

Using Rose

Rational Rose® 2001

VERSION: 2001.03.00

PART NUMBER: 800-023909-000

support@rational.com
<http://www.rational.com>

Rational®
the e-development company™

COPYRIGHT NOTICE

Copyright © 2000 Rational Software Corporation. All rights reserved.

THIS DOCUMENT IS PROTECTED BY COPYRIGHT AND CONTAINS INFORMATION PROPRIETARY TO RATIONAL. ANY COPYING, ADAPTATION, DISTRIBUTION, OR PUBLIC DISPLAY OF THIS DOCUMENT WITHOUT THE EXPRESS WRITTEN CONSENT OF RATIONAL IS STRICTLY PROHIBITED. THE RECEIPT OR POSSESSION OF THIS DOCUMENT DOES NOT CONVEY ANY RIGHTS TO REPRODUCE OR DISTRIBUTE ITS CONTENTS, OR TO MANUFACTURE, USE, OR SELL ANYTHING THAT IT MAY DESCRIBE, IN WHOLE OR IN PART, WITHOUT THE SPECIFIC WRITTEN CONSENT OF RATIONAL.

U.S. GOVERNMENT RIGHTS NOTICE

U.S. GOVERNMENT RIGHTS. Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in the applicable Rational License Agreement and 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.

TRADEMARK NOTICE

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.

Visual C++, Visual Basic, Windows NT, Developer Studio, and Microsoft are trademarks or registered trademarks of the Microsoft Corporation. BasicScript is a trademark of Summit Software, Inc. All other names are used for identification purposes only and are trademarks or registered trademarks of their respective companies.

Portions of Rational Rose include source code from Compaq Computer Corporation; Copyright 2000 Compaq Computer Corporation.

U.S. Registered Patent Nos. 5,193,180 and 5,335,344 and 5,535,329. Licensed under Sun Microsystems Inc.'s U.S. Pat. No. 5,404,499. Other U.S. and foreign patents pending.

Printed in the U.S.A.

Contents

Preface	xxi
Audience	xxi
Other Resources	xxi
Contacting Rational Technical Publications	xxi
Contacting Rational Technical Support	xxii
1 Introduction to Visual Modeling Using Rational Rose	1
Visual Modeling	1
Modeling with Rational Rose	2
Notations	3
Features	3
Extending Rational Rose	4
2 Getting Started with Rational Rose	5
Application Window	6
Title Bar	6
Control-Menu Box	6
Minimize, Restore, and Close Buttons	7
Menu Bar	7
Toolbar	7
Toolbox	10
Customizing the Toolbox	11
Browser	11
Documentation Window	11
Diagram Window	12
Overview Window	13
Specification Window	13
Printing Diagrams and Specifications	14
Print Preview	14
Apply Filter Dialog Box	15
Saving in Various Formats	15
Modifying the Rose.ini File	16
3 The Browser	17
Overview	17
Viewing the Browser	17

Hiding and Displaying the Browser	18
Positioning the Browser	18
Docking and Undocking the Browser	18
Navigating a Model	18
Expanding and Collapsing the Browser Tree	19
Creating and Editing Model Elements	20
Naming an Element in the Browser	20
Selecting Multiple Elements in the Browser	20
Sorting Packages in the Browser	21
Using Drag-and-Drop in the Browser	21
Browser to Browser Capabilities	22
Browser to Diagram Capabilities	23
Browser to Specification Capabilities	24
4 Introduction to Diagrams	25
Overview	25
Diagram Windows	25
Viewing Diagrams	26
Displaying Multiple Diagrams	27
Creating, Linking, Displaying, Renaming, and Deleting Diagrams	28
Creating a New Diagram	28
Linking a Diagram	28
Displaying a Diagram	29
Renaming a Diagram	29
Deleting a Diagram	29
Creating and Naming Model Elements	30
Creating an Element on the Diagram	30
Creating an Element in the Browser	30
Naming Model Elements	30
Reassigning Model Elements	32
Manipulating Icons	33
Selecting Icons	33
Deselecting Icons	33
Resizing an Icon	34
Moving One or More Icons	34
Changing from One Kind of Element or Relationship to Another	34
Cutting, Copying, and Pasting Icons	35
Deleting Model Elements	36
Shallow Delete	36

Deep Delete	36
Correlations	37
Creating Correlations Between Elements	37
Bending a Correlation Icon.	37
Reconnecting a Correlation Icon from One Icon to Another	38
Naming a Correlation.	38
Laying Out a Diagram.	38
Laying Out All Shapes in a Diagram	39
Laying Out Selected Shapes in a Diagram	40
Adorning the Diagrams.	40
Placing Text in a Diagram.	40
Manipulating Text	40
Understanding Model Workspaces.	41
Differences Between a Saved Model and a Model Workspace	41
Model Workspace Scenario	42
Saving a Model Workspace	43
Loading a Model Workspace	43
5 Introduction to Specifications	45
Displaying Specifications	45
Custom Specifications	46
Editing Specifications	46
Common Specification Elements	47
Dialog Boxes	47
General Tab	47
Detail Tab	49
Files Tab.	49
Tab Buttons	51
Navigating the Tabs	52
Adding and Deleting Entries.	52
Editing Entries	52
6 Class Diagrams and Specifications	53
Class Diagram Overview	53
Class Diagram Toolbox	54
Creating and Displaying a Class Diagram	55
Assigning a Class to Another Logical Package	55
Adding and Hiding Classes and Filtering Class Relationships.	55
Class Specification	56
Class Specification—General Tab	56

Type	57
Parent	57
Stereotype	57
Export Control	58
Class Specification—Detail Tab	59
Cardinality	60
Space	60
Persistence	61
Concurrency	62
Abstract	62
Formal Arguments	62
Class Specification—Operations Tab	63
Show Inherited	64
Class Specification—Attributes Tab	65
Class Specification—Relations Tab	67
Class Specification—Component Tab	68
Class Specification—Nested Tab	69
Class Specification—Files Tab	71
Class Attribute Specification	71
Class Attribute—General Tab	72
Class	72
Show Classes	72
Type	72
Initial Value	73
Class Attribute—Detail Tab	73
Containment	73
Static	74
Derived	74
Operation Specification	74
Operation Specification—General Tab	75
Return Type	75
Operation Specification—Detail Tab	76
Arguments	76
Protocol	76
Qualifications	77
Exceptions	77
Size	77
Time	77
Concurrency	77
Operation Specification—Preconditions Tab	78

Preconditions	78
Interaction Diagram	78
Operation Specification—Semantics Tab	79
Semantics	79
Interaction Diagram	79
Operation Specification—Postconditions Tab	80
Postconditions	80
Interaction Diagram	80
Operation Specification—Files Tab	80
Parameter Specification	81
Defining a New Parameter	81
Parameter Specification—General Tab	82
Default	82
Owner	82
Type	82
Association Specification	83
Association Specification—General Tab	83
Parent	83
Stereotype	84
Role	84
Element	84
Association Specification—Detail Tab	84
Derived	85
Link Element	85
Name Direction	85
Constraints	85
Association Specification—Role B General Tab	86
Association Specification—Role A and B Detail Tab	87
Navigable	87
Aggregate	87
Static	88
Friend	88
Containment of	88
Keys/Qualifiers	89
Generalize Specification	89
Generalize Specification—General Tab	89
Friendship Required	90
Virtual Inheritance	90
Realize Specification	90
Realize Specification—General Tab	90

Dependency Specification	91
Dependency Specification—General Tab	91
Has Relationship (Booch Only)	92
Has Specification—General Tab	92
Has Specification—Detail Tab	93
Key/Qualifier Specification	93
Defining a New Key/Qualifier	93
Key/Qualifier Specification—General Tab	94
Owner	94
7 Use-Case Diagrams and Specifications	95
Use-Case Diagram Overview	95
Actors	95
Use Case	96
Flow of Events	97
Relationships	97
Association	97
Dependency	98
Extend Stereotype	98
Include Stereotype	98
Refine Stereotype	99
Generalization	99
Use-Case Diagram Toolbox	100
Use-Case Specification	101
Use-Case Specification—General Tab	101
Name	101
Package	102
Rank	102
Abstract	102
Use-Case Specification—Diagram Tab	102
Diagram List	103
Use-Case Specification—Relations Tab	103
Relations	103
Generalize Specification	104
Generalize Specification—General Tab	104
Stereotype	104
Friendship Required	105
Virtual Inheritance	105
Actor Specification	105

8	State Machine Diagrams and Specifications	107
	Creating and Displaying a State Machine Diagram	107
	State Machine Specification	107
	State Machine Specification—General Tab	108
	Statechart Diagram Overview	108
	Creating a Statechart Diagram	109
	Automatic Transmission Example	110
	Activity Diagram Overview	111
	Using Activity Diagrams	111
	Understanding Workflows	111
	Creating an Activity Diagram	112
	Workflow Modeling	112
	Purposes of Workflow Modeling	112
	Defining a Workflow	113
	Modeling a Workflow with an Activity Diagram	114
	Activity Diagram-Specific Model Elements	115
	Activities	115
	Swimlanes	115
	Objects	115
	Object Flow	116
	Understanding Objects and Object Flows	117
	Changing the State of an Object	118
	Shared State Machine Diagram Model Elements	118
	States	118
	Start and End States	118
	Transitions	119
	Transition to Self	119
	Decisions	119
	Synchronizations	119
	Swimlane Specification	120
	Swimlane Specification—General Tab	120
	State and Activity Specification	121
	State and Activity Specification—General Tab	121
	State and Activity Specification—Actions Tab	122
	Type	122
	Action Expression	123
	State and Activity Specification—Transitions Tab	123
	State and Activity Specification—Swimlanes Tab	124
	Action Specification	124

State Transition Specification	126
State Transition Specification—General Tab	126
Transition Specification—Detail Tab	127
Guard Condition	127
Transition Between Substates	127
Decision Specification	128
Decision Specification—General Tab	128
Decision Specification—Transitions Tab	129
Decision Specification—Swimlanes Tab	130
Synchronization Specification	130
Synchronization Specification—General Tab	131
Synchronization Specification—Transitions Tab	132
Object Specification (Activity Diagrams)	132
Object Specification—General Tab	133
Object Specification—Incoming Object Flows Tab	134
Object Specification—Outgoing Object Flows Tab	135
Object Flow Specification	135
Object Flow Specification—General Tab	136
9 Interaction Diagrams and Specifications	137
Interaction Diagram Overview	137
Creating and Displaying an Interaction Diagram	137
Collaboration Diagrams	138
Sequence Diagrams	139
Toolboxes	140
Collaboration Diagram Toolbox	140
Sequence Diagram Toolbox	141
Common Collaboration and Sequence Diagram Icons	141
Object	141
Messages	142
Message Numbering	144
Assigning an Operation to a Message	144
Collaboration-Specific Toolbox Icons	145
Links	145
Sequence Numbering	145
Top-Level Numbering	146
Hierarchical Numbering	146
Scripts	146
Focus of Control	147
Displaying Focus of Control	148

Coloring Focus of Control	148
Moving the Focus of Control	149
Nested Focus of Control	149
Creating Alternative Diagrams	149
Toggling between Interaction Diagrams	149
Creating a Collaboration Diagram from a Sequence Diagram	150
Creating a Sequence Diagram from a Collaboration Diagram	150
Object Specification (Interaction Diagrams)	150
Object Specification—General Tab	151
Name	151
Class	151
Persistence Field	152
Multiple Instances Check Box	152
Class Instance Specifications	152
Class Instance Specification—General Tab	153
Class	153
Link Specification	154
Link Specification—General Tab	154
Assoc	155
Supplier and Client Visibility	155
Shared	156
Role	156
Link Specification—Messages Tab	157
Icon	157
Sequence	157
Message Name	157
Receiver	158
Message Specification	158
Message Specification—General Tab	158
Class	159
Message Specification—Detail Tab	159
Synchronization	160
Frequency	160
10 Component Diagrams and Specifications	161
Component Diagram Overview	161
Creating and Displaying a Component Diagram	161
Component Diagram Toolbox	162
Assigning a Component to Another Package	162
Component Specification	163

Component Specification—General Tab	163
Stereoype (Component)	164
Language	164
Component Specification—Detail Tab	164
Declarations	165
Component Specification—Realizes Tab	165
Show All Classes	166
Classes	166
Language	166
Component Specification—Files Tab	166
Package Specification	166
Package Specification—General Tab	167
Package	167
Package Specification—Detail Tab	168
Component Diagrams	168
Package Specification—Realizes Tab	168
Package Specification—Files Tab	168
11 Deployment Diagrams and Specifications	169
Deployment Diagram Overview	169
Creating and Displaying a Deployment Diagram	169
Deployment Diagram Toolbox	170
Processor Specification	170
Processor Specification—General Tab	171
Processor Specification—Detail Tab	172
Characteristics	172
Processes	172
Scheduling	173
Device Specification	173
Device Specification—General Tab	174
Device Specification—Detail Tab	175
Connection Specification	175
Process Specification	176
Process Specification—General Tab	176
Processor	177
Priority	177
12 Stereotypes	179
Overview	179
Benefits to Using Stereotypes	179
User-Defined Stereotypes	179

Viewing Stereotypes	180
Diagram Tab	180
Browser Tab	182
Creating Stereotypes	183
Creating a New Stereotype for the Current Model	183
Creating a New Stereotype Configuration File	183
Creating a New Stereotype for All Rose Models	184
Creating Stereotype Icons	185
Creating a Diagram Icon	186
Creating Diagram Toolbox and List View Icons	186
Adding Stereotypes to the Diagram Toolbox	187
Subsystem Stereotype Package	188
Subsystem Stereotype Sample	188
13 Framework Wizard Add-In	189
Activating the Framework Wizard Add-In	189
Creating a New Model from a Framework	190
Creating and Deleting Frameworks	191
The Framework Library	191
Creating a New Framework	192
Changing or Deleting a Framework	193
14 Type Library Importer	195
What Is a Type Library?	195
Why Would I Want to Import Type Libraries into the Model?	196
What COM Components Can Be Imported into the Model?	196
How Is a Type Library Presented?	197
A Type Library in Rational Rose	197
Type Library in the OLE Viewer in Visual Studio	202
Type Library in the Object Browser in Visual Basic	202
Importing a Type Library into the Model	203
Importing a New Version of an Existing Type Library	204
Hiding Type Library Items	205
Show Hidden Items Selected	205
Show Hidden Items Cleared	206
Using an Imported Type Library	207
Adding Class Members to a Quick Import Type Library	207
Customizing the Type Library Importer	208

A Upgrading from a Previous Release 211
Upgrading from Rational Rose 3.0 or Later 211
Upgrading from Releases Prior to Rational Rose 3.0 211
Understanding Petal File Versions 211

Index 213

Figures

Figure 1	Application Window	6
Figure 2	Standard Toolbar	7
Figure 3	Application Window	17
Figure 4	Navigating a Model	19
Figure 5	Browser—Collapsed and Expanded Tree	19
Figure 6	Diagram Window	26
Figure 7	Multiple Diagrams—Cascade Windows	27
Figure 8	Multiple Diagrams—Tiled Windows	27
Figure 9	Selected Elements in a Diagram	33
Figure 10	Example Layout of a Class Diagram	39
Figure 11	Model Workspace Loaded Units	42
Figure 12	General Tab	47
Figure 13	Detail Tab	49
Figure 14	Files Tab	50
Figure 15	Tab Buttons	51
Figure 16	Class Diagram Example	53
Figure 17	Class Diagram Toolbox	54
Figure 18	Class Specification—General Tab	56
Figure 19	Class Specification—Detail Tab	59
Figure 20	Class Specification—Operations Tab	63
Figure 21	Class Specification—Attributes Tab	65
Figure 22	Class Specification—Relations Tab	67
Figure 23	Class Specification—Component Tab	68
Figure 24	Class Specification—Nested Tab	69
Figure 25	Class Attribute—General Tab	72
Figure 26	Class Attribute—Detail Tab	73
Figure 27	Operations Specification—General Tab	75
Figure 28	Operation Specification—Detail Tab	76
Figure 29	Operation Specification—Preconditions Tab	78
Figure 30	Operations Specification—Semantics Tab	79
Figure 31	Operation Specification—Postconditions Tab	80
Figure 32	Parameter Specification—General Tab	82
Figure 33	Association Specification—General Tab	83
Figure 34	Association Specification—Detail Tab	84
Figure 35	Association Specification—Role A and B General Tab	86
Figure 36	Association Specification—Role A and B Detail Tab	87

Figure 37	Generalize Specification—General Tab	89
Figure 38	Realize Specification—General Tab	90
Figure 39	Dependency Specification—General Tab	91
Figure 40	Has Specification—General Tab	92
Figure 41	Has Specification—Detail Tab	93
Figure 42	Key/Qualifier Specification—General Tab	94
Figure 43	Use Case Diagram Toolbox	100
Figure 44	Use-Case Specification—General Tab	101
Figure 45	Use-Case Specification—Diagram Tab	102
Figure 46	Use-Case Specification—Relations Tab	103
Figure 47	Generalize Specification—General Tab	104
Figure 48	State Machine Specification—General Tab	108
Figure 49	Automatic Transmission Example	110
Figure 50	Objects on an Activity Diagram Sample	116
Figure 51	Object Flow Sample	117
Figure 52	CD Player Sample	117
Figure 53	Swimlane Specification—General Tab	120
Figure 54	State and Activity Specification—General Tab	121
Figure 55	State and Activity Specification—Actions Tab	122
Figure 56	State and Activity Specification—Transitions Tab	123
Figure 57	State and Activity Specification—Swimlanes Tab	124
Figure 58	State Transition Specification—General Tab	126
Figure 59	State Transition Specification—Detail Tab	127
Figure 60	Decision Specification—General Tab	128
Figure 61	Decision Specification—Transitions Tab	129
Figure 62	Decision Specification—Swimlanes Tab	130
Figure 63	Synchronization Specification—General Tab	131
Figure 64	Synchronization Specification—Transitions Tab	132
Figure 65	Object Specification—General Tab	133
Figure 66	Object Specification—Incoming Object Flows Tab	134
Figure 67	Object Specification—Outgoing Object Flows Tab	135
Figure 68	Object Flow Specification—General Tab	136
Figure 69	Collaboration Diagram Example	138
Figure 70	Sequence Diagram Example	139
Figure 71	Collaboration Diagram Toolbox	140
Figure 72	Sequence Diagram Toolbox	141
Figure 73	Multiple Object Diagram	142
Figure 74	Focus of Control Diagram Example	148
Figure 75	Object Specification—General Tab	151
Figure 76	Class Instance Specification—General Tab	153
Figure 77	Link Specification—General Tab	154

Figure 78	Link Specification—Messages Tab	157
Figure 79	Message Specification—General Tab	158
Figure 80	Message Specification—Detail Tab	159
Figure 81	Component Diagram Example	161
Figure 82	Component Diagram Toolbox	162
Figure 83	Component Specification—General Tab	163
Figure 84	Component Specification—Detail Tab	164
Figure 85	Component Specification—Realizes Tab	165
Figure 86	Package Specification—General Tab	167
Figure 87	Package Specification—Detail Tab	168
Figure 88	Deployment Diagram Example	169
Figure 89	Deployment Diagram Toolbox	170
Figure 90	Processor Specification—General Tab	171
Figure 91	Processor Specification—Detail Tab	172
Figure 92	Device Specification—General Tab	174
Figure 93	Device Specification—Detail Tab	175
Figure 94	Process Specification—General Tab	176
Figure 95	Options Dialog Box—Diagram Tab	180
Figure 96	Options Dialog Box—Browser Tab	182
Figure 97	Subsystem Stereotype Sample	188
Figure 98	Create New Model Dialog Box	190
Figure 99	Framework Wizard Specification Page	192
Figure 100	Framework Wizard Summary Page	193
Figure 101	Component View of the Microsoft Scripting Runtime Type Library	197
Figure 102	Component Overview Diagram for a Model	197
Figure 103	Logical View of the Microsoft Scripting Runtime Type Library	198
Figure 104	Overview Diagram of the Microsoft Scripting Runtime Type Library	199
Figure 105	OLE Viewer in Visual Studio	202
Figure 106	Object Browser in Visual Basic	203
Figure 107	Type Library with Show Hidden Items Option Selected	205
Figure 108	Type Library with Show Hidden Items Option Cleared	206
Figure 109	COM Properties Dialog Box	208

Tables

Table 1	Print Dialog Box Tabs	14
Table 2	Browser to Browser Capabilities	22
Table 3	Browser to Diagram Capabilities	23
Table 4	Browser to Specification Capabilities.	24
Table 5	Export Control Field Options	58
Table 6	Cardinality Field Options	60
Table 7	Persistence Field Options	61
Table 8	Class Concurrency Options.	62
Table 9	Physical Containment Options.	73
Table 10	Concurrency Field Options	77
Table 11	Containment Field Options	88
Table 12	Persistence Field Options	152
Table 13	Supplier and Client Visibility Options.	155
Table 14	Synchronization Options	160
Table 15	Frequency Options	160
Table 16	Scheduling Field Options.	173
Table 17	COM Stereotypes	200
Table 18	Rational Rose Petal File Version	211

Preface

This manual provides an introduction to Rational Rose. Rational Rose is the visual modeling tool that is part of a comprehensive set of tools that embody software engineering best practices and span the entire software development life cycle. Rational Rose helps improve communication both within teams and across team boundaries, reducing development time and improving software quality.

Audience

This guide is intended for all users of Rational Rose, including administrators, analysts, architects, and developers.

Other Resources

- Online Help is available for Rational Rose and its add-ins. In Rational Rose, select an option from the **Help** menu.
- Manuals for Rational Rose and its add-ins are available. All manuals are available online in either HTML or PDF format. The online manuals are on the *Rational Solutions for Windows Online Documentation* CD.
- A Rational Rose tutorial is available for Rational Rose. The tutorial is on the *Rational Solutions for Windows Online Documentation* CD.
- For more information on training opportunities, see the Rational University Web site: <http://www.rational.com/university>.

Contacting Rational Technical Publications

To send feedback about documentation for Rational products, please send e-mail to our Technical Documentation department at techpubs@rational.com.

Contacting Rational Technical Support

If you have questions about installing, using, or maintaining this product, contact Rational Technical Support as follows:

Your Location	Telephone	Facsimile	E-mail
North America	(800) 433-5444 (toll free) (408) 863-4000 Cupertino, CA	(781) 676-2460 Lexington, MA	support@rational.com
Europe, Middle East, Africa	+31 (0) 20-4546-200 Netherlands	+31 (0) 20-4545-201 Netherlands	support@europe.rational.com
Asia Pacific	+61-2-9419-0111 Australia	+61-2-9419-0123 Australia	support@apac.rational.com

Note: When you contact Rational Technical Support, please be prepared to supply the following information:

- Your name, telephone number, and company name
- Your computer's make and model
- Your operating system and version number
- Product release number and serial number
- Your case ID number (if you are following up on a previously-reported problem)

Introduction to Visual Modeling Using Rational Rose

1

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 you 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: the Unified Modeling Language (UML). The UML provides a smooth transition between the business domain and the computer domain. Using the UML, all members of a design team can work with a common vocabulary, minimizing miscommunication and increasing efficiency.

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. 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 alternative to 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 a class operation or the workflow of a business process.

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 component diagrams that focus on the actual software module organization within the development environment. The deployment architecture is captured in deployment diagrams that map software to processing nodes—showing the configuration of run-time processing elements and their software processes.

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 concrete forms and tools that can be manipulated

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) 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 files (.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 CD ROM or the application 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.

Getting Started with Rational Rose

2

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. For further information on the Framework Wizard, refer to the *Framework Wizard Add-In* on page 189.

Independent of Frameworks, you can use Rational Rose's graphical user interface to display, create, modify, manipulate, and document the elements in a model using these windows:

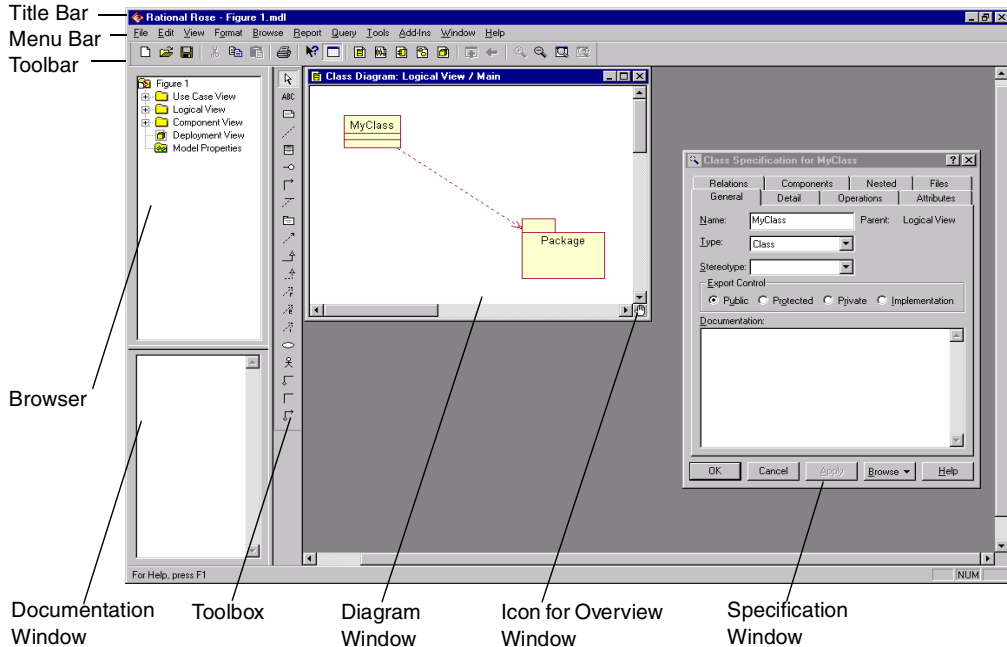
- Application window
- Browser window
- Documentation window
- Diagram window
- Overview window
- Specification window

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

Application Window

An application window contains a title bar, menu bar, toolbar, and a work area where the toolbox, browser, documentation window, diagram window, and specification window appear.

Figure 1 Application Window



Title Bar

The title bar always displays the diagram type. Additional information (like the view or diagram name) is often displayed depending on the diagram/model being viewed. The title bar includes a Control-Menu box, Minimize button, Restore button, and Close button.

Control-Menu Box

Clicking the Control-Menu box (on the application or diagram window) displays a menu with the following commands:

- 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 fit the entire screen.
Close	Closes the window.

Minimize, Restore, and Close Buttons

These buttons allow you to minimize, restore, or close the diagram or application window.

Menu Bar

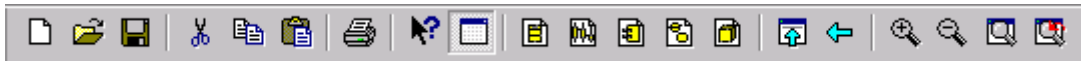
The menu bar changes depending on which diagram you are working. For a description of each menu and command, refer to the Rational Rose online Help.

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.

Figure 2 Standard Toolbar



New Model

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

Open Model

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

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

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 on 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 online Help available. Click this icon and then click the item with which you want help.

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 view more information.

Fit in Window

Clicking the **Fit in Window** icon centers and displays a diagram within the limits of the window. This command changes the zoom factor so that the entire diagram appears.

Undo Fit in Window

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

Help Topics

Clicking the **Help Topics** icon opens 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 **Selector**, **Separator**, and **Lock** icons.

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 separators as you want, but you must have at least one.

Lock Icon

The lock 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. Refer to *Customizing the Toolbox on page 11*.

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. Refer to *Adding Stereotypes to the Diagram Toolbox* on page 187 for more details.

Customizing the Toolbox

To access the **Customize Toolbar** dialog box in order to modify the displayed toolbox:

- 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 the **Toolbars** tab. 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 more information about the browser, refer to *The Browser* on page 17.

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 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 open.

Only one documentation window can be open at a 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 of the docked and floating states of the window are as follows:

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 (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 by clicking and dragging along the border in a vertical or horizontal direction.
- The window title displays the type (class or object) and 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:

- Right-click the icon in either the diagram or browser, and then click **Open Specification** from the shortcut menu.
- Click the icon in either the diagram or browser, and then click **Browse > Specification**.
- Double-click on the icon in either the diagram or browser. (If you have selected the **Double-Click to Diagram** option in the **Options** dialog box, a diagram may appear instead of a specification.)

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. Table 1 describes the tabs in the **Print** dialog box.

Table 1 Print Dialog Box Tabs

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

Print Preview

The print preview option allows 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** 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.
- Zoom Out**
- Print** Click **Print** to display the **Print** dialog box.

One 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.
Two Page	
Close	Click Close to return to an active window.

Apply Filter Dialog Box

The **Apply Filter** dialog box allows 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. For example:

- 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**. Refer to *Modifying the Rose.ini File* on page 16.

Modifying the Rose.ini File

On start-up, Rational Rose extracts information from the registry and from 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. 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. The rose.ini file is stored in these locations:

Operating System	Path
Windows 95/98	Same folder as the Rational Rose executable
Windows NT	C:\WINNT\Profiles\<<username>>\Application Data\Rational\Rose\6.0
Windows 2000	C:\Documents and Settings\<<username>>\Application Data\Rational\Rose\6.0
UNIX	Home directory

- 3 Make your changes to the rose.ini file. For more information on the rose.ini file, refer to the Rose.ini Settings (Overview) topic in the online Help.
- 4 Save and close the rose.ini file.
- 5 Restart Rose.

Overview

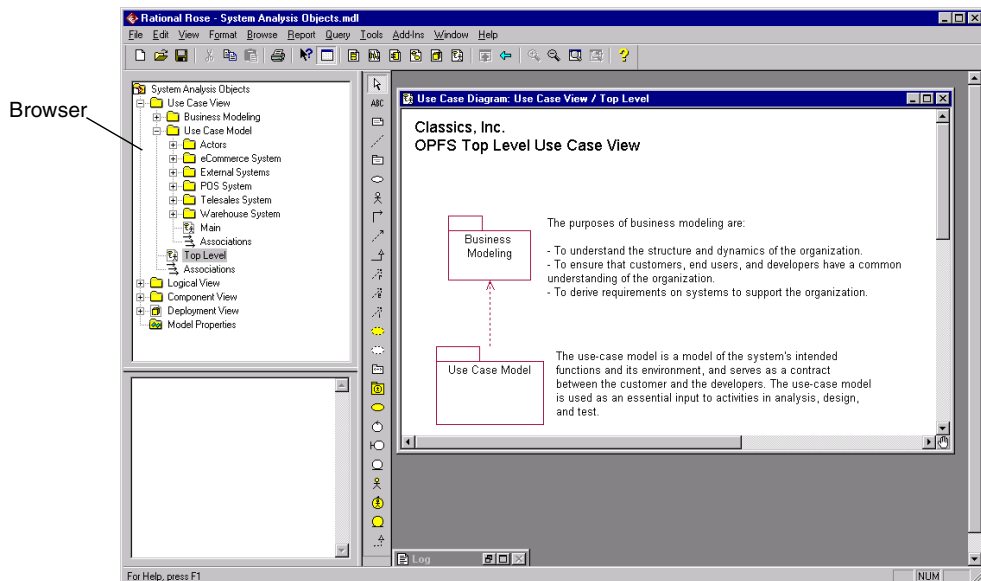
The browser is an easy-to-use alternative to menus and toolbars for visualizing, navigating, and manipulating items within your model. The browser provides:

- 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 a 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, click **View > 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 dock 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.

Navigating a Model

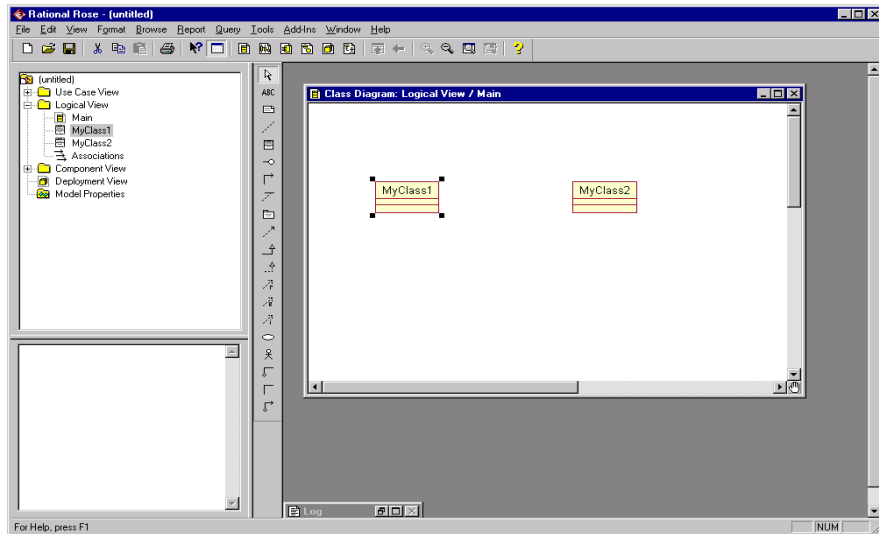
The browser provides a visual representation of your model's hierarchy. As you make changes in a diagram window or in the browser window, the windows remain synchronized:

- 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.

Figure 4 shows MyClass1 highlighted in both the browser and class diagram.

Figure 4 Navigating a Model

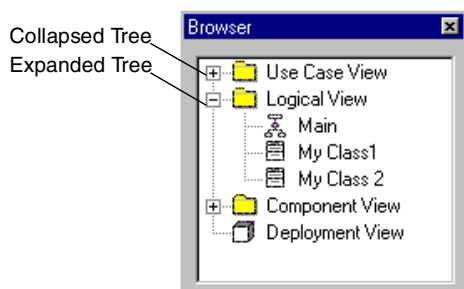


Expanding and Collapsing the Browser Tree

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 5 Browser—Collapsed and Expanded Tree



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 message appears and states 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 either click **Cancel**, which ignores the name, or **OK**. If you do not want to see this dialog box any more, select the **Don't warn anymore this session** check box. To start seeing this dialog box again, restart the application.

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 allows 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.

To select 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: To deselect an item, press the CTRL key and click the item.

To select sequential items:

- 1 Select an item in the browser.
- 2 Hold down the SHIFT key.
- 3 Select another item in the browser. Notice that the browser selects every item between the two items that you selected.

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 and drop all of the packages you want to sort 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.

Note: Your new folder organization is temporary. If the folders are collapsed under the parent icon or Rose is shut down, the packages will be rearranged in alphabetical order the next time that the parent icon is expanded.

Using Drag-and-Drop in the Browser

The drag-and-drop feature allows you to move 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, 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

Table 2 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 ^a	<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

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

Browser to Diagram Capabilities

Table 3 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 ^a	<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

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

Browser to Specification Capabilities

Table 4 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

Overview

Diagrams are views of the information contained in a model. Rational Rose automatically maintains consistency between the diagram and its 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

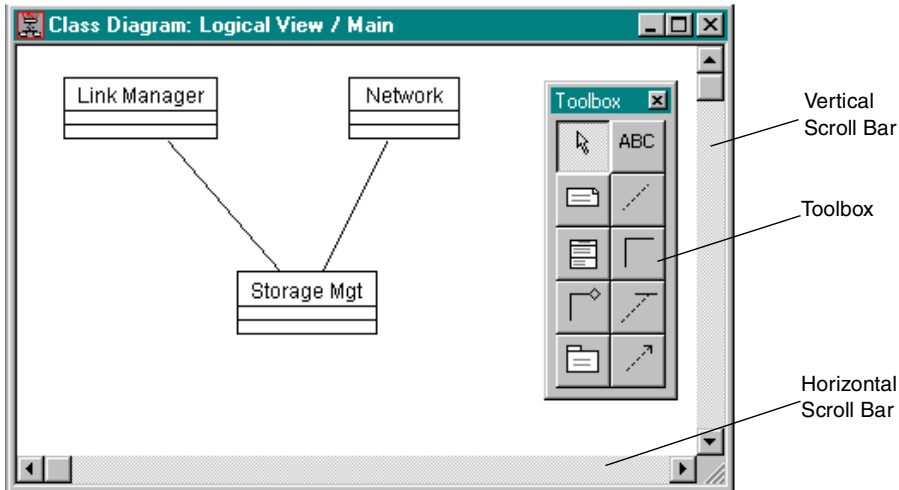
Each icon on a diagram represents an element in the model. Since diagrams illustrate multiple views of a model, each model element can appear in none, one, or several of a model's diagrams. You can control which elements and properties appear on each diagram.

To add icons to a diagram, click **Tools > Create** and click one of the model elements. Click 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 its minimize button.

Displaying Multiple Diagrams

You can display multiple diagrams simultaneously in the application window. To display diagrams in cascaded windows (Figure 7) or tiled windows (Figure 8), click **Window > Cascade** or **Tile**.

Figure 7 Multiple Diagrams—Cascade Windows

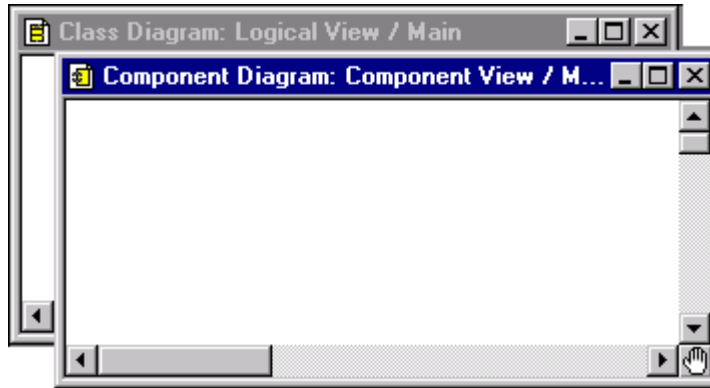
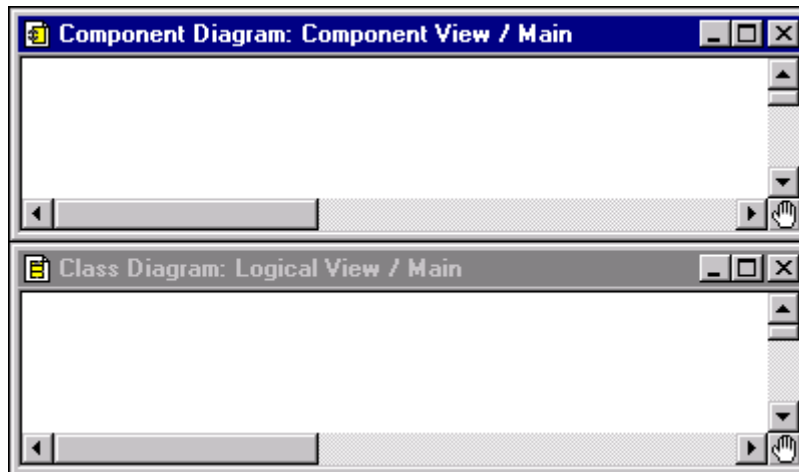


Figure 8 Multiple Diagrams—Tiled Windows



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 it.

Creating, Linking, Displaying, Renaming, and Deleting Diagrams

Creating 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 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). Also, the fully qualified name is displayed in an underline font.

Note: You may need to resize the note to see the entire name.

- 5 Change the text in the note (if desired) to something more meaningful to your project.
- 6 Double-click the note to view the linked diagram.

Displaying 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**.

Renaming a Diagram

Note: You cannot rename a deployment 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 **Rename**.
- 5 Type a new diagram title.
- 6 Click **OK**.

Deleting 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.

Creating and Naming Model Elements

Creating an Element on the Diagram

- 1 Click the appropriate creation tool.
- 2 Click 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.

Creating an Element in the Browser

- 1 Click the appropriate package.
- 2 From the shortcut menu, click **New**, and 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 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 its definition of a class “Customer” do 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).

To name 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 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 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.

To create/name 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 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 the **Don't warn anymore this session** check box. To start seeing the dialog box again, restart the application.

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

To place 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.

To use 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. For example, Logical View::Package B::Class 1 is a fully qualified name.

To rename model elements:

- 1 Click the name of an icon to display the insertion point (flashing vertical bar).

- 2 Backspace and type additional text.

Note: Stereotypes in the form <<stereotype>> are extracted from the name of an item when you edit it.

- 3 Click outside the named icon.

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

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

- 3 Choose the model element that 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 includes selecting, deselecting, moving, and resizing. These features are similar to those you might find in most major drawing tools.

Selecting Icons

To select a single icon:

- Left-click the icon to be selected.

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

To select multiple icons:

- Press and hold the CONTROL or SHIFT key and click each icon to be selected.

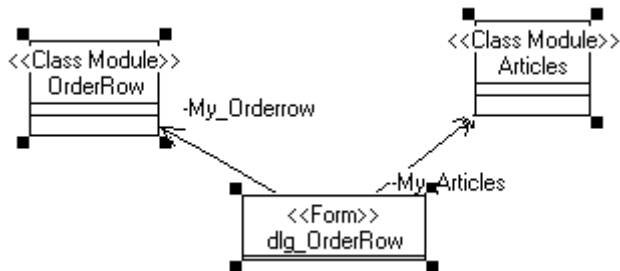
- or -

- 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.

Figure 9 shows multiple elements selected in a diagram:

Figure 9 Selected Elements in a Diagram



Note: You can select any element in the diagram.

Deselecting Icons

To deselect all icons:

- Click any open area of the diagram.

To deselect a specific icon:

- 1 Press and hold the CONTROL or SHIFT key.
- 2 Click the icon.

Other icons that were previously selected remain selected.

Resizing an Icon

- 1 Click 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.

Moving One or More Icons

To move icons using the mouse:

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

To move icons using the keyboard:

- 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**.

Changing from One Kind of Element or Relationship to Another

- 1 Click the toolbox tool bearing the desired icon.
- 2 Press and hold the ALT or META key.
- 3 Click 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.

Cutting, Copying, and Pasting Icons

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

To cut, copy, and paste:

Clicking **Edit > Cut**, **Copy**, or **Paste** can manipulate selections containing icons and text in diagrams, and text information in specification fields. Clicking **Edit > Cut** performs a delete operation on some diagrams and a delete from model operation on others.

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 portions 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.

To use other menu commands:

Clicking **Edit > Undo** reverses the last **Delete**, **Delete From Model**, or **Cut**.

The **Edit** menu also provides commands that allow 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 right-click an icon, Rational Rose displays a shortcut menu. This menu allows you to modify properties (for icons that represent relationships) or select properties to be displayed within the icon.

Deleting Model Elements

There are two ways to delete model elements in Rational Rose: you can perform a shallow delete or a deep delete. A shallow delete removes the element icon 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 icon from a diagram but keep the model element in the model. A shallow delete keeps the model element in the browser and removes the icon of the element from the diagram.

To perform a shallow delete on a selected model element that appears on a diagram:

- Click **Edit > Delete**.
- Press DELETE.

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

Deep Delete

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

To perform a deep delete on a selected diagram model element(s):

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

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

- 1 Click 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 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.

Bending 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.

Reconnecting 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.

Naming a Correlation

To name a newly-created correlation:

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

To change the name of a correlation:

- 1 Click 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.

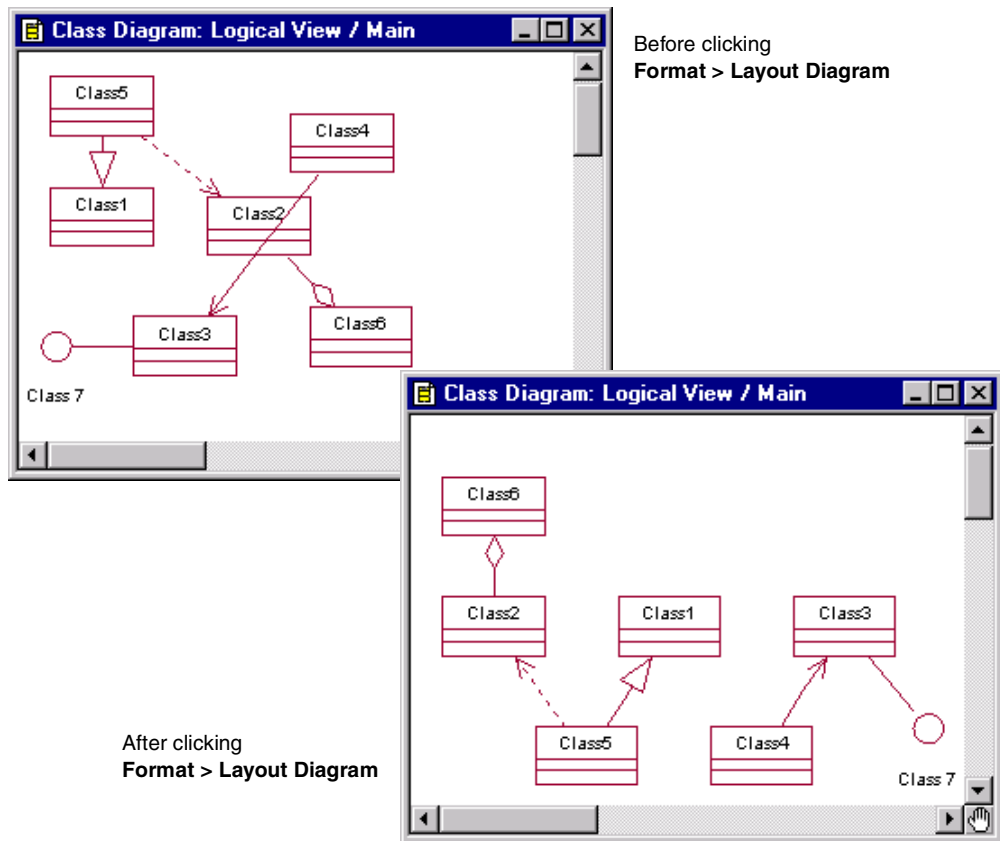
Alternatively, you can change the name in the **Name** field of the specification.

Laying Out a Diagram

When a diagram contains many elements (also called shapes) and many relationships (also called correlations), it can become difficult to read. The layout diagram feature is designed to make a diagram easier to read by rearranging elements on a diagram to clarify their relationships. This is done by minimizing the number of crossed links and positioning shapes in an order that reflects their relationships.

Figure 10 on page 39 provides an example of how the **Layout Diagram** command rearranges classes in a class diagram. Additional information and examples about the feature can be found in the “Layout Diagram (Overview)” topic in the online Help.

Figure 10 Example Layout of a Class Diagram



When you perform a layout command, Rational Rose follows these two rules:

- Shapes that have relationships with other shapes are rearranged based on their relationship(s) in the diagram. Refer to the [Layout Diagram \(Overview\)](#) topic in the online Help for more information.
- Shapes that do not have relationships (called “unconnected shapes”) are placed in one or more rows at the bottom of the diagram or selected area. Examples of unconnected shapes include any element (e.g., class, actor, package) that does not have a relationship to another element, notes that are not attached to elements, and text boxes created with the **ABC Text** tool.

Laying Out All Shapes in a Diagram

- 1 Click **Tools > Options** to display the **Options** dialog box.
- 2 On the **General** tab, set the options under **Layout Options**. For help on an option, click the question mark **?**, and then click the option.

- 3 Click **OK** to save the settings and close the dialog box.
- 4 Click **Format > Layout Diagram**.

Rational Rose rearranges all shapes in the diagram according to the settings specified in the **Options** dialog box.

Note: To undo the layout, select **Edit > Undo**.

Laying Out Selected Shapes in a Diagram

- 1 Select the shapes and relationships in the diagram that you want to rearrange.

Shapes will be rearranged either in an order based on their relationships if the relationships between them are selected or in horizontal rows if only the shapes are selected. If you are working in an activity diagram or a three-tier diagram, all shapes that you select must be within the same swimlane.

- 2 Click **Format > Layout Selected Shapes**.

Rational Rose rearranges the selected shapes within the area of the diagram that they currently occupy.

Note: To undo the layout, select **Edit > Undo**.

Adorning the Diagrams

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

Placing Text in a Diagram

- 1 Choose the **ABC** tool from the toolbox.
- 2 Click a location in the diagram.

Manipulating Text

To change the default font parameters:

- 1 Ensure that nothing is selected by clicking an empty region in any diagram.
- 2 In 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 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 using the mouse:

- 1 Click 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 using the keyboard:

- 1 Click 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.

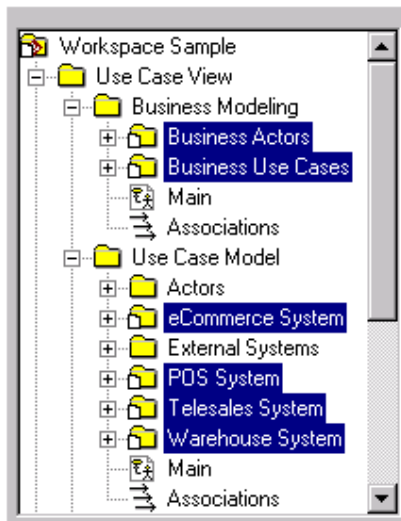
Note: Rational Rose stores all workspace files (*.wsp) in the workspaces folder.

Model Workspace Scenario

The following scenario 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 Figure 11). 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.

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. (See Figure 11.) The workspace configuration will also display some important class and activity diagrams in the diagram window.

Figure 11 Model Workspace Loaded Units



The model workspace will help the new developer by:

- Automatically loading the controlled units for which the developer is responsible
- Displaying some of the more important diagrams the developer should examine first
- Saving the developer time because Rational 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 for greater efficiency.

Saving 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

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

A specification allows 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 an item in a diagram or browser.
- Click a diagram item, and then click **Browse > Specification**.
- Right-click an item, and then 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 a logical or component package, you must turn off the **Double-Click to diagram** option. To disable this option, click **Tools > Options**. Click 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 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 chapter specific to that specification.

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 12 General Tab



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

- Name:** A text input field containing "Logical View".
- Parent:** A text input field that is currently empty.
- Stereotype:** A dropdown menu that is currently empty.
- Documentation:** A large text area with a vertical scrollbar on the right side, currently empty.
- Buttons:** A row of five buttons at the bottom: "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

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**.

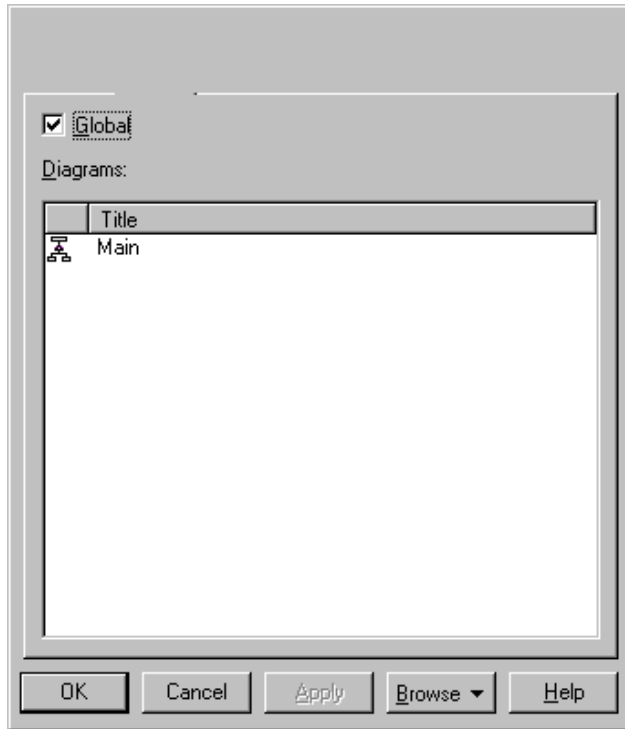
Rational Rose does not display this field in the diagram.

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 placeholder for the responsibilities of the class that you will determine during development.

Detail Tab

The **Detail** tab contains information specific to the model element you have selected.

Figure 13 Detail Tab



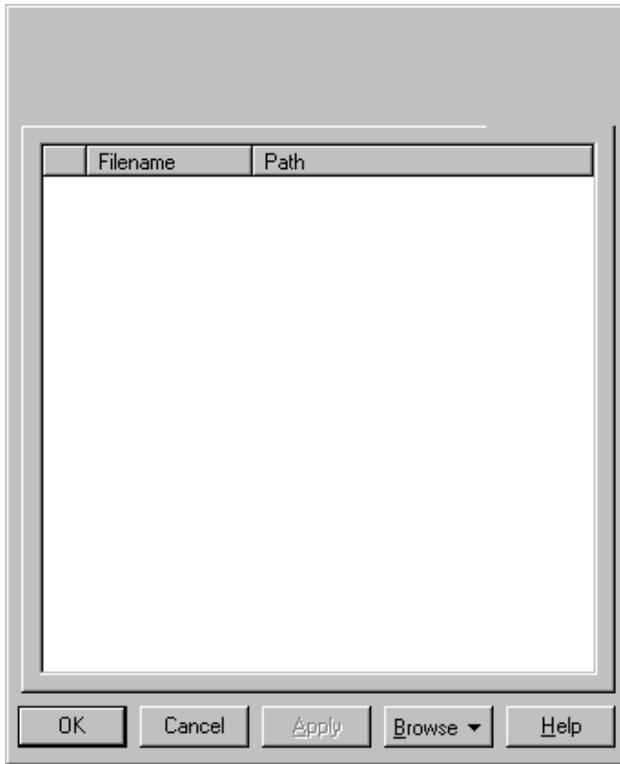
Files Tab

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 (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 14 Files Tab



Viewing Existing Files or URLs

If a file is already inserted, the file name and path are displayed on the tab. To open the document or go the Web site, double-click either the file name or path, or right-click the file name or path, and then click **Open File/URL** from the shortcut menu.

Inserting New Files

You can insert (or attach) files by:

- Using the drag-and-drop technique.
- Right-clicking in the text box, clicking **Insert File** from the shortcut menu, and navigating through the dialog box to locate your file.

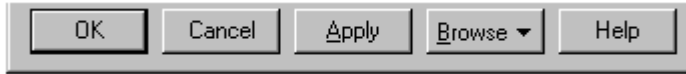
Inserting New URLs

Right-click in the text box, and then click **Insert URL** from the shortcut menu; this will insert the default address www.rational.com. Edit the file name 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 15 Tab Buttons



OK

Clicking **OK** applies the changes made to the specification, closes the dialog box, and returns 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 and leaves the specification open.

Changes to a **Specification** field are not enacted until you click **OK** or **Apply**. These buttons are disabled if the model element is assigned to a controlled unit that 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 that 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. 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 or by pressing the ENTER or TAB key) or cancel the addition (by pressing the ESC key).

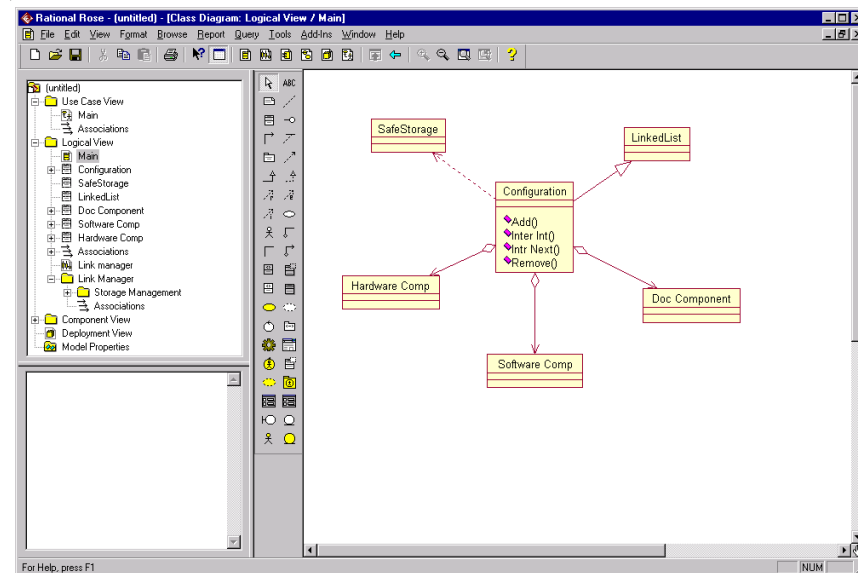
To open the specification for an element displayed in a list, select the row and column and click **Specification** from the shortcut menu or double-click the column. For example, double-clicking the **Name** column in the **Relations** tab of the class specification will open the specification for the relation, while double-clicking 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.

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 16 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 represent logical packages and classes in class diagrams.

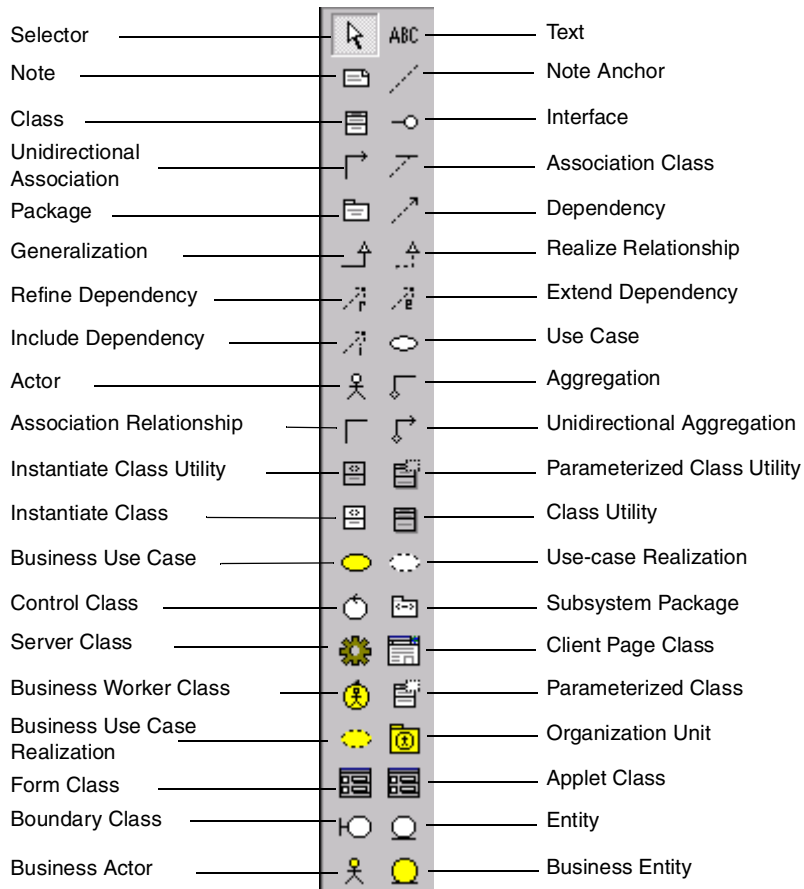
You can change properties or relationships by editing the specification or modifying the icon in the diagram. The associated diagrams or specifications are automatically updated.

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 11* for more information about adding or deleting tools in 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 17 Class Diagram Toolbox



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 the class diagram icon.

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. For example, a 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 updates all class diagrams to reflect the new assignment. Like classes, logical packages are also assigned to logical packages—permitting nesting to an arbitrary depth. You can assign and reassign logical packages and classes.

Adding and Hiding Classes and Filtering Class Relationships

The commands on the **Query** menu allow you to control 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 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**.

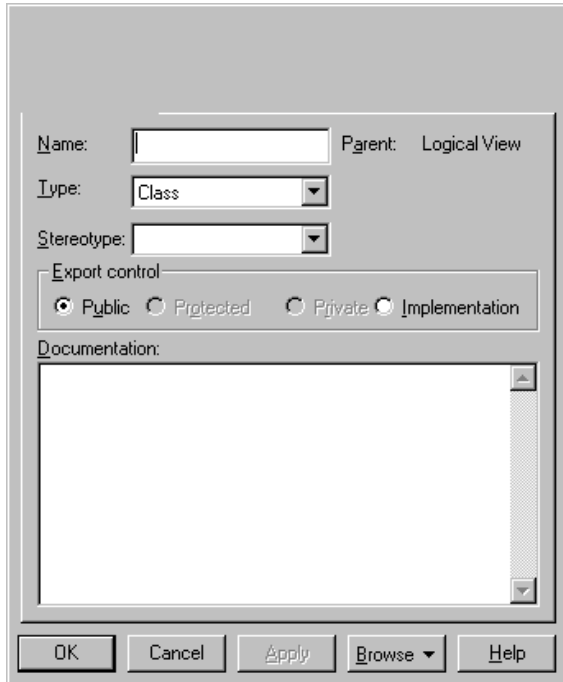
If you have not clicked **Tools > Options** and selected the **Double-Click to diagram** check box, you can double-click any icon representing the class. You can also click **Specification** from the shortcut menu.

Specification Content

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

Class Specification—General Tab

Figure 18 Class Specification—General Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for 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 to which the class belongs (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. You can also define your own stereotypes.

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. Click **Tools > Options** to display the **Options** dialog box. Refer to the *Stereotypes* chapter for more information on stereotypes.

To show stereotypes on the diagrams, right-click a class, and then click **Options > Stereotype Display > None, Label, Decoration, or Icon** from the shortcut menu. These commands display the following information.

Command	Description
None	Stereotype information is not displayed.
Label	The name of the stereotype is displayed between angle brackets (for example, <<stereotype>>).
Decoration	A small icon is displayed in the class icon to indicate the stereotype.
Icon	The class icon is transformed into a stereotype icon.

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

Option	Description
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 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

Figure 19 Class Specification—Detail Tab

Cardinality:

Space:

Persistence

Persistent

Transient

Concurrency

Sequential

Guarded

Active

Synchronous

Abstract

Formal Arguments:

Name	Type	Default Value

OK Cancel Apply Browse Help

Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements 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> ^a	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

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

To display class cardinality on an icon, right-click 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 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.

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 thread 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, select or clear 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.

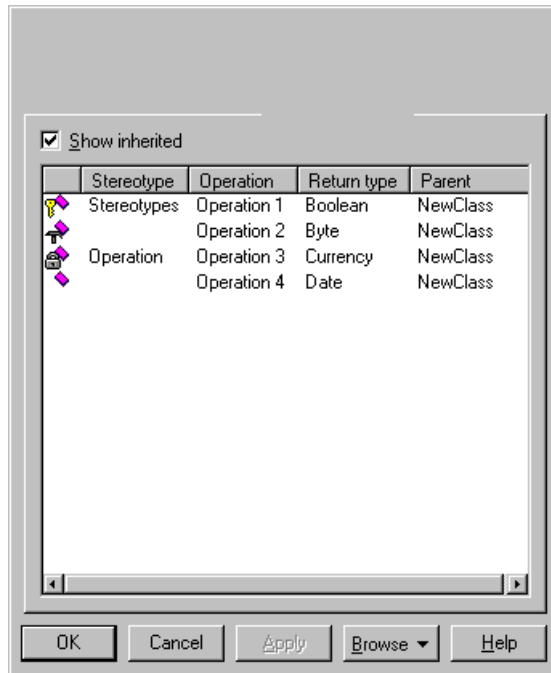
Parameters are displayed on class diagrams.

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 20 Class Specification—Operations Tab







Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements 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 the class that defines the operation.

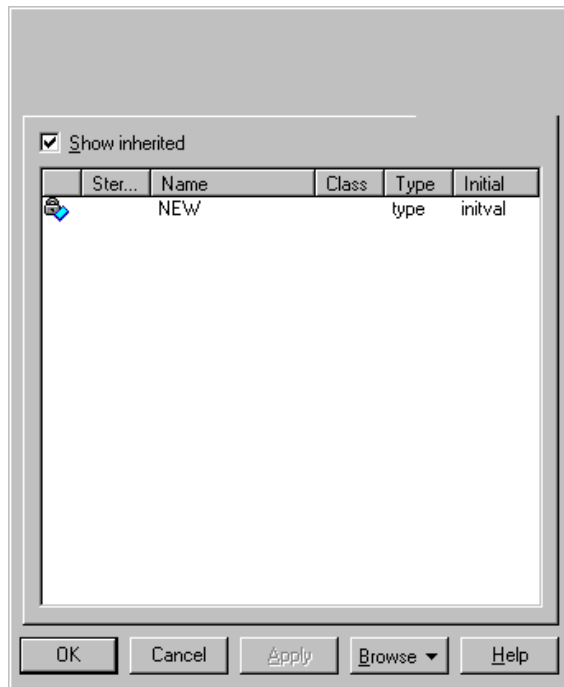
The **Operation** tab is active for all class types. In the class diagram, you can display operation names in the class compartment.

Show Inherited

Select the **Show Inherited** check box 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 21 Class Specification—Attributes Tab



Refer to the descriptions earlier in this chapter and in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

The Rational Unified Process 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 22 Class Specification—Relations Tab



Refer to the descriptions earlier in this chapter or in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

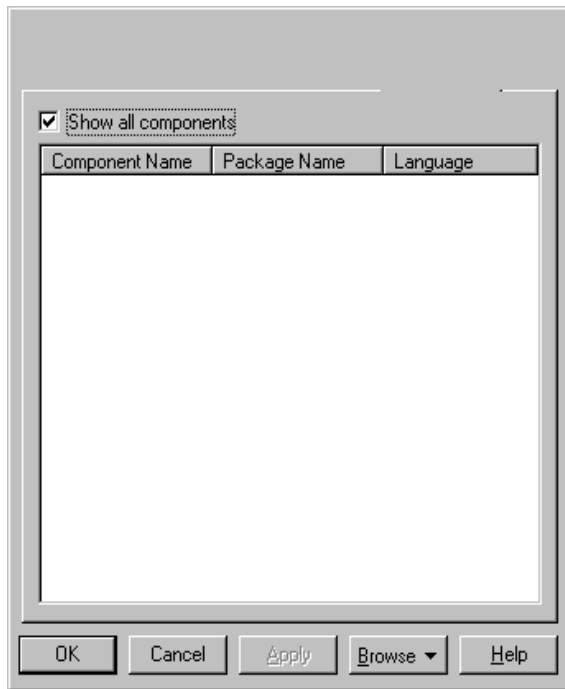
Rational Rose automatically updates this list when you draw relationships between classes.

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 23 Class Specification—Component Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements 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

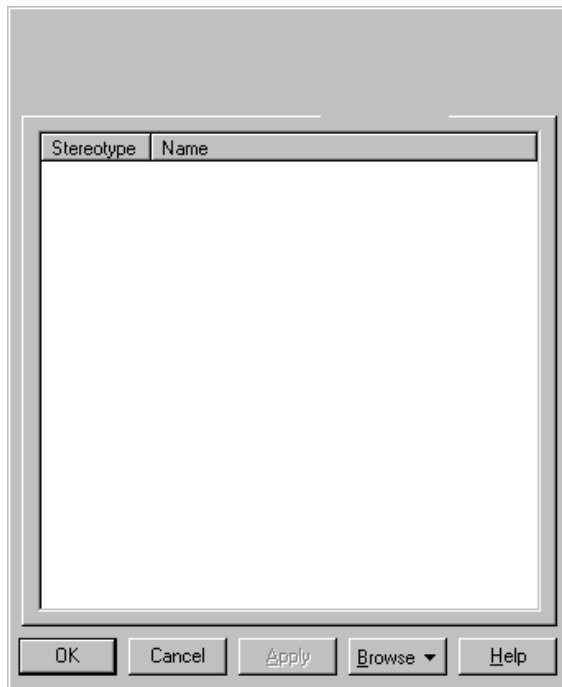
The **Language** field identifies the implementation language assigned to this element.

Note: When you change the implementation language of a component, the data types that are used in the specification of operations or 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 assigned to other components, a dialog box is displayed and asks 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.

Figure 24 Class Specification—Nested Tab



Refer to the descriptions earlier in this chapter and in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

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 to the overall functionality of the computer.

Note: Nested classes can be cut and pasted.

To add a nested class to a class specification:

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

To delete a nested class from a class specification:

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

- or -

- 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 message will appear to verify the deletion.

To relocate 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 *The Browser* chapter.

To move 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 Specifications* chapter for information on this tab.

Class Attribute Specification

A **Class Attribute Specification** allows 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 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

Figure 25 Class Attribute—General Tab

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

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

Refer to the descriptions earlier in this chapter and in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

Class

The class to which the attribute belongs 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

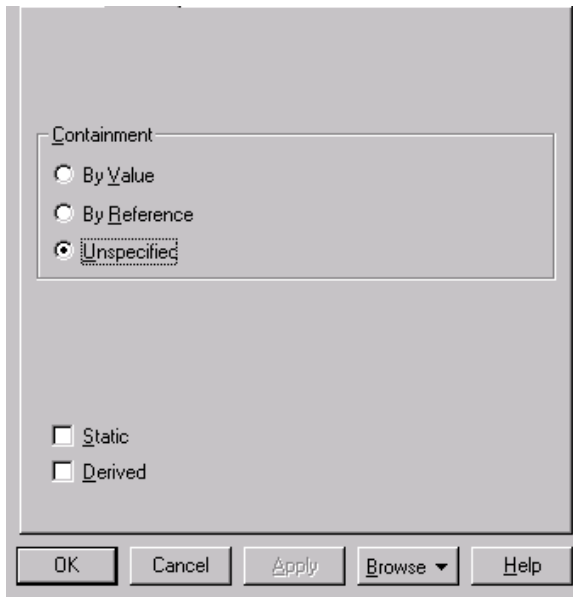
Attribute types can either be classes or language-specific types. When the attribute is a data value, the type is defined as a language-specific type. You can enter the type in the **Type** field of the **Class Attribute Specification**. Rational Rose displays the type beside the attribute name in the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

Class Attribute—Detail Tab

Figure 26 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 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 (derived) 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 the property of a class operation 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 Specification** consists of the following tabs: **General**, **Detail**, **Preconditions**, **Semantics**, **Postconditions**, and **Files**.

Operation Specification—General Tab

Figure 27 Operations Specification—General Tab

The dialog box is titled "Operations 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.
- Show classes:** A checked checkbox.
- 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".

Refer to the descriptions earlier in this chapter or in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

Return Type

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

Figure 28 Operation Specification—Detail Tab

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

Protocol:

Qualification:

Exceptions:

Size:

Time:

Abstract

Concurrency: Sequential Guarded Synchronous

OK Cancel Apply Browse Help

Refer to the descriptions in the *Introduction to Specifications* chapter for 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

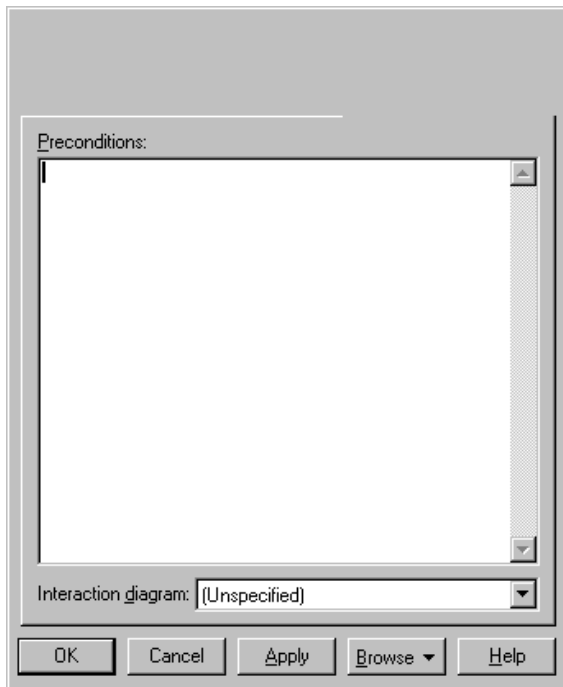
Type	Description
Sequential (default)	The semantics of the operation are guaranteed only in the presence of a single thread of control. Only one thread of control can be executing in the method at any one time.
Guarded	The semantics of the operation are guaranteed in the presence of multiple threads 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 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 29 Operation Specification—Preconditions Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

Preconditions

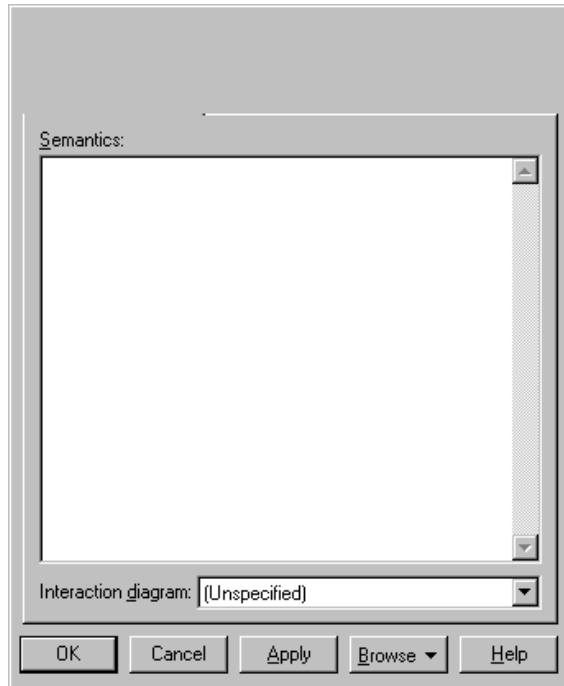
Invariants that are assumed by the operation (the entry behavior of an operation) are listed.

Interaction Diagram

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

Operation Specification—Semantics Tab

Figure 30 Operations Specification—Semantics Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for 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 31 Operation Specification—Postconditions Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

Postconditions

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 Specifications* chapter for information on this tab.

Parameter Specification

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

Specification Content

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

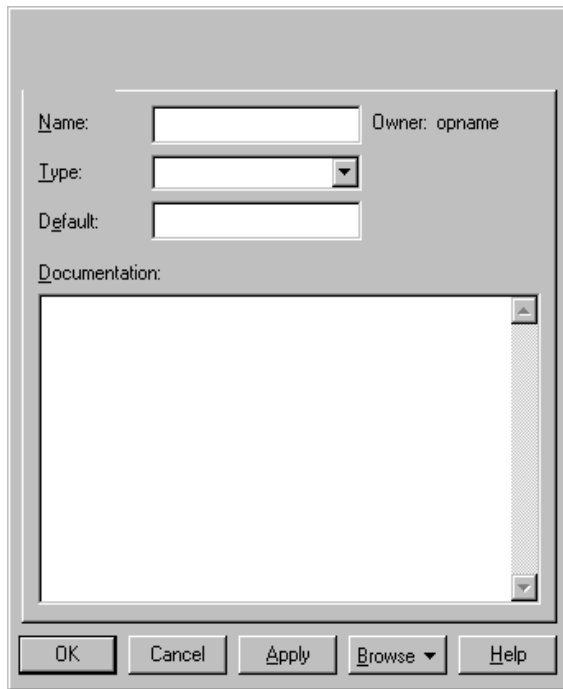
Defining a New Parameter

To display a Parameter Specification:

- 1 From a **Class Specification Operation Tab**, double-click an operation to display the **Operation Specification**.
- 2 Click 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 the argument to display the **Parameter Specification**.

Parameter Specification—General Tab

Figure 32 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 with a small cursor icon on the left. To its right, the text "Owner: opname" is displayed.
- Type:** A dropdown menu with a small downward-pointing arrow on the right side.
- Default:** A text input field.
- Documentation:** A large, empty 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".

Refer to the descriptions in the *Introduction to Specifications* chapter for 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.

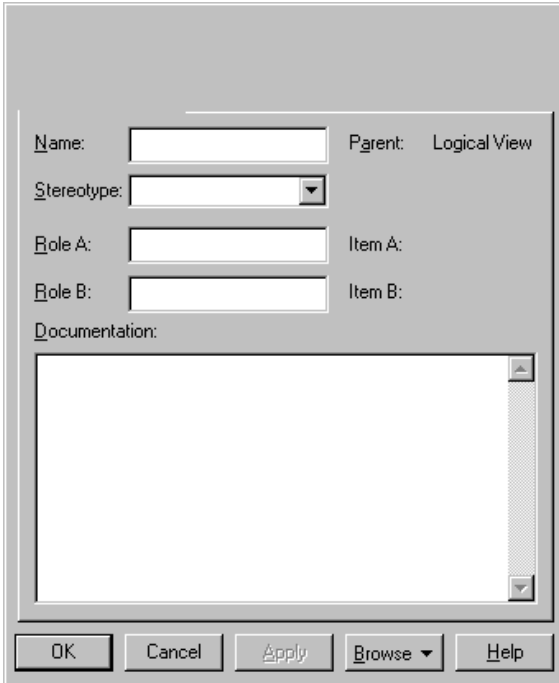
To display the association specification, double-click any icon representing the processor or click **Browse > Specifications**.

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

Figure 33 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 "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:** "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

Parent

The parent to which the component belongs (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 predefined. You can define your own stereotypes.

Role

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

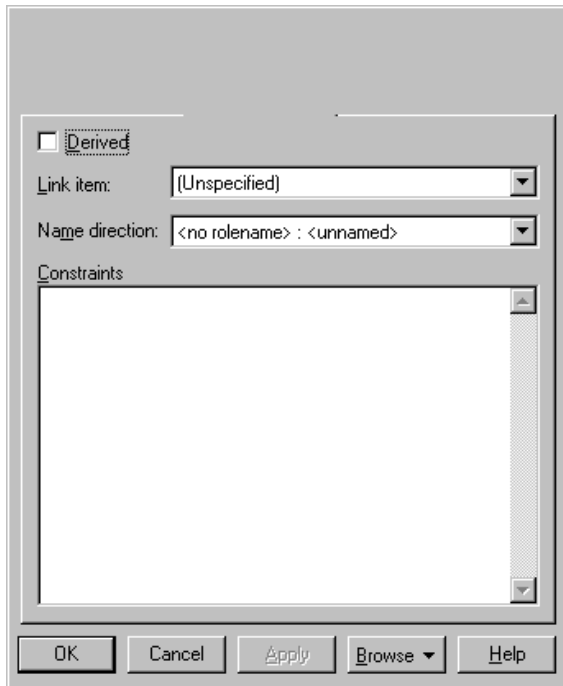
To enter a role name, click in the **Role** field and enter the text.

Element

The **Element** field describes the two elements linked by this association. This field cannot be edited.

Association Specification—Detail Tab

Figure 34 Association Specification—Detail Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

Derived

This field indicates whether the element was computed (derived) 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 an association name.

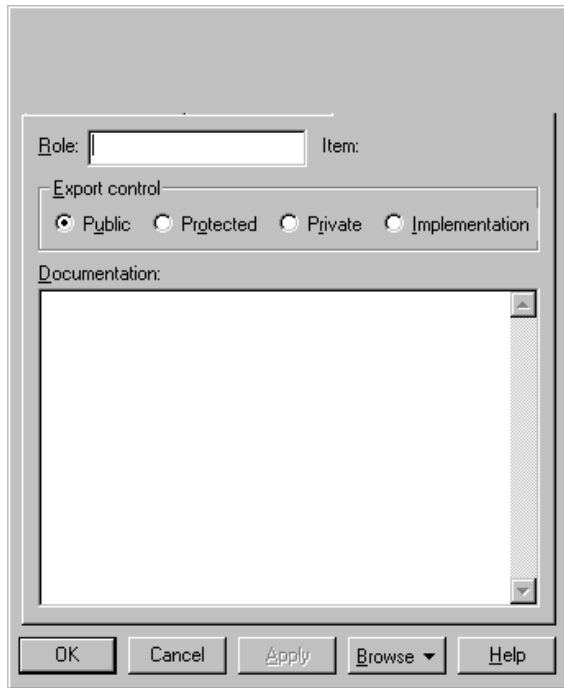
Constraints

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 the **Constraint** field and enter the text. A constraint is displayed notationally, surrounded by braces under the role to which it applies.

Association Specification—Role B General Tab

Figure 35 Association Specification—Role A and B General Tab



Refer to the descriptions earlier in this chapter or in the *Introduction to Specifications* chapter for information on the specification elements.

Association Specification—Role A and B Detail Tab

Figure 36 Association Specification—Role A and B Detail Tab

The dialog box is titled "Association Specification—Role A and B Detail Tab". It contains the following elements:

- Role:** A text input field.
- Item:** A text input field.
- Constraints:** A large empty text area with a vertical scrollbar.
- Cardinality:** A dropdown menu.
- Checkboxes:** **Navigable**, **Aggregate**, **Static**, and **Friend**.
- Containment of:** Radio buttons for **By value**, **By reference**, and **Unspecified**.
- Keys / Qualifiers:** A table with two columns: **Name** and **Type**. The table is currently empty.
- Buttons:** **OK**, **Cancel**, **Apply**, **Browse** (with a dropdown arrow), and **Help**.

Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

Navigable

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

To set a role's navigation, select the **Navigable** check 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, select the **Aggregate** check 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** check box designates that the supplier class has granted rights to a client class to access its non-public parts.

You can select this check box 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 Specification* on page 93.

Generalize Specification

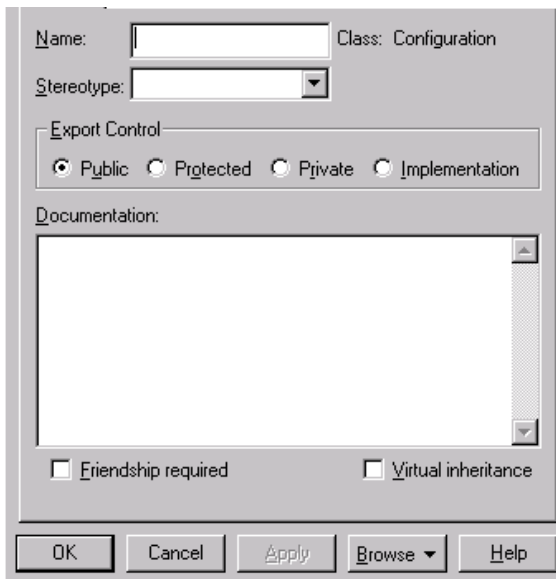
A generalize relationship between classes shows that one class shares the structure or behavior defined in one or more other classes.

Specification Content

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

Generalize Specification—General Tab

Figure 37 Generalize Specification—General Tab



The screenshot shows a dialog box titled "Generalize Specification—General Tab". It has a "Name:" text box and a "Class:" label with the value "Configuration". Below that is a "Stereotype:" dropdown menu. The "Export Control" section contains four radio buttons: "Public" (selected), "Protected", "Private", and "Implementation". The "Documentation:" section is a large empty text area. At the bottom, there are two checkboxes: "Friendship required" and "Virtual inheritance", both of which are unchecked. The dialog box has standard buttons: "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

Refer to the descriptions earlier in this chapter or in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

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.

Realize Specification

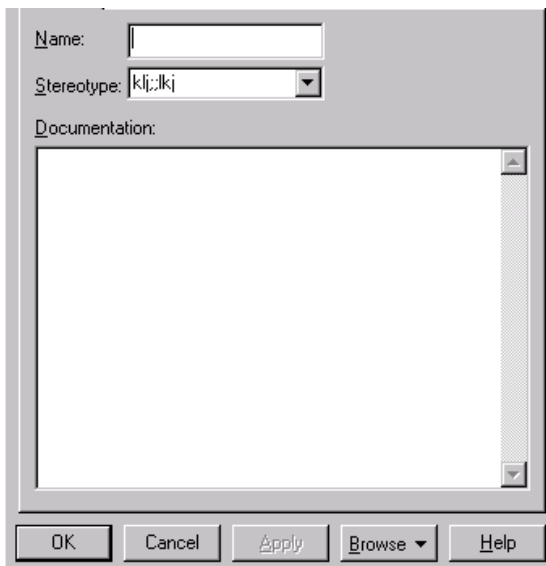
A realize relationship connects a class to an interface or a component to an interface.

Specification Content

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

Realize Specification—General Tab

Figure 38 Realize Specification—General Tab



Refer to the descriptions earlier in this chapter or in the *Introduction to Specifications* chapter for information on the specification elements.

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** in Figure 39. Only the name, class, stereotype, and documentation fields are present.

Specification Content

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

Dependency Specification—General Tab

Figure 39 Dependency Specification—General Tab

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

- Name:** A text input field.
- Class:** A dropdown menu with "Configuration" selected.
- Stereotype:** A dropdown menu.
- Friendship required:** A checkbox, currently unchecked.
- Export Control:** A group box containing four radio buttons: "Public" (selected), "Protected", "Private", and "Implementation".
- Multiplicity from:** A dropdown menu.
- Multiplicity to:** A dropdown menu.
- Documentation:** A large text area with a vertical scrollbar.
- Buttons:** "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

Refer to the descriptions earlier in this chapter or in the *Introduction to Specifications* chapter for information on the specification elements.

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.

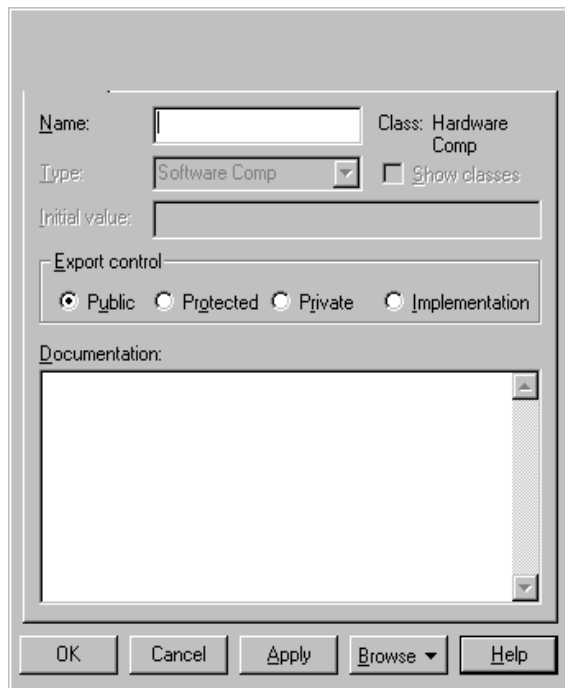
To display a *has* relationship’s specification, select any icon representing the *has* relationship and either double-click or click **Browse > Specifications**.

Specification Content

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

Has Specification—General Tab

Figure 40 Has Specification—General Tab



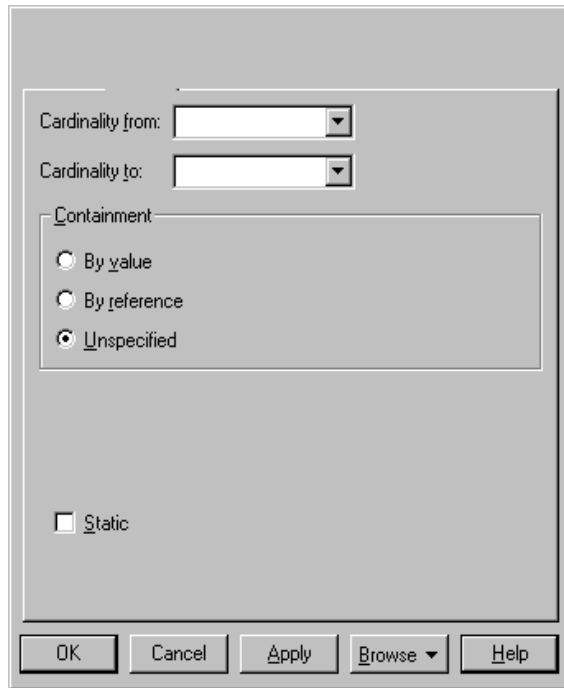
The screenshot shows a dialog box titled "Has Specification—General Tab". It contains the following fields and controls:

- Name:** A text input field.
- Class:** A label with the text "Hardware Comp".
- Type:** A dropdown menu currently showing "Software Comp".
- Show classes:** An unchecked checkbox.
- Initial value:** A text input field.
- Export control:** A group box containing four radio buttons: "Public" (selected), "Protected", "Private", and "Implementation".
- Documentation:** A large text area with a vertical scrollbar.
- Buttons:** "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

Refer to *Class Attribute—General Tab* on page 72 for more information.

Has Specification—Detail Tab

Figure 41 Has Specification—Detail Tab



The screenshot shows a dialog box titled "Has Specification—Detail Tab". It contains the following elements:

- Two dropdown menus: "Cardinality from:" and "Cardinality to:".
- A section titled "Containment" with three radio buttons: "By value", "By reference", and "Unspecified". The "Unspecified" option is selected.
- A checkbox labeled "Static" which is currently unchecked.
- A row of five buttons at the bottom: "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

Refer to the descriptions earlier in this chapter or in the *Introduction to Specifications* chapter for information on the specification elements.

Key/Qualifier Specification

A **Key/Qualifier Specification** allows 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:

- 1 Double-click an association or aggregation.
- 2 From either the **Association Specification** or the **Aggregation Specification**, click the **Role A Detail** or **Role B Detail** tab.
- 3 Move the pointer to the **Key/Qualifier** section of either specification.

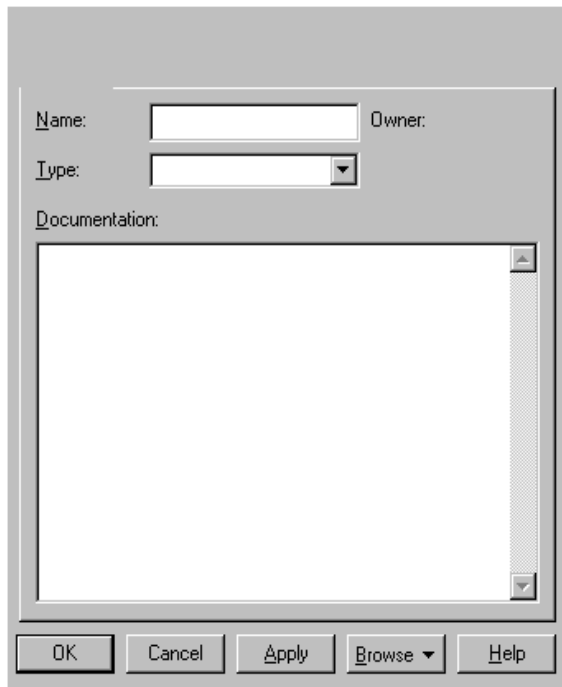
- 4 Right-click to display the shortcut menu.
- 5 Click **Insert** to add a key/qualifier.
- 6 Double-click the entry to display the **Key/Qualifier Specification**.

Specification Content

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

Key/Qualifier Specification—General Tab

Figure 42 Key/Qualifier Specification—General Tab



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

- Name:** A text input field.
- Owner:** A text input field.
- Type:** A dropdown menu.
- Documentation:** A large text area with a vertical scrollbar.
- Buttons:** A row of five buttons at the bottom: **OK**, **Cancel**, **Apply**, **Browse** (with a dropdown arrow), and **Help**.

Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

Owner

The **Owner** field identifies the name, or owner, of the role from which the key/qualifier evolved.

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.

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.

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. Flows 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 the *Files Tab* on page 49 for a discussion 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 flows of events.

Relationships

Relationships show interactions between actors and use cases. Association, dependency, and 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 are typically verbs or verb phrases and are 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 the association to view the **Association Specification**. Click the appropriate **Role A** (or **B**) **Detail** tab, select the **Navigable** check box, 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 in the use-case toolbox. See *Customizing the Toolbox on page 11* for information on adding or deleting diagram toolbox tools.

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 generalization 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.

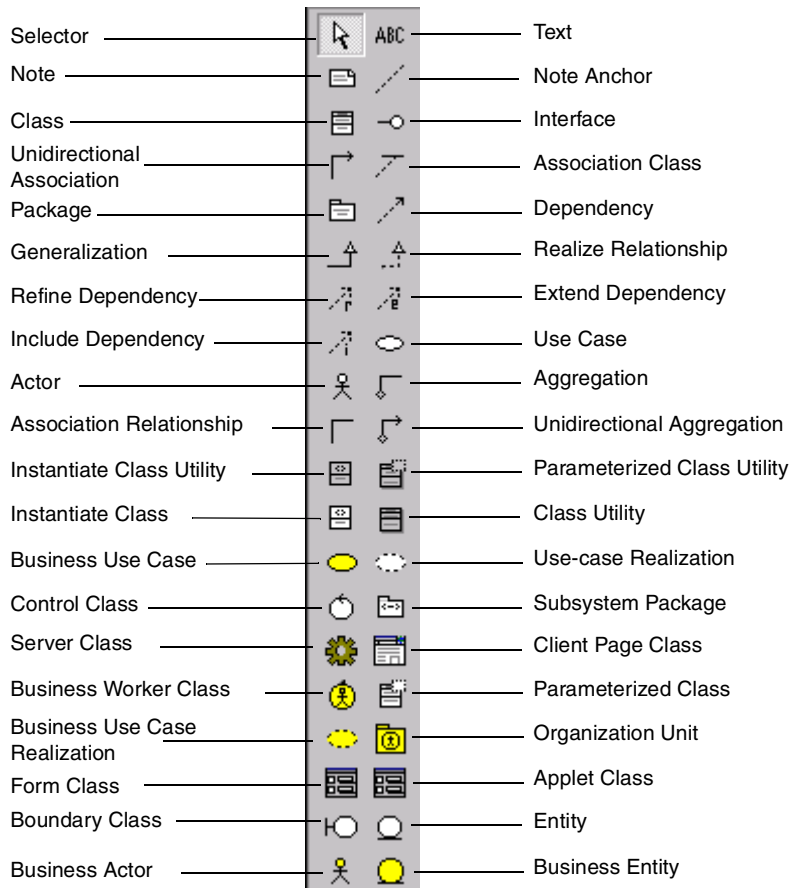
Use-Case Diagram Toolbox

The graphic below shows all the tools that can be placed on the use-case diagram toolbox. Refer to “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 43 Use Case Diagram Toolbox



Use-Case Specification

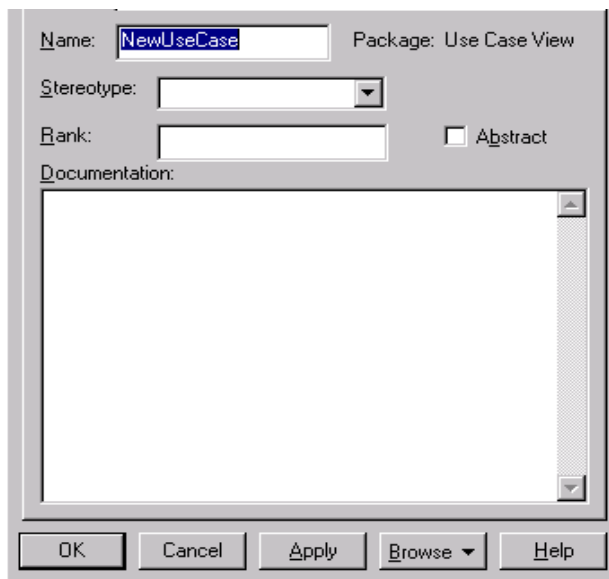
A **Use-Case Specification** allows you to display and modify the properties and relationships of a use case in the current model.

Specification Content

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

Use-Case Specification—General Tab

Figure 44 Use-Case Specification—General Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification 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

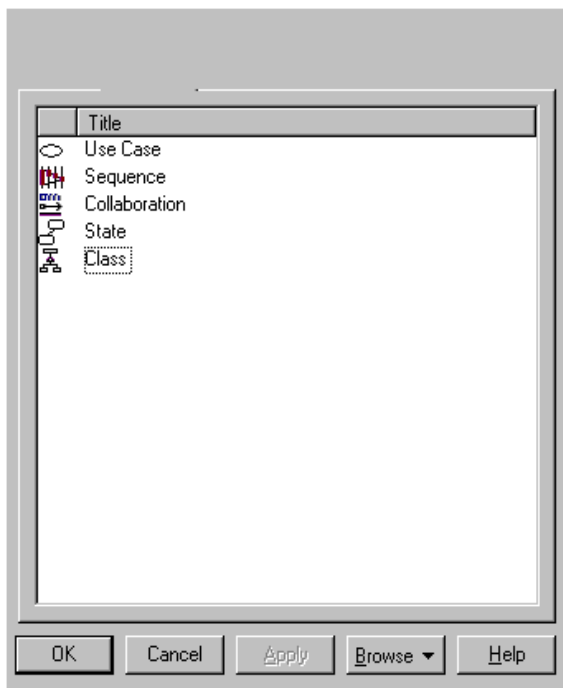
The **Rank** field prioritizes use cases. For example, you can use the rank field to plan the iteration in the development cycle at which a use case should be implemented.

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 45 Use-Case Specification—Diagram Tab



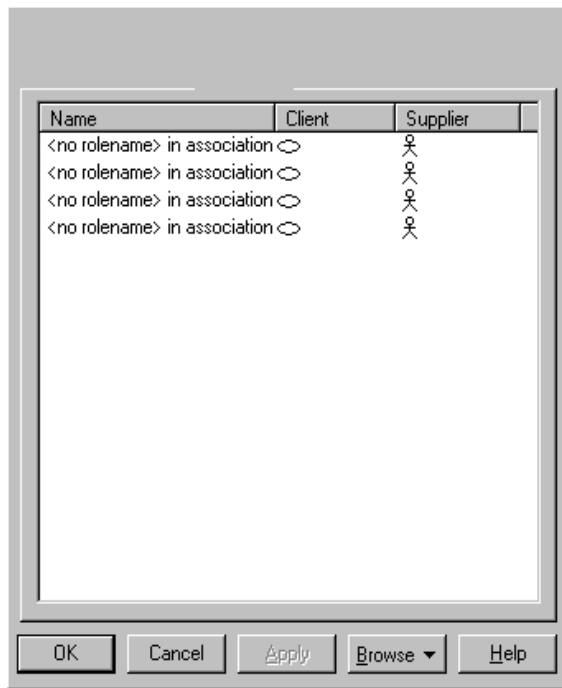
Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

Diagram List

The diagram list contains 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 46 Use-Case Specification—Relations Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for 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

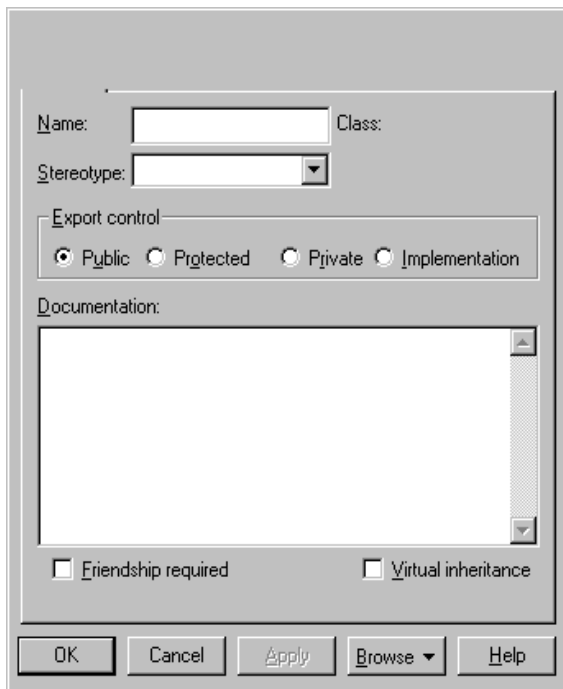
A **Generalize Specification** allows you to display and modify the properties and relationships of a use case in the current model.

Specification Content

The **Generalize Specification** contains the **General** tab.

Generalize Specification—General Tab

Figure 47 Generalize Specification—General Tab



Refer to the descriptions earlier in this chapter or in the *Introduction to Specifications* chapter for information on the specification 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 that 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** is similar 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 *Class Specification* on page 56 for more information.

State Machine Diagrams and Specifications

8

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:

- 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** allows 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.

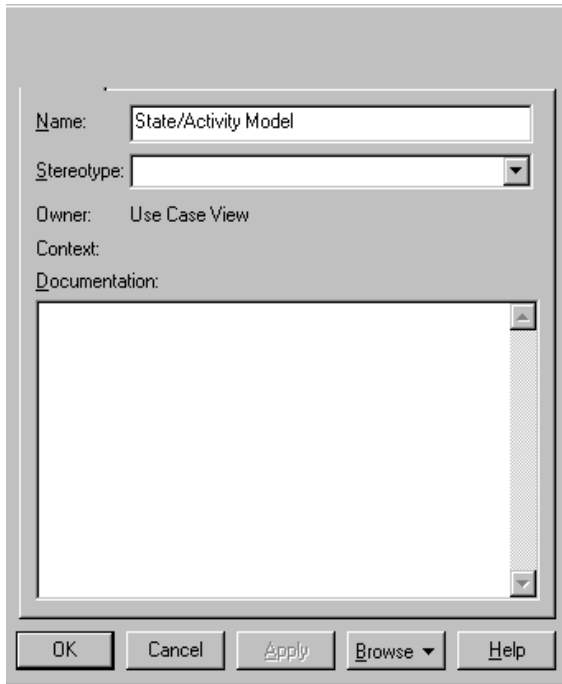
Changes made either through the specification or directly on the icon are automatically updated throughout the model.

Specification Content

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

State Machine Specification—General Tab

Figure 48 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:

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

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 49 Automatic Transmission Example

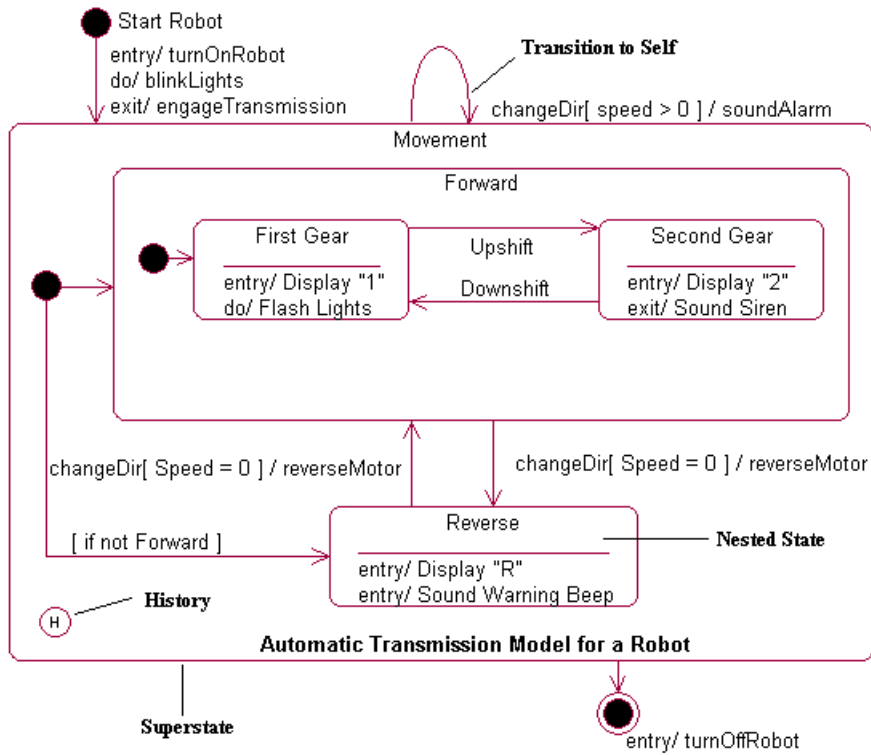


Figure 49 illustrates some of the major model elements in a statechart diagram:

- 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 of approvals for 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
- States
- Swimlanes
- Synchronizations
- Objects
- Transitions
- Object flows
- Start state
- Activities
- 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:

- 1 In the browser, right-click 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.

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 overall responsibility for 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 end of the workflow. Start 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 Define 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

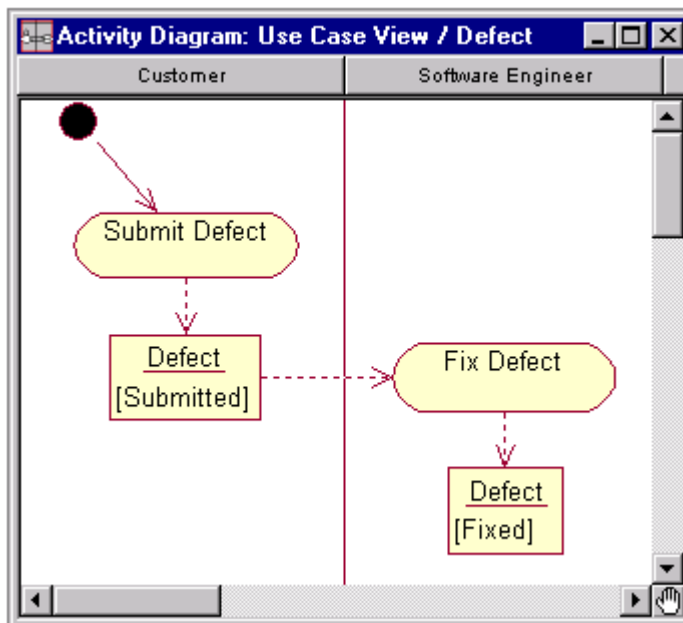
Swimlanes are helpful when modeling a business workflow because they can represent organizational units or roles within a business model. Swimlanes are very similar to objects because they provide a way to tell who is performing a certain role. Swimlanes only appear on activity diagrams. You should place 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 swimlane views appear between 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 Figure 50 on page 116, the `Submit Defect` and `Fix Defects` activities can be thought of as the verbs and the `Defect` objects can be thought of as the nouns in the activity diagram vocabulary. Objects are connected to activities through object flows.

Figure 50 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.

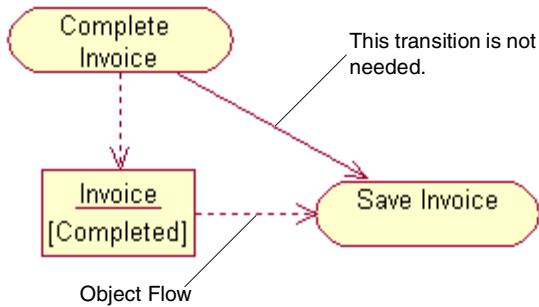
Object Flow

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 sample in Figure 51 on page 117 does not require a transition because the transition is redundant.

Figure 51 Object Flow Sample

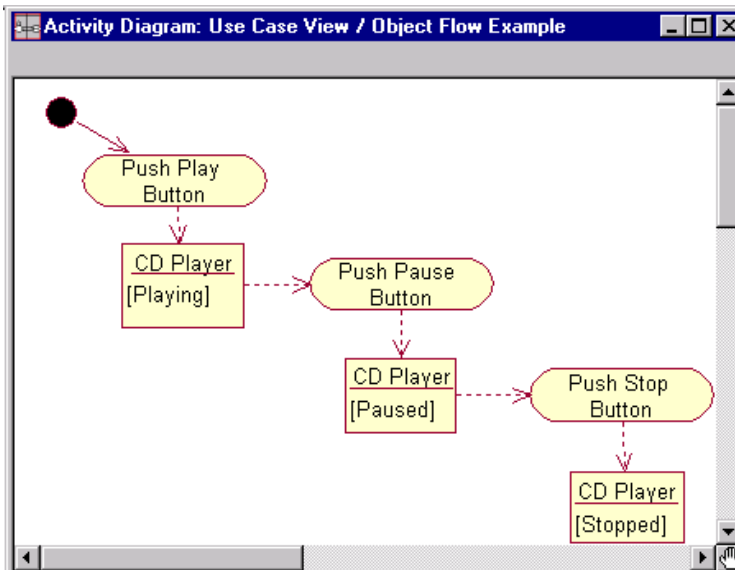


Understanding Objects and Object Flows

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 states.
- Objects connect with activities through object flows.

Figure 52 CD Player Sample



In Figure 52 on page 117, 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 the 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 the object to display the **Object Specification**.
- 2 Select **New** from the **State** list.
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

This section describes the elements that can appear in both activity diagrams and statechart diagrams:

States

A state represents a condition or situation in 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 or 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 cannot 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 allow 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** allows you to display and modify the properties and relationships of a swimlane on an activity diagram.

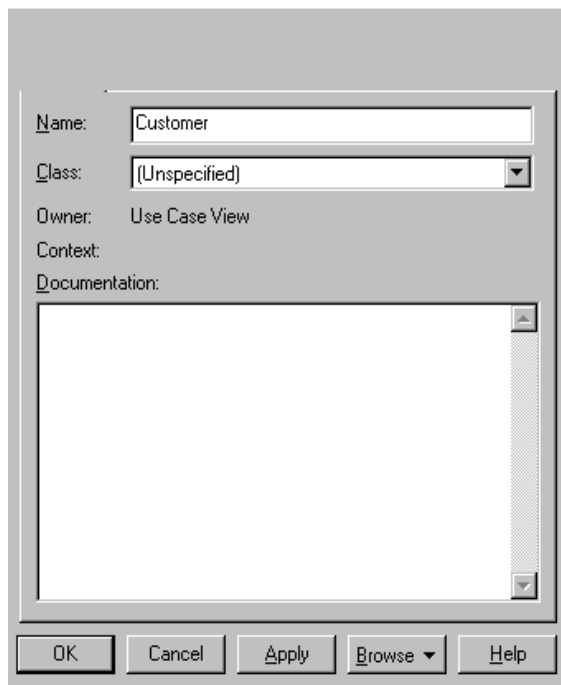
To display a **Swimlane Specification**, select the swimlane header on an activity diagram and double-click. You may also double-click on the swimlane icon in the browser.

Specification Content

The **Swimlane Specification** consists of the **General** tab.

Swimlane Specification—General Tab

Figure 53 Swimlane Specification—General Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements.

State and Activity Specification

A **State and Activity Specification** allows 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 different purposes. Start states and end states use the same specifications as states because they are a type of state. 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 54 State and Activity Specification—General Tab

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

- Name:** A text input field.
- Stereotype:** A dropdown menu.
- Owner:** A text field containing "Use Case View".
- Context:** A text field.
- Documentation:** A large text area with a vertical scrollbar.
- State/activity history**
- Sub state/activity history**
- Buttons: **OK**, **Cancel**, **Apply**, **Browse** (with a dropdown arrow), and **Help**.

Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

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

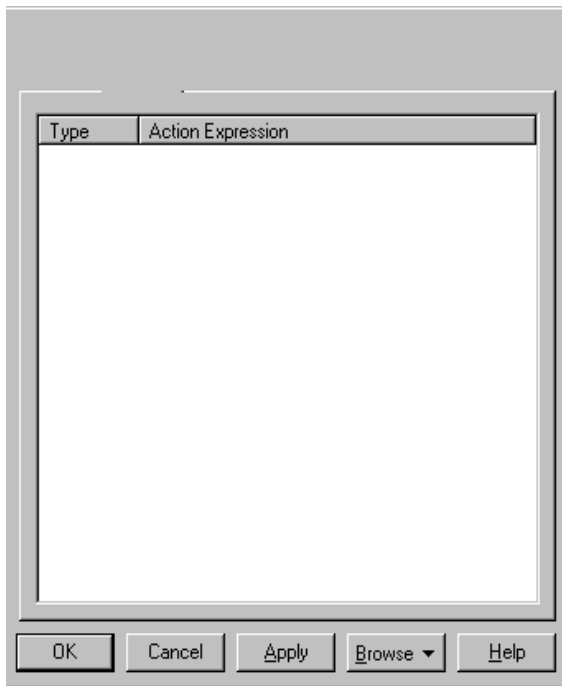
State/Activity History

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 55 State and Activity Specification—Actions Tab



Type	Action Expression
------	-------------------

OK Cancel Apply Browse Help

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**.

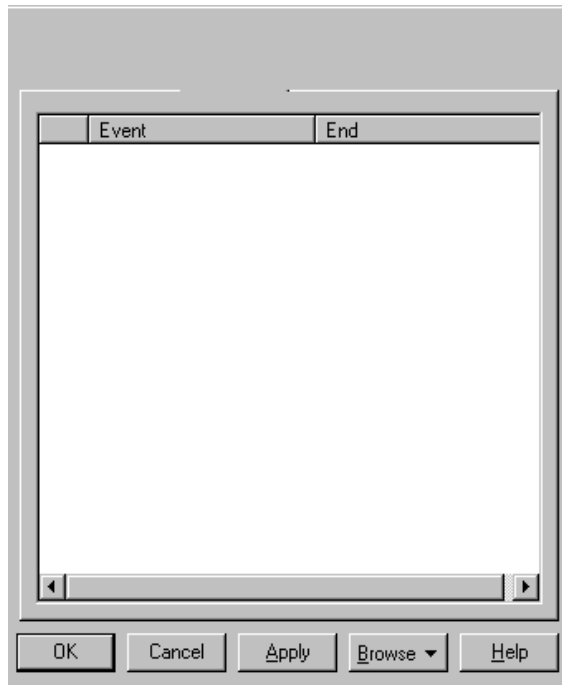
Action Expression

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 **Action Specification**, refer to the *Action Specification* on page 124.

State and Activity Specification—Transitions Tab

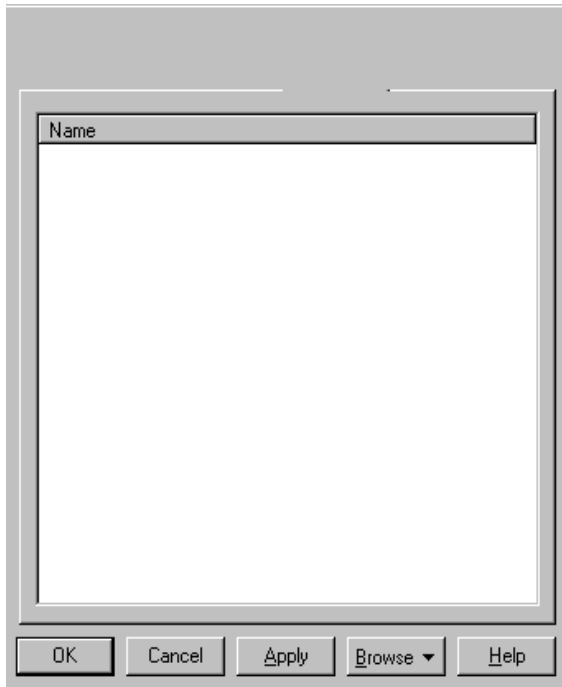
Figure 56 State and Activity Specification—Transitions Tab



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

State and Activity Specification—Swimlanes Tab

Figure 57 State and Activity Specification—Swimlanes Tab



Information about the swimlane name is displayed on this tab.

Action Specification

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

To define a new action on a state or activity:

- 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 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—Consist 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 **Detail** tab.

State Transition Specification

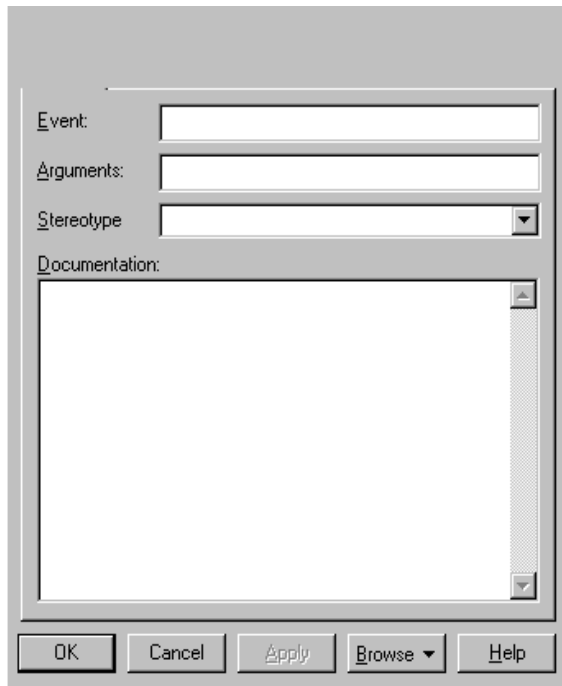
A **State Transition Specification** allows you to display and modify the properties and relationships of a transition on a statechart diagram or activity diagram. The **State Transition Specification** lists the events and actions that are comprised by the transition.

Specification Contents

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

State Transition Specification—General Tab

Figure 58 State Transition Specification—General Tab



Refer to the descriptions earlier in this chapter or in the *Introduction to Specifications* chapter for information on the specification elements.

Transition Specification—Detail Tab

Figure 59 State Transition Specification—Detail Tab

The screenshot shows a dialog box titled "State Transition Specification—Detail Tab". It contains several input fields and a section for substate transitions. At the bottom, there are buttons for "OK", "Cancel", "Apply", "Browse", and "Help".

Guard Condition:	<input type="text"/>
Action:	<input type="text"/>
Send event:	<input type="text"/>
Send arguments:	<input type="text"/>
Send target:	<input type="text"/>
Transition between substates	
From	<input type="text" value="Order Part"/>
To	<input type="text" value="Make Payment"/>

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 **Guard 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 to 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.

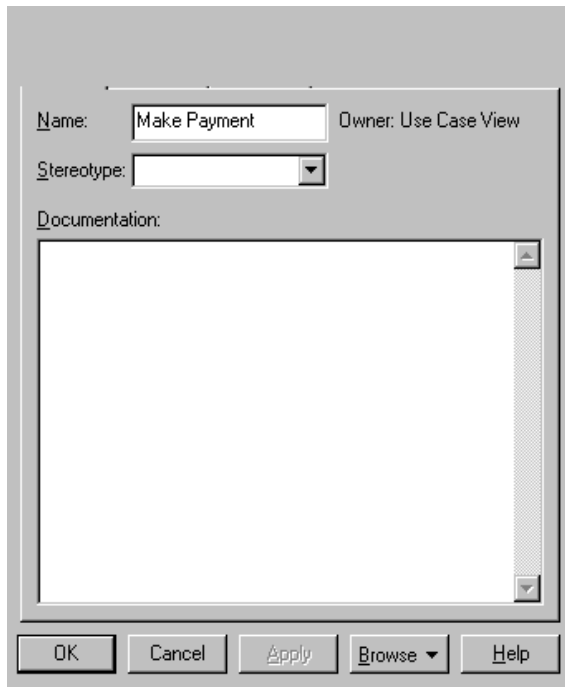
Decision Specification

A **Decision Specification** allows 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

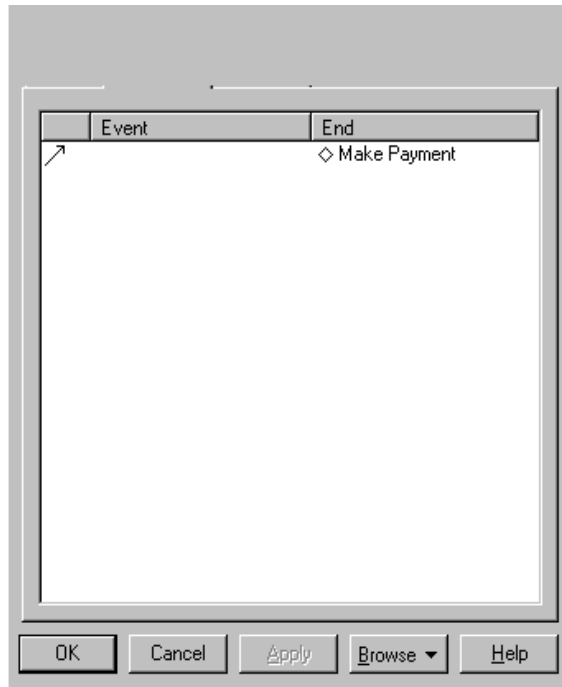
Figure 60 Decision Specification—General Tab



Refer to the descriptions earlier in this chapter or in the *Introduction to Specifications* chapter for information on the specification elements.

Decision Specification—Transitions Tab

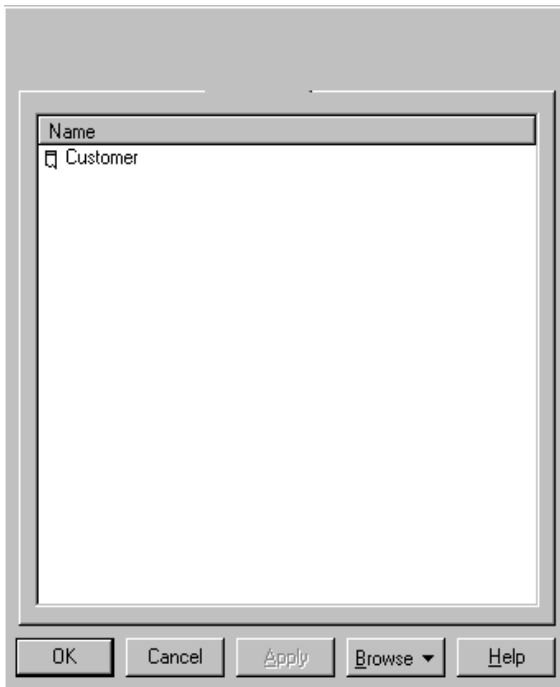
Figure 61 Decision Specification—Transitions Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements.

Decision Specification—Swimlanes Tab

Figure 62 Decision Specification—Swimlanes Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements.

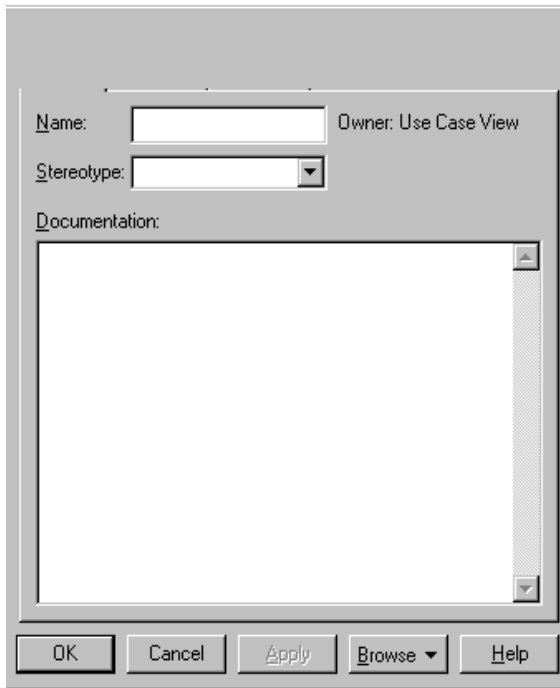
Synchronization Specification

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

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

Synchronization Specification—General Tab

Figure 63 Synchronization Specification—General Tab



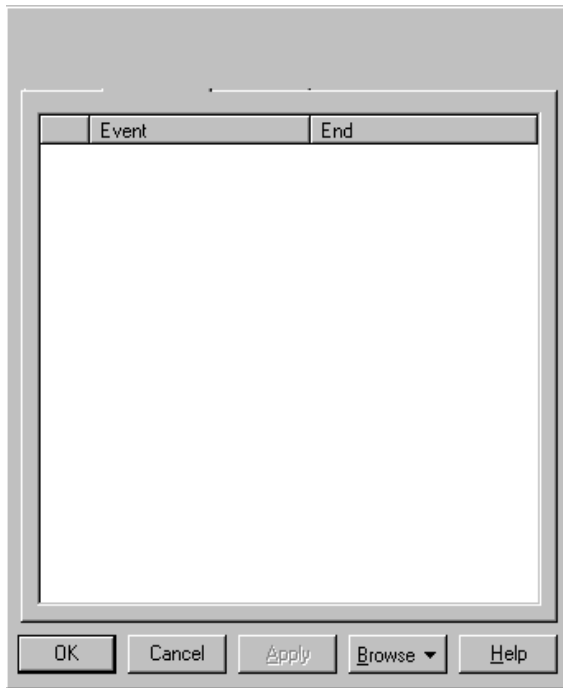
The image shows a dialog box titled "Synchronization Specification—General Tab". It contains the following elements:

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

Refer to the descriptions earlier in this chapter or in the *Introduction to Specifications* chapter for information on the specification elements.

Synchronization Specification—Transitions Tab

Figure 64 Synchronization Specification—Transitions Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements.

Object Specification (Activity Diagrams)

An **Object Specification** allows you to display and modify the properties of an activity diagram object. The object specifications that appear from objects on an activity diagram are slightly different from 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

Figure 65 Object Specification—General Tab

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

- Name:** A text input field.
- Class:** A dropdown menu with "(Unspecified)" selected.
- State:** A dropdown menu with "(Unspecified)" selected.
- 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
- Buttons:** OK, Cancel, Apply, Browse (with a dropdown arrow), and Help.

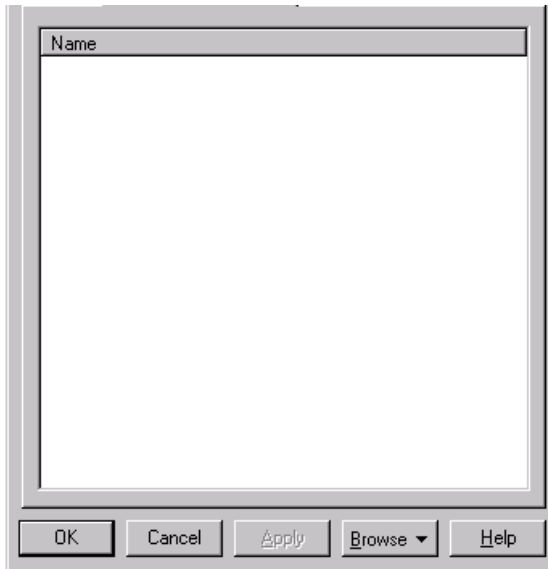
Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

State

The **State** drop-down list specifies and displays the object state.

Object Specification—Incoming Object Flows Tab

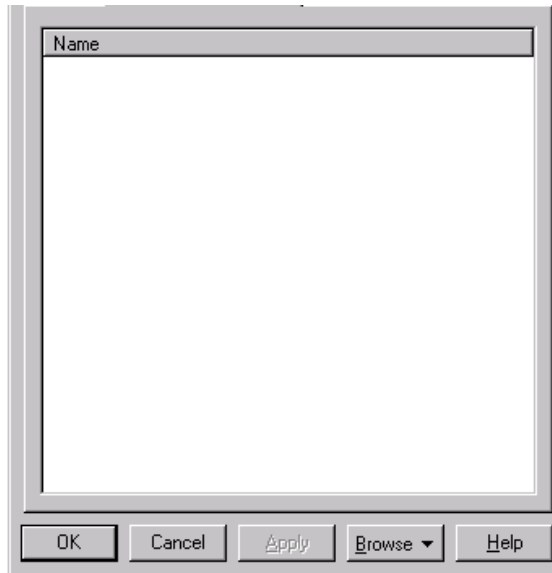
Figure 66 Object Specification—Incoming Object Flows Tab



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

Object Specification—Outgoing Object Flows Tab

Figure 67 Object Specification—Outgoing Object Flows Tab



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

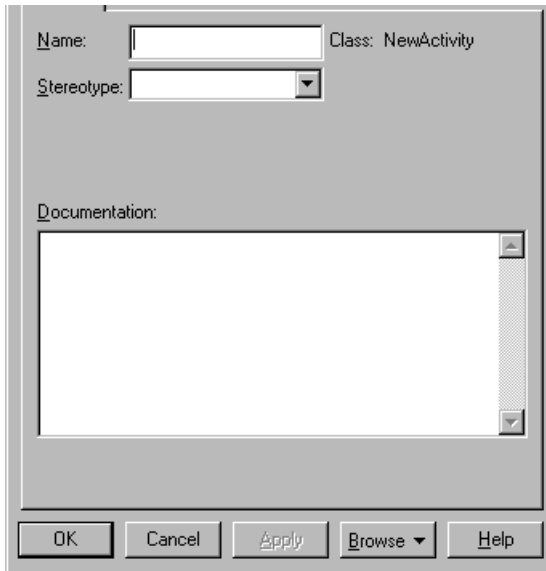
Object Flow Specification

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

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

Object Flow Specification—General Tab

Figure 68 Object Flow Specification—General Tab



The image shows a dialog box titled "Object Flow Specification—General Tab". It has a light gray background and a standard Windows-style border. At the top, there are two input fields: "Name:" followed by an empty text box, and "Class: NewActivity" to its right. Below these is a "Stereotype:" label followed by a dropdown menu showing a downward arrow. In the center, there is a "Documentation:" label above a large, empty text area with a vertical scrollbar on the right side. At the bottom, there is a row of five buttons: "OK", "Cancel", "Apply", "Browse" (with a small downward arrow), and "Help".

Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements.

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. The main difference between sequence and collaboration diagrams is that 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, and 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 show 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.

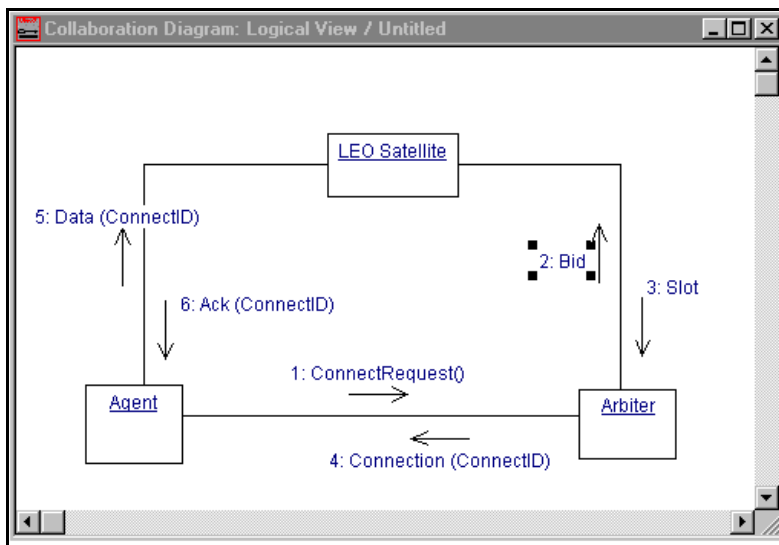
During analysis, collaboration diagrams can indicate the semantics of the primary and secondary interactions.

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 69 Collaboration Diagram Example



Sequence Diagrams

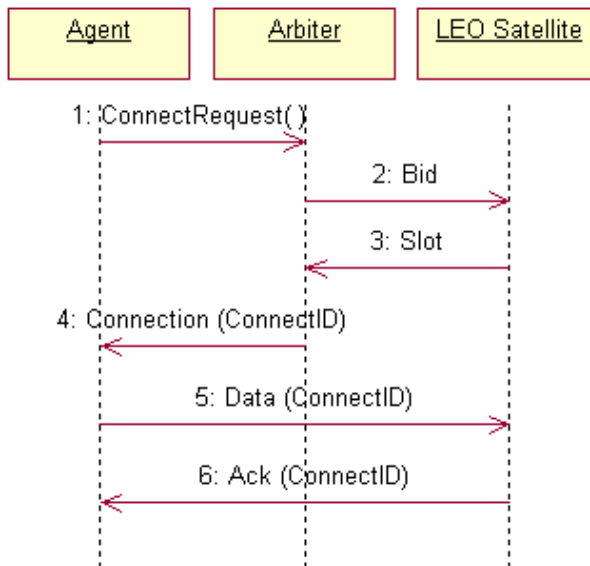
A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence—what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces. Sequence diagrams 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. Figure 70 shows a sequence diagram.

Figure 70 Sequence Diagram Example



Toolboxes

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

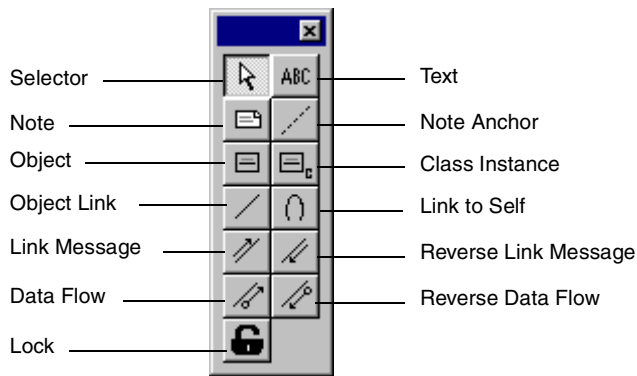
Collaboration Diagram Toolbox

The graphic below shows all the tools that can be placed on the collaboration diagram toolbox. Refer to *Customizing the Toolbox on page 11* 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 71 Collaboration Diagram Toolbox



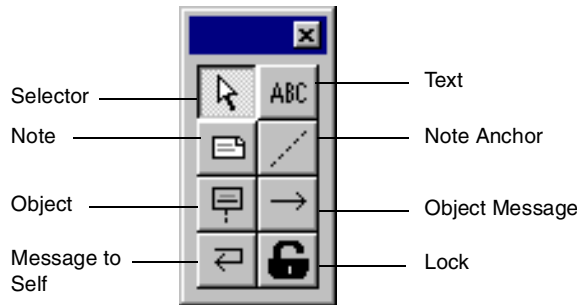
Sequence Diagram Toolbox

The graphic below shows all the tools that can be placed on the sequence diagram toolbox. Refer to *Customizing the Toolbox on page 11* 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 72 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 differ slightly, 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

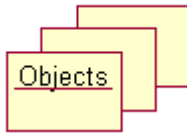
If you have multiple objects that are instances of the same class, you can modify the object icon by selecting the **Multiple Instances** check box in the **Object Specification**. When you select this check box, the icon is changed from one object to three staggered objects.

To create an icon representing multiple objects:

- 1 Create an object.
- 2 Double-click its icon to display its specification.
- 3 Select the **Multiple Instances** check box.
- 4 Click **OK**.

Rational Rose displays the **Multiple Object** icon.

Figure 73 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 a 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 up and to the right, places a message icon from client to supplier. The reverse message tool, bearing an arrow pointing down and to the left, 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 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 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 an icon representing the message to display the message specification. Then, modify the **Name** field and click **OK**.

Message Numbering

To enable or disable the display of message numbers click **Tools > Options**. Click 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 allows 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 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 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 its name.

You can also create multiple messages associated with the same message icon using the **Link Specification**. This method is described in *Link Specification* on page 154.

To change a message's assigned operation, display its specification by double-clicking its message icon. If the message icon represents multiple messages, double-click 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 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 (a 1, 2, 3 pattern) or hierarchical numbering (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 hierarchal numbering, the hierarchal 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 **Tools > Options**.
- 2 Click the **Diagram** tab.
- 3 Select the **Sequence Numbering** check box.
- 4 Select the **Hierarchical Messages** check box.

Scripts

Scripts are used to enhance messages on sequence diagrams; they are text fields that attach to messages.

To create and attach a script:

- 1 Click the message icon and drag it between two objects.
- 2 Create text by either:
 - Using the **ABC** icon.
 - Clicking **Tools > Create >Text**.
- 3 Select one or more labels. Press the CTRL or SHIFT key to enable multiple selections.
- 4 Select one message.
- 5 Click **Edit > Attach Script** to attach the script to the message.

To move a script:

- 1 Select the message and drag it.

The script moves next to the message.

- 2 Select only the script and drag it.

The script moves independently of the message.

To detach a script:

- 1 Select either a script or a message.

- 2 Click **Edit > Detach Script**.

To delete messages, scripts, or objects:

- Messages

Click the message, and then click **Edit > Delete** to delete all dependent messages and attached scripts.

- Scripts

Click the script, and then click **Edit > Delete** to delete the script with no effect on the messages.

- Objects

Click the object, and then click **Edit > Delete** to delete all messages and attached scripts.

To undo:

- Click **Edit > Undo** to reverse the latest change.

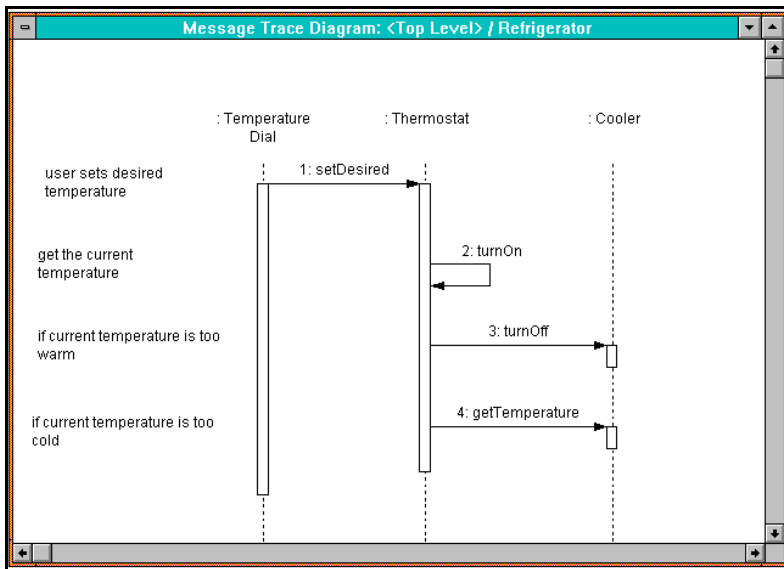
Focus of Control

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 an 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 an FOC vertically off the source FOC to make it detached and independent.

Figure 74 illustrates a sequence diagram with FOC notation and scripts.

Figure 74 Focus of Control Diagram Example



Displaying Focus of Control

To enable the Focus of Control notation on a sequence diagram:

- 1 Click **Tools > Options**.
- 2 Click the **Diagram** tab.
- 3 Select the **Focus of control** check box.

Coloring Focus of Control

To help distinguish a particular FOC from other items in a sequence diagram, you can fill an FOC with a color.

To color an FOC:

- 1 Select the message icon that enters the FOC you want to color.
- 2 Click **Format > Fill Color**.
- 3 Click 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 an FOC and all corresponding messages. If the FOC has an entry point message, you can move the message.

Otherwise, follow these 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 Click the source message and move it to the desired location on a sequence diagram.

The source FOC changes locations.

Nested Focus of Control

A nested Focus of Control is an 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 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 **Browse > Create Collaboration Diagram**. 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 **Browse > Create Sequence Diagram**. 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 (Interaction Diagrams)

An object specification allows you to display and modify the properties and relationships of an object in the current model.

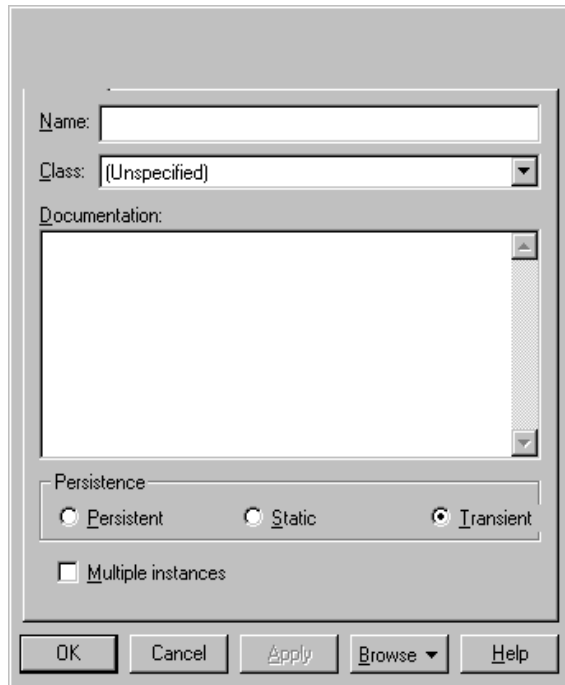
To display an **Object Specification**, double-click any icon representing an object, or click **Browse > Specifications**.

Specification Content

The **Object Specification** consists of the **General** tab.

Object Specification—General Tab

Figure 75 Object Specification—General Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for 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 an icon representing the object, and click **Show Persistence**.

Multiple Instances Check Box

Select 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.

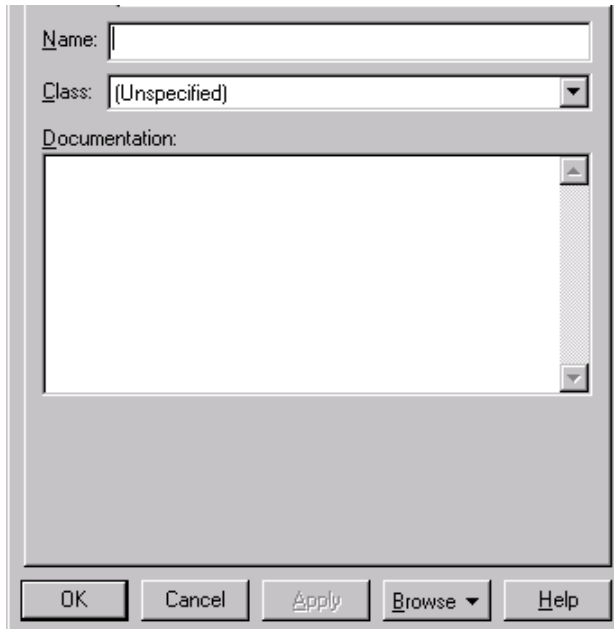
To display a **Class Instance Specification**, double-click any icon representing a class instance, or click **Browse > Specifications**.

Specification Content

The **Class Instance Specification** consists of the **General** tab.

Class Instance Specification—General Tab

Figure 76 Class Instance Specification—General Tab



The image shows a dialog box titled "Class Instance Specification—General Tab". It has a "Name:" text box at the top, which is currently empty. Below it is a "Class:" dropdown menu showing "(Unspecified)". Underneath is a "Documentation:" label followed by a large, empty text area with a vertical scrollbar on the right. At the bottom of the dialog box are five buttons: "OK", "Cancel", "Apply", "Browse" (with a small downward arrow), and "Help".

Refer to the descriptions earlier in this chapter or in the *Introduction to Specifications* chapter for 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 to the right of the **Class** field. A list box will display all the possible class selections, including **<New>**. Double-click **<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 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

Figure 77 Link Specification—General Tab

The screenshot shows a dialog box titled "Link Specification—General Tab". It contains the following elements:

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

Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

Assoc

The **Assoc** 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 and 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 and 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 the 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 a visibility type in the **Supplier Visibility** section.
- To set visibility for the client object, click 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, select or clear the **Shared** check box below the appropriate visibility section.

- If you select the **Shared** check box, the visibility adornment changes from a filled square to an open square of the corresponding type.
- If you clear the **Shared** check box, the visibility adornment changes to a filled square of the corresponding type.

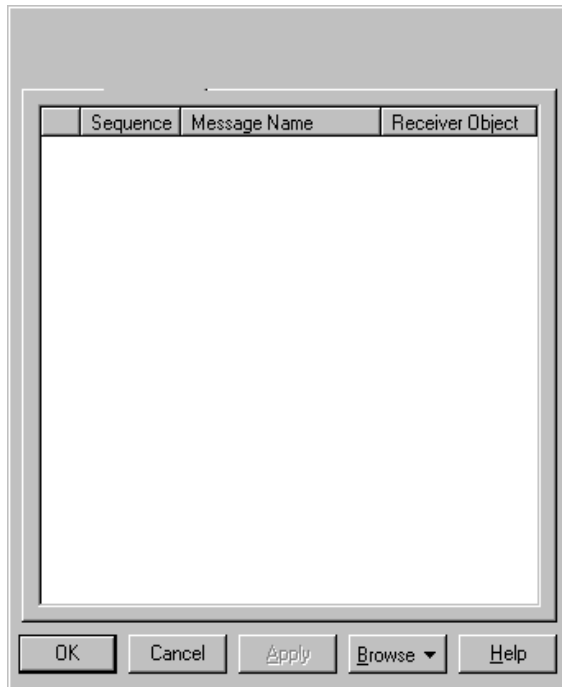
Role

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 78 Link Specification—Messages Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for 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 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 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.

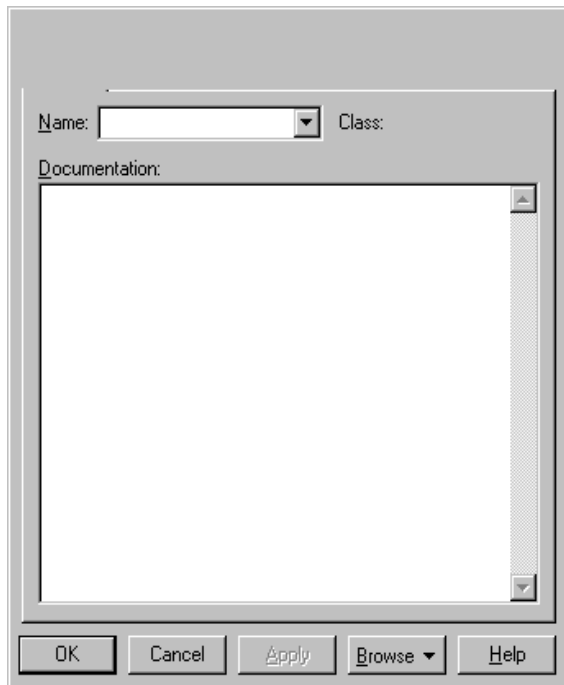
To display a **Message Specification**, double-click any icon representing a message, or click **Browse > Specifications**.

Specification Content

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

Message Specification—General Tab

Figure 79 Message Specification—General Tab



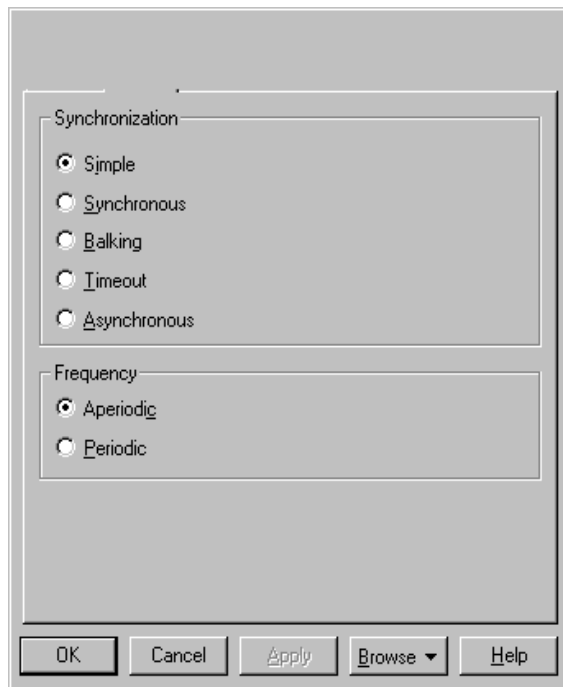
Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

Class

The **Class** field displays the name of the class to which the element belongs.

Message Specification—Detail Tab

Figure 80 Message Specification—Detail Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for 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.

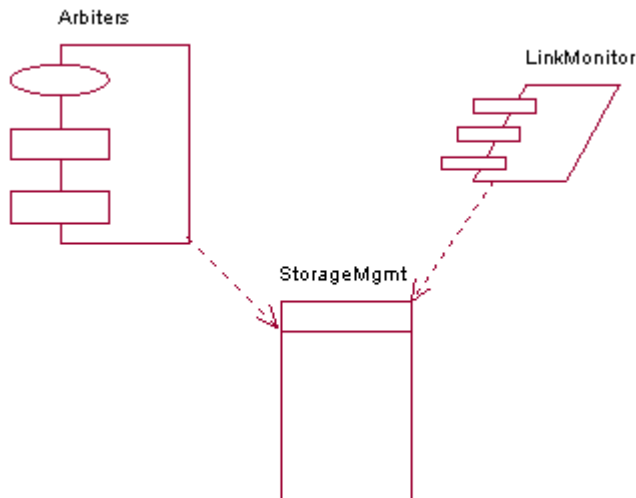
Table 15 Frequency Options

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 81 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 in which the diagram is contained.

Creating and Displaying a Component Diagram

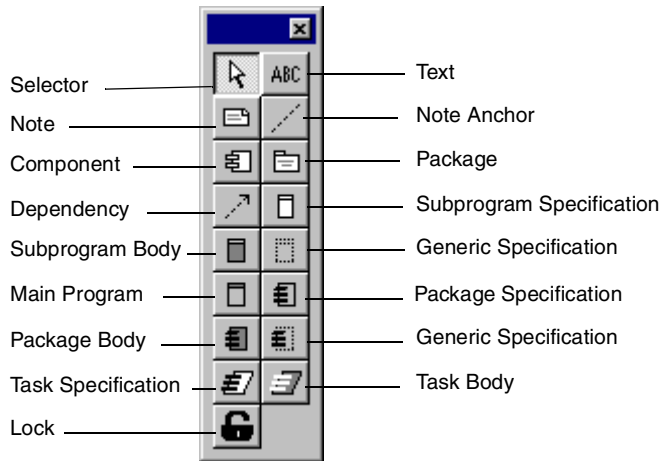
You can create or display the component diagram in one of three ways:

- Click **Browse > Component Diagram**.
- On the toolbar, click the component diagram icon.
- On the browser, double-click the component diagram icon.

Component Diagram Toolbox

The application window displays the following toolbox when the current window contains a component diagram and **View > As Unified** is selected.

Figure 82 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 Specification

A **Component Specification** displays and modifies the properties and relationships of each component in the current model. The same specification is used for all kinds of components.

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 any icon representing the component, or click **Browse > Specifications**.

Specification Content

The **Component Specification** consists of the following tabs: **General**, **Detail**, **Realizes**, and **Files**.

Component Specification—General Tab

Figure 83 Component Specification—General Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

Stereotype (Component)

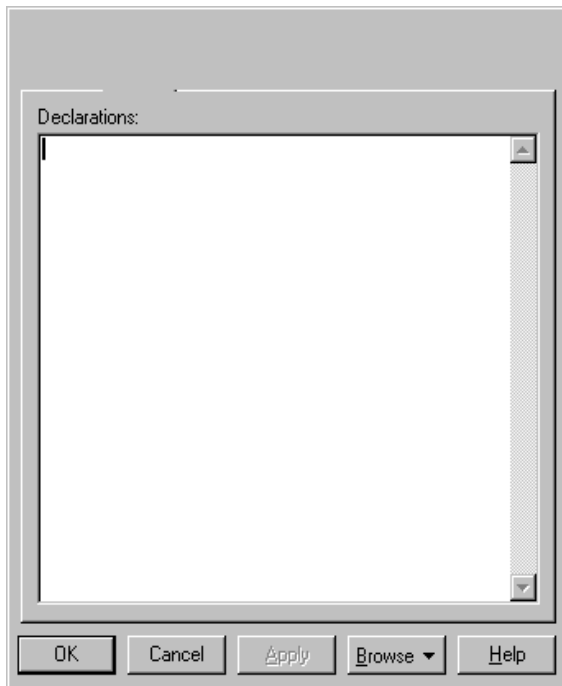
A component stereotype represents the subclassification of an element. The most common type of components are already predefined as stereotypes, including Main Program, Package Body, Package Specification, Subprogram Body, Subprogram Specification, Task Body, and Task Specification. You can also define 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 that allows you to specify how to handle those classes appears.

Component Specification—Detail Tab

Figure 84 Component Specification—Detail Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

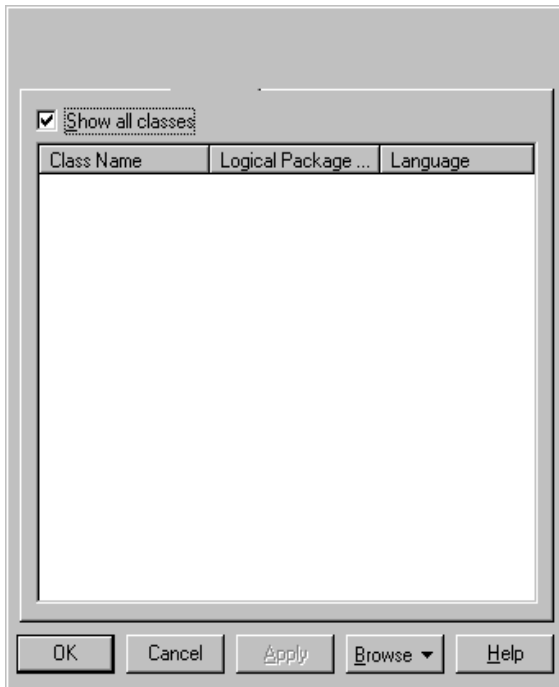
Declarations

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 85 Component Specification—Realizes Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for 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 check box is cleared, 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. 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.

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 in which you must specify how to handle those classes.

Component Specification—Files Tab

Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements.

Package Specification

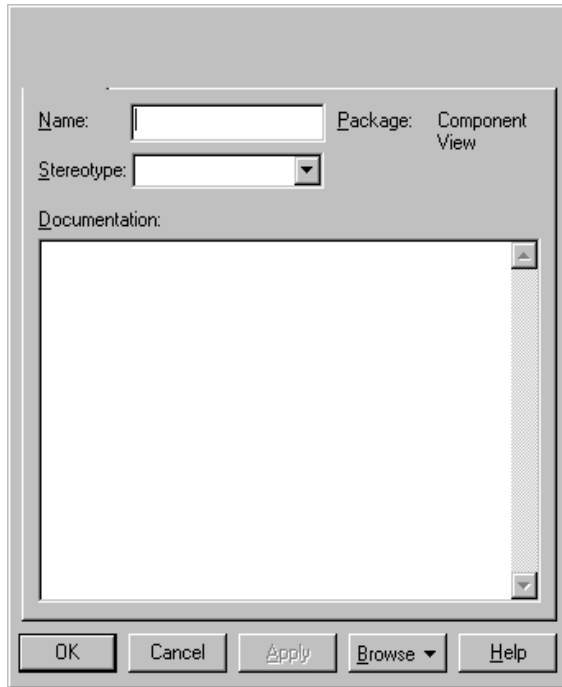
A **Package Specification** displays and modifies the properties and relationships of a package in the current model.

To display a **Package Specification**, double-click any icon representing the package, or click **Browse > Specifications**.

The **Package Specification** consists of the following tabs: **General**, **Detail**, **Realizes**, and **Files**.

Package Specification—General Tab

Figure 86 Package Specification—General Tab



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

- Name:** A text input field.
- Package:** A static text field containing the text "Component 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".

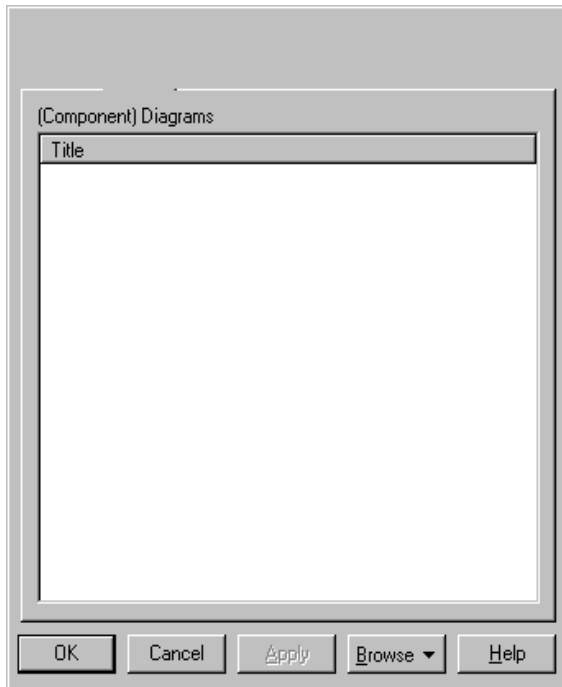
Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements not covered in the following section.

Package

The package the component belongs to is displayed in this static field.

Package Specification—Detail Tab

Figure 87 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 click **Browse > 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 its entry.

Package Specification—Realizes Tab

Refer to description earlier in this chapter for information on this tab.

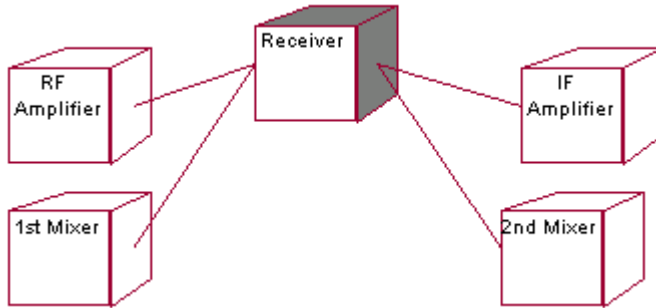
Package Specification—Files Tab

Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements.

Deployment Diagram Overview

A deployment diagram shows processors, devices, and connections. Each model contains a single deployment diagram that shows the connections between processors and devices, and the allocation of its processes to processors.

Figure 88 Deployment Diagram Example



Creating and Displaying a Deployment Diagram

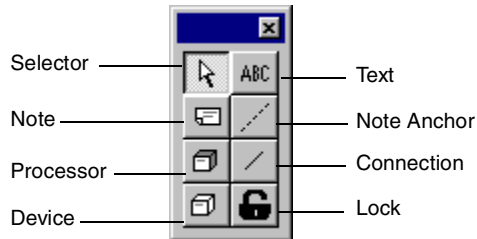
You can create or display the deployment diagram in one of three ways:

- Click **Browse > Deployment Diagram**.
- On the toolbar, click the deployment diagram icon.
- In the browser, double-click 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 89 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.

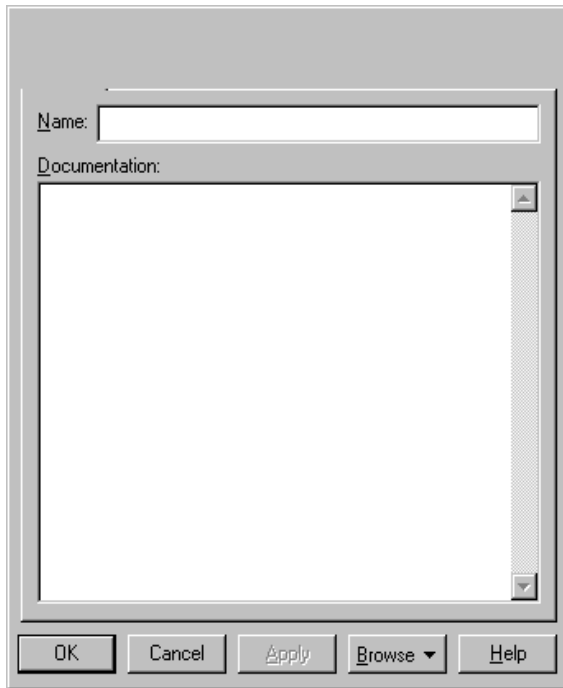
To display a **Processor Specification**, double-click any icon representing a processor, or click **Browse > Specifications**.

Specification Content

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

Processor Specification—General Tab

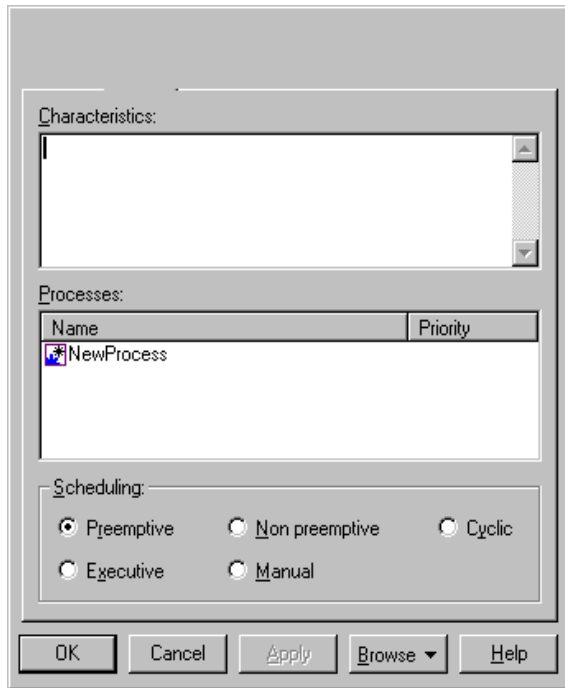
Figure 90 Processor Specification—General Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements.

Processor Specification—Detail Tab

Figure 91 Processor Specification—Detail Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for 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 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, right-click in the processes area and click **Insert** from the shortcut menu. A new process 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 (default)	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.
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 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.

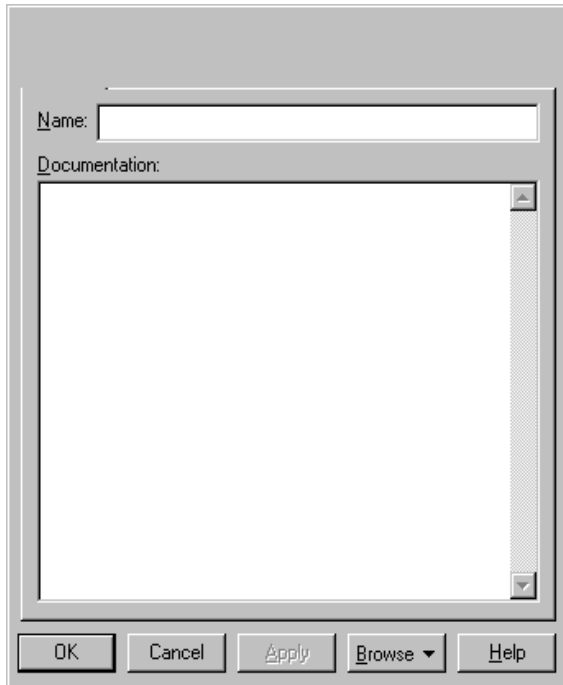
To display a **Device Specification**, double-click any icon representing a device, or click **Browse > Specifications**.

Specification Content

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

Device Specification—General Tab

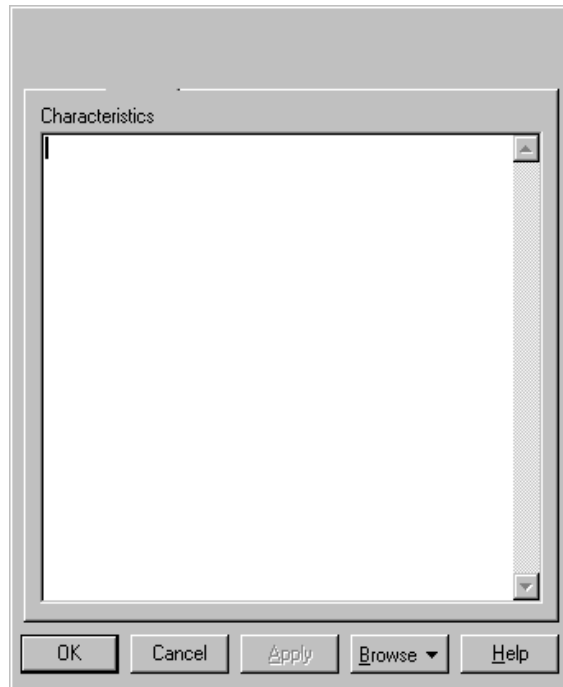
Figure 92 Device Specification—General Tab



Refer to the descriptions in the *Introduction to Specifications* chapter for information on the specification elements.

Device Specification—Detail Tab

Figure 93 Device Specification—Detail Tab



Refer to the descriptions earlier in this chapter or in the *Introduction to Specifications* chapter for information on the specification elements.

Connection Specification

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 RS-232 cable. It can also represent an indirect coupling.

To display a **Connection Specification**, double-click any icon representing a connection, or click **Browse > Specifications**.

The **Connection Specification** consists of two tabs, which contain the same elements as the **Device Specification**. Refer to *Device Specification* on page 173.

Process Specification

Processes are threads of control that execute on a processor. One process specification documents one thread of control.

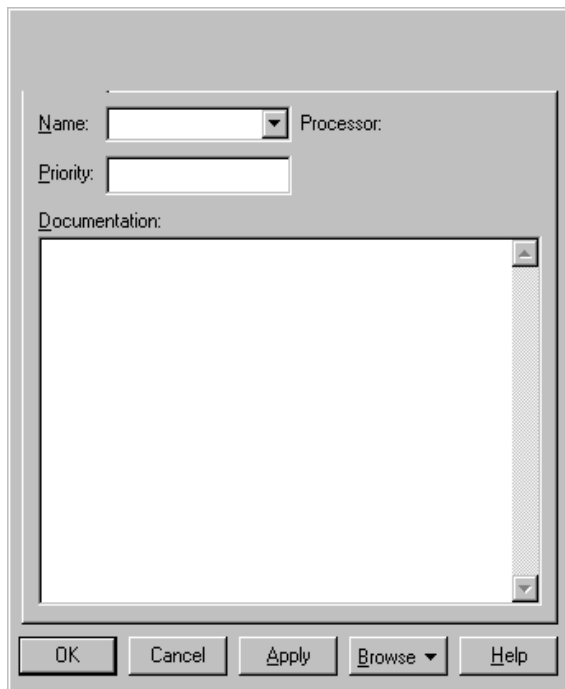
You access the **Process Specification** through the **Processes** field of a **Processor Specification**. None of the information contained in the **Process Specification** is displayed in a diagram; thus, process properties can only be viewed and modified through a **Process Specification**.

Specification Content

The **Process Specification** consists of the **General** tab.

Process Specification—General Tab

Figure 94 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".

Refer to the descriptions in the *Introduction to Specifications* chapter for 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.

Overview

A stereotype is a modeling element subclassification that carries a specific meaning. Stereotypes can be applied to:

- Activities
- Association Relationships
- Attributes
- Classes
- Components
- Connections
- Dependency Relationships
- Devices
- Generalization Relationships
- Objects (on activity diagrams)
- Operations
- Packages
- Processors
- States
- Use Cases

A stereotype can be depicted by either a name or 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 greater detail.

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

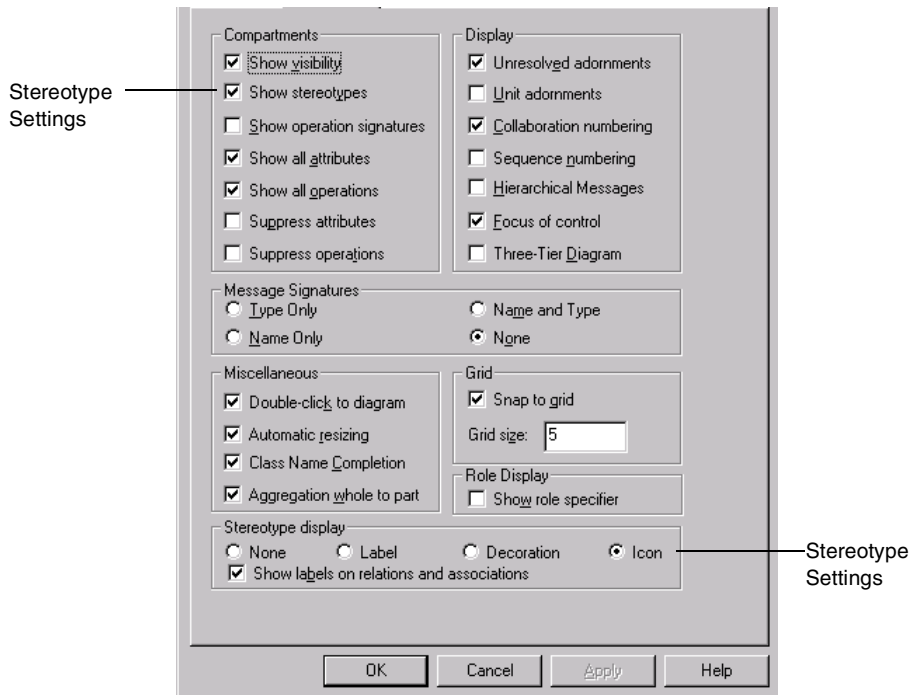
For more information on these icons, refer to the online Help.

Viewing Stereotypes

You can control how stereotypes are displayed. Click **Tools > Options** to display the **Options** dialog box. The settings are found on the **Diagram** tab.

Diagram Tab

Figure 95 Options Dialog Box—Diagram Tab



The following selections are applied to new model elements that are added to the diagrams. To make changes to existing model elements, use the shortcut menu.

Compartments—Show Stereotypes

This check box 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 on model elements (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 guillemets (<< >>).

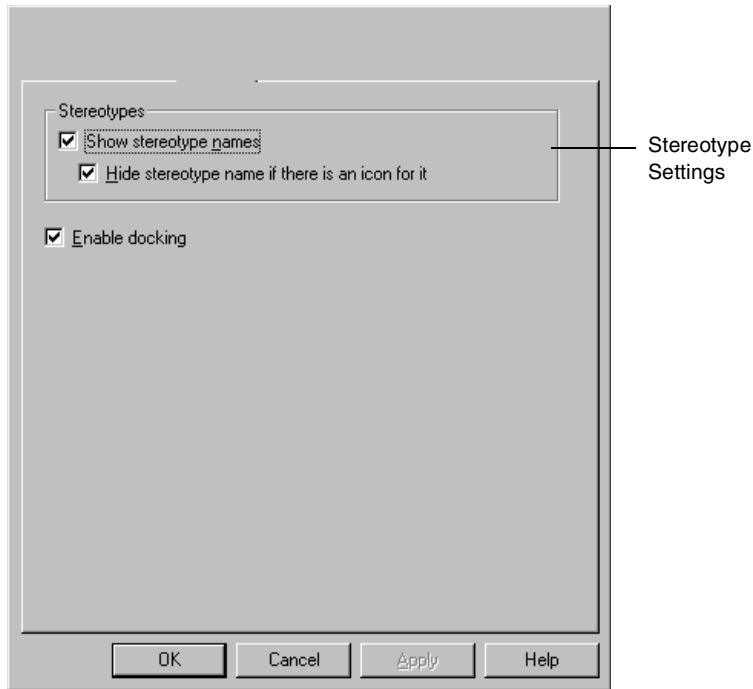
Decoration displays a graphic marker such as highlighting an icon or tool. Selecting **None** means the name is not displayed, while selecting **Icon** means the icon created for that stereotype is displayed.

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 names of stereotypes appear inside guillemets (<< >>).

Browser Tab

Figure 96 Options Dialog Box—Browser Tab



The changes made on the **Browser** tab are reflected in the browser.

Show stereotype names

This check box 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 check box to display stereotype icons, but not stereotype names, of model elements in the browser. This check box 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, refer to *Creating a New Stereotype for All Rose Models on page 184*.

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 *Creating a New Stereotype for the Current Model on page 183*.
- 4 Run the Windows Registry Editor (regedit.exe) by clicking **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 > 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. Refer to *Creating Stereotype Icons* on page 185.

- 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 an ampersand (&) 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 in which 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 in which 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 in which 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 assign values other than defaults.
- 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, refer to *Adding Stereotypes to the Diagram Toolbox* on page 187.

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, refer to *Viewing Stereotypes* on page 180.

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. Diagram toolbox and list view icons must be created in bitmap (.bmp) format.

Note: If you create an icon (for example, a diagram icon), 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.
- 2 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** check box is selected 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. 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 an icon (for example, a list view icon), 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** list in the **Save as** dialog box.

Note: Some of your icon colors may not match 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, refer to *Creating Stereotypes* on page 183.
- 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**.

The **Customize Toolbars** dialog box is displayed. The leftmost column provides the list of available icons.

- 4 Select the icon you want to appear on the diagram toolbox and click **Add**.

Subsystem Stereotype Package

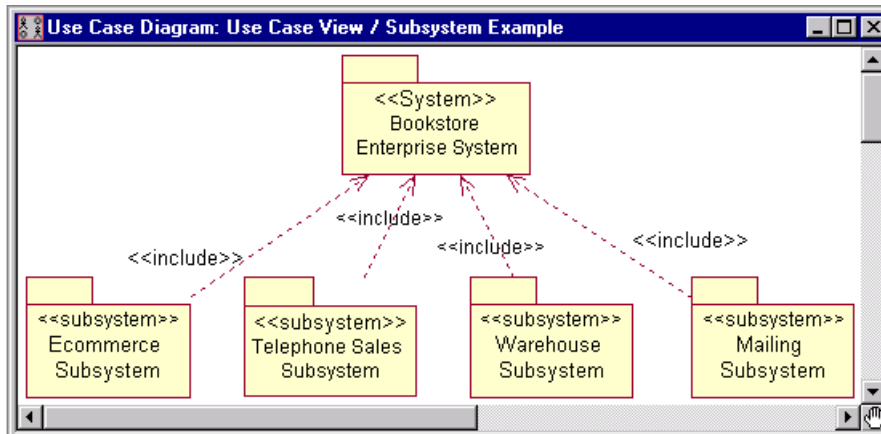
Although closely related to a system, a subsystem is a group of model elements that have specific behavior and objectives. A subsystem is a stereotyped package and is represented by the package icon with the subsystem stereotype.

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 97 Subsystem Stereotype Sample



The four subsystems in the subsystem sample together make up all the functionality of the *Bookstore Enterprise System*. Note that through 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.

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 (refer to *Activating the Framework Wizard Add-In* on page 189).

Activating the Framework Wizard Add-In

In order to create models from frameworks and add new frameworks, the Framework Wizard Add-In 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.

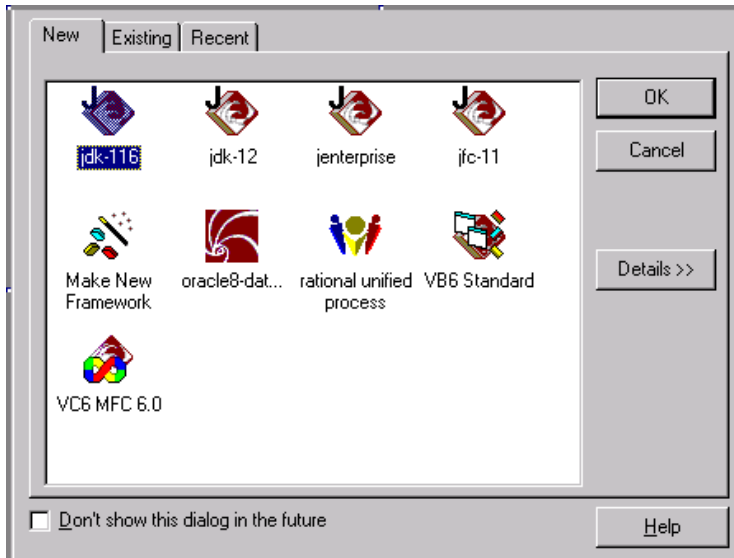
Creating a New Model from a Framework

To create a new model from a framework

- 1 Click **File > New**.

The **Create New Model** dialog box opens.

Figure 98 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**.

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 the package in a diagram, or click **File > Units > 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 (refer to *Activating the Framework Wizard Add-In* on page 189).

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 is 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 that appears in the **Create New Model** dialog box when the user clicks **Details**. If there is no .rtf file, default description text appears.
- *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

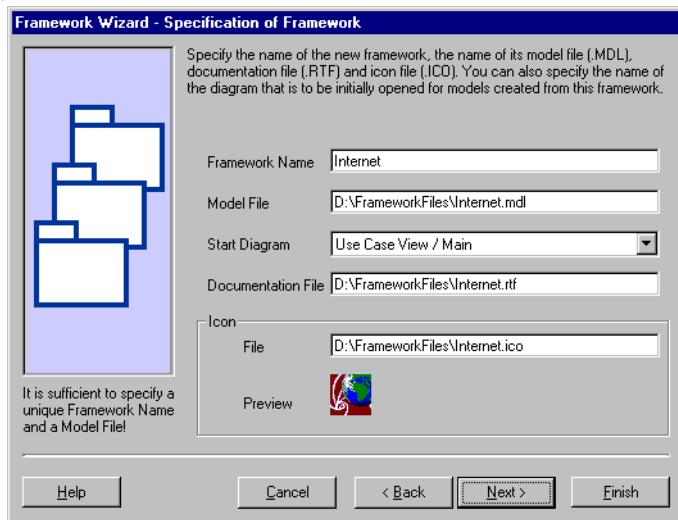
To create a new framework:

- 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.
- 2 Write a description of the framework in any word processor and save the document in RTF (Rich Text Format) format in any folder.
- 3 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).
- 4 Click **File > New**.

The **Create New Model** dialog box opens.

- 5 Open the “Make New Framework” framework, which starts the Framework Wizard. (If the welcome page appears, click **Next**.)

Figure 99 Framework Wizard Specification Page

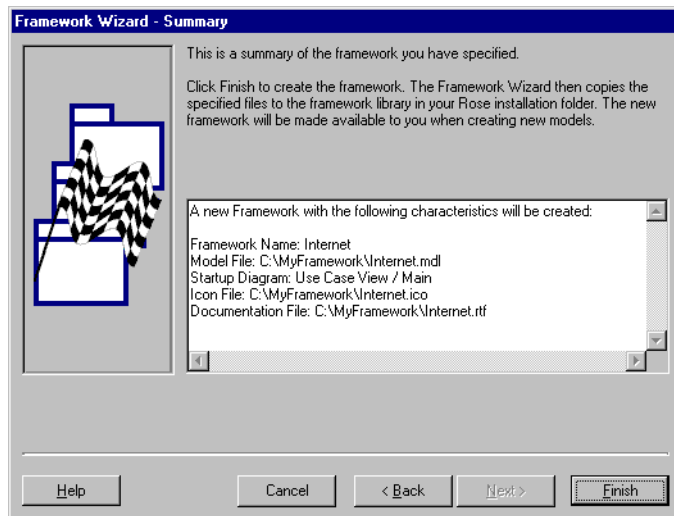


- 6 In the **Framework Name** field, specify the name of the new framework. The name must be unique among the existing frameworks, and it can only contain characters that are allowed in folder names.
- 7 In the **Model File** field, specify the name and location of the file that constitutes the framework model. (To browse to the file, click in the field. Then, click the displayed button.)

- 8 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.
- 9 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.)
- 10 Click **Next**.

A summary of the new framework appears.

Figure 100 Framework Wizard Summary Page



- 11 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 from the \Framework\Frameworks folder.

The Type Library Importer allows you to import a type library of a COM component into the model by dragging the COM component from Windows Explorer and dropping it in Rational Rose. You can also click **Tools > COM > Import Type Library**.

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, refer to the *Customizing the Type Library Importer* on page 208.

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, and so on 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, and so on. For 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 icon indicates whether the file will be imported or attached to a model element:

- 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.

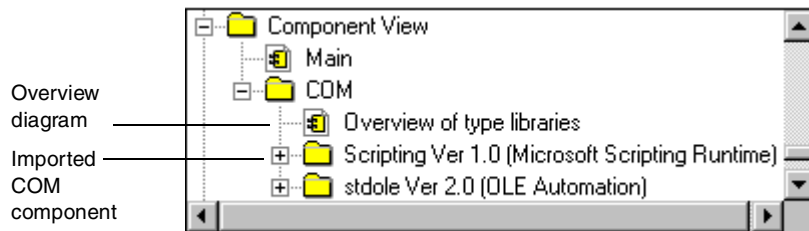
How Is a Type Library Presented?

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 101, in the component view for an imported type library.

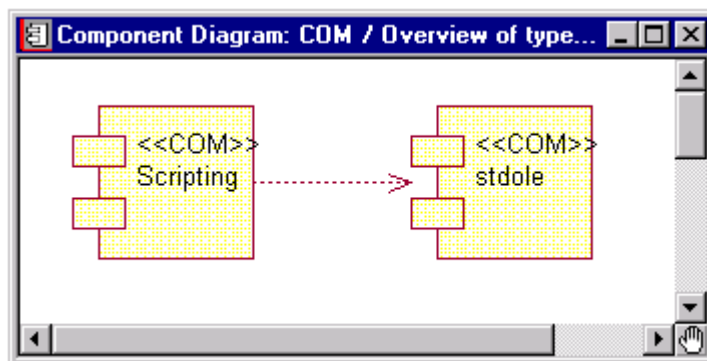
Figure 101 Component View of the Microsoft Scripting Runtime Type Library



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 102 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.

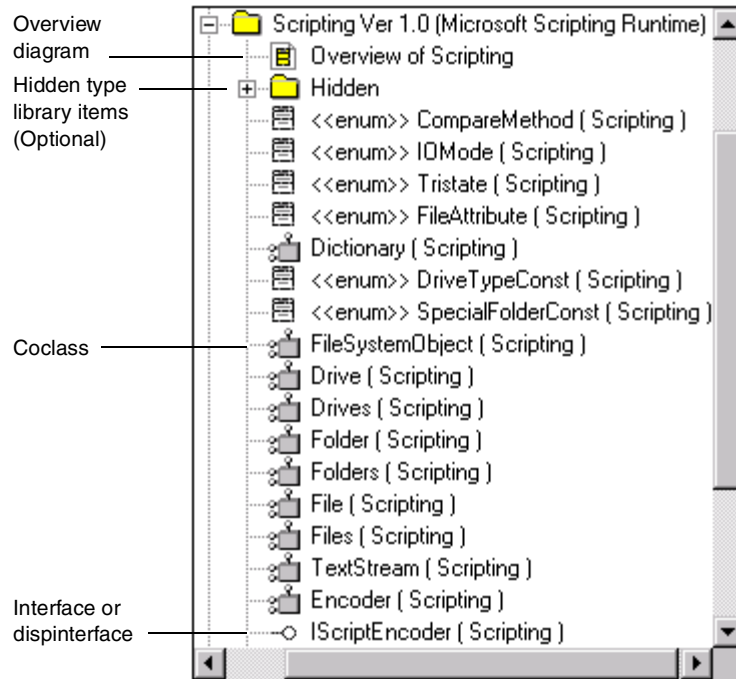
Figure 102 Component Overview Diagram for a Model



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 103.

Figure 103 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, and so on.

Each item in the type library is represented by a class, such as the coclass `FileSystemObject` in Figure 103. 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 (refer to *COM Stereotypes* on page 200). In Figure 103, you can see that coclasses have their own icon in the browser:

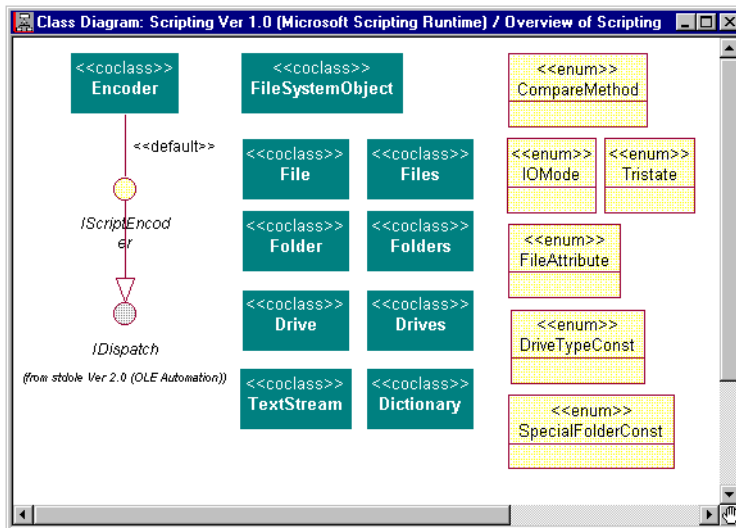
The kind model property on an interface class indicates whether the class corresponds to an interface or a dispinterface in COM.

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; refer to *Adding Class Members to a Quick Import Type Library* on page 207.

The Type Library Overview Diagram

An overview diagram is created in the logical view, which shows the contents of the imported type library. Figure 104 shows the overview diagram for the Microsoft Scripting Runtime type library. As you can see, coclasses, such as `Encoder` in Figure 104, are shaded (green) in type library overview diagrams.

Figure 104 Overview Diagram of the Microsoft Scripting Runtime Type Library



Hidden Items

If the **Show hidden items** check box 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 103 on page 198 and Figure 104 on page 199 was imported with the **Show hidden items** check box cleared.

For more information, refer to *Hiding Type Library Items* on page 205.

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 102 on page 197 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 104 on page 199, 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	Description
Components	
COM	The Type Library Importer assigns the stereotype and language “COM” to the component representing an imported type library.
Classes	
coclass	Represents a coclass in a type library.
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.

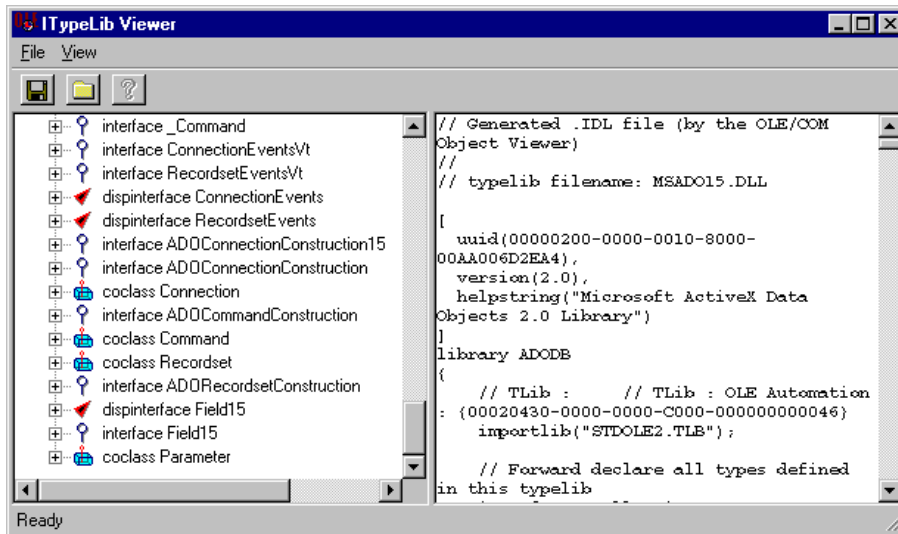
Table 17 COM Stereotypes (continued)

Stereotype	Description
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	
none	Represents any realize relationship between type library items.
source	Indicates that the realized interface contains the coclass' event procedures.
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.

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.

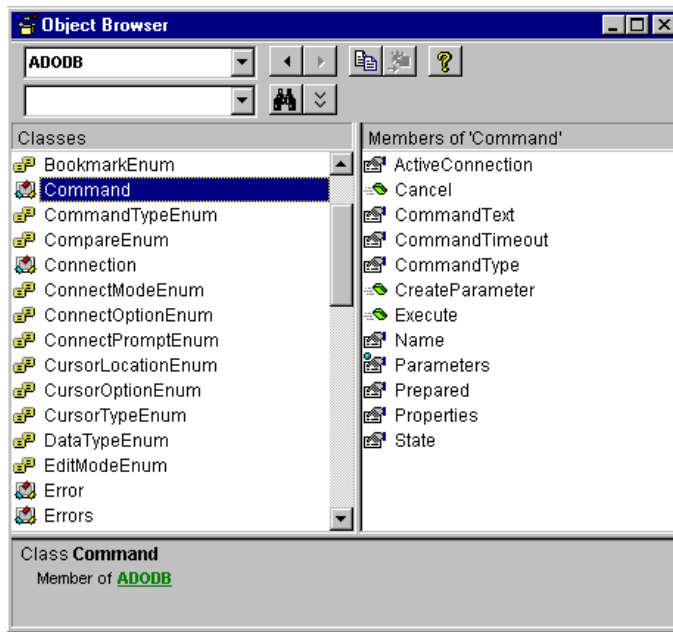
Figure 105 OLE Viewer in Visual Studio



Type Library in the Object Browser in Visual Basic

Figure 106 on page 203 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 106 Object Browser in Visual Basic



Since 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** check box, 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, refer to *Hiding Type Library Items* on page 205.

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, click **Tools > COM > Properties** to display the **COM Properties** dialog box. In it, you can control how to import type libraries. For more information, refer to *Customizing the Type Library Importer* on page 208.

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. Refer to *Hiding Type Library Items* on page 205.
- 2 Drag the file—DLL, EXE, OCX, OLB, or TLB—from 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; this 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, refer to *Adding Class Members to a Quick Import Type Library* on page 207.

The following results occur:

- 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 the component that represents the imported type library and click **Upgrade to Latest Version**.

Hiding Type Library Items

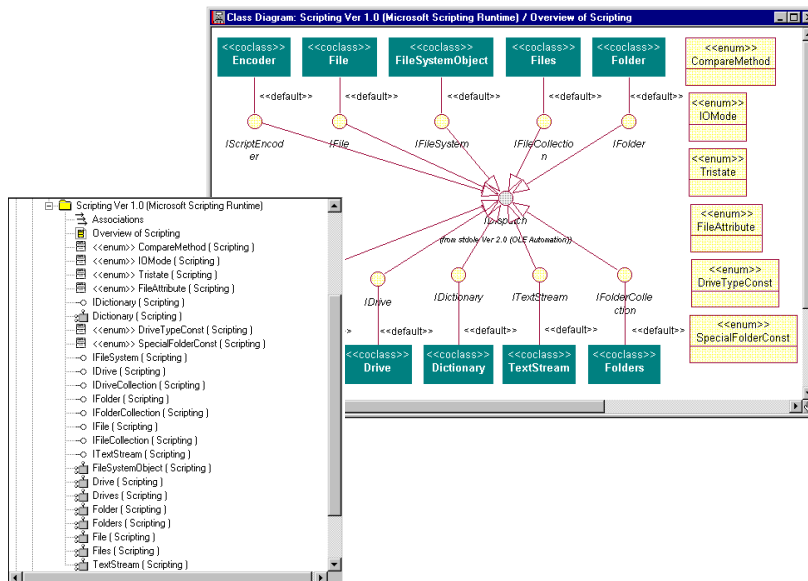
When importing a type library, the created type library in the model is represented differently depending on whether the **Show hidden items** check box is selected in the **COM Properties** dialog box. Click **Tools > COM > Properties** to display the **COM Properties** dialog box.

Show Hidden Items Selected

If the **Show hidden items** check box is selected when importing a type library, all type library items, including coclasses, dispinterfaces, and interfaces, 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 107 shows how the Microsoft Scripting Runtime component, `scrrun.dll`, is presented when imported with the **Show hidden items** check box selected. This view is recommended when developing clients in languages other than Visual Basic.

Figure 107 Type Library with Show Hidden Items Option Selected

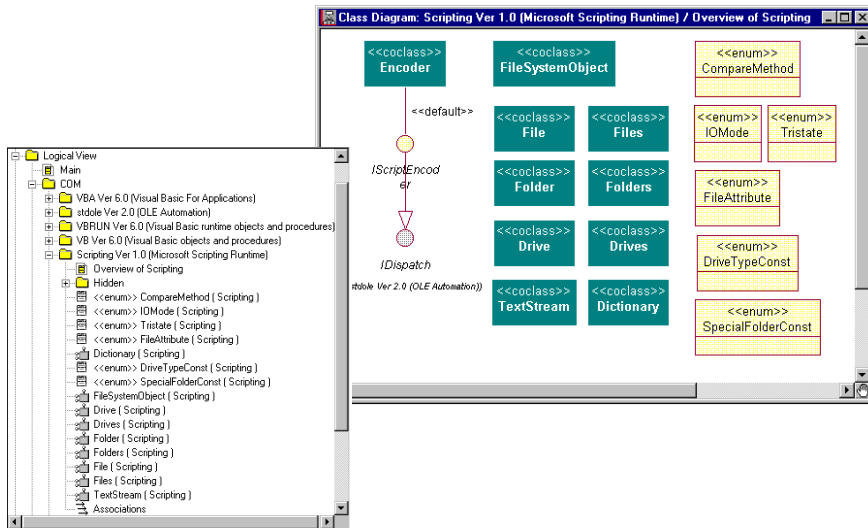


Show Hidden Items Cleared

If the **Show hidden items** check box 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 108 shows how the Microsoft Scripting Runtime component, `scrrun.dll`, is presented when imported with the **Show hidden items** check box is cleared.

Figure 108 Type Library with Show Hidden Items Option Cleared



In this view, only the type information that is relevant for Visual Basic clients is shown on the type libraries overview diagram, and all hidden type library items are inserted into a separate package called `Hidden`. 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 106 on page 203).

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 depends on the programming language. For more 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:
 - an interface to import its class members into the model.
 - a coclass to import the class members of its interfaces into the model.
 - a 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.

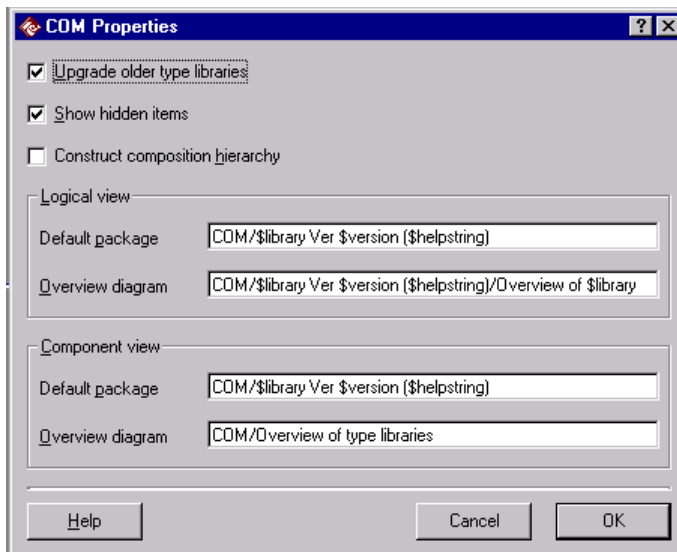
Customizing the Type Library Importer

In the **COM Properties** dialog box, you can control 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, 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 109 COM Properties Dialog Box



Note: Changing the settings in the **COM Properties** dialog box affects only type libraries that are imported after the settings are changed.

To replace existing type libraries when importing new versions:

Select the **Upgrade older type libraries** check box 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 check box 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”:

Refer to *Hiding Type Library Items* on page 205.

To show the composition hierarchy for imported type libraries:

Select the **Construct composition hierarchy** check box. 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 in which 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 in which 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`.

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 file format 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 Version

Rose Version	Petal File Version	Rose Format
Rose 3.0	Petal 37	3.0 Model
Rose 4.0	Petal 40	4.0 Model

Table 18 Rational Rose Petal File Version (continued)

Rose Version	Petal File Version	Rose Format
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

Index

Symbols

(-) Minus Sign 19

(+) Plus Sign 19

A

Abstract 62, 102

Action Expression 123

Action Specification 124

Actions Tab 122

Activities 115

Activity Actions 125

Activity Diagram

 Creating 112

 Overview 111

 Using 111

 Workflows 111

Activity Specification 121

 Actions Tab 122

 General Tab 121

 Swimlanes Tab 124

 Transitions Tab 123

Actor Specification 105

Actors 95

Add-In

 Installing 4

 Manager 4

Adding

 Classes 55

 Stereotypes to the Toolbox 187

Adorning Diagrams 40

Aggregate 87

Application Window 6

Apply Button 51

Argument 76

Assigning a Component to a Package 162

Association 97, 155

Association Specification 83

 Detail Tab 84

 General Tab 83

 Role A and B Detail Tab 87

 Role A and B General Tab 86

Attributes Tab 65

B

Bending Correlations 37

Browse Button 51

Browse Class Diagram Button 8

Browse Component Diagram Button 9

Browse Deployment Diagram Button 8

Browse Interaction Diagram Button 9

Browse Parent Button 9

Browse Previous Diagram Button 9

Browse State Machine Diagram Button 9

Browse Use-Case Diagram Button 9

Browser

 Collapsing a Tree 19

 Creating Icons for 186

 Description 11, 17

 Displaying 18

 Docking 18

 Drag-and-Drop 21

 Expanding a Tree 19

 Hiding 18

 Illustration 6, 17

 Naming an Element 20

 Navigating 18

 Positioning 18

 Viewing 17

Browser to Browser Capabilities 22

Browser to Diagram Capabilities 23

Browser to Specification Capabilities 24

Buttons

 on Specifications 51

 on the Toolbar 7

C

- Cancel Button 51
- Cardinality 60
- Changing Model Elements 34
- Changing the State of an Object 118
- Characteristics 172
- Class 72, 151, 153, 159
- Class Attribute Specification 71
 - Detail Tab 73
 - General Tab 72
- Class Diagram
 - Creating 55
 - Displaying 55
 - Overview 53
 - Re-assign a Class 55
 - Toolbox 54
- Class Instance Specification 152
- Class Specification 56
 - Attributes Tab 65
 - Component Tab 68
 - Detail Tab 59
 - Files Tab 71
 - General Tab 56
 - Nested Tab 69
 - Operations Tab 63
 - Relations Tab 67
- Classes 166
- Client Visibility 155
- Close Button 7
- Coclass 198
- Collaboration Diagram
 - Creating 137, 150
 - Overview 138
 - Toolbox 140
- Collapsing a Browser Tree 19
- COM
 - Components 196
 - Properties Dialog Box 208
 - Stereotypes 200
- Component Diagram 168
 - Creating 161
 - Displaying 161
 - Overview 161
 - Toolbox 162
- Component Name 68
- Component Specification 163
 - Detail Tab 164
 - Files Tab 166
 - General Tab 163
 - Realizes Tab 165
- Component Tab 68
- Concurrency 62, 77
- Connection Specification 175
- Connections 169
- Constraints 85
- Containment 73, 88
- Context Sensitive Help Button 8
- Control-Menu Box 6
- Copy Button 8
- Copying Icons 35
- Correlation
 - Bending 37
 - Creating 37
 - Description 37
 - Naming 38
 - Reconnecting 38
- Creating
 - Activity Diagrams 112
 - Class Diagrams 55
 - Collaboration Diagrams 137, 150
 - Component Diagram 161
 - Correlations 37
 - Diagrams 28
 - Icons 186
 - List View Icons 186
 - Model Elements 20, 30
 - New Stereotype Configuration File 183
 - Overloaded Elements 31
 - Sequence Diagrams 137, 150
 - Statechart Diagrams 109
 - Stereotypes 183
 - Toolbox Icons 186
- Customizing the Toolbox 11
- Cut Button 8
- Cutting Icons 35

D

- Decision Specification 128
 - General Tab 128
 - Swimlanes Tab 130
 - Transitions Tab 129
- Decisions 119
- Declarations 165
- Deep Delete 36
- Default 82
- Default Font Parameters 40
- Deleting
 - Deep vs. Shallow 36
 - Diagrams 29
 - Messages 147
 - Model Elements 36
 - Objects 147
 - Scripts 147
- Dependency 98
- Dependency Specification 91
- Deployment Diagram
 - Creating 169
 - Displaying 169
 - Overview 169
 - Toolbox 170
- Derived 74, 85
- Deselecting Icons 33
- Detail Tab
 - About 49
 - Association Specification 84
 - Class Attribute Specification 73
 - Class Specification 59
 - Component Specification 164
 - Device Specification 175
 - Has Specification 93
 - Logical Package Specification 93
 - Message Specification 159
 - Operation Specification 76
 - Package Specification 168
 - Processor Specification 172
 - State Transition Specification 127
- Device Specification 173
 - Detail Tab 175
 - General Tab 174
- Devices 169
- Diagram Icon, see Icons
- Diagram List 103
- Diagram Tab 102
- Diagram Toolbox, see Toolbox
- Diagram Window
 - Cascading 27
 - Description 12, 25
 - Illustration 6
 - Tiling 27
 - Viewing 26
- Diagrams
 - Adorning 40
 - Creating 28
 - Deleting 29
 - Displaying 29
 - Laying Out 38
 - Linking 28
 - Placing Text 40
 - Printing 14
 - Renaming 29
 - Saving 15
 - Type Library 199
 - Viewing 26
- Dialog Box, see Specification
- Dispinterface 198
- Displaying
 - Browser 18
 - Diagrams 29
 - Focus of Control 148
 - Multiple Diagrams 27
 - Specifications 45
 - Stereotype Names 182
- Docking 12, 18
- Documentation 48
- Documentation Window
 - Description 11
 - Illustration 6
- Drag-and-Drop 20, 21

E

- Editing Model Elements 20, 33
- Editing Specifications 46
- Element 84

Elements, see Model Elements
End State 118
Exceptions 77
Expanding a Browser Tree 19
Export Control 58
Extend Stereotype 98
Extending Rational Rose 4

F

F5 Key 149
Features 3
Files 15, 16, 211
Files Tab
 About 49
 Class Specification 71
 Component Specification 166
 Operation Specification 80
 Package Specification 168
Filtering 15, 55
Fit in Window Button 9
Floating 12, 18
Flow of Events 97
Focus of Control
 Coloring 148
 Description 147
 Displaying 148
 Moving 149
 Nested 149
Formal Arguments 62
Framework
 Activating the Framework Wizard 189
 Changing 193
 Creating a Model from a Framework 190
 Creating a New Framework 192
 Deleting 193
 Library 191
 Wizard 189
Frequency 160
Friend 88
Friendship Required 90, 105
Full Import 207
Fully Qualified Names 32

G

General Tab
 About 47
 Association Specification 83
 Class Attribute Specification 72
 Class Instance Specification 153
 Class Specification 56
 Component Specification 163
 Decision Specification 128
 Dependency Specification 91
 Device Specification 174
 Generalize Specification 89, 104
 Has Specification 92
 Key/Qualifier Specification 94
 Link Specification 154
 Logical Package Specification 92
 Message Specification 158
 Object Flow Specification 136
 Object Specification 133, 151
 Operation Specification 75
 Package Specification 167
 Parameter Specification 82
 Process Specification 176
 Processor Specification 171
 Realize Specification 90
 State and Activity Specifications 121
 State Machine Specification 108
 State Transition Specification 126
 Swimlane Specification 120
 Synchronization Specification 131
Generalization 99
Generalize Specification 89, 104
Guard Condition 127

H

Has Relationship 92
Has Specification 92
 Detail Tab 93
 General Tab 92
Help Button 51
Help Topics Button 10

- Hiding
 - Browser 18
 - Classes 55
 - Stereotype Names 182
 - Type Library Items 205

I

- Icon 157
- Icons
 - Copying 35
 - Creating on the Diagram 30
 - Creating User-Defined Icons 186
 - Cutting 35
 - Deselecting 33
 - Moving 34
 - Pasting 35
 - Resizing 34
 - Selecting 33
- Implementation 58
- Importing a Type Library 203
- Include Stereotype 98
- Incoming Object Flows Tab 134
- Ini File 16
- Initial State, *see* Start State
- Initial Value 73
- Installing an Add-In 4
- Interaction Diagram 78, 79, 80
 - Creating 137
 - Overview 137
- Interface
 - in Type Library 198
 - of COM Component 198

K

- Key/Qualifier Specification 93
- Keys 89

L

- Language 69, 164, 166
- Laying Out a Diagram 38

- Library
 - Frameworks 191
 - Imported 207
 - Type 195
- Link Element 85
- Link Specification 154
 - General Tab 154
 - Messages Tab 157
- Linking Diagrams 28
- Links
 - Creating 145
 - Description 145
 - see also* Correlation 145
- Loading a Model Workspace 43
- Logical Package Specification 92
 - Detail Tab 93
 - General Tab 92

M

- Manipulating Icons 33
- Manuals *xxi*
- Maximize Button 7
- Menu Bar
 - Description 7
 - Illustration 6
- Menu Control Box 6
- Message Name 157
- Message Specification 158
 - Detail Tab 159
 - General Tab 158
- Message Tab 157
- Messages
 - Assigning an Operation to 144
 - Creating 143
 - Deleting 147
 - Description 142
 - Naming 143
 - Numbering 144
- Minimize Button 7
- Minus Sign 19

- Model
 - Navigating 18
 - Printing 14
 - Saving 15
- Model Elements
 - Changing 34
 - Creating 20
 - Deleting 36
 - Editing 20
 - Laying Out 38
 - Naming 30
 - Reassigning 32
 - Selecting in the Browser 20
- Model File 211
- Model Workspaces
 - Loading 43
 - Saving 43
 - Understanding 41, 42
- Modeling with Rational Rose 2
- Moving Icons 34
- Multiple Instances 152
- Multiple Objects 142

N

- Name 48, 101, 151
- Name Direction 85
- Naming
 - Correlations 38
 - Diagrams 29
 - Element 20
 - Element in the Browser 20
 - Fully Qualified Names 32
 - Model Elements 30, 31, 32
 - Overloaded Elements 31
- Navigable 87
- Navigating a Model 18
- Navigating the Tabs 52
 - Adding and Deleting Entries 52
 - Editing Entries 52
- Nested Class 70
- Nested Focus of Control 149
- Nested Tab 69
- New Model Button 7

- Notations 3
- Numbering Messages 144
- Numbering Sequences 145

O

- Object 141
- Object Browser in Visual Basic 202
- Object Flow 116
- Object Flow Specification 135
- Object Specification 132, 150
 - General Tab 133
 - Incoming Object Flows Tab 134
 - Outgoing Object Flows Tab 135
- Objects 115
- OK Button 51
- OLE Viewer 202
- On Event 125
- Online Help xxi
- Open Model Button 7
- Operation Specification 74
 - Detail Tab 76
 - Files Tab 80
 - General Tab 75
 - Postconditions Tab 80
 - Preconditions Tab 78
 - Semantics Tab 79
- Operations Tab 63
- Outgoing Object Flows Tab 135
- Overloaded Elements 31, 32
- Overview Window
 - Description 13
 - Icon 6
- Owner 82, 94

P

- Package 55, 57, 102, 167, 188
- Package Name 68
- Package Specification 166
 - Detail Tab 168
 - Files Tab 168
 - General Tab 167
 - Realizes Tab 168

- Parameter Specification 81
- Parent 9, 57, 83
- Paste Button 8
- Pasting Icons 35
- Persistence 61
- Persistence Field 152
- Petal File 211
- Plus Sign 19
- Positioning the Browser 18
- Postcondition 80
- Postconditions Tab 80
- Preconditions 78
- Preconditions Tab 78
- Print Diagrams Button 8
- Printing 14
- Priority 177
- Private 58
- Process Specification 176
- Processes 172
- Processor 177
- Processor Specification 170
 - Detail Tab 172
 - General Tab 171
- Processors 169
- Protected 58
- Protocol 76
- Public 58

Q

- Qualifications 77
- Qualifiers 89
- Quick Import 207

R

- Rank 102
- Rational Technical Support xxii
- Realize Specification 90
- Realizes Tab 165, 168
- Reassigning Model Elements 32
- Receiver 158
- Redo Command 35
- Referenced Type Libraries 200

- Refine Stereotype 99
- Relations 103
- Relations Tab 67, 103
- Relationship, see Correlation
- Relationships 97
- Renaming Model Elements 32
- Resizing Icons 34
- Return Class 75
- Role 84, 156
- Role A and B Detail Tab 87
- Role A and B General Tab 86
- Rose.ini File 16

S

- Save Model, Log, or Script Button 8
- Saving
 - Files 15
 - Model Workspaces 43
- Scheduling 173
- Scripts
 - Creating 146
 - Deleting 147
 - Detaching 147
 - Moving 147
- Selecting Elements 20
- Selecting Icons 33
- Semantics 79
- Semantics Tab 79
- Sequence 157
- Sequence Diagram
 - Creating 137, 150
 - Overview 139
 - Toolbox 141
- Sequence Numbering 145
- Shallow Delete 36
- Shapes, see Model Elements
- Shared 156
- Show all Classes 166
- Show All Components 68
- Show Classes 72
- Show Inherited 64
- Show labels 181
- Show Stereotype Names 182

- Show Stereotypes 181
- Size 77
- Snap-to-grid 34
- Sorting Packages 21
- Space 60
- Specification
 - About Common Elements 47
 - Action 124
 - Activity 121
 - Actor 105
 - Association 83
 - Class 56
 - Class Attribute 71
 - Class Instance 152
 - Component 163
 - Connection 175
 - Decision 128
 - Dependency 91
 - Device 173
 - Displaying 45
 - Editing 46
 - Generalize 89, 104
 - Has 92
 - Illustration 6
 - Key/Qualifier 93
 - Link 154
 - Logical Package 92
 - Message 158
 - Navigating the Tabs 52
 - Object 132, 150
 - Object Flow 135
 - Operation 74
 - Package 166
 - Parameter 81
 - Process 176
 - Processor 170
 - Realize 90
 - State 121
 - State Machine 107
 - State Transition 126
 - Swimlane 120
 - Synchronization 130
 - Tab Buttons 51
 - Use-Case 101
 - Window 13
- Standard Toolbar 7
- Start State 118
- State 133
- State Actions 125
- State Machine
 - Creating 107
 - Overview 107
- State Machine Specification 107
- State Specification 121
 - Actions Tab 122
 - General Tab 121
 - Swimlanes Tab 124
 - Transitions Tab 123
- State Transition 119
- State Transition Specification 126
 - Detail Tab 127
 - General Tab 126
- State/Activity History 122
- State/Activity Model Icon 107
- Statechart Diagram
 - Creating 109
 - Overview 108
- States 118
- Static 74, 88
- Stereotype 57, 84, 104, 164
 - Benefits 179
 - Creating 183
 - Creating Configuration File 183
 - Creating Icons 185
 - Description 179
 - Display 181
 - Icon 185
 - Sample 188
 - User-Defined 179
 - Viewing 180
- Subsystem Stereotype Package 188
- Supplier Visibility 155
- Support xxii
- Swimlane Specification 120
- Swimlanes 115
- Swimlanes Tab 124, 130
- Synchronization 160
- Synchronization Specification 130
 - General Tab 131
 - Transitions Tab 132

Synchronizations 119

T

Tabs

- Buttons 51
- Common 47
- Navigating 52

Technical Support xxii

Text

- Changing Default Font Parameters 40
- Moving 41
- Placing in Diagrams 40
- Resized the Text Box 41

Time 77

Title Bar

- Description 6
- Illustration 6

Toolbar

- Description 7
- Illustration 6

Toolbox

- Adding Stereotypes 187
- Class Diagram 54
- Collaboration Diagram 140
- Component Diagram 162
- Creating Icons for 186
- Customizing 11
- Deployment Diagram 170
- Description 10
- Illustration 6
- Sequence Diagram 141
- Use-Case Diagram 100

Training xxi

Transition Between Substates 127

Transition to Self 119

Transitions 119

Transitions Tab 123, 129, 132

Type 57, 72, 82, 122

Type Libraries

- About 195
- Adding Class Members 207
- Customizing the Importer 208
- Hiding and Displaying Items 205
- Importing into a Model 203
- Overview Diagram 199
- Reasons for Importing 196
- Referenced 200
- Types of Files to Import 196
- Updating 204
- Using 207
- Viewing in Rational Rose 197
- Viewing in Visual Studio 202

U

Undo Fit in Window Button 10

Undo Mistakes 35

Upgrading 211

URLs 50

Use Cases 96

Use-Case Diagram

- Overview 95
- Toolbox 100

Use-Case Specification 101

- Diagram Tab 102
- Relations Tab 103

User-Defined Stereotype 179

V

View Documentation Button 8

Virtual Inheritance 90, 105

Visibility 155

Visual Modeling 1

W

Workflows

Defining 113

Modeling with Activity Diagrams 114

Purposes 112

Understanding 111

Workspaces, see Model Workspaces

Z

Zoom In Button 9

Zoom Out Button 9