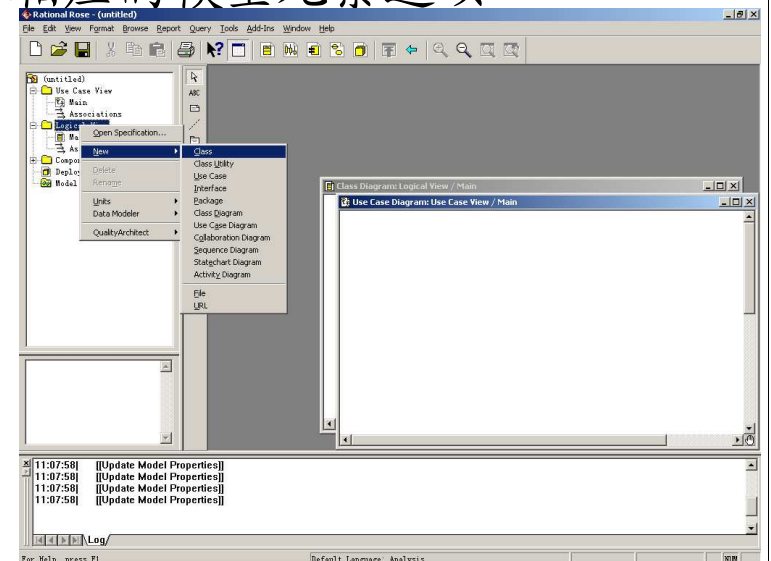
1 浏览器

■ 1.2 操作模型元素

— 创建一个模型元素

- 利用快捷菜单创建模型元素
 - 右单击新模型元素所属的父元素（可以是视图、模型图、包等），从快捷菜单中选择New
 - 在New下拉菜单栏中选择相应的模型元素选项

- 运用拖放功能
 - 注意源位置的标识
“from ...”





1 浏览器

■ 1.2 操作模型元素

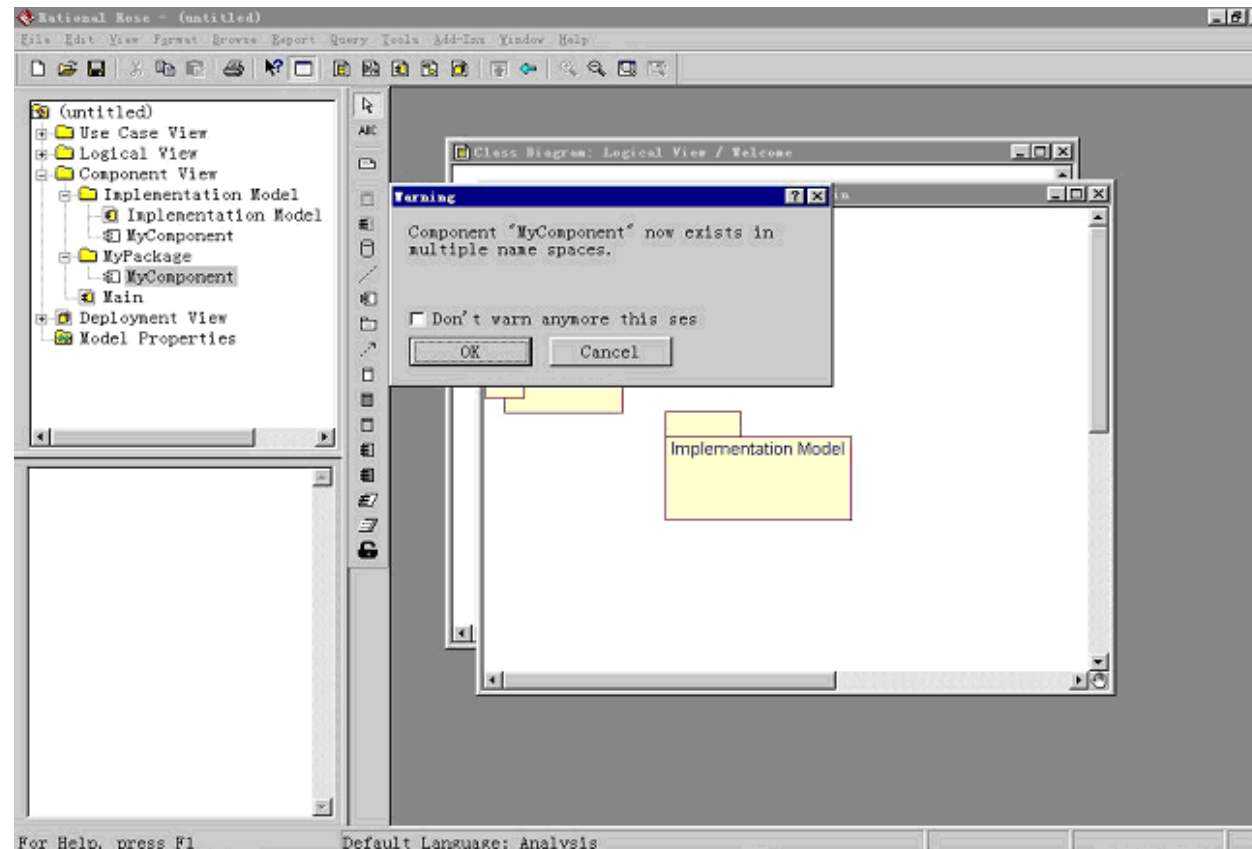
— 删除模型元素

- 从浏览器中删除一个模型元素，将把该模型元素从模型中永久删除，同时还将删除该元素的关系
- 可以一次删除多个模型元素
 - 按下Ctrl或者Shift键选取要删除的多个模型元素

— 命名模型元素

- 直接在浏览器中输入模型元素的名称
- 注意多元素同名的命名错误

1 浏览器



不同元素相同命名出现的警告对话框



1 浏览器

■ 1.3 拖放功能

- 浏览器内部的拖放功能
- 浏览器与模型图之间的拖放功能
- 浏览器与规范窗口之间的拖放功能

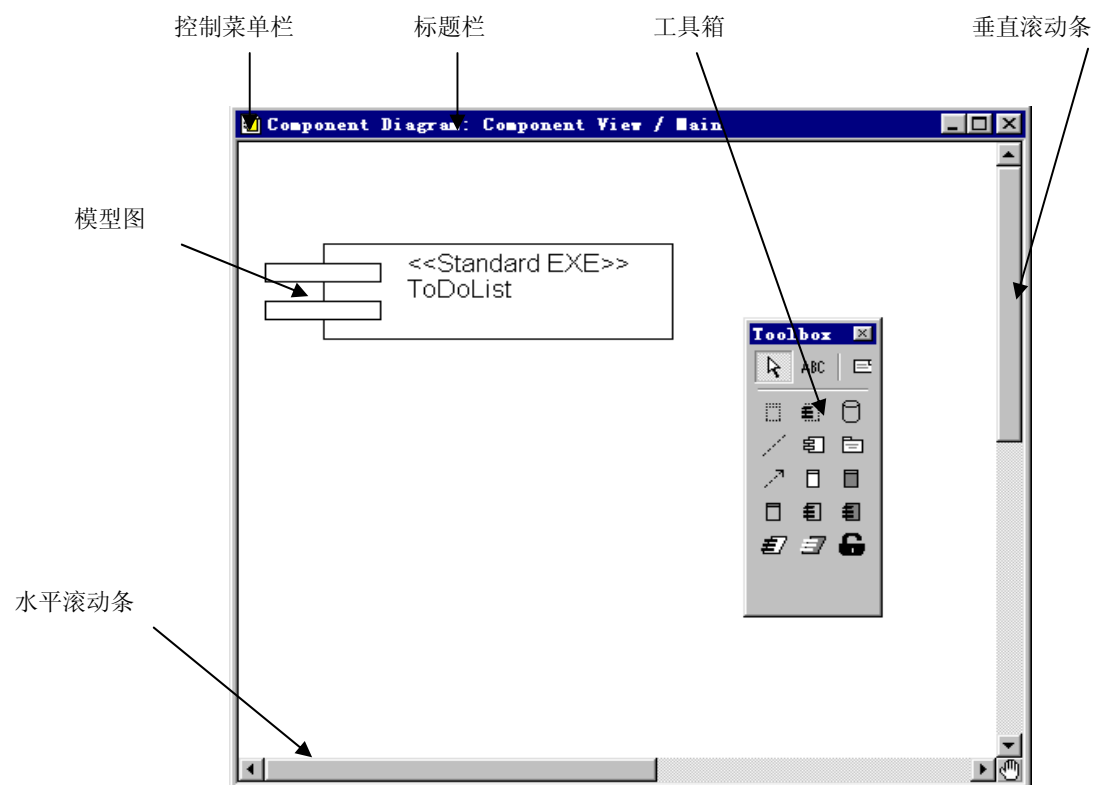
2 模型图

Rose支持的模型图

模型图	图标	描述	建模角度
类图 Class diagram		显示系统中的类和包，提供系统构件及其相互关系	静态结构建模
用例图 Use-case diagram		用例图从用户的角度描述系统功能的使用者和主要的系统操作流程。显示用例与参与者及其相互关系	系统功能建模
协作图 Collaboration diagram		从对象组织结构的角度显示用例中特定情形的操作流程	动态行为建模
顺序图 Sequence diagram		按时间顺序显示用例中特定情形的操作流程	动态行为建模
状态图 Statechart diagram		显示系统中类的对象所有可能的状态以及事件发生时状态的转换条件	动态行为建模
活动图 Activity diagram		描述满足用例要求所需进行的活动以及活动间的关系的图	动态行为建模
构件图 Component diagram		描述代码构件的物理结构以及构件之间的依赖关系。组件图有助于分析和理解组件之间的影响程度	静态结构建模
部署图 Deployment diagram		描述系统中的物理结构	静态结构建模

2 模型图

■ 2.1 模型图窗口





2 模型图

■ 2.1 模型图窗口

- 可以在模型图窗口中创建和修改模型的图形视图
- 模型图中的每个图标表示模型中的一个元素
- 每个模型图只展示系统模型多种不同视图中的某一个
- 可以同时应用程序窗口中显示多个不同的模型图



2 模型图

■ 2.2 模型图操作

- 下面的6种常见模型图操作，既可以在浏览器中进行，也可以通过菜单栏中的Browse项进行：
 - 创建一个新的模型图
 - 删除一个模型图
 - 显示一个模型图
 - 重命名一个模型图
 - 链接一个模型图
 - 在模型图窗口中选择多个元素



2 模型图

■ 2.3 模型元素操作

— 创建模型元素

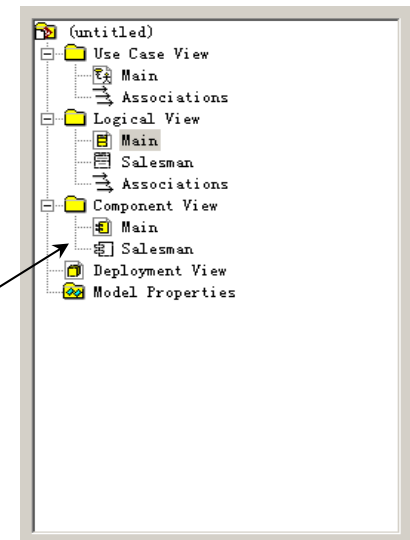
- 1) 单击工具箱中适当的创建工具
- 2) 在模型图窗口中的某一位置单击鼠标左键

2 模型图

■ 2.3 模型元素操作

— 命名模型元素

- 不在同一个包内的参与者、用例、类、构件和包，名称可以相同。不同的模型元素拥有相同的名称时，这些元素被称为“重载”
 - 重载允许你进行基于多语言构件的开发。
 - 重载允许用例视图中的参与者和逻辑视图中的类拥有相同的名称





2 模型图

■ 2.3 模型元素操作

- 创建一个重载元素

- 1) 从工具箱中创建一个新的元素
- 2) 双击新元素，或者单击
Browse>Specification, 打开规范窗口
- 3) 在名称字段中输入名称
- 4) 单击OK按钮



2 模型图

■ 2.4 操纵模型元素图标

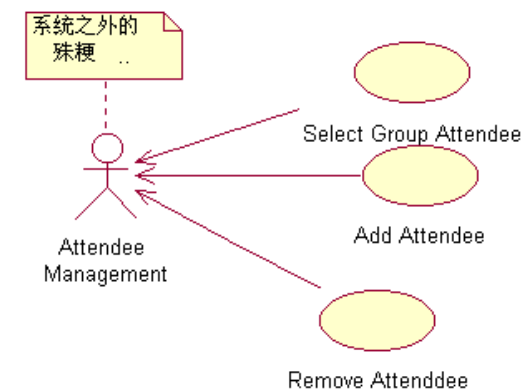
- 选择图标
- 撤销图标选择
- 调整图标大小
- 移动图标
- 改变图标表示的元素类型
- 剪切、复制和粘贴图标
- 图注

2 模型图

■ 2.4 操纵模型元素图标

- 图注的操作

- 图注是加进模型图中的少量文本，可以和模型图中的特定元素相联系
- 图注用于说明模型元素





2 模型图

■ 2.5 相互关系

- 对于一个选定的模型图，相互关系可以是一种关系、链接、依赖、转移或者连接
- 创建两个元素之间的相互关系
 - 1) 单击工具箱中的关系工具
 - 2) 将光标移到模型图中的客户（client）图标
 - 3) 按下鼠标左键
 - 4) 将箭头拖动到模型图中的提供者（supplier）图标
 - 5) 在提供者元素上点击，释放鼠标



2 模型图

■ 2.5 相互关系

- 命名相互关系

- 1) 单击图标
- 2) 输入名称
- 3) 在已命名的图标之外单击鼠标



2 模型图

■ 2.6 装饰模型图

- 将文本放进模型图中
 - 1) 从工具箱中选择ABC工具
 - 2) 在模型图的某个位置单击鼠标左键
 - 3) 编辑文本
- 颜色填充和线条着色
 - 使用不同的颜色可以突出显示一些重要的元素



2 模型图

■ 2.6 装饰模型图

- 颜色填充

- 1) 右单击模型元素图标
- 2) 选中Use Fill Color
- 3) 再次右单击模型元素图标，从快捷菜单中选择Format>FillColor
- 4) 在弹出的颜色对话框中选择颜色



2 模型图

■ 2.6 装饰模型图

- 线条着色

- 1) 右单击模型元素图标
- 2) 从快捷菜单中选择Format>FillColor
- 3) 选择适当的颜色并确定



2 模型图

■ 2.7 模型工作空间

- 模型工作空间是当前所有已加载的单元和已打开的模型图的快照
- 通过定义一个或多个工作空间，可以建立在 Rational Rose 中的工作环境，并在每次准备好工作的时候返回到工作空间
- 加载工作空间时，Rose 通过加载特定的控制单元和打开正确的模型图，取回它们的快照



2 模型图

■ 2.7 模型工作空间

- 已保存的模型和模型工作空间

- 一个完整的、已保存的模型包含模型图、元素和控制单元。一个模型工作空间包含的是已保存的模型、已打开的模型图和控制单元在特定时间的实际状态。
- 可以让多个模型工作空间与一个模型相关
- 模型工作空间的保存不影响该模型如何在另一台机器上加载



2 模型图

■ 2.7 模型工作空间

- 保存模型工作空间

- 在默认的情况下，Rational Rose以“<模型名>-<操作系统用户名>.wsp”的形式命名工作空间
- 要保存一个模型工作空间
 - 1) 单击File>Save Model Workspace (Rose同时保存模型和工作空间文件)
 - 2) 在Save As对话框中输入工作空间文件的名称



2 模型图

■ 2.7 模型工作空间

- 加载模型工作空间

- 1) 单击File>Load Model Workspace
- 2) 选择要加载的模型工作空间文件名
- 3) 单击Open按钮



3 模型元素的规范

- 规范窗口用于显示和修改模型元素的属性和关系
- 在规范窗口中显示的一些信息也可以在图标中显示出来，用于来描绘模型图中的模型元素
- 规范窗口中提供了诸如字段、列表框、选项按钮和复选框之类的标准接口元素



3 模型元素的规范

■ 3.1 显示规范窗口

– 显示模型元素规范窗口的方法:

- 在模型图或者浏览器中双击一个项
 - 注意: 通过双击一个逻辑或构件包时显示规范窗口, 必须关掉Tools>Options>Diagram>Double-Click to Diagram选项
- 单击模型图中的一个项, 然后单击Browse>Specification
- 右单击模型图中的一项以显示快捷菜单, 然后单击Open Specification
- 选中模型图中的一个项, 然后按下CTRL + B
- 选中模型图中的一个项, 然后按下F4键



3 模型元素的规范

■ 3.2 自定义规范窗口

- 如果语言插件支持模型元素中相应的规范，则打开一个已经映射到语言的模型元素的规范窗口时，将显示语言自定义的规范。否则，只显示标准的Rose规范窗口。



3 模型元素的规范

■ 3.3 编辑规范

- 通过编辑一个模型元素的规范，或者通过修改模型图中的图标来改变该元素的属性或关系，Rose会自动地更新相应的模型图和规范
- 一个模型元素如果是写保护的，或者被包含在一个写保护的控制单元内，则该元素规范的OK按钮将被禁用，以阻止对它的修改

3 模型元素的规范

■ 3.4规范窗口的常见元素

- 对话框

- 所有的规范窗口都以对话框的形式显示，并且带有可以导航到特定页和特定项的标签

- General 标签

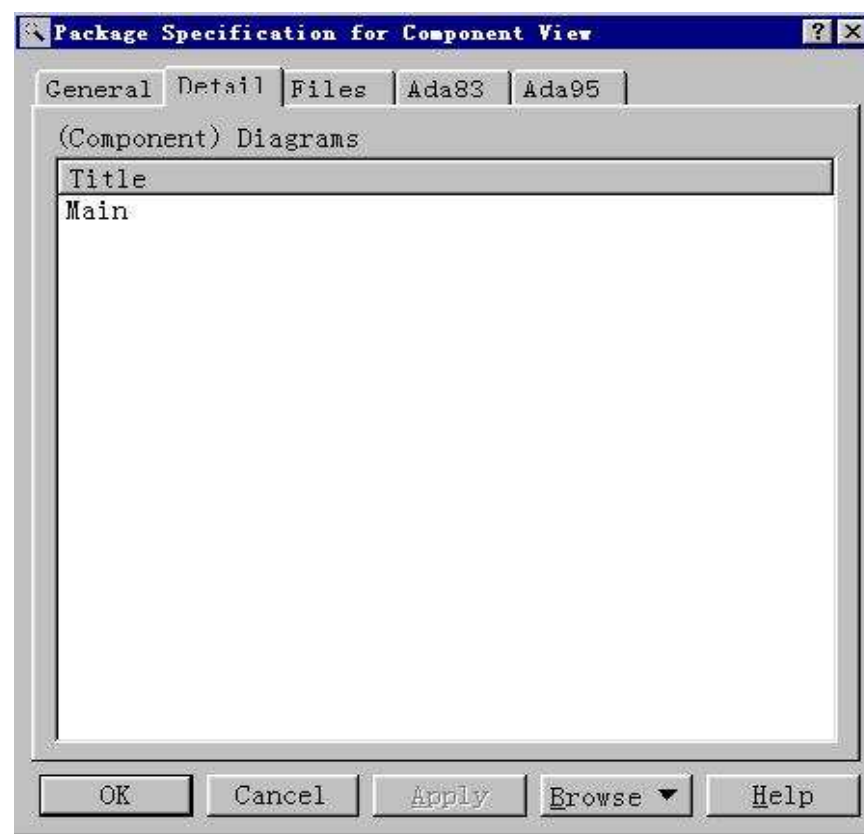
- 名称 (Name)
- 文档 (Documentation)
- 构造型 (Stereotype)



3 模型元素的规范

■ 3.4 规范窗口的常见元素

– Detail 标签

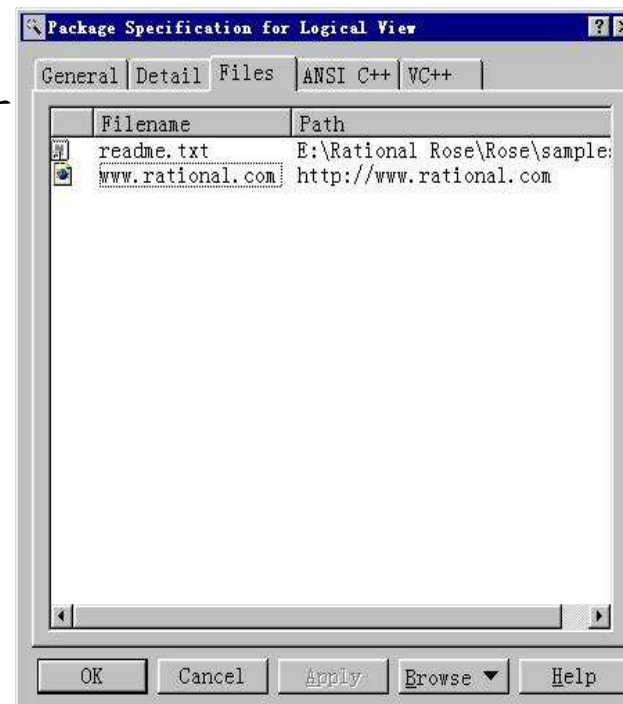


3 模型元素的规范

■ 3.4规范窗口的常见元素

- Files标签

- 显示文件、URL、视图文件，以及已经插入到或链接到模型元素和模型图的URL。
- Files标签在操纵补充文档的链接时十分有用
- 当在浏览器中展开元素或模型图时，所有在Files标签中列出的URL和文件都将显示出来



3 模型元素的规范

■ 3.4 规范窗口的常见元素

- 5个控制按钮

- OK
- Cancel
- Apply
- Browse



- Select in Browser: 高亮显示浏览器中所选项
 - Browse Parent: 打开所选项父项的规范窗口
 - Browse Selection: 打开当前被选项的规范窗口
 - Show Usage: 显示一个所有当前被选项支持的图的列表（在协作图中，显示消息的用法列表）
- Help

3 模型元素的规范

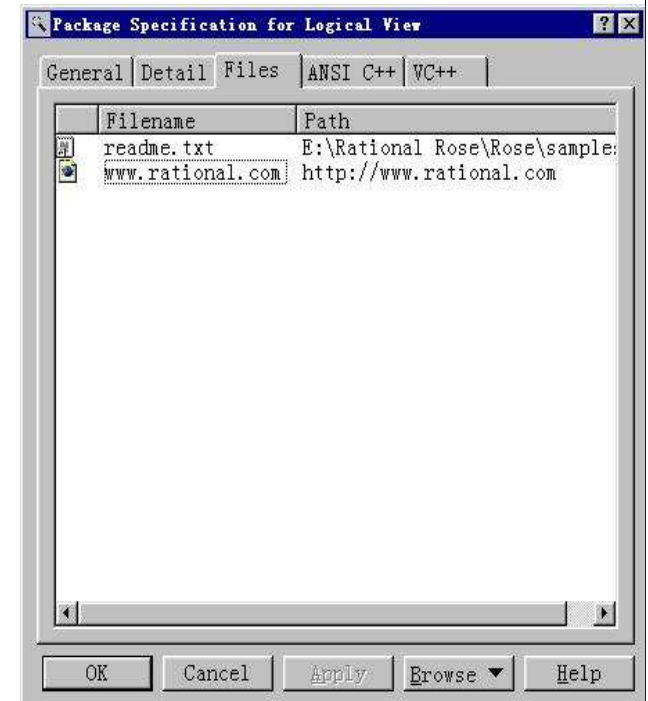
■ 3.5 标签导航

– 插入操作（插入一个新的行）

- 单击快捷菜单中的Insert
- 或者，按INSERT键

– 删除操作（删除一行）

- 选中行，单击快捷菜单中的Delete
- 或者，按DELETE键



第3章 用例图及其应用



《Rational Rose 2003基础教程》

配套电子教案



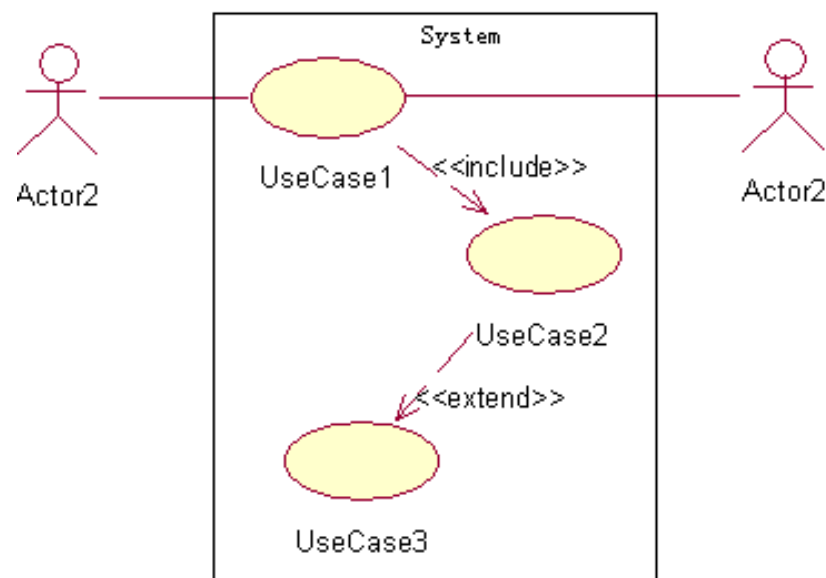
内 容

- 基本概念
- 关系及其应用
- 参与者规范及应用
- 用例规范及应用
- 用例视图

1 基本概念

用例图由三部分组成：

- 参与者
- 一组（个）用例
- 关系



1 基本概念

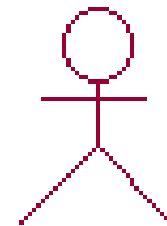
■ 1.1 参与者

— 定义

- 是直接与系统相互作用的系统、子系统或类的外部实体的抽象。它是用户所扮演的角色，是系统的用户。每个参与者定义了一个角色集合。通常，一个参与者可以代表一个人、一个计算机子系统、硬件设备或者时间等角色。典型的参与者如销售部经理、销售员和结帐系统。

— 图形表示

- 用小人图符表示



Student



1 基本概念

■ 1.1 参与者

— 参与者的识别

- 谁将使用系统的主要功能?
- ■ 谁将需要系统的支持来完成他们的日常任务?
- ■ 谁必须维护、管理和确保系统正常工作?
- ■ 谁将给系统提供信息、使用信息和删除信息?
- ■ 系统需要处理哪些硬件设备?
- ■ 系统使用了外部资源吗?
- ■ 系统需要与其他什么系统交互吗?
- ■ 谁或者什么对系统产生的结果感兴趣?
- ■ 一个人同时使用几种不同的规则吗?
- ■ 几个人使用相同的规则吗?
- ■ 系统使用遗留下来的应用吗?

1 基本概念

■ 1.2 用例

– 定义

- 对一组动作序列的描述，系统通过执行这一组动作序列为参与者产生一个可观察的结果

– 用例特征

- 说明了系统具有的一种行为模式
- 说明了一个参与者与系统执行的一个相关的事务序列
- 提供了一种获取系统需求的方法
- 提供了一种与最终的用户和领域专家进行沟通的方法
- 提供了一种测试系统的方法

– 图形表示

- 用椭圆形表示，用例的名字显示在图标在下面



Purchase Ticket



1 基本概念

■ 1.2 用例

— 用例识别

- 参与者要向系统请求什么功能?
- 每个参与者的特定任务是什么?
- 参与者需要读取、创建、撤消、修改、或存储系统的某些信息吗?
- 是否任何一个参与者都要向系统通知有关突发性的、外部的改变? 或者必须通知参与者关于系统中的发生的事件?
- 这些事件代表了哪些功能?
- 系统需要哪些输入/输出?
- 这些输入输出来自哪里或者到哪里去?
- 哪些用例支持或维护系统?
- 是否所有功能需求都被用例使用了?
- 系统当前实现的主要问题是什么?



1 基本概念

■ 1.3 事件流

- 事件流是用例完成需求行为的事件描述。
- 事件流的目的是建立用例中逻辑流程的文档，详细描述系统用户的工作和系统本身的工作，既包括正常状态下系统完成需求行为的事件，也包括在其他状态下不能完成需求行为的事件。
- 事件流通常包括：
 - 简要说明
 - 前置条件
 - 事件流
 - 后置条件



1 基本概念

■ 1.4 用例模型

一个用例模型由一个或者多个用例图和所有的支持文件（诸如用例规范和参与者定义等）所构成。用例规范是大多数用例模型的产物，而用例图充当将需求模型综合在一起的粘胶剂。用例模型应当从项目投资者的角度进行开发，而不是从开发者的（通常是技术）观点去开发。



2 关系及其应用

关系反应了参与者和用例之间、用例和用例之间以及参与者和参与者之间的相互作用。

在一个用例图中，可能会出现关联关系、依赖关系、泛化关系以及这三种关系的扩展形式：扩展关系、包含关系和精化关系。



2 关系及其应用

■ 2.1 关联关系

关联关系表示一种通信路径，它存在于参与者和用例之间，提供用例和参与者之间的通信途径。建立通信之后，信息可以双向流动。

关系方向显示的不是信息的流动方向，而是谁启动信息。

2 关系及其应用

■ 2.1 关联关系

— 表示

- 工具箱中：一个直角直线

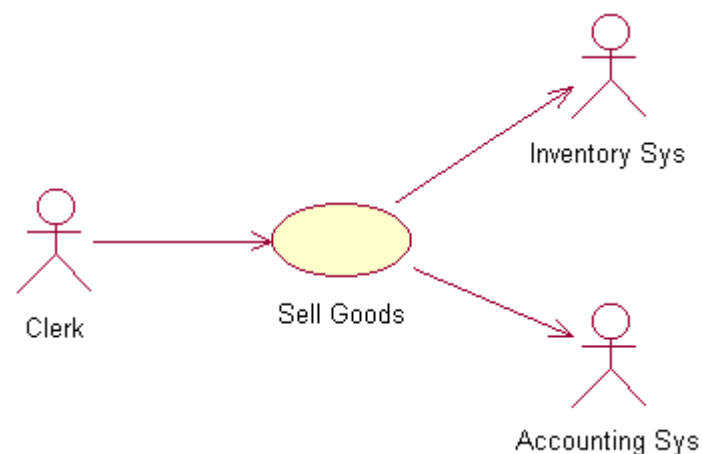


- 模型图中：一条直线或者一条带箭头的直线



— 关联命名

- 一个动词或者一个动词短语，用于指明关系的类型或者目的。



关联关系表示通信途径

2 关系及其应用

■ 2.1 关联关系

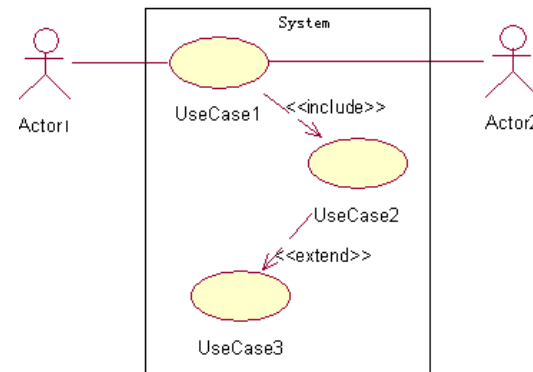
- 在用例图中，通常存在两种类型的关联：
 - 单向关联



- 双向关联

Actor1 与 UseCase1

Actor2 与 UseCase1



2 关系及其应用

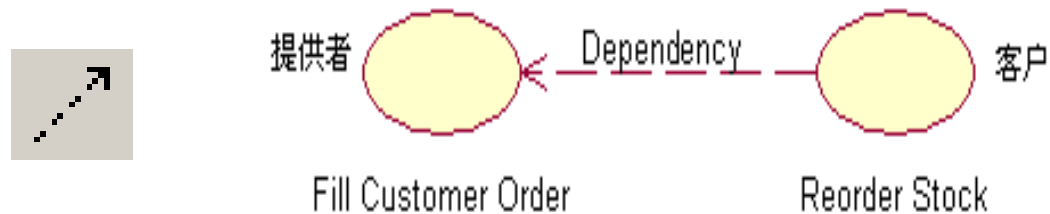
■ 2.2 依赖关系

— 定义

- 存在于两个模型要素之间的一种关系，其中一个模型要素的改变将影响另一个模型要素

— 表示方法

- 工具箱和模型图中均表示为一个带箭头的虚线
- 画图时，拖动鼠标从客户到提供者画出关联关系




2 关系及其应用

■ 2.3 泛化关系

— 定义

- 在一个更一般的模型要素和另一个较具体的模型要素之间存在的一种关系，通常用于表示类（包括用例、参与者等）之间的继承关系

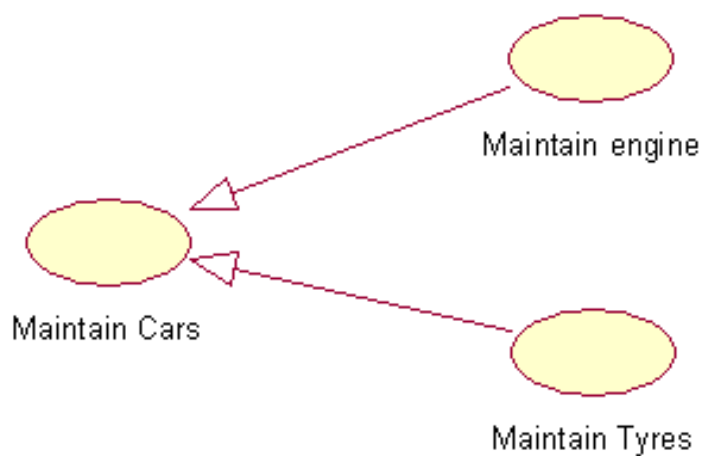
— 表示方法

- 工具箱中：
- 模型图中：一条带空心三角形箭头的实线（箭头方向从具体用例指向一般用例）

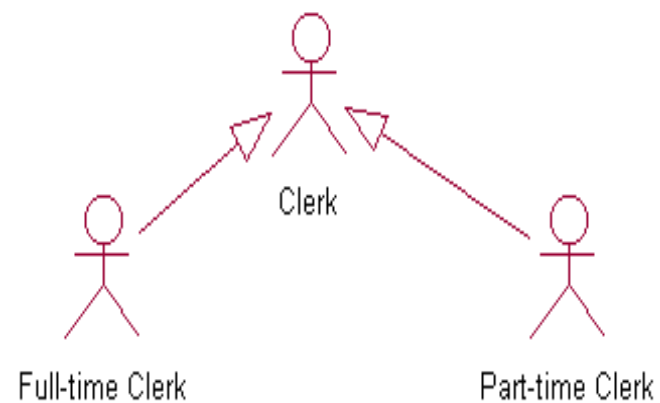


2 关系及其应用

■ 2.3 泛化关系



用例之间的泛化关系



参与者之间的泛化关系

2 关系及其应用

■ 2.4 关系的扩展

– 1) 扩展关系

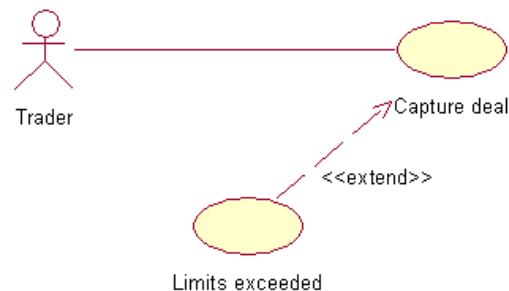
- 扩展关系可以放置在所有的关系上，大多数扩展构造型都放置在依赖关系和关联关系上
- 扩展关系用带箭头的虚线表示，沿线上加一个用双尖括号括起来的“extend”

`<<extend>>`
----->

2 关系及其应用

■ 2.4 关系的扩展

- 常见的几种扩展关系
 - a. 两个用例相似但不完全相同时（如图）
 - b. 当要对多个额外情况逐一建模时，可以使用扩展关系，用一个独立的用例替代每个额外的情况
 - c. 如果用例涵盖了所有的情况变化，则该用例将会变得十分复杂，应该考虑使用扩展关系

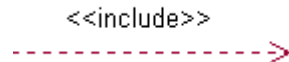


2 关系及其应用

■ 2.4 关系的扩展

– 2) 包含关系

- 是一种构造型关系，它将一个基用例连接到一个包含用例
- UML1.1中为使用关系，在1.3中改为包含关系
- 包含关系在一个用例中重用另一个用例中的步骤
- 包含关系用带箭头的虚线表示，沿线上加一个用双尖括号括起来的“include”





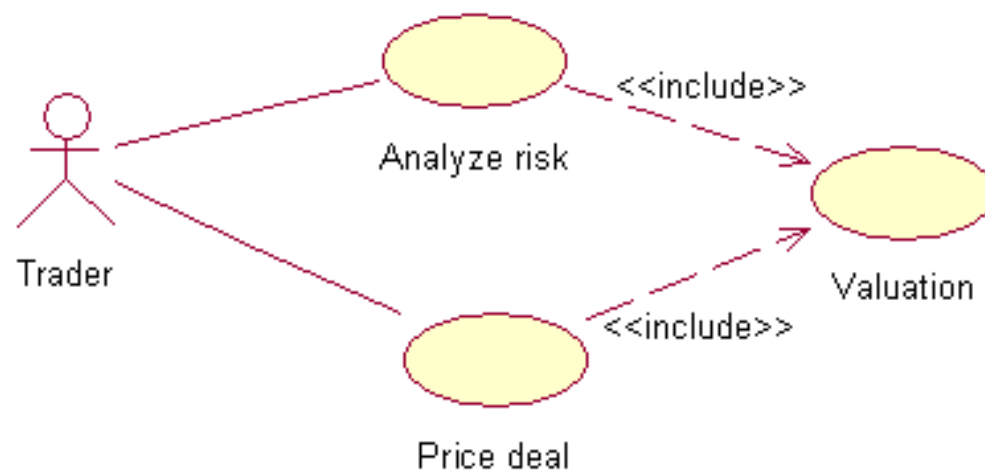
2 关系及其应用

■ 2.4 关系的扩展

- 使用包含关系的三种情况:
 - a.如果有多个用例，并且这些用例包含大量类似的行为，应该考虑将这些类似的行为通过包含关系包含到用例中
 - b.对两个或多个互相独立的用例建模时做了重复的工作，可以通过包含关系包含这些重复的工作
 - c.如果某个行为可能会引入冗余，或者，当行为发生变化时可能导致不一致性，这时，应该对这种行为进行孤立建模并将它包含到用例中

2 关系及其应用

■ 2.4 关系的扩展



包含关系举例



2 关系及其应用

■ 2.4 关系的扩展

— 3) 精化关系

- 精化关系在不同的语义层或者开发阶段连接两个或者多个模型要素。它表示了某些在一个特定的细节层次上规定的东西的更加全面的规格说明。例如，一个设计类就是一个分析类的一种精化。在一个精化关系中，源模型要素是一般的，在定义上更加概括；而目标模型要素更加具体并得到了进一步的精化。



3 参与者规范及应用

■ 3.1 参与者规范

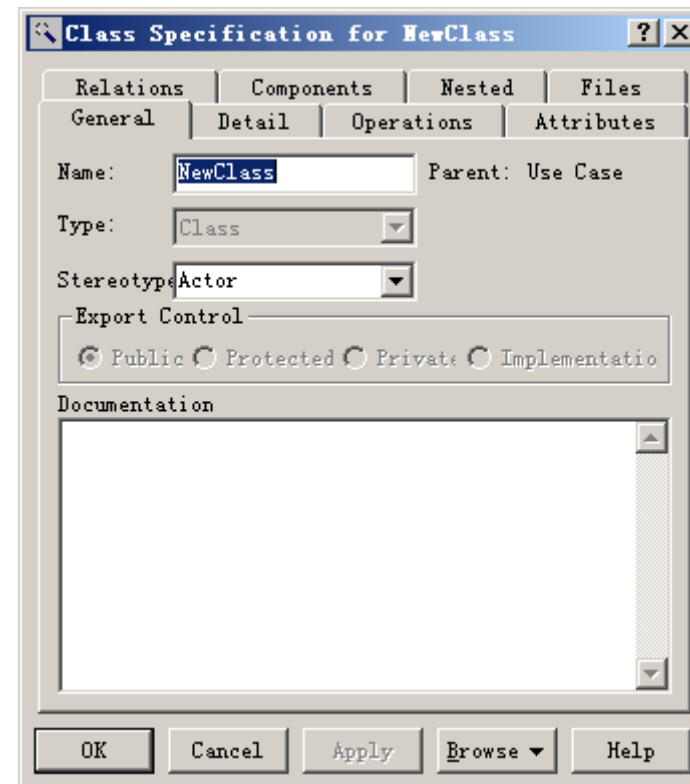
- Rose在实现中对参与者和类使用相同的规范窗口，包括如下一些标签：
 - General
 - Detail
 - Operations
 - Attributes
 - Relations
 - Components
 - Nested
 - Files

3 参与者规范及应用

■ 3.1 参与者规范

– General标签

- Name
- Stereotype
- Documentation



The image shows a 'Class Specification for NewClass' dialog box. It has a title bar with a question mark and a close button. Below the title bar are four tabs: 'Relations', 'Components', 'Nested', and 'Files'. The 'General' tab is selected, and it contains sub-tabs: 'General', 'Detail', 'Operations', and 'Attributes'. The 'General' sub-tab is active. It features a 'Name' field with 'NewClass' entered, a 'Parent' field with 'Use Case', a 'Type' dropdown menu set to 'Class', and a 'Stereotype' dropdown menu set to 'Actor'. Below these is an 'Export Control' section with four radio buttons: 'Public' (selected), 'Protected', 'Private', and 'Implementation'. At the bottom is a large 'Documentation' text area. The dialog box has 'OK', 'Cancel', 'Apply', 'Browse', and 'Help' buttons at the bottom.

3 参与者规范及应用

■ 3.1 参与者规范

– Detail标签

- Multiplicity（参与者基数）

基数	含义
0..0	0
0..1	0或者1
0..n	0或者多
1..1	1
1..n	1或者多
n	许多

- Abstract（抽象参与者）

Class Specification for NewClass

Relations Components Nested Files

General Detail Operations Attributes

Multiplic

Space:

Persistence ☐ Persistent ☒ Transient

currency ☐ Sequential ☐ Guarded ☐ Active ☐ Synchronous

☐ Abstract

Formal Arguments:

Name	Type	Default Value
------	------	---------------

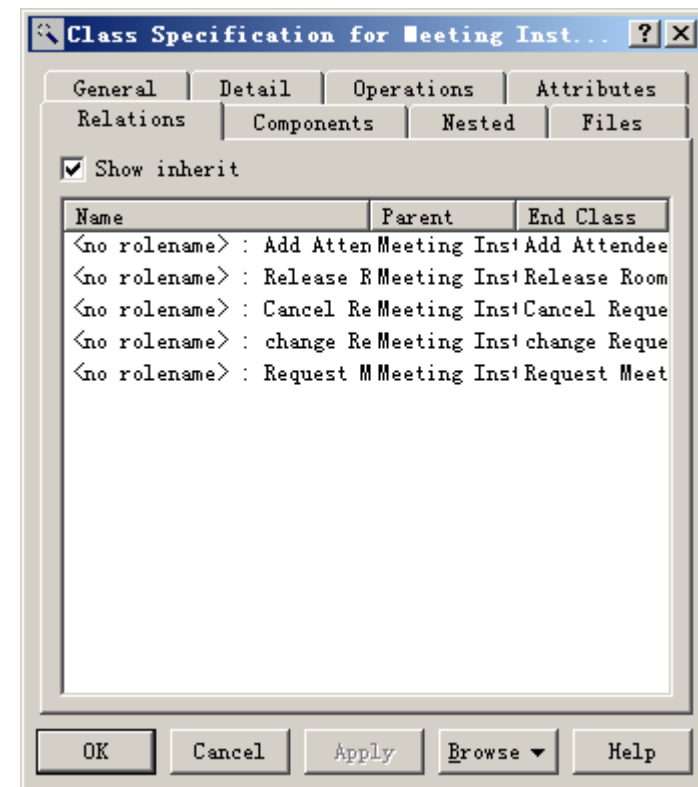
OK Cancel Apply Browse Help

3 参与者规范及应用

■ 3.1 参与者规范

– Relations标签

- 列出了参与者参与的所有关系。包括参与者与用例、参与者与其他参与者的一切关系





3 参与者规范及应用

■ 3.2 参与者的操作

- 1) 增加参与者
- 2) 删除参与者

4 用例规范及应用

■ 4.1 用例规范

- General标签
- Diagrams标签
- Relations标签
- Files标签

The screenshot shows a software dialog box titled "Use Case Specification for Price Deal". It features four tabs: "General", "Diagrams", "Relations", and "Files", with "General" currently selected. The "General" tab contains the following fields and controls:

- Name:** A text box containing "Price Deal".
- Package:** A text box containing "Use Case View".
- Stereotype:** A dropdown menu.
- Rank:** A text box.
- Abstract:** A checkbox labeled "Abstract" which is currently unchecked.
- Documentation:** A large, empty text area for notes.

At the bottom of the dialog box, there are five buttons: "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

4 用例规范及应用

■ 4.1 用例规范

– General标签

- Name
- Package
- Stereotype
- Rank
- Abstract
- Documentation

The screenshot shows a software dialog box titled "Use Case Specification for Price Deal". It has four tabs: "General", "Diagrams", "Relations", and "Files", with "General" currently selected. The "General" tab contains the following fields and controls:

- Name:** A text box containing "Price Deal".
- Package:** A text box containing "Use Case View".
- Stereotype:** A dropdown menu.
- Rank:** A text box.
- Abstract:** A checkbox, currently unchecked.
- Documentation:** A large, empty text area for notes.

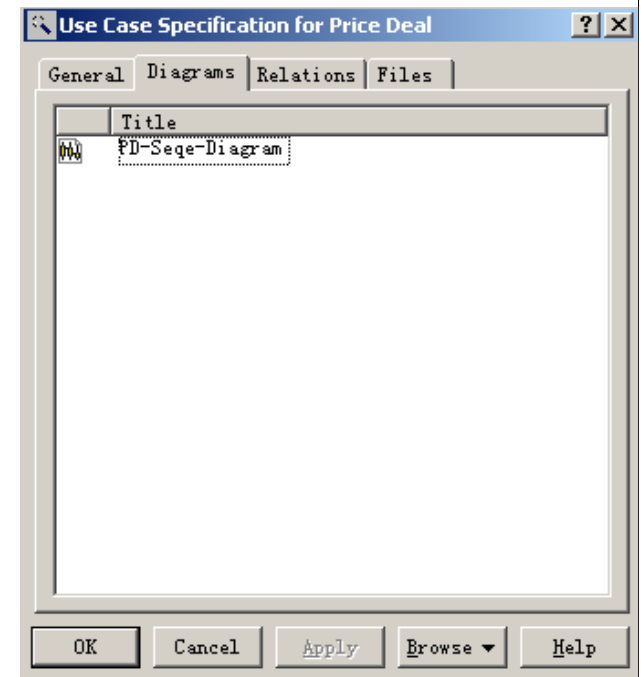
At the bottom of the dialog box are five buttons: "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

4 用例规范及应用

■ 4.1 用例规范

— Diagrams标签

- 用例所拥有的模型图的信息，其中第一列（没有标题）显示模型图的图标，第二列（Title）显示图的名称

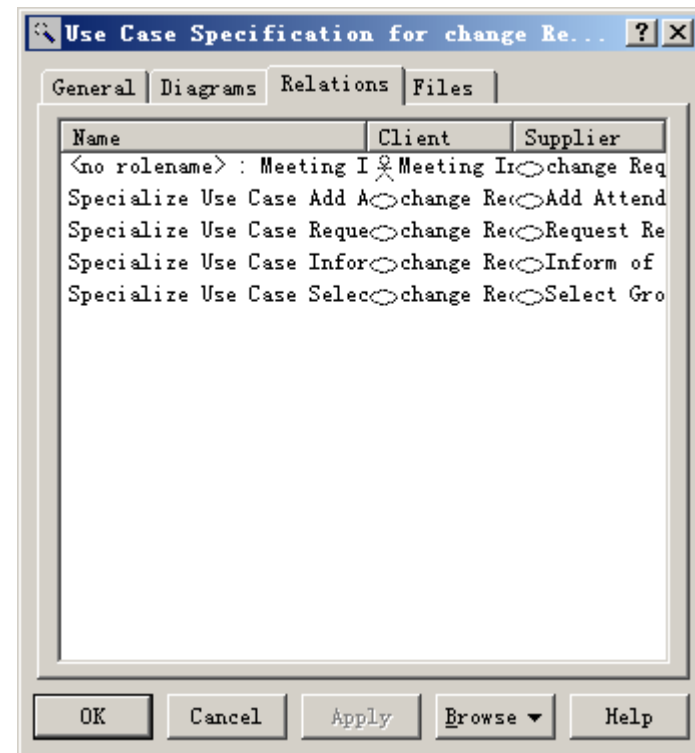


4 用例规范及应用

■ 4.1 用例规范

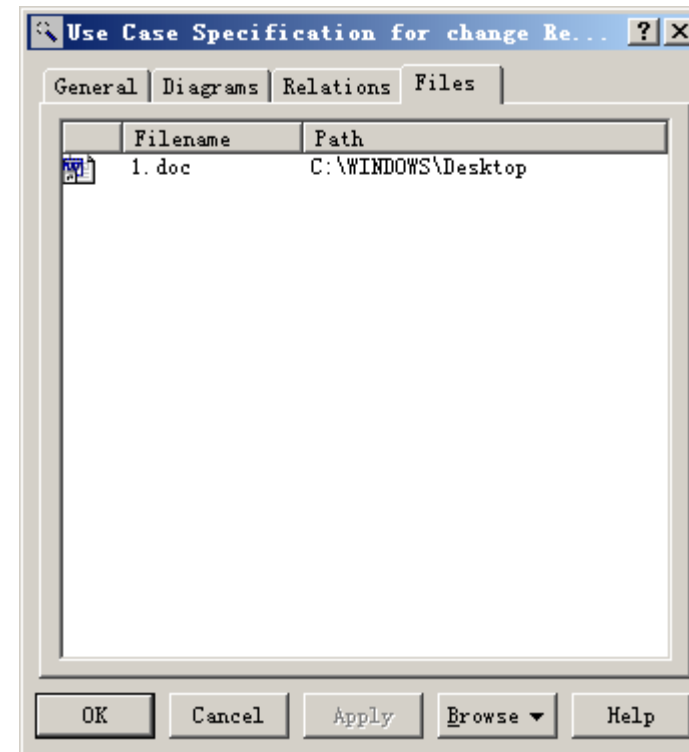
– Relations标签

- 用例与其他用例或参与者之间存在的所有关联关系



4 用例规范及应用

- 4.1 用例规范
 - Files标签





4 用例规范及应用

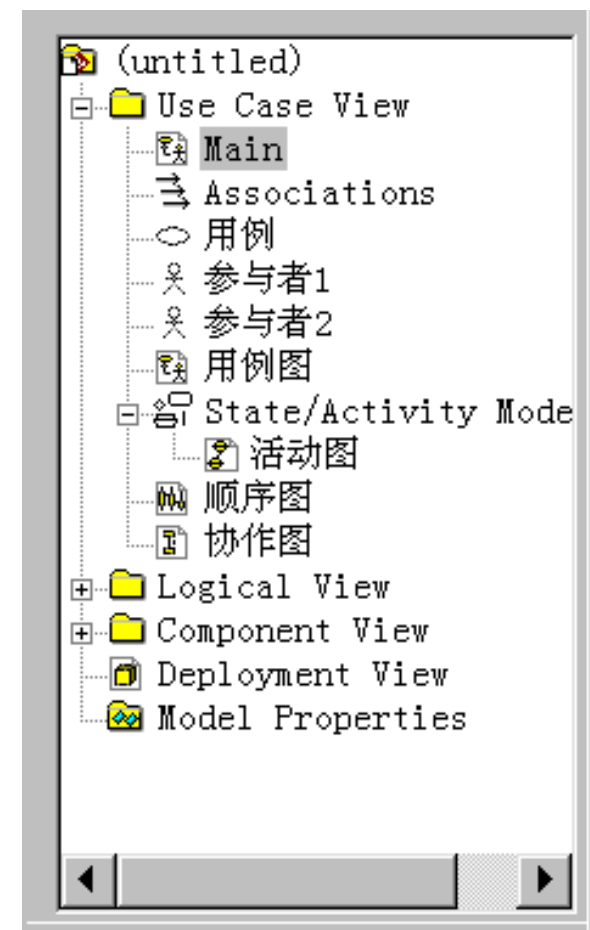
■ 4.2 用例的操作

- 增加用例
 - 将新的用例加入用例图
 - 将现有的用例加入用例图
- 删除用例
 - 仅仅从一个用例图中删除一个用例
 - 从整个模型中删除用例
- 添加文件和链接URL

5 用例视图

– 用例图包含的内容

- 用例
- 参与者
- 用例与参与者之间的通信关系
- 用例之间的包含和扩展关系
- 参与者的泛化关系
- 用例图
- 用例实现
- 顺序图
- 协作图





5 用例视图

■ 5.1 用例图操作

- 创建新的用例图
- 打开已有的用例图
- 删除用例图
- 链接用例图
- 重命名用例图



5 用例视图

■ 5.1 用例图工作箱

- 常用工具
 - 10个按钮
- 定制工具箱

第4章 类图及其应用



《Rational Rose 2003基础教程》

配套电子教案



内 容

- 基本概念
- 类图操作
- 类规范与类的应用
- 属性规范与应用
- 操作规范与应用
- 关联规范与应用
- 泛化规范与应用
- 依赖规范与应用
- 聚合规范与应用
- 逻辑包规范与应用

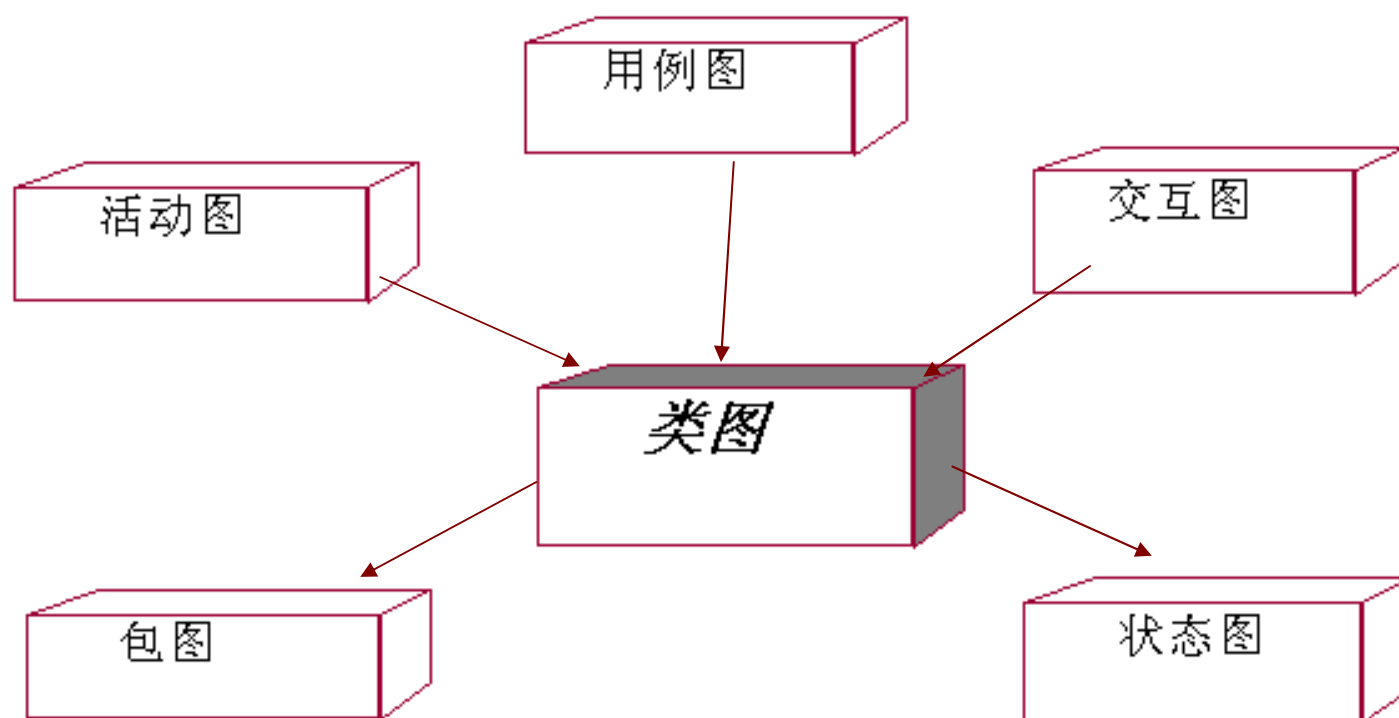


引言

类图是逻辑视图的重要组成部分，用于对系统的静态结构建模，涉及到具体的实现细节。

在系统分析阶段，类图主要用于显示角色和提供系统行为的实体的职责；在系统设计阶段，类图主要用于捕捉组成系统体系结构的类结构；在系统编码阶段，根据类图中的类及它们之间的关系实现系统的功能。

类图的地位和作用



1 基本概念

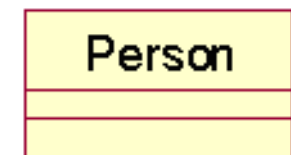
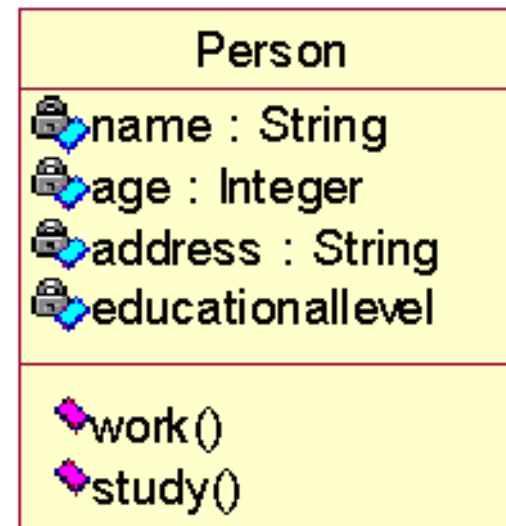
■ 1.1 类

– 定义

- 类是一组具有相同属性、相同行为、和其他对象有相同关系、有相同表现的对象描述
- 类具有属性和操作

– 图形表示

- 类名、属性、操作
- 可以简化表示





1 基本概念

■ 1.1 类

— 属性

- 一般描述类的某个特征，可以确定并区分对象以及对象的状态
- 完整的语法表示
 - [可见性]属性名[多重性][: 类型] = [初值][{特殊串}]

— 操作

- 操作是与类相关联的行为，表示类提供的服务
- 完整的语法表示
 - [可见性]操作名[(参数表)]: 返回类型 [{特殊串}]

1 基本概念

■ 1.1 类

– 对象

- 对象是一个类的实例，对象的每一个属性都有具体的值
- 图形表示
 - John是对象名，Person是类名
- 对象在交互图和状态机中使用

John : Person

– 接口

- 一组可重用的操作，描述类的部分行为
- 图形表示
 - 两种表示方法

<<Interface>>
Window



Window

1 基本概念

■ 1.2 关系

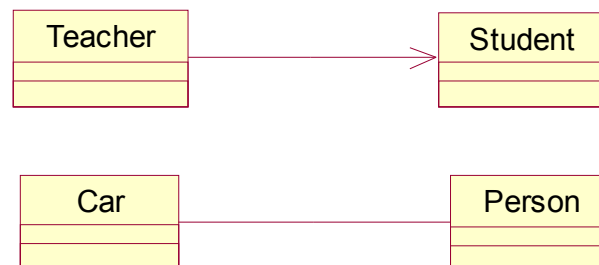
– 1) 关联 (association)

- 定义

- 两个类在概念上有连接关系时，类之间的连接称为关联；提供了不同类的对象可以相互作用的连接

- 图形表示

- 用一根连接类的实线表示，用箭头表示关联的方向；如果不明确指明方向，则默认关联是双向的



1 基本概念

■ 1.2 关系

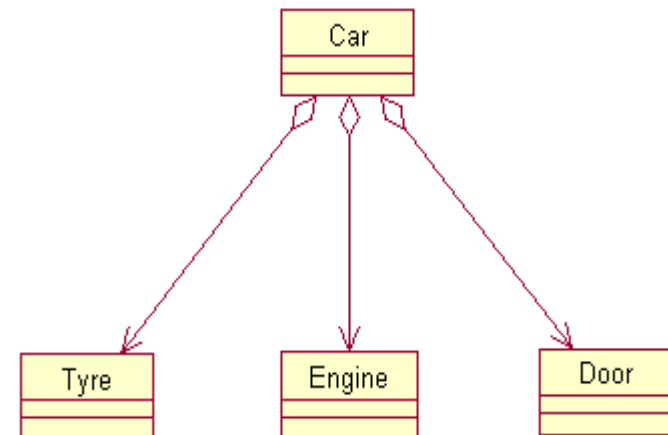
– 2) 聚合 (aggregation)

- 定义

- 类之间的一种整体与部分的关系
- 体现了一种层次结构，整体类位于部分类的上层，多个部分类处于并列的层次

- 图形表示

- 尾端带一个菱形的单箭头直线
- 菱形指向整体部分



1 基本概念

■ 1.2 关系

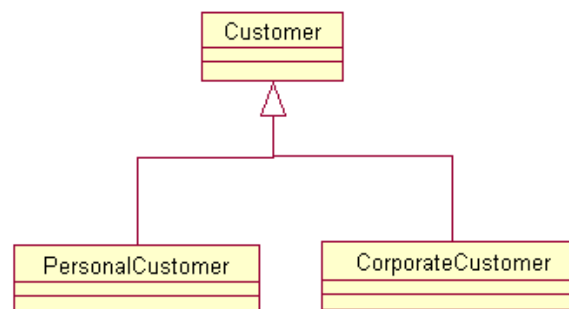
– 3) 泛化 (generalization)

- 定义

- 泛化关系是继承机制中产生的类与类之间的关系
- “is a part of”关系：一个事物是另一个事物的种类

- 图形表示

- 一条带有空心大箭头的有向实线，箭头指向父类



1 基本概念

■ 1.2 关系

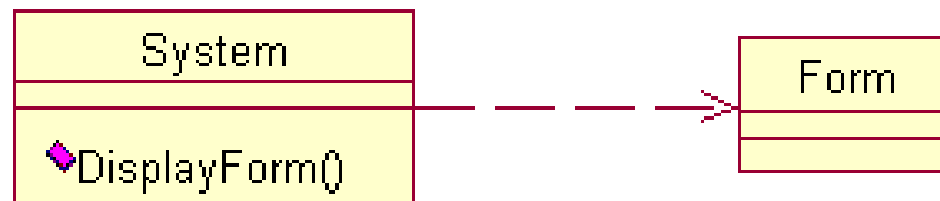
– 4) 依赖

- 定义

- 两个元素之间的一种关系，其中一个元素（提供者）的变化将影响另一个元素（客体），或向它提供所需信息
- 显示一个类引用另一个类

- 图形表示

- 用两个模型元素之间带箭头的虚线表示，箭尾处的模型元素（客户）依赖于箭头处的模型元素（提供者）



1 基本概念

■ 1.2 关系

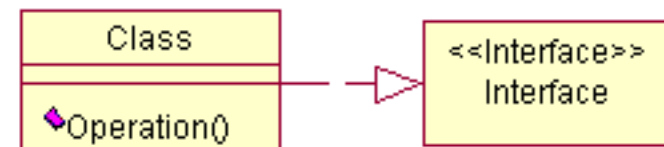
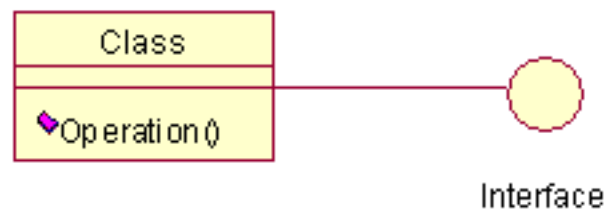
– 5) 实现

- 定义

- 类和接口之间的关系是实现关系，表示类实现接口提供的操作
显示一个类引用另一个类

- 图形表示

- 因接口的表示方法而异

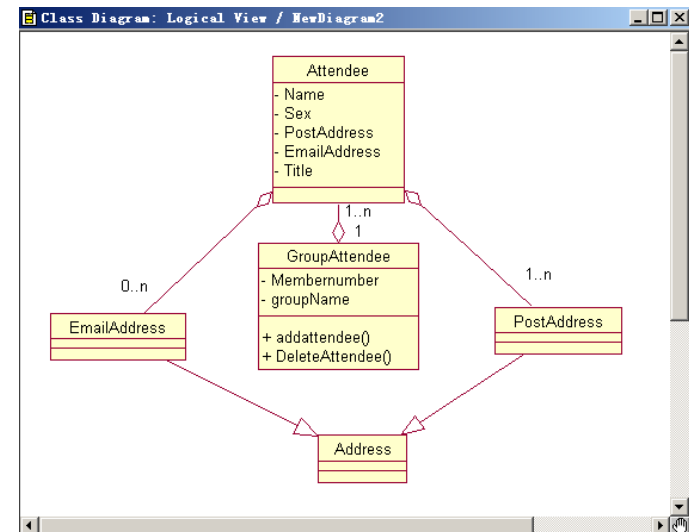


2 类图操作

■ 2.1 创建新的类图

— 步骤

- 右单击浏览器中的逻辑视图名称“Logical View”；
- 单击快捷菜单中的New > Class diagram；
- 输入新类图的名称。





2 类图操作














■ 2.2 删除类图

- 删除类图时，类图中的类并不删除，它们仍然在浏览器和其它模型图中
- 要删除一幅类图：
 - 右单击浏览器中的类图；
 - 单击快捷菜单中的Delete。

2 类图操作

■ 2.3 类图工具箱

- 常用工具如图
- 可以定制工具箱
 - 右单击工具箱;
 - 单击快捷菜单中的“Customize.....”;
 - 在弹出的“自定义工具栏”对话框中, 选择相应的工具图标, 然后单击“添加”或“删除”将选定的工具图标添加至窗口的工具箱中, 或从工具箱中移走。

工具	用途
	将光标变成箭头, 以便选择项目
	在模型图中添加文本框
	在模型图中添加注释
	连接注释和模型图项目
	在模型图中添加新类
	在模型图中添加新接口类
	在模型图中画关联关系
	在模型图中画聚合关系
	将关联类和关联关系链接起来
	在模型图中添加新的包
	在模型图中画依赖关系
	在模型图中画泛化关系
	在模型图中画实现关系

3 类规范与类的应用

■ 3.1 类规范

— 8个标签

- General标签
- Detail标签
- Operation标签
- Attributes标签
- Relations标签
- Components标签
- Nested标签
- Files标签

Class Specification for NewClass

Relations Components Nested Files

General Detail Operations Attributes

Name: NewClass Parent: Logical View

Type: Class

Stereotype

Export Control

☒ Public ☐ Protected ☐ Private ☐ Implementation

Documentation

OK Cancel Apply Browse Help

3 类规范与类的应用

■ 3.1 类规范

– 1) General标签

- 名称 (Name)
- 构造型 (Stereotype)
- 输出控制 (Export Control)

选项	含义
Public	系统中所有其它类都能访问这个类
Protected	这个类可以在嵌套类、友元或同一个类中访问
Private	这个类可以在友元或同一个类中访问
Implementation	这个类只能由同一包中的其它类访问

Class Specification for NewClass

Relations Components Nested Files

General Detail Operations Attributes

Name: NewClass Parent: Logical View

Type: Class

Stereotype

Export Control

☒ Public ☐ Protected ☐ Private ☐ Implementation

Documentation

OK Cancel Apply Browse Help

3 类规范与类的应用

■ 3.1 类规范

- 1) General标签
 - 构造型 (Stereotype)

构造型	
Actor	参与者
boundary	边界
Business actor	业务参与者
Business entity	业务实体
Business worker	业务工人
Control	控制
Domain	域
Entity	实体
Interface	接口
Table	表格
View	视图

3 类规范与类的应用

■ 3.1 类规范

– 2) Detail标签

- 多重性 (Multiplicity)
- 存储需求 (Space)
- 并发性 (Concurrency)

类型	描述
Sequential	默认设置。当只有一个控制线程时，类能正常工作；有多个控制线程时，不能保证类能否正常工作
Guarded	存在多个控制线程时，类正常工作但不同种的类应相互协作，保证不会互相干扰
Active	类有自己的控制线程
Synchronous	存在多个控制线程时，类的正常工作不需要与其他类相互协作，类本身能处理互斥情形

The image shows a 'Class Specification for transmitter' dialog box with the 'Detail' tab selected. The 'Multiplicity' is set to 'n'. The 'Space' field is empty. Under 'Persistence', 'Transient' is selected. Under 'Concurrency', 'Sequential' is selected. The 'Abstract' checkbox is unchecked. The 'Formal Arguments' section is empty. The dialog has buttons for 'OK', 'Cancel', 'Apply', 'Browse', and 'Help'.



3 类规范与类的应用

■ 3.1 类规范

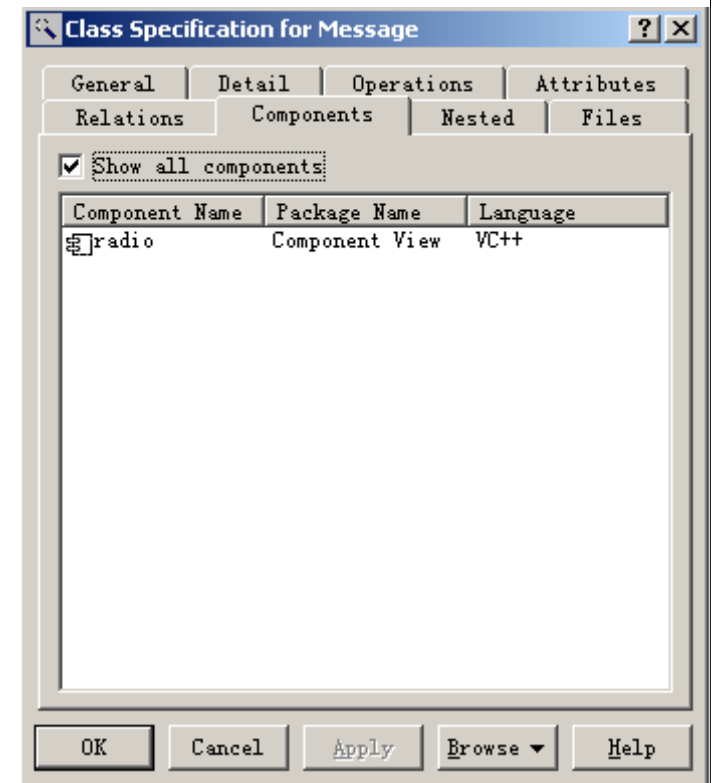
- 3) Attributes标签
- 4) Operation标签
- 5) Relations标签
- 6) Nested标签
 - 创建嵌套类
 - 右单击标签中任一空白处，从快捷菜单中选择Insert，输入嵌套类名称；
 - 按一般方法创建父类和嵌套类：在浏览器中，将嵌套类拖放到父类。

3 类规范与类的应用

■ 3.1 类规范

– 7) Components标签

- 显示用于实现系统中类的构件
- 复选标记框“Show all components”表示是否要显示模型中的所有构件；如果选中该标记，则显示系统中的所有构件；否则，只显示实现所选类的构件
- 显示构件时，同时还显示构件所属的包，以及实现所采用的编程语言





3 类规范与类的应用

■ 3.2 类的操作

— 添加一个类

- 单击工具箱中的类图标;
- 在模型图任一位置单击, 新类取名为NewClass;
- 输入新的类名。

— 删除一个类

- 单击模型图中要删除的类;
- 选择Edit > Delete from Model。

4 属性规范与应用

■ 4.1 属性规范

— 3个标签:

- General标签
- Detail标签
- DDL标签

The screenshot shows a dialog box titled "Class Attribute Specification for TimeLength". It has two tabs: "General" and "Detail". The "General" tab is active. The dialog contains the following fields and controls:

- Name:** A text field containing "TimeLength".
- Type:** A dropdown menu.
- Stereotype:** A dropdown menu.
- Initial:** A text field.
- Class Message:** A checkbox labeled "Show classes" which is checked.
- Export Control:** A group box containing four radio buttons: "Public", "Protected", "Private" (which is selected), and "Implementation".
- Documentation:** A large text area for documentation.
- Buttons:** "OK", "Cancel", "Apply", "Browse", and "Help" at the bottom.

4 属性规范与应用

■ 4.1 属性规范

— 1) General标签

- 属性数据类型 (Type)
- 属性构造型 (Stereotype)
- 属性的初值 (Initial)
- 输出控制 (Export Control)

Class Attribute Specification for TimeLength

General | Detail

Name: TimeLength Class Message

Type: [dropdown] ☒ Show classes

Stereotype: [dropdown]

Initial: [text box]

Export Control

☐ Public ☐ Protected ☒ Private ☐ Implementation

Documentation

[text area]

OK Cancel Apply Browse Help

选项	含义
Public	公共属性。任何其它类都可以浏览或修改属性的之值
Protected	保护属性。类及其派生类可以访问该属性
Private	专用属性。其他类不能访问该属性
Implemented	实现属性。只能被同一包中的类访问

可见性	Rose图注	UML图注
Public		+
Protected		#
Private		-
Implemented		<无>

4 属性规范与应用

■ 4.1 属性规范

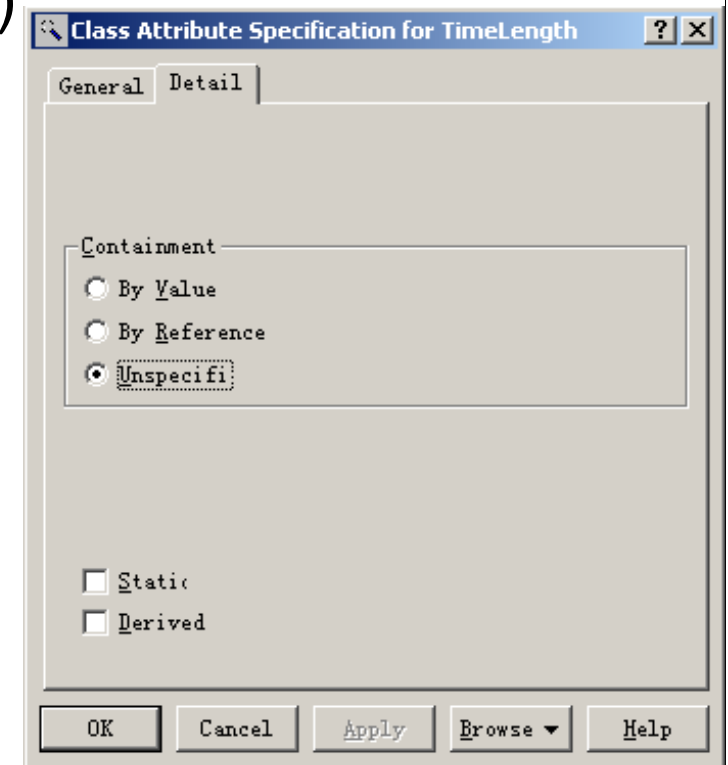
– 2) Detail标签

- 属性控制 (Containment)

- By value: 按数值
- By reference: 按引用
- Unspecified: 未指定

- 静态属性 (Static)

- 派生属性 (Derived)





4 属性规范与应用

■ 4.2 属性操作

— 增加一个属性

- 右单击浏览器或类图中的类，选择**New > Attribute**，用**Name:DataType = Initial Value**格式，输入属性名。或者，
- 打开类的规范窗口，选择“**Attributes**”标签，右单击属性区的任何位置，在快捷菜单中选择**Insert**。

— 删除一个属性

- 右单击浏览器某个属性，从快捷菜单选择**Delete**。或者，
- 选择类图中的某个属性，用退格键删除模型图中的属性名、数据类型和初始值，单击模型图中任一位置，确认删除。或者，
- 打开属性类的类规范窗口，选择“**Attributes**”标签，右单击要输出的属性，从快捷菜单中选择**Delete**，确认删除。

5 操作规范与应用

■ 5.1 操作规范

— 1) General标签

- 返回值的类型 (Return)
- 构造型 (Stereotype)
- 输出控制 (Export Control)

可见性	含义
Public	操作可以被其它类访问。
Protected	操作可以被子类、友元类或本身访问
Private	操作可以被友元类或本身访问
Implemented	操作是公开的，但只被同一包中的类访问

Operation Specification for appendAudioBlock

Preconditions Semantics Postconditions Files

General Detail Exceptions

Name: Class Message

Return ☒ Show classes

Stereotype:

Export Control

☒ Public ☐ Protected ☐ Private ☐ Implemented

Documentation

OK Cancel Apply Browse Help

5 操作规范与应用

■ 5.1 操作规范

– 2) Detail标签

- 操作变元 (argument)
- 操作协议 (Protocol)
- 操作限制 (Qualification)
- 内存长度 (Size)
- 时间 (Time)
- 并发性 (Concurrency)
 - Sequential: 只有一个控制线程时, 操作正常工作。
 - Guarded: 存在多个控制线程时, 不同线程的类相互协作, 操作可正常工作。
 - Synchronous: 存在多个控制线程时, 操作可正常工作。

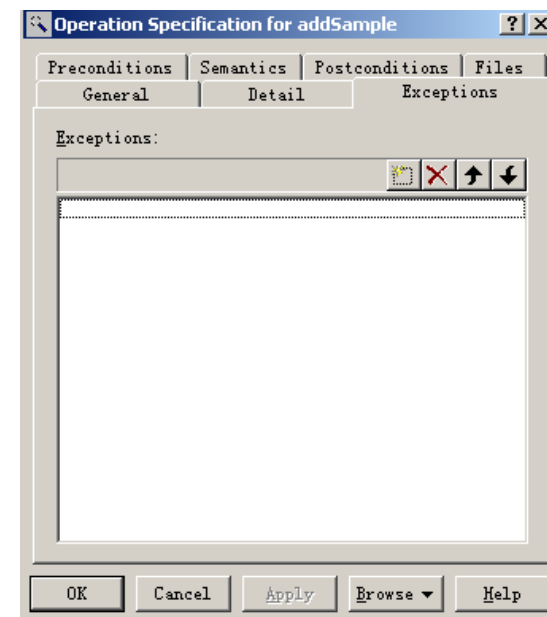
The screenshot shows a dialog box titled "Operation Specification for addcourse". It has several tabs: "Preconditions", "Semantics", "Postconditions", "Files", "General", "Detail", and "Exceptions". The "Detail" tab is currently selected. Inside the "Detail" tab, there is a section for "Arguments" with a table that has columns "Name", "Type", and "Default". Below this, there are input fields for "Protocol:", "Qualificat:", "Size:", and "Time:". At the bottom, there is a "Concurrency" section with three radio buttons: "Abstract", "Sequential" (which is selected), "Guarded", and "Synchronous". The dialog box also has "OK", "Cancel", "Apply", "Browse", and "Help" buttons at the bottom.

5 操作规范与应用

■ 5.1 操作规范

– 3) 异常 (Exceptions) 标签

- 操作可抛出的异常
- 通过其中的4个按钮添加、删除和上、下移动异常



5 操作规范与应用

■ 5.1 操作规范

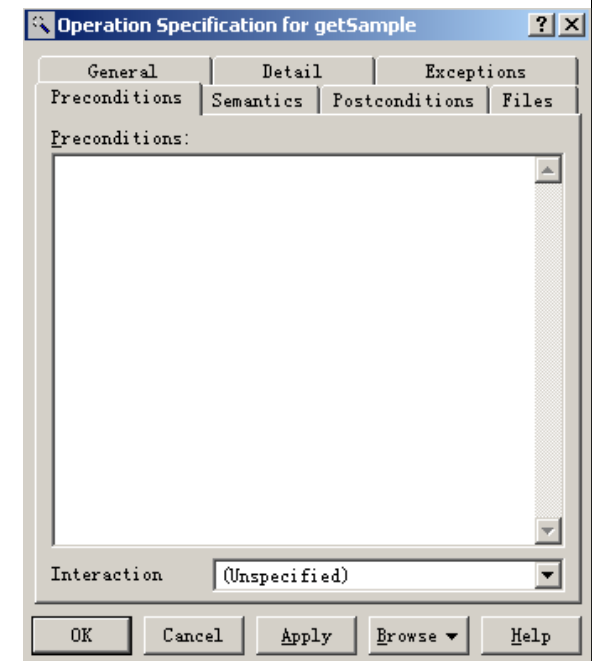
– 4) 前置条件 (Preconditions) 标签

- 前置条件 (Preconditions)

- 指定操作运行之前应满足的条件，即操作的入口行为，通常是不等式

- 交互图 (Interaction Diagram)

- 说明操作语义的交互图

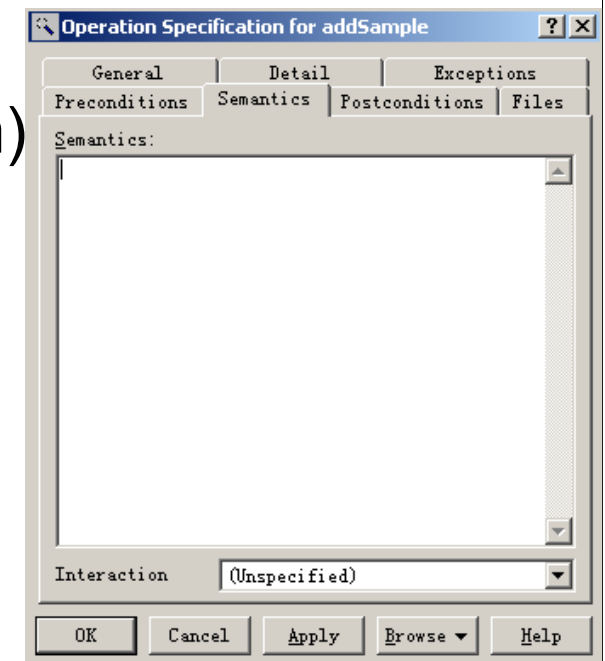


5 操作规范与应用

■ 5.1 操作规范

– 5) 语义 (Semantics) 标签

- 语义 (Semantics)
 - 指定操作的工作
 - 在文本框中用伪代码描述操作逻辑
- 交互图 (Interaction Diagram)
 - 说明了操作语义的交互图



5 操作规范与应用

■ 5.1 操作规范

– 6) 后置条件

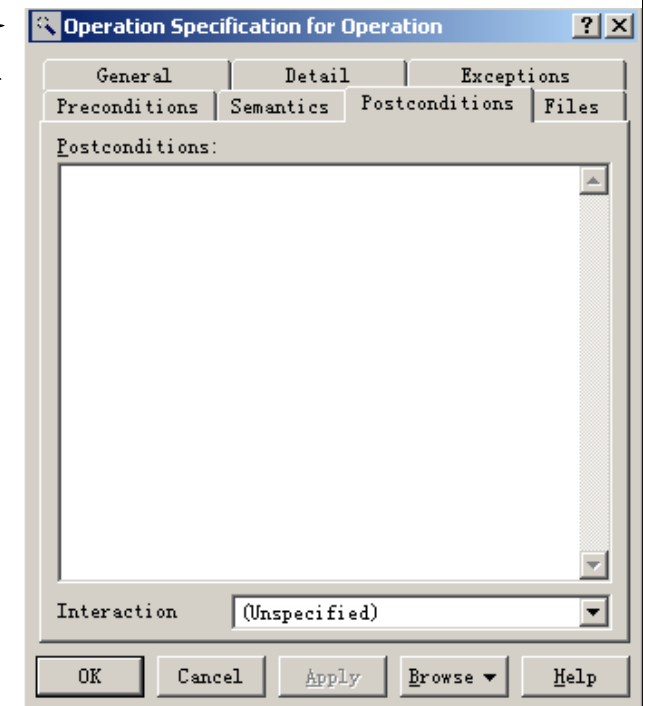
(**Postconditions**)标签

- 后置条件 (Postconditions)

- 指定操作运行之后要符合的条件，即操作的出口行为，通常是不等式

- 交互图 (Interaction Diagram)

- 说明操作语义的交互图





5 操作规范与应用

■ 5.2 操作应用

- 增加一个操作
 - 类似于添加一个属性的操作
- 删除一个操作
 - 类似于删除一个属性的操作

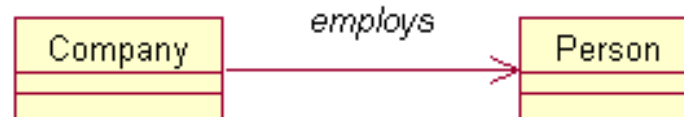
6 关联规范与应用

■ 6.1 关联规范

– 1) **General**标签

- 关系名 (Name)

- 一个动词或动词短语，描述关系的作用，是可选的
- 关系名在关系线上斜体显示



The screenshot shows the 'Association Specification for Untitled' dialog box. It has a title bar with a question mark and a close button. The dialog is divided into several tabs: 'Role B General', 'Role A Detail', 'Role B Detail', 'General', 'Detail', and 'Role A General'. The 'General' tab is currently selected. Inside the 'General' tab, there are fields for 'Name:' (empty), 'Stereotype:' (a dropdown menu), 'Role A:' (empty), and 'Role B:' (empty). To the right of these fields, there are labels for 'Parent:' (Use Case View), 'Element' (AudioSample), and 'Element' (AudioBlock). At the bottom of the dialog, there is a large 'Documentation' text area and a row of buttons: 'OK', 'Cancel', 'Apply', 'Browse', and 'Help'.

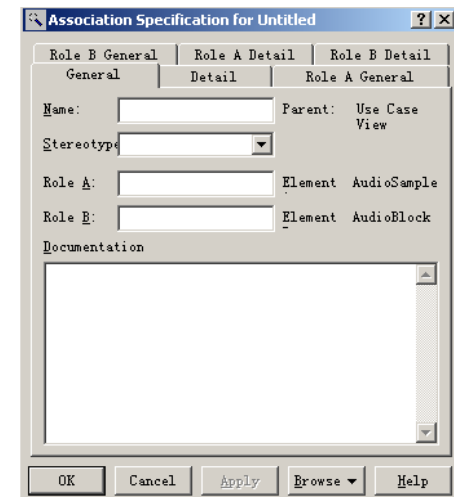
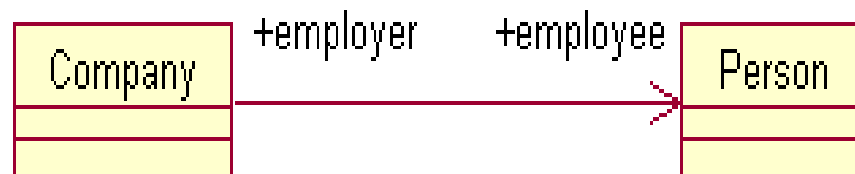
6 关联规范与应用

■ 6.1 关联规范

– 1) General标签

- 角色 (Role)

- 描述类在关系中的作用
- 通常是名词或名词短语，显示在起这个作用的类旁边
- 角色名前带有一个加号“+”，

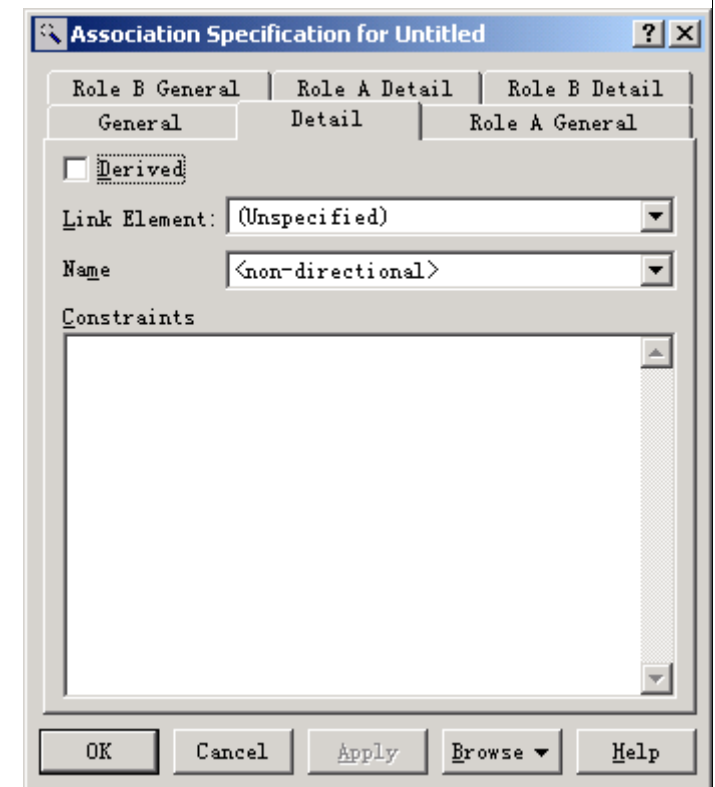


6 关联规范与应用

■ 6.1 关联规范

— 2) Detail标签

- 链接元素 (Link Element)
 - 链接元素也称为关联类，可以放置与关联相关的属性
- 使用限制 (Constraints)
 - 限制是必须符合的条件。在Rose中可以设置关系的限制条件
 - 输入的限制条件在生成代码时成为说明语句



6 关联规范与应用

■ 6.1 关联规范

– 3) Role A General 和Role B General 标签

- Rose在生成代码时要给关联建立一个属性
- Export Control字段设置该属性的可见性

可见性	含义
Public	公共属性。任何其它类都可以访问或修改属性的值
Protected	保护属性。类及其派生类可以访问该属性
Private	专用属性。其他类不能访问该属性
Implemented	实现属性。只能被同一包中的类访问

Association Specification for Untitled

Role B General | Role A Detail | Role B Detail

General | Detail | Role A General

Role: [] Element: MeetingRoom

Export Control

☒ Public ☐ Protected ☐ Private ☐ Implemented

Documentation

[]

OK Cancel Apply Browse Help

6 关联规范与应用

■ 6.1 关联规范

— 4) Role A Detail和Role B Detail标签

- 多重性 (Multiplicity)
 - 表示关联中一个类的几个实例与另一个类的一个实例相联系
- 包容 (Containment)
 - 设置关联属性是按值还是按引用包容

选项	含义
By value	按数值。属性放在类中
By reference	按引用。属性放在类外，类指向这个属性
Unspecified	未指定。还没有指定控制类型

- Static
 - 类所有的实例共享的属性
- Friend
 - 表示客户类能访问提供者类的非公共属性和操作

Association Specification for Untitled

General | Detail | Role A General
Role B General | Role A Detail | Role B Detail

Role: [] Element: MeetingRoom

Constraints

Multiplic 1 [] ☒ Navigable

☐ Aggregate ☐ Static ☐ Friend

Containment of MeetingInstance

☐ By Value ☐ By Reference ☒ Unspecified

Keys/Qualifie

Name	Type

OK Cancel Apply Browse Help

6 关联规范与应用

■ 6.2 Key/Qualifier规范

— 定义

- 即限定词，。在一对多的关联中，索引范围很广，使用限定符可以缩小关联的范围



- 按下面的步骤可以打开Key/Qualifier规范窗口：
 - 双击关联，打开关联规范窗口；
 - 单击“Role A Detail”标签；
 - 双击Key/Qualifier字段列表中的限定词。

6 关联规范与应用

■ 6.2 Key/Qualifier规范

- 定义限定词
 - 右单击关联规范窗口Key/Qualifier字段列表，从快捷菜单中选择Insert，直接在列表中输入限定词和类型
- 删除限定词
 - 右单击关联规范窗口Key/Qualifier字段列表，从快捷菜单中选择Delete
- 修改限定词
 - 单击关联规范窗口Key/Qualifier字段列表，直接在列表中编辑限定词和类型。

Association Specification for Untitled

General | Detail | Role A General
Role B General | Role A Detail | Role B Detail

Role: Element: Student

Constraints

Multiplic ☒ Navigable

☐ Aggregate ☐ Static ☐ Friend

Containment of Teacher

☐ By Value ☐ By Reference ☒ Unspecified

Keys/Qualifiers

Name	Type
Number	Integer

OK Cancel Apply Browse Help



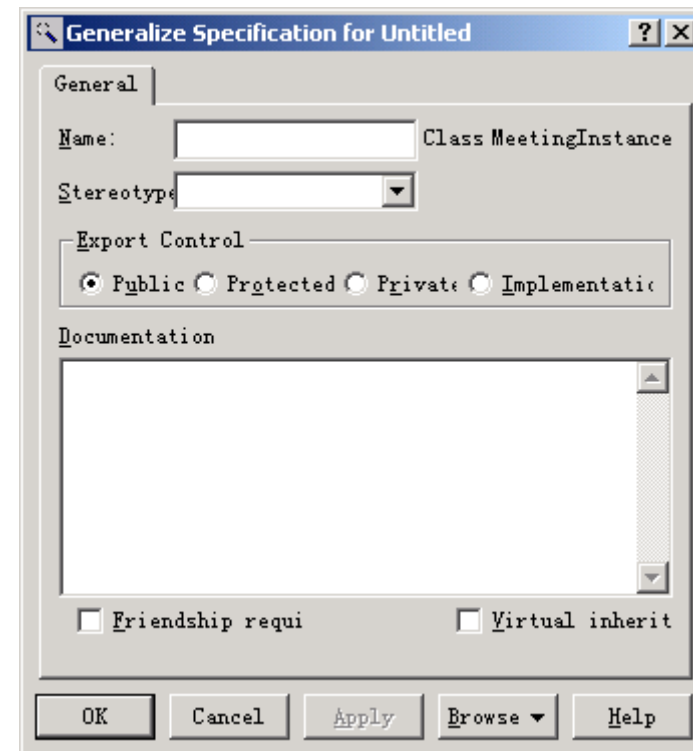
6 关联规范与应用

■ 6.3 关联的操作

- 增加关联
- 改变关系的导航
 - 在要移动的关系端单击右键，在快捷菜单中选择Navigable
- 删除关联

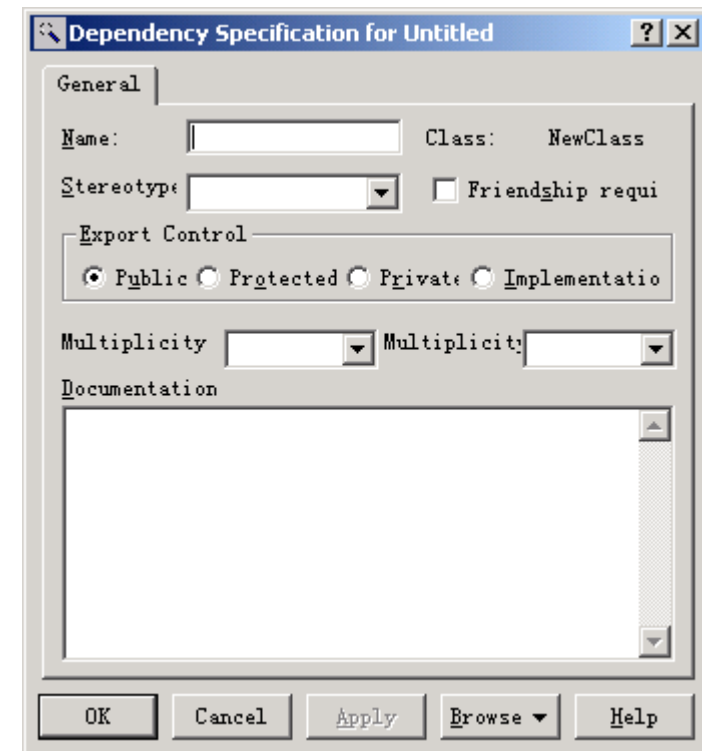
7 泛化规范与应用

- 7.1 泛化规范
 - **General** 标签
- 7.2 泛化操作
 - 创建一个泛化关系
 - 删除一个泛化关系



8 依赖规范与应用

- 8.1 依赖规范
 - General 标签
- 8.2 依赖操作
 - 创建一个依赖关系
 - 删除一个依赖关系





9 聚合规范与应用

- 9.1 聚合规范
 - 参考关联规范
- 9.2 聚合操作
 - 创建一个聚合关系
 - 删除一个聚合关系

10 逻辑包规范与应用

■ 10.1 逻辑包操作

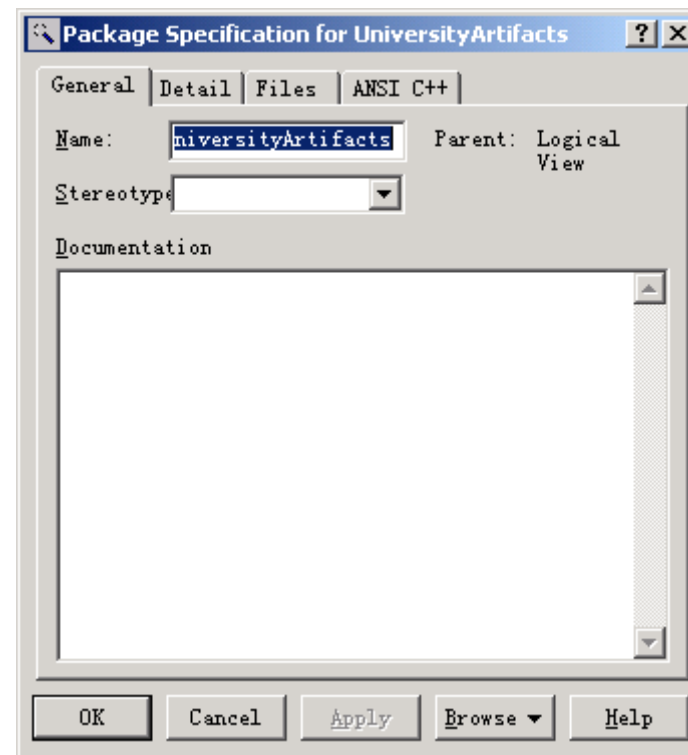
– 包的定义

- 包是一种分组机制，将模型中紧密相关的部分放在同一个包中，减少了模型的规模
- 包之间的关系表现为依赖关系、泛化关系和精化关系
- 包的操作

- 增加一个包
- 删除一个包

■ 10.2 逻辑包规范

- 4个标签，如图



第5章 交互图及其应用



《Rational Rose 2003基础教程》

配套电子教案



内 容

- 基本概念
- 顺序图
- 协作图
- 顺序图和协作图共有的元素
- 对象规范与应用
- 消息规范与应用
- 消息编号
- 协作图特有的要素——链
- 顺序图所特有的元素——控制焦点
- 顺序图与协作图之间的转换



引言

交互图可以用于对一个用例的事件流程进行建模，也可以单独使用，用于可视化、详述、构造和文档化一个特定对象群体的动态方面。交互图显示一个交互，由一组对象和它们之间的关系构成，其中包括：需要什么对象、对象相互发送什么消息、什么角色启动消息以及消息按什么顺序发送。



1 基本概念

■ 1.1 交互图

– 分类

- 交互图分为两种：顺序图和协作图
 - 顺序图强调消息发送的时间顺序
 - 协作图则强调接收和发送消息的对象的结构
- 顺序图和协作图语义等价
 - 顺序图和协作图在语义上是等价的，共享相同的基本模型
 - 两个图都可以表示另一个图所不能表示的某些东西
 - 顺序图和协作图可以实现两者之间的等价转换，而不丢失任何信息



1 基本概念

■ 1.2 对象

- 对象是类的实例，具有特定的属性和操作。在交互图中，属性展示了对象的信息和状态，操作展示了对象的行为和功能。



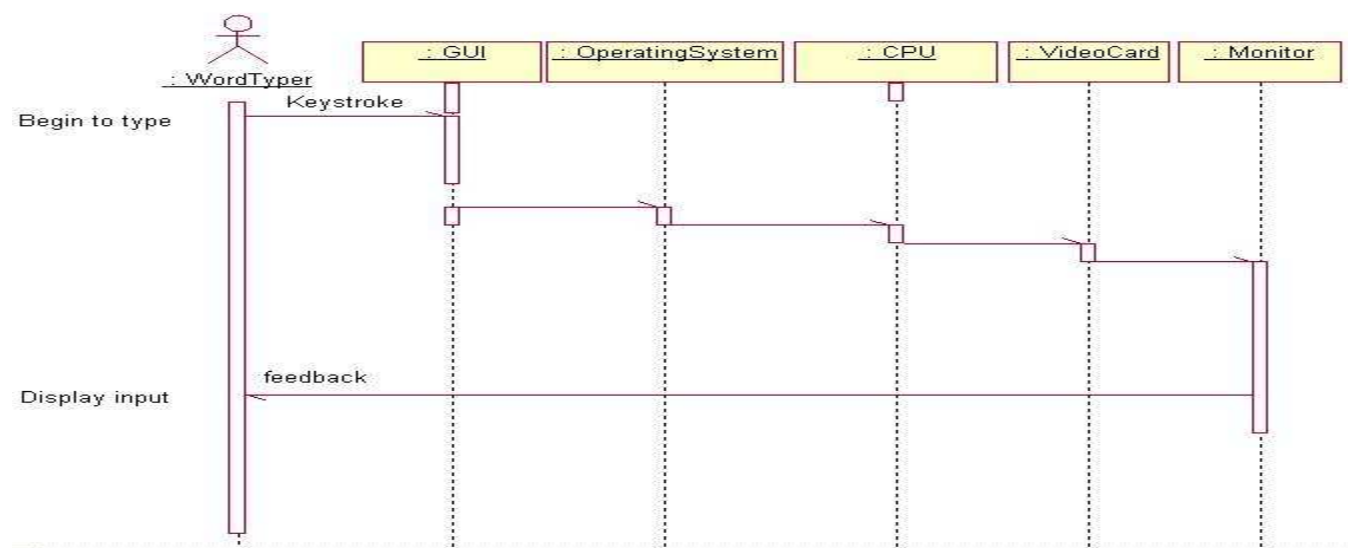
1 基本概念

■ 1.3 消息

- 消息是从一个对象到另一个或者几个其他对象的信息传递，简单地说，消息就是对象与对象、参与者与参与者，或者对象与参与者之间的某种通信方式。
- 消息可以是一个信号或一次操作调用，收到消息即为事件。可以有两种消息，一种是从发送者向接收者发送信号，另一种是由调用者调用接收者的操作
- 对象之间的协作通过相互发送消息实现。

2 顺序图

顺序图是描述消息时间顺序的交互图。在图形上，顺序图是一张表，其中显示的对象沿横轴排列，从左到右分布在图的顶部；而消息则沿纵轴按时间顺序排序。创建顺序图时，以能够使图尽量简洁为依据布局。



2 顺序图

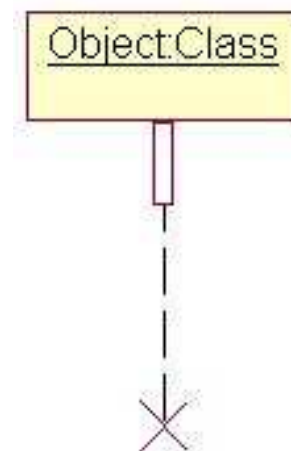
■ 2.1 顺序图中的要素

— 顺序图包含的内容

- 1) 对象
- 2) 对象生命线
- 3) 消息
- 4) 控制焦点

— 对象、对象生命线和控制焦点

- 表示方法
 - 对象向下方垂直伸展的虚线称为该对象的生命线，表示该对象存在的时间长度
 - 大“X”标记表明生命的结束
 - 在生命线上的窄矩形条称为控制焦点，控制焦点表示该对象正在执行某个操作。窄矩形的长度表示操作的持续时间。



2 顺序图

■ 2.1 顺序图中的要素

— 消息的表示方法

- 用消息线（消息图标）表示。一个对象到另一个对象的消息用跨越对象生命线的一条消息线表示出来，包括反身消息。
- 三种表示消息的方法
 - 消息线本身
 - 消息线加消息编号
 - 消息线加消息编号加消息名



2 顺序图

■ 2.1 顺序图中的要素

– 消息的分类

- UML的消息分为：
 - 简单（simple）
 - 同步（synchronous）
 - 异步（asynchronous）
- Rose的消息分类
 - 7种类型，见消息规范





2 顺序图

■ 2.2创建顺序图

– 创建方法

- 单击Browse > Interaction Diagram，弹出Select Interaction Diagram对话框；
- 在对话框左边选择预建立顺序图的包，单击OK。弹出New Interaction Diagram对话框；
- 在Title字段输入新框图的名字，单击diagram type中的Sequence，单击OK；

2 顺序图

2.3 在顺序图中添加脚本

— 目的

- 对消息增加说明
- 移动顺序图中的消息时，相应的脚本也将随之移动到新的位置
- 要避免在框图中放入太多的逻辑条件，以保持框图的简洁性

— 添加方法

- 选择工具箱中的文本图标；
- 单击框图中要放入脚本的位置，通常将脚本放在框图的左边；
- 在文本框中键入脚本文本；
- 选中脚本文本框，按下shift键后选中脚本所描述的消息；
- 选择Edit > Attach Script。

— 撤销方法

- 选择脚本和消息中的任意一项；
- 选择Edit > Detach Script。



2 顺序图

■ 2.4 顺序图的删除

— 方法步骤

- 右键单击浏览器中的顺序框图名；
- 在快捷菜单中选择“Delete”。

2 顺序图

- 2.5 顺序图工具箱
 - 12种常见工具按钮
 - 可以定制

按钮	用途
	将光标变成箭头，以便选择项目
	在框图中添加文本框
	在框图中添加注释
	连接注释和框图项目
	在框图中添加新对象
	在两个对象之间绘制消息
	绘制反身消息
	显示过程调用返回的消息
	显示对象删除
	绘制两个对象之间过程调用
	绘制两个对象之间的异步消息
	锁定按钮



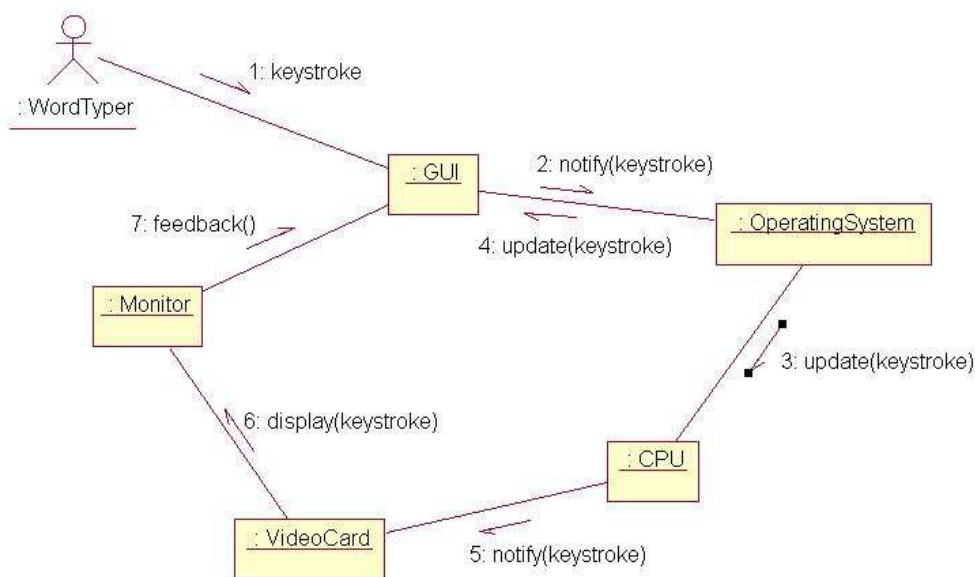
3 协作图

协作图强调发送和接受消息的对象之间的结构组织的交互图，显示对象、对象之间的链接以及对象之间的消息，还可以显示当前模型中的简单类实例和类实体实例。

3 协作图

■ 3.1 协作图包含的元素

- 对象
- 链
- 消息



3 协作图

■ 3.2 协作图的创建和删除

- 创建一个新协作图的步骤如下：
 - 1) 单击Browse > Interaction Diagram。弹出Select Interaction Diagram对话框；
 - 2) 在对话框左边选择预建立协作图的包，单击OK。弹出New Interaction Diagram对话框；
 - 3) 在Title字段输入新框图的名字，单击diagram type中的Collaboration，单击OK按钮。
- 删除一个协作图的步骤如下：
 - 1) 右键单击浏览器中的协作框图名；
 - 2) 在快捷菜单中选择“Delete”。

3 协作图

■ 3.4 协作图工具箱

- 13种常见工具
- 可以定制

按钮	用途
	将光标变成箭头，以便选择项目
	在框图中添加文本框
	在框图中添加注释
	连接注释和框图项目
	在框图中添加新对象
	在框图中添加新的类实例
	创建对象之间的链接
	创建对象反身链接
	在两个对象之间或一个对象本身增加消息
	在两个对象之间或一个对象本身从相反方向增加消息
	显示两个对象之间的信息流
	在相反方向显示两个对象之间的信息流
	锁定按钮

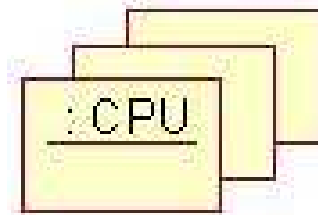
4 顺序图和协作图共有的元素

■ 4.1 对象

— 表示方法

- Object指明对象，Class定义Object的类型
- 如果对象已经映射到类，用户便可以在Rose中选择性的显示对象名、类名
- 可以用一个多实例图标表示类的多个实例


Object:Class

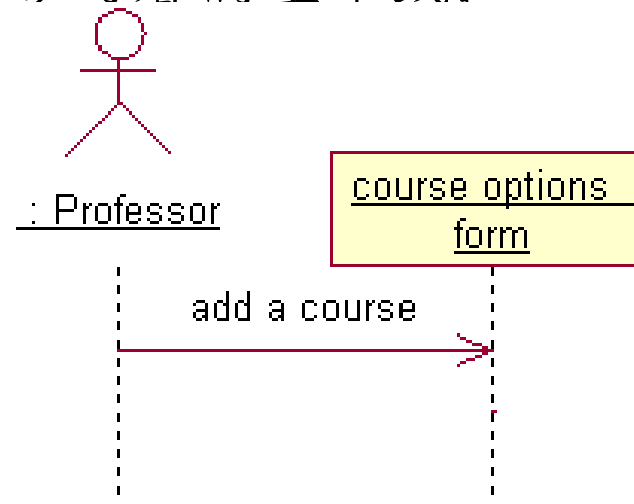


4 顺序图和协作图共有的元素

■ 4.2 消息

– 1) 将消息加进顺序图


- 单击工具箱中的 按钮;
- 将鼠标从发送消息的对象或参与者的生命线拖动到接受消息的对象或参与者的生命线:
- 输入消息文本。

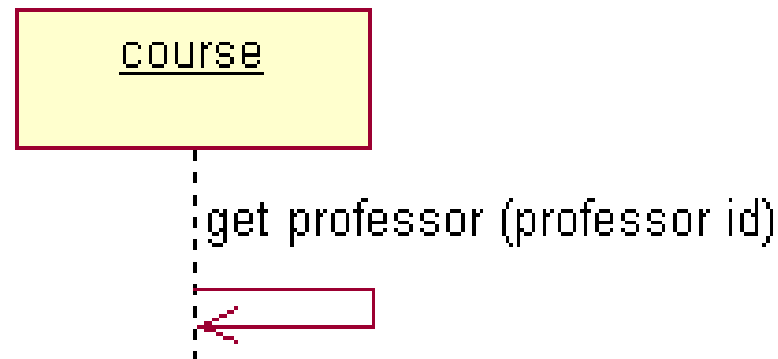


4 顺序图和协作图共有的元素

■ 4.2 消息

– 2) 将反身消息加进顺序图


- 单击工具箱中的  按钮;
- 单击收发消息的对象生命线;
- 输入消息名。



4 顺序图和协作图共有的元素

■ 4.2 消息

– 3) 将消息加进协作图

- 单击工具箱中的  按钮;
- 单击框图中对象间的链接;
- 输入消息名。

– 4) 删除消息

- 选择要删除的消息;
- 选择 Edit > Delete from Model 或按 Ctrl+D。
- 注意：在协作图中消息删除后，链接仍然存在于模型中



4 顺序图和协作图共有的元素

■ 4.3 消息编号

- Rose中默认顺序图中不显示消息编号，协作图显示消息编号。顺序图中，消息根据对象生命线从上往下从1开始进行编号；协作图中，消息根据建立的顺序从1开始进行编号。删除消息时，Rose自动将其余消息重新编号。



4 顺序图和协作图共有的元素

■ 4.3 消息编号

- 打开或关闭消息编号：
 - 选择Tools > Options;
 - 选择Diagram标签;
 - 复选或取消Collaboration Numbering 或者 Sequence Numbering。



4 顺序图和协作图共有的元素

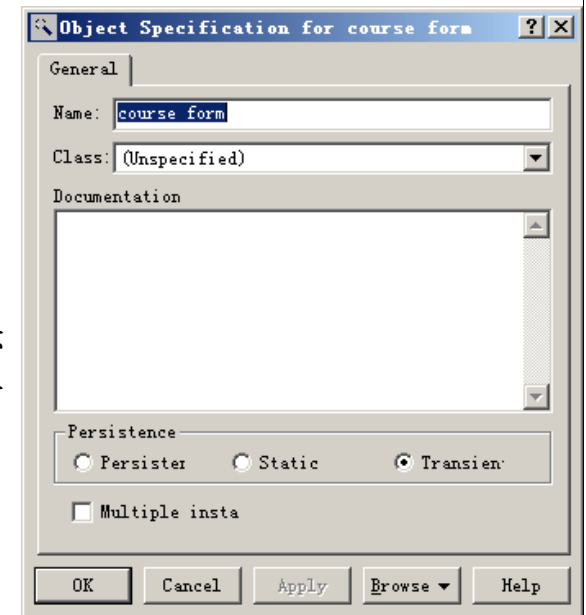
■ 4.4 为消息分配操作

- 要给消息分配操作：
 - 双击框图中的消息图标;
 - 从弹出的列表选择一个操作。或者单击 <new operation>定义一个新操作。
- 改变消息指定的操作：
 - 双击框图中的消息;
 - 在消息规范窗口“Name”字段列表框中选择新的操作名。

5 对象规范与应用

■ 5.1 对象规范

- 1) 映射类 (**Class**)
 - 在Class下拉列表框中选择类名；或者，
 - 将浏览器中的类拖动到框图中的对象上。
- 2) 对象持续性 (**Persistence**)
 - Persistent: 对象保存到数据库或者其他形式的永久存储体中
 - Static: 对象保存在内存中直到程序中止
 - Transient: 对象只在短时间内保存在内存中
- 3) 多实例 (**Multiple instances**)
 - 选中该复选框之后，协作图中的对象图标将显示为多实例图标。但在顺序图中，对象图标仍然是单个对象的图标。

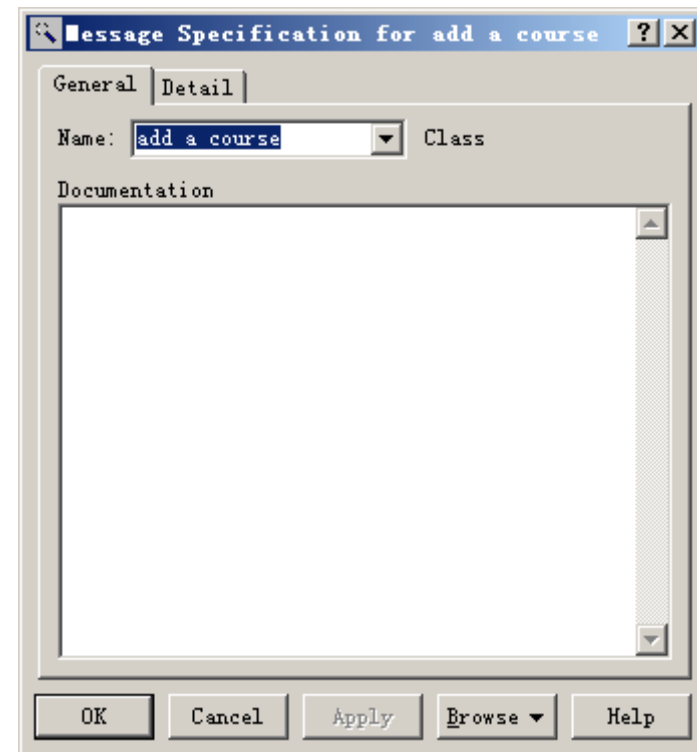


6 消息规范与应用

■ 6.1 消息规范

– 1) **General**标签

- Name: 消息名
- Documentation: 对消息的附加说明

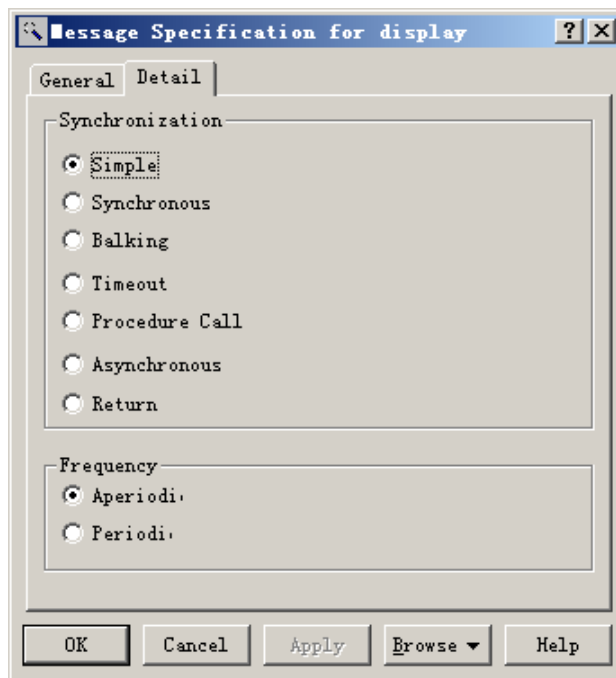


6 消息规范与应用

6.1 消息规范

– 2) Detail标签

- 消息同步类型
(Synchronization)



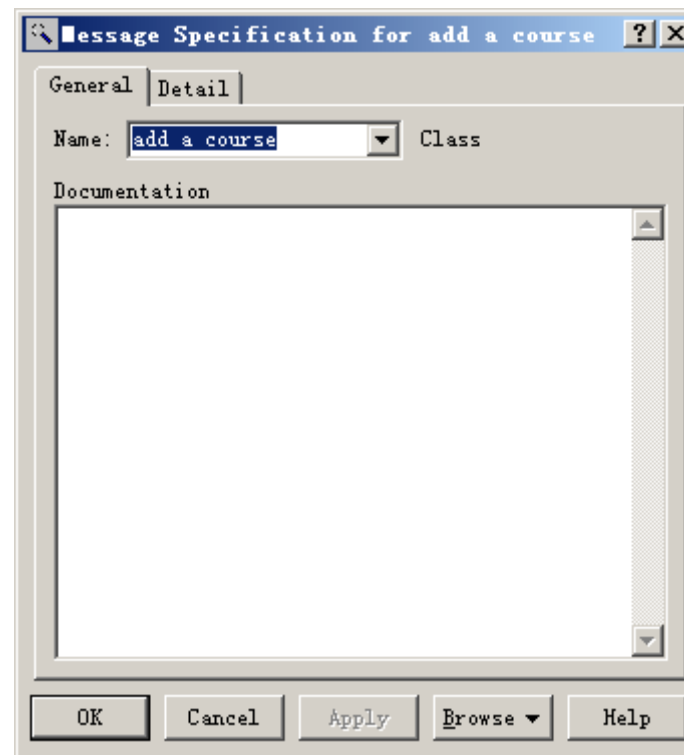
选项	Rose图标	含义
Simply		默认值。
Synchronous		同步。客户（发送消息的对象）发出消息后，等待供应者（即消息接收者，或操作提供者）响应消息。客户等供应者执行完操作之后才能继续其本身的进程，在执行操作的这段时间内，客户一直处于等待状态，直到它收到从操作提供者那里返回的消息。
Balking		阻止。客户只能在操作提供者能立即接收消息的情况下发送消息。如果操作提供者没有准备好接收消息，则客户将放弃该消息。
Timeout		超时。客户发出消息并指定等待时间，如果供应者不能在指定时间内处理消息，则客户将放弃该消息。
Asynchronous		异步。客户发出消息后，不等待消息是否接收，无需等待供应者的应答，可以直接继续自身的操作。
Procedure Call		过程调用。客户发出消息，等待处理消息的整个嵌套顺序完成之后才能继续。
Return		返回。从过程调用返回。返回箭头可以省略，因为这种消息隐式地表示一个活动的结束。

6 消息规范与应用

■ 6.1 消息规范

– 1) General标签

- Name: 消息名
- Documentation: 对消息的附加说明

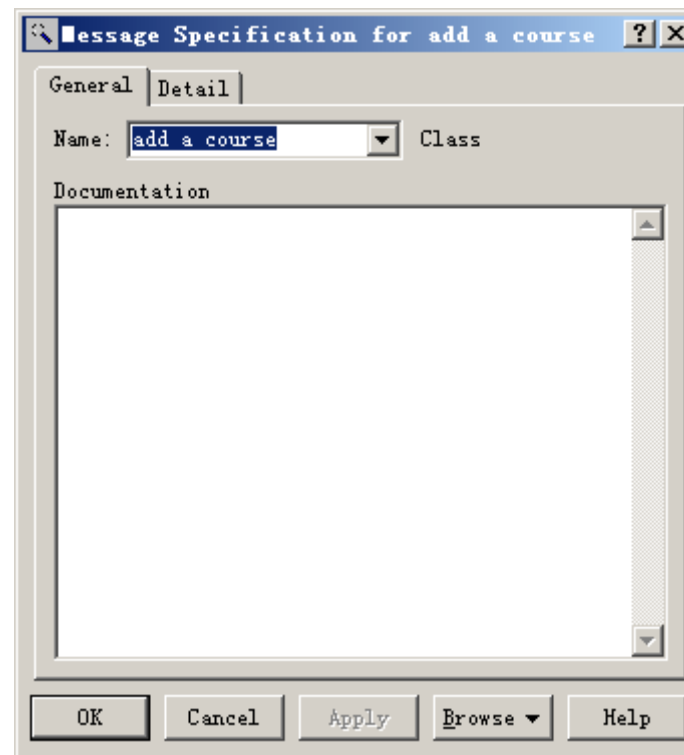


6 消息规范与应用

■ 6.1 消息规范

– 1) General标签

- Name: 消息名
- Documentation: 对消息的附加说明

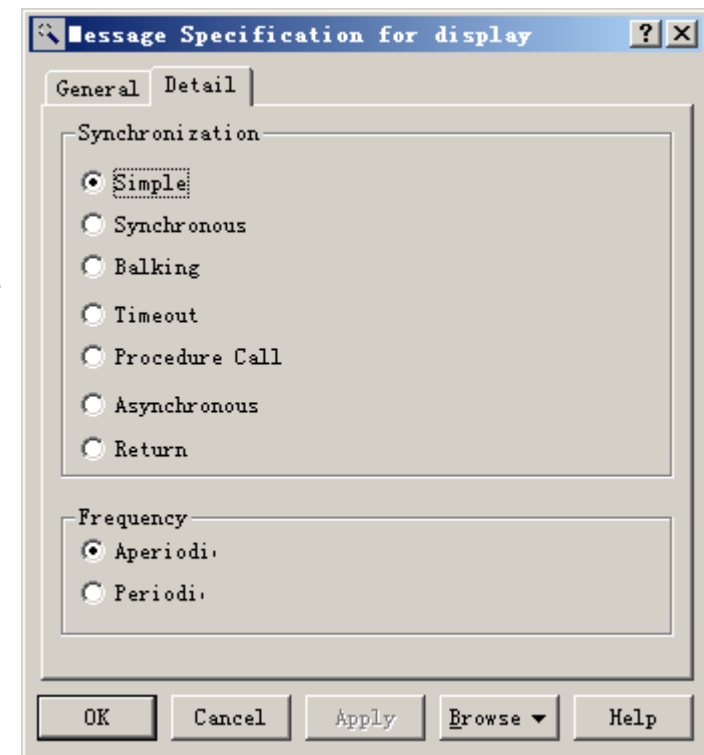


6 消息规范与应用

■ 6.1 消息规范

– 1) Detail标签

- 消息频率 (Frequency)
 - 消息频率可以让消息按规定的
时间间隔发送
 - 不定期 (Aperiodic): 将消
息设定为不定期发送的消息、
只发送一次, 或者不规则时
间间隔发送
 - 定期 (Periodic): 将消息
设定为定期发送的消息





7 消息编号

消息编号用来表示一个消息的时间顺序，通过消息的顺序编号可以更清楚地看出各消息之间的时间顺序以及相互之间的关系。

在交互图中可以选择显示消息编号。在顺序图中，消息可以按两种方式编号：Top-Level（顶级编号）方式，如1、2、3；或者Hierarchical（等级编号）方式，如1.1、1.1.2、1.1.3。在协作图中，消息只能采用Top-Level编号，但如果协作图是由顺序图转换而来，图中也可以使用Hierarchical编号。

7 消息编号

■ 7.1 Top-Level 编号

– 定义

- 顶级编号。编号由1开始，后续消息的编号单调递增，消息编号只有一个单一的数字，没有数字子集。

– 举例

- 如用户设定了编号为1., 1.1., 1.2.和1.2.1.的消息，在顶级编号时将被编号为1, 2, 3和4

– 一般用于对象和消息很少的顺序图中

– 要对消息进行顶级编号：

- 单击Tools > Options;
- 单击Diagram标签;
- 选中Sequence Numbering复选框。

7 消息编号

■ 7.2 Hierarchical编号

– 定义

- 即等级编号，Hierarchical 编号使用带小数点的号码，方便显示消息嵌套。

– 举例

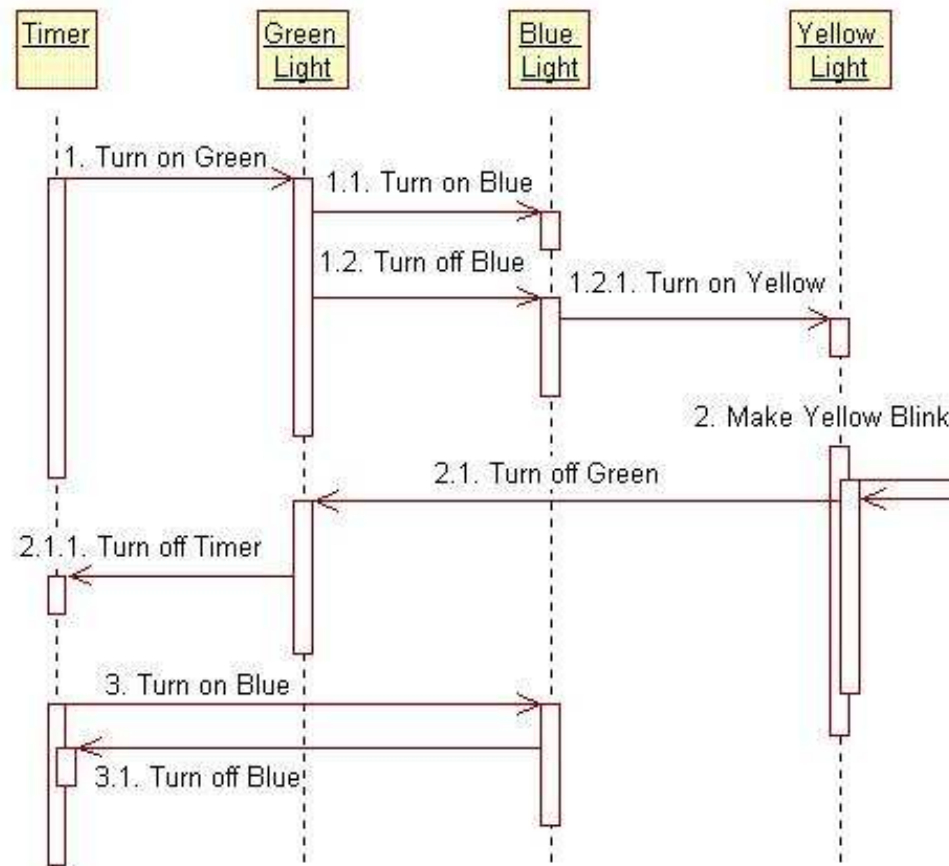
- 如1表示第一个消息，1.1表示嵌套在消息中的第一个消息，1.2表示嵌套在消息中的第二个消息。注意：如果删除消息1，则嵌套在消息1中的其他消息也将同时被删除。

– 要对消息进行等级编号：

- 选择Tools > Options;
- 单击Diagram标签;
- 同时选中Sequence Numbering和Hierarchical Messages复选框。

7 消息编号

■ 7.2 Hierarchical编号







8 协作图特有的要素——链

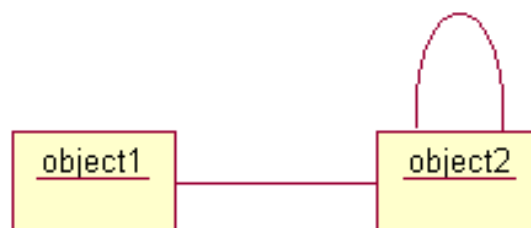
■ 8.1 链的定义

- 链是关联的实例，当一个类与另一个类之间有关联时，这两个类的实例之间就有链，一个对象就能向另一个对象发送消息。所以链是对象间的发送消息的路径。
- 要在协作图中增加消息，必须先建立对象之间的链接。
- 链接一般建立在两个对象或者两个类实例之间，也可以建立反身链接。

8 协作图特有的要素——链

■ 8.2 链的操作

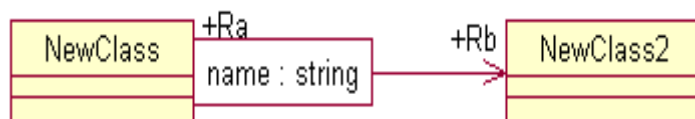
- 创建链接：
 - 单击工具箱中的  按钮；
 - 从一个对象拖动到另一个对象，创建链接。
- 创建反身链接
 - 单击工具箱中的  按钮；
 - 单击发送消息的对象，建立反身链接。



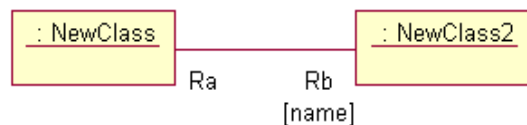
8 协作图特有的要素——链

■ 8.3 链的规范

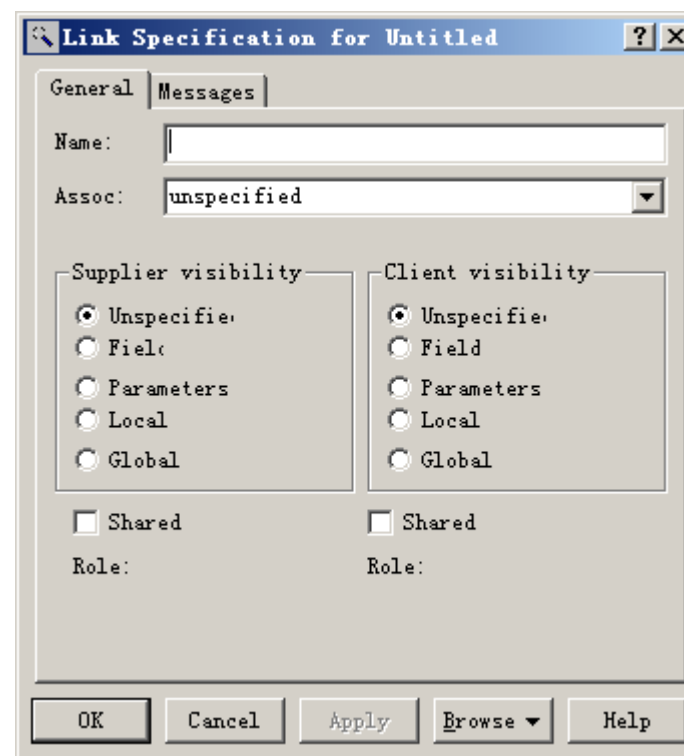
- 1) General标签
 - 关联 (Assoc)：两个对象的类之间建立的关联



类关联



对象之间的链接



8 协作图特有的要素——链

■ 8.3 链的规范

— 1) General标签

- 可见性 (Visibility)：一个对象看到另一个对象的能力

选项	含义
Unspecified (Default)	不指定对象的可见性（默认值）
Field	供应者对象可见。
Parameters	供应者对象可见。供应者对象是客户对象操作中的一个参数。
Local	供应者对象局部可见。供应者对象是客户对象操作中的一个局部变量。
Global	供应者对象全局范围内可见。

The screenshot shows a dialog box titled "Link Specification for Untitled". It has two tabs: "General" and "Messages". The "General" tab is active. It contains the following fields and options:

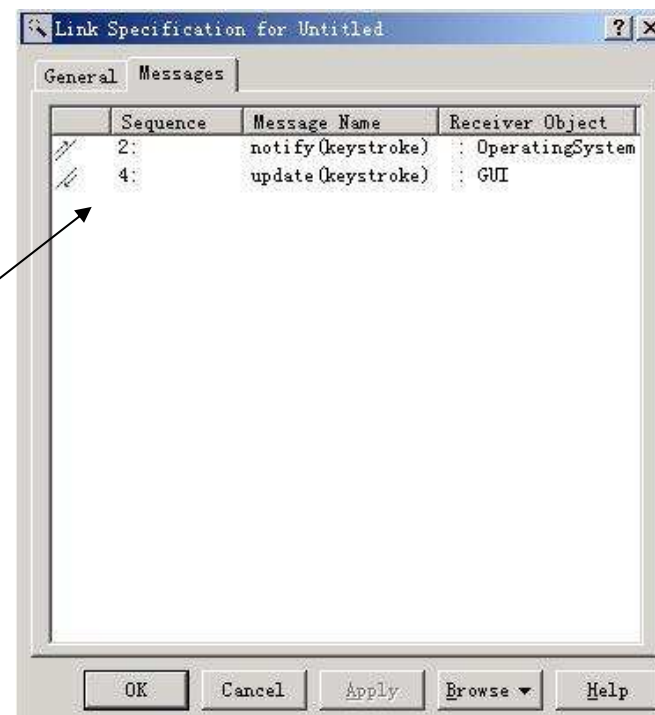
- Name:** A text input field.
- Assoc:** A dropdown menu currently showing "unspecified".
- Supplier visibility:** A group box containing five radio buttons: "Unspecified" (selected), "Field", "Parameters", "Local", and "Global".
- Client visibility:** A group box containing five radio buttons: "Unspecified" (selected), "Field", "Parameters", "Local", and "Global".
- Shared:** Two checkboxes, one for "Supplier" and one for "Client", both currently unchecked.
- Role:** Two text input fields, one for "Supplier" and one for "Client".
- Buttons:** "OK", "Cancel", "Apply", "Browse", and "Help" at the bottom.

8 协作图特有的要素——链

■ 8.3 链的规范

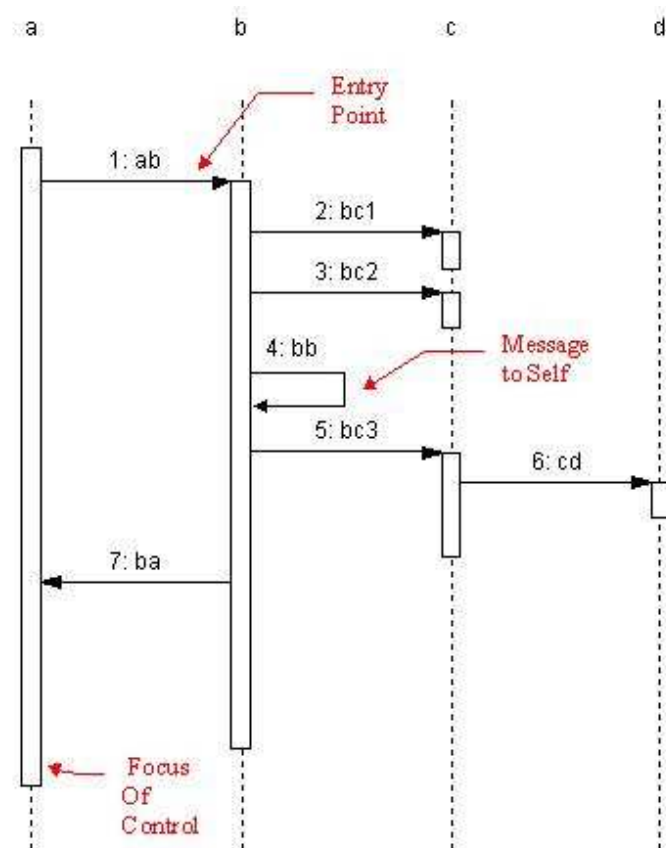
— 2) Messages标签

- 显示了当前链接上的消息信息
- 右图说明:
 - 一个是正向的链消息（由客户发给供应者），一个是逆向的链消息（由供应者返回给客户）；消息的编号分别为2和4；消息名分别为notify(keystroke)和update(keystroke)，接收对象分别为OperatingSystem类的实例和GUI类的实例。



9 顺序图所特有的元素——控制焦点

控制焦点是顺序图所特有的元素。控制焦点是对象生命线上一个窄矩形，用于装饰对象生命线，表示对象执行一个动作所经历的时间长度。矩形的顶部表示动作的开始，底部表示动作的结束。上下移动控制焦点时，依附在其上的每个独立的消息线也将发生相应的移动。





9 顺序图所特有的元素——控制焦点

■ 9.1 显示控制焦点

- 显示或关闭控制焦点的方法：
 - 单击Tools > Options;
 - 单击Diagram标签;
 - 选中“Focus of Control”左边的复选框，表示在顺序图中显示控制焦点；否则，在顺序图中不显示控制焦点。

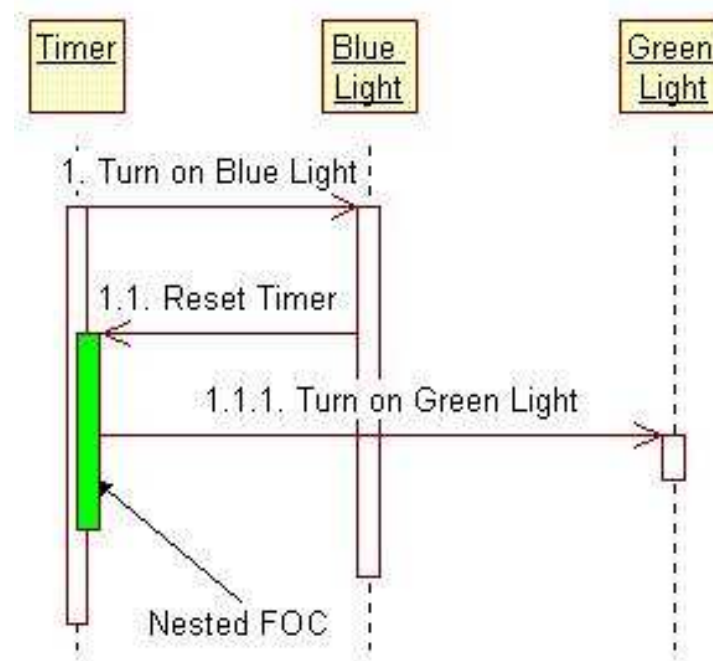
■ 9.2 控制焦点着色

- 单击箭头指向控制焦点的消息。
- 选择Format > Fill Color;
- 在颜色框中单击要选择的填充颜色;
- 单击OK。

9 顺序图所特有的元素——控制焦点

■ 9.3 控制焦点的嵌套

- 嵌套的控制焦点依附在另外一个控制焦点之上，它可以确切地区分一个消息开始和终止的具体位置。
- 要显示嵌套的控制焦点，必须同时选中Options对话框中Diagram标签中的Hierarchical Messages和Focus of Control两个选项。





9 顺序图所特有的元素——控制焦点

■ 9.4 移动控制焦点

- 移动控制焦点的方法：
 - 选中要移动的控制焦点上的第一个消息线；
 - 按下Alt键；
 - 单击并拖动源消息至目标位置。



10 顺序图与协作图之间的转换

■ 10.1 从顺序图转换成协作图

- 从sequence框图创建collaboration框图：
 - 打开sequence 框图；
 - 选择Browse > Create collaboration Diagram; 或者，直接按下F5键；
 - 浏览协作图并调整图中模型元素的位置。

■ 10.2 从协作图转换成顺序图

- 从collaboration 框图创建sequence框图：
 - 打开collaboration 框图；
 - 选择Browse > Create Sequence Diagram; 或者，直接按下F5键；
 - 浏览顺序图并调整图中模型元素的位置。

第6章 状态机图及其应用



《Rational Rose 2003基础教程》

配套电子教案



内 容

- 基本概念
- 状态图
- 活动图
- 状态机共享的模型元素
- 活动图专有的模型元素
- 状态规范和活动规范
- 动作规范
- 转换规范
- 判断规范
- 同步规范



1 基本概念

■ 1.1理解状态机

- 状态机图通过对类对象的生存周期建立模型来描述对象随时间变化的动态行为，也可以用来描述用例、协作和方法的动态行为，它是展示状态与状态转换的图。
- 状态机是一个类的对象所有可能的生命历程的模型。
- 状态机包括状态图和活动图两种表示方法。
 - 状态图用于对系统的动态方面建模。
 - 活动图用于对计算流程和 workflows 建模，展示的主要内容是对象的活动状态。
 - 状态图以状态为中心，活动图以活动为中心。



1 基本概念

■ 1.2 状态机操作

– 定义

- 在UML规格文件中，状态机被定义为是一种行为，说明对象或交互在它们的声明周期中为响应事件所经历的状态序列，以及它们的响应和动作。
- 创建一个状态图或活动图时，Rational Rose自动创建一个状态/活动模型。
- Rational Rose限定每个所有者只能拥有一个状态/活动模型。



1 基本概念

■ 1.2 状态机操作

- 创建一个状态/活动模型：
 - 单击Browse>State Machine Diagram, 在左边的“State Machine”下面选择新创建的状态/活动模型所要放置的位置;
 - 双击New:
 - 在Title文本框中为图命名;
 - 指定要创建的图类型: 活动图 (Activity) 或状态图 (Statechart);
 - 单击OK按钮。



2 状态图

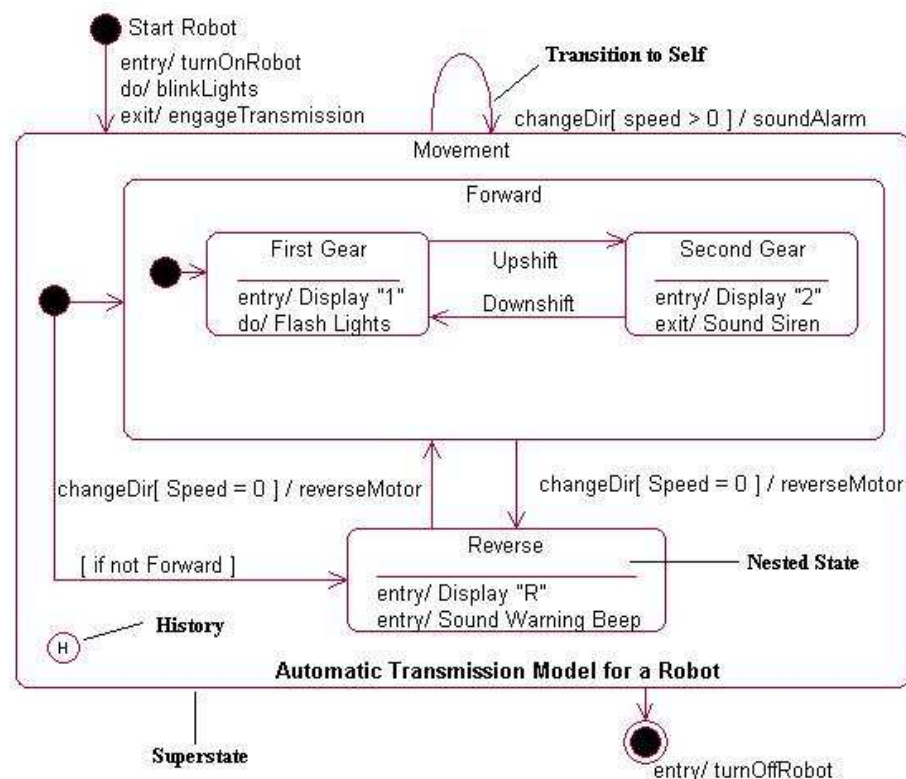
状态图显示一个对象从创建到摧毁的整个生命周期。**Rose**可以对每个类创建一个或者多个状态图，在状态图中体现类的所有状态和各种过渡转换，状态变化是状态图显示的焦点。状态图可以帮助系统分析员、设计人员和开发人员理解系统中各个对象的行为。在**Rose**中，状态图和类图相互配合，以便完整描述类的特征。仅用类图是不够的，因为它只能描述类对象的静态特征，而状态图可以对类对象动态行为进行建模。

2 状态图

■ 2.1 状态图所包含的内容

— 状态图主要显示以下3种内容:

- 对象在生命周期中所经历的状态序列;
- 诱发对象从一个状态转换到另一个状态的事件;
- 状态改变所导致的动作。





2 状态图

■ 2.2 创建状态图

— 步骤如下：

- 在浏览器中，用鼠标右键单击模型元素（除了属性、关系和出现在构件视图中的模型元素）；
- 单击New>Statechart Diagram；
- 或者，
- 单击Browse>State Machine Diagram；
- 单击New；
- 在New State Machine对话框中选中Statechart Diagram复选框；
- 输入状态图标题；
- 单击OK按钮。

2 状态图

■ 2.3 状态图工具箱

- 13种常用的工具:
- 可以定制;

按钮	用 途
	将光标变成箭头，以便选择项目
	在框图中添加文本框
	在框图中添加注释
	连接注释和框图项目
	在框图中添加新状态
	起始状态
	终止状态
	状态转换
	水平同步棒
	垂直同步棒
	转换到自身
	判断
	锁定按钮

3 活动图

■ 3.1 使用活动图

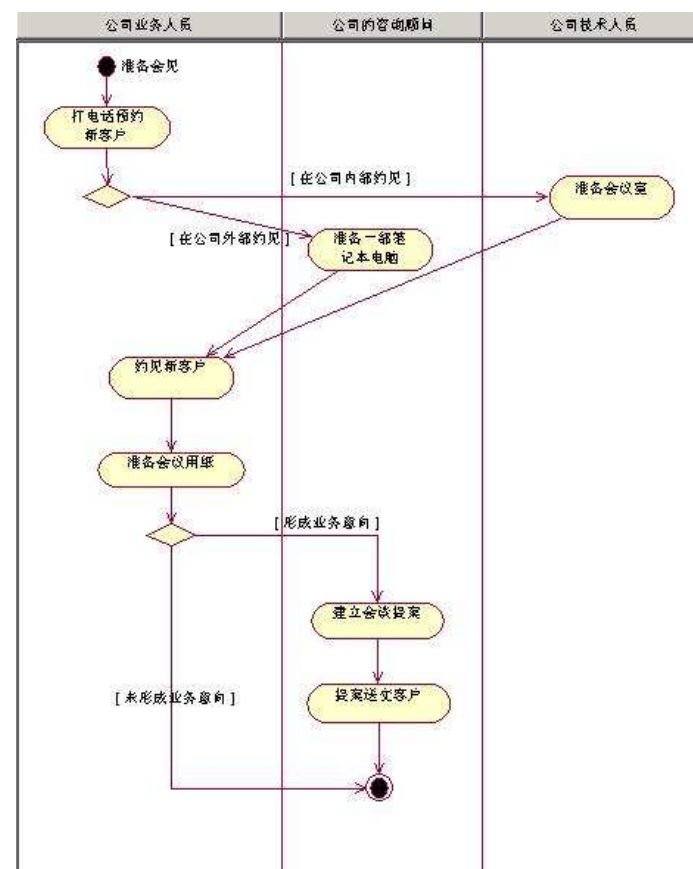
活动图可以对多种不同类型的工作流建模。如果从更简单或者更具体的角度考虑，可以更容易理解活动图。即，活动图被设计用于简化描述一个过程或操作的工作步骤。例如，软件公司可以用活动图对一个软件的开发过程建模；会计师事务所可以用活动图对任意数目的财务往来进行建模；公司可以用活动图对订单批准过程进行建模；还可以对诸如求Fibonacci数列第 n 个数的数值之类的操作进行建模。

3 活动图

■ 3.2 活动图所包含的内容

— 活动图包含的内容:

- 判断 (Decisions)
- 状态 (States)
- 泳道 (Swimlanes)
- 同步 (Synchronizations)
- 对象 (Objects)
- 对象流 (Object flows)
- 转换 (Transitions)
- 活动 (Activities)
- 初始状态 (Start state)
- 终止状态 (End state)





3 活动图

■ 3.3 创建活动图

— 步骤如下：

- 在浏览器中，单击模型元素（除了属性、关系和出现在构件视图中的模型元素）；
- 单击New>Activity Diagram;
- 在浏览器中对新建的活动图重命名，或者双击NewDiagram图标来显示活动图。

3 活动图

■ 3.4 活动图工具箱

- 19种常用工具
- 可以定制

按钮	用途
	将光标变成箭头，以便选择项目
	在框图中添加文本框
	在框图中添加注释
	连接注释和框图项目
	状态
	活动
	起始状态
	终止状态
	状态转换
	转换到自身
	水平同步棒
	垂直同步棒
	判断
	泳道
	对象
	对象流
	业务活动
	商务交易
	锁定按钮



3 活动图

■ 3.5 理解 workflow

– 定义

- 是一个良好定义的动作序列，执行时将产生一个可观察的值，或者产生一个个体或实体的对象。

– 建模目的

- 理解一个组织的结构和动态特性；
- 确保客户、最终用户和开发人员对组织形成一致的理解；
- 导出用于支持组织的系统需求。



3 活动图

■ 3.5 理解工作流

– 识别工作流

- 谁（who）或什么（what）将对工作流负总责？
- 要实现目标，需要执行哪些活动？
- 谁（who）将对执行各种活动和状态负责？
- 活动会创建或者修改对象吗？
- 考察模型中的其他元素，活动和状态应该在何处出现？
- 为什么这个活动或状态需要出现？



3 活动图

■ 3.6 用活动图对 workflow 建模

— 步骤:

- 识别 workflow 的目标;
- 确定从起始状态到终止状态 workflow 的前置条件和后置条件;
- 定义并识别为实现目标而必须发生的活动和状态, 按逻辑顺序将它们放进模型图, 并对它们命名;
- 定义并画出所有要在模型图中创建和修改的对象, 将这些对象和活动同对象流连接起来;
- 按泳道决定谁 (who) 或什么 (what) 将对执行这些活动和状态负责;
- 从主流程开始, 用转换符号连接所有的元素 ;
- 在流程可能要分裂出一个候补流 (alternate flow) 的地方放置一个判断;
- 评估模型图, 看是否有并发 workflow, 如果有, 用同步表示分叉 (forking) 和结合 (joining) ;
- 在每个模型元素的规范窗口中设置动作、触发器和监护条件。

4 状态机共享的模型元素

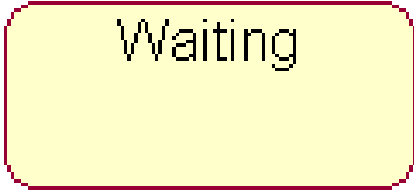
■ 4.1 状态

— 定义

- 状态（**state**）是指在对象的生命周期中满足某些条件、执行某些活动或等待某些事件的条件（**condition**）或状况（**situation**）

— 图形表示

- 圆角矩形表示，状态的名称放在圆角矩形中



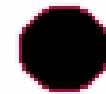
Waiting

4 状态机共享的模型元素

■ 4.2 起始状态和终止状态

— 起始状态

- 又称“初始状态”，在状态图中，起始状态显式地给出一个状态机执行的起始点，表示导致转换（transition）的事件中的起始事件；在活动图中，起始状态显式地给出一个工作流的起始点。
- 一个状态机只能有一个起始状态。

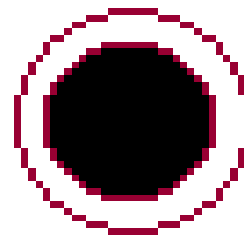


4 状态机共享的模型元素

■ 4.2 起始状态和终止状态

– 终止状态

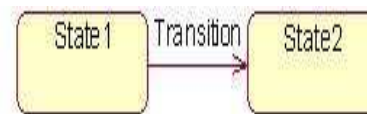
- 表示活动的结束或者一个最终状态，即对象从内存中删除之前所处的状态
- 状态图和活动图都可以有0个、1个或者多个终止状态。



4 状态机共享的模型元素

■ 4.3 状态转换

- 说明一个处于源状态的对象将要执行某种（些）指定的动作，并且当特定的事件发生或者某种条件被满足时，将进入目标状态。
- 状态转换是两个状态之间、两个活动之间或者一个活动和一个状态之间的关系。
- 可以从一个状态发出一个或多个状态转换，前提是转换要唯一。从一个状态发出的转换不能具有相同的事件，除非事件中有多个条件。转换既可以出现在状态图中，也可以出现在活动图中
- 图形表示：

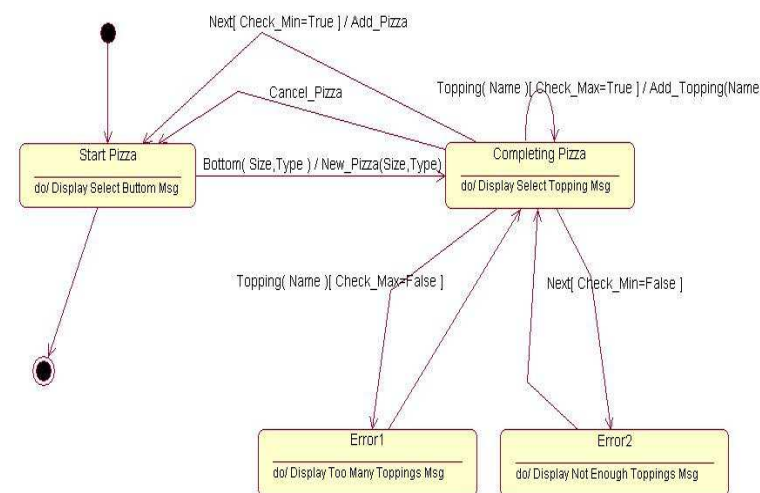


4 状态机共享的模型元素

■ 4.3 状态转换

- 状态转换通常采用如下的语法进行标记：
事件名（参量）[监护条件]/动作^目标.发送事件（参量）
- 转换及转换动作

转换种类	含 义	UML语法
入口动作	进入某一状态时执行的动作	entry/action n
出口动作	离开某一状态时执行的动作	exit/action
外部转换	引起状态改变的转换或自身转换，同时执行一个具体的动作，包括引起入口动作和出口动作被执行的转换	e(a:T)[exp] /action
内部转换	引起一个动作的执行但不改变状态或不引起入口动作和出口动作的执行	e(a:T)[exp] /action



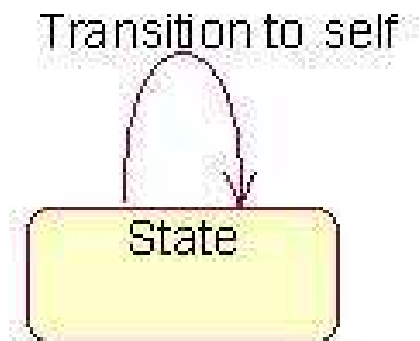
4 状态机共享的模型元素

■ 4.4 转换到自身

— 定义

- 包含的源状态（活动）和目标状态（活动）相同，所包含的动作和事件与转换完全相同。

— 图形表示



4 状态机共享的模型元素

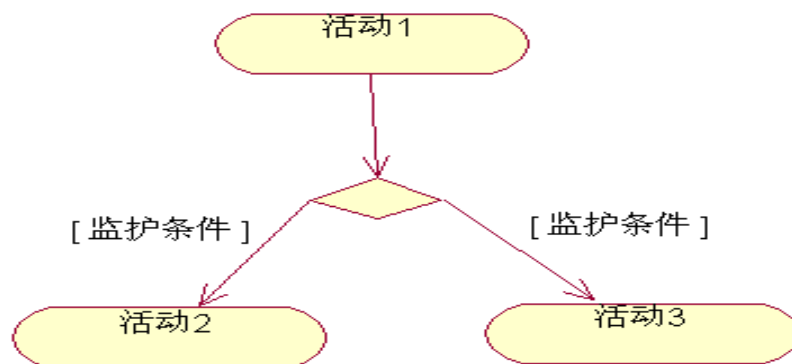
■ 4.5 判断

— 定义

- 表示 workflow 基于监护条件将出现分支的位置

— 图形表示

- 判断在活动图和状态图中表示为一个菱形



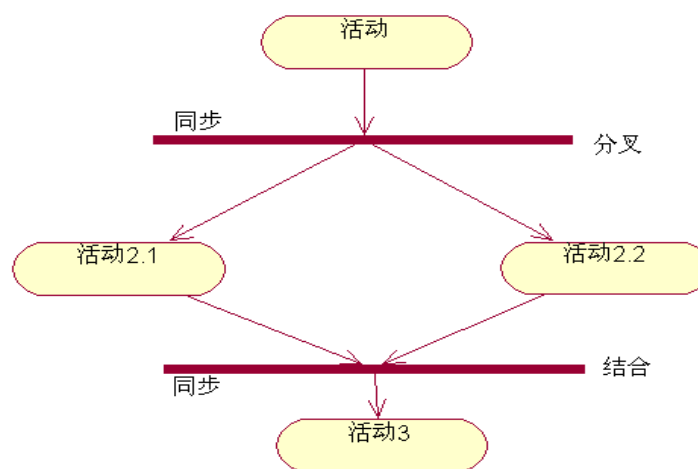
4 状态机共享的模型元素

■ 4.6 同步

– 定义

- 在活动图和状态图中，同步可视化地定义那些表示并行工作流的分叉和结合
- 在浏览器中并不显示同步

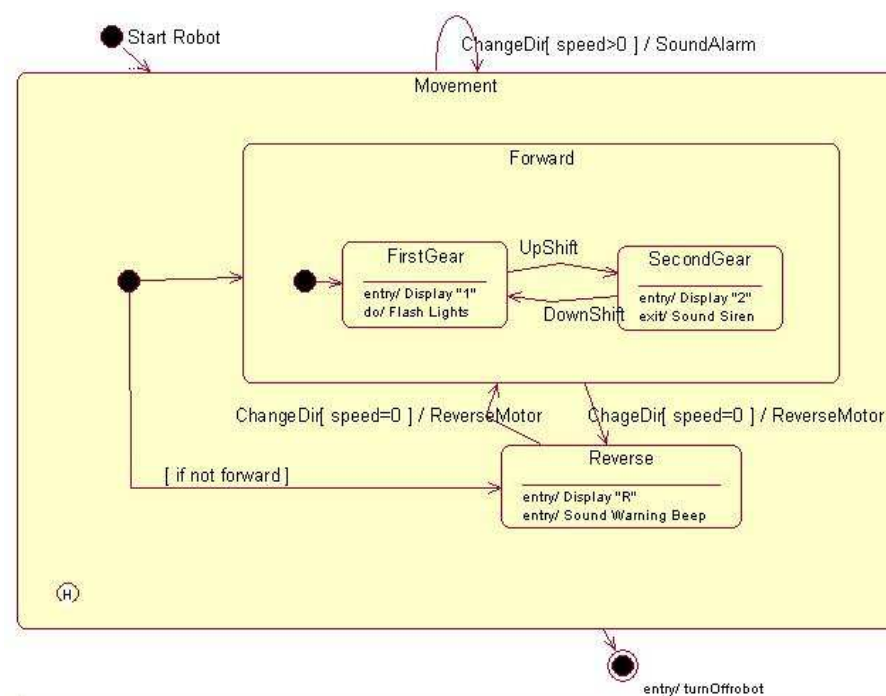
– 图形表示：一条粗的水平或垂直的棒条



4 状态机共享的模型元素

■ 4.7 例子：机器人自动传输的状态图

- 起始状态
- 终止状态
- 超状态和子状态
- 嵌套状态
- 历史状态



5 活动图专有的模型元素

■ 5.1 活动

— 定义

- 活动表示一个工作流中“任务”或“职责”的执行，也可以表示某个过程中一条语句的执行。一个活动类似于状态，但它表达了这样一种含义：在一个活动中没有明显的等待（事件）。

— 图形表示



Press Play
Button

5 活动图专有的模型元素

■ 5.2 泳道


— 定义

- 将活动图中的活动分组，每一组指明了谁（who）或者是什么（what）对执行活动或状态负责

— 泳道的两个重要特点：

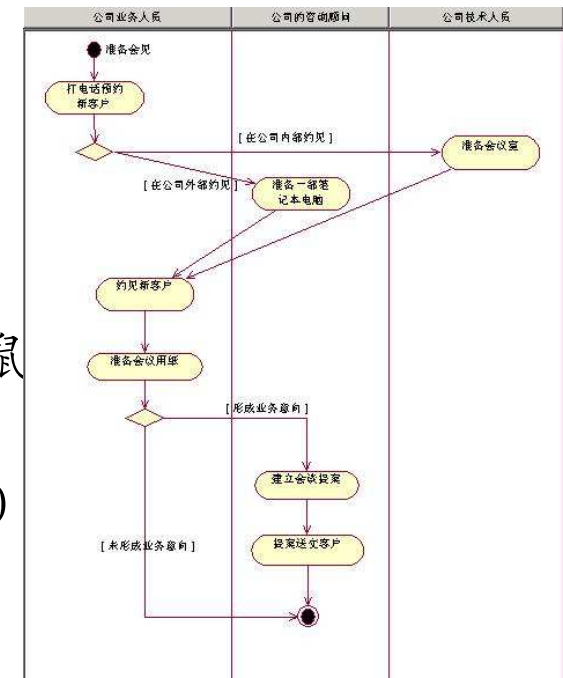
- 指明了活动图中的活动主体
- 转换可以跨越不同的泳道

— 在活动图中创建泳道：

- 点击活动图工具箱中的泳道图标 
- 在活动图中要创建泳道的位置点击鼠标

— 删除泳道

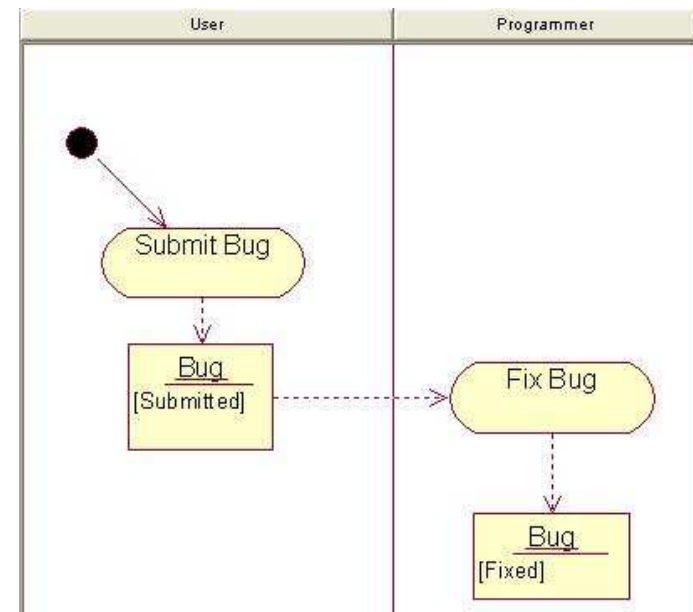
- 点击活动图中的泳道头（泳道名称）
- 按Del键将泳道从活动图中删除；或者按Ctrl+D将泳道从模型中删除



5 活动图专有的模型元素

■ 5.3 对象

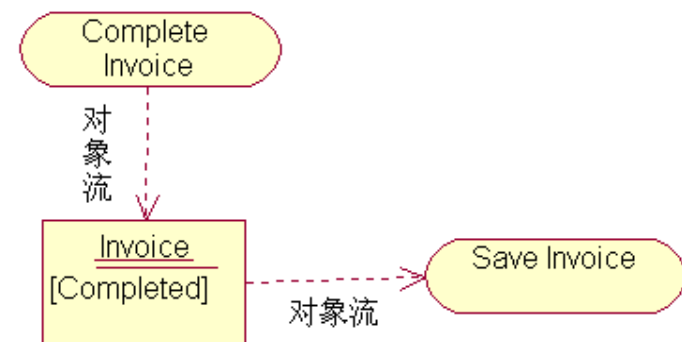
- Rational Rose 2003不支持专门的对象图建模，但是允许对象出现在活动图、协作图和顺序图中
- 活动图中的对象可以表示活动之间的输入输出关系
- 对象可以以多种状态出现



5 活动图专有的模型元素

■ 5.4 对象流

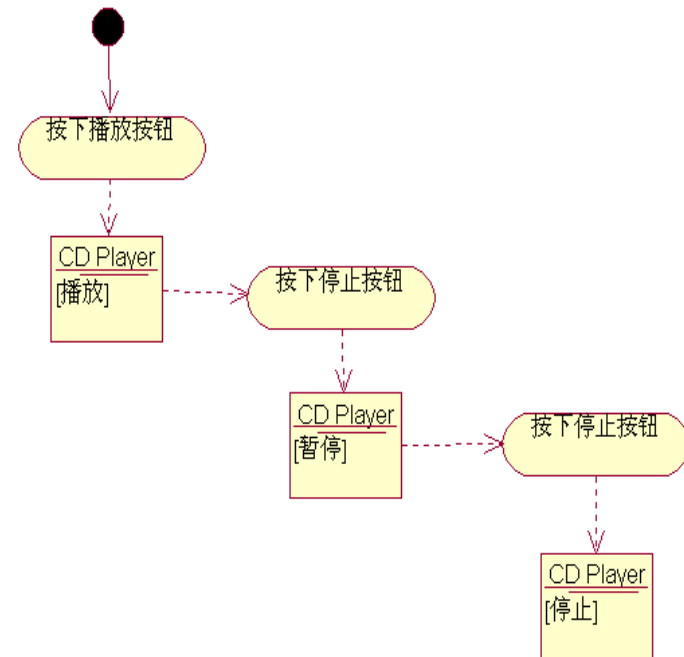
- 在UML规格文件中，对象流是指将对象状态作为输入或输出的控制流。
- 活动图中的对象流表示对象和一个活动的关系，对象可以创建一个对象流（作为输出），也可以使用对象流（作为输入）。
- 图形表示
 - 用虚线箭头来表示对象流



5 活动图专有的模型元素

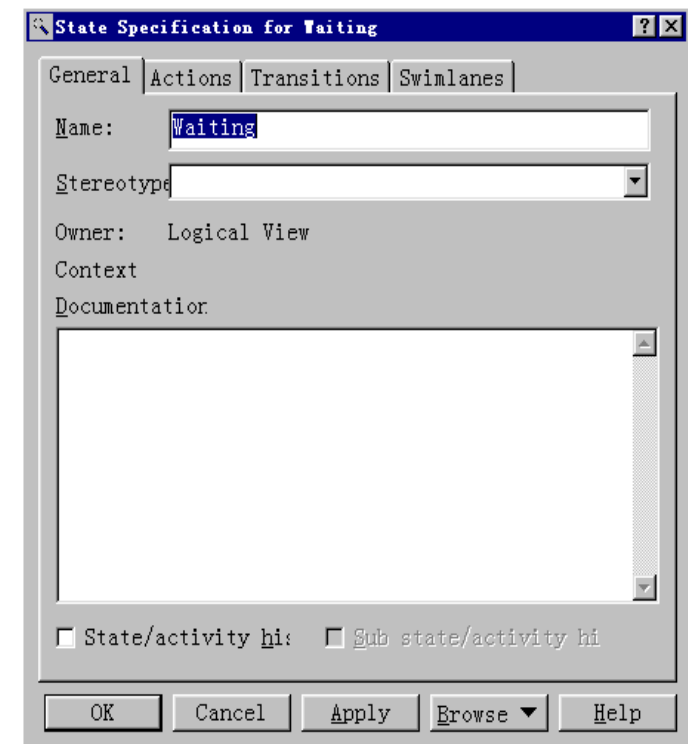
■ 5.5 理解对象和对象流

- 对象可以以不同的状态出现多次;
- 活动可以改变对象的状态;
- 对象通过对象流和活动相连。



6 状态规范和活动规范

- 状态(起始状态和终止状态)和活动的规范窗口都包含下列标签:
 - General标签
 - Actions标签
 - Transitions标签
 - Swimlanes标签
- 1) General标签
 - 复选标记State/activity history: 状态/活动历史, 历史提供了一种通过子状态直接转换到最近访问过的状态的机制。



6 状态规范和活动规范

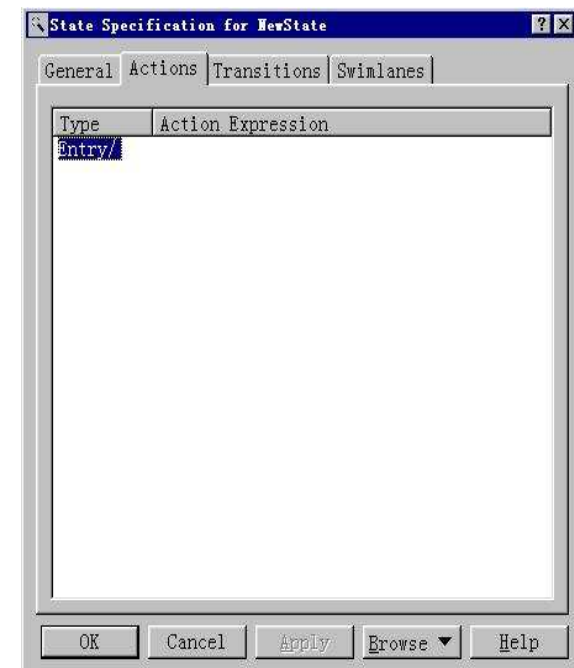
- 2) Actions标签
 - Type: 类型字段标识符栏列出了动作规范中指定的动作种类
 - Action Expression: 列出了4种可能的时间选项，规定了一个动作执行的时机和被执行动作的类型。见动作规范的Detail标签



7 动作规范

■ 7.1 创建一个新的动作

- 在状态图或活动图规范窗口的**Actions**标签中：
 - 单击鼠标右键，显示快捷菜单；
 - 单击**Insert**添加一个条目（**entry**）项；
 - 双击添加的“**Entry/**”项，显示动作规范窗口；
 - 在**Name**栏输入动作描述。如果该栏未被激活，在**Type**栏单击**Action**。



7 动作规范

■ 7.2 状态和活动动作

- 在一个状态或活动中，有以下4中可能的动作：
 - On Entry
 - On Exit
 - Do
 - On Event
- 1) On Event
 - Event: 对一个在时间和空间上占有一定位置的有意义事情的规格说明
 - Arguments: 由所有与事件相关的可选参量组成
 - Condition: 可能包含一个条件布尔表达式

Detail

When: On Event

On Event

Event:

Arguments:

Condition:

Type: Action

Name:

Send arguments:

Send target:

OK Cancel Apply Browse Help

8 转换规范

■ 8.1 General标签

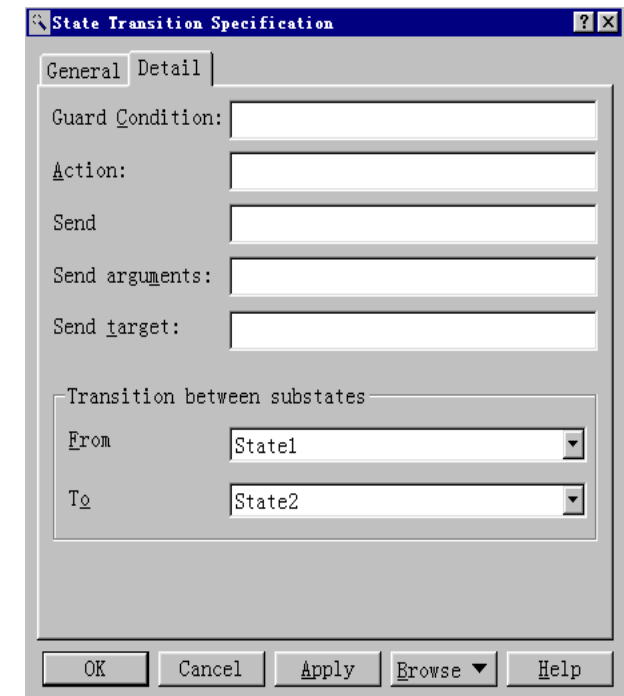
■ 8.2 Detail标签

- 监护条件（Guard Condition）

- 条件状态转换只有当条件表达式为真时才被触发。监护条件显示在转换的附近，表示在方括号内：

事件名（参量）[条件]/动作^目标。
事件（参量）

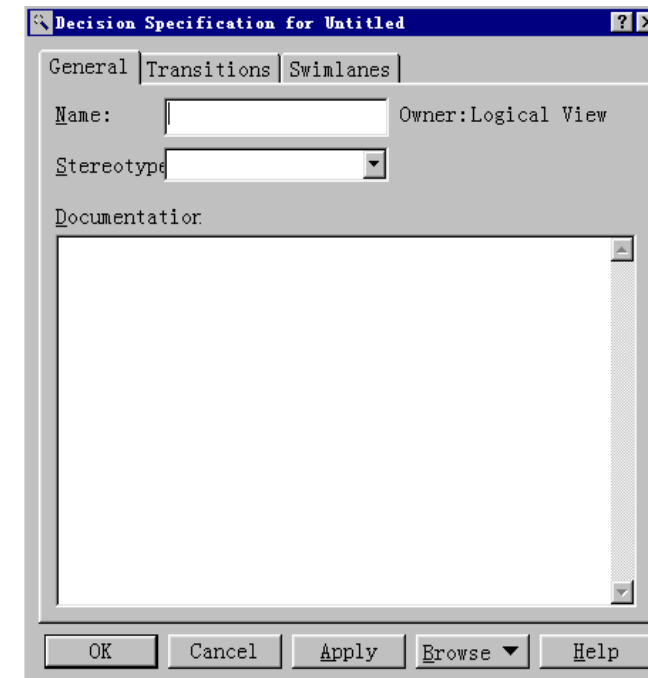
- 子状态之间的转换：当转换被放置在视图中隐藏的子状态，或者转换来自于视图中隐藏的子状态时，子状态之间的转换变得十分有用。From栏显示了转换出发时所在状态的状态名，To栏转换所指的状态名。From和To任何时候都处于激活（active）状态



The image shows a dialog box titled "State Transition Specification". It has two tabs: "General" and "Detail". The "General" tab is selected. The dialog contains several input fields: "Guard Condition:", "Action:", "Send", "Send arguments:", and "Send target:". Below these is a section titled "Transition between substates" with two dropdown menus: "From" (set to "State1") and "To" (set to "State2"). At the bottom, there are buttons for "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

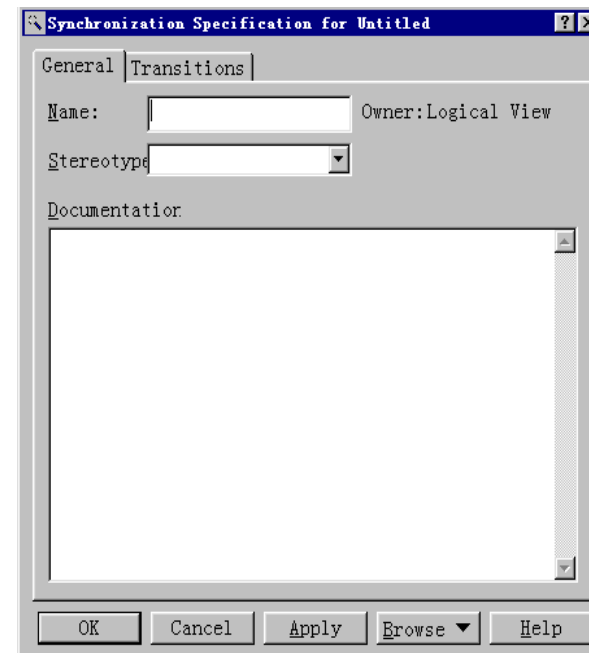
9 判断规范

- 9.1 General标签
- 9.2 Transitions标签
- 9.3 Swimlanes标签



10 同步规范

- 10.1 General标签
- 10.2 Transitions标签



第7章 构件图及其应用



《Rational Rose 2003基础教程》

配套电子教案



内 容

- 基本概念
- 构件图操作
- 构件规范
- 构件包规范



1 基本概念

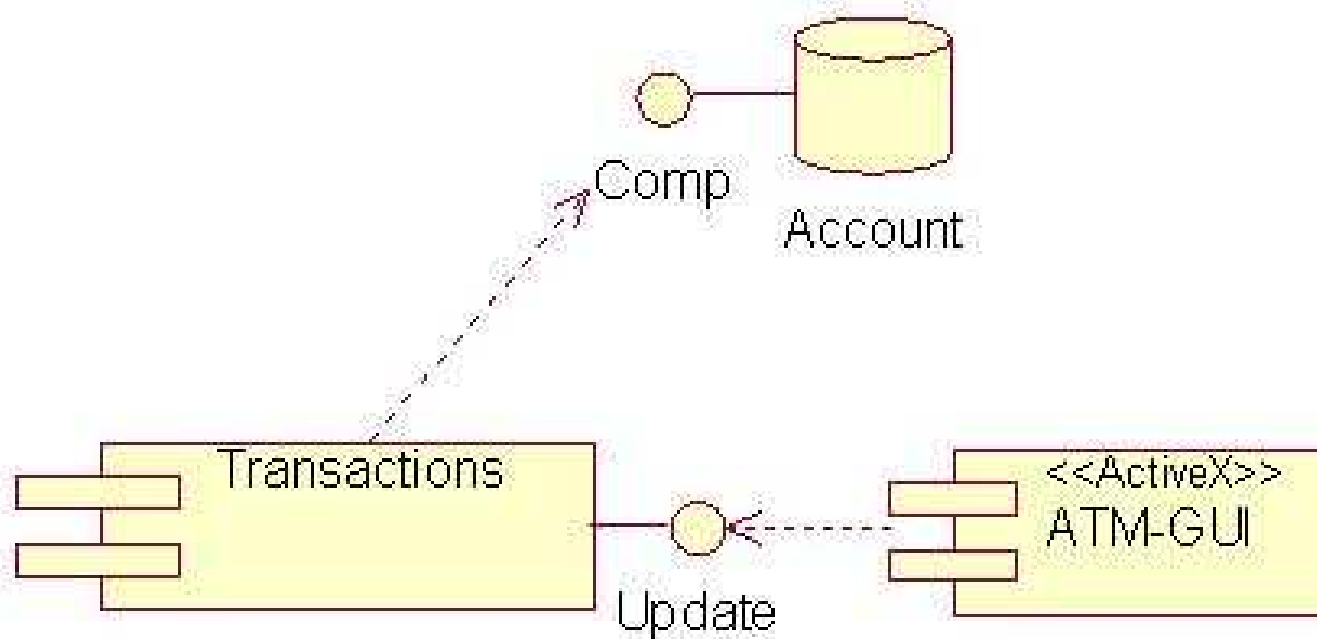
构件图提供当前模型的物理视图，对系统的静态实现视图建模。构件图显示一个系统物理设计时，构件所映射的类和对象的配置。

一个构件图可以表示一个系统全部或者部分的构件体系。从组织内容看，构件图显示软件构件的组织以及构件之间的依赖关系，包括源代码构件、二进制代码构件以及可执行构件。

构件图主要包含以下几种内容：构件、接口、依赖关系以及构件包。

1 基本概念

■ 构件图举例





1 基本概念

■ 1.1 构件和接口

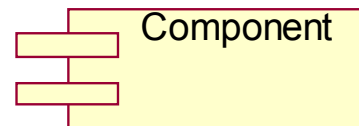
- 构件是系统中遵从一组接口并提供其实现的物理的、可替换的部分，是定义了良好接口的软件模块，如源代码、二进制代码、可执行文件以及动态连接库等。
- 构件的接口由该构件支持的一个或多个接口元素表示。
- 在建模中，构件用于显示编译和运行时的依赖关系，以及接口和软件模块之间调用的依赖关系。
- 一个系统可以包含多个不同类型的软件模块，每个软件模块都由模型中一个构件表示。

1 基本概念

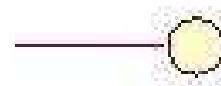
■ 1.1 构件和接口

— 图形表示

- 构件用左边带有两个标签的矩形图标表示



- 接口用于描述构件所提供的服务的一组操作集合，指定了构件的外部可见操作。构件和接口之间的关系叫做实现关系。可以通过接口是访问一个构件。



1 基本概念

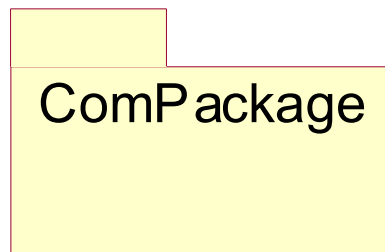
■ 1.2 依赖关系

- 依赖关系表示一个实体使用另一个实体提供的服务。
- 在构件图中，常见的两种依赖关系：
 - 编译依赖
 - 调用依赖
- 使用依赖关系的三种情况：
 - 客户类要访问一个在提供者类（接口）中定义的值（常量或变量）；
 - 客户类的操作激活提供者类（接口）的操作；
 - 客户类的操作具有返回到提供者类（接口）的实例或参量的特征标记。

1 基本概念

■ 1.3 构件包

- 定义：
 - 包含一组逻辑相关的构件或者系统的主要构件，它所扮演的角色和作用类似于类图中的逻辑包。
- 构件包的名称通常就是文件系统的路径名。
- 图形表示





2 构件图操作

■ 2.1 创建和显示构件图

- 可以通过下面三种方式中的某一种来显示和创建构件图：
 - 单击Browse>Component Diagram;
 - 在工具文本框上双击构件图图标;
 - 在浏览器中，在构件图图标上双击。

2 构件图操作

■ 2.2 构件图工具箱

- 18种常见工具
- 可以定制

按钮	用途
	将光标变成箭头，以便选择项目
	在框图中添加文本框
	在框图中添加注释
	连接注释和框图项目
	构件
	包
	依赖
	子程序规范
	子程序实体
	主程序
	包规范
	包实体
	任务规范
	任务实体
	通用子程序
	通用包
	数据库
	锁定按钮



2 构件图操作

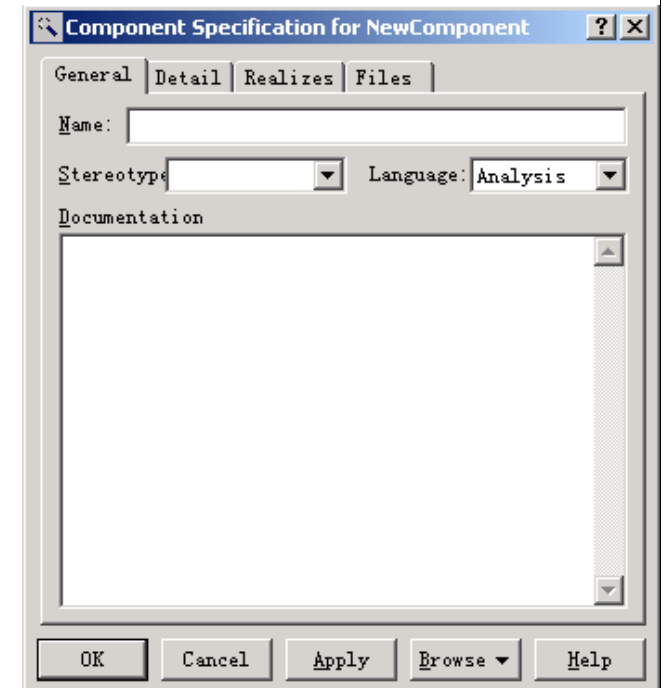
■ 2.3 将构件映射到另一个包

- 每个构件都对应地属于一个包。当用构件图工具箱中的创建工具创建了一个构件之后，新的构件将被映射到包含构件图的包。
- 要将一个包（包A）中的构件重新映射到给另一个包（包B）：
 - 在被包（A）直接包含的图中选中要重新映射的构件图标；
 - 单击Edit>Relocate。

3 构件规范

■ 3.1 General标签

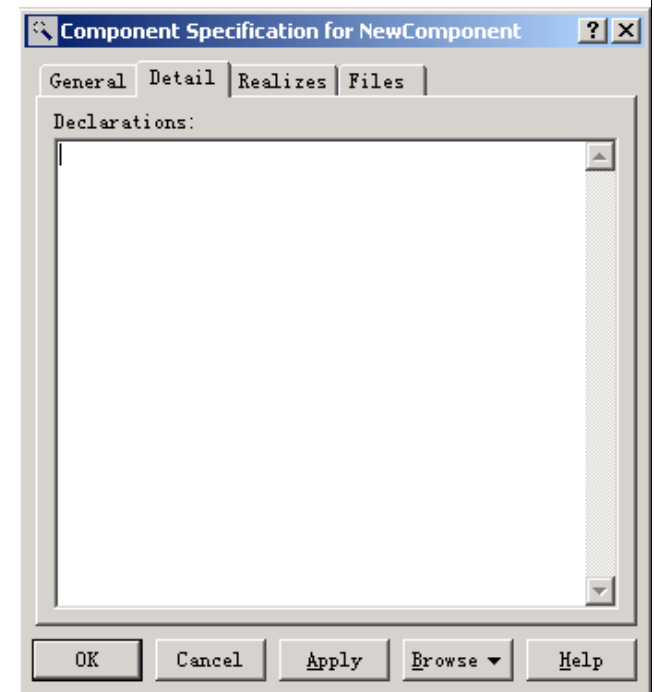
- Name（名称）
- Stereotype（构造型）
- Language（语言）
- Documentation（说明）



3 构件规范

■ 3.2 Detail标签

- Declarations（声明）：声明文本框包含一个声明列表，如类名、变量以及其它一些语言专有特性（`#includes`或类似的结构）。



3 构件规范

■ 3.3 Realizes标签

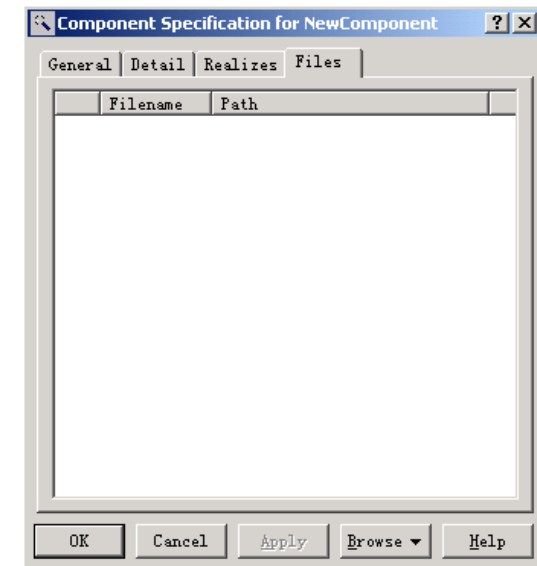
- Show all Classes: 显示所有的类
- Classes Name: 类名
- Logical Package Name: 逻辑包名
- Language: 语言



3 构件规范

■ 3.4 Files标签

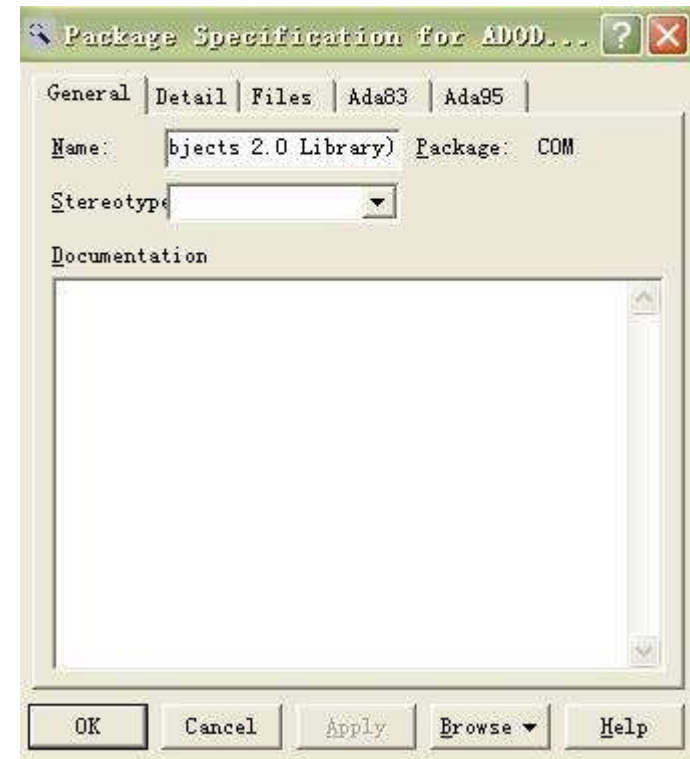
- 通过单击快捷菜单中的“Insert File”，可以插入一个新的与构件相关的文件；或者，单击“Insert URL”，将一个新的URL与构件相连接。



4 构件包规范

■ 4.1 General标签

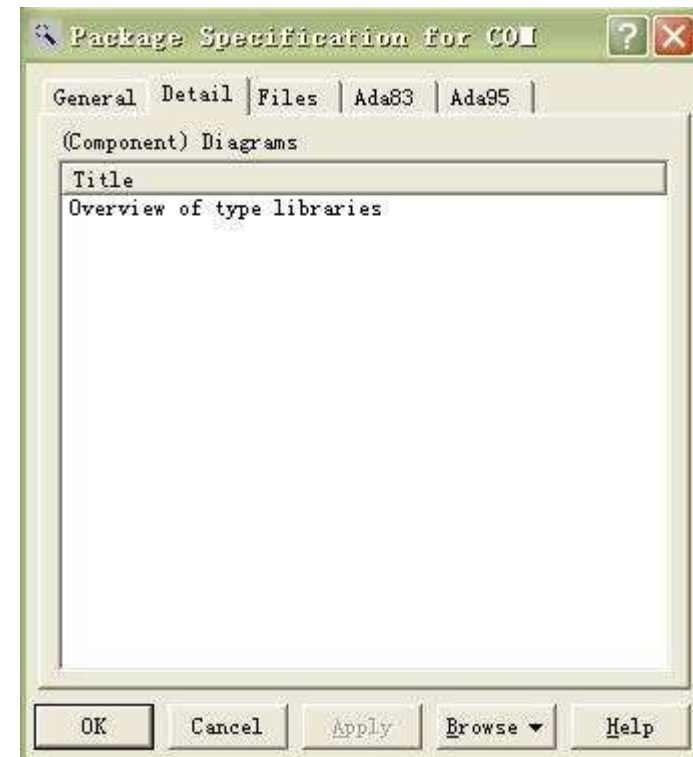
- Name字段:
- Stereotype字段 :



4 构件包规范

■ 4.2 Detail标签

- Detail标签显示构件图文本框，该文本框列出了包中所含的构件图。



4 构件包规范

■ 4.3 Files标签

- Files标签在操纵补充文档的链接时十分有用。补充文档用于说明构件包，其中Filename列显示文件名，Path列显示文件所在的路径。



4 构件包规范

■ 4.4 Ada83标签和Ada95标签

- Set字段：属性设置
- “Edit Set...”按钮：点击该按钮，将弹出一个相应的属性设置窗口



第8章 部署图及其应用



《Rational Rose 2003基础教程》

配套电子教案



内 容

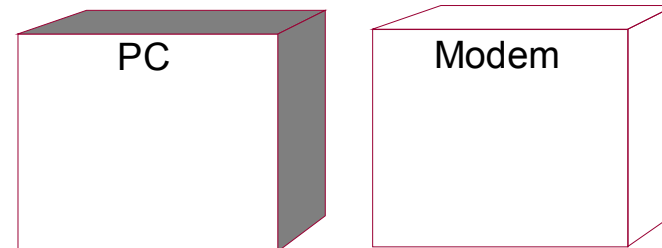
- 部署图概述
- 部署图操作
- 处理器规范
- 设备规范
- 进程规范

1 部署图概述

部署图对面向对象系统的物理方面建模,描述系统运行时节点、构件实例及其对象的配置。

节点是各种计算资源的通用名称,包括处理器和设备两种类型,两者的区别是处理器能够执行程序,而设备是一种不具备计算能力的硬件构件(如打印机)。

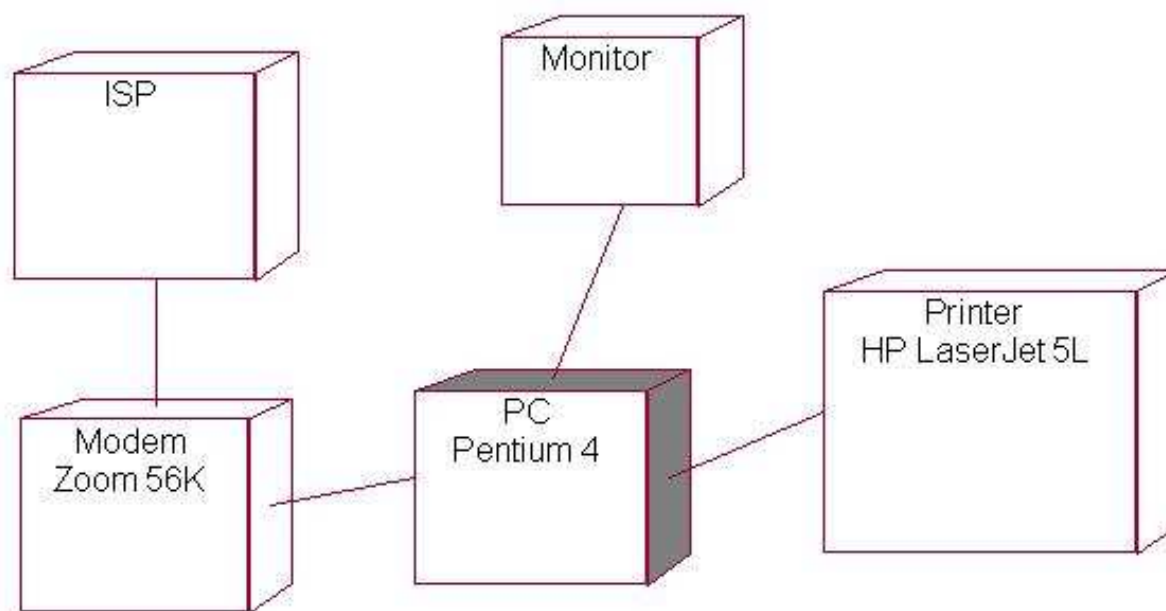
处理器和设备都用箱子图形表示,区别是处理器的侧面有阴影。



1 部署图概述

■ 部署图举例

- 每一个模型都包含一个独立的部署图，显示模型的处理器及其设备之间的连接，以及处理器到处理器的布置。





2 部署图操作








■ 2.1 创建和显示部署图

- 可以通过下面三种方式中的某一种来创建或显示部署图：
 - 单击Browse>Deployment Diagram;
 - 在工具栏上，单击部署图图标;
 - 在浏览器中，双击部署图图标。

2 部署图操作

■ 2.2 部署图工具箱

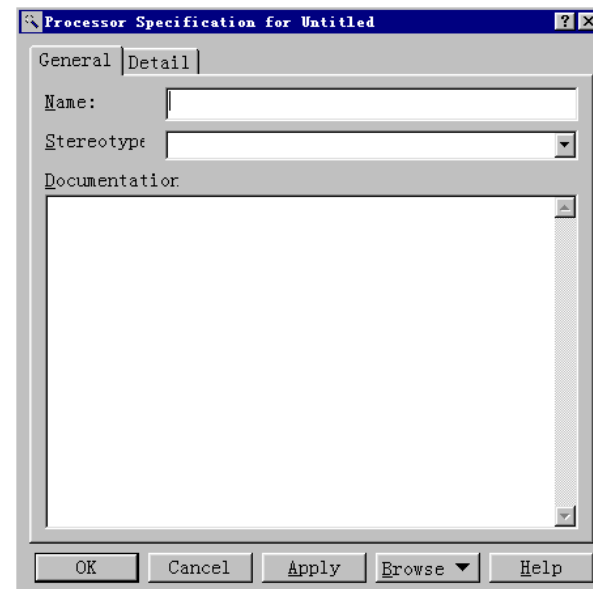
- 8种常用工具
- 可以定制

工具图标	
	选择工具
	文本
	注解
	注解锚定
	处理器
	连接
	设备
	锁定

3 处理器规范

■ 3.1 General标签

- Name: 显示处理器的名称
- Stereotype: 显示处理器的构造型
- Documentation: 显示对处理器的附加说明

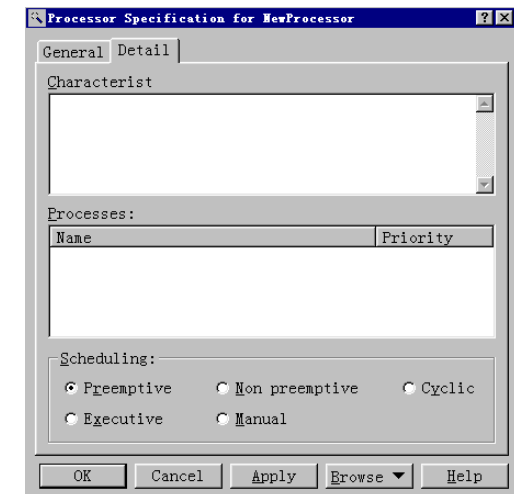


3 处理器规范

■ 3.2 Detail标签

- Characteristics: 指定处理器的物理描述
- Processes: 指派给该处理器的进程
- Scheduling: 处理器所使用的进程调度类型

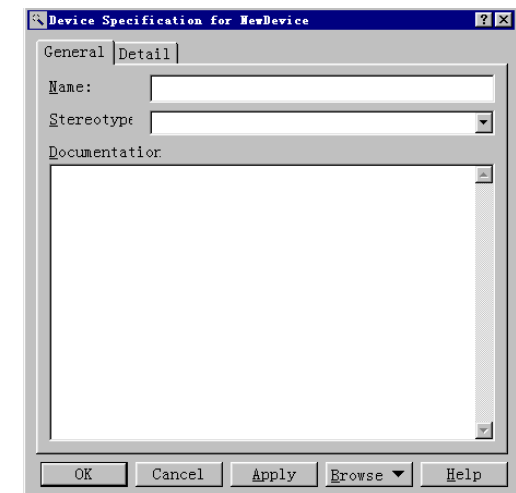
类型	描述
Preemptive	已经就绪的、较高优先权的进程可以抢占当前正在执行的、较低优先权的进程的资源。相同优先权的进程将被赋予一个执行的时间片，允许平均分配计算资源（默认的情况）。
Non preemptive	当前进程将持续执行，直到它放弃控制。
Cyclic	控制从一个进程传递给另一个进程，每个进程有一固定的处理时间长度。
Executive	由一种算法控制进程调度。
Manual	进程由系统外部的一个用户调度。



4 设备规范

■ 4.1 General标签

- Name: 显示设备的名称
- Stereotype: 显示设备的构造型
- Documentation: 显示对设备的附加说明



4 设备规范

■ 4.2 Detail标签

- Characteristics: 指定设备的物理描述



5 进程规范

■ 定义

- 是在一个处理器中执行的控制线程（thread）

■ 进程规范

- 对一个控制线程的文档说明
 - Name: 显示进程的名称
 - Processor: 显示进程的所有者
 - Priority: 进程的相对优先权
 - Documentation: 对进程的附加说明



第9章 Rose的双向工程



《Rational Rose 2003基础教程》

配套电子教案



内 容

- **Rose**对编程语言的支持
- **Rose**的双向工程
- **RTE**举例
- **Rose**的正向工程
- **Rose**的逆向工程



1 Rose对编程语言的支持

支持的语言	企业版	专业版	Rose 2003 Modeler
ANSI C++	X	C++版本	
Visual C++	X	C++版本	
Visual Basic	X	VB版本	
Java	X	J版本	
Ada	X	Ada版本	
CORBA IDL	X		
MIDL	X		
XML DTD	X		



2 Rose的双向工程

Rose支持UML模型与编程语言之间的相互转换，采用双向工程（RTE, Round Trip Engineering）的解决方案。Rose可以分析代码的改动，对模型进行修改，构建出与代码相关的更好的模型。

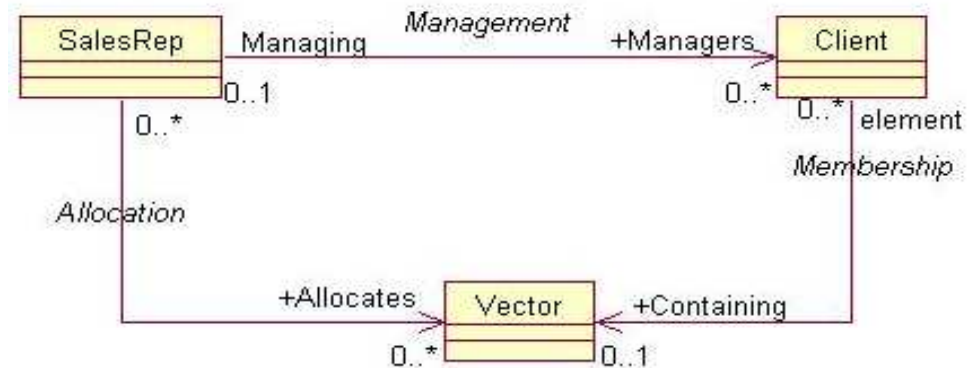
2 Rose的双向工程

语言特性↕	Rose 2003 企业版↕	Rose 2003 专业版↕	Rose 2003 Modeler↕
源代码和模型文件自动同步↕	Java、ANSI C++↕	Java、ANSI C++版本↕	↕
Code templates (代码模板)↕	X↕	VB、VC++语言↕	↕
Framework Wizard (框架向导)↕	X↕	X↕	X↕
ANSI C++代码生成和逆向工程↕	X↕	C++ 版本↕	↕
Visual C++代码生成和逆向工程↕	X↕	C++版本↕	↕
Visual Basic 代码生成和逆向工程↕	X↕	VB 版本↕	↕
COM 构件的逆向工程↕	X↕	VB、C++ 版本↕	↕
<u>Typelib</u> 逆向工程↕	X↕	VB、C++ 版本↕	↕
<u>Ada</u> 代码生成和逆向工程↕	X↕	<u>Ada</u> 版本↕	↕
Java 代码生成和逆向工程↕	X↕	Java 版本↕	↕
J2EE 构件代码生成和逆向工程↕	X↕	Java 版本↕	↕
Gang-Of-Four patterns↕	X↕	Java 版本↕	↕
CORBA/IDL 代码生成和逆向工程↕	X↕	C++、 <u>Ada</u> 版本↕	↕
MIDL 代码生成和逆向工程↕	X↕	↕	↕
XML DTD 代码生成和逆向工程↕	X↕	↕	↕
Web 建模的ASP和JSP页面(双向工程)↕	X↕	↕	↕
从对象模型到数据模型，再到数据库 (或 DDL 脚本)的正向工程↕	X↕	Data Modeler 版本↕	↕
从数据库(或DDL脚本)到数据模型， 再到对象模型的逆向工程↕	X↕	Data Modeler 版本↕	↕

3 RTE举例

■ 3.1 从模型到代码——正向工程

- 类SalesRep: 从模型到代码



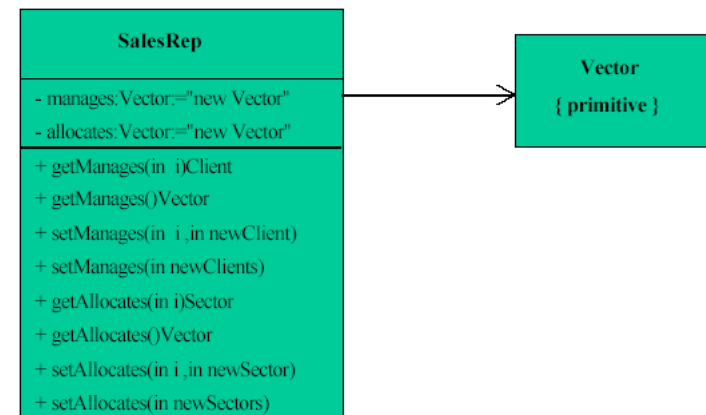
```
import java.util.Vector;
public class SalesRep {
    public Client getManages(int i) { ... } //Management -> manages Client
    public Vector getManages() { ... }
    public void setManages(int i, Client newClient) { ... }
    public void setManages(Vector newClients) { ... }
    public Secteur getAllocates(int i) { ... } // -> allocates Vector
    public Vector getAllocates() { ... }
    public void setAllocates(int i, Vector newVector) { ... }
    public void setAllocates(Vector newVectors) { ... }
    private Vector manages = new Vector();
    private Vector allocates = new Vector();
}
```

3 RTE举例

■ 3.2 从代码到模型——逆向工程

— 类SalesRep: 从代码到模型

```
import java.util.Vector;
public class SalesRep {
    public Client getManages(int i) { ... } //Management -> manages Client
    public Vector getManages() { ... }
    public void setManages(int i, Client newClient) { ... }
    public void setManages(Vector newClients) { ... }
    public Secteur getAllocates(int i) { ... } // -> allocates Vector
    public Vector getAllocates() { ... }
    public void setAllocates(int i, Vector newVector) { ... }
    public void setAllocates(Vector newVectors) { ... }
    private Vector manages = new Vector();
    private Vector allocates = new Vector();
}
```





3 RTE举例

■ 3.3 一致性问题

- 对于任何一种CASE工具，在进行模型/代码转换时，都存在一致性问题。即便最先进的CASE工具也很难保证模型/代码与代码/模型转换之间的一致性（同步）。
 - 在进行逆向工程之后，类SalesRep模型已经丢失了逻辑关系——与类Client和类Vector之间的关联。



4 Rose的正向工程

■ 4.1 代码生成的步骤

– 五个步骤:

- 1) 检查模型;
- 2) 创建构件;
- 3) 将类映射到构件;
- 4) 设置代码生成属性;
- 5) 选择要进行代码转换的类、构件和包。



4 Rose的正向工程

■ 4.1 代码生成的步骤

– 1) 检查模型 (**Check Model**)

- 引用问题
- 非法访问 (Access violations) 问题
- 语言语法问题

– 2) 创建构件

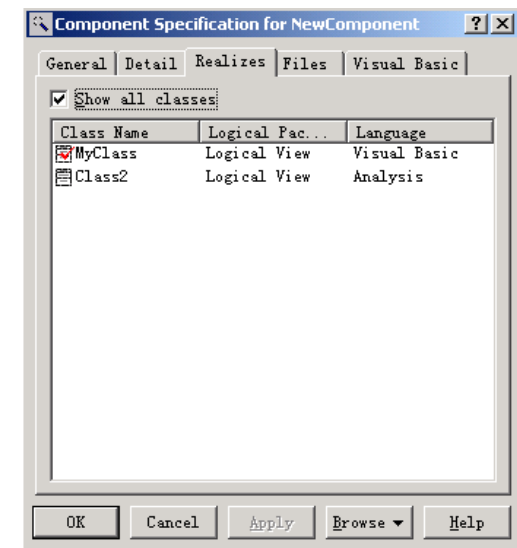
- Check Model是独立于语言的。如果你想将模型转换成下面的三种语言之前进行模型检查，则：
 - Java: 单击Tools>Java/J2EE>Syntax Check
 - CORBA: 单击Tools>CORBA>Syntax Check
 - Oracle8: 单击Tools> Oracle8>Syntax Checker

4 Rose的正向工程

■ 4.1 代码生成的步骤

– 3) 将类映射到构件

- 构件之间的依赖性决定了系统的编译依赖性。
- 指定构件实现的类：
 - 选中构件图或浏览器中的用于实现类的构件图标；
 - 打开构件规范窗口；
 - 选中Realizes标签；
 - 选中Show all classes复选框；
 - 在类列表中找到并右击所要实现的类，单击快捷菜单中的Assign。





4 Rose的正向工程

■ 4.1 代码生成的步骤

– 4) 设置代码生成属性

- 直接影响生成的语言代码框架，代码生成属性控制模型元素转换到代码时的具体细节。在生成代码之前，最好先检查代码生成属性并进行必要的修改。因为模型元素（包括类、属性、构件等）有多种代码生成属性可供选择，而Rose只是提供了常用的代码生成属性的默认设置。
- 要查看代码生成属性，单击**Tools>Options**，然后选择相应的语言标签

4 Rose的正向工程

■ 4.1 代码生成的步骤

– 5) 选择类、构件和包

- Rose允许分步骤、有选择的进行代码转换，即在生成代码时，可以一次选择一个类、一个构件或者一个包，通过代码转换将其转换成相应的语言代码；也可以一次生成多个类、构件或者包的代码；或者是它们的组合。

选中一个或多个	生成的代码
类	每个被选中的类
逻辑包	每个逻辑包中的类
构件	映射到每个构件的类
构件包	映射到构件包中各个包中的类



4 Rose的正向工程

■ 4.2 代码生成

- 1) 语言工具定制
 - 从Tools菜单中选择特定的语言工具选项并设置相应的语言选项
- 2) 控制源代码的框架内容
 - 任何造型工具都不能创建完整的应用程序。
 - Rose强大的代码生成能力在于它能生成大量的框架代码。
- 3) 设定源代码文件的位置
 - 默认的情况下，包含已生成的代码的文件和路径位于当前工作空间所在的目录下。
- 4) 代码生成之后的工作
 - 精化模型系统的程序实现：编写每个类的操作（方法实现）和设计图形用户界面（GUI）



5 Rose的逆向工程

■ 定义

- 通过源代码或者对源代码进行修改，得出UML模型，这就是逆向工程。

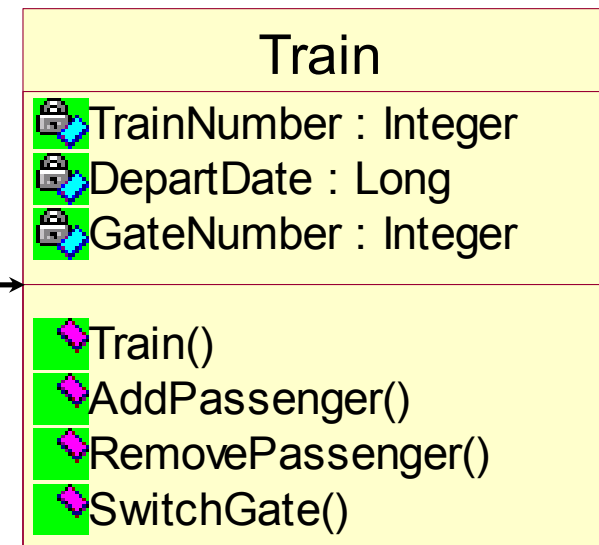
■ 模型的信息来源

- 1) 类、属性和操作
- 2) 关系
- 3) 包和构件

5 Rose的逆向工程

■ 1) 类、属性和操作

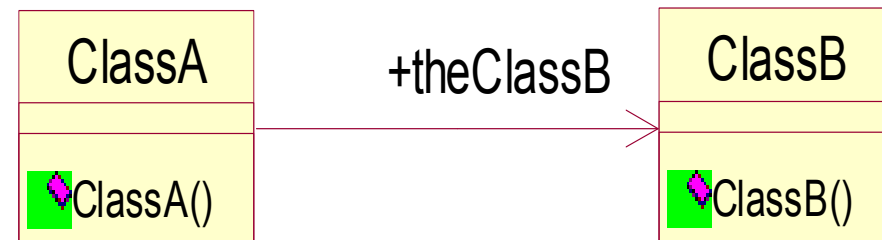
```
public class Train
{
    private int TrainNumber;
    private long DepartDate;
    private int GateNumber;
    public Train()
    {
    }
    public boolean AddPassenger()
    {
    }
    public boolean RemovePassenger()
    {
    }
    public boolean SwitchGate()
    {
    }
}
```



5 Rose的逆向工程

■ 2) 关系

```
//ClassA+  
public class ClassA+  
{+  
    public ClassB theClassB;+  
    public ClassA()+  
    {+  
    }+  
    +  
}+  
+  
//ClassB+  
public class ClassB+  
{+  
    public ClassB()+  
    {+  
    }+  
    +  
}+  
+
```





5 Rose的逆向工程

■ 3) 包和构件

- 代码中包和构件的信息也会在逆向工程时体现在Rose模型中。
- Rose对构件的处理方法因语言而异。

第10章 ANSI C++的 Rose双向工程



《Rational Rose 2003基础教程》

配套电子教案



内 容

- **ANSI C++语言插件**
- **ANSI C++的正向工程**
- 代码生成属性
- 设置代码生成属性
- 生成**ANSI C++**代码
- **ANSI C++的逆向工程**



1 ANSI C++语言插件

ANSI C++是Rose最新提供的支持C++编程语言的插件。该插件提供了如下功能

- 支持从分析到设计的模型开发
- 支持独立于开发商编译器的C++语言
- 支持C++代码生成
- 支持从C++代码到模型的逆向工程
- 支持模型和代码之间的迭代式同步双向工程
- 支持所有C++结构（包括类、模板、名字空间、继承以及类成员函数）的设计、建模和可视化
- 支持大型框架结构
- 支持用户对生成的代码格式化的风格设计



2 ANSI C++的正向工程

■ 正向工程的步骤

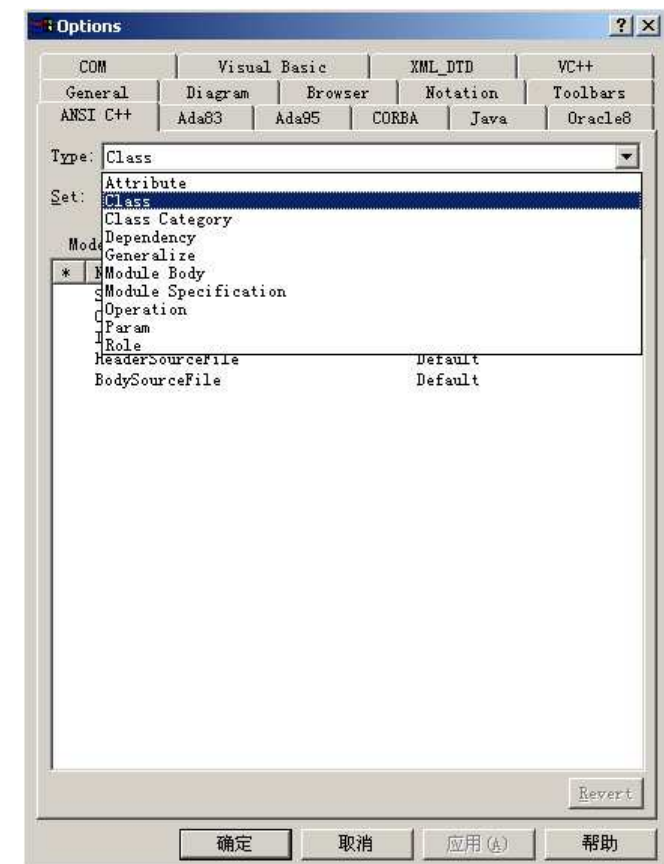
- 1).检查模型;
- 2).生成构件;
- 3).将类映射到构件并将构件的实现语言设为ANSI C++ ;
- 4).设置代码生成属性 ;
- 5).选择类图和 (或)构件图中要进行代码生成的类和构件;
- 6).选择Tools> ANSI C++>Code Generation, 或者, 右单击类和构件图标, 选择ANSI C++>Generate Code....

步骤**1)-4)** 可选

3 代码生成属性

■ ANSI C++语言属性标签

- Attribute (类属性)
- Class (类)
- Class Category (类类别, 包)
- Dependency (依赖关系)
- Generalize (泛化关系)
- Module Body (模块体)
- Module Specification (模块规范)
- Operation (类操作)
- Param (操作参数)
- Role (角色)

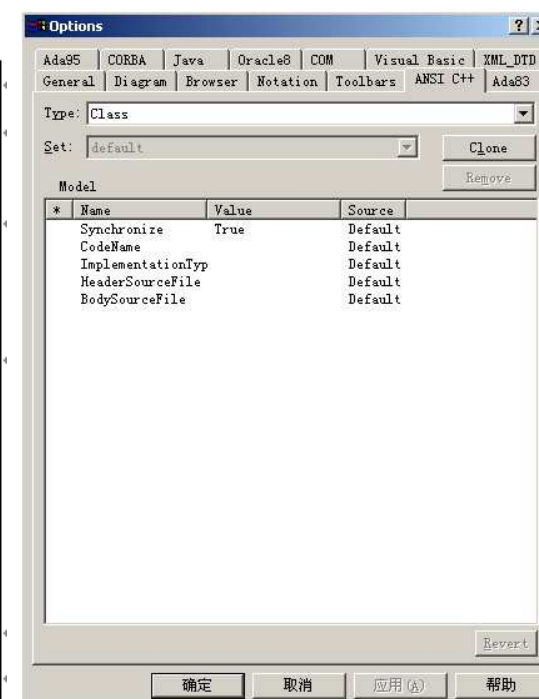


4 设置代码生成属性

■ 4.1 类的属性

- 类的**ANSI C++**代码生成属性
 - 统一设置模型中类的属性、用途和默认值。

属性	类型	描述	默认值
Synchronize	Boolean	如果为 True，该模型元素将参与代码生成和逆向工程；如果为 False，则不参与。	True
CodeName	String	指定类在代码中的名称。修改该属性可以在代码中得到与模型中不相同的类名。如果 Rose 模型和代码用不同的语言描述，该属性显得非常有用。该属性值应该是一个有效的 C++ 标识符。	空
ImplementationType	String	只当类具有构造型 “typedef” 时才使用，此时该属性不为空。该属性值对所有其它的类都可以忽略，而且必须为一个有效的 C++ 类型的表达式。指定返回类型定义声明，例如，如果类 StringArray 具有构造型 typedef，并且该属性值为： std::vector<std::string>，则生成的代码应该为 std::vector<std::string> StringArray;	空
HeaderSourceFile	String	类头文件的文件名（即 .h 文件的文件名）	空
BodySourceFile	String	类实现文件的文件名（即 .cpp 文件的文件名）	空

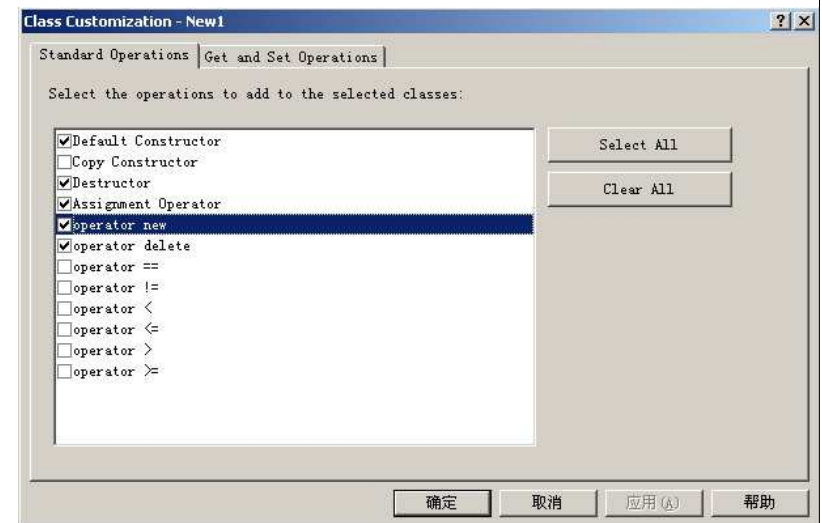


4 设置代码生成属性

■ 4.1 类的属性

– 定制类的代码生成属性

- 在类定制窗口中设置：
 - 1).在Class框图中选择一个或几个类;
 - 2).右单击类选择ANSI C++>Class Customization。
- 设置代码生成属性
 - Standard Operations标签
 - Get and set Operations

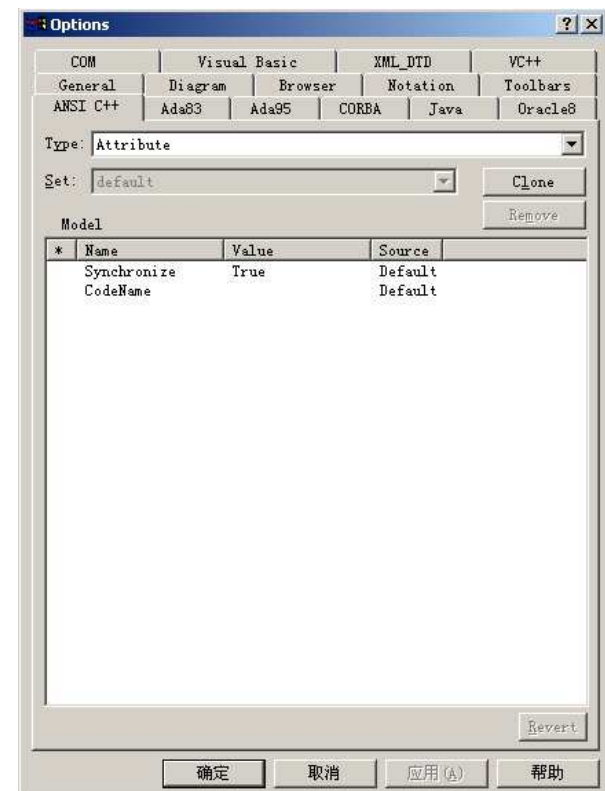


4 设置代码生成属性

■ 4.2 属性的属性

– 在Options窗口对模型中所有类的属性的代码生成属性进行设置

- Synchronize
 - 控制属性是否参与双向工程过程，默认值为True
- CodeName: 控制生成代码中类的属性名，在该窗口中不可修改，即使用模型中的属性名



4 设置代码生成属性

■ 4.3 操作的属性

属 性	描 述	默 认 值
Synchronize	控制操作是否参与双向工程	True
CodeName	生成代码中的操作名	空
InitialCodeBody	控制操作要包括的代码。这些代码在首次运行代码生成过程时在操作中生成，并且在后续代码生成过程中不会被替换。	空
Inline	控制是否内联操作	False
GenerateFunctionBody	控制是否生成函数体。默认生成函数体	Default



4 设置代码生成属性

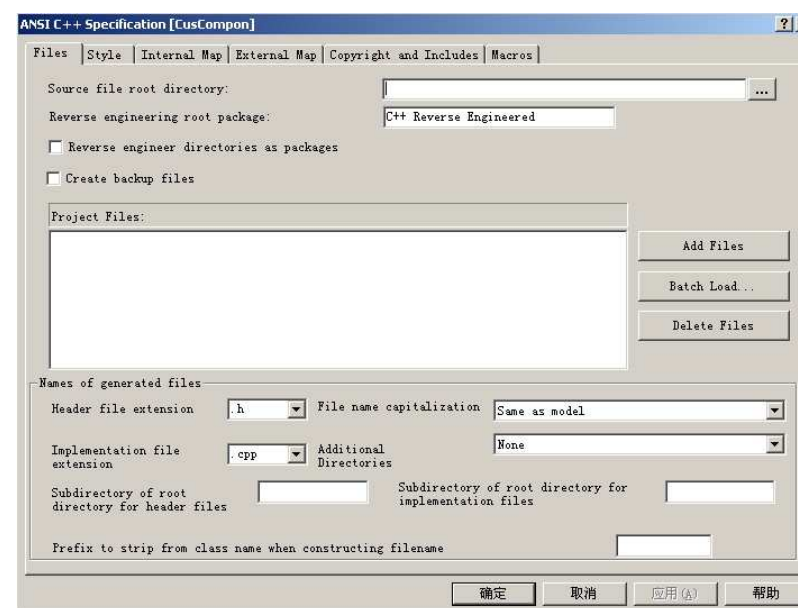
■ 4.4 参数属性

- 操作参数的唯一代码生成属性为 **CodeName**。该属性控制操作参数在代码中的名字，默认使用模型中的参数名。

4 设置代码生成属性

■ 4.5 模块体属性和模块规范属性

- 是与双向工程中的.cpp与.h文件相关的属性
- 构件的代码生成属性
 - 构件ANSI C++规范窗口

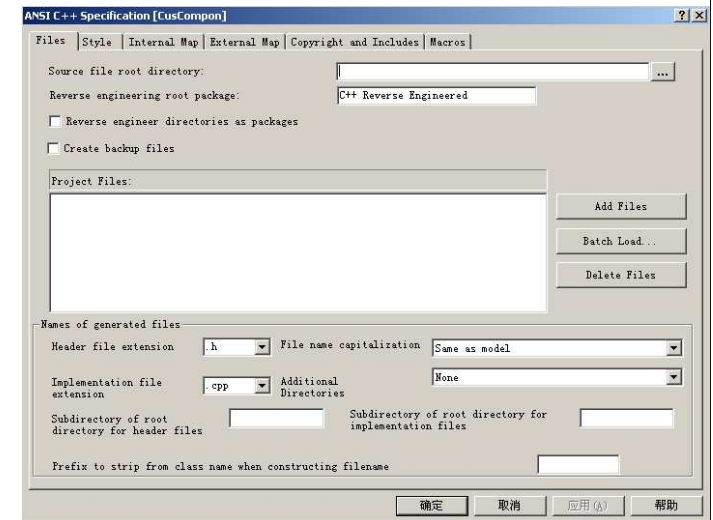


4 设置代码生成属性

■ 构件ANSI C++规范窗口

– 1) Files标签

- Source file root directory: 逆向工程中源文件的根目录
- Reverse engineering root package: 逆向工程的根目录，默认值为C++ Reverse Engineered
- Reverse engineering directories as packages: 将目录在逆向工程中转化为逻辑视图包
- Create backup files: 创建备份文件
- Project Files: 可以在该字段中添加和删除映射到该构件的文件，包括与这个构件相关的.cpp、.h以及其他源代码文件
- Names of generated files: 关于生成的代码文件的名称设置

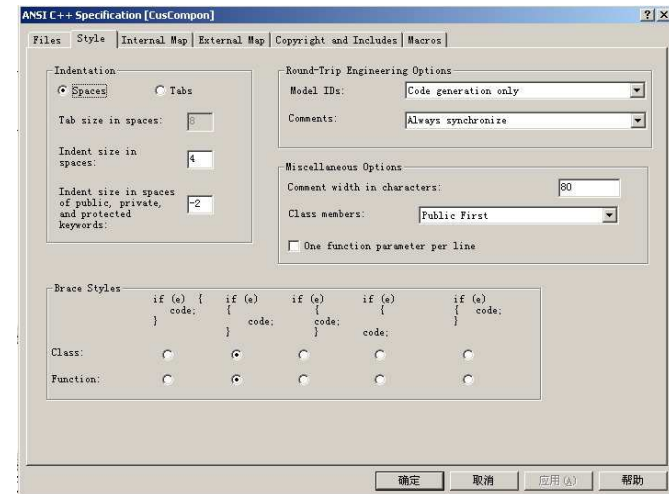


4 设置代码生成属性

■ 构件ANSI C++规范窗口

– 2) Style标签

- Indentation组合框：源代码的缩排格式
- Round-Trip Engineering Options：双向工程选项
- Miscellaneous Options：杂选项
- Brace Styles：代码中类和函数所使用的大括号的格式

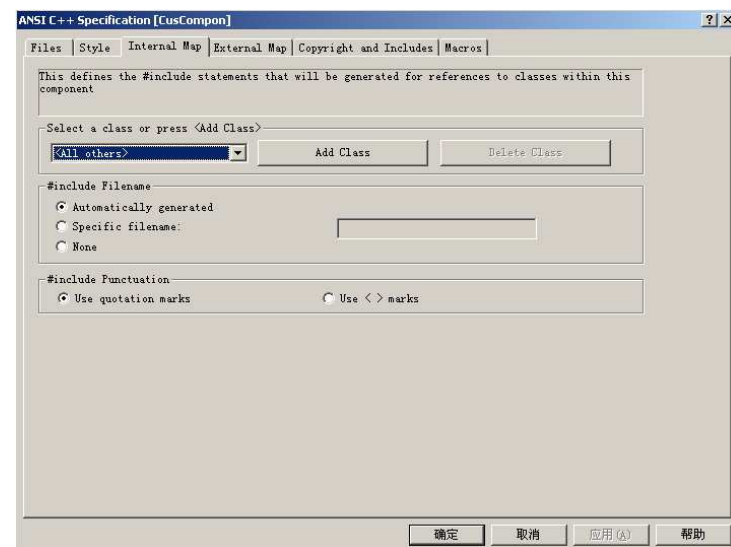


4 设置代码生成属性

■ 构件ANSI C++规范窗口

– 3) Internal Map和External Map标签

- Select a class or press <Add Class>: 添加引用的类
- #include Filename: 选择#include语句中的文件名
- #include Punctuation: 选择在#include语句中使用双引号或者使用“<>”标记

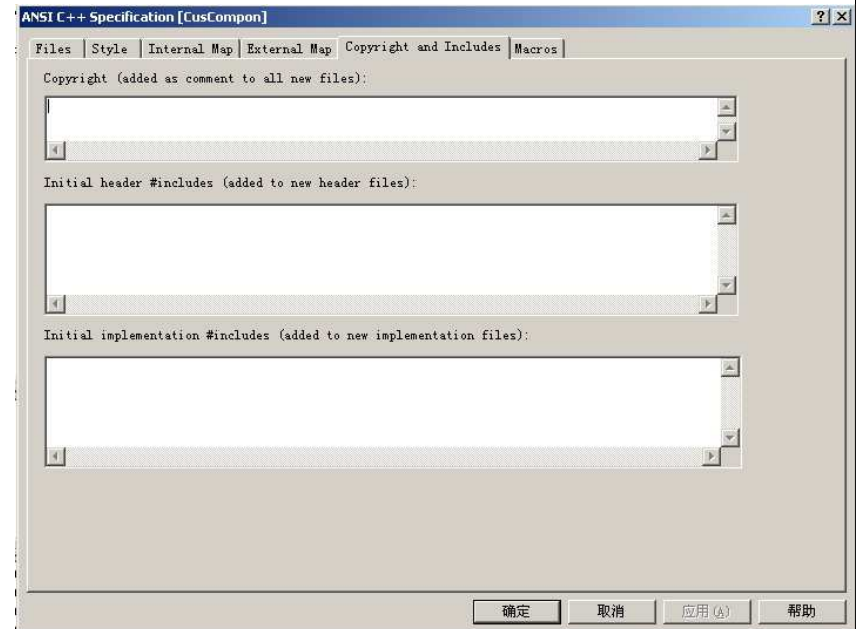


4 设置代码生成属性

■ 构件ANSI C++规范窗口

– Copyright and Includes标签

- Copyright: 构件版权信息，以注释的形式添加到新的文件中
- Initial header #includes: 构件所要包含的所有头文件
- Initial implementation #includes: 构件所要包含的实现头文件

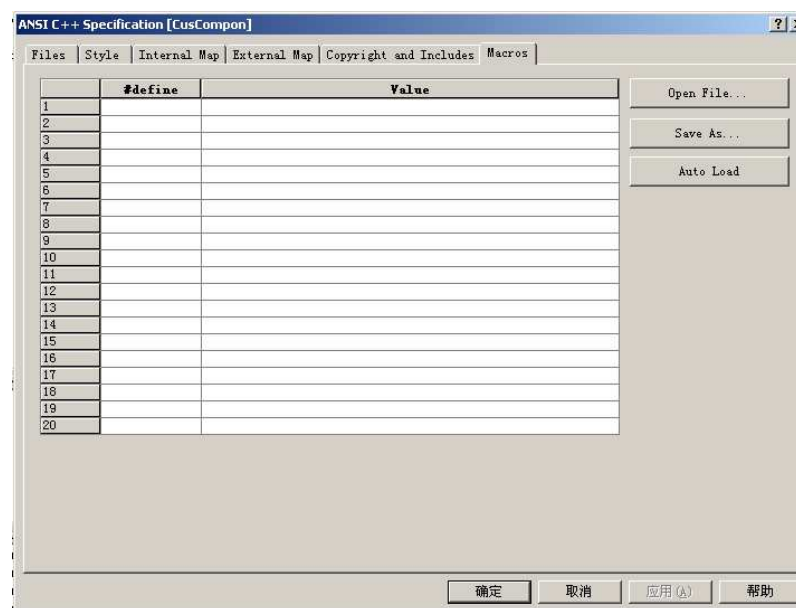


4 设置代码生成属性

■ 构件ANSI C++规范窗口

— 5) Macros标签

- 在该标签中为构件设置任意数目的预定义宏，预定义宏在逆向工程中会被添加到模型中





4 设置代码生成属性

■ 4.6 角色属性

– 3个属性:

- Synchronize: 控制角色是否参与双向工程, 默认值为True
- CodeName: 代码中角色的名称, 默认为空
- InitialValue: 代码中数据成员的初始值, 默认为空

■ 4.7 泛化属性

- Synchronize: 控制泛化关系是否参与双向工程, 默认值为True



4 设置代码生成属性

■ 4.8 依赖属性

- BodyReferenceOnly: 控制#include语句是否只能由与客户类关联的模块体生成，默认值为False

■ 4.9 类类别属性

- CodeName: 设置名称空间名，默认为空
- IsNameSpace: 指定类类别是否为名称空间，默认为False



5 生成ANSI C++代码

- 代码生成的操作步骤:
 - 1).选择类图和 (或)构件图中要进行代码生成的类和构件
 - 2).选择Tools> ANSI C++>Code Generation
- 对模型中的类进行ANSI C++代码生成操作，将变成相应的C++类。每个类生成两个文件，一个.h头文件和一个.cpp实现文件。
类代码中的信息包括:
 - 类名
 - 类可见性
 - 构造函数和析构函数 (由代码生成属性决定)
 - 类的属性 (可见性、Get和Set操作)
 - 类的操作 (参数)
 - 类之间的关系 (角色、依赖、泛化)



6 ANSI C++的逆向工程

- ANSI C++不仅支持一个空模型的逆向工程，也支持向一个已存在的模型中添加逆向工程的模型元素
- 要进行逆向工程，模型必须包含用ANSI C++实现的构件。对所要添加类到模型中的文件，其内容必须是有效的C++代码。



6 ANSI C++的逆向工程

– 逆向工程的步骤:

- 1) 在构件视图中创建并命名新的构件;
- 2). 双击新的构件打开构件规范窗口, 将构件的实现语言改为ANSI C++;
- 3). 右单击新构件, 选择快捷菜单中的ANSI; C++>Open ANSI C++ Specification..., 打开构件ANSI C++规范窗口;
- 4). 在构件规范窗口的Files标签中添加源文件;
- 5). 关闭构件ANSI C++规范窗口;
- 6). 单击构件, 选择Tools>ANSI C++>Reverse engineer..., 打开“选择要添加到模型中的类”的窗口;
- 7). 选择要添加到模型中的类, 单击Ok按钮, 开始逆向工程。

第11章 VC++的 Rose双向工程



《Rational Rose 2003基础教程》

配套电子教案



内 容

- **Rose VC++**的正向工程
- 代码生成属性
- **Code Update Tool**与代码生成
- 生成的代码内容
- **VC++**的逆向工程



1 Rose VC++的正向工程

■ 步骤

- 1.检查模型;
- 2.创建构件, 在构件规范窗口的Language下拉列表框中选择VC++, 并将构件映射到对应的VC项目;
- 3.将类映射到构件;
- 4.设置代码生成属性;
- 5.右单击要生成代码的构件, 选择Update Code..., 或者单击Tools>VC++>Update Code..., 激活Code Update Tool;
- 6. 按照Code Update Tool提示操作。



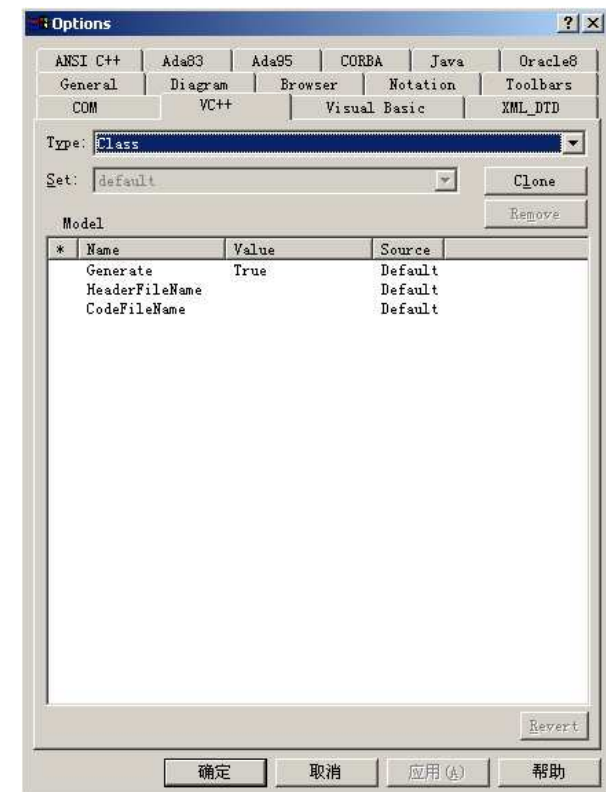
2 代码生成属性

- 设置代码生成属性的途径
 - VC++语言属性窗口：设置VC++语言属性
 - VC++属性对话框（Properties Dialog）：为新建的模型或当前模型设置默认的VC++语言属性
 - 构件属性对话框：设定应用于构件所要实现的类的模型属性
 - Model Assistant工具：精确设置模型中的类与代码之间的对应关系
 - 模型元素的规范窗口

2 代码生成属性

■ 2.1 VC++语言属性窗口

- 类的属性
- 类属性的属性
- 类操作的属性
- 依赖关系 (**Dependency**)
- 角色 (**Role**)
- 包的属性 (**Class Category**)
- 构件的属性 (**Module Specification**)
- 项目属性 (**Project**)

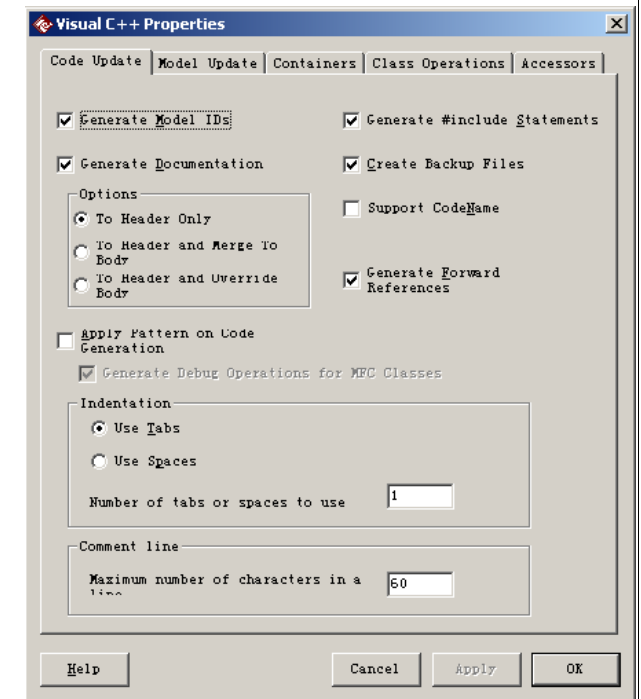


2 代码生成属性

2.2 VC++属性对话框

– Code Update标签

- Generate Model ID: 生成模型ID
- Generate Documentation: 信息作为注释添加到代码中
- Generate #include Statements: 为头文件生成#include语句
- Apply Pattern on Code Generation: 生成代码时是否选中Class Operations标签和Accessors标签中的原型
- Generate Debug Operations for MFC Classes: 为MFC类CObject的派生类生成DUMP和AssertValid成员函数
- Create Backup Files: 如果选中该选项, 则在对源代码文件进行修改之前, 在备份区域生成该文件的备份
- Support CodeName: 为每个模型元素指定与模型中不相同的名字

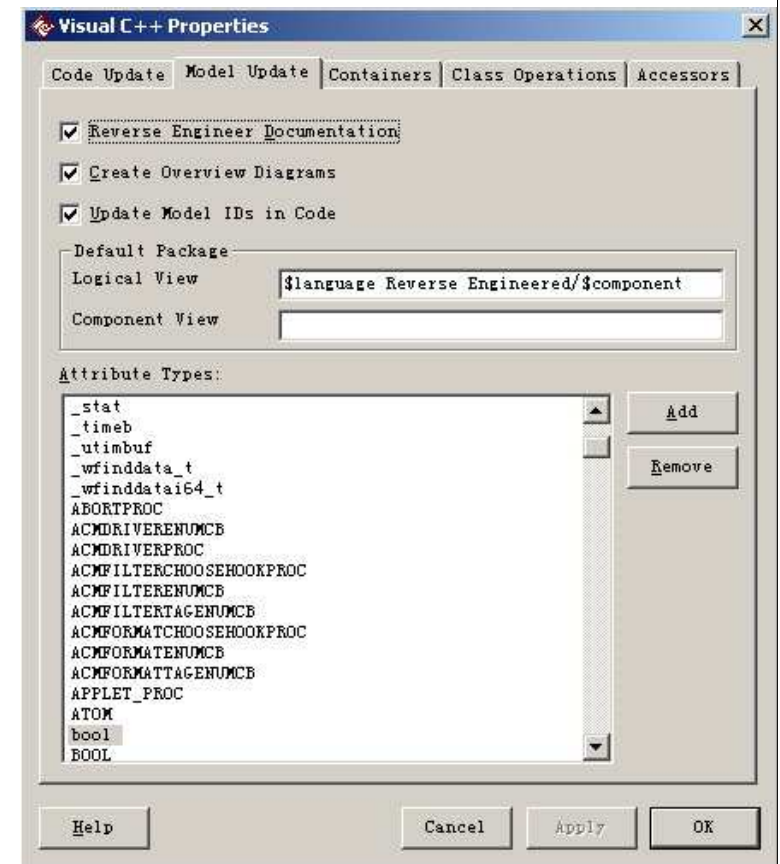


2 代码生成属性

■ 2.2 VC++属性对话框

— Model Update标签

- Create Overview Diagrams: 自动为每个逆向工程的构件创建一张综合图
- Default Package: 新模型元素所在包的名字
- Attribute Types: 首次逆向工程时应该作为属性（而不是角色）进行建模的VC++属性类型

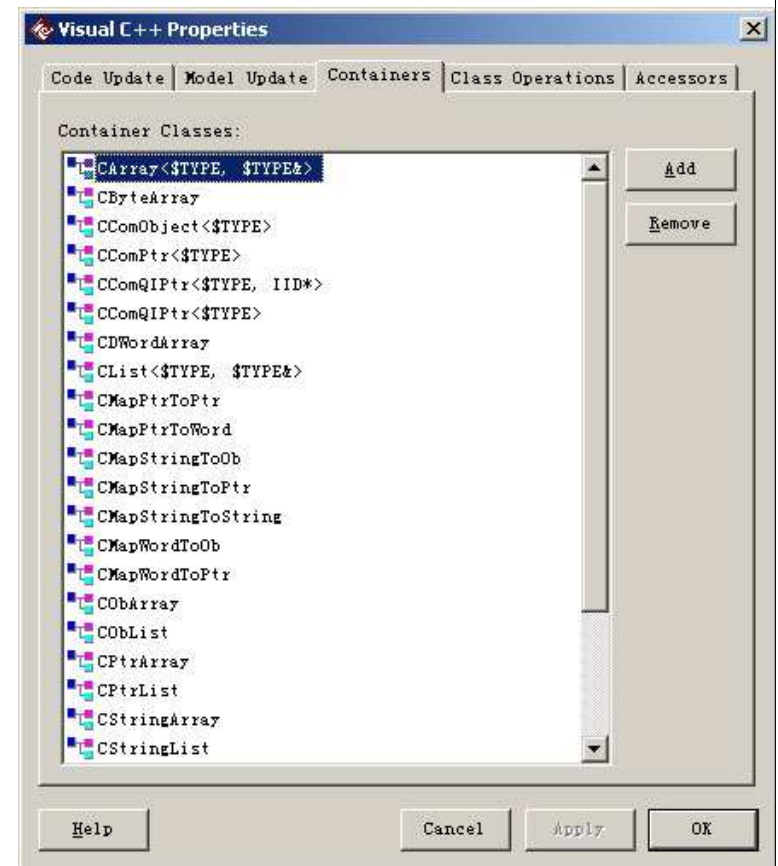


2 代码生成属性

■ 2.2 VC++属性对话框

– Containers标签

- 列出了在Model Assistant>Role标签中Implementation下拉列表框可选用的类。在默认的情况下，列表中提供的是最常用MFC容器类。也可以将用户自定义的容器类添加到列表中。右边的Add和Remove按钮用于添加和删除列表中的某个（或某些）容器类。
- 要修改其中的容器类，右单击要改动的类，在快捷菜单中选中Edit进行编辑

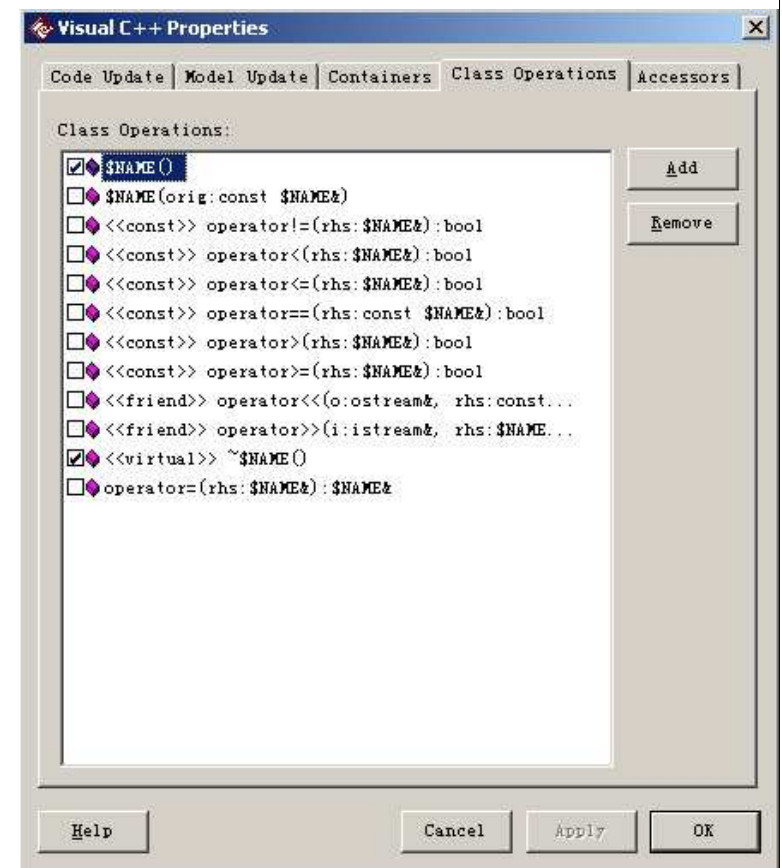


2 代码生成属性

■ 2.2 VC++属性对话框

– Class Operations标签

- 类操作的代码生成与否还依赖于Code Update中的Apply Pattern on Code Generation选项。左边选中的成员函数只在第一次代码生成过程中生成。
- Model Assistant将用该列表控制所能加入类的操作

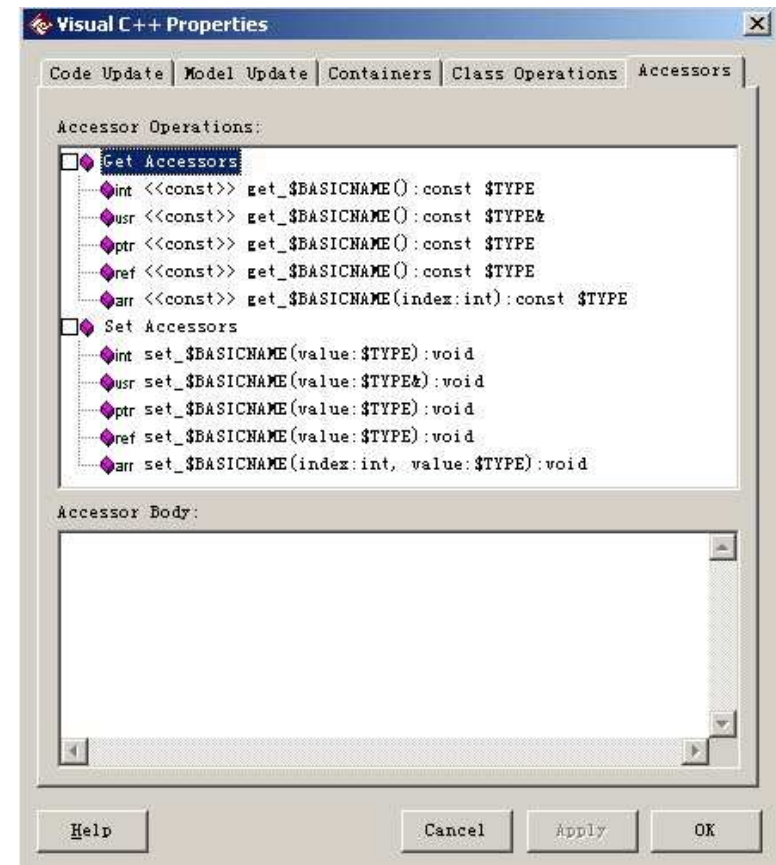


2 代码生成属性

■ 2.2 VC++属性对话框

– Accessors标签

- Accessor Operations:
访问函数（操作）。每个访问函数的原型由模型中角色和属性的类型决定
- Accessor Body: 访问操作的函数体。

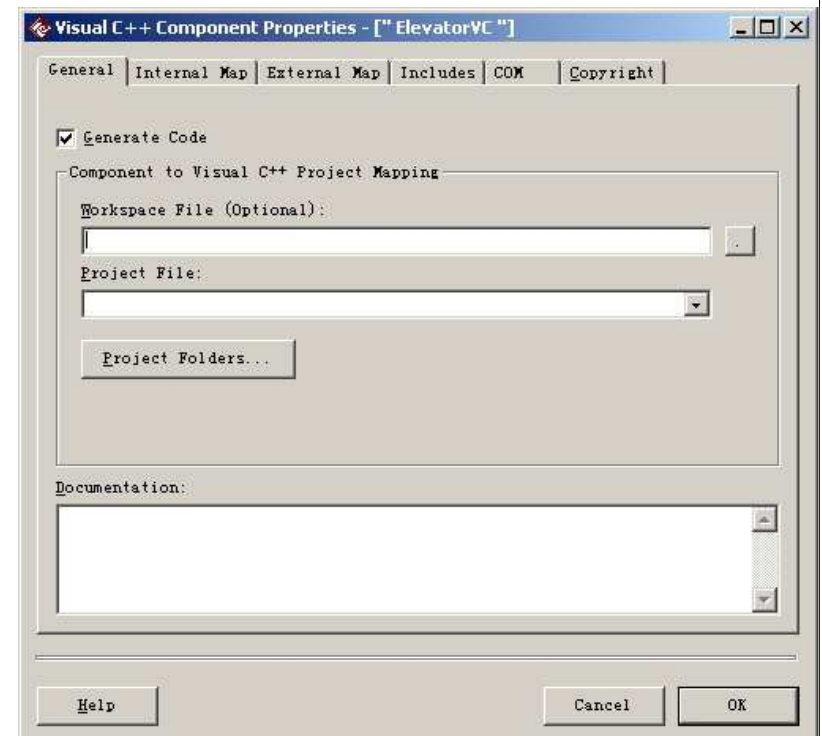


2 代码生成属性

■ 2.3 构件属性对话框

— General 标签

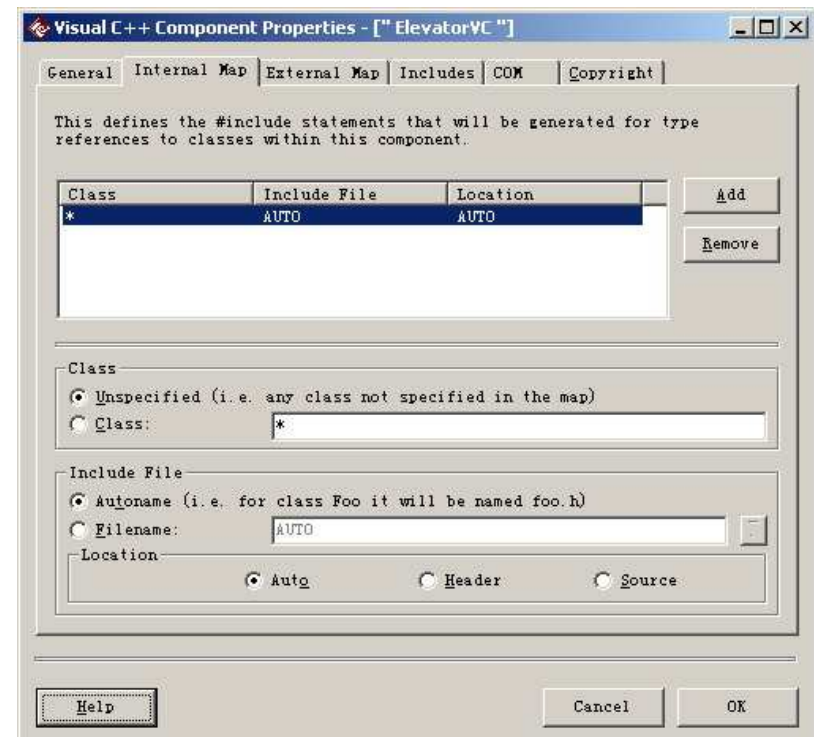
- Workspace File (Optional): VC++ 项目的工作空间和项目文件的名称和路径
- Workspace File: VC++ 项目工作空间文件的文件名和路径
- Project File: VC++ 项目文件的文件名和路径



2 代码生成属性

■ 2.3 构件属性对话框

- Internal Map（和 External Map）标签
 - Location: 控制在何处定义#include语句，有三种选择：
 - Auto: 根据类型依赖规则决定写入头文件或实现文件
 - Header: 总是写入头文件中
 - Source: 总是写入实现文件中



2 代码生成属性

■ 2.3 构件属性对话框

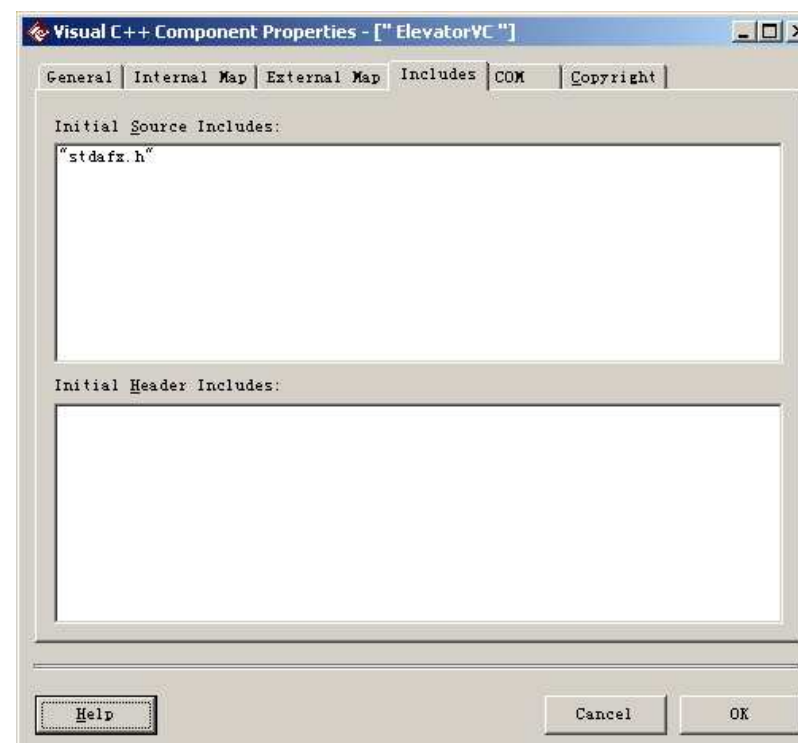
– Includes标签

- Initial Source

Includes: 写入类实现文件的#include语句中的文件名。如果有多个包含文件，则每个文件名占一行。

- Initial Header

Includes: 写入类头文件的#include语句中的文件名，每个文件名各占一行。

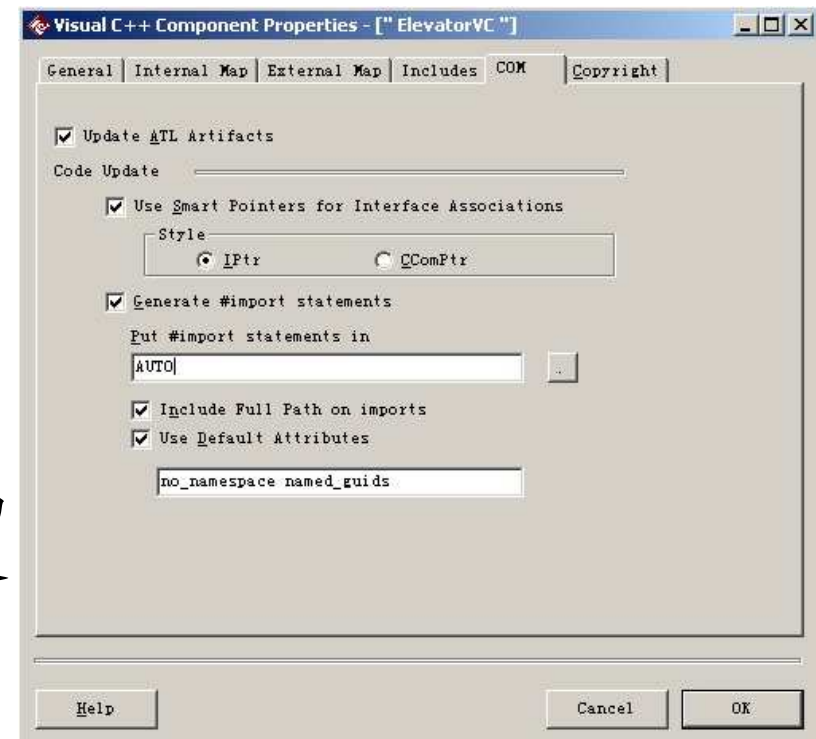


2 代码生成属性

■ 2.3 构件属性对话框

— COM标签

- Update ATL Artifacts:
控制是否为服务器端简单的ATL对象生成代码
- Use Smart Pointers for Interface Associations:
设定是否要在双向工程中使用VC++的 Smart Pointer特性。
- Generate #import statements: 控制是否为服务器端简单的ATL对象生成#import语句





2 代码生成属性

■ 2.3 构件属性对话框

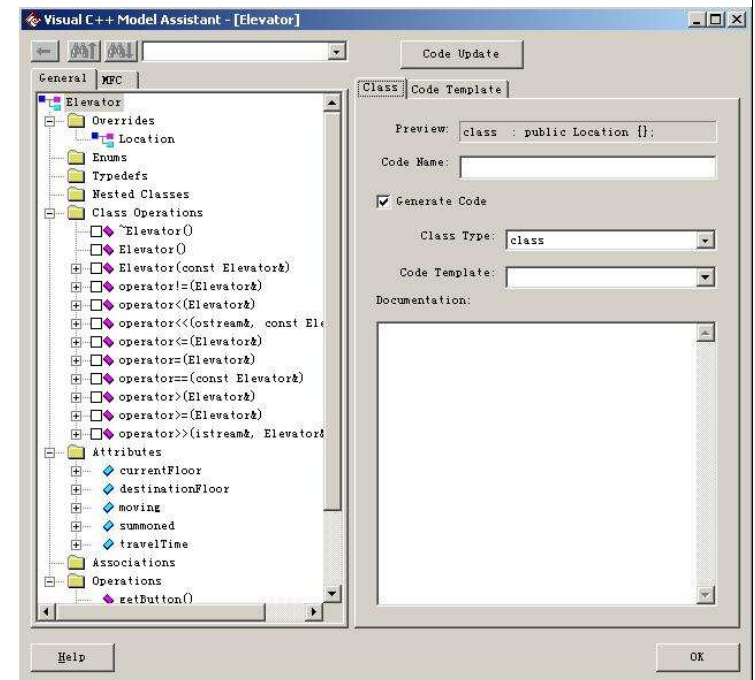
– Copyright标签

- 添加构件的版权信息，默认的价值为“Copyright (C) 1991 - 1999 Rational Software Corporation”。

2 代码生成属性

■ 2.4 Model Assistant工具

- Model Assistant是一个功能强大的工具，可以用来设定类以下层次的模型元素（包括类、操作、属性、关联等）的代码生成属性，精确定制生成的代码框架。利用该工具可以提高代码的准确性和简洁性。
- 启动Model Assistant工具：
 - 1.右单击浏览器或类图中的类；
 - 2.选择Model Assistant；

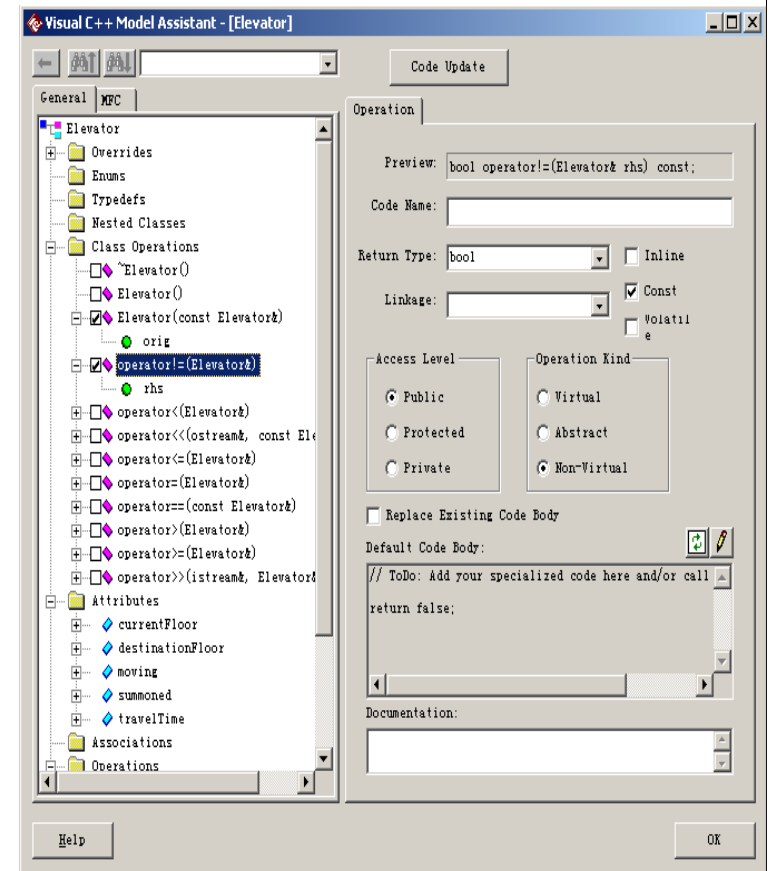


2 代码生成属性

■ 2.4 Model Assistant工具

– General树视图窗口及其标签

- 类及标签
- Class Operations文件夹
- Attributes 文件夹
- Operations文件夹
- Associations文件夹



2 代码生成属性

■ 2.4 Model Assistant工具 – MFC树视图窗口及标签

文件夹	描述
Class Node	MFC视图窗口允许你修改和设定一个MFC派生类的代码生成属性。可以在右边的Class标签中设置该类的代码生成属性。
MFC Overrides	显示根节点类中所有可用于重载的虚拟操作（只适用于MFC派生类）。点击操作前面的复选框可以将其设定为根节点类的重载操作。可以在Operation和Parameters标签中设置虚拟操作的属性。
Windows Messages	列出了根节点类所有可接收的Windows消息句柄。点击操作前面的复选框，可以在右边的Message Handler标签中修改和设定操作的代码生成属性。
Command Handlers	列出了基类的所有命令句柄。点击操作前面的复选框，可以在右边的Command Handler标签中修改和设定操作的代码生成属性。
Notification Handlers	列出了基类的所有notification句柄。点击操作前面的复选框，可以在右边的Notification Handler标签中修改和设定操作的代码生成属性。



3 Code Update Tool与代码生成

- Code Update Tool(代码生成向导) 简化正向工程操作：
 - 可以同时生成和更新多个用不同语言实现的源代码项目；
 - 可以保证模型和源代码之间的同步；
 - 可以将类映射到构件，方便操作；
 - 可以在代码生成向导中打开Model Assistant，设定类及其成员的代码生成属性，进一步设置类与代码之间的映射。
 - 可以在将模型转换成代码之前，提前预览类和类成员的代码，及时发现模型中的错误并加以修改。



3 Code Update Tool与代码生成

- 利用代码生成向导进行正向工程的步骤：
 - 第一步：启动Code Update Tool；
 - 第二步：选择要进行代码转换的类、构件以及实现语言：
 - 将构件映射到VC项目；
 - 指定要实现的类及其成员；
 - 第三步：生成代码；
 - 第四步：查看代码生成结果。



4 生成的代码内容

■ 4.1 概述

生成代码时，模型中的构件对应于VC项目，模型中的类对应于代码中的类。模型中的其他细节（包括属性、操作、关系、可见性等）在类的头文件和实现文件中体现

■ 4.2 头文件 (*.h) 框架代码

- 类声明
- 类的数据成员和成员函数声明
- 注释
- 反映代码生成属性设置的代码



4 生成的代码内容

■ 4.3 实现文件 (*.cpp) 框架代码

- #include语句
- 成员函数和数据成员的定义
- 反映代码生成属性设置的代码



5 VC++的逆向工程

利用Rose VC++的Model Update Tool可以方便地进行逆向工程的操作。在需要进行逆向工程时，都需要使用Model Update Tool，包括：

- 1) 根据VC++项目创建一个新的模型；
- 2) 针对代码的改动，更新一个已存在的模型；
- 3) 往模型中添加一个外部VC++构件。



5 VC++的逆向工程

■ 逆向工程的步骤

- 1.编译要转换的VC++项目，确保源代码文件中没有任何语法错误；
- 2.如果是创建新的模型，则需要创建一个构件，并设置构件的实现语言为VC++；如果是更新模型，则可以省略这一步；
- 3.单击Tools > Visual C++ > Update Model from Code，激活Model Update Tool，单击Next，接着出现Select Components and Classes窗口，在该窗口中进行构件和类的选择。
- 4.将VC++项目与模型中已有（或者新创建）的构件关联；



5 VC++的逆向工程

■ 逆向工程的步骤（续）

- 5.在要进行逆向工程转换的项目的各项名称前面选中复选框，或者选择构件边上的复选框，对整个项目进行逆向工程。
- 6.单击Next，出现显示Finish页面。
- 7.检查所要生成的模型元素，确定之后单击Finish按钮，也可以单击Back按钮返回并改动前面的设定。单击Finish按钮之后，开始逆向工程过程，进度在Progress页面显示。
- 8.在Summary页面查看结果。单击OK按钮，关闭Model Update Tool。
- 9.将生成的类移动到模型逻辑视图中相应的逻辑包中。
- 10.保存新的模型。

第12章 Visual Basic 的Rose双向工程



《Rational Rose 2003基础教程》

配套电子教案



内 容

- 正向工程的操作步骤
- 设置代码生成属性
- 生成的代码
- **VB**的逆向工程



引言

Rose VB集成了以下工具，对VB语言的双向工程提供支持：

1) Class Wizard: 可以在模型中创建和修改一个新的VB类；

2) Model Assistant: 用于正向工程过程中更新和精确指定一个类的代码生成属性；

3) Component Assignment Tool: 提供一个易于使用的接口，方便在模型中创建新的构件、将构件与源代码项目关联以及将类映射到构件。

4) Code Update Tool: 对模型和VB源代码项目进行双向工程。

5) Model Update Tool: 从源代码中收集信息，更新应用程序的设计模型。



1 正向工程的操作步骤

■ 正向工程的步骤如下：

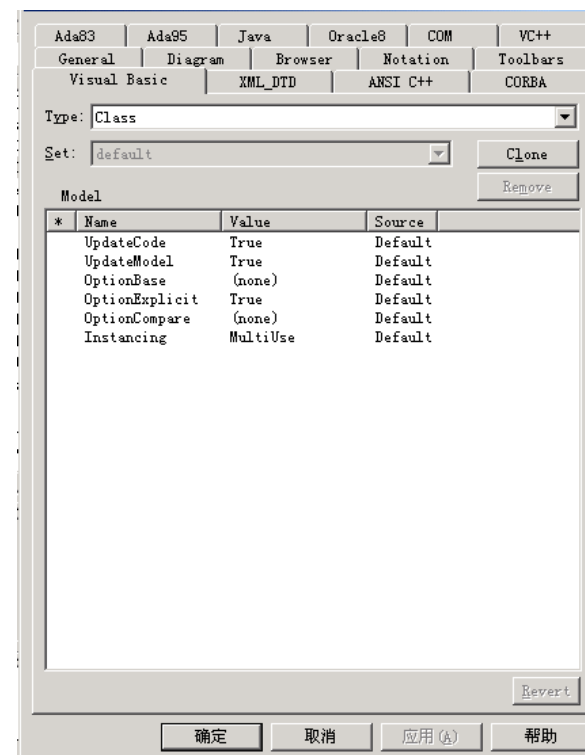
- 1.检查模型
- 2.创建构件；
- 3.将类映射到构件；
- 4.设置代码生成属性；
- 5.选择Class或Component框图中要生成代码的类或构件；
- 6.选择Tools>Visual Basic>Update Code, Rose VB插件启动Code Update Tool;
- 7.根据代码生成向导的提示，逐步完成正向工程。

2 设置代码生成属性

■ 2.1 VB语言属性窗口

— 1.Class属性

- UpdateCode: 控制是否为该类生成代码，默认值为True。
- UpdateModel: 控制类是否参与逆向工程过程，默认值为True。
- OptionBase: 设定数组的起始下标，默认值为空。
- OptionExplicit: 控制类代码中的变量名是否需要显式声明，默认值为True。
- OptionCompare: 设定字符串数据的默认比较方法，默认值为空。
- Instancing: 设定类对其它应用程序的可见性，默认值为MultiUse。



2 设置代码生成属性

■ 2.1 VB语言属性窗口

— 2.Attribute的属性

属性名	默认值	描述
New	False	控制Rose是否在模块变量声明之前添加New关键字。
ProcedureID	空	设定VB过程的ID。
PropertyName	空	设定属性的名字。属性包含一个数据成员和相关的属性过程，Model Assistant工具在显示数据成员及其相关的属性过程时，以属性名作为它们的节点文件夹（见Model Assistant工具）。
Subscript	空	设定Rose在为属性和角色生成数据成员时所使用的数组下标。默认为空表示Rose不生成下标；可接受值的形式有两种：“(1 to MaxLen)”和“()”，前者表示Rose将生成以给定的数字为起止下标的数组，后者表示Rose将生成一个动态数组。
WithEvents	False	设定生成代码时是否用关键字WithEvents修饰属性。



2 设置代码生成属性

■ 2.1 VB语言属性窗口

– 3.Operation的属性

- AliasName: 设定DLL中过程调用时的过程别名。
- DefaultBody: 指定生成代码时插入方法体中的代码和注释，默认值为空。
- IsStatic: 控制Rose是否为用户通过关键字Static指定的函数过程生成代码默认值为False。
- LibraryName: 指定包含方法的DLL名字。
- ProcedureID: 设定操作过程的ID。
- ReplaceExitingBody: 控制是否总是用默认的方法体覆盖现有的方法体，默认值为False



2 设置代码生成属性

■ 2.1 VB语言属性窗口

– 4.Param的属性

- ByRef: 设定VB参数的默认传递机制。值为True表示按引用传递。
- ByVal: 设定VB参数的默认传递机制，值为True，表示按值传递。
- Optional: 设定VB参数的传递机制为可选，默认值为False。
- ParamArray: 设定VB数组的传递机制，默认值为False。



2 设置代码生成属性

■ 2.1 VB语言属性窗口

– 5.Module Specification (构件)的属性

- ImportBinary: 控制是否加载从该构件编译出来的二进制类型库默认值为False。
- ImportReferences: 控制是否加载VB项目中所需的COM构件的类型库, 默认值为True。
- ProjectFile: 设定映射到构件的VB项目的.vbp文件路径, 默认值为空。



2 设置代码生成属性

■ 2.1 VB语言属性窗口

— 6.Generalize的属性

- FullName: 控制在Implements语句中是否使用包含构件名在内的完整的实现类名字，默认值为False。
- ImplementsDelegation: 控制泛化关系是否由实现委托功能实现，默认值为True。



2 设置代码生成属性

■ 2.1 VB语言属性窗口

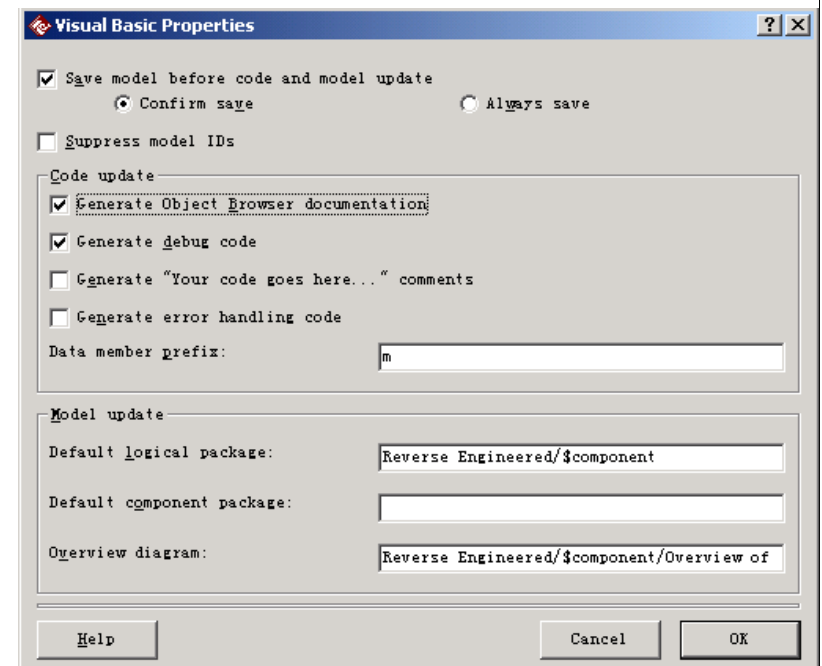
– 7.Role的属性

- **FullName**: 控制在角色声明语句中是否使用包含构件名在内的完整的引用类名字，默认值为True。
- **New**: 控制Rose是否在模块的变量声明语句中使用New关键字。
- **ProcedureID**: 设置VB过程ID，默认值为空。
- **PropertyName**: 设定属性的名字，见Attribute的代码生成属性或Model Assistant工具。
- **UpdateCode**: 控制是否为角色生成代码。

2 设置代码生成属性

■ 2.2 VB属性对话框

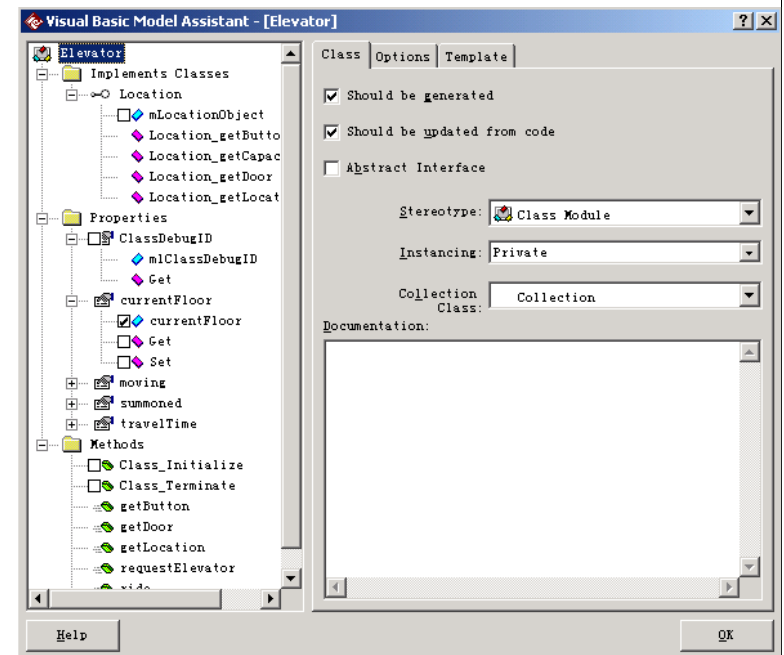
- Save model before code and model update :
- Suppress model IDs:
- Code update: 5个选项
用于正向工程
- Model update: 3个选项
用于逆向工程



2 设置代码生成属性

■ 2.3 Model Assistant工具

- 用于精确控制代码生成的内容
- 与VC++代码生成的作用类似，参考第11章

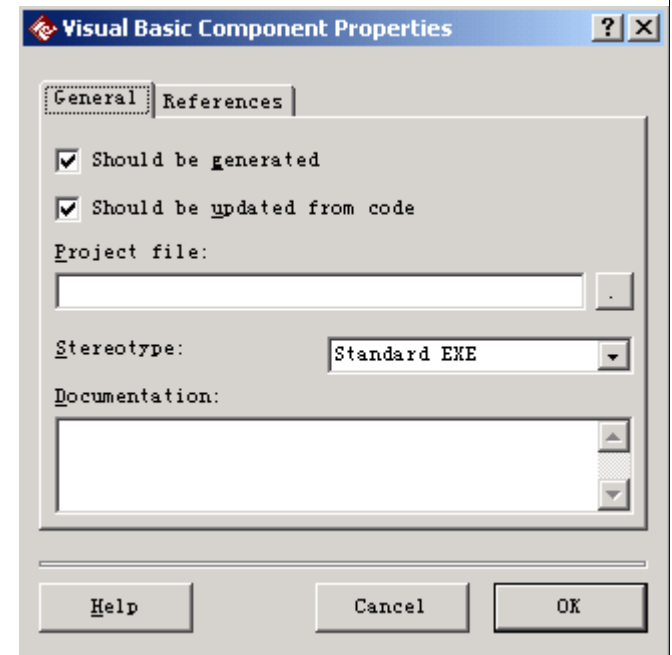


2 设置代码生成属性

■ 2.4 VB构件属性对话框

– General标签

- Should be generated: 设定构件是否参与正向工程。
- Should be updated from code: 设定构件是否参与逆向工程。
- Project file: 设定构件对应的VB项目文件。
- Stereotype:: 构件的构造型，对应于VB项目的类型。
- Documentation: 构件的说明文档。

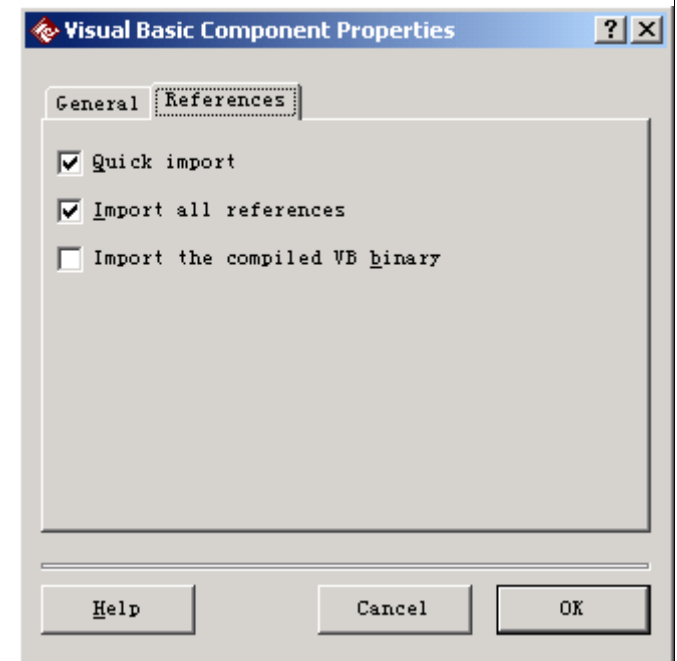


2 设置代码生成属性

■ 2.4 VB构件属性对话框

– References标签

- Quick import: 设定用VB项目更新构件时，是否加载模型中所引用的COM构件。
- Import all references: 设定是否加载所有的COM构件。
- Import the compiled VB binary: 设定是否加载从该构件编译出来的二进制类型库。





3 生成的代码

■ 3.1 类、属性和方法

- 类的属性代码内容包括：属性名、数据类型、可见性、初始值以及Get、Set、Let方法。具体的代码因（模型属性、代码生成属性）设置而异。
- 类操作在代码中体现为方法，代码内容包括：方法名、可见性、参数、参数数据类型、参数传递机制、返回值、错误处理、提示注释及调试语句等。具体的代码因（模型属性、代码生成属性）设置而异。
- Rose VB不仅生成完整的方法签名，而且还生成完整的方法体。

3 生成的代码

■ 3.2 类之间的关系

– 1) 关联关系

- 在VB中，关联关系映射为与关联关系的角色同名的属性（数据成员）



Public Users As B



Private Users As Collection

3 生成的代码

■ 3.2 类之间的关系

– 2) 泛化关系

- 因为VB并不支持继承，泛化关系变成实现委托



```
类 A: ↵
Option Explicit↵
↵
' ##ModelId=3F80E2D30148↵
Implements B↵
↵
' ##ModelId=3F80E2A10271↵
Private AttributeA As Variant↵
↵
' ##ModelId=3F80E302031C↵
Private mBObject As New B↵
↵
' ##ModelId=3F80E2AA033C↵
Public Sub OperationA()↵
↵
End Sub↵
↵
' ##ModelId=3F80E302033C↵
Private Sub B OperationB()↵
... Call mBObject.OperationB↵
End Sub↵
```

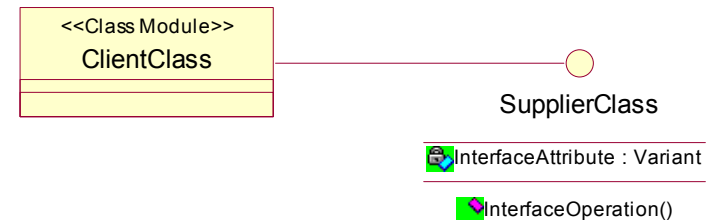
```
类 B: ↵
Option Explicit↵
↵
' ##ModelId=3F80C2B2002E↵
Private AttributeB As Boolean↵
↵
' ##ModelId=3F80C2C00251↵
Public Sub OperationB()↵
↵
End Sub↵
```

3 生成的代码

■ 3.2 类之间的关系

– 3) 实现关系

- 当一个类（客户类）需要调用接口（提供者类）中的操作，即客户类实现提供者类的接口时，两个类之间存在实现关系。在这种情况下，客户类要同时提供接口的属性和操作的实现。



```
类 ClientClass: ↵
Option Explicit↵
↵
' ##ModelId=3F81202101F4↵
Implements SupplierClass↵
↵
' ##ModelId=3F811F9B01E4↵
Public NewProperty As SupplierClass↵
↵
' ##ModelId=3F81204700EA↵
Private Sub SupplierClass_InterfaceOperation()↵
↵
End Sub↵
↵
' ##ModelId=3F81204700FA↵
Private Property Set SupplierClass_NewProperty(ByVal RHS As ClientClass)↵
↵
End Property↵
↵
' ##ModelId=3F8120470148↵
Private Property Get SupplierClass_NewProperty() As ClientClass↵
↵
End Property↵
```




3 生成的代码

■ 3.2 类之间的关系

– 4) 聚合关系

- 聚合关系在概念上等同于关联关系，正向工程中得到的代码与关联关系的代码相同。模型中的聚合表示一种整体与部分的关联，是概念意义上的抽象，对于代码生成没有其它特殊的影响。而且，在逆向工程中也反映不出聚合这种抽象关联。

– 5) 依赖关系

- 依赖关系在代码中没有体现，即并不生成属性。



4 VB的逆向工程

■ 逆向工程的步骤如下：

- 1.创建构件并将VB项目赋予构件；
- 2.选择Tools>Visual Basic>Update Model Tool；
- 3.选择要更新的模型和构件；
- 4.选择Finish完成逆向工程过程。