



Rational Rose 2000e
Using Rose

**Copyright © 1993–2000 Rational Software Corporation.
All rights reserved.**

Part Number: 800-023321-000

Revision 7.0, March 2000, (Software Release 2000e)

This document is subject to change without notice.

GOVERNMENT RIGHTS LEGEND: Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in the applicable Rational Software Corporation license agreement and as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (Oct 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14, as applicable.

Rational, the Rational logo, Rational Rose, ClearCase, and Rational Unified Process are trademarks or registered trademarks of Rational Software Corporation in the United States and in other countries. All other names are used for identification purposes only and are trademarks or registered trademarks of their respective companies.

Visual SourceSafe is a trademark or registered trademark of Microsoft Corporation.

T Quatrani VISUAL MODELING WITH RATIONAL ROSE AND UML, (pages 3,4,29,39,73,142,153). © 1998 Addison Wesley Longman Inc., -Reprinted by permission of Addison Wesley Longman. All rights reserved.



Contents

Contents iii

List of Figures xvii

List of Tables xxi

Preface xxiii

How this Manual is Organized xxiii

Online Help xxv

Online Manuals xxv

Related Documentation xxvi

File Names xxvi

Starting Rational Rose xxvii

Rose.ini Location xxvii

Chapter 1 Introduction to Visual Modeling Using Rational Rose 1

Visual Modeling 1

Modeling with Rational Rose 2

Notations 4

Features 4

Extending Rational Rose 5

Chapter 2 Getting Started with Rational Rose 7

- The Application Window 8
 - Control-Menu Box 8
 - Title Bar 9
 - Minimize and Maximize Buttons 9
 - Menu Bar 9
 - Toolbar 9
 - Toolbox 12
 - Customizing the Toolbox 14
 - Browser 14
- Documentation Window 14
- Diagram Window 15
- Overview Window 16
- Specification Window 17
- Printing Diagrams and Specifications 17
 - Print Preview 18
 - Zoom In and Zoom Out 18
 - Print 18
 - One Page and Two Page 18
 - Close 18
 - Apply Filter Dialog Box 18
- Saving in Various Formats 19
- Modifying the Rose.ini File 19
- Deleting Model Elements 20
 - Shallow Delete 20
 - Deep Delete 20

Chapter 3 The Browser 21

- Overview 21
- Viewing the Browser 22
 - Hiding and Displaying the Browser 22
 - Positioning the Browser 22

Docking and Undocking the Browser	23
Expanding and Collapsing the Browser Tree	23
Selecting Multiple Elements in the Browser	24
Navigating a Model	24
Creating and Editing Model Elements	25
Naming an Element in the Browser	26
Using Drag-and-Drop in the Browser	26
Browser to Browser Capabilities	27
Browser to Diagram Capabilities	28
Browser to Specification Capabilities	29
Sorting Packages in the Browser	29

Chapter 4 Introduction to Diagrams 31

Overview	31
Diagram Windows	31
Viewing Diagrams	32
Displaying Multiple Diagrams	33
Creating, Linking, Displaying, Renaming, and Deleting Diagrams	34
Create a New Diagram	34
Linking a Diagram	35
Display a Diagram	35
Rename a Diagram	36
Delete a Diagram	36
Selecting Multiple Elements in the Diagram	36
Creating and Naming Model Elements	37
Create an Element on the Diagram	37
Create an Element in the Browser	37
Naming Model Elements	37
Naming an Element on the Diagram	38
Creating/Naming an Overloaded Element on the Diagram	39
Placing an Overloaded Element on the Diagram from the Browser	39
Fully Qualified Names	40
Renaming Model Elements	40
Reassigning Model Elements	40

Manipulating Icons	41
Deleting, Cutting, Copying, and Pasting Icons	42
Correlations	43
Creating Correlations Between Elements	43
Adorning the Diagrams	45
Manipulating Text	45
Understanding Model Workspaces	46
Differences between a Saved Model and a Model Workspace	46
Model Workspace Sample	47
Saving a Model Workspace	49
Loading a Model Workspace	49

Chapter 5 Introduction to Specifications 51

Displaying Specifications	51
Custom Specifications	52
Editing Specifications	52
Common Specification Elements	53
Dialog Boxes	53
General Tab	53
Detail Tab	55
Files Tab	55
Tab Buttons	57
Navigating the Tabs	58
Adding and Deleting Entries	58
Editing Entries	58

Chapter 6 Class Diagrams and Specifications 61

Class Diagram Overview	61
Creating and Displaying a Class Diagram	62
Class Diagram Toolbox	62
Assigning a Class to Another Logical Package	64
Adding and Hiding Classes, and Filtering Class Relationships	64

Class Specification	65
Class Specification—General Tab	66
Type	66
Parent	66
Stereotype	67
Export Control	67
Class Specification—Detail Tab	68
Cardinality	69
Space	69
Persistence	70
Concurrency	71
Abstract	71
Formal Arguments	72
Class Specification—Operations Tab	72
Show Inherited	74
Class Specification—Attributes Tab	75
Class Specification—Relations Tab	77
Class Specification—Component Tab	78
Class Specification—Nested Tab	79
Class Specification—Files Tab	81
Class Attribute Specification	82
Class Attribute—General Tab	82
Class	83
Show Classes	83
Type	83
Initial Value	83
Class Attribute—Detail Tab	84
Containment	84
Static	85
Derived	85
Operation Specification	86
Operation Specification—General Tab	87
Return Class	87
Operation Specification—Detail Tab	88
Arguments	88
Protocol	88
Qualifications	89
Exceptions	89

Size	89
Time	89
Concurrency	90
Operation Specification—Preconditions Tab	91
Preconditions	91
Interaction Diagram	91
Operation Specification—Semantics Tab	92
Semantics	92
Interaction Diagram	92
Operation Specification—Postconditions Tab	93
Postcondition	93
Interaction Diagram	93
Operation Specification—Files Tab	94
Parameter Specification	94
Defining a New Parameter	94
Parameter Specification—General Tab	95
Default	95
Owner	95
Type	96
Association Specification	96
Association Specification—General Tab	97
Parent	97
Stereotype	97
Role	98
Element	98
Association Specification—Detail Tab	98
Derived	99
Link Element	99
Name Direction	99
Constraints	99
Association Specification—Role B General Tab	100
Association Specification—Role A and B Detail Tab	101
Navigable	101
Aggregate	102
Static	102
Friend	102
Containment of	102
Keys/Qualifiers	103

Generalize Specification	103
Generalize Specification—General Tab	104
Friendship Required	104
Virtual Inheritance	104
Realize Specification	104
Realize Specification—General Tab	105
Dependency Specification	105
Dependency Specification—General Tab	106
Has Relationship (Booch Only)	106
Has Specification—General Tab	107
Has Specification—Detail Tab	108
Key/Qualifier Specification	109
Defining a New Key/Qualifier	109
Key/Qualifier Specification—General Tab	110
Owner	110

Chapter 7 Use-Case Diagrams and Specifications 111

Use-Case Diagram Overview	111
Actors	112
Use Case	112
Flow of Events	113
Relationships	113
Association	114
Dependency	114
Extend Stereotype	115
Include Stereotype	115
Refine Stereotype	115
Generalization	116
Use-Case Diagram Toolbox	117
Use-Case Specification	118
Use-Case Specification—General Tab	118
Name	119
Package	119
Rank	119
Abstract	119

Use-Case Specification—Diagram Tab	120
Diagrams	120
Use-Case Specification—Relations Tab	121
Relations	121
Generalize Specification—General Tab	122
Stereotype	122
Friendship Required	123
Virtual Inheritance	123
Actor Specification	123
Chapter 8 State Machine Diagrams and Specifications	125
Creating and Displaying a State Machine Diagram	125
State Machine Specification	126
State Machine Specification General Tab	126
Statechart Diagram Overview	127
Creating a Statechart Diagram	127
Automatic Transmission Example	128
Activity Diagram Overview	129
Using Activity Diagrams	129
Understanding Workflows	129
Creating an Activity Diagram	130
Workflow Modeling	131
Purposes of Workflow Modeling	131
Defining a Workflow	131
Modeling a Workflow with an Activity Diagram	132
Activity Diagram-Specific Model Elements	133
Activities	133
Swimlanes	133
Objects	133
Object Flow	134
Understanding Objects and Object Flows	135
Changing the State of an Object	136
Shared State Machine Diagram Model Elements	137

States	137
Start and End States	137
Transitions	137
Transition to Self	138
Decisions	138
Synchronizations	138
Swimlane Specification	138
Swimlane Specification General Tab	139
State and Activity Specifications	139
State and Activity Specification General Tab	140
State and Activity Specification Actions Tab	141
Type	141
Action Expression	141
State and Activity Specification Transitions Tab	142
State and Activity Specification Swimlanes Tab	143
Action Specification	143
Transition Specification	145
Transition Specification – General Tab	145
State Transition Specification Detail Tab	146
Guard Condition	146
Transition Between Substates	147
Decision Specification	147
Decision Specification General Tab	148
Decision Specification Transitions Tab	149
Decision Specification Swimlanes Tab	150
Synchronization Specification	150
Synchronization Specification General Tab	151
Synchronization Specification Transitions Tab	152
Object Specification (Activity Diagrams)	152
Object Specification General Tab	153
Object Specification Incoming Object Flows Tab	154
Object Specification Outgoing Object Flows Tab	155
Object Flow Specification	155

Object Flow Specification General Tab 156

Chapter 9 Interaction Diagrams and Specifications 157

Interaction Diagram Overview 157

 Creating and Displaying an Interaction Diagram 157

Collaboration Diagrams 158

Sequence Diagrams 159

Toolboxes 160

 Collaboration Diagram Toolbox 160

 Sequence Diagram Toolbox 161

 Common Collaboration and Sequence Diagram Icons 162

 Object 162

 Messages 163

 Message Numbering 164

 Assigning an Operation to a Message 165

 Collaboration Specific Toolbox Icons 166

 Links 166

Sequence Numbering 167

 Top-Level Numbering 167

 Hierarchical Numbering 167

 Scripts 167

Focus of Control 169

 Displaying Focus of Control 169

 Coloring Focus of Control 170

 Moving the Focus of Control 170

 Nested Focus of Control 170

Creating Alternative Diagrams 170

 Toggling between Interaction Diagrams 171

 Creating a Collaboration Diagram from a Sequence Diagram 171

 Creating a Sequence Diagram from a Collaboration Diagram 171

Object Specification 171

 Object Specification—General Tab 172

 Name 172

 Class 173

Persistence Field	173
Multiple Instances Check Box	173
Class Instance Specifications	174
Class Instance Specification—General Tab	174
Class	175
Link Specification	175
Link Specification—General Tab	176
Assoc	176
Supplier & Client Visibility	177
Shared	178
Role	178
Link Specification—Messages Tab	179
Icon	179
Sequence	179
Message Name	180
Receiver	180
Message Specification	180
Message Specification—General Tab	181
Class	181
Message Specification—Detail Tab	182
Synchronization	183
Frequency	183

Chapter 10 Component Diagrams and Specifications 185

Component Diagram Overview	185
Creating and Displaying a Component Diagram	186
Component Diagram Toolbox	186
Assigning a Component to Another Package	186
Component Specifications	187
Component Specification—General Tab	188
Stereotype (Component)	188
Language	188
Component Specification—Detail Tab	189
Declarations	190
Component Specification—Realizes Tab	190
Show all Classes	191

Classes	191
Language	191
Component Specification—Files Tab	191
Package Specification	192
Package Specification—General Tab	192
Package	193
Package Specification—Detail Tab	193
Component Diagrams	193
Package Specification—Realizes Tab	194
Package Specification—Files Tab	194

Chapter 11 Deployment Diagrams and Specifications 195

Deployment Diagram Overview	195
Creating and Displaying a Deployment Diagram	196
Deployment Diagram Toolbox	196
Processor Specification	196
Processor Specification—General Tab	197
Processor Specification—Detail Tab	198
Characteristics	198
Processes	199
Scheduling	199
Device Specification	200
Device Specification—General Tab	200
Device Specification—Detail Tab	201
Connection Specifications	201
Process Specification	202
Process Specification—General Tab	203
Processor	203
Priority	203

Chapter 12 Stereotypes 205

Overview	205
Benefits to Using Stereotypes	205
User-Defined Stereotypes	206

Viewing Stereotypes	206
Diagram Tab	207
Browser Tab	208
Creating Stereotypes	209
Creating a New Stereotype for the Current Model	209
Creating a New Stereotype Configuration File	209
Creating a New Stereotype for All Rose Models	210
Creating Stereotype Icons	211
Creating a Diagram Icon	212
Creating Diagram Toolbox and List View Icons	212
Adding Stereotypes to the Diagram Toolbox	213
Subsystem Stereotype Package	214
Subsystem Stereotype Sample	214
Chapter 13 Framework Wizard Add-In	217
Activating the Framework Wizard Add-In	217
Creating a New Model from a Framework	218
Creating and Deleting Frameworks	219
The Framework Library	219
Creating a New Framework	220
Changing or Deleting a Framework	222
Chapter 14 Type Library Importer	223
What Is a Type Library?	223
Why Would I Want to Import Type Libraries into the Model?	224
What COM Components Can Be Imported into the Model?	224
How Is a Type Library Presented?	225
A Type Library in Rational Rose	225
A Type Library in the OLE Viewer in Visual Studio	231
A Type Library in the Object Browser in Visual Basic	232
Importing a Type Library Into the Model	233

Contents

Importing a New Version of an Existing Type Library	234
Hiding Type Library Items	234
Show Hidden Items Selected	234
Show Hidden Items Cleared	235
Using an Imported Type Library	236
Adding Class Members to a Quick Import Type Library	237
Customizing the Type Library Importer	237

Appendix A Upgrading From a Previous Release 241

Upgrading from Rational Rose 3.0 or Later	241
Upgrading from Releases Prior to Rational Rose 3.0	241
Understanding Petal File Versions	242

Appendix B Contacting Technical Support 243

When Contacting Rational Technical Support	243
How to Contact Rational Customer Support	244
Telephone and E-mail	244
Fax	245
Rational Web Site	245
Rational Technical Support Call Center Contact Information	245
North America	245
Europe	245
Asian Pacific	246

Index 247



List of Figures

- Figure 1 Application Window 8
- Figure 2 Standard Toolbar 9
- Figure 3 Application Window 22
- Figure 4 Browser—Collapsed and Expanded Tree 23
- Figure 5 *Navigating a Model* 25
- Figure 6 Diagram Window 32
- Figure 7 Multiple Diagrams—Cascade Windows 33
- Figure 8 Multiple Diagrams—Tiled Windows 34
- Figure 9 Selected Elements in a Diagram 37
- Figure 10 Model Workspace Loaded Units 48
- Figure 11 General Tab 53
- Figure 12 Detail Tab 55
- Figure 13 Files Tab 56
- Figure 14 Tab Buttons 57
- Figure 15 Class Diagram Example 61
- Figure 16 Class Diagram Toolbox 63
- Figure 17 Class Specification—General Tab 66
- Figure 18 Class Specification—Detail Tab 68
- Figure 19 Class Specifications—Operations Tab 73
- Figure 20 Class Specification—Attributes Tab 75
- Figure 21 Class Specification—Relations Tab 77
- Figure 22 Class Specification—Component Tab 78
- Figure 23 Class Specification—Nested Tab 80
- Figure 24 Class Attribute—General Tab 82
- Figure 25 Class Attribute—Detail Tab 84

List of Figures

- Figure 26 Operations Specification—General Tab 87
- Figure 27 Operation Specification—Detail Tab 88
- Figure 28 Operation Specification—Precondition Tab 91
- Figure 29 Operations Specification—Semantics Tab 92
- Figure 30 Operation Specification—Postcondition Tab 93
- Figure 31 Parameter Specification—General Tab 95
- Figure 32 Association Specification—General Tab 97
- Figure 33 Association Specification—Detail Tab 98
- Figure 34 Association Specification—Role A and B General Tab 100
- Figure 35 Association Specification—Role A and B Detail Tab 101
- Figure 36 Generalize Specification—General Tab 104
- Figure 37 Realize Specification—General Tab 105
- Figure 38 Dependency Specification—General Tab 106
- Figure 39 Has Specification—General Tab 107
- Figure 40 Has Specification—Detail Tab 108
- Figure 41 Key/Qualifier Specification—General Tab 110
- Figure 42 Use-Case Diagram Toolbox 117
- Figure 43 Use-Case Specification—General Tab 118
- Figure 44 Use-Case Specification—Diagram Tab 120
- Figure 45 Use-Case Specification—Relations Tab 121
- Figure 46 Generalize Specification—General Tab 122
- Figure 47 State Machine Specification—General Tab 126
- Figure 48 Automatic Transmission Example 128
- Figure 49 Objects on an Activity Diagram Sample 134
- Figure 50 Object Flow Sample 135
- Figure 51 CD Player Sample 136
- Figure 52 Swimlane Specification—General Tab 139
- Figure 53 State and Activity Specification—General Tab 140
- Figure 54 State and Activity Specification—Actions Tab 141
- Figure 55 State and Activity Specification—Transitions Tab 142
- Figure 56 State and Activity Specification—Swimlanes Tab 143
- Figure 57 State Transition Specification—General Tab 145
- Figure 58 State Transition Specification—Detail Tab 146
- Figure 59 Decision Specification—General Tab 148
- Figure 60 Decision Specification—Transition Tab 149
- Figure 61 Decision Specification—Swimlane Tab 150

Figure 62 Synchronization Specification—General Tab 151
Figure 63 Synchronization Specification—Transitions Tab 152
Figure 64 Object Specification—General Tab 153
Figure 65 Object Specification—Incoming Object Flows Tab 154
Figure 66 Object Specification—Outgoing Object Flows Tab 155
Figure 67 Object Flow Specification—General Tab 156
Figure 68 Collaboration Diagram Example 159
Figure 69 Sequence Diagram Example 160
Figure 70 Collaboration Diagram Toolbox 161
Figure 71 Sequence Diagram Toolbox 161
Figure 72 Multiple Object Diagram 163
Figure 73 Focus of Control Diagram Example 169
Figure 74 Object Specification—General Tab 172
Figure 75 Class Instance Specification—General Tab 174
Figure 76 Link Specification—General Tab 176
Figure 77 Link Specification—Message Tab 179
Figure 78 Message Specification—General Tab 181
Figure 79 Message Specification—Detail Tab 182
Figure 80 Component Diagram Example 185
Figure 81 Component Diagram Toolbox 186
Figure 82 Component Specification—General Tab 188
Figure 83 Component Specification—Detail Tab 189
Figure 84 Component Specification—Realizes Tab 190
Figure 85 Package Specification—General Tab 192
Figure 86 Package Specification—Detail Tab 193
Figure 87 Deployment Diagram Example 195
Figure 88 Deployment Diagram Toolbox 196
Figure 89 Processor Specification—General Tab 197
Figure 90 Processor Specification—Detail Tab 198
Figure 91 Device Specification—General Tab 200
Figure 92 Device Specification—Detail Tab 201
Figure 93 Process Specification—General Tab 203
Figure 94 Options Dialog Box—Diagram Tab 207
Figure 95 Options Dialog Box—Browser Tab 208
Figure 96 Subsystem Stereotype Sample 214
Figure 97 Create New Model Dialog Box 218

List of Figures

- Figure 98 Framework Wizard Specification Page 221
- Figure 99 Framework Wizard Summary Page 222
- Figure 100 The Component View of the Microsoft Scripting Runtime Type Library 225
- Figure 101 The Component Overview Diagram for a Model 226
- Figure 102 The Logical View of the Microsoft Scripting Runtime Type Library 227
- Figure 103 The Logical Overview Diagram of the Microsoft Scripting Runtime Type Library 228
- Figure 104 The OLE Viewer in Visual Studio 231
- Figure 105 The Object Browser in Visual Basic 232
- Figure 106 Type Library with Show Hidden Items Option Selected 235
- Figure 107 Type Library with Show Hidden Items Option Cleared 236
- Figure 108 The COM Properties Dialog Box 238



List of Tables

Table 1	Print Dialog Box Tabs	17
Table 2	Browser to Browser Capabilities	27
Table 3	Browser to Diagram Capabilities	28
Table 4	Browser to Specification Capabilities	29
Table 5	Export Control Field Options	67
Table 6	Cardinality Field Options	69
Table 7	Persistence Field Options	70
Table 8	Class Concurrency Options	71
Table 9	Physical Containment Options	84
Table 10	Concurrency Field Options	90
Table 11	Containment Field Options	103
Table 12	Persistence Field Options	173
Table 13	Supplier & Client Visibility Options	177
Table 14	Synchronization Options	183
Table 15	Frequency Options	183
Table 16	Scheduling Field Options	199
Table 17	COM Stereotypes	229
Table 18	Rational Rose Petal File Versions	242



Preface

The *Rational Rose 2000e, Using Rose* manual provides important usage and reference information to effectively use Rational Rose. This manual is designed for all users of Rational Rose, and provides information on the following Rational Rose features and concepts:

- Diagrams
- Specifications
- The Browser
- Standard Rational Rose Add-Ins

The *Rational Rose 2000e, Using Rose* manual does not contain information on the standard Rational Rose add-ins. For more information on these add-ins, refer to the language-specific documentation shipped with this edition.

To use Rational Rose 2000e, you should be comfortable with basic Windows techniques.

Rational Rose includes an online tutorial which provides a hands-on introduction to Rational Rose features. The Rational Rose tutorial resides on the Documentation CD. However, the primary source for reference information is the online help.

How this Manual is Organized

Chapter 1—Introduction to Visual Modeling Using Rational Rose

Introduces you to Visual Modeling and Rational Rose. It also gives a brief explanation of UML Notation and describes the many features of Rational Rose.

Chapter 2—Getting Started with Rational Rose

Describes how the Rational Rose graphical user interface displays, creates, modifies, manipulates, and documents the elements of a model using the Application, Documentation, Diagram, and Specification windows.

Chapter 3—The Browser

Explains the various capabilities of the browser including how to navigate in a model.

Chapter 4—Introduction to Diagrams

Explains how to create, display and modify diagrams in a model.

Chapter 5—Introduction to Specifications

Explains how to display and edit a specification. This chapter also explains the common elements of a specification and how to navigate through the default tabs.

Chapter 6—Class Diagrams and Specifications

Describes the class diagram, the class specification and all of the specifications that derive from a class diagram. The Class Attribute, Operation, Association, Generalize, Dependency, Logical Package, Has Relationship, and Key/Qualifier Specifications are all described in this chapter.

Chapter 7—Use-Case Diagrams and Specifications

Explains how use-case diagrams graphically depict system behavior. This chapter also lists the features of the Use-Case and Actor Specifications.

Chapter 8—State Machine Diagrams and Specifications

Describes the two diagrams contained within a state machine: statechart diagrams and activity diagrams. The chapter explains the concept of workflow modeling in activity diagrams. The chapter also explains the specifications and model elements associated with statechart and activity diagrams.

Chapter 9—Interaction Diagrams and Specifications

Describes the differences and similarities between collaboration and sequence Diagrams. The chapter also explains Focus of Control, Message Numbering, and the various specifications associated with both diagrams.

Chapter 10—Component Diagrams and Specifications

Explains the component diagram and toolbox. The chapter also describes the Component and Package Specifications.

Chapter 11—Deployment Diagrams and Specifications

Provides an overview of the deployment diagram and explains the Processor, Device, Connection and Process Specifications.

Chapter 12—Stereotypes

Describes the benefits of using stereotypes and how to create user-defined stereotypes. This chapter also explains how to create and add stereotypes to a model.

Chapter 13—Framework Add-In

Provides instruction on activating the Framework Add-In, creating a new model from a framework, and deleting a framework. This chapter also describes the framework library.

Chapter 14—Type Library Importer

The Type Library Importer Add-In allows you to import the type library of COM (Component Object Model) components into your model by dragging the file from the Windows Explorer and drop it in Rational Rose.

Online Help

Rational Rose 2000e includes comprehensive online help with hypertext links and a two-level search index.

Online Manuals

Rational Rose 2000e includes versions of all user manuals online on the Documentation CD.

Related Documentation

After installation and before you begin using Rational Rose, please review any `readme.txt` files and **Release Notes** to ensure that you have the latest information about the product. The release notes are included with your product documentation and are available online from the **Start** menu.

For additional resources on UML, Visual Modeling, and Rational Rose, refer to the following publications:

- Quatrani, Terry. *Visual Modeling with Rational Rose and UML*. Reading, MA. Addison Wesley Longman, Inc., 1998 (Included with product documentation).

This book introduces the most popular and influential elements—the Rational Objectory Process, the Unified Modeling Language (UML), and Rational Rose—and offers practical direction on specifying visualizing, documenting, and creating software solutions.

- Kruchten, Philippe. *The Rational Unified Process - An Introduction*. Reading, MA. Addison Wesley Longman, Inc., 1998.

This book introduces you to the Rational Unified Process.

- Booch, Grady; Rumbaugh, James; Jacobson, Ivar. *The Unified Modeling Language User Guide*. Reading, MA. Addison Wesley Longman, Inc., 1999.

This book teaches you how to use and apply the Unified Modeling Language.

File Names

Where file names appear in examples, Windows syntax is depicted. To obtain a legal UNIX file name, eliminate any drive prefix and change the backslashes to slashes:

`c:\project\username`

becomes

`/project/username`

Starting Rational Rose

On start-up, Rational Rose extracts information from the registry and from the rose.ini file.

Rose.ini Location

The rose.ini file is located in the following locations:

- In **Windows 95/98**, the rose.ini file is located in the same folder as the Rational Rose executable.
- In **Windows 2000**, the rose.ini file is located at: C:\Documents and Settings\<<username>>\Application Data\Rational\Rose\6.0
- In **Windows NT**, the rose.ini file is located at:
C:\WINNT\Profiles\<<username>>\Application
Data\Rational\Rose\6.0
- On UNIX platforms, the rose.ini file is located in the in home directory.

For more information on the rose.ini file, refer to the online help.



Chapter 1

Introduction to Visual Modeling Using Rational Rose

Rational Rose provides support for two essential elements of modern software engineering: component based development and controlled iterative development. While these concepts are conceptually independent, their usage in combination is both natural and beneficial. [Rational Rose's model-diagram architecture facilitates use of the Unified Modeling Language (UML), Component Object Modeling (COM), Object Modeling Technique (OMT), and Booch '93 method for visual modeling. Using semantic information ensures correctness by construction and maintaining consistency.]

Visual Modeling

Increasing complexity, resulting from a highly competitive and ever-changing business environment, offers unique challenges to system developers. Models help organize, visualize, understand, and create complex things.

Visual Modeling is the mapping of real world processes of a system to a graphical representation. Models are useful for understanding problems, communicating with everyone involved with the project (customers, domain experts, analysts, designers, etc.), modeling complex systems, preparing documentation, and designing programs and databases. Modeling promotes better understanding of requirements, cleaner designs, and more maintainable systems.

As software systems become more complex, we cannot understand them in their entirety. To effectively build a complex system, the developer begins by looking at the big picture without getting caught up in the details. A model is an ideal way to portray the abstractions of a complex problem by filtering out nonessential details. The developer

must abstract different views or blueprints of the system, build models using precise notations, verify that the models satisfy the requirements of the system, and gradually add detail to transform the models into an implementation.

The models of a software system are analogous to the blueprints of a building. An architect could not design a structure in its entirety with one blueprint. Instead a blueprint is drawn up for the electrician, the plumber, the carpenter, and so on. When designing a software system, the software engineer deals with similar complexities. Different models are drawn up to serve as blueprints for marketing, software developers, system developers, quality assurance engineers, etc. The models are designed to meet the needs of a specific audience or task, thereby making them more understandable and manageable.

Visual modeling has one communication standard: Unified Modeling Language (UML). The UML provides a smooth transition between the business domain and the computer domain. By communicating with the UML, all members of a design team work with a common vocabulary, minimizing miscommunication and increasing efficiency. Visualizing the components and relationships of a complex system make it easier to understand than describing it with words.

Visual modeling captures business processes by defining the software system requirements from the user's perspective. This streamlines the design and development process. Visual modeling also defines architecture by providing the capability to capture the logical software architecture independent of the software language. This method provides flexibility to your system design since the logical architecture can always be mapped to a different software language. Finally, with visual modeling, you can reuse parts of a system or an application by creating components of your design. These components can then be shared and reused by different members of a team allowing changes to be easily incorporated into already existing development software.

Modeling with Rational Rose

Rational Rose is the visual modeling software solution that lets you create, analyze, design, view, modify and manipulate components and implement systems in a way that makes them truly easy to communicate. You can graphically depict an overview of the behavior of your system with a use-case diagram. Rational Rose provides the collaboration diagram as an alternate graphical representation of a

use-case diagram. It shows object interactions organized around objects and their links to one another. The statechart diagram provides additional analysis techniques for classes with significant dynamic behavior. A statechart diagram shows the life history of a given class, the events that cause a transition from one state to another, and the actions that result from a state change. Activity diagrams provide a way to model the workflow of a business process or a way to model a class operation.

Rational Rose provides the notation needed to specify and document the system architecture. The logical architecture is captured in class diagrams that contain the classes and relationships that represent the key abstractions of the system under development. The component architecture is captured in the component diagrams that focus on the actual software module organization within the development environment. The deployment architecture is captured in a deployment diagram that maps software to processing nodes—it shows the configuration of run-time processing elements and the software processes living in them.

Rational Rose lets you realize all the benefits of visual modeling. It is designed to provide software developers with a complete set of visual modeling tools for development of robust, efficient solutions to real business needs.

Notations

Notation plays an important part in any application development activity—it is the glue that holds the process together. UML provides a very robust notation, which grows from analysis into design. Certain elements of the notation (that is, use cases, classes, associations, aggregations, inheritance) are introduced during analysis. Other elements of the notation (that is, containment indicators and properties) are introduced during design.

Notation has the following roles:

- Communicates decisions that are not obvious or cannot be inferred from the code itself
- Provides semantics that capture important strategic and tactical decisions
- Offers a concrete form and tools to manipulate

Features

Rational Rose provides the following features to facilitate the analysis, design, and iterative construction of your applications:

- Use-Case Analysis
- Object-Oriented Modeling
- User-Configurable Support for UML, COM, OMT, and Booch '93
- Semantic Checking
- Support for Controlled Iterative Development
- Round-Trip Engineering
- Parallel Multiuser Development through Repository and Private Support
- Integration with Data Modeling Tools
- Documentation Generation
- Rational Rose Scripting for Integration and Extensibility
- OLE Linking
- OLE Automation
- Multiple Platform Availability

Extending Rational Rose

The add-in feature allows you to quickly and accurately customize your Rational Rose environment depending on your development needs. Using the add-in tool, you can install language (for example, Visual Basic, Visual Java, etc.) and non- language (for example Microsoft Project) tools while in Rational Rose.

When an add-in is installed, it is automatically in an activated state. Add-Ins can install:

- Menus (.mnu file)
- Help files (.hlp file)
- Contents tab file (.cnt file)
- Properties (.pty file)
- Executables (.exe)
- Script files (.ebs script source file and .ebx compiled script file)
- OLE servers (.dll file)

Additionally, an add-in can define fundamental types, predefined stereotypes, and metafiles. Note that an add-in is not to be considered strictly a one-to-one association with a round-trip engineering (RTE) integration.

Add-In Manager

The Add-In Manager allows you to control the state of the add-in, whether it is activated or deactivated. If the add-in is deactivated, it is still visible through the Add-In Manager. However, the add-in's properties and menus are not available.

Installing an Add-In

Use the following steps to install an add-in on your Windows 95, Windows 98, or Windows NT system:

1. Exit Rational Rose.
2. Insert the application's CD ROM that you wish to install.
3. Run the `setup.exe` program.
4. Respond to the dialogs to complete your installation.
5. Restart Rational Rose. Confirm that your add-in is activated using the **Add-In Manager** menu.



Chapter 2

Getting Started with Rational Rose

When you first start Rational Rose, some editions will display a **Framework** dialog box. From this dialog box, you can load a model with predefined model elements, allowing you to focus your modeling efforts on the parts that are unique to your system, instead of “reinventing the wheel.” For further information on the Framework Wizard, refer to the 13 Framework Wizard Add-In.

Independent of Frameworks, Rational Rose’s graphical user interface displays, creates, modifies, manipulates, and documents the elements in a model using four kinds of windows:

- Application window
- Documentation window
- Diagram window
- Specification window

Rational Rose displays the diagram, specification and documentation windows within the application window.

The Application Window

An application window contains a control-menu box, a menu bar, a title bar, a toolbar, a toolbox, a minimize button, a maximize button, the browser, and the documentation window.

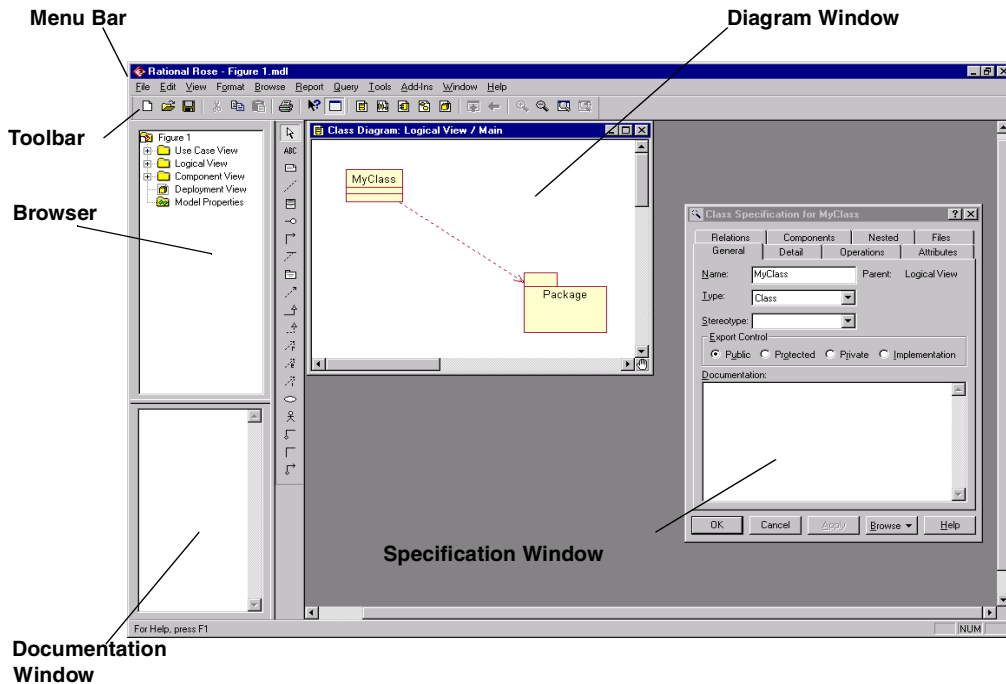


Figure 1 Application Window

Control-Menu Box

Clicking the Control-Menu Box (on the application or diagram window) displays a drop-down list box with the following options:

Restore: Restores focus to that diagram window.

Move: Highlights the border of the window. Move your pointer to the Title Bar, click and drag the window to the desired location.

Size: Highlights the border of the window. Move your pointer to the border and resize the window as desired.

Minimize: Reduces the window to an icon placing it in the bottom of the application window.

Maximize: Enlarges the window to consume the entire screen.

Close: Closes the window.

Title Bar

The title bar always displays diagram type. Additional information (like the view or diagram name) is often displayed depending on the diagram/model being viewed.

Minimize and Maximize Buttons

These buttons allow you to minimize or maximize the diagram or application window.

Menu Bar

The menu bar changes depending on which diagram you are currently working on. For more information regarding the menu bar see the *Diagram Overview* chapter later in this manual.

Toolbar

The standard toolbar is displayed directly under the menu bar, along the top of the application window. This toolbar is independent of the open diagram window.

The following icons are available for use on the standard toolbar, independent of the open diagram window.

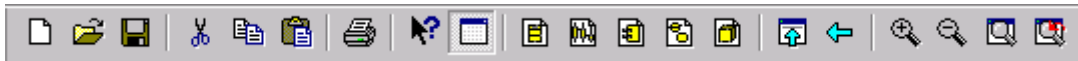


Figure 2 Standard Toolbar

Create New Model

Clicking the **New File** icon creates a new model.

Open Existing Model

Clicking the **Open Model** icon from the toolbar opens the **Load Model** dialog box. You can open a model from anywhere within the design.

Create and **Open** icons: If you have a model open when you click either the **Create** or **Open** icon, you are asked to save your current model. Clicking **No** discards all changes since your last save. Clicking **Yes** saves your changes and opens a new model, or displays the **Load Model** dialog box automatically.

Save Model or Log

Clicking the **Save Model** icon opens the **Save Model to** dialog box. Enter a new file name. After the model is named and saved, clicking this icon automatically saves your changes to the current model without displaying the dialog box. This will also save the log if the log window is open.

Cut

Clicking the **Cut** icon removes icons from your model. Element(s) must be selected to activate the icon. Cutting an element will also cut associated relationships. You can cut multiple selected items.

Copy

Clicking the **Copy** icon copies an element to a new location on the same model, or to a new model, without affecting the original model.

Paste

Clicking the **Paste** icon pastes a previously cut or copied element to the clipboard onto another location.

Print Diagrams

Clicking the **Print** icon prints diagrams to the default printer.

Context Sensitive Help

Clicking the **Context Sensitive Help** icon makes all topics covered in the on-line help material available. Click on this icon, drag to the item and release the mouse.

View Documentation

Clicking the **View Documentation** icon displays the documentation window on the diagram.

Browse Class Diagram

Clicking the **Browse Class Diagram** icon opens the **Select Class Diagram** dialog box.

Browse Interaction Diagram

Clicking the **Browse Interaction Diagram** icon opens the **Select Interaction Diagram** dialog box.

Browse Component Diagram

Clicking the **Browse Component Diagram** icon opens the **Select Component Diagram** dialog box.

Browse State Machine Diagram

Clicking the **Browse State Machine Diagram** icon opens the **Select Statechart Diagram** or **Activity Diagram** dialog box.

Browse Deployment Diagram

Clicking the **Browse Deployment Diagram** icon opens the **Deployment Diagram** dialog box.

Browse Use-Case Diagram

Clicking the **Browse Use-Case Diagram** icon opens the **Selected Use Case Diagram** dialog box.

Browse Parent

Clicking the **Browse Parent** icon displays the “parent” of the selected diagram or specification. If you have a specification selected, the specification for the parent of the “named” item is displayed.

Browse Previous Diagram

Clicking the **Browse Previous Diagram** icon displays the last displayed diagram.

Zoom In

Clicking the **Zoom In** icon magnifies the current diagram to view an area in detail.

Zoom Out

Clicking the **Zoom Out** icon minimizes the current diagram allowing you to “pull back” to view more information.

Fit in Window

Clicking the **Fit In Window** icon centers and displays any diagram within the limits of the window. This command changes the zoom factor so that the entire diagram shows.

Undo Fit in Window

Clicking the **Undo Fit In Window** icon undoes the actions performed on the previous Fit In Window command.

List Help Topics

Clicking the **List Help Topics** icon brings up the online help contents.

Toolbox

The diagram toolbox consists of tools that are appropriate for the current diagram. Changing diagrams automatically displays the appropriate toolbox.

When a modifiable diagram window is active, a toolbox with tools appropriate for the current diagram is displayed. If the current diagram is contained by a controlled unit or the model is write-protected, the toolbox is not displayed.

While each diagram has a set of tools applicable for the current diagram, all toolboxes have the following three icons:

- Selector Icon
- Separator Icon
- Lock Icon

Selector Icon

The selector icon is used to select icons on the diagram. This icon cannot be removed from the toolbox.

Separator Icon

The separator icon is used to put a small space between icons on the toolbox. You can have as many as you want, but one must always remain on the toolbox.

Lock Icon

This icon can be set to locked or unlocked. In the locked mode, any tool icon stays in the selected state until the diagram loses focus or another tool button is selected. This option facilitates the rapid placement of several identical icons without repeatedly returning to the diagram toolbox.

This icon is usually not displayed, but you can add it to the toolbox. See *Customizing the Toolbox* below.

You can obtain the lock functionality without the icon through the shortcut menu or by pressing the SHIFT key while placing an element. Releasing the SHIFT deactivates the lock feature.

The toolbox for each diagram type is discussed in the appropriate chapter.

Note: *You can also extend the toolbox. This allows you to view stereotype icons and additional tools if applicable. See the Stereotype chapter for more details.*

Customizing the Toolbox

You can access the **Customize Toolbar** dialog box to modify the displayed toolbox using any of the following methods:

- Right-click anywhere on the toolbox and then click **Customize** from the shortcut menu.
- Double-click anywhere on the toolbox not occupied by a button.
- Click **View > Toolbars > Configure**.
- Click **Tools > Options**. On the **Option** dialog box, click **Toolbars**. This approach gives you the ability to modify all the diagram toolboxes without first displaying a specific diagram type.

Browser

The browser is a hierarchical navigational tool that allows you to view the names and icons of interaction, class, use case, statechart, activity, and deployment diagrams as well as many other model elements.

When a class or interface is assigned to a component, the browser displays the assigned component name in an extended name. The extended name is a comma-separated list within parenthesis to the right of the class and interface name. The extended list includes all the assigned components.

For additional information on the workings of the browser, refer to the chapter entitled *The Browser*.

Documentation Window

The documentation window is used to describe model elements or relationships. The description can include such information as the roles, keys, constraints, purpose, and essential behavior of the element. You can type information in this free-form text either here or through the documentation field of a specification.

To view the documentation window, click **View > Documentation**. A check mark next to documentation indicates the window is opened.

Only one documentation window can be open at one time, but as you select different items, the window will be updated accordingly.

When the window is first displayed, it will be docked to the lower left corner. To move the window, click and drag on the border. The window outline indicates the window state: a thin, crisp line indicates the window will be docked, while a thicker, hashmark-type border indicates it will be floating.

Characteristics unique to the window state (docked or floating) are discussed below:

Docked

- The window can be moved within the dockable region of the model, but it remains positioned along the border.
- The size remains fixed.
- The title can be displayed through a tool tip (simply place your pointer anywhere in the window).
- The window may be docked at any time.

Floating

- The window can be moved to any location, and is always displayed on top of the diagram.
- Size can be changed via click and drag along the border in a vertical or horizontal direction.

The window title displays the type (class or object) and the name of the class or object.

Diagram Window

Diagram windows allow you to create and modify graphical views of the current model. Each icon in a diagram represents an element in the model. Since diagrams are used to illustrate multiple views of a model, each model element can appear in none, one or several of a model's diagrams. This means you can control which elements and properties appear on each diagram.


Diagrams are contained by the model elements they represent:

- A logical package (also User Services, Business Services, and Data Services) contains an automatically created class diagram called “Package Overview,” and user created class diagrams, collaboration diagrams, interaction diagrams, and three-tiered diagrams.
- A component package contains component diagrams.
- A class contains its state diagrams.
- A model contains the diagram for its top level components, its three-tiered service model diagram, its deployment diagram, and the diagram contained by its logical package and component packages. These top-level components can be classes, components, devices, connections, and processors.

Overview Window

The overview window is a navigational tool that helps you move to any location on all Rational Rose diagrams. When a diagram is larger than the viewable area within the diagram window, it is not possible to see the whole diagram without scrolling. The overview window provides a scaled-down view of the current diagram so you can see the entire diagram.

To move to an exact area of your diagram, use the following steps:

1. Move the pointer over the hand  located in the lower, right-side of the diagram window. Notice how the pointer appears as a + when the pointer is located over the active hand.
2. Click on the hand icon so the overview window appears.
3. Hold down the mouse button and move the box inside the overview window to a desired diagram location.

Note: *The overview window closes automatically when you release the mouse button.*

Specification Window

A specification enables you to display and modify the properties and relationships of a model element, such as a class, a relationship, an operation, or an activity. The information in a specification is presented textually; some of this information can also be displayed inside icons representing the model element in diagrams.

You can change properties or relationships by editing the specification or modifying the icon on the diagram. The associated diagram or specification is automatically updated.

To display a specification, click the icon in either the diagram or the browser and use one of the following methods:

- Right-click to display the shortcut menu.
- Click **Browse > Specification**.
- Double-click on the icon (if you have not executed the **Double-Click to Diagram** command).

The specifications are displayed as tabs and you can easily navigate through them.

Printing Diagrams and Specifications

The **Print** dialog box allows you to print diagrams and specifications. The **Print** dialog box has the following tabs:

Table 1 *Print Dialog Box Tabs*

General tab	Allows you to specify a printer, a selection of diagrams and specifications, and the number of copies to be printed.
Diagrams tab	Allows you to select and view a list of diagrams to be printed.
Specifications tab	Allows you to select and view a list of specifications to be printed.
Layout tab	Allows you to select layout settings for printing diagrams and specifications.

Print Preview

The print preview option enables you to see how a diagram will appear when printed. Also, print preview displays the total number of pages the diagram will take to print on the status bar.

Zoom In and Zoom Out

Click either **Zoom In** or **Zoom Out** to view a diagram at different magnified sizes. Also, you can click on any part of the diagram to get a magnified view.

Print

Click the **Print** button to display the **Print** dialog box.

One Page and Two Page

Click **Two Page** to display the diagram in two pages or click **One Page** to view the diagram in one page. When diagrams are viewed in two pages, the **Next Page button** becomes active and enables you to view other pages. The **Previous Page button** becomes active when there is a previous page to view.

Close

Click **Close** to return to an active window.

Apply Filter Dialog Box

The **Apply Filter dialog box** enables you to search for diagrams and specifications within your model. The filter is especially useful when you print diagrams from large models.

To print a specific diagram in a model, type in the name, type, or path of the diagram you are trying to print.

Name—Provides a list of all diagram names depending on search criteria.

Type—Provides a list of all diagram types depending on search criteria.

Path—Provides a list of each path for diagrams displayed.

Next, press the **OK** button to locate the diagram. Then, with the diagram selected, press **OK** from the **Print** dialog box to print the diagram.

To search for a diagram or a specification in the Apply Filter dialog box, you can use the * (asterisk) wildcard character:

- A* matches any name beginning with the letter A
- *A matches any name ending with the letter A
- *A* matches any name containing the letter A

Saving in Various Formats

If you want to save a Rational Rose model as a different format, you may select any of the following options from the **Save As Type** list in the **Save Model To** dialog box:

- Models *.mdl (the current version of Rose)
- Petal *.ptl
- Rose 6.1/6.5 Model
- Rose 4.5/6.5 Model
- Rose 4.0 Model
- Rose 3.0 Model

If you prefer, you can modify the rose.ini file to always save in a specified format, eliminating the need to select **Save As**.

Modifying the Rose.ini File

Use the following steps to modify the rose.ini file:

1. Make sure that Rational Rose is not running before you modify the rose.ini file.

Note: *You will lose all changes to the rose.ini file if Rational Rose is running while you make your changes.*

2. Open the rose.ini file in a text editor. To find the location of the Rose.ini file, look at the Rose.ini Settings (Overview) topic in the online help.
3. Make your changes to the rose.ini file.

4. Save and close the rose.ini file.
5. Restart Rose.

Deleting Model Elements

There are two ways to delete model elements in Rational Rose: shallow delete and deep delete. A shallow delete removes the view of the model elements from a diagram. A deep delete removes model elements from a model completely.

Shallow Delete

A shallow delete is useful when you want to remove a model element from a diagram but keep the model element in the model. A shallow delete keeps the model element in the browser and removes the view of the element from the diagram.

Use one of the following commands to perform a shallow delete on selected model element(s) that appear on a diagram:

- Click **Edit > Delete**
- Press **CTRL + X**
- Press the **DELETE** key

***Note:** If you perform a shallow delete on an element without a name, Rational Rose will delete the model element completely out of the model.*

Deep Delete

A deep delete is useful when you want to remove a model element completely out of a model.

Use one of the following commands to perform a deep delete on selected diagram model element(s):

- Click **Edit > Delete** from Model
- Press **CTRL + D**
- Right-click on an element in the browser and then select **Delete** from the shortcut menu



Chapter 3

The Browser

Overview

The browser is an easy-to-use alternative to menus and toolbars for visualizing, navigating and manipulating items within your model. The browser provides:

- A hierarchical view of many items in a model
- Drag-and-drop capabilities that change a model's characteristics
- Automatic updating of model items to reflect changes in the browser

Viewing the Browser

When you start Rational Rose, the browser is visible by default. It appears in docked position, to the left of the toolbox and diagram windows.

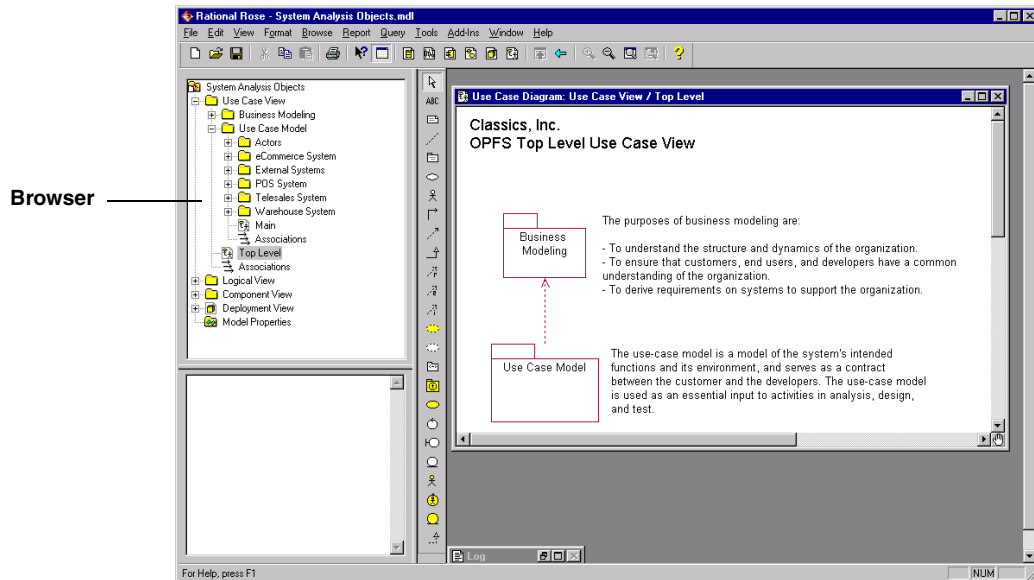


Figure 3 Application Window

Hiding and Displaying the Browser

To hide (or display) the browser window, on the **View** menu, click **Browser**. A check mark next to the word Browser indicates the browser is visible.

Positioning the Browser

You can change the size and position of the browser according to your own preferences. The browser can be:

- **Docked:** Positioned along the border with a fixed size
- **Floating:** Moved to any location with a variable size

Docking and Undocking the Browser

The browser is in a docked position by default.

To redock the browser:

1. Click on any border of the browser.
2. Drag the browser to any application window border.

To undock the browser:

1. Click on any border of the browser.
2. Drag the browser to the desired position.
3. Resize the browser window, if necessary.

Note: As with any resizable window, you can resize the browser by pointing to a border and dragging the pointer to increase or decrease the window's dimensions.

Expanding and Collapsing the Browser Tree

The current model's hierarchy is visible in the tree structure of the browser window:

- A plus (+) sign next to an icon indicates that the icon is collapsed, that is, it contains other model elements. Click the + sign to expand the icon and view its subordinate items.
- A minus (-) sign next to an icon indicates that the icon is fully expanded. Click the minus (-) sign to collapse the item.

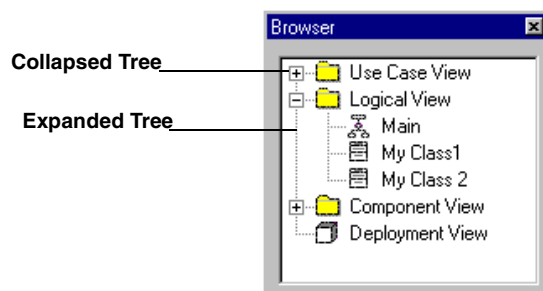


Figure 4 Browser—Collapsed and Expanded Tree

Selecting Multiple Elements in the Browser

You can select multiple elements in the browser to manipulate items within your model for version control purposes. Version control functionality is available through the Version Control add-in or through ClearCase. Selecting multiple elements in the browser enables you to check in or check out more than one file at a time using a version control system. When multiple icons are selected, only the browser options are available on the shortcut menu.

Note: *Add-ins have the ability to modify shortcut menus.*

Use the following steps to select items in the browser:

Selecting Multiple Items in any Order

1. Select an item in the browser.
2. Hold down the CTRL key.
3. Click each item in the browser that you want to select.

Note: *You can deselect an item by pressing the CTRL key.*

Selecting Sequential Items

1. Select an item in the browser.
2. Hold down the SHIFT key.
3. Select another item in the browser.

Notice how the browser selects every item between the two items that you selected.

Navigating a Model

The browser provides a visual representation of your model's hierarchy. As you make changes in a diagram window or in the browser window, the windows remain synchronized:

- To display a diagram window, double-click on its name or icon in the browser window.
- To display an item's specification, double-click on the item in the browser or in a diagram window. (Any changes you make to the specification are automatically reflected in both the browser and the diagram).

- To focus an item in the current diagram, click the item in the browser or in the diagram window.

The following figure shows **My Class1** highlighted in both the browser and the class diagram:

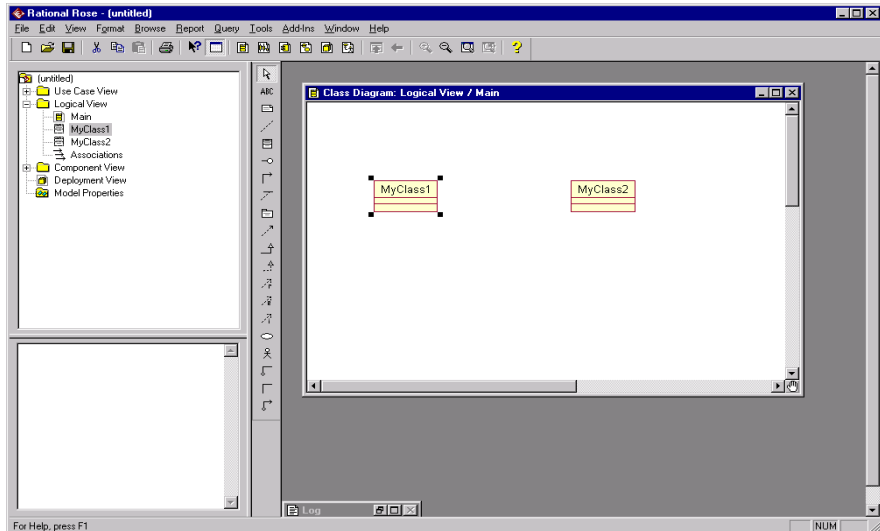


Figure 5 Navigating a Model

Creating and Editing Model Elements

You can use the drag-and-drop capabilities in the browser to create and edit model elements in two ways:

- Drag-and-drop one item in the browser to another item in the browser. Your diagram will automatically be updated to reflect the changes in the browser.
- Drag-and-drop elements from the browser to the appropriate diagrams.
- If the class belongs to a parent different from the diagram, and **Show Visibility** is on, the class is annotated with the term '(from x)' where x is the class' location. If **Show Visibility** is off, only the class name is displayed.

Naming an Element in the Browser

1. Create or select an element.
2. Type in a new name.

If this name already exists in another package, a warning dialog is displayed telling you that the name of the element and type already exist in another package. For example: “Class AA now exists in multiple name spaces.”

You can dismiss this box by either clicking **Cancel** which ignores the name or **OK**. If you do not want to see this dialog box any more, select “Don’t warn anymore this session” button. If you want to start seeing this warning again, you must restart the application.

Using Drag-and-Drop in the Browser

The drag-and-drop feature allows moving elements within the browser and from the browser to diagrams and specifications.

Specifically, you can use drag-and-drop to do the following tasks:

- Assign classes and interfaces to components
- Move class operations and attributes between classes
- Move class diagrams, sequence and collaboration diagrams between packages
- Move component diagrams between component packages
- Move nested classes from one specification to another
- Place components and component packages on component diagrams
- Place classes interfaces and component packages on class diagrams
- Place objects, class instances (and class assignments) on interaction diagrams
- Relocate components and component packages between component packages
- Relocate classes, nested classes, use cases, interfaces, associations and packages between packages
- Place activity diagram model elements on an activity diagram

Note: *You cannot re-order elements on the browser.*

Browser to Browser Capabilities

The following table lists the actions you can perform by dragging-and-dropping objects within the browser:

Table 2 Browser to Browser Capabilities

Capability	Description
Add	<ul style="list-style-type: none"> ■ Class to class diagram ■ Logical package to class diagram ■ Component to component diagram ■ Component package to component diagram
Assign	<ul style="list-style-type: none"> ■ Component to class and interface ■ Class and interface to component ■ Logical package to component package
Move	<ul style="list-style-type: none"> ■ Class diagram to logical package ■ Interaction diagram to logical package ■ Collaboration diagram to logical package ■ Component diagram to component package ■ State/Activity model to the logical or use-case view ■ Process to processor ■ Activities and states to different state machines
Move/Copy*	<ul style="list-style-type: none"> ■ Operation to class and interface ■ Class attribute to class and interface
Relocate	<ul style="list-style-type: none"> ■ Class and interface to logical package ■ Class to nested class ■ Logical package to logical package ■ Component to component package ■ Component package to component package ■ Use case to package

* The default action is *Move*. To *Copy*, hold down the CTRL key while dragging the element to its destination.

Browser to Diagram Capabilities

The following table lists the actions you can perform by dragging-and-dropping elements from the browser to diagrams:

Table 3 Browser to Diagram Capabilities

Capability	Description
Add	<ul style="list-style-type: none"> ■ Class and interface to class diagram ■ Logical package to class diagram ■ Component to component diagram ■ Component package to component diagram ■ Processor to deployment diagram ■ Device to deployment diagram ■ Add activities and objects to activity diagrams
Assign	<ul style="list-style-type: none"> ■ Component to class and interface ■ Class and interface to component ■ Component package to package ■ Logical package to component package
Move/Copy*	<ul style="list-style-type: none"> ■ Operation to class and interface ■ Class attribute to class and interface
Relocate	<ul style="list-style-type: none"> ■ Class to logical package ■ Logical package to logical package ■ Component to component package ■ Component package to component package
Create Object	<ul style="list-style-type: none"> ■ Class in interaction diagram ■ Class in collaboration diagram

* The default action is *Move*. To *Copy*, hold down the CTRL key while dragging the element to its destination.

Browser to Specification Capabilities

The following table lists the actions you can perform by dragging and dropping model elements from the browser to a specification:

Table 4 Browser to Specification Capabilities

Capability	Description
Assign	<ul style="list-style-type: none"> ■ Class and interface to/from Component Specification Realizes tab ■ Component to Class Specification Components tab
Move/Copy	<ul style="list-style-type: none"> ■ Operations to/from Class Specification Operations tab ■ Attributes to/from Class Specification Attribute tab

Sorting Packages in the Browser

Use the following steps to sort packages in the browser:

1. Create a new package in the browser and name it **Temp**.
2. In the **browser**, drag all of the packages you want to sort and drop them into the **Temp** package.
3. In the browser, retrieve the packages one by one from the **Temp** package and place them back in the original location.
4. Delete the **Temp** package.



Chapter 4

Introduction to Diagrams

Overview

Diagrams are views of the information contained in a model. Rational Rose automatically maintains consistency between the diagram and their specifications. You can change properties or relationships by editing the specification or modifying the icon on the diagram. The associated diagrams or specifications are automatically updated.

Diagram Windows

In a diagram window, you can create and modify graphical views of the model. Rational Rose supports the following kinds of diagrams:

- Class Diagram
- Use-Case Diagram
- Collaboration Diagram
- Sequence Diagram
- Component Diagram
- Statechart Diagram
- Deployment Diagram
- Activity Diagram

Each icon on a diagram represents an element in the model. Since diagrams illustrate multiple views of a model, each model element can appear in none, one, or several of a model's diagrams. You can control which elements and properties appear on each diagram.

To create or add icons to a diagram, click **Tools > Create** and click one of the model elements. Click on the diagram to place the element.

Viewing Diagrams

When a diagram is opened, it is displayed in a window within the application window. This diagram window has its own control-menu box, title bar, minimize button, and maximize button. Each diagram window also has vertical and horizontal scroll bars for panning across diagrams larger than the window. The application window presents a toolbox that contains tools appropriate for the current diagram.

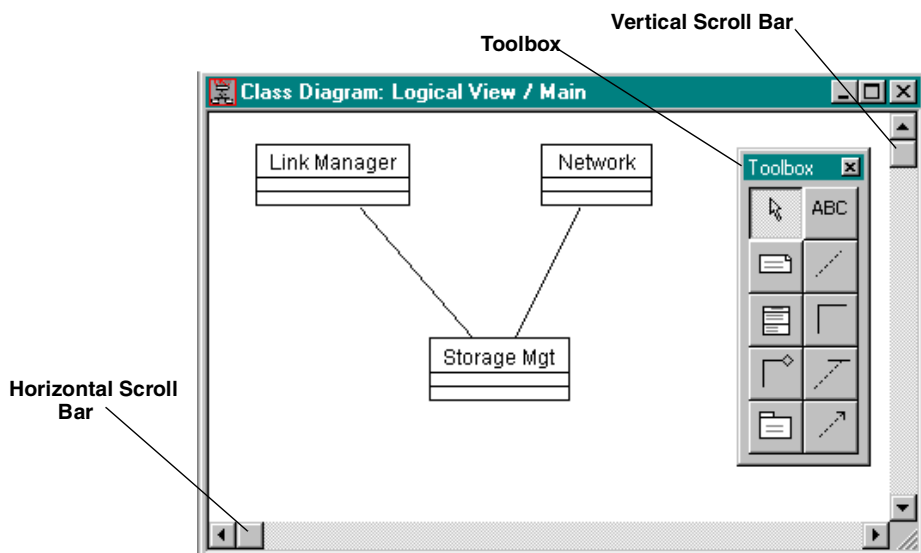


Figure 6 *Diagram Window*

You can resize a diagram window by using the left mouse button to drag a side or corner of the diagram's border. You can reduce a diagram to an icon by clicking on its minimize button.

Displaying Multiple Diagrams

You can display multiple diagrams simultaneously in the application window.

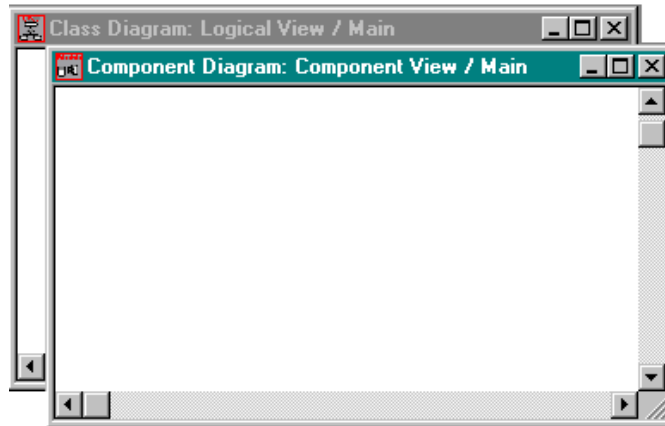


Figure 7 Multiple Diagrams—Cascade Windows

The shaded title bar indicates that it is the current diagram. Diagram-specific commands apply to the current diagram and the application window displays the toolbox associated with the current diagram. Menu commands and toolbox icons not appropriate for the current diagram are “dimmed” and cannot be used. You can make a diagram “current” by clicking on it.

To display multiple diagrams in cascaded windows, as shown in Figure 7, or as tiled windows in Figure 8, click **Window > Cascade** or **Tile**.

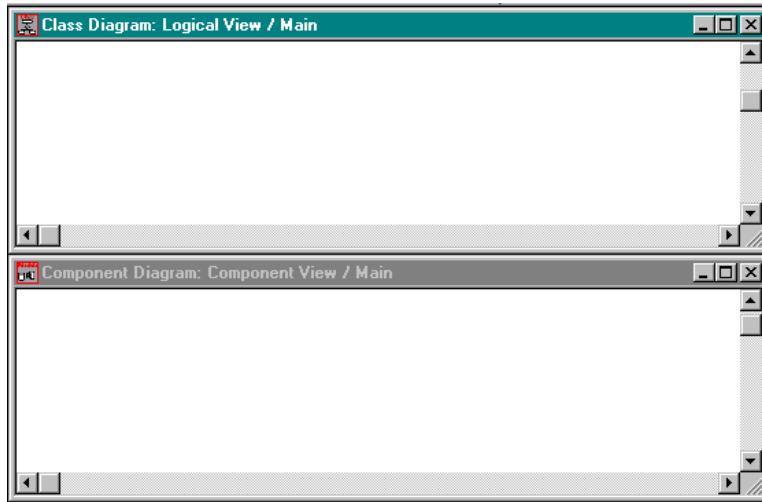


Figure 8 Multiple Diagrams—Tiled Windows

Creating, Linking, Displaying, Renaming, and Deleting Diagrams

Create a New Diagram

1. Click **Browse > xxx Diagram**, where xxx is the diagram type. (If you select **Deployment Diagram**, the diagram is immediately displayed and the following steps can be ignored.)
2. In the resulting dialog box, select a view from the list on the left.
3. Click **<New>** from the list on the right. (If you are creating a new interaction diagram, you must click either **Sequence** or **Collaboration** from the **New Interaction** dialog box.)
4. Click **OK**.
5. Type the diagram title. If you do not enter a title, the diagram is labeled untitled.
6. Click **OK**.

Linking a Diagram

You can link one diagram to another diagram through the note icon. This feature works somewhat like the shortcut method you may be familiar with in the Windows operating environment. Once the diagram is linked, you can double-click on the note and the linked diagram is immediately displayed. A linked diagram is indicated by underlined text in the note.

1. Create a note on any diagram.
2. Display the browser if not already visible.
3. In the browser, locate the diagram that you want to link.
4. Drag the diagram icon from the browser onto the note icon on the diagram.
 - As you position the cursor onto the note, you will see the shortcut symbol (a dotted square and a curved arrow inside a solid square).
5. The fully qualified name is displayed in an underline font.

Note: *You may need to resize the note to see the entire name.*
6. Change the text in the note (if desired) to something more meaningful to your project.
7. Double-click on the note to view the linked diagram.

Display a Diagram

1. Click **Browse > xxx Diagram**, where xxx is the diagram type. (If you select **Deployment Diagram** the diagram is immediately displayed and the following steps can be ignored.)
2. In the resulting dialog box, select an element from the list on the left.
3. Select a diagram from the list on the right.
4. Click **OK**.

Rename a Diagram

1. Click **Browse > xxx Diagram**, where xxx is the diagram type. (**Note:** you cannot rename a deployment diagram.)
2. In the resulting dialog box, select the package containing the diagram from the list on the left.
3. Select the diagram from the list on the right.
4. Click **Rename**.
5. Type a new diagram title.
6. Click **OK**.

Delete a Diagram

1. Click **Browse > xxx Diagram**, where xxx is the diagram type.
2. In the resulting dialog box, select the package containing the diagram from the list on the left.
3. Select the diagram from the list on the right.
4. Click **Delete**.
5. Click **Yes** on the confirmation box.

Selecting Multiple Elements in the Diagram

You can select multiple elements in the diagram by using the following steps:

1. Select an elements in a diagram.
2. Hold down the CTRL key.
3. Click each element in the diagram that you want to select. Notice the squares on each corner of the elements that indicate a selected element.

The diagram example below shows multiple elements selected in a diagram:

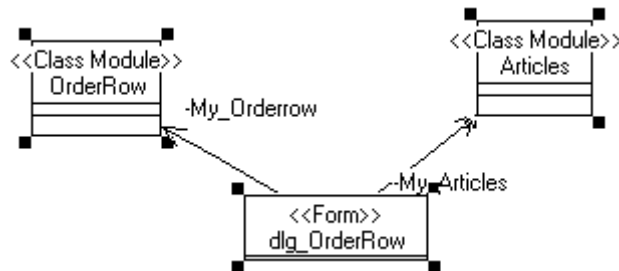


Figure 9 Selected Elements in a Diagram

Note: You can select any element in the diagram.

Creating and Naming Model Elements

Create an Element on the Diagram

1. Click on the appropriate creation tool.
2. Click on a location in the diagram.

Rational Rose creates a model element of the appropriate kind and places an icon representing this element on the diagram and in the browser.

Create an Element in the Browser

1. Click on the appropriate package.
2. From the shortcut menu, click **New** then point to the element you want to create.

The element exists only in the browser until you drag it on a diagram.

Naming Model Elements

You can name your model elements with any combination of characters that are meaningful to you. Depending on the model element and its location, you may or may not be restricted to unique names.

For example, actors, use cases, classes, components, and packages that reside in different packages do not require unique names. When different elements have the same name, the elements are said to be “overloaded.”

Overloading gives you the flexibility of using existing software libraries that may have the exact names you have in your code or in another software library.

Overloading also allows you to do multi-lingual component based development. For example, an application can be modeled even if the GUI for screen input is in VB or Java, the processing is in C++, and the database in Oracle. In this example, each application can have their own definition of a class “Customer” where Customer does different things.

Another useful feature of overloading is the ability to have actors in the use-case view and classes in the logical view with the same name.

When naming an element, it is important to note that in some cases an overloaded element is created, while in other cases, the existing element with the same name is used (and therefore an overloaded element is not created).

Naming an Element on the Diagram

1. Create a new element on the diagram from the toolbox.
2. Type in a name. As soon as you start typing, a pop-up box listing all the available class names in the model is displayed.

You can select one of the highlighted names by double-clicking on a name or by pressing the `ENTER` or `TAB` key. Otherwise, you can continue typing (and click outside the edit area) to enter a new name.

- If you do not want to see this window, you can turn this option off. To do so, click **Tools > Options**. Click on the **Diagram** tab. Under the Miscellaneous section on the lower left, click **Class Name Completion** to turn the feature off.

If the name you select is an overloaded name, clicking outside the box displays a secondary window, asking you to select the name from the fully qualified path.

Creating/Naming an Overloaded Element on the Diagram

If you want to create an overloaded element name on the diagram, you must enter the name through the specification. If you instead enter the duplicate name on the element in the diagram, you will be using an existing element rather than creating a new one with its own characteristics.

1. Create a new element on the diagram from the toolbox.
2. Double-click on the element or click **Browse > Specification**, to display the specification.
3. Type a name in the name field.
4. Click **OK**.

If this name already exists in another package, a warning dialog is displayed telling you the name of the element and type already exists in another package. For example: "Class AA now exists in multiple name spaces."

You can dismiss this box either by clicking **Cancel** which ignores the name or **OK**. If you do not want to see the dialog box anymore, select "Don't warn anymore this session" button. If you want to start seeing the warnings again, you must restart the application.

The element is now named with a duplicate name, but has its own unique characteristics.

Placing an Overloaded Element on the Diagram from the Browser

From the browser, drag the element onto the diagram.

If the element belongs to a parent different from the diagram, and **Show Visibility** is on, the element is annotated with the term '(from x)' where x is the element's location. If **Show Visibility** is off, only the element name is displayed.

Fully Qualified Names

A fully qualified name is displayed as you place your pointer over the model element. A fully qualified name consists of the element hierarchy (starting at the package level), where each level is separated by double colons.

Example: Logical View::Package B::Class 1

Renaming Model Elements

1. Click on the name of a selected icon to display a flashing vertical bar that designates the insertion point.
2. Backspace and type additional text.
3. Note that stereotypes in the form <<stereotype>> are extracted from the name of an item when you edit it.
4. Click outside the named icon.

Alternatively, you can double-click on an icon to display its specification; modify the **Name** field and click **OK**.

If double-clicking on a logical package icon displays the main class diagram, click **Tools > Options** and click the **Diagram** tab. Click on the **Double-Click to Diagram** check box to turn off this option. With this option turned off, double-clicking on a package will display the specification.

Reassigning Model Elements

This feature allows you to make a selected icon represent a different model element.

1. Select the icon to reassign.
2. Click **Edit > Reassign**.

The dialog box lists the packages in the model on the left and a list of the valid elements to choose from on the right. Choose the model element which the selected icon will represent. This operation affects only the selected icon; other icons representing the original model element—on the current diagram and all other diagrams—maintain their original representation. Model elements involved in the operation of this command are themselves unchanged.

Manipulating Icons

Manipulating icons include selecting, deselecting, moving and resizing. These features are similar to those you might find in most major drawing tools.

Selecting Icons (Option 1):

- To select a single icon, left-click on the icon to be selected. Rational Rose displays the icon's selection handles, and deselects all other icons.
- To select several icons, hold the `CONTROL` or `SHIFT` key down and click on each icon to be selected.

Selecting Icons (Option 2):

1. Point near the border of one of the icons to be selected.
2. Left-drag to create a dashed selection box around the icons you want to select.
3. Release the left mouse button.

Rational Rose displays each icon's selection handles, and deselects all other icons.

Deselecting Icons:

- To deselect all icons, click in any open area of the diagram.
- To deselect a specific icon:
 1. Press and hold the `CONTROL` or `SHIFT` key.
 2. Click on the icon.

Any other icons that were previously selected remain selected.

To resize an icon:

1. Click on the icon to be resized.
2. Choose the appropriate selection handle and left-drag to the new dimension.

Rational Rose redraws the icon at the new size, preserving its proportions. To change the proportions of an icon, press the `CTRL` key while resizing it.

To move one or more icons (Option 1):

1. Select the icon(s).
2. Left-drag to the desired location.
3. Release the left mouse button.

To move one or more icons (Option 2):

1. Select the icon(s).
2. Use the four directional arrow keys to move the icons by one pixel in the indicated direction, or press the `CTRL` key while using the arrow keys to move eight pixels in the indicated direction.

If the snap-to-grid operation is enabled, icons and text boxes that are created or moved will be aligned with the nearest grid coordinate. To enable or disable this operation, in the **Options** dialog box, click **Snap To Grid**. To specify the size of the grid in pixels, on the **Options** dialog box, click **Grid Size**.

To change from one kind of element or relationship to another:

1. Click on the toolbox tool bearing the desired icon.
2. Press and hold the `ALT` or `META` key.
3. Click on the icon to be changed.

Rational Rose redraws the icon and updates the model to reflect the change, or reports an error if the change is not legal.

Deleting, Cutting, Copying, and Pasting Icons

You can delete cut, copy, and paste icons between different diagram windows using commands on the **Edit** menu or the tools on the toolbar.

Delete

Rational Rose has two distinct delete functions. Use:

- **Delete** to remove the selected icon from the current diagram with no change to the model element it represents.
- **Delete from Model** to remove the represented element from the model, along with any icon representing it on any diagram.

On the **Edit** menu, clicking **Cut** performs a delete operation on some diagrams and a delete from model operation on others.

Cut, Copy, and Paste

Clicking **Cut**, **Copy**, or **Paste** can manipulate selections containing icons and text in diagrams, and text information in specification fields.

Clicking **Copy** will copy the selected icons to the platform clipboard. Clicking **Cut** performs this same operation and then performs a delete operation. You can use these commands to move some or all of a class diagram to other tools that support the platform clipboard.

Clicking **Paste** in a class diagram adds icons from the clipboard to the center of the current diagram as if you manually created them with the toolbox.

Other Menu Commands

On the **Edit** menu, clicking **Undo** reverses the last **Delete**, **Delete From Model**, or **Cut**.

The **Edit** menu also provides commands that enable you to **Select All**, **Find**, and **Rename** icons.

The **Browse** menu provides commands to navigate among diagrams, and create, rename, and delete them.

When you click the right mouse button on an icon, Rational Rose displays a shortcut menu. This modifies properties (for icons that represent relationships) or selects properties to be displayed within the icon.

Correlations

Depending on the diagram selected, a correlation can be a relationship, a link, a dependency, a transition or a connection. The word correlation can stand for any of the items previously listed.

Creating Correlations Between Elements

To place a correlation between two or more icons:

1. Click on the relationship's tool in the toolbox.
2. Point to the client icon on the diagram.
3. Press and hold the left mouse button.

4. Drag the pointer to the supplier icon on the diagram.
 - You can create vertices while placing the relation, by releasing the mouse button while still on the diagram. A new vertex is created each time you lift the mouse button.
 - You can modify a vertex by dragging on a selected vertex.
 - Joining an inherits relationship to another inherits relationship will create a tree, rather than a hierarchical structure.
5. Release the mouse button at the supplier element.

Rational Rose inserts and selects the relationship, deselecting any other icons. Moving the relationship or class element(s) automatically adjusts the size or vertices as necessary.

To bend a correlation icon:

1. Point to the section of the icon to introduce or modify a bend.
2. Left-drag the pointer to the new location for that section of the icon.
3. Release the mouse button.

When you release the mouse button, Rational Rose redraws the correlation icon with the new or modified bend. If the modification nearly eliminates a bend, Rational Rose will replace the bend with a straight segment.

To reconnect a correlation icon from one icon to another:

1. Point to the end you want to reconnect.
2. Left-drag to the new icon.
3. Release the left mouse button.

Rational Rose redraws the relationship between the two icons and updates the model to reflect the change, or reports an error if the change is not legal.

To name a newly-created correlation:

1. Click on the icon.
2. Type the name.
3. Click outside the named icon.

To change the name of a correlation:

1. Click on the name to display a flashing vertical bar that designates the insertion point.
2. Backspace and type additional text.
3. Click outside the named icon.

Alternatively, you can change the name in the **Name** field of the specification.

Adorning the Diagrams

You can select which adornments (symbols) to display on the diagram through the shortcut menu. The shortcut menu is displayed by clicking the right mouse button on an icon. You can click on the menu choices to enable and disable them; a check mark indicates that a choice is enabled. You can also adorn your diagram with annotation that you add. This annotation or adornment is typically used as notes to yourself or others about specification features or functions not noted by Rational Rose.

Manipulating Text

To place text in a diagram:

1. To select non-default font parameters, on the **Options** dialog box, click **Font** or **Font Size**.
2. Choose the tool bearing **ABC** from the toolbox.
3. Click on the location in the diagram.

To change the default font parameters:

1. Ensure that nothing is selected by clicking on an empty region of any diagram.
2. To select the desired default font parameters, on the **Options** dialog box, click **Font** or **Font Size**.

The default font parameters apply to all diagrams.

To change the dimensions of the invisible box containing text in a diagram:

1. Click on the text to make the text box's selection handles visible.

2. Left-drag the appropriate selection handle to resize the text box.

To move the invisible box containing text in a diagram: (Option 1):

1. Click on the text.
2. Left-drag the text to the location.
3. Release the left mouse button.

To move the invisible box containing text in a diagram: (Option 2):

1. Click on the text.
2. Use the four arrow keys to move the text box by one pixel in the indicated direction, or press the `CTRL` key while using the arrow keys to move eight pixels in the desired direction.

Understanding Model Workspaces

A model workspace is a snapshot of all currently loaded units and open diagrams. By defining one or more workspaces, you can set up your working environment in Rational Rose and return to that environment each time you are ready to work. When you load the workspace, Rose restores the snapshot by loading the specified controlled units and opening the correct diagrams.

If you are working with large models that are divided into many controlled units, you will notice even greater productivity gains by using workspaces to load predefined units and diagrams.

Differences between a Saved Model and a Model Workspace

A saved Rational Rose model contains the diagrams, elements, and controlled units that make up the complete model. A model workspace contains the actual state of open diagrams and controlled units for a specific saved model at a given point in time.

It is possible to have multiple workspaces corresponding to only one model. For example, during analysis and design, you might want to define one model workspace that displays the most important analysis diagrams and controlled units, and another model workspace that displays the most important design diagrams and controlled units. Each workspace is different but points to the same model.

It is also important to note that saving a model workspace will not affect how the model is loaded on another machine. If a co-worker wants to load a model using a model workspace you defined on your machine, the co-worker must have a copy of the model workspace and model located in the same folder on his or her machine.

By default, Rational Rose will name the workspace **<model name>-<Operating System User Name>.wsp**. For example, the name of a saved model workspace might look like **MyModelName-JillUser.wsp**.

Note: *Rational Rose stores all workspace files (*.wsp) in the workspaces folder.*

Model Workspace Sample

The following sample shows how using model workspaces can benefit a team working on a large model.

A new software developer has just joined a distributed team that is working on a very large model containing over 200 controlled units. Through the course of the next several months, the new developer will model several systems in the Use Case Model and will modify the Business Actors and Use Cases (as shown in browser illustration). In order to help the new developer, the team's project manager created a model workspace that will load all of the units the software developer will be responsible for, as well as some of the more important diagrams.

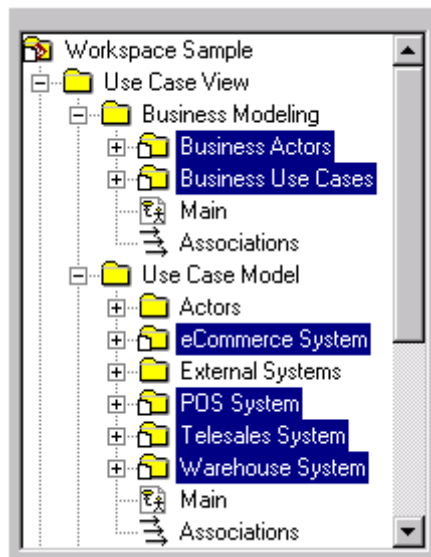


Figure 10 Model Workspace Loaded Units

When the developer loads the model workspace, the **Business Actors**, **Business Use Cases**, **eCommerce System**, **POS System**, **Telesales System**, and **Warehouse System** controlled units all load. The workspace configuration will also display some important class and activity diagrams in the diagram window.

The model workspace will help the new developer by:

- Loading the controlled units that the developer is responsible for automatically
- Displaying some of the more important diagrams the developer should examine first
- Saving the developer time because Rose only has to load six out of 200+ controlled units
- Eliminating confusion by limiting the scope of information the developer sees

After working in the model, the developer can easily customize the model workspace the project manager created, or create additional model workspaces to create efficiency.

Saving a Model Workspace

To save a Model Workspace:

1. Click **File > Save Model Workspace**. Rational Rose will save both the model and workspace files.
2. Name your workspace file in the **Save As** dialog box. By default, Rational Rose will name the workspace <model name>-<Operating System User Name>.wsp. For example, the name of a saved model workspace might look like **MyModelName-JillUser.wsp**.

Note: Rational Rose stores all workspace files (*.wsp) in the *workspaces* folder.

Loading a Model Workspace

To load a model workspace:

1. Click **File > Load Model Workspace**.
2. Select the name of workspace file (*.wsp) to load.
3. Click **Open**.



Chapter 5

Introduction to Specifications

A specification enables you to display and modify the properties and relationships of a model element, such as a class, a relationship, or an operation.

Some of the information displayed in a specification can also be displayed inside icons representing the model element in diagrams.

The specification fields are standard interface elements such as text boxes, list boxes, option buttons, and check boxes.

Displaying Specifications

You can display a specification in the following ways:

- Double-click on an item in a diagram or browser.
- Click a diagram item and then click **Browse > Specification**.
- Click **Open Specification** from the shortcut menu.
- Select the diagram item and press CTRL+B.

Rational Rose displays a specification that corresponds to the selected item.

In order to view a specification when you double-click on a logical or component package, you must turn off the **Double-Click to Diagram** option. To disable this option, click **Tools > Options**. Go to the **Diagram** tab. A check mark inside the **Double-Click to diagram** check box indicates the main diagram will be displayed when you double-click. If there is no check mark in the check box, double-clicking on a logical or component package displays the package specification.

Custom Specifications

When you open the specification of an element that has an assigned language, a custom specification will be displayed if supported. If not supported, the standard Rose specification will be displayed.

The following specifications can be customized by language add-ins:

- Association
- Class
- Class Attribute
- Generalize
- Key/Qualifier
- Parameter
- Operation
- Component
- Class Instance

Editing Specifications

If you change a model element's properties or relationships by editing its specification or modifying the icons on the diagram, Rational Rose will automatically update the corresponding diagrams or specifications.

If a model element is write-protected, or contained by a controlled unit that is write-protected, the **OK** button on the specification will be disabled to prevent the element from being modified.

Specifications can be resized by placing the pointer on a specification corner. Click and drag the specification to the desired size.

Specifications can also be printed by clicking **File > Print**.

Common Specification Elements

The specifications share a number of common elements which are discussed on the following pages. For details on specific specifications and their unique elements, refer to the following chapters.

Dialog Boxes

All specifications are presented in a dialog box format and contain tabs for navigating to specific pages or items. You can resize all specifications.

General Tab

The first tab presented in all specifications is labeled **General** and usually contains information such as **Name** and **Documentation**.



Figure 11 General Tab

Name

Every model element and each relationship can be labeled with a word or phrase that denotes the semantics or purpose of the relationship. You can enter the name in the diagram or in the **Name** field of a specification.

- If you enter the name in the diagram, Rational Rose displays the entry in the **Name** field.
- If you enter the name in the specification, Rational Rose displays the new name in the icon and updates the information in the model.

You can rename an element using one of the following methods:

- Change its name in the diagram or browser.
- Change its name in the specification.

Documentation

Use the **Documentation** field to describe relationships. The description can include such information as the roles, keys, constraints, purpose, and essential behavior of the element. You can enter information in the documentation field in one of two ways:

- Enter text directly in the free form text field.
- Click **View > Documentation**.

Rational Rose does not display this field in the diagram.

Note: *If you document a class and identify the concepts or functions represented by the entity, you can use the field to form a basis of a more traditional data dictionary. You can also list the statements of obligation to provide certain behavior with the class. You can use this entry as a place holder for the responsibilities of the class that you will determine during development.*

Detail Tab

Another common tab is the **Detail** tab. The **Detail** tab contains information specific to the model element you have selected. The **Detail** tabs and their relationship to a diagram element are described in their respective chapters.

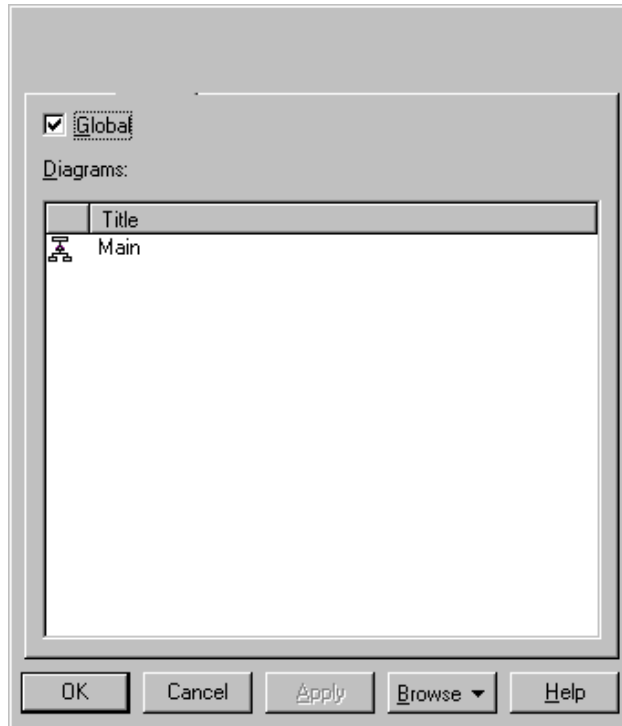


Figure 12 *Detail Tab*

Files Tab

The Files tab allows you to insert new files or URLs or view files and URLs already inserted or attached to your model element or diagram.

The Files tab is useful for maintaining links to supplemental documentation about the system being built (for example, Vision Documents, GUI sketches, project plans, etc.).

Any attached URLs or files listed here are also displayed when the element or diagram is expanded in the browser.

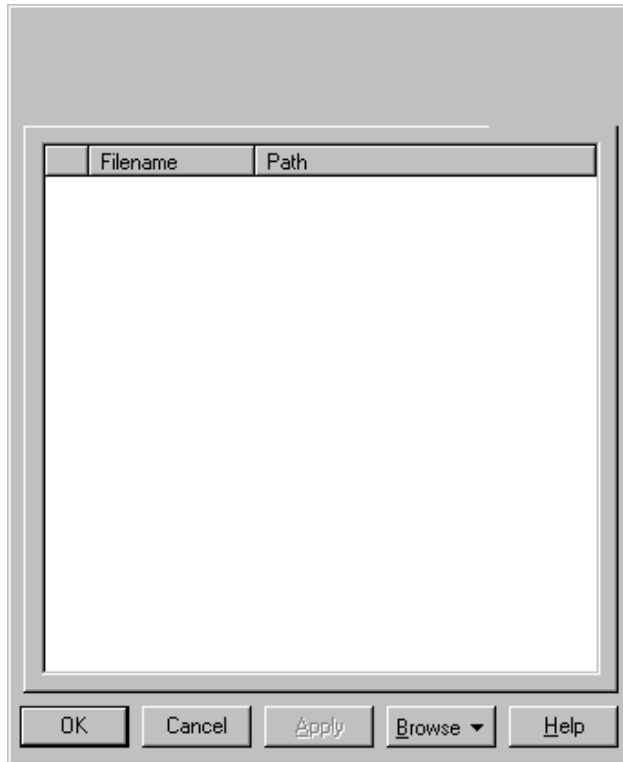


Figure 13 Files Tab

Viewing Existing Files or URLs

If a file is already inserted, the Filename and Path are displayed on the tab. To open the document or go the web site, double-click on either the Filename or Path, or choose **Open File/URL** from the shortcut menu.

Inserting New Files

You can insert (or attach) files by:

- Using the drag-and-drop technique.
- Clicking **Insert File** from the shortcut menu and navigating through the dialog box to locate your file.

Inserting New URLs

Click **Insert URL** from the shortcut menu; this will insert the default address `www.rational.com`. Edit the Filename and Path to point to the correct web site.

Tab Buttons

The bottom of each tab, regardless of type (General, Detail, etc.), contains five buttons to control the actions on each tab.

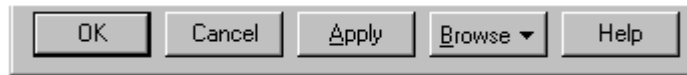


Figure 14 *Tab Buttons*

OK

Clicking **OK** applies the changes made to the specification and closes the dialog box, returning focus to the diagram.

Cancel

Clicking **Cancel** ignores all changes made to the specification since the last **Apply**, closes the dialog box, and returns focus to the diagram.

Apply

Clicking **Apply** enacts the changes made to the specification while leaving the specification open.

Changes to a **Specification** field are not enacted until you click on the specification's **OK** or **Apply**. These will be disabled if the model element is assigned to a controlled unit which is write-protected.

Browse

Clicking **Browse** displays four choices:

- **Select in Browser** which highlights the selected item in the browser.
- **Browse Parent** which opens the specification for the parent of the selected item.
- **Browse Selection** which opens the specification for the currently selected item.

- **Show Usage** which displays a list of all diagrams in which the currently selected element is the supplier, or in the case of a collaboration diagram, a list which shows the usage of a message.

Help

Clicking **Help** invokes the online help topic related to the dialog box.

Navigating the Tabs

Many tabs contain lists of elements related to the specification and are described in detail in their respective chapters. However, the methods used to navigate through the specifications are very similar throughout all tabs.

The lists typically consist of one row per related element. The rows are typically divided into columns, describing aspects of the rows (e.g. **Filename** and **Path** on the **Files** tab). To navigate between rows and columns in the list, either select the row and column with the pointer or use the arrow keys on the keyboard.

Adding and Deleting Entries

To insert a new row in a list, click **Insert** from the shortcut menu or press the INSERT key. An untitled entry is added.

To delete a row, select the row and click **Delete** from the shortcut menu or press the DELETE key.

Editing Entries

To edit a column in a row, select the column and press F8 or select the column twice with the pointer. Enter text into the column or select an entry from the drop-down menu (if available). After the column has been edited, either accept the change (by clicking outside the column, pressing the ENTER or TAB key) or cancel the addition (by pressing the ESC key).

To open the specification for an element displayed in a list, select the row and column and click **Specification** from the shortcut menu or double-click on the column. For example, double-clicking on the **Name**

column in the **Relations** tab of the class specification will open the specification for the relation, while double-clicking on the **End Class** column in the same list will open the specification of the related class.

To reorder the rows in a list, select the row to be moved and drag it to the new location in the list. It is not possible to reorder rows in every list tab. To move an element in a list to another specification, the browser, or to an open diagram, select the row and drag it to the new location.



Chapter 6

Class Diagrams and Specifications

Class Diagram Overview

A class diagram is a picture for describing generic descriptions of possible systems. Class diagrams and collaboration diagrams are alternate representations of object models. Class diagrams contain classes and object diagrams contain objects, but it is possible to mix classes and objects when dealing with various kinds of metadata, so the separation is not rigid.

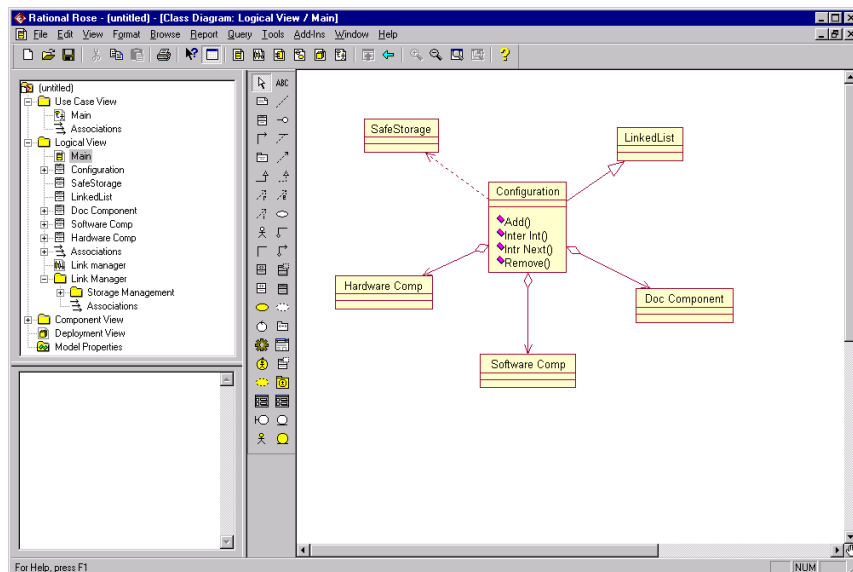


Figure 15 Class Diagram Example

Class diagrams contain icons representing classes, interfaces, and their relationships. You can create one or more class diagrams to depict the classes at the top level of the current model; such class diagrams are themselves contained by the top level of the current model. You can also create one or more class diagrams to depict classes contained by each package in your model; such class diagrams are themselves contained by the package enclosing the classes they depict; the icons representing logical packages and classes in class diagrams.

You can change properties or relationships by editing the specification or modifying the icon on the diagram. The associated diagrams or specifications are automatically updated.

Creating and Displaying a Class Diagram

You can create or display a class diagram in one of three ways:

- Click **Browse > Class Diagram**.
- On the toolbar, click the class diagram icon.
- On the browser, double-click on the class diagram icon.

Class Diagram Toolbox

The graphic below shows all the tools that can be placed on the class diagram toolbox. See “Customizing the Toolbox” on page 14 for more information on adding or deleting tools on a diagram toolbox.

The application window displays the following toolbox when the current window contains a class diagram, you have selected **View > As Unified**, and you have customized the toolbox to display all the tool options.

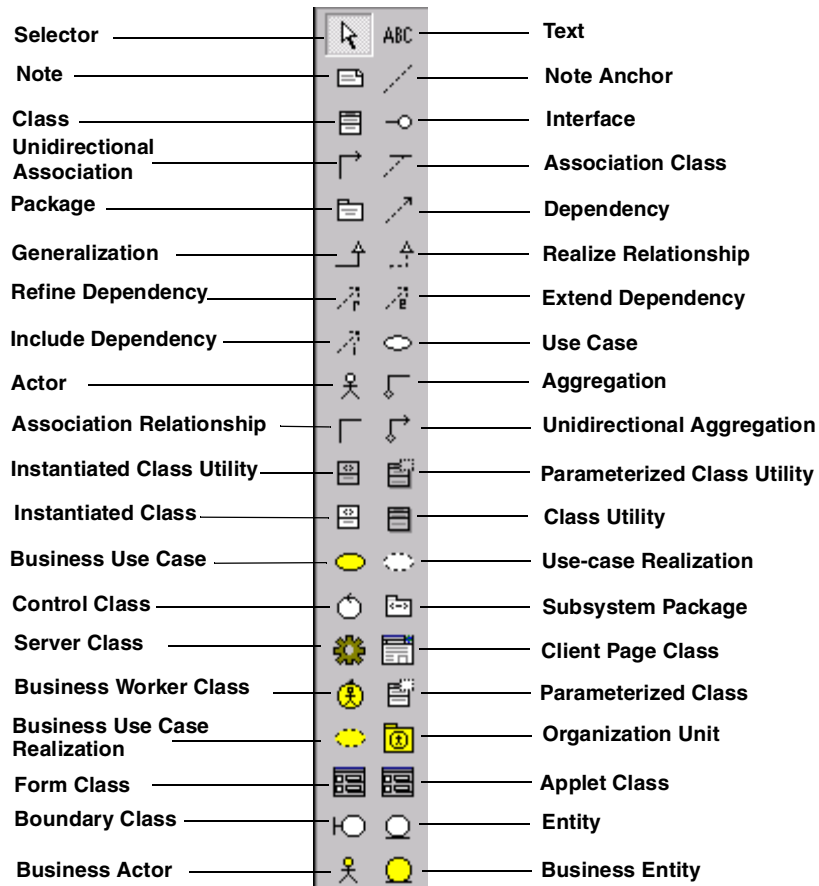


Figure 16 *Class Diagram Toolbox*

Assigning a Class to Another Logical Package

Every class is assigned to a logical package. When you create a class using a creation tool from the class diagram toolbox, the class is assigned to the logical package containing the class diagram. In Figure 15, “Class Diagram Example,” on page 61 the class diagram named *Main* is directly contained by the logical package named *LinkManager*. All of the classes depicted on *Main* are assigned to *LinkManager*, except the class *SafeStorage*. This is assigned to logical package *StorageManagement*. Rational Rose annotates the icon representing *SafeStorage* with the phrase from Storage Management.

To re-assign a class from one logical package to another:

1. Select an icon (or icons) representing the class in a diagram contained by the logical package to which the class should be assigned. (You might need to create such a diagram or icon if one does not currently exist.)
2. Click **Edit > Relocate**.

Rational Rose will update all class diagrams to reflect the new assignment.

Like classes, logical packages are also assigned to logical packages—permitting nesting to an arbitrary depth. Assigning and re-assigning can be applied to logical packages and classes.

Adding and Hiding Classes, and Filtering Class Relationships

The commands on the **Query** menu provide powerful facilities for controlling which model elements are represented by icons in the current diagram.

On the **Query** menu, clicking:

- **Add Classes** adds classes to the diagram by name.
- **Add Use Cases** adds use cases to the diagram by name.
- **Expand Selected Elements** adds classes to the diagram based on their relationships to selected classes.
- **Hide Selected Elements** removes selected classes from the diagram and optionally removes their clients or suppliers from the diagram.
- **Filter Relationships** controls which kinds of relationships appear in the current diagram.

Class Specification

A **Class Specification** displays and modifies class properties and relationships. Some of the information in the specification can also be displayed inside class icons.

If a field does not apply to a particular class type, the field is unavailable and you cannot add or change information in the field.

To display a **Class Specification**, click an icon representing the class in a class diagram and click **Browse > Specification**.

If you have not gone clicked **Tools > Options** and clicked **Double-Click to Diagram**, you can double-click on any icon representing the class. You can also click **Specification** from the shortcut menu.

Specification Content

The **Class Specification** consists of the following tabs: **General**, **Detail**, **Operations**, **Attributes**, **Relations**, **Component**, **Nested**, and **Files**.

Class Specification—General Tab

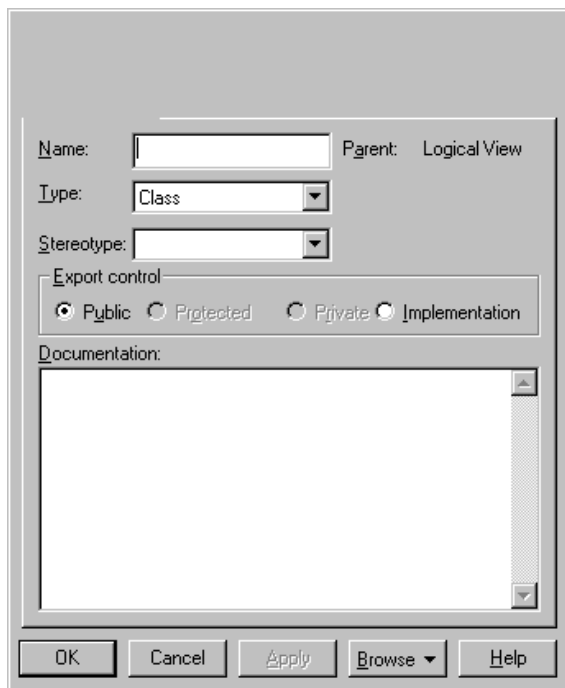


Figure 17 Class Specification—General Tab

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Type

Your **Type** choices include: Class, Parameterized Class, Instantiated Class, Class Utility, Parameterized Class Utility, Instantiated Class Utility, and Metaclass.

Parent

The parent the class belongs to (its package) is displayed in this static field.

Stereotype

A stereotype represents the subclassification of an element. It represents a class within the UML metamodel itself, that is, a type of modeling element. Some stereotypes are already predefined, but you can also define your own to add new kinds of modeling types.

Stereotypes can be shown in the browser and on diagrams. The name of the stereotype may appear in angle brackets <<>>, depending on the settings found in either the **Diagram** or **Browser** tabs of the **Options** dialog box located under the **Tools** menu. Refer to the *Stereotype* chapter for more information on stereotypes.

To show stereotypes on the diagrams, click **Options** from the shortcut menu and click **Stereotype Name** or **Stereotype Icon**. **Stereotype Name** displays the name in angle brackets (ie. <<stereotype>>). **Stereotype Icon** displays the graphical representation.

Export Control

The **Export Control** field specifies how a class and its elements are viewed outside of the defined package.

Table 5 *Export Control Field Options*

Select:	To Indicate:
Public	The element is visible outside of the enclosing package and you can import it to other portions of your model. Operations are accessible to all clients.
Protected	The element is accessible only to subclasses, friends, or the class itself.
Private	The element is accessible only to its friends or to the class itself.
Implementation	The element is visible only in the package in which it is defined. An operation is part of the implementation of the class.

The **Export Control** field can be set only in the specification. No special annotation is related to access control properties.

To change the export control type for the class, click on the appropriate option in the **Export Control** field. You can display the implementation export control in the component compartment. You can display visibility in an icon through the shortcut menu.

Class Specification—Detail Tab

Cardinality: 1

Space:

Persistence

- Persistent
- Transient

Concurrency

- Sequential
- Guarded
- Active
- Synchronous

Abstract

Formal Arguments:

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

OK Cancel Apply Browse Help

Figure 18 Class Specification—Detail Tab

Refer to the description in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification element not covered in the following section.

Cardinality

The **Cardinality** field specifies the number of expected instances of the class. In the case of relationships, this field indicates the number of links between each instance of the client class and the instance of the supplier. You can set a specific cardinality value for the client class, supplier class, or both.

Use the following syntax to express cardinality:

Table 6 Cardinality Field Options

Type	Description
n (default)	Unlimited number of instances
1	One instance only
0..n	Zero or more instances
1..n	One or more instances
0..1	Zero or one instance
<literal>*	Exact number of instances
<literal>..n	Exact number or more instances
<literal>..<literal>	Specified range of instances
<literal>..<literal>,<literal>	The number of instances will be in the specified range or an exact number of instances
<literal>..<literal>, <literal>..<literal>	The number of instances will be in one of the specified ranges

*Where <literal> is any integer greater or equal to one.

To display class cardinality on an icon, right-click on the icon and select a cardinality through the shortcut menu. A literal value can only be specified on the specification.

Space

Use the **Space** field to document the amount of storage required by objects of the class during execution.

Persistence

Persistence defines the lifetime of the instances of a class. A persistent element is expected to have a life span beyond that of the program or one that is shared with other threads of control or other processes. Use this field to identify the persistence for elements of this class:

Table 7 Persistence Field Options

Type	Description
Persistent (Default)	The state of the element transcends the lifetime of the enclosing element.
Transient	The state and lifetime of the element are identical.
Static	The element exists during the entire execution of a program.

The persistence of an element must be compatible with the persistence that you specified for its class. If a class persistence is set to **Persistent**, then the object persistence is either persistent, static or transient. If a class persistence is set to **Transient**, then the object persistence is either static or transient.

You can set the persistence only through the specification. This field is inactive for class utilities, parameterized class utilities, and instantiated class utilities.

To set the persistence, click on the applicable option in the **Persistence** field. You can display the persistence in the diagram by clicking **Show Persistence** from the shortcut menu.

Concurrency

A class concurrency defines its semantics in the presence of multiple threads of control. Choose one of the following options:

Table 8 Class Concurrency Options

Type	Description
Sequential (default)	The semantics of the operation are guaranteed only in the presence of a single thread of control. Only one thread of control can be executing in the method at any one time.
Guarded	The semantics of the operation are guaranteed in the presence of multiple threads of control. A guarded class requires collaboration among client threads to achieve mutual exclusion.
Active	The class has its own tread of control.
Synchronous	The semantics of the operation are guaranteed in the presence of multiple threads of control; mutual exclusion is supplied by the class.

Abstract

The **Abstract** check box identifies a class that serves as a base class. An abstract class defines operations and states that will be inherited by subclasses. This field corresponds to the abstract class adornment displayed inside the class icon.

To toggle the abstract adornment, click on the abstract check box in the class specification.

When you click **Abstract** and you view the model in Booch notation, the abstract class adornment is displayed in the lower left corner of the class icon.

You can change the abstract class adornment only through the specification.

The **Abstract** field is inactive for metaclasses, class utilities, parameterized class utilities, and instantiated class utilities.

Formal Arguments

In the **Parameterized Class** or **Parameterized Class Utility Specification**, the formal, generic parameters declared by the class or class utility are listed.

In the **Instantiated Class** or **Instantiated Class Utility Specification**, the actual arguments that match the generic parameters of the class being instantiated are listed.

You can add, update, or delete parameters only through the **Class Specification**. This field applies only to parameterized classes, parameterized class utilities, instantiated classes, and instantiated class utilities.

To define the parameters for a class, position the pointer within the **Parameters** field and click **Insert** from the shortcut menu or press the **INSERT** key.

Parameters are displayed on class diagrams.

Class Specification—Operations Tab

Operations denote services provided by the class. Operations are methods for accessing and modifying **Class** fields or methods that implement characteristic behaviors of a class.

The **Operations** tab lists the operations that are members of this class. Rational Rose stores operation information in an operation specification. You can access **Operation Specifications** from the **Class Specification** or from the **Browser**.

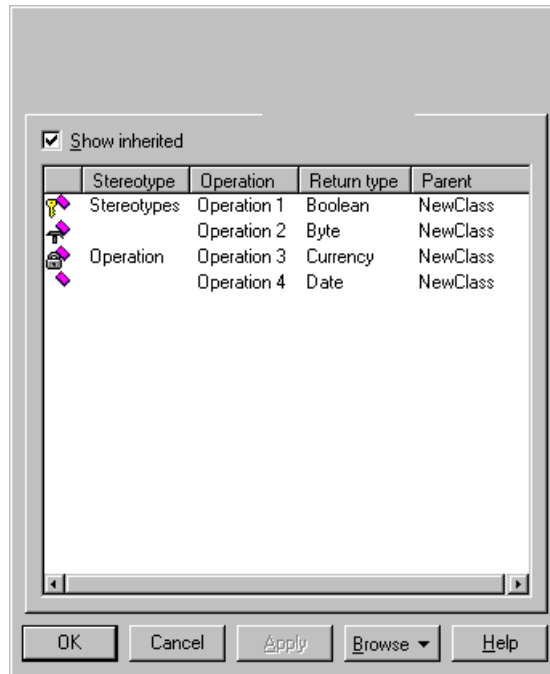






Figure 19 Class Specifications—Operations Tab

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification element not covered in the following section.

To enter an operation in the **Class Specification**, use **Insert** from the shortcut menu. Rational Rose adds the operation name to the operations list.

The descriptions for each field on the **Operations** tab are discussed below:

■ **Access Control Adornment (Unlabeled):**

-  Public—members of a class are accessible to all clients.
-  Protected—members of a class are accessible only to subclasses, friends, or to the class itself.
-  Private—members of a class are accessible only to the class itself or to its friends.
-  Implemented—the class is accessible only by the implementation of the package containing the class.

■ **Stereotype**—displays the name of the stereotype.

■ **Operation**—displays the name of the operation.

■ **Return Type**—identifies the type of value returned from the operation.

■ **Parent**—identifies which class defines the operation.

The **Operation** tab is active for all class types. In the class diagram, you can display operation names in the class compartment.

Show Inherited

Click the **Show Inherited** option to see operations inherited from other classes. If there is no check mark in this field, you can view only operations associated with the selected class.

Class Specification—Attributes Tab

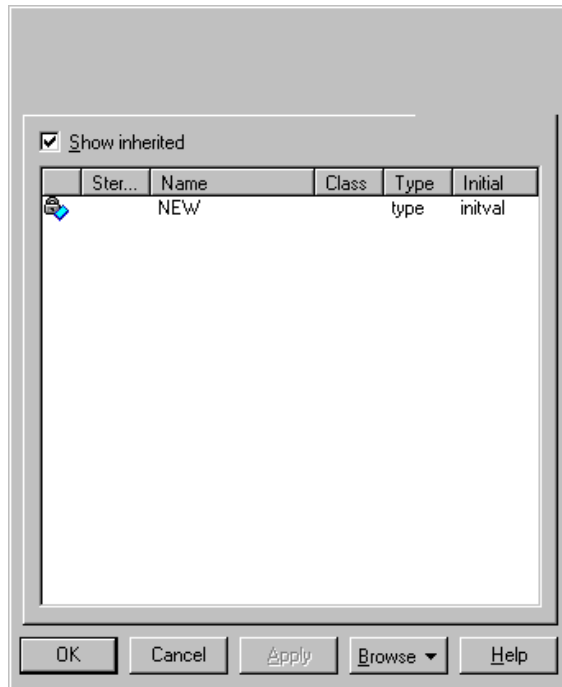


Figure 20 Class Specification—Attributes Tab

Refer to the descriptions earlier in this chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

The Rational Unified Method asserts that attributes are data values (string or integer) held by objects in a class. Thus, the **Attributes** tab lists attributes defined for the class through the **Class Attribute Specification**.

You can add an attribute relationship through **Insert** on the shortcut menu or by pressing the **INSERT** key. An untitled entry is added.

Attributes and relationships created using this technique are added to the model, but do not automatically appear in any diagrams.

The descriptions for each field are discussed below:

■ **Access Control Adornment (Unlabeled):**

- ◆ Public—members of a class are accessible to all clients.
- ◆ Protected—members of a class are accessible only to subclasses, friends, or to the class itself.
- ◆ Private—members of a class are accessible only to the class itself or to its friends.
- ◆ Implemented—the class is accessible only by the implementation of the package containing the class.

- **Stereotype**—displays the name of the stereotype.
- **Name**—displays the name of the attribute.
- **Class**—identifies where the attribute is defined.
- **Type**—this can be a class or a traditional type, such as `int`.
- **Initial**—displays the initial value of an object.

This **Attribute** tab is active for all class types.

Class Specification—Relations Tab

Classes collaborate with other classes in a variety of ways. The **Relations** tab identifies the relationships in which this class is the client (class) and the corresponding supplier (end) class. If you labeled the relationship, Rational Rose displays its name after the kind of relationship.



Figure 21 Class Specification—Relations Tab

Refer to the descriptions earlier in this chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification element not covered in the following section.

Rational Rose automatically updates this list when you draw relationships between classes.

The description for each field is discussed below:

- **Name**—displays the name of the relationship.
- **Parent**—displays the client name.
- **End Class**—displays the supplier name.

Class Specification—Component Tab

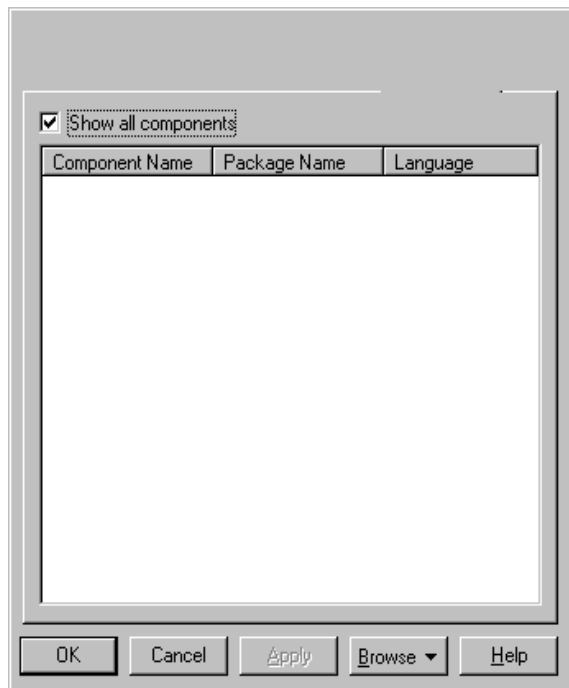


Figure 22 *Class Specification—Component Tab*

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification element not covered in the following section.

Show All Components

Select this option if you want to get a list of all components in a model. If this option is not selected, you will see only the components to which this class is assigned.

Component Name

The component list identifies the components to which this class is assigned (with a check mark). A class can be assigned to a note or to several components with the same implementation language assigned.

You can assign the class to a component through **Assign** on the shortcut menu, or by dragging a component from the browser and dropping it in the list.

Package Name

This field displays the package that the component belongs to.

Language

The **Language** field identifies the implementation language that is assigned to this element.

Note: *When you change the implementation language of a component, the data types that are used in the specification of operations are attributes of the assigned classes, are not automatically converted to data types in the new implementation language. Also if you change the implementation language for a component with classes that are assigned to other components, a dialog box is displayed asking how to handle those classes.*

Class Specification—Nested Tab

A nested class is a class that is enclosed within another class. Classes may contain instances of, inherit from, or use a nested class.

Enclosing classes are referred to as parent classes, and a class that lies underneath the parent class is called a nested class.

A nested class is typically used to implement functionality for the parent class. In many designs, a nested class is closely coupled to the parent class and is often not visible outside of the parent class. For example:

Think of your computer as a parent class and its power supply as a nested class. While the power supply is not visible outside the computer, the task it completes is crucial for the overall functionality of the computer.

Note: *Nested classes can be cut and pasted.*

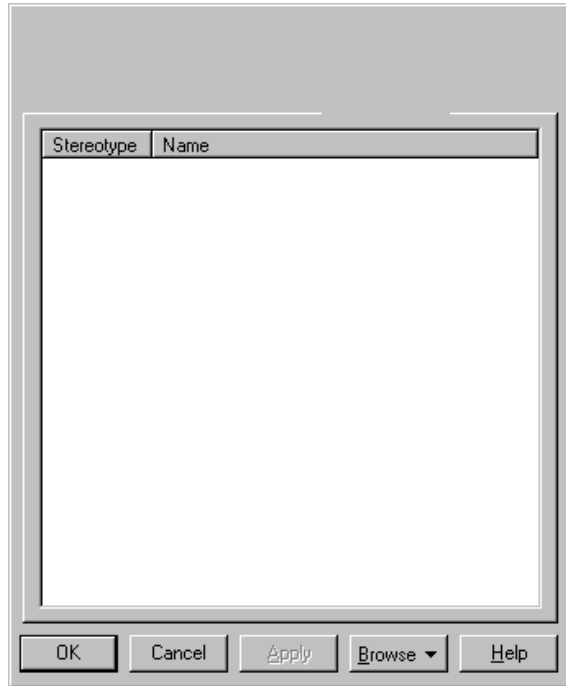


Figure 23 Class Specification—Nested Tab

Refer to the description earlier in this chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Adding a Nested Class from a Class Specification:

1. Create and name a class.
2. Display the **Class Specification**.
3. Click on the **Nested** tab.
4. Right-click to display the shortcut menu, then click **Insert**.

An untitled class entry is inserted. A nested class entry with a default class name is inserted.

To display a nested class:

1. Click **Query > Add Classes**.
2. Select the nested class and place it in the **Selected Classes** list box.

Deleting a Nested Class from a Class Specification

1. Select the nested class from the **Nested** tab in the **Class Specification**.
2. Right-click on the class to display a shortcut menu.
3. From the shortcut menu, click **Delete**.

Or, use the following steps to delete a nested class:

1. Select the name of the nested class from the Nested Classes list box in the Class Specification.
2. Press the DELETE key.

If you delete a nested class that is also a parent to other nested classes, all the nested classes will be deleted.

Note: When you attempt to delete a nested class from a Class Specification, a warning dialog will appear to verify the deletion.

Relocating Nested Classes from the Browser to a Specification

Classes and Nested Classes can be moved from the browser to the Class Specification Nested Tab. If you move a class (NewClassA) from the browser and place it directly on top of a class (NewClassB) on the Nested tab, NewClassA becomes nested underneath NewClassB. However, only one level of class nesting appears on the Nested tab. You can view all levels of nesting in the browser.

For additional information on the browser, refer to Chapter 3, *The Browser*.

Moving Nested Classes between Class Specifications

Nested classes can be dragged and dropped between Class Specification Nested tabs.

Class Specification—Files Tab

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on this tab.

Class Attribute Specification

A **Class Attribute Specification** enables you to display and modify the properties of a class attribute in the current model.

To display an **Attribute Specification**, select the entry on the **Attribute** tab of the **Class Specification** and click **Insert** from the shortcut menu. Alternatively, double-clicking on the entry will display the **Class Attribute Specification**.

Specification Content

The **Class Attribute Specification** consists of the following tabs: **General** and **Detail**.

Class Attribute—General Tab

The screenshot shows a dialog box titled "Class Attribute—General Tab". It contains the following fields and controls:

- Name:** A text box containing "name".
- Class:** A label "Class: Configuration" next to the name field.
- Type:** A dropdown menu.
- Stereotype:** A dropdown menu.
- Initial value:** A text box.
- Export Control:** A group box containing four radio buttons: Public, Protected, Private, and Implementation.
- Documentation:** A large text area with a vertical scrollbar.
- Buttons:** A row of buttons at the bottom: OK, Cancel, Apply, Browse (with a dropdown arrow), and Help.

Figure 24 Class Attribute—General Tab

Refer to the descriptions earlier in this chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Class

The class the attribute belongs to is displayed in this static field.

Show Classes

Select the **Show Classes** check box to list all classes defined in the model and any fundamental types that reside in the model.

If you clear this check box, the selection lists include only the fundamental types that reside in the model.

Type

Attribute types can either be classes or language specific types. When the attribute is a data value, the type is defined as a language specific type. You can enter the type in the **Type** field of the **Class Attribute Specification**. Rational Rose displays the type beside the attribute name in the class icon and updates the information in the model.

Initial Value

You can assign an initial value to your class attribute through this field. Click in the **Initial Value** field and enter the value.

Class Attribute—Detail Tab

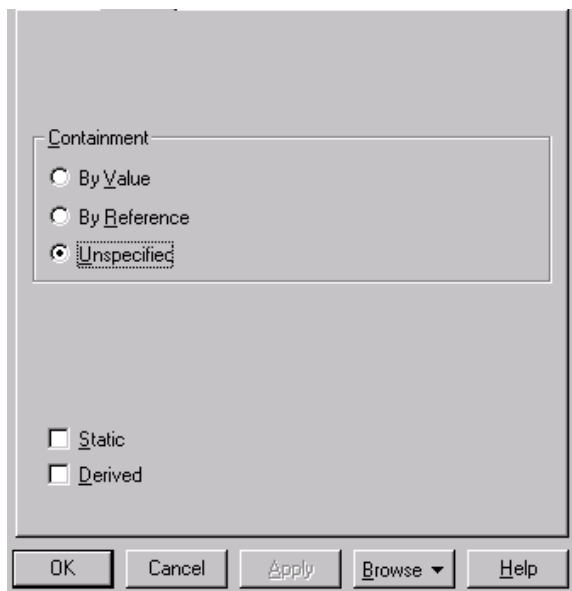


Figure 25 Class Attribute—Detail Tab

Containment

Physical containment plays a role in the construction and destruction of an aggregate's parts through semantics. The specification of physical containment is necessary for meaningful code generation from the model.

You can set one of the following types of physical containment:

Table 9 Physical Containment Options

Type	Description
By Value	Physical containment of a value of the part.
By Reference	Physical containment of a pointer or reference to the part.
Unspecified (default)	The type of physical containment has not been specified.

To set or change the containment type in the **Relationship Specification**, click on the applicable option in the **Containment** field. The application places an adornment at the supplier end of the relationship. You can also select a value from the shortcut menu.

Static

Select the **Static** check box to specify that the client class, not the client's instances, owns the supplier class. In the case of an attribute, a static attribute is an attribute whose value is common to a class of objects rather than a value peculiar to each instance.

You can set this field in the specification or through the shortcut menu.

Derived

The **Derived** check box indicates whether the element was computed or implemented directly.

To define a element as derived, select the **Derived** check box. The element name is adorned by a “/” in front of the name.

Operation Specification

You should complete one **Operation Specification** for each operation that is a member of a class and for all free subprograms.

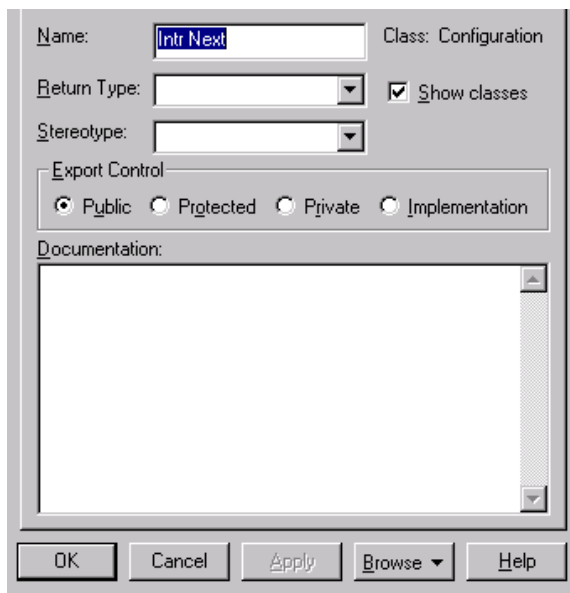
If you change a class operations property by editing its specification, Rational Rose will update all class diagrams containing icons representing that class.

To access the **Operation Specification**, select an entry on the **Operation** tab of the **Class Specification** and double-click the entry or click **Insert** from the shortcut menu. You can also bring the specification up through the shortcut menu.

Specification Content

The **Operation Specifications** consists of the following tabs: **General**, **Detail**, **Preconditions**, **Semantics**, **Postconditions**, and **Files**.

Operation Specification—General Tab



The screenshot shows a dialog box titled "Operation Specification—General Tab". It contains the following fields and controls:

- Name:** A text box containing "Intr Next".
- Class:** A text box containing "Configuration".
- Return Type:** A dropdown menu.
- Stereotype:** A dropdown menu.
- Export Control:** A group box containing four radio buttons: Public, Protected, Private, and Implementation.
- Documentation:** A large empty text area with a vertical scrollbar on the right.
- Buttons:** A row of five buttons: "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

Figure 26 Operations Specification—General Tab

Refer to the descriptions earlier in this chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Return Class

For operations that are functions, set this field to identify the class or type of the function's result. If show classes is set, the list box displays all the classes in the package. If **Show Classes** is not set, only the predefined set of return class types is displayed.

If you enter a class name and it does not exist in your model, the application does not create one.

Operation Specification—Detail Tab

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

Protocol:

Qualification:

Exceptions:

Size:

Time:

Abstract

Concurrency: Sequential Guarded Synchronous

OK Cancel Apply Browse Help

Figure 27 Operation Specification—Detail Tab

Refer to the description in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Arguments

This field contains a list of the arguments of the operation. You may express these arguments in your selected implementation language.

The argument list can be rearranged with the click and drag technique. Select an argument from the list, drag it to the location, and release. The list will reflect the new order.

Protocol

This field lists a set of operations that a client can perform on an object and the legal orderings in which they might be invoked. The protocol of an operation has no semantic impact.

Qualifications

This field identifies language-specific features that qualify the method. You will find this especially useful in Common Lisp Object System (CLOS), in which methods can be described as before or after.

Exceptions

This field contains a list of the exceptions that can be raised by the operation. Enter the name of one or more classes identifying the exception.

Size

This field identifies the relative or absolute amount of storage consumed by the invocation of the operation.

Time

This field contains a statement about the relative or absolute time required to complete an operation. Use this field to budget time for the operation.

Concurrency

This field denotes the semantics in the presence of multiple threads of control. The **Concurrency** field shows the concurrency for the elements of a class. The concurrency of an operation should be consistent with its class.

Table 10 *Concurrency Field Options*

Type	Description
Sequential (default)	The semantics of the operation are guaranteed only in the presence of a single thread of control. Only one thread of control can be executing in the method at any one time.
Guarded	The semantics of the operation are guaranteed in the presence of multiple threads of control. A guarded class requires collaboration among client threads to achieve mutual exclusion.
Synchronous	The semantics of the operation are guaranteed in the presence of multiple threads of control; mutual exclusion is supplied by the class.

You can set the concurrency of a class only through the **Class Specification**. The **Concurrency** field is inactive for class utilities, parameterized class utilities, and instantiated class utilities.

To change the concurrency, click on an applicable option in the **Concurrency** field. You can display the concurrency in the class diagram by clicking **Show Concurrency** from the shortcut menu.

Operation Specification—Preconditions Tab

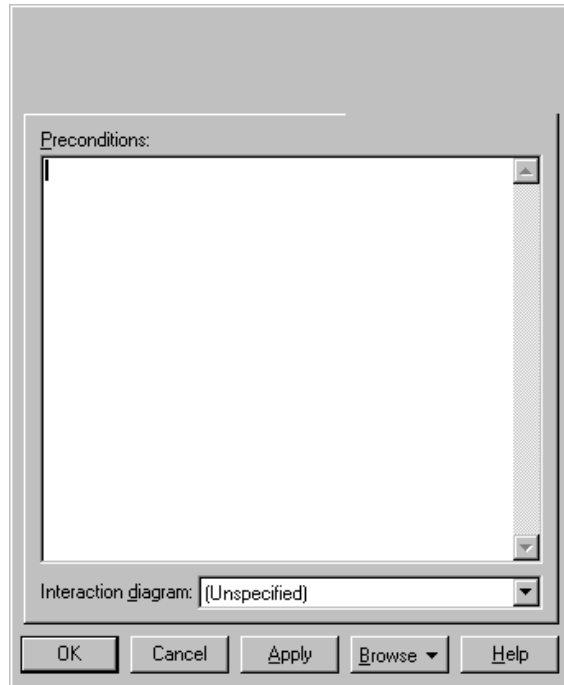


Figure 28 Operation Specification—Precondition Tab

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Preconditions

Invariants that are assumed by the operation (the entry behavior of an operation) are listed.

Interaction Diagram

Select an interaction diagram from the list box that illustrates the appropriate semantics.

Operation Specification—Semantics Tab

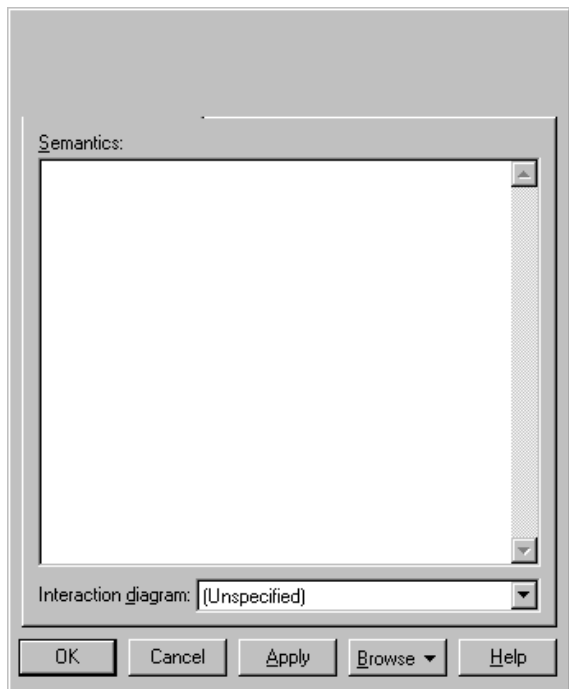


Figure 29 Operations Specification—Semantics Tab

Refer to the description in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Semantics

The action of the operation is shown in this area.

Interaction Diagram

Select an interaction diagram from the list box that illustrates the appropriate semantics.

Operation Specification—Postconditions Tab

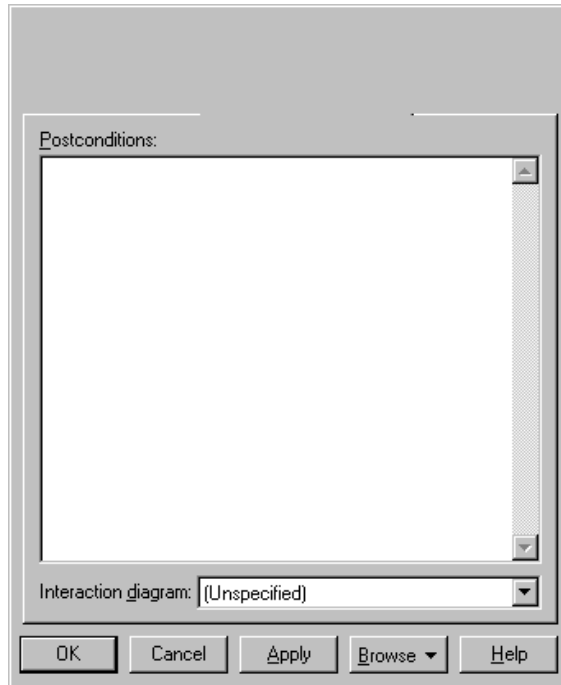


Figure 30 Operation Specification—Postcondition Tab

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Postcondition

Invariants that are satisfied by the operation (the exit behavior of an operation) are listed in this area.

Interaction Diagram

Select an interaction diagram from the list box that illustrates the appropriate semantics.

Operation Specification—Files Tab

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on this tab.

Parameter Specification

A **Parameter Specification** enables you to modify an argument of an operation.

Specification Content

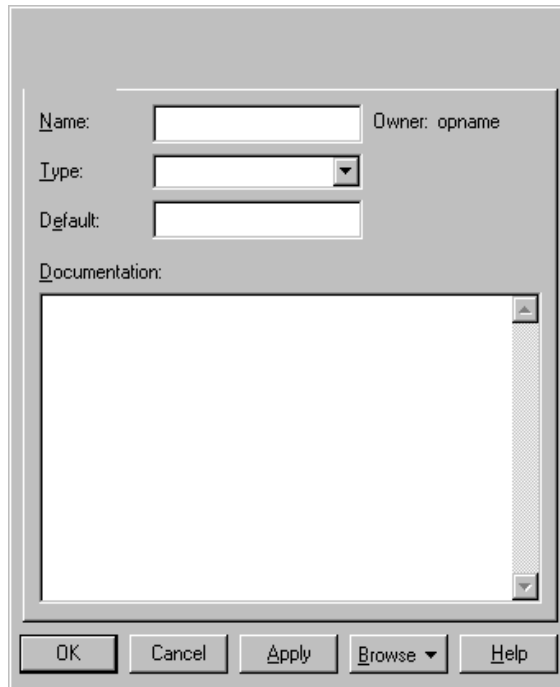
The **Parameter Specification** consists of the following tab: **General**.

Defining a New Parameter

To display a **Parameter Specification**, use the following steps:

1. From a Class Specification's Operation Tab, double-click an operation to display the **Operation Specification**.
2. From the **Operation Specification**, select the **Detail** tab.
3. Move the pointer to the arguments section.
4. Right-click to display the shortcut menu.
5. Click **Insert**, and a new argument is added.
6. Double-click on the argument to display the **Parameter Specification**.

Parameter Specification—General Tab



The image shows a dialog box titled "Parameter Specification—General Tab". It contains the following fields and controls:

- Name:** A text input field.
- Owner:** A label with the value "opname" next to it.
- Type:** A dropdown menu.
- Default:** A text input field.
- Documentation:** A large text area with a vertical scrollbar.
- Buttons:** A row of five buttons: "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

Figure 31 *Parameter Specification—General Tab*

Refer to the description in the *Introduction to Diagrams and Specification* chapter if you need information on the specification elements not covered in the following section.

Default

The default field may contain a value that an instance takes unless otherwise specified.

Owner

The operation is the owner of the parameter.

Type

Type is a description of a set of instances that share the same operations, abstract attributes and relationships, and semantics. Depending upon the language installed, different types will appear.

Association Specification

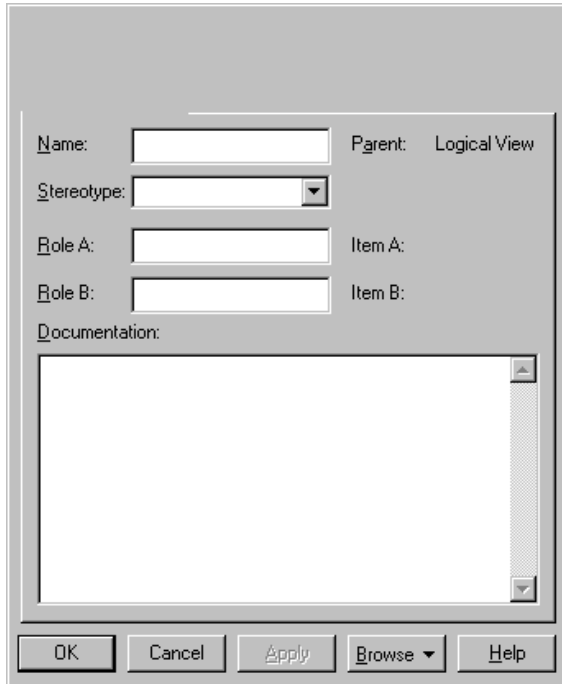
An association represents a bidirectional semantic relationship between two classes.

To display the association specification, double-click any icon representing the processor or click **Browse > Specifications**.

Specification Content

The **Association Specification** consists of the following tabs: **General**, **Detail**, **Role A and Role B General**, and **Role A and Role B Detail**.

Association Specification—General Tab



The screenshot shows a dialog box titled "Association Specification—General Tab". It contains the following fields and controls:

- Name:** A text input field.
- Parent:** A static text field containing the value "Logical View".
- Stereotype:** A dropdown menu.
- Role A:** A text input field.
- Item A:** A text input field.
- Role B:** A text input field.
- Item B:** A text input field.
- Documentation:** A large text area with a vertical scrollbar.
- Buttons:** A row of five buttons: "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

Figure 32 Association Specification—General Tab

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Parent

The parent the component belongs to (its package) is displayed in this static field.

Stereotype

A stereotype represents the subclassification of an element. It represents a class within the UML metamodel itself (that is, a type of modeling element). Some stereotypes are already predefined, but you can also define your own to add new kinds of modeling types.

Role

Use this field to label the role with a name that denotes the purpose or capacity wherein one class associates with another.

To enter a role name, click in the **Role** field and enter the text.

Element

The **Element** field describes the two elements which this association associates. This field cannot be edited.

Association Specification—Detail Tab

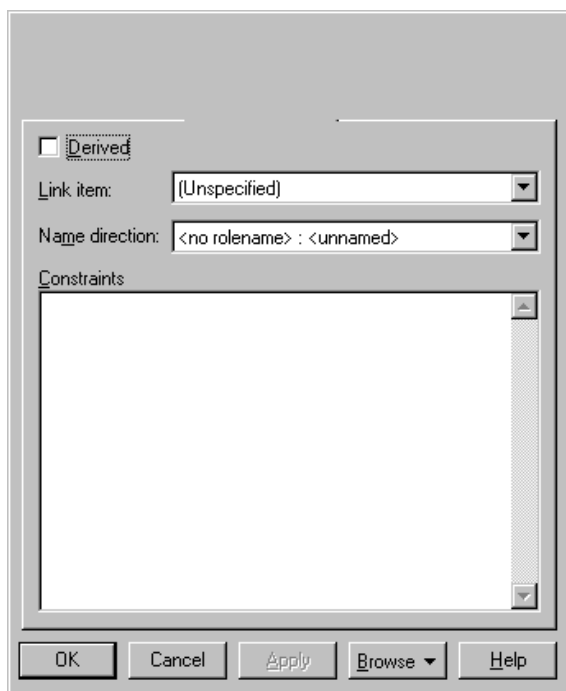


Figure 33 Association Specification—Detail Tab

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Derived

This field indicates whether the element was computed or implemented directly.

To define an element as derived, select the **Derived** check box. The element name is adorned by a “/” in front of the name.

Link Element

This field lists the attributed associations linked to the association. These attributed associations apply to the association as a whole.

Name Direction

This field defines the direction of a role.

Constraints

The constraint is an expression of some semantic condition that must be preserved while the system is in a steady state. The constraint on the **Detail** tab applies to the association as a whole, while the constraint on the **Detail A** or **Detail B** tab applies to a particular role.

To apply a constraint, click in the **Constraint** field and enter the text. Constraints are displayed notationally, surrounded by braces under the role for which it applies.

Association Specification—Role B General Tab

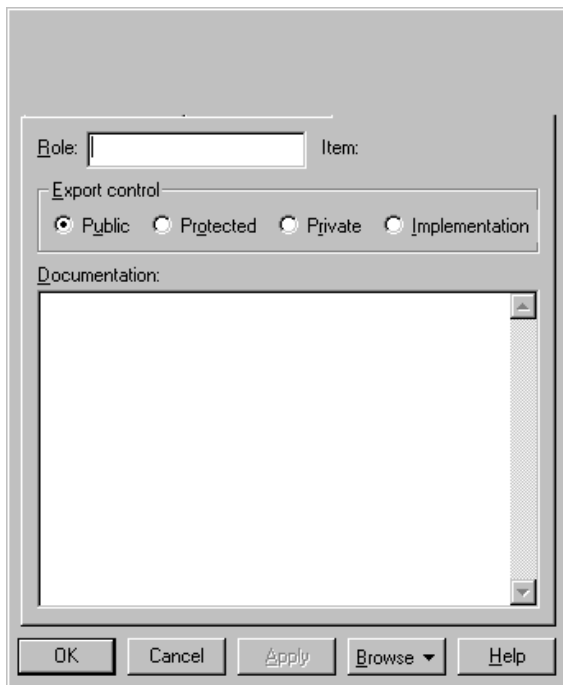


Figure 34 Association Specification—Role A and B General Tab

Refer to the descriptions earlier in this chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the elements shown on this specification.

Association Specification—Role A and B Detail Tab

Figure 35 Association Specification—Role A and B Detail Tab

Refer to the descriptions earlier in the chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following sections.

Navigable

The **Navigable** field indicates in which direction the role is navigating. By default, roles are bidirectional and no navigation notation is provided.

To set a role's navigation, click on the **Navigable** box in the **Association Specification** or click **Navigable** through the shortcut menu. The navigable arrowhead points in the direction of the role, unless a containment adornment is displayed. Containment adornments override navigable adornments.

Aggregate

Use the **Aggregate** field to set a direction to either all or part of the relationship among instances of these classes. Only one end of the relationship can be aggregate.

To set the aggregate adornment, click on the **Aggregate** box in the **Association Specification** or click **Aggregate** through the shortcut menu. The adornment is a diamond on the relationship.

Static

Use the **Static** field to specify that the client class, not the client's instances, owns the supplier class. In the case of an attribute, a static attribute is an attribute whose value is common to a class of objects rather than a value peculiar to each instance.

You can set this field in the specification or through the shortcut menu. To switch the static adornment in the **Relationship Specification**, select the **Static** check box.

Friend

The friend field designates that the supplier class has granted rights to a client class to access its non-public parts.

You can set this field in the **Relationship Specification** or through the relationship's shortcut menu.

Containment of

Physical containment has semantics that play a role in the construction and destruction of an aggregate's parts. The specification of physical containment is necessary for meaningful code generation from the model.

You can set one of the following types of physical containment:

Table 11 Containment Field Options

Type	Description
By Value	Physical containment of a value of the part.
By Reference	Physical containment of a pointer or reference to the part.
Unspecified (default)	The type of physical containment has not been specified.

You can change the containment type in the **Relationship Specification** or you can select a value from the relationship's shortcut menu.

Keys/Qualifiers

A key or qualifier is an attribute that uniquely identifies a single target object. The attributes allow 1..n or n..n associations and reduce the number of instances. The list box will display all keys or qualifiers currently defined.

To enter a key or qualifier, click **Insert** from the shortcut menu or press the **Insert** key. An untitled entry is placed in the name and type field. To change the entry, select to highlight and type in a new name.

For information on the Key/Qualifier specification, refer to the *Key/Qualifier* section later in this chapter.

Generalize Specification

A generalize relationship between classes shows that one class shares the structure or behavior defined in one or more other classes.

Specification Content

The **Generalize Specification** consists of the following tab: **General**.

Generalize Specification—General Tab

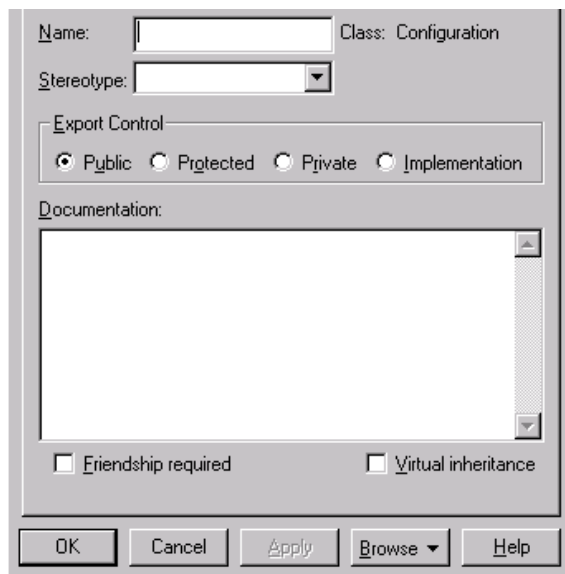


Figure 36 Generalize Specification—General Tab

Refer to the descriptions earlier in this chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the elements not covered in the following section.

Friendship Required

Select the **Friendship required** check box to specify the supplier class has granted rights to the client class to access its non-public members.

Virtual Inheritance

Select the **Virtual Inheritance** check box to ensure that only one copy of the base class will be inherited by descendants of the subclasses.

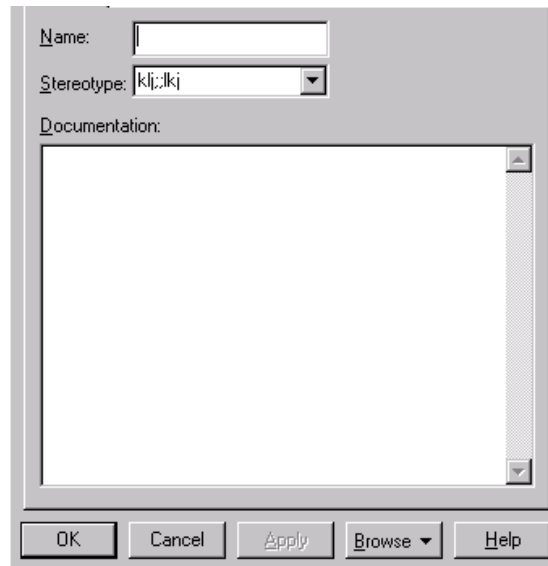
Realize Specification

A realize relationship connects a class to an interface or a component to an interface.

Specification Content

The **Realize Specification** consists of the following tab: **General**.

Realize Specification—General Tab

The image shows a dialog box titled "Realize Specification—General Tab". It contains the following fields and controls:

- Name:** A text input field.
- Stereotype:** A dropdown menu with the value "klj;lkj" selected.
- Documentation:** A large text area for entering documentation.
- Buttons:** "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

Figure 37 Realize Specification—General Tab

Refer to the descriptions earlier in this chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the elements shown on this specification.

Dependency Specification

The dependency relationship indicates that the client class depends on the supplier class to provide certain services. One class can use another class in a variety of ways. Typically, a dependency relationship indicates that the operations of the client invoke operations of the supplier. Dependency relationships appear on component diagrams and they can also be used to connect use cases.

Note: A dependency that connects two use cases together contains a simpler form of the dependency specification pictured below. Only the name, class, stereotype, and documentation fields are present.

Specification Content

The **Dependency Specification** consists of the following tab: **General**.

Dependency Specification—General Tab

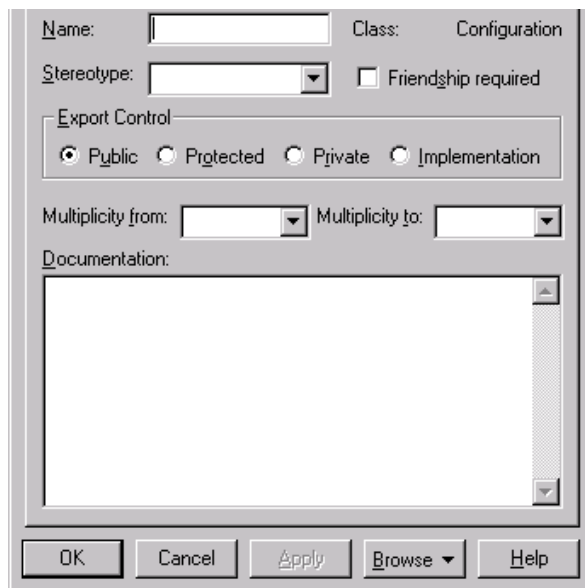


Figure 38 *Dependency Specification—General Tab*

Refer to the descriptions earlier in this chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the elements shown on this specification.

Has Relationship (Booch Only)

A *has* relationship shows a whole and part relationship between two classes, where one class is the whole and the other is the part. The whole class contains or owns its parts. This relationship is also called an aggregation relationship.

Because attributes for a class can be expressed by a *has* by-value relationship with cardinality of "1," attributes are also defined in the *has* relationship specifications.

To display a *has* relationship's specification, select any icon representing the *has* relationship and either double-click or click **Browse > Specifications**.

Specification Content

The **Has Specification** consists of the following tab(s): **General** and **Detail**.

Has Specification—General Tab

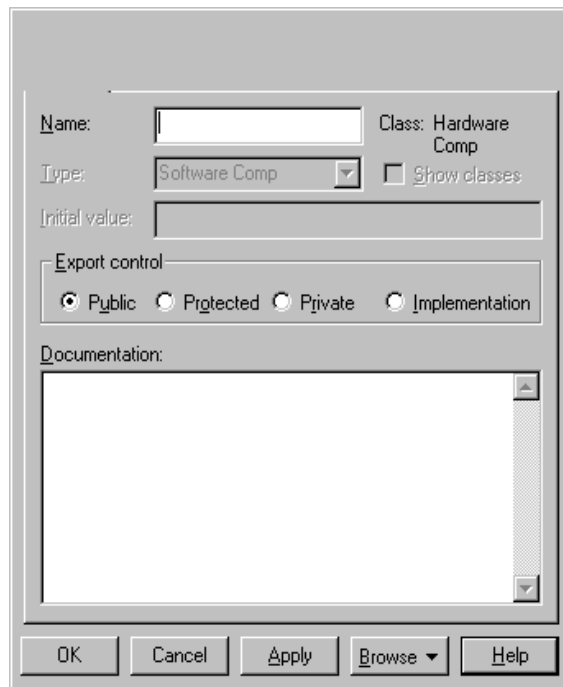


Figure 39 *Has Specification—General Tab*

Refer to the section on the *Class Attribute General Tab* earlier in this chapter for more information.

Has Specification—Detail Tab

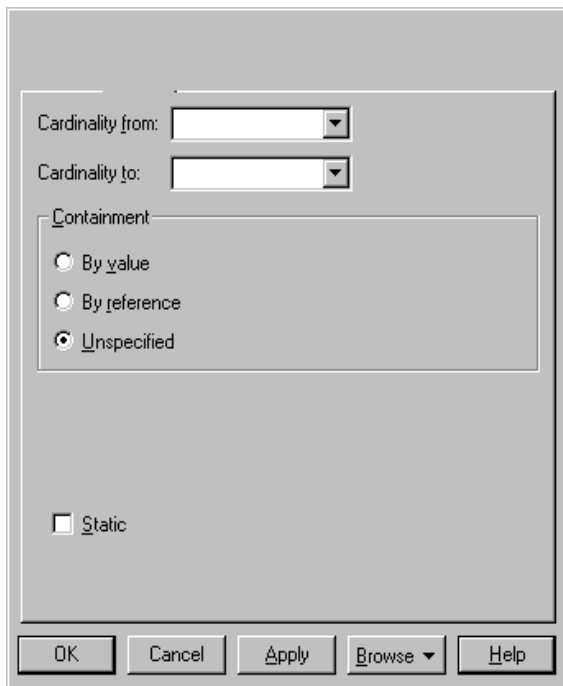


Figure 40 *Has Specification—Detail Tab*

Refer to the descriptions earlier in this chapter, or in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements on this tab.

Key/Qualifier Specification

A **Key/Qualifier Specification** enables you to modify a specific attribute whose value uniquely identifies a single target object.

Defining a New Key/Qualifier

To display a **Key/Qualifier Specification**, use the following steps:

1. Double-click on an association or aggregation.
2. From either the **Association Specification** or the **Aggregation Specification**, select the **Role A Detail** or **Role B Detail**.
3. Move the pointer to the **Key/Qualifier** section of either specification.
4. Right-click to display the shortcut menu.
5. Click **Insert**, and a Key/Qualifier is added.
6. Double-click on the entry to display the **Key/Qualifier Specification**.

Specification Content

The **Key/Qualifier Specification** consists of the following tab: **General**.

Key/Qualifier Specification—General Tab

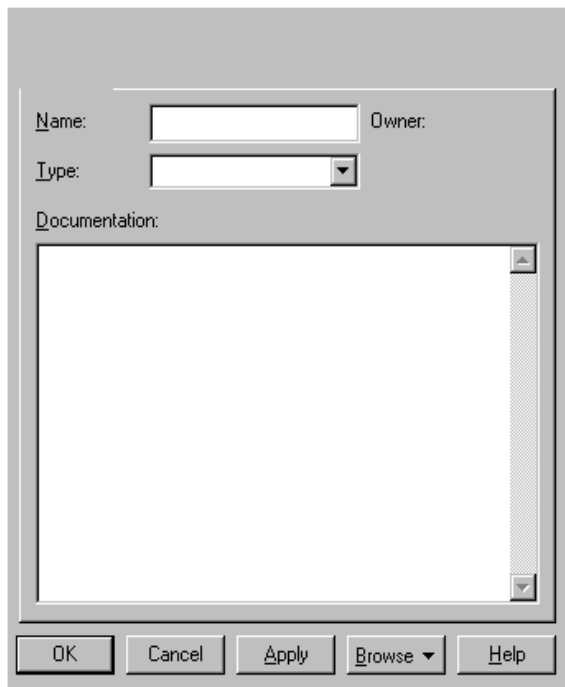


Figure 41 *Key/Qualifier Specification—General Tab*

Refer to the description in the *Introduction to Diagrams and Specification* chapter if you need information on the specification elements not covered in the following section.

Owner

The owner static field identifies the name, or owner, of the role that the key/qualifier evolved from.



Chapter 7

Use-Case Diagrams and Specifications

Use-Case Diagram Overview

Use-case diagrams present a high-level view of how a system is used as seen from an outsider's (or actor's) perspective. These diagrams graphically depict system behavior (also known as use cases). A use-case diagram may depict all or some of the use cases of a system.

A use-case diagram can contain:

- Actors ("things" outside the system).
- Use cases (system boundaries identifying what the system should do).
- Interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations.

Use-case diagrams can be used during analysis to capture the system requirements and understand how the system should work. During the design phase, use-case diagrams can be used to specify the behavior of the system as implemented.

Actors

Actors represent system users. They help define the system and give a clear picture of what the system should do. It is important to note that an actor interacts with, but has no control over the use cases.

An actor is someone or something that:

- Interacts with or uses (but is not part of) the system.
- Provides input to and receives information from the system.
- Is external to the system and has no control over the use cases.

Actors are discovered by examining:

- Who directly uses the system.
- Who is responsible for maintaining the system.
- External hardware used by the system.
- Other systems that need to interact with the system.

An actor is a stereotype of a class and is depicted as a “stickman” on a use-case diagram. The name of the actor is displayed below the icon.

Use Case

A use case is a sequence of events (transactions) performed by a system in response to a trigger initiated by an actor. A use case contains all the events that can occur between an actor-use case pair, not necessarily the ones that will occur in any particular scenario.

In its simplest form, a use case can be described as a specific way of using the system from a user’s (actor’s) perspective. A use case also illustrates:

- A pattern of behavior the system exhibits
- A sequence of related transactions performed by an actor and the system

Use cases provide a means to:

- Capture system requirements
- Communicate with the end users and domain experts
- Test the system

Use cases are best discovered by examining what the actor needs and defining what the actor will be able to do with the system; this helps ensure that the system will be what the user expects.

Since all the needs of a system typically cannot be covered in one use case, it is usual to have a collection of use cases. Together this use case collection specifies all the ways of using the system.

A use case may have a name, although it is typically not a simple name. It is often written as an informal text description of the actors and the sequences of events between objects. Use case names often start with a verb.

The name of the use case is displayed below the icon.

Flow of Events

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the **Files** tab of a model element. See Chapter 6 for a discussion on the **Files** tab.

A flow of events should include:

- When and how the use case starts and ends
- Use case/actor interactions
- Data needed by the use case
- Normal sequence of events for the use case
- Alternate or exceptional flows

You can use activity diagrams to further model flow of events.

Relationships

Relationships show interactions between actors and use cases. Association, dependency, generalization relationships can be drawn from an actor to a use case. The generalize relationship can be drawn between actors.

Any association relationships are also presented in a text format on the **Relations** tab (described later) for a selected use case or actor.

Association

An association provides a pathway for communication between use cases and actors. Associations are the most general of all relationships and consequentially, the most semantically weak. If two objects are usually considered independently, the relationship is an association. The association name and its stereotype is typically a verb or a verb phrase and is used to identify the type or purpose of the relationship.

There are two different types of associations connected with use-case diagrams: uni-directional and bi-directional.

Uni-directional association: By default, associations in use cases are uni-directional and drawn with a single arrow at one end of the association. The end with the arrow indicates who or what is receiving the communication.

Bi-directional association: To change the communication to be bi-directional, double-click on the association to view the **Association Specification**. Select the appropriate **Role A** (or **B**) **Detail** tab, click **Navigable** to add a check mark, and click **Apply**. You have now made the association bi-directional. The graphic changes from a line with an arrow at one end to a line with no arrow.

If you prefer, you can also customize the toolbox to include the bi-directional tool to the use-case toolbox. See *See “Customizing the Toolbox” on page 14.* for information on adding or deleting tools on a diagram toolbox.

You can create or display a use-case diagram in one of three ways:

- Click **Browse > Use Case Diagram**.
- On the toolbar, double-click the use-case diagram icon.
- In the browser, double-click the use-case diagram icon.

Dependency

A dependency is a relationship between two model elements in which a change to one model element will affect the other model element. Use a dependency relationship to connect model elements with the same level of meaning. Typically, on class diagrams, a dependency relationship indicates that the operations of the client invoke operations of the supplier.

You can connect model elements with dependencies on any diagram except state machine diagrams and object diagrams. For example, you can connect a use case to another use case, a package to another package, and a class to a package. Dependencies are also used on component diagrams to connect model elements.

Extend Stereotype

An extend relationship is a stereotyped relationship that specifies how the functionality of one use case can be inserted into the functionality of another use case. You can place extend stereotypes on all relationships. However, most extend stereotypes are placed on dependencies or associations. Extend relationships are important because they show optional functionality or system behavior.

Include Stereotype

An include relationship is a stereotyped relationship that connects a base use case to an inclusion use case. An include relationship specifies how behavior in the inclusion use case is used by the base use case. Include relationships are important because they represent that the inclusion use case functionality is used by the base use case.

Refine Stereotype

A refine relationship is a stereotyped relationship that connects two or more model elements at different semantic levels or development stages. It represents a fuller specification of something that has already been specified at a certain level of detail. For example, a design class is a refinement of an analysis class. In a refine relationship, the source model element is general and more broadly defined whereas the target model element is more specific and refined.

Generalization

A generalize relationship is a relationship between a more general class or use case and a more specific class or use case. A generalization is shown as a solid-line path from the more specific element to a more general element. The tip of a generalization is a large hollow triangle pointing to the more general element.

You can place a stereotype on any generalization through the Generalization Specification. However, three common stereotypes for generalizations are extends, includes and generalization.

Use-Case Diagram Toolbox

The graphic below shows all the tools that can be placed on the use-case diagram toolbox. See See “Customizing the Toolbox” on page 14. for information on adding or deleting diagram toolbox tools.

The application window displays the following toolbox when the current window contains a use-case diagram and **As Unified** is selected from the **View** menu.

Some icons will be different if *As Booch* or *As OMT* is selected from the *View* menu.

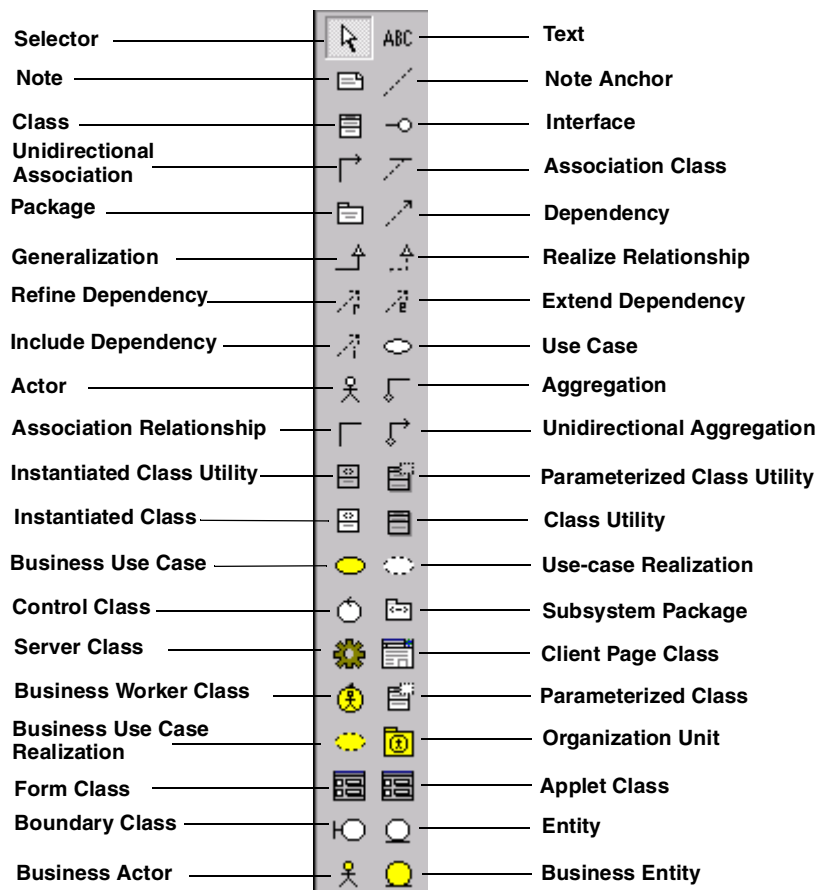


Figure 42 Use-Case Diagram Toolbox

Use-Case Specification

A **Use-Case Specification** enables you to display and modify the properties and relationships of a use case in the current model.

Specification Content

The **Use-Case Specification** contains the following tabs: **General**, **Diagrams**, **Relations**, and **Files**.

Use-Case Specification—General Tab

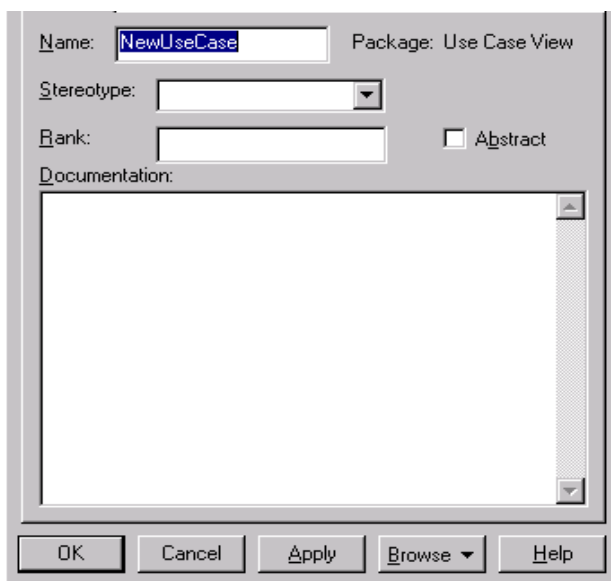


Figure 43 Use-Case Specification—General Tab

Refer to the description in the *Introduction to Diagrams and Specifications* chapter if you need information on the specifications elements not covered in the following section.

Name

A use case name is often written as an informal text description of the external actors and the sequences of events between elements that make up the transaction. Use-case names often start with a verb. The name can be entered or changed on the specification or directly on the diagram.

Package

This static field identifies the package to which the components belong.

Rank

The **Rank** field prioritizes use cases. For example, you can use the rank field to plan what iteration in the development cycle a use case should be implemented.

Abstract

An abstract notation indicates a use case that exists to capture common functionality between use cases (uses) and to describe extensions to a use case (extends).

Use-Case Specification—Diagram Tab

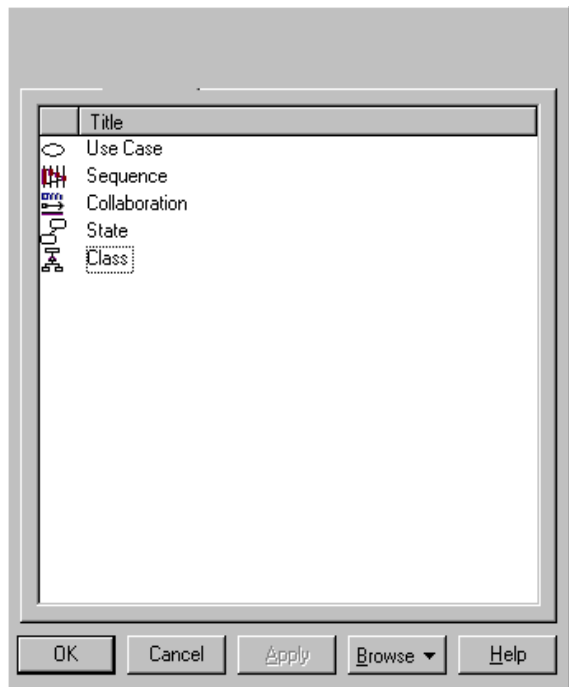


Figure 44 Use-Case Specification—Diagram Tab

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Diagrams

The **Diagrams** list box lists all the diagrams owned by the use case. The diagram list consists of two columns. The first (unlabeled) column displays the diagram icon type for the diagram. The second column displays the diagram name. To insert a new diagram in the list, click one of the **Insert** choices in the shortcut menu that corresponds to the diagram type.

Use-Case Specification—Relations Tab

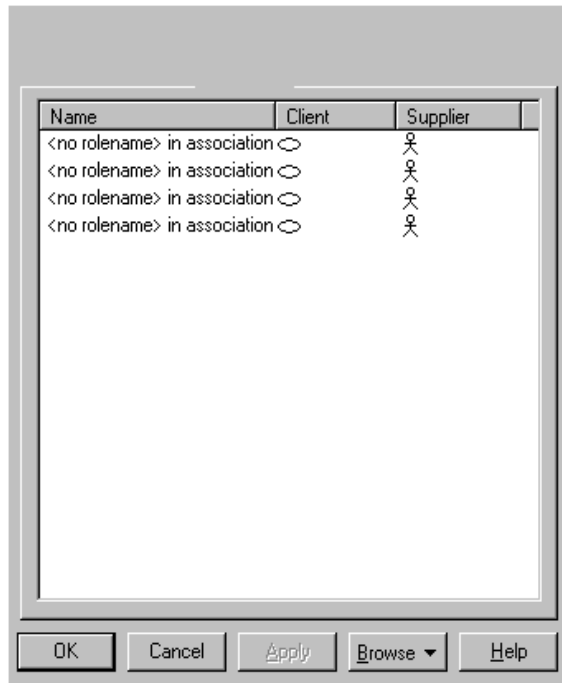


Figure 45 Use-Case Specification—Relations Tab

Refer to the description in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Relations

The **Relations** tab lists all the association relationships that correspond to the selected use case. The client and supplier names and type icons are displayed to the right of the relation name. Double-clicking on any column in a row displays the element's specification.

Generalize Specification—General Tab

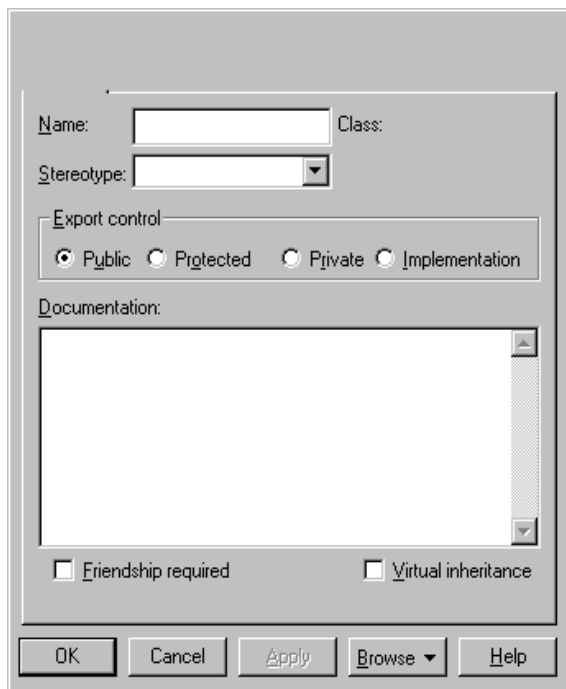


Figure 46 Generalize Specification—General Tab

Refer to the descriptions earlier in this chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the elements not covered in the following section.

Stereotype

Stereotypes allow you to provide additional distinctions in your model that are not explicitly supported by the UML. The use of stereotypes makes it easy to add information about modeling elements that may be specific to a project or process.

The Generalize Specification uses stereotypes to create two new use-case relationships which can be attached to a model element to indicate a special relationship between use cases.

Friendship Required

Select the **Friendship Required** check box to specify that the supplier class has granted rights to the client class to access its non-public members.

Virtual Inheritance

Select the **Virtual Inheritance** check box to ensure that only one copy of the base class will be inherited by descendants of the subclasses.

Actor Specification

An **Actor Specification** looks identical to a **Class Specification**, except that the stereotype field is set to actor. However, some of the fields in the class specification are not applicable to actors and are therefore disabled. Refer to the class specification in the *Class Diagrams and Specifications* chapter for more information.



Chapter 8

State Machine Diagrams and Specifications

The state/activity model icon that appears in the browser can be thought of as a “container” for statechart and activity diagrams and all of their model elements. A state/activity model owns statecharts and activity diagrams and is represented semantically with a state machine. A state machine can be defined as a behavior that specifies the valid sequences of activities that an object or interaction goes through during its life in response to events, together with its responses and actions.

Rational Rose automatically creates one state/activity model when you create a statechart or activity diagram. A state/activity model can be relocated to a new owner, such as a class operation or a use case, by dragging it to a new location in the browser. Rational Rose limits you to only one state/activity model per owner.

Creating and Displaying a State Machine Diagram

To create a State/Activity Model, use the following steps:

1. Click **Browse > State Machine Diagram**.
2. Double-click **New**.
3. Name the diagram.
4. Specify the type of diagram you want to create: **Activity** or **Statechart**.
5. Click **OK**.

State Machine Specification

A State Machine Specification enables you to display and modify the properties and relationships of a State/Activity Model. A state/activity model contains statechart and activity diagrams.

To view the State Machine Specification, double-click the State/Activity Model in the browser.

Changes made either through the specification or directly on the icon are automatically updated throughout the model.

Specification Content

The **State Machine Specification** consists of the following tab: **General**.

State Machine Specification General Tab

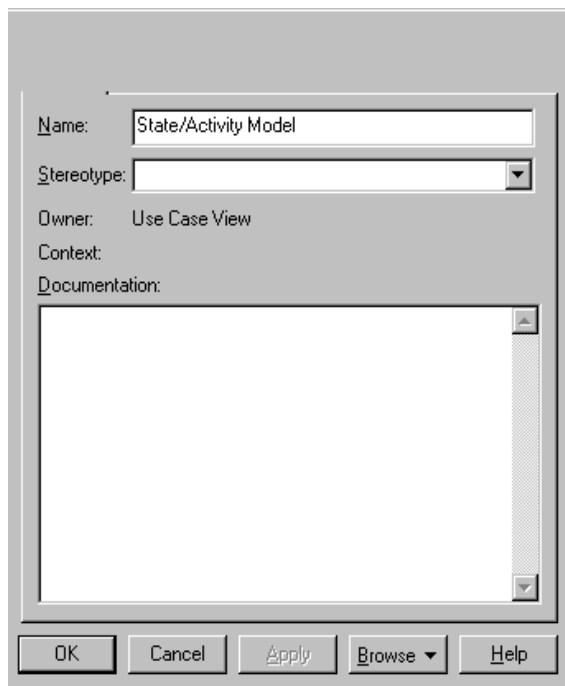


Figure 47 State Machine Specification—General Tab

Statechart Diagram Overview

Statechart diagrams model the dynamic behavior of individual classes or any other kind of object. They show the sequences of states that an object goes through, the events that cause a transition from one state or activity to another, and the actions that result from a state or activity change.

Statechart diagrams are closely related to activity diagrams. The main difference between the two diagrams is statechart diagrams are state centric, while activity diagrams are activity centric. A statechart diagram is typically used to model the discrete stages of an object's lifetime, whereas an activity diagram is better suited to model the sequence of activities in a process.

Each state represents a named condition during the life of an object during which it satisfies some condition or waits for some event. A statechart diagram typically contains one start state and multiple end states. Transitions connect the various states on the diagram. As with activity diagrams, decisions and synchronizations may also appear on statechart diagrams.

Creating a Statechart Diagram

You can create statechart diagrams on most model elements except for attributes, associations, or model elements that appear in the component view.

To create a statechart diagram, use the following steps:

1. In the browser, right-click on any model element except for attributes, associations, or model elements that appear in the component view.
2. Click **New > Statechart Diagram**.

Here is another way to create a statechart diagram:

1. Click the **Browse State Machine Diagram** button from the toolbar.
2. Click **New**.
3. Select the **Statechart Diagram** check box in the **New State Machine** dialog box.
4. Enter the statechart diagram title.

5. Click **OK**.

Automatic Transmission Example

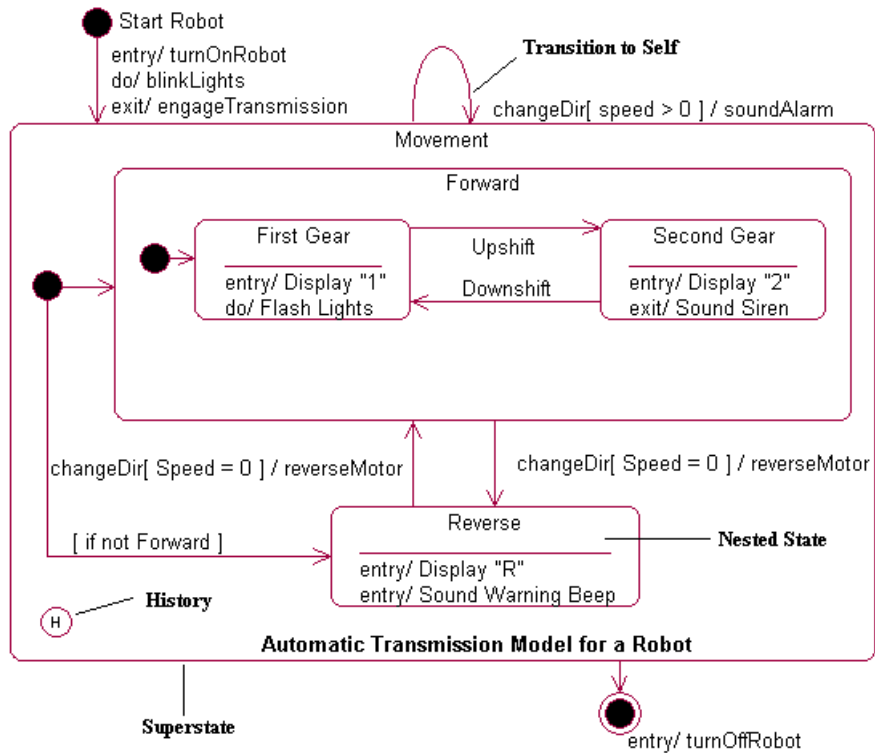


Figure 48 Automatic Transmission Example

Figure 48 illustrates some of the major model elements in a statechart diagram:

- Decisions
- Synchronizations
- States
- Transitions
- Start States
- End States

Activity Diagram Overview

Activity diagrams provide a way to model the workflow of a business process. You can also use activity diagrams to model code-specific information such as a class operation. Activity diagrams are very similar to a flowchart because you can model a workflow from activity to activity. An activity diagram is basically a special case of a state machine in which most of the states are activities and most of the transitions are implicitly triggered by completion of the actions in the source activities. The main difference between activity diagrams and statecharts is activity diagrams are activity centric, while statecharts are state centric. An activity diagram is typically used for modeling the sequence of activities in a process, whereas a statechart is better suited to model the discrete stages of an object's lifetime.

Using Activity Diagrams

Activity diagrams can model many different types of workflows. For example, a company could use activity diagrams to model the flow for an approval of orders or to model the paper trail of invoices. An accounting firm could use activity diagrams to model any number of financial transactions. A software company could use activity diagrams to model a software development process.

Understanding Workflows

Each activity represents the performance of a group of actions in a workflow. Once the activity is complete, the flow of control moves to the next activity or state through a transition. If an outgoing transition is not clearly triggered by an event, then it is triggered by the completion of the contained actions inside the activity. A unique activity diagram feature is a swimlane that defines who or what is responsible for carrying out the activity or state. It is also possible to place objects on activity diagrams. The workflow stops when a transition reaches an end state.

You can attach activity diagrams to most model elements in the use case or logical views. Activity diagrams cannot reside within the component view.

You can use the following tools on the activity diagram toolbox to model activity diagrams:

- Decisions
- Swimlanes
- Objects
- Object Flows
- Activities
- States
- Synchronizations
- Transitions
- Start State
- End State

Creating an Activity Diagram

You can create activity diagrams on most model elements except for attributes, associations, or model elements that appear in the component view.

To create an activity diagram, use the following steps:

1. In the browser, right-click on any model element except for attributes, associations, or model elements that appear in the component view.
2. Click **New > Activity Diagram**.
3. Rename or double-click to display the **NewDiagram** icon in the browser.

Here is another way to create an activity diagram:

1. Click the **Browse State Machine Diagram** button from the toolbar.
2. Click **New**.
3. Select the **Activity Diagram** check box in the **New State Machine** dialog box.
4. Enter the activity diagram title.
5. Click **OK**.

Workflow Modeling

In business and in other industries, there are many manual and automated systems. Each of these systems contains one or more workflows. A workflow is best defined as a well-defined sequence of activities that produces an observable value or objective to an individual or entity when performed. You can model workflows with activity diagrams.

Purposes of Workflow Modeling

The purposes of workflow modeling are threefold:

- To understand the structure and dynamics of an organization
- To ensure that customers, end users, and developers have a common understanding of the organization
- To derive requirements on systems to support the organization

Defining a Workflow

When you define a workflow, your activity diagram should answer the following questions:

Who or what has the overall responsibility of the workflow?

- A use case or class could own each activity diagram, for example.

What activities need to be performed to meet your objective or goal?

- Define all of the high-level activities that need to take place in the workflow. You do not need to define every activity or state, just the ones with the greatest importance in the workflow.

Who will be responsible for performing the various activities and states?

- Define each activity within a swimlane so you know who is responsible for carrying out the activity. Any element within a swimlane is owned and should be carried out by the swimlane.

Do the activities create or modify objects?

- Connect objects and activities with object flows. Specify the state of the object through the state specification.

Where do the activities and states take place with respect to other elements on your diagram?

- Placement of your activities on the diagram determines the order of your workflow.

Why does this activity or state need to take place?

- The reason or purpose for each activity or state should be placed in the specification documentation field.

Modeling a Workflow with an Activity Diagram

Modeling a workflow in an activity diagram can be done several ways; however, the following steps present just one logical process:

1. Identify a workflow objective. Ask, "What needs to take place or happen by the end of the workflow? What needs to be accomplished?" For example, if your activity diagram models the workflow of ordering a book from an online bookstore, the goal of the entire workflow could be getting the book to the customer.
2. Decide the pre and post-conditions of the workflow through a start state and an end state. In most cases, activity diagrams have a flowchart structure so start and end states are used to designate the beginning and ending of the workflow. State and end states clarify the perimeter of the workflow.
3. Define and recognize all activities and states that must take place to meet your objective. Place and name them on the activity diagram in a logical order.
4. Define and diagram any objects that are created or modified within your activity diagram. Connect the objects and activities with object flows.
5. Decide who or what is responsible for performing the activities and states through swimlanes. Name each swimlane and place the appropriate activities and states within each swimlane.
6. Connect all elements on the diagram with transitions. Begin with the "main" workflow.
7. Place decisions on the diagram where the workflow may split into an alternate flow. For example, based on a Boolean expression, the workflow could branch to a different workflow.

8. Evaluate your diagram and see if you have any concurrent workflows. If so, use synchronizations to represent forking and joining.
9. Set all actions, triggers and guard conditions in the specifications of each model element.

Activity Diagram-Specific Model Elements

Activities

An activity represents the performance of “task” or “duty” in a workflow. It may also represent the execution of a statement in a procedure. An activity is similar to a state, but expresses the intent that there is no significant waiting (for events) in an activity.

Swimlanes

Swimlanes are helpful when modeling a business workflow because they can represent organizational units or roles within a business model. Swimlanes are very similar to an object because they provide a way to tell who is performing a certain role. Swimlanes only appear on activity diagrams. You should place activities within swimlanes to determine which unit is responsible for carrying out the specific activity.

When a swimlane is dragged onto an activity diagram, it becomes a swimlane view. Swimlanes appear as small icons in the browser while a swimlane views appear between the thin, vertical lines with a header that can be renamed and relocated.

Objects

Rational Rose allows objects on activity, collaboration, and sequence diagrams. Specific to activity diagrams, objects are model elements that represent something you can feel and touch. It might be helpful to think of objects as the nouns of the activity diagram and activities as the verbs of the activity diagram. Further, objects on activity diagrams allow you to diagram the input and output relationships between activities. In the following diagram, the **Submit Defect** and **Fix**

Defects can be thought of as the verbs and the defect objects as the nouns in the activity diagram vocabulary. Objects are connected to activities through object flows.

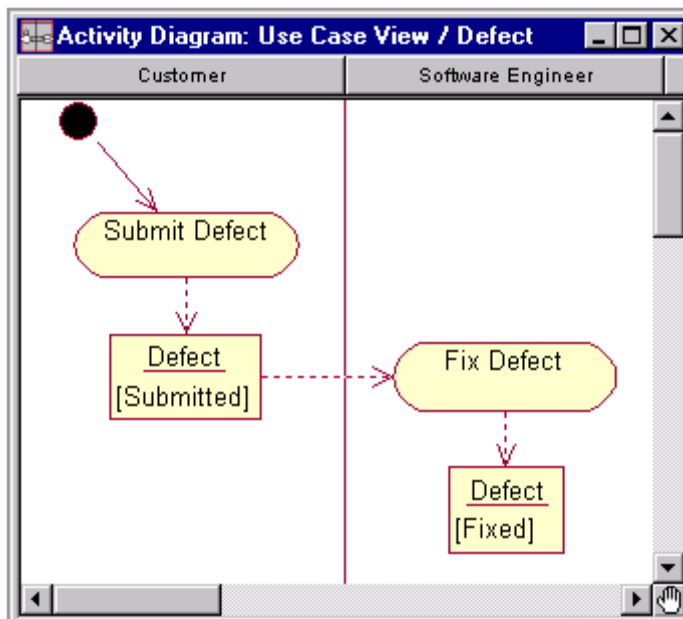


Figure 49 Objects on an Activity Diagram Sample

Most objects can appear in an infinite number of states. For example, look at both instances of the defect object. In one instance, the Customer (noted by the swimlane) placed the defect in a **[submitted]** state. In the other, the software engineer (noted by the swimlane) placed the defect in a **[fixed]** state. Each time you associate a new state with an object, a new state appears in the browser along with the object. You may specify more details of the object's state in the state specification.

Object Flow

An object flow on an activity diagram represents the relationship between an activity and the object that creates it (as an output) or uses it (as an input).

Rational Rose draws object flows as dashed arrows rather than solid arrows to distinguish them from ordinary transitions. Object flows look identical to dependencies that appear on other diagram types.

You do not need a transition if your diagram has two activities connected through an object and two corresponding object flows. The scenario below does not require a transition because the transition is redundant:

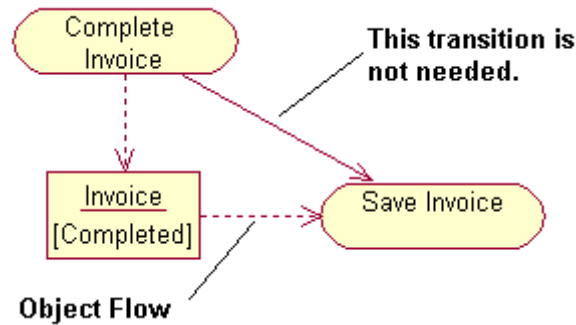


Figure 50 *Object Flow Sample*

Understanding Objects and Object Flows

The object flow sample demonstrates how activities affect object state on activity diagrams. The object flow sample illustrates three important aspects of activity diagram objects:

- Objects may appear more than once and in several states
- activities may change object state
- objects connect with activities through object flows

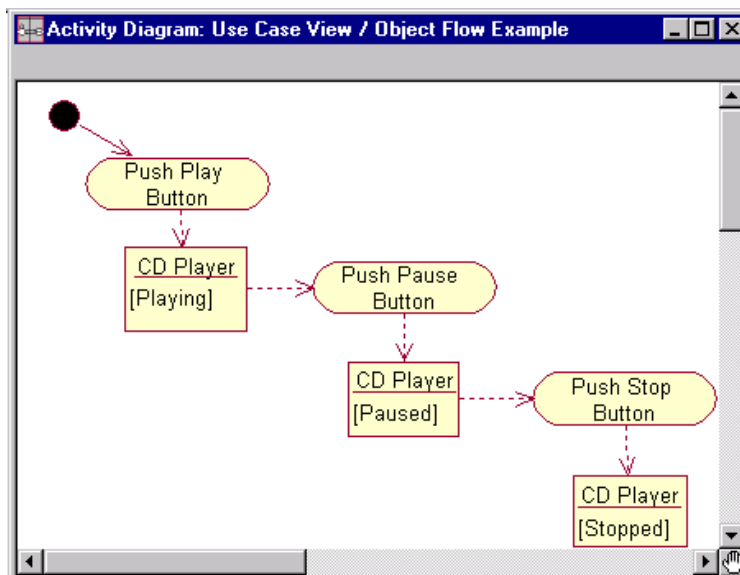


Figure 51 CD Player Sample

In the object flow sample, notice that the CD Player object appears on the diagram more than once. However, each object appears in a different state: playing, paused, and stopped. Each activity in the sample changes the state of the CD Player when you push the various buttons or perform the activity. For example, when you Push Pause Button the state of the CD Player changes to [Paused].

In most cases, the same object may be (and usually is) the output of one activity and the input of one or more subsequent activities.

Changing the State of an Object

To change the state of an object on an activity diagram:

1. Double-click on the object to display the object specification.
2. Select New from the State drop-down menu. A new state specification appears.
3. Enter descriptive information about the object state in the state specification.
4. Click **OK** to close the state specification.
5. Click **OK** to close the object specification.

Shared State Machine Diagram Model Elements

The following paragraphs contain the elements that can appear in both activity diagrams and statechart diagrams:

States

A state represents a condition or situation during the life of an object during which it satisfies some condition or waits for some event. Each state represents the cumulative history of its behavior.

Start and End States

A start state explicitly shows the beginning of a workflow on an activity diagram or the beginning of the events that cause a transition on a statechart. You can have only one start state on a statechart or activity diagram.

An end state represents a final or terminal state on an activity diagram or statechart diagram. Place an end state when you want to explicitly show the end of a workflow on an activity diagram or the end of a statechart diagram.

Transitions

A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied. A state transition is a relationship between two states, two activities, or between an activity and a state.

You can show one or more state transitions from a state as long as each transition is unique. Transitions originating from a state can not have the same event, unless there are conditions on the event. Transitions appear on statechart and activity diagrams.

You should label each state transition with the name of at least one event that causes the state transition. You do not have to use unique labels for state transitions because the same event can cause a transition to many different states or activities.

Transitions are labeled with the following syntax:

```
event (arguments) [condition] / action ^ target.sendEvent  
(arguments)
```

Only one event is allowed per transition, and one action per event.

Events, conditions and actions must be added by editing the label or through the State Transition Specification.

Transition to Self

A transition to self is very similar to a state transition; however, it does not move the focus of control to another state or activity when an event occurs. A transition to self contains the same source and target state or activity.

A transition to self contains actions and events just like transitions.

The icon for a transition to self is a looped line with an arrowhead pointing toward the same source state or activity. The transition to self arc appears on the top of an activity or state icon.

Decisions

A decision represents a specific location on an activity diagram or statechart diagram where the workflow may branch based upon guard conditions. There may be more than two outgoing transitions with different guard conditions, but for the most part, a decision will have only two outgoing transitions determined by a Boolean expression.

Synchronizations

Synchronizations enable you to see a simultaneous workflow in an activity diagram or statechart diagram. They also visually define forks and joins representing parallel workflow.

Swimlane Specification

A Swimlane Specification enables you to display and modify the properties and relationships of a swimlane on an activity diagram.

To display a Swimlane Specification, select the swimlane header on an activity diagram and double-click. You may also double-click on the swimlane icon in the browser.

Specification Content

The swimlane specification consists of the following tab: **General**.

Swimlane Specification General Tab

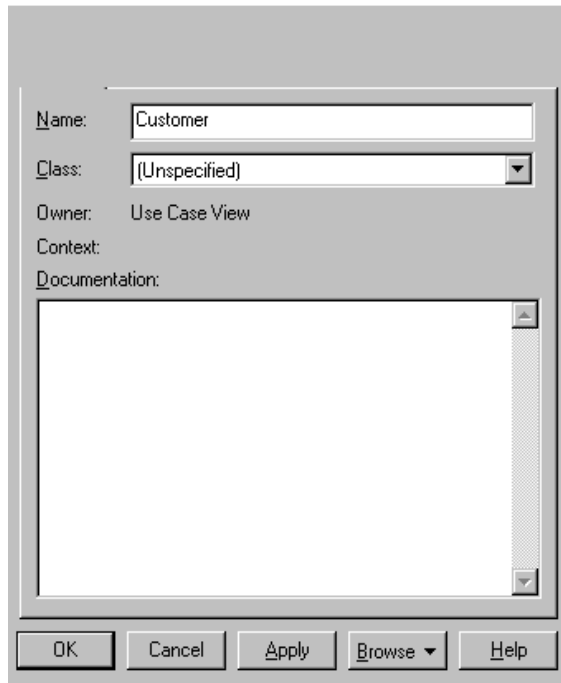


Figure 52 Swimlane Specification—General Tab

State and Activity Specifications

A State and Activity Specification enables you to display and modify the properties and relationships of a state or activity on a statechart diagram or activity diagram. Although a state and activity have almost identical features, they are used for very different purposes. Also, start and end states have identical features as a state specification. Start and End States are actually states. However, they appear as circles on statechart and activity diagrams.

Specification Content

The **State, Activity, Start State, and End State Specifications** consists of the following tabs: **General, Action, Transitions, and Swimlanes.**

State and Activity Specification General Tab

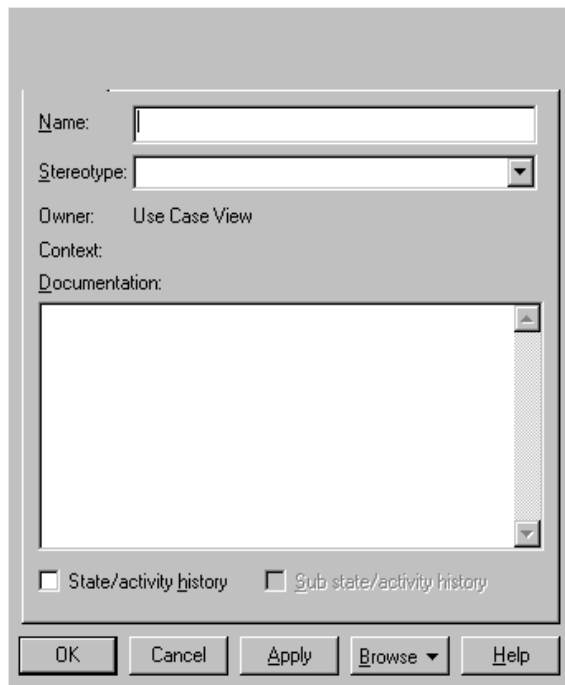


Figure 53 State and Activity Specification—General Tab

Information about the name, stereotype, owner, context, documentation, state/activity history and sub state/activity history is entered or displayed on this tab.

Refer to the descriptions in the Introduction to Diagrams and Specifications chapter if you need information on the elements shown on this specification.

State/Activity History

History provides a mechanism to return to the most recently visited state when transitioning directly to a state with substates. History applies to the level in which it appears. It may also be applied to the lowest depth of nested states.

To apply history at the state or activity level, click **State/activity history**. Click **Sub state/activity history** to apply history to all the depths of nested states or activities within the state or activity level.

State and Activity Specification Actions Tab

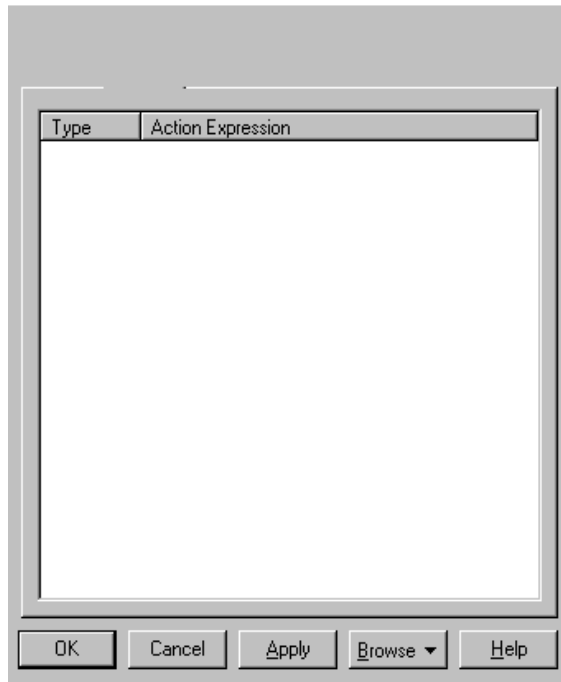


Figure 54 *State and Activity Specification—Actions Tab*

Information about the type and action expression is entered or displayed on this tab.

Type

The Type field identifier bar lists the kind of action specified in the Action Specification.

Action Expression

The Action Expression field identifier bar lists the four possible timing options that specify when to carry out an action and it specifies the types of actions that are carried out. You can modify the action settings through the Action Specification Detail tab.

For information on the Actions Specification, refer to the Action Specification.

State and Activity Specification Transitions Tab

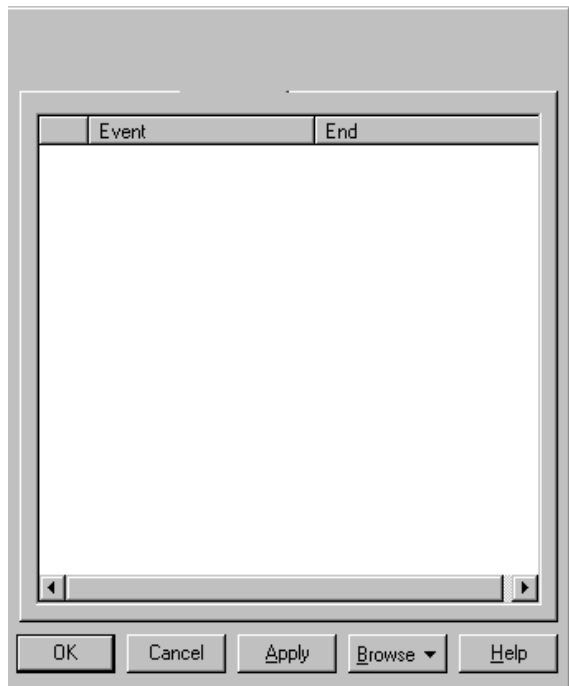


Figure 55 State and Activity Specification—Transitions Tab

Information about the icon, event and end is displayed on this tab.

State and Activity Specification Swimlanes Tab

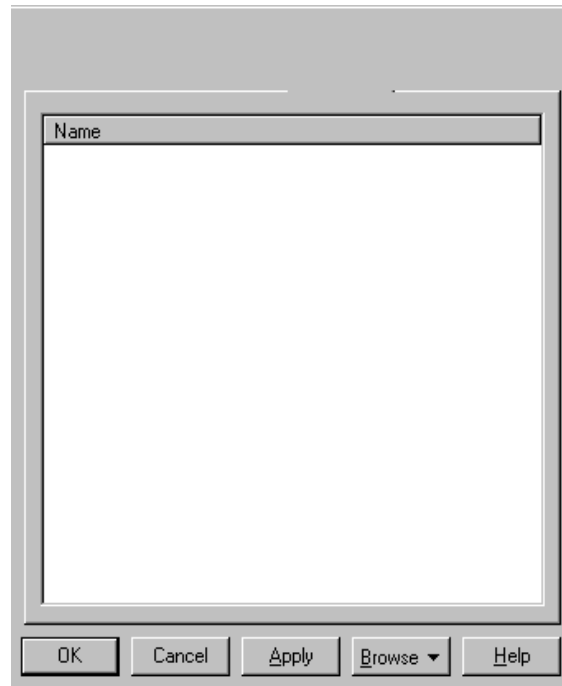


Figure 56 *State and Activity Specification—Swimlanes Tab*

Information about the swimlane **name** is displayed on this tab.

Action Specification

An Action Specification enables you to display and modify the action properties in a statechart diagram or activity diagram.

How to Define a New Action

Use the following steps to define a new action on a state or activity from a State or Activity Specification Actions tab:

1. Click the **Actions** tab of a **State Specification** or **Activity Specification**.
2. Right-click to display the shortcut menu.
3. Click **Insert** and an entry item is added.

4. Double-click on the entry to display the **Action Specification**.
5. Type the action description in the **Name** field. If this field is not active, click **Action** on the **Type** field.

If you select Send Event, you may type optional arguments to the triggered event in the Send Arguments field and the name of another object in the model in the Send Target field.

State and Activity Actions

Each state and activity on a statechart or activity diagram may contain any number of internal actions. An action is best described as a “task” that takes place while inside a state or activity. There are four possible actions within a state or activity:

- On Entry
- On Exit
- Do
- On Event

On Event

The On Event parameters are only enabled when you set the On Event timing parameter.

Event—In a statechart or activity diagram, an event is an occurrence that can trigger a state transition. Type the name of the event that will trigger the action.

Arguments—Consists of any optional arguments associated with the event.

Condition—May contain a conditional Boolean expression.

There is an advantage to using an **On Event** state action rather than a transition to self. Transitions to self trigger all the actions associated with a state, whereas state actions handle internal state transitions. This provides you with the control to process an internal event without triggering the entry and exit actions. The Trigger specification contains the same features as the Action Specification. The Trigger Specification defines the properties of a trigger.

Specification Content

The **Action Specification** consists of the following tab: **Detail**.

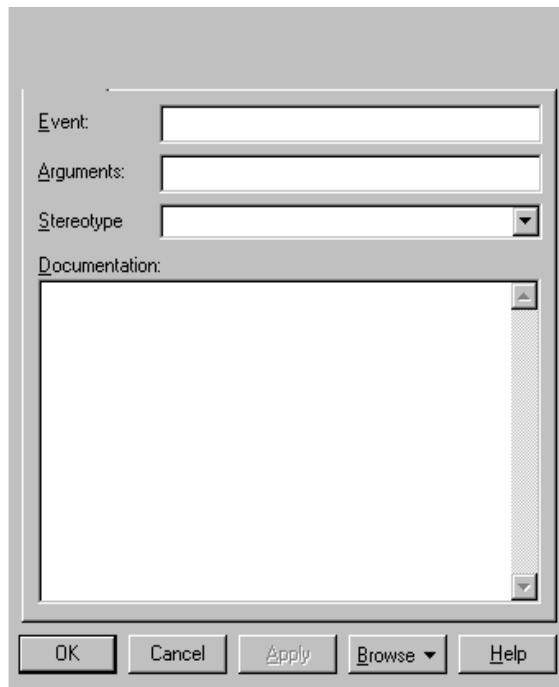
Transition Specification

A State Transition Specification enables you to display and modify the properties and relationships of a transition on a statechart diagram or activity diagram. The state transition specification lists the events and actions that comprise the transition.

Specification Contents

The **State Transition Specification** consists of the following tabs: **General**, and **Detail**.

Transition Specification – General Tab



The image shows a dialog box titled "State Transition Specification" with the "General" tab selected. The dialog contains the following fields and controls:

- Event:** A text input field.
- Arguments:** A text input field.
- Stereotype:** A dropdown menu.
- Documentation:** A large text area with a vertical scrollbar.
- Buttons:** OK, Cancel, Apply, Browse (with a dropdown arrow), and Help.

Figure 57 *State Transition Specification—General Tab*

Refer to the descriptions earlier in this chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

State Transition Specification Detail Tab

The dialog box is titled "State Transition Specification Detail Tab". It contains the following fields and controls:

- Guard Condition:
- Action:
- Send event:
- Send arguments:
- Send target:
- Transition between substates:
 - From:
 - To:

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

Figure 58 State Transition Specification—Detail Tab

Guard Condition

Conditional state transitions are triggered only when the conditional expression evaluates to true. Conditions are denoted by surrounding brackets:

`Event (args) [condition] / Action ^target.someEvent (args)`

To add a condition, click **Condition** on the State Transition Specification and type the conditional expression. You may also include a condition by selecting the event label and changing the text.

Transition Between Substates

Transition between substates is useful when a transition is placed to or from a substate that has been hidden from view. The **From** field displays the state name from which the transition is initiated. The **To** field displays the state name from which the transition is pointing. Both fields are active at all times.

To enter a transition substate, click the scrolling arrow on the right side of the field. A list of potential transition substates will be presented. The list includes the name of all the states that reside within the bounds of the top level superstate, including the superstate. Select a state from the list.

Decision Specification

A Decision Specification enables you to display and modify the properties and relationships of a decision on a statechart diagram or activity diagram.

The **Decision Specification** consists of the following tabs: **General**, **Transitions**, and **Swimlanes**.

Decision Specification General Tab



Figure 59 Decision Specification—General Tab

Refer to the descriptions earlier in this chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Decision Specification Transitions Tab

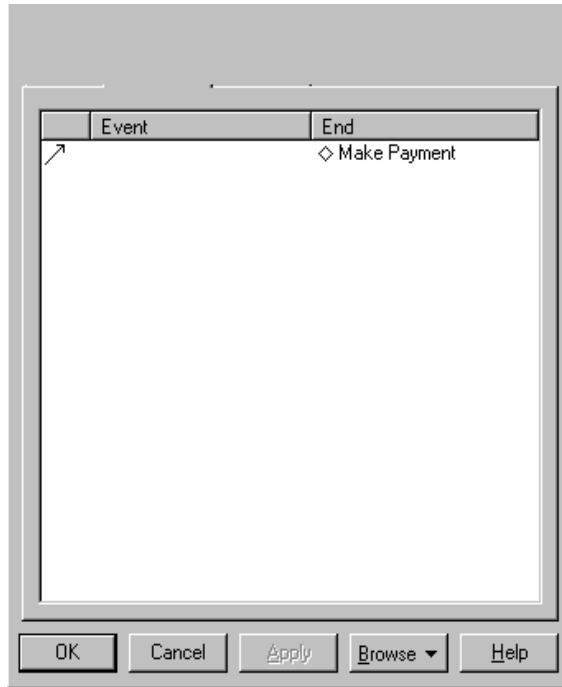


Figure 60 *Decision Specification—Transition Tab*

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Decision Specification Swimlanes Tab

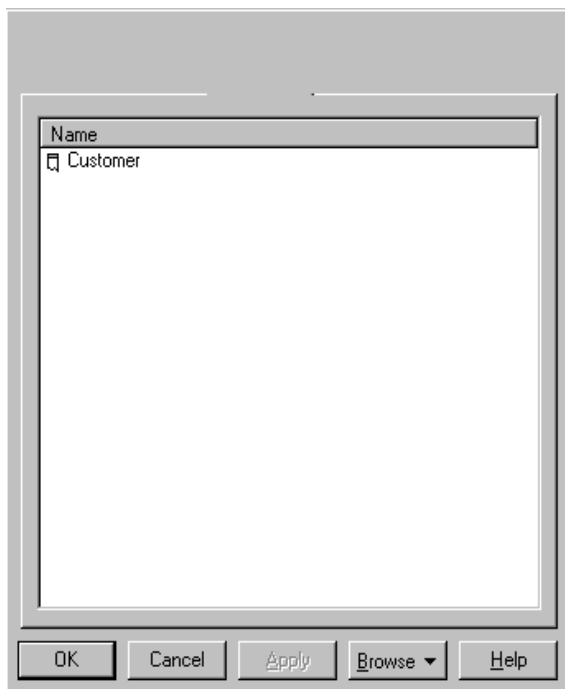


Figure 61 Decision Specification—Swimlane Tab

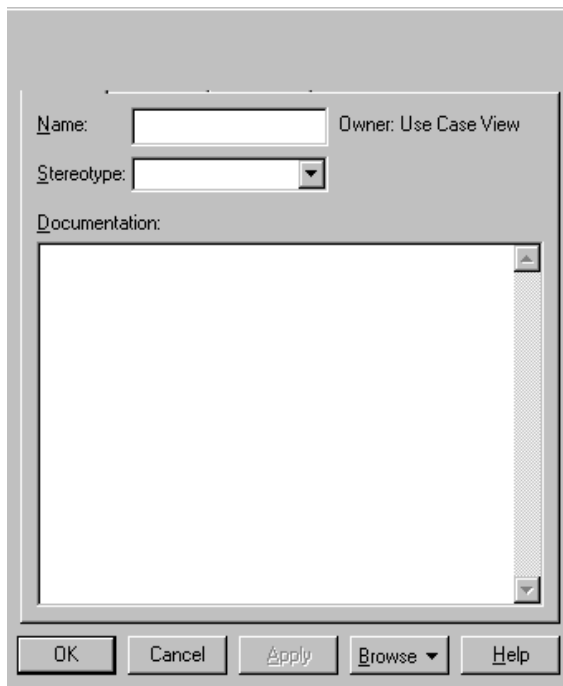
Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Synchronization Specification

A Synchronization Specification enables you to display and modify the properties and relationships of a synchronization on a statechart diagram or activity diagram.

The **Decision Specification** consists of the following tabs: **General** and **Transitions**.

Synchronization Specification General Tab



The image shows a dialog box titled "Synchronization Specification General Tab". It contains the following elements:

- Name:** A text input field.
- Owner:** A label with the text "Use Case View".
- Stereotype:** A dropdown menu.
- Documentation:** A large text area with a vertical scrollbar on the right side.
- Buttons:** A row of five buttons at the bottom: "OK", "Cancel", "Apply", "Browse" (with a small downward arrow), and "Help".

Figure 62 Synchronization Specification—General Tab

Refer to the descriptions earlier in this chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Synchronization Specification Transitions Tab

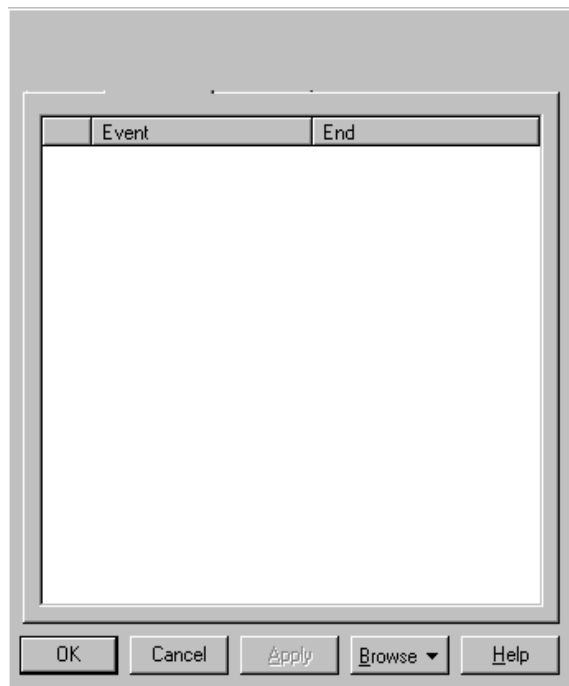


Figure 63 Synchronization Specification—Transitions Tab

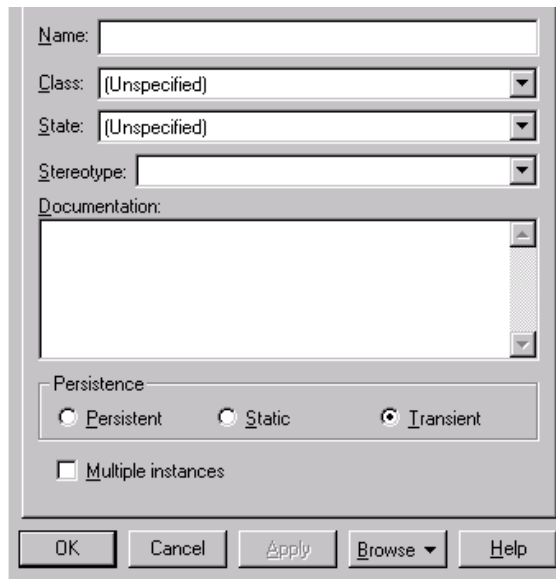
Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section

Object Specification (Activity Diagrams)

An Object Specification enables you to display and modify the properties of an activity diagram object. The object specifications that appear from objects on an activity diagrams are slightly different than the object specifications derived from a sequence or collaboration diagram. Activity diagram object specifications contain state and stereotype menus.

The **Object Specification** for an activity diagram consists of the following tabs: **General**, **Incoming Object Flows**, and **Outgoing Object Flows**

Object Specification General Tab



The image shows a dialog box titled "Object Specification General Tab". It contains several input fields and controls:

- Name:** A text input field.
- Class:** A dropdown menu currently showing "(Unspecified)".
- State:** A dropdown menu currently showing "(Unspecified)".
- Stereotype:** A dropdown menu.
- Documentation:** A large text area with a vertical scrollbar.
- Persistence:** A group box containing three radio buttons: Persistent, Static, and Transient.
- Multiple instances:** A checkbox that is currently unchecked.
- Buttons:** At the bottom, there are five buttons: "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

Figure 64 Object Specification—General Tab

State

The state drop-down menu specifies and displays the object state.

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Object Specification Incoming Object Flows Tab

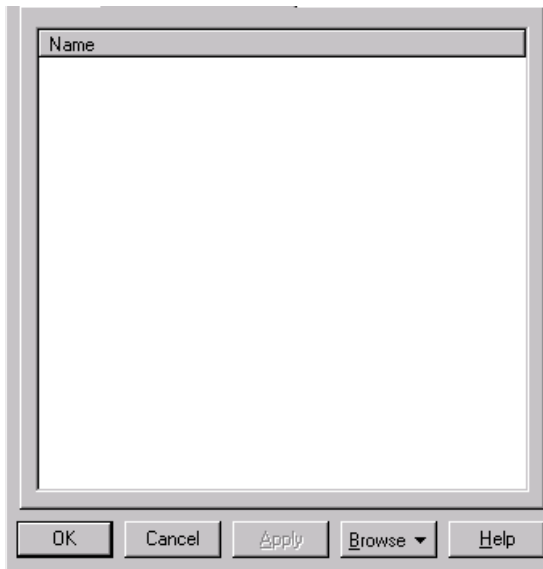


Figure 65 *Object Specification—Incoming Object Flows Tab*

The **Incoming Object Flows** tab displays the name of all incoming object flows.

Object Specification Outgoing Object Flows Tab

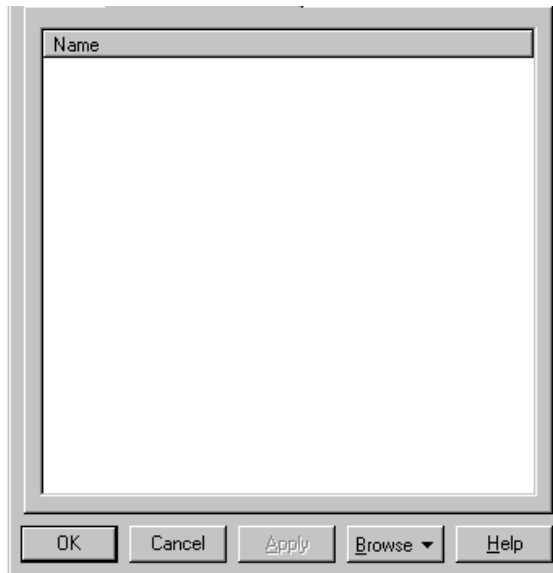


Figure 66 *Object Specification—Outgoing Object Flows Tab*

The **Outgoing Object Flows** tab displays the name of all outgoing object flows.

Object Flow Specification

An Object Flow Specification enables you to display and modify the properties and relationships of an object flow on an activity diagram.

The **Object Flow** Specification consists of the following tab: **General**.

Object Flow Specification General Tab

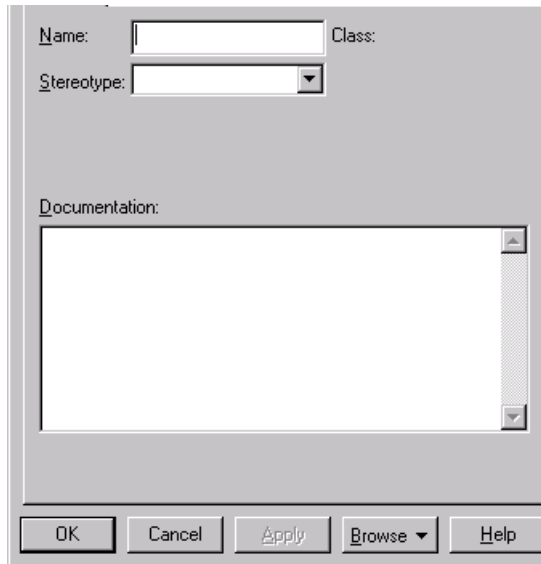


Figure 67 Object Flow Specification—General Tab

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.



Chapter 9

Interaction Diagrams and Specifications

Interaction Diagram Overview

An interaction is an important sequence of interactions between objects. Rational Rose provides two alternate views or representations of each interaction—a collaboration and sequence diagram. These are collectively referred to as interaction diagrams. There are two main differences between sequence and collaboration diagrams: sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other.

You can specify and modify an interaction with either kind of diagram, or with both. Rational Rose automatically reflects all changes made either to a sequence or collaboration diagram in the corresponding collaboration or sequence diagram, if one has been created.

Creating and Displaying an Interaction Diagram

To create or display a collaboration or sequence diagram:

1. Click **Browse > Interaction Diagram**. The **Select Interaction Diagram** dialog box is displayed.
2. Select a package to “own” the diagram.
3. On the right side of the dialog box, click the diagram name, then click **OK**.

4. From the **New Interaction Diagram** dialog box, enter the diagram title and click the diagram type. Your choices are **Sequence** or **Collaboration**. Each diagram type is described in detail later in this chapter.

Collaboration Diagrams

A collaboration diagram is an interaction diagram which shows the sequence of messages that implement an operation or a transaction. These diagrams shows objects, their links, and their messages. They can also contain simple class instances and class utility instances. Each collaboration diagram provides a view of the interactions or structural relationships that occur between objects and object-like entities in the current model.

You can create one or more collaboration diagrams to depict interactions for each logical package in your model. Such collaboration diagrams are themselves contained by the logical package enclosing the objects they depict.

During analysis, collaboration diagrams can indicate the semantics of the primary and secondary interactions.

During design, collaboration diagrams can show the semantics of mechanisms in the logical design of the system.

Use collaboration diagrams as the primary vehicle to describe interactions that express your decisions about the behavior of the system. They can also be used to trace the execution of a scenario by capturing the sequential and parallel interaction of a cooperating set of objects.

Collaboration diagrams may also depict interactions that illustrate system behavior.

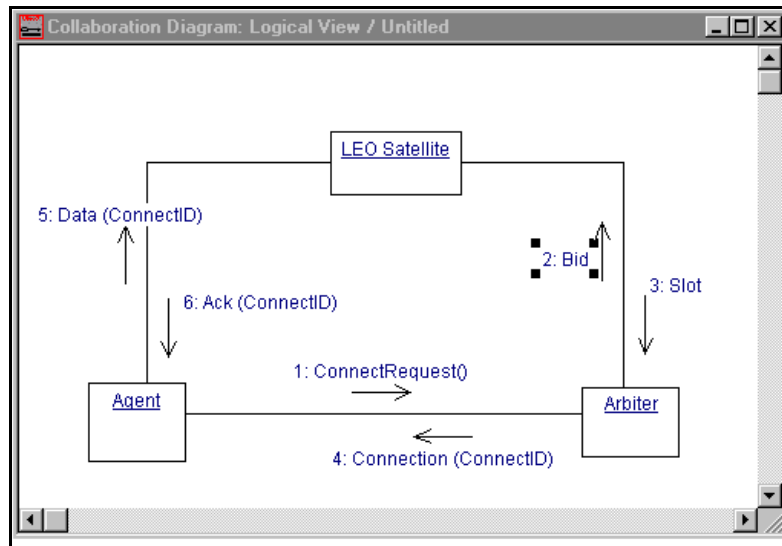


Figure 68 Collaboration Diagram Example

Sequence Diagrams

A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence—what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces. Sequence diagrams are normally associated with use cases.

This type of diagram is best used during early analysis phases in design because they are simple and easy to comprehend. A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

Sequence diagrams are closely related to collaboration diagrams and each are alternate representations of an interaction.

A sequence diagram traces the execution of a scenario in time. The following example shows a sequence diagram:

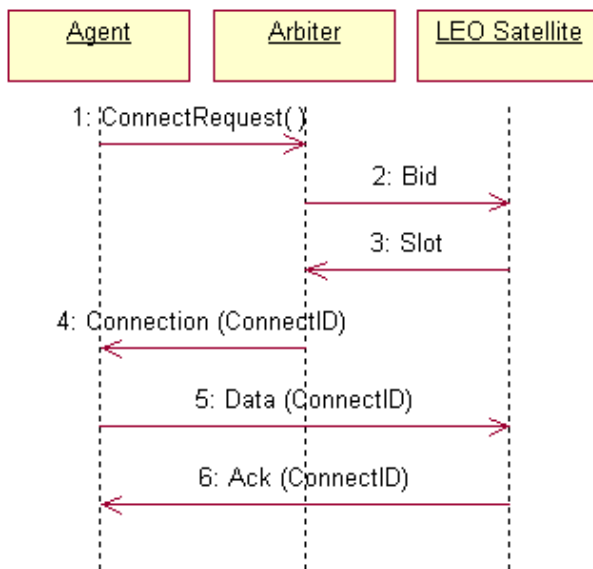


Figure 69 Sequence Diagram Example

Toolboxes

Each diagram type has its own unique toolbox. The collaboration and sequence diagram toolboxes are illustrated in this section.

Collaboration Diagram Toolbox

The graphic below shows all the tools that can be placed on the collaboration diagram toolbox. See “Customizing the Toolbox” on page 14. for information on adding or deleting tools on a diagram toolbox.

The application window displays the following toolbox when the current window contains a collaboration diagram and you have selected **View > As Unified**.

Note: Some icons will be different if you have selected **View > As Booch** or **View > As OMT**.

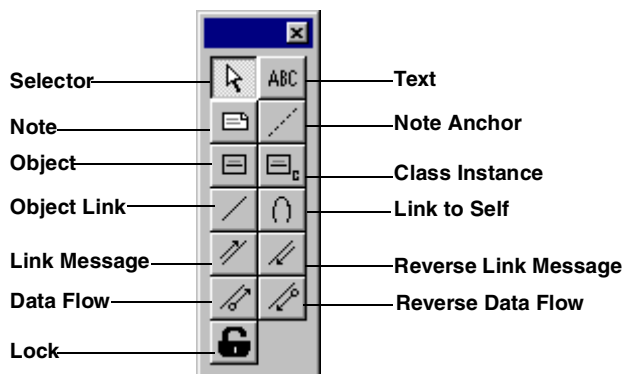


Figure 70 Collaboration Diagram Toolbox

Sequence Diagram Toolbox

The graphic below shows all the tools that can be placed on the sequence diagram toolbox. See “Customizing the Toolbox” on page 14. for information on adding or deleting tools on a diagram toolbox.

The application window displays the following toolbox when the current window contains a sequence diagram and you have selected **View > As Unified**.

Note: Some icons will be different if you have selected **View > As Booch** or **View > As OMT**.

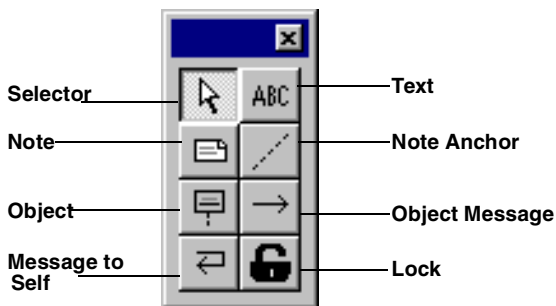


Figure 71 Sequence Diagram Toolbox

Note: The object and message icons are also found in the collaboration toolbox.

Common Collaboration and Sequence Diagram Icons

There are a number of common tools that are used on both collaboration and sequence diagrams. Although they may look slightly different, they illustrate common concepts or elements. Tools unique to a specific diagram type are discussed after this section.

Object

One of the primary elements of a collaboration or sequence diagram is an object. An object has state, behavior, and identity. The structure and behavior of similar objects are defined in their common class. Each object in a diagram indicates some instance of a class. An object that is not named is referred to as a class instance.

The object icon is similar to a class icon except that the name is underlined.

If you use the same name for several object icons appearing in the same collaboration diagram, they are assumed to represent the same object; otherwise, each object icon represents a distinct object. Object icons appearing in different diagrams denote different objects, even if their names are identical. Objects can be named three different ways: object name, object name and class, or just by the class name itself.

Multiple Objects

If you have multiple objects that are instances of the same class, you can modify the object icon by clicking **Multiple Instances** in the **Object Specification**. When you select this check box, the icon is changed from one object to three staggered objects.

To create an icon representing multiple objects:

1. Create an object.
2. Double-click on its icon to display its specification.
3. Click the **Multiple Instances** check box.
4. Click **OK**.

Rational Rose displays the **Multiple Object** icon:

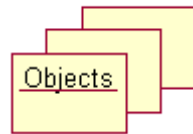


Figure 72 Multiple Object Diagram

Messages

A message icon represents the communication between objects, indicating that an action will follow.

Each message icon represents a message passed between two objects, and indicates the direction of message is going. A message icon in a collaboration diagram can represent multiple messages. A message icon in a sequence diagram represents exactly one message.

A message is the communication carried between two objects that triggers an event. A message carries information from the source focus of control to the destination focus of control.

A message is represented on collaboration diagrams and sequence diagrams by a message icon which visually indicates its synchronization. The synchronization of a message can be modified through the message specification.

If all messages represented by a message icon do not have the same synchronization, the “simple” message icon is displayed. You can change the synchronization of the message by editing the message specification.

The sequence diagram toolbox contains two message tools. The message icon tool appears as a horizontal arrow. The message to self icon appears as a message that returns to itself.

The collaboration diagram toolbox contains two message tools. The forward message tool, bearing an arrow pointing “northeast,” places a message icon from client to supplier. The reverse message tool, bearing an arrow pointing “southwest,” places a message icon from supplier to client. The default synchronization for a message is “simple.”

Scripts may be attached to messages to enhance the messages.

If a message is deleted, the link on the collaboration diagram remains intact.

To create a client-to-supplier message and assign it to a link between two objects (collaboration diagram only):

1. Click the **Message** icon.
2. Click on an icon representing the link.

Rational Rose creates an unnamed, empty message assigned to the designated link. The source of this message is the client object, and the destination of this message is the supplier object.

To create a supplier-to-client message, use the **Reverse Message Creation** tool in the above procedure. The source of the resulting message is the supplier-object, and the destination of this message is the client-object.

To name an unnamed message:

1. Click the icon representing the message.
2. Type the name.
3. Click outside the named icon.

Rational Rose will name the message as specified, and assign it a sequence number based on creation order, starting with 1.

To Change Message Names in Interaction Diagrams:

1. Click on the name to display a flashing vertical bar that designates the insertion point.
2. Enter additional text.
3. Click outside the named icon.

Alternatively, you can double-click on an icon representing the message to display the message specification; modify the **Name** field and click **OK**.

Message Numbering

To enable or disable the display of message numbers click **Tools > Options**. Click on the **Diagram** tab and click **Collaboration Numbering** (for collaboration diagrams) or **Sequence Numbering** (for sequence diagrams).

To Change Messages Numbering in Interaction Diagrams:

1. Create or display the interaction's sequence diagram, click **Browse > Create Sequence Diagram** or **Browse > Go to Sequence Diagram**.
2. Reorder the messages by dragging the message icons into the preferred order.
3. Redisplay the interaction diagram by clicking **Browse > Go to Collaboration Diagram** or **Browse > Go to Sequence Diagram**.

Assigning an Operation to a Message

Rational Rose enables you to assign an operation to a message by presenting a list of all operations accepted by the destination object. The list of valid operations is defined by the specification of the object's parent class, and the specifications of the parent class' superclasses, as specified by its inheritance hierarchy. The **Class** field of the destination object's specification must therefore be set to identify the destination object's parent class before operations can be assigned to a message to that destination object.

Assigning an operation to a message changes the name of the message to the name of the operation.

To assign an operation to a message:

1. Right-click on the message icon.
2. Click an operation from the pop-up list, or click **<new operation>** to add and specify a new operation to the destination object's class specification.

If you click **<new operation>**, you must repeat this procedure after specifying the new operation to assign the newly-created operation to the message.

You can associate multiple messages with a message icon. Each new message is represented by an independent name and sequence number. If a message icon represents multiple messages, you must select a specific message by clicking on its name.

You can also create multiple messages associated with the same message icon using the link specification. This method is described in the *Link Specification* section in this chapter.

To change a message's assigned operation, display its specification by double-clicking on its message icon. If the message icon represents multiple messages, double-click on the name of the message whose operation you select to change. Select the desired operation from the specification's **Referenced Operation** field, or directly enter an operation name in the **Name** field.

Collaboration Specific Toolbox Icons

Links

Objects interact through their links to other objects. A link is an instance of an association, analogous to an object being an instance of a class.

A link should exist between two objects, including class utilities, only if there is a relationship between their corresponding classes. The existence of a relationship between two classes symbolizes a path of communication between instances of the classes: one object may send messages to another.

Links can support multiple messages in either direction. If a message is deleted, the link remains intact.

The link is depicted as a straight line between objects or objects and class instances in a collaboration diagram. If an object links to itself, use the loop version of the icon.

To create a link between two objects:

1. Click the **Link** tool.
2. Drag the pointer between the two object icons.

Rational Rose will create and display an unnamed link.

To create a reflexive link—a link between an object and itself:

1. Click the **Link to Self** tool.
2. Click on an icon representing the object.

Rational Rose will create and display an unnamed reflexive link.

Sequence Numbering

Sequence numbering allows you to clearly see how messages interact and relate to one another. Numbering messages can be done two ways on sequence diagrams: top level numbering; in a 1, 2, 3, pattern or hierarchical numbering; in a 1.1, 1.1.2, 1.1.3, pattern. Only top level numbering is available on collaboration diagrams. However, if you create a collaboration diagram from a sequence diagram with hierarchical numbering the hierarchical numbering is retained.

Top-Level Numbering

Top-level numbering gives each message or message to self a single number. There are no number subsets. Top-level numbering is useful in small sequence diagrams with few objects and messages.

Hierarchical Numbering

Hierarchical numbering bases all messages on a dependent message. For example, you could have messages numbered 1., 1.1, 1.2, 1.2.1, where message number 1 is an independent message. All other message numbers numbered 1.x and beyond are dependent on message 1. If you remove independent message 1 from the diagram, all dependent messages will be removed.

To Display Hierarchical Numbering:

1. Click **Options** from the **Tools** menu.
2. Click the **Diagram** tab.
3. Click on the **Sequence Numbering** check box.
4. Click on the **Hierarchical Messages** check box.

Scripts

Scripts are used to enhance messages on sequence diagrams. They are simple text fields that attach to messages.

To create and attach a script:

1. Click on the message icon and drag between two objects.
2. Create text by either:
 - Using the **ABC** icon.

- Clicking **Text** from the **Create** option on the **Tools** menu.
- 3. Select one or more labels.
 - Press the CTRL or SHIFT key to enable multiple selections.
- 4. Select ONE message.
 - The order of steps 2 & 3 is inconsequential.
- 5. Click **Edit > Attach Script**.
 - The script is now attached to the message.

To move a script:

1. Select the message and move with the drag technique.
 - The script moves next to the message.
2. Select only the script and move with the drag technique.
 - The script moves independently of the message.

To detach a script:

1. Select either script or message.
2. Click **Edit > Detach Script**.

To delete a(n):

1. Message
 - All dependent messages and attached scripts are deleted.or...
2. Script
 - Script is deleted with no effect on the messages.or...
3. Object
 - All messages and attached scripts are deleted.

To undo:

1. Click **Edit > Undo** will reverse the latest change.

Focus of Control

Focus of Control (FOC) is an advanced notational technique that enhances sequence diagrams. This technique shows the period of time during which an object is performing an action, either directly or through an underlying procedure.

FOC is portrayed through narrow rectangles that adorn lifelines (the vertical lines descending from each object). The length of a FOC indicates the amount of time it takes for a message to be performed. When you move a message vertically, each dependent message will move vertically as well. Also, you can move a FOC vertically off the source FOC to make it detached and independent.

A sequence diagram with FOC notation and scripts follows:

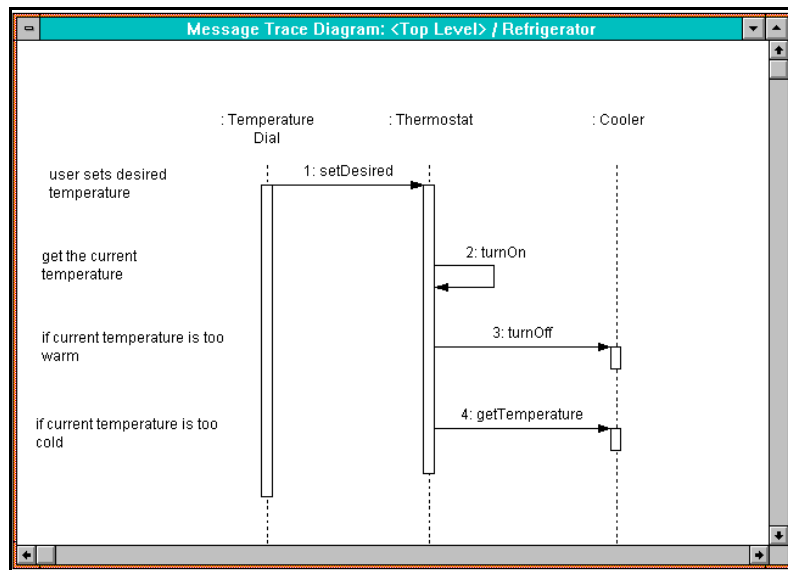


Figure 73 Focus of Control Diagram Example

Displaying Focus of Control

To enable or suppress the Focus of Control notation on a sequence diagram:

1. Click **Tools > Options**.

2. Click the **Diagram** tab.

A check mark beside Focus of Control means that the notation will be displayed.

Coloring Focus of Control

To help distinguish a particular FOC from other items in a sequence diagram, you can fill a FOC with a color. To color a FOC:

1. Select the message icon that enters the FOC you want to color.
2. Click **Format > Fill Color**.
3. Click on the color you want to make the selected FOC.
4. Click **OK**.

Moving the Focus of Control

Sometimes it is helpful to move the starting point of a FOC and all corresponding messages. If the FOC has an entry point message, you can move the message. Otherwise, use the following steps to move the FOC on a sequence diagram:

1. Select the first message from the FOC you want to move.
2. Press the ALT key.
3. Grab the source message and move it to the desired location on a sequence diagram. Note how the source FOC changes locations.

Nested Focus of Control

A Nested Focus of Control is a FOC that resides on another FOC. Nested FOC allows you to distinguish exactly where a message starts and where it ends. If you want to add a message to an existing sequence diagram, the Nested FOC feature helps you determine where to place it.

Creating Alternative Diagrams

The **Create Collaboration Diagram** command creates a collaboration diagram from information contained in the sequence diagram. The **Create Sequence Diagram** command creates a sequence diagram from

information contained in the interaction's collaboration diagram. The **Go to Sequence Diagram** and **Go to Collaboration Diagram** commands traverse between an interaction's two representations.

Toggling between Interaction Diagrams

When you work in either a collaboration or sequence diagram, it is possible to view the corresponding diagram by pressing the **F5** key. For example, if you are working on a sequence diagram, you can press **F5** and Rational Rose will automatically create a collaboration diagram with the same diagram name and model elements. If you make a change to one diagram and then press **F5**, the change will appear on the corresponding diagram as well.

***Note:** When toggling from a sequence diagram to a collaboration diagram, you may need to rearrange the collaboration diagram model elements.*

Creating a Collaboration Diagram from a Sequence Diagram

To create a collaboration diagram from a sequence diagram, click anywhere on the sequence diagram and click **Create Collaboration Diagram** from the **Browse** menu. Note that if this collaboration diagram already exists, the **Browse** menu will instead present the **Go to Collaboration Diagram** option.

Creating a Sequence Diagram from a Collaboration Diagram

To create a sequence diagram from a collaboration diagram, click anywhere on the collaboration diagram and click **Create Sequence Diagram** from the **Browse** menu. Note that if this sequence diagram already exists, the **Browse** menu will instead present the **Go to Sequence Diagram** option. Class instances in the collaboration diagram are represented as objects in the sequence diagram.

Object Specification

An object specification enables you to display and modify the properties and relationships of an object in the current model.

To display an **Object Specification**, double-click on any icon representing an object or on the **Browse** menu, click **Specifications**.

Specification Content

The **Object Specification** consists of the following tab: **General**.

Object Specification—General Tab

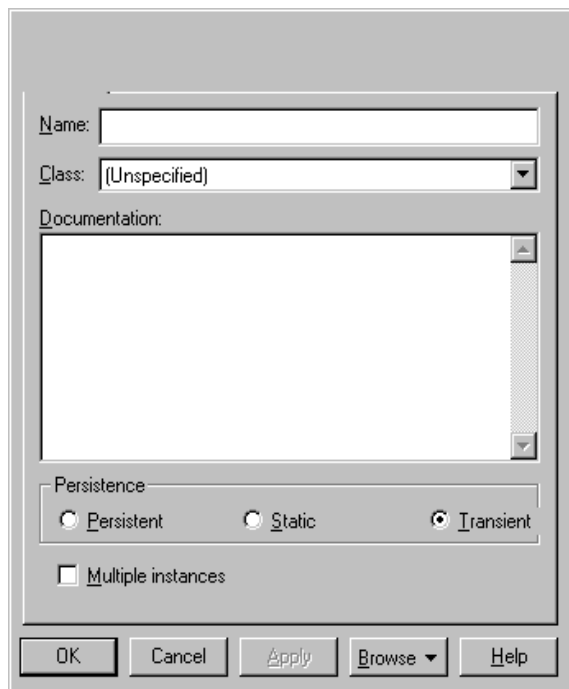


Figure 74 *Object Specification—General Tab*

Refer to the description in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Name

If you specify the name of the object's class in the **Object Specification**, the name must identify a class defined in the model.

Class

The **Class** field displays the name of the object's parent class. The default class for a newly created object is **Unspecified**.

The object will accept messages conveying the operations of its parent class, and the operations of the superclasses of its parent class.

If you subsequently delete the class from the model, its name will be displayed in parentheses. If you recreate the class or create a new class with the same name, the object becomes an instance of this class.

Persistence Field

Use these options to specify the object's persistence:

Table 12 Persistence Field Options

Type	Description
Persistent	The object exists after the termination of the program in which it was created.
Static	The object exists during the entire execution of a program.
Transient	The object is created and destroyed dynamically during the execution of a program.

To display an object's persistence in a collaboration diagram, right-click on an icon representing the object to display the shortcut menu and click **Show Persistence**.

Multiple Instances Check Box

Check the **Multiple Instances** check box to indicate that this object represents multiple instances of the same class. When you select this field, the icon changes from one object to three staggered objects. The object group is considered one entity, but this icon indicates that several objects are involved.

Class Instance Specifications

A class instance places a representation of a class on a collaboration diagram.

To display an **Class Instance Specification**, double-click on any icon representing an class instance or on the **Browse** menu, click **Specifications**.

Specification Content

The **Class Instance Specification** consists of the following tab: **General**.

Class Instance Specification—General Tab

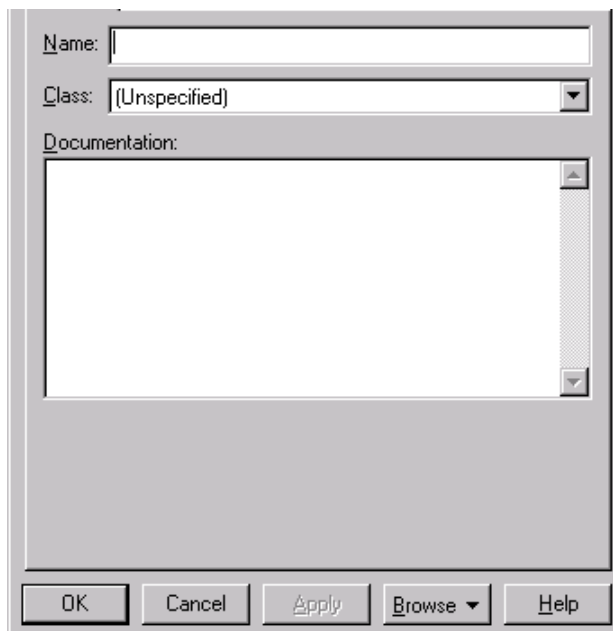


Figure 75 *Class Instance Specification—General Tab*

Refer to the descriptions earlier in this chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Class

The class the element belongs to is displayed here. The default class for a newly created element is **(Unspecified)**. If you specify an object's class in the **Object Specification**, the class name must identify a class defined in the model, or you may create a new class.

To create a new class through the **Object Specification**, click the scroll arrow on to the right of the **Class** field. A list box will display all the possible class selections, including **<New>**. Double-click on **<New>**. A **Class Specification** dialog box is displayed. Enter the information regarding the new class.

If you delete a class from the model after you have associated it with one or more objects, the class name is enclosed in parentheses. If you re-create the class or create a new class with the same name, the object becomes an instance of the new class.

You can set this field only through the dialog box.

Link Specification

A link is the path of communication between two objects. A link can exist between two objects, between an object and a class instance, or between an object and itself.

To display a **Link Specification**, double-click on any icon representing a link or click **Browse > Specifications**.

Specification Content

The **Link Specification** consists of the following tabs: **General** and **Messages**.

Link Specification—General Tab

The screenshot shows a dialog box titled "Link Specification—General Tab". It contains the following elements:

- A text field labeled "Name:" with an empty input box.
- A dropdown menu labeled "Assoc:" with "unspecified" selected.
- Two panels for visibility settings:
 - Supplier visibility:** Radio buttons for "Unspecified" (selected), "Field", "Parameters", "Local", and "Global".
 - Client visibility:** Radio buttons for "Unspecified" (selected), "Field", "Parameters", "Local", and "Global".
- Two checkboxes labeled "Shared" for the supplier and client sides.
- Two text labels labeled "Role:" positioned below the "Shared" checkboxes.
- A row of buttons at the bottom: "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

Figure 76 Link Specification—General Tab

Refer to the description in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Assoc

The **Association** field lists any valid role(s) or association(s) tied to the classes belonging to the two objects.

Select an association from the drop-down list. The name of the role tied to the association is displayed beside the link on the diagram. The keys are displayed in brackets under the role, and the constraints are displayed in braces under the keys.

Supplier & Client Visibility

Visibility is the ability of one object to see another object.

You can specify the following visibility types for the supplier object, the client object or both:

Table 13 Supplier & Client Visibility Options

Type	Description
Unspecified (Default)	The object visibility has not been specified.
Field	The supplier object is visible because it is a field of the client.
Parameters	The supplier object is visible to the client because it is a parameter for one of the client's operations.
Local	The supplier is local to an operation of the client object.
Global	The supplier object is global to client.

An object visibility adornment is a letter inside a box placed at the supplier end of the link. Each letter identifies the type of visibility used. The adornment box is either open (shared) or filled (unshared).

You can set link visibility through the **Link Specification** or through the shortcut menu. These fields correspond to visibility adornments displayed in the collaboration diagram.

- To set visibility for the supplier object, click on a visibility type in the Supplier Visibility section.
- To set visibility for the client object, click on a visibility type in the Client Visibility Section.

The visibility adornment is placed at the appropriate end of the link. The unspecified object visibility does not have a corresponding visibility adornment. Use this adornment only when you need to document an important tactical decision.

Shared

If visibility is an important detail in your software model, use visibility adornments to show these details in a collaboration diagram.

Shared visibility indicates structural sharing of the given object; that is, the shared object's state can be altered through more than one path. Unshared visibility represents unique access given to the client object. When you create a link, unshared visibility is the default.

You can set the shared indicators in the **Link Specification** or by selecting a visibility value from the shortcut menu.

To toggle the shared indicator, click on the **Shared** check box below the appropriate visibility section.

- If you specify shared visibility, the visibility adornment changes from a filled square to an open square of the corresponding type.
- If you specify unshared visibility, the visibility adornment changes to a filled square of the corresponding type.

Role

This field lists the role names tied to the selected associations. This is especially useful since many associations are not named. This field cannot be edited.

Note: *The Link to Self Specification contains only the Name, Visibility and Shared elements.*

Link Specification—Messages Tab

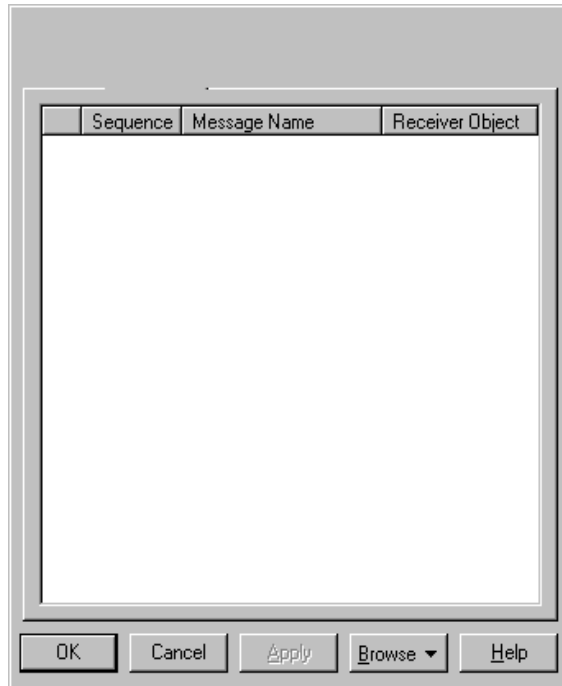


Figure 77 Link Specification—Message Tab

Refer to the description in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

You can add a message either directly on the diagram or through **Insert** on the shortcut menu.

Icon

This left-most unlabeled field contains a small version of the link message icon indicating the direction of the message.

Sequence

This is a system-assigned, sequential message number.

Message Name

Click on the item to see the Picklist box showing all available operations on the class. This is the only editable column on this tab.

Receiver

This is the object receiving the message.

Note: You can double-click on every field except the icon field to display the message specification.

Message Specification

A message conveys an operation through a link between objects. A message's specification identifies the operation it conveys, its synchronization, its frequency, and its associated documentation.

To display a **Message Specification**, double-click on any icon representing a message or on the **Browse** menu, click **Specifications**.

Specification Content

The **Message Specification** consists of the following tabs: **General** and **Detail**.

Message Specification—General Tab

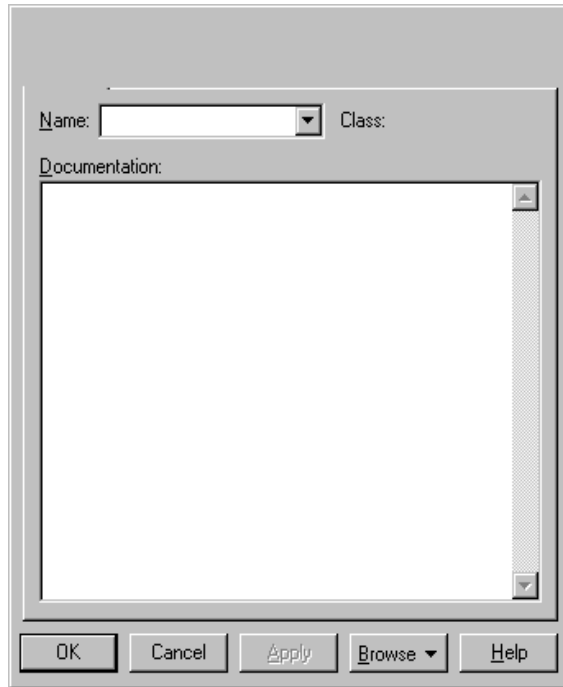


Figure 78 *Message Specification—General Tab*

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the elements not covered in the following section.

Class

The **Class** field displays the name of the class to which the element belongs.

Message Specification—Detail Tab

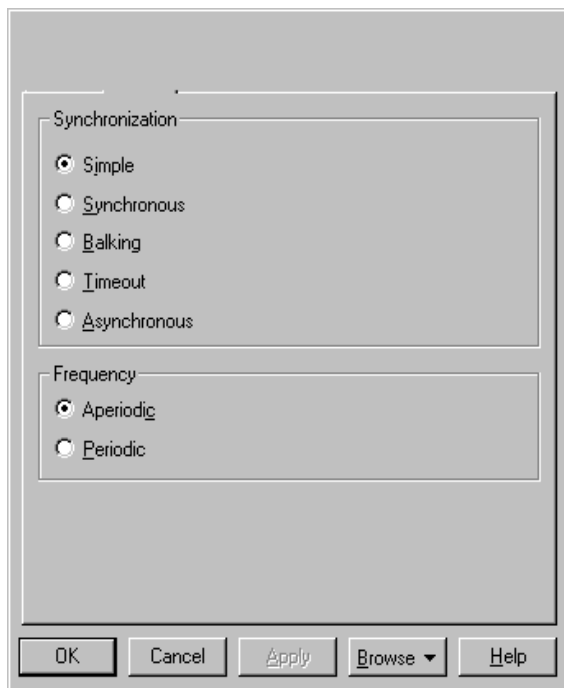


Figure 79 *Message Specification—Detail Tab*

Refer to the description in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Synchronization

Use these options to specify concurrency semantics for the operation named in the **Synchronization** field:

Table 14 Synchronization Options

Type	Description
Simple (default)	The message has a single thread of control.
Synchronous	The operation proceeds only when the client sends a message to the supplier and the supplier accepts the message.
Balking	The client passes a message only if the supplier is immediately ready to accept the message; the client abandons the message if the supplier is not ready.
Timeout	The client abandons a message if the supplier cannot handle the message within a specified amount of time.
Asynchronous	The client sends a message to the supplier for processing and continues to execute its code without waiting for or relying on the supplier's receipt of the message.

Frequency

Use these options to indicate whether the message is sent periodically or aperiodically:

Table 15 Frequency Options

Type	Description
Aperiodic	The message is sent at irregular intervals, or does not have a regular interval.
Periodic	The message is sent at regular intervals.



Component Diagram Overview

A component diagram shows the physical dependency relationships (mapping to a file system) between components—main programs, subprograms, packages, and tasks—and the arrangement of components into component packages.

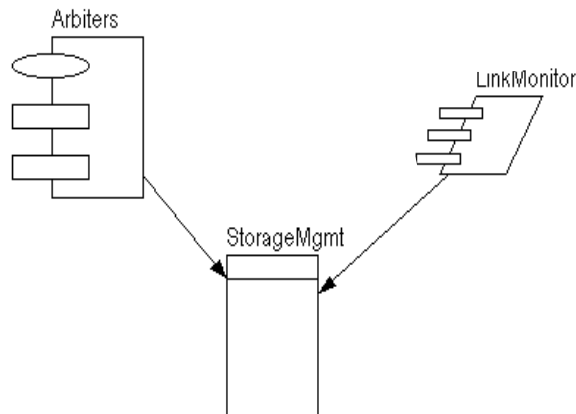


Figure 80 *Component Diagram Example*

Component diagrams are contained (owned) either at the top level of the model, or by a package. This means the diagram will depict the components and packages where the diagram is contained.

Creating and Displaying a Component Diagram

You can create or display the component diagram in one of three ways:

- Click **Browse > Component Diagram**.
- On the toolbar, double-click the component diagram icon.
- On the browser, double-click on the component diagram icon.

Component Diagram Toolbox

The application window displays the following toolbox when the current window contains a component diagram and **As Unified** is selected from the **View** menu.

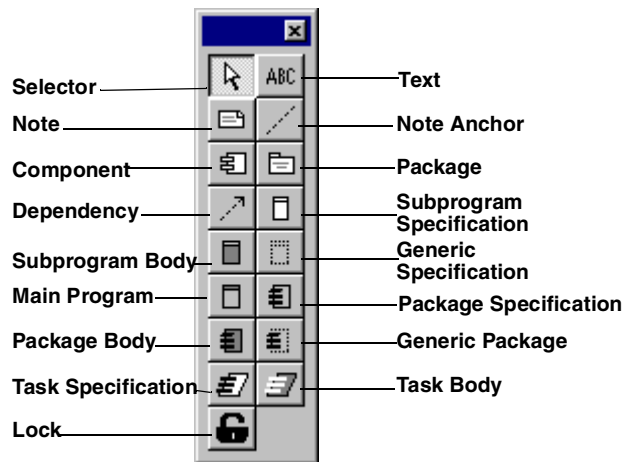


Figure 81 Component Diagram Toolbox

Assigning a Component to Another Package

Every component is assigned to a package. When you create a component using a creation tool from the component diagram toolbox, the component is assigned to the package containing the component diagram.

To reassign a component from one package to another:

1. Select a component icon in a diagram directly contained by the package to which the component should be assigned. (You might need to create such a diagram or icon if one does not currently exist.)
2. Click **Edit > Relocate**.

Rational Rose will update all component diagrams to reflect the component's new assignment.

Like components, packages are also assigned to packages, permitting nesting to an arbitrary depth. The mechanisms previously described can be applied to packages as well as components.

Component Specifications

A **Component Specification** displays and modifies the properties and relationships of each component in the current model. The same specification is used for all component kinds.

Some of the information on this specification can also be displayed inside icons representing the component in a component diagram.

To display a **Component Specification**, double-click on any icon representing the component or on the **Browse** menu, click **Specifications**.

Specification Content

The **Component Specification** consists of the following tabs: **General**, **Detail**, **Realizes**, and **Files**.

Component Specification—General Tab



Figure 82 *Component Specification—General Tab*

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the elements shown on this specification tab.

Stereotype (Component)

A component stereotype represents the subclassification of an element. The most common type of components are already predefined as stereotypes, including Main Program, Package Body, Package Specification, Subprogram Body, Subprogram Specification, Task Body and Task Specification. You can also define and add your own kinds of stereotypes.

Language

This field identifies the implementation language that is assigned to this component.

Note that when changing the implementation language of a component, the data types that are used in the specification of operations and attributes of the assigned classes, are not automatically converted to data types in the new implementation language. Also, if you change the implementation language for a component with classes that are assigned to other components, a dialog box is displayed where you have to decide how to handle those classes.

Component Specification—Detail Tab



Figure 83 Component Specification—Detail Tab

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Declarations

The **Declarations** field contains a list of declarations, such as class names, variables, and other language-specific features (such as **#includes** or similar constructs). Declarations can include classes, objects and any other language-specific declarations.

Use this field to list the elements that physically reside in the component. You can view this field only through the component specification.

Component Specification—Realizes Tab



Figure 84 Component Specification—Realizes Tab

Refer to description in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Show all Classes

Select this check box if you want to view a list of all classes in the model. If this option is not selected, you will see only the classes that are assigned to this component.

Classes

The list identifies the classes and interfaces that are assigned to this component (indicated with check marks). The Logical Package column shows to which package a class belongs, and the Language column shows the programming language that is assigned to a specific class.

You assign a class or interface to a component through **Assign** on the shortcut menu in the list, or by dragging a class or interface from the browser and dropping it in this list. Note that you can only assign classes that are unassigned, or classes that are assigned to components with the same implementation language as this component.

Language

This field identifies the implementation language that is assigned to this component.

Note that when the changing the implementation language of a component, the data types that are used in the specification of operations and attributes of the assigned classes, are not automatically converted to data types in the new implementation language. Also, if you change the implementation language for a component with classes that are assigned to other components, a dialog box is displayed where you have to decide how to handle those classes.

Component Specification—Files Tab

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on this tab.

Package Specification

A **Package Specification** displays and modifies the properties and relationships of a package in the current model.

To display a **Package Specification**, double-click on any icon representing the package or on the **Browse** menu, click **Specifications**.

The **Package Specification** consists of the following tabs: **General**, **Detail**, **Realizes** and **Files**.

Package Specification—General Tab

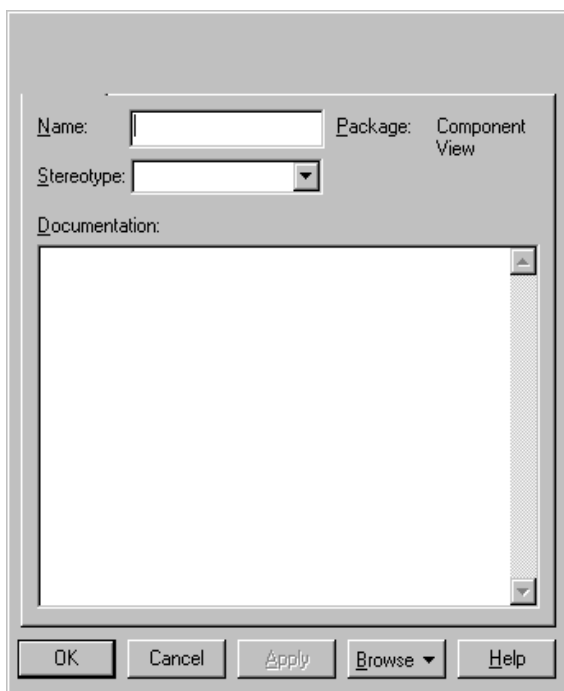


Figure 85 *Package Specification—General Tab*

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the elements of this specification not covered in the following section.

Package

The package the component belongs to is displayed in this static field.

Package Specification—Detail Tab

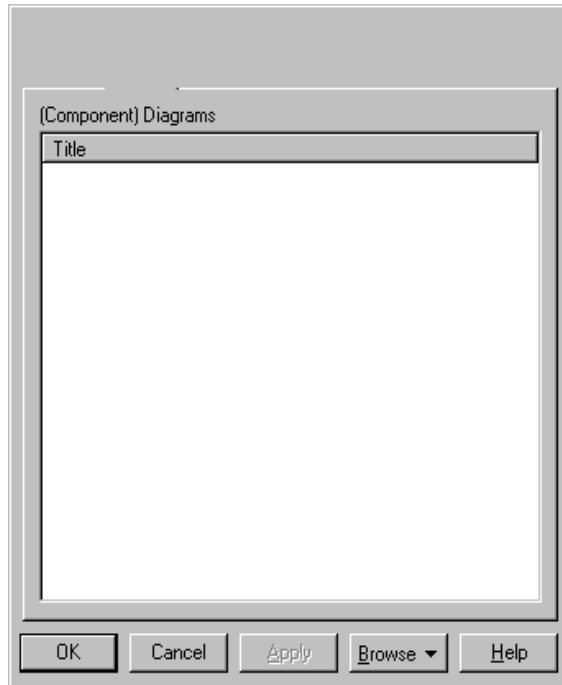


Figure 86 *Package Specification—Detail Tab*

Component Diagrams

This field lists the component diagrams contained in the package. You can create a new component diagram in the package through **Insert** on the shortcut menu, or on the **Browse** menu, click **Component Diagram**. You may rename or delete existing component diagrams from this field.

To display a specific component diagram listed in this field, double-click on its entry.

Package Specification—Realizes Tab

Refer to description earlier in this chapter if you need information on this tab.

Package Specification—Files Tab

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on this tab.



Deployment Diagram Overview

A deployment diagram shows processors, devices, and connections. Each model contains a single deployment diagram that shows the connections between its processors and devices, and the allocation of its processes to processors.

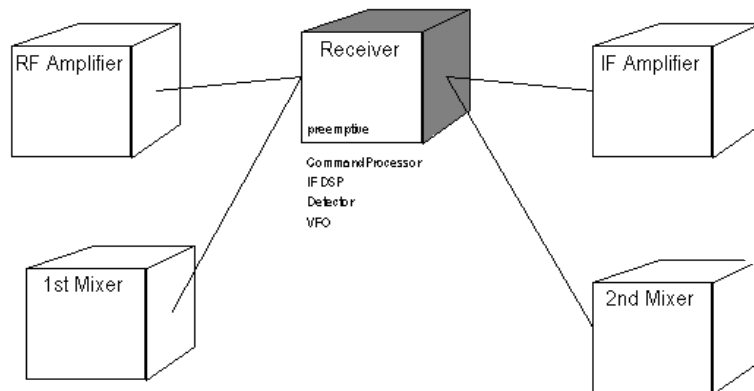


Figure 87 Deployment Diagram Example

Creating and Displaying a Deployment Diagram

You can create or display the deployment diagram in one of three ways:

- Click **Browse > Deployment Diagram**.
- On the toolbar, double-click on the deployment diagram icon.
- In the browser, double-click on the deployment diagram icon.

Deployment Diagram Toolbox

The application window displays the following toolbox when the current window contains a deployment diagram and you have selected **View > As Unified**:

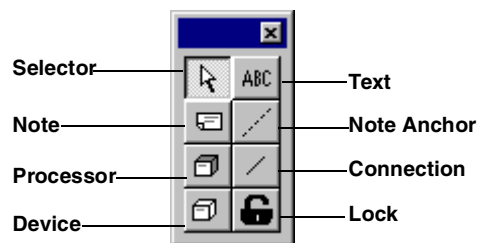


Figure 88 Deployment Diagram Toolbox

Processor Specification

A **Processor Specification** displays and modifies the properties and relationships of a processor in the current model. Some of the information on the specification can also be displayed inside icons representing the processor in a model's deployment diagram.

To display a **Processor Specification**, double-click on any icon representing a processor or click **Browse > Specifications**.

Specification Content

The **Processor Specification** consists of the following tabs: **General** and **Detail**.

Processor Specification—General Tab

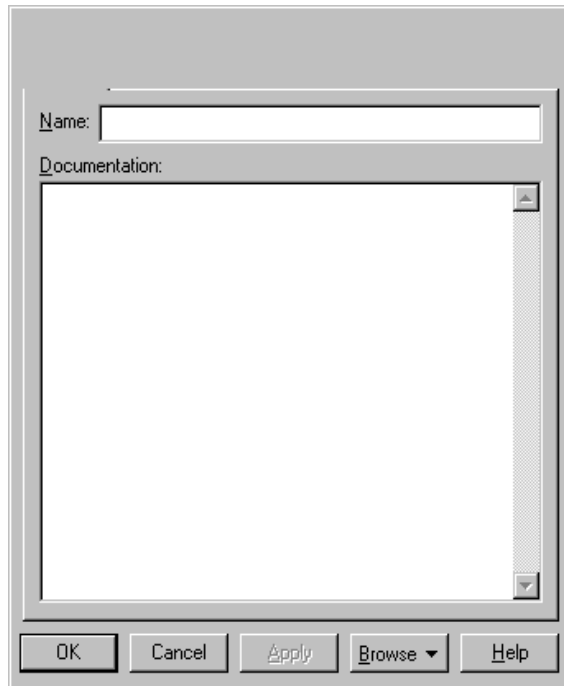


Figure 89 Processor Specification—General Tab

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the elements shown on this specification.

Processor Specification—Detail Tab

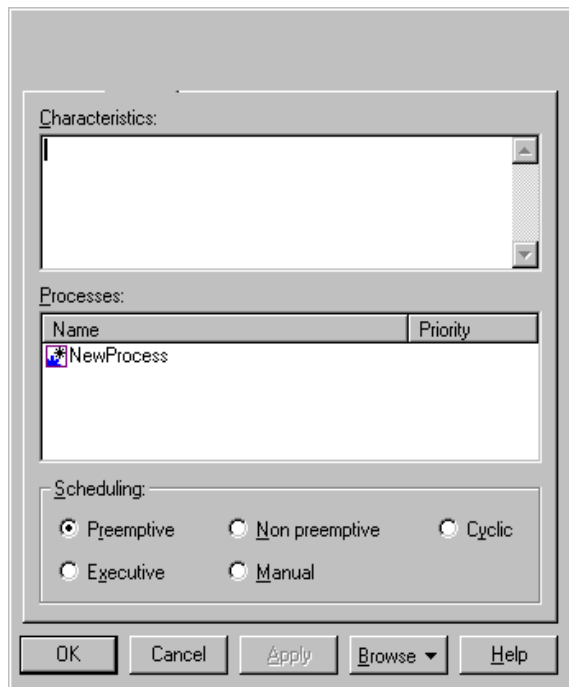


Figure 90 Processor Specification—Detail Tab

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Characteristics

Use the **Characteristics** field to specify a physical description of an element. For example, you can describe the kind and bandwidth of a connection, the manufacturer, model, memory, and disks of a machine, or the kind and size of a device. You can set this field only through the specification. This information is not displayed in the deployment diagram.

To update this field, click in the **Characteristics** field and enter the information.

Processes

Use this field to identify the processes assigned to this processor. Processes denote either the root of a main program from a component diagram or the name of an active object from a collaboration diagram.

To create a process, place the pointer in the Processes area and click **Insert** using the shortcut menu. A NewProcess entry is created. To change the name or priority, click the item and type the changes.

You can display a list of the processes by selecting the processor icon and clicking **Show Processes** from the shortcut menu.

Scheduling

The **Scheduling** field specifies the type of process scheduling used by the processor. Use these options to specify the appropriate scheduling:

Table 16 Scheduling Field Options

Type	Description
Preemptive	Higher-priority processes that are ready to execute can preempt lower-priority processes that are currently executing. Processes with equal priority are given a time slice in which to execute, allowing computation resources to be fairly distributed (default).
Non preemptive	The current process continues to execute until it relinquishes control.
Cyclic	Control passes from one process to another; each process is given a fixed amount of processing time.
Executive	An algorithm controls process scheduling.
Manual	Processes are scheduled by a user outside of the system.

You can set this field only through the specification. To set the scheduling type, click on the applicable option button in the **Scheduling** field.

You can display the scheduling type in the processor icon by clicking **Show Scheduling** from the shortcut menu.

Device Specification

A **Device Specification** displays and modifies the properties and relationships of a device in the current model. Some of the information on this specification can also be displayed inside icons representing the device in a deployment diagram.

To display a **Device Specification**, double-click on any icon representing a device or click **Browse > Specifications**.

Specification Content

The **Device Specification** consists of the following tabs: **General** and **Detail**.

Device Specification—General Tab

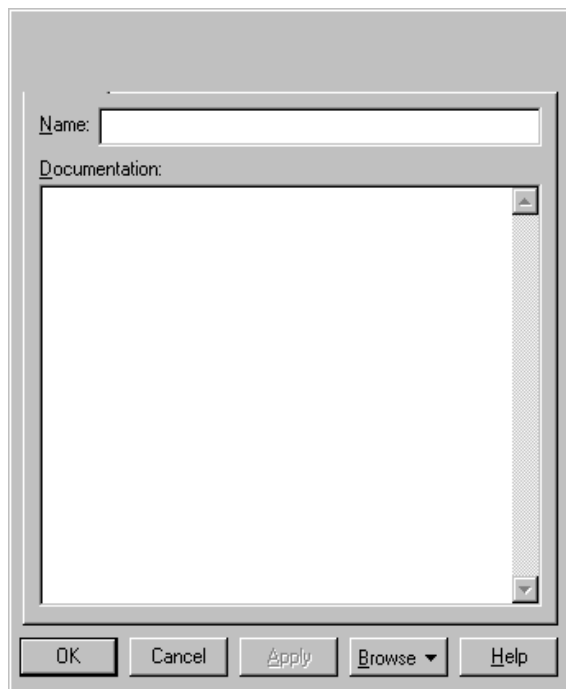


Figure 91 *Device Specification—General Tab*

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the elements shown on this specification.

Device Specification—Detail Tab

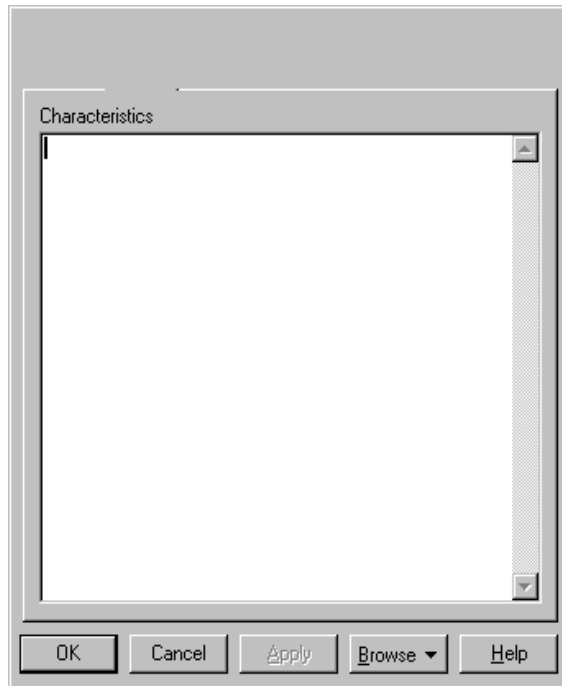


Figure 92 *Device Specification—Detail Tab*

Refer to the descriptions earlier in this chapter or in the *Introduction to Diagrams and Specifications* chapter if you need information on the elements shown on this specification.

Connection Specifications

A **Connection Specification** indicates a communication path between two processors, two devices, or a processor and a device. A connection usually represents a direct hardware coupling, such as an RS232 cable. It can also represent an indirect coupling.

To display a **Connection Specification**, double-click on any icon representing a connection or click **Browse > Specifications**.

The **Connection Specification** consists two tabs, which contain the same elements as the **Device Specification**, and is not repeated here.

Process Specification

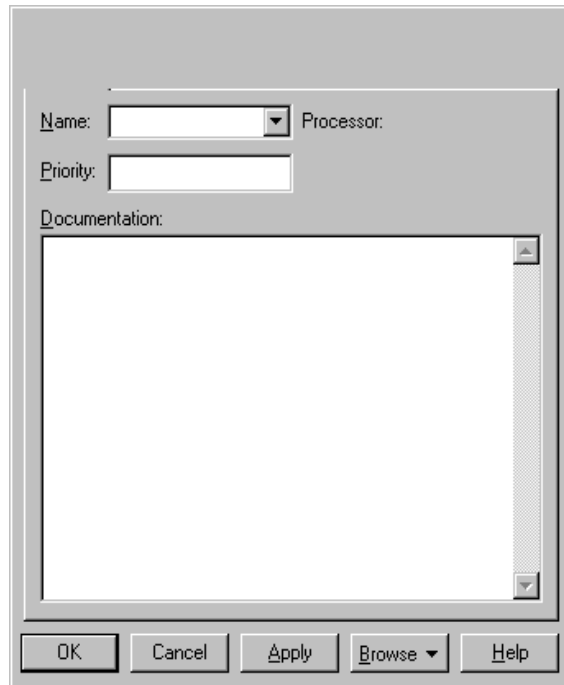
Processes are threads of control that execute on a processor. One process specification documents one thread of control.

You access the **Process Specification** through the **Processes** field of a **Processor Specification**. None of the information contained in the **Process Specification** is displayed in a diagram; thus, process properties can only be viewed and modified through a **Process Specification**.

Specification Content

The **Process Specification** consists of the following tabs: **General**.

Process Specification—General Tab



The image shows a dialog box titled "Process Specification—General Tab". It contains the following fields and controls:

- Name:** A text input field with a dropdown arrow on the right.
- Processor:** A text input field.
- Priority:** A text input field.
- Documentation:** A large text area with a vertical scrollbar on the right side.
- Buttons:** A row of five buttons at the bottom: "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

Figure 93 *Process Specification—General Tab*

Refer to the descriptions in the *Introduction to Diagrams and Specifications* chapter if you need information on the specification elements not covered in the following section.

Processor

The owner of the process is shown here.

Priority

This field contains the relative priority of this process, if there is one. You can use this information with the scheduling type identified in the **Processor Specification** to schedule process execution.



Chapter 12

Stereotypes

Overview

A stereotype is a modeling element subclassification that has been given a more specific meaning. Stereotypes can be applied to:

- Packages
- Components
- Classes
- Use Cases
- Attributes and Operations
- Association, Generalization, and Dependency Relationships
- Processors, Devices, and Connections
- Objects (on activity diagrams)
- Activities
- State

A stereotype can be depicted by either a name or by an icon.

Benefits to Using Stereotypes

A stereotype allows you to provide additional distinctions in your model that are not explicitly supported by the UML. The use of stereotypes:

- Allows for customization of your development process
- Provides mnemonic help and visualization aids
- Allows you to make presentations with more pizzazz

User-Defined Stereotypes

Some stereotypes are predefined, but you can also define your own to add new kinds of modeling types. User-defined stereotypes are defined in a stereotype configuration file. For each stereotype, you can customize icons to be displayed in diagrams, in the browser, and in the toolbox.

Rational Rose offers ten stereotype icons that you can use when modeling a business:

- Business Use Case
- Use-Case Realization
- Boundary Class
- Business Actor
- Business Entity
- Business Worker
- Entity Class
- Control Class
- Business Use Case Realization
- Organization Unit Package

For more information on these icons, refer to the Rational Unified Process.

Viewing Stereotypes

You can control how stereotypes are displayed. The settings are found on the **Diagram** tab and the **Browser** tab, located in the **Options** dialog box under the **Tools** menu.

Diagram Tab

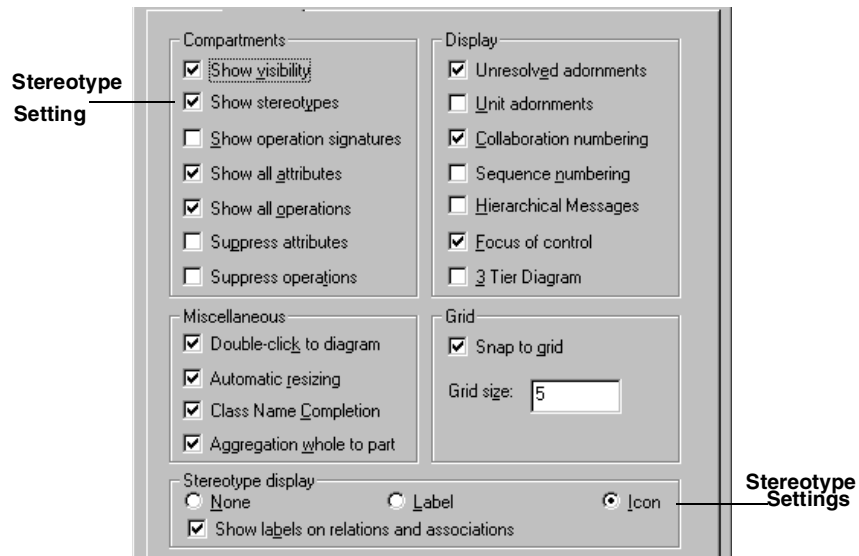


Figure 94 Options Dialog Box—Diagram Tab

The following selections are applied to new model elements are added to the diagrams. To make changes to existing model elements, use the shortcut menu.

Compartments—Show Stereotypes

This option allows you to control the display of stereotype names for operations and attributes of new classes in the class compartments.

Stereotype Display—None, Label, Decoration, Icon

These options allow you to control the display of stereotypes (except for relationships) in diagrams. The selection is applied to new model elements that are added to diagrams.

Selecting **Label**, makes the name of the stereotype appear inside angled brackets, << >>. Decorating displays a graphic marker such as highlighting an icon or tool. Selecting **None** means the name is not displayed, while selecting **Icon** means the icon created for that stereotype is displayed.

Stereotype Display—Show labels on relations and associations

This option allows you to control the display of stereotype labels on new relationships in diagrams. If enabled, the name of the stereotypes will appear inside angle brackets, << >>.

Browser Tab

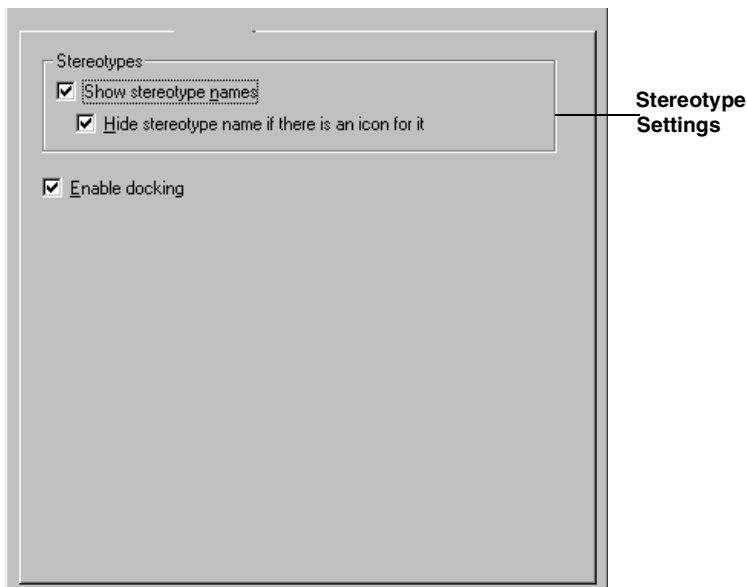


Figure 95 Options Dialog Box—Browser Tab

The changes made on the **Browser** tab are reflected in the browser.

Show stereotype names

This option allows you to control the display of stereotype names of model elements in the browser.

Hide stereotype name if there is an icon for it

Select this option to display stereotype icons, but not stereotype names, of model elements in the browser. This option is only relevant for stereotypes that have an icon.

Creating Stereotypes

Creating a New Stereotype for the Current Model

You can create a new stereotype by typing a new name in the **Stereotype** field of a model element's specification. The new stereotype will then be available in the **Stereotype** field for all model elements of that type (which are assigned the same language), in the current model.

If you want the stereotype to be available in all Rose models, follow the steps below. If you already have a stereotype configuration file, skip to the section titled *Creating a New Stereotype for All Rose Models*.

Creating a New Stereotype Configuration File

The stereotypes in Rational Rose must be defined in a stereotype configuration file. Rational Rose is delivered with a default stereotype configuration file, called `DefaultStereotypes.ini`. If possible, add your stereotypes to that file. If you do not want to use that file, follow these steps to create a new stereotype configuration file:

1. Close Rational Rose.
2. Create a text file (called, for example, `MyStereotypes.ini`) using Notepad or another text editor, and save it in the Rose installation folder.
3. Edit the new stereotype configuration file. For information on how to create a new stereotype and add it to a stereotype configuration file, please refer to the *Creating a New Stereotype* section.
4. Run the Windows Registry Editor (`regedit.exe`) by selecting **Run** on the **Start** menu. Type "regedit" and click **OK**.
5. Locate and select the section entitled [HKEY_LOCAL_MACHINE\SOFTWARE\Rational Software\RoseStereotypeCfgFiles] in the registry list.
6. Click **Edit > New** and click **String Value**. Give the new registry key the name "file#", where # is the next consecutive number (1, 2, or 3, etc.).
7. Double-click the new key, and enter the name of your configuration file (for example, `MyStereotypes.ini`).

8. Close the registry. Next time you open a model in Rational Rose, the stereotypes defined in your new stereotype configuration file will be available in the model.

Creating a New Stereotype for All Rose Models

To create a new stereotype and make it available in all models in Rose:

1. Close Rational Rose.

Note: *Optionally, create icons for the stereotype to be used in diagrams, lists, and diagram toolboxes. Please refer to the Creating Stereotype Icons section later in this chapter.*

2. Open the default stereotype configuration file, DefaultStereotypes.ini.
3. In the stereotype configuration file, add a line for the new stereotype in the section called [Stereotyped Items]. For example, to add the class stereotype Controller to an existing configuration file, add a corresponding line as follows:

```
[Stereotyped Items]
Class:Model
Class:View
Class:Controller
```

4. Create a section for the new stereotype, named exactly as the line you added in the [Stereotyped Items] section, for example:

```
[Class:Controller]
Item=Class
Stereotype=Controller
```

5. If you have created a diagram icon for the stereotype, specify the name of that file (Metafile). Note that you can use "&" instead of the folder of the stereotype configuration file. For example:

```
Metafile=&\MyStereotypeIcons\controller.emf
```

6. If you want to create a diagram toolbox button for this stereotype, specify the name of the file where you created the corresponding small toolbox icon (SmallPaletteImages) and the location of the icon in that file (SmallPaletteIndex). You can also specify the name of the file where the corresponding large toolbox icon is defined (MediumPaletteImages) and the location of the icon in that file (MediumPaletteIndex). For example:

```
SmallPaletteImages=&\MyStereotypeIcons\
small_palette_icons.bmp
```

```
SmallPaletteIndex=3
MediumPaletteImages=&\MyStereotypeIcons\
    medium_palette_icons.bmp
MediumPaletteIndex=3
```

7. If you want to graphically display this stereotype in specification lists or in the browser, specify the name of the file where you created its list icon (ListImages) and the location of the icon in that file (ListIndex). For example:

```
ListImages=&\MyStereotypeIcons\list_icons.bmp
ListIndex=2
```

8. Add any other setting needed to define the new stereotype. For a list of all available settings, information on the meaning of each setting, the possible values, and the default values, please refer to the “Stereotype Configuration File” topic in the online help. Note, however, that you only have to include settings for which you want to give other values than their default values.
9. Save your changes to the stereotype configuration file.
10. Run Rational Rose. View the log window to make sure there were no problems loading your icons.
11. If you created a diagram toolbox icon for the new stereotype, and want to add it as a button on a diagram toolbox, please refer to the *Adding Stereotypes to the Diagram Toolbox* section later in this chapter.

The new stereotype is now available in Rational Rose. For information on how to control the display of the new stereotype in diagrams and in the browser, please refer to the *Viewing Stereotypes* section earlier in this chapter.

Creating Stereotype Icons

For each stereotype, four different icons may be supplied:

- A diagram icon (to customize the appearance of model elements with this stereotype in diagrams).
- A small and a large diagram toolbox icon (to be able to add a button for this stereotype to the diagram toolbox). Two different sizes correspond to the **Use Large Buttons** option on the **Toolbars** tab of the **Options** dialog box.
- A list view icon (to graphically display the stereotype for model elements in specification lists and in the browser).

Creating a Diagram Icon

Diagram icons are “symbols” or “elements” that can be placed on a diagram from the browser, toolbar, or menu. Diagram icons have to be created in Windows Metafile (.wmf) or Enhanced Metafile (.emf) format. You can download drawing packages that support these formats at various shareware sites on the Internet. Enhanced Metafiles are recommended if possible. Diagram toolbox and list view icons must be created in bitmap (.bmp) format.

Note: *Note: If you create a diagram icon, for example, you will most likely want to create the other three corresponding icons: a list view icon, a small toolbar icon, and a large toolbar icon.*

To create a diagram icon:

1. Using a vector based (as opposed to bitmap) drawing application, draw your icon in the size you want it to appear in Rational Rose. It is not recommended that you use a drawing application that forces the icon to fit a certain area, such as a page as with PowerPoint.
2. Consider the following: Make sure that the scaling factor is set to 100% when deciding on the size of the icon. Use colors if you like. If you want the name of the model element to appear within the stereotype icon, leave some blank space for it.
3. Select the icon and export it in either the Windows Metafile (.wmf) format or the Enhanced Metafile (.emf) format. If you use CorelDraw, make sure the Include header option is checked if you save your selection as a Windows Metafile.

Creating Diagram Toolbox and List View Icons

Diagram toolbox icons and list view icons (icons that appear in the browser) are created in bitmap (.bmp) format and Rational Rose only supports bitmap files saved in the 256 color bitmap scheme. You can create one bitmap file containing several icons, arranged horizontally side by side. Note that the SmallPaletteIndex setting in the configuration file of a stereotype specifies the diagram toolbox icon that belongs to a specific stereotype. The ListIndex setting specifies the list icon that belongs to a specific stereotype. Diagram icons can only be created in Windows Metafile (.wmf) or Enhanced Metafile (.emf) format.

Note: If you create a list view icon, for example, you will most likely want to create the other three corresponding icons: a diagram icon, a small toolbar icon, and a large toolbar icon.

To create a new icon:

1. Create or open a bitmap file using a program such as Microsoft Paint or the bitmap editor in Microsoft Visual Studio.

Note: If you are adding several icons to the same bitmap file in Microsoft Visual Studio, use the grid setting in the Image menu to help you see the borders of each icon.

2. Create an icon of the following size and background color:
 - **Small diagram toolbox icon** - 15 pixels high and 16 pixels wide, using a gray background (which is RGB = 192, 192, 192 in Rational Rose).
 - **Large diagram toolbox icon** - 24 pixels high and 24 pixels wide, using a gray background (which is RGB = 192, 192, 192 in Rational Rose).
 - **List view icon** - 16 pixels high and 16 pixels wide, using a white background.
3. Save the icon as a 256 color bitmap file (.bmp). To save the color setting in Microsoft Paint, select 256 color bitmap from the Save as type menu in the Save as dialog box.

Note: Some of your icon colors may not show up precisely because the color palette used by the toolbars is limited.

Adding Stereotypes to the Diagram Toolbox

To make a stereotype available as a button on a diagram toolbox:

1. Create a stereotype and a corresponding diagram toolbox icon. For information on how to do that, please refer to the *Creating a New Stereotype* section earlier in this chapter.
2. Click **Tools > Options**, and click the **Toolbars** tab.
3. Under **Customize Toolbars**, click on the diagram type for which you want to change the toolbox.

or,

In an open diagram, right-click in the diagram toolbox and select **Customize**.

4. The **Customize Toolbars** dialog box is displayed. The leftmost column provides the list of available icons. Select the icon you want to appear on the diagram toolbox and click **Add**.

Subsystem Stereotype Package

Although closely related to a system, a subsystem is a group of model elements that has specific behavior and objectives. A subsystem is a stereotyped package and is represented by the package icon with the subsystem stereotype.

Note: *The term subsystem is also used in the Rose Extensibility Interface (REI). However, there is a specific distinction between each term. In the REI, any package that resides in the component view is considered a subsystem. A subsystem on a Rose diagram is a stereotyped package.*

Subsystem Stereotype Sample

The subsystem stereotype sample demonstrates how subsystems collaborate to make up a larger system. The sample below illustrates a bookstore system and the smaller subsystems that make up the total system.

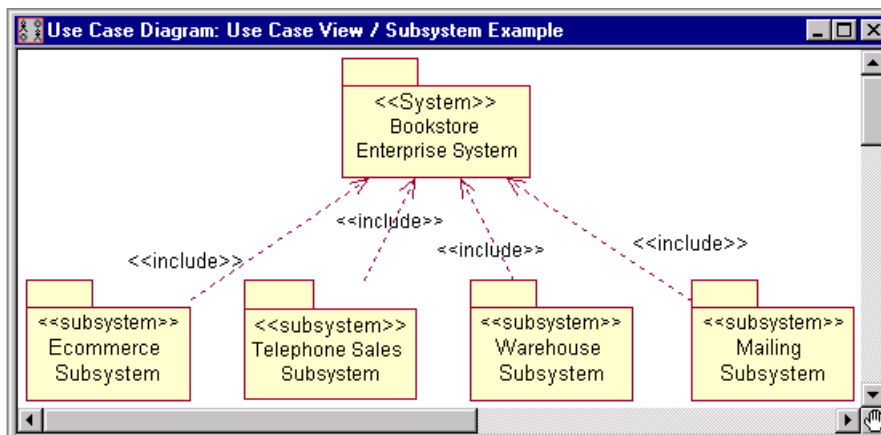
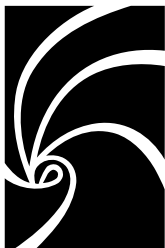


Figure 96 Subsystem Stereotype Sample

The four subsystems in the subsystem sample together make up all the functionality of the Bookstore Enterprise System. Noted throughout the <<include>> relationships, each subsystem provides a certain piece of the Bookstore system functionality.

Instead of going into separate subsystems, a user of the Bookstore Enterprise System can verify stock in the Warehouse subsystem or check the status of a shipped book in the mailing subsystem, for example. All subsystems make up a much larger system. Each stereotyped package subsystem is just a means of organizing model elements and diagrams together.



Chapter 13

Framework Wizard Add-In

The Framework Wizard Add-In provides a library of frameworks that can be used as templates when creating new models. If the Framework Wizard Add-In is active, the **File > New** command in Rational Rose displays a dialog box from which you can choose one of the available frameworks. By choosing an appropriate framework when you create a new model, the model is automatically initialized with a predefined architecture and a set of reusable model elements. This way, you can focus your modeling efforts on the parts that are unique to your system, instead of “reinventing the wheel.”

The Framework Wizard Add-In also provides a wizard to help you add additional frameworks. The Wizard is started by opening the “Make New Framework” framework.

Note: *The Framework Wizard Add-In is only available on Windows, and only in some Rational Rose editions. Also, in order to create models from frameworks and add new frameworks, the Framework Wizard Add-In must be active (see the Activating the Framework Wizard Add-In section.)*

Activating the Framework Wizard Add-In

In order to create models from frameworks and add new frameworks, the Framework Wizard Add-In it must be active. It is active if the **File > New** command in Rational Rose displays a **Create New Model** dialog box. If the **File > New** command just opens a new empty model, the Framework Wizard Add-In is not active.

To Install the Framework Wizard Add-In

1. Run the Rational Rose setup program.
2. Select a custom install, and select the Rose Framework Add-In feature. If the Framework Wizard Add-In feature is not present, the add-in is not available in your Rational Rose edition.

To activate the Framework Wizard Add-In:

1. Click **Add-Ins > Add-In Manager** in Rational Rose.
2. Select the **Framework Wizard** option and click **OK**. If the option is not present, the Framework Wizard Add-In is not installed.

Creating a New Model from a Framework

To create a new model from a framework, use the following steps:

1. Click **File > New**. The **Create New Model** dialog opens.

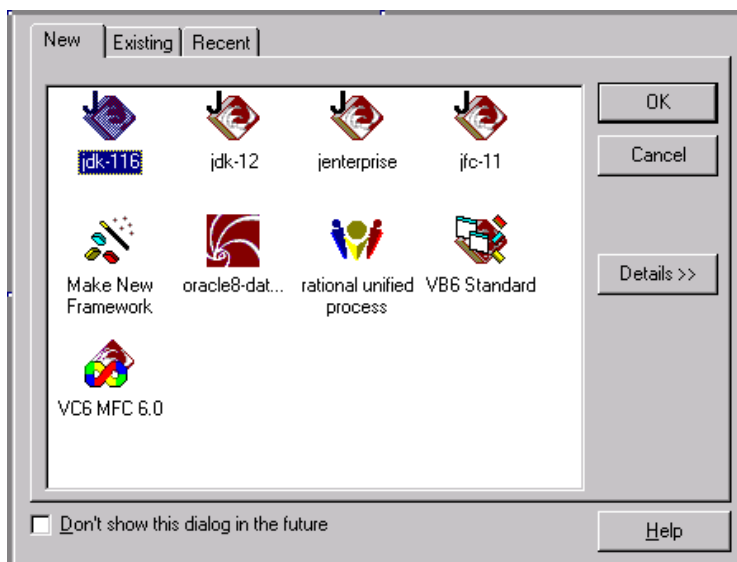


Figure 97 Create New Model Dialog Box

2. Open the framework that corresponds to the system you are going to develop. A new model is created and initialized with the contents of the chosen framework. (If you don't want to use any of the frameworks, click **Cancel**. A new model with only the default contents is created.)
3. Save the new model and give it a name by clicking **File > Save As**.

Note: *Each package in a framework is stored as a controlled unit in a separate file. To access the contents of a package in a framework, you have to load the corresponding controlled unit. To load a unit, double-click on the package in a diagram, or click **File > Units** and click **Load**.*

Creating and Deleting Frameworks

Rational Rose provides you with a Framework Wizard that helps you create a new framework and add it to the framework library. To use the Framework Wizard, the Framework Wizard Add-In must be installed and activated (see the *Activating the Framework Wizard Add-In* section).

The Framework Library

The Framework Wizard Add-In provides a library of predefined frameworks. The frameworks are located in the `\Framework\Frameworks` folder in your Rational Rose installation folder. When creating a new model, you can choose to create the model from one of the listed frameworks. The set of available frameworks are displayed with the **File > New** command.

In the framework library, all files that work together to define a specific framework are located in a folder with the same name as the framework. Each framework is defined by the following files:

- **FrameworkName.mdl**, which contains the model framework itself. This model is an ordinary Rational Rose model.
- **FrameworkName.ico**, which includes the icon that symbolizes the framework in the **Create New Model** dialog box. If there is no **.ico** file, Rational Rose displays a default icon for the framework.
- **FrameworkName.rtf**, which includes a description of the framework, which is shown in the **Create New Model** dialog box when the user clicks **Details**. If there is no **.rtf** file, Rational Rose displays a default description text.

- **Parameters**, which holds the name of the diagram that is initially opened for models created from this framework. The Framework Wizard automatically creates this file and enters the name of the diagram as a line with the following syntax:
“StartDiagram=ParentPackage / DiagramName.” For example:
“StartDiagram=Logical View / Framework Overview.”

Creating a New Framework

To create a new framework, perform the following steps:

1. Create and save a model with the contents of the framework in any folder. That model will be used as the template when creating new models from this framework.

Note: *Optionally, write a description of the framework in any word processor and save the document in RTF (Rich Text Format) format in any folder.*

Note: *Optionally, use a drawing tool to create an icon that symbolizes the new framework. Save the icon as an .ico file in any folder. (Or look for a suitable existing .ico file.)*

2. Click **File > New**. The **Create New Model** dialog box opens.
3. Open the “Make New Framework” framework, which starts the Framework Wizard. (If the welcome page is shown, click **Next**.)

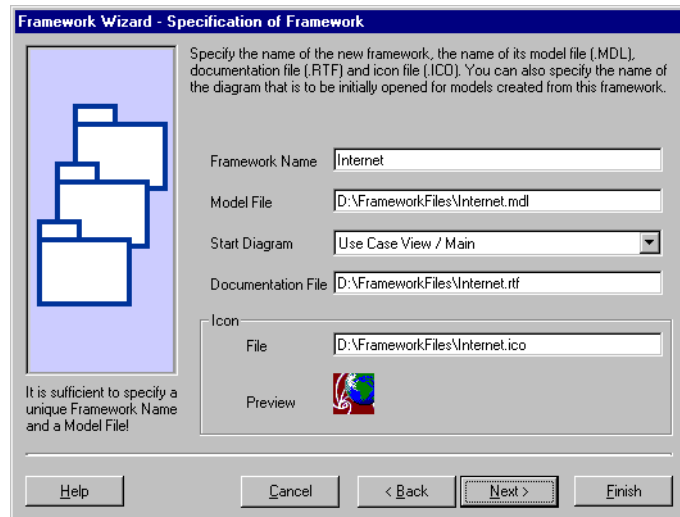


Figure 98 Framework Wizard Specification Page

4. In the **Framework Name** field, specify the name of the new framework. The name must be unique among the existing frameworks, and it can only contain characters that are allowed in folder names.
5. In the **Model File** field, specify the name and location of the file that constitute the framework model. To browse to the file, click in the field. Then click the displayed button.

Note: *Optionally, click in the **Start Diagram** field to specify a diagram that is to be initially opened for models created from this framework. The specified model opens. Click the arrow to the right of the **Start Diagram** field and select one of the diagrams.*

Note: *Optionally, specify the name and location of the documentation and icon files in the **Documentation File** and **Icon File** fields. (To browse to a file, click in the field. Then click the displayed button.)*

6. Click **Next**. A summary of the new framework is shown.

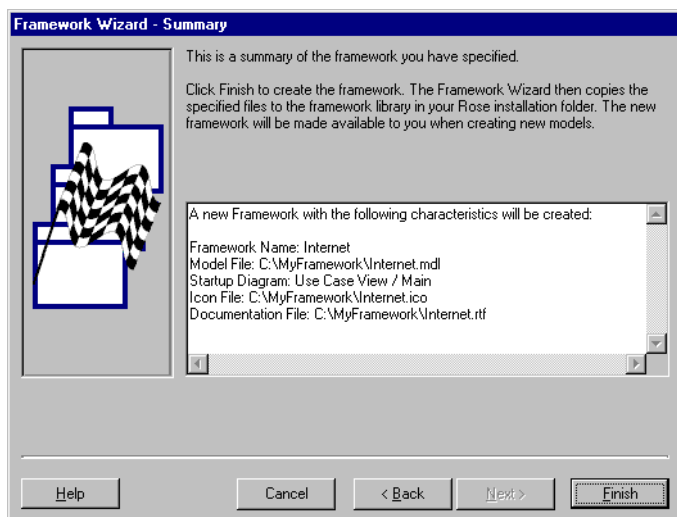


Figure 99 Framework Wizard Summary Page

7. If you are satisfied with the framework specification, click **Finish**. Otherwise, go back and change your settings.

When the Framework Wizard is finished, a folder with the same name as the new framework, containing the specified files, will be created in the `\Framework\Frameworks` folder. The new framework is now available for creating new models.

Changing or Deleting a Framework

To change the contents of a framework model, its icon, its description, or the initial diagram to be opened, update the appropriate file in the framework's folder.

To delete a framework, delete its folder in the `\Framework\Frameworks` folder.



Chapter 14

Type Library Importer

The Type Library Importer allows you to import a type library of a COM component into the model, by dragging the COM component from the Windows Explorer and drop it in Rational Rose. Or, you can use the **Tools > COM > Import Type Library** command.

You can control several aspects of how type libraries are imported into the model. For example, you can control:

- What should happen with existing type libraries when importing new versions
- The name and location of new type libraries in the model
- The name and contents of the overview diagrams that are created when importing type libraries

For further information, see the *Customizing the Type Library Importer* section later in this chapter.

Note: *Importing a component is not the same as reverse engineering a component into the model. Imported components are still external to the system, because you import only the type library of the components, whereas reverse engineering a component means that a model of the component's source code is added to the model.*

What Is a Type Library?

A type library contains a description of a COM (component object model) component as viewed from the outside. The description includes the coclasses, interface items, dispinterfaces, properties (called attributes in UML), methods (called operations in UML), data

types, etc. of the component. Type library information is needed to provide a language-neutral description of the interfaces and data types that a COM component exposes.

This chapter does not explain the different kinds of type library items—coclasses, interfaces, dispinterfaces, etc. For detailed information about COM components and type libraries, refer to:

- Don Box, *Essential COM*, Addison-Wesley Pub Co, ISBN 0201634465
- <http://msdn.microsoft.com/library>—for example the *Inside OLE* section in the *Books* section.

Why Would I Want to Import Type Libraries into the Model?

By importing type libraries into the model, you can show how classes in the model use and depend upon classes in other components, regardless of their implementation language. For example, you can:

- Reuse COM components—that is, to show how the classes in the model instantiate, use, and communicate with the items in a COM component.
- Show how classes in the model implement (or realize) the interface items of a COM component.
- Show dependencies between components.
- Use the data types defined by a COM component when specifying attributes and operations on the classes in the model.



What COM Components Can Be Imported into the Model?

The following file types can be imported into a Rational Rose model:

- Dynamic Link Libraries (.dll)
- Executables (.exe)
- ActiveX Components (.ocx)

- Object Libraries (.olb)
- Type Libraries (.tlb)

The file must contain valid type information. If you drop a file without type information on an element in the browser, Rational Rose treats it as any file and attaches the dropped file to the model element that you drop it on. When you drop the file in Rational Rose, the cursor indicates whether the file will be imported or attached to a model element. A cursor with:

- a  icon means that Rational Rose imports the file
- a  icon means that the file is attached to the model element that you drop it on

How Is a Type Library Presented?

In Rational Rose, an imported type library is represented as a component in the component view and a logical package containing the type library items. In your implementation environment, a type library is presented differently in different implementation language environments.

A Type Library in Rational Rose

The Type Library Importer creates a component, such as Scripting Ver 1.0 (Microsoft Scripting Runtime) in Figure 100, in the component view for an imported type library.

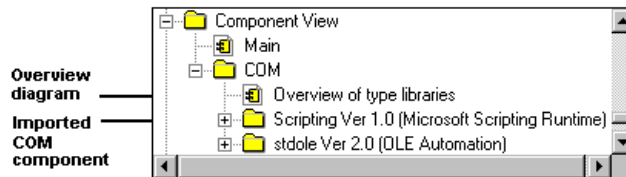


Figure 100 *The Component View of the Microsoft Scripting Runtime Type Library*

The Component Overview Diagram

The component is automatically inserted into a type library overview diagram in the component view. For example, the overview diagram in Figure 101 shows that two type libraries, Scripting and stdole, have been imported into this model and that the Scripting type library depends upon the stdole type library.

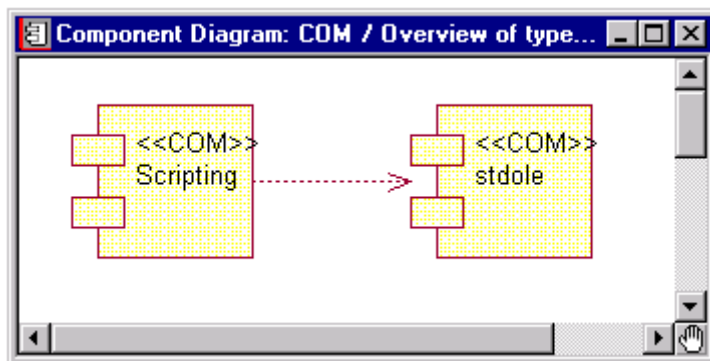


Figure 101 The Component Overview Diagram for a Model

The Logical View of a Type Library

The logical view contains a package for the imported COM component, such as Scripting Ver 1.0 (Microsoft Scripting Runtime) in Figure 102.

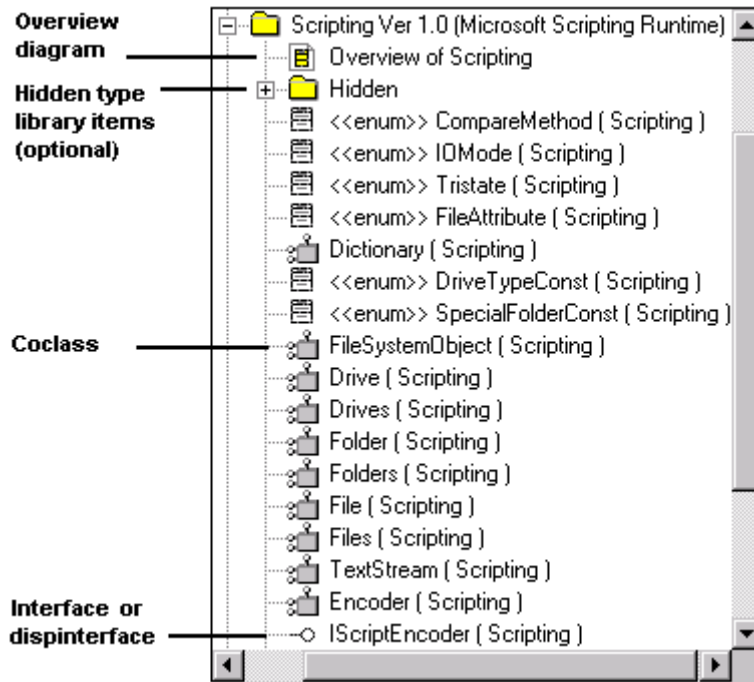



Figure 102 The Logical View of the Microsoft Scripting Runtime Type Library

Type Library Items

The logical package contains the type library items that are defined by the type information of the imported COM component—coclasses, interfaces, dispinterfaces, etc.

Each item in the type library is represented by a class, such as the coclass “FileSystemObject” in Figure 102. The stereotype of the type library’s classes in the model indicates the kind of project item—coclass, interface, enum, type, module, struct, and union (see the *COM Stereotypes* section below). As you can see in Figure 102, coclasses have their own icon in the browser: .

The “kind” model property on an interface class indicates whether the class corresponds to an interface or a dispinterface in COM.

Note: *If the type library was imported using a quick import, the Type Library Importer does not create any class members (attributes and operations) on the imported items. If you chose a full import, the class members are created. You can later import the class members for a type library item, see the Adding Class Members to a Quick Import Type Library section in this chapter.*

The Type Library Overview Diagram

An overview diagram is created in the logical view, which shows the contents of the imported type library. Figure 103 shows the overview diagram for the Microsoft Scripting Runtime type library.

As you can see, coclasses, such as Encoder in Figure 103, are green in type library overview diagrams.

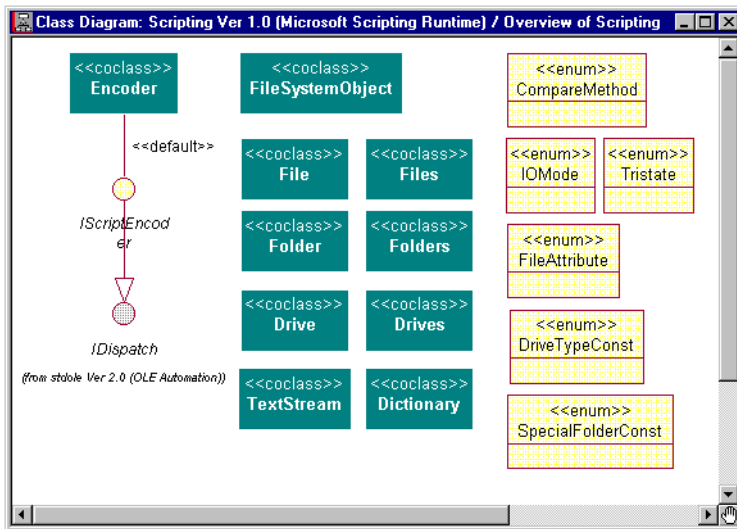


Figure 103 The Logical Overview Diagram of the Microsoft Scripting Runtime Type Library

Hidden Items

If the **Show hidden items** option in the **COM Properties** dialog box is cleared when a type library is imported, all hidden items and items with names beginning with “_” are placed in a separate logical package called “Hidden”. Those items are not displayed on the overview diagram. The Microsoft Scripting Runtime type library in Figure 102 and Figure 103 was imported with the **Show hidden items** option cleared.

For more information, see the *Hiding Type Library Items* section in this chapter.

Referenced Type Libraries

Any referenced type libraries are automatically imported. When importing a type library item, for example A, all items that A refers to must exist in the model. If the referenced items did not exist in the model before, the Type Library Importer automatically imports the type libraries of the corresponding COM components and adds dependency relationships between them. For example, the `stdole` type library in Figure 101 was automatically imported when the Scripting type library was imported, because Scripting refers to the `stdole` type library.

Referenced type libraries are imported using a quick import. Also, type library items that are referenced by the current type library, such as `IDispatch` in Figure 103, are gray in the type library’s overview diagram.

COM Stereotypes

The Type Library Importer uses the stereotypes in Table 17 for the model elements that represent a type library in the model.

Table 17 COM Stereotypes

Stereotype:	Meaning:
Components	
COM	The Type Library Importer assigns the stereotype and language “COM” to the component representing an imported type library.
Classes	

Stereotype:	Meaning:
coclass	Represents an enum type definition in a type library. The class members of the enum in the type library become attributes with initial values on the class in the model.
enum	Represents an enum type definition in a type library. The class members of the enum in the type library become attributes with initial values on the class in the model.
interface	Represents an interface or dispinterface in a type library. The “kind” model property on the class in Rational Rose indicates whether the class is an interface or a dispinterface. Type library interfaces are always abstract—that is, the Abstract box in the interface’s Class Specification is checked.
module	Represents a module in a type library.
struct	Represents a struct type definition in a type library. The class members of the struct in the type library become typed attributes on the class in the model.
union	Represents a union type definition in a type library. The class members of the union in the type library become typed attributes on the class in the model.
Operations	
propget	Corresponds to a property-accessor function on an interface or dispinterface.
propput	Corresponds to a property-setting function on an interface or dispinterface.
propputref	Corresponds to a property-setting function that uses a reference instead of a value.
Realize relationship	
<i>none</i>	Represents any realize relationship between type library items.
source	Indicates that the realized interface contains the coclass’ event procedures.

Stereotype:	Meaning:
default	Indicates that the realized interface is the default interface of the coclass.
Dependency relationship	
imports	Indicates that the supplier COM component was automatically imported when the client component was imported, because the supplier component is referenced by the client component.

A Type Library in the OLE Viewer in Visual Studio

The contents of a type library as shown in the OLE Viewer correspond to how Rational Rose presents an imported type library. Figure 105 shows how the OLE Viewer in Visual Studio presents the Microsoft ActiveX Data Objects type library, MSADO15.dll. All the type library items displayed by the OLE Viewer can be found in the model after importing the type library.

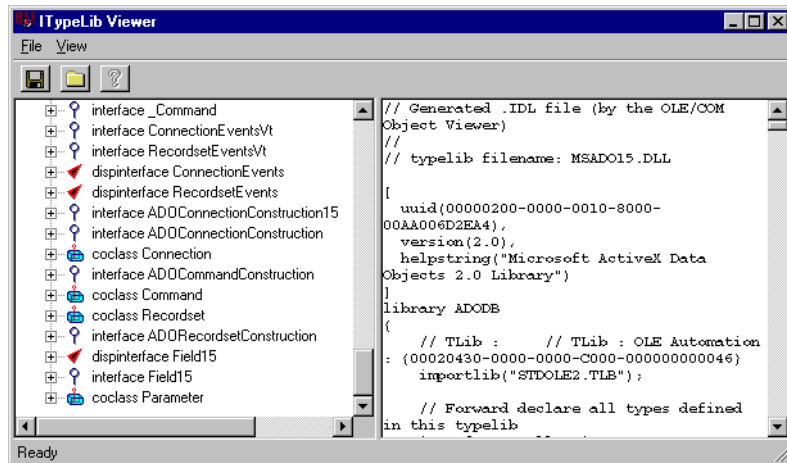


Figure 104 The OLE Viewer in Visual Studio

A Type Library in the Object Browser in Visual Basic

Figure 105 shows how the Object Browser in Visual Basic presents the Microsoft ActiveX Data Objects type library, MSADO15.dll. The Object Browser in Visual Basic shows only those type library items that are relevant in Visual Basic. For example, it does not list the default interfaces of the coclasses, because Visual Basic assumes the default interface when referring to a coclass.

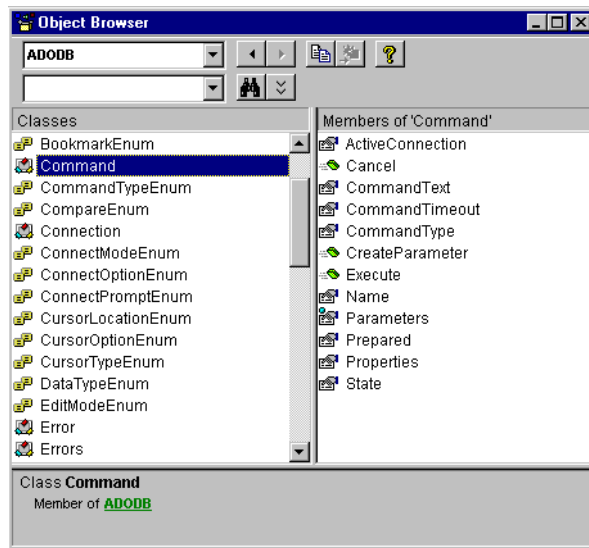


Figure 105 The Object Browser in Visual Basic

As Rational Rose supports many different programming languages, all items in an imported type library are shown in the model. However, by clearing the **Show hidden items** option, the top level of the packages of the type libraries that you import, as well as their overview diagrams, give the same view of the libraries as the Object Browser in Visual Basic. For more information, see the *Hiding Type Library Items* section in this chapter.

Importing a Type Library Into the Model

The Type Library Importer in Rational Rose allows you to import a type library of a COM component into the model. By doing that, you can show how classes in the model use and depend upon classes in other components.

If you want to change the default behavior of the Type Library Importer, the **Tools > COM > Properties** dialog box allows you to control how to import type libraries. For more information, see the *Customizing the Type Library Importer* section in this chapter.

To import a type library into the model:


1. If a type library is to be used by Visual Basic classes only, you may want to show only the type information that is relevant for Visual Basic classes. See the *Hiding Type Library Items* section in this chapter.
2. Drag the file—DLL, EXE, OCX, OLB, or TLB—from the Windows Explorer and drop it in the browser or in a diagram. (The drop target is not important, because the type library is created in the packages defined by the **Default package** options in the **COM Properties** dialog box.)

Note: *If the Rational Rose application window is hidden or minimized, point to the Rational Rose icon in the Windows task bar before dropping the file, which brings the application to the front. Instead of dragging and dropping the file, you can click **Tools > COM > Import Type Library** and select the appropriate file.*

3. On the displayed menu, select whether to import the full component (**Full Import**), including all operations and attributes of the type library items, or to perform a **Quick Import** which excludes the members.

Note: *You can later import the members of a quickly imported type library item, see the *Adding Class Members to a Quick Import Type Library* section in this chapter.*

4. The following now happens:
 - If the selected COM component contains all the necessary type information, the Type Library Importer creates a representation of the type library in the model.

- If a dragged and dropped COM component does not contain valid type information, and if you have dropped the component on an element in the browser, Rational Rose attaches the dropped file to that element if possible. That is, if the cursor turns into an arrow with a  icon, Rational Rose will attach and not import the file.

Importing a New Version of an Existing Type Library

To import a new version of a type library that already exists in the model, right-click on the component that represents the imported type library and click **Upgrade to Latest Version**.

Hiding Type Library Items

When importing a type library, the created type library in the model is represented differently depending on the **Show hidden items** option in the **Tools > COM > Properties** dialog box.

Show Hidden Items Selected

If the **Show hidden items** option is selected when importing a type library, all type library items, coclasses, dispinterfaces, interfaces, etc. are shown on the type library's overview diagram. Also, the type library items are inserted directly under the type libraries package in the browser.

Figure 106 shows how the Microsoft Scripting Runtime component, `scrrun.dll`, is presented when imported with the **Show hidden items** option is selected. This view is recommended when developing clients in other languages than Visual Basic.

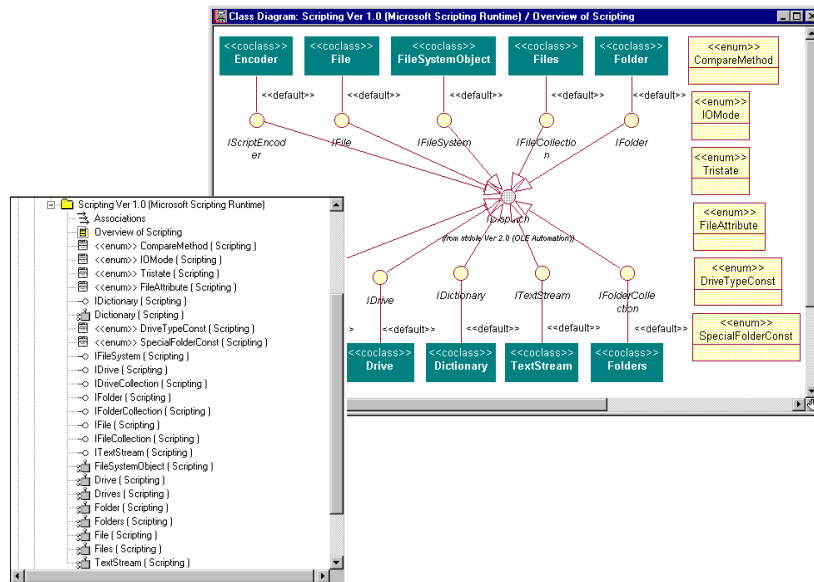


Figure 106 Type Library with Show Hidden Items Option Selected

Show Hidden Items Cleared

If the **Show hidden items** option is cleared when importing scrrun.dll, no hidden type library items are shown on the type library’s overview diagram. Also, the hidden type library items are inserted into a separate package called “Hidden” in the type library’s package in the browser.

Figure 107 shows how the Microsoft Scripting Runtime component, scrrun.dll, is presented when imported with the **Show hidden items** option is cleared.

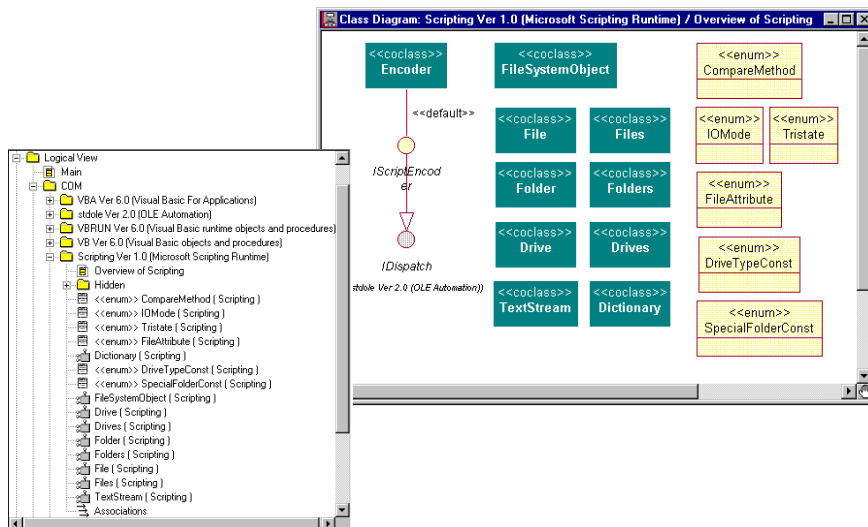


Figure 107 Type Library with Show Hidden Items Option Cleared

This means that with this view, only the type information that is relevant for Visual Basic clients are shown on the type libraries overview diagram, and all hidden type library items are inserted into a separate package called “Hidden”. Thus, this view is recommended when developing Visual Basic clients, because it corresponds to the view that is shown by the Object Browser in Visual Basic (see Figure 105).

Using an Imported Type Library

By importing type libraries into the model, you are able to show how classes in the model use and depend upon classes in other components, regardless of their implementation language. The application you are modeling can use a type library in several ways, for example:

- Classes can use the data types defined by a type library
- Classes in the model can implement the interface of a COM component
- A COM component can be reused by the application

How to use a type library differs between different programming languages. For further information, refer to the Rational Rose documentation of each language integration.

Adding Class Members to a Quick Import Type Library

If a type library was imported using a quick import, the Type Library Importer did not create any class members (attributes and operations) on the imported items. You can later import the class members of a type library item by doing a full import of that item.

To add class members to a type library:

1. In the browser or in a diagram, right-click on a(n):
 - interface to import its class members into the model
 - coclass to import the class members of its interfaces into the model
 - COM component to import the class members of all the items in that type library
2. Click **Full Import** on the displayed menu.

Note: *It may take several minutes for Rational Rose to perform a full import of an entire COM component. If you do not want to import the entire type library, perform a full import of only those type library items that you are using.*

Customizing the Type Library Importer

In the **Tools > COM > Properties** dialog box, you can control several aspects of how type libraries are imported into the model. For example, you can control the following:

- What should happen with existing type libraries when importing new versions
- The name and location of new type libraries in the model
- The name, location, and contents of the overview diagrams that are created when importing type libraries

To open the **COM Properties** dialog box, click **Tools > COM > Properties**:

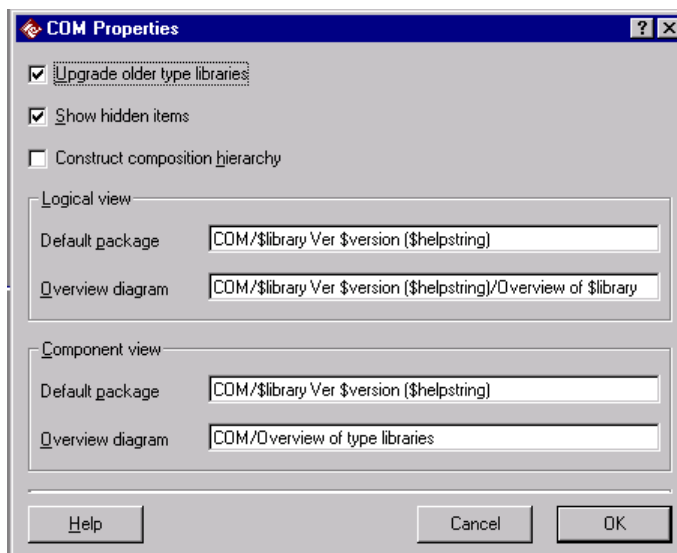


Figure 108 The COM Properties Dialog Box

Note: Changing the settings in the **COM Properties** dialog box affects only type libraries that are imported hereafter.

To replace existing type libraries when importing new versions:

Select the **Upgrade older type libraries** option in the **COM Properties** dialog box. The next time you import a new version of a type library, the current version is replaced by the new version. If this option is cleared when you import a new version of a type library, the model will contain both versions.

To hide items that are defined as hidden or called “_item”:

See the *Hiding Type Library Items* section in this chapter.

To show the composition hierarchy for imported type libraries:

Select the **Construct composition hierarchy** option. The next time you import a type library, the Type Library Importer adds association relationships between its related interfaces, which indicate the type library's composition hierarchy.

To change the name of the logical packages where type libraries are created:

In the **Default package** box under **Logical view** in the **COM Properties** dialog box, type the name of the package including the path of any enclosing packages. You can use the following variables in the package name:

- \$library — the name of the imported type library, which corresponds to the library model property
- \$version — the version of the imported type library, which corresponds to the version model property
- \$helpstring — a description of the type library, which corresponds to the helpstring model property

For example, “COM/\$library Ver \$version (\$helpstring)” means that the following logical package is created for a new type library called stdole:

```
Logical View
COM
    stdole Ver 2.0 (OLE Automation)
```

To change the name of the component packages where type library components are created:

In the **Default package** box under **Component view** in the **COM Properties** dialog box, type the name of the package including the path of any enclosing packages. You can use the same variables as above.

For example, “COM/\$library Ver \$version (\$helpstring)” means that the following component package is created for a new type library called stdole:

```
Component View
COM
    stdole Ver 2.0 (OLE Automation)
```

You can change the name and location of the diagrams on which type libraries are displayed.

In the **Overview diagram** box under **Logical view** or **Component view** in the **COM Properties** dialog box, type the name of the diagram including the path of any enclosing packages. You can use the same variables as above in the diagram name.

For example, the default value for the logical view is “COM/\$library Ver \$version (\$helpstring)/Overview of \$library”, which means that the Type Library Importer creates a diagram called “Overview of stdole Ver 2.0 (OLE Automation)” when you import a COM component called stdole.

The default value for the component view is “COM/Overview of type libraries”. This means that the Type Library Importer inserts all imported type library components into the same diagram, which is called “Overview of type libraries”.



Appendix A

Upgrading From a Previous Release

This appendix describes procedures for converting models developed under previous versions of Rational Rose.

Upgrading from Rational Rose 3.0 or Later

If you are upgrading from release 3.0 or later of Rational Rose for Windows, your models are converted automatically when you open them. When you save your model, Rational Rose asks you if you want to save your model in the new format.

Upgrading from Releases Prior to Rational Rose 3.0

If you are upgrading from a Rational Rose release prior to 3.0, please contact technical support for assistance.

Understanding Petal File Versions

Petal files and Model files are very similar. However, a petal file is a portion of a model file whereas a model file is really the complete or entire model. You can create a petal file by saving your model in petal through the **File > Save As** command. Petal files are also created when you export part of a model through the **File > Export** command.

The following table contains the petal file version numbers for each Rational Rose release. If you save a model as an older version of Rose, some model elements and features will be lost. For example, if you save a Rose 2000 model in a Rose 98i format, your model will not include activity diagrams.

Table 18 Rational Rose Petal File Versions

Rose Version	Petal File Version	Rose Format
Rose 3.0	Petal 37	3.0 Model
Rose 4.0	Petal 40	4.0 Model
Rose 98 and Rose 98i	Petal 42	4.5/6.0 Models
Rose 98i Service Pack 1 and Rose 2000	Petal 43	6.1/6.5 Models
Rose 2000e	Petal 44	7.0 Model



Appendix B

Contacting Technical Support

This appendix describes procedures for interacting with Rational Software Corporation's technical support services.

When Contacting Rational Technical Support

When contacting Rational technical support, please be prepared to supply the following information:

- Name, telephone number, and company name
- Computer make and model
- Make and version number of operating system
- Product release number
- Your log number (if you're calling about a previously reported problem)

If your site has a designated, on-site support person, please try to contact that person before contacting Rational technical support.

How to Contact Rational Customer Support

Rational customer support can provide information and assistance by:

- Telephone
- Electronic Mail
- Fax
- Rational Web Site

Please note that all information was current at the time of printing. To be assured of the latest information, from the **Help** menu, point to **Rational on the Web** and select **Technical Support**.

Telephone and E-mail

Telephone and E-mail support is available Monday through Friday (except holidays) in four major Rational Technical Support Call Centers around the world. Specific call center contact information is located at the end of this Appendix, as well as on our web site at www.rational.com.

When contacting Technical Support through e-mail, please include the requested information outlined in the previous section “When Contacting Rational Technical Support,” along with a detailed description of your problem. Upon receipt of your request, Rational Technical Support will send you an electronic response with your Log id # and point of contact for your issue. When sending e-mail concerning a previously reported problem, please include in the subject field: “re: Log XXXX,” substituting your assigned support log id for the XXXX.

Fax

Rational Technical Support Engineers will sometimes ask you to fax information to help them diagnose problems. Please mark faxes "Attention: Technical Support" and add your fax number to the information requested earlier.

Rational Web Site

You can also contact Rational technical support through our web site at www.rational.com.

Rational Technical Support Call Center Contact Information

This information was accurate at the time of printing. If you experience any difficulty contacting us using this information, please check our web site at www.rational.com/support for the most up-to-date information.

North America

18880 Homestead Road
Cupertino, CA 95014
Telephone: 800-433-5444 or 408-863-4000
E-mail: support@rational.com

Europe

Beechavenue 30
1119 PV Schiphol-Rijk
The Netherlands
Telephone: +31 (0)20 4546 200
E-mail: support@europe.rational.com

Asian Pacific

Level 13, 821 Pacific Hwy

Chatswood NSW 2067

Australia

Telephone: +61-2-9419-0111

E-mail: support@apac.rational.com



Index

Symbols

+ (plus) sign 23

A

Abstract 71, 119

Action 143

Activities 133

Activity diagrams 129

activity diagrams 125

Actor Specification 123

Add icons to a diagram 32

Add-In

 Installing 5

 Manager 5

Adding

 Classes 64

 Stereotypes to the Diagram Toolbox
 213

Add-Ins 5

Adorning the Diagrams 45

Aggregate 102

Application Window 8

 documentation window 8

 maximize button 8

 menu bar 8

 title bar 8

 toolbar 8

Arguments 88

Assigning a Component to Another Package 186

Assigning Classes 64

Association 176

Association Specification 96

 Detail Tab 98

 Constraints 99

 Derived 99

 Link Element 99

 Name Direction 99

 General Tab 97

 Element 98

 Parent 97

 Role 98

 Stereotype 97

 Role A and B Detail Tab 101

 Aggregate 102

 Containment 102

 Friend 102

 Keys/Qualifiers 103

 Navigable 101

 Static 102

 Role A and B General Tab 100

B

Browse

 Class Diagram 11

 Component Diagram 11

 Deployment Diagram 11

- Interaction Diagram 11
- Parent 12
- Previous Diagram 12
- Browser 21, 22, 23
 - Collapsing 23
 - Creating Icons 212
 - Docked 22
 - Docking 23
 - Drag-and-Drop 26
 - Hiding 22
 - Hiding and Displaying 22
 - Naming an Element 26
 - Navigating 24
 - Redock 23
 - Undocking 23
 - Viewing 22
- Browser to Browser Capabilities 27
- Browser to Diagram Capabilities 28
- Browser to Specification Capabilities 29
- Browser Tree
 - Collapsing 23
 - Expanding 23
- C**
- Cardinality 69
- Change the default font parameters 45
- Changing the State of an Object 136
- Characteristics 198
- Class 83, 173, 175, 181
- Class Attribute Specification 82
 - Detail Tab 84
 - Containment 84
 - Derived 85
 - Static 85
 - General Tab 82
 - Class 83
 - Initial Value 83
 - Show Classes 83
 - Type-Class Attribute 83
- Class Diagram 11, 61
 - Creating 62
 - Displaying 62
 - Re-assign a class 64
 - Toolbox 62
- Class Instance Specification 174
 - General Tab 174
 - Class 175
- Class Specification 65
 - Attributes Tab 75
 - Detail Tab 68
 - Abstract 71
 - Cardinality 69
 - Concurrency 71
 - Formal Arguments 72
 - Persistence 70
 - Space 69
 - Files Tab 81
 - General Tab 66
 - Export Control 67
 - Parent 66
 - Stereotype 67
 - Type 66
 - Operation Tab 72
 - Show Inherited 74
 - Relations Tab 77
- Classes 191
- Client Visibility 177
- coclass 227
- Collaboration 157
- Collaboration Diagram 158
 - Create 170
 - Toolbox 159
- COM components 223
- COM Properties dialog box 237
- Common Specification Elements 53
- Component Diagram 11, 193
 - Creating 186
 - Displaying 186
 - Toolbox 186
- Component Name 78
- Component Specification 187
 - Detail Tab 189
 - Declarations 190

- General Tab 188
 - Language 188
 - Stereotype 188
- Realizes Tab 190
 - Classes 191
 - Language 191
 - Show all Classes 191
- Concurrency 71, 90
- Connection Specification 201
- Connections 195
- Constraints 99
- Containment 84, 102
- Context Sensitive Help 11
- Control-Menu Box 8
- Copy 10
- Correlation 43, 44
 - Bending 44
 - Changing the Name 45
 - Reconnecting 44
- Creating
 - Alternative Diagrams 170
 - Collaboration Diagram from a Sequence Diagram 171
 - Component Diagram 186
 - Diagram Icon 212
 - Elements 37
 - Icons for the Diagram Toolbox or Browser 212
 - Icons on a diagram 32
 - Model Elements 25
 - New Model 9
 - New Stereotype Configuration File 209
 - Overloaded Element on the Diagram 39
 - Sequence Diagram from a Collaboration Diagram 171
- Creating an Activity Diagram 130
- Creating Diagram Toolbox and List View Icons 212
- Creating Model Elements 37
- Cut 10

D

- Decisions 138
- Declarations 190
- Deep Delete 20
- Default Font Parameters
 - Changing 45
- Delete 42
 - From Model 42
 - Icons 42
- Deleting 42
- Deleting Model Elements 20
- Dependency 114
- Dependency Specification 105
 - General Tab 106
- Deployment Diagram 11, 195
 - Creating 196
 - Displaying 196
 - Toolbox 196
- Derived 85, 99
- Deselecting Icons 41
- Detail Tab 55
- Device Specification 200
 - Detail Tab 201
 - General Tab 200
- Devices 195
- Diagram 120
 - Adorning 45
 - Creating 34, 170
 - Deleting 36
 - Displaying 35
 - Placing Text 45
 - Renaming 36
- Diagram Icon 212
- Diagram Toolbox
 - Adding Stereotypes 213
 - Creating Icons 212
- Diagram Window 31, 32
- Diagrams 120
- Dialog Boxes 53
- dispinterface 227
- Displaying 22

- Browser 22
- Multiple Diagrams 33
- Specifications 51
- Docked 22
- Documentation 54
- Documentation Window 8
- Drag-and-Drop 21, 25, 26

E

- Editing Model Elements 25
- Elements 98
- Exceptions 89
- Expanding 23
- Export Control 67
- Extend 115
- Extending Rose 5

F

- F5 171
- Features 4
- Files Tab 55, 81
- Filtering Class Relationships 64
- Fit in Window 12
- Floating 22
- Focus of Control
 - Diagrams 174
 - Scripts 167
- Font 45
- Formal Arguments 72
- Framework 217
 - Activating the Framework Wizard Add-In 217
 - Changing 222
 - Creating a Model from a Framework 218
 - Creating a New Framework 219
 - Deleting 222
- Framework Library 219
- Frequency 183
- Friend 102

- Friendship Required 104, 123
- full import 237
- Fully Qualified Names 40

G

- Generalize Specification 103
 - General Tab 104, 122
 - Friendship Required 104, 123
 - Virtual Inheritance 104, 123

H

- Has Relationship 106
- Has Specification 106
 - Detail Tab 108
 - General Tab 107
- Hide Stereotype Name 208
- Hiding Classes 64
- Hiding the Browser 22

I

- Icons 179
 - Copying 42
 - Cutting 42
 - Deleting 42
 - Pasting 42
- importing a type library 233
- Include 115
- Initial Value 83
- Installing an Add-In 5
- Interaction Diagram 11, 91, 92, 93, 157
- interface
 - in type library 227
 - of COM component 227

K

- Keys 103

L

- Language 79, 188, 191
- Link a Diagram 35
- Link Element 99
- Link Specification 175
 - General Tab 176
 - Association 176
 - Roles 178
 - Shared 178
 - Messages Tab 179
 - Icon 179
 - Message Name 180
 - Receiver 180
 - Sequence 179
- Linking 34
- Loading a Model Workspace 49
- Logical Package
 - Assigning 64
- Logical Package Specification 106

M

- Manipulating Icons 41
- Manipulating Text 42
- Maximize Button 8
- Menu Bar 8
- Menu Control Box 8
- Message Name 180
- Message Specification 180
 - Detail Tab 182
 - Frequency 183
 - Synchronization 183
 - General Tab 181
 - Class 181
- Messages 163
- Minimize Button 8
- Model
 - Navigating 24
- Model Elements 37
 - Creating 25
 - Editing 25

- Model files 242
- model workspace 46
- Model Workspace Sample 47
- Moving Icons 42
- Multiple Instances Check Box 173
- Multiple Objects 162

N

- Name 119
- Name Direction 99
- Naming 44
 - an Element on the Diagram 38
 - an Overloaded Element on the Diagram 39
 - Element 26
 - Element in the Browser 26
 - Model Elements 37
- Naming Model Elements 37
- Navigable 101
- Navigating a Model 24
- Navigating the Tabs 58
 - Adding and Deleting Entries 58
 - Editing Entries 58
- Notations 4

O

- Object Browser in Visual Basic 232
- Object Flow 134
- Object Flow Specification 155
- Object Specification 171
 - General Tab 172
 - Class 173
 - Persistence Field 173
- Objects 133
- OLE Viewer 231
- Open Existing Model 10
- Operation Specification 86
 - Detail Tab 88
 - Arguments 88
 - Concurrency 90

- Exceptions 89
- Protocol 88
- Qualifications 89
- Size 89
- Time 89
- Files Tab 94
- General Tab 87
 - Return Class 87
- Post Conditions Tab 93
 - Interaction Diagram 93
- Postconditions Tab
 - Postcondition 93
- Pre Condition Tab
 - Pre Conditions 91
- Pre Conditions Tab 91
 - Pre Conditions 91
- Semantics Tab 92
 - Interaction Diagram 91, 92
 - Semantics 92
- Overloaded Element 39
- Overloading 38

P

- Package 66, 119, 193, 214
- Package Name 79
- Package Specification 192
 - Detail Tab 193
 - Component Diagram 193
 - General Tab 192
 - Package 193
- Parent 12, 66, 97
- Paste 10
- Persistence 70
- Persistence Field 173
- Petal File 242
- Petal files 242
- Positioning 22
- Postcondition 93
- PreConditions 91
- Previous Diagram 12
- Print Diagrams 10

- Priority 203
- Private 67
- Process Specification 202
 - General Tab 203
 - Priority 203
 - Processor 203
- Processes 199
- Processor 195, 203
- Processor Specification 196
 - Detail Tab 198
 - Characteristics 198
 - Processes 199
 - Scheduling 199
 - General Tab 197
- Protected 67
- Protocol 88
- Public 67

Q

- Qualifications 89
- Qualifiers 103
- quick import 237

R

- Rank 119
- Reassigning Model Elements 40
- Receiver 180
- Refine 115
- Relations 121
- Renaming Model Elements 40
- Resizing an Icon 41
- Return Class 87
- Role 98, 178
- Rose.ini File 19

S

- Save Model or Log 10
- Saving 19
- Saving a Model Workspace 49

- Scheduling 199
- Scripts 174
 - Create 167
 - Delete 168
 - Detach 168
 - Move 168
 - Undo 168
- Semantics 92
- Sequence 179
- Sequence Diagram 159
 - Toolbox 170
- Shallow Delete 20
- Shared 178
- Show all Classes 191
- Show Classes 83
- Show labels 208
- Show Stereotype Names 208
- Show Stereotypes 207
- Size 89
- Snap-to-grid 42
- Sorting Packages 29
- Space 69
- Specification
 - Actor 123
 - Association 96
 - Class 65
 - Class Attribute 82
 - Class Instance 174
 - Component 187
 - Connection 201
 - Dependency 105
 - Detail Tab 55
 - Device 200
 - Dialog Boxes 53
 - Displaying 51
 - Files Tab 55
 - General Tab
 - Documentation 54
 - Name 54
 - Generalize 103
 - Has 106
 - Link 175
 - Logical Package 106
 - Message 180
 - Object 171
 - Operation 86
 - Package 192
 - Process 202
 - Processor 196
 - Tab Buttons 57
 - Apply 57
 - Browse 57
 - Cancel 57
 - Help 58
 - OK 57
 - Start and End States 137
 - State Machine 125
 - state transition 137
 - State Transition Specification
 - Detail Tab
 - Transition Between Substates 147
 - state/activity model icon 125
 - statechart 125
 - Statechart diagrams 127
 - States 137
 - Static 85, 102
 - Stereotype 67, 97, 188, 205, 209
 - Benefits 205
 - Creating 209
 - Creating Configuration File 209
 - Creating Icons 211
 - Display 207
 - Icon 211
 - User-Defined 206
 - Viewing 206
 - stereotype 214
 - Stereotype Sample 214
 - subsystem 214
 - Subsystem Stereotype Package 214
 - Supplier Visibility 177
 - Support 243
 - Swimlanes 133
 - Synchronization 183

Synchronizations 138

T

Tab Buttons 57

 Apply 57

 Browse 57

 Cancel 57

 Help 58

 OK 57

Tabs

 Navigating 58

Technical Support 243

Time 89

Title Bar 8

Toolbar 8, 34

Toolbox

 Class Diagram 62

 Collaboration Diagram 159

 Component Diagram 186

 Deployment Diagram 196

 Sequence Diagram 170

Transition Between Substates 147

Transition to Self 138

Transitions 137

Type 66

type libraries 223

Type Library Importer 223

Type-Class Attribute 83

U

Undo Fit in Window 12

Undock 23

Upgrading 241

URLs 56

Use-Case Specification

 Diagram Tab 120

 Diagram 120

 General Tab

 Abstract 119

 Package 119

 Rank 119

 Relations Tab 121

 Relations 121

User Interface 5

User-Defined 206

V

Viewing

 Diagrams 32

 Documentation 11

Virtual Inheritance 104, 123

Visibility 177

W

Workflow Modeling 131

Workflows 129

Workspaces 46

Z

Zoom In/Out 12