



USER MANUAL

version 17.0

No Magic, Inc.
2011

All material contained herein is considered proprietary information owned by No Magic, Inc. and is not to be shared, copied, or reproduced by any means. All information copyright 1998-2011 by No Magic, Inc. All Rights Reserved.

CONTENTS

Contents 3

1 INTRODUCING MAGICDRAW 18

About MagicDraw and UML 18

MagicDraw Editions and Features 19

 MagicDraw Editions 19

- MagicDraw Personal Edition 19
- MagicDraw Standard Edition 19
- MagicDraw Professional Edition 19
- MagicDraw Architect Edition 20
- MagicDraw Enterprise Edition 20
- MagicDraw Reader Edition 20

 Other MagicDraw Features and Add-ons 20

- Reports Generation 20
- Floating License 20
- Teamwork Server 21
- Code and Database Engineering 21
- OpenAPI 21
- Integrations 22
- MagicDraw Customization 22
- MagicDraw Plugins 22
- MagicDraw RConverter 25

 MagicDraw Welcome Screen 25

 MagicDraw News Reader 30

 MagicDraw Documentation and Support 31

 New and Noteworthy 32

 Manuals and User Guides 32

 Help 32

 Hints Associated to User's Actions 32

 Productivity Tips Displayed in Progress Window 33

 Tutorials 33

 Other Documentation 35

 Support 35

 FAQ 35

 Forum 35

 Knowledge Base. New issue registration 36

 Customer support levels 36

 End-of-life policy 37

 Reporting issues directly from MagicDraw 37

 View and submit internal errors 39

2 GETTING STARTED 40

 System Requirements 40

 Java Virtual Machine (JVM) 40

 Operating System- dependent issues 41

 Installation Procedure 41

CONTENTS

Using the installer **42**

Windows 2000/2003/NT/XP/Vista/7 **42**

Unix **42**

MAC OS X **42**

Using no-install package **42**

Windows 95/ 98/ NT/ 2000/ XP/ Vista*/ 7 **43**

Unix **43**

Mac OS X **43**

Licensing Information **43**

Removing Unused Licenses **44**

MagicDraw Configuration **44**

MagicDraw configuration files location **44**

Unlocking MagicDraw **46**

Activating the commercial license after the purchase (v16.9 or later) **47**

Commercial license activation process **48**

Online activation **48**

Offline activation **50**

Commercial license types **51**

Host ID **51**

Determining a Host ID using lmhostid **51**

Identification **52**

Registration **54**

Deactivation **54**

Deactivating from the License Activation Management screen **55**

License Deactivation ID **55**

Confirmed deactivation case **55**

Rehost limit **56**

User Registration **56**

Registration Workflow **57**

Adding a License **58**

Registering **58**

Confirming Your Registration **59**

Logging in to Your Dedicated Area at www.magicdraw.com **59**

Bug Report **60**

Troubleshooting **61**

Updating **62**

Auto-Check for Updates dialog box **62**

3 USING MAGICDRAW **63**

Customizing and Selecting Perspective **63**

Customizing MagicDraw Perspectives **64**

MagicDraw Startup dialog box **65**

Select Perspective dialog box **66**

Customize Perspectives dialog box **67**

Understanding MagicDraw User Interface (UI) **69**

Menus **70**

Toolbars **70**

Main toolbars **71**

Diagram toolbars **71**

Customizing toolbars **72**

Using the Model Browser **73**

CONTENTS

| | |
|--|----|
| Containment tree | 76 |
| Data branch | 79 |
| Code engineering sets | 80 |
| Diagrams tree | 81 |
| Inheritance tree | 82 |
| Model Extensions tree | 83 |
| Search Results Tree | 84 |
| Working with Elements in the Model Browser | 84 |
| Multiple selection | 85 |
| Zoom panel | 86 |
| Documentation panel | 87 |
| Properties panel | 87 |

Customizing Environment Options 89

| | |
|--|-----|
| General Options Tab | 90 |
| Diagram Options Tab | 94 |
| Browser Options Tab | 97 |
| Teamwork Options Tab | 98 |
| Floating Options Tab | 99 |
| CVS Options Tab | 100 |
| Update Options Tab | 101 |
| Network Options Tab | 102 |
| Keyboard Options Tab | 103 |
| Plugins Options Tab | 104 |
| Resources Options Tab | 105 |
| Path Variables Options Tab | 106 |
| Spelling Tab | 106 |
| Launchers Options Tab | 107 |
| Experience Options Tab | 108 |
| External Tools Options Tab | 109 |
| Eclipse UML2 (v1.x) XMI Options Tab | 110 |
| Eclipse UML2 (v2.x / v3.x) XMI Options Tab | 112 |
| Macros Options Tab | 114 |
| Report Wizard Options Tab | 114 |

Look and Feel: Controlling the Interface 117

Single and Multiple Windows interface styles 118

Assigning Shortcut Keys 118

4 WORKING WITH PROJECTS 119

Creating a Project 119

| | |
|---|-----|
| Creating a new project | 119 |
| Working with multiple projects | 121 |
| Creating a new project from the existing source code | 122 |
| Creating a new project from a previously created template | 122 |
| Creating a new Use Case project | 124 |

Saving a Project 125

Autosave 126

Opening a Project 127

Importing a Project 127

Exporting a Project 127

Exporting a project as a template 128

Exporting a project as an Eclipse UML2 (v1.x) XMI file 128

CONTENTS

| | |
|---|------------|
| Exporting a project as an Eclipse UML2 (v2.x/v3.x) XMI file | 128 |
| Exporting to BPEL | 129 |
| Setting Project Options | 132 |
| Project Options | 132 |
| Searching | 134 |
| Java Regular Expressions | 138 |
| Metacharacters | 138 |
| Character Sets | 138 |
| Grouping | 138 |
| Quantifiers | 138 |
| Boundary Matchers | 139 |
| Embedded Flag Expressions | 139 |
| Replacing | 140 |
| Project Partitioning | 141 |
| Partitioning the model | 141 |
| Exporting the module of a project | 142 |
| Sharing the module of a project | 143 |
| Managing modules (the Modules dialog box) | 144 |
| Analyzing Dependencies Among Elements | 145 |
| Using the module of a project | 145 |
| The Use Module Wizard | 145 |
| Reusing model parts between models | 147 |
| Reloading the module of a project | 148 |
| Importing the module of a project | 149 |
| Working with partially loaded projects | 149 |
| Advanced Concepts | 150 |
| Indexing | 150 |
| Missing elements for the proxies (orphaned proxies) | 151 |
| Displaying orphaned proxies in a project | 152 |
| Searching for orphan proxies | 152 |
| Resolving orphaned proxies | 152 |
| Creating New Substitute | 153 |
| Deleting orphan proxies | 154 |
| Exporting a Project as MOF | 154 |
| MOF Domain model | 155 |
| MOF export | 155 |
| Metamodeling template | 156 |
| Validation constraints | 156 |
| Ecore Support | 157 |
| Introduction | 157 |
| Preparing Ecore Models | 157 |
| Annotation modeling | 161 |
| Exporting | 162 |
| Validation | 164 |
| Importing | 165 |
| Working with Standard Profiles | 165 |
| Standard Profiles as System Resources | 165 |
| Plugin and Profile Versions | 166 |
| Standard Profiles in Teamwork Server | 166 |

CONTENTS

5 DIAGRAMMING 167

Working with Diagrams 167

Diagram Basics 167

Diagrams Dialog 170

Diagram Properties 171

Diagram Name and its Context Name Synchronization 173

Diagram Frame 173

Drawing Diagram Shapes 175

NEW! Overviewing Other Diagrams 178

Creating a diagram overview shape 178

Modifying the content of a diagram overview shape 183

Diagram overview shape environment 186

How to... 192

Table with Diagram Information 193

Changing the Diagram Type 194

Working with Shapes of Model Elements 195

Working with Paths and Relationships 198

Inserting a Shape on the Path 200

Creating Relations from the Model 201

Smart Manipulation 202

Selection and Multiple Selections 204

Copying/Pasting Text or Images to Diagrams 206

Nesting Image Shapes 207

Dragging, Copying, Cutting, and Pasting 207

Zooming 214

Using the Grid 215

Layout 216

Orthogonal Layout Tool 216

Hierarchic Layout Tool 217

Tree Layout Tool 218

Organic Layout Tool 220

Circular Layout Tool 221

Orthogonal Path Router 222

Organic Path Router 222

Class Diagram Layout Tool 223

Activity Diagram Layout Tool 224

Business Process Diagram Layout tool 225

Quick Diagram Layout feature 227

Label layout in the diagram 227

Default label positions 227

Labels positions after moving a path, shape or related element 227

Showing Diagrams in Full Screen 229

Floating Diagram Window 230

Saving as an Image 230

Setting image saving options 232

Printing 232

Print Range tab 233

Print Options Tab 234

Print Header/Footer Tab 236

CONTENTS

6 WORKING WITH MODEL ELEMENTS 237

Specification Window 237

Functions of Specification Window 239

Specification Window Toolbar 240

Quick filter 242

Specification window tabs 243

General tab 243

Documentation/Hyperlinks tab 244

Attributes tab 246

Usage in Diagrams tab 246

Operations tab 247

Template Parameters tab 248

Relations tab 249

Tags tab 250

Constraints tab 252

Traceability tab 252

Buttons available in the Specification window 253

Default Property Values 253

Sharing the default property values 254

Formatting Symbols 255

Displaying icon or image 256

Style Engine 258

Symbol Property Styles Tree 258

Working with Properties Styles 264

Properties extension by diagram 265

Properties Inheritance 267

General Style Properties 267

Shape, Path and Diagram Properties 267

Stereotype properties 268

Defining Hyperlinks Between Elements 270

Adding a hyperlink to the model element 270

Edit Hyperlink dialog box 273

Owner of the Model Element 274

Owner display mode 275

Qualified name starting from model library 276

Relations Changes Ownership when Client or Supplier is Moved to Other Owner 276

Selecting an Element 278

Element selection views 278

Element search criteria and modes 279

Element creation mode 280

Elements multiple selection 281

Refactoring 281

Converting an Element 281

Replacing an Element 282

HTML Editor 283

Advanced HTML editor 289

Copying/Opening Element URLs 291

7 TOOLS 292

Model Merge 293

CONTENTS

| | |
|--|------------|
| Definitions | 293 |
| Introduction to Merging | 293 |
| 3-way merge | 294 |
| 2-way merge | 294 |
| Model Merge Concepts | 295 |
| Model Merge demonstration | 295 |
| Usage scenario 1 - 3-way Merge and Analysis | 295 |
| Usage scenario 2 - Merging in Teamwork System | 299 |
| Usage scenario 3 - Copying changes from the branch to Trunk (in Teamwork) with conflict resolution | 305 |
| Usage scenario 4 - Representing DSL elements in the Merged results | 313 |
| Merging concepts in details | 315 |
| Change types | 315 |
| Accepting or Rejecting changes | 316 |
| Dependent changes | 316 |
| Conflicting changes | 317 |
| Building change tree | 317 |
| Analyzing Merging Results | 319 |
| The Merge window description | 319 |
| Viewing changes in diagrams | 333 |
| Finishing projects merge | 338 |
| Controlling Merge memory usage | 341 |
| Pattern Wizard | 342 |
| Creating Setters / Getters | 348 |
| Implementing or Overriding Operations | 349 |
| Model Transformation Wizard | 350 |
| Available Transformations | 351 |
| Working with Model Transformation Wizard | 351 |
| Selecting transformation type | 352 |
| Selecting source and destination models | 353 |
| Selecting type mappings | 354 |
| Setting custom transformation properties | 355 |
| Transformation Type Mapping | 356 |
| Controlling Type Mapping Rule Behavior | 357 |
| Controlling direction | 357 |
| Multiple rules for the same source type | 357 |
| Controlling type inheritance, any, and empty types | 358 |
| Type modifiers | 360 |
| Transformation Traces and Update | 360 |
| Profile Migration Transformation | 362 |
| Profile Migration Transformation mapping | 362 |
| Starting Profile Migration Transformation | 363 |
| Sample of the Profile Migration Transformation | 363 |
| Resource Manager | 366 |
| Spelling Checker | 368 |
| Spell checking as you type | 369 |
| Spell checking for the whole project or the selected scope | 371 |
| Checking spelling for the whole project | 371 |
| Checking spelling for the selected scope | 372 |
| Analyzing the Check Spelling (the Validation Results window) | 373 |
| Solving the spell checking errors | 373 |
| Setting the spell checking options | 375 |
| Spell checking options | 376 |

CONTENTS

| | |
|---|------------|
| Spell checking dictionaries | 377 |
| Defining properties of the customized element to be spell checked | 377 |
| Import Data to MagicDraw | 378 |
| Import data from Rational Software Architect/Modeler using MagicDraw RSXConverter | 378 |
| Import data from Rational Rose using MagicDraw RConverter | 378 |
| Import data from other tools | 378 |
| 8 MODEL ANALYSIS | 380 |
| Displaying Related Elements | 381 |
| Analyzing Usages and Dependencies | 382 |
| Understanding Used By | 382 |
| Understanding Depends On | 382 |
| Searching for Usages/Dependent Elements | 382 |
| Traceability | 386 |
| Creating Traceability Relations | 387 |
| Traceability Relations Representation | 388 |
| Traceability properties in Specification window | 389 |
| Traceability properties in Properties panel | 392 |
| Traceability properties in notes | 392 |
| Traceability properties in Go To submenu | 393 |
| Traceability properties in Traceability Report | 393 |
| Navigating between Different Levels of Abstraction | 394 |
| Analyzing Traceability Relations | 395 |
| Analysis using Relation Map | 395 |
| Analysis using Dependency Matrix | 395 |
| Predefined Traceability Rules | 397 |
| Relation Map | 398 |
| Creating the Relation Map | 399 |
| Defining Relation Map filters, layout, and depth | 399 |
| Choosing and changing context element | 400 |
| Choosing the relation criterion | 401 |
| Manipulation of Relation Map | 406 |
| Symbol Usage in Diagrams | 408 |
| Projects Comparison | 409 |
| Comparing Projects | 410 |
| Understanding model differences | 411 |
| Diagrams Comparison | 413 |
| Metrics | 415 |
| Metric Suites | 416 |
| Displaying Metrics | 416 |
| Starting Metrics | 417 |
| Metrics window | 419 |
| Exporting Metrics | 419 |
| Comparing metrics | 421 |
| Metrics Options | 421 |
| Metrics Properties | 424 |
| Dependency Matrix | 425 |
| Creating the Dependency Matrix | 426 |
| Dependency Matrix View | 427 |
| Dependency Matrix pane | 429 |
| Working with a Dependency Matrix Template | 431 |

CONTENTS

| | |
|--|------------|
| Dialog boxes in Dependency Matrix functionality | 432 |
| Dependency Matrix Templates dialog box | 432 |
| The Add / Remove Elements dialog box | 434 |
| The Dependency Matrix Specification dialog box, tags group | 435 |
| Analyzing Package Dependencies | 436 |
| Package Dependencies panel | 438 |
| Analyzing Dependencies Among Elements | 440 |
| Package dependencies by relationship | 440 |
| Dependencies by reference | 442 |
| Diagram dependencies | 443 |
| Unresolved dependencies | 443 |
| Validation | 444 |
| Introduction | 444 |
| Constraint Types | 444 |
| Predefined Validation Suites | 445 |
| Validating | 446 |
| Validation Results Window | 448 |
| Model Validation Example | 448 |
| Validation Rules | 450 |
| OCL Constraints | 453 |
| Binary Constraints | 454 |
| Validation Suites | 454 |
| Advanced Topics | 458 |
| Global validation rules | 458 |
| Expressions in error messages | 460 |
| Modeling other types OCL2.0 constraints/expressions | 460 |
| Unsupported OCL2.0 features | 461 |
| Adding/customizing severity levels | 462 |
| Performance Issues | 462 |
| Active Validation | 462 |
| Detecting errors in the model | 463 |
| Failure indicator | 463 |
| Marking errors in the Browser | 465 |
| Highlighting errors on the diagram | 465 |
| Handling incorrect model | 466 |
| Changing the Active Validation Options | 466 |
| The Active Validation Suites | 467 |
| Validating Parameters and Arguments Synchronization | 467 |
| Shape Ownership | 467 |
| UML model correctness | 471 |
| Validating the Orphaned Proxies (OP) | 471 |
| Customizing the Active Validation | 471 |
| Validate element that has no representation in diagram | 471 |
| Model Visualizer | 473 |
| Class Diagram Wizard | 473 |
| Package Dependency Wizard | 476 |
| Displaying package dependencies | 479 |
| Package Overview Diagram Wizard | 480 |
| Hierarchy Diagram Wizard | 485 |
| Realization Diagram Wizard | 488 |
| Activity Decomposition Hierarchy Wizard | 491 |
| Content Diagram Wizard | 494 |

CONTENTS

Sequence Diagram from Java Source Wizard **497**

9 UML DIAGRAMS **502**

Architectural Views **503**

 Use Case View **503**

 Structural View **503**

 Behavioral View **503**

 Implementation view **504**

 Environment view **504**

Class Diagram **504**

 Class diagram elements **505**

Use Case Diagram **510**

 Use Case diagram elements **511**

Communication Diagram **512**

 Communication Diagram elements **513**

Sequence Diagram **514**

 Sequence diagram elements **515**

 Combined fragments 517

 Messages 519

State Machine Diagram **520**

 State Machine Diagram elements **521**

Protocol State Machine Diagram **524**

 Protocol State Machine Diagram elements **524**

Activity Diagram **527**

 Activity Diagram elements **529**

 Smart Activity Diagram layout **534**

 Dynamic centerlines 534

 Diagram orientation 535

Implementation Diagram **536**

 Component diagram overview **536**

 Deployment diagram overview **537**

 Implementation Diagram elements **538**

Composite Structure Diagram **542**

 Composite Structure Diagram elements **543**

 Displaying existing Parts on the Composite Structure diagram creation **544**

Interaction Overview Diagram **545**

10 EXTENSION DIAGRAMS **547**

Common Elements **549**

User Interface Modeling Diagram **550**

 The Reasons Why Prototyping is so Important? 550

 Working with User Interface Modeling Diagrams **551**

 User Interface Modeling **552**

 User Interface Modeling diagram elements 552

 Modifying a table 557

 Modifying a tree 559

 Nesting 559

 Reusability 560

 Specifying elements 560

 Icon usage 562

CONTENTS

| | |
|--|------------|
| Using symbol properties | 563 |
| User interface prototyping | 564 |
| Case Studies for User Interface Modeling | 564 |
| Case Study 1 - Modelling User Interface for the Report Wizard Window | 565 |
| Case Study 2 - Slider Example | 568 |
| Case Study 3 - User Interface Prototyping Example | 570 |
| Content Diagram | 572 |
| Content Diagram Elements | 574 |
| Robustness Diagram | 577 |
| Robustness Diagram Elements | 579 |
| Web Diagram | 579 |
| Web Diagram Elements | 581 |
| CORBA IDL Diagram | 583 |
| CORBA IDL Diagram Elements | 584 |
| Generic DDL Diagram | 584 |
| Generic DDL Diagram Elements | 585 |
| Oracle DDL Diagram | 585 |
| Oracle DDL Diagram Elements | 586 |
| WSDL Diagram | 587 |
| WSDL Diagram Elements | 588 |
| XML Schema Diagram | 589 |
| XML Schema Diagram Elements | 590 |
| Time Diagram | 591 |
| Time Diagram Elements | 592 |
| Struts Diagram | 592 |
| Struts Diagram Elements | 593 |
| Networking Diagram | 594 |
| Networking Diagram Elements | 595 |
| Business Process Diagram | 597 |
| Business Process Diagram Elements | 598 |
| Free Form Diagram | 605 |
| Relation Map Diagram | 605 |
| Dependency Matrix | 605 |
| NEW! Generic Table | 605 |
| Creating Generic Tables | 606 |
| Using Generic Tables | 611 |
| Setting detailed column names | 612 |
| Modifying generic table | 612 |
| Manipulations in generic table | 615 |
| Generic Table Environment | 616 |
| Generic table toolbars | 616 |
| Generic table shortcut menu | 617 |
| Element row shortcut menu | 619 |
| 11 MODEL ELEMENTS | 620 |
| Common Model Elements in the Diagrams | 620 |
| Note, Comment | 621 |
| Anchor | 622 |
| Constraint path | 622 |
| Image Shape | 622 |
| Separator | 622 |

CONTENTS

| | |
|----------------------------------|---|
| Documentation | 623 |
| UML Extension Elements | 623 |
| Stereotype | 624 |
| | Stereotype Specification dialog box 627 |
| | Assigning a stereotype 628 |
| | Changing the stereotype display mode 628 |
| | Stereotype notation 629 |
| Tags | 630 |
| | Editing Tagged Value 630 |
| Constraint | 632 |
| | The Time and Duration Constraints 633 |
| | Working with Constraints 633 |
| | The Constraint Specification dialog box 634 |
| | OCL 635 |
| Profiles | 637 |
| | Profiles 638 |
| | Working with profiles 638 |
| | Creating Profiles 638 |
| | Using and Importing Profiles 638 |
| | Exporting Profiles 639 |
| Action | 639 |
| | Accept Event Action 639 |
| | Call Behavior Action 642 |
| | Call Operation Action 643 |
| | Opaque Action 644 |
| | Send Signal Action 645 |
| | Working with actions 648 |
| | Advancing actions: applying duration constraint 649 |
| Actor | 650 |
| Association | 651 |
| | Association End 654 |
| | Advancing actions: navigable owned association ends 658 |
| Association in Use Case Diagrams | 659 |
| Attribute | 659 |
| Class | 663 |
| | Working with classes 663 |
| | Creating A Structured Class 665 |
| | Design Patterns 666 |
| | Class presentation options 666 |
| Collaboration | 668 |
| Combined Fragment | 670 |
| Component | 671 |
| Connector | 673 |
| Containment | 676 |
| Data type | 677 |
| | Enumeration 678 |
| | Primitive 679 |
| Decision Node | 680 |
| Dependency | 680 |
| | Template Binding dependency 681 |
| | Abstraction 682 |
| | Usage 682 |

CONTENTS

| | |
|---|------------|
| Package Merge | 683 |
| Package Import | 683 |
| Element Import | 683 |
| Access | 683 |
| Deployment | 684 |
| Deployment Specification | 684 |
| Events | 684 |
| Event types | 685 |
| Usability | 685 |
| Exception Handler | 686 |
| Extend | 686 |
| Flow Final Node | 688 |
| Fragment | 688 |
| Alternative Fragment | 688 |
| Combined Fragment | 689 |
| Function Behavior | 691 |
| Gate | 691 |
| The formal gate and actual gate usage in the sequence diagram | 693 |
| Generalization | 694 |
| Generalizable elements | 695 |
| Generalization sets | 695 |
| Include | 699 |
| Information Flow | 700 |
| Information Item | 701 |
| Instance | 702 |
| Instance Specification | 705 |
| Interaction Use | 706 |
| Interface | 706 |
| Provided and Required Interfaces | 707 |
| Provided and Required Interfaces in the Composite Structure diagram | 708 |
| Provided/required interfaces in the Component diagram | 709 |
| Internal transition | 710 |
| Lifeline | 710 |
| Lifeline in the Sequence Diagram | 712 |
| Link | 712 |
| Manifestations | 713 |
| Message | 713 |
| Assigning/Creating operation for message | 717 |
| Assigning/Creating signal reception for message | 718 |
| Predecessors and activators | 718 |
| Message in Sequence Diagram | 719 |
| Creating nested activation | 719 |
| Model | 721 |
| Node | 722 |
| Structured activity node | 724 |
| Activity parameter node | 725 |
| Expansion Region and Expansion Nodes | 725 |
| If, Loop and Sequence Conditional Nodes | 725 |
| Object | 726 |
| Object Flow | 726 |

CONTENTS

| | |
|--|-----|
| Object Node | 727 |
| Opaque Behavior | 728 |
| Operation | 728 |
| Package | 733 |
| Working with packages | 733 |
| Parameter | 736 |
| Parameters synchronization with Arguments | 736 |
| Rules of synchronization between parameters and arguments | 737 |
| Synchronization between Operation parameters and Behavior parameters | 738 |
| Synchronization between Activity parameters and Activity Parameter Nodes | 740 |
| Synchronization between Operation parameters and pins on Call Operation Action | 741 |
| Synchronization between Behavior parameters and pins of Call Behavior Action | 742 |
| Synchronization between Interaction parameters and Interaction Use arguments | 744 |
| Synchronization between Operation parameters and Message arguments | 744 |
| Synchronization between Interaction Parameters and Lifelines | 746 |
| The Parameters Synchronization dialog box | 747 |
| Port | 752 |
| Pseudo State | 755 |
| Initial | 755 |
| Final state | 755 |
| Terminate | 756 |
| Entry Point | 756 |
| Exit Point | 756 |
| Deep History | 756 |
| Shallow History | 756 |
| Junction | 756 |
| Choice | 756 |
| Fork and Join | 757 |
| Realization | 757 |
| Reception | 759 |
| Send Signal Action | 760 |
| State | 761 |
| Changing State to Composite/submachine/orthogonal State | 764 |
| Composite State | 764 |
| Submachine | 765 |
| Adding connection point reference | 765 |
| Defining State Invariant | 766 |
| Assigning behavior to state | 766 |
| Subsystem | 766 |
| Swimlane | 767 |
| Template / Parameterized class | 771 |
| Transition | 772 |
| Use Case | 776 |
| Use Case Extension | 777 |
| Value Specification | 779 |
| 12 APPENDIX I: MENUS, BUTTONS, AND ICONS | 781 |
| Menu System | 782 |
| File menu | 782 |
| Edit menu | 784 |
| View menu | 787 |

CONTENTS

| | |
|---|------------|
| Layout menu | 788 |
| Diagrams menu | 791 |
| Options menu | 791 |
| Tools menu | 793 |
| Analyze menu | 796 |
| Teamwork menu | 797 |
| Window menu | 797 |
| Help menu | 798 |
| Main Toolbars | 798 |
| File Toolbar | 800 |
| Diagrams Toolbar | 801 |
| Custom Diagrams Toolbar | 801 |
| Diagrams Navigation Toolbar | 803 |
| Opened Projects Toolbar | 804 |
| Perspectives Toolbar | 804 |
| Validation Toolbar | 804 |
| Diagram Toolbars | 804 |
| Browser Window | 807 |
| Floating, Auto-hide and Close Buttons | 807 |
| Buttons from the Containment Tree | 808 |
| Buttons from the Inheritance Tree | 809 |
| Buttons from the Diagrams Tree | 809 |
| Buttons from the Model Extensions Tree | 810 |
| Buttons from the Search Results Tree | 810 |
| Icons of general elements | 810 |
| Icons of relationships | 820 |
| Icons from Modules and Profile mechanism | 823 |
| 13 APPENDIX II: UML 2.3 SUPPORT | 825 |
| Introduction | 825 |
| Metamodel Changes | 825 |
| Notation Changes | 827 |
| Opening up Older Models | 828 |
| Index | 829 |

1 INTRODUCING MAGICDRAW

In the “Introducing MagicDraw” chapter, you will find the introductory information about MagicDraw.

- “About MagicDraw and UML” on page 18
- “MagicDraw Editions and Features” on page 19
- “MagicDraw Welcome Screen” on page 25
- “MagicDraw News Reader” on page 30
- “MagicDraw Documentation and Support” on page 31

About MagicDraw and UML

Today's graphical software can be extremely complex in its structure and architecture, but that does not mean it must be difficult to use. We have learned much from the hardware industry, where everything you see is scattered pieces. This approach also works well in the software world – objects at a higher abstraction level are treated like “software pieces”. To simplify the process further, we may use pictures instead of textual descriptions to show the relationships between objects in a complex system. Though pictures work better than textual descriptions alone, experience has proven that communicating complex ideas effectively requires more than simple flowcharts.

Early methodologies, such as Booch notation, OMT, and others served the same purpose: to graphically express the software's architecture information. However, these methodologies accomplished this in slightly different ways and with different levels of thoroughness. In 1994, Grady Booch, Jim Raumbaugh, and Ivar Jacobson came together to unify their varied methods and experience. The UML (Unified Modeling Language) was the fruit of their joint effort. UML was crafted with two objectives: To reflect the best practices of the industry and to demystify the process of software system modeling.

In short, UML provides standardized pictures of your software applications and allows your development team to quickly grasp the functionality contained within the application. UML is a language and a process with neutral notation. This means that you can use it to design your entire OO system in any programming language and any software development process.

The development of a model for an industrial-strength software system, prior to its construction or renovation, is as essential as having a blueprint for a large building. Good models are vital for effective communication among project teams.

In the early 1990s, the tools for OO software modeling emerged, followed by the development of the visual modeling approach. Visual modeling means that you first design your system by drawing diagrams (blueprints) and then employ tools to convert those diagrams into code. The value of such an approach is that the often tedious framework coding is done automatically, freeing the programmer to focus on design issues. The transition from the design to the implementation phase is smoother and more straightforward. Moreover, using the features of reverse engineering and code generation, the developer can move back and forth between the code and the design that is being expressed in the diagrams.

Today, visual modeling tools provide many features that replace some of the more tedious tasks for the designer, programmer, and documentation writer. Some of the leading tools provide so-called round-trip code engineering capabilities – the structure of reverse engineered code is changed in the modeling tool and is generated back without the implementation of specific information (e.g. method bodies, comments) being lost.

MagicDraw is a visual UML modeling and CASE tool with teamwork support. Designed for Business Analysts, Software Analysts, Programmers, QA Engineers, and Documentation Writers, this dynamic and versatile

development tool facilitates analysis and design of Object Oriented (OO) systems and databases. It provides the industry's best code engineering mechanism (with full round-trip support for Java, C#, C++, WSDL, XML Schema, and CORBA IDL programming languages), as well as database schema modeling, DDL generation and reverse engineering facilities.

As of version 16.8, MagicDraw supports UML v2.3. For the detailed information about supported changes of UML specification from version 2.2 to 2.3, see “Appendix II: UML 2.3 Support” on page 825.

MagicDraw Editions and Features

A detailed list of MagicDraw features can be found at:

<http://www.magicdraw.com/files/brochures/a4/MagicDrawDataSheet.pdf>

MagicDraw Editions

MagicDraw Personal Edition

MagicDraw Personal Edition contains powerful UML diagramming capabilities, including full UML 2 support and extensibility features, basic reporting functionality, and image export. Exported files are stored in XMI format.

All model elements can be accessed via the MagicDraw Open API.

In this edition, you will find everything you need to draw, edit, and publish your UML models.

Personal Edition is available only in a standalone version and is not designed for use with MagicDraw Teamwork Server.

MagicDraw Standard Edition

MagicDraw Standard Edition provides all of the Features of Personal Edition and adds WAE, content, and Robustness diagrams. Standard Edition also adds model analysis and facilitation features, customizable and extendable patterns, integrations with most popular IDEs, and a set of predefined model templates and UML profiles.

Standard Edition supports UNISYS XMI and the latest Model Driven Architecture (MDA) tool offerings. UNISYS XMI diagramming extensions allow the interchange of MagicDraw models with other UML modeling tools.

Standard Edition is available in standalone, floating and mobile license versions and is fully compatible with MagicDraw Teamwork Server.

Standard Edition is ideally suited for analysts and architects who need various model extensions and modeling facilitations.

MagicDraw Professional Edition

Professional Edition is built on the Standard Edition capabilities and is available in one of three programming language specific versions-Java, C++ and C#. In addition to the Standard Edition features, Professional Edition adds code generation and reverse engineering functionality. Depending on the language version selected, the user will receive:

- **Java version** - Code engineering for Java, Java bytecode. Integration with Java IDEs.
- **C++ version** - Code engineering for C++.

- **C# version** - Code engineering for C#, CIL (MSIL).

Professional Edition is ideal for anyone who wants to generate code from an existing model or create a UML model from an existing project.

MagicDraw Architect Edition

The Architect Edition is specially packaged to provide the optimal price and technical features necessary for architects that do not need the full capabilities of the Enterprise Edition. This edition combines the common functionality of the Standard Edition together with some powerful options from the Enterprise Edition. These include: advanced modeling facilitations and analysis, reverse engineering and code generation for DDL, WSDL, CORBA IDL and XML. Architects have less need for IDE integrations as well as Java and C++ code engineering, so these capabilities are not included.

MagicDraw Enterprise Edition

MagicDraw Enterprise Edition represents the top of the line in the MagicDraw family of products, as well as the ultimate solution for all your modeling needs. Enterprise Edition combines all of the functionality of the Personal and Standard Editions, and all three versions of the Professional Edition, into a comprehensive state-of-the-art UML programming solution. But the Enterprise Edition does not stop there, adding code engineering and diagramming functionality in CORBA IDL, EJB, WSDL and XML schema. For working with DB structure, Enterprise Edition not only provides code engineering and diagramming, but also provides structure retrieval via JDBC.

Enterprise Edition is a must when working with multiple development technologies and databases.

The MagicDraw family of award-winning products represents the most powerful and best value in the UML modeling industry today.

MagicDraw Reader Edition

MagicDraw Reader Edition is made for reading and previewing UML models created with MagicDraw and it is free of charge. It is extremely useful when you want to share your ideas expressed in UML with partners, colleagues, or clients, who do not have a copy of MagicDraw. Printing and image export capabilities are also included.

Since MagicDraw version 14.0, Reader Edition has the ability to open and review Teamwork Server projects.

Other MagicDraw Features and Add-ons

Reports Generation

You will find a complete description of the MagicDraw Report Wizard, related OpenAPI, and tutorial in "[MagicDraw ReportWizard UserGuide.pdf](#)".

Floating License

The Floating license agreement does not limit the number of clients you can install on different computers. It only limits the number of applications that can run at the same time. To control loaded applications, a server is required. The server can be installed on several computers, but simultaneously can be started only on the one of them. The license key of the floating server provides information to the server about how many applications may run simultaneously for the particular MagicDraw edition. If you upgrade the MagicDraw version, you do not

1 INTRODUCING MAGICDRAW

MagicDraw Editions and Features

have to obtain a new license key for the server. The Administrator's Console is used to manage client connections and configure the server.

For more information about MagicDraw Floating License, see "[MagicDraw FloatingLicense UserGuide.pdf](#)".

Teamwork Server

| | |
|-------------|---|
| NOTE | The Teamwork Server is available with MagicDraw application (Standard, Professional, Architect, and Enterprise editions). |
|-------------|---|

With MagicDraw Teamwork Server, you can assign as many developers as needed to work simultaneously on the same project using multiple workstations. The resulting Teamwork project is saved on the server for sharing by other MagicDraw applications. Users with administrator rights can create new users by giving them their own login name and various permissions to work on projects. Depending on permissions, users can update, commit, edit, create, and delete model elements, diagrams, and projects.

To enable Teamwork support, you should install and run the MagicDraw Teamwork Server software. Each MagicDraw application acts as a client in the Teamwork system.

Teamwork Server functionality is available with MagicDraw client Standard, Professional, and Enterprise Editions only. MagicDraw Reader Edition is allowed to open and review teamwork projects.

NEW! As of version 17.0 you can use the secure connection (SSL) while working with Teamwork Server.

For more information about Teamwork License Server, see "[MagicDraw Teamwork UserGuide.pdf](#)".

Code and Database Engineering

MagicDraw code engineering provides a simple and intuitive graphical interface for merging code and UML models, as well as preparing both code skeletons out of UML models and models from code.

MagicDraw code engineering features can be very useful in several situations:

- You already have code that needs to be reversed to a model.
- You wish to have the implementation of the created model.
- You need to merge your models and code.

The tool may generate code from models and create models out of code (reverse). Changes in the existing code can be reflected in the model, and model changes may also be seen in your code. Independent changes to a model and code can be merged without destroying data in the code or model.

MagicDraw UML code engineering supports Java, Java Bytecode, C++ (ANSI, CLI, Managed), C#, CIL, CIL Disassembler, CORBA IDL, DDL (Cloudscape, DB2, Microsoft Access, Microsoft SQL server, MySQL, Oracle, Pervasive, Pointbase, PostgreSQL, Standard SQL, Sybase), XML Schema, WSDL, and EJB 2.0 notation.

For more information on working with code engineering and databases, see "[MagicDraw Code&DatabaseEngineering UserGuide.pdf](#)".

OpenAPI

This document describes the MagicDraw Open Java API and provides instructions on how to write your own plug-ins, create actions in the menus and toolbars, change UML model elements, and create new patterns.

For more information on working with OpenAPI, see "[MagicDraw OpenAPI UserGuide.pdf](#)".

Integrations

MagicDraw supports the following integrations:

- Eclipse
- RAD
- BEA Workshop
- IntelliJ
- JBuilder
- NetBeans
- OAW
- AndroMDA
- ProActivity
- CVS
- CaliberRM

For more information about MagicDraw integrations, see “[MagicDraw Integrations UserGuide.pdf](#)”.

MagicDraw Customization

MagicDraw introduces several advanced customization engines, based on UML Profiles:

- Custom Diagram Wizard allows creating your own diagram types for custom profile. You may include your own toolbars, stereotyped elements, symbol styles, and custom smart manipulators. Such customization is saved in the special “diagram descriptor” that could be exchanged between users. This allows others to use your custom diagrams.
- Domain Specific Language Customization Engine (DSL customization engine) allows “tuning” domain specific profiles, customizing multiple GUI, model initialization, and semantic rules, creating your own specification dialogs. DSL customization is model-driven approach, based on UML profiling. Customization is saved as a UML model.
- Advanced UML Profiling allows the use of some profiling enhancements that are not defined in UML, but helps to solve some common problems like tag grouping, unwanted stereotypes, tags hiding, etc.

For more information about MagicDraw customization, see “[UML Profiling and DSL UserGuide.pdf](#)”.

MagicDraw Plugins

For the full MagicDraw plugins list, see the following website: www.magicdraw.com/plugin.

For the third party add-ons, check out https://secure.nomagic.com/third_party_plugins

| | |
|-------------|---|
| NOTE | For information about MagicDraw and Plugins compatibility see the following website: http://www.magicdraw.com/compatibility . The table shows which versions of MagicDraw and MagicDraw Plugins can work together. |
|-------------|---|

UPDM

The UPDM plugin for MagicDraw UML/SysML modeling solution fully supports building integrated enterprise architectures meeting DoDAF and MODAF requirements ensuring mission critical project success. The plugin supports all DoDAF 1.5, DoDAF 2.0, and MODAF 1.2 viewpoints and views dependent on the selected user

1 INTRODUCING MAGICDRAW

MagicDraw Editions and Features

environment. Each user environment provides architecture framework specific concepts, artifacts, new project templates, samples, and architecture framework specific usability features. A user environment can be changed any time by fully converting model to meet requirements of the selected architecture framework.

More information at https://secure.nomagic.com/updm_plugin

SysML

The SysML plugin includes SysML profile, template, all SysML diagrams, SysML samples project, SysML usability features, and System Engineer perspective. The System Engineer perspective (the specific mode of the MagicDraw user interface for SysML modeling) includes SysML specific menus, toolbars, diagrams, specification dialogs and user interface.

More information at <https://secure.nomagic.com/sysml>

Cameo DataHub

Cameo DataHub™ is a tool that allows the user to import, export, synchronize, and make references between Cameo Requirements+™, MagicDraw, SysML Plugin, DoDAF Plugin, Telelogic DOORS, Rational RequisitePro, and CSV files.

More information at <https://secure.nomagic.com/cameodatahub>

Cameo SOA+

Cameo SOA+™ leverages the Unified Modeling Language® (UML®) along with the latest SOA modeling standard, SoaML™, to provide both architects and developers an integrated solution for creating optimal SOA architectures and implementations. Cameo SOA+™ brings together SOA at both the business and technology levels to address the full spectrum of services. From Enterprise and Business Architectures to implementing, using and composing services on your favorite enterprise service bus (ESB) or application server, this integrated plug-in is versatile enough for both personal and team-based development. SoaML helps create and use services based on new and existing capabilities using composite services.

More information at https://secure.nomagic.com/cameo_soa

Cameo Data Modeler

Cameo Data Modeler™ plugin for MagicDraw® provides support for Entity-Relationship modeling. It expands previous Business Entity-Relationship diagram (a limited form of ER diagram) to full featured Entity-Relationship diagram - including extended entity-relationship concepts - generalization support.

More information at https://secure.nomagic.com/cameo_data_modeler

Cameo Business Modeler

Cameo Business Modeler™ plugin for MagicDraw provides support for BPMN 2.0 profile, diagrams, user perspective, usability features for BPMN modeling, manual, samples, and import from BPMN 1.1 models that were created with MagicDraw. All four BPMN 2.0 diagrams are supported.

More information at <https://secure.nomagic.com/cameobusinessmodeler>

Merge

Model Merge enables copying changes between different project versions. This functionality is usually needed when there are several branches that reflect different releases or versions of the product, e.g. when certain fixes have to be copied from a release branch to the mainstream development.

More information at <https://secure.nomagic.com/merge>

1 INTRODUCING MAGICDRAW

MagicDraw Editions and Features

MagicRQ

MagicRQ plugin for MagicDraw is the first true Requirements Hub that can move requirements between Telelogic's DOORS or IBM Rational® RequisitePro® seamlessly. All of the requirements information is moved into MagicDraw for maximum traceability and interaction.

More information at <https://secure.nomagic.com/magicrq>

ParaMagic

ParaMagic plugin using the quantitative information and constraint relationships displayed in SysML diagrams, model-builders can run simulations from the earliest stages of system design. In traditional domains of system engineering like aerospace and transportation, users can explore system performance, estimate cost and allocate resources. Developers leveraging MagicDraw's DoDAF and business modeling capabilities can add parametric simulation using SysML submodels for defense planning, business process analysis and computational finance.

More information at <https://secure.nomagic.com/paramagic>

DoDAF

The DoDAF plugin includes DoDAF profile, template, all DoDAF diagrams, DoDAF documentation generation feature, DoDAF samples project, DoDAF usability features, and DoDAF Architect perspective. The DoDAF Architect perspective (the specific mode of the MagicDraw user interface for DoDAF modeling) includes DoDAF specific menus, toolbars, diagrams, specification dialogs and user interface.

More information at <https://secure.nomagic.com/dodaf>

Methodology Wizards

Methodology Wizards Plugin automates modeling tasks and extends methodology support in MagicDraw. This Wizard guides you through model creation process according to a set methodology.

More information at https://secure.nomagic.com/methodology_wizards

MARTE Profile

This specification of a UML® profile adds capabilities to UML for model-driven development of Real Time and Embedded Systems (RTES). This extension, called the UML profile for MARTE (in short MARTE), provides support for specification, design, and verification/validation stages. This new profile is intended to replace the existing UML Profile for Schedulability, Performance and Time.

More information at https://secure.nomagic.com/marte_profile

CSV Import

The CSV Import plugin is a MagicDraw plugin that will read values in a comma separated values (CSV) file and create model elements, diagrams and relationships from that data. MagicDraw users will have the capability to create MagicDraw models from their data stored in spreadsheets, relational databases and other repositories.

More information at https://secure.nomagic.com/csv_import_plugin

SPEM

SPEM - Adopted standard for software engineering process description. Standard provides generic elements those allow to describe any software development process. The purpose of SPEM is to support the definition of software development processes specifically including those processes that involve or mandate the use of UML. The SPEM plugin includes SPEM profile, template, all SPEM diagrams and properties customization.

1 INTRODUCING MAGICDRAW

MagicDraw Welcome Screen

More information at https://secure.nomagic.com/spem_plugin

SoaML Profile

Service oriented architecture Modeling Language (SoaML) profile. The goals of SoaML are to support the activities of service modeling and design and to fit into an overall model-driven development.

Enterprise Architect Import Plugin

MagicDraw has the capability to import UML models that conform to various XMI versions (including XMI 2.1) from other tools. Sparx Systems Enterprise Architect (EA) is one of those tools. EA provides a XMI export functionality which makes it possible to export UML2.1(XMI2.1). However, the XMI exported from EA contains some XMI conflicts and EA-specific data that do not conform to UML standards. The main purposes of Enterprise Architect Import Plugin are thus to solve XMI conflicts between EA and MagicDraw that cause problems when loading the XMI to MagicDraw and also to transform some EA-specific data to the form of UML elements with stereotypes.

MagicDraw RConverter

MagicDraw RConverter generates data files in Rational Rose's (*.mdl) to MagicDraw's (*.xml). By obtaining information from Rational Rose via Rational Rose API (REI) and using VB6.0 to calculate the change, the resulting file is saved in the MagicDraw file (xml file extension) format.

For information on working with RConverter, see “[NM_MagicDraw_RConverter_UserManual.pdf](#)”.

MagicDraw Welcome Screen

The Welcome screen is displayed in the MagicDraw desktop when no projects are opened. It helps to manage projects, provides quick access to the product descriptions, samples, the latest news and updates. See the Welcome screen in Figure 1 on page 26.

To open the Welcome screen from the **Help** main menu, choose the **Show Welcome Screen** command (note that no projects should be opened).

1 INTRODUCING MAGICDRAW

MagicDraw Welcome Screen

The **Show Welcome Screen** option is added to the **Environment Options** dialog box, **General** branch, **Display** group. Using this option you can set whether the Welcome Screen will be displayed.

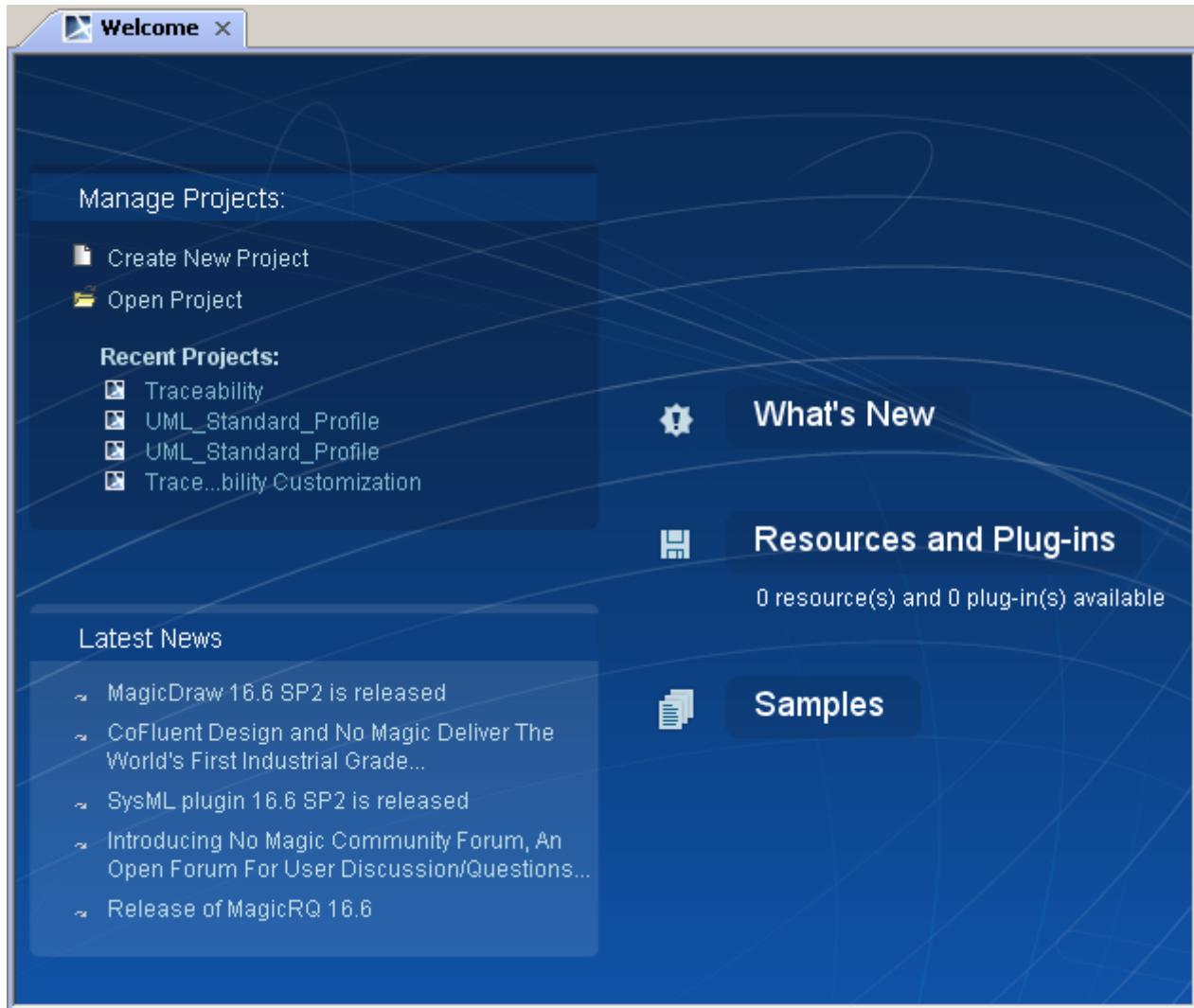


Figure 1 -- The MagicDraw Welcome Screen

Manage Projects

- Click **Create New Project**, to create a blank project.
- Click **Open Project**, to open existing project.
- The **Recent Projects** list contains list of the recently opened projects.

Latest News

- Click on the particular news to read more detailed description. The news will be displayed in the MagicDraw News Reader. For more information, see “MagicDraw News Reader” on page 30.

What's New tab

Click the description to open the following items:

- Introduction to MagicDraw. The introductory MagicDraw sample is opened.

1 INTRODUCING MAGICDRAW

MagicDraw Welcome Screen

- New and noteworthy. The www.magicdraw.com/newandnoteworthy webpage is opened. See the list of a new MagicDraw features for the newest version.
- MagicDraw news. The MagicDraw News Reader is opened. For more information, see “MagicDraw News Reader” on page 30.
- MagicDraw updates. The **Updates Information** dialog box opens, with a list of the latest updates. Here you will be able to see what updates are available and update to a newer version.
- MagicDraw home page. Click this link to navigate to the www.magicdraw.com webpage.

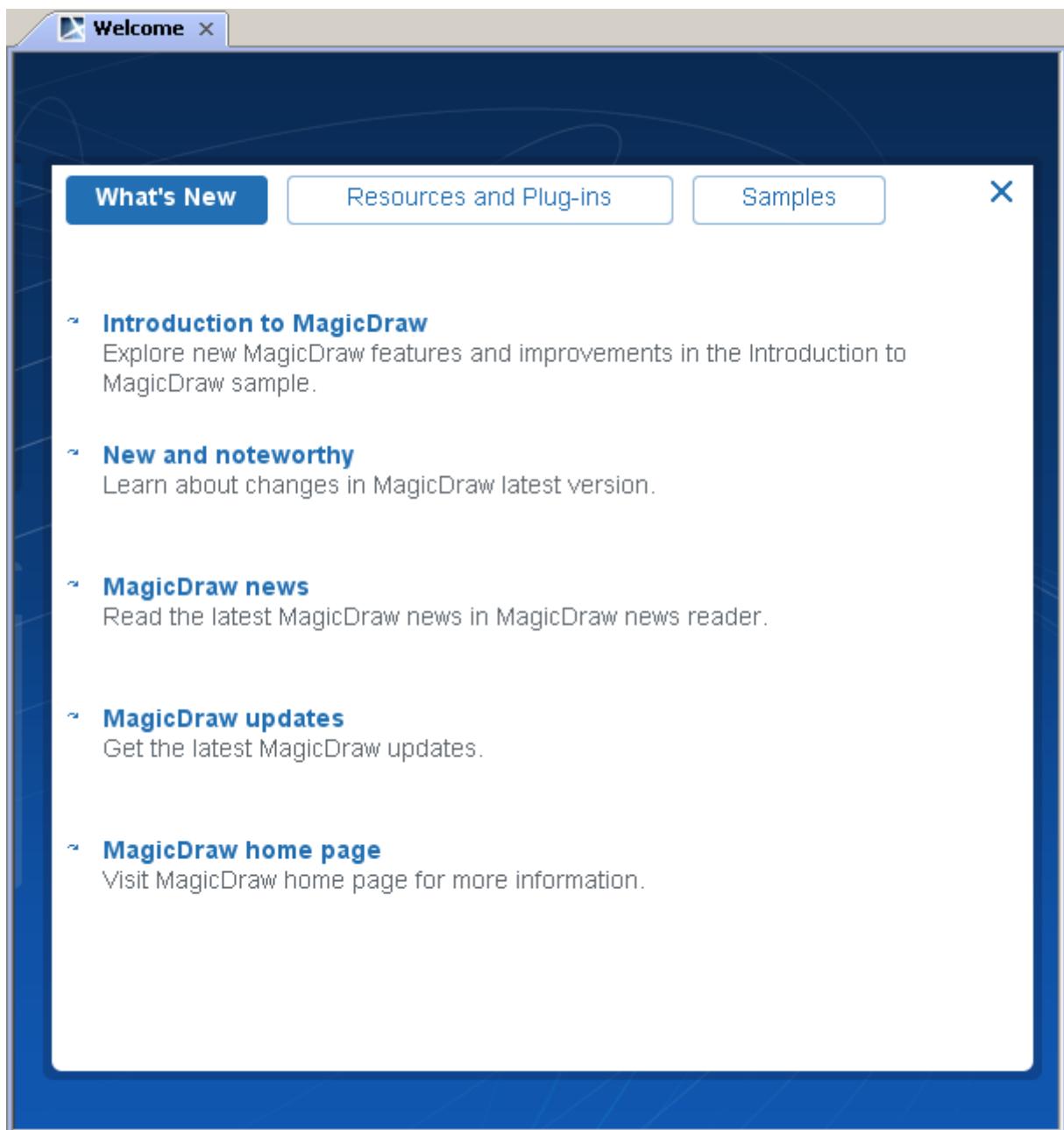


Figure 2 -- The Welcome Screen, What's New tab

Resources and Plugins

- Review available plugins. The www.magicdraw.com/plugin page is opened. All available plugins descriptions are available in this page.

1 INTRODUCING MAGICDRAW

MagicDraw Welcome Screen

- Install available resources. Click to open the **Resource/Plugin Manager** window. Download and install available resources and plugins using this manager. For more information, see “Resource Manager” on page 366.
- MagicDraw manuals. Click to open the <MagicDraw installation directory>\manuals folder, which contains MagicDraw manuals.
- Online demos. The www.magicdraw.com/viewlets webpage is opened. View demos, which introduces MagicDraw and MagicDraw features.
- MagicDraw eSchool. The <http://school.nomagic.com/> webpage is opened.

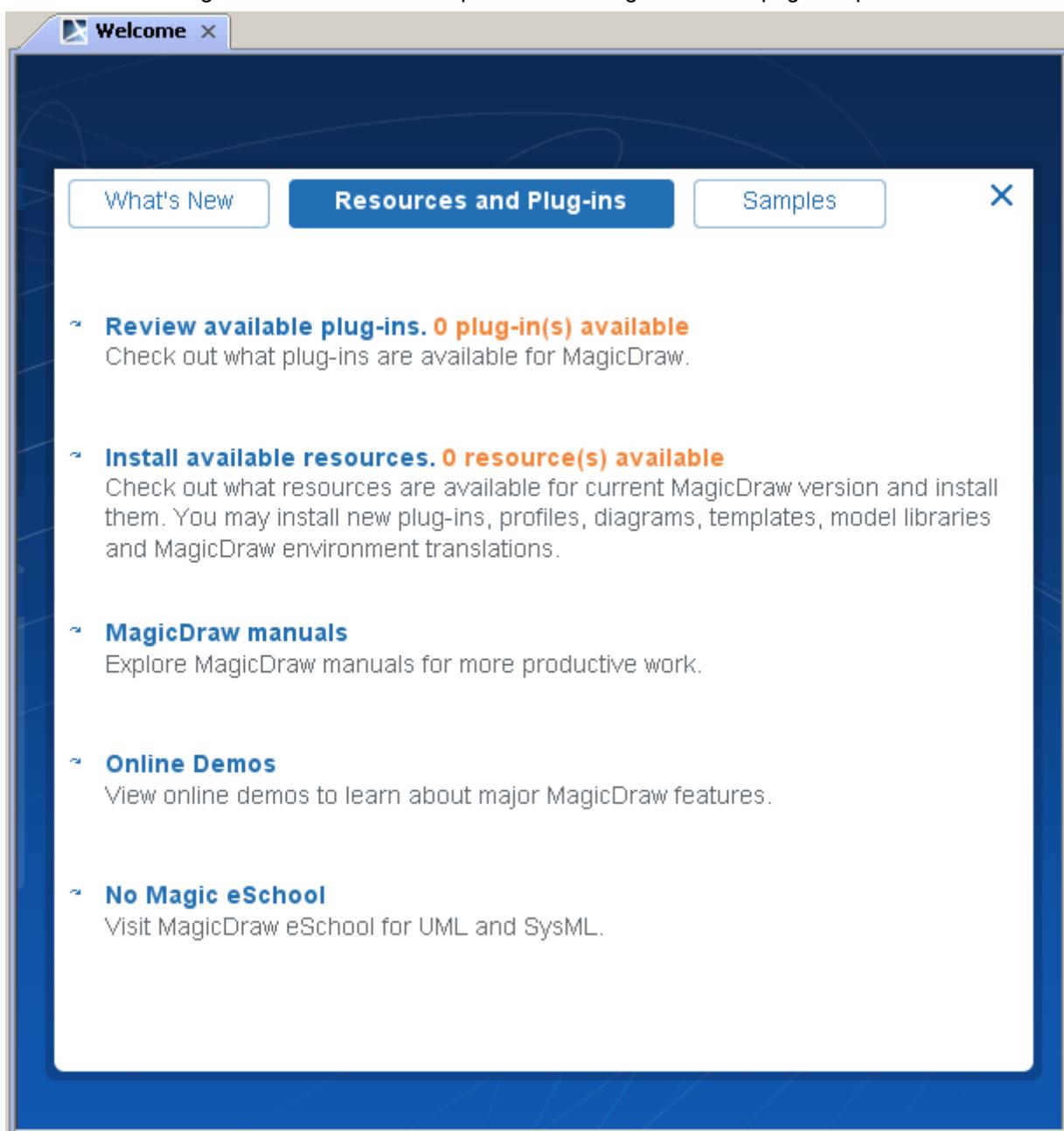


Figure 3 -- The Welcome Screen, Resources and Plugins tab

1 INTRODUCING MAGICDRAW

MagicDraw Welcome Screen

Samples

Quick navigation to MagicDraw samples. All samples from <MagicDraw installation folder> \samples directory are accessible in this page.

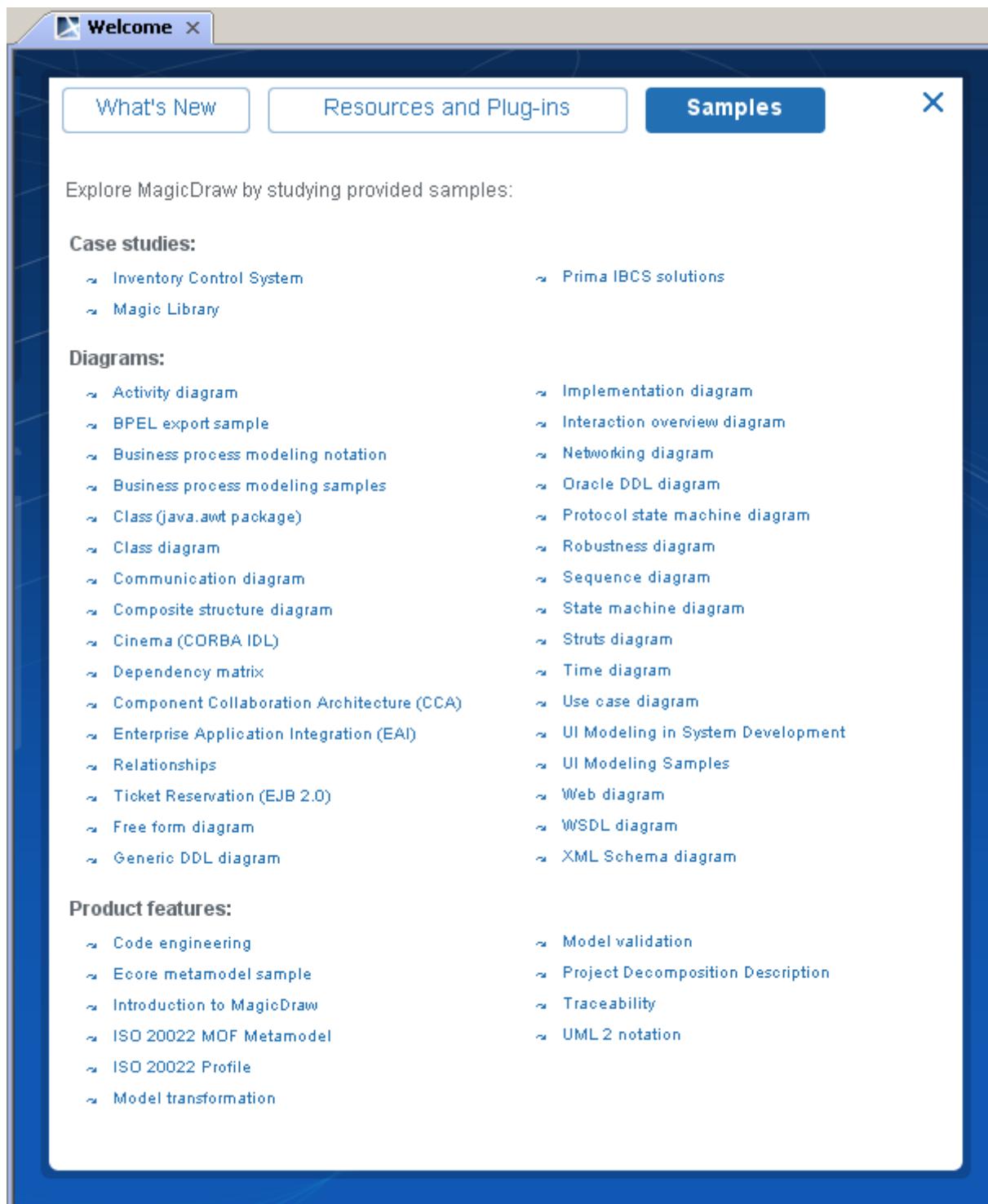


Figure 4 -- The Welcome Screen, Samples tab

MagicDraw News Reader

Information about the latest MagicDraw events is provided in the new MagicDraw News Reader. The News Reader is accessible from the MagicDraw **Help** main menu, the **News Reader** command (see Figure 5 on page 31).

MagicDraw News Reader informs about:

- No Magic News. All news regarding company news, product news, new services provided, etc.
- New Versions. Messages about new MagicDraw releases, betas, service packs, plug-ins.
- Resources. Messages about MagicDraw resources updates.

When some news is available, a small envelope icon will be displayed at the right of MagicDraw status bar. Click on this icon to invoke the MagicDraw News Reader (see Figure 6 on page 31).

Reading news in the MagicDraw News Reader

Select the news channel at the left side of the MagicDraw News Reader and then select one of the news in the list. Below the title of the selected news, the description is presented. Unread messages are displayed in bold.

Click the **Open in Browser** link to read description on www.magicdraw.com website.



To refresh news, click the **Refresh** button at the left top corner of the **MagicDraw News Reader** window.



To mark all the selected items as read, click the **Mark item as read** button.

Setting options of the MagicDraw News Reader

To change the checking period, change the **Check for News** property in the **General** section of the **Environment Options** dialog. Property provides the following options:

- Once a day
- Once a week
- Once a month
- Do not check.

1 INTRODUCING MAGICDRAW

MagicDraw Documentation and Support

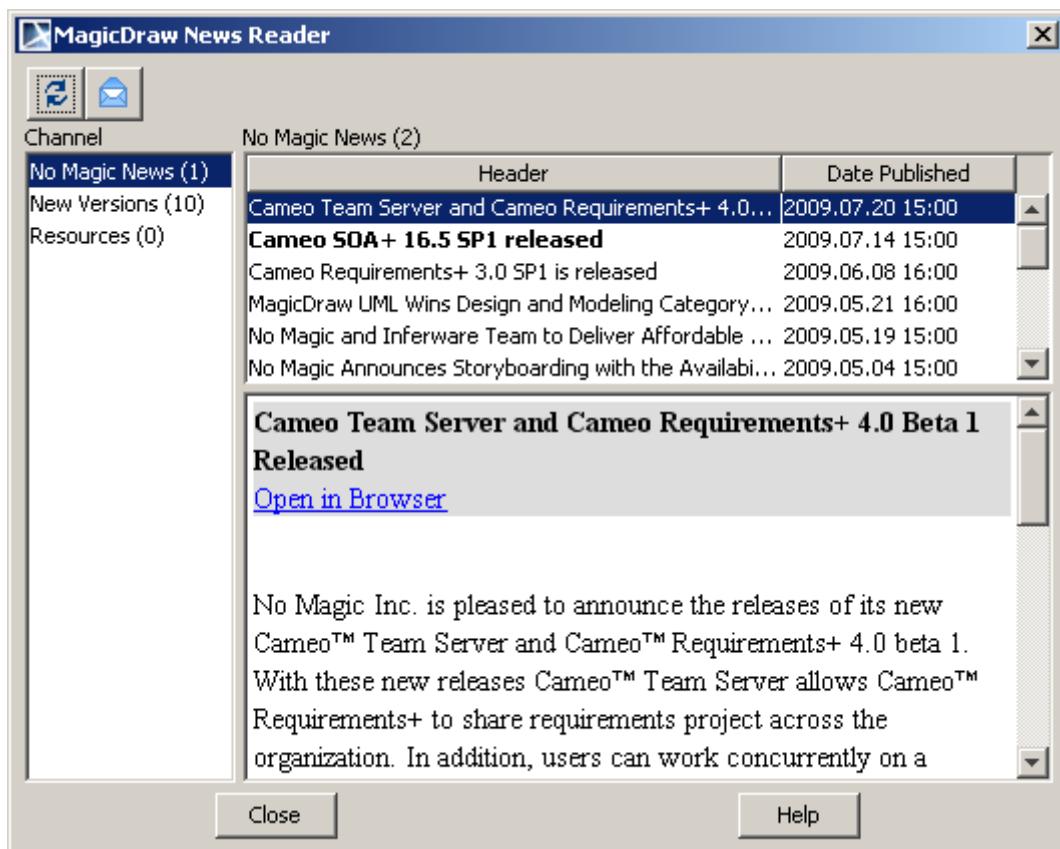


Figure 5 -- MagicDraw News Reader

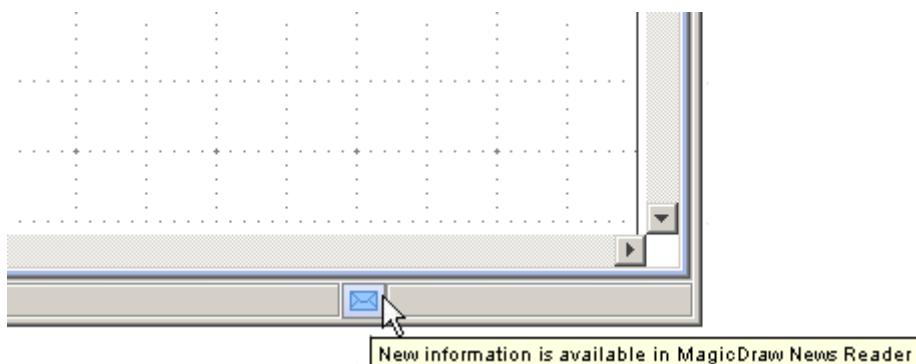


Figure 6 -- Envelope icon informs about news in MagicDraw News Reader

MagicDraw Documentation and Support

MagicDraw provides several kinds of documentation. Choose the way you want to learn.

The main source of information about MagicDraw is www.magicdraw.com and documentation can be downloaded from www.magicdraw.com/documentation.

New and Noteworthy

For information on MagicDraw new features, see New and Noteworthy at <http://www.magicdraw.com/newandnoteworthy>.

Manuals and User Guides

You can find the MagicDraw manual and user guides in “<MagicDraw installation>\manual” folder.

Help

The integrated help within MagicDraw is based on JavaHelp. MagicDraw help provides detailed descriptions of all MagicDraw dialog boxes, commands, and shortcut menus. You will also find a How-to list, as well as main descriptions and examples of all UML model elements.

Hints Associated to User's Actions

MagicDraw now provides hints to help you MagicDraw (Figure 7 on page 32). Hints related to your actions will open and inform you about the functionality that is available in MagicDraw and show you how to perform some operations more effectively.

Hints are displayed in the lower right-hand corner of the MagicDraw application. Figure below shows an example of a hint.

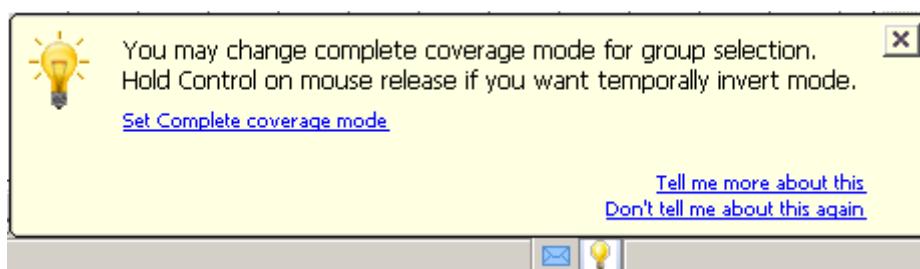


Figure 7 -- Hints Associated to User's Actions

Hint is displayed for a 10 seconds or for a period while mouse is rolled on hint. Small icon will be displayed in a status bar after hint will hide. Clicking on this icon, you can open hint again.

To change hints display mode

Change the **Hints display mode** property in the **Environment Options** dialog box, **General** branch, **Display** group.

The **Hints display mode** property specifies whether hints on MagicDraw functionality related to user actions will be displayed. Custom hint set does not include hints, that are asked not to be shown by user. Select option **Display all hints again** to reset custom hint set to show all hints. Selecting value **Show all hints again** will delete the list of the hints that should not be displayed.

Productivity Tips Displayed in Progress Window

MagicDraw now displays productivity tips in a progress window whenever it performs a long task (Figure 8 on page 33).

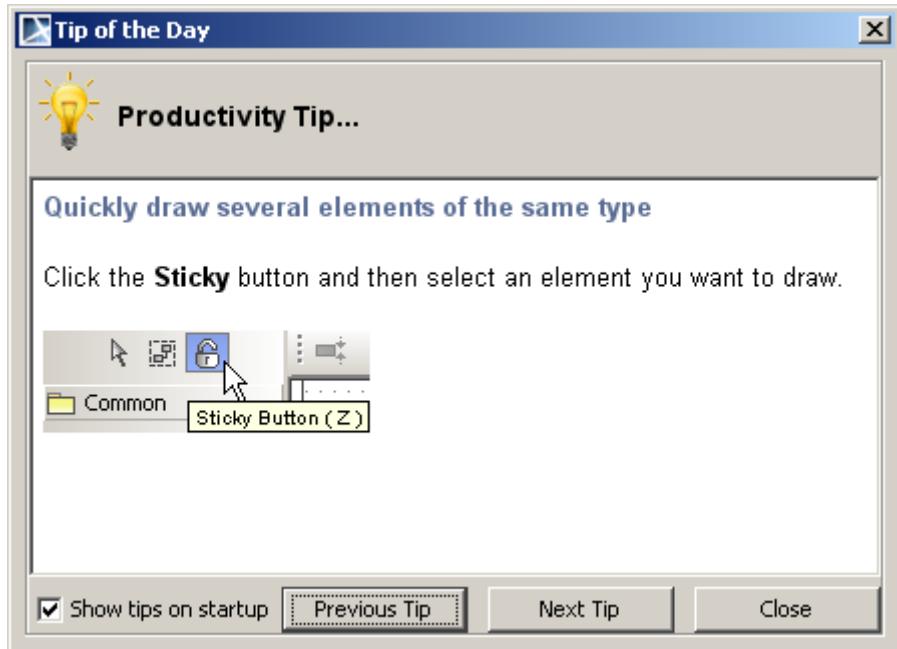


Figure 8 -- Productivity Tip

To show or hide productivity while running long task:

Select or clear the **Show tips while running long tasks** property in the **Environment Options** dialog box, **General** branch, **General** group.

Tutorials

Quick Start tutorials for UML diagrams provide the step-by-step on-screen instructions on how to work with UML diagrams and start modeling with MagicDraw. They also give the links to the associated information in the other MagicDraw user's documentation and to the worldwide tutorials sources as well.

Try the Quick Start tutorials for the new diagram, Class diagram, Use Case diagram, Activity diagram, Sequence diagram, and others.

To open the Quick Start tutorials:

1. From the **File** menu, select **New Project**. The **New Project** dialog will open (Figure 9 on page 34).
2. Select the **Project from Template** icon on the left-hand side of the **New Project** dialog.
3. Select **Guide to UML Diagrams Project > Guide to UML Diagrams Project** in the Select template tree.
4. Click **OK**. The MagicDraw project with its tutorials will open (Figure 10 on page 35).

1 INTRODUCING MAGICDRAW

MagicDraw Documentation and Support

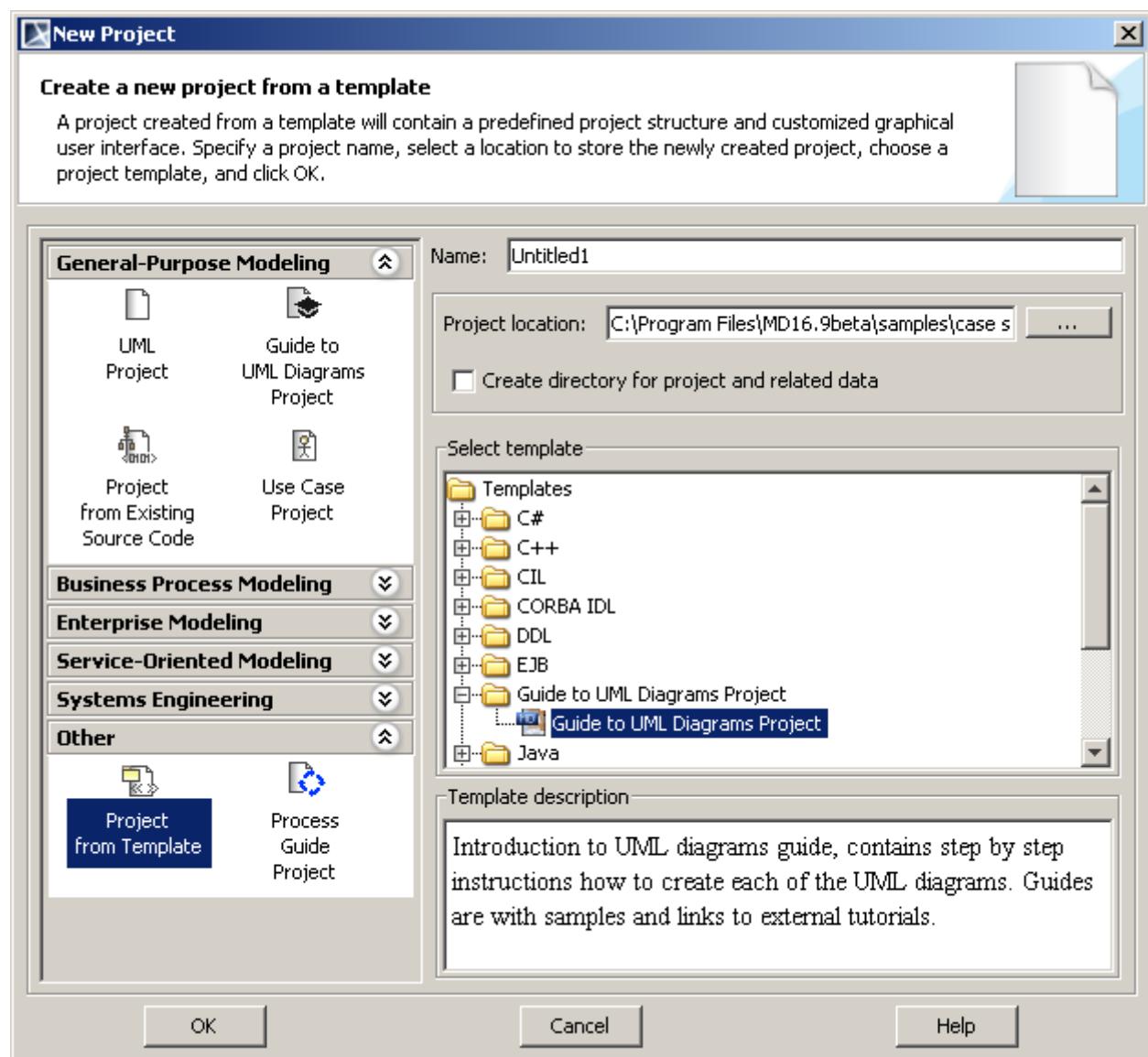


Figure 9 -- The New Project Dialog

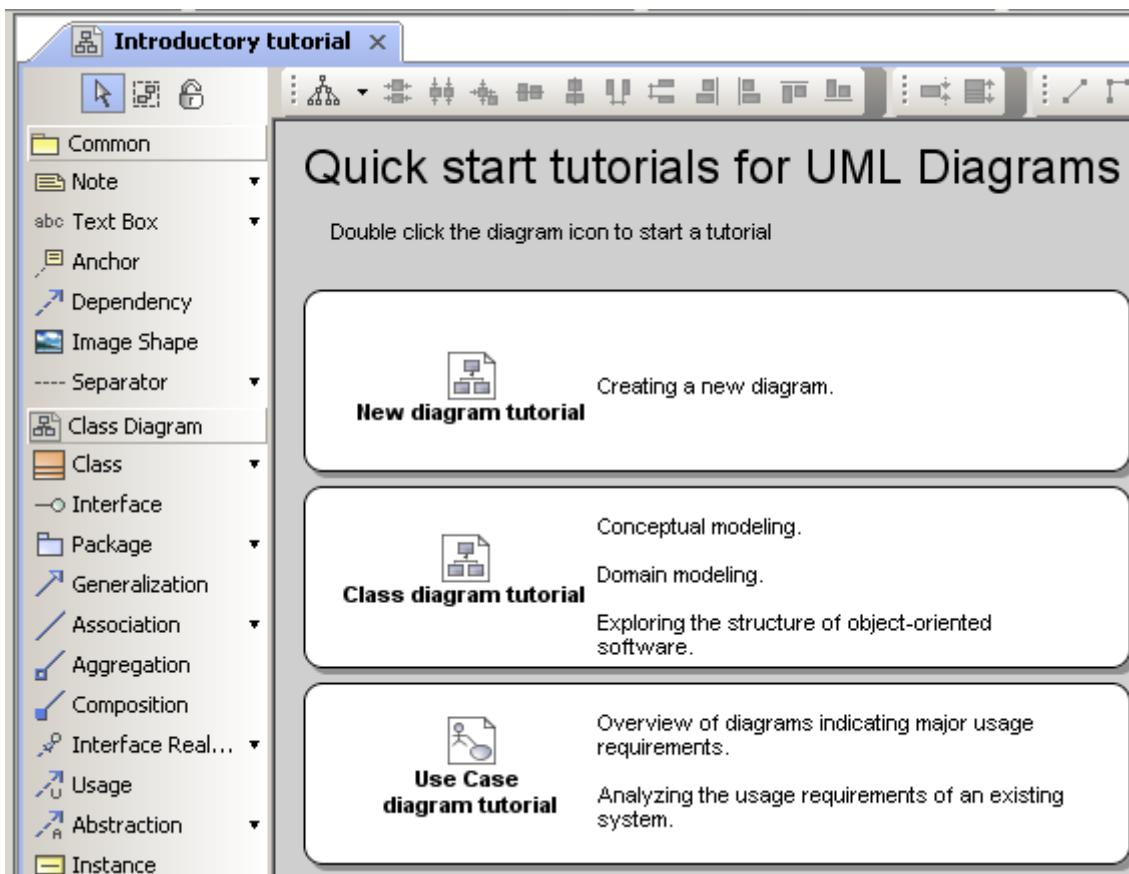


Figure 10 -- Quick Start Tutorials for UML Diagrams

Other Documentation

- The [readme.html](#) file is located in the main MagicDraw installation directory. Readme documents are also available for MagicDraw integrations.
- [Viewlets](#). View online demos introducing MagicDraw and particular functionalities. You may find online demos at www.magicdraw.com/viewlets.
- [Samples](#). In the MagicDraw installation directory (samples directory), you will find the samples of MagicDraw projects.

Support

FAQ

Before calling or registering an issue, please have a look in our [FAQ section](#). It is constantly updated and may contain an answer to your question.

Forum

Discuss and get answers about No Magic products in [No Magic Community Forum](#). With respect to the growing interest in No Magic products, we have expanded the capabilities that had been provided by our news-groups and have moved the content to the online Forum.

You are welcome to post your comments and questions here.

Note: If you are interested in old discussions from the newsgroups, you can find them in the Forum as well as in the newsgroups in a read-only mode.

- If you are a new user, ask questions and get started on learning about UML and MagicDraw by the NEW USER section.
- Discuss issues related to technologies and processes in the STANDARDS/METHODOLOGIES section.
- Share your professional experience in terms of using MagicDraw, including installation and running, MagicDraw OpenAPI, scripting, integrations with other 3rd party tools, etc. in the MAGICDRAW section.
- Find new product versions, updates, and major events in the ANNOUNCEMENT section.
- Post your suggestions for improvements and new features that you would like to see added to the Forum in the OTHER section, the Suggestions subsection.

Knowledge Base. New issue registration

- Knowledge base and New issues registration - <https://support.nomagic.com>.
- sales@magicdraw.com - e-mail address for questions regarding academic or site license discounts and quotes.
- contact@magicdraw.com - e-mail address for the other contacts.

The support system <https://support.nomagic.com> provides:

- Submitted issues status tracking.
- Ability to submit private and public questions, suggestions, improvements, and problems.
- Ability to search through the existing public issues, view status of your issues, provide your vote for suggestions.

We provide free professional support for:

- Registered users with a valid No Magic Software Assurance contract (SA). SA provides you with technical support, software upgrades and maintenance releases at no additional cost for a contract period.
- Pre-sales users during the evaluation period.
- New customers for 30 days.
- Inquiries about registration, licensing and product updates.

Customer support levels

A customer support level determines customer access rights and the response time for support requests. To find out yours customer support level, please, visit the knowledgebase - <https://support.nomagic.com>.

| Support level | Applies for | Ensured support |
|-----------------|---|--|
| Limited support | Customer with no Software Assurance (SA) | Self-help: FAQ, Knowledge base, documentation, online demos, No Magic Community Forum. |
| Full support | <ul style="list-style-type: none">• Customer with SA• Inquiries about registration, licensing, and product updates.• With valid evaluation key.• 30 days after purchase. | Response within business hours, with 24 - 48 hours response time. |
| Premium support | VIP customer* (marked as VIP) | Response within business hours, with 24 hours response time. |

* VIP customer usually is our partner.

Note: During the support period covered by our SA, you can also report any software problems or errors. If it is determined that a reported reproducible issue in the software actually exists, and this significantly impacts the usability of the software, No Magic agrees to make reasonable efforts to provide a usable workaround solution or to correct the issue in an upcoming maintenance release or update.

End-of-life policy

Public service packs are primarily released for the newest version. As we understand that switching from one version to another can take some time, in case of significant issues we provide service packs (or private patches) for up to one year old versions when applicable.

We support and provide patches to older versions only when you own Software Assurance (SA) contracts https://secure.nomagic.com/software_assurance only on your purchased products, so please be sure you are covered.

We always strongly recommend using the newest version as it contains all newest fixes, usability features, new capabilities, and support for standards and technologies.

Reporting issues directly from MagicDraw

We highly recommend to submit report any problem, suggest improvements, or ask about new features directly from MagicDraw. Report will create an issue in to our Online Customer Support System <https://support.nomagic.com>.

These reports help us address issues in a more timely manner, as well as speeding up maintenance releases that are free of known defects.

To report an issue directly from MagicDraw

1. On the help menu, click **Report an Issue**.
2. Fill in the first and last names together with your email address.

NOTE If you supplied the email address used when registering at www.magicdraw.com, you will be able to track the status of your submitted issue in our Online Customer Support System located at <https://support.nomagic.com>. If you used another email address, you will only receive email notifications of status changes.

3. Choose the product, issue type, and component.
4. Describe your issue by providing as much information as possible.

NOTE Please note that we provide a professional support for registered MagicDraw users with the valid No Magic Software Assurance Contract (SA). SA provides you with technical support together with the major software updates and maintenance releases at no cost throughout the contract period. Issues are normally handled within one or two business days during regular business hours.

5. In the **Attachments** tab, select files you would like to send together with your issue report:
 - **Attach log file** - the log file will be sent together with your issue report.
 - **Attach project file** - the opened and last saved project file will be sent together with your issue report.
 - **Attach used modules** - modules that have been used in the opened project file will be sent together with your issue report.

1 INTRODUCING MAGICDRAW

MagicDraw Documentation and Support

- **Attach diagrams images** - choose diagrams, whose images you would like to send together with your issue report. Also specify the images format.

6. If you are registered user, track your issue at <https://support.nomagic.com>.

NOTE If you are already registered personal information will be filled into the Report an Issue dialog.

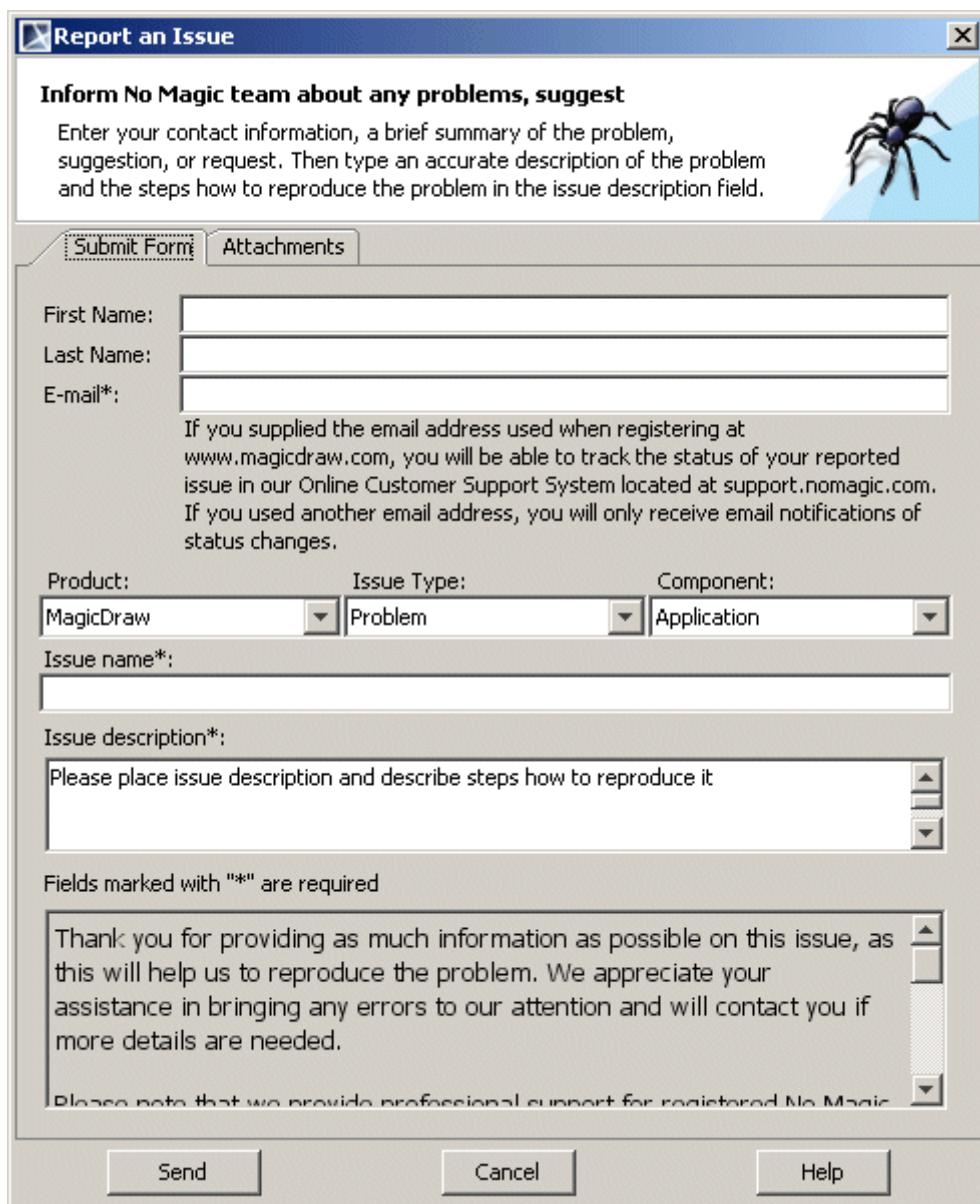


Figure 11 -- Report an Issue dialog box, Submit Form tab

To report an issue when application is unresponsive

If MagicDraw becomes unresponsive, a separately executable tool is provided for analyzing the status of the process to aid in bug submission. In these situations, manually start the submitbug.exe file (located in the <MagicDraw installation directory>\bin folder) and follow directions. After submitbug.exe is started, the **Report an Issue** dialog box opens. For more information about the **Report an Issue** dialog box, see "To report an issue directly from MagicDraw" on page 37.

View and submit internal errors

If an error occurs, an error message will appear at the bottom of the MagicDraw window.

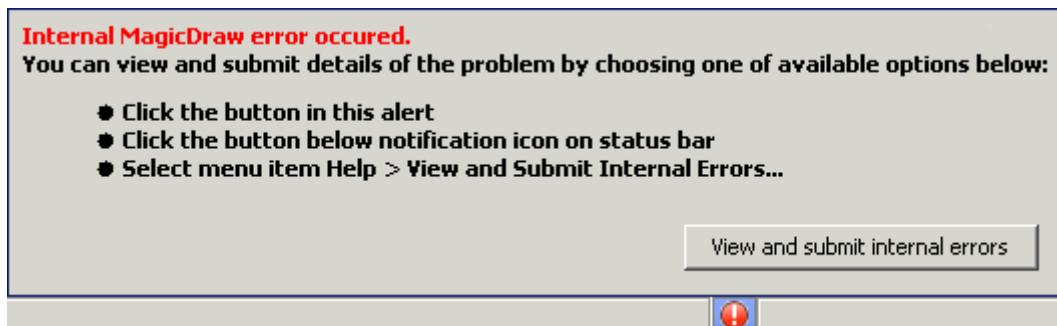


Figure 12 -- Message, about MagicDraw internal error.

To view internal errors

To view internal errors you have to open the **Unhandled Errors** dialog box. There are three methods for to open the **Unhandled Errors** dialog.

- Click the **View and submit internal errors** button in the message window.
- From the **Help** main menu, choose the **View and Submit internal errors** command.
- Click the notification icon  on the status bar.

Note: The **View and Submit internal errors** command is only active in the **Help** menu and the red button at the bottom of the status bar only exists if the **Submit errors** dialog box contains errors.

To submit an error

1. Open the **Unhandled Errors** dialog box.
2. Click the **Submit** button in the **Unhandled Errors** dialog box. The **Submit Error** dialog box appears.
3. Fill in **Your name**, **Your e-mail** and **Bug description** fields and click the **Send** button. The error will be sent to the MagicDraw support team.

2 GETTING STARTED

In the Getting Started chapter, you will find information about how to install, update, register, and configure MagicDraw.

1. "System Requirements", on page 40
2. "Installation Procedure", on page 41
3. "Licensing Information" on page 43
4. "MagicDraw Configuration", on page 44
5. "Unlocking MagicDraw", on page 46
6. "Activating the commercial license after the purchase (v16.9 or later)" on page 47
7. "User Registration", on page 56
8. "Updating" on page 62

System Requirements

To run MagicDraw UML, your system must meet the following requirements:

| Resource type | Minimum | Recommended |
|-----------------------------------|---|--|
| Memory | 1 GB | 2 GB RAM recommended; more memory generally improves responsiveness. 4 GB RAM is recommended for MicrosoftTM Windows Vista and for very large projects |
| Disk space | 500 MB | 500 MB or more |
| Video mode | 800*600 @ 64k Colors | 1280*1024 @ 64k Colors |
| Operating system | All that have Java compatible JVM 1.6.0: Windows 95/98/NT/2000/XP/Vista/7, Linux, Mac OS X (most testing procedures and debugging were performed on these operating systems) | |
| HTML browser | Any | Mozilla Firefox or IE. Safari for Mac OS X. |
| Java Virtual Machine (JVM) | JDK 1.6.0 | Sun's JDK 1.6.0_22 |

Java Virtual Machine (JVM)

You may have the JVM installed on your computer or install JVM together with MagicDraw specifically for the MagicDraw application. JVM is an application that provides the interpretation of the Java bytecode Java class files. Different operating systems may have different JVM implementations, therefore some bugs may be specific to the operating system or JVM.

MagicDraw is a stable environment, if it is configured properly and if a stable JVM is used. **USE THE RECOMMENDED JVM TO AVOID MOST PROBLEMS.** All recommendations are written below. Because MagicDraw is a Java application, most of the stability and performance depends on the JVM implementation. Refer to the JVM specification and problems description, if you have problems on a specific OS.

JDK 1.1.x, 1.2.x, 1.3.x, 1.4.x, and 1.5.x are not supported.

2 GETTING STARTED

Installation Procedure

You can review your system and the JVM information in the **About** dialog box, **Environment** tab (from the **Help** menu, select **About**). The JDK version can also be encountered from the command line by typing:

```
java -version
```

To change Java version

All platforms:

Change line in file mduml.properties (this file is in <MagicDraw installation directory>/bin folder):

JAVA_HOME = <path to new JDK>

NOTE

Integrated MagicDraw runs on the JVM specified by the IDE. In order to change the JVM, you need to modify the startup properties for the IDE that MagicDraw integrates with. If you are running MagicDraw integrated with IDE, read the appropriate readme.html for specific integration. This file can be found in <MAGICDRAW installation directory>/integrations/<IDE directory>.

To redirect output to the console instead of a log file

By default all MagicDraw output info goes into a file. The output can be redirected to the console instead of a log file.

All platforms:

Pass application argument (not java property) -verbose.

Add argument into the JAVA_ARGS line in the mduml.properties file (this file is in <MagicDraw installation directory>/bin folder):

APP_ARGS =-verbose

Operating System- dependent issues

Because MagicDraw is 100% Pure Java, it is platform independent and runs on a variety of operating systems. However, for Java applications to interact properly with the operating system, Java Virtual Machine (JVM) software is required. JVM software varies depending on the operating system and MagicDraw may perform inadequately with the wrong one.

The performance of Java applications depends on the performance of the Java Virtual Machine. A badly designed Machine may degrade the performance of MagicDraw. It could even cause MagicDraw to fail or crash. To avoid these problems, we recommend that you use the following Virtual Machines:

- Sun (JDK standard) for Linux. JDK 1.6.0_22 is recommended.
- Sun (JDK standard) for Windows (95/98, 2000, 2003, NT, Vista, 7). JDK 1.6.0_22 for both 32-bit and 64-bit Windows OS is recommended.
- Sun (JDK standard) for Mac OS X Leopard and Mac OS X Snow Leopard. JDK 1.6.0_22 for both 32-bit and 64-bit Mac OS X is recommended.

Installation Procedure

First, obtain the MagicDraw installation files. You can download the latest version from the MagicDraw homepage (<http://www.magicdraw.com/>). Because MagicDraw is a Java application, you will need more than the installation files to run the tool successfully. You can also install JVM together with MagicDraw or you may

2 GETTING STARTED

Installation Procedure

already have it installed. Information about the latest Java ports is available at http://www.magicdraw.com/jvm_list.htm.

NOTE If an installation is for Windows, and has a JVM you do not need anything else.

Using the installer

If you use an operating system other than Windows, Unix, or Mac OS X, we recommend to run MagicDraw from the no-install package (see section "[Using no-install package](#)" on page 42).

Windows 2000/2003/NT/XP/Vista/7

1. Download the installer:

- *MD_UML_<version number>_win.exe* for 32-bit Windows (or *MD_UML_<version number>_PE_win.exe* in case of downloading Personal edition)
- *MD_UML_<version number>_win64.exe* for 64-bit Windows (or *MD_UML_<version number>_PE_win64.exe* in case of downloading Personal edition)

2. Double-click the installer.

The Setup Wizard automatically adds MagicDraw UML shortcuts to the start menu and the desktop. You may also execute the shortcuts from the installation directory.

NOTE MagicDraw runs on Windows Vista as of version 12.5.

Unix

1. Download *MD_UML_<version number>_unix.sh* (or *MD_UML_<version number>_PE_unix.sh* in case of downloading Personal edition).

2. Using the command-line prompt go to the directory wherein you have downloaded the installer.

3. Type the command:

```
sh ./MD_UML_<version number>_unix.sh
```

IMPORTANT! Be sure you have JVM installed.

MAC OS X

1. Download *MD_UML_<version number>_mac.dmg* (or *MD_UML_<version number>_PE_mac.dmg* in case of downloading Personal edition).

2. Double-click the installer.

3. Drag the MagicDraw UML folder to the Applications or some other folder.

IMPORTANT! Be sure you have JVM installed.

Using no-install package

Download the no-install package to run MagicDraw on any operating system.

Windows 95/ 98/ NT/ 2000/ XP/ Vista*/ 7

1. Download *MD_UML_<version number>_no_install.zip* (or *MD_UML_<version number>_PE_no_install.zip* in case of downloading Personal edition).
2. Extract the file.
3. Go to the bin directory and run:
 - *mduml.exe* to start MagicDraw on 32-bit Windows operating system
 - *mduml64.exe* to start MagicDraw on 64-bit Windows operating system

NOTE MagicDraw runs on Windows Vista as of version 12.5.

Unix

1. Download *MD_UML_<version number>_no_install.zip* (or *MD_UML_<version number>_PE_no_install.zip* in case of downloading Personal edition).
2. Extract the file.
3. Using the command-line prompt go to the bin directory and type the command:

```
sh ./mduml
```

IMPORTANT! Be sure you have JVM installed.

Mac OS X

1. Download *MD_UML_<version number>_no_install_mac.zip* (or *MD_UML_170_PE_no_install_mac.zip* in case of downloading Personal edition).
2. Extract the file.
3. Using the command-line prompt go to the bin directory and type the command:

```
./mduml
```

IMPORTANT! Be sure you have JVM installed.

Licensing Information

Information about installed or needed licenses and the status is presented in the MagicDraw **About** screen with the ability to remove unused licenses (see Figure 13 on page 44).

The following information about installed or needed licenses are available:

| Information item | Description |
|--------------------------|--|
| User ID | User ID is displayed in the About dialog box, Licensing tab. Please refer to user ID when contacting support or sales. |
| Not installed indication | If license is not installed, "Not installed" text is displayed after the license name. |

| Information item | Description |
|------------------------|---|
| Not started indication | If licenses is installed but not started "Not started" text is displayed after license name and reason is be given. Possible reasons: <ul style="list-style-type: none"> • Required resource is not installed. • Plugin startup failed. • Other. |

Removing Unused Licenses

Press button **Remove Unused Licenses** in the **About** screen, **Licensing** tab to remove licenses of not installed plugins.

NOTE If you have any questions or issues, please report them using the **Report an Issue** dialog. For more information, see “View and submit internal errors” on page 39.



Figure 13 -- The MagicDraw About window, Licensing tab

MagicDraw Configuration

MagicDraw configuration files location

By default MagicDraw configuration and auxiliary files are stored in the user home directory - <User home directory>/.magicdraw/<version>.

2 GETTING STARTED

MagicDraw Configuration

You may also save configuration files:

- In the MagicDraw installation directory.
- In your chosen directory.

You may check the exact path to the configuration files in the MagicDraw **About** screen, **Environment** tab, **Configuration files** line (invoke the About screen from the **Help** main menu, **About MagicDraw** command).

To store MagicDraw configuration files in the MagicDraw installation directory

Add argument to JAVA_ARGS line in the file mduml.properties file (this file is in <MagicDraw installation directory>/bin):

```
JAVA_ARGS=-DLOCALCONFIG=false
```

To store MagicDraw configuration files to your chosen location

You may define custom path for configuration files in two ways:

- In the mduml.properties file, JAVA_ARGS line add the property
`-Dlocalconfig.location=<custom path>`
- Define custom path in a newly created file:
 - 1 In the <User home directory>/.magicdraw/<version> folder, create a file named *magicdrawredirect*.
 - 2 In the created file, type the absolute path where the MagicDraw configuration and auxiliary files will be saved, for example: *C:\<directory name>*.

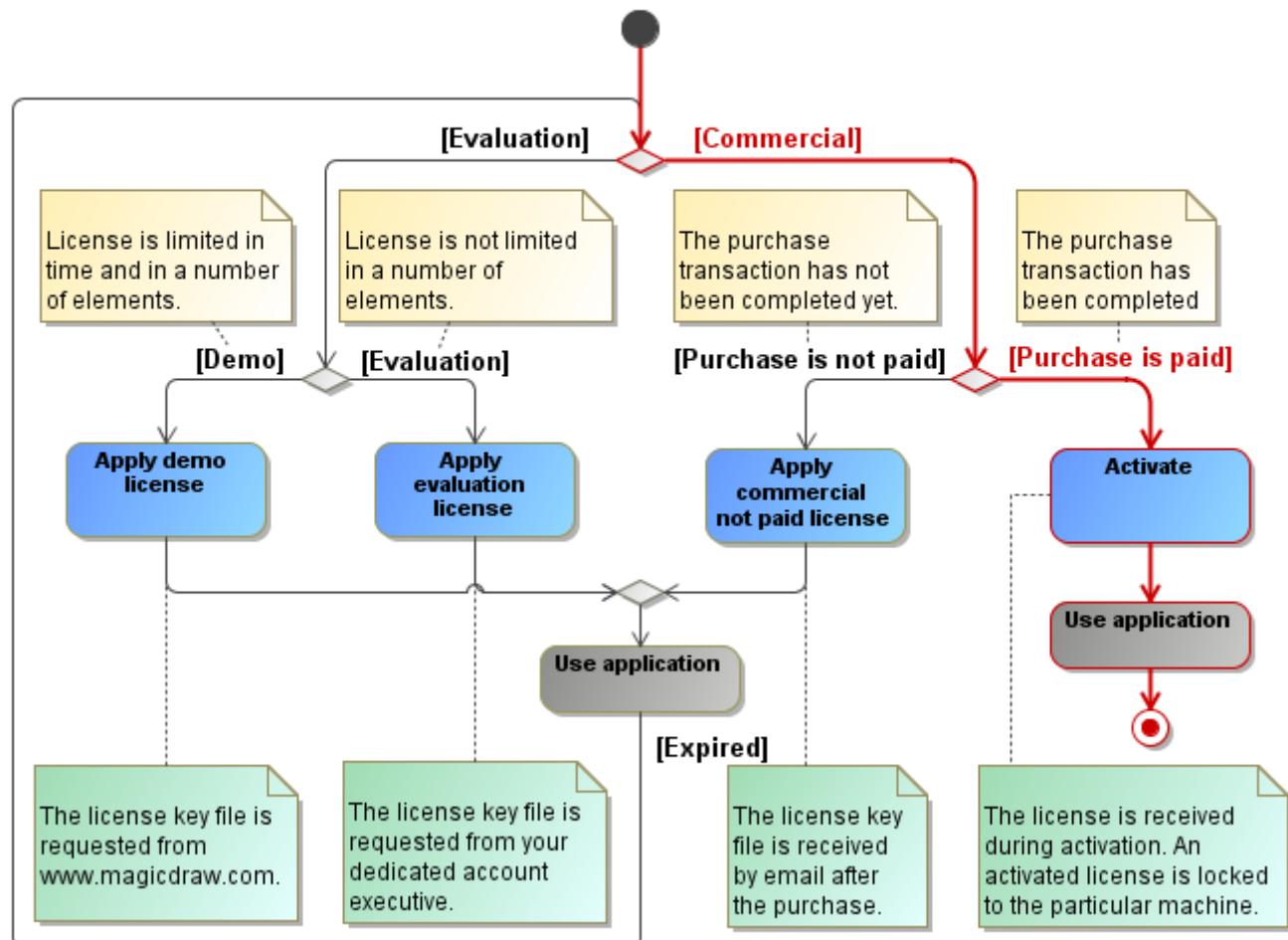
NOTE

If MagicDraw is configured to store files in the MagicDraw installation directory (see the topic “To store MagicDraw configuration files in the MagicDraw installation directory” above), files will not be stored to your chosen location.

Unlocking MagicDraw

Unlock MagicDraw with evaluation, demo, or commercial license key file.

How to get the needed license?



To unlock MagicDraw:

1. After downloading MagicDraw v16.9 or later, you will get license key file to your mailbox.
2. Save this key file into the local file system before starting MagicDraw.
3. When starting MagicDraw for the first time, you will be asked to choose the license key file. Click the **Select Licenses Key Files** button and select the file from the directory you have saved the file.
4. System will automatically start MagicDraw and you will be able to use it.

2 GETTING STARTED

Activating the commercial license after the purchase (v16.9 or later)

Activating the commercial license after the purchase (v16.9 or later)

As of version 16.9, in order to improve software protection and to prove our customers' investments, the solid product licensing is introduced. Licenses are locked to the machine. You will be able to monitor and control, on what machines the commercial licenses are used.

After updating to v16.9, you will be requested to activate and receive the commercial licenses dedicated for the particular machine.

Flexera Software FLEXnet licensing system, the industry leader helping to manage and secure flexible software licensing, is used to manage No Magic products licenses.

How to get the commercial Activated license?

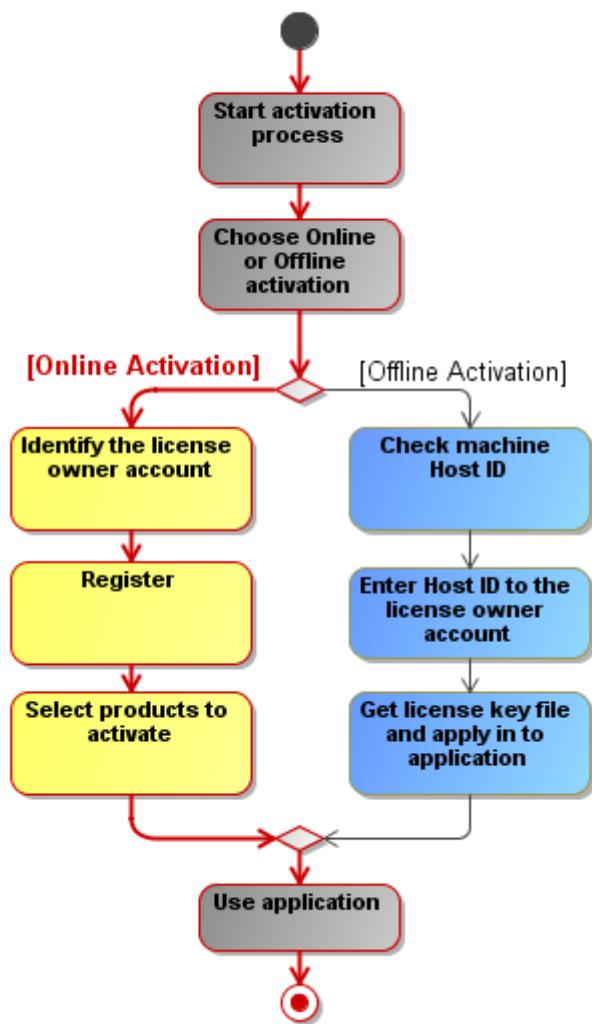


Figure 14 -- Activation process. How to get a commercial license.

All the updated information about the commercial license activation you may also find at <http://www.magicdraw.com/CommercialLicenseActivation>.

In this section you will find the following subsections:

- “Commercial license activation process” on page 48
- “Commercial license types” on page 51

2 GETTING STARTED

Activating the commercial license after the purchase (v16.9 or later)

- “Host ID” on page 51
- “Identification” on page 52
- “Registration” on page 54
- “Deactivation” on page 54

Commercial license activation process

The commercial license activation is required after the purchase transaction has been completed. The activation process allows receiving the commercial license dedicated for the particular machine.

NOTES

- Only commercial activated licenses are locked to the particular machine.
- During the activation process, identification and registration is required. Any key file (evaluation, demo, commercial not activated and other) can be used for the identification of the license owner.
- Commercial not activated keys allow using the application for 7 days according to obtained licenses, before requesting activation. Not activated keys can be used during activation process for identification of the account on which the purchases are.

Online and offline activation is available.

Online activation

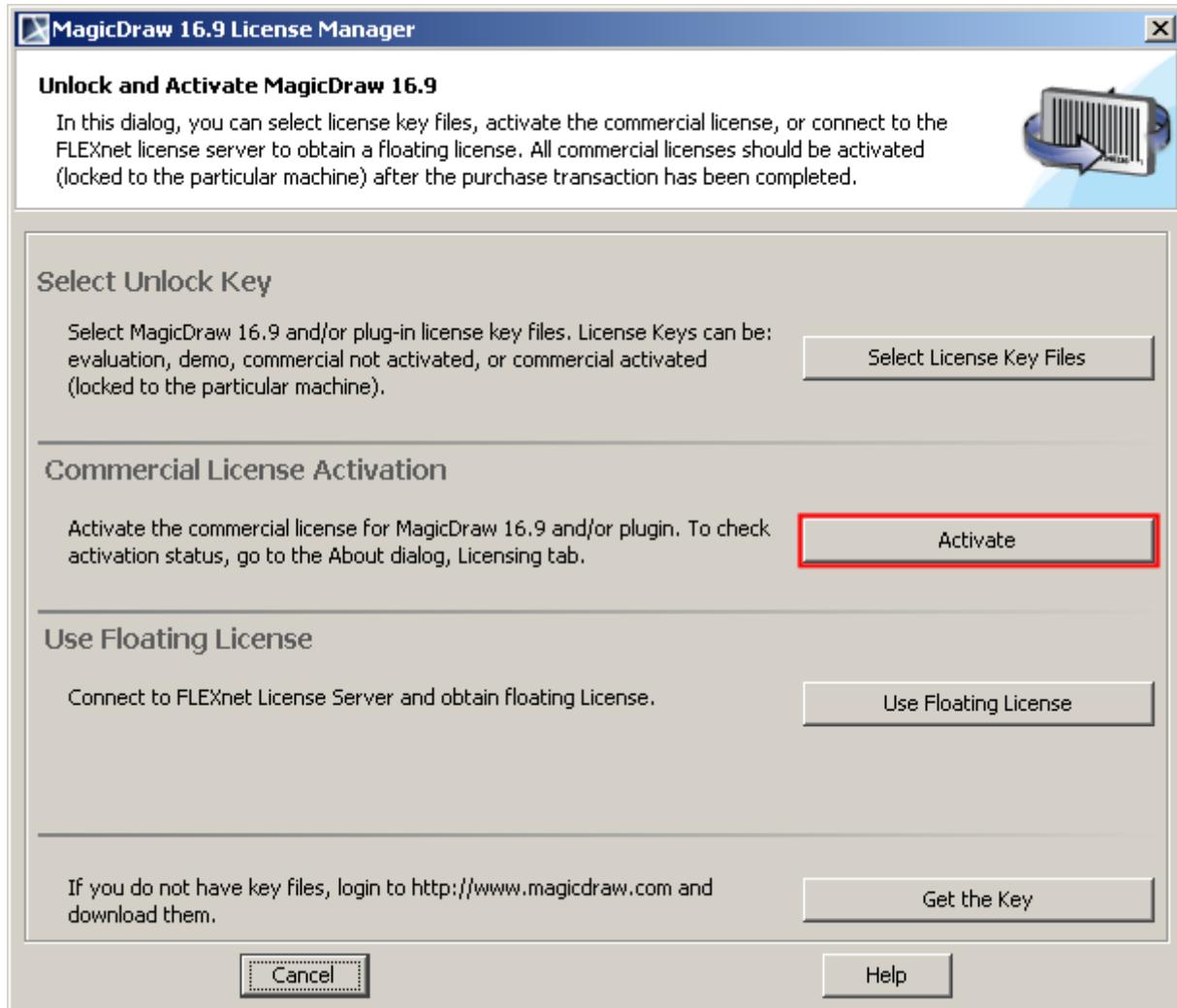
Choose online activation and enable commercial activated (without expiration) licenses automatically.

2 GETTING STARTED

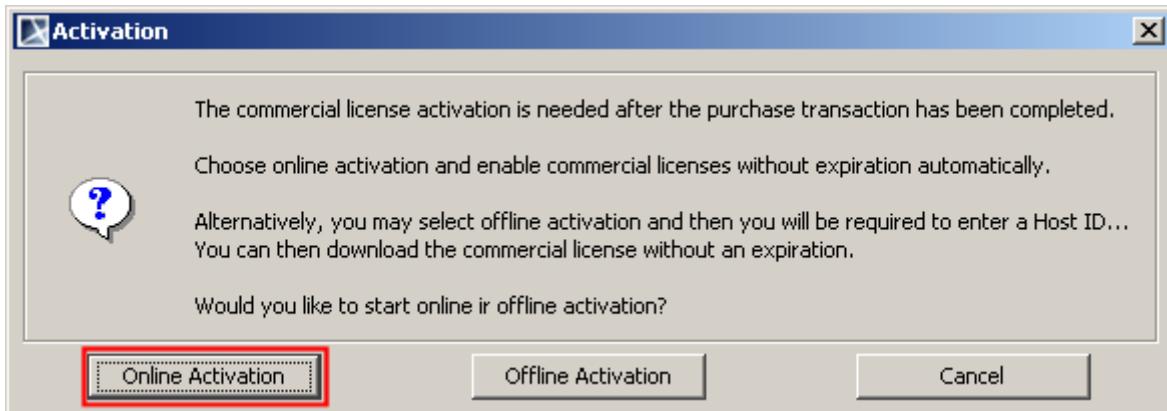
Activating the commercial license after the purchase (v16.9 or later)

To activate the license online

1. Click the **Activate** button in the **License Manager** dialog of the application.



2. Click **Online Activation**.



3. If you will be requested, identify yourself and register.
4. Select licenses you need to activate.

2 GETTING STARTED

Activating the commercial license after the purchase (v16.9 or later)

Offline activation

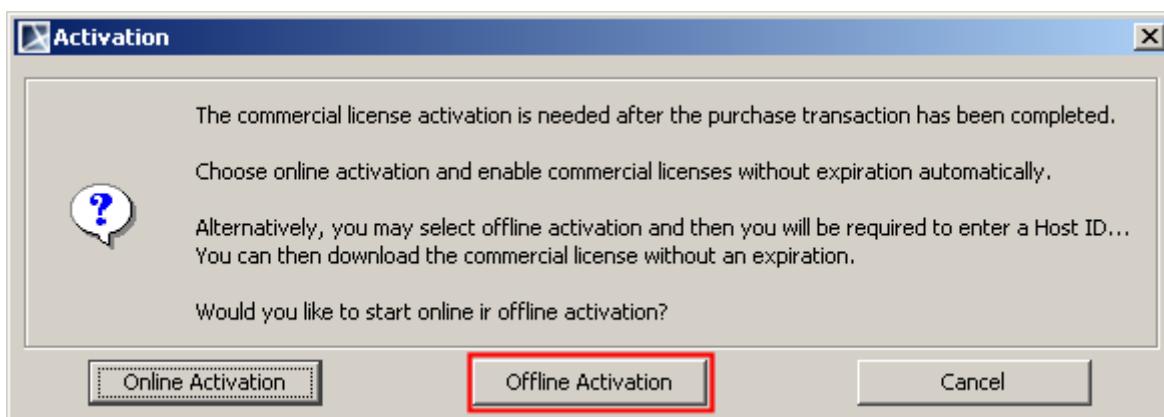
Alternatively, you may select offline activation, get the Host ID, enter it in the license owner account and then download the commercial activated license.

To activate the commercial license offline

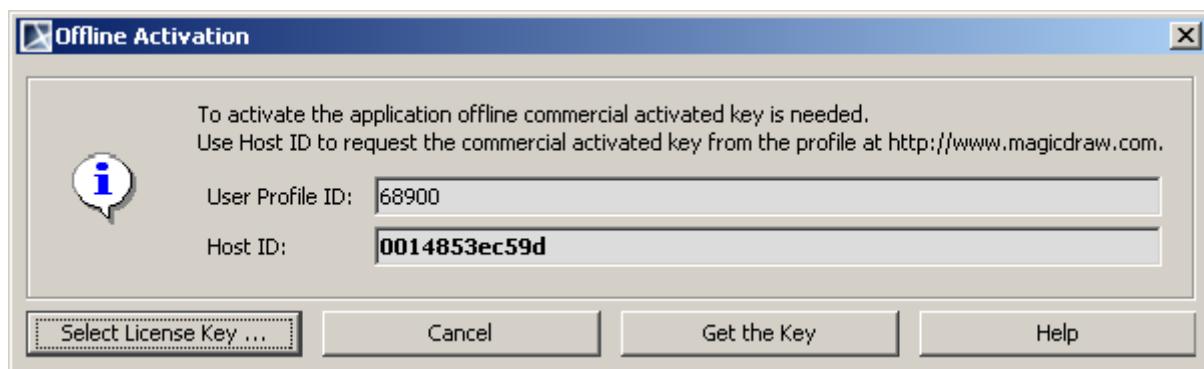
1. Login to the license owner account on www.magicdraw.com.
2. Click the **LicenseActivationManagement** (www.magicdraw.com/LicenseActivationManagement) item in the **Members** menu.
3. Click the **Create New License Activation Record** button and enter user details and Host ID of the machine.
4. Download or send the key to your e-mail.

Determining a Host ID (for offline activation)

1. Click the **Activate** button in the **License Manager** dialog of the application.
2. Click **Offline Activation**.



3. Host ID is shown.



NOTE

Host ID is the Ethernet address of the machine on which application will run.

Related topics

- [Commercial license types](#)
- [Host ID](#)
- [Identification](#)
- [Registration](#)
- [Deactivation](#)

2 GETTING STARTED

Activating the commercial license after the purchase (v16.9 or later)

Commercial license types

DEFINITIONS

- **Commercial not activated license** is used only to determine the account on www.magicdraw.com and allows using the application for 7 days according to obtained licenses, before requesting the activation.

Example: MagicDraw_16_9_Professional_C#_Standalone_Non activated key.txt

- **Commercial activated license** is the license without expiration, locked to the particular machine, and can be deactivated.

Example: MagicDraw_16_9_Professional_C#_Standalone_Activated_key_for_<name>_<surname>.txt

After the purchase transaction has been completed, you will receive the **commercial not activated license**. They are used ONLY to determine the account on www.magicdraw.com from which activated keys should be requested. Any other license, evaluation, demo, or commercial can be used for identification of the license owner account during activation.

Commercial not activated license requires activation. It allows using the application for 7 days according to obtained licenses, before requesting the activation. After the online or offline activation you will get the **commercial activated license**.

After the online activation, commercial activated license will be applied automatically and you will be able to use the application on the particular machine.

During the offline activation you will be required to enter a Host ID <link to Host ID section> in the license owner account at www.magicdraw.com/LicenseActivationManagement. You can then download the commercial activated license and select it from the application.

Related topics

[Commercial license activation process](#)

[Host ID](#)

[Identification](#)

[Registration](#)

[Deactivation](#)

Host ID

A Host ID is the value of a specific system attribute (MAC / Ethernet address) that uniquely identifies the host / machine under which an application is running. A Host ID is used for the locked to particular machine license. Only commercial activated licenses are locked to the particular machine.

For the Host ID determining instructions refer to “Determining a Host ID (for offline activation)” on page 50.

Determining a Host ID using lmhostid

One of the ways for determining Ethernet address (Host ID) is using lmhostid utility. This way can be useful when the list of Host IDs of multiple machines should be discovered, even before installing the applications itself. If you are a system administrator, you can provide end users with commercial activated licenses without requesting licenses from them.

The lmhostid is a command line utility, which prints the Host ID that is required. The lmhostid utility is available as an executable called lmutil. Download it for a particular OS from http://www.globes.com/support/fnp_utilities_download.htm.

2 GETTING STARTED

Activating the commercial license after the purchase (v16.9 or later)

Run lmutil with lmhostid parameter from the command line. Lmhostid displays the default hostid for the current platform.

For exact commands for specific OS refer to: License Administration Guide (Obtaining System Hostids).

NOTE If two or more Ethernet addresses are available on the particular machine, the first one printed by the hostid command should be used.

Related topics

[Commercial license activation process](#)

[Commercial license types](#)

[Identification](#)

[Registration](#)

[Deactivation](#)

Identification

To determine a license owner, the user identification is requested during activation process.

Identification is requested only if you have not added any license key file, have not registered your installation, or have not performed identification.

There are two methods available for the user identification:

- **License file based identification.** Select the license key file to identify the license owner account. Evaluation, demo, or commercial license files can be used for the identification.

2 GETTING STARTED

Activating the commercial license after the purchase (v16.9 or later)

- **Login name based identification.** Use the license owner login name and password to identify on which account on www.magicdraw.com your purchases are.

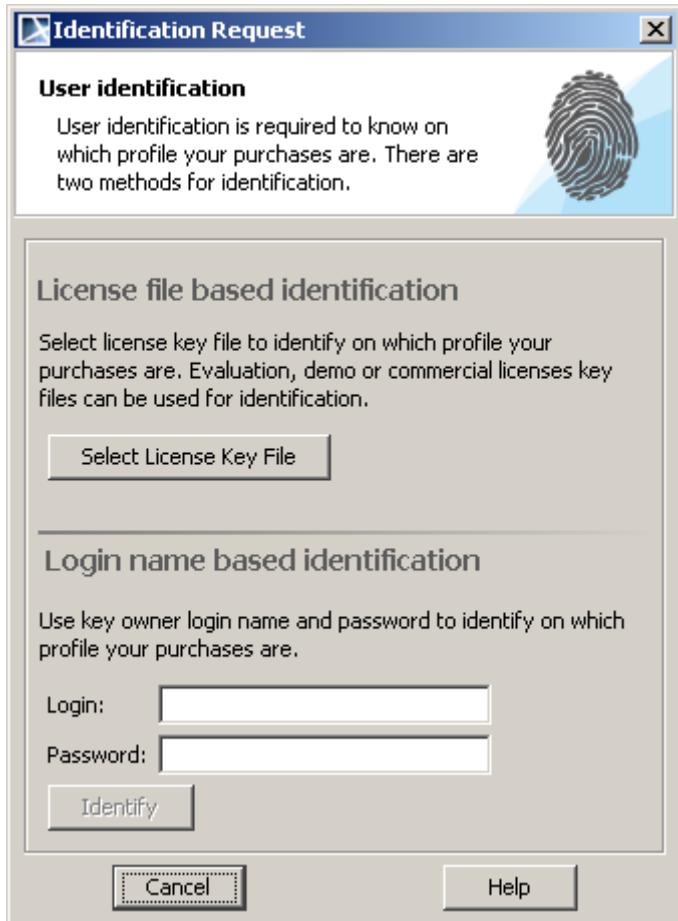


Figure 15 -- **Identification Request** dialog

Once identification is performed, it will not be requested the next time during activation.

NOTE In case you need to change identification records, just apply the new license key file. The last applied license key file is used for the identification.

Related topics

- [Commercial license activation process](#)
- [Commercial license types](#)
- [Host ID](#)
- [Registration](#)
- [Deactivation](#)

2 GETTING STARTED

Activating the commercial license after the purchase (v16.9 or later)

Registration

The user registration is required during activation process in order to obtain the commercial activated license.

The screenshot shows a Windows-style registration dialog box titled "Registration". The main title bar says "Registration". The header contains the instruction "Please identify yourself and register installation" and a small icon of a person holding a briefcase. The left side of the window has a section titled "Are these your details? If not please change:" with fields for First Name (Martin), Last Name (Smith), E-mail (martin.smith@nomagic.com), Phone (+1-214-291-9100), Role (Developer), Industry (Information Technology), Company (No Magic, Inc.), and Country (United States). Below these fields is a note: "Please make sure your email address is correct. A confirmation link will be sent to this address." To the right of the details is a section titled "Registration will enable us to provide you with professional services:" which lists three bullet points: "Customer support for all problems, questions, and suggestions.", "Access to early releases and evaluations.", and "New versions, patches, and updates.". At the bottom of the window are three buttons: "Register", "Cancel", and "Go to Web Page".

Figure 16 -- Registration window

The registration will help us to provide you with:

- Customer support. Issues, questions, and suggestions can be reported and existing ones tracked on <https://support.nomagic.com>.
- Access to early releases and evaluations.
- New versions, patches, and updates.

Related topics

[User Registration](#)

[Commercial license activation process](#)

[Commercial license types](#)

[Host ID](#)

[Identification](#)

[Deactivation](#)

Deactivation

The deactivation is the process, which allows returning commercial activated licenses. You may remove and return all activated (locked to the particular machine), not used commercial licenses by deactivating them.

2 GETTING STARTED

Activating the commercial license after the purchase (v16.9 or later)

To deactivate a license in the application

1. Click the **Deactivate** button in the **License Manager** dialog of the application or uninstall the application.
2. Commercial licenses will be deactivated automatically and returned online.

If the online licenses return is not available, licenses will be deactivated and offline deactivation - licenses return message with License Deactivation ID - will be shown. To update the account information manually, return the used license by entering the License Deactivation ID to the license owner account at www.magicdraw.com/LicenseActivationManagement.

3. The application is deactivated and licenses are returned.

NOTES

- The deactivated license still will be active for 7 days.
- The returned licenses can be activated on the same or another machine.
- Online deactivation from the application side or the deactivation with the License Deactivation ID is treated as the confirmed deactivation case and does not decrees an available rehost limit.

Deactivating from the License Activation Management screen

You can deactivate licenses not only from application, but also from the License Activation Management screen in the license owner account at www.magicdraw.com/LicenseActivationManagement.

NOTE

The deactivation from the License Activation Management screen should be used only if the deactivation from application side is not available: installation has been lost or corrupted.

To deactivate an application from the license owner account:

1. Go to license owner account at www.magicdraw.com/LicenseActivationManagement.
2. Remove all products assigned for the particular installation in the **Activated Products** column.
3. Application will be deactivated on the next start-up. Licenses will be returned.

NOTE

The number of available, not confirmed, deactivations from the License Activation Management screen is 1 for each obtained license per year. The Confirmed deactivation case from the application side will not decrease the rehost limit.

License Deactivation ID

License Deactivation ID is the value of a specific system attribute that uniquely identifies license deactivation/return from the particular host/machine under which application was running. The License Deactivation ID is used for the license offline deactivation.

To determine License Deactivation ID (for offline deactivation):

1. Click the Deactivate button in the License Manager dialog of the application.
2. The license Deactivation ID will be shown if the online deactivation will not be available.

Confirmed deactivation case

The license deactivation, about which confirmation is received in the license owner account at www.magicdraw.com/LicenseActivationManagement is treated as confirmed deactivation case.

The deactivation confirmation can be received either online, during online deactivation, or offline, with the help of the License Deactivation ID, which is shown during the offline deactivation.

2 GETTING STARTED

User Registration

The deactivation from the License Activation Management screen without the License Deactivation ID, is treated as NOT confirmed deactivation and decreases the rehost limit. An exception is, if after such the deactivation, automatic confirmation is received from the application side that the license has been also deactivated on the application side.

Rehost limit

The rehost limit is the number of available, not confirmed, deactivations from the License Activation Management screen. By default such number is 1 for each obtained license per year. The confirmed deactivation case from the application side will not decrease the rehost limit.

If the rehost limit is exceeded you can request extension from your dedicated account executive by filling the reason of the request and clicking the Request Rehost Increase button.

Rehosting - moving the license from one machine to another

You may move of the activated (locked to the particular machine) licenses from one machine to another is called rehosting.

To rehost a license

1. Deactivate the license on the old installation.
2. Install an application on the new machine.
3. Activate the license on the new installation.

Related topics

[Commercial license activation process](#)

[Commercial license types](#)

[Host ID](#)

[Identification](#)

[Registration](#)

User Registration

User Registration allows you to access dedicated resources on MagicDraw Website such as dedicated online support, answers database, new products evaluation, and beta products.

The **Registration** window will open the first time MagicDraw is started (Figure 17 on page 57). You can complete Product Registration at any time by selecting **Register** on the MagicDraw **Help** menu.

NOTES

- No Magic, Inc. respects your privacy. We will only use your personal information for communications and management of your online account, and the products you register with your account.
- Registration for the owners of commercial licenses is mandatory.

2 GETTING STARTED

User Registration

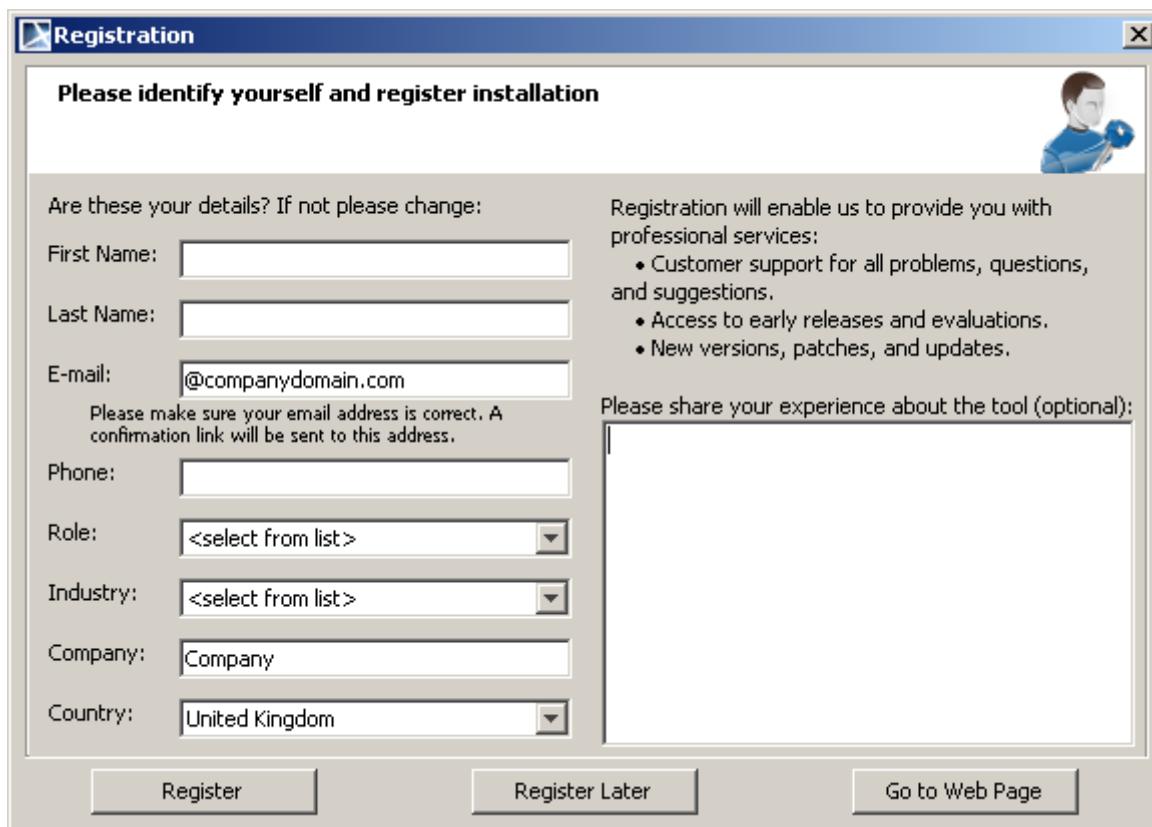


Figure 17 -- The Registration Dialog

Registration Workflow

The registration process is straightforward. After a new key application, you will be requested to register your installation. If you have successfully sent the online registration form, you will receive an e-mail with a link to confirm your registration and the correctness of information furnished through online registration.

Upon confirmation, a dedicated account will be created for you at www.magicdraw.com.

NOTE If you have an existing profile at www.magicdraw.com you may register with the same user information and the same profile information will be used for registration.

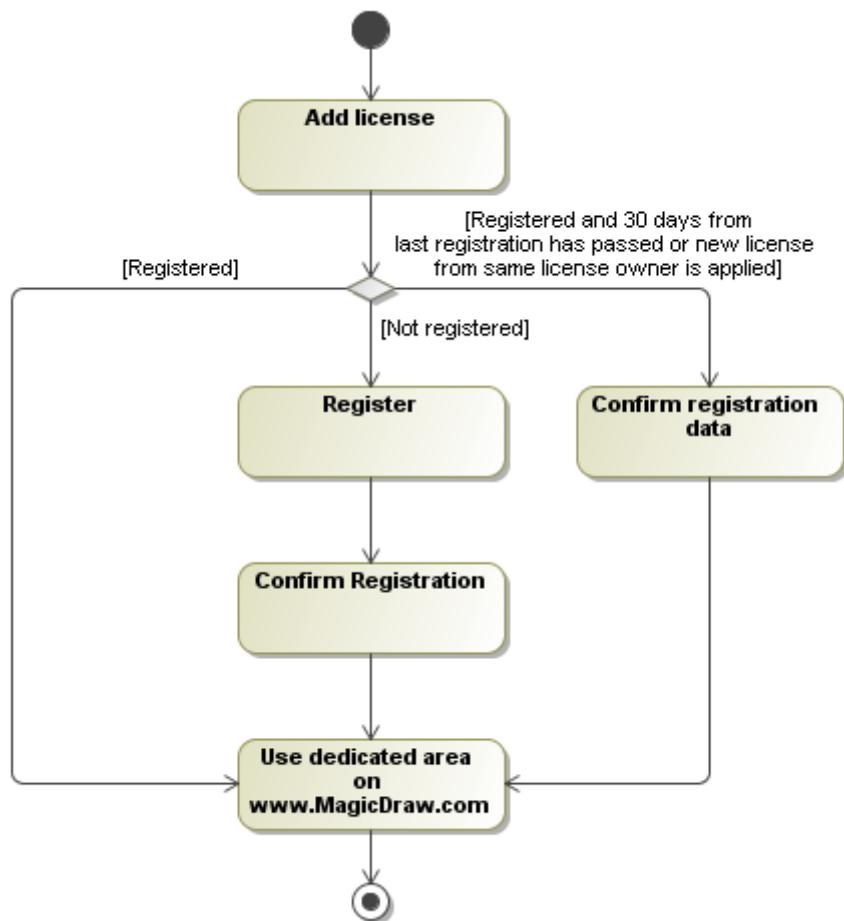


Figure 18 -- Registration Workflow

Adding a License

MagicDraw always checks the registration status at startup after license has been provided, an unlock key has been added, or a Floating server has been selected.

Registering

If you are not a registered user, MagicDraw will open the **Registration** dialog at startup, prompting you to register.

| | |
|-------------|--|
| NOTE | You do not have to complete the online User Registration to use MagicDraw, however, it is recommended to do so to receive the benefits available for a registered user. You can complete Product Registration at any time by clicking Register on the MagicDraw Help menu. |
|-------------|--|

To complete Product Registration:

1. On the MagicDraw main menu, click **Help > Register**. The **Registration** dialog will open (Figure 19 on page 59).
2. Provide the requested information in the **Registration** dialog (some information has been profiled from the key owner profile in order to provide a more usable registration process).
3. Share your experience about the tool (optional).
4. Click **Register** to send data to the server and to receive a confirmation email later on.

2 GETTING STARTED

User Registration

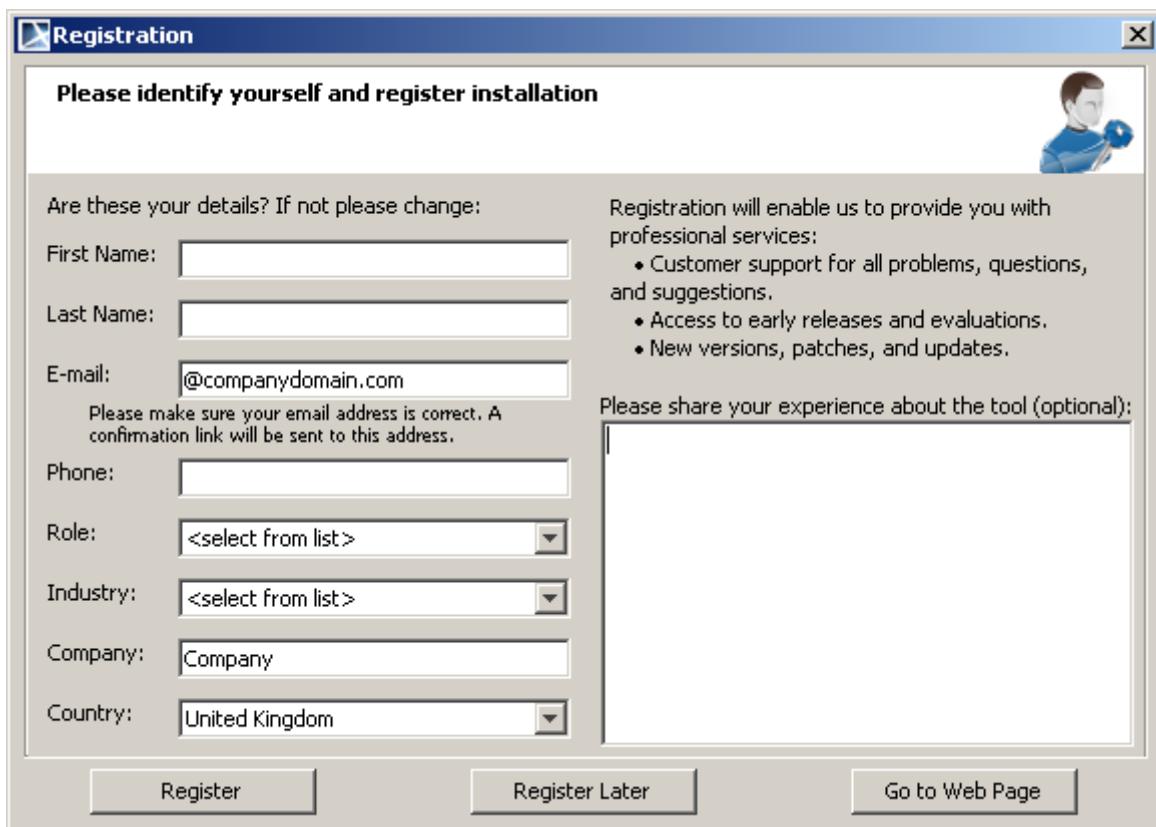


Figure 19 -- The Registration Dialog

NOTES

- You can click the **Confirm Later** button to register at a later time and close the **Registration Confirmation** dialog.
- A message reminding about Registration Confirmation will appear at MagicDraw startup if you do not complete the registration process.
- No Magic, Inc. respects your privacy. We will only use your personal information for communications and management of your online account, and the products you register with your account.

Confirming Your Registration

An e-mail with registration data and a confirmation link will be sent to the e-mail address provided during registration. Click the confirmation link to confirm the registration process and create or navigate (if it has been created) to your dedicated area at www.magicdraw.com.

Logging in to Your Dedicated Area at www.magicdraw.com

Use your login name and password received upon confirming your registration to log in to the user dedicated area at www.magicdraw.com.

Your registration will enable us to provide you with the following professional services:

- Dedicated customer support for all problems, questions, and suggestions.
- Access to early releases and evaluations.
- New versions, patches, and updates.

2 GETTING STARTED

User Registration

Registration Data Confirmation

After a period of 30 days has passed since the date of your registration or a new license from the same licensed owner has been applied, you will be requested to confirm that you are the one using the software installation at MagicDraw startup. A profiled **Registration Confirmation** dialog will open. Check the correctness of data and click the **Confirm** button.

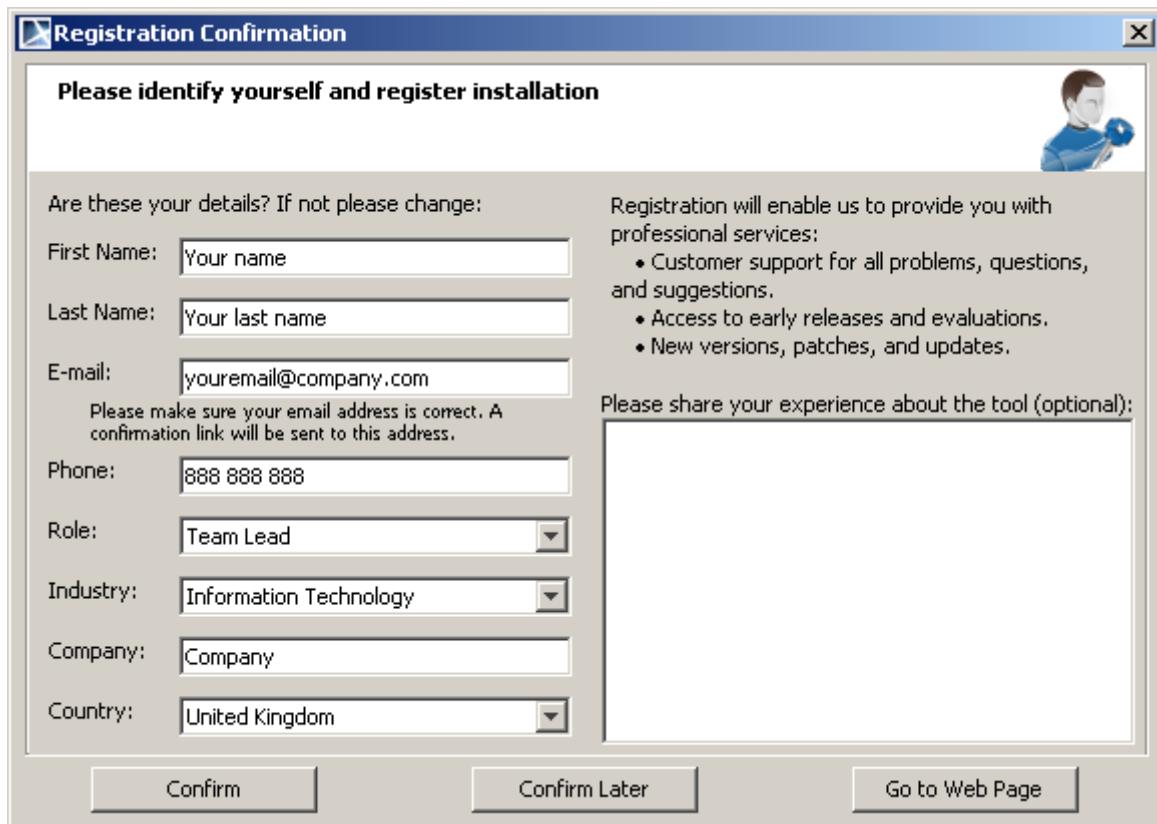


Figure 20 -- Registration Confirmation

NOTE

You will receive no email upon confirming your registration.

If you are registering as a new user, with different profile information, the **Update Existing Profile** question will appear. You can either update your existing profile or identify yourself as a different user using the product installation.



Figure 21 -- Updating Existing Profile

Bug Report

If you are a registered user, your personal information will be provided in the **Report an Issue** dialog. Submit notifications of software errors dialog is available from **Help menu > Report an Issue**.

2 GETTING STARTED

User Registration

If the provided information does not correspond to the information you have provided during registration, once you click the **Send** button to submit the bug, you will be asked to register or update your information. The **Registration** dialog will open and profiled with your personal information from previous registration with changes from the **Report an Issue** dialog or details from the **Report an Issue** dialog if you have not yet registered.

For more information about bug reporting, see “Reporting issues directly from MagicDraw” on page 37.

Troubleshooting

| Issue | Solution |
|--|---|
| Registration is requested on each startup | <p>The Registration Confirmation dialog will open each time MagicDraw starts until you register the product installation.</p> <p>The registration process is straightforward and requires minimum data for you to access dedicated online support, answers database, new products evaluation, and beta products.</p> <p>Clicking the Confirm Later button will close the Registration Confirmation dialog.</p> |
| Registration confirmation is requested at each startup | <p>If you do not click the registration confirmation link, installation will not be confirmed.</p> <p>You may request a new confirmation email from the registration confirmation message.</p> |
| No email with a confirmation link is received | Your spam or virus filter may filter out the e-mail with a confirmation link. If that is the case, you can request a new confirmation e-mail by clicking Help > Register on the MagicDraw main menu. |
| The No connection to registration server message opens | <p>Check your internet connection and try to reconnect.</p> <p>NOTE: User Registration is encouraged, but is not required to use MagicDraw.</p> |
| You are getting can not connect to server message on registration dialog invocation from Help menu -> Registration | <p>Check your internet connection and try to reconnect.</p> <p>NOTE: User Registration is encouraged, but is not required to use MagicDraw.</p> |
| The Registration dialog does not opens at Startup | <ul style="list-style-type: none">• You have already registered and the period of 30 days since the date of your last registration has not been used up.• There are network limitations to check your registration status. <p>Note: User Registration is encouraged, but is not required to use the MagicDraw.</p> |

NOTE

If you encounter problems during the registration process, please contact us registration@nomagic.com

Updating

An automatic updates feature is implemented in MagicDraw. Notification and update of all the patches can be done automatically.

To enable an automatic checking for MagicDraw updates

1. Open the **Environment Options** dialog box, **Update** pane. For a detailed description of this dialog box, see “Update Options Tab” on page 101.
2. Select a period when MagicDraw should check for updates from the **Check for Updates** drop-down list box: **Manually, On startup, Once a day, Once a week, Once a month**. It is recommended to check for updates once a month.

NOTE

You may also manually check for updates: go to the **Help** menu and select the **Check for Updates** command.

Auto-Check for Updates dialog box

The dialog box is opened when automatic checking for MagicDraw updates is enabled on the **Update** pane in the [Environment Options](#) dialog box and the **Show Auto-Checking Confirmation Dialog Box** check box is selected.

| Element Name | Function |
|--------------------------------|--|
| Show this tip next time | If selected, the dialog box is opened each time that MagicDraw should be updated according to the schedule selected on the Update pane in the Environment Options dialog box. |
| Check | Starts checking for MagicDraw updates in the http://www.magicdraw.com page. |
| Cancel | Closes the dialog box without saving changes. |
| Help | Displays MagicDraw Help . |

3 USING MAGICDRAW

In “Using MagicDraw”, you will find information about how to define MagicDraw according to perspective, an introduction to the MagicDraw User Interface, and defining your environment.

- "Customizing and Selecting Perspective", on page 63.
- "Understanding MagicDraw User Interface (UI)", on page 69.
- "Customizing Environment Options", on page 89.
- "Look and Feel: Controlling the Interface", on page 117
- "Assigning Shortcut Keys", on page 118.

Customizing and Selecting Perspective

Launch MagicDraw for the first time and after the application starts, the **MagicDraw Startup** dialog box appears. In this dialog box you may select your work perspective.

Due to the growing number of MagicDraw features, many features may be configured for standard or expert user. MagicDraw can satisfy the needs of different software development process roles. In order to better satisfy user needs, MagicDraw configuration depends on Perspective.

Perspectives allow:

- The selection of a predefined MagicDraw configuration and features according to your software development process role.
- Finding features faster, because there are less of them.
- To choosing a suitable MagicDraw experience mode with a single click.
- Customizing a set of predefined features and configuration based on user needs.

There are five perspectives in MagicDraw:

- **Business Analyst** - this perspective provides features for the Business Analyst. This role is responsible for defining business architecture. Code engineering, transformations, extensions, and other features are hidden.
- **Full Featured** - Perspective provides all features available in MagicDraw and installed plugins.
- **Quick Start** - Quick Start perspective provides basic features dedicated for modelling and not overcrowded interface for quick learning. Code engineering, transformations and other advance features are hidden, however easily reachable in expert mode of this perspective.
- **Software Architect** - this perspective provides features primarily involved in designing and implementing projects. It is a set of roles consisting of Software Architect, Designer, Interface designer, and Database designer. This is the default MagicDraw configuration. All functionalities are available for expert mode.
- **System Analyst** - this perspective provides features primarily dedicated to obtaining requirements and modeling the system. Analysis features are highlighted. Configuration is modeling oriented. Code engineering, transformations and other features are hidden.

3 USING MAGICDRAW

Customizing and Selecting Perspective

To set the perspective for the MagicDraw environment

- Launch MagicDraw for the first time. The **MagicDraw Startup** dialog box will appear with the possibility to switch between perspectives. Select the desired perspective from the list and click the **OK** button.
- From the **Options** menu, choose **Perspectives** and then click **Perspectives**. The **Select Perspective** dialog box opens. Select the desired perspective and click the **Apply** button.
- On the **Perspectives** toolbar, select the desired perspective from the available perspectives list.

NOTE

For more information about the Perspectives toolbar, see “**Perspectives Toolbar**” on page 804.

Customizing MagicDraw Perspectives

Perspective customization allows the grouping of functional MagicDraw features to standard/expert modes. Customization also allows the user to hide unnecessary commands, which makes MagicDraw simpler and faster to use.

To open the **Customize Perspectives** dialog box

- From the **Options** menu, choose **Perspectives** and then **Customize**.
- In the **Perspectives** dialog box, click the **Customize** button.

MagicDraw has six customizable areas in different perspectives. Each of these areas has a set of commands, which can be shown in standard/expert mode, or hidden:

- Main menu;
- Main toolbars;
- Diagram toolbars;
- Diagram modeling elements toolbar;
- Context menu actions;
- Reports.

To customize the selected MagicDraw area in a predefined perspective

1. In the **Customize Perspectives** dialog box, select the perspective and click the **Edit** button near the selected MagicDraw area. The appropriate **Customize** dialog box opens.
2. Expand tree sections and select radio buttons beside items in the tree depending on your choice for **Standard and Expert**, **Expert only** or **Hidden** modes.
3. Click **OK** to save changes, then **OK** in the **Customize Perspectives** dialog box, and then **Apply** in the **Select Perspective** dialog box (if needed).

To switch between Standard/Expert menu, toolbar, or diagram toolbar modes

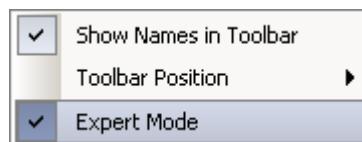
- From the toolbar shortcut menu, select/clear the **Expert Menu Mode** or **Expert Toolbar Mode** check box.



3 USING MAGICDRAW

Customizing and Selecting Perspective

- From the diagram buttons toolbar menu, select/clear **Expert Mode** check box.



MagicDraw Startup dialog box

Launch MagicDraw for the first time. The **MagicDraw Startup** dialog box will appear with the possibility to switch between perspectives.

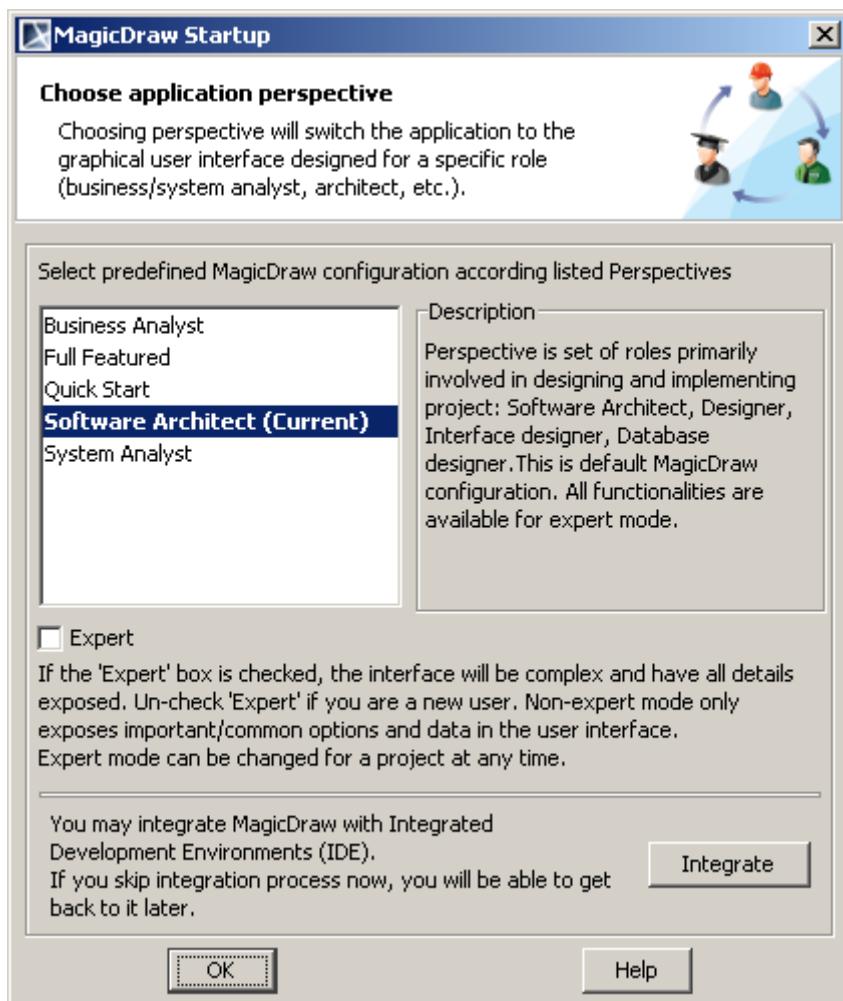


Figure 22 -- MagicDraw Startup dialog box

| Element Name | Function |
|---------------------------|--|
| Business Analyst | Possible perspectives to set, which will load a predefined MagicDraw configuration. |
| Full Featured | |
| Quick Start | |
| Software Architect | |
| System Analyst | |
| Expert | If not selected, a simplified MagicDraw interface with the most popular items and features will be opened and the specification properties will be shown in Standard Mode. |
| Integrate | Opens the Integrations dialog box for quick integration with the selected tool. |

3 USING MAGICDRAW

Customizing and Selecting Perspective

| Element Name | Function |
|--------------|--|
| OK | Loads MagicDraw with the selected perspective. |
| Help | Displays MagicDraw Help . |

Select Perspective dialog box

From the **Options** main menu, choose **Perspectives** and then **Perspectives**.

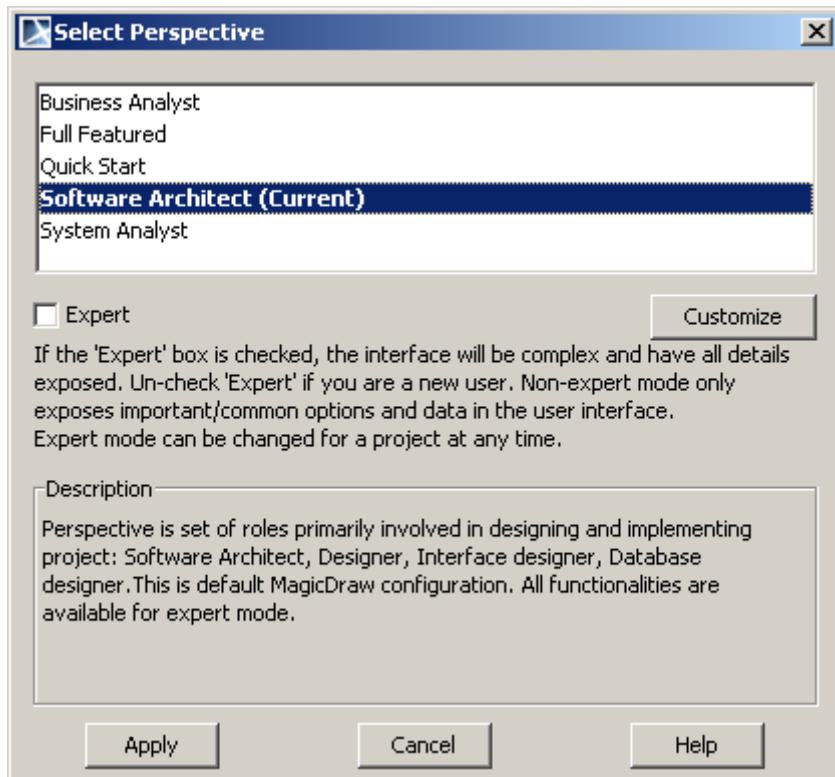


Figure 23 -- Select Perspective dialog box

| Element Name | Function |
|---------------------------|--|
| Business Analyst | List of possible perspectives, which will load a predefined MagicDraw configuration. |
| Full Featured | |
| Quick Start | |
| Software Architect | |
| System Analyst | |
| Customize | Opens the Customize Perspective dialog box. |
| Expert | If not selected, a simplified MagicDraw interface with the most popular items and features will be opened and the specification properties will be shown in Standard Mode. |
| Apply | The selected perspective will be applied for current MagicDraw mode. |
| Cancel | Closes the dialog box without saving changes. |
| Help | Displays MagicDraw Help . |

Customize Perspectives dialog box

From the **Options** main menu, choose **Perspectives** and then **Customize**.

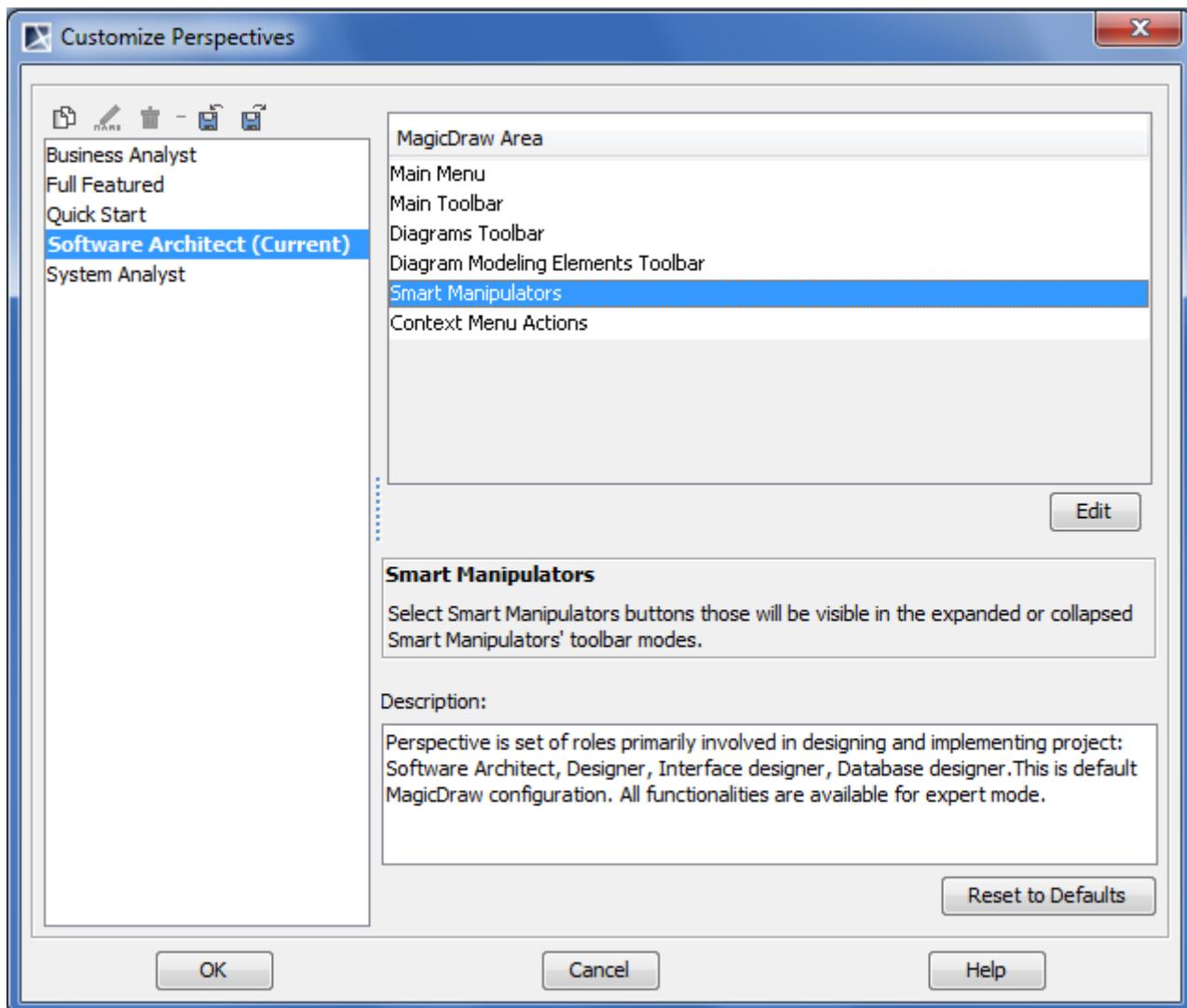


Figure 24 -- Customize Perspectives dialog box

| Element Name | Function |
|-----------------------------|--|
| Clone Selected Perspective | Copies the selected perspective to a new one. |
| Rename Selected Perspective | The Enter Perspective Name dialog box opens. Change name of the perspective and click OK . Renaming can also be performed using the F2 shortcut key. |
| Remove Selected Perspective | Deletes the selected perspective from the list. |
| Import New Perspective | The Open dialog box appears. Select *.umd extension file and click Open to import the perspective into the MagicDraw environment. |

3 USING MAGICDRAW

Customizing and Selecting Perspective

| Element Name | Function |
|---|--|
| Export Selected Perspective  | The Save dialog box opens. Type a name for the created perspective and click Save to store it as *.umd extension file. |
| Business Analyst Software Architect System Analyst | List of possible perspectives, which will load the predefined MagicDraw configuration. |
| MagicDraw Area | List of customizable toolbars and command sections. |
| Edit | Click the Edit button to open the Customize Main Menu dialog box in which a commands mode could be changed by selecting radio buttons. |
| Description | Displays short description about each selected area. |
| Reset to Defaults | Resets changes back to the default configuration. |

Understanding MagicDraw User Interface (UI)

MagicDraw main window has the following parts:

- Main menu
- Main toolbars
- Model Browser
- Diagram toolbars
- Diagram pallet
- Diagram pane

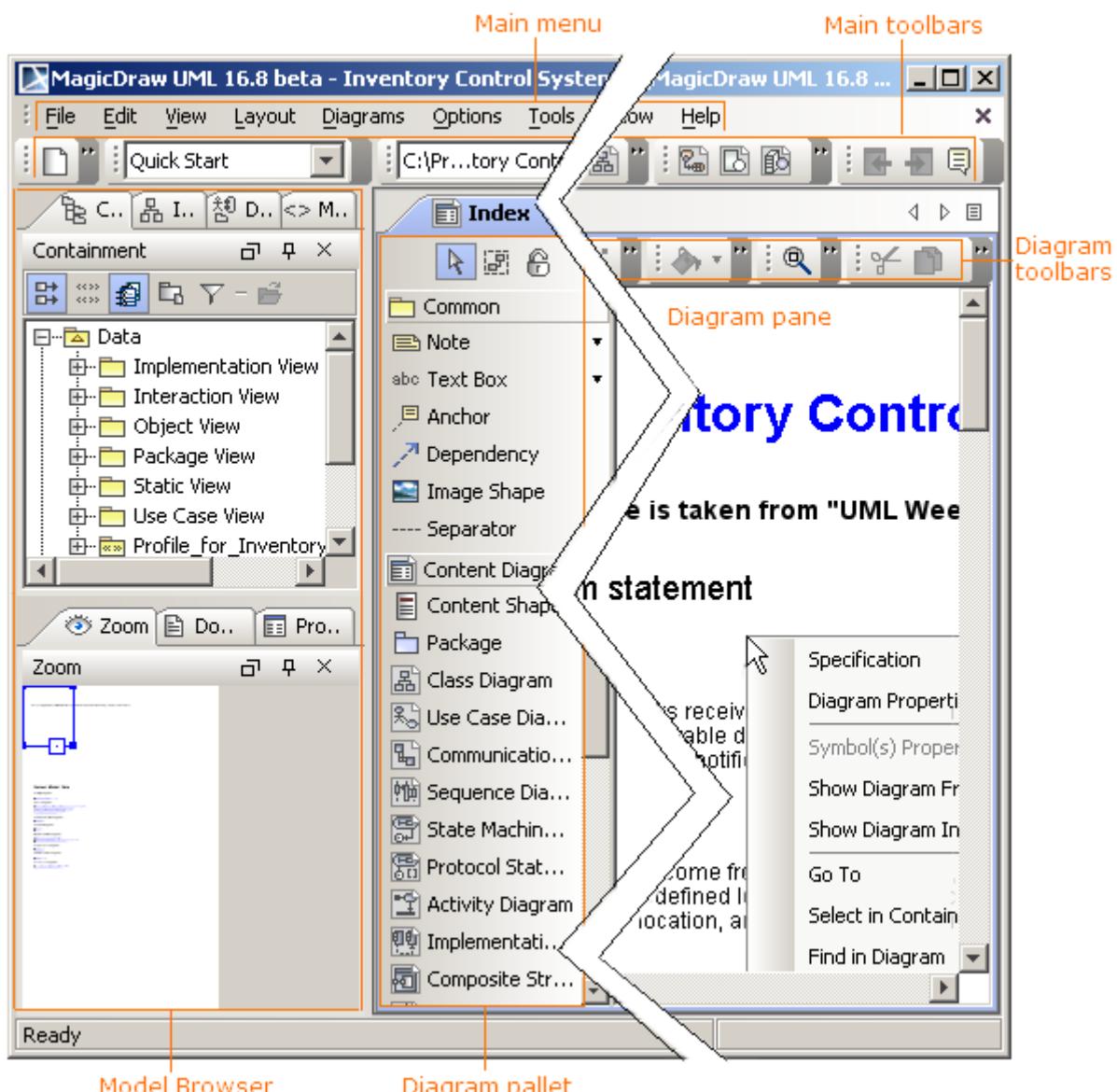


Figure 25 -- MagicDraw main UI structure

Nearly all MagicDraw commands can be accessed from multiple places within MagicDraw:

- Main menu
- Main toolbars
- Model Browser
- Diagram toolbars

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

- Shortcut menus (accessible by right-clicking)
- Shortcut keys
- Smart manipulators (accessible by selecting a symbol on the diagram pane).

The table below shows the accessibility of several commands in various ways:

| Function | Accessibility |
|---|--|
| Main operations of editing (copy, cut, paste, delete) | <ul style="list-style-type: none">• Edit menu.• Main toolbar buttons.• Shortcut keys.• Shortcut menu commands from Browser. |
| Opening of the Specification dialog boxes | <ul style="list-style-type: none">• By double-clicking the model element.• Specification command from the element shortcut menu on the diagram or Browser.• When adding one model element to another model element from the Specification dialog box or Browser. |
| Defining symbols properties (font, color, etc.) | <ul style="list-style-type: none">• Symbol shortcut menu -> Symbol(s) Properties.• Options menu -> Project.• Main toolbar buttons |
| NOTES | <p>The Symbol shortcut menu is also accessible from the Edit menu, Symbol.</p> <p>The toolbar of a particular diagram presents the paths and shapes available for the corresponding diagram. If an arrow is placed on the diagram toolbar button, select a button representing the corresponding model element by right-clicking the button.</p> |

Menus

The description of all menu commands you may find in Section "Menu System" on page 782

You may customize menu items by selecting and/or modifying perspectives. More information about customizing perspectives, you may find in Section "Customizing and Selecting Perspective" on page 63.

NOTE Various plugins might bring additional menu items.

Toolbars

Toolbars help to speed up your work with MagicDraw, when performing commonly used tasks. There are main toolbars and diagram toolbars in the main window of MagicDraw.

To show or hide different toolbars

- Right-click the toolbars area and then select or clear the check boxes of the toolbars you want to be displayed or hidden.

You may also save your own toolbars configuration and set it as a default one (for more information see "Customizing toolbars", on page 72).

Main toolbars

The main window of MagicDraw contains the following main toolbars:

- **File** (main)
- **Diagrams**
- **Custom Diagrams**
- **Diagrams Navigation**
- **Opened Projects**
- **Perspectives**
- **Teamwork**
- **External Tools**
- **Validation**

For detailed information about the commands of the main toolbars, see "Main Toolbars", on page 798.

Diagram toolbars

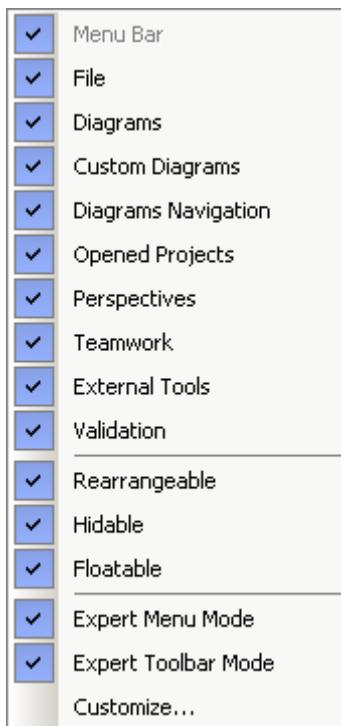
The main window of MagicDraw contains the following diagram toolbars:

- **Symbol Editing**
- **Shape Editing**
- **Path Editing**
- **Edit**
- **View**
- **Layout**

For detailed information about the commands of the diagram toolbars, see "Diagram Toolbars" on page 804.

Customizing toolbars

Toolbars configuration shortcut menu has the following commands:



| Check box | Function |
|----------------------------|---|
| Rearrangeable | If selected, it is possible to change the toolbar position by selecting the dotted line in front of the desired toolbar group and dragging it to a new location. |
| Hidable | If selected, there is no possibility to close a separately opened toolbar group (for example, dragged diagram pane) with the X button on the right top corner. |
| Floatable | If selected, the toolbar group can be dragged to any desirable position inside the MagicDraw borders. |
| Expert Menu Mode | If selected, all menu commands will be listed on the menu. Otherwise, the command list will be shortened and you can expand it by clicking the arrow on the bottom. |
| Expert Toolbar Mode | If selected, displays all toolbar buttons, which were marked to be shown in the Expert mode perspective. |
| Customize | Opens the Customize Toolbars dialog. |

To add a new toolbar

1. From the toolbars configuration shortcut menu, choose **Customize**. The **Customize Toolbars** dialog opens.
2. Click **Add > New Toolbar**.
3. Type the name for a new toolbar.
4. Click **OK**.

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

To add a new button to the selected toolbar

1. From the toolbars configuration shortcut menu, choose **Customize**. The **Customize Toolbars** dialog opens.
2. Click **Add > Button**.
3. Select the desired command.
4. Click **OK**.

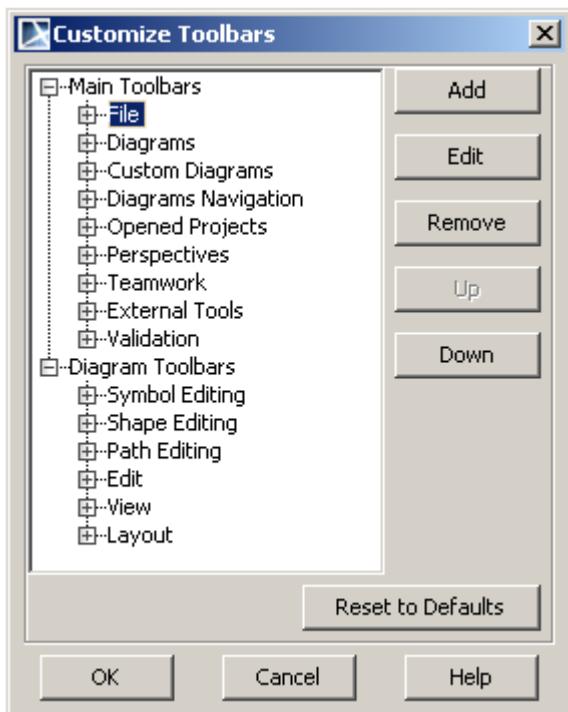


Figure 26 -- The Customize Toolbars dialog

| Button | Function |
|--------------------------|--|
| Add | Adds a new button or a toolbar. |
| Edit | The Edit Icon dialog opens. Click the “...” button to add an icon to the selected toolbar button. |
| Remove | Removes the selected button from the toolbar section. |
| Up | Moves the selected button up the toolbar list. |
| Down | Moves the selected button down the toolbar list. |
| Reset to Defaults | Resets changes made to the toolbar back to the default settings. |

Using the Model Browser

[View Online Demo](#)

MagicDraw Basics

The Model Browser provides a visual representation of the hierarchy of your model elements. The items in this hierarchy are either:

- Compressed - a plus sign next to an icon indicates that the icon is compressed, which means that it contains other model elements. This is the default setting when you start your application. Click the plus sign to expand the icon and view its subordinate items.
- Expanded - a minus sign next to an icon indicates that the icon is fully expanded. Click the minus sign to collapse the item.

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

If there is no sign next to an icon, it does not contain other model elements.

The Model Browser is a hierarchical navigation tool that allows you to manage your model data, including packages, components, classes, all UML diagrams, extension mechanisms, and other data. The Model Browser may be used as an alternative tool to the menus and toolbars that are in MagicDraw. It is easier to work with project diagrams and data elements using the Model Browser. The Model Browser performs the following operations:

- Creation and specification of model elements without viewing them.
- Copying, cutting, and pasting of model elements.
- Opening and deleting of model elements.
- Dragging and dropping of model elements to the Diagram pane and inside the Model Browser.
- Dragging and dropping of data in the Code engineering sets (you may create data in the Data branch, drag it to the Code Engineering sets, and then the round trip object is created automatically).
- Hierarchical viewing of all model elements.
- Trace viewing for the selected model element.
- Symbol creation for the selected model element in the current diagram.
- Managing diagrams.
- Managing extension mechanisms such as constraints, stereotypes, and tagged values.
- Java reversing of a class directly from the classpath.
- Adjusting the code engineering sets.
- Code generation for particular sets.
- Filtering of the visible items, by any model type,(e.g. class, package, operation, component, state and others - for both views and dates), when the Filter from the Model Browser shortcut menu is selected.
- Sorting of the visible items for the selected model element.
- Sorting of all model elements.
- Displaying search results.

The Model Browser window is divided into two parts:

- **Containment tree/Diagrams tree/Inheritance tree/Model Extensions tree/Search Results.**
The Containment tree tab groups data in the logical sets.
The Diagrams tab groups diagrams that are represented on the diagram pane according to the diagrams type or shows them as a list.
The Inherence tree tab represents the class hierarchy of the project.
The Model Extensions tree tab represents all predefined and created constraints and stereotypes.
The Search results tab displays search results.
- **Documentation/Zoom Control** part. The Documentation tab shows documentation associated with the selected item. The **Zoom Control** tab is responsible for zooming the current diagram.

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

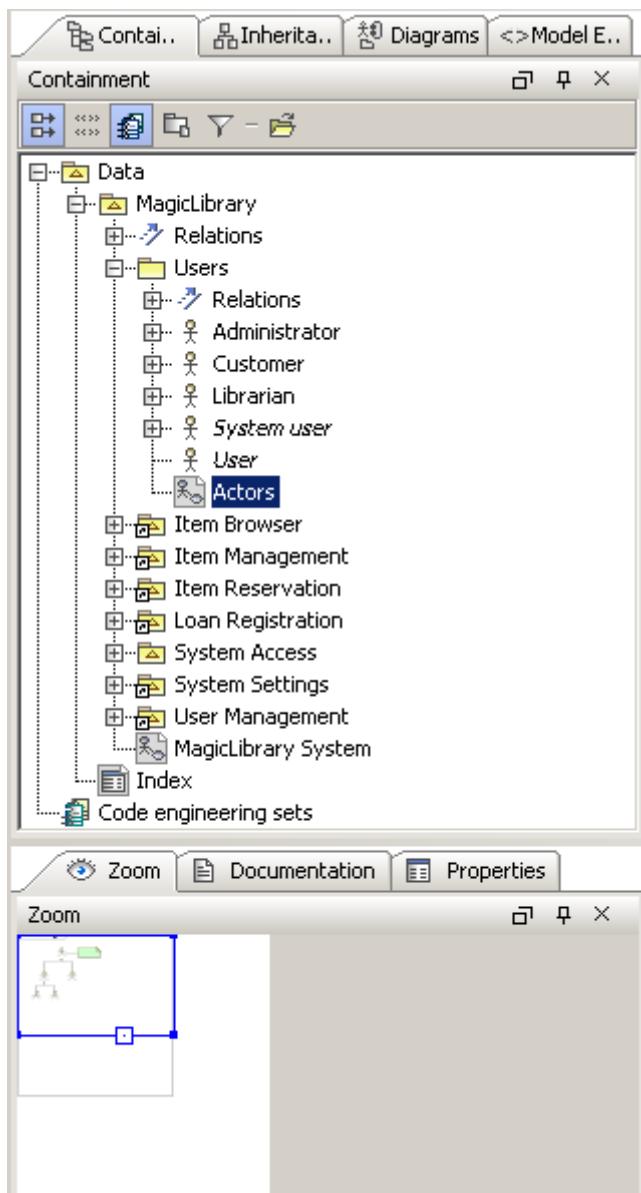


Figure 27 -- The Model Browser window

When at least one Project is open, the Model Browser is placed to the left side of the main window. Beginning with MagicDraw version 7.5, it is possible to move the Model Browser to any place on the the MagicDraw application. Also, all tabs can be viewed separately and you may set up the Model Browser according to your needs by hiding the desired tabs.

To change the size of either part of the Model Browser

- Drag the bar that separates the two parts.

To change the Model Browser position from the **Options** menu

1. From the **Options** menu, select **Environment**. The **Environment Options** dialog opens.
2. Click on the **Browser** tab.
3. In the Browser environment options pane, change the **Browser Position** property to **Right** or **Left**.

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

To close or reopen the desired tab of the Model Browser

- From the **Window** menu, choose the tab you want to close/open.

To reset all Model Browser tabs to the default position

- From the **Window** menu, choose **Reset Windows Configuration** command.

To sort items in Model Browser alphabetically

- From the **Options** menu, select **Environment**. The **Environment Options** dialog opens.
- Click on the **Browser** tab.
- In the Browser environment options pane, set the **Sort Always** check box to “true” (default “true”).

Containment tree

The Containment tree displays model data, grouping it in logical sets.

To open the Containment tree

- At the top of the Model Browser, click the **Containment** tab.
- If the Containment tree is hidden, from the **Window** menu, select **Containment**.

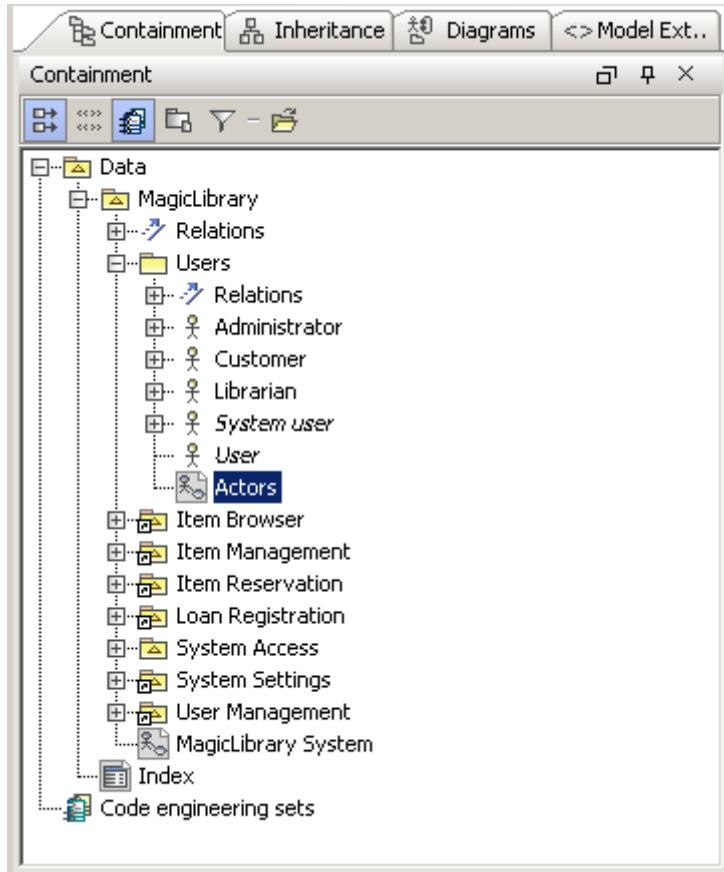


Figure 28 -- Containment tree

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

To show full information of operations, attributes, and relationships in the Containment Tree

Do one of the following:

- In the **Containment** tab, click the **Show Full Types in Browser** button  .
- Do the following:
 1. Choose **Environment** command from the **Options** menu. The **Environment Options** dialog box appears.
 2. In the **Browser** tab, set the Show Full Types in Browser check box to “true” (default “false”).

To show stereotypes in the Containment tree

Do one of the following:

- In the **Containment** tab, click the **Show Stereotypes in Browser** button  .
- Do the following:
 1. Select **Environment** from the **Options** menu. The **Environment Options** dialog box opens.
 2. In the **Browser** tab, set the **Show Stereotypes in Browser** check box to “true” (default “false”).

To show/hide Code Engineering sets branch

Do one of the following:

- In the **Containment** tab, click the **Show Code Engineering Sets** button  .
- Do the following:
 1. Select **Environment** from the **Options** menu. The **Environment Options** dialog box opens.
 2. In the **Browser** tab, set the **Show Code Engineering Sets** check box to “true” (default “true”).

To show/hide Modules

Do one of the following:

- In the **Containment** tab, click the **Show Auxiliary Resources** button  .
- Do the following:
 1. Select **Environment** from the **Options** menu. The **Environment Options** dialog box opens.
 2. In the **Browser** tab, set the **Show Auxiliary Resources** check box to “true/false” (default “true”).

To filter types of elements to be displayed

To improve accessibility the **Filter** button has been added to the Containment Tree toolbar (Figure 29 on page 78). Previously the **Filter** menu was accessible only from the Containment tree shortcut menu (Figure 30 on page 78).

The **Items Filter** dialog allows you to choose what types of elements to be displayed in the Containment tree (Figure 31 on page 79).

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

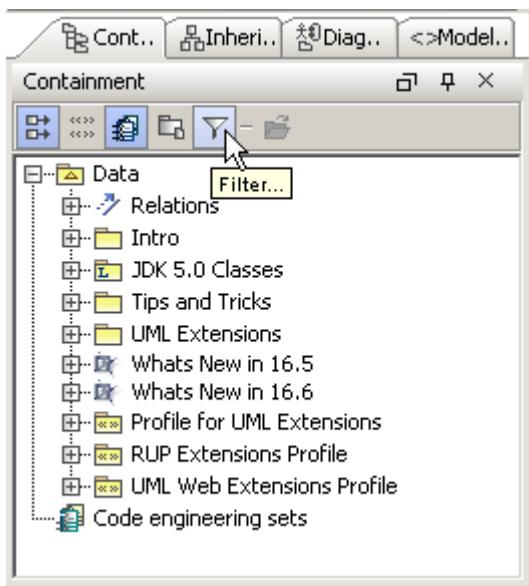


Figure 29 -- The Filter Button in the Containment Tree

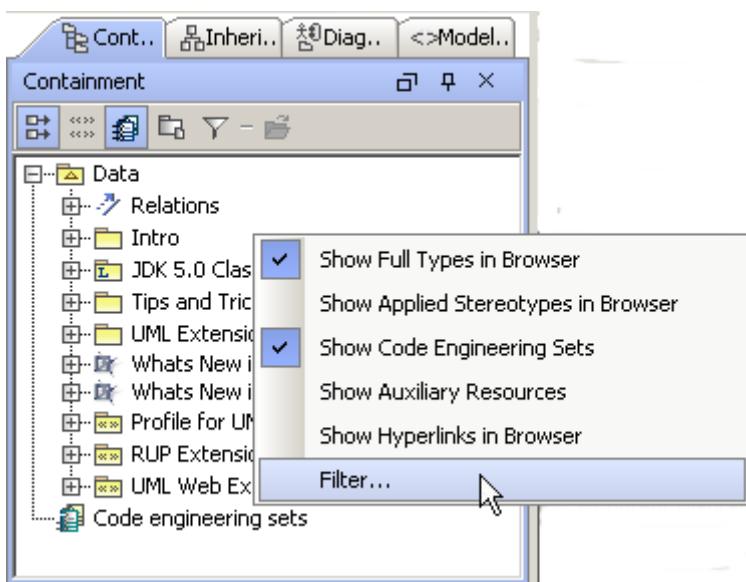


Figure 30 -- The Filter command in the Model Browser shortcut menu

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

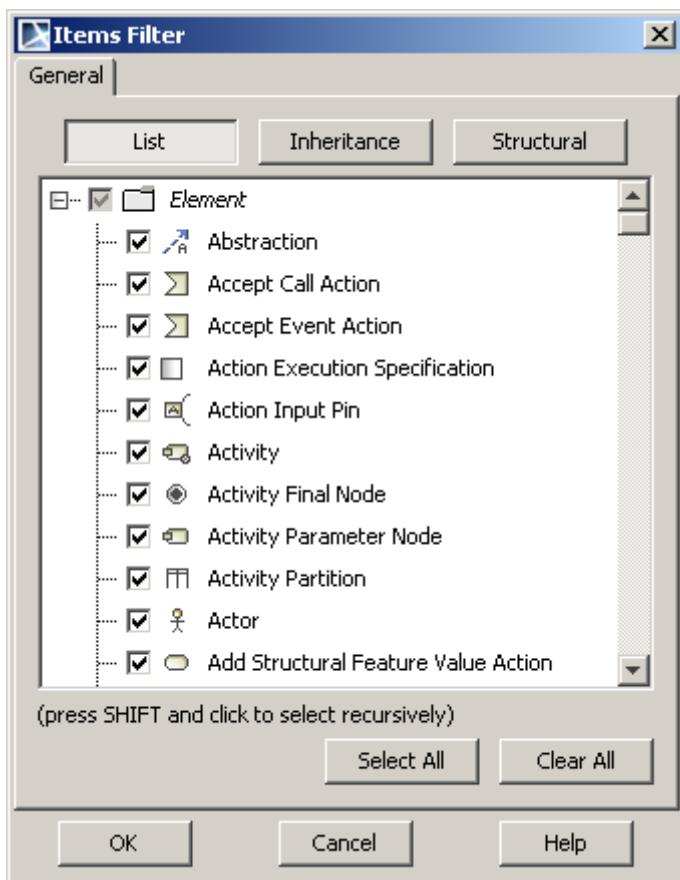


Figure 31 -- The Items Filter dialog box

To open package contents in a new tab

- In the **Containment** tab, click the **Open in New Tab** button . New tab with package name and tree contents will be opened in the Model Browser.

Data branch

The **Data** branch represents the model and structure of a project. All model elements are stored in packages. This helps you distribute data into logical groups. By default, all new model element data (inner structure) are stored in the **Data** package. You may create your own packages for storing your model element data.

The **Data** branch also contains the **File View** package, **UML Standard Profile** with stereotypes and data types, and **Relations** package (appears only when at least one path is drawn on the diagram pane).

The **File View** package is where the components are placed during code engineering.

The **UML Standard Profile** contains a list of stereotypes, data types, and elements from the UML 2 metamodel.

To create a new element

- In the Containment tree, from the package shortcut menu, choose **New Element** and select the desired element from the list.
- Type the name of the element directly in the Containment tree.

For more information about managing model elements from the Browser, see [UPDATE]“Working with Elements in the Model Browser” on page 84.

Code engineering sets

The Code engineering sets branch is a gateway between your source code and model data. Using sets, you can perform Java, C++, IDL, DDL, EJB, CIL, and C# round-trip code engineering (code generation + reverse engineering).

To create a new Code engineering set

1. Right-click the **Code Engineering Sets** item and select **New** from the item shortcut menu or open the **Code Engineering Sets** dialog box and click the **New** button. The **New Set** dialog box opens.
2. Type the set name and select the programming language from the drop-down list (by default - Java).
3. Click **OK** to finish the set creation.

To edit the selected set

- Select **Edit** from the set shortcut menu. The **Round Trip Set** dialog box opens. Add/remove files or classes from the Code engineering set.

To rename the selected set

- Select **Rename** from the set shortcut menu and type the set name.

To change the code generation properties

- Select **Properties** from the set shortcut menu. The **CG Properties Editor for Data** dialog box opens.

To delete the selected set

- Select **Delete** from the set shortcut menu.

To restore the deleted set

- From the **Edit** menu, select **Undo** or press the shortcut keys **CTRL+Z**.

To generate code from the selected set

1. Select **Generate** from the set shortcut menu. The **Code Generation Options** dialog box opens.
2. Adjust the code generation options.
3. Click **OK**. The **Messages Window** dialog box opens. Information about generated files is shown.

To check syntax

- Select **Check Syntax** from the set shortcut menu. If no errors are found, then a message box opens stating there are no syntax errors in the model.

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

To reverse the selected set

1. Select **Reverse** from the set shortcut menu.
2. The **Reverse Options** dialog box opens. Define options and click **OK**.
For the detailed descriptions about the reverse, see Section Reverse in the Code Engineering User's Guide.

To reverse files that have been changed

1. Select **Refresh** from the set shortcut menu.
2. The **Reverse Options** dialog box opens. Define the options and click **OK**.
For detailed description about the reverse process, see the "Reverse" Section in the Code Engineering User's Guide.

TIP!

All functions listed above can be performed in the **Code Engineering Sets** dialog box.

To select a text editor for source code

1. In the **Options** menu, select **Environment**, then select the **Launchers** group in the dialog box that opens.
2. In the **Default launchers** field, click the "..." button and select the directory where the text editor is located.
3. Click **OK**.

Diagrams tree

The Diagrams tree in the Browser represents the external structure of a diagram.

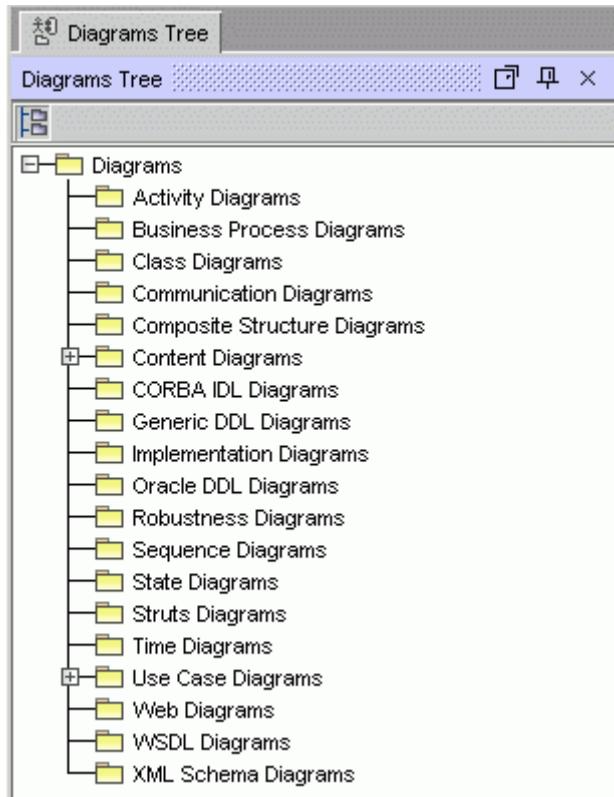


Figure 32 -- Diagrams tree

In the Diagrams tree, with the selected diagrams, you can perform the operations listed below.

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

To group diagrams according to their diagram type

- Click the **Group by Diagram Type** button  in the Diagrams tree.
- From the shortcut menu, select the **Group by Diagram Type** check box.

To open the selected diagram from the Browser

- Select **Open** from the item shortcut menu or double-click the item in the diagram.

To delete the selected diagram

- Select **Delete** from the selected diagram shortcut menu.

To rename the selected diagram

- Select **Specification** from the diagram shortcut menu. The corresponding **Diagram Specification** dialog box opens. Type the diagram name and click **OK**.

TIP!

In the **Diagram Specification** dialog box you can add documentation to the diagram, view the relationships in which the diagram participates, and define hyperlinks, stereotypes, constraints, and tagged values.

To print the selected diagram

- Select **Print** from the diagram item shortcut menu. If the diagram is empty, it will not be printed.

Inheritance tree

The Inheritance tree represents classifiers, packages, data types, and stereotypes hierarchy within your project. Inheritance according to the UML Specification is shown using the generalization relationship.

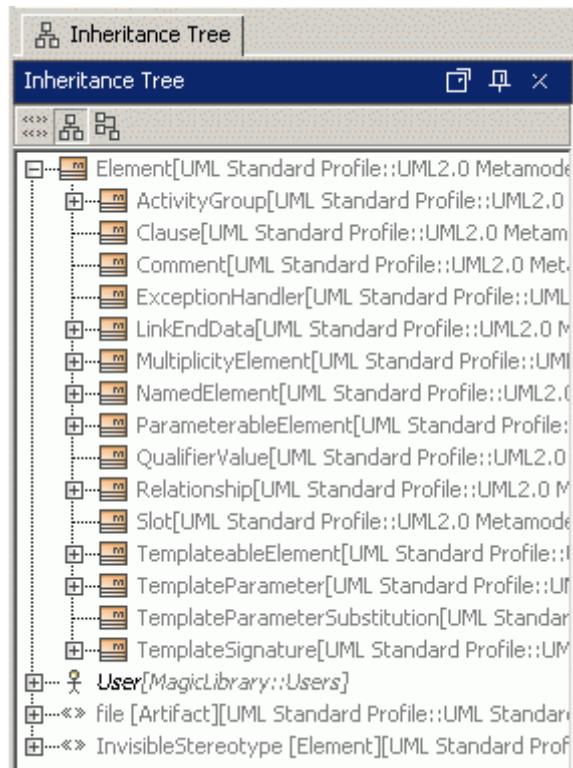


Figure 33 -- Inheritance tree

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

To show stereotypes in the Inheritance tree

- In the **Inheritance Tree** tab, click the **Show Stereotypes in Browser** button .
- 1. Select **Environment** command from the **Options** menu. The **Environment Options** dialog box opens.
- 2. In the **Browser** tab, set the **Show Stereotypes in Browser** check box to “true” (default “false”).

To show classifier hierarchies in the Inheritance tree

- In the **Inheritance Tree** tab, click the **Show only Hierarchies** button .
- 1. Select **Environment** command from the **Options** menu. The **Environment Options** dialog box opens.
- 2. In the **Browser** tab, set the **Show Only Hierarchies** check box to “true/false” (default “false”).

To invert tree in the Inheritance tree

- In the **Inheritance Tree** tab, click the **Invert Tree** button .
- 1. Select **Choose Environment** command from the **Options** menu. The **Environment Options** dialog box opens.
- 2. In the **Browser** tab, set the **Invert Tree** check box to “true/false” (default “false”).

Model Extensions tree

The Model Extensions Tree contains all Stereotypes that are predefined and created manually in the project.

In this tree you can create, review, copy/paste, and delete extension mechanisms.

It is mainly used for the work of a team using Teamwork server for locking for edit/unlocking extension mechanisms.

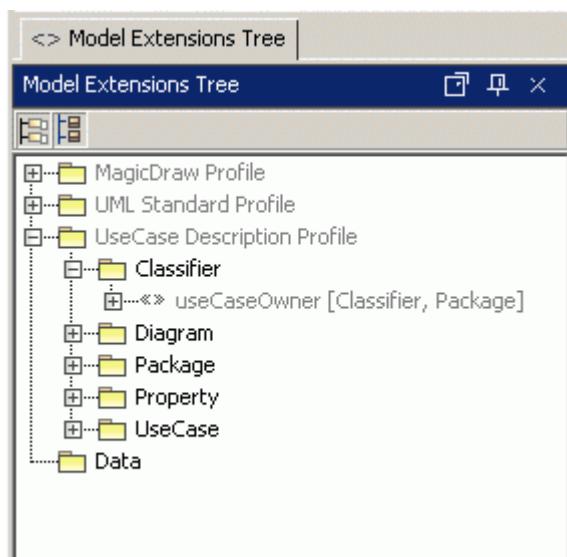


Figure 34 -- Model Extensions Tree

To group extensions by profiles

- In the Model Extensions Tree, click the **Group by Profiles** button .

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

- From the Model Extensions Tree shortcut menu, select the **Group by Profiles** check box.

To group extensions by metaclasses

- In the Model Extensions Tree, click the **Group by Metaclasses** button .
- From the Model Extensions Tree shortcut menu, select the **Group by Metaclasses** check box.

Search Results Tree

The Search Results tree shows results of a search, which may be performed through the [Find dialog box](#).

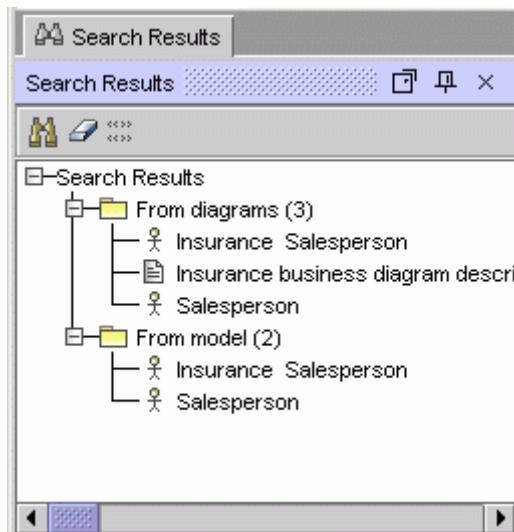


Figure 35 -- Search Results tab

The results in the tab are displayed in two packages:

- From Diagrams - elements are shown that are displayed on the diagram(s).
- From Model - elements are shown that are created in the model data.

For information about how to perform a search, see “Searching” on page 134.

To open the **Find** dialog box from the **Search Results Tree**

- Click the **Find** button  , or press CTRL+F keys.

To clear the results of the previous search

- Click the **Clear Results** button  or select the **Clear Results** button from the **Search Results** tab.

Working with Elements in the Model Browser

To create an element in the Model Browser

- In the Containment tree, right-click the package.
- Click **New Element** and select the model element you wish to create.

To create a diagram in the Model Browser

- In the Containment tree, right-click the package.

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

2. Click **New Diagram** and select the diagram you wish to create.

To copy/cut and paste the selected model element in the Model Browser

1. From the element's shortcut menu, select **Copy** or **Cut**.
2. Select the container, where you wish to put the model element.
3. From the container's shortcut menu, select **Paste**.

To copy/cut and paste the selected model element among different projects

1. From the item shortcut menu, select **Copy** or **Cut**.
2. From the **Opened Projects** main toolbar, select the other project.
3. Select the container, where you wish to put the element.
4. From the container's shortcut menu, select **Paste**.

To delete a model element from the Model Browser

- From the element's shortcut menu, select **Delete**.

To drag-and-drop the selected item in the Model Browser

1. Make sure that the place you wish to drag the item is visible.
2. Drag the selected item to the destination and drop it.

To draw a symbol on the diagram

- From the items shortcut menu, select **Create Symbol**.

TIP!

You may draw a symbol by dragging and dropping an item to the Diagram pane.

To show/hide the model elements in the Model Browser

1. Do one of the following:
 - From the Browser shortcut menu, select **Filter**.
 - On the Containment tree toolbar, click **Filter**.
2. In the **Items Filter** dialog, clear the check boxes of those model elements that you would not like to appear in the Model Browser.

For more information about how to filter items, see “To filter types of elements to be displayed” on page 77.

Multiple selection

A group of model elements can be selected within the Browser tree and you can edit all the selected model elements at the same time.

To make multiple selections

- Hold down the SHIFT key and click the last element you wish to include in the multiple selection.
- For more precise selection, hold down the CTRL key and click (while holding the key down) with the mouse on the elements you wish to select.

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

To select all model elements or all browser tree items

- Click the mouse pointer in the area you want to select all elements and press the shortcut keys **CTRL+A**.

NOTE All selected model elements can be moved or deleted as a single selected element. While moving the group of elements, a border appears denoting the area you have selected. Be careful when deleting multiple elements because no confirmation dialog box will appear.

Zoom panel

The **Zoom** panel enables you to preview any selected diagram. To select a diagram go to Model Browser > Containment/Diagram tree. You can also use this panel for zooming in or out the active diagram.

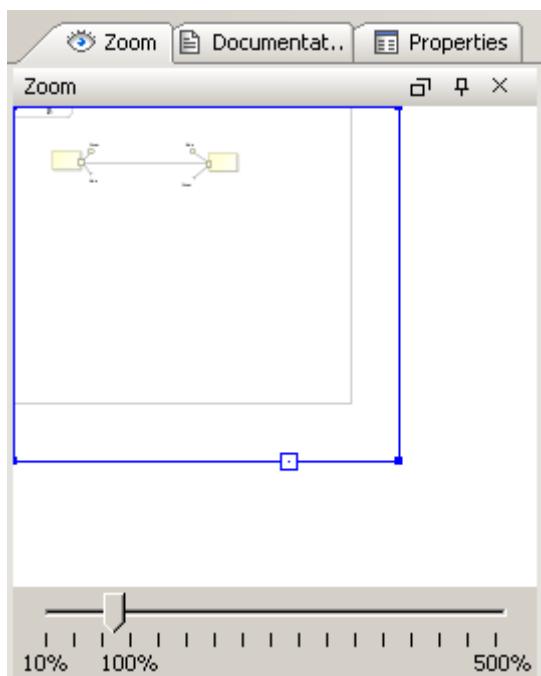


Figure 36 -- Zoom panel

To open the **Zoom** panel

Do one of the following:

- In the Model Browser, click the **Zoom** panel.
- On the **Window** menu, click **Zoom**.

To zoom in or out the active diagram

- In the **Zoom** panel, drag corners of the blue square.

NOTE By default the zoom slider is not visible. To display the zoom slider in the **Zoom** panel, go to **Options > Environment**. In the **Environment Options** window select the **Browser** tab. Then select the **Show diagram zoom slider** check box.

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

To access quickly any part of the diagram

- In the **Zoom** panel, drag the blue square to the desired part of the diagram. The desired diagram part will be displayed in the diagram window.

To fit the diagram to the window

- In the **Zoom** panel, on the blue square, click the **Fit in Window** button .

Documentation panel

The **Documentation** panel shows the information associated with the selected model element in the Model Browser or on a diagram pane.

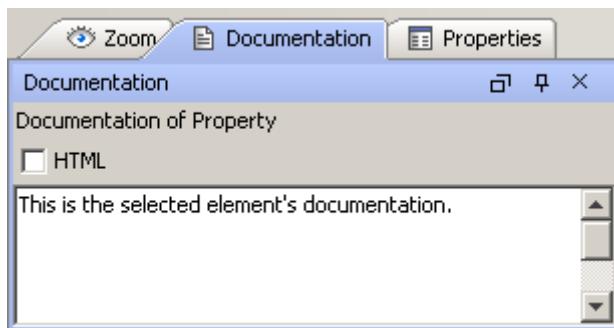


Figure 37 -- Documentation panel

To open the **Documentation** panel

Do one of the following:

- In the Model Browser, click on the **Documentation** panel.
- On the **Window** menu, click **Documentation**.

NOTE

If there is no documentation for the selected element, the text area is empty.

To write documentation for the selected model element

- In the **Documentation** panel, click on the text area and type the text.

Properties panel

The Properties panel in the Model Browser has been available since MagicDraw version 8.0 was released. It enables you to quickly access the basic information about the selected element or diagram.

To open the **Properties** panel

Do one of the following:

- In the Model Browser, click on the **Properties** panel.
- On the **Window** menu, click **Properties**.

The **Properties** panel includes the following tabs:

| Tab name | Description |
|----------|---|
| Element | Contains the main properties in the element's or diagram's specification. |

3 USING MAGICDRAW

Understanding MagicDraw User Interface (UI)

| Tab name | Description |
|---------------------|---|
| Symbol | Contains the element symbol's properties. NOTE: This tab is available for symbols only. |
| Language properties | Contains the same data as the Language Properties tab in the element's Specification window. NOTE: This tab is available only for the elements, which are used for code generation, e.g., class, attribute, operation. |
| Traceability | Contains the same data as the Traceability tab in the element's Specification window. For more information about traceability, see "Traceability" on page 386. |

All **Property** panel tabs, except the one for traceability properties, has two modes: **Standard** and **Expert**. Choose the mode that best suits your needs.

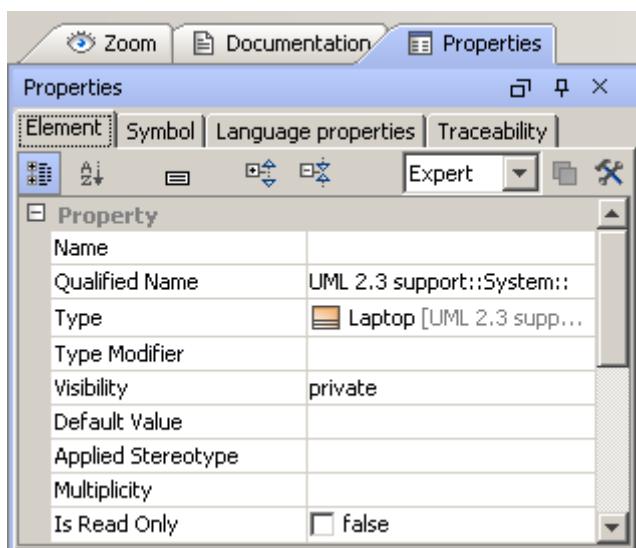


Figure 38 -- Properties panel

3 USING MAGICDRAW

Customizing Environment Options

Click the **Customize**  button to open the **Customize Properties** dialog wherein you can set which properties will be visible in the **Standard** and/or **Expert** modes and which ones will be hidden in any mode.

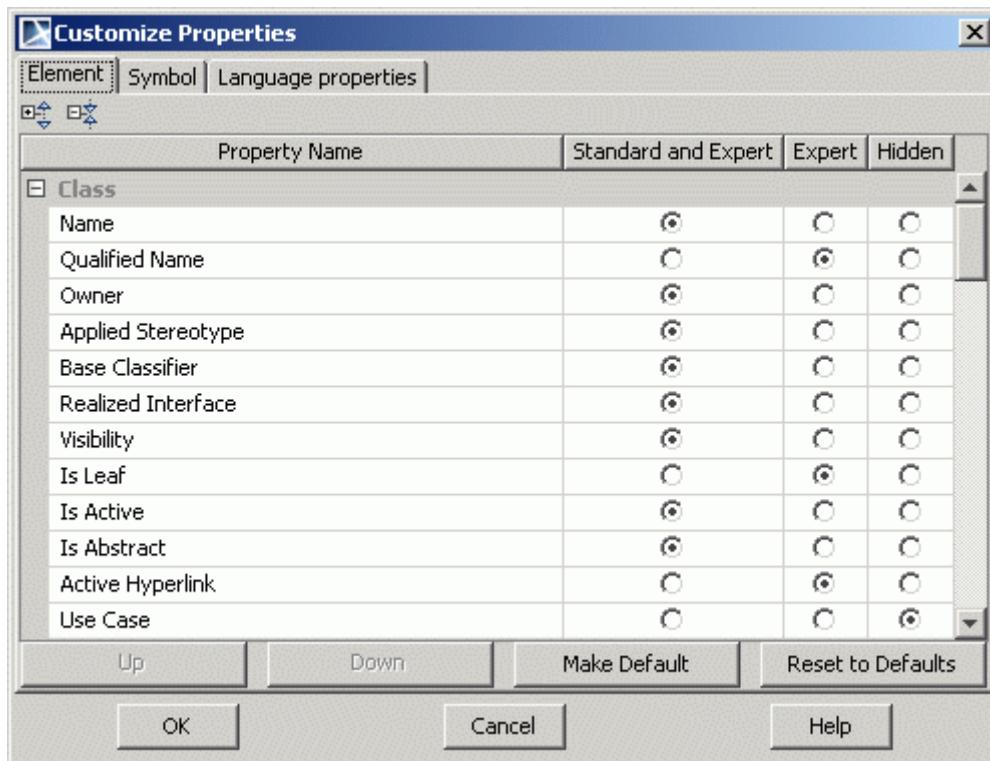


Figure 39 -- Customize Properties dialog box

Customizing Environment Options

You can customize MagicDraw according to your preferences: change the application settings in the **Environment Options** dialog box.

To open the **Environment Options** dialog

- From the **Options** menu, select **Environment**.

Common fields available in the **Environment Options** dialog:

| | |
|---------------------------|---|
| Quick filter field | Type in this field to filter properties. More information about the Quick filter field you may find at Section "Quick filter" on page 242. |
| Reset to Defaults | Resets all properties to the default settings. |
| OK | Saves changes and exits the dialog box. |
| Cancel | Exits the dialog box without saving changes. |
| Help | Displays MagicDraw Help. |

General Options Tab

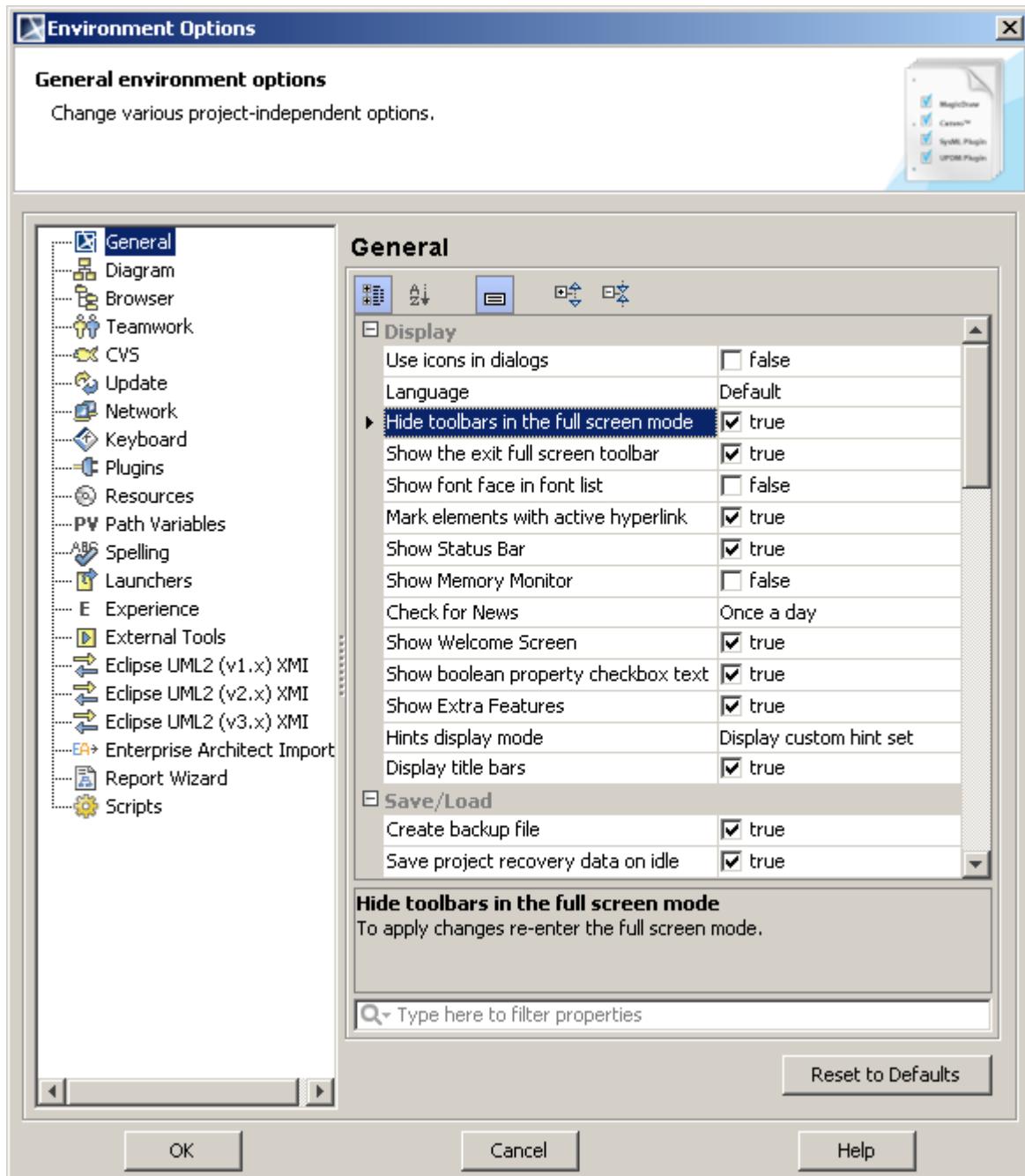


Figure 40 -- Environment options dialog. General options tab

Change the appropriate fields according to your preferences. If the check box is selected, the description False changes to True.

| Property name | Function |
|-----------------------------|--|
| Display | |
| Use icons in dialogs | Icons are shown on the button of dialog boxes. |

3 USING MAGICDRAW

Customizing Environment Options

| Property name | Function |
|--|--|
| Language | <p>Select the language for the MagicDraw GUI:</p> <ul style="list-style-type: none"> ● Default – according to your application. It is set by default. ● English ● French ● German ● Japanese ● Russian ● Thai <p>NOTE: Restart MagicDraw to apply changes to the language property.</p> |
| Hide toolbars in full screen mode | The main toolbar is hidden, when the diagram full screen mode is turned on. To display the main toolbar, clear the Hide toolbars in full screen mode check box in the Environment Options dialog box, General pane, General group. |
| Show the exit full screen toolbar | Shows the Full Screen toolbar for closing the full screen mode. |
| Show font face in font list | Font names that are listed in the Font drop-down list box on the main toolbar will be displayed in the original font. |
| Mark elements with active hyperlink | Shows active hyperlink arrow on the element symbol. |
| Show Status Bar | Shows status bar line at the bottom of the MagicDraw window. |
| Show Memory Monitor | Shows the project memory usage line at the bottom of the MagicDraw window. |
| Check for MagicDraw News | Specifies how often MagicDraw will check for news. |
| Show Welcome screen | Shows the Welcome screen, if there are no opened projects. |
| Show boolean property checkbox text | Shows the check box labels for boolean properties. |
| Show Extra Features | Shows features that are available only in a higher edition or not installed plugins. |
| Hints display mode | <p>Specifies whether hints on MagicDraw functionality related to the user actions will be displayed. Select one of the following options:</p> <ul style="list-style-type: none"> ● Display custom hint set (default) - shows a custom hint set excluding the hints that are selected by the user not to be shown. ● Display all hints - shows all hints. ● Do not display - displays all hints. <p>NOTE: Hints are displayed in the lower right-hand corner of the application.</p> |
| Save/ Load | |
| Create backup file | Backup files of your projects will be created. Backups are saved with the name pattern old_name.xml.bak or old_name.xml.zip.bak. |
| Save project recovery data on idle | Saves AutoRecovery file of the open project(s) when a system is not in use. |
| Idle time to activate recovery save | Enter the system idle time (in minutes). This is the amount of time until the file recovery save is activated. |
| Open last project on startup | The last project of your previous session will be opened next time you start MagicDraw. |

3 USING MAGICDRAW

Customizing Environment Options

| Property name | Function |
|---|--|
| Diagrams load mode | Select how to load diagrams after opening a MagicDraw project. <ul style="list-style-type: none"> ● Load all Diagrams – opens all diagrams that are in the project. ● Load Only Open Diagrams (default) – opens only diagrams that were not closed in the previous use of the project. ● Do not Load Diagrams – no diagrams will be opened when opening the project. |
| Save rich XMI | Saves maximum additional information to an xmi file, which is not required for MagicDraw load but may be needed when using other tools. |
| Save UUID | If selected, the “xmi:uuid” attribute will be added for each model element in the XMI file. |
| Check dependency on module export/share | Select one of the following options: <ul style="list-style-type: none"> ● Do not check ● Allow dependencies (default) ● Do not allow dependencies |
| Blank project template | Default location and template file name: <install.root>\templates\template.xml |
| Global Modules Path | Select the global part of the path, where the project modules are located. TIP: There is a specific part of the path also, and it can be specified the specific project options. |
| Save stereotype information within element | Specifies where in the XMI file the applied on the element stereotype's information is saved. If the value is true, the stereotype's information is stored at the end of the file. If the value is false, the stereotype's information is stored within the element information. |
| Display title bars | Switch on or off the title bar area in MagicDraw dialogs. |
| General | |
| Save settings on exit | Specifies whether the altered settings within the current MagicDraw session will be stored and applied for all future sessions. |
| Show tip of the day | Shows the Tip of the Day dialog every time you start MagicDraw. |
| Show tips while running long tasks | Shows the Tip of the Day dialog inside the progress bar while running long tasks. |
| Recent files list size | Type the number of projects to appear in the File menu. Click the file name to open it. The maximum file list size is 10. The file name list shows files that were used in MagicDraw. |
| Recent windows list size | Type the number of recently used windows that are listed in the Window menu. Click the window name to make it active. The maximum windows list size is 10. |
| Undo list size | Type the number of actions that will be available for undo/redo operations. They will be listed in undo/redo drop down lists on the main toolbar. The maximum undo/redo list size is 500. |
| Enable view and submit internal errors | Enables viewing and submitting internal errors. NOTE: In order to escape negative after-effects, which are possible, please, keep this option selected. |
| Image Export | |
| Save diagram background in image | Saves the diagram with background. By default, the diagram background is white after saving as image. |

3 USING MAGICDRAW

Customizing Environment Options

| Property name | Function |
|---|--|
| JPEG Compression Quality | This is a JPEG format option. |
| Use SVG <text> tag for text output | This is an SVG format option. |
| Image resolution (DPI) | This property is DPI property value with numeric value range from 1 to 4800. Default value is 72. |
| Exported image size [%] | Define image size in percent. Default value is 100%. For example; if 200% is defined, then the view is enlarged (zoomed) before generating an image. Raster image will not lose quality as additional pixels are introduced. |
| TIFF Color Space | This is a TIFF format option. |
| TIFF Compression | This is a TIFF format option. |
| Editing | |
| Update parameter's properties while it has reference on other argument | Synchronizes the parameter's properties with the referenced argument's updates. |
| Opaque Expression default language | Select one of the following options: <ul style="list-style-type: none"> ● English ● OCL |
| Opaque Expression displaying lines | Select one of the following options: <ul style="list-style-type: none"> ● 1 ● 2 ● 3 ● 4 ● 5 ● All Lines |
| Synchronize diagram name with it's context name | Synchronizes the diagram name with the diagram context name. For example, an activity diagram is automatically named after the name of the activity, which owns the diagram. |
| Help | |
| Help font | Choose the font that will be used in the Help window. |
| Help server port | The default Help server port is 1111. Change it, if some other server is using this port. |
| Specifications | |
| Strip Multiline Text | If selected, shows striped multiline text. |
| Specification opening mode | Select one of the following options: <ul style="list-style-type: none"> ● In the same window ● In a new window |
| Recent specifications list size | Specify the list size for specifications. Possible range 1 to 20. |
| Specifications history size | Specify the history size for open element specifications. Possible range 1 to 20. |
| Active Validation | |
| Enable Active Validation | Enables the active validation on run time. |

3 USING MAGICDRAW

Customizing Environment Options

| Property name | Function |
|---|--|
| Active Validation Period (seconds) | Specify the period (in seconds) for the active validation in the model. TIP: Specify the higher period for the better performance. |
| Merge | |
| Optimize for | Select one of the following options: <ul style="list-style-type: none"> • Speed • Memory |

Diagram Options Tab

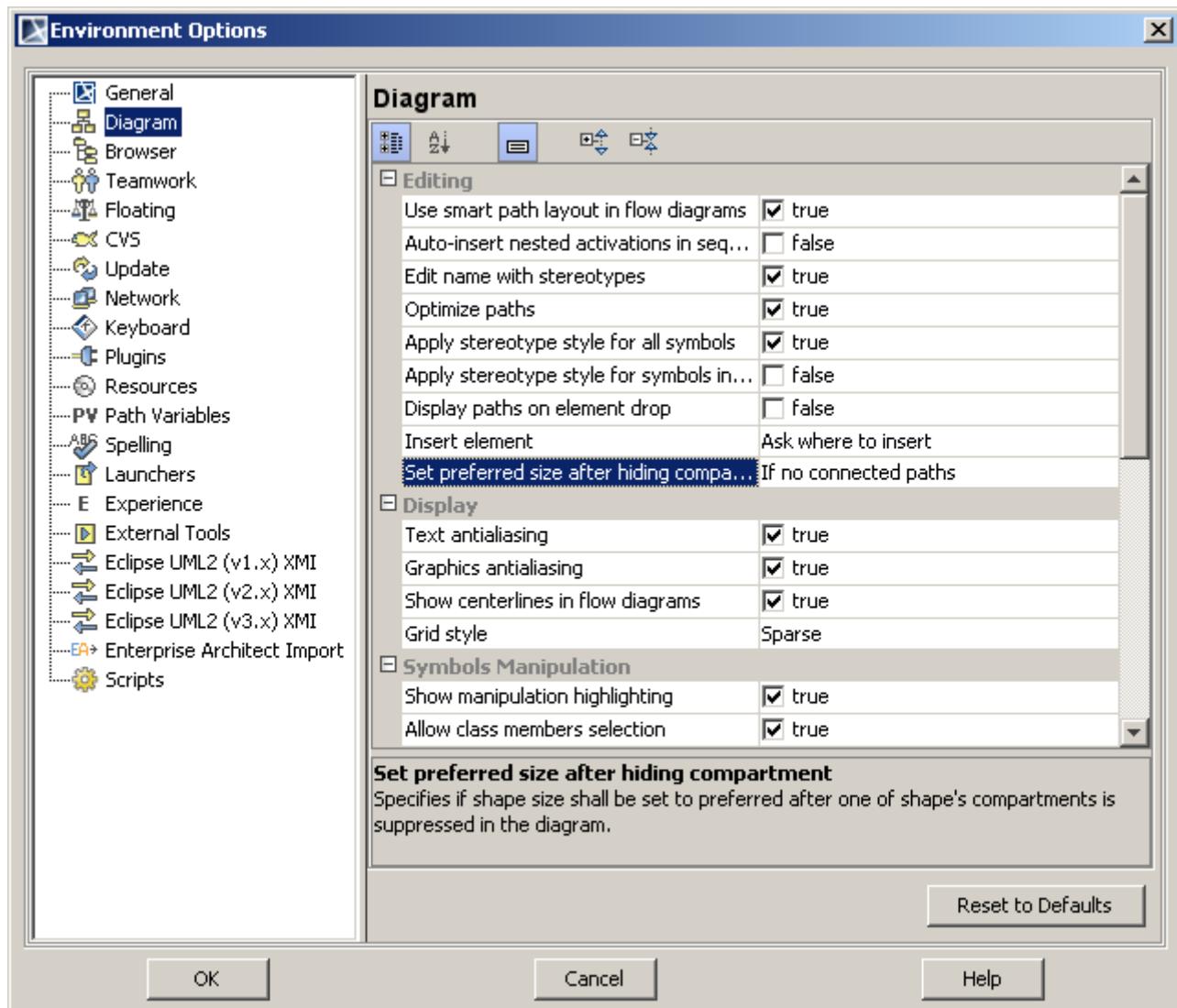


Figure 41 -- Environment Options dialog. Diagram options tab

| Property name | Function |
|------------------------------|---|
| Text antialiasing | Smooths the jagged edges of text. |
| Graphics antialiasing | Smooths the jagged edges of graphics. |
| Grid style | Select one of the following styles of the grid: Dense or Sparse . |

3 USING MAGICDRAW

Customizing Environment Options

| Property name | Function |
|--|--|
| Validate Shape Ownership | Validates if the symbol owner on the diagram matches the actual element owner in the model. Using the symbol ownership validation feature, shape ownership problems will be highlighted on the diagram pane and you will be able to easily see and resolve them. For more information see “Resource Manager” on page 366. |
| Shape Ownership Validation Period (seconds) | Type the period in seconds to check diagram for symbol ownership problems. |
| Edit name with stereotypes | Allows the management of stereotypes by editing the stereotype name directly on the Diagram pane. |
| Show manipulation highlighting | Enable error highlighting of the modeling process. This helps you to see the errors in the model at drawing time. The valid and invalid actions will be highlighted in different colors (defaults are red for invalid actions, blue for valid actions). |
| Allow class members selection | Operations and attributes can be selected, then dragged & dropped to another class. |
| Optimize paths | When a path is drawn in various directions, this option removes loops from the path. |
| Apply stereotype style for all symbols | Applies the created stereotype style to a symbol when adding the stereotype to an element. |
| Apply stereotype style for symbols in unlocked diagrams | Applies the stereotype style in teamwork for unlocked diagrams. |
| Mark selected labels | Mark path labels after a path is selected. |
| Display paths on element drop | When dragging and dropping an element from the Browser or executing a Create Symbol action, the path to the existing symbols is created. |
| Insert Element | Splits the path into two paths by drawing a symbol on it and inserts this symbol according to the choice. Possible choices: <ul style="list-style-type: none"> ● Before path ● After path ● Do not insert ● Ask where to insert NOTE: This is valid only in state and activity diagrams. |
| Set preferred size after hiding compartment | Specifies if shape size shall be set to preferred after one of shape's compartments is suppressed in the diagram. Possible choices: <ul style="list-style-type: none"> ● Never; ● Always; ● If no connected paths (default). |
| Auto completion includes metaclasses | If selected, the list of available elements to choose , element types, or stereotypes includes metaclasses (in MagicDraw metaclasses are placed in the <i>UML Standard Profile</i> . Default value is <i>false</i> . [update] |
| Auto completion includes elements from profiles and modules | When selected, the list of available elements to choose, element types, or stereotypes includes elements, which are placed in profiles and modules. Default value is <i>true</i> . NOTE: This option toggles all profiles except <i>UML Standard Profile</i> . [update] |
| Zoom step size | Type in the step value of zooming your diagram views. Step value should be in the range from 0.1 to 1.0. |

3 USING MAGICDRAW

Customizing Environment Options

| Property name | Function |
|---|--|
| Center on the mouse location when zooming | Centers zooming according to mouse arrow. |
| Show Smart Manipulations | Set smart manipulator visibility. For a detailed description about the smart manipulation feature, see "Smart Manipulation" on page 202 |
| Mouse's Left Button Action | Possible choices: Create element - target element is created during the creation of a path after a mouse click on the diagram pane. Make a break point - set a break point for the path after a mouse click. |
| Smart Manipulator fade time on mouse over | Set the smart manipulators fade time in seconds. |
| Smart Manipulator appearance | Possible choices: <ul style="list-style-type: none">● Mouse Over● Shift |
| Show Names in Toolbar | Shows element names in toolbars near icon. |
| Toolbar Position | Change toolbar appearance near diagram pane. Possible choices: <ul style="list-style-type: none">● Right● Left |
| Inform about the removal of existing elements, when changing a context element | If any elements exist in the Relation Map, when changing a context element, show information message. |
| Show question dialog when restoring layout | Show the question dialog, which asks to preserve or not manually suppressed/expanded branches and hidden elements when restoring layout. |
| Preserve manually suppressed / expanded branches and hidden elements true | When restoring layout, preserve manually suppressed/expanded branches and hidden elements. |

Browser Options Tab

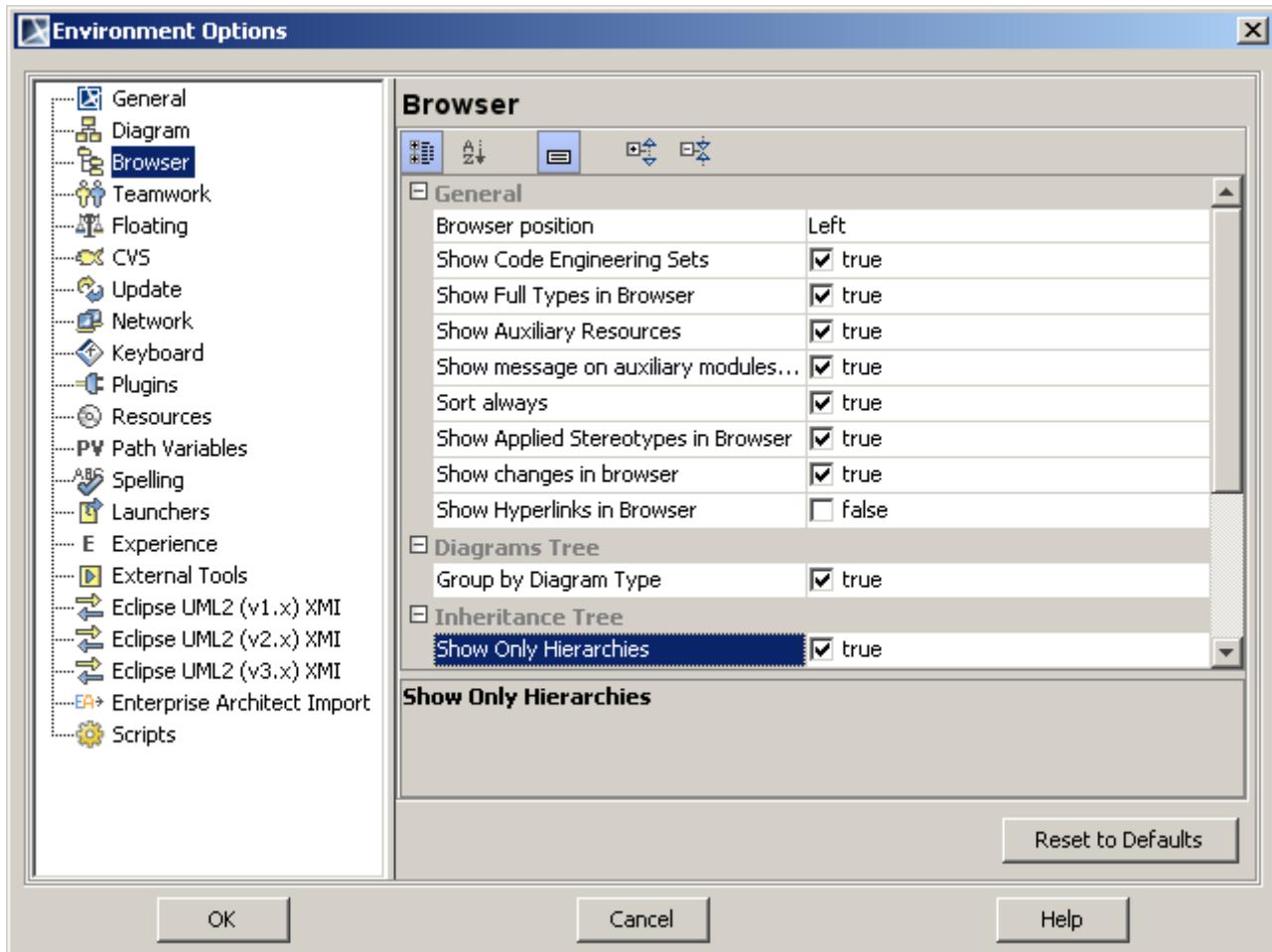


Figure 42 -- Environment Options dialog. Browser options tab

| Property name | Function |
|--|--|
| Browser position | In the drop-down list, select Left if you wish the browser to appear on the left side of the workspace. Select Right if you wish the browser to appear on the right side of the workspace. |
| Show code engineering sets | Shows code engineering set items in the browser. |
| Show full types in browser | Shows the full attributes, operations, and relationships data. |
| Show Auxiliary Resources | Shows/hides the profiles, modules with applied <>auxiliaryResources<> stereotype in the Browser. |
| Show message on auxiliary modules visibility during use | If Show Auxiliary Resources option is false and profiles/modules are not displayed in the Browser, while exporting module, a message will appear. |
| Sort always | Always sort browser items alphabetically. |
| Show applied stereotypes in browser | Shows stereotypes that are assigned to the model elements. |
| Show changes in browser | Highlight edited or added model elements and diagrams in the Browser. |
| Show hyperlinks in browser | Shows hyperlinks that are assigned to the model elements. |

3 USING MAGICDRAW

Customizing Environment Options

| Property name | Function |
|--|--|
| Group by diagram type | Groups diagrams according to diagram type in the Diagrams tree. |
| Show Only Hierarchies | When selected, classifiers without hierarchies are not displayed in Inheritance tree. In this case, Inheritance tree displays only classifiers connected with the generalization relationship. |
| Invert Tree | When selected, hierarchies are inverted so that at the top of the hierarchy, the specific element is displayed. |
| Display warning when element is not found in inheritance tree | Displays warning when element was not found in the Inheritance tree. |
| Group by Profiles | Stereotypes, tagged values, and constraints are grouped by profiles in the Model Extensions Tree. |
| Group by Metaclasses | Groups stereotypes by Metaclass in the Model Extensions Tree. |
| Show diagram zoom slider | Shows zoom slider in the Zoom Control tab. |

Teamwork Options Tab

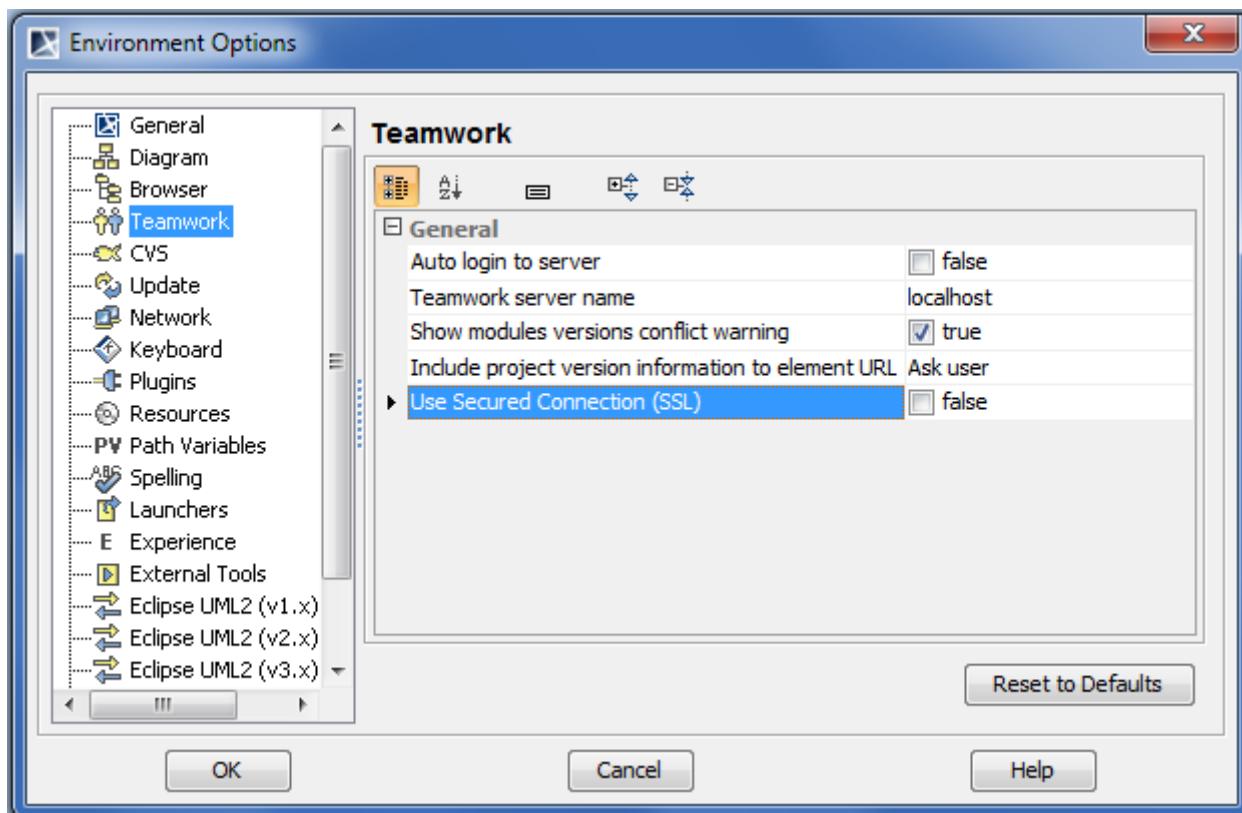


Figure 43 -- Environment Options dialog. Teamwork options tab

| Property name | Function |
|-----------------------------|--|
| Auto login to server | Enables the automatic logging in to Teamwork Server upon starting MagicDraw. |

3 USING MAGICDRAW

Customizing Environment Options

| Property name | Function |
|---|---|
| Include project version information to element URL | You can copy a project element URL to a clipboard and share it with other as a quick reference to model elements. Select option to include project version information to element URL. For more information about Copying/Opening Element URLs, see "Copying/Opening Element URLs" on page 291. |
| Show modules versions conflict warning | Enables the automatic display of a warning message when two modules (or module and a project) use the same submodules. |
| Teamwork server name | Define the name of the Teamwork Server. The name is automatically set once you have logged in to the Teamwork Server. |
| NEW! Use Secured Connection (SSL) | Enables using the secured connection (SSL) automatically, when connecting to the server. For the instructions on how to configure the secured connection in the server side please refer to " MagicDraw Teamwork UserGuide.pdf ". |

Floating Options Tab

This tab is available only if you use FLEXnet license server.

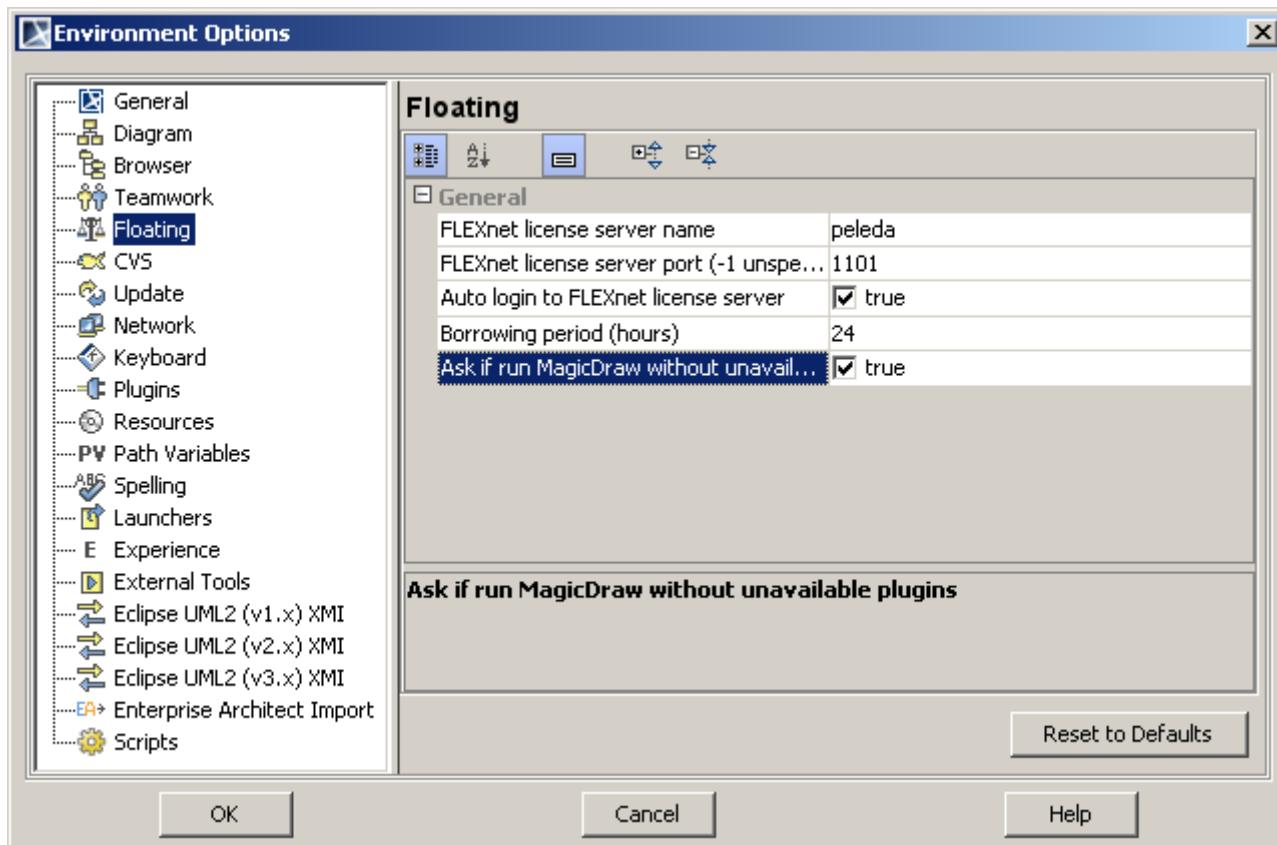


Figure 44 -- Environment Options dialog box. Floating options tab

| Property name | Function |
|------------------------------------|---|
| FLEXnet license server name | Specify the name of the FLEXnet license server. |
| FLEXnet license server port | Enter the port number of the FLEXnet license server. The server can be started on many different ports. |

3 USING MAGICDRAW

Customizing Environment Options

| Property name | Function |
|---|--|
| Auto login to FLEXnet license server | If selected, automatically logs in to the FLEXnet license server, when MagicDraw starts. |
| Borrowing period (hours) | Specify a period of time for borrowing licenses from the server and using them offline. NOTE: The maximum license borrowing period is one year. |
| Ask if run MagicDraw without unavailable plugins | If selected, when starting the application, the user will be asked, if he or she wants to run MagicDraw without so-called unavailable plugins, if there are some. An unavailable plugin is the one, whose license is not available in the FLEXnet license server. |

CVS Options Tab

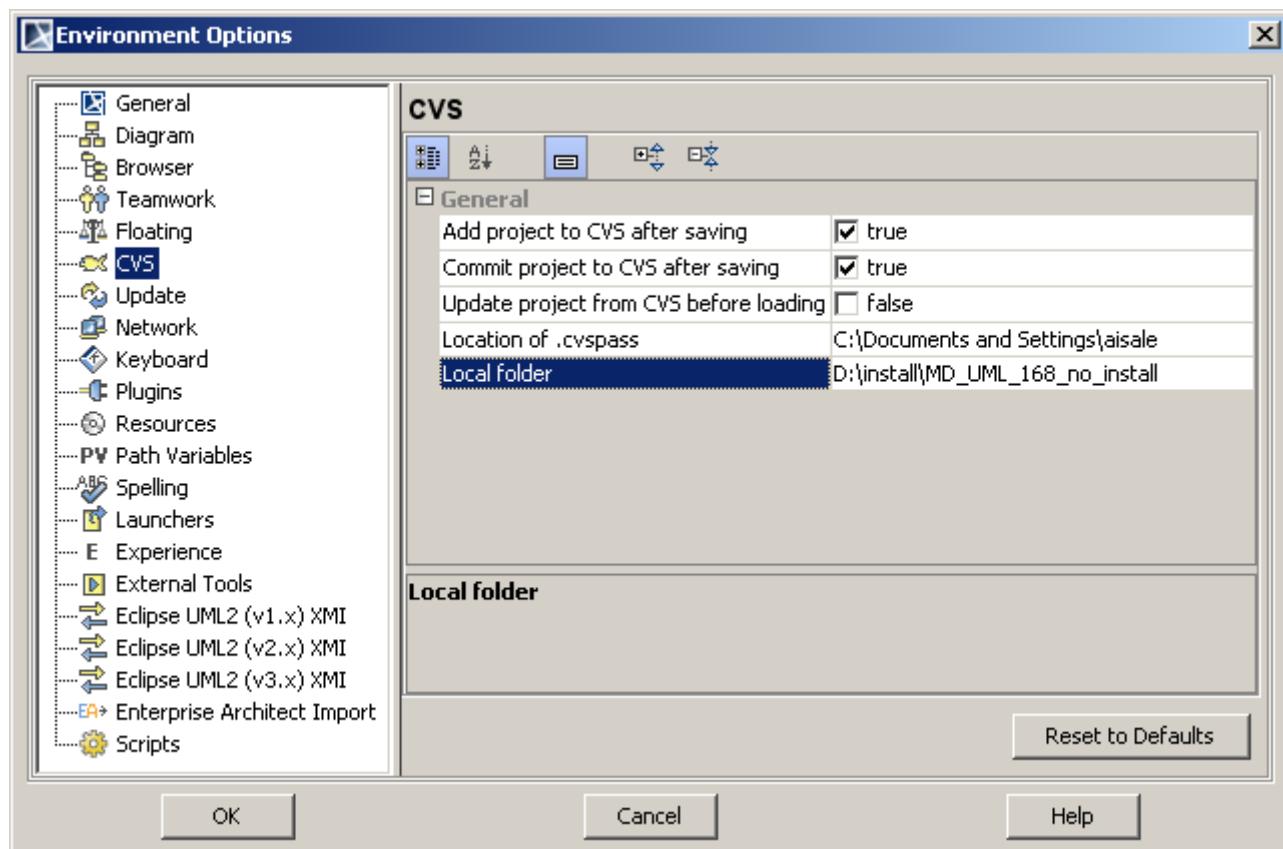


Figure 45 -- Environment Options dialog. CVS options tab

| Property name | Function |
|---|---|
| Add project to CVS after saving | When a new project is saved, it will be added to the CVS checked out directory. The Add Project to CVS dialog box appears. |
| Commit project to CVS after saving | Commits project to CVS after saving it. The Commit Project to CVS dialog box opens. |
| Update project from CVS before loading | Updates the project from CVS before loading. The Update CVS Project dialog box opens. |
| Location of .cvspass | The path where the .cvspass is located. You may type it or choose the path from the Open dialog box. |

3 USING MAGICDRAW

Customizing Environment Options

| Property name | Function |
|---------------|---|
| Local folder | The path where the module will be saved on checkout action. You may type it or choose the path from the Open dialog box. |

Update Options Tab

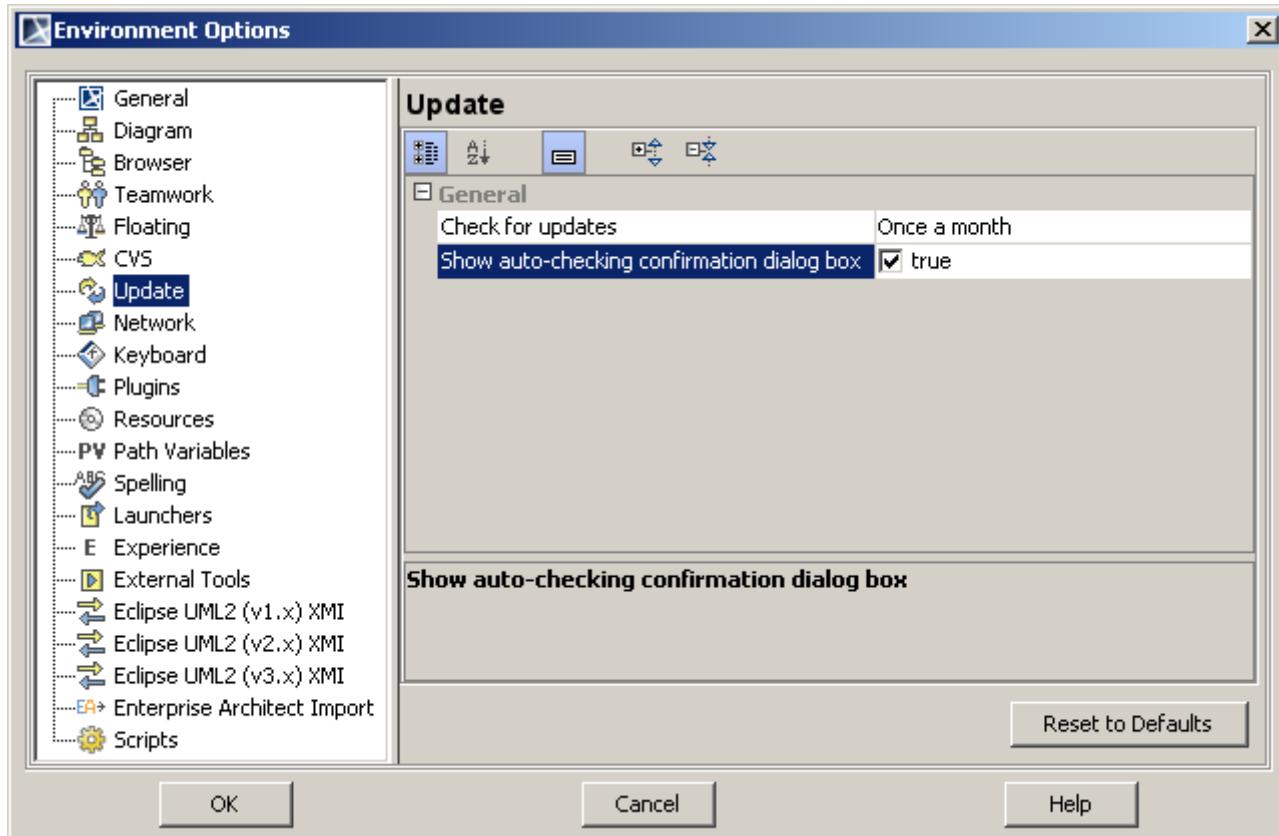


Figure 46 -- Environment Options dialog. Update options tab

| Property name | Function |
|--|---|
| Check for Updates | Select the period for checking MagicDraw for updates: <ul style="list-style-type: none">● Manually● On startup● Once a day● Once a week● Once a month (recommended) |
| Show auto-checking confirmation dialog box | If selected, shows the confirmation dialog each time before auto-checking. |

Network Options Tab

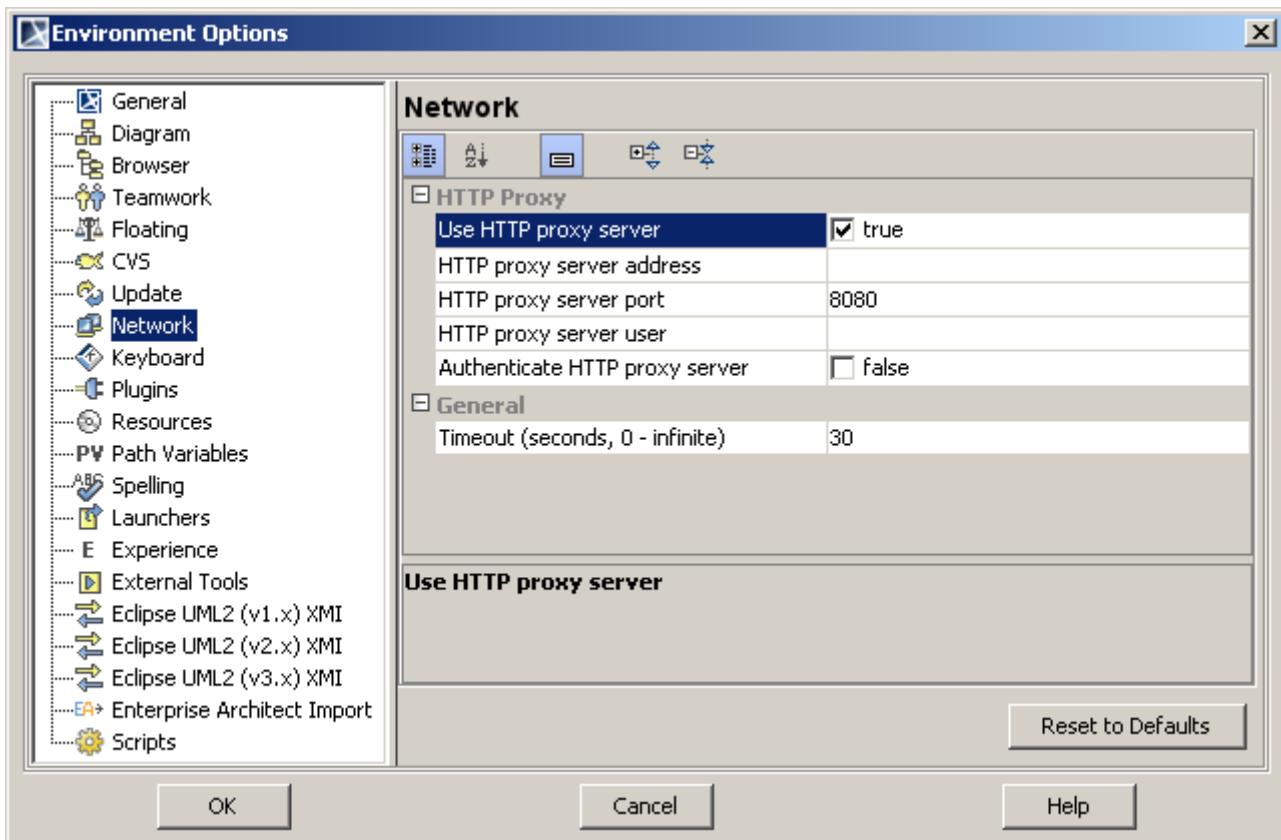


Figure 47 -- Environment Options dialog. Network options tab

| Property name | Function |
|--|---|
| Use HTTP proxy server | If selected, uses the HTTP Proxy Server when MagicDraw is checking for updates. |
| HTTP proxy server address | Type the HTTP Proxy Server name or IP address. |
| HTTP proxy server port | Type the port of the HTTP Proxy Server. The default number is 8080. |
| HTTP proxy server user | Type the name of the HTTP Proxy Server user. |
| Authenticate HTTP proxy server | Authenticates the HTTP Proxy Server. |
| Timeout (seconds, 0 - infinite) | Seconds to wait before canceling a connection attempt. |

Keyboard Options Tab

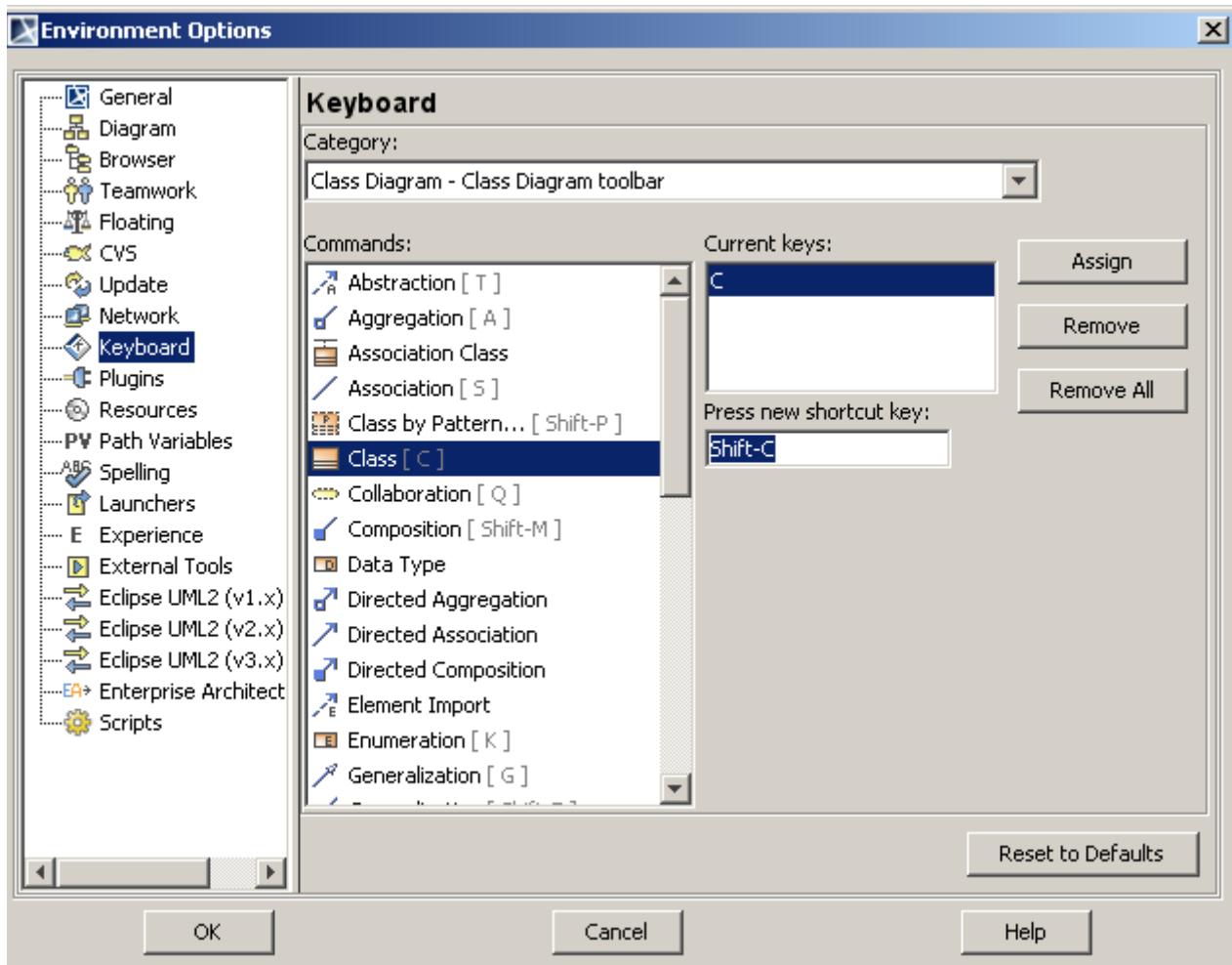


Figure 48 -- Environment Options dialog. Keyboard options tab

Lists and fields available in this tab:

| | |
|-------------------------------|---|
| Category | The list of all available categories of the keys (File, Edit, Tools, etc.). The default item in this choice is always All Categories . Selecting the category opens its content in the Commands list. |
| Commands | The list of commands from the selected category. If All Categories is selected in the Categories list, all commands of an application are shown in this list. Commands are sorted alphabetically. Selecting the command shows the preset shortcut keys in the Current keys list. |
| Current keys | The list of keys assigned to the selected command. |
| Press new shortcut key | The field for entering a new shortcut key. Puts a human representation of any key pressed in the field. If the key pressed is already assigned to another command, the Currently Assigned To label appears with the current command name using that key. |
| Currently assigned to | Shows the name of the command for the new key entered when this key is already assigned to some other command. By default is hidden. This label is only visible if the entered key is already assigned to another command. |

Buttons available in this tab:

| | |
|---------------|--|
| Assign | Assigns the key entered to the selected command. The key entered is added to the Current Keys list. |
|---------------|--|

3 USING MAGICDRAW

Customizing Environment Options

| | |
|-------------------|--|
| Remove | Removes the selected keys from the selected command. |
| Remove All | Removes all keys from the selected command. |

Plugins Options Tab

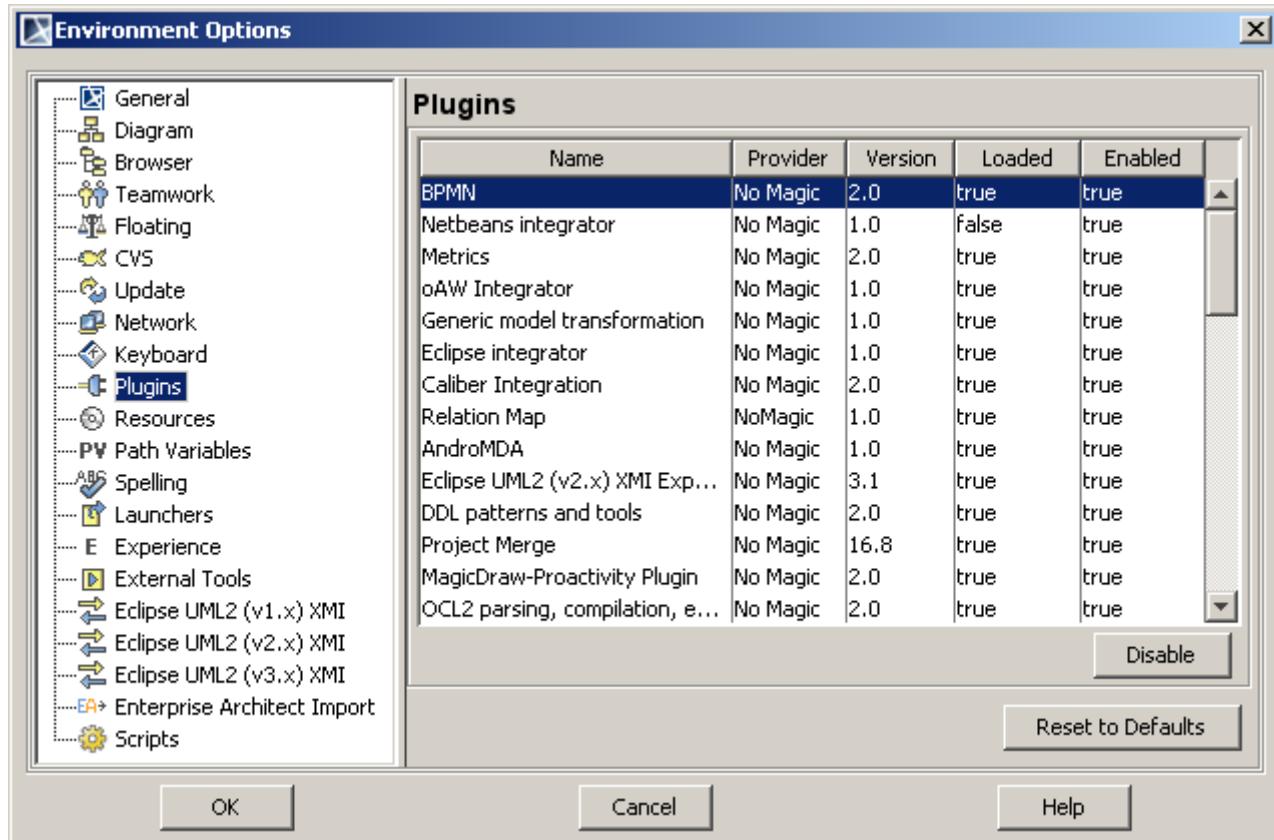


Figure 49 -- Environment Options dialog. Plugins options tab

| Column | Function |
|-----------------|---|
| Name | The name of the plugin. |
| Provider | The name of the company which provides the plugin. |
| Version | Version number of the selected plugin. |
| Loaded | Indicates if the plugin is correctly loaded. |
| Enabled | Shows the status of the plugin. If <i>true</i> , the selected plugin is activated in MagicDraw. |

Buttons available in this tab:

Disable/ Enable Disables or enables the selected plugin.

Resources Options Tab

Displays the external resources such as dtd, xsd, etc. that are used in MagicDraw.

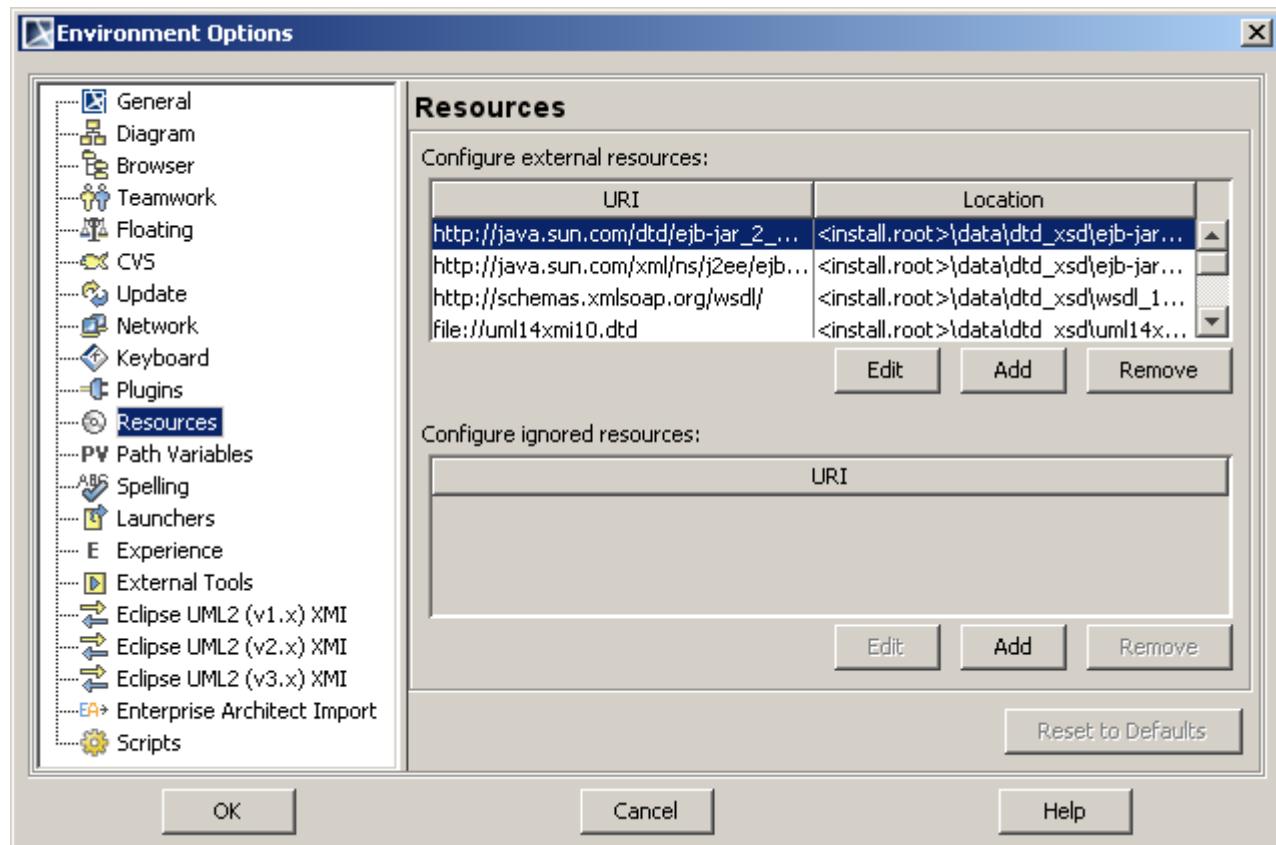


Figure 50 -- Environment Options dialog. Resources options tab

| Property name | Function |
|-------------------------------------|---|
| Configure External Resources | Define external resources that will be used in XML parsing. |
| Location | The location of the external resource. |
| URI | The URI of the external resource. |
| Edit | Edit the location and the URI of the resource in the External Resource dialog box. |
| Add | Add a new external resource. |
| Remove | Remove the selected resource form the list. |
| Configure Ignored Resources | |
| URI | The URI of the external resource that will not be ignored in MagicDraw. |
| Edit | Edit the URI of the ignored external resource in the External Resource dialog box. |
| Add | Add a new external resource that will be ignored. |
| Remove | Remove the selected ignored resource from the list. |

Path Variables Options Tab

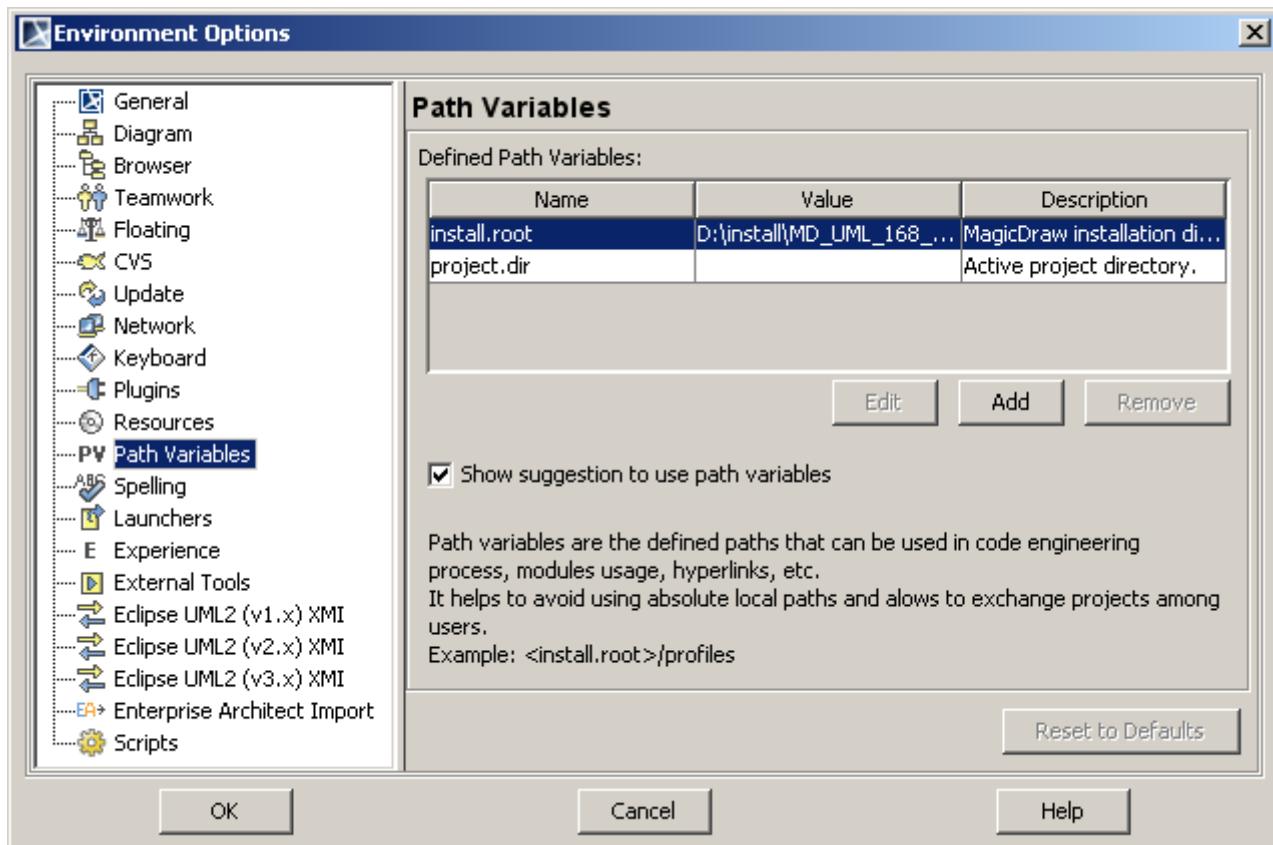


Figure 51 -- Environment Options dialog. Path Variables options tab

| Column | Function |
|--------------------|-----------------------------------|
| Name | Any name for a path variable. |
| Value | Path to the folder. |
| Description | Description of the path variable. |

Buttons available in this pane:

| | |
|--|---|
| Edit | In the list, select an existing path variable. Double click on the item in the list, or click the Edit button. The Path Variables dialog box opens. Edit the path variable. |
| Add | The Path Variables dialog box opens. Specify a name, value, and description for the new path variable. |
| Remove | Remove the selected path variable. |
| Show suggestion to use path variables | In all places where path variables can be used, after specifying absolute path is suggested to use proper path variables. For this, the Use Path Variables dialog box will open. |

Spelling Tab

The description of the **Spelling** tab, you may find in Section “Setting the spell checking options” on page 375.

Launchers Options Tab

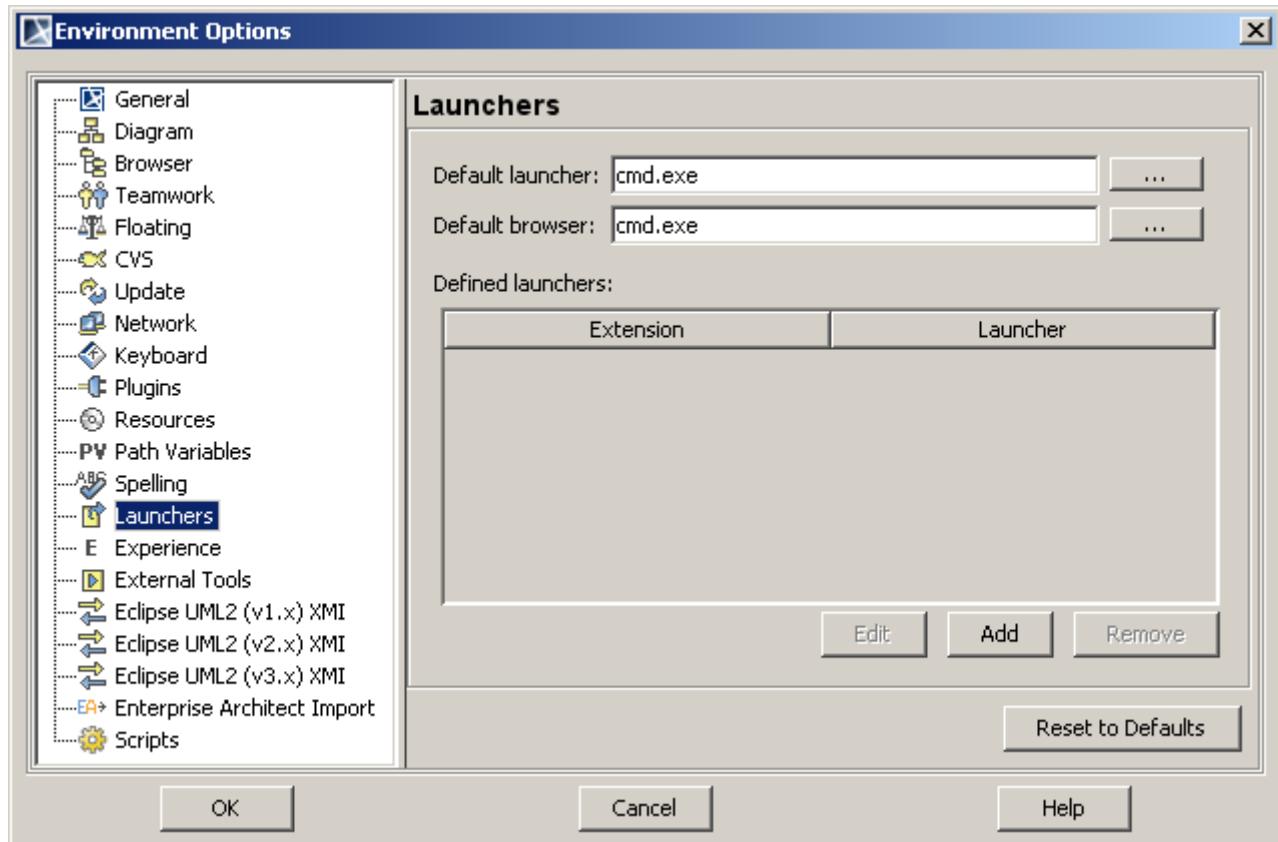


Figure 52 -- Environment Options dialog. Launchers options tab

Fields available in this pane:

| | |
|--------------------------|--|
| Default Launcher | The Launcher Properties dialog opens. |
| Default Browser | The Launcher Properties dialog opens. |
| Defined Launchers | |

Buttons available in this pane:

| | |
|---------------|---|
| Edit | Opens the Launcher Properties dialog box for editing launcher information. |
| Add | Adds a launcher in the Launcher Properties dialog box. |
| Remove | Removes a launcher from the list. |

Experience Options Tab

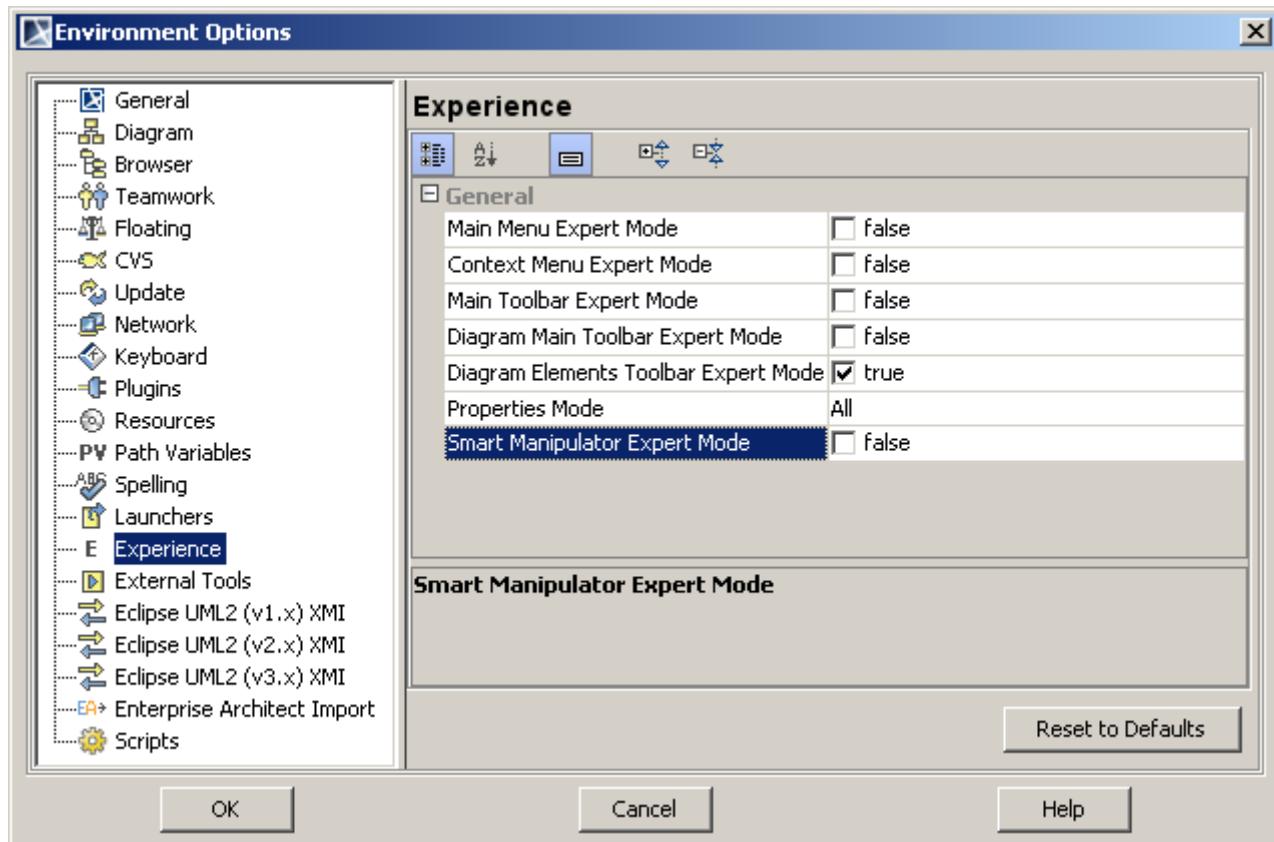


Figure 53 -- Environment Options dialog. Experience options tab

| Property name | Function |
|---|--|
| Main Menu Expert Mode | Toggles visibility of the main menu by adding/removing arrows to expand menus. |
| Context Menu Expert Mode | Toggles visibility of the context menu actions by adding/hiding them, depending on which mode they were set to appear. |
| Main Toolbar Expert Mode | Toggles visibility of the toolbar menu buttons by adding/hiding them depending on which mode they were set to appear. |
| Diagram Main Toolbar Expert Mode | Toggles visibility of the diagram toolbar menu buttons by adding/hiding them depending on which mode they were set to appear. |
| Diagram Elements Toolbar Expert Mode | Toggles visibility of the diagram elements toolbar menu buttons by adding/hiding them depending on which mode they were set to appear. |
| Properties Mode | Possible choices: <ul style="list-style-type: none"> • Standard • Expert • All |
| Smart Manipulator Expert Mode | Toggles visibility of buttons in the smart manipulators menu depending on which mode they were set to appear. |

External Tools Options Tab

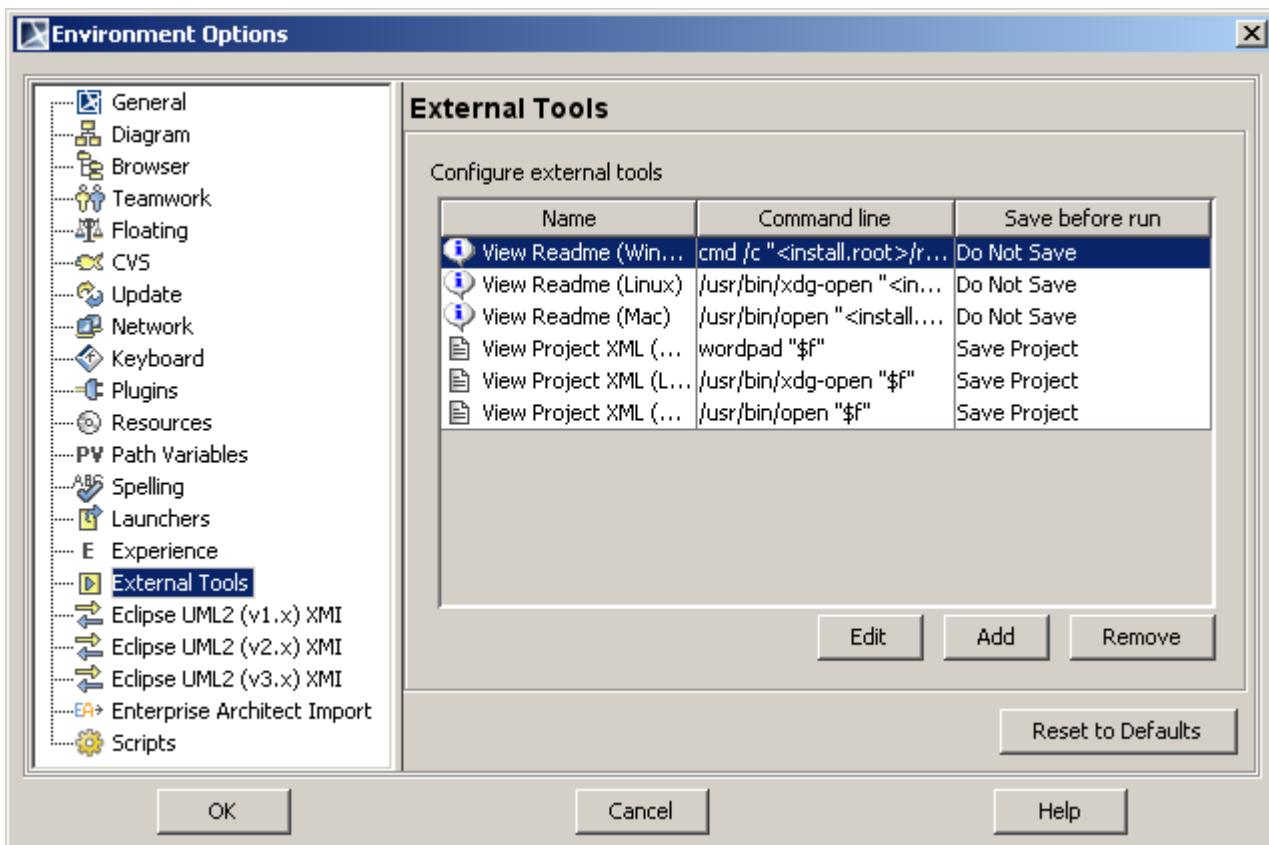


Figure 54 -- Environment Options dialog. External tools options tab

| Column | Function |
|------------------------|---|
| Name | Name of the command, which will appear in the external toolbar. |
| Command line | The command line is executed upon selection. The command line can include the project file name variable, \$project_file\$, if it depends on the MagicDraw project file name. |
| Save before run | Actions, which can be made with a project (save/export) before starting external tools that use a MagicDraw project file. Possible choices: <ul style="list-style-type: none"> ● Save project. ● Do not save. ● Eclipse UML2 (v1.x) XMI. ● Eclipse UML2 (v2.x) XMI. ● Eclipse UML2 (v3.x) XMI. |

Buttons available in this pane:

| | |
|---------------|--|
| Edit | The External Tool dialog opens. You may change the name, icon, command line text, and project save/export options here. |
| Add | The External Tool dialog box opens for new external tool configuration. |
| Remove | Removes the selected external tool from the list. |

Eclipse UML2 (v1.x) XMI Options Tab

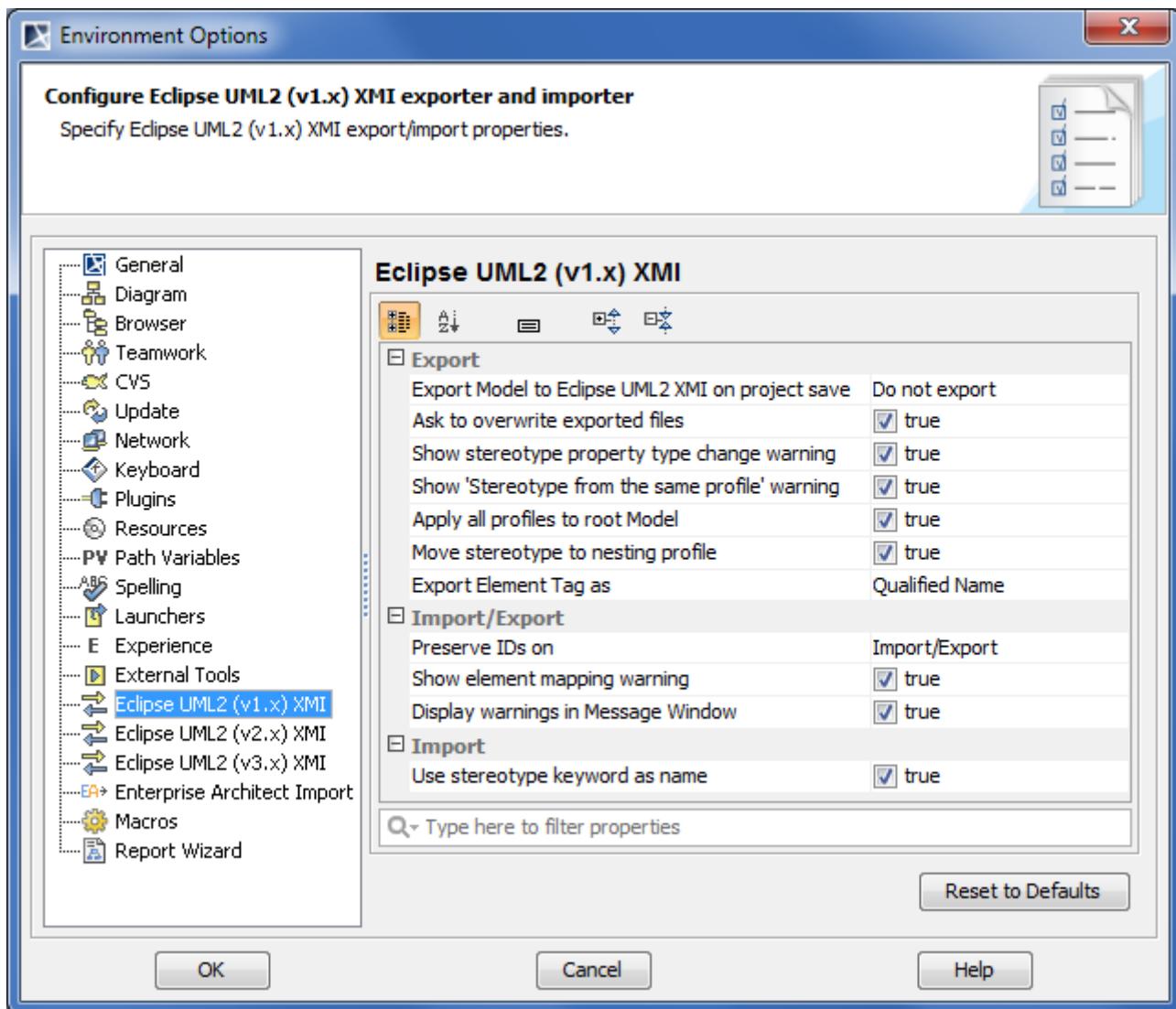


Figure 55 -- Environment Options dialog. Eclipse UML2 (v1.x) XMI options tab

| Property name | Function |
|---|--|
| Export | |
| Export Model to Eclipse UML2 XMI on project save | Exports MagicDraw models (without diagrams) to UML2 format. Possible choices: <ul style="list-style-type: none">• Do not Export• Ask Before Export• Always Export |
| Ask to overwrite exported files | If selected, a message is displayed for requesting confirmation to overwrite previously exported files. |
| Show stereotype property type change warning | If selected, the warnings will appear when exporting a model to the Eclipse UML2 XMI file after the stereotype property type has been changed. |
| Show 'Stereotypes from the same profile' warning | If selected, the warnings that the stereotype will not be applied to the element, if this element is in the same profile with the stereotype, will appear when exporting. |

3 USING MAGICDRAW

Customizing Environment Options

| Property name | Function |
|---|--|
| Apply all profiles to root Model | If selected, all the exported profiles will be applied to the root model. If cleared, all the exported profiles will be applied to the nearest nesting package. |
| NEW! Add stereotype to nesting profile | If selected, all stereotypes that are not directly contained by a profile are moved to the nesting profile, when exporting a model to an Eclipse UML2 (v1.x) XMI. |
| Export Element Tag as | Possible choices: <ul style="list-style-type: none">● Qualified Name● Name● ID |
| Import/ Export | |
| Preserve IDs on | <ul style="list-style-type: none">● Do not preserve● Export● Import● Export/Import |
| Show element mapping warning | If selected, the warnings showing the elements' (metaclasses and data types) mappings between MagicDraw and Eclipse UML2 will appear when exporting or importing a model. |
| Display warnings in Message Window | If selected, warnings related to project export will be listed in the Message Window. |
| Import | |
| Use stereotype keyword as name | In the Eclipse UML2 models, the stereotypes have both names and keywords. If selected, the keywords will be used as names for the stereotypes, when importing a model from the Eclipse UML2. |

3 USING MAGICDRAW

Customizing Environment Options

Eclipse UML2 (v2.x / v3.x) XMI Options Tab

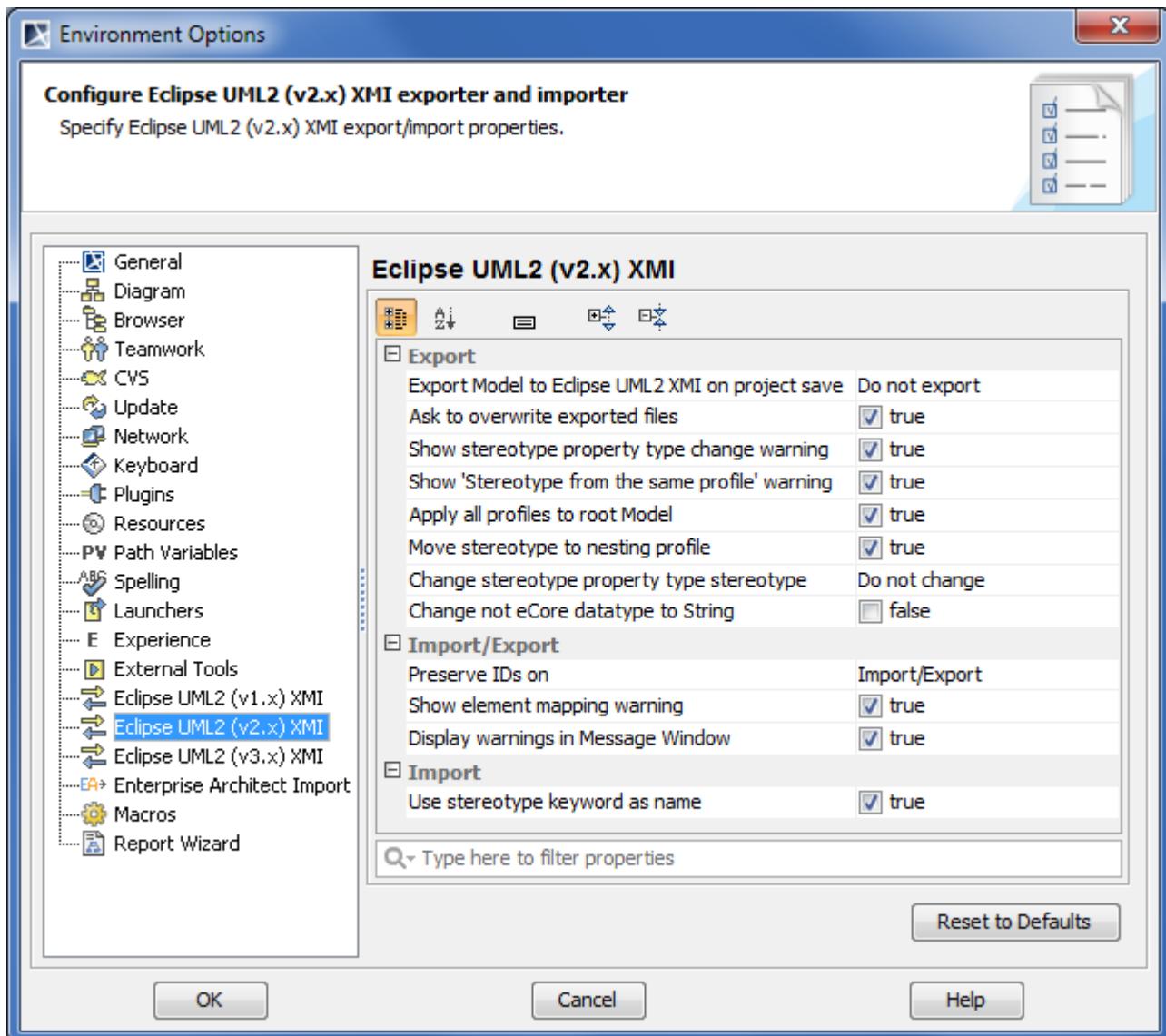


Figure 56 -- Environment Options dialog. Eclipse UML2 (v2.x / v3.x) XMI options tab

| Property name | Function |
|---|--|
| Export | |
| Export Model to Eclipse UML2 XMI on project save | Exports MagicDraw models (without diagrams) to UML2 format. Possible choices: <ul style="list-style-type: none">• Do not Export• Ask Before Export• Always Export |
| Ask to overwrite exported files | If selected, a message is displayed for requesting confirmation to overwrite previously exported files. |
| Show stereotype property type change warning | If selected, the warnings will appear when exporting a model to the Eclipse UML2 XMI file after the stereotype property type has been changed. |
| Show 'Stereotypes from the same profile' warning | If selected, warnings that the stereotype will not be applied to the element, if this element is in the same profile with the stereotype, will appear when exporting. |

3 USING MAGICDRAW

Customizing Environment Options

| Property name | Function |
|---|---|
| Apply all profiles to root Model | If selected, all the exported profiles will be applied to the root model. If cleared, all the exported profiles will be applied to the nearest nesting package. |
| NEW! Add stereotype to nesting profile | If selected, all stereotypes that are not directly contained by a profile are moved to the nesting profile, when exporting a model to an Eclipse UML2 (v2.x / v3.x) XMI. |
| Change stereotype property type stereotype | Specifies changing the stereotype's property type's stereotype on model export. Select one of the following options: <ul style="list-style-type: none">● Do not change● Single extended metaclass only● Common super of extended metaclasses |
| Change not eCore datatype to String | Changes the stereotype's property type to String on export, if the type is not eCore data type. |
| Import/ Export | |
| Preserve IDs on | <ul style="list-style-type: none">● Do not preserve● Export● Import● Export/Import |
| Show element mapping warning | If selected, the warnings showing the elements' (metaclasses and data types) mappings between MagicDraw and Eclipse UML2 will appear when exporting or importing a model. |
| Display warnings in Message Window | If selected, warnings related to project export will be listed in the Message Window. |
| Import | |
| Use stereotype keyword as name | In the Eclipse UML2 models, the stereotypes have both names and keywords. If selected, the keywords will be used as names for the stereotypes, when importing a model from the Eclipse UML2. |

Macros Options Tab

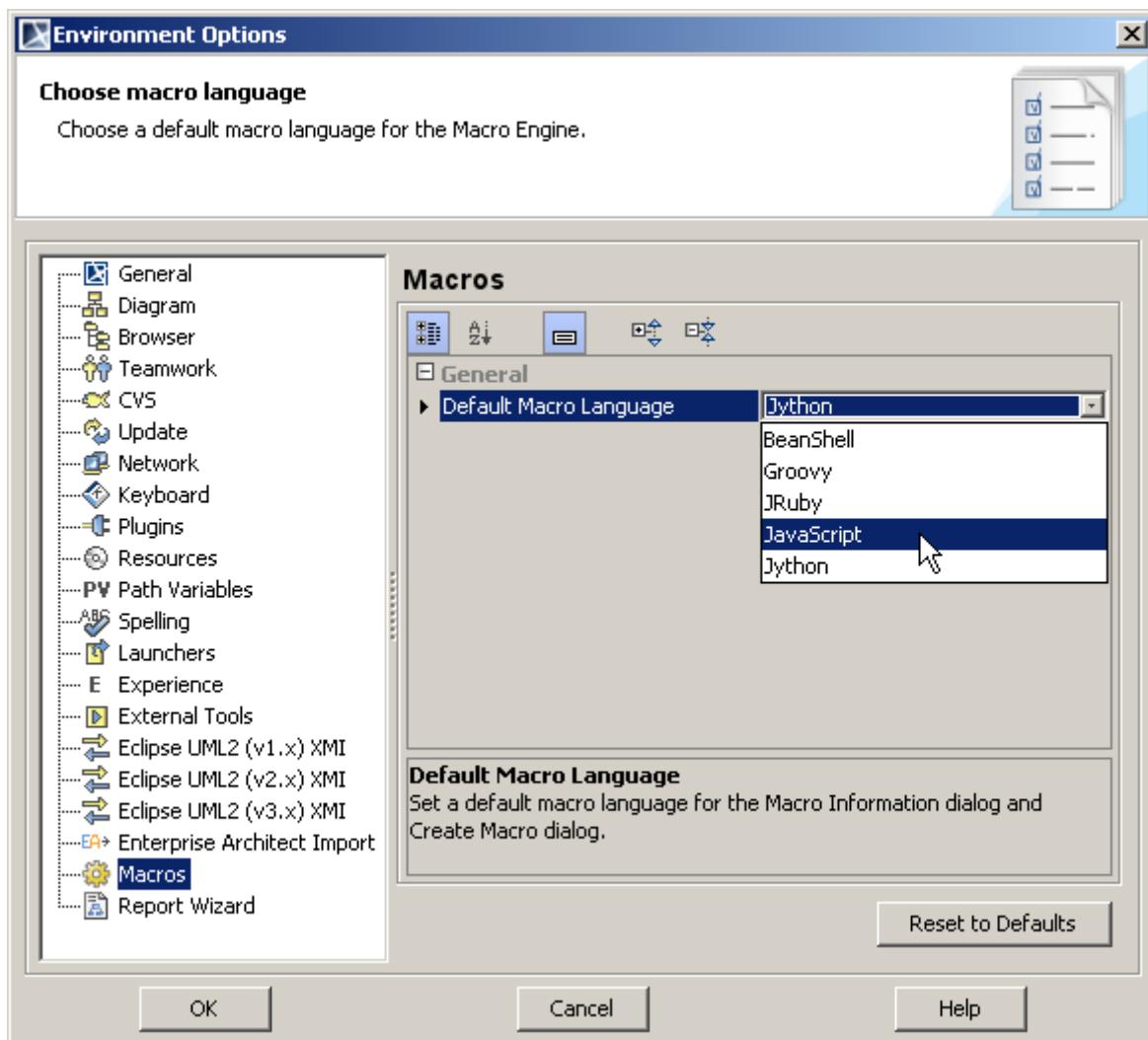


Figure 57 -- The Environment Options dialog. Macros options tab

| Property name | Function |
|-------------------------------|--|
| Default Macro Language | The default macro language for the Macro Engine. |

For detailed information about Macro Engine, see "[MagicDraw MacroEngine UserGuide.pdf](#)".

Report Wizard Options Tab

The Report Wizard environment options allow configuring the report engine. There are three things that you can configure:

1. Report engine properties.
2. Template mappings.
3. Template folder.

3 USING MAGICDRAW

Customizing Environment Options

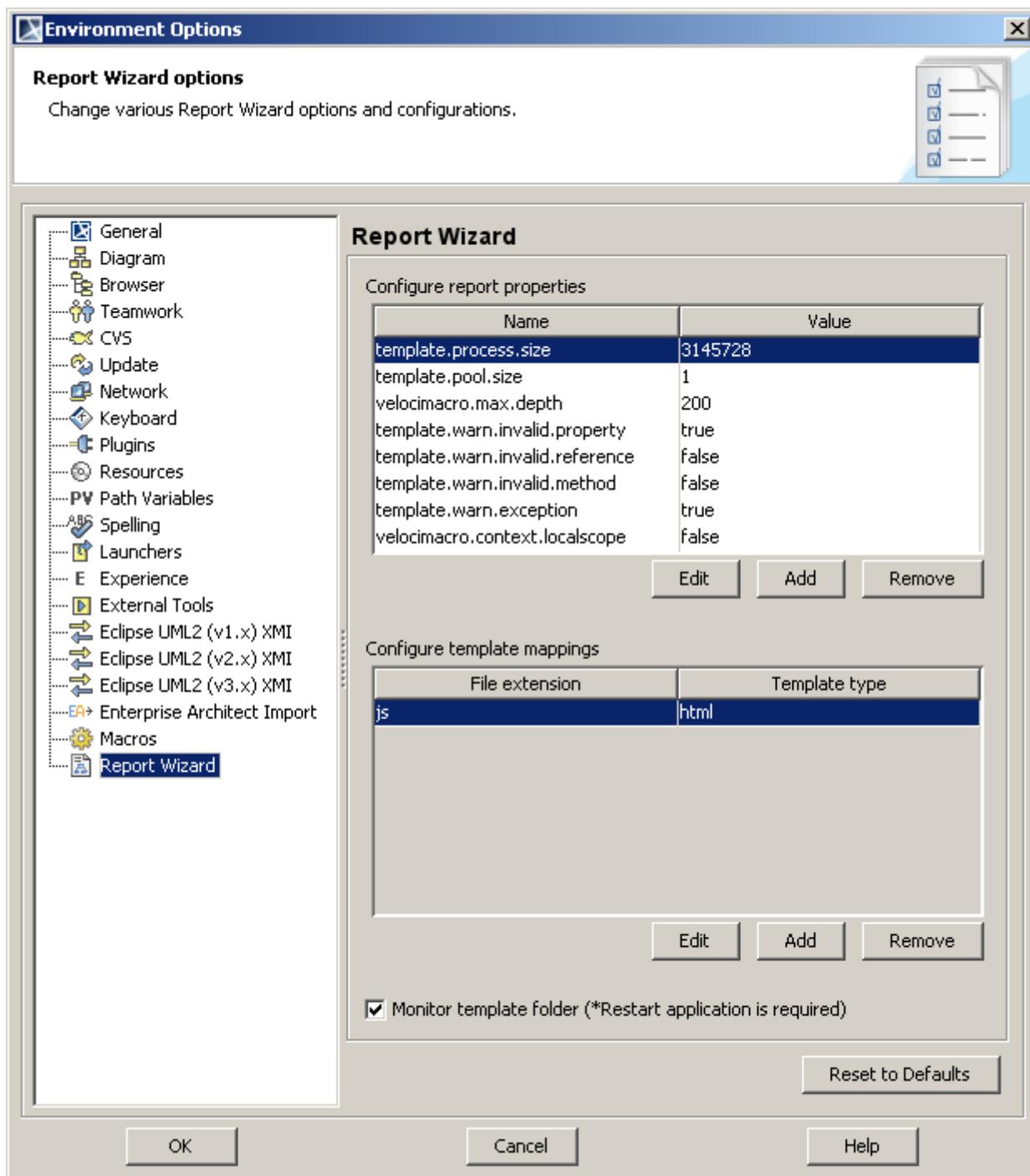


Figure 58 -- Environment Options dialog. Report Wizard options tab

1. Configuring report engine properties

You can configure report engine properties, such as template process size, template pool size, various velocity and debug report template properties.

Columns available in the **Configure report properties** table:

| Column | Function |
|--------|--------------------------------|
| Name | A report engine property name. |

3 USING MAGICDRAW

Customizing Environment Options

| Column | Function |
|--------|---------------------------------|
| Value | A report engine property value. |

Buttons associated with the **Configure report properties** table:

| Button | Function |
|--------|--|
| Edit | Opens the Report Properties dialog to edit the selected report engine property. |
| Add | Opens the Report Properties dialog to create a new report engine property. |
| Remove | Removes the selected report engine property from the list. |

2. Configuring template mappings

Template mappings associate a template type with the template engine. The report engine allows you to determine how to handle different types of files, such as JS or DOCBOOK file. All you have to do is specify a file-name extension and select an engine name. The report engine also allows developers to add a user custom engine into this option.

Columns available in the **Configure template mappings** table:

| Column | Function |
|----------------|--|
| File extension | A filename extension of the template mapping. IMPORTANT! The filename extension is case-insensitive. |
| Template type | A template type of the template mapping. IMPORTANT! The template type is case-sensitive. |

Buttons associated with the **Configure template mappings** table:

| Button | Function |
|--------|--|
| Edit | Opens the Template Mappings dialog to edit the selected template mapping. |
| Add | Opens the Template Mappings dialog to create a new template mapping. |
| Remove | Removes the selected template mapping from the list. |

3. Monitor template folder option

You can either make available or make unavailable the MRZIP template file automatic deployment option by selecting or clearing the **Monitor template folder** check box.

For detailed information about Report Wizard options and configurations, see section “10. Report Wizard Environment Options” in [“MagicDraw ReportWizard UserGuide.pdf”](#).

Look and Feel: Controlling the Interface

The appearance of MagicDraw windows, dialog boxes, menus, and everything inside them can be changed. The **Look and Feel** submenu allows you to personalize the user interface of the MagicDraw. You can set your favorite colors and fonts.

To make changes to the interface:

- From the **Options** menu, choose **Look and Feel** and then choose the style you wish to apply.

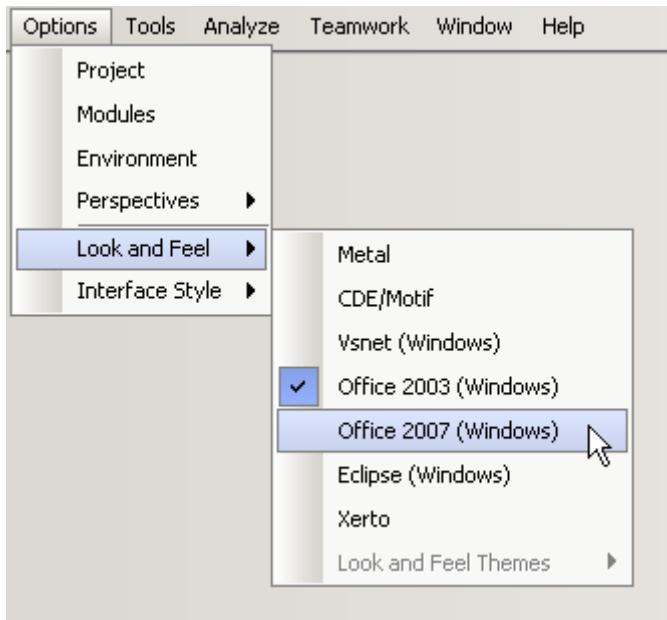
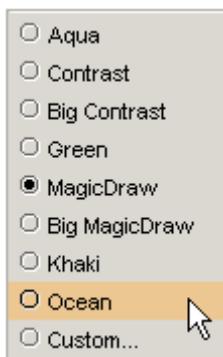


Figure 59 -- Interface styles for MagicDraw

The chosen style will not look exactly like the applications in those operating systems because every style of graphical interface is implemented within a Swing library, but it will look quite similar.

Depending on the operating system you use, some choices might be unavailable for you. For example, Windows9x/NT users may not switch to the Mac interface style.

Note that themes listed in **Look and Feel Themes** are valid only for the **Metal** style. You can choose any of the following themes:



After choosing the Custom theme, the **Properties** dialog for setting your own options will be opened.

Single and Multiple Windows interface styles

Beginning with MagicDraw version 7.5, the modern JIDE library is implemented (called Single Window interface style). Using the **JIDE** interface style, it is possible to work with the Browser window in a more flexible way, use documentation, zoom, and dock message windows above the main window. You can arrange the Browser window in combinations or even hide the desired Browser windows.

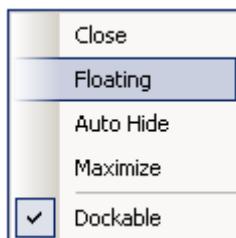
Also, you may use the different **Multiple Windows** style. It allows you to manage all windows independently, as if they belong to different applications. For instance, you can overlap the windows across each other, resize them independently, and so on. There is no main window containing all the other windows.

To set the interface style

- From the **Options** menu, select **Interface Style**, and then select one of the desired interface styles: **Single Window** or **Multiple Windows**.

To make the Model Browser a separate window

- From the Model Browser shortcut menu, select **Floating** and move the window to any desired position.



NOTE

If the **Dockable** check box is selected, the floating tab window will appear in a fixed edge position after trying to move it outside the MagicDraw window borders.

Assigning Shortcut Keys

To assign or change a command shortcut key

- From the **Options** menu, select **Environment**.
- The **Environment Options** dialog box opens.
- Select the **Keyboard** tab and assign the desired shortcut keys in the right pane of the dialog box.

For more information about the **Keyboard** tab, see “Keyboard Options Tab” on page 103.

4 WORKING WITH PROJECTS

The term “project” is used to describe the problem that must be solved, including all the possible solutions for how the problem can be resolved and finally developed. All work in MagicDraw™ UML is organized into projects. Project is the top entity where all model-related data (the set of diagrams) is held. Project data is organized by object orientation, which makes its management intuitive and in accordance with the problem that is being solved.

In this Section, you will find the following chapters:

1. "Creating a Project", on page 119.
2. "Saving a Project", on page 125
3. "Opening a Project", on page 127
4. "Importing a Project", on page 127
5. "Exporting a Project", on page 127
6. "Setting Project Options", on page 132
7. "Searching", on page 134
8. "Project Partitioning", on page 141
9. "Exporting a Project as MOF" on page 154
10. "Ecore Support" on page 157
11. "Working with Standard Profiles" on page 165

Creating a Project

[View Online Demo](#)

MagicDraw Basics

Creating a new project

All project information is stored in a single file. A project name matches the file name where the project is saved.

The newly created project consists of the following packages:

- **Data** package is empty and holds all model elements.
- **File View** package contains components that are created during code engineering and represent source files. Adding a resident element to a particular component causes that element to be generated within the source file.
- **UML Standard Profile** contains stereotypes that are necessary for working with MagicDraw, primitive data types and constraints (which are UML standard), and UML 2 metamodel elements. The following data types are specified in MagicDraw: boolean, byte, char, date, double, float, int, Integer, long, short, void, and string.

You can also create your own packages for holding the model elements. By default, packages cannot be deleted or renamed in a project (except for the File View package).

To start a new project, you must create a new workspace for it.

You can select a project type from the following domains:

4 WORKING WITH PROJECTS

Creating a Project

- **General-Purpose Modeling** (UML, Use Case, Guide to UML Diagrams projects, Project from Existing Source Code)
- **System Engineering** (SyML project)
- **Enterprise Modeling** (DoDAF, DoDAF 2.0, MODAF projects)
- **Business Process Modeling** (BPMD2 project)
- **Service-Oriented Modeling** (Cameo SOA+ project)
- **Other** (Project from Template, Process Guide project)

On a new project creation the **General-Purpose Modeling** domain opens by default.

To create a new workspace for a blank project

- From the **File** menu, select **New Project**.
- On the main toolbar, click the **New Project** button.
- Press shortcut key **CTRL+N**.

In all cases, the **New Project** dialog box opens.

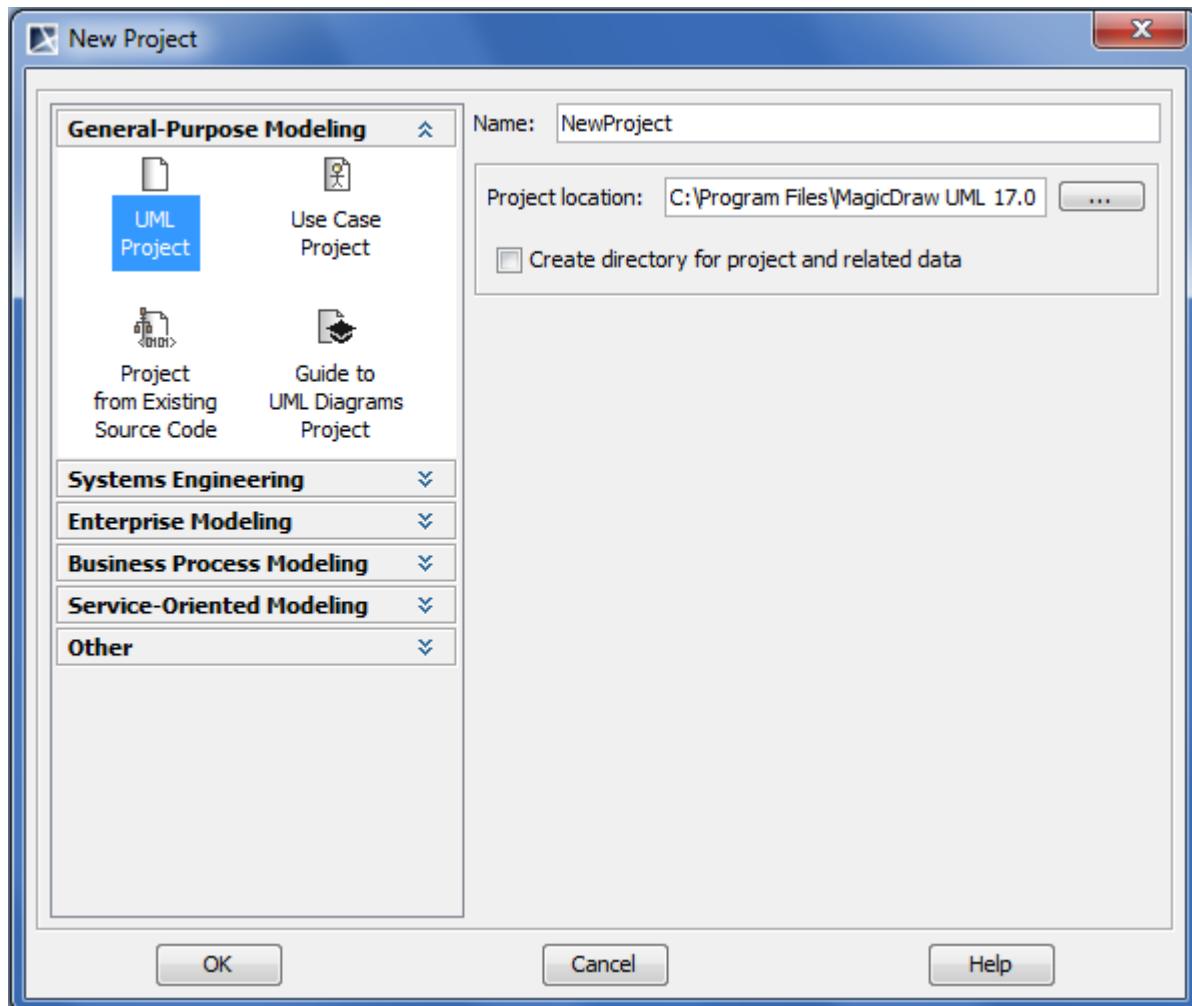


Figure 60 -- New Project dialog box

1. Select the **UML Project** icon in the **General-Purpose Modeling** domain.
2. Specify the file name in the **Name** text box.
3. Click the “...” button to select the location to store a newly created project in your computer.
Click **OK**.

Working with multiple projects

Because you may need to manage several projects at the same time, MagicDraw allows you to work with several projects simultaneously.

All open projects are held in separate workspaces. Different active projects may exchange data. Entities from one project can be copied or moved to another.

To switch between loaded projects

- In the **Projects** drop-down combo box, click the additional project you wish to open.
- Select **Projects** from the **File** menu, click the name of the project you wish to open.

To close all open projects

Select **Close All Projects** from the **File** menu. The **Question** message box appears.



Figure 61 -- Question message box

Choose the way your projects will be closed:

| | |
|-------------------|---|
| Yes | The project you are currently closing will be saved (its name appears in the question). The dialog box is displayed again when the next project closes. |
| Yes To All | Save all projects without prompting. The Save dialog box will not appear for each open project. |
| No | Project you are currently closing will not be saved. The dialog box is displayed again when the next project closes. |
| No To All | All the projects will be closed without saving or further prompting. |
| Cancel | Cancel saving projects. |

To exchange model entities between open projects

- Use the **Cut**, **Copy**, and **Paste** commands in the **Edit** menu, or the appropriate shortcut keys: Ctrl+X, Ctrl+C, Ctrl+V or the toolbar buttons.
- Drag-and-drop the created model element from the Browser tree to the Diagram pane.

NOTE Data may only be exchanged between projects that are currently open within MagicDraw. You may not copy/paste elements between instances of different tools that are currently running or to other applications.

Creating a new project from the existing source code

To create a new project from existing source files

- From the **File** menu, select **New Project**.
- On the main toolbar, click the **New Project** button.
- Press shortcut key **CTRL+N**.

In all cases, the **New Project** dialog box opens.

- Select the **New Project from Existing Source** icon.
- Specify the file name in the **Name** text box.
- Click the “...” button to select the location to store a newly created project in your computer.
- Select a code engineering language from the list by clicking the “...” button and click **OK**.

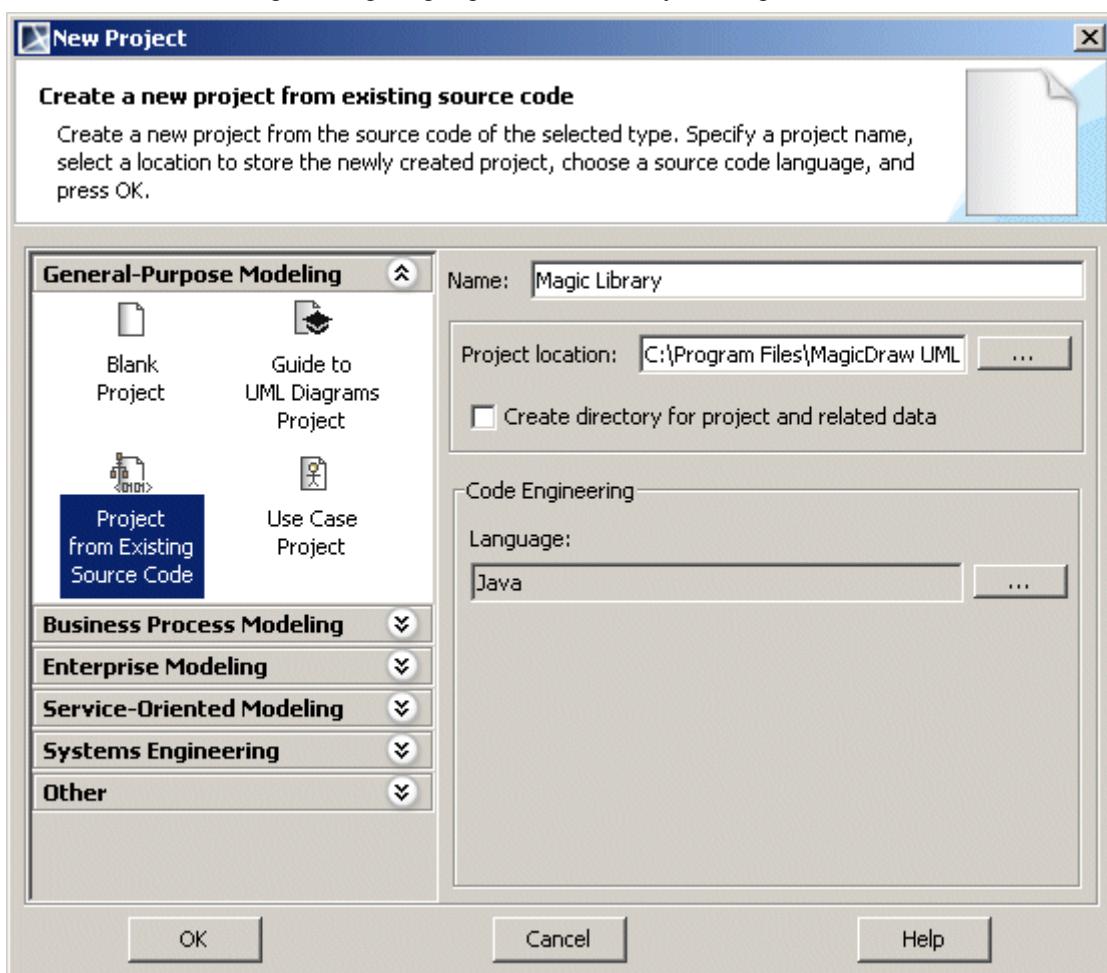


Figure 62 -- New Project dialog box - New Project from Existing Source

- The **Round Trip Set** dialog box opens. Add the source files to enable code engineering to reverse them into a newly created project.

Creating a new project from a previously created template

NOTE: This functionality is available in Standard, Professional, Architect and Enterprise editions only.

C++, CIL, Java, C#, DDL, EJB, WSDL, XML Schema, Metamodeling, RUP extensions, CORBA IDL, and UML-WebExtension templates are available in the **New Project** dialog box.

4 WORKING WITH PROJECTS

Creating a Project

To create a new project from a specified template

- From the **File** menu, select **New Project**.
- On the main toolbar, click the **New Project** button.
- Press shortcut key **CTRL+N**.

In all cases, the **New Project** dialog box opens.

- Select the **New Project from Template** icon.
- Specify the file name in the **Name** text box.
- Click the “...” button to select the location to store a newly created project in your computer.
- Select the template from the templates tree and click **OK**.

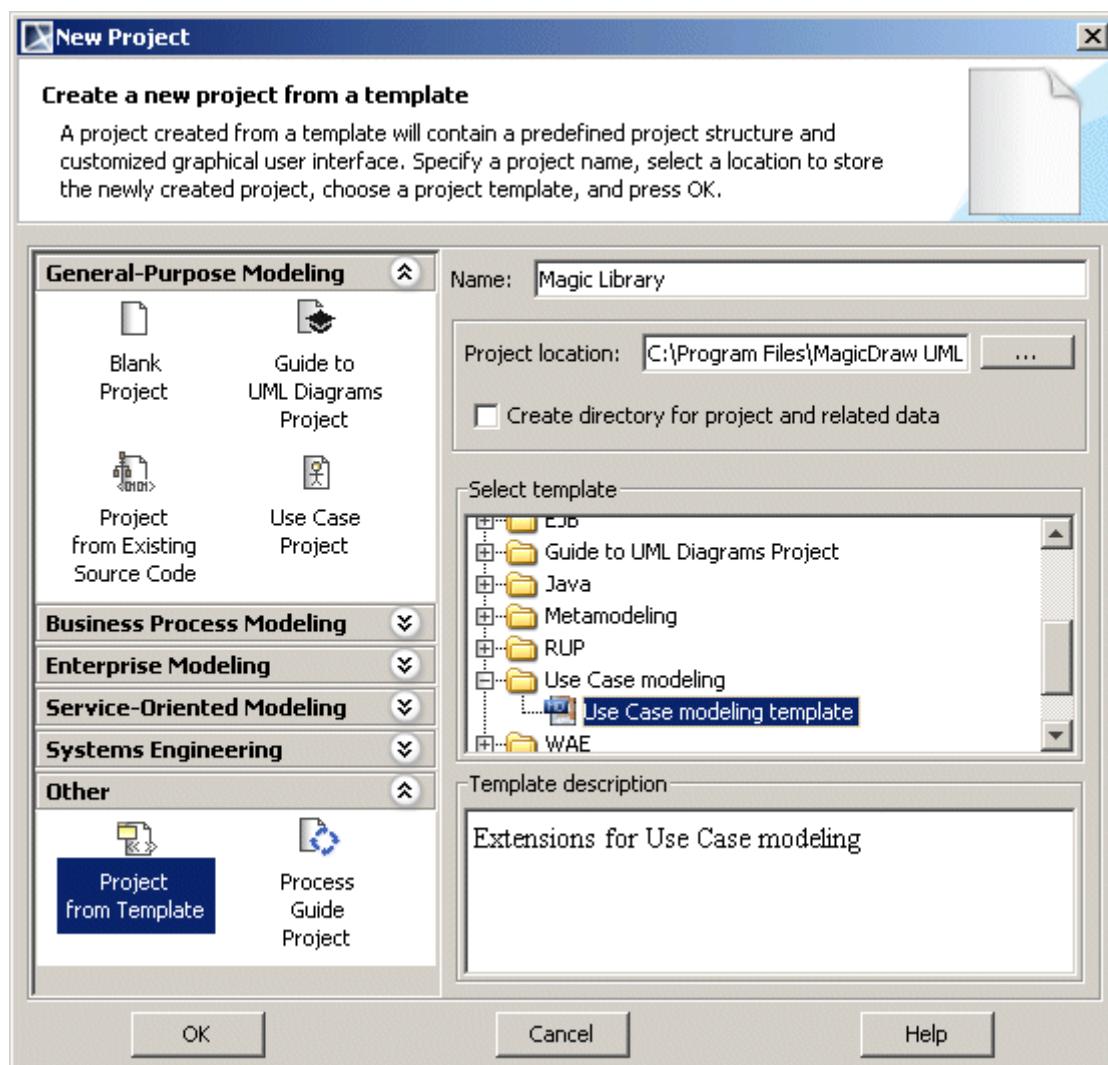


Figure 63 -- New Project dialog box - New Project from Template

The newly created project from a template will contain specific model elements and stereotypes.

TIP!

All MagicDraw templates are located in the <MagicDraw installation directory>/templates folder so you can import the desired template into your previously created project using the **Import MagicDraw Project** command from the **File** menu.

Creating a new Use Case project

To create a new project from existing source files

- From the **File** menu, select **New Project**.
- On the main toolbar, click the **New Project** button.
- Press shortcut key **CTRL+N**.

In all cases, the **New Project** dialog box opens.

1. Select the **Use Case Project** icon.
2. Specify the file name in the **Name** text box.

Click the “...” button to select the location to store a newly created project in your computer.

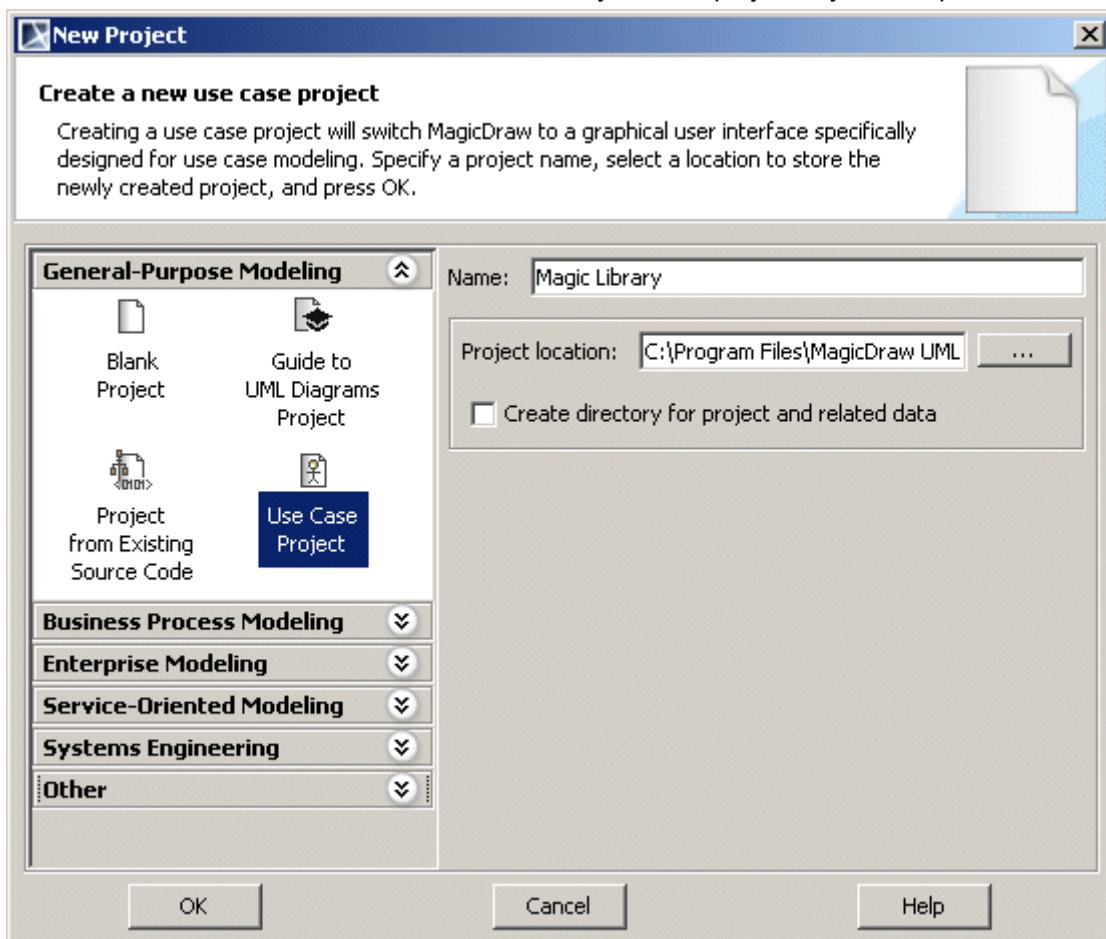


Figure 64 -- New Project dialog box - Use Case Project

The newly created project will automatically load the **UseCase Description Profile**. Also **Actor**, **High-Level Use Case** and **System-Level Use Case** packages will be created in the Data tree. Additional properties will be displayed in the newly created use cases **Specification** dialog box.

Saving a Project

IMPORTANT! The native MagicDraw format is *.mdzip and *.mdxml. Saving in *.xml, *.xml.zip format will also be allowed.

To save changes for later sessions, revised projects must be saved. While saving, you can edit the name of the project and the file format.

To save the project

1. From the **File** menu, select the **Save Project** or **Save Project As** command. Alternatively, you can click the **Save** button on the main toolbar or press the shortcut keys CTRL+S. The **Save** dialog box opens.
2. Select the destination directory (where you wish to save the project) and type the chosen file name.

NOTE

Default project save location is set according to your OS:

- Windows: My Documents/
- Linux: /home/<username>/
- Mac: Users/Documents/

4 WORKING WITH PROJECTS

Saving a Project

3. Select the format for saving a project: **Packed MagicDraw File Format (*.mdzip)** (default), **MagicDraw File Format (*.mdxml)**, or **XML (*.xml)**.

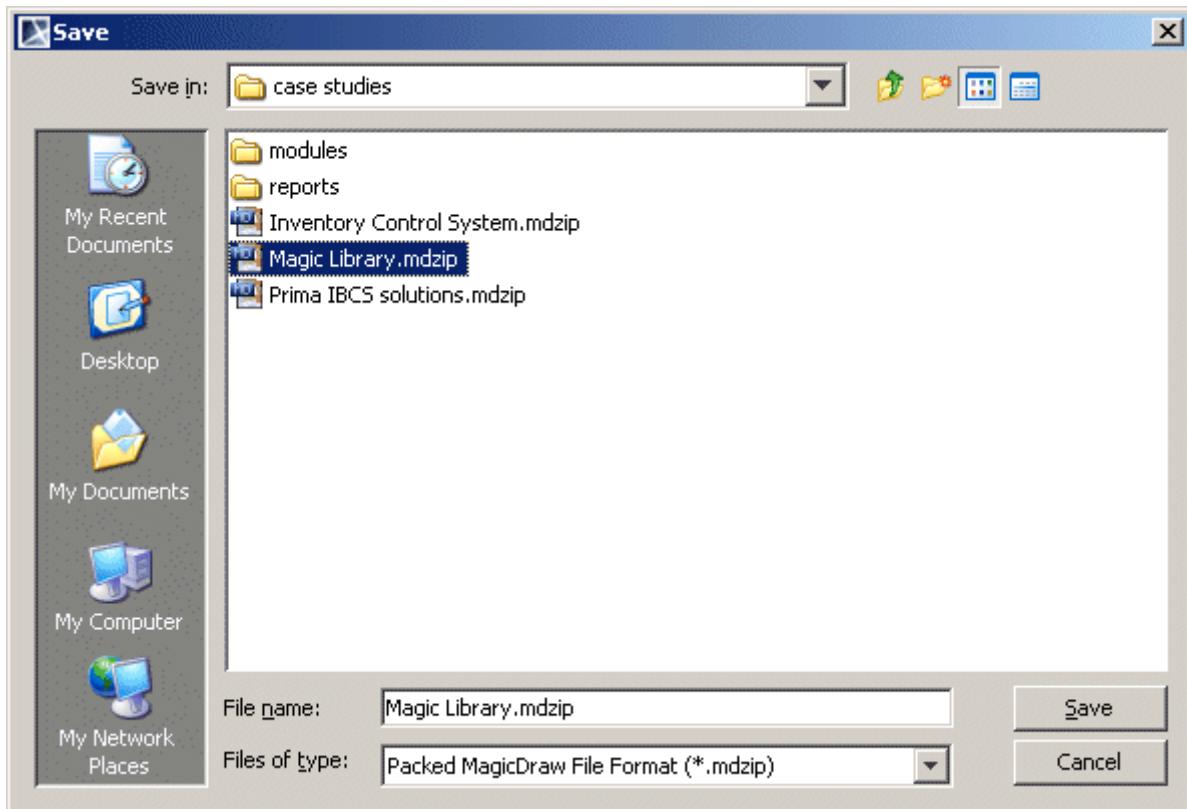


Figure 65 -- Save dialog box

NOTES

- If the **Create Backup File** check box is selected in the **Environment Options** dialog box, MagicDraw always creates a backup file of the previously saved project. The backup is held in a file with a name identical to that of the project. For a detailed description of the **Environment Options** dialog box, see “Customizing Environment Options” on page 89.
- If you want to save maximum additional information to an xmi file (not required in loading to MagicDraw load, but may be useful when using other tools), select the **Rich XMI** check box in the **Environment Options** dialog.

Autosave

After you stop working with MagicDraw, an idle time passes and the current project is saved to a special file called the AutoRecovery file.

If the application is terminated normally, the AutoRecovery file is removed. If the application crashes, the AutoRecovery file is left. On startup, MagicDraw checks for an AutoRecovery file. If it exists, MagicDraw suggests loading the project from this file.

To save an AutoRecovery file of the open project(s) when a system is not in use.

1. Open the **Environment Options** dialog box.
2. In the **General** pane, select the **Save Project Recovery Data on Idle** check box. Enter the system idle time (in minutes) in the **Idle Time to Activate Recovery Save** text box. This is the length of time the system must be idle in order to activate an AutoRecovery save.

Opening a Project

IMPORTANT! The native MagicDraw format is *.mdxml, or *.mdzip.

To edit or review previously created projects

1. From the **File** menu, select **Open Project** or click the **Open Project** button on the main toolbar.
2. In the **Open** dialog box, select the project you wish to open and click **Open**.

TIP: Drag-and-drop the project you wish to open directly from the open window to MagicDraw and the project is started immediately.

Double-click a project file with the *.mdxml or *.mdzip extension, it will open a new MagicDraw application window.

NOTES: If the **Open Last Project on Startup** check box in the **Environment Options** dialog box is selected, the last project you worked with will be opened right after MagicDraw starts.
XMI v2.1 is the native MagicDraw file format for model storage. But this format does not specify how to store diagrams. So, if you use MagicDraw to open an XMI file exported from another tool, only the model will be loaded (not diagrams or views). MagicDraw can not "import" XMI, it can only open it.
For model interchange you should use the RConverter. For more about RConverter, See "Reusing model parts between models" on page 147.

You may load more than one project within the same MagicDraw session. A separate workspace will be created for each project that is opened.

Importing a Project

To import a previously created project to an open project

1. From the **File** menu, select **Import**.
2. The Import dialog box opens. Select the project you want to import and click **OK**. The diagrams of the imported project are placed in the open project.

Exporting a Project

You can export part of a project as a module and share it with other users/projects.

For more information about data partitioning, see "Project Partitioning" on page 141.

For more information about exporting project as MOF, see "Exporting a Project as MOF" on page 154.

For more information about Ecore support, see "Ecore Support" on page 157.

Exporting a project as a template

NOTE: This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

You can save (export) the created project as a template and use the same project for creating other new projects.

To export project as template

1. Open a project you want to export as a template. From the **File** menu, select **Export** and then **Template**.
2. The **Export Template** dialog box opens.
3. Type the name and a description of the template.
4. Click **OK**.

Exporting a project as an Eclipse UML2 (v1.x) XMI file

The export of a MagicDraw model to an Eclipse based UML2 (v1.x) compatible XMI file enables the interchange of the UML v2 models for the further manipulations and transformations with the most popular MDA tools, i.e., AndroMDA, OpenArchitectureWare, etc.

To export a project as an Eclipse UML2 (v1.x) XMI

1. Open a project you want to export as an Eclipse UML2 (v1.x) XMI file.
2. On the **File** menu, click **Export To > Eclipse UML2 (v1.x) XMI File**.
3. Select a directory for the exported project files location.
4. Click **OK**. The project and profiles will be saved in *.uml2 extension.

To change the export options

1. From the **Options** menu, select **Environment**.
2. Click on the **Eclipse UML2 (v1.x) XMI** tab and change the options in the right side pane. For more information, see “Eclipse UML2 (v1.x) XMI Options Tab” on page 110.

Exporting a project as an Eclipse UML2 (v2.x/v3.x) XMI file

The export of a MagicDraw model to an Eclipse based UML2 (v2.x/v3.x) compatible XMI file enables the interchange of UML v2/v3 models for the further manipulations and transformations with the most popular MDA tools, i.e., AndroMDA, OpenArchitectureWare, etc.

To export a project as an Eclipse UML2 (v2.x/v3.x) XMI

1. Open a project you want to export as an Eclipse UML2 (v2.x/v3.x) XMI file.
2. On the **File** menu, click **Export To > Eclipse UML2 (v2.x/v3.x) XMI File**.
3. Select a directory for the exported project files location.
4. Click **OK**. The project and profiles will be saved with the *.uml extension.

To change the export options

1. From the **Options** menu, select **Environment**.

2. Click on the **Eclipse UML2 (v2.x/v3.x) XMI** tab and change the options in the right side pane.
For more information, see “Eclipse UML2 (v2.x / v3.x) XMI Options Tab” on page 112.

Exporting to BPEL

The Business Process Execution Language (BPEL) is an XML schema language for specifying the automated business processes bidirectionally between multiple enterprises using Web Service technology. The BPEL is suitable for all technical staff to manipulate data of process and execute the business process.

The MagicDraw UML BPEL Export exports the Business Process diagram to the valuable BPEL, based on BPEL specification v.1.1.

The Business Process Diagram (BPD) is a helpful tool for business analysts/users to generate a model for business processes. Different from other UML diagrams such as an Activity Diagram or a Use Case Diagram, the BP diagram allows technical staff to easily understand and implement business diagrams. To optimize the valuation of BPD, converting BPD to the execution form (BPEL), which can use the benefit of Web Service, is quite a challenge.

To export a BP diagram to BPEL

1. Create Business Process diagram.
2. From the **File** menu, select **Export** and then **BPEL**. The **BPEL Export** dialog box opens.
3. Select the existing Business Process diagram in the tree and click the **Export** button. The **Save** dialog box opens.
4. Save the BPEL file in the selected location. A message stating the export was successful will appear.

If the BPEL export can not be performed, a Message Window appears with a detailed description of problems.

To evaluate validation rules for export

After exporting the BPMN notations from the BPD diagram to the BPEL files, some additional process data and Web Services information require manipulation in order to deploy the BPEL files on the vendor-specific servers.

It is because BPEL version 1.1 is not a standard yet. There are still many open issues and questions about implementation. Each vendor has made their own decision on specific implementation on the open issues and those questions. BEA is the chosen vendor of this version of MagicDraw BPEL Export. This MagicDraw BPEL Export version will focus on the BEA Web Logic Workshop (WLW) and the BEA Web Logic Integration (WLI); therefore the limitations and open issues from BEA are concerned.

In order to export a BP Diagram based on BPMN specification version 1.0 to a BPEL file based on BPEL4WS Version 1.1 specification with a flavor of BEA, the validation rules for mapping between two standards are required and can be found below.

Ignored Notations

Note, Note HTML, Text Box, Text Box HTML, Anchor to Note, Constraint, Separator, RectangularShape and Group will be ignored and will not be mapped.

UnMappable Notations

Start Event None, Intermediate Event None, Intermediate Event Link, Intermediate Event Cancel, Intermediate Event Multiple, End Event None, End Event Cancel, End Event Multiple, Compensation Task, Gateway Complex, Manual Task, Reference Task, Expanded Sub-Process Compensation, Collapsed Sub-Process Compensation, Collapsed Ad-Hoc Sub-Process, Expanded Ad-Hoc Sub-Process, Association, Directional Association, Compensation Association, Message Flow, Data Object, Horizontal Pool, and Vertical Pool cannot be mapped.

Event Notations

Start Event

- A process level must have one and only one Start Event notation.
- Start Event notation must be a source of Sequence Flow. This means the Start Event notation must have an outgoing Sequence Flow.

End Event

- At least one End Event notation must be present in the diagram.
- Multiple End Events are allowed in the diagram.
- Every branch of the process must have the End Event notation. However, in Expanded Sub-Process notation, the End Event None notation can be used as the ending point.

Intermediate Event

- Intermediate Event notation must be between Start Event and End Event notations.
- Only Message, Timer, Error, and Rule Intermediate Event can be used in normal flow.
- Intermediate Message, Timer, Error, and Rule must have only one incoming Flow.
- Intermediate Event must have only one outgoing Sequence Flow.
- Intermediate Event Link will not be mapped because the methodology of mapping needs to track the flow to the element. Therefore, if the Intermediate Event Link does not have a flow to it, it will not be recognized.

Task and Sub-Process

- Task and Sub-Process cannot have any Event attached to them.
- Task notations must have only one incoming Sequence Flow. In case there are multiple incoming Sequence Flows for a Task, it is recommended to use the Gateway for convergence, instead.
- A diagram (process) inside the Expanded Sub-Process must follow the same rules as the outer diagram (process).

An exception to this is the End Event None can be used in the Expanded Sub-Process.

All notations inside the Expanded Sub-Process cannot have an incoming Sequence Flow or an outgoing Sequence Flow linked directly to notations outside the Expanded Sub-Process.

Gateway

- If a Gateway has only one incoming and one outgoing Sequence Flow, the gateway is not needed in the BP Diagram.
- A Gateway cannot have both multiple incoming Sequence Flows and multiple outgoing Sequence Flows. If it has multiple incoming Sequence Flows, it must have only one outgoing Sequence Flow. If it has multiple outgoing Sequence Flows, it must have only one incoming Sequence Flow.

DataBased XOR Gateway

- If there is only one incoming Sequence Flow to the Data-based XOR Gateway and more than one outgoing Sequence Flow, the Data-based XOR Gateway will act as a convergence. A Data-based XOR Gateway must have one outgoing Default Sequence Flow and at least one outgoing Condition Sequence Flow with condition expression. The normal Sequence Flow cannot be an outgoing Flow from a Data-based XOR, which acts as a convergence.
- If there are more than one incoming Sequence Flows to a Data-based XOR Gateway and only one outgoing Sequence Flow, the Data-based XOR Gateway will act as a divergence. The only

4 WORKING WITH PROJECTS

Exporting a Project

outgoing Sequence Flow from a Data-based XOR Gateway must be a normal Sequence Flow and not a Condition Sequence Flow or Default Sequence Flow.

EventBased XOR

- For an EventBased XOR Gateway, the outgoing Sequence Flow can only target a Task, with the TaskType set to Receive or an Intermediate Event with a Trigger attribute set to Message, Timer, or Rule.
- If there is only one incoming Sequence Flow to an Event-based XOR Gateway and more than one outgoing Sequence Flow, the Event-based XOR Gateway will act as a convergence. The only outgoing Sequence Flows from an Event-based XOR Gateway must be a normal Sequence Flows and not a Condition Sequence Flow or Default Sequence Flow.
- If there are more than one incoming Sequence Flows to an Event-based XOR Gateway and only one outgoing Sequence Flow, the Event-based XOR Gateway will act as a divergence. The only outgoing Sequence Flow from an Event-based XOR Gateway must be a normal Sequence Flow and not a Condition Sequence Flow or Default Sequence Flow.
- If the outgoing Sequence Flow targets a Receive Task, then an Intermediate Event with a Trigger Message cannot be a target of the other outgoing Sequence Flow of this Event-based XOR Gateway.

Inclusive OR

- If there is only one incoming Sequence Flow to an Inclusive OR Gateway and more than one outgoing Sequence Flow, the Inclusive OR Gateway will act as a convergence. An Inclusive OR Gateway must have one outgoing Default Sequence Flow and at least one outgoing Conditional Sequence Flow with condition expression. The normal Sequence Flow cannot be an outgoing Flow from an Inclusive OR, which acts as a convergence.
- If there are more than one incoming Sequence Flows to an Inclusive OR Gateway and only one outgoing Sequence Flow, the Inclusive OR Gateway will act as a divergence. The only outgoing Sequence Flow from an Inclusive OR Gateway must be a normal Sequence Flow and not a Condition Sequence Flow or Default Sequence Flow.

Parallel AND

- If there is only one incoming Sequence Flow to a Parallel AND Gateway and more than one outgoing Sequence Flow, the Parallel AND Gateway will act as a convergence (Parallel AND Fork). The only outgoing Sequence Flows from a Parallel AND Gateway must be a normal Sequence Flows and not a Condition Sequence Flow or Default Sequence Flow.
- If there are more than one incoming Sequence Flows to a Parallel AND Gateway and only one outgoing Sequence Flow, the Parallel AND Gateway will act as a divergence (Parallel AND Join). The only outgoing Sequence Flow from a Parallel AND Gateway must be a normal Sequence Flow and not a Condition Sequence Flow or Default Sequence Flow.

Sequence Flow

- All notations, except Start Event must have an incoming Sequence Flow
- All notations, except End Event, must have an outgoing Sequence Flow.
- Isolate notation is prohibited in the diagram. This means that any notations without any incoming and outgoing flow are not allowed in the diagram.
- All notations, except Gateway, must not have more than one incoming Sequence Flow.
- Default Sequence Flow and Condition Sequence Flow are only allowed to be used with the Gateway notation.

Setting Project Options

Project Options

The **Project Options** dialog box is used for defining properties for the model elements (shapes and paths) and diagrams, for creating your own project style, for importing or exporting a created project style, for applying the default element properties, and for defining code engineering options.

To define Project Options

From the **Options** main menu, select **Project**. The **Project Options** dialog box opens.

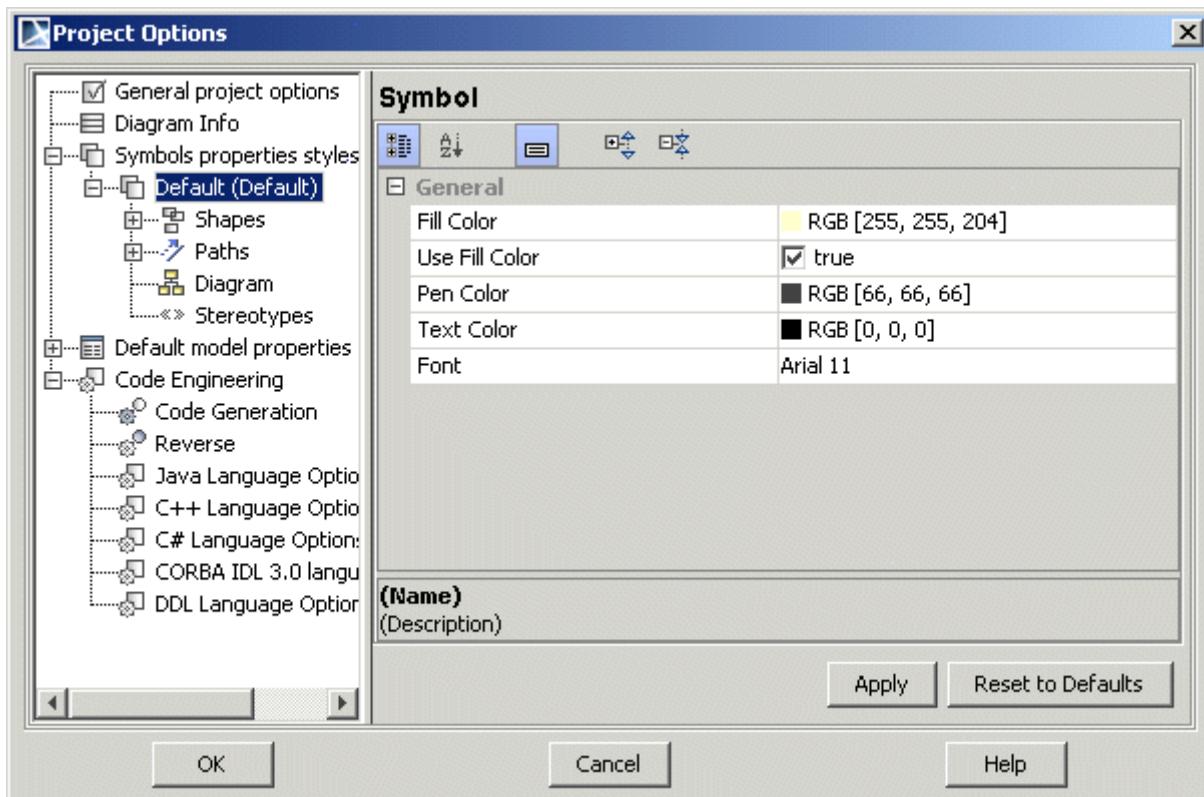


Figure 66 -- Project Options dialog box

4 WORKING WITH PROJECTS

Setting Project Options

The **Project Options** dialog box has several collections of customizable options:

General Project Options

- In the **Modules Path** text box, enter the path where by default modules will be stored. Define the default packages where the extension elements will be stored. Also, if desired, you can select the package every time when creating a new extension element, the dialog box asking about the storage place will open. In the **Modules Path** text box, enter the path where, by default, modules will be stored.
- If the **Auto synchronize Parameters and Arguments** check box is selected, arguments will be created on parameters creation and they will be synchronized. After **Auto synchronize Parameters and Arguments** check box is cleared parameters and arguments are not synchronized anymore. By default the **Auto synchronize Parameters and Arguments** check box is selected. For more information about parameters and arguments synchronization see "Parameters synchronization with Arguments" on page 736.
- The **Qualified name display style** option:
 - **Absolute**.
 - **Model Library relative** (Model Library is a package with *modelLibrary* stereotype).
 - **Model relative** means that element's qualified name shown on its symbol (not property) should be cropped from the nearest model found in hierarchy without its name.
 - **Model or model library relative** option combines two options "Model relative" and "Model Library relative" in one option.

Diagram Info

Customize how and what information will be displayed in the Diagram Info table.

Symbols Properties Styles

Expands the tree hierarchy of all the styles defined within the project. You can create as many of these styles as you like.

NOTE To apply the changed project options to an open project, click the **Apply** button.

For more information about the style engine, see "Style Engine" on page 258.

Default model properties

You may define model element properties in the **Project Options** dialog box, in the **Default model properties** branch. Select the exact element in the tree and change the property value in the right side pane. To reset the element properties to the default value, click the **Reset to Defaults** button. To reset the property values for all elements select the **Default model properties** branch and click the **Reset to Defaults** button.

For more information about setting the default element property values, see "Default Property Values" on page 253.

Code Engineering

Define general code engineering options, as well as options specific to languages. For more information, see Code Engineering user's guide.

To find elements to change, browse the options tree in the **Project Options** dialog box. The items in this hierarchy are either:

- **Compressed** - a plus sign next to an icon indicates that the icon is compressed, and contains other model elements. This is the default setting when you start your application. Click the plus sign to expand the icon and view its subordinate items.

- **Expanded** - a minus sign next to an icon indicates that the icon is fully expanded. Click the minus sign to collapse the item.

If there is no sign next to an icon, it does not contain other model elements.

Searching

The MagicDraw search mechanism enables searching within model elements data, symbols, and extensions.

You may also search for usages and dependant elements of the selected elements. This functionality is described in the “Tooltip text” on page 209.

To quickly find the needed classifier or diagram

1. From the **Edit** menu, select **Quick Find**.
2. In the opened dialog, type the name of the classifier or diagram (also, you can select it from the drop-down list) and choose one of the option buttons for the advanced search.

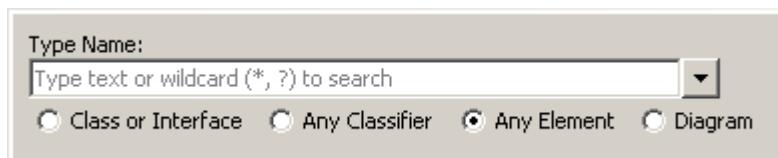


Figure 67 -- Quick Find dialog

Filters in the autocomplete dialog allow the filtering of rarely used items, such as “metaclasses” and “elements from modules”. This allows comfortable and clear usage of the autocomplete dialog for modeling, without active usage of the elements from profiles and it increases modeling speed.

At the bottom of the drop-down list box, you will find buttons to perform filtering:

- **The Auto completion includes metaclasses button.** When pressed, the list of available elements, element types, or stereotypes includes metaclasses (in MagicDraw metaclasses are placed in the *UML Standard Profile*) appears.
- **The Auto completion includes elements from profiles and modules button.** When pressed, the list of available elements, element types or stereotypes includes elements, which are placed in modules appears. (**Note!** This option toggles all profiles except the *UML Standard Profile*.)
- **The Auto completion uses camel case button.** When pressed, you may search for elements via the capital letter patterns. For example, instead of typing *ArrayIndexOutOfBoundsException* you may type *AIOOBE*.

Find all elements of the same type

1. From the **Edit** menu, select **Find**, or press corresponding shortcut key Ctrl+F, or click the **Find** button on the main toolbar. The **Find** dialog box opens.
2. Type the “*” symbol in the **Name** text box.
3. Click the “...” button near the **Type** text box to open the **Select Element/Symbol Type** drop-down combo box. Select the types of elements and click **OK**. The Model elements in this box are listed according to the metamodel.

4 WORKING WITH PROJECTS

Searching

4. Click the **Find** button to start search. The search results will appear in the **Search Tree** in the Browser.

TIP!

To generalize the beginning or ending of the name, add the "?" symbol to the front or to the end of the string.

Find model elements and symbols in your project

1. Choose **Find** from the [Edit](#) menu or press the corresponding shortcut key Ctrl+F or click the **Search** tab in the Browser.
2. In the **Name** text box, type the name of the element. If you want to find all elements of the selected type, enter the "*" symbol in the **Name** text box.
3. Click the "..." button near the **Type** text box to open the **Select Model Element/Symbol Type** drop-down combo box. Select the types of elements and click **OK**. The Model elements in this box are listed according to the metamodel.
4. To start a search, click the **Find** button. The search results will appear in the **Search Results** tree in the Browser.

TIP!

To generalize the beginning or ending of the name, add the "?" symbol to the front or to the end of the string.

NOTE

If the **Clear Previous Results** check box is cleared, new results are appended to the previous search results in the tree.

TIP!

Select **Search Data Unused in Diagrams** check box to find only elements without shapes.

To search for symbols in an active diagram

Search in the active diagram using the **Find** or **Quick find** dialog is a time saving feature.

You can search for the symbols of elements, which are drawn in the open diagram:

1. Select the **Find** in Diagram command from the diagram shortcut menu. The **Find** dialog box opens. In the **Find** dialog box, the **Limit results to active diagram** check box is selected.
2. Type the element name for the symbol you are searching for. Click the **Find** button.
3. In the **Search Results** tree, double click on the element and the symbol of this element is selected on the diagram pane.

-or-

- Press the **Shift+F** key. The **Find** dialog box opens.
- Press the **Ctrl+Shift+F** key to open the **Quick find** dialog.

4 WORKING WITH PROJECTS

Searching

Find dialog box

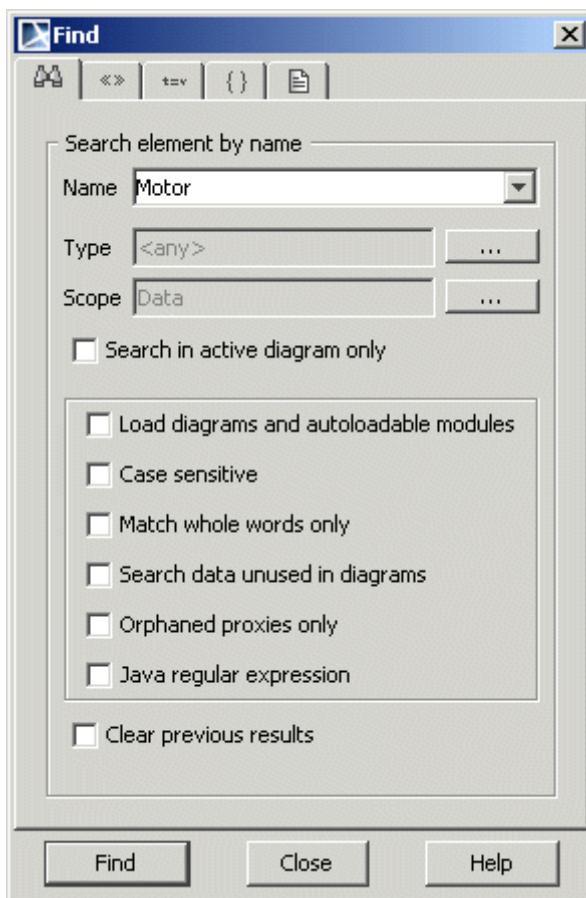


Figure 68 -- Find dialog box

The **Find** dialog box contains five tabs:

| Tab Name | Tab Icon |
|------------------------------------|---------------|
| Search Element by Name | oculars icon |
| Search Element by Stereotype | <> icon |
| Search Element by Tagged Value | +=v icon |
| Search Element by Constraint Value | { } icon |
| Search Element by Documentation | document icon |

4 WORKING WITH PROJECTS

Searching

Search Elements

| Element | Function |
|--|---|
| Name | <p>Type the name of the item you wish to find.</p> <p>NOTE: You may define wildcards <*> and <?> for the search. For example, if you define the following input string <a*b>, the system looks for items with <a> at the beginning and at the end of the string. If you define the string as <?agicDraw>, all strings containing <agicDraw> will be found.</p> |
| Type | Select an element type from the proposed items, or leave the default value of <any>. |
| Scope | Specify a package where the content search will be performed. |
| Limit results to active diagram | <p>The search scope is limited to the symbols of active diagrams. The Check box is disabled if all diagrams are closed or if the active diagram is empty.</p> <p>The default value is <i>false</i>.</p> <p>If the Find dialog box is opened from the diagram shortcut menu, the Limit results to active diagrams check box is selected.</p> <p>NOTE: The Limit results to active diagrams check box does not exist in the Find dialog box when it is opened from the model comparing dialog.</p> |
| Value | <p>Select or input a value of the Tagged Value or Constraint Value.</p> <p>NOTE: Only available for Search Element by Tagged Value and Constraint Value tab.</p> |
| Load elements (not loaded) and autoloadable modules | <p>If the model has diagrams or modules that are not loaded, select this check box to load all elements to be included in the search.</p> <p>NOTE: Elements will not be included in the search if the module load mode is set to Manual load.</p> |
| Case Sensitive | <p>Search for items that have capitalization exactly as defined in the string entered in the Item to Find box.</p> <p>When the check box is cleared, MagicDraw does not distinguish between uppercase and lowercase characters of the item name entered in the Name box while searching.</p> |
| Match Whole Words Only | <p>Search for items with names that exactly match the string entered in the Item to Find box.</p> <p>When the check box is cleared, MagicDraw searches for items with names matching the first part of the string entered in the Name field.</p> |
| Search Data Unused in Diagrams | Only searches elements that do not contain symbols in any diagram. |
| Java Regular Expression | In the “Java Regular Expressions” on page 138, you can find several expressions that will help you to make a search in MagicDraw. |
| Clear Previous Results | Removes all previous search results from the Browser tree. |
| Find | Searches for items and displays the results in the found items list field. If MagicDraw does not find any items, a message is displayed. |
| Close | Exits the dialog box. |
| Help | Displays MagicDraw Help . |

Java Regular Expressions

Metacharacters

There are several characters supported, which are used to form search patterns $([^\$])?^*$.

There are two ways to force a metacharacter to be treated as an ordinary character:

- + Precede the metacharacter with a backslash
- + Enclose it within \Q (starts the quote) and \E (ends it).

Character Sets

| | |
|------------------------------|--|
| [abc] | Any character of a, b, or c. |
| [^abc] | Any character except a, b, or c (negation). |
| [a-z] | All characters from a to z (range). |
| [a-z[A-Z]] | All characters from a to z and A to Z (union). |
| [a-z&&[r-z]] | Characters from r to z (intersection). |
| [a-z&&[^r-z]] | Characters from a to q (subtraction). |

Predefined character sets:

| | |
|-----------|---------------------------------------|
| . | Any character. |
| \d | Any digit character. |
| \D | Any non digit character. |
| \s | White space character (\t\n\x0B\f\r). |
| \S | Any non white character. |
| \w | Word character (a-z, A-Z, _, 0-9). |
| \W | Any non word character. |

Example:

Regular expression: [ABC][^\s]\d

Matched text: any sequence starting with an "A", "B", or "C" symbol, followed by any non white space character and any digit.

Grouping

Capturing groups helps to treat multiple characters as a single unit.

Example:

Regular Expression: ABC|(\dABC)

Matched text: any text containing ABC symbol set or ABC symbol set beginning with any digit symbol.

Quantifiers

Quantifiers allow specify a number of character (X) appearances.

| | |
|-----------|---------------------------|
| X? | Match X zero or one time. |
|-----------|---------------------------|

4 WORKING WITH PROJECTS

Searching

| | |
|--------|---|
| X* | Match X zero or many time. |
| X+ | Match X one or many time. |
| X{n} | Match X exactly n times. |
| X{n,} | Match X at least n times. |
| X{n,m} | Match X exactly n times, but not more than m times. |

Example:

Regular expression: Cla(s{2})

Matched text: any sequence starting with "Cla" symbols, followed by "s" symbol two times. It will match any text containing the string "Class".

Boundary Matchers

Boundary matchers help to match strings more precisely. Boundary matchers help by matching a particular word, beginning or end of line, or beginning or end of the input.

| | |
|----|-------------------------|
| ^ | Beginning of the line. |
| \$ | The end of the line. |
| \b | A word boundary. |
| \B | A non word boundary. |
| \A | Beginning of the input. |
| \z | End of the input. |

Example:

Regular expression: \bCla(s{2})\b

Matched text: any sequence starting with "Cla" symbols, followed by "s" symbol two times. It will match any text containing string "Class" as whole word ("Classs" won't be matched).

Embedded Flag Expressions

Allows setting to set properties for a regular expression matcher.

| | |
|------|--|
| (?i) | Case insensitive matching. |
| (?x) | Ignores white spaces in regular expression. |
| (?m) | Enables multi line option. If not specified, boundary matches ^ and \$ matches beginning of the input and end respectively. |
| (?s) | Enables expression "." to match any character including line terminators. If not specified, dot in expression does not match line terminators. |
| (?u) | Enable Unicode-aware case folding. When this flag is specified, case insensitivity is applied to the Unicode standard. |
| (?d) | Enable Unix lines mode. Only terminator "\n" is recognized in behavior of ".", "^", "\$" |

Example:

Regular expression: (?m)^\bCla(s{2})\b

Matched text: any sequence from a new line, starting with "Cla" symbols, followed by "s" symbol two times.

References

<http://java.sun.com/docs/books/tutorial/extra/regex/index.html>

Replacing

The Find and Replace functionality allows replacing one specified model value with another value.

You can change the values for the following properties:

- Names
- Documentation
- Tag values
- Text included to Notes
- Text included to Text Boxes
- Expressions.

To replace a value:

1. From the **Edit** menu, select **Find and Replace**. The **Find and Replace** dialog box opens.

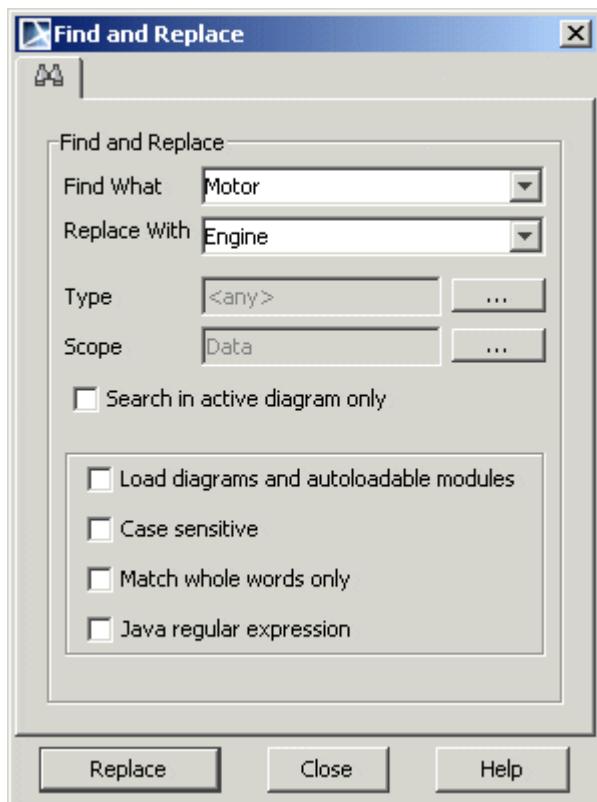


Figure 69 -- Find and Replace dialog box

2. Type the value to be replaced into the **Find What** field.
3. In the **Replace With** field, type the value that will replace the value of the found element.
4. Specify the search criteria. For more information about the search criteria, see "Searching" on page 134.

- Click the **Replace** button to start the replacement. On each property replacement the question appears. You may choose to replace value, replace all values or not replace.

NOTE You will see the error when changing value to not valid:

- For example, if value type is *boolean*, you may replace values from *true* to *false* or from *false* to *true*, but if you will try to replace the value *true* or *false* to other, for example, to *Motor*, an error message will be displayed.
- You will not be allowed to change the *Integer* value to *String* when it is a part of the value. For example, if you have the *120* value and trying to replace *20* with *AB*, an error message will be displayed.

Project Partitioning

[View Online](#) [Shared Packages](#)
[Demo](#)

Partitioning the model

If you developed, or are developing, a large model that has several weakly dependent parts, it is advisable to split it into several module files. Partitioning opens up possibilities for reusing model parts in several related projects and may improve performance on very large projects, when modules are loaded selectively.

Partitioning has a package level granularity. Smaller elements cannot be split into separate modules. In principle each package in a containment tree could be partitioned into a separate module, however this is excessive.

The decision on how to split a model into parts should be made carefully. You should isolate model parts, which form some cohesive, logically complete piece of structure (subsystem, code library, profile) and have light inter-dependencies.

When there are many one-way dependencies to some model part (parts **A**, **B**, **C** depend on part **D**, but part **D** does not depend on any of the parts **A**, **B**, **C**), this part is a good candidate for placement into module.

When one big project is used to store all the modeling information of the project models (use case models, high level architectural models of the project, detailed implementation level class, sequence, state, etc.), it may be useful to partition the models according to the modeling domains (use cases in one module, architectural models in another, implementation level models in yet another). This allows unloading unnecessary modules while working on one part or another (saving computer and improving performance), but still retain the relationships between domains and load modules, on demand.

NOTE Avoid partitioning a model into parts, which have circular dependencies.

(A \leftrightarrow B or A->B->C->A situations)!

Usually programmers are very adept at splitting large code bases into libraries. The very same criteria should be applied to splitting the large models into modules.

MagicDraw module functionality allows two important possibilities:

- possibility to work without all modules loaded;
- read-write modules.

Modules are often used for profile storing, however a module **is not** a profile and it is important not to mix the two. Any model part can be stored in the module.

Exporting the module of a project

NOTE: This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

Using the **Export Module** dialog box, you can partition the model and save the content of a selected package as a separate module. Once exported, the package and its containing elements are read-only and the module name is displayed in brackets next to the package name in the Browser tree.

To export any module using the **File** menu

1. From the **File** main menu, select **Export**, and then select **Module**. The **Export Profile/Module** dialog box opens.

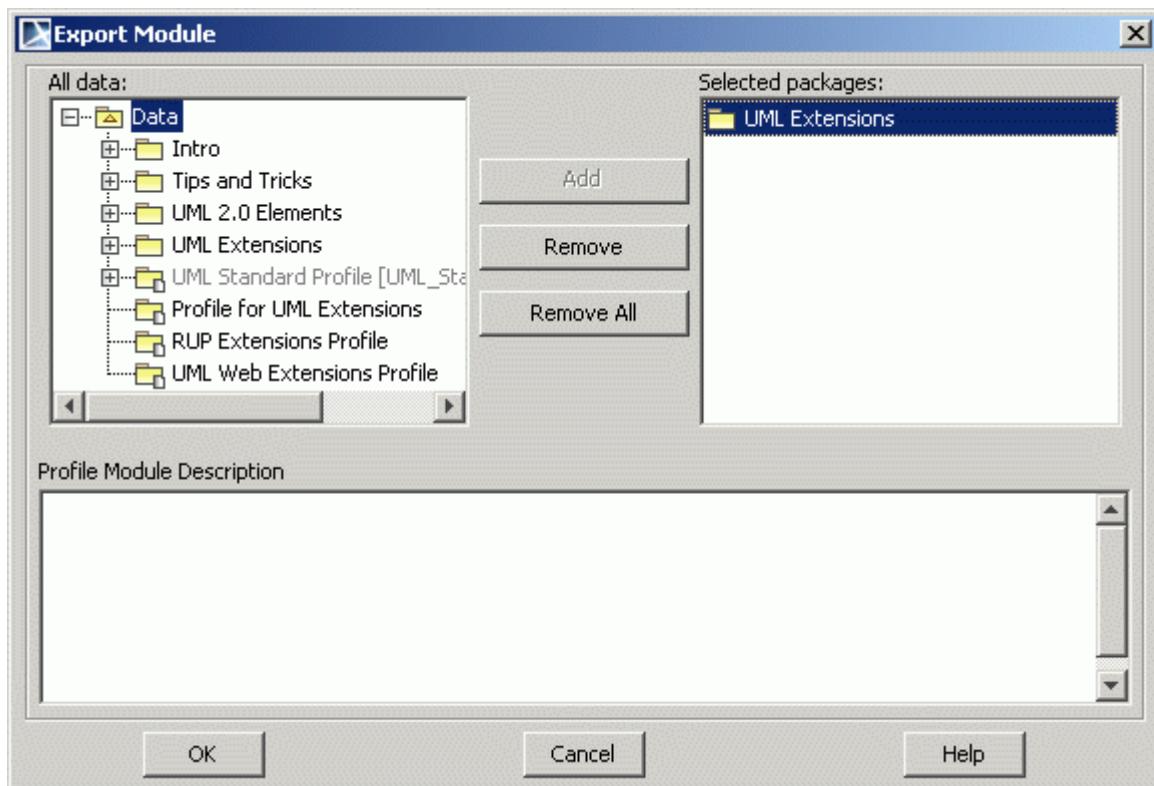


Figure 70 -- The Select Package dialog box for module exportation

2. In the **All Data** list, select the package you want to save as a separate module. Click **Add**. The package is added to the **Selected Objects** list.
3. If desired, type a description of the module in the **Profile Module Description** window. This description is displayed in the documentation of the package.
4. Click **OK**.

To export the selected module using the package shortcut menu

1. In the Browser tree, select the package you want to save as a separate module (you can also select multiple packages).
2. From the package shortcut menu, select **Modules**, and then select **Export Module**. The **Export Module** dialog box opens.

3. In the **All Data** list, select the package you want to save as a separate module. Click **Add**. The package is added to the **Selected Objects** list.
4. If desired, type the description of the module in the **Profile Module Description** window. This description is displayed in the documentation of the package.
5. Click **OK**.

MagicDraw will check for dependencies from the exported part of the model to the part of the model not. You will have to resolve them. The dependency resolution process is the same as for dependency resolution between shared and parts of the module not shared.

When dependencies are resolved, MagicDraw will ask for the file and export the module.

This action can be thought as consisting of 3 elementary steps:

- Saving model elements into the module file.
- Sharing the entire contents of the module.
- Using the module in the main project.

Alternatively, if you have several small, related projects, you can join them together into a larger, partitioned project to work with all the information from one place. This is achieved by using the **Use Module** command that was previously mentioned.

NOTE: Only packages can be exported as modules. To export the created diagram, you must move it to a package containing elements to export.

Sharing the module of a project

NOTE: This functionality is available in Standard, Professional, Architect and Enterprise editions only.

Not all module contents are visible in the project being used. The Module has a shared part and private part. Only contents of the shared part are visible in the project being used. The concept is similar to the public/private parts of modules in programming languages (e.g. Pascal).

To designate packages of the module as shared

- From the **File** main menu, select **Shared Packages**.
- From the package shortcut menu, select **Modules** and then **Shared Packages**. The **Shared Packages** dialog box opens. Use the **Add** button to select more packages for multiple simultaneous sharing, if needed. Click **OK**.

Only the package selected is shared and everything else is not shared.

When the module with shared package(s) is used in the project, the shared part(s) is mounted into the module of the project. Each shared package can have a different mount point. Modules of profiles are typically mounted directly under the top level **Data** element of the package being used, however this can be changed.

Example:

Shared package "util" from the module can be mounted on the "com::company" path in the main project - to form the "com::company::util" path. The Preferred Path of the Shared Package (can be tuned in the **Shared Packages** dialog box) of the module, serves as a hint for MagicDraw on where to mount the package.

Modules form a recursive data structure - the main project uses one or several modules; these modules in turn can use other modules; those other modules can use yet another set of modules and so on. All model pieces

from these modules are gathered and connected into the integral model, which is shown in the model Browser when the main project is opened.

Managing modules (the Modules dialog box)

From the **Options** main menu, select **Modules**.

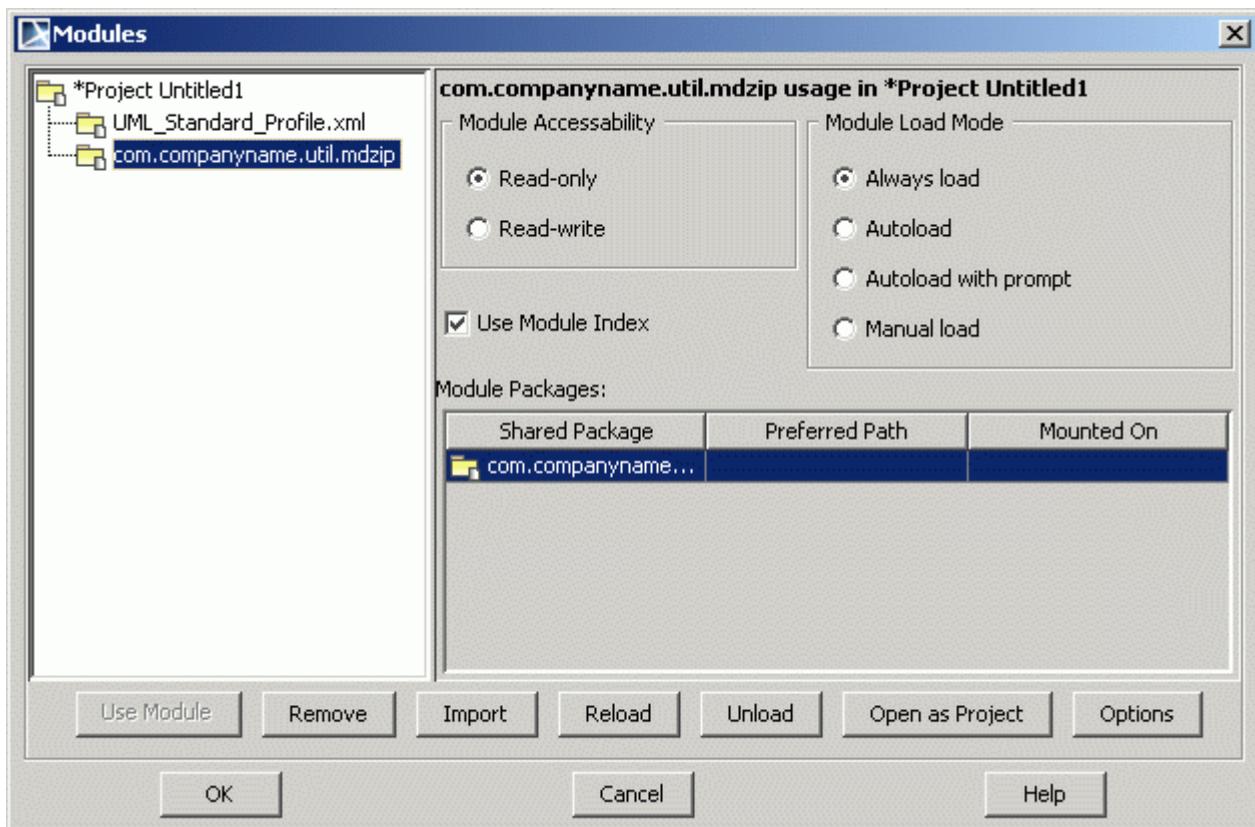


Figure 71 -- The Modules dialog box

| Box name | Function |
|-----------------------------|--|
| Module Accessibility | Specifies the way a module can be used in a project: <ul style="list-style-type: none"> • read-only modules are not editable within the project using it. • read-write modules can be edited in place - directly in the project using it. |
| Module Load Mode | Sets the module loading mode: <ul style="list-style-type: none"> • Always load (default) - modules are always loaded when the project is opened. • Autoload - module is not loaded when the project using it is loaded. However, MagicDraw monitors user activities in the project and tries to anticipate guess when the user might want to use the model piece from the unloaded module. • Autoload with prompt - mode is similar to Autoload. However, MagicDraw will ask the user before loading the module. • Manual load - module is not loaded when the project using it is loaded. |

4 WORKING WITH PROJECTS

Project Partitioning

| | |
|-------------------------|---|
| Use Module Index | If selected, uses the indexing scope specified in the Project Options dialog. For more information about indexing, see “Indexing” on page 150. |
| Shared Package | Name of the shared package. |
| Preferred Path | Carries the information about the path where the shared package should be placed in the project using it. |
| Mounted On | Holds the packages of the project on which the corresponding module share is mounted. Click the “...” button to change package or create a new one. |
| Use Module | Enabled, when the module is selected as read-write. Allows the use of the module in the selected module. |
| Remove | Removes module from the project. |
| Import | Imports module to the project. |
| Reload | Reloads module in the project. |
| Unload | Unloads module from the project. |
| Open as Project | Opens the selected project as a module. |
| Options | The Project Options dialog box opens. |

Analyzing Dependencies Among Elements

A package can be exported to an independent module only if it does not depend on external elements (except other modules). Cyclical dependencies between several modules are not allowed.

All the information about analyzing and checking package dependencies you may find in the Section “Analyzing Package Dependencies” (see on page -436)

There are three types of dependencies:

- Dependency by relationship
- Dependency by reference
- Diagram dependencies

Using the module of a project

When a module is used in another project, its contents are linked-in and made accessible in the model tree of the project using them as if it were part of the project.

To use a module in a project

1. From the **File** main menu, select **Use Module**.
 2. In the **Use Module** dialog box, select the module you want to use in your project, specify the module settings and click **OK**.
- The model elements are still stored separately; module elements - in the module file and main project elements - in the main project file.

The Use Module Wizard

From the **File** menu, select the **Use Module** command.

4 WORKING WITH PROJECTS

Project Partitioning

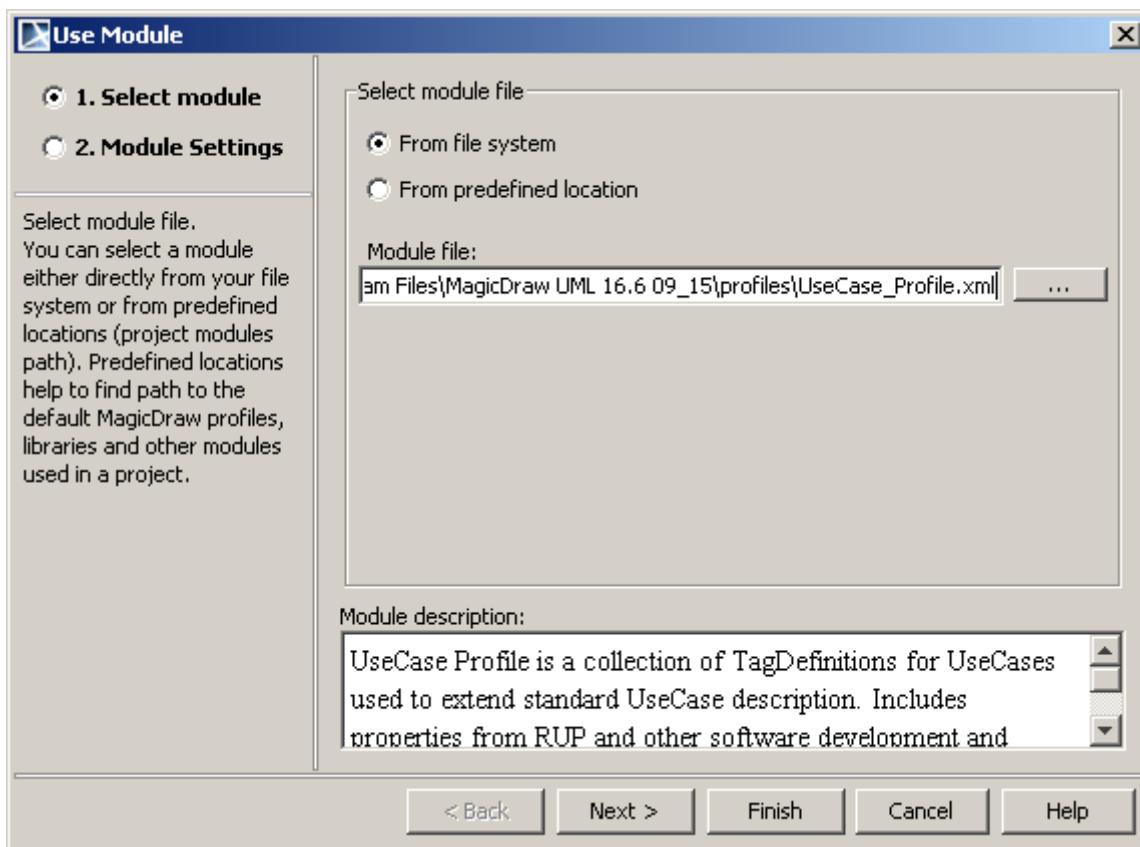


Figure 72 -- The Use Module Wizard, Select Module step, the From file system radio button

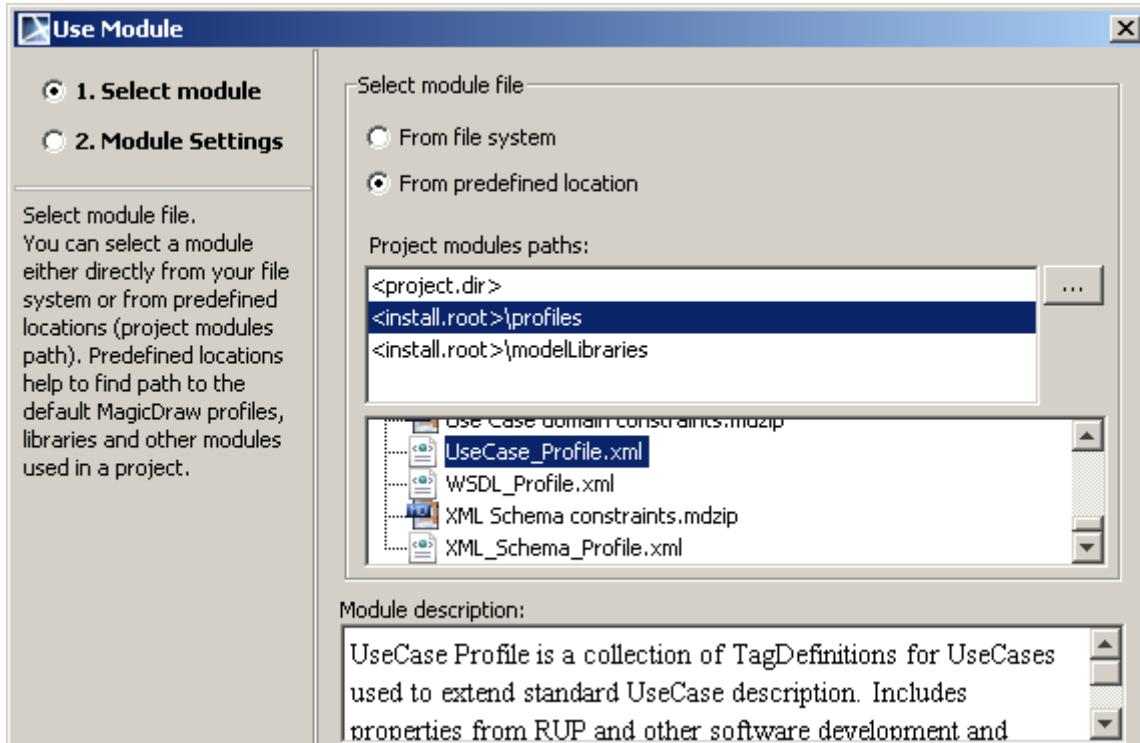


Figure 73 -- The Use Module Wizard, Select Module step, the From predefined location radio button

4 WORKING WITH PROJECTS

Project Partitioning

| Box name | Function |
|---------------------------------|--|
| From file system | Allows selecting module file from your file system. Click the “...” button next to the Module file text box, to browse to module file (Figure 72 on page 146). |
| From predefined location | Allows selecting modules from predefined locations. Select project module path from a paths list, and then select module file from the list below. Click the “...” button next to the Project modules paths text box to add new module path to your project. The Select Folder dialog box will open (Figure 73 on page 146). |
| Module description | Displays the module description. |
| Next | Proceeds to the next step. |
| Finish | Saves changes and closes the dialog box. |
| Cancel | Cancels the dialog box without saving changes |
| Help | Displays MagicDraw Help. |

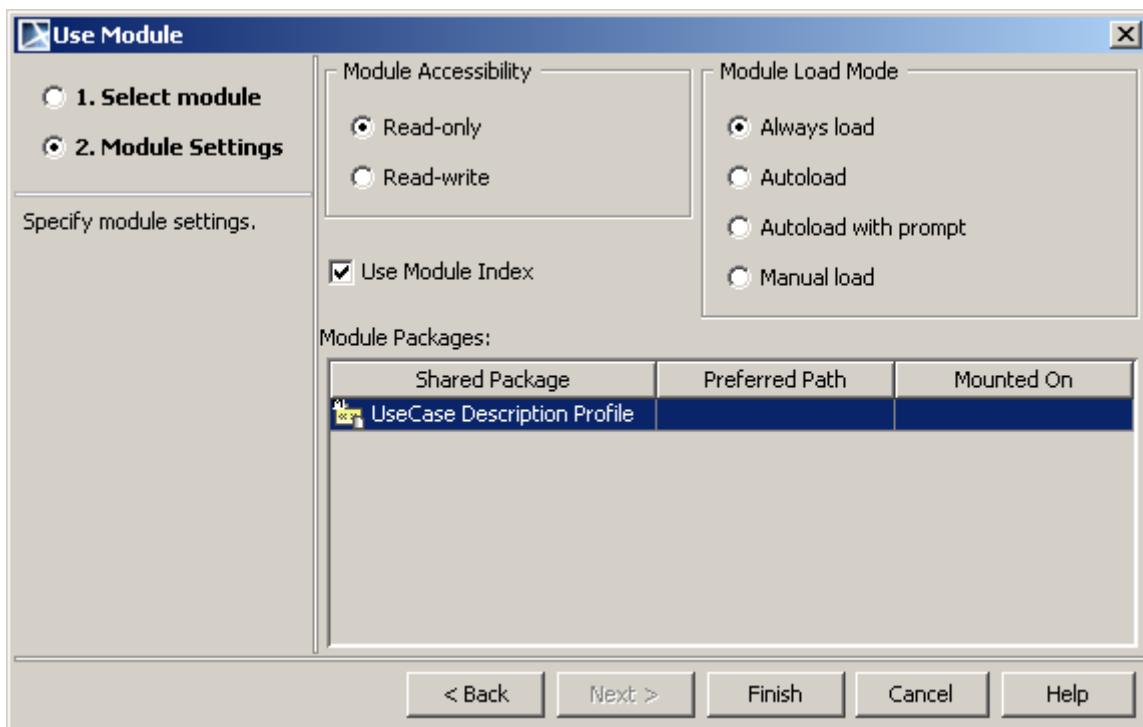


Figure 74 -- The Use Module Wizard, Step2 - Module Settings

For more information about the options in this step, see “Managing modules (the Modules dialog box)” on page 144.

Reusing model parts between models

When models are properly partitioned, model parts can be reused in other projects.

4 WORKING WITH PROJECTS

Project Partitioning

For example, a common situation in Java software projects is this layout of the packages (in project **A**):



Common and *util* packages are good candidates for refactoring into modules. Then in project **B** these modules can be reused.

There are two ways to use the module in the project:

- read-only modules are not editable within the project using it.
- read-write modules can be edited in place - directly in the project using it.

The usage mode can be specified in the **Use Module** Wizard, Step 2. By default, the module is used in the project in read-only mode.

To change module accessibility mode

1. From the **Options** main menu, select **Module**. The **Module Options** dialog box opens.
2. Select module in the tree and change the **Module Accessibility** option from read-only to read-write or vice versa. Click **OK**.

You can change content of a module and make its inner elements editable by selecting **Open Module As Project** (from the module shortcut menu, **Modules** submenu). The module opens as a separate project.

When to use read-only module?

The decision to use a module as read-only or read-write depends on the maturity of the module and the organization ownership/responsibility rules for the projects, developing modules.

If the library in the module is mature (changes to it are not expected/likely/possible) it should be used in read-only mode.

If the module is owned by a team, working on one project, and this team is responsible for this module and the module is reused in another project, the module should be used in the other project as read-only. This prevents inadvertent changes to the library.

When to use read-write module?

In the case where a module is actively developed and evolves together with the projects that are using it, a module should be used read-write.

In this case, if there are multiple projects using the module, you should be careful and remember, that your changes to the module will be reflected in other projects, therefore care should be exercised. Usage of team-work server might be advisable in this case. And, of course, there can be mixed usage situations - when a module is used read-only in some projects and read-write in others.

Reloading the module of a project

The best way to access the latest changes to your module is to reload it. All modifications made in the other project for this module and then exported as modules with the same name, are reloaded in the current project.

To reload the module

In the Browser tree, from the exported module shortcut menu, select **Modules**, and then select **Reload Module**.

NOTE: If you open your module as a project, be sure to save any changes you have made (by using the Save command). All modifications appear after reloading the module in the other open project, which includes your module as a component.

Importing the module of a project

If you wish to store model elements of the module in the main project file, you can import the module into the project.

To import a module into a project

From the package shortcut menu in the Browser, select **Modules** and then **Import Module**.

All the model elements from the module will be copied into the main project, and the module will be unlinked from the project.

Working with partially loaded projects

In MagicDraw v11.5, the possibility to work with some modules unloaded was added. Prior to this release, all the modules were required to be loaded when the main project was loaded. If the module file was missing, it was an error.

This feature allows some memory to be saved and improved performance when working with very large projects. "Large" for MagicDraw is several thousands of classes and other complex elements. If counting all the small elements, such as properties, methods, method parameters, ~50K of elements is considered a large project. A good example of a large module is a module having a model of Java rt.jar reversed into it. Also diagrams are large elements - 20 or more complex diagrams should be considered large.

When working with a large project partitioned into several modules, at any moment a module can be unloaded. When editing a project, if you see that the module will not be used for some time (perhaps you are working on a different part of the large project), you can unload it - this will save resources.

To unload a module from a project

From the module shortcut menu, select **Modules** and then **Unload Module**.

An unloaded module can be loaded at any time.

To load a module in a project

From the module shortcut menu, select **Modules** and then **Load Module**.

When the module is unloaded, there are some model elements left in the place where the module was mounted. These elements are not editable, and they have a small M in the upper right corner of their icon.

These are the so-called "proxy" elements of the real elements from the module. Instead of the real model elements, the proxy carries only the name and kind of the model element information - it is a lightweight surrogate for the real model element. The proxies are left in the place of those module elements, which are referenced from the main project. These proxies are normal and necessary to maintain project integrity (so that there are no dangling ends of relationships, types of properties do not disappear, etc.).

There are 4 module loading modes:

- Always load (default) - this mode of operation closely mimics the pre-v11.5 MagicDraw functionality. In this mode, modules are always loaded when the project is opened. They can be unloaded if the user deems it necessary.
- Autoload - module is not loaded when the using project using it is loaded. However, MagicDraw monitors user activities in the project and tries to anticipate when the user might want to use the model piece from the unloaded module. E.g. if the user does the search, finds usages/dependencies, reports, metrics, transformations, or code engineering actions with a scope that touches the unloaded module, MagicDraw will load the module.
- Autoload with prompt - mode differs from the *Autoload* mode in this way: MagicDraw will ask the user before loading the module.
- Manual load - module is not loaded when the project using it is loaded. It can be loaded, using the aforementioned **Load Module** command.

To change the module loading mode

1. From the **Options** main menu, select **Module**. The **Module Options** dialog box opens.
2. Select a module in the tree and change **Module Load Mode** by selecting the appropriate radio button.

Modules, which are used very frequently, should be set in the *Always load* mode.

Modules, which are used only occasionally, should be set in the *Autoload* mode (or *Autoload with prompt* if you like to have more control on the loading behavior).

Modules, which are used only very rarely, can be put in the *Manual load* mode. Another frequent case where modules can be set into *Manual load* mode is when modules represent some software library, which is not expected to change. See the paragraph 1.6.1 Indexing below.

Advanced Concepts

Indexing

Indexing can be considered as an intermediate form of work, between working with a fully loaded module and working with the module unloaded.

When a module is unloaded/not loaded in the project, only necessary proxy elements are shown in the place of the module. However, there is a possibility to retain more proxies from the unloaded module than is kept by default. There is one case, where this functionality is particularly useful.

Consider the large software library module in a project. Let's say, only the various classes are used in the main project - some library classes are set as types of properties in the model classes, some model classes inherit from the library classes, etc. In this case, structural information of the library classes (their properties and methods) is not important. If proxies of all classes could be retained when the module is unloaded, this library module could be used in the main project in the unloaded state (saving a considerable amount of computer resources). The indexing feature allows achieving this functionality.

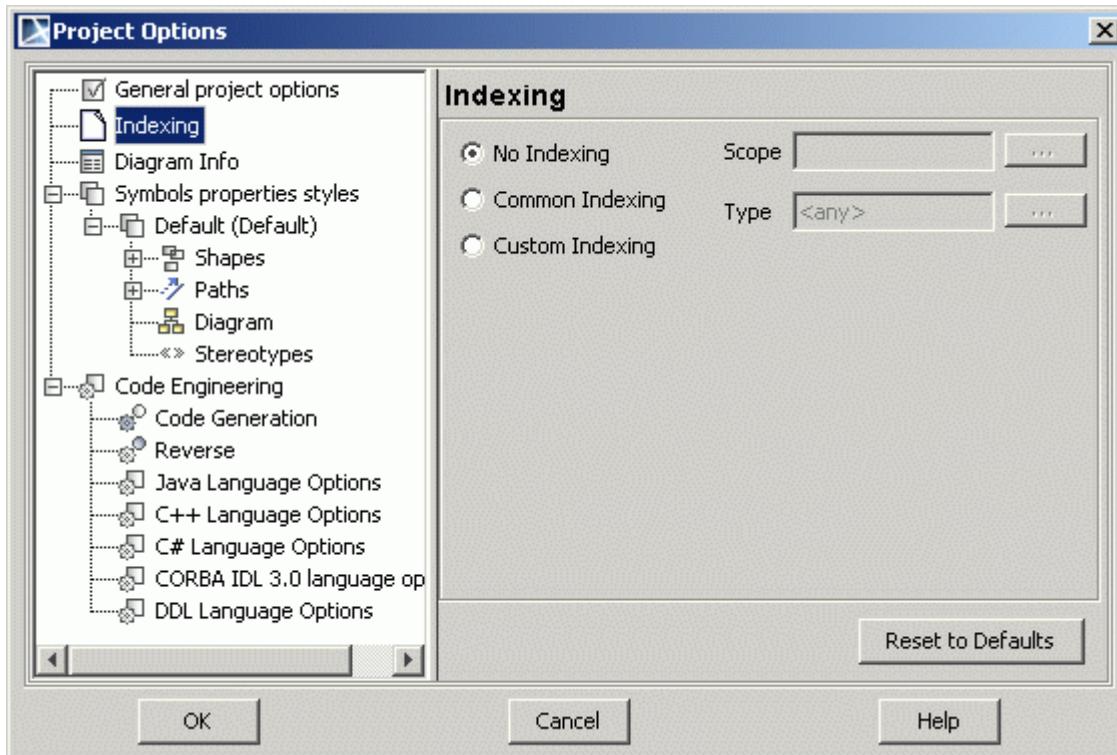
To specify indexing scope

1. First open the module as a project.

4 WORKING WITH PROJECTS

Project Partitioning

- From the **Options** menu, select **Project**. The **Project Options** dialog box opens. Select the **Indexing** section.



- Select the **Common Indexing** radio button. This enables indexing of the module and determines what information will be indexed.
- When common indexing is chosen, classifiers and their inheritance relationships will be indexed. If you want more elements to be indexed, select the **Custom Indexing** option and fine-tune what element types (properties, methods, etc) should be indexed. The more elements you select, the more elements will be accessible in the project using them as proxies. However, your gains in performance from the module unloading will also diminish. Hence, a balance is needed when customizing the index. It is usually better to use the common indexing variant. Click **OK**.
- In the project using the indexed module, go to the options of this module (**Options** menu - >**Modules**) and select the **Use Module Index** check box for that module.

Such setup causes all the classes of the module to be visible as proxies when the module is not loaded (it is also advisable to change the loading mode of this module to Manual loading).

These proxies can be used as normal model elements in the project using them, without ever loading the module. They can be set as types of properties of the classes in the main project, they can be set as an association ends, classes may be derived from them, etc. If you ever need more information from that module, you can load it at any time to access the full data in the module.

An example could be in the module holding standard Java classes (rt.jar was reversed into it). This module is large, having all the details of standard Java classes. Many of these details are unused in the project; frequently only class information is used in the project for modeling tasks.

Missing elements for the proxies (orphaned proxies)

Orphaned proxy is really an indication of the dangling reference. Appearance of the proxy is indication that some other elements for example from outside the module (i.e. elements in the main project or other modules) reference to the element in the module that was previously there but no longer exists. Element was deleted/removed/somehow made unavailable in the module. In such case MagicDraw creates so called "orphaned

proxy" in place of the missing element - a surrogate/not real element in place where real element was once in the past.

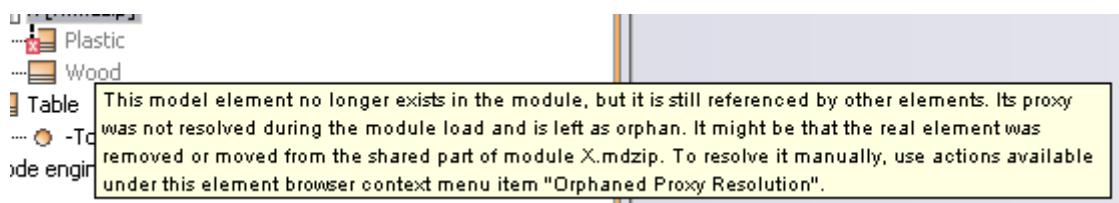
Displaying orphaned proxies in a project

MagicDraw distinguishes orphaned proxies from normal elements in the project:

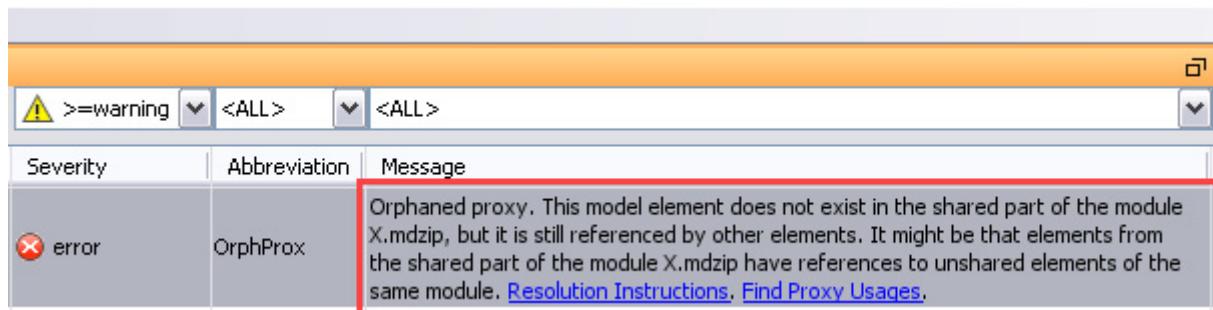
- Orphaned proxies are indicated with [!] adornment in the model tree.



- A special Tool Tip is displayed for every orphaned proxy. The Tool Tip describes the reason why the orphaned proxy has been created and gives a suggestion how to resolve it.



- Active Validation finds and marks orphaned proxies as validation errors in the projects.



Searching for orphan proxies

Run a search (**Ctrl+F**) and check the **Orphaned Proxies Only** check box. This will give you all the orphan proxies in your model in the search results.

Resolving orphaned proxies

Each orphaned proxy can be resolved.

To resolve the orphaned proxy:

- Right-click the orphaned proxy in the search result or Containment Tree and select the menu item **Orphaned Proxy Resolution**.

There are 4 actions you can do with the orphaned proxies:

- Clear Proxy Usages. This resolution can also be triggered by simply pressing the **Delete** button. The action clears all the references to this non-existing element, hence there is no more need for the proxy to appear.

2. Replace With. This resolution replaces all references to the orphaned proxy with references to the chosen element.
3. Create New Substitute. This resolution “resurrects” the element which is missing.
Note: This resolution is disabled when the orphaned proxy containing module is read-only.
4. Find Proxy Usages. This command works as a helper to the other resolutions. It lists all usages of the orphaned proxy in the project. This also helps to identify from which module there are usages of orphaned proxies.

Which one of them to choose depends on wherever these missing elements are necessary or are they unnecessary.

If they are necessary (i.e. you want these elements to exist; they must be there), this means that they were deleted by mistake at some point in the past. Maybe MagicDraw can not find the required module? Or elements were moved to some other module? Or maybe the old version of the module is used, which hasn't got these elements? The causes may be numerous.

MagicDraw can help to identify why orphaned proxy appeared, as it remembers, if possible, the last existence of the real element in the module. There are tree main causes of orphaned proxies:

- If an element was removed from the module or moved from the shared part of the module to the unshared part. In this case all such orphaned proxies shall be located under the module shared package where that module is being used.
Resolving such orphaned proxies must be done in the proxy-containing module. This module is provided in the Tool Tip of the orphaned proxy.
- If elements in the module of the shared part have some references to the elements that are located not in the shared part of the module. In this case, all such orphaned proxies shall be located not under the proxy-containing module package (which is provided in the Tool Tip of the orphaned proxy). To check if this is the case, open the proxy containing module as a project and execute the module dependency checker: **Tools->Dependency Checker**. If the dependency checker finds any errors, try to fix them.
- If elements in the module of the shared part have some references to the elements that are located in another project. This could happen if the module was mounted in the project in the read-write mode and somebody added a reference to the project data. In such case, there will be no orphaned proxies in that project, but if the module is being opened as a project, orphaned proxies are created for such another-project elements.

To resolve the orphaned proxies into the real element, you have to find. where real elements are in your modules/main project.

- If they were moved to some other module, you have to use that module into your main project (**File>Use module**).
- If real elements were in the part of the module which was unshared, share this module part again.
- If MagicDraw can not find the module on disk, it should ask you to provide path to it on project load.
- If elements were deleted from the module/main project, you have to roll back to the previous version of the module/project (in your version control system or Teamwork server, or wherever you back your files up into) which still had these elements.

Creating New Substitute

As the last resort, if you have no version saved, where these elements still exist, you can try to “resurrect” them. Right-click each proxy>**Orphaned Proxy Resolution>Create New Substitute**. MagicDraw will recreate the missing element from the bits of information it still has (which is not much - ID, name and kind of the element).

NOTE: This action might be disabled if the substitute to be created must be in the module, but this module read-only. In this case simply change module to read-write in the **Module Options** dialog (**Options>Modules**) and the action will be enabled.

Deleting orphan proxies

If these elements are unnecessary (i.e. you want them to disappear; they must not be there), this means that they were deleted properly. Now all we have to do is clear the dangling references, which still exist in the other modules/main project to these non existing elements.

To delete orphan proxy:

- Right-click each proxy>**Orphaned Proxy Resolution>Clear Proxy Usages**.
- Press **Delete** on the selected orphan proxy.

You can also do this en masse:

1. Run a search (**Ctrl+F**).
2. Check the **Orphaned Proxies Only** check box. This will give you all the orphan proxies in your model in the search results.
3. Select them all and press **Delete**.

When references to them are cleared, orphan proxies will disappear.

NOTES:

- Note that if references to these non existing elements are in the modules, which are mounted read-only, this action can not clear them. MagicDraw will list the modules that still has references to the orphaned proxy but currently could not be removed (because module(s) are mounted as read-only).
- Clear Proxy Usages or Del button resolution method works temporarily only- when project is loaded next time orphan proxies will reappear. To delete proxies fully you have to open each module as project and clean orphan proxies there.

Exporting a Project as MOF

This feature renews the MagicDraw metamodeling portfolio. MagicDraw is able to export/import the UML model into/from the MOF (both CMOF and EMOF) XMI format.

To export a project as a MOF

- To export the model into the MOF, from the **File** main menu, select **Export** and then **MOF**.

To import a project as a MOF

- To import the MOF from the **File** main menu, select **Open Project** and select to open *.emof or *.cmof file.

Data which can not be exported

- Diagram data can not be exported.

4 WORKING WITH PROJECTS

Exporting a Project as MOF

- None of the model features that are available only in UML will be exported (behavioral models in particular, etc).

MOF Domain model

Domain model for CMOF can be glanced at the MOF2.0 spec, 06-01-01.pdf document from OMG.

MOF export

Under File->Export->MOF group there are 2 items – MOF Whole Model, MOF Selection.

Exporting the whole MOF model

To export the whole model into the MOF, from the **File** main menu, select **Export, MOF**, and then **MOF Whole Model**. The **Select Target File** dialog box opens (see Figure 75 on page 155).

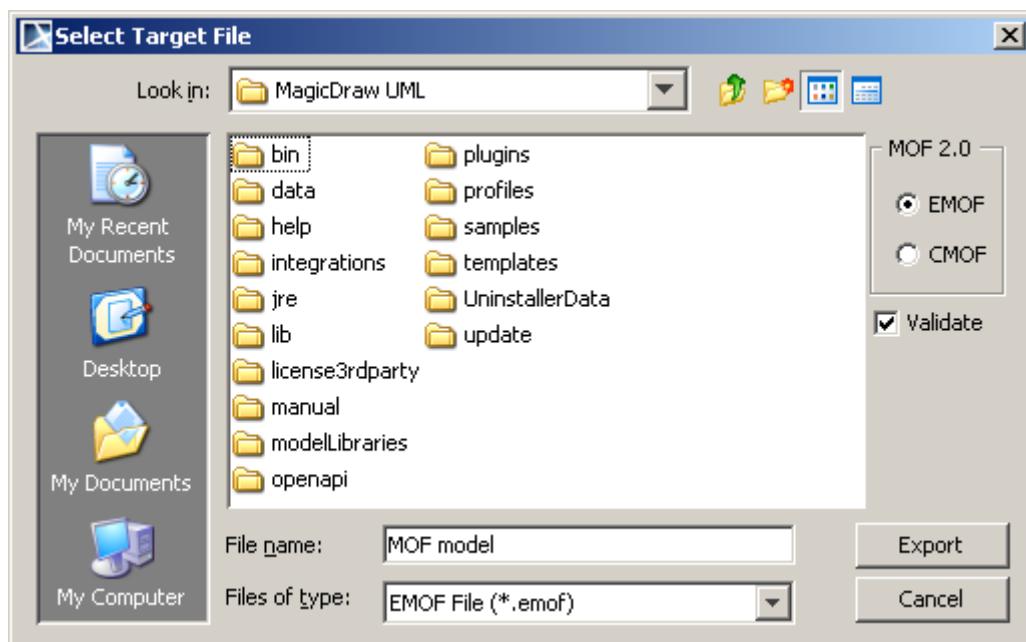


Figure 75 -- The Select Target File dialog box

Select directory to export the file, type the file name.

At the right side of dialog box you can choose to export EMOF or CMOF. The extension of file is by default .cmof or .emof correspondingly. After this the whole model is exported except the model parts marked as auxiliary resources (UML standard profile, Metamodeling profile, etc).

Select or clear the **Validate** check box for validating the MOF model export. When export runs, the exported model is also checked using the appropriate (CMOF/EMOF) validation suite. For more information about MOF model validation, see “Validation constraints” on page 156.

Exporting the selected part of the MOF model

To export the selected part of the model into the MOF, from the **File** main menu, select **Export, MOF**, and then **MOF Selection**. The **Select Packages to Export** dialog box opens (see Figure 76 on page 156). Select one or

4 WORKING WITH PROJECTS

Exporting a Project as MOF

multiple packages to choose the model part to be exported. After this the selected model part (package granularity level) is exported to CMOF or EMOF correspondingly.

After the package selection, click OK and the Select Target File dialog box opens. Detailed description, see above “Exporting the whole MOF model” on page 155.

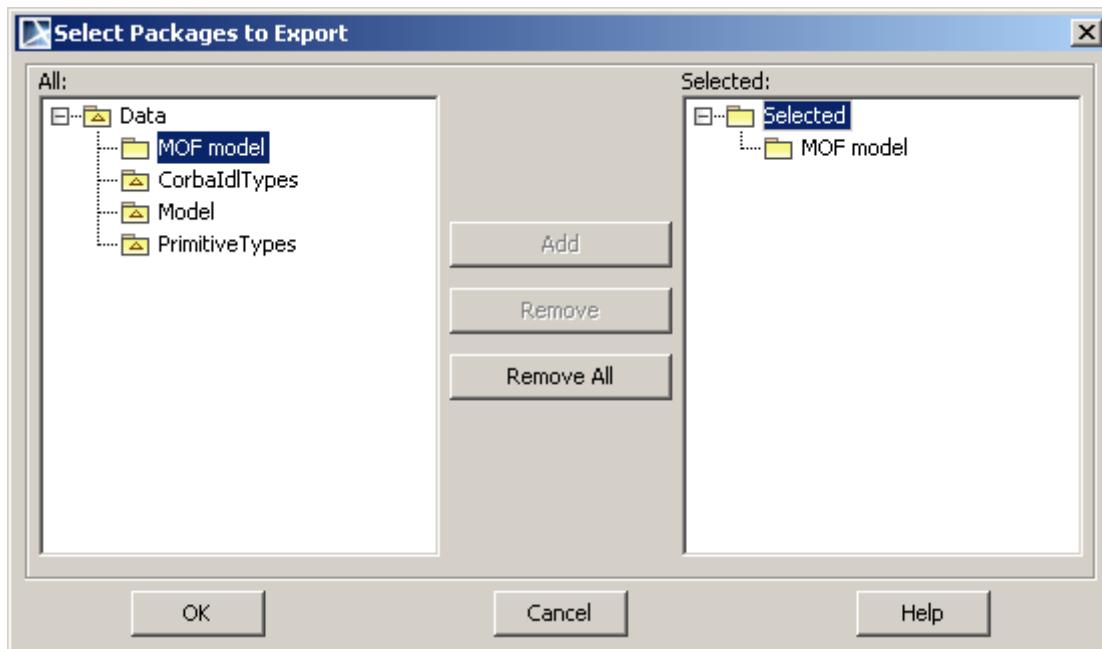


Figure 76 -- The Select Packages to Export dialog box

Metamodeling template

You can create a new project using a Metamodeling template. Template contains Metamodeling profile usage. It can be used as a starting point for developing your own metamodels. Also look at the Ecore Metamodel Example in the

s directory for a larger example of how to use the profile.

For more information about templates usage, see "Creating a new project from a previously created template", on page 122.

Validation constraints

There are 2 new validation suites for testing the model being exported to MOF (1 each for CMOF and EMOF). These suites contain batches of rules, which display warnings about elements not being exported. Rules are run together with model export (if the Validate check box is checked).

Rules are warnings only - they warn about elements not being exported (diagrams, behavioral elements etc) but do not preclude the model from being exported. Unsuitable elements are simply skipped.

Ecore Support

Introduction

The Ecore model is the one type of models that Eclipse Modeling Framework (EMF) supports. This type of models is also colloquially called EMF models (even though EMF supports many types of models, e.g., the UML model). Ecore models can be used for various purposes. Ecore models can be used for metamodeling purposes, where its expressive power is roughly similar (slightly higher) to EMOF. Ecore can also be used for simple class modeling purposes as a subset of UML.

MagicDraw has the Ecore **export** functionality. Ecore models, prepared in MagicDraw, can be exported as Ecore models for the further processing with other EMF tools - a generation of model repositories, a code, XML parsing and storing, etc.

Preparing Ecore Models

To start the modeling in Ecore, you can use the “new project from template” functionality. Select **File > New Project** from the main menu, then, in the **New Project** dialog, the **Other** domain, choose the **Project from Template** icon, and select the **Metamodeling > Ecore Template** template from the template list.

If you already have some UML model or a CMOF/EMOF model, which you want to develop into the Ecore model, you can just attach the Ecore profile from the MagicDraw's profiles folder (from the main menu, select **File > Use Module**, choose **Ecore_Profile.xml** in MagicDraw profiles). If your Ecore model references some standard Ecore elements (such as standard data types - like **EShort**, or standard metaclasses - like **EStructuralFeature**), you also need to use the standard Ecore library (from the main menu, select **File > Use Module**, choose **Ecore.mdzip** in MagicDraw model libraries).

There are no separate custom diagrams for the Ecore model editing. You can use the same Class diagrams as you use for your UML models. Since Ecore is almost a subset of UML (with a few additions), familiar UML elements are used for the modeling. You can also develop Ecore models without using the Ecore profile at all - if your Ecore model uses only UML-specific information, you can develop it using plain UML and export it to Ecore without a problem.

Ecore is even more similar to EMOF. The Ecore profile is built on and depends on the Metamodeling profile (which implements the EMOF modeling). Element properties, that are relevant to MOF (Package namespace URI and Property isID settings) are also relevant in Ecore. You can also have the one model and export it as Ecore and then export it also as EMOF.

Class, DataType, Enumeration, Package, Operation, Parameter have a direct one-to-one correspondence between UML and Ecore.

Ecore has two flavors of structural features - EAttribute and EReference where UML has just one - Property. Fortunately a differentiation between an attribute and a reference is unambiguous and automatically resolved (the property, whose type is a data type is treated as EAttribute; the property, whose type is a class is treated as EReference), hence the user does not need worry about this - he/she can simply use properties.

There are no standalone Association, Generalization model elements in Ecore, but there is analogous information in Ecore (two EReferences, pointing to each other by their **opposite** fields is equivalent to the association; the EClass::eSuperTypes field is equivalent to a generalization). Hence it is possible and meaningful to draw associations and generalizations in your model for exporting this information to Ecore.

Ecore generics (templates) are also supported. You can use the UML template support to model Ecore generics. While the modeling is not trivial (and not one-to-one due to weak semantics of Ecore's EGenericType), it is possible to model all cases of template types, even ones with complexly nested type bounds - like, for example, **SortedList<T extends Comparable<? super T>>**.

4 WORKING WITH PROJECTS

Ecore Support

Your models can also contain any other UML elements, which are not present in Ecore. These elements are simply skipped during the export to Ecore. A warning is given about these elements - see Section "Validation" on page 164.

There are few Ecore-specific properties, which are brought in when the Ecore profile is used. These properties are used to capture Ecore specific information, not existing in UML. MOF-specific properties are also relevant for Ecore. These special properties are the following:

- The Ecore package has an additional **nsUri** (shared with MOF models) and **nsPrefix** fields:

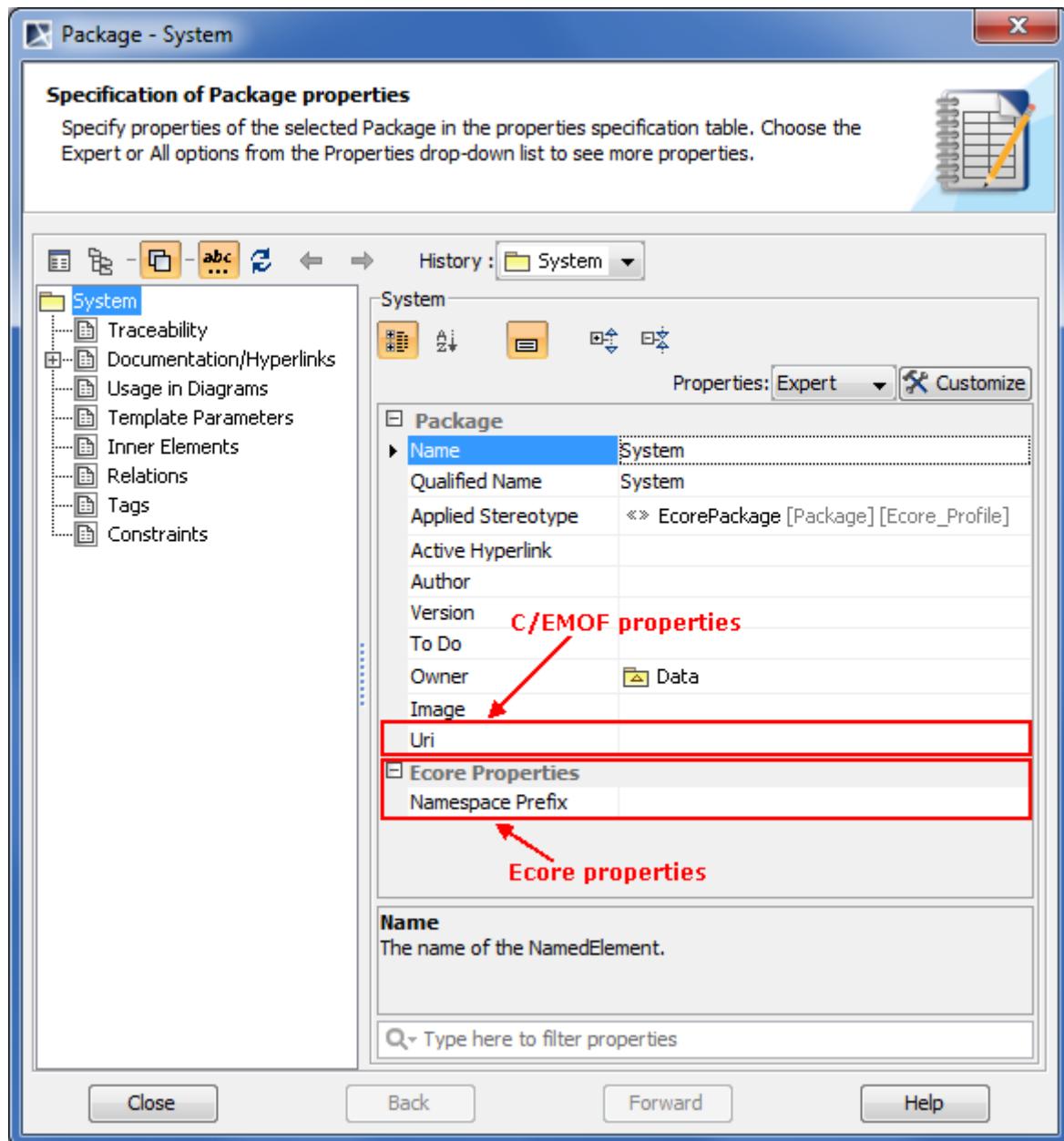


Figure 77 -- Ecore Package Additional Properties

4 WORKING WITH PROJECTS

Ecore Support

- The Ecore classifier (class, data type, enumeration) has an additional **instanceClassName**, **instanceTypeName** fields:

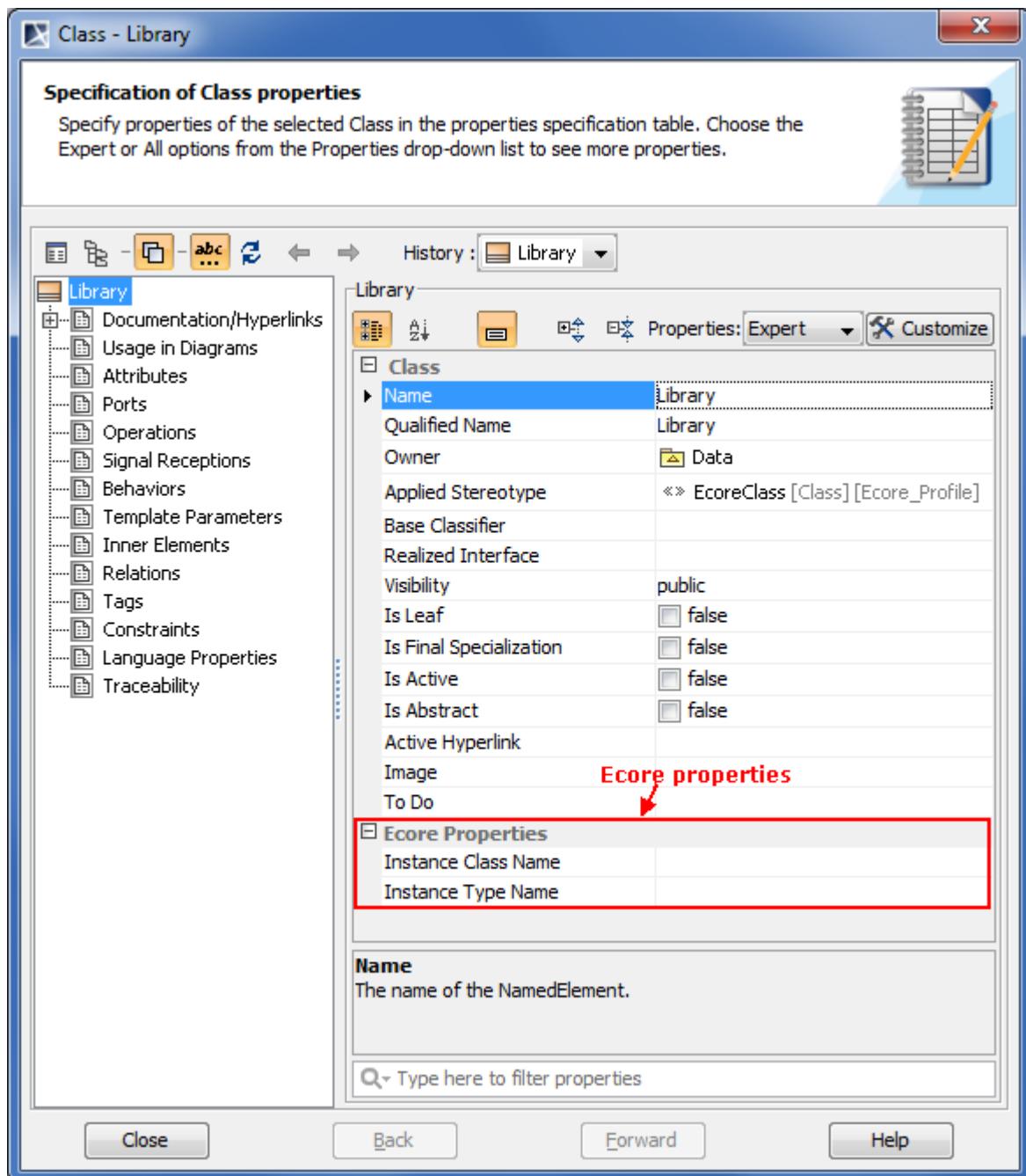


Figure 78 -- Ecore Classifier Additional Properties

4 WORKING WITH PROJECTS

Ecore Support

- Ecore attributes and references (modeled as the UML property) have additional **volatile**, **transient**, **unsettable**, **ID** (only for attributes, shared with MOF models), and **resolveProxies** (only for references) fields:

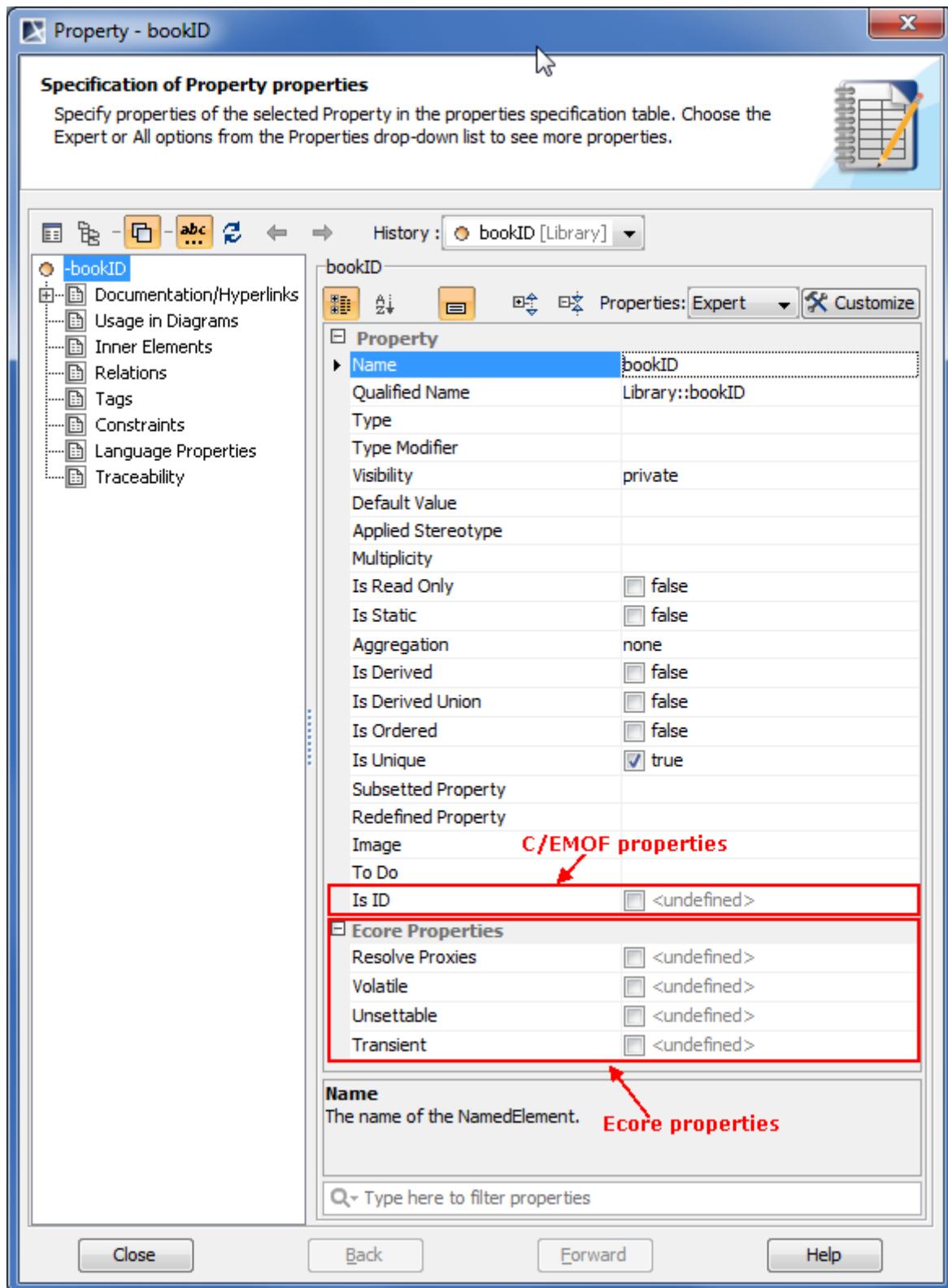


Figure 79 -- Ecore Attribute, Reference Additional Properties

Annotation modeling

Ecore annotations are modeled as UML comments. For simple annotations no additional actions are necessary.

However Ecore annotations have the more powerful semantic than UML comments - they can have an internal substructure. In particular they can have an additional key-value map. For this additional information, there is a special <<EcoreAnnotation>> stereotype, that can be applied on an annotating comment. After this application, the key-value map can be entered in a separate node in the **Specification** window. Key-value pairs are stored as internal subcomment elements of the annotation. This process is illustrated in the following pictures :

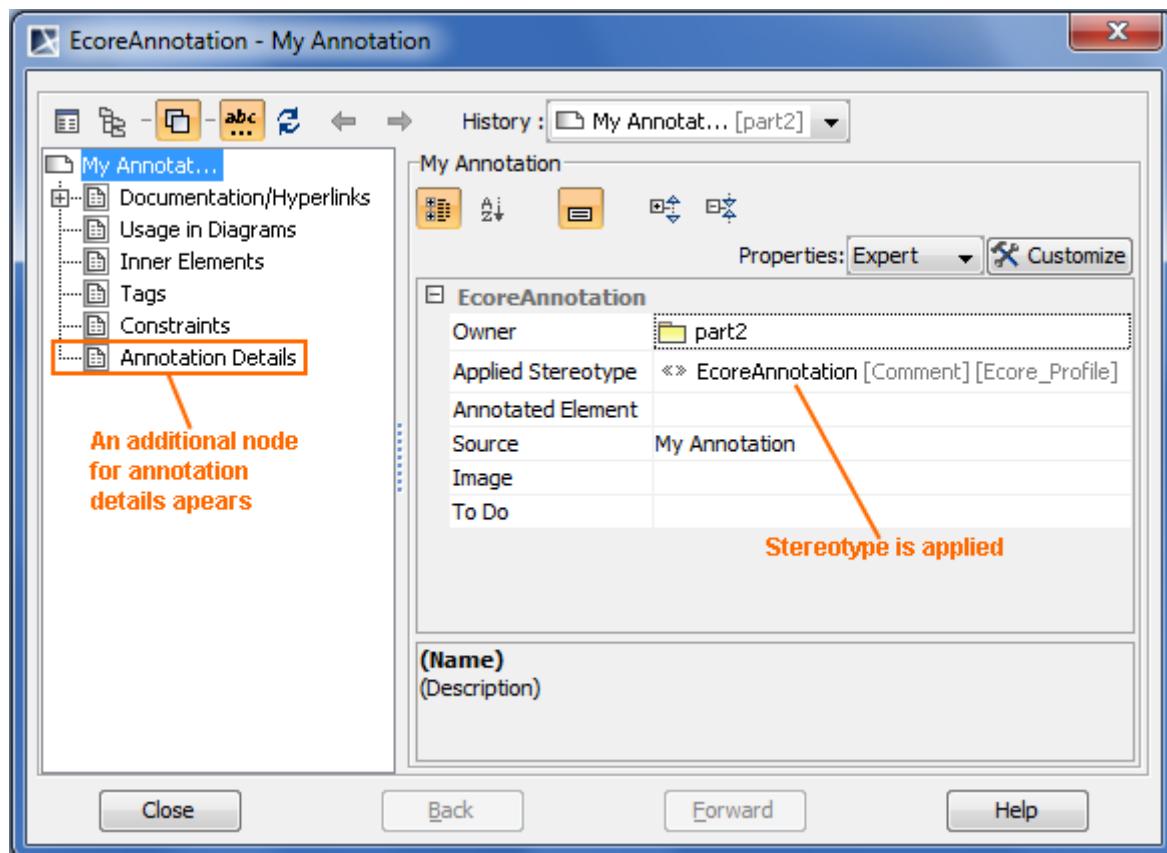


Figure 80 -- Applying EcoreAnnotation Stereotype

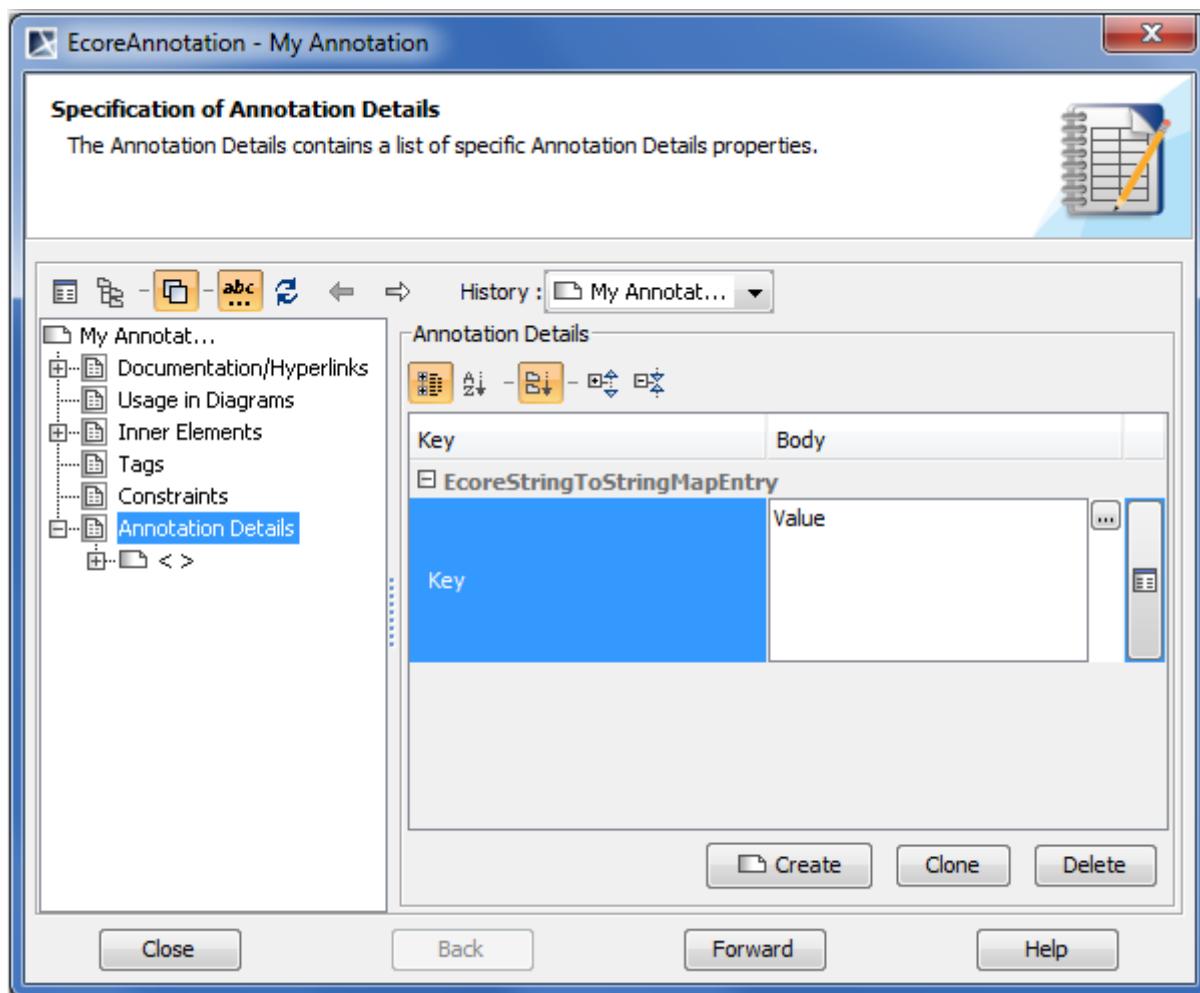


Figure 81 -- Filling In Annotation Details - Key - Value Map

Exporting

The Ecore model exporting is very similar to the EMOF/CMOF model exporting. After the Ecore model is prepared, you can export either whole model or only selected package(s) to the *.ecore file format.

To export the whole model, use the **File > Export To > EMF Ecore File > Ecore Whole Model** menu item. Choose the destination file in the opened dialog and press **Export**.

4 WORKING WITH PROJECTS

Ecore Support

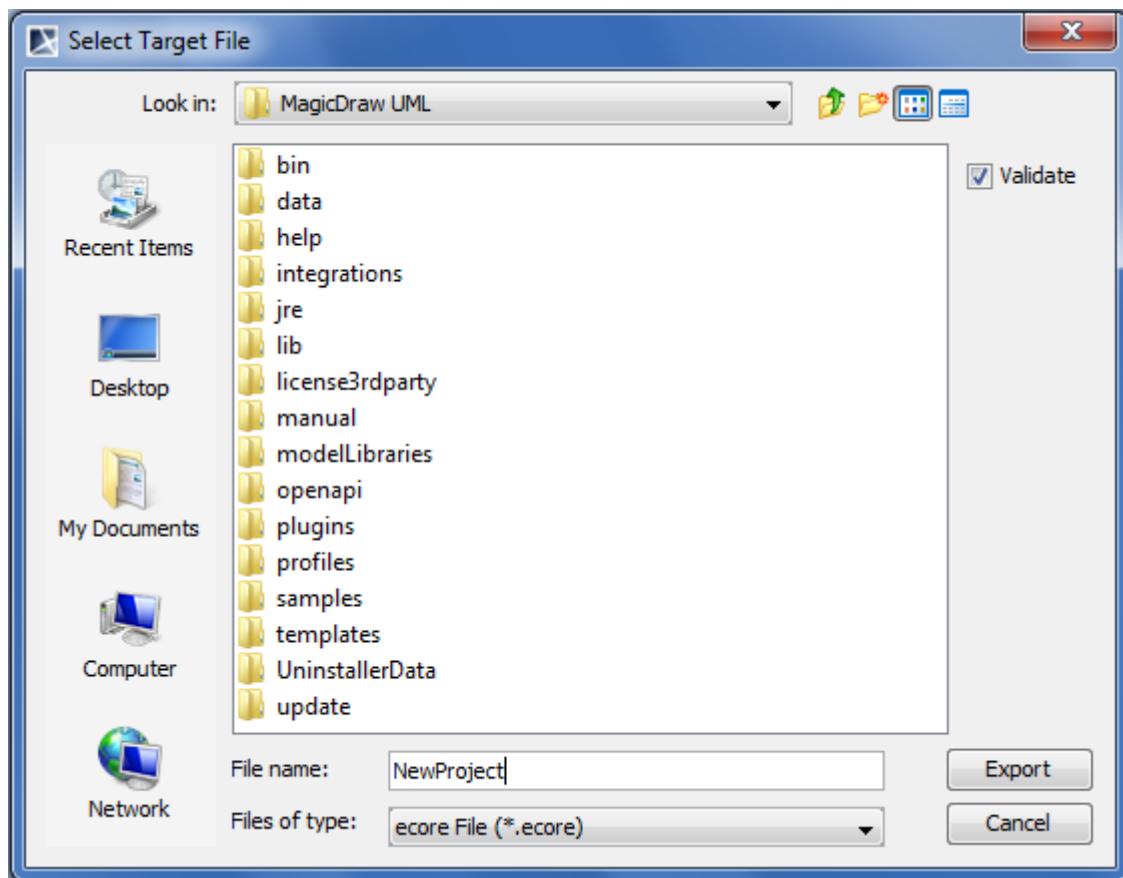


Figure 82 -- Selecting Target File for Ecore Export

To export a part of the model, on the main menu, select **File > Export To > EMF Ecore File > Ecore Selection**. In the opened dialog, select packages you want to export. Then specify the target file and the location.

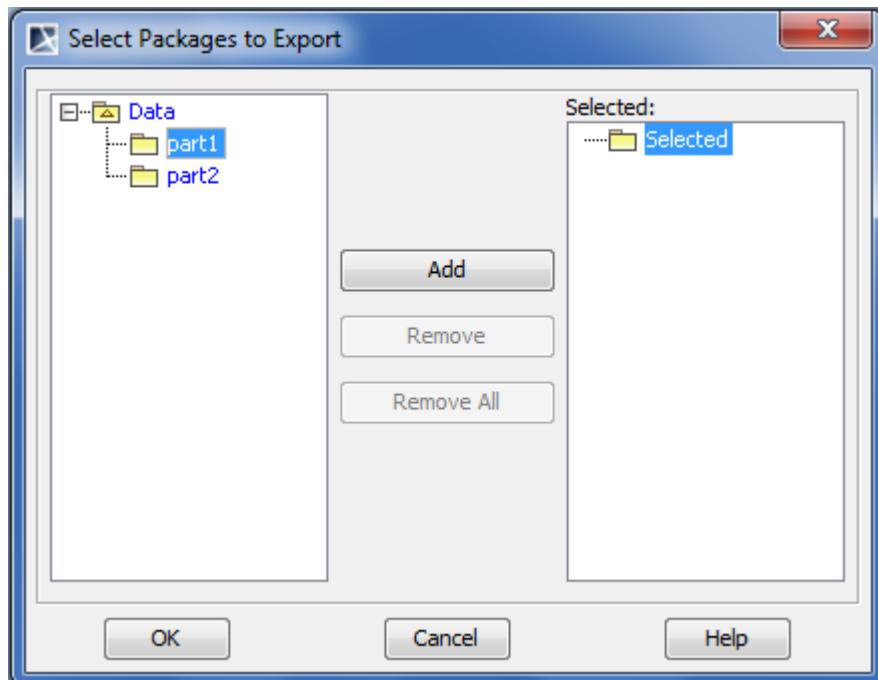


Figure 83 -- Selecting Part of the Model for Ecore Export

You can validate the model being exported - see Section "Validation" on page 164.

Standard data types from UML - **String**, **Boolean**, **Integer**, **UnlimitedNatural** are exported as standard Ecore data types - **EString**, **EBoolean**, **EInt**, **EInt** correspondingly.

Standard types from the MagicDraw profile - **boolean**, **byte**, **char**, **date**, **double**, **float**, **int**, **long**, **short** are exported as Ecore types - **EBoolean**, **EByte**, **EChar**, **EDate**, **EDouble**, **EFloat**, **EInt**, **ELong**, **EShort** correspondingly. The **void** type is exported as an absence of the type.

References to Ecore model elements, defined in the "Ecore" model library in MagicDraw (standard datatypes like **EInt**, metaclasses like **EStructuralFeature**) are exported as standardized Ecore references to Ecore meta-model elements (the resource identifier part of the **Href** is <http://www.eclipse.org/emf/2002/Ecore>).

Exporting preserves module structure

When you export a project, only direct contents of that project are exported, any modules used in the project are not exported. References to modules are exported as Ecore references. Hence the export result is the Ecore file/model containing the direct content of the MagicDraw project and referencing other Ecore files/modules existing in the MagicDraw project. To export modules to Ecore, you have to open each module as a project and trigger the Ecore export.

NOTE It is recommended to export modules to Ecore before exporting the main project.

There is one important complication when dealing with projects consisting of modules. It arises because Ecore references, crossing a resource boundary (when the element in one file references the element in another file), are qualified-name based, not id based - as is the case with CMOF, EMOF or UML. When exporting the project with references to elements in other projects/modules, the export tool must know the full path of elements in the module's Ecore file. This information cannot be obtained normally. The following approach is adopted to resolve this problem:

- When the module is exported to Ecore ("module" here means - any project, which has its contents shared - because any of such project can be used as a module in another project), information about paths/qualified names of the elements is stored in special stereotypes/tags of shared packages (the **EcoreExportServiceInformation** stereotype, the **ecoreExportPath** tag). This also means that the project is modified during the export. This is a little counterintuitive, but we need to save this information for the later usage. You need to save your MagicDraw project afterwards, to preserve these changes.
- When a project is exported to Ecore and this project references elements from the module, the exporter uses information about the element path, which was stored when the module was exported to Ecore. If this information is missing (e.g., the project is being exported before modules are exported), the exporter tool tries to guess the correct element path in the module and gives a warning in the messages window about this.

Because of this complication, it is strongly recommended that modules should be exported to Ecore first, and projects, using those modules, should be exported after that.

Validation

You can check the Validate check box to run a validation during an export - in the same manner as the validation is done for the CMOF/EMOF export. The validation will be run during the export, and all UML elements, that are not suitable for the Ecore export, will be highlighted. The validation just produces warnings about not suitable/skipped elements; having warnings does not preclude the export itself.

You can also run this validation suite at any time while you are developing the Ecore model.

To run a validation

1. From the main menu, select **Analyze > Validation > Validate**, The **Validation** dialog will open.

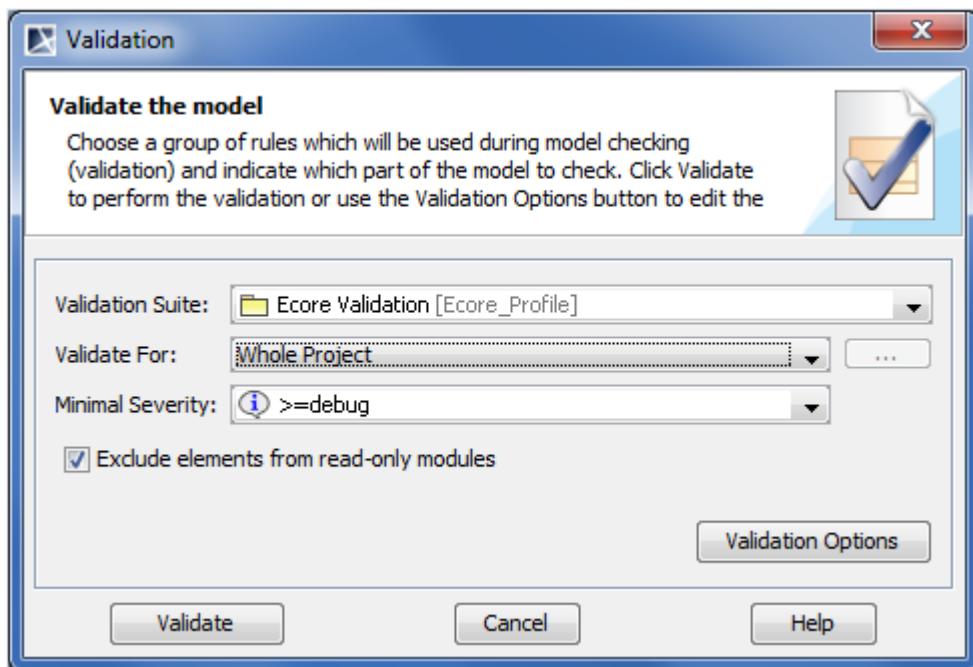


Figure 84 -- Validation dialog

2. In the **Validation Suite** drop-down list, select **Ecore Validation**.
3. In the **Validate For** drop-down list, select the scope of the validation.
4. In the **Minimal Severity** drop-down list, select the level of severity.
5. Click **Validate**.

Importing

There is currently no possibility to import Ecore files directly. Ecore files could be imported using the EMOF import functionality. In the Eclipse environment, open the *.ecore file and save it as *.emof. Eclipse will export the Ecore file in the EMOF format, which can be opened in MagicDraw. Please note that this indirect way loses some Ecore-specific model details, which are not existent in EMOF.

Working with Standard Profiles

Standard Profiles as System Resources

All profiles and libraries, which are bundled with MagicDraw family products are considered as standard/system resources, which are non-modifiable and are essential for the correct tool behavior.

We highly recommend not to modify our provided standard profiles and libraries as it could cause problems on version updates, plugins, core MagicDraw tool malfunctions, and model corruptions.

Users will be warned on any intentional or unintentional attempt to modify profiles in the following ways:

- Open a profile as a project.
- Use a module in the read-write mode.

- Import a module into a project.
- Merge projects.
- Any other cases.

Plugin and Profile Versions

Standard profiles are usually upgraded to support newest versions of the specification of standards they represent (e.g., SysML 1.2 or UML 2.3 versions) in the every MagicDraw release cycle. The MagicDraw application, the plugins code, and the behavior are modified accordingly to reflect these changes.

There is a very high probability that the new version of MagicDraw or plugins can not work with older or newer profile versions and may cause an unpredictable behavior or even model distortions. For example, MagicDraw SysML 16.5 requires to use SysML Profile of the version 16.5, as it could malfunction when using SysML Profile from the version 16.0 or 16.6.

To protect the user from such cases, every MagicDraw project starting from the version 16.6 knows which profiles or plugins versions were used to create it and are required to load data correctly.

Every standard profile has the version number. Normally it is the same as the MagicDraw (or a plugin) release version number.

MagicDraw requires to use the corresponding version of the profile with the corresponding software version. You will get a warning, if your used plugins or profiles are obsolete or you miss some plugins or profiles .

Resource Manager with selected missing resources will be launched automatically, so you will be able to install missing plugins/profiles in few clicks.

If new versions of plugins are not purchased or you simply do not want to install it, but need to take a look at the project content, warnings may be ignored and the project may be loaded. In this case, proxy elements for missing profile elements will be created to retain missing references. Missing custom diagrams will be loaded as regular UML diagrams or will be restricted for a review. Do not save such project! Use it for the preview only.

Old projects will be loaded/converted without any warnings, if you have newest versions of corresponding plugins and profiles as MagicDraw is always backward compatible.

Standard Profiles in Teamwork Server

Standard / system profiles and modules are not added into Teamwork Server, because every user has recent versions installed locally. As profiles/modules are non-modifiable, the version control is not needed. It solves multiple profiles usage / modification / update issues in the teamwork and at the same time increases the teamwork performance, as standard profiles will not be transferred via networks.

For more information about Standard Profiles in Teamwork Server, see Section “**Working with Standard Profiles in Teamwork**” [MagicDraw Teamwork UserGuide.pdf](#).

Information about an updating UML Standard Profile due to the migration to UML 2 is presented in the Teamwork Server readme file. You will find this file in the <MagicDraw Teamwork Server installation directory> folder.

5 DIAGRAMMING

This chapter offers an overview of working with diagrams and symbols. In general, the topics discussed apply to all supported diagram types.

When working with diagrams it is helpful to keep in mind the following concepts:

- A **shape** refers to a notation of a model element, such as a package, class, state, use case, object, etc.
- A **path** refers to the notation for the various kinds of relationships such as associations, aggregations, dependency, message, and links.
- Both paths and shapes are defined as **symbols**.

In the “Diagramming” chapter, you will find the following sections:

1. "Working with Diagrams", on page 167
2. "Working with Shapes of Model Elements", on page 195
3. "Working with Paths and Relationships", on page 198
4. "Smart Manipulation", on page 202
5. "Selection and Multiple Selections", on page 204
6. "Copying/Pasting Text or Images to Diagrams", on page 206
7. "Nesting Image Shapes", on page 207
8. "Dragging, Copying, Cutting, and Pasting", on page 207
9. "Zooming", on page 214
10. "Using the Grid", on page 215
11. "Layout", on page 216
12. "Showing Diagrams in Full Screen", on page 229
13. "Floating Diagram Window", on page 230
14. "Saving as an Image", on page 230
15. "Printing", on page 232

Working with Diagrams

[View Online](#)
[Demo](#)

MagicDraw Basics

Diagram Basics

To create a new diagram

-
- From the toolbar:
Click the desired diagram button on the **Diagrams** or **Custom Diagrams** toolbar. The **Create Diagram** dialog opens. Type the name of the diagram and select or create a package where you wish to create your diagram.
 - From the **Diagrams** menu:
On the main menu point to **Diagrams** and then select the desired diagram. The corresponding

Diagrams dialog opens. Click **Add**. The **Create Diagram** dialog box opens. Type the name of the diagram and select or create a package where you wish to create your diagram.

- From the Model Browser:
Right-click the desired model element in which you would like to create a diagram and, from the shortcut menu, select **New Diagram**. Type the name for diagram directly in the Model Browser.
- From the model element's Specification window:
Open the **Inner Elements** tab in the Package's, Profile's, Model's, or other element's Specification window. Click **Create**. On the shortcut menu point to **New Diagram** and then select the diagram type. Define the diagram name, enter documentation, define stereotypes, and add tagged values and/or constraints.
- Using a diagram creation wizard:
On the main menu click **Diagrams > Diagram Wizards** and then select the appropriate wizard depending on what diagram you want to create. Wizards for creating a Class, Generic Table, Package Dependency, Package Overview, Hierarchy, Activity Decomposition Hierarchy, Realization, Sequence Diagram from Java Source, Content diagram are available.
- Using a diagram creation wizard (from the Model Visualizer dialog):
On the main menu click **Analyze > Model Visualizer**. The **Model Visualizer** dialog with the list of available diagram creation wizards will open. Select one and click the **Start** button to open the appropriate diagram creation wizard.

NOTE You should create a diagram in a package, model, or profile.

To open a diagram

- From the Model Browser:
Select **Open** from the diagram shortcut menu or double-click the diagram.
- From the Diagrams menu:
On the main menu point to **Diagrams** and then select the desired to open diagram type. The corresponding **Diagrams** dialog opens. Select the diagram you want to open and click the **Open** button.
- From the Content Diagram (available in the Standard, Professional, Architect, and Enterprise editions), if the diagram is added to the table of contents or a shape of the diagram is drawn on the diagram pane.
- Double-click a model element, to which the diagram is assigned.

TIP 1! To load all diagrams that have been created in the project, from the **Diagrams** menu, select **Load All Diagrams**.

TIP 2! To open the list of diagrams that have been most recently closed, from the **View** menu, select **Recently Closed Diagrams** and double-click the diagram you want to open. The F12 key also activates this command.

TIP 3! In the **General** pane of the **Environment Options** dialog box, you can select a method for loading diagrams while opening a project. Three options are available:

- **Load all Diagrams** – loads all diagrams that exist in the project.
- **Load Only Open Diagrams** – loads only diagrams that were not closed in earlier usages of the project.
- **Do not Load Diagrams** – all diagrams are not loaded and closed after opening a project.

5 DIAGRAMMING

Working with Diagrams

To close a diagram

- Click the **Close** button on the diagram pane or select **Close Diagram** from the diagram shortcut menu.

Define a diagram in its Specification window.

To split diagrams in new horizontal or vertical group

- Select the open diagram tab and drag it to the diagram pane. The shortcut menu with commands appears.



2. Select **New Horizontal Group** or **New Vertical Group** to split diagram pane and have more than one diagram opened at the same time.

Commands can be found in the diagram tab shortcut menu, when so many diagrams are opened that diagram tabs are filled in the toolbar line.

To show the diagram owner on the diagram tab

1. In the package, create diagram.
2. Select the open diagram tab and right-click to open the shortcut menu.
3. Select the **Show Owner** check box. The package name appears on the diagram tab.

Diagram Specification window

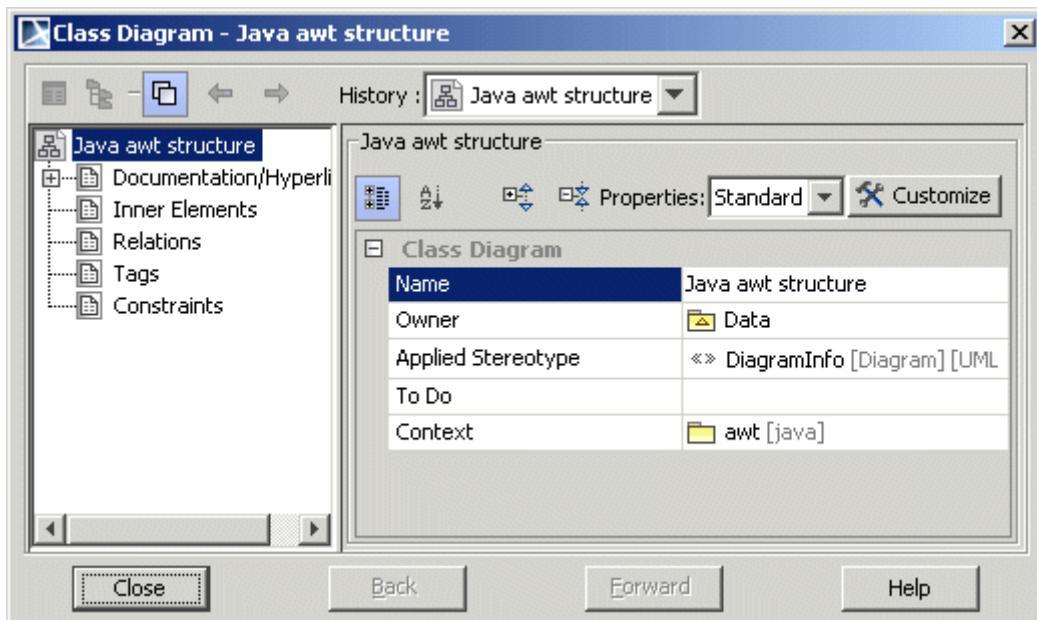


Figure 85 -- Diagram Specification window

To rename a diagram

- Open the diagram Specification window and type a new diagram name.

5 DIAGRAMMING

Working with Diagrams

To change the diagram context

- Open the diagram Specification window, select the **Context** property value, and click the “...” button. Then, the element Selection dialog appears. Select the new diagram context. More information about element Selection dialog, see Section “Selecting an Element” on page 278.

Diagrams Dialog

The **Diagrams** dialog is used for the following purposes:

- For viewing the owner of the diagram.
- For creating a new corresponding type of diagram.
- For editing the name and other characteristics of the diagram.
- For removing a diagram from the project.
- For opening a diagram.

To open the corresponding **Diagrams** dialog box

From the **Diagrams** menu, select one of the diagrams. Depending on the type of diagram, the dialog box that opens has a corresponding title.

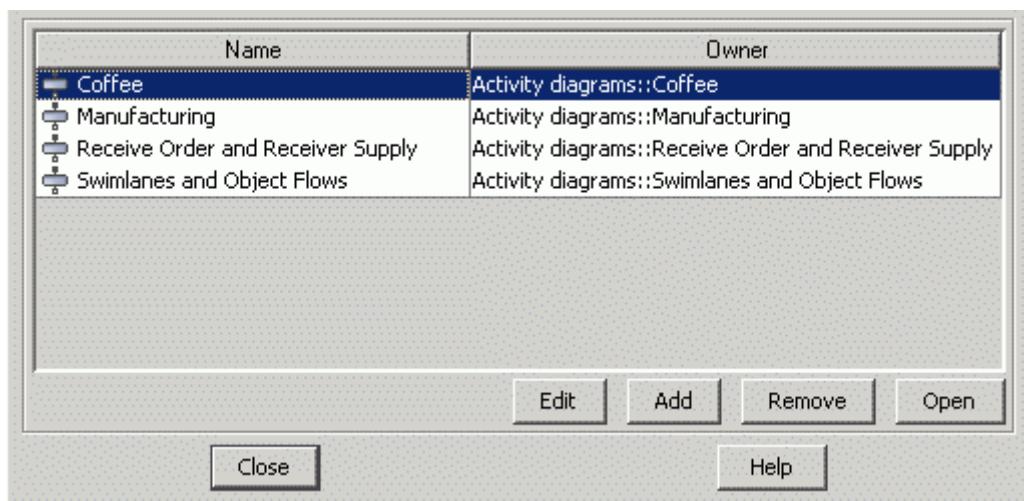


Figure 86 -- Diagrams dialog box

The **Diagrams** dialog box contains the following elements:

| Element name | Function |
|---------------|---|
| Name | The names of all created corresponding diagrams in the open project. |
| Owner | The name of the package that owns the diagram. |
| Edit | The Diagram Specification dialog box opens. Type the diagram name, select a package, and click OK . |
| Add | Creates a new diagram. The Create Diagram dialog box opens. Type the diagram name, select a package, and click OK . |
| Remove | Deletes the selected diagram. |
| Open | Opens the selected diagram. |
| Close | Saves all actions performed during the session and exits the dialog box. |
| Help | Displays the MagicDraw Help. |

Diagram Properties

Customize the diagram style (color, grid) in the **Diagram Properties** dialog.

To open the **Diagram Properties** dialog

- Select **Diagram Properties** from the diagram shortcut menu.
- On the **Edit** menu, point to **Symbol**, and then click **Diagram Properties**.
- Press SHIFT+ENTER.

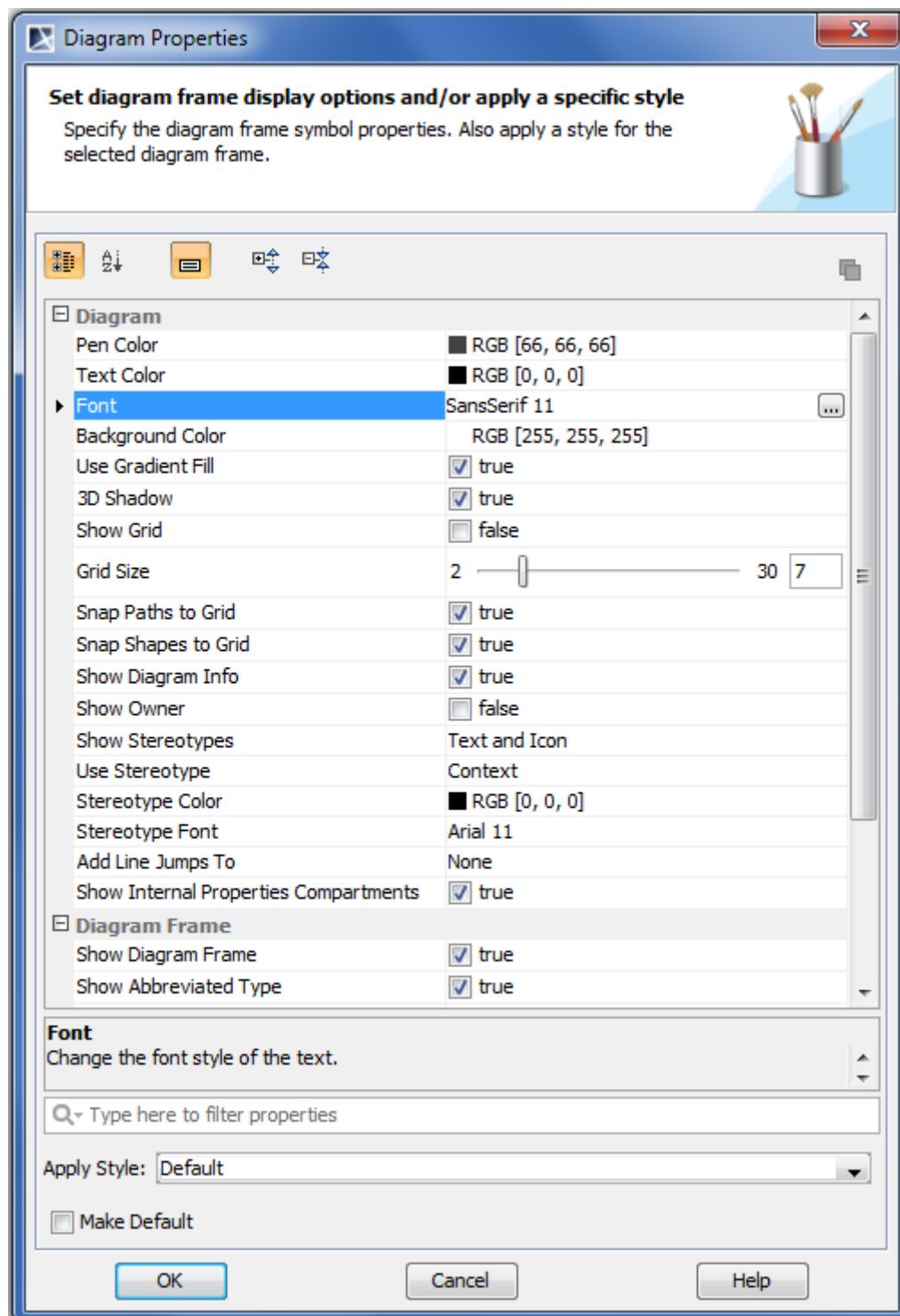


Figure 87 -- Diagram Properties dialog

The **Diagram Properties** dialog contains the following properties.

| Property name | Description |
|--|--|
| Pen Color | Change the color of paths or diagram frame line. |
| Text Color | Change the text color of the diagram frame name. |
| Font | Change the font style of the text. |
| Background Color | Set the diagram background color. Click the “...” button. The Color dialog box opens. Set the color in one of three different ways: using Swatches , HSB , or RGB tabs. |
| Use Gradient Fill | Sets the gradient for the shapes fill color. |
| 3D Shadow | Displays 3D shadow on symbols. |
| Show Grid | Shows the grid on the diagram pane. |
| Grid Size | Sets grid size from 2 to 30. |
| Snap Paths to Grid | Uses the grid on the diagram for drawing paths. |
| Snap Shapes to Grid | Uses the grid on the diagram for drawing shapes. |
| Show Message Numbers | Displays message numbers on the diagram. NOTE: This property is available only for the activity, communication, and sequence diagrams. |
| Use Advanced Numbering | Displays more detailed message numbering on the diagram. NOTE: This property is available only for the activity, communication, and sequence diagrams. |
| Show Diagram Info | Displays a table on the diagram that contains information about the diagram (Diagram name, Author, Creation date, Modification date, etc.). You can specify what information you want to include in the Project Options dialog (on the main menu select Options > Project). |
| Show Owner | Displays diagram owner on the diagram tab. |
| Show Stereotypes | Specifies the representation of the diagram stereotype that is displayed in the diagram frame header. |
| Use Stereotype | Utilizing the Use Stereotype property, you may choose to display context stereotype or diagram stereotype in the diagram frame and on the diagram shape. For more information about the Use Stereotype property, see “To display the context stereotype icon instead of the diagram stereotype icon on the diagram frame” on page 175. |
| Stereotype Color | Changes the color of the stereotyped text label. |
| Stereotype Font | Changes the font style of the stereotyped text label. |
| Diagram Orientation | Available in the activity and business process diagrams for correct rectilinear path braking and drawing paths between shapes from side to side, or from bottom to top shape borders. |
| Add Line Jumps To | Specifies how line jumps are added to link intersections on the diagram. |
| Show Internal Properties Compartments | Displays SysML internal properties compartments. |

Diagram Name and its Context Name Synchronization

The diagram name and its context name are synchronized automatically.

For example, create an Activity diagram. Type a name for the Activity diagram, for example, Receive. The name of the Activity automatically changes to Receive. And conversely - change the name of the Activity and the Activity diagram name will be changed automatically. This is synchronization of a diagram name and its context name:

Synchronization works in the following cases:

- Activity and Activity diagram inside.
- Interaction and Communication or Sequence diagram inside.
- (Protocol) State Machine and (Protocol) State Machine diagram inside.
- Class and all available inner diagrams inside.

To turn off the synchronization

Clear the **Synchronize the diagram name with it's context name** check box in the **Environment Options** dialog box, **General** branch, and **Editing** group.

NOTE If the second diagram will be created in the branch, diagram names will not be synchronized.

Diagram Frame

As of MagicDraw version 12.0, a UML diagram has the content area that is restricted by the diagram frame as it is stated in UML specification. The diagram frame is primarily used in cases where the diagrammed element has graphical border elements (like ports for classes and components, entry/exit points on statemachines).

By default the diagram frame is displayed on the diagram pane when a new diagram is created. The frame is a rectangle in all diagrams, but state machine and activity. State machine and activity diagram frames have rounded corners.

The frame can be resized manually by dragging its corners or borders.

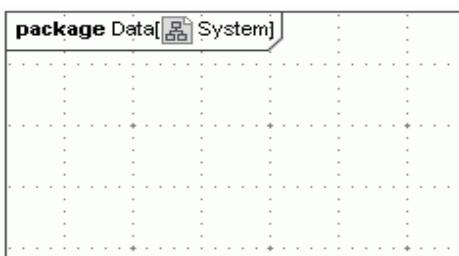


Figure 88 -- An example of the diagram frame

To show/ hide the diagram frame, do one of the following

- From the diagram pane shortcut menu, select/ click to clear the **Show Diagram Frame** check box.
- In the **Diagram Properties** dialog, select/ click to clear the **Show Diagram Frame** check box.

To change the diagram frame properties

Do either:

1. On the **Options** menu, select **Project**. The **Project Options** dialog will open.
2. In the tab tree, expand **Symbols properties styles** and then click **Diagram**. The Diagram options pane will be displayed on the right of the **Project Options** dialog.
3. Change the properties in the **Diagram Frame** property group.

Or:

1. Open the **Diagram Properties** dialog (the ways for opening the dialog are described in Section "[Diagram Properties](#)" on page 171).
2. Change the properties in the **Diagram Frame** property group.

| Property name | Description |
|-------------------------------|--|
| Show Diagram Frame | Displays the diagram frame on the diagram pane. |
| Show Abbreviated Types | Shows full/abbreviated diagram keyword type on the diagram frame header. |
| Show Diagram Name | Shows the diagram name and icon in the diagram frame header. |
| Show Parameters | Diagram context element parameters are displayed in the diagram frame header. |
| Show Context Name | Diagram context element name is displayed in the diagram frame header. |
| Show Context Type | Diagram context element type is displayed in the diagram frame header. |
| Show Diagram Type | Shows the diagram type in the diagram frame header. |
| Show Context Kind | Shows context kind, which is a keyword predefined in UML (e.g. package, class, activity) in the diagram frame header. |
| Autosize | Adjusts the size of the diagram frame to the contained information so that it uses minimum space. Autosizing is automatically switched off when the shape is being resized by the user. |

To hide the icon on the diagram frame

1. Open the **Diagram Properties** dialog (the ways for opening the dialog are described in Section "[Diagram Properties](#)" on page 171).
2. Change the **Show Stereotypes** property value to **Text**.

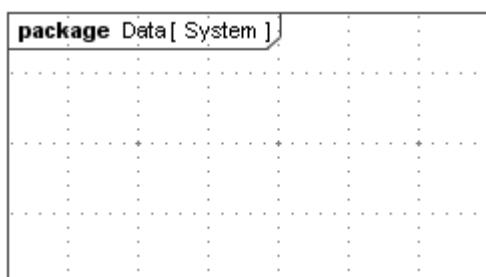


Figure 89 -- Diagram Frame with hidden diagram icon

When a new diagram is created, the diagram stereotype icon is displayed on the diagram frame header by default, though you can change it to the context stereotype icon.

To display the context stereotype icon instead of the diagram stereotype icon on the diagram frame

1. Open the **Diagram Properties** dialog (the ways for opening the dialog are described in Section ["Diagram Properties"](#) on page 171).
2. In the **Diagram Frame** property group, change the **Use Stereotype** property value to **Context**. The context stereotype icon will be displayed instead of the diagram stereotype icon in the diagram frame header.

NOTES

- The **Use Stereotype** property takes effect only in case the **Show Stereotypes** property value is **Text and Icon** or **Icon**.
- For more information about the context of a diagram, see the procedure ["To change the diagram context"](#) on page 170.

There is a possibility to show the abbreviation of a diagram type instead of the full diagram type in the diagram frame header.

To display the abbreviated diagram type

1. Open the **Diagram Properties** dialog (the ways for opening the dialog are described in Section ["Diagram Properties"](#) on page 171).
2. In the **Diagram Frame** property group, select the **Show Abbreviated Type** check box.

The abbreviated diagram types are listed in the following table.

| Diagram name | Abbreviation |
|---------------------|--------------|
| Use Case | uc |
| Communication | comm |
| Sequence | sd |
| State Machine | stm |
| Activity | sct |
| Implementation | impl |
| Composite Structure | cs |

Drawing Diagram Shapes

You can draw diagram shapes in any type of diagram.

For more information about working with shapes you can find in section ["Working with Shapes of Model Elements"](#) on page 195.

To display a diagram shape on a diagram pane

1. Select the diagram in the Model Browser.

5 DIAGRAMMING

Working with Diagrams

2. Drag the diagram to the diagram pane. The diagram shape will be drawn on the diagram pane (see the example in the following picture).

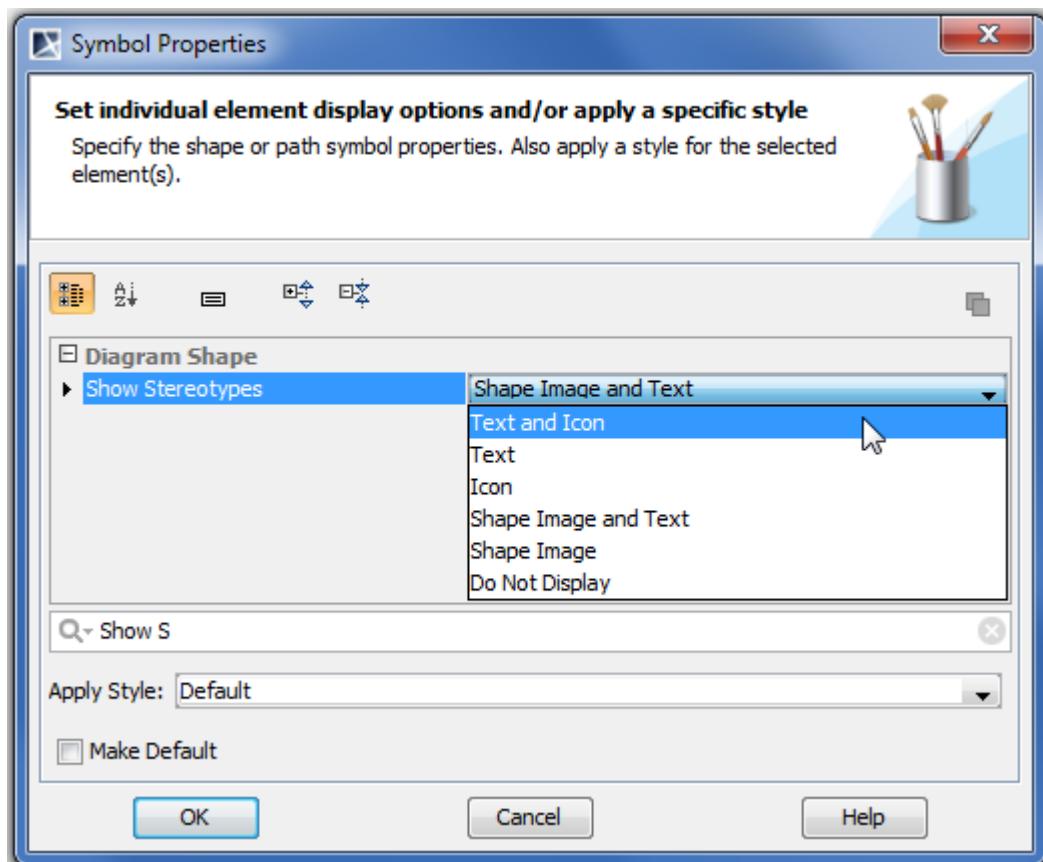


You can change the representation of the diagram shape by changing the diagram stereotype display mode.

To change the representation of the diagram shape

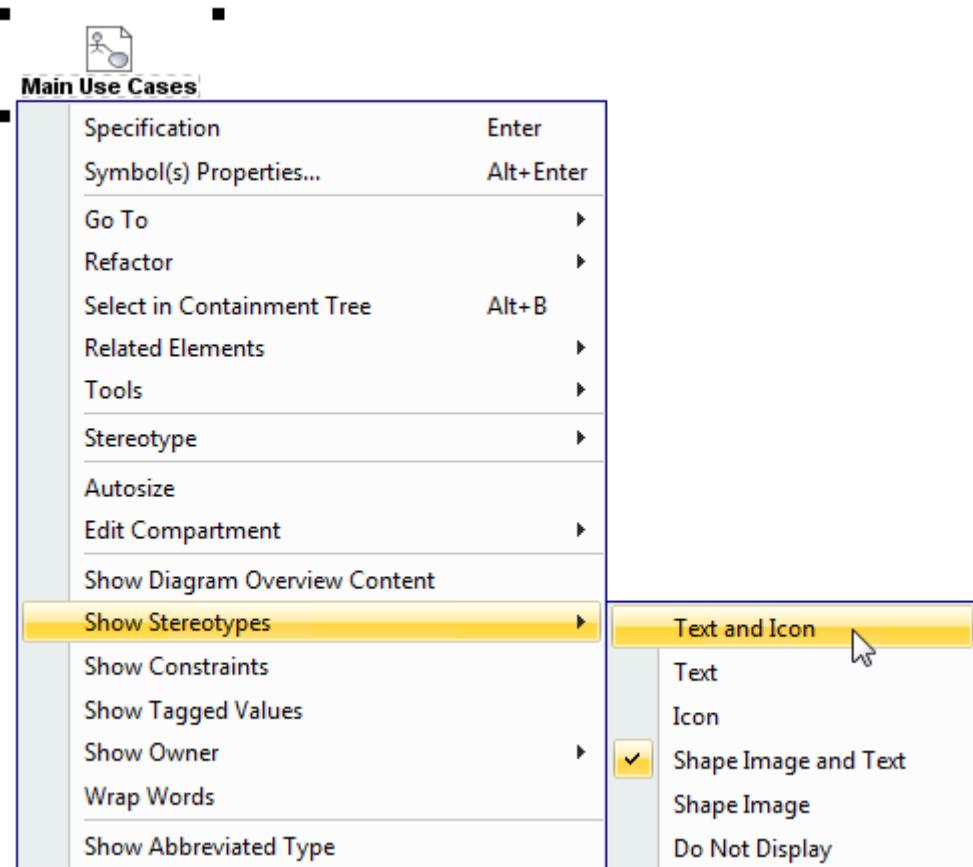
Do either:

1. Right-click the shape and from the shortcut menu select **Symbol(s) Properties**.
2. In the **Symbol Properties** dialog select a new value for the **Show Stereotypes** property (see the following picture).



Or:

1. Right-click the shape and from the shortcut menu select **Show Stereotypes**.
2. From the opened submenu select a desired diagram stereotype display mode (see the following picture).



For more information about the shape stereotype display modes, see Section ["Changing the stereotype display mode"](#) on page 628.

You can also select showing or hiding diagram stereotype constraints or tagged values on the diagram shape.

To show diagram stereotype constraints and tagged values on the diagram shape

Do either:

1. Right-click the diagram shape and from the shortcut menu select **Symbol(s) Properties**.
2. In the **Symbol Properties** dialog select the **Show Constraints** and **Show Tagged Values** check boxes.

Or:

1. Right-click the diagram shape and from the shortcut menu select **Show Constraints**.
2. Right-click the diagram shape and from the shortcut menu select **Show Tagged Values**.

To display the context stereotype icon instead of the diagram stereotype icon on the diagram shape

1. Right-click the diagram shape and from the shortcut menu select **Symbol(s) Properties**.

2. In the **Symbol Properties** dialog, change the **Use Stereotype** property value to **Context**. The context stereotype icon will be displayed instead of the diagram stereotype icon on the diagram shape.

NOTES

- The **Use Stereotype** property takes effect only in case the **Show Stereotypes** property value is **Text and Icon** or **Icon**.
- For more information about the context of a diagram, see the procedure "[To change the diagram context](#)" on page 170.

There is a possibility to show the abbreviation of a diagram type instead of the full diagram type on the diagram shape.

To display the abbreviated diagram type

Do either:

1. On the diagram pane select the diagram shape.
2. In the shortcut menu, select the **Show Abbreviated Type** check box.

Or:

- From the diagram shape shortcut menu, open the **Symbol Properties** dialog box and select the **Show Abbreviated Type** check box.

The abbreviated diagram types are listed at the end of Section "[Diagram Frame](#)" on page 173.

NEW! Overviewing Other Diagrams

NOTE

This feature is available in Standard, Professional, Architect, and Enterprise editions.

TIP!

You can also learn about overviewing diagrams while analyzing the Diagram overview sample.

To open the sample, do any of the following:

- On the Welcome screen, select **Samples** and then in the **Product Features** section click **Diagram overview**.
- Go to the folder <MagicDraw installation directory>\samples\product features and open the *diagram overview.mdzip* file.

As of version 17.0 you can overview other diagrams, including dependency matrices, tables, and relation maps, on a diagram pane. For this the diagram overview shape can be used.

Read and learn how to use the diagram overview shape in the following sections:

- "[Creating a diagram overview shape](#)" on page 178.
- "[Modifying the content of a diagram overview shape](#)" on page 183.
- "[Diagram overview shape environment](#)" on page 186.
- "[How to...](#)" on page 192.

Creating a diagram overview shape

You can use one of the following ways to create a diagram overview shape:

- **Drag the diagram shape from the Model Browser and change it into the diagram overview shape afterwards.** This way created diagram overview shape *shows the full content* of the corresponding diagram (see Figure 90 on page 183) and is updated automatically

according to *all changes* made in this diagram. For detailed description of the procedure see section "[To create a diagram overview shape by using a drag-and-drop operation](#)" on page 179.

- **Use the  Diagram Overview button that is located in the diagram pallet.** This way created diagram overview shape shows *the full content* of the corresponding diagram (see Figure 90 on page 183) and is updated automatically according to *all changes* made in this diagram. For detailed description of the procedure see section "[To create a diagram overview shape by using the diagram pallet](#)" on page 181.
- **Paste a copied part of the diagram's content using a special command from the main menu.** This way created diagram overview shape shows *any copied part* of the corresponding diagram's content (see Figure 91 on page 183) and is updated automatically according only to *the changes made in this copied part*. Note that this way can not be used for dependency matrixes, tables, and relation maps, as it is not allowed to copy and paste different parts of them. For detailed description of the procedure see section "[To create a diagram overview shape by pasting any part of diagram's content](#)" on page 182.

To create a diagram overview shape by using a drag-and-drop operation

1. Open a diagram wherein you want to create the diagram overview shape.
2. In the Model Browser select a diagram you want to overview.
3. Drag the diagram shape to the opened diagram pane. The diagram shape will be drawn on the diagram pane (see the example in the picture below).



Instructor Use Cases

NOTE

If you can not drag the diagram shape to the opened diagram pane within a teamwork project, make sure you have the right to edit model of this project and then try to lock for edit this diagram.

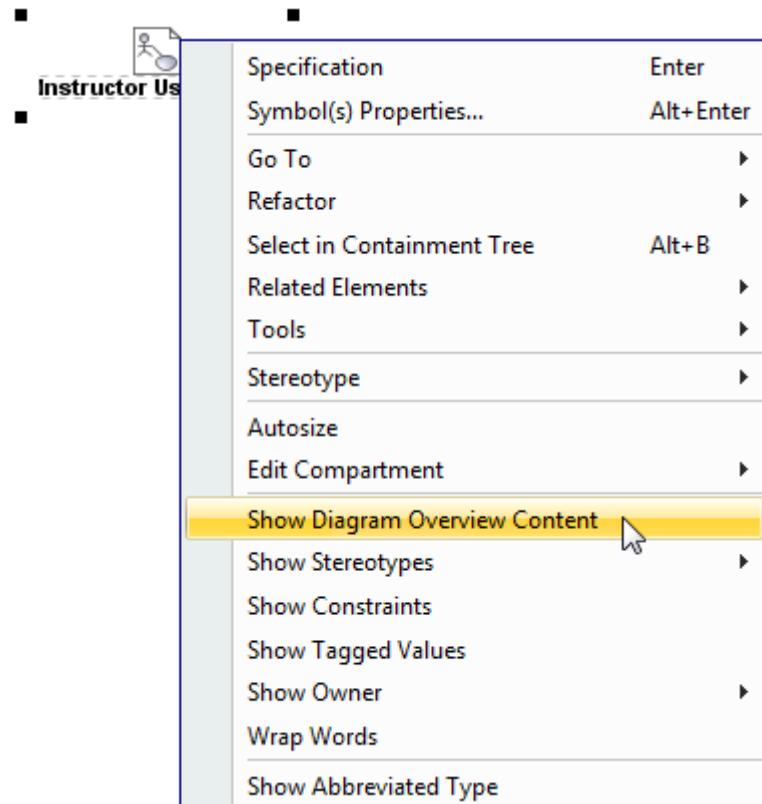
For more information about locking elements please refer to section "Locking Model Elements for Editing" in "[MagicDraw Teamwork UserGuide.pdf](#)".

4. Make the diagram shape show the full content of the corresponding diagram using any of the following GUI features:

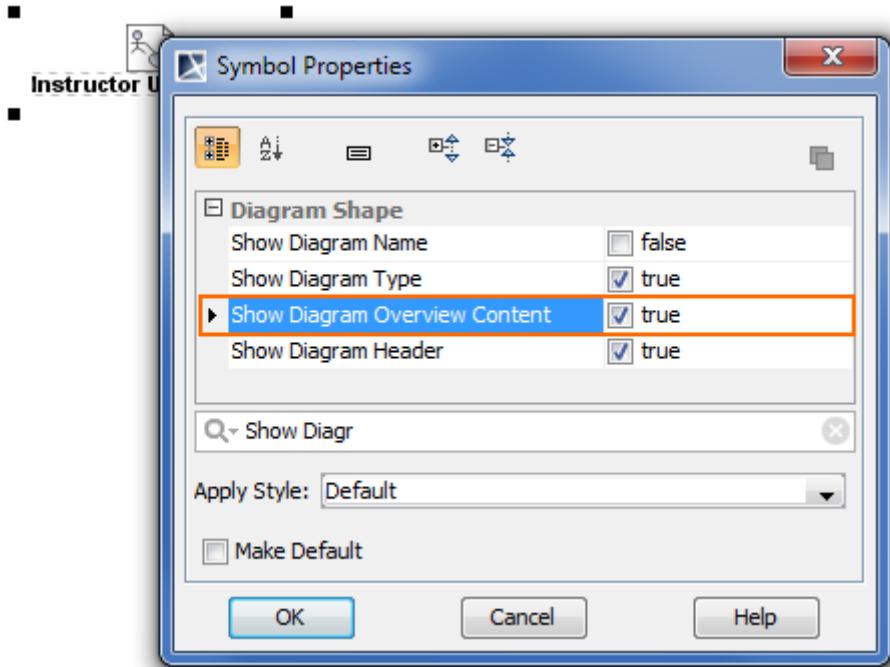
5 DIAGRAMMING

Working with Diagrams

- Right-click the diagram shape and from the shortcut menu select **Show Diagram Overview Content**.



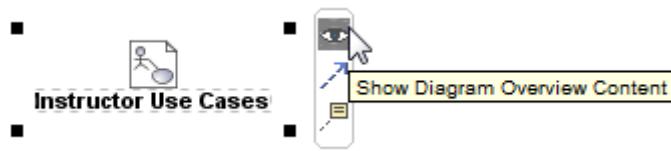
- Right-click the diagram shape and from the shortcut menu select **Symbol(s) Properties**. In the opened dialog, set the **Show Diagram Overview Content** property value to *true*.



5 DIAGRAMMING

Working with Diagrams

- Click the diagram shape and on the Smart Manipulator toolbar select the button.



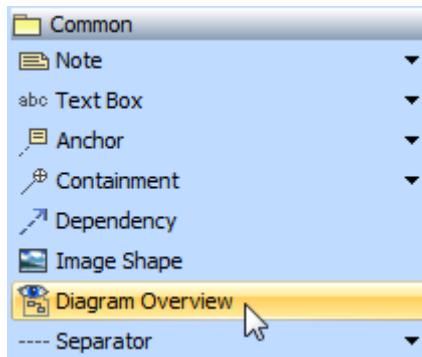
The diagram shape will be changed to the diagram overview shape that shows the full content of the corresponding diagram (see Figure 90 on page 183).

NOTE The content of this way created diagram overview shape will be updated automatically according to *all changes* made in the corresponding diagram.

TIP! Double-click the diagram overview shape to open the corresponding diagram.

To create a diagram overview shape by using the diagram pallet

- Open a diagram wherein you want to create the diagram overview shape.
- On the diagram pallet click the **Diagram Overview** button that is located within the **Common** buttons group.



NOTE If the diagram pallet is inactive within a teamwork project, make sure you have the right to edit model of this project and then try to lock for edit this diagram.

For more information about locking elements please refer to section "Locking Model Elements for Editing" in "[MagicDraw Team-work UserGuide.pdf](#)".

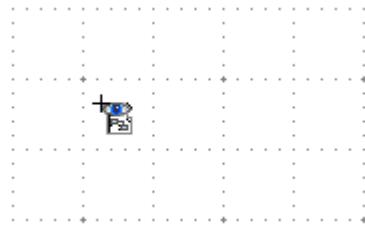
The **Select Diagram** dialog will open (to learn more about using the dialog please refer to section "[Selecting an Element](#)" on page 278).

- Select a diagram for which you want create a diagram overview shape and click **OK**.

5 DIAGRAMMING

Working with Diagrams

4. Click a free space of the diagram pane.



The created diagram overview shape will show the full content of the corresponding diagram (see Figure 90 on page 183).

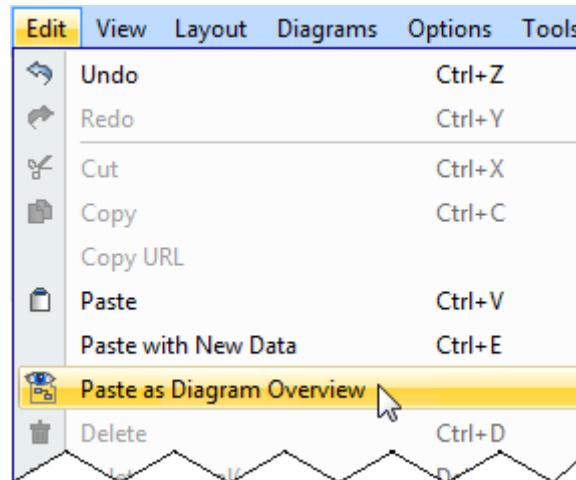
NOTE The content of this way created diagram overview shape will be updated automatically according to *all changes* made in the corresponding diagram.

TIP! Double-click the diagram overview shape to open the corresponding diagram.

To create a diagram overview shape by pasting any part of diagram's content

NOTE This way can not be used for dependency matrixes, tables, and relation maps, as it is not allowed to copy and paste different parts of them.

1. Open a diagram for that you want to create the diagram overview shape.
2. Select the element shapes that you want to appear in the diagram overview shape and then copy them.
3. Open the diagram wherein you want to create the diagram overview shape.
4. On the **Edit** menu, click **Paste as Diagram Overview**.



NOTE If the command is inactive within a teamwork project, make sure you have the right to edit model of this project and then try to lock for edit the diagram wherein you want to create the diagram overview shape.

For more information about locking elements please refer to section “Locking Model Elements for Editing” in [“MagicDraw Teamwork UserGuide.pdf”](#).

The created diagram overview shape will show only the copied part of the corresponding diagram's content (see Figure 91 on page 183)

NOTE The content of this way created diagram overview shape will be updated automatically according only to the *changes made in the copied part* of the corresponding diagram.

5 DIAGRAMMING

Working with Diagrams

TIP!

Double-click the diagram overview shape to open the corresponding diagram.

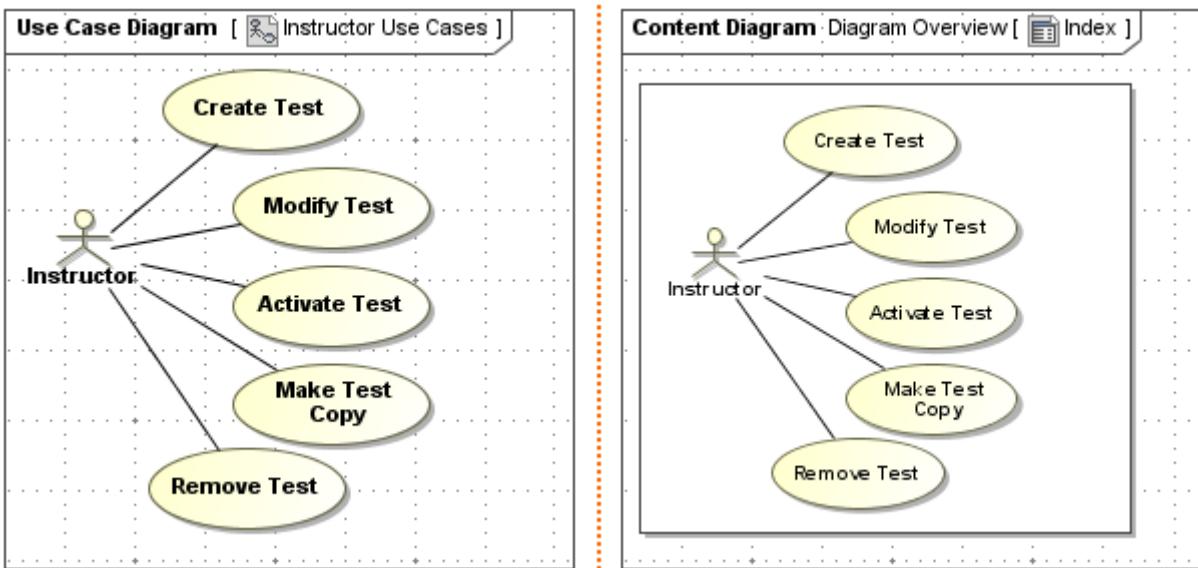


Figure 90 -- An example of diagram and its diagram overview shape showing the full content of the diagram

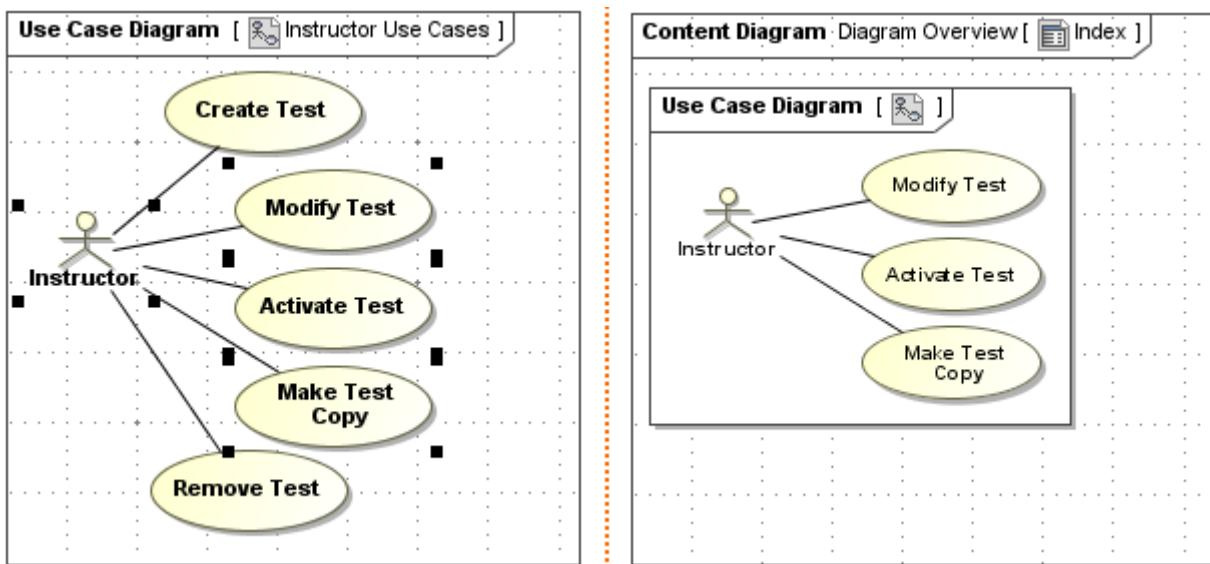


Figure 91 -- An example of diagram and its diagram overview shape showing only the part of the diagram content

Modifying the content of a diagram overview shape

Once the diagram overview shape is created, you can modify its content: add more element shapes or remove no more needed ones by using the **Compartment Edit** dialog.

To open the **Compartment Edit** dialog for modifying the content of a diagram overview shape

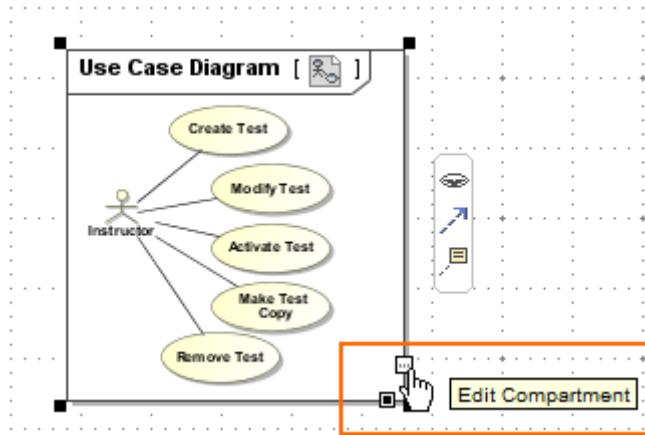
Do any of the following:

- Right-click the diagram overview shape, on the shortcut menu point to **Edit Compartment**, and then select **Diagram Overview Content**.

5 DIAGRAMMING

Working with Diagrams

- Click the diagram overview shape and on the shape area click the “...” button.



The **Compartment Edit** dialog will open with the **Diagram Overview Content** tab selected.

There are two lists in the **Diagram Overview Content** tab:

- | | |
|-----|--|
| All | <ul style="list-style-type: none">If the Selected list is <i>not empty</i>, lists element shapes that are not shown in the diagram overview shape.If the Selected list is <i>empty</i>, lists all element shapes meaning that all of them are shown in the diagram overview shape. In this case the diagram overview shape will reflect any changes in the corresponding diagram. |
|-----|--|

- | | |
|----------|---|
| Selected | Lists element shapes that are shown in the diagram overview shape. In this case the diagram overview shape will reflect only changes of the selected shapes in the corresponding diagram. |
|----------|---|

5 DIAGRAMMING

Working with Diagrams

If the diagram overview shape is created by using instructions given either in the first or the second procedure in section "[Creating a diagram overview shape](#)" on page 178, then you will see that all element shapes are listed in the **All** list within the **Compartment Edit** dialog.

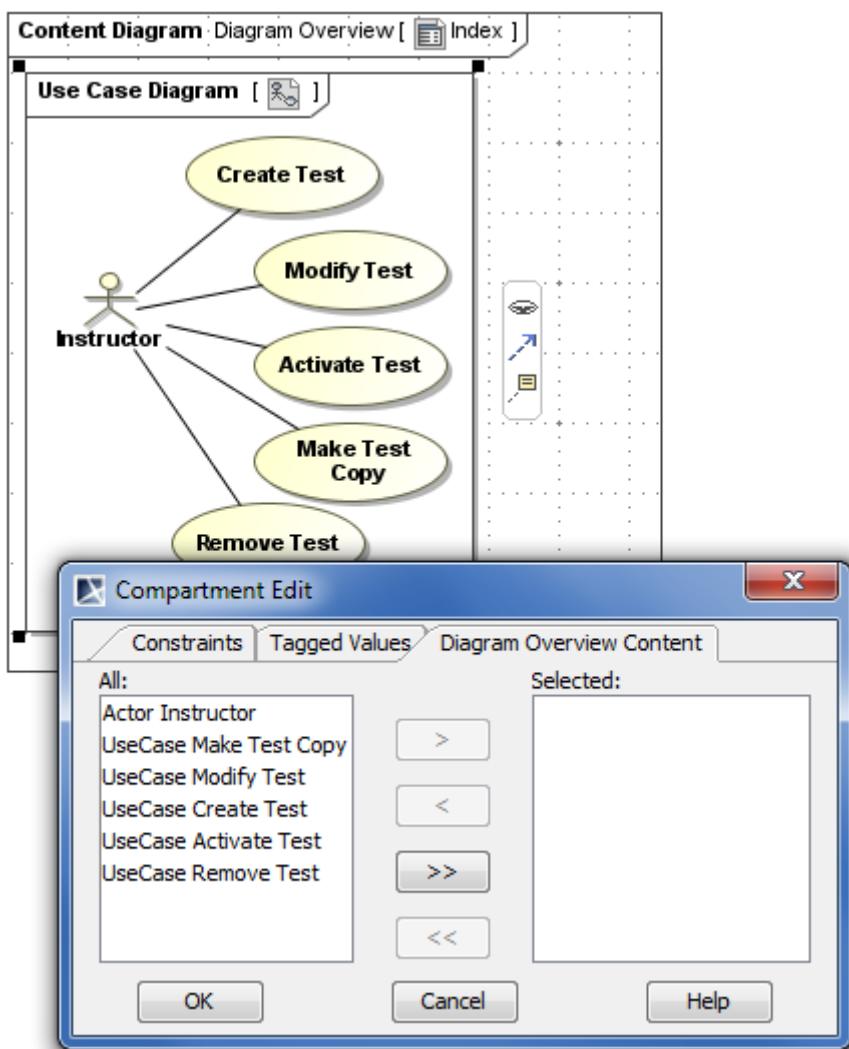


Figure 92 -- An example of the Compartment Edit dialog in case of showing the full content of the diagram in the diagram overview shape

If the diagram overview shape is created by using instructions given in the third procedure in section "[Creating a diagram overview shape](#)" on page 178, then you will see that only the element shapes which are displayed in the diagram overview shape are listed in the **Selected** list (even in case of copying all shapes).

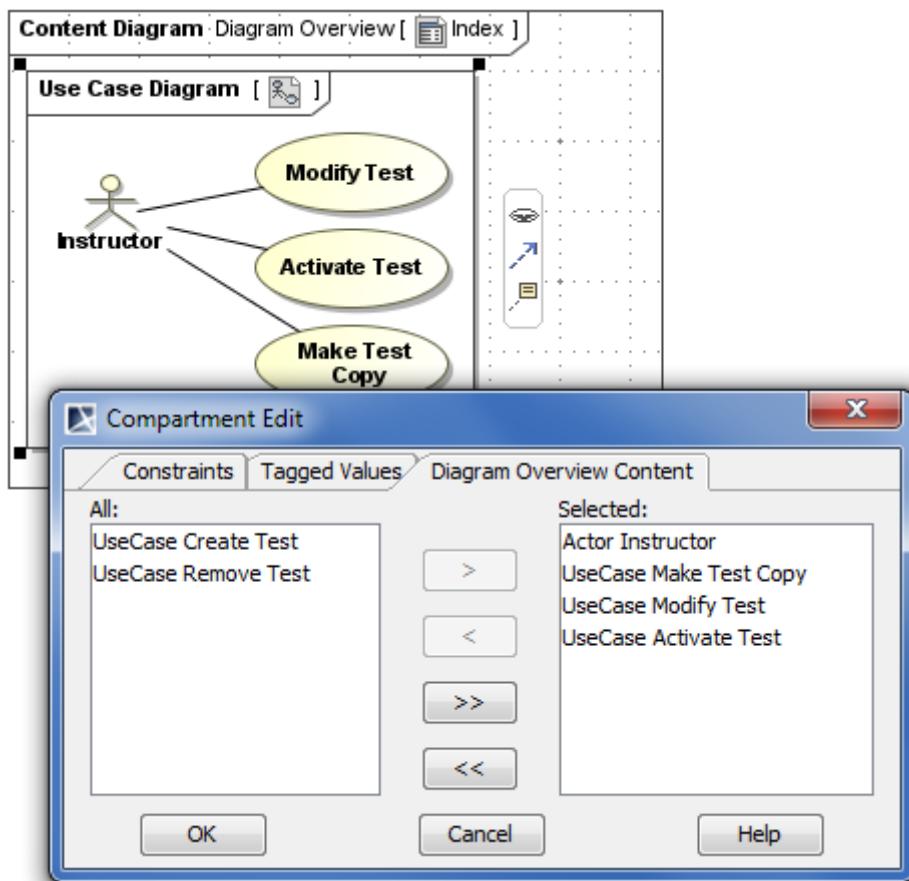


Figure 93 -- An example of the Compartment Edit dialog in case of showing only the part of the diagram content in the diagram overview shape

To modify the content of the diagram over view shape

1. Open the **Compartment Edit** dialog using one of the ways described in the procedure above.
2. Use the **>**, **>>**, **<**, and **<<** buttons to manage item moving between the lists. Remember the rules of showing and not showing shapes included in each list.
3. Click **OK** when you are finished.

Diagram overview shape environment

This section describes GUI features that can be used for handling the diagram overview shape:

- "[Shortcut menu](#)" on page 187.
- "[Symbol Properties dialog](#)" on page 189.
- "[Smart Manipulator toolbar](#)" on page 191.
- "[Shape Area Buttons](#)" on page 192.

Shortcut menu

This section introduces diagram overview shape-specific commands, but not all commands from the diagram overview shape shortcut menu.

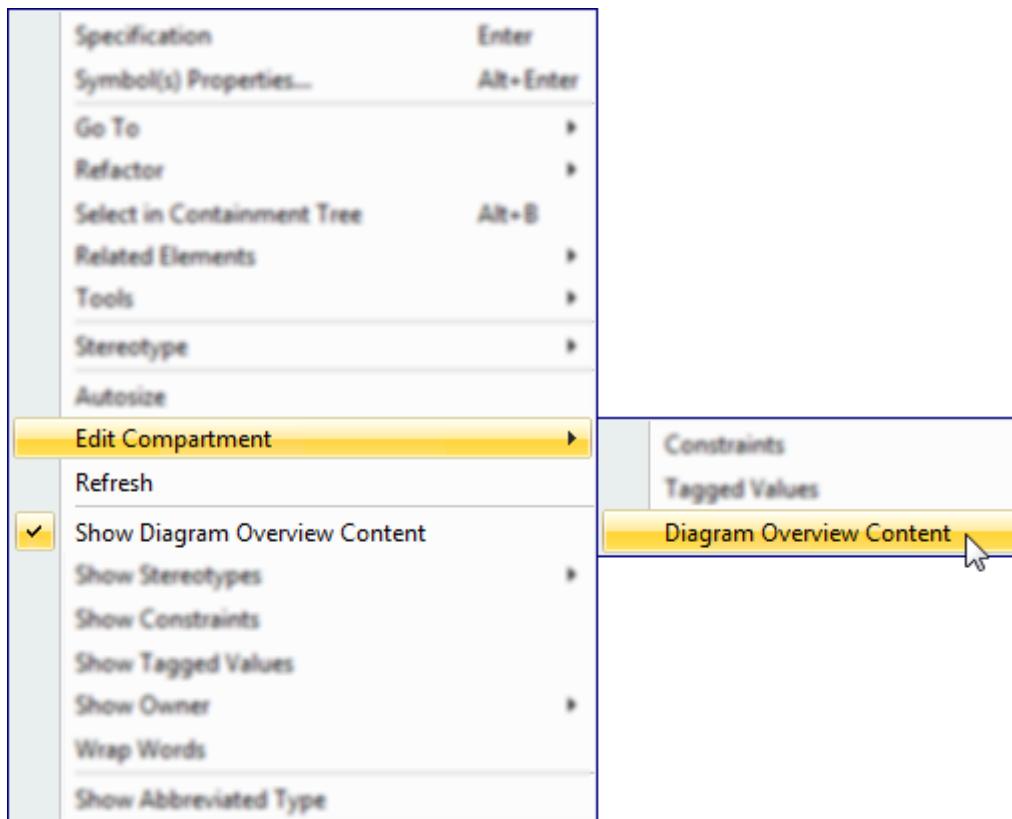


Figure 94 -- Diagram overview shape-specific commands in the shortcut menu

| Command | Description |
|---|--|
| Edit Compartment > Diagram Overview Content | <p>Opens the Compartment Edit dialog with the Diagram Overview Content tab selected.</p> <p>TIP! You can also open this dialog by clicking the “...” button that is on the frame of a diagram overview shape area (to see the button, click the diagram overview shape).</p> |
| Refresh | <p>Updates the content of the diagram overview shape according to changes in the model.</p> <p>NOTE: This button is available in the shortcut menu only if a dependency matrix, table, or relation map is displayed in the diagram overview shape.</p> <p>TIP! You can also update the content of the diagram overview shape by clicking the button that is on the frame of a diagram overview shape area (to see the button click the diagram overview shape).</p> |

| Command | Description |
|--------------------------------------|--|
| Show Diagram Overview Content | <p>Select to change the diagram shape into the diagram overview shape that shows the content or a part of the content of a corresponding diagram.</p> <p>Unselect to change the diagram overview shape into the diagram shape.</p> <p>For more information about using this command refer to step #4 of the procedure "To create a diagram overview shape by using a drag-and-drop operation" on page 179.</p> |

Symbol Properties dialog

This section introduces diagram overview shape-specific properties, but not all properties in the **Symbol Properties** dialog of a diagram overview shape.

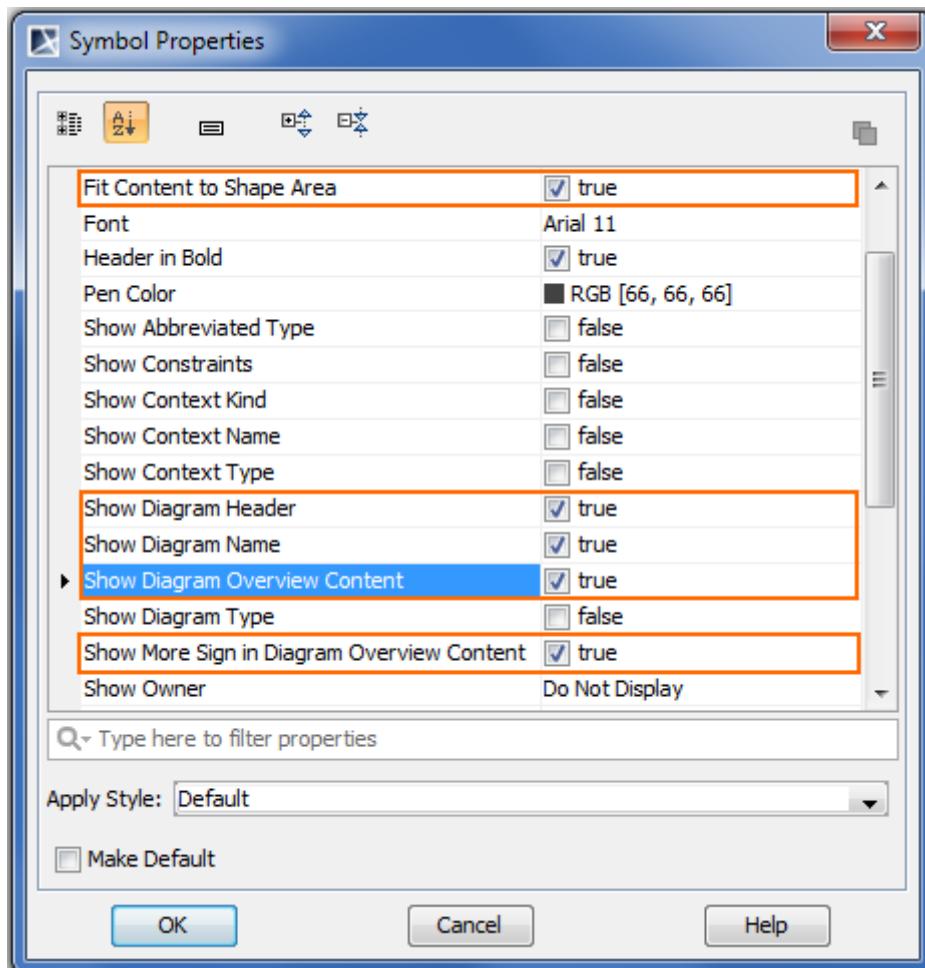


Figure 95 -- Diagram overview shape-specific properties in the Symbol Properties dialog

| Property | Description |
|----------------------------------|---|
| Fit Content to Shape Area | Select to adjust the zoom ratio of the diagram content so that it fits the size of the diagram overview shape area. Unselect to fix the current zoom ratio of the diagram content. It will persist in any size change. NOTE: Any value setting for this property can take effect on the diagram overview shape in case the Autosize property value is <i>false</i> . If you need to reset the diagram content ratio to 1:1, set the Autosize property value to <i>true</i> . |
| Show Diagram Header | Select to show the diagram header in the diagram overview shape. |
| Show Diagram Name | Select to show the diagram icon and name in the diagram header of the diagram overview shape, in case the diagram header is displayed, i.e., the Show Diagram Header property value is <i>true</i> . |

5 DIAGRAMMING

Working with Diagrams

| Property | Description |
|---|---|
| Show Diagram Overview Content | Select to change the diagram shape into the diagram overview shape that shows the content or a part of the content of a corresponding diagram. Unselect to change the diagram overview shape into the diagram shape. For more information about using this command refer to step #4 of the procedure " To create a diagram overview shape by using a drag-and-drop operation " on page 179. |
| Show More Sign in Diagram Overview Content | Select to display the more sign at the bottom of the diagram overview shape area, when only a part of the diagram content is displayed the diagram overview shape (see Figure 96 on page 190). |

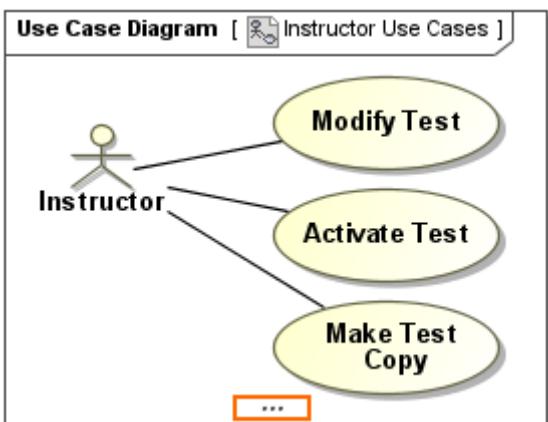


Figure 96 -- An example of the more sign on diagram overview shape area

Smart Manipulator toolbar

In the Smart Manipulator of the diagram (overview) shape there are two buttons that need to be described.

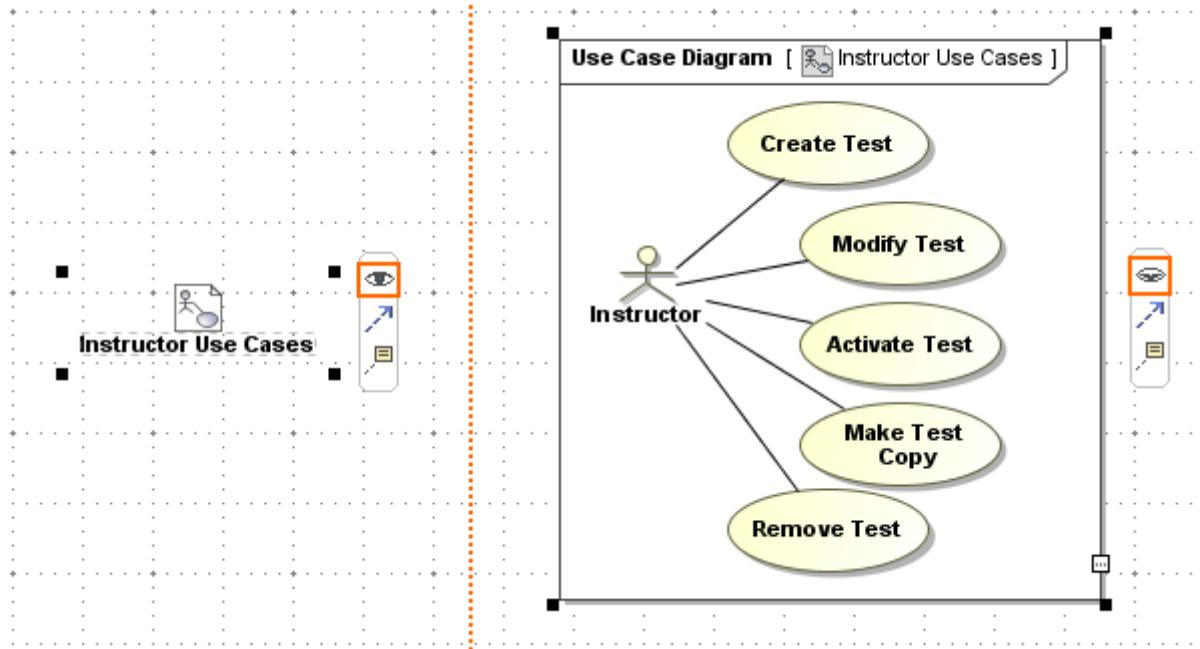


Figure 97 -- Buttons for showing (on the left) and hiding (on the right) the diagram overview content in the Smart Manipulator toolbar

| Button | Description |
|--------|---|
| | Changes the diagram shape to the diagram overview shape that shows in its content the selected-to-overview shapes of the corresponding diagram. For more information about using this command refer to step #4 of the procedure " To create a diagram overview shape by using a drag-and-drop operation " on page 179. |
| | Changes the diagram overview shape to the diagram shape. |

Shape Area Buttons

There are different sets of buttons on the diagram overview shape area showing a diagram and the one showing a dependency matrix, table, or relation map.

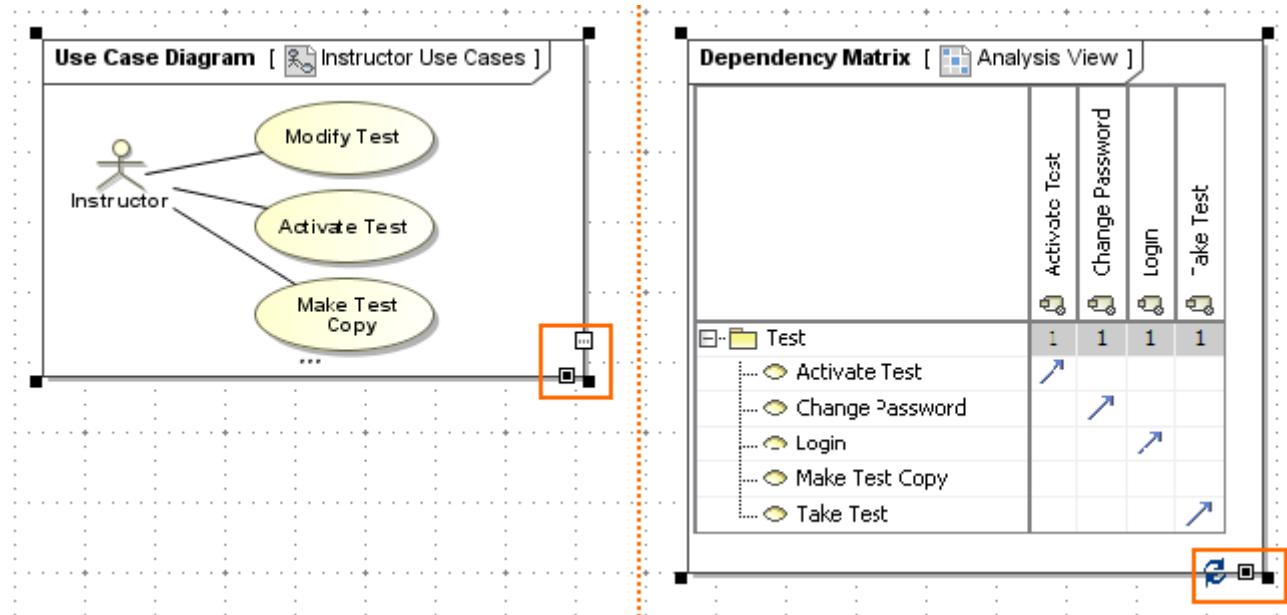


Figure 98 -- Diagram overview shape area buttons in case of overviewing diagram (on the left) and dependency matrix (on the right)

| Button | Description |
|--------|--|
| | Opens the Compartment Edit dialog with the Diagram Overview Content tab selected. |
| | Updates the content of the diagram overview shape according to changes in the model. NOTE: This button is available on the diagram overview shape area only if a dependency matrix, table, or relation map is displayed in the diagram overview shape. |
| | Resets the diagram overview shape's content ratio to 1:1. |

How to...

This section gives some remarks that can be useful when struggling to arrange sizes of your diagram overview shapes. The solutions we offer are as follows:

- ["How to make a couple of diagram overview shapes to be of the same size?"](#) on page 193.
- ["How to set desired shape sizes in the a diagram overview shape?"](#) on page 193.

How to make a couple of diagram overview shapes to be of the same size?

Let's say you have two diagram overview shapes. You want their zoom ratio to be 1:1 and shape areas to be of the same height (or width) at the same time.

Solution

1. For both diagram overview shapes set the **Fit Content to the Shape Area** property value to *false*.

2. For both diagram overview shapes set the **Autosize** property value to *false*.
3. Make both diagram overview shapes of preferred size.
4. Make both diagram overview shapes of the same height (or width) by dragging corners of the shape areas.

How to set desired shape sizes in the a diagram overview shape?

Solution

1. Set the **Fit Content to the Shape Area** property value to *true* for this diagram overview shape.
2. Set the desired shape content size by dragging corners of the shape area.
3. Set the **Fit Content to the Shape Area** property value to *false* for this diagram overview shape.

In case the diagram overview shape shows the full content of the corresponding diagram, all new shapes added to the diagram pane in the diagram overview shape will be of the previously set size.

If you want to reset the shape sizes to ratio zoom 1:1, do any of the following:

- Set for the diagram overview shape the **Autosize** property value to *true*.
- Make the diagram overview shape of the preferred size.

Table with Diagram Information

On the diagram, you may display a table containing various diagram details: its name, author, status, the dates it was created and modified, etc. By default, the Diagram info table is displayed at the right top corner of the diagram frame, but you can drag and drop it to any other position on the diagram.

The table includes the following fields:

- Diagram name
- Author
- Creation date
- Modification date
- Last Modified by
- Other available tag definitions

To show the table containing the diagram information

1. From the diagram shortcut menu, select **Show Diagram Info**.
2. The table with the predefined information will be displayed on the diagram.

| Diagram name | Requirements |
|-------------------|------------------|
| Author | andre |
| Creation date | 5/18/07 9:02 AM |
| Modification date | 2/26/10 11:34 AM |
| Last modified by | martin |

To define information that will be included in the table

1. From the diagram info table shortcut menu, select **Customize**, or from the **Options** menu, select **Project**. The **Project Options** dialog box opens.
2. Open the **Diagram Info** pane.

3. In the **Source** pane, select the type of information you want to include in the table: **Standard Mode** or **Custom Mode**.
4. **Standard Mode** contains the following fields that will be shown in the table: Author, Creation date, Modification date, and all other tag definitions that can be assigned to the diagram. In the **Custom Mode** field, you may create your own table or any other object in HTML.
5. Preview the selected table or other created object in the **Preview** pane.

Changing the Diagram Type

The diagram type may be changed to another type of diagram if both diagram types are compatible. Note: diagram elements are not converted.

Changing the diagram type is usable:

- To migrate with existing project to the a diagram type which was not available till then. For example, to migrate Class diagram to the SysML Block Definition diagram.
- To migrate with existing project from the diagram type, which will be dropped from your project. For example, if user has decided to drop plug-in, and needs to convert plugin specific diagram to standard MagicDraw diagrams.

Diagram conversion scenarios:

- Any static diagram may be converted to another type of static diagram.
- Any dynamic diagram may be converted to another diagram, if both diagrams are based on the same diagram type and diagrams are compatible.

To change the diagram type:

1. Select one or more the same type diagrams in the Browser (Containment or Diagrams tree).
2. From the shortcut menu, choose command **Change Type To** and select desired diagram type from the list (see Figure 99 on page 195).

5 DIAGRAMMING

Working with Shapes of Model Elements

You can also convert multiple the same type diagrams – select them in the Browser and choose command **Change Type To**.

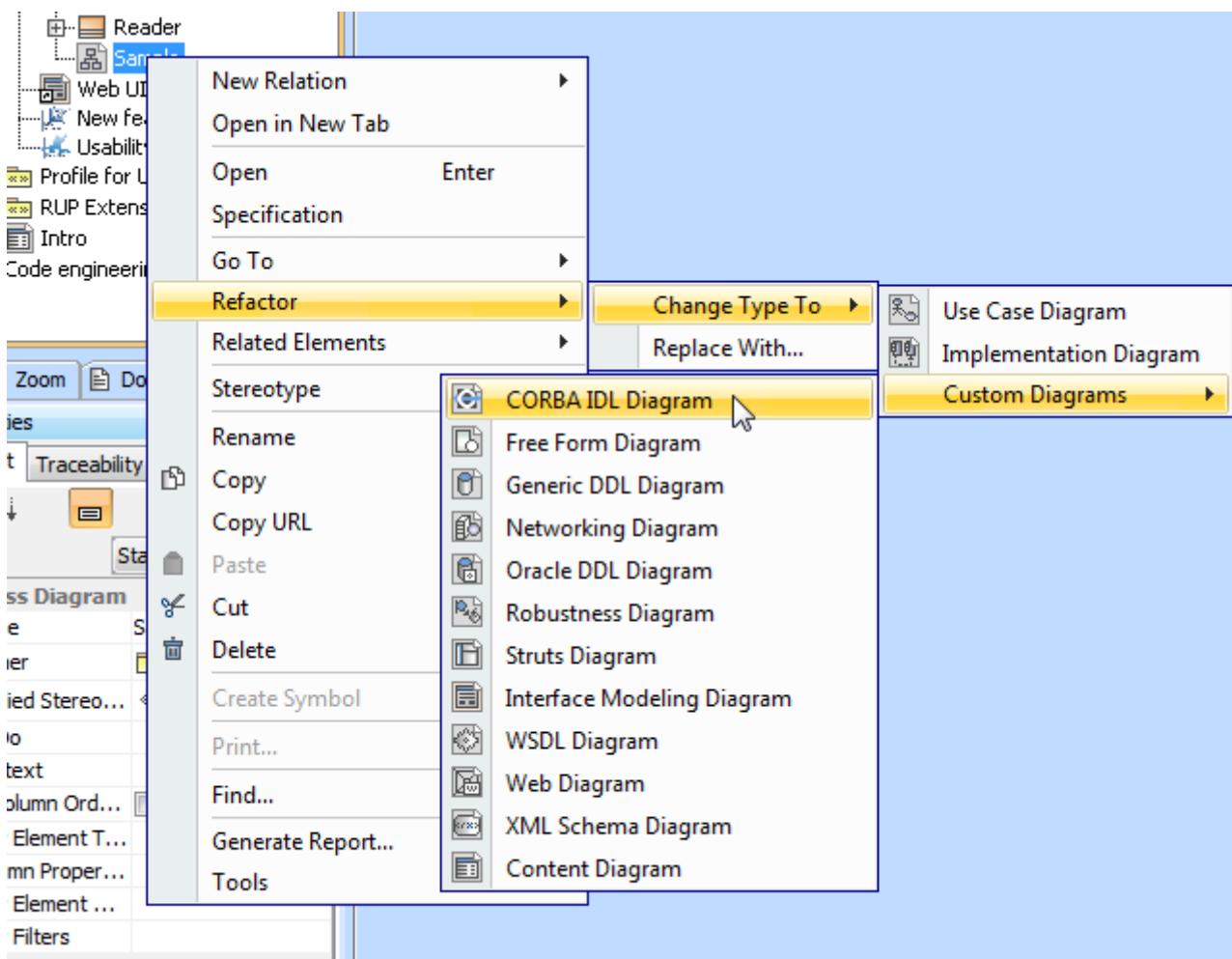


Figure 99 -- Changing the diagram type

Working with Shapes of Model Elements

[View Online Demo](#)

MagicDraw Basics

To draw a shape on the Diagram pane

1. Click the shape button on the diagram toolbar, or press the appropriate shortcut key for the shape you wish to draw (the button remains pressed). For a detailed description of diagram toolbars, see “You may customize menu items by selecting and/or modifying perspectives. More information about customizing perspectives, you may find in Section “Customizing and Selecting Perspective” on page 63.” on page 70.
2. Click the desired location on the diagram pane. The new shape is placed on the diagram pane at the point you click.
OR
 - Click the shape button on the diagram toolbar, hold down the left mouse button, and drag shape from the toolbar to the diagram. The new shape is placed on the diagram pane at the point you will release the mouse.

- Create the desired model element in the Browser tree. From the created item shortcut menu, select **Create Symbol** or drag and drop the selected model element to the diagram pane.

To draw a number of shapes on the diagram pane

1. Click the shape button on the diagram toolbar, or press the appropriate shortcut key for the shape you wish to draw (the button remains pressed.)
2. Click the **Sticky** button on the diagram toolbar (shortcut key is Z.)
3. Click the desired location on the diagram pane. The new shape is placed on the diagram at the point you click (the button remains pressed.)
4. Click the next location on the diagram pane. The next shape is placed on the diagram pane. Repeat this until you draw the desired number of shapes.
5. To undo the shapes, click the **Sticky** button on the diagram toolbar (shortcut key is Z).

To draw a shape for the selected item in the Browser tree

1. Activate a diagram on which you wish to draw a shape.
2. From the Browser tree, select an item you wish to draw.
3. From the item shortcut menu, select **Create Symbol** or drag and drop the selected model element onto the diagram pane.

To specify the name of the shape (when it is allowed)

1. Double-click the shape or select **Specification** from the shape shortcut menu. The corresponding **Specification** dialog box opens.
2. Type the shape name in the **Name** text box and click **Close**.
 - Type the shape name directly on the selected shape on the Diagram pane.
 - Type the shape name after slowly double-clicking the shape in the Browser tree.

To create several shapes with the same data

- From the shortcut menu of the desired item in the Browser tree, select **Create Symbol**. Or, drag and drop the selected model element onto the diagram pane.
- Type the same name for multiple shapes directly on the shape after the text cursor appears in that area.
 1. Specify a shape name.
 2. Draw another shape of the same kind on the Diagram pane.
 3. Click the shape in the name area. The list of existing shape names appears.
 4. Select a name for the shape from the list.

NOTES

- These shapes will contain identical data.
- Auto completion for entering names is available for all elements.

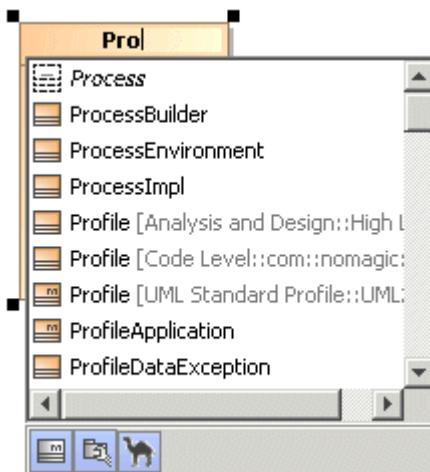
IMPORTANT!

If you attempt to enter an existing name in the corresponding **Specification** dialog box, an error message alerts you to the existence of the current name of the shape. You may not specify a name for a new shape if another shape of the same name and kind is already present in the package.

To use autocompletion when typing the element name

- On the diagram pane, click on the element and then click on the element name area. The autocompletion list with already created elements appears.

- Type the first letter(s) of your searched element and the list is reduced according to the letters you typed. For, example, if you are searching for *Profile* class, type the *Pro* letters and all classes, which begins with *Pro* will be shown in the list.,



You can also press the **Ctrl+Space** or **Ctrl+Backspace** to invoke the autocompletion list.

Filters in the autocompletion dialog allow the filtering of rarely used items as “metaclasses” and “elements from profiles”. This allows comfortable and clear usage of the autocompletion dialog for modeling without active usage of elements from profiles and it increases modeling speed.

At the bottom of the drop-down list box, you will find buttons to perform filtering:

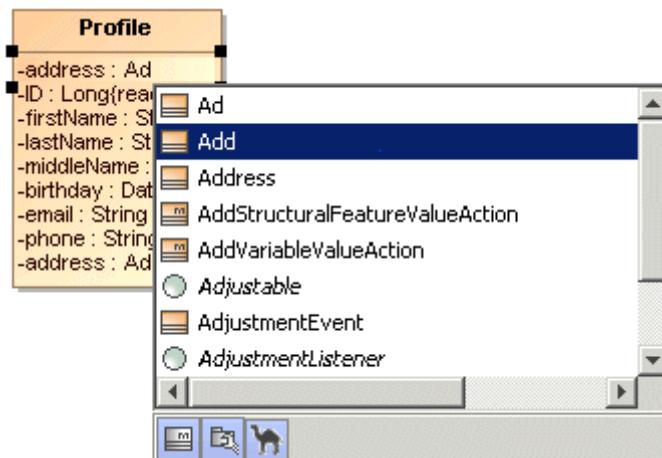
- The Auto completion includes metaclasses button.** When pressed, the list of available elements, element types, or stereotypes includes metaclasses (in MagicDraw metaclasses are placed in the *UML Standard Profile*) appears.
- The Auto completion includes elements from profiles and modules button.** When pressed, the list of available elements, element types or stereotypes includes elements, which are placed in modules appears. (**Note!** This option toggles all profiles except the *UML Standard Profile*.)
- The Auto completion uses camel case button.** When pressed, you may search for elements via the capital letter patterns. For example, instead of typing *ArrayIndexOutOfBoundsException* you may type *A/OOBE*.

To assign an existing type or to create a new type to an element

You may quickly assign a type for the attribute, operation, parameter, instance, and lifeline using the auto-completion list:

- On the diagram pane in the element name area type ":" and now you may assign the element its type:
- Type the name of a non-existent element. A new class and the assigned type are created in the project.
 - Type the name of an already existing classifier and it will be assigned as the type.
 - If in the project two classifiers exist with the same title after the classifier name is typed, the Select Classifier dialog box opens. Select the element you want to assign as type.

2. On the diagram pane in the element name area type ":" and press **Ctrl+Space** or **Ctrl+Backspace**. The list of possible elements to assign opens. To find the element in the list by name, type its name.



To delete the selected model element or symbol

- From the **Edit** menu, select **Delete** (both data and symbol are deleted.)
- On the main toolbar, click **Delete** (both data and symbol are deleted.)
- Press **CTRL+D** keys (both data and symbol are deleted.)
- Press **Delete** key (only the symbol is deleted, leaving the data intact.)

NOTE

When deleting paths by pressing the **DELETE** key, the **Delete Data?** message appears. To delete the relationship data from the model, click **Yes**.

When you use other methods to delete relationships, the relationship data is automatically deleted.

Working with Paths and Relationships

[View Online Demo](#)

MagicDraw Basics

To create a path between shapes

- Click the appropriate path button on the diagram toolbar for the relationship you wish to draw. For a detailed description of the diagram toolbars, see "You may customize menu items by selecting and/or modifying perspectives. More information about customizing perspectives, you may find in Section "Customizing and Selecting Perspective" on page 63." on page 70.
- Click the first (source) shape of the path.
- Drag the path to the second (target) shape of the path and drop it there.

To remove the selected path between shapes

- To remove the selected path from the diagram, press Delete. After the deletion, the relationship will no longer be referenced by the deleted path (symbol), but still can be referenced by other appropriated symbols.

- To remove the selected relationship from the model, press Ctrl + D.

To create a number of paths

1. Click the appropriate path button on the diagram toolbar.
2. Click the **Sticky** button on the Diagram toolbar (shortcut key is Z.)
3. Click the first (source) shape of the path.
4. Drag the path to the second (target) shape of the path and drop it there.
5. Click the first (source) shape of the path to draw the next path.
6. Drag the path to the target shape and drop it there. The new path is created between the two shapes. Repeat this until you create the desired number of paths of that type.
7. Click the **Sticky** button on the Diagram toolbar (shortcut key is Z.)

To change a path appearance style

1. Select the path.
2. Right-click the path or select **Path** from the **Edit** menu, and select the commands you need:
 - To set the path as rectilinear, oblique, or bezier, select **Path Style**.
 - To select a path style, select **Change Path Style** (shortcut keys CTRL+L.)
 - To reset path labels to the default position, select **Reset Labels Positions**.
 - To remove all angles of the path, select **Remove Break Points**.

NOTES

- Every diagram has the Manipulation Highlighting feature. When drawing a path between two model elements, you will see that those shapes are bordered with a red or blue rectangle. The red color indicates that the path may not be drawn between these shapes. Blue rectangle allows a path to be drawn.
- Remove the manipulation highlighting in the **Environment Options** dialog box, **Diagram** section, **Edit** group. For more information, see “Customizing Environment Options” on page 89.
- For drawing a path, you can use smart manipulations menu, which appears near the element symbol. Select a path and drag it to the target shape.

To make the path corners rounded

1. Select the path.
2. From the paths shortcut menu, select the **Rounded Corners** check box.
- Right-click the path and select **Symbol(s) Properties** and in the **Properties** dialog box, set the **Rounded Corners** property to *true*

To create line jumps

Line jumps represent an intersection of lines. If you have a large diagram with lots of intersecting paths, line jumps make the diagram easier to understand.

By default line jumps are not displayed. You can configure a diagram to display line jumps in the following way:

1. From the diagram shortcut menu, choose **Diagram Properties**. The **Properties** dialog box appears.
2. Change the **Add Line Jumps To** property.

The **Add Line Jumps To** property has the following options:

5 DIAGRAMMING

Working with Paths and Relationships

- **None.** Line jumps are not displayed. (Default value)

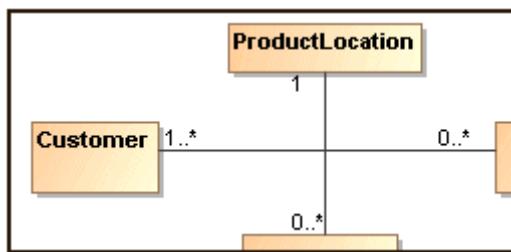


Figure 100 -- Diagram with no line jumps

- **Horizontal Line.** Line jumps are displayed on horizontal lines.

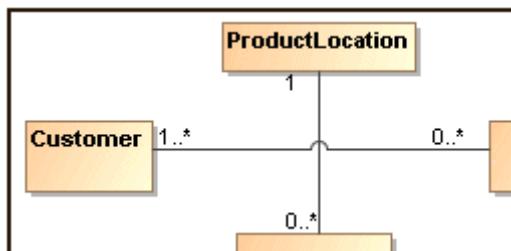


Figure 101 -- Diagram with horizontal line jumps

- **Vertical Line.** Line jumps are displayed on vertical lines.

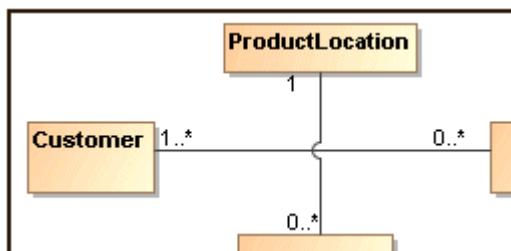


Figure 102 -- Diagram with vertical line jumps

Changing the **Add Line Jumps To** property for a particular diagram from within the diagram **Properties** dialog box, will change line jumps for the current diagram only. For more information about how to change line jumps for the whole project, see “Style Engine” on page 258.

Inserting a Shape on the Path

NOTE: This functionality is available in the State and Activity diagrams.

In the State and Activity diagrams you may split a path into two paths, by drawing a symbol on it. This is valid for Transition / Control Flow / Object Flow relationships and allowed to connect with these path elements.

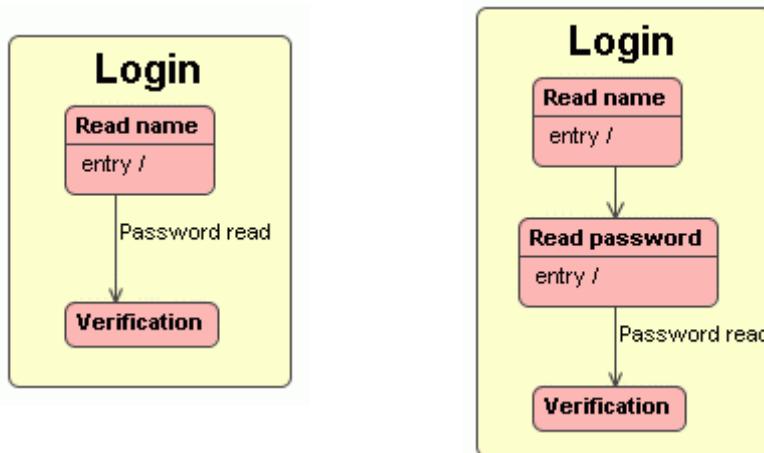
To insert a new shape splitting path on the diagram pane

1. Select the symbol you want to insert or click the diagram toolbar button to create a new one.
2. Drag it on the path. The path is highlighted in blue.
3. Drop the symbol. A Message dialog box appears asking if you want to insert the symbol on the path.

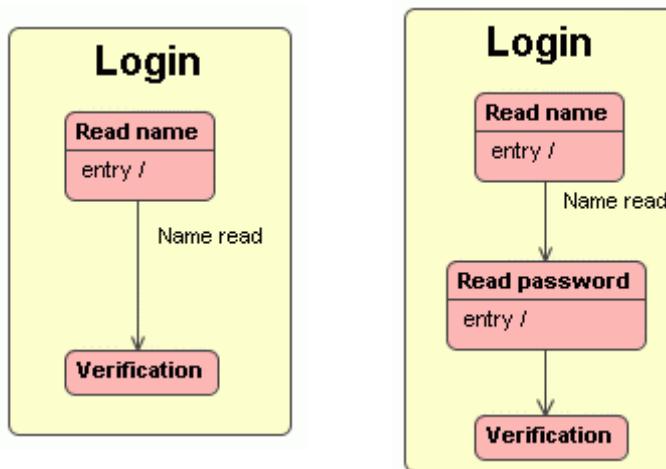
Possible solutions:

- **Before <path type>.** Symbol is inserted before the path. It means a new path is created, then the dropped element symbol is drawn and then the existing path is drawn. For example:

Password read transition is drawn from *Read Name* state to *Verification* state. If you want to insert the *Read password* state before the *Password read* transition, drop the *Read password* state on the transition and in the open dialog, click the **Before Transition** button.



- **After <path type>**. Symbol is inserted before path. It means, the existing path is created, then the dropped element symbol is drawn, and then a new path is drawn. For example: *Name read* transition is drawn from *Read Name* state to *Verification* state. If you want to insert the *Read password* state after the *Name read* transition, drop the *Read password* state on the transition and in the open dialog, click the **After Transition** button.



- **Do not insert**. Action is cancelled and the dialog is closed.

Select the Remember my choice check box and the next time an element will be inserted before or after the path, depending on your selection made this time.

Creating Relations from the Model

The main purpose of this functionality is to allow connecting and create traceability, according to UML, among elements, which are not from the same diagram. In other words, to link elements from a model without the need to place them in the same diagram.

Advantages of this implementation:

- Working time is saved on creating a diagram just to link elements for traceability.

- Some elements cannot be added to the same diagram and linked (elements from Behavior diagrams - Actions, States, Lifelines cannot be added to Static diagrams), this feature will allow the linking of such elements.
- Capability to relate multiple elements to a single element quickly. This is the usual case when a single element is represented with many elements in different abstraction levels or domains.
- The allocation relationship can provide an effective means for navigating the model by establishing cross relationships and ensuring the various parts of the model are properly integrated. For example, activity allocation to a bloc in SysML.
- Another example is creating an abstraction relationship with a stereotype <>trace></> between the model elements or sets of model elements that represent the same concept in different models.
- Smart manipulators allow connecting to any existing element quickly. Also, any Relation used with an element has a smart manipulator included. Draw any relation from any element and you will be able to select the existing target element from the browser.

To create a new relation for an element

1. From an element shortcut menu in the browser, select **New Relation** and then select the desired link from the group of **Outgoing** or **Incoming** relations. The **Create New <relation name> To (From)** dialog box opens.
2. In the model element tree, select an element to (from) which you want to create a relation. Click **OK**. The link will appear in the Browser. Type the name or leave it unnamed.
-or-
1. In the element **Specification** dialog box, select the **Relations** group.
2. Click the **Outgoing** or **Incoming** button and then select the desired link from the list. The **Create New <relation name> To (From)** dialog box opens.
3. In the model element tree, select an element to (from) which you want to create a relation. Click **OK**. The link will appear in the **Relations** group.
-or-
1. On the diagram pane, select an element and then select the desired link from the smart manipulator relations list that opens.
2. Right-click to open the target element list and select the **Select From Model** command. The **Create New <relation name> To** dialog box opens.
3. In the model element tree, select an element to which you want to create a relation. Click **OK**. The link will be drawn on the diagram pane.

Smart Manipulation

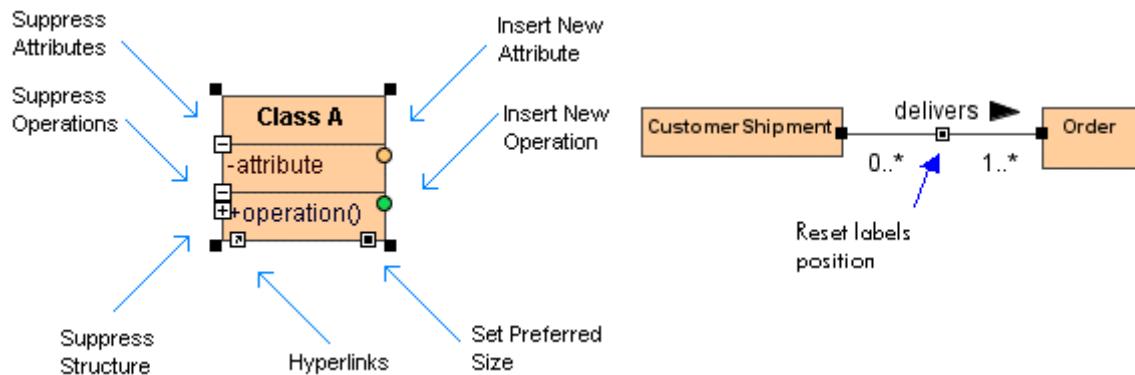
[View Online Demo](#)

Smart Manipulators

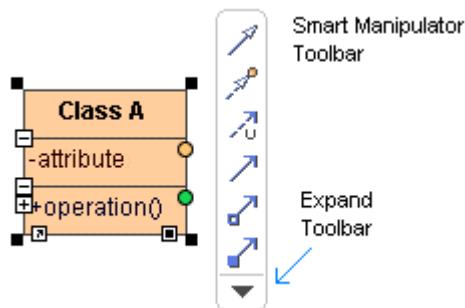
Smart Manipulation is a feature designed to make working with MagicDraw even easier. Use Smart Manipulation to suppress attributes and operations, set an auto-size option, reset a label position on a path, and draw relationships with most commonly used elements. MagicDraw offers varying smart mechanisms depending on the shapes involved.

There are two types of Smart Manipulators:

1. Small buttons are displayed within the symbol on the diagram pane.



2. Smart Manipulator toolbar, which appears when elements are selected on the diagram pane.



In the Smart Manipulator toolbar, smart manipulators are divided into *standard* and *extra* modes. You can toggle between these two modes by clicking the **Expand** button - the arrow symbol - located at the bottom of every Smart Manipulator toolbar. The program remembers your mode choice and displays it for all elements.

Use the Smart Manipulators toolbar to quickly perform simple actions and create new elements.

To create a new element connected to a particular element

1. Select a symbol on the diagram pane. The Smart Manipulator toolbar appears. In the toolbar, select the relationship you want to draw. The drawing of the selected relationship is initiated and the mouse cursor displays the new element which will be created.

For example, create a class symbol. In the class Smart Manipulator toolbar, select the Directed Association relationship for drawing. The drawing of this Directed Association relationship is initiated and the mouse cursor displays a class icon. Click the left mouse button. The element displayed on the mouse cursor is created together with the relationship.

2. Use the Smart Manipulator toolbar to select which element you want to draw at the other end of relationship. In the toolbar, select the relationship and then click the right mouse button. The list of elements available for creating appears. Select the element from the list and it will be created.

TIP! To create a path breakpoint use the following keyboard combination:
Ctrl key + Mouse click.

TIP! To cancel the drawing of an element, press **Esc**.

The Autosize option is automatically added for all shapes.

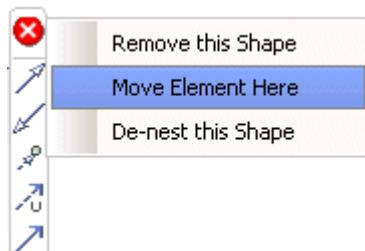
NOTE Autosizing is automatically switched off when the shape is being resized by the user.

To hide smart manipulation

1. From the **Options** menu, select **Environment**. The **Environment Options** dialog box opens.
2. In the **Diagram** pane, **Smart Manipulators** group, clear the **Show Smart Manipulation** check box and click **OK**.

To solve the detected symbol ownership problem

1. Select the element from the diagram pane, which is drawn on an incorrect ownership (which is highlighted in red). The Smart Manipulator toolbar appears.
2. Click the red button, which is at the top of the Smart Manipulator toolbar. The menu with the possible problem solving solutions appears.



For more information see “Resource Manager” on page 366.

Selection and Multiple Selections

To select a shape

- Click the desired shape on the Diagram pane.

To deselect the selected shape

Click outside the shape on the Diagram pane.

To select all shapes of the same type

Press ALT and click the shape. All shapes of the same type are selected.

To select all shapes on the diagram

From the **Edit** menu, select **Select All** (shortcut keys CTRL+A.) To make multiple selections

1. Click the shape on the Diagram pane.
2. Hold down the SHIFT key and click another shape. Repeat until you select the desired number of shapes.
- Drag the cursor diagonally across the area you wish to select. All shapes in the selected area will be selected.

To select a group of shapes

To select a group of shapes drag the cursor diagonally across the area you wish to select. This is a simple and fast way to select a group of shapes on the diagram.

See the sample of the rectangular selection in Figure 1.

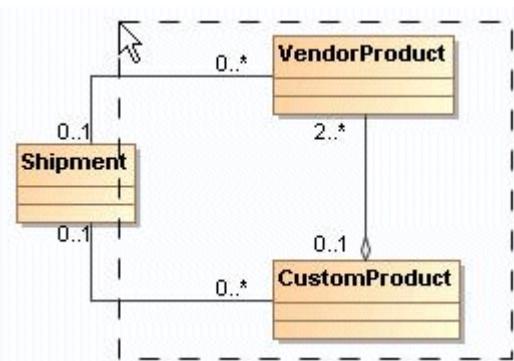


Figure 103 -- Rectangular selection

After the selection process represented in the Figure 1, the following shapes are selected, as shown in Figure 2.

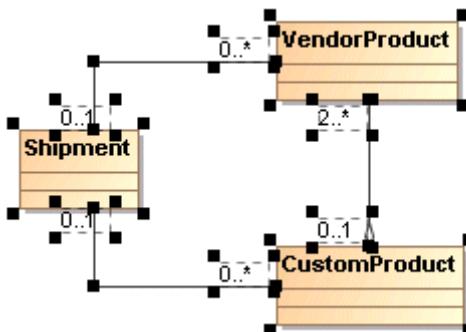


Figure 104 -- Rectangular selection result with partial selection coverage mode

The following rectangular selection modes are available:

1. *Partial coverage*. Symbols which are only partly covered with the rectangular selector are selected. See Figure 2. After the selection process represented in Figure 1, the class *Shipment* is selected by the rectangular selector, even though it was only partially covered.
2. *Complete coverage*. Only those symbols that are fully covered by the rectangular selection process will be selected. See Figure 3. For example, after the selection process shown in the Figure 1, the class *Shipment* and the associations are not selected because these symbols were not fully covered.

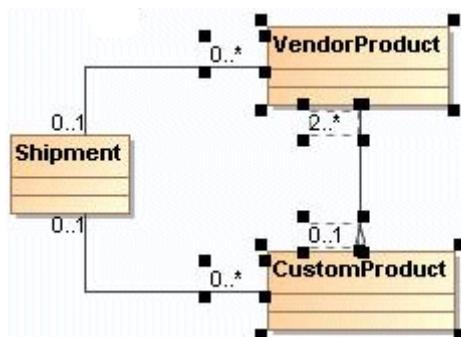


Figure 105 -- Rectangular selection result with complete coverage selection mode

5 DIAGRAMMING

Copying/Pasting Text or Images to Diagrams

Default selection mode is Partial coverage.

To quickly change the group selection mode from Partial coverage to Complete coverage mode or conversely:

- Press the **Ctrl** key and then drag the cursor diagonally across the area you want to select.

To change the group selection mode for the whole project:

- In the main diagram toolbar press the **Complete coverage mode for group selection** button (see Figure 106 on page 206).

OR

1. From the **Options** main menu, select **Environment**. The **Environment Options** dialog box appears.
2. In the **Diagram** branch, **Symbols Manipulation** group, change the property of the **Group selection mode** option.

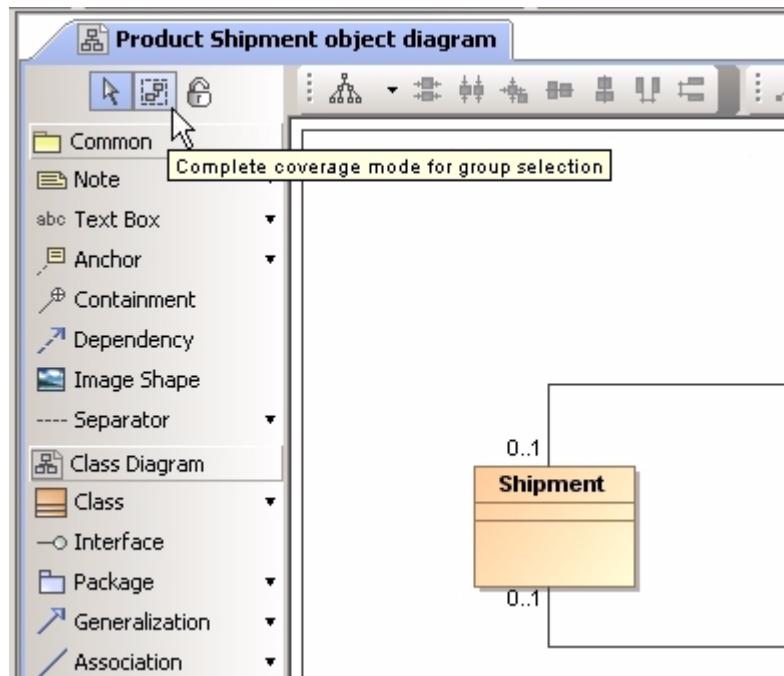


Figure 106 -- The Complete coverage mode for group selection button

Copying/Pasting Text or Images to Diagrams

It is now possible to copy and paste text or images to a diagram. A text box and an image shape will be available for the copied text or image. MagicDraw supports HTML and plain text, .gif, .jpg, .svg, and .png image file types. To copy and paste text or image:

1. Copy text or image(Ctrl+C).
2. Open a MagicDraw diagram.
3. Paste the copied text or image (Ctrl+P). The **Paste Special** dialog will open (Figure 107 on page 207).

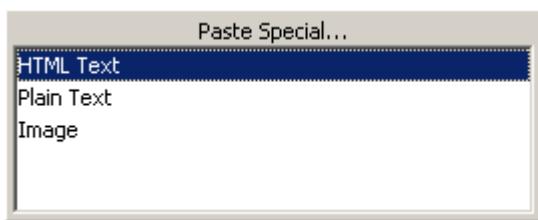


Figure 107 -- The Paste Special Dialog

NOTES

- This copy and paste feature functionality has been extended, allowing you to drag and drop from other applications such as Web browsers.
- The **Paste Special** dialog will open only if the clipboard contains any text or images, text, or HTML text formats.

Nesting Image Shapes

You can now drag an image to any elements in a diagram as nested a element (Figure 108 on page 207).

To drag an image to an element:

1. Select an image in the diagram pane.
2. Drag it to the image shape.

Dragged images will be nested by the following elements: Package, Model, Subsystem, Instance, Node, Part, Combined Fragment, Composite State (State diagram), Interruptible Activity Region, Structured Activity Node, Expansion Region, and Conditional Node (Activity diagram).



Figure 108 -- Samples of Images Nested to the Products Package and Server Component

Dragging, Copying, Cutting, and Pasting

Move a shape to another location on the diagram pane by dragging-and-dropping.

5 DIAGRAMMING

Dragging, Copying, Cutting, and Pasting

To drag multiple selected symbols

Select the symbols and drag them to the desired area on the diagram pane.

To copy a shape using dragging-and-dropping

Hold down the CTRL key while dragging the selected shape to the area where you wish to make a copy.

Drag and drop items from the browser to the diagram pane

1. In the Browser tree, select the created model element you wish to draw.
2. Drag it to the desired location on the diagram pane and drop it there.

NOTES

- You may select several model elements and draw them on the diagram pane.
- If the selected model elements are not compatible with the open diagram, you will not be allowed to draw those model elements.

To copy/cut and paste the selected shape on the diagram

1. From the **Edit** menu, select **Copy/Cut** (shortcut keys CTRL+C/CTRL+X.)
2. From the **Edit** menu, select **Paste** (shortcut keys CTRL+V.)

NOTE:

You may copy/paste many (but not all) model elements among various diagrams.

To paste one or more copied model elements by creating new data and symbols

From the **Edit** menu, select the **Paste With New Data** command (shortcut keys CTRL+E.)

To copy the whole diagram and paste it to MS Office or other application

1. Select or deselect all model elements on the diagram.
2. From the **Edit** menu, select **Copy as BMP Image**, **Copy as EMF Image**, **Copy as JPG Image**, or **Copy as PNG Image** (shortcut keys CTRL+SHIFT+B, CTRL+SHIFT+E, CTRL+SHIFT+J, or CTRL+SHIFT+P)
3. Open the desired application and paste the copied diagram.

To copy the selected model elements and paste to MS Office or other application

1. Select the desired model elements on the diagram pane.
2. From the **Edit** menu, select **Copy as BMP Image**, **Copy as EMF Image**, **Copy as JPG Image**, or **Copy as PNG Image** (shortcut keys CTRL+SHIFT+B, CTRL+SHIFT+E, CTRL+SHIFT+J, or CTRL+SHIFT+P).
3. Open the desired application and paste the copied model elements.

TIP!

You can drag and drop source code files from the native file manager to a MagicDraw Code Engineering Set.

NOTE

You can copy or cut and paste the text only when using the shortcut keys CTRL+C or CTRL+X and CTRL+V. When you use the buttons or commands, the whole element is copied/cut and pasted.

5 DIAGRAMMING

Dragging, Copying, Cutting, and Pasting

Tooltip text

MagicDraw now displays a tooltip that shows supplementary information of what will happen whenever you drag any elements (Figure 110 on page 210, Figure 111 on page 210, Figure 113 on page 211).

Dragging a File to an Element

The improved drag-and-drop capability allows you to drag any files from your file system to any element in the browser or in a diagram. A hyperlink will be automatically created for the element to which the file is dragged, allowing you to open the file by double-clicking the element.

To drag a file on an element:

1. Select a file in your Explorer (Figure 109 on page 209).
2. Drag it to the element in the browser or in a diagram in MagicDraw (Figure 110 on page 210). A hyperlink to the file will be created.

Figure 110 on page 210 show how a hyperlink from the Products package to the Products_description.doc file is created.

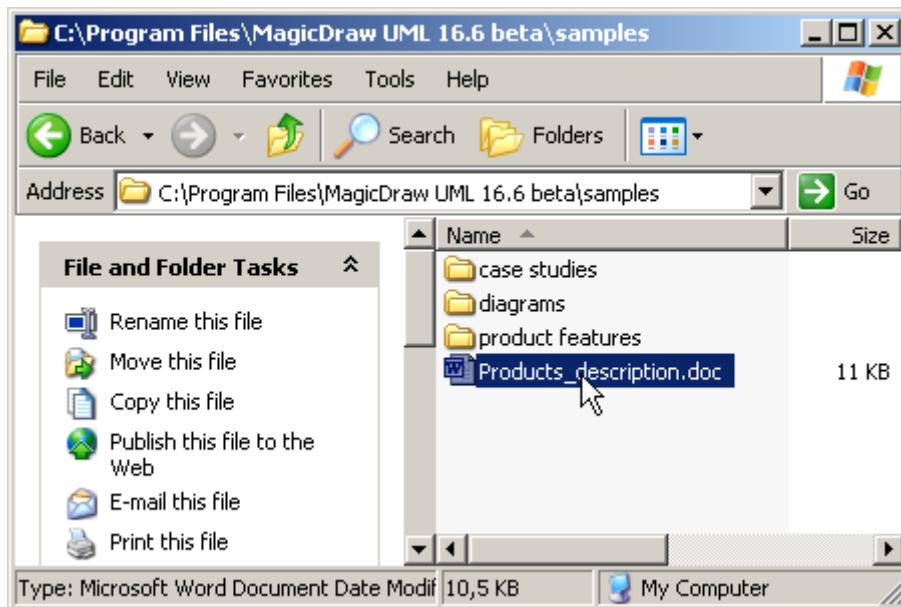


Figure 109 -- Selecting File in Your Explorer

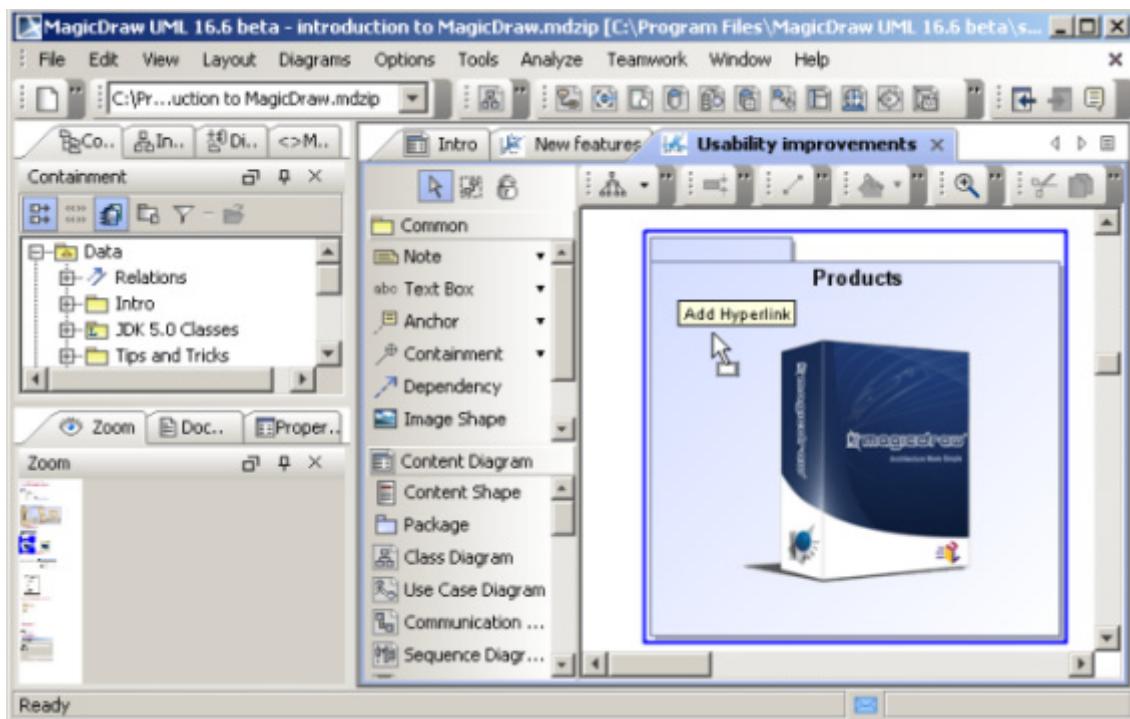


Figure 110 -- Dragging a File to the Element and Creating a Hyperlink

Drag and drop MagicDraw file on a diagram to open project

You can now drag MagicDraw project file from your file system and drop it on a diagram or any non-element. MagicDraw project will open (Figure 111 on page 210).

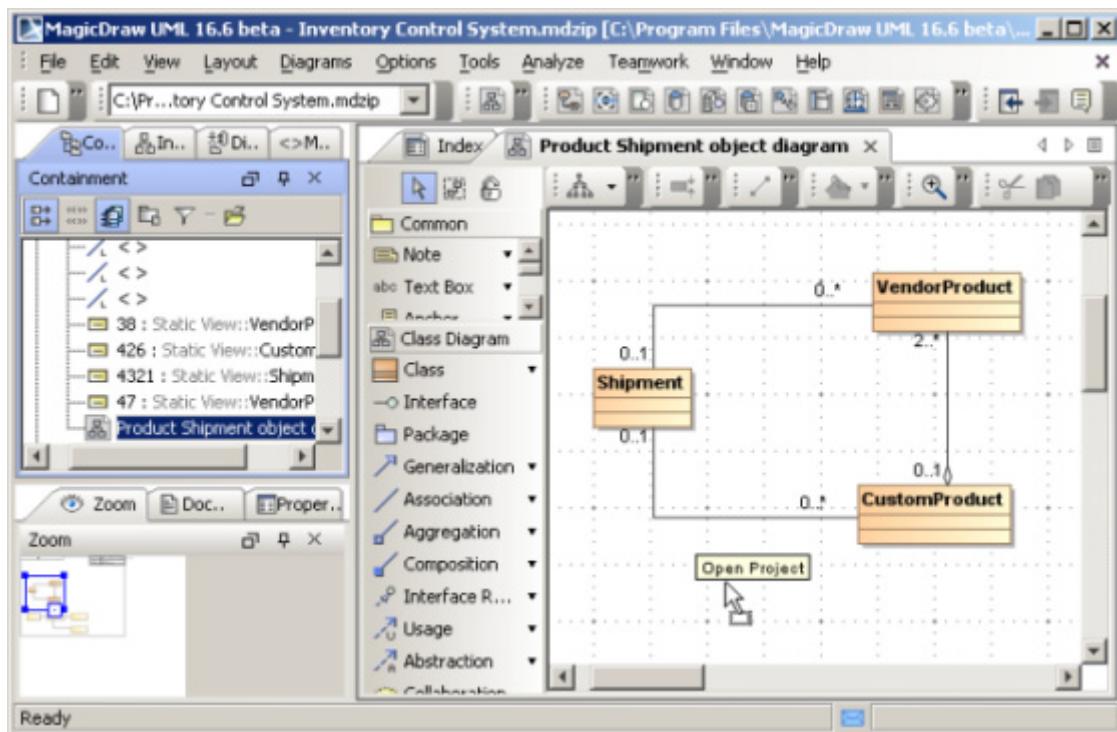


Figure 111 -- Dropping MagicDraw project file to diagram

5 DIAGRAMMING

Dragging, Copying, Cutting, and Pasting

Dragging an Image to an Element

You can now drag an image file from your file system to an element in the browser or in a diagram (Figure 112 on page 211). The image will be set as the value of the Image property of the element (Figure 113 on page 211).

The image will be set as a Stereotype icon if it is dragged to a Stereotype.



Figure 112 -- Image was Dragged and Dropped to the Class Shape

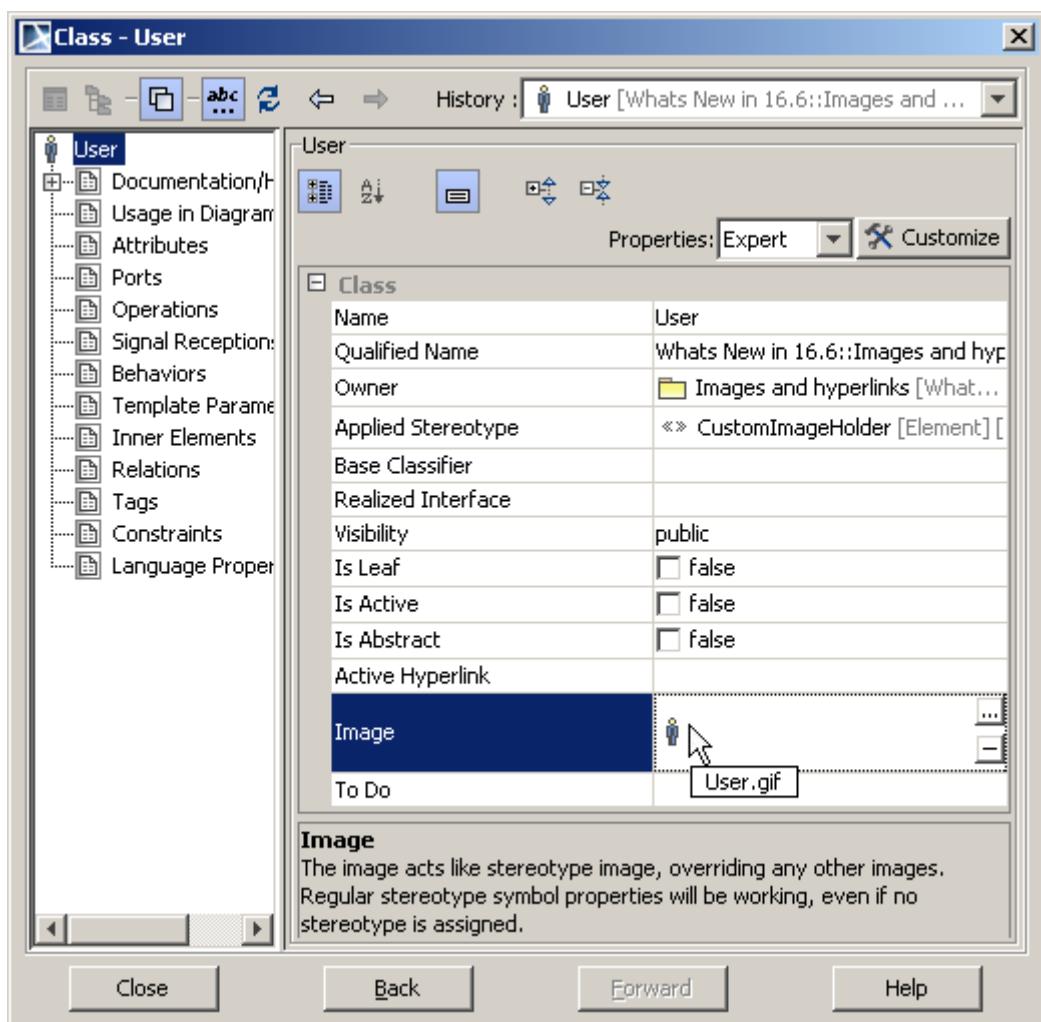


Figure 113 -- The Image Property in the Class Specification Dialog

Dragging Elements in the Specification Dialog

You can now drag any elements to any properties in the Specification dialog. For example, you can drag a Class element from the Containment tree to the **Type** property in the **Operation** Specification dialog. The Specification dialog will then assign that Class element as the type of the Operation element. In this case, the step-by-step example is as follows:

1. Open the **Customer Class** specification dialog, the **Operations** branch, and select the *getProfile* Operation.
2. Select the *CustomerProfile* class in the Containment tree () and drag it to the **Type** property area in the open Customer **Class** specification dialog (Figure 114 on page 212). The *getProfile* operation type will be assigned to the *CustomerProfile* class.

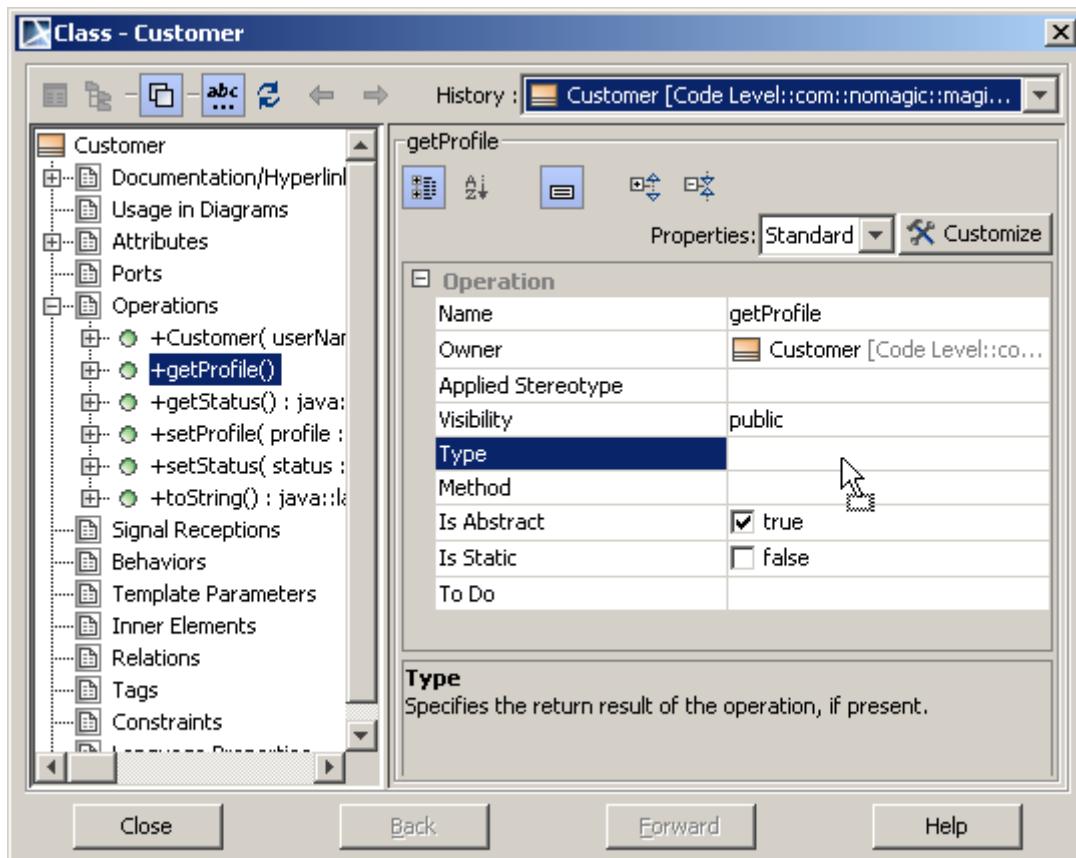


Figure 114 -- Dragging Class to the Property in the Specification Dialog

Dragging elements from the Specification dialog

You can now drag any elements from the Specification dialog to a diagram or to the browser. For example:

1. Open the Class specification dialog.
2. Select User Class, which is assigned as the Base Classifier (Figure 19).
3. Either (i) drag it to a diagram in the empty diagram pane to create a User Class symbol.

- Or (ii) drag it to the existing shape to create a new Attribute with Type.

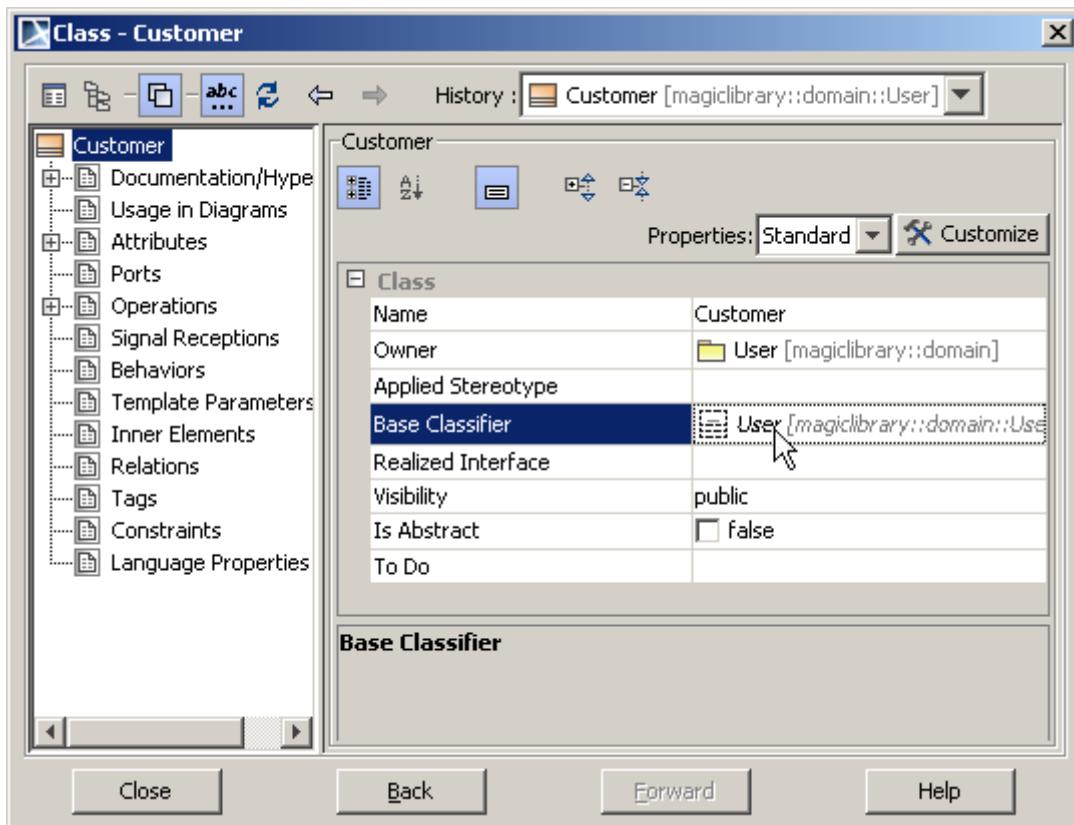


Figure 115 -- Dragging from the Customer Class Specification Dialog

Drag and drop Stereotype

You can now drag Stereotype from Browser or Diagram on any other element to apply it.

Drag and drop in Sequence diagram

- You can now drag an Operation from the browser to a Message in a Sequence diagram. The message will become a Call Message with once the operation has been assigned.

NOTE

The Lifeline type must have/inherit this operation.

- Dragging a Signal to a Message in a Sequence diagram will convert the Message into a Send Signal Message and assign the Signal to the Message.

Drag-and-drop in State Machine Diagrams

You can now drag an Event element to a Transition element in a State Machine diagram. A Trigger with this Event will be created for that Transition element.

Drag-and-drop in Activity Diagrams

It is now possible to:

- Drag a Signal to an Activity diagram to create a Send Signal Action.
- Drag a Signal to a Send Signal Action to set or change the Signal.

- Drag a Signal to a Accept Event Action to set or change the Signal.

NOTE

Once the Signal is assigned to the Accept Event Action, a Signal Event for the diagram and a Trigger for the action will be created automatically.

- Drag an Event to an Activity diagram to create an Accept Event Action.
- Drag an Event to an Accept Event Action to set the Event.
- Drag a Signal Event to an Activity diagram to create an Accept Event Action.

Zooming

Zooming allows you to select a particular part of a diagram, zoom into it, and make changes while working with a finer level of detail. You can also gain an overview of a diagram by zooming out from it.

To fit the current diagram in the window

- From the **View** menu or from the diagram shortcut menu, select **Fit in Window** (shortcut keys **CTRL+W**.)
- In the Browser **Zoom** tab, click **Fit in Window** .

To zoom into the current diagram

- From the **View** menu or from the diagram shortcut menu, select **Zoom In** (shortcut keys **CTRL+NumPad PLUS SIGN (+)** or scroll.)
- Click the **Zoom In** toolbar button .

To zoom out from the current diagram

- From the **View** menu or from the diagram shortcut menu, select **Zoom Out** (shortcut keys **CTRL+NumPad MINUS SIGN (-)** or scroll.)
- Click the **Zoom Out** toolbar button .

TIP!

You can zoom in or zoom out using the **CTRL+wheel** keys.

To restore the diagram to the original size

- From the **View** menu or from the diagram shortcut menu, select **Zoom 1:1** (shortcut keys **CTRL+NumPad SLASH MARK (/)**.)
- Click the **Zoom 1:1** toolbar button .

To view the selected shapes at maximum size

Select the shapes and then from either the **View** menu or the diagram shortcut menu, select **Zoom to Selection** (shortcut keys **CTRL+NumPad ASTERICS MARK (*)**.)

To select the zoom settings

1. From the **Options** menu, select **Environment**. The **Environment Options** dialog box opens.
2. Open the **Diagram** pane and change the **Zoom Step Size** property. The maximum number is 1.0 (you may zoom a diagram twice.)

NOTE

You may also zoom in or out of the diagram using the zoom panel in the Browser window. For the detailed description, see “Zoom panel” on page 86.

Using the Grid

The grid helps to arrange diagram symbols on the diagram pane. By default the grid is in the visible state.

To change the grid state (visible, not visible)

From either the **View** menu or from the diagram shortcut menu, select **Grid** and then select/clear the **Show Grid** check box.

To pull a path with the intersection of gridlines

From either the **View** menu or from the diagram shortcut menu, select **Grid** and then select/clear the **Snap Paths to Grid** check box.

To pull a shape with the intersection of gridlines

From either the **View** menu or from the diagram shortcut menu, select **Grid** and then select/clear the **Snap Shapes to Grid** check box.

To change the grid size

1. From either the **View** menu or from the diagram shortcut menu, select **Grid** and then select **Grid Size**.
2. The **Grid Size** dialog box opens.
3. Enter a grid size between 2 and 30 (default is 10).
4. Click **OK**.

To change the grid style

1. From the **Options** menu, select **Environment**. The **Environment Options** dialog box opens.
2. From the **Grid Style** drop-down list, select one of the following styles:
 - Dense
 - Sparse (default)

Layout

NOTE The diagram layout engine has nine layout options available in Standard, Professional, Architect, and Enterprise editions only.

With MagicDraw, it is easy to manage simple or complex diagrams using the automated layout features that optimize diagram layout for viewing.

Arrange your symbols on the Diagram pane using the **Layout** menu, or you can use the symbol shortcut menu when two or more symbols are selected. Since MagicDraw version 8.0, a new layout component has been applied with many more possibilities for arranging your models.

To resize the selected shape

Drag the corner of the shape to the desired size.

To automatically resize the selected shape to a preferred size

From the shape shortcut menu, select **Autosize**.

To alter the routing of the path line

Drag any point of the selected path in any direction.

The MagicDraw layout mechanism is built on various layout tools. All layout tools could be separated into 2 different groups: general layout tools and specific diagram layout tools. MagicDraw has only one specific diagram layout tool - the Class Diagram Layout Tool.

The General Layout Tools are:

- Orthogonal Layout Tool
- Hierarchic Layout Tool
- Tree Layout Tool
- Organic Layout Tool
- Circular Layout Tool
- Path Router

These layout tools are provided by Files layout tool component. You can arrange each diagram (except Sequence and Time Diagram) by using any of the 6 general layout tools. The Class diagram can also be arranged by using the Class Diagram Layout Tool.

Orthogonal Layout Tool

The Orthogonal Layout is well suited for medium sized sparse diagrams. It produces compact drawings with no shape overlaps, few crossings and few bends. All edges will be routed in an orthogonal style, i.e. only rectilinear style paths will be used.

5 DIAGRAMMING

Layout

Orthogonal layout options

| Option | Values | Default Value | Description |
|---------------------------------------|---------|---------------|--|
| Use Existing Drawing as Sketch | Boolean | False | The layout tool tries to "orthogonalize" the given sketch by interpreting it and without making too many modifications in respect to the original drawing. |
| Layout Only Top Level Symbols | Boolean | False | Keeps the relative position of symbols inside packages and performs the top level layout. |
| Group Layout Quality | 0-1 | 1 | Set the desired layout quality. Higher values result in less connection crossings and smaller layout area, but also increased computation time. |
| Orthogonal Grid | Integer | 50 | Defines the virtual grid spacing used by the layout tool. Each shape center point lies on a grid point. |

Hierarchic Layout Tool

The Hierarchic layout can be used to highlight the main direction or flow within a diagram. Cyclic dependencies of shapes will be automatically detected and resolved. Shapes will be placed in layers, arranged by hierarchy. Additionally, the ordering of nodes within each layer is chosen in such a way that the number of path crossings is minimal.

Hierarchy layout options

| Option | Values | Default Value | Description |
|--|---------|---------------|---|
| Reverse Orientation in Activity Diagram | Boolean | True | If True, orientation is reversed in activity diagram. |
| Minimal Layer Distance | Integer | 40 | Determines the minimal distance between shapes that reside in adjacent layers. |
| Minimal Shape Distance | Integer | 30 | Determines the minimal distance between adjacent shapes that reside in the same layer. |
| Minimal Path Distance | Integer | 30 | Determines the distance between adjacent pairs of horizontal path segments and between horizontal path segments and shapes. |
| Minimal First Segment Length | Integer | 10 | Determines the minimal length of the first and last segments for orthogonal path routings, i.e. the length from the intersection points to the first or last bend point respectively. |

5 DIAGRAMMING

Layout

| Option | Values | Default Value | Description |
|--------------------------------------|--|---------------|---|
| Orientation | Top to Bottom, Bottom to Top, Left to Right, Right to Left | Top to Bottom | Determines main layout orientation. |
| Shape Placement | Linear Segments, Polyline, Simplex, Pendulum, Median Simplex, Tree | Simplex | <p>Linear Segments. Aligns shapes in such a way that path segments tend to have very few bends. It is a very good choice in combination with Path Routing set to Orthogonal. However, this greatly increases layout width.</p> <ul style="list-style-type: none"> • Polyline. Aligns shapes by slightly reducing the width without shape overlaps. Although, paths will have lots of bends. • Pendulum. A sound combination of Linear Segments and Polyline. • Simplex. Produces high quality drawings. Similar to Linear Segments, aligns shapes in such a way that path segments tend to have very few bends. Additionally, the resulting layout will be more balanced and more compact. • Median Simplex. Tends to produce more locally symmetric layouts for the sake of a few more bends. • Tree. Produces very nice layout, when the graph is a tree. If the graph is not a tree, the placement policy of Linear Segments will be used. |
| Path Routing | Oblique, Orthogonal | Orthogonal | <ul style="list-style-type: none"> • Oblique. Paths are routed to oblique style with a certain number of bends. • Orthogonal. Paths are routed to orthogonal style. Orthogonal path routing increases the height of the layout. |
| Randomization Rounds | Integer | 40 | Determines the number of rounds that are initialized using different randomized starting positions. Greater values can lead to fewer crossings and longer running times. Huge diagrams with lots of inherent crossings should be processed using smaller values. |
| Layout Only Top Level Symbols | Boolean | False | Keeps the relative position of symbols inside packages and performs the layout only on the top level. |
| Make Sub Trees | Boolean | True | Inheritance paths are joined into inheritance arcs. |

Tree Layout Tool

The Tree layouter organizes diagram shapes into a tree structure. The Tree layout tool might be applied on shapes that have no undirected cyclic paths between them.

5 DIAGRAMMING

Layout

Table 3, Visible Tree layout options

| Option | Values | Default Value | Description |
|--------------------------------|--|-----------------|---|
| Layout Style | Directed, Balloon, Horizontal-Vertical | Directed | <ul style="list-style-type: none"> Directed. The tree is a hierarchy layout hierarchically with the root shape on the top. This is a good choice for directed trees with a unique root shape and a moderate number of shapes on a single hierarchy level. This layout style uses the current diagram layout as a sketch to determine the order of siblings at a common shape. Balloon. The tree is routed in a radial style. This is a good choice for undirected, dense, or huge trees with a high number of shapes on a single hierarchy level. Horizontal-Vertical. Children of a shape are either arranged on a horizontal or on a vertical line. Paths are routed orthogonally. This layout can be very compact if you choose the right alignment type for children of a node. |
| Make Sub Trees | Boolean | True | Inheritance paths are joined into inheritance arcs. |
| Directed | | | |
| Minimal Layer Distance | Integer | 50 | Determines the minimal distance between parent and child shapes. |
| Minimal Shape Distance | Integer | 50 | Determines the minimal distance between siblings of a shape. |
| Orientation | Top to Bottom, Bottom to Top, Left to Right, Left to Right | Top to bottom | Determines the main layout orientation. The layout tool tries to arrange shapes in such a way that all paths point in the main layout direction. |
| Port Style | Border Centered, Border Distributed | Border Centered | Determines the way paths are attached to shapes. <ul style="list-style-type: none"> Border Centered. Paths are attached to the center of the border of corresponding shapes. Border Distributed. Path attachment points are distributed along the border of corresponding shapes. |
| Orthogonal Path Routing | Boolean | True | If selected, all paths are routed orthogonally in a bus-like fashion. If not selected, paths are routed as straight line segments. |
| Balloon | | | |

5 DIAGRAMMING

Layout

| Option | Values | Default Value | Description |
|------------------------------|----------------------------|---------------|--|
| Root Shape Position | Directed Root, Center Root | Center Root | <ul style="list-style-type: none"> • Directed Root. Selects a shape with indegree zero if present. A good choice for directed root trees. • Center Root. Selects the root shape in such a way that the depth of the resulting tree is minimized. |
| Preferred Child Wedge | 0-360 | 340 | Determines the angular range of the sector that will be reserved for children of a shape. The remaining angular range will be automatically used to accommodate the edge that connects to the root node. |
| Preferred Root Wedge | 0-360 | 360 | Determines the angular range of the sector that will be reserved around the root shape to accommodate attached subtrees. |
| Minimal Path Length | Integer | 50 | Determines the minimal length of a path. |
| Compactness Factor | 0.1-0.9 | 0.5 | A smaller compactness factor will result in shorter paths and a more compact overall layout. |

Horizontal-Vertical

| | | | |
|---------------------------|---------|----|--|
| Horizontal Spacing | Integer | 20 | The minimal horizontal distance between adjacent shapes. |
| Vertical Spacing | integer | 20 | The minimal vertical distance between adjacent shapes. |

Organic Layout Tool

The organic layout is well-suited for the visualization of highly connected backbone regions with attached peripheral ring or star structures. These structurally different regions of a network can be easily identified by looking at a drawing produced by this layout tool.

Organic layout options

| Option | Values | Default Value | Description |
|------------------------------|---------|---------------|---|
| Preferred Path Length | Integer | 50 | Specify the preferred length of all paths. The layout tool tries to arrange the shapes in such a way that paths have a determined path length. |
| Obey Shape Size | Boolean | True | If True, the distance between two shapes is calculated with respect to the size of the shape. |
| Gravity Factor | -0.2-2 | 2 | Regulates the tendency of the shapes to be placed near the center of the diagram. The greater the factor is, the closer shapes are placed to the center of diagram. Negative values lead to huge layouts. |

5 DIAGRAMMING

Layout

| Option | Values | Default Value | Description |
|--------------------------------------|---------|---------------|--|
| Path Attraction | 0-2 | 2 | Higher values make Layout tool obey the given preferred path length. |
| Shape Repulsion | 0-2 | 0 | Higher values result in greater shape distances. |
| Activate Tree Beautifier | Boolean | True | If True, optimizes tree-like substructures of the diagram. The optimization process might ignore some layout options. |
| Layout Only Top Level Symbols | Boolean | False | If True, the algorithm keeps the relative position of the symbols inside the packages and performs the layout only on the top level. |
| Package Shape Compactness | 0-1 | 0.2 | Control the compactness of the package shape. Larger values lead to more compact package shapes, but paths between packages may be longer and the shapes inside packages tend to get clutched together at the center of the package. |

Circular Layout Tool

The Circular Layout produces arrangements that emphasize group and tree structures within a network. It partitions shapes into groups by analyzing the connectivity structure of the network. The detected groups are arranged on separate circles. The circles themselves are arranged in a radial tree layout fashion.

Circular Layouter options

| Option | Values | Default Value | Description |
|-------------------------------|---------------------------------|---------------|--|
| Layout Style | Compact, Isolated, Single Cycle | Compact | <ul style="list-style-type: none"> Compact. Each group will consist of shapes that are reachable by two disjoint paths. Shapes that belong to more than one group will be assigned exclusively to one group. Isolated. Each group will consist of shapes that are reachable by two path disjoint paths. All shapes belonging to more than one group will be assigned to an isolated group. Single Cycle. All shapes will be arranged on a single circle. |
| Minimal Shape Distance | Integer | 100 | Determines the minimal distance between borders of two adjacent shapes on a common circle. The smaller the distance, the more compact the resulting layout. |
| Auto Circle Radius | Boolean | True | If True, automatically determines the radius of each circle in the layout. An automatically chosen radius is usually the smallest possible radius that obeys Minimal Node Distance. |

5 DIAGRAMMING

Layout

| Option | Values | Default Value | Description |
|--------------------------------|---------|---------------|--|
| Fixed Circle Radius | Integer | 200 | If Auto Circle Radius is not set, this option determines the fixed radius for all circles in the resulting layout. Minimal Node Distance will be ignored in this case. |
| Preferred Child Wedge | 0-360 | 340 | Determines the angular range of the sector that will be reserved for children of a shape. The remaining angular range will be automatically used to accommodate the paths that connect to the root node. |
| Minimal Path Length | Integer | 50 | Determines the minimal length of a path that connects two shapes that lie on separate circles. The smaller the chosen value, the more compact the resulting layout. |
| Maximal Deviation Angle | 10-360 | 100 | The bigger the chosen value, the more compact the resulting layout. If the value is smaller than 90 degrees, the tree-edges might cross through the circularly arranged groups of shapes. |
| Compactness Factor | 0.1-0.9 | 0.5 | The smaller the compactness factor, the shorter paths and the more compact the overall layout. |

Orthogonal Path Router

This layout routes paths using only vertical and horizontal line segments, while keeping the positions of shapes in the diagram fixed. The routed paths usually will not cross any shapes and will not overlap any other paths.

Orthogonal Path layout options

| Option | Values | Default Value | Description |
|-----------------------------|---------|---------------|---|
| Minimal Distance | Integer | 20 | Specifies the minimal allowed distance between shapes and paths. |
| Use Existing Bends | Boolean | False | Specifies whether existing bends should be used as an initial solution for the new routing. |
| Route Only Necessary | Boolean | False | If True, only paths that violate the minimal distance criterion will be rerouted. |

Organic Path Router

This layout routes paths using an oblique path style, while keeping fixed positions of shapes on a diagram. The routed paths usually will not cross any shapes and will not overlap any other paths.

5 DIAGRAMMING

Layout

Organic Path layout options

| Option | Values | Default Value | Description |
|--|---------|---------------|--|
| Minimum Path Distance | Integer | 1 | Determines the minimum distance between any two path segments. |
| Custom Minimum Distance to Nodes | Integer | 10 | Determines the distance between any path segment and any shape side. The path router strictly adheres to the set value. Router does not use this value by default due to increased calculation times. |
| Route on Grid | Boolean | True | If True, all paths are routed on grid lines from the predefined grid. |
| Space Driven Vs. Center Driven Search | 0-1 | 1 | Determines the ratio between two complementary weighting strategies when looking for a path, namely "center driven" and "space driven" weighting. Values closer to 0 lead to paths that are more distributed over the available space. Values closer to 1 give more emphasis to paths closer to a path center. |
| Local Crossing Minimization | Boolean | True | If False, the number of crossings seen at a shape side can increase considerably. Since this option has a positive effect on a diagram "readability," it is enabled by default. |

Class Diagram Layout Tool

The Class diagram layout tool uses different layout algorithms to improve class diagram readability.

Class Diagram layout options

| Option | Values | Default Value | Description |
|-------------------------------|--|---------------|--|
| Minimal Layer Distance | Integer | 50 | Determines the minimal distance between parent and child shapes. |
| Minimal Shape Distance | Integer | 50 | Determines the minimal distance between the siblings of a shape. |
| Orientation | Top to Bottom, Bottom to Top, Left to Right, Right to Left | Top to bottom | Determines the main layout orientation. The layout tool tries to arrange shapes in such a way that all paths point in the main layout direction. |
| Compactness Factor | 0-1 | 1 | Adjusting this value can lead to a variety of differing layouts. For small values, the resulting layout will use more space and shapes tend to be far away from each other. Values around 0.5 lead to evenly distributed shapes, whereas values near 1.0 produce highly compact layouts. |

5 DIAGRAMMING

Layout

| Option | Values | Default Value | Description |
|--|---------|---------------|--|
| Space Driven Vs. Center Driven Search | 0-1 | 1 | Determines the ratio between two complementary weighting strategies when looking for a path, namely "center driven" and "space driven" weighting. Values closer to 0 lead to paths that are more distributed over the available space. Values closer to 1 give more emphasis to paths closer to a path center. |
| Build Generalization Hierarchies | Boolean | True | Classes connected by generalization paths are organized into hierarchies. |
| Build Realization Hierarchies | Boolean | False | Classes connected by realization paths are organized into hierarchies. |
| Build Containment Hierarchies | Boolean | False | Classes connected by containment paths are organized into hierarchies. |
| Make Sub Trees | Boolean | True | If enabled, inheritance paths will be joined into inheritance arcs. |

Activity Diagram Layout Tool

The Activity diagram layout tool uses different layout algorithms to improve activity diagram readability.

Activity Diagram layout options

| Option | Values | Default Value | Description |
|-------------------------------------|--|---------------|---|
| Minimal Layer Distance | Integer | 40 | Determines the minimal distance between parent and child shapes. |
| Minimal Shape Distance | Integer | 30 | Determines the minimal distance between the siblings of a shape. |
| Minimal path distance | Integer | 30 | Determines the distance between adjacent pairs of horizontal path segments and between horizontal path segments and shapes. |
| Minimal first segment length | Integer | 30 | Determines the minimal length of the first and last segments for orthogonal path routings, i.e. the length from the intersection points to the first or last bend point respectively. |
| Orientation | Top to Bottom, Bottom to Top, Left to Right, Right to Left | Top to bottom | Determines the main layout orientation. |

5 DIAGRAMMING

Layout

| Option | Values | Default Value | Description |
|--------------------------------------|--|---------------|---|
| Shape Placement | Linear segments Polyline Simplex Pendulum Median simplex Tree | Simplex | Linear segments - aligns shapes in such a way that path segments tend to have very few bends. It is a very good choice in combination with Path Routing set to Orthogonal. However, this greatly increases the width of the layout. Polyline - aligns shapes by slightly reducing the width without shape overlaps. Although paths will have lots of bends. Pendulum - a sound combination of Linear Segments and Polyline. Simplex - produces high quality drawings. Similar to Linear Segments, aligns shapes in such a way that path segments tend to have very few bends. Additionally, the resulting layout will be more balanced and more compact. Median Simplex - tends to produce more locally symmetric layouts for the sake of a few more bends. Tree - produces very nice layouts, when the graph is a tree. If the graph is not a tree, a placement policy of Linear Segments will be used. |
| Path Routing | Oblique Orthogonal | Orthogonal | Oblique - paths are routed to oblique style with a certain number of bends. Orthogonal - paths are routed to orthogonal style. Orthogonal path routing increases the height of the layout. |
| Randomization Rounds | Integer | 40 | Determines the number of rounds that are initialized using different randomized starting positions. Greater values can lead to fewer crossings and longer running times. Huge diagrams with lots of inherent crossings should be processed using smaller values. |
| Layout only top level symbols | Boolean | False | Keeps the relative position of symbols inside packages and performs the top level layout. |

For more information about the smart layout feature drawing diagram, see “Smart Activity Diagram layout” on page 534.

Business Process Diagram Layout tool

The Business process diagram layout uses different layout algorithms to improve business process diagram readability.

Business process diagram layout options

| Option | Values | Default Value | Description |
|-------------------------------|---------|---------------|--|
| Minimal Layer Distance | Integer | 40 | Determines the minimal distance between shapes that reside in adjacent layers. |

5 DIAGRAMMING

Layout

| Option | Values | Default Value | Description |
|-------------------------------------|--|---------------|--|
| Minimal Shape Distance | Integer | 30 | Determines the minimal distance between borders of two adjacent shapes on a common circle. The smaller the distance, the more compact the resulting layout. |
| Minimal Path Distance | Integer | 30 | Determines the distance between adjacent pairs of horizontal path segments and between horizontal path segments and shapes. |
| Minimal first segment length | Integer | 30 | Determines the minimal length of the first and last segments for orthogonal path routings, i.e. the length from the intersection points to the first or last bend point respectively. |
| Orientation | Top to Bottom, Bottom to Top, Left to Right, Right to Left | Top to bottom | Determines the main layout orientation. |
| Shape Placement | Linear segments Polyline Simplex Pendulum Median simplex Tree | Tree | <p>Linear segments - aligns shapes in such a way that path segments tend to have very few bends. It is a very good choice in combination with Path Routing set to Orthogonal. However, this greatly increases the width of the layout.</p> <p>Polyline - aligns shapes by slightly reducing the width without shape overlaps. Although paths will have lots of bends.</p> <p>Pendulum - a sound combination of Linear Segments and Polyline.</p> <p>Simplex - produces high quality drawings. Similar to Linear Segments, aligns shapes in such a way that path segments tend to have very few bends. Additionally, the resulting layout will be more balanced and more compact.</p> <p>Median Simplex - tends to produce more locally symmetric layouts for the sake of a few more bends.</p> <p>Tree - produces very nice layouts, when the graph is a tree. If the graph is not a tree, a placement policy of Linear Segments will be used.</p> |
| Path Routing | Oblique Orthogonal | Orthogonal | <p>Oblique - paths are routed to oblique style with a certain number of bends.</p> <p>Orthogonal - paths are routed to orthogonal style. Orthogonal path routing increases the height of the layout.</p> |
| Randomization Rounds | Integer | 40 | Determines the number of rounds that are initialized using different randomized starting positions. Greater values can lead to fewer crossings and longer running times. Huge diagrams with lots of inherent crossings should be processed using smaller values. |

| Option | Values | Default Value | Description |
|-------------------------------|---------|---------------|---|
| Layout only top level symbols | Boolean | False | Keeps the relative position of symbols inside packages and performs the top level layout. |

Quick Diagram Layout feature

You may use the Quick Layout feature when editing diagrams other than class or sequence. This feature applies recommended layout tool with default options on active diagram.

Label layout in the diagram

In MagicDraw 16.0 version there is improved label layout in the diagram. The following label positions are improved for paths, relationship ends, and shapes:

- Default label positions were reviewed and improved.
- Label positions after moving a path, shape, or related element.

Default label positions

Default label positions leaves after moving a path, shape, or related element if it is semantically logical decision. See an example below, there association multiplicities leaves at their default positions after class is moved.

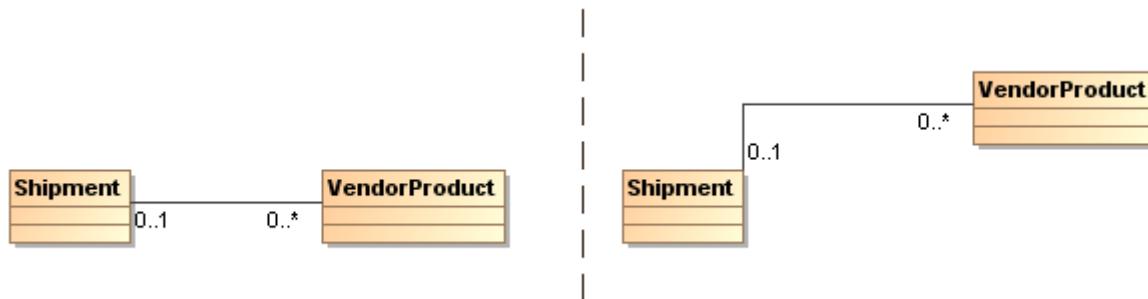


Figure 116 -- Default label position example

NOTE: If labels are at their default position, reset labels position functionality is disabled.

Labels positions after moving a path, shape or related element

After moving a path, shape or related element default label positions leaves if it is semantically logical decision.

For nicer representation of labels in diagram in the following cases labels positions are reseted to their default position automatically:

1. Symbol properties edit. When symbol properties edit causes label text box addition or removal from diagram pane labels positions are reseted.
2. Path, path end or port properties edit. When path, path end or port data edit causes label text box addition or removal from diagram pane labels positions are recalculated. See an example when qualifier is added in Figure 118 on page 228.
3. Path, shape or related element movement. See an example, when related element is moved Figure 119 on page 228.

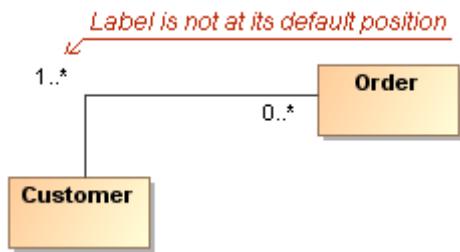


Figure 117 -- Label position before changes

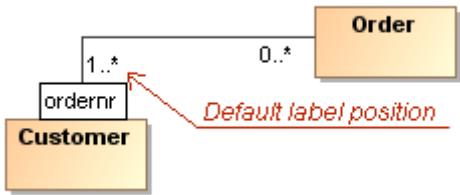


Figure 118 -- Label position is reseted to its default position after qualifier is added



Figure 119 -- Label position is reseted to its default position after Order class movement

Displaying label deviation from default position

While moving text box from default position, dotted line shows deviation from default position. This helps to see the current labels owner (See Figure 120 on page 228).



Figure 120 -- Dotted line, which shows deviation from the default position

Indicating label if it is not at its default position

If label is not at its default labels position, label right bottom corner is marked after label owner (path or shape) selection on diagram pane (see Figure 121 on page 229).

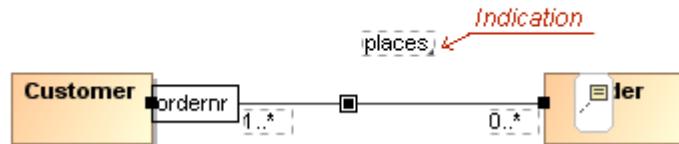


Figure 121 -- Marking the label when it is not at its default position

Showing Diagrams in Full Screen

If you want to see your diagrams in full screen and work exclusively from the diagram, use the (**show diagrams in full screen**) functionality. In full screen mode all necessary modeling commands will be visible, with option to hide, and the Browser will be in auto hide mode. You may also manage the MagicDraw interface components to be displayed or hidden.

To turn on diagram full screen mode

There are four ways to turn on the diagram full screen mode:

- Double click on the diagram tab with the diagram name at the top of diagram.
- From the diagram shortcut menu, select the **Show Diagrams in Full Screen** command.
- From the **View** menu, select the **Show Diagrams in Full Screen** command.
- Press the F11 key.

NOTE: You may change the **Show diagrams in full screen** shortcut key in the **Environment Options** dialog box, **Keyboard** pane. For more information, see “Assigning Shortcut Keys” on page 118.

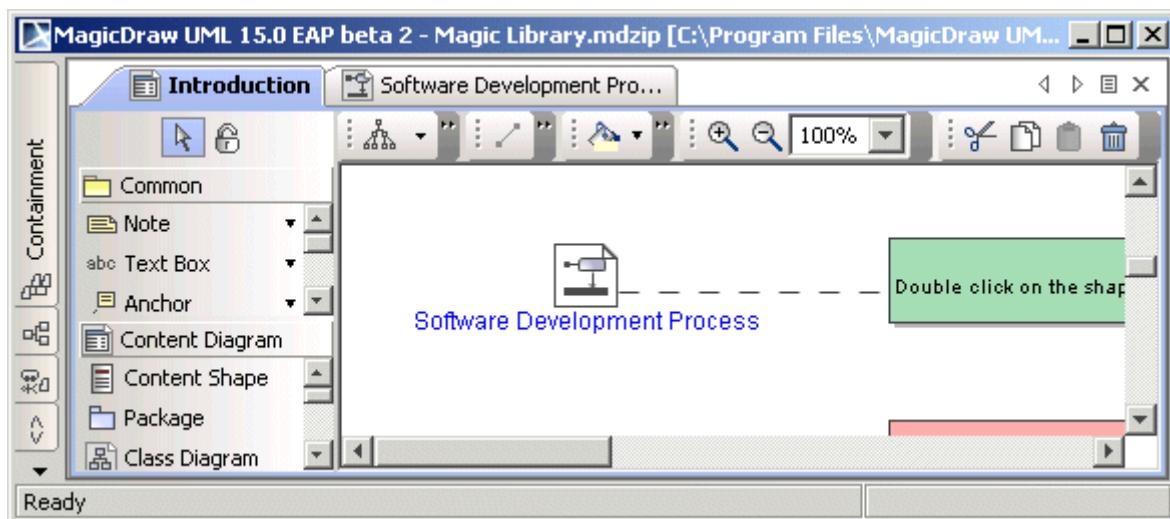


Figure 122 -- Diagram displayed in full screen mode

You may turn off the diagram full screen mode in the same way as turning it on.

Managing the MagicDraw interface components in the diagram full screen mode:

1. *Browser windows* are in auto hide mode when the diagram full screen mode is turned on. Located at the left side of the window are tabs of browser windows. To display the browser, move the mouse cursor over the browser window tab (for example, on the Containment tab) and the browser window will open. For more information about working with windows in auto hide mode, see “Using the Model Browser” on page 73.
2. *The main toolbar* is hidden when the diagram full screen mode is turned on. To display the main toolbar, clear the **Hide toolbars in full screen mode** check box in the **Environment Options** dialog box, **General** pane, **General** group.
3. *The diagram toolbar* is displayed when the diagram full screen mode is turned on. Right click the diagram toolbar to manage it.

NOTE

Showing the diagram in full screen mode is available only in Single Window (JIDE) interface style.

Floating Diagram Window

Floating diagram windows can be enabled by clicking the **Floating** command from the Diagram tab shortcut menu (Figure 123 on page 230).

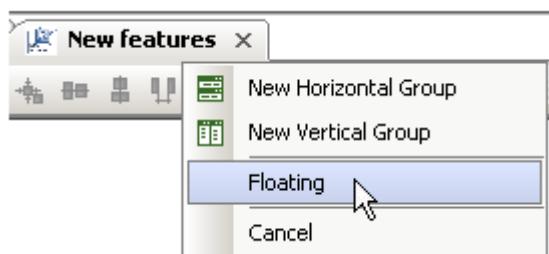


Figure 123 -- The Floating Command on the Diagram tab shortcut menu

Saving as an Image

Diagrams and symbols that were created in the model can be saved as an image in the following formats:

- Enhanced Metafile Format (*.emf) - supports language specific symbols.
- Encapsulated PostScript (*.eps)
- Joint Photographic Experts Group (*.jpg, *.jpeg)
- Portable Network Graphics (*.png)
- Scalable Vector Graphics (*.svg)
- Tagged Image File Format (*.tif, *.tiff)
- Windows Metafile Format (*.wmf)

MagicDraw version 15.0 and above allows exporting of a created diagram to Tagged Image File Format (TIFF) and choice of desired color space and compression. TIFF images can be edited and resaved without suffering

5 DIAGRAMMING

Saving as an Image

a compression loss and it is a flexible and adaptable file format for high color depth images. TIFF format is superior to JPG format.

To save the current diagram or the selected elements within the diagram as an image

1. From the **File** menu, select **Save As Image**. The **Save As Image** dialog box opens.
2. Select the **Active Diagram** or the **Selected Symbols** option button.
3. Select the image format (*.emf, *.eps, *.jpg, *.png, *.svg, *.tif, *.tiff, *.wmf), file name, and the location directory.

To save the selected diagrams of your project as images

1. Select the **Save As Images** command from the **File** menu. The **Save As Image** dialog box opens.
2. Check the **Selected diagrams** radio button, and select the diagrams you want to save as images from the **Not empty diagrams** list.
3. In the **Working Directory** field, type in the name of the destination directory or click the ‘...’ button to browse to the directory list.
4. Select the graphical file format in the **Image Format** drop down list (EMF, EPS, JPG, PNG, SVG, TIF, or WMF) and click **Save**.

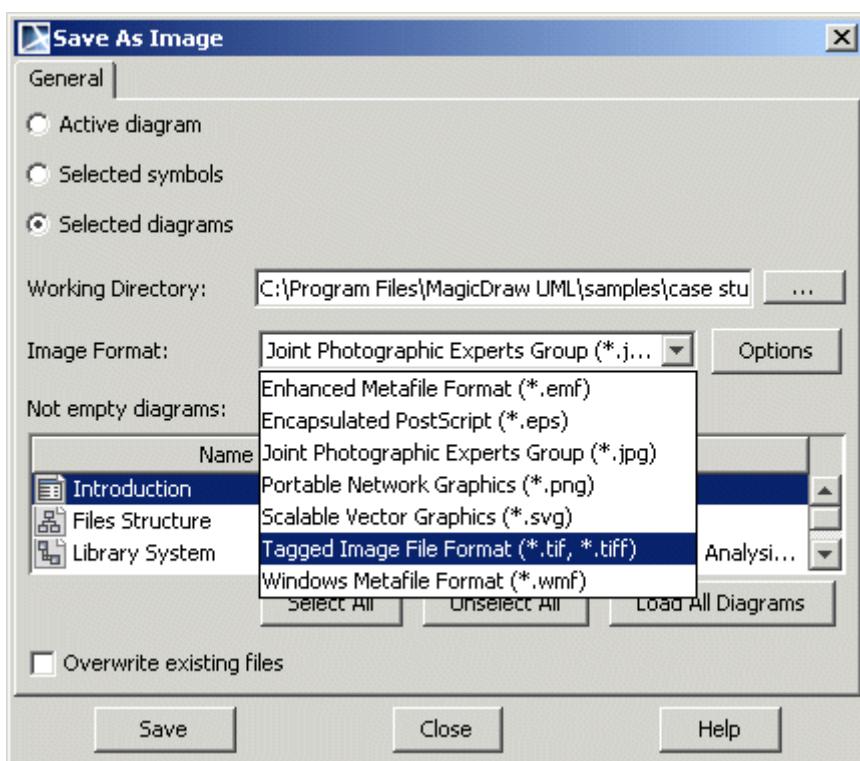


Figure 124 -- Save As Image dialog box

The filename of the saved diagram will be the same as the name of that diagram.

The **Not empty diagrams** list contains all exportable diagrams that contain UML elements. Select the diagrams you wish to export.

To make multiple selections

Press the CTRL key and click the diagrams you wish to export.

To select or clear all diagrams

Click the **Select All** button (press CTRL+A keys) or **Unselect All** button.

To display the list of all diagrams that are available in the project

Click the **Load All Diagrams** button in the **Save As Image** dialog box.

Setting image saving options

MagicDraw version 15.0 and above allows changing image size, resolution (DPI), and specifying other image properties specific to the selected image format.

To specify image options:

1. Select the **Save As Images** command from the **File** menu. The **Save As Image** dialog box opens.
2. Select the graphical file format in the **Image Format** drop down list.
3. Click the **Options** button near the **Image Format** drop down list. The **Image Export Options** dialog box opens.

Modify image export properties, which are described in the table below:

| Property name | Description | Formats |
|---|---|---|
| Save diagram background in image | Saves the diagram with background. By default, the diagram background is white after saving as an image. | This property is included in all format options. |
| Image resolution (DPI) | This property is the DPI property value with numeric value range from 1 to 4800. Default value is 72. | Property is not included in SVG and WMF format options. |
| Exported image size [%] | Define image size in percent. Default value is 100%. For example, if 200% is defined, then the view is enlarged (zoomed) before generating the image. Raster image will not lose its quality as additional pixels are introduced. | Property is not included in WMF format options list. |

Other image export options:

- **JPEG Compression Quality** property is included in **JPEG** format options list.
- **Use SVG <tag> for text output** property is included in **SVG** format options list.
- **Compression, Color space** properties are included only in **TIFF** format options list.

You may also define image saving options in the **Environment Options** dialog box. For more information see “Customizing Environment Options” on page 89.

Printing

In MagicDraw you can print an active diagram, multiple diagrams, or selected model elements. All printing menu commands are found in the **File** menu. You can also use the toolbar buttons or the shortcut keys.

Before printing, use the **Print** dialog box to set your printing options.

NOTE If the size of the text is too small, it may not be visible on the printed page.

To open the **Print** dialog box

- From the **File** menu, select **Print**.
- In the **Print Preview** screen, click the  button.

The **Print** dialog box contains the following tabs: **Print Range**, **Print Options**, and **Print Header/Footer**. Descriptions for these tabs appear in the following sections.

Print Range tab

In the **Print Range** tab, select the item you want to print.

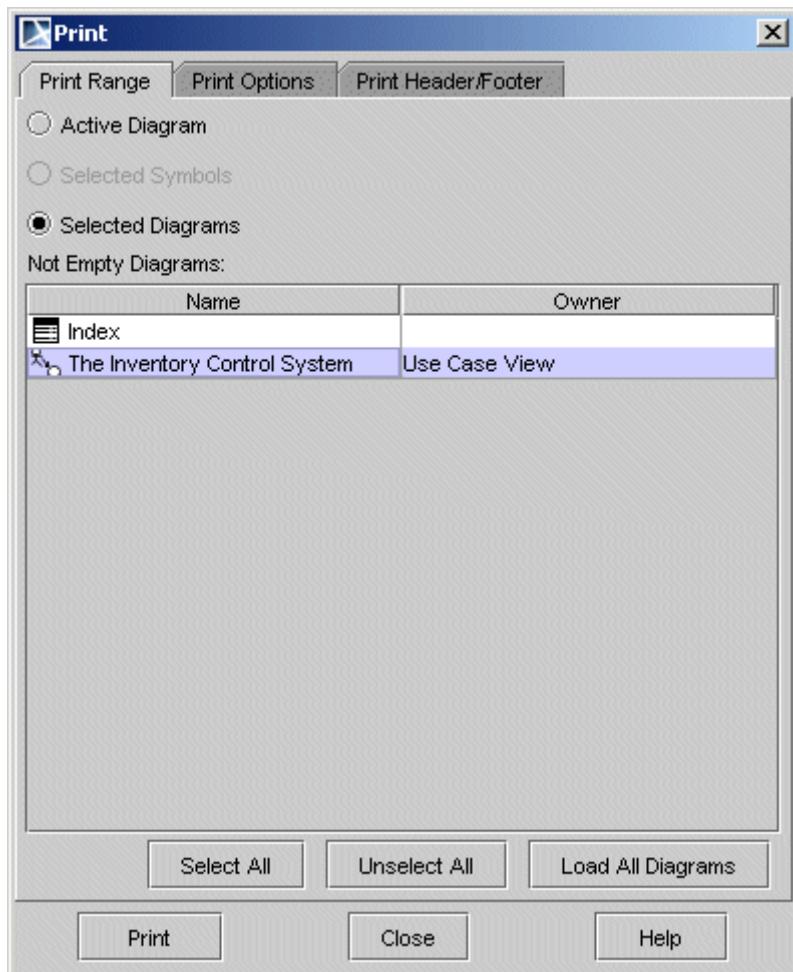


Figure 125 -- Print dialog box. Print Range tab

| Element name | Function |
|--------------------------|---|
| Active Diagram | Print the currently open diagram. |
| Selected Symbols | Print symbols you select on the diagram. The desired symbols should be selected to activate this option button. |
| Selected Diagrams | From the Not Empty Diagrams list, select the diagrams you want to print. |

| Element name | Function |
|--------------------------|--|
| Name | Show available diagrams in the project. To select the diagram for printing, click the name of the diagram in the list. The selected diagrams are highlighted. Press CTRL or SHIFT to select more than one diagram. |
| Owner | The name of the model element that owns the particular diagram. |
| Select All | Select all diagrams in the list for printing. |
| Unselect All | Remove the selection. |
| Load All Diagrams | Load all diagrams belonging to a project. By default only open diagrams are shown in the Print dialog box. |
| Print | Print the selected diagram(s). |
| Close | Close the dialog box. |
| Help | Display MagicDraw UML Help. |

Print Options Tab

Click the **Print Options** tab to customize the printing jobs.

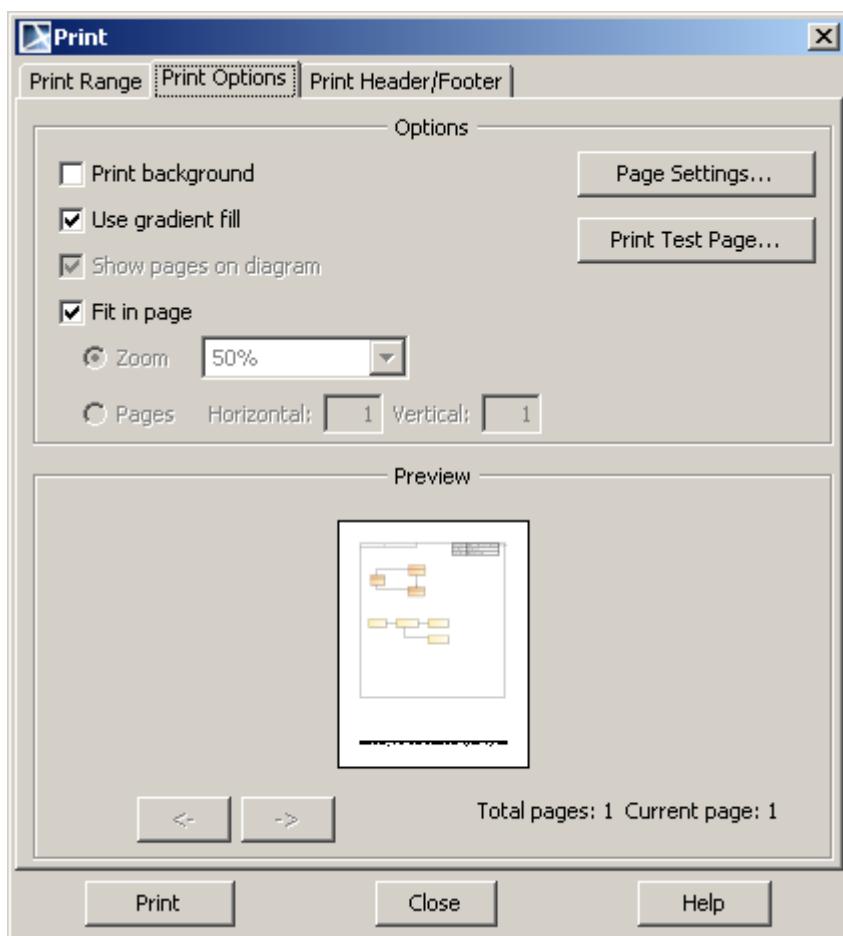


Figure 126 -- Print dialog box. Print Options tab

| Element name | Function |
|--------------------------|---|
| Print Background | Print the background color of your diagrams. |
| Use gradient fill | Select this option to enable diagram symbols gradient fill in printing. |

| Element name | Function |
|------------------------------|--|
| Show Pages on Diagram | Show the page boundary on the diagram pane. NOTE You will not see any boundary if the Fit in Page check box is selected. |
| Fit in Page | The printed diagram fits in one page. If the Fit in Page check box is cleared and the Show Pages in Diagram check box is selected, the gridlines of pages are shown on the diagram pane. |
| Zoom | Zoom the selected diagram to the size you want for printing. NOTE You are not allowed to zoom a diagram if the Fit in Page check box is selected. |
| Pages | Set the number of pages on which you want to print the diagram. <ul style="list-style-type: none"> ● Vertical. The number of vertical pages on which the diagram will be displayed. ● Horizontal. The number of horizontal pages on which the diagram will be displayed. |
| Page Settings | The Page Setup dialog box opens. |
| Print Test Page | Print the test page. Set print options in the Print Options dialog box. |
| Preview | Preview the diagram appearance before printing. |
| < | Preview the previous page. |
| > | Preview the next page. |

Print Header/Footer Tab

Click the **Print Header/Footer** tab to customize the header and footer of the printed pages.

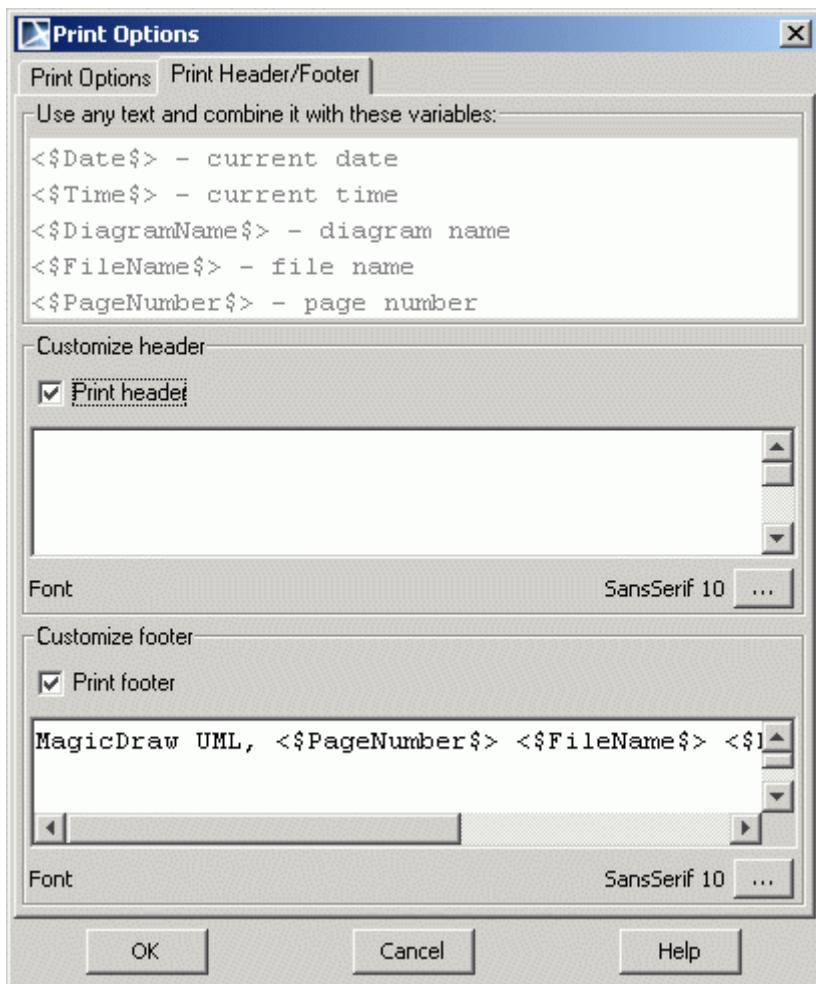


Figure 127 -- Print dialog box. Print Header/Footer tab

- **Use any text and combine it with these variables** box - use this box to indicate which fields you want to include in the header/footer.
- **Customize header** group box - prints the header. Select the **Print Header** check box and type the text you wish to be printed. Use the "..." button to select the desired font.
- **Customize footer** group box - prints the footer. Select the **Print Footer** check box and type or change the text you wish to print. Use the "..." button to select the desired font. By default <\$PageNumber\$> <\$FileName\$> <\$DiagramName\$> <\$Date\$> <\$Time\$> is printed.

6 WORKING WITH MODEL ELEMENTS

The “Working with Elements” Chapter includes the following sections:

- “Specification Window” on page 237
- “Default Property Values” on page 253
- “Formatting Symbols” on page 255
- “Style Engine” on page 258
- “Defining Hyperlinks Between Elements” on page 270
- “Owner of the Model Element” on page 274
- “Selecting an Element” on page 278
- “Refactoring” on page 281
- “Replacing an Element” on page 282
- “HTML Editor” on page 283
- “Copying/Opening Element URLs” on page 291

Specification Window

[View Online Demo](#)

Specification Windows

You can define all model elements in the Specification window.

The MagicDraw shortcut menus, toolbars, and browser help ease the task of editing model elements.

IMPORTANT

Beginning with version 8.0, MagicDraw enables the editing of model elements and symbol properties directly from the Browser, located in the **Properties** panel. For more information, see “Properties panel” on page 87.

6 WORKING WITH MODEL ELEMENTS

Specification Window

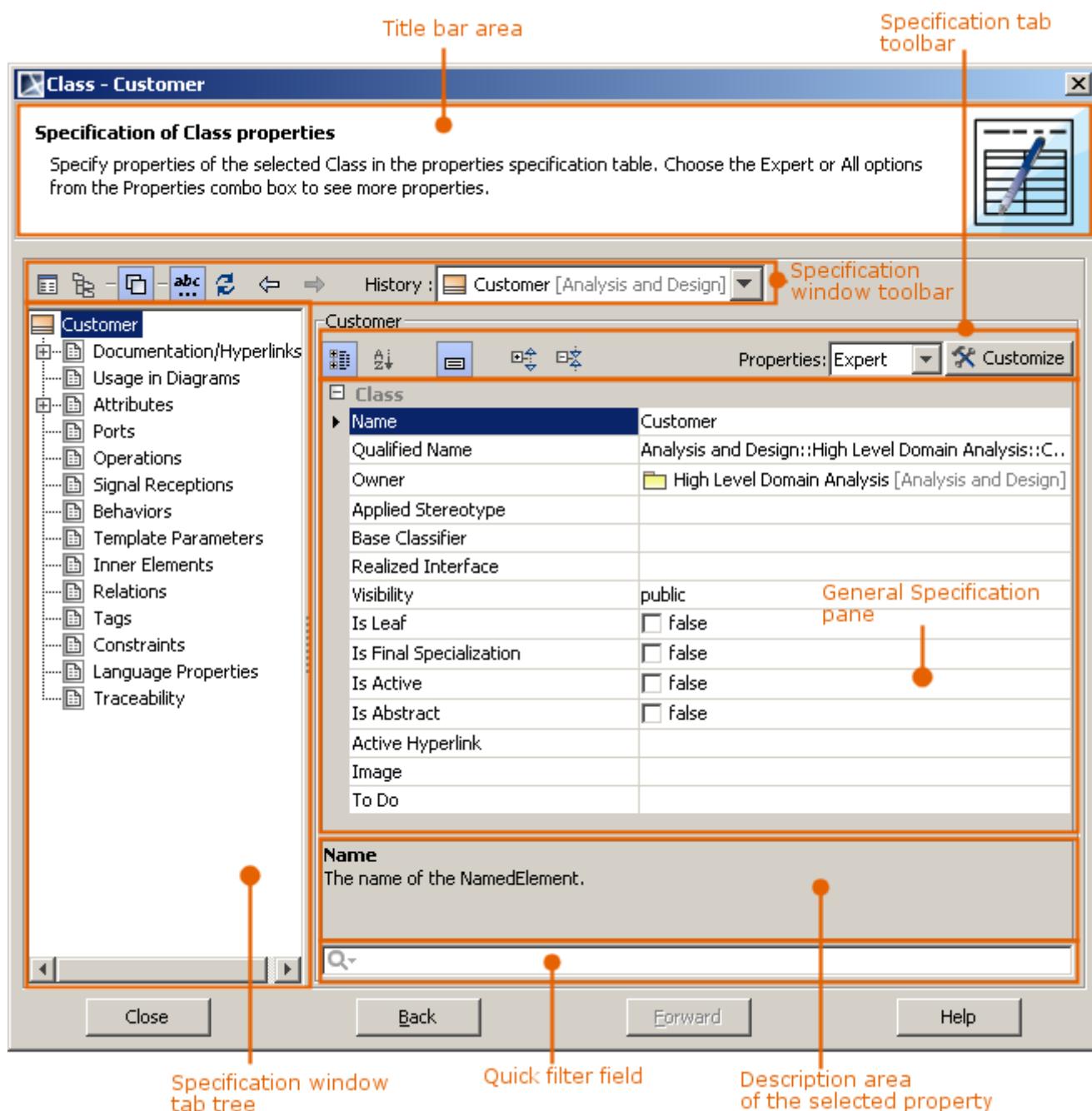


Figure 128 -- Specification window structure. General specification tab

The Specification window is used to define UML model elements such as class, package, activity, and others. Specification window is a non-modal window in which you may edit model element properties and work with model simultaneously.

Since MagicDraw version 10.0, tabs in Specification window have been replaced with a tree structure. This helps to avoid opening many specification windows. For example, class and its properties of all inner elements can be modified in the same window.

To open the corresponding Specification window

- From the selected symbol shortcut menu, select **Specification**.
- Double-click a symbol on the Diagram pane or in the Model Browser.

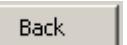
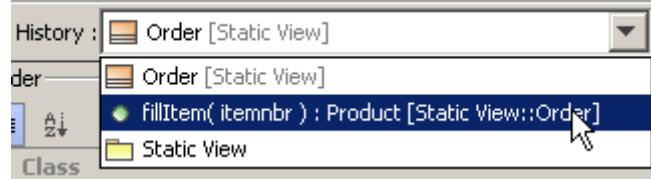
6 WORKING WITH MODEL ELEMENTS

Specification Window

- Select a symbol on the Diagram pane and press the ENTER key.

Functions of Specification Window

Using Specification window you may perform various actions that are necessary when working with model elements. Every model element has its own specification. In the table below, you will find described only the common functions that Specification window is used for.

| Function | How to |
|--|--|
| Add/modify model element properties and inner elements. | From the Specification window tab tree, choose the desired tab and fill its properties in the properties list. |
| Open referenced elements specifications and work with them in the same window. | Click  or  |
| Navigate between specifications that were opened | <ul style="list-style-type: none">• Click  or .• Click  or • Choose element, which specification you want to open, from the History list  |
| Manage relations | <ul style="list-style-type: none">• View all the relationships in which element participates.• Modify the name of the relationship.• View/change the direction of the relationship.• Modify the target element.  <ul style="list-style-type: none">• Create new outgoing  or • incoming  relationships. |
| Add/edit element documentation | <p>Add or edit documentation of the element. Documentation also can be written in HTML.</p> <p>More information about working with HTML text, see in Section “HTML Editor” on page 283.</p> |
| Manage element hyperlinks | <p>Open, edit, add, or remove hyperlinks from the selected model element to file, web page, other element/symbol, or requirements.</p> <p>More information about working with hyperlinks, see in Section “Defining Hyperlinks Between Elements” on page 270.</p> |

6 WORKING WITH MODEL ELEMENTS

Specification Window

| Function | How to |
|---|---|
| Manage element tags and their values | More information about working with tags you may find in Section “Editing Tagged Value” on page 630. |
| Manage element constraints | More information about working with constraints, you may find in Section “Working with Constraints” on page 633 |
| Track element’s symbol usage in diagrams and open these diagrams. | Find out in which diagrams the symbol is used. |
| Select model element in Containment tree | Click  or  Select in Containment Tree |
| Track elements traceability | More information about the traceability find in Section “Traceability” on page 386. |
| TIP! | Right-click the specification property and choose the action you want to perform. |

Specification Window Toolbar

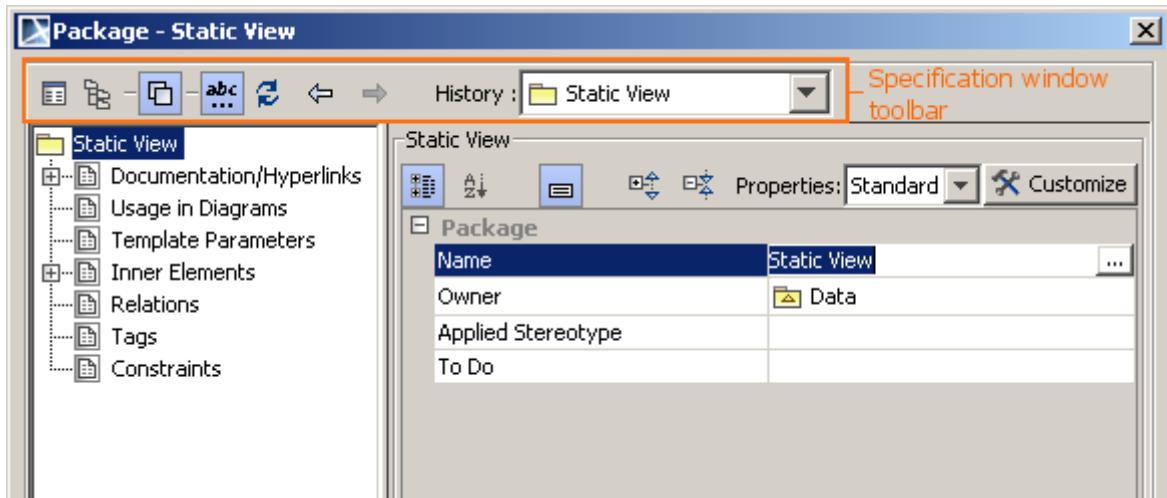


Figure 129 -- Specification window toolbar

| Icon | Name | Function |
|---|----------------------------|---|
|  | Open Specification | Opens Specification of the selected referenced element in the same window. |
|  | Select in Containment Tree | Selects the selected element in the Containment tree of the Model Browser. |
|  | Specification opening mode | If selected, the referenced Specification windows are opened in the same window. |
|  | Strip multiline text | If selected, property's text (e.g. ToDo), which covers more than five rows, is striped, not showing all of it, by adding three dots at the end of text. |

| Icon | Name | Function |
|------|---|---|
| | Refresh | Refresh the data of the Specification window. |
| | Back to <previously opened element specification> Forward to <previously opened element specification> | Use these buttons for switching between different specifications. You may also use Back , Forward buttons, and History drop-down list for switching between specifications. |

Quick filter

If the general Specification window pane contains 10 or more properties, Quick filter field appears. Using the Quick filter field you may quickly find the required property in the property list. This is especially handy when the properties list is rather long. Properties can be filtered by the text entered in this field.

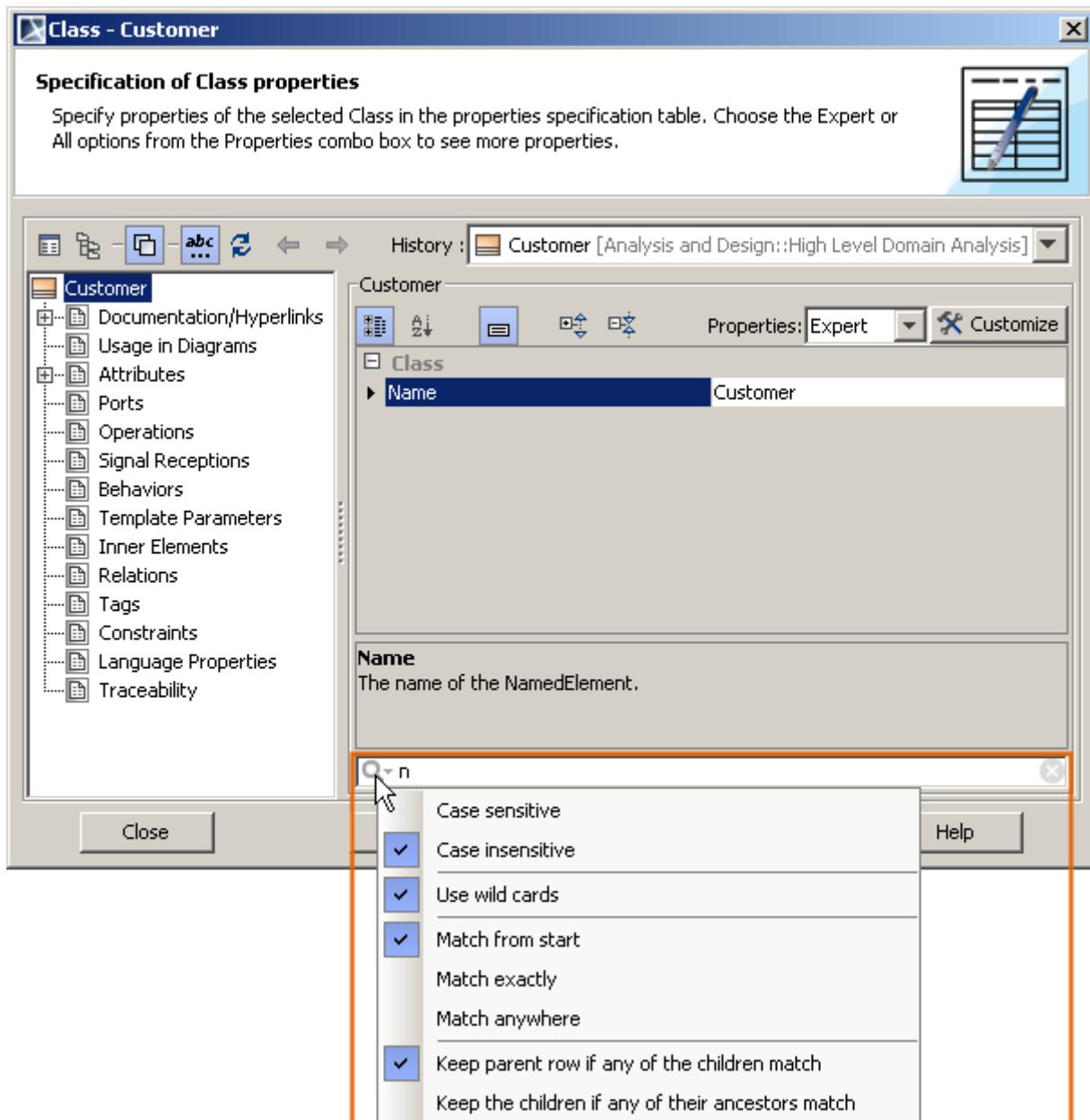


Figure 130 -- Quick filter field with its options.

| Option name | Function |
|-------------------------|---|
| Case sensitive | Words differs in meaning based on differing use of uppercase and lowercase letters. |
| Case insensitive | Words do not differ in meaning based on differing use of uppercase and lowercase letters. |
| Use wild cards | Increase the flexibility and efficiency of a quick filter search by using wildcard characters that substitute any of a class of characters in a search. |

6 WORKING WITH MODEL ELEMENTS

Specification Window

| | |
|--|--|
| Match from start | The search will be performed according to the first letters of the property. |
| Match exactly | The search will be performed according to the exact name of the property. |
| Match anywhere | The search will be performed according to any of the letter of the property. |
| Keep parent row if any of the children match | Show the category name of the found property. |
| Keep the children if any of their ancestors match | Show all other properties that are in the same category as the found property. |

Specification window tabs

Model elements that may participate in the relationships contain the **Relations** tab. All model element specification dialog boxes have **Template Parameters**, **Tags**, **Constraints**, and **Documentation/Hyperlinks** tabs. Descriptions of these tabs are presented in the following sections.

To open the corresponding Specification dialog box

- Select **Specification** from the selected symbol shortcut menu.
- Double-click a symbol on the Diagram pane or Browser.
- Select a symbol on the Diagram pane and press the ENTER key.
- The element Specification dialog box opens when you add one model element to another model element from another Specification dialog box. The second Specification dialog box opens on top of the first. Use the **Back to** or **Forward to** arrow buttons for switching between windows.

General tab

Name text box

| | |
|------|--|
| Name | |
|------|--|

Type or view the model element name. If you enter the name of an existing model element, an error message opens.

For some model elements (attribute, operation, and so forth), the default name *Untitled 1* is set. You can change this name to a preferred name.

Is Active, or Is Abstract check boxes

| | |
|-------------|--------------------------------|
| Is Active | <input type="checkbox"/> false |
| Is Abstract | <input type="checkbox"/> false |

When one of these check boxes is selected, the model element is correspondingly set as an active or abstract generalizable model element.

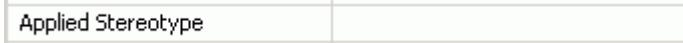
6 WORKING WITH MODEL ELEMENTS

Specification Window

A generalizable element is a model element that may participate in a generalization relationship.

| Name | Function |
|--------------------|--|
| Is Abstract | Specifies whether the generalizable element may or may not have a direct instance. True indicates that an instance of the generalizable element must be an instance of a child of the generalizable element. False indicates that there may be an instance of the generalizable element that is not an instance of a child. An abstract generalizable element is not instantiable since it does not contain all necessary information. |

Applied Stereotype



Click the "..." button to open the list of all available applied stereotypes, select the check box for the chosen stereotype and click **Apply**.

Visibility



To define an element access level, use the drop down list to set its visibility. There are four levels of access:

- **Public.** The element can be accessed by any outside object.
- **Package.** The element can be accessed by any classifier declared in the same package (or a nested subpackage, to any level).
- **Private.** The element can be accessed only from inside the current class.
- **Protected.** The element can be accessed from inside the current class and classes derived from that class.

ToDo



Type or view information about an element. The **To Do** property is used for keeping special information, exclusive cases, or additional records.

Image

Click the "..." button to assign the image to the element. Assigned image can be displayed on the shape or instead of the shape.

For more information, about changing the image display mode, see “Displaying icon or image” on page 256.

Documentation/Hyperlinks tab

Use the **Documentation/Hyperlinks** tab to add comments to the selected element and to assign hyperlinks. The hyperlink can direct the user to a model element, web page, or a file.

6 WORKING WITH MODEL ELEMENTS

Specification Window

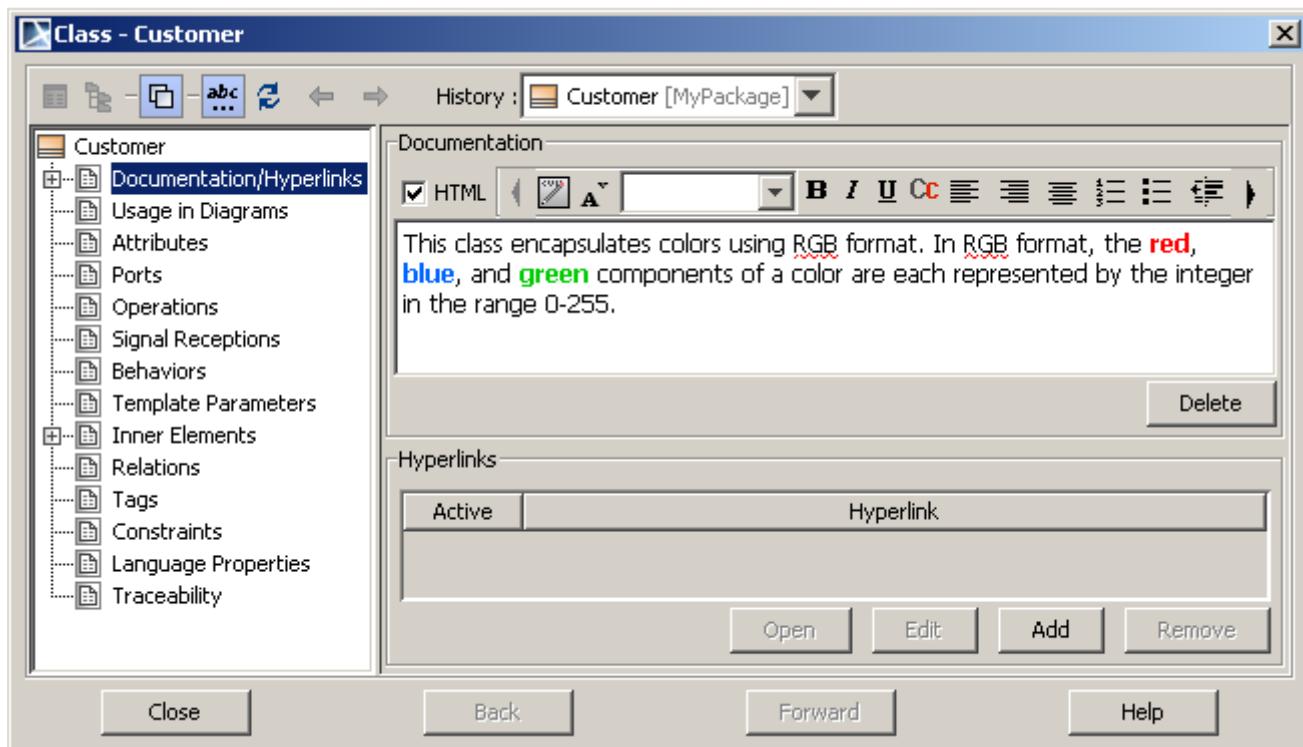


Figure 131 -- Specification window. Documentation/Hyperlinks tab

Writing HTML documentation

To write documentation in HTML format, simply select the HTML check box to display a menu with the available text formatting options.

For more information about HTML editor toolbar, see Section ["HTML editor toolbar"](#) on page 289.

Adding Hyperlinks

In the **Hyperlinks** group, manage the hyperlinks you want to add to the model element

| | |
|------------------|--|
| Active | If selected, the hyperlink is activated and will work when double-clicking the model element. |
| Hyperlink | Display information about the hyperlink: diagram or element name, file path, or URL name. |
| Open | Opens the previously assigned hyperlink. |
| Edit | The Insert Hyperlink dialog box opens. Edit the selected hyperlink. |
| Add | The Insert Hyperlink dialog box opens. Select the hyperlink you want to add to the model element. |
| Remove | Remove the selected hyperlink from the model element. |

6 WORKING WITH MODEL ELEMENTS

Specification Window

Attributes tab

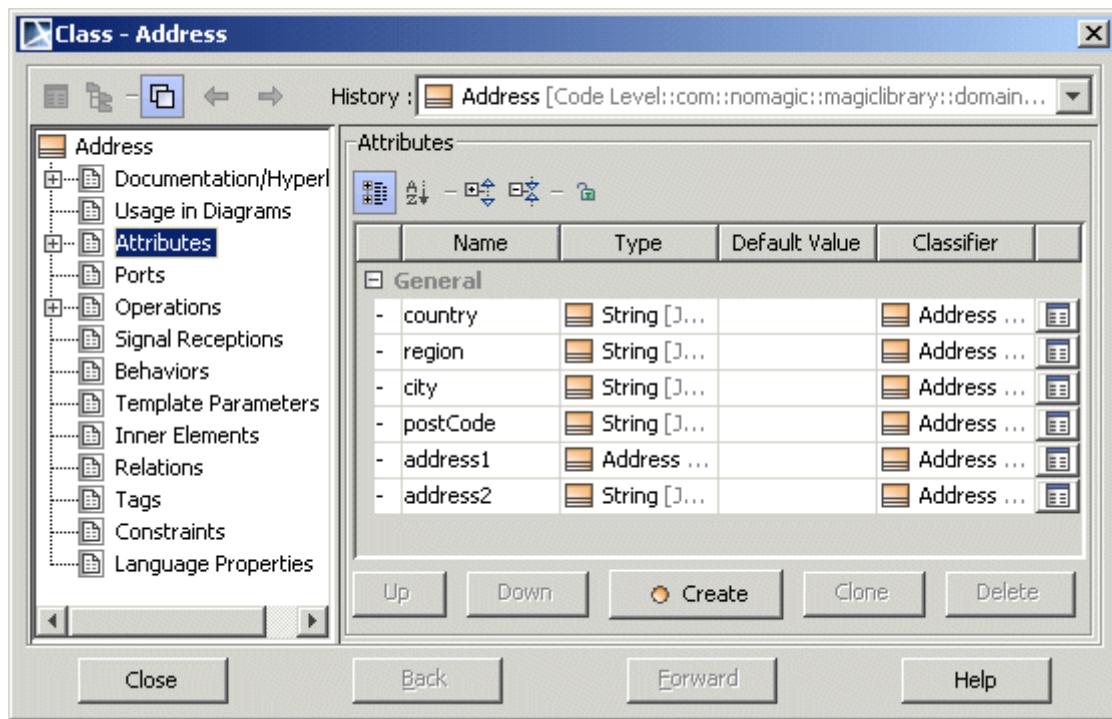


Figure 132 -- Attributes tab

The **Attributes** tab contains the model element attributes list and buttons for editing the attributes list.

| | |
|---|--|
| Name | Attribute name. |
| Type | Attribute type. It can be a primitive type or another class. |
| Default Value | Attribute default value. |
| Classifier | Class name that contains the current attribute. |
| Up | Move item to upper position in the list. |
| Down | Move item to lower position in the list. |
| Create | Add a new attribute to the class. The Attribute Specification dialog box opens. TIP! You may also add a new attribute from the classifier by pressing Insert. |
| Clone | Enabled when the element is selected in the list. A new element will be created. The new element derives all properties from the cloned element. The name will be changed to "<element_name><number>". |
| Delete | Remove the selected attribute from the classifier. TIP! You may also delete an attribute from the classifier by pressing Delete. |
|  | Click this button to open the Attribute Specification dialog box. |

Usage in Diagrams tab

For more information about searching for symbol usage in diagrams from the **Usage In Diagrams** branch, see "To search for diagrams in which symbol is used from the element Specification window" on page 408.

6 WORKING WITH MODEL ELEMENTS

Specification Window

Operations tab

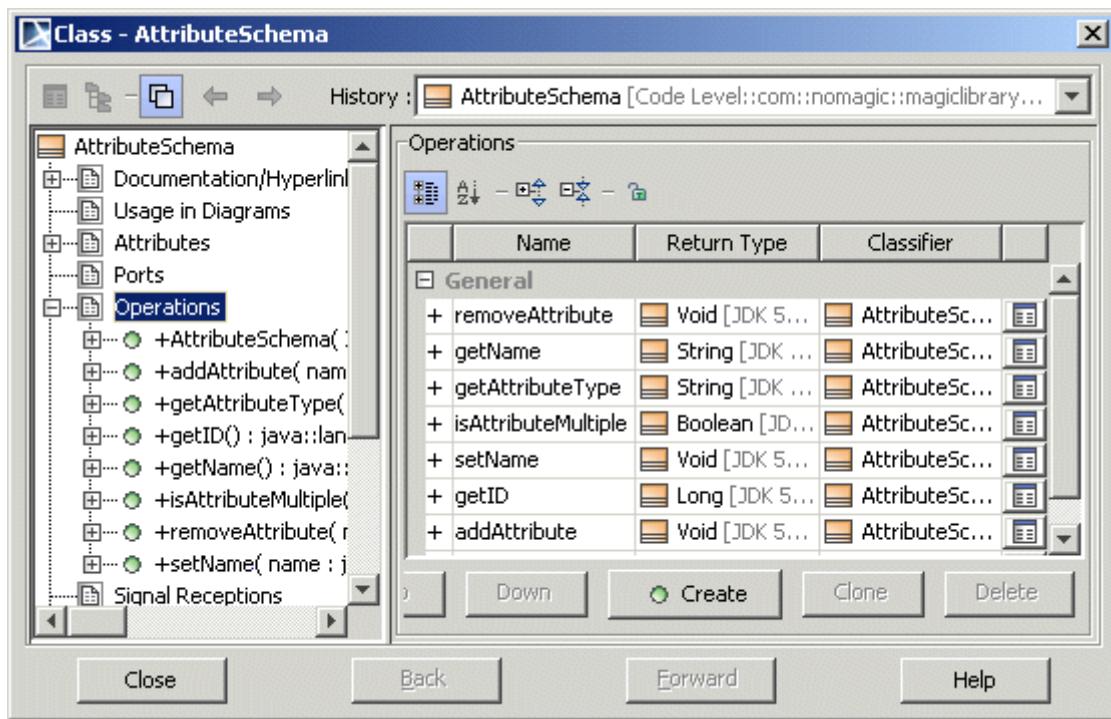


Figure 133 -- Operations tab

The **Operation** tab contains the model element operations list and buttons for managing this list.

| | |
|--------------------|--|
| Name | Operation name. |
| Return type | Operation return type. |
| Classifier | The name of the classifier containing the current operation. |
| Up | Move item to upper position in the list. |
| Down | Move item to lower position in the list. |
| Create | Add a new operation to the model element. The Operation Specification dialog box opens. TIP! You may also add a new operation from the classifier by pressing Insert. |
| Clone | Enabled when the element is selected in the list. A new element will be created. The new element derives all properties from cloned element. The name will be changed to "<element_name><number>". |
| Delete | Remove the selected operation from the model element. TIP! You may also delete an operation from the classifier by pressing Delete. |
| | Click this button to open the Operation Specification dialog box. |

Template Parameters tab

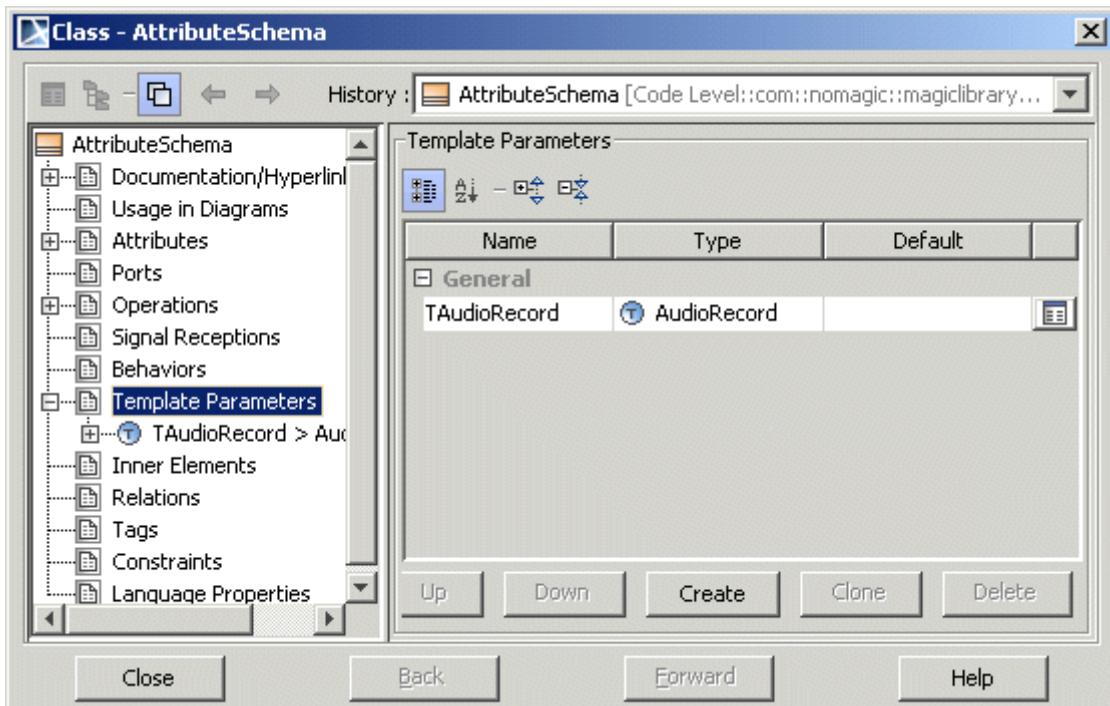


Figure 134 -- Operations tab

The **Template Parameter** tab contains the model element template parameters list and buttons for managing this list.

| | |
|---|--|
| Name | The name of the template parameter. |
| Type | Type of template parameter: classifier or data type. |
| Default | The Select Element dialog box opens. Here you can assign the element as the default element for the template parameter. |
| Up | Move the item to the upper position. |
| Down | Move the item to the lower position. |
| Create | Create a new template parameter. TIP! You may also delete a template parameter from the classifier by pressing Delete. |
| Clone | Enabled when the element is selected in the list. A new element will be created. The new element derives all properties from the cloned element. The name will be changed to "<element_name><number>". |
| Delete | Remove the template parameter from the class. TIP! You may also delete a template parameter from the classifier by pressing Delete. |
|  | Click this button to open the Template Parameter Specification dialog box. |

6 WORKING WITH MODEL ELEMENTS

Specification Window

Relations tab

The **Relations** tab contains the list of relationships in which the appropriate model element participates.

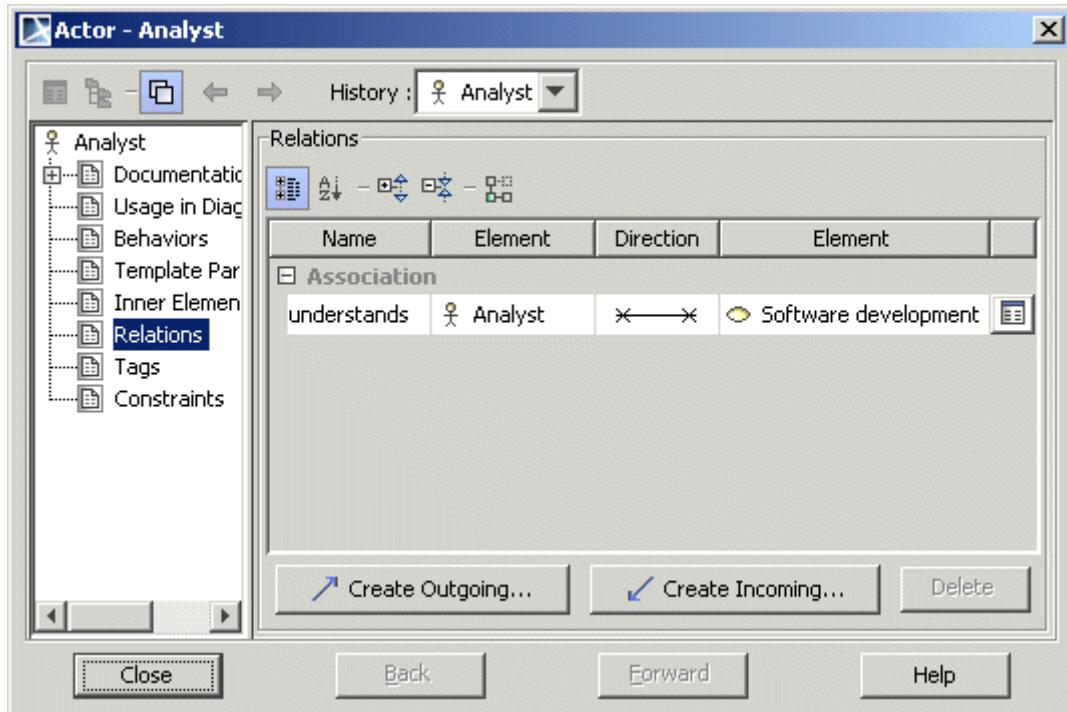
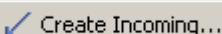
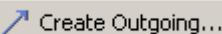


Figure 135 -- Specification dialog box. Relations tab

| Element name | Function |
|---|---|
| Name | Name of the relationship (optional). |
| Element | One of the relationship endings. |
| Direction | Shows a relationship's direction, helps to specify source and target. |
| Element | Another relationship ending. |
| Create Incoming | Create a new incoming relationship by choosing the relationship type from the appeared list.  |
| Create Outgoing | Create a new outgoing relationship by choosing the relationship type from the appeared list.  |
|  | After clicking this button, the relationship Specification dialog box opens. |
| Delete | Removes the selected relationship from the list. |

6 WORKING WITH MODEL ELEMENTS

Specification Window

Tags tab

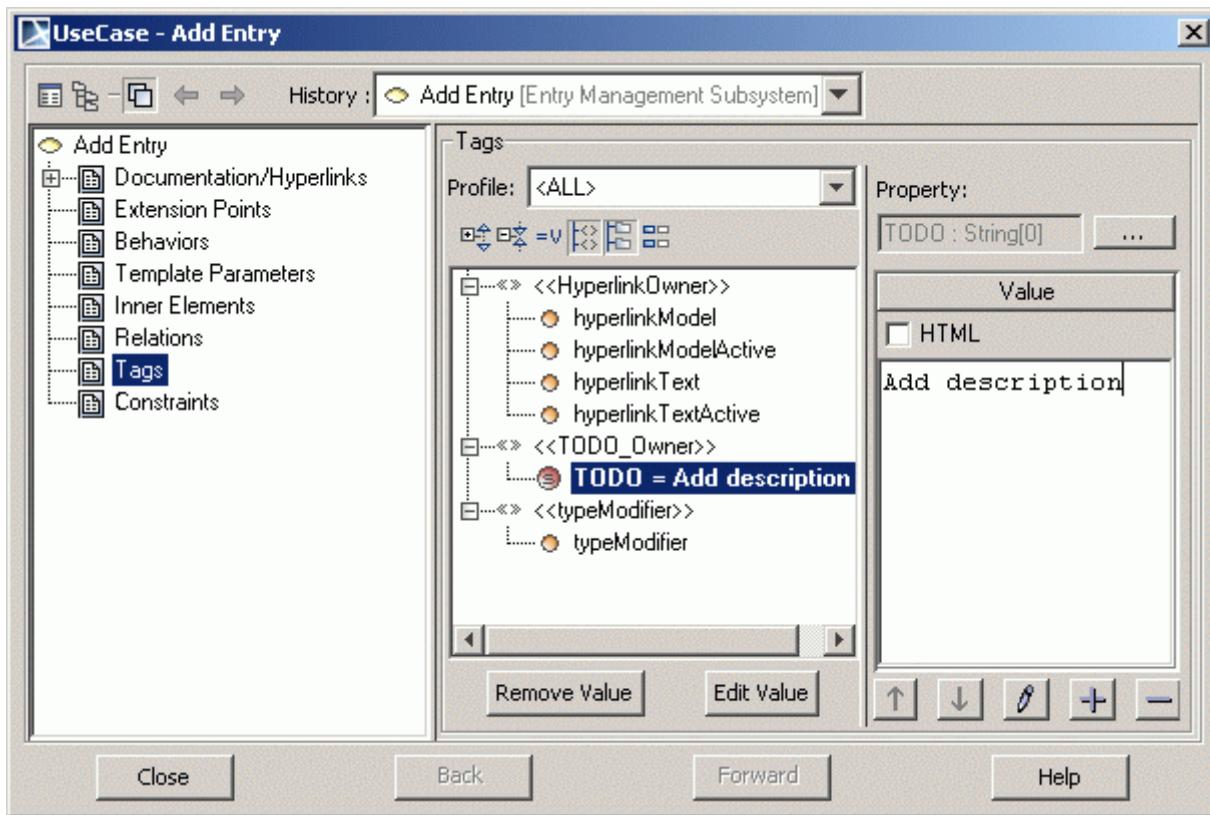


Figure 136 -- Specification dialog box. Tagged Values tab

| Element name | Function |
|---|--|
| Profile | Lists the profiles available for the current project. |
| Expand All Tree Branches | If tag definitions are grouped and those groups are collapsed, expands the groups. |
| Collapse All Tree Branches | If tag definitions are grouped and those groups are expanded, collapses the groups. |
| Show Tag Definitions with Values | Only displays in the list those tag definitions that have values. |
| Show Tag Definitions Groups | If tag definitions are grouped into specific “packages”, shows those “packages” on the list by grouping tag definitions. |
| Group By Stereotype | Sorts tag definitions by the assigned stereotypes. |
| Show Tag Definitions Types | Displays types of tag definitions in the list. |

| Element name | Function |
|---|--|
| | The list of tag definitions with values that are selected for the current model element. |
| Create Value | Creates a value for the selected tag definition. The right pane of the dialog box is activated. Select or enter the value. All data types and types of metamodel can be types of values. TIP! You may also create value by dragging and dropping an element from the Browser. |
| Remove Value (available only when the tag definition has a value) | Removes the value(s) from the selected tag definition. |
| Edit Value | The Tagged Value Specification dialog box opens, allowing you to edit or extend the selected tagged value. |
| Edit Tag Definitions | The Profiles dialog box opens. Edit, add, or remove tag definitions for the current model element. |
| Right pane of the dialog box | |
| Tag Definition “...” | Click the “...” button and edit the selected tag definition in the Tag Definition Specification dialog box. |
| HTML | Set the tagged value text as HTML. |
| Value (if the value is added) | Type or select the value. |
| Edit | Edit the selected value. |
| Add | Add a new value. |
| Remove | Remove the selected value. |

For more information about how to create a new tagged value, see “To create a new tag definition” on page 630.

6 WORKING WITH MODEL ELEMENTS

Specification Window

Constraints tab

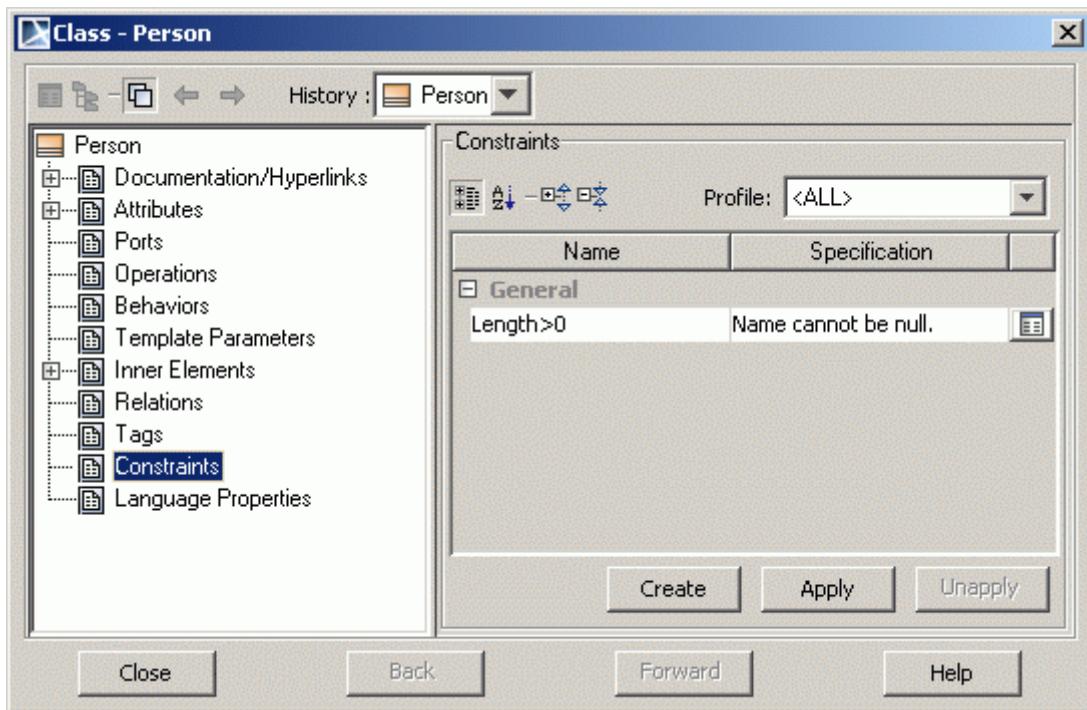


Figure 137 -- Specification dialog box. Constraints tab

| Element name | Function |
|----------------------|---|
| | The list of constraints assigned to the current model element. |
| Name | Enter the name of the constraint. |
| Specification | A comment associated with the selected constraint. To edit the comment, double click the Specification line. Click the ‘...’ button to open the Edit Specification dialog box. This allows you to edit expressions and select the Object Constraint Language (OCL) to check the expression syntax. |
| | Click this button to open the Constraint Specification dialog box. |
| Create | Creates a constraint. |
| Apply | The Select Elements dialog box opens. Select an existing constraint from the model and apply it to the element. |
| Unapply | Removes the selected constraint from the list. |

Traceability tab

The Traceability tab in element's Specification window is one of the places, wherein the element's traceability properties, showing its realizing and/ or more specific elements, are represented.

For more information about traceability feature, see section “Traceability” on page 386

Buttons available in the Specification window



| Button | Function |
|----------------|---------------------------------------|
| Close | Save changes and exit the dialog box. |
| Back | Return to the previous dialog box. |
| Forward | Proceed to the next dialog box. |
| Help | Display MagicDraw Help. |

Default Property Values

MagicDraw version 15.0 and above allows for defining the initial (default) properties for elements.

The Default element properties can be defined for:

- the whole project.
- the specific diagram.

To set the default properties for the whole project

1. From the **Options** main menu, select the **Project** command. The **Project Options** dialog box opens.
2. Expand the **Default model properties** branch. Select the exact element and in the right pane side, change the property value.

After creating a new element it will have new property values. Values for previously created elements will not be changed.

To reset element properties to the default value, click the **Reset to Defaults** button. To reset property values for all elements select the **Default model properties** branch and click the **Reset to Defaults** button.

(Exception: interface attribute default visibility will always be #public, no matter what your settings.)

To set the default properties for the specific diagram

1. From the **Diagrams** main menu, select the **Customize** command. The **Customize Diagram** wizard opens.
2. Define the new or created diagram properties and in the **Specify toolbar buttons** step, click the **Add** button. In the appeared menu that opens, select the **New Button** command. The **Edit button** dialog box opens.

6 WORKING WITH MODEL ELEMENTS

Default Property Values

3. Open the **Element Properties** tab. Select the **Specify own values** radio button and change the default element property values.

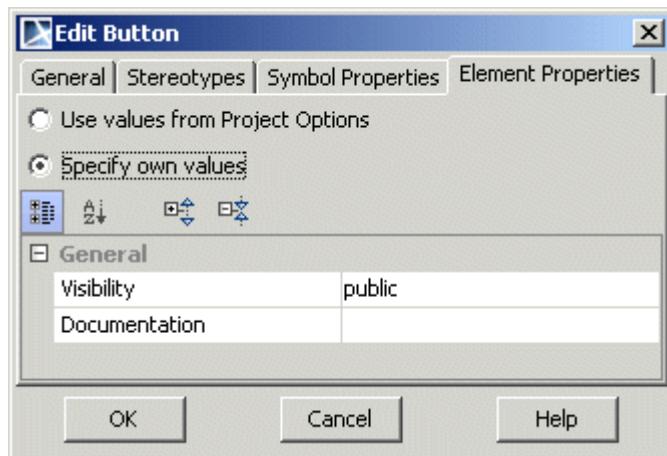


Figure 138 -- The **Edit Button** dialog box, **Element Properties** tab

Create a new element from the customized diagram toolbar and the element will have the defined, default properties.

For more information about the **Customize Diagram** wizard, see *UML Profiling and DSL UserGuide.pdf*, which is located in the <MagicDraw installation directory>/Manual folder.

Sharing the default property values

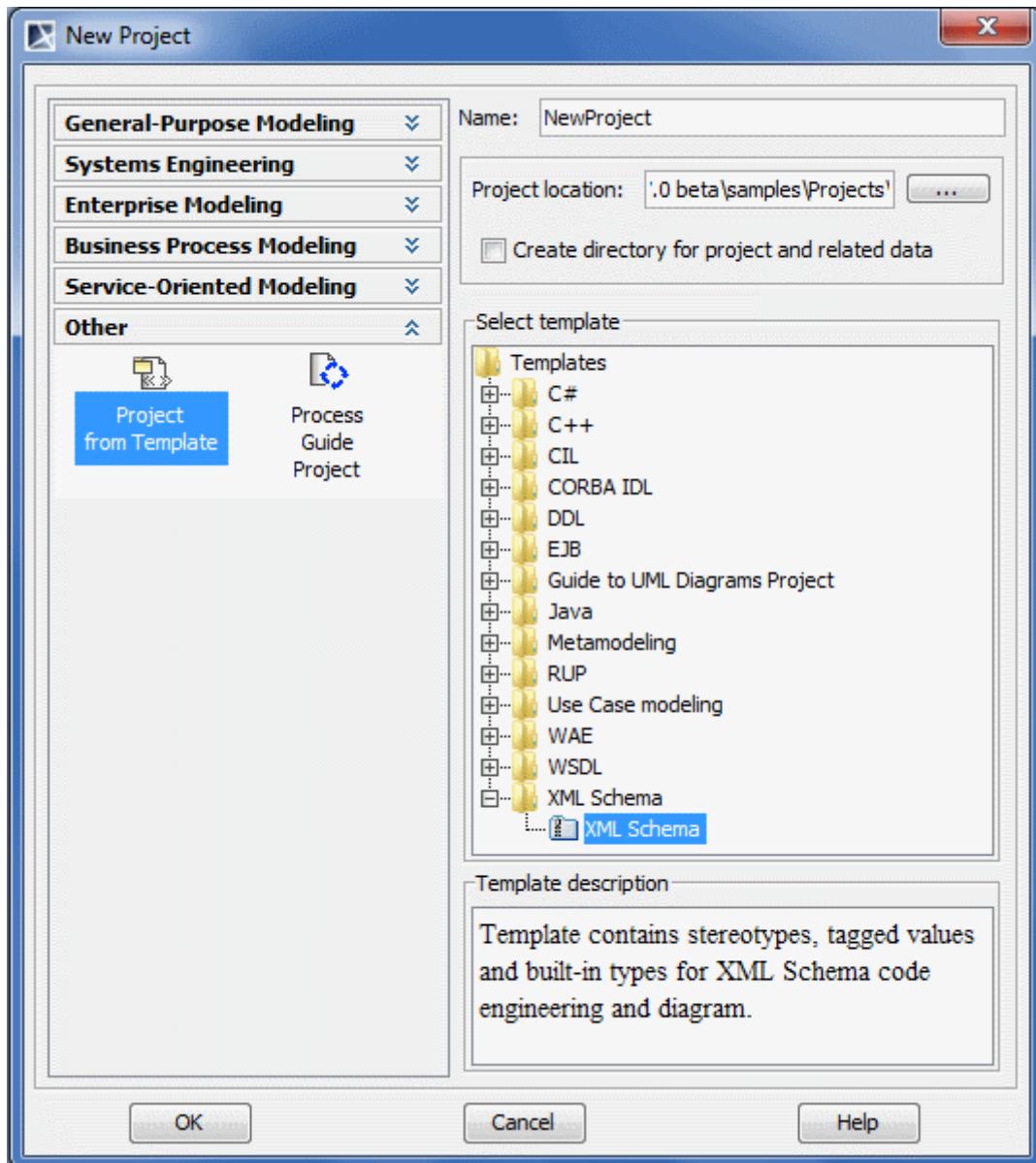
If you want to share the default properties with other users for their new project, change the property values in the **Project Options** dialog box and then create a project template, which other users may use:

1. From the **File** main menu select the **Export > Template...** command and save the project as a template. A template will be created in the <MagicDraw installation directory>/templates directory.
2. To import the created template to a new project from the **File** main menu, select the **New Project** command. The **New Project** dialog box opens. Select the **Project from Template** icon from

6 WORKING WITH MODEL ELEMENTS

Formatting Symbols

the **Other** domain and in the **Select template** tree, select your template. The project options are imported to the project together with the template.



Parent topic: "Diagramming", on page 167.

Related topics:

"Project Options" on page 132.

Formatting Symbols

In MagicDraw, every symbol may have its own style: color, font, size, and so forth. There are several ways to define the symbol properties:

- From the symbol properties dialog box: from the selected symbol shortcut menu or from the **Edit** menu, select **Symbols** and then **Symbol(s) Properties**.

6 WORKING WITH MODEL ELEMENTS

Formatting Symbols

- From the **Project Options** dialog box: this dialog box enables you to change all available symbol properties, create your own style for the project, apply different symbol properties for different diagrams, define stereotype properties that may be bound to the symbol, and more.
- From the MagicDraw main toolbar: using this toolbar, you may change the color, font, and path style of a symbol.

For more information about symbol properties, see “Project Options” on page 132.

To set the symbol properties (fill color, text color, and others)

- From the symbol shortcut menu, select **Symbol(s) Properties**.
- From the **Options** menu, select **Project**. The **Project Options** dialog box opens. Select the desired options for the selected model elements.
- Select a symbol and from the **Edit** menu, select **Symbol**, and then select **Symbol(s) Properties**.

NOTE

To apply changes made in the **Project Options** dialog box, click **Apply**.

Changing symbol properties dialog for multiple symbols

Symbol properties dialog may be invoked now when multiple symbols are selected in the diagram: select multiple symbols, invoke context menu and choose command **Symbol(s) Properties**.

Common selected symbols properties are displayed by default. If you wish to see all properties, turn on **Show All Properties** button that is located above properties table.

Note: Multiple symbol properties also may be changed using quick properties panel at the bottom of the Browser.

To show/hide model element constraints, stereotype and/or tagged values on the diagram pane

- Select/clear the **Show Constraints**, **Show Stereotype**, and/or **Show Tagged Values** check boxes from the symbol shortcut menu.
1. From the **Options** menu, select **Project**.
 2. The **Project Options** dialog box opens. Select the **Show Constraints**, **Show Stereotype**, and/or **Show Tagged Values** check boxes for the selected model elements.

NOTE

To apply changes made in the **Project Options** dialog box, click the **Apply** button in the **Styles** tab.

Displaying icon or image

You may display icon or image on the symbol or instead the symbol.

6 WORKING WITH MODEL ELEMENTS

Formatting Symbols

Get acquainted with the following definitions in order to understand this functionality better.

| Definition | Description |
|--------------------------|--|
| Symbol | The term "symbol" means a visual representation of some model elements in the diagram. Symbols are subdivided into shapes and paths (lines in the model, for displaying various relationships). |
| Symbol properties | Every symbol may have its own style: color, font, size, and so forth. Symbol properties may be defined for the concrete symbol, for all symbol of one element, or according to the diagram type. For more information about symbol properties definition, see "Formatting Symbols" on page 255, about style engine, see "Style Engine" on page 258. |
| Stereotype | "A stereotype defines how an existing metaclass may be extended, and enables the use of platform or domain specific terminology or notation in place of, or in addition to, the ones used for the extended metaclass." [The OMG UML specification (UML 2.3: Superstructure)]. For more information about stereotype, see "Stereotype" on page 624. For more information about applying stereotype properties, see "Stereotype properties" on page 268. |
| Icon | Icon of stereotype. Icon is a small image displayed in the top-right corner of shape. To assign icon to stereotype in the Stereotype specification dialog box, define the Icon property. For more information on how to assign Icon for stereotype, see "To create a stereotype with an image" on page 624. |
| Text | Stereotype name, displayed on the symbol. |
| Image | Image which can be assigned to element and displayed as icon or instead of element shape. To assign image to element in the element specification dialog box, assign the Image property. For more information on how to assign icon for element, see "Image" on page 244. |

To change the icon visibility mode on the element shape:

1. From the element shape shortcut menu, select the **Presentation Options > Show Stereotypes** command and then select the desired property mode.
2. You may change the stereotype/icon visibility mode in the symbol **Properties** dialog box > **Show Stereotypes** combo box.

Select one of the six property modes for **Show Stereotypes**. The property modes are described in the table below:

| Show Stereotypes Property Mode | Displayed on the diagram pane | Icon of the stereotype and stereotype name are assigned to element | Image to element is assigned |
|--------------------------------|-------------------------------|--|---|
| Icon and Text | | Icon of stereotype displayed. Name of stereotype displayed. | Image of element is displayed in the corner of shape. |
| Icon | | Icon of stereotype displayed in the corner of symbol. Name of stereotype is not displayed. | Image of element is displayed in the corner of shape. |

6 WORKING WITH MODEL ELEMENTS

Style Engine

| Show Stereotypes Property Mode | Displayed on the diagram pane | Icon of the stereotype and stereotype name are assigned to element | Image to element is assigned |
|--------------------------------|--|--|---|
| Text Only |  | Icon of stereotype is not displayed. Name of stereotype is displayed. | Image of element is not displayed. |
| Shape Image and Text* |  | Icon of stereotype is displayed instead of shape. Name of stereotype is displayed. | Image of element is displayed instead of shape. |
| Shape Image* |  | Icon of stereotype is displayed instead of shape. Name of stereotype is not displayed. | Image of element is displayed instead of the shape. |
| Do Not Display |  | Icon of stereotype is not displayed. Name of stereotype is not displayed. | Image of element is not displayed. |

* - To display the icon of a stereotype instead of the element shape all element compartments should be suppressed.

TIP! If element has assigned both - image and stereotype icon - then image of element will be displayed on the shape.

NOTE **Shape Image and Text** and **Shape Image** properties are not added to the Path element properties list.

Style Engine

The Style engine is a part of the MagicDraw UML system that defines diagrams, shapes, paths, and stereotype properties. There may be few property styles defined, but all symbols are created according to the style that is selected as default. There is a possibility to apply different presentation styles for diagram/shape/path/stereotype depending on the diagram type.

Symbol Property Styles Tree

Expands the tree hierarchy of all the styles defined within the project. You may use as many of these styles as you wish.

Shape and **Path** trees have the inner structure to help you find the model element, the representation of which must be changed. The right side of the dialog box contains possible choices and instruments to manage them.

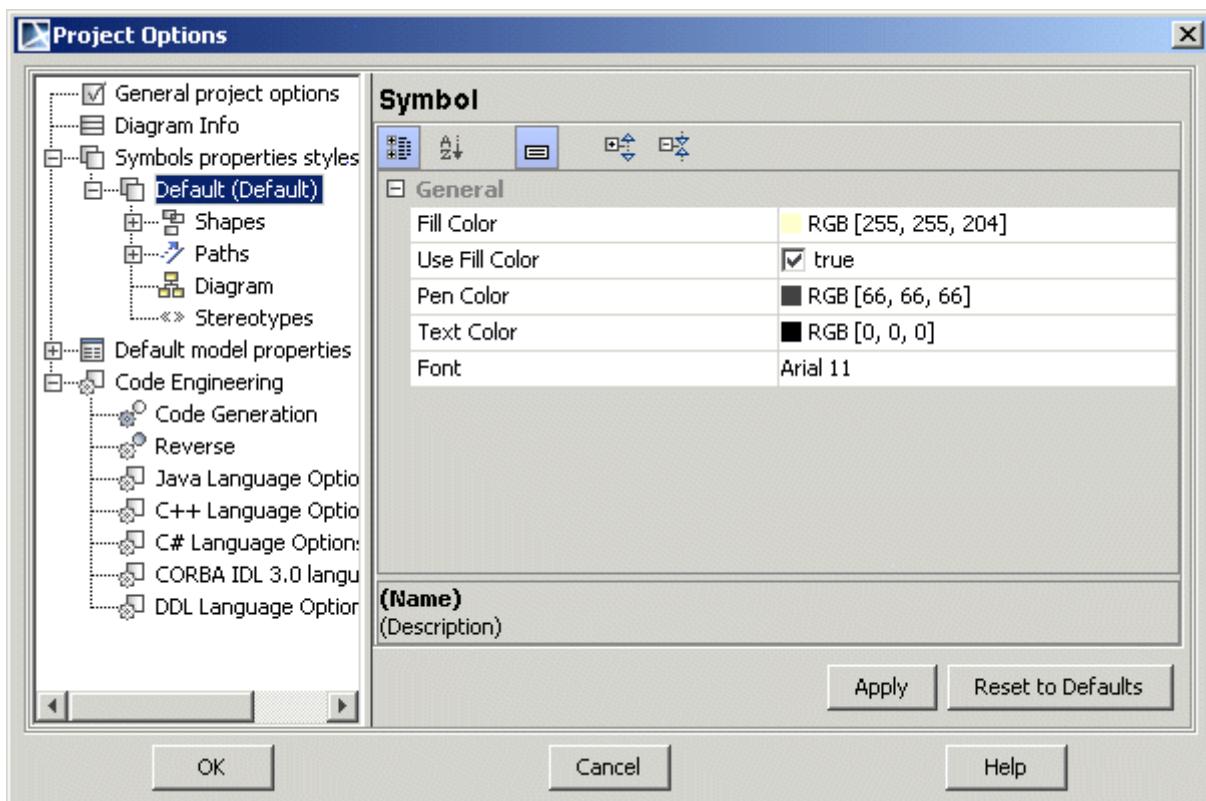


Figure 139 -- Project Options dialog box. Styles

The following buttons are available in the **Project Options** dialog box, the **Symbols properties styles** branch:

| Button name | Function |
|--------------------------|--|
| Clone | Clone (duplicate) the selected style. |
| Rename | Change the name of all existing styles. Type a new name for a style in the Enter Style Name dialog box. |
| Delete | Remove the selected style. |
| Make Default | Make the selected style the default style for newly created projects. |
| Apply | Apply the selected style to the open project. |
| Import | The “ Open dialog box” opens. Select the style you wish to import (*.stl). |
| Export | The “ Save dialog box” opens. Select the directory where you wish to export a style. |
| Reset to Defaults | Change all settings to the default configuration. |

To create a new style by cloning the existing one

1. Select the default style in the **Styles** list box and click the **Clone** button.
2. Type a name for the new style in the **Enter Style Name** dialog box.
3. Change options of the new style.

To change the name of the selected style

1. Select a style you want to rename and click the **Rename** button.
2. Type a new name for the style in the **Enter Style Name** dialog box.

6 WORKING WITH MODEL ELEMENTS

Style Engine

To remove the selected style

Click the **Delete** button in the **Project Options** dialog box.

To make a selected style your default style for newly created projects

Click the **Make Default** button in the **Project Options** dialog box.

To apply the selected style or changed option to a current project

- Click the **Apply** button in the **Project Options** dialog box, **Styles** pane.

NOTE You can also apply the desired options to the selected diagram model elements. Click the **Apply** button in the specific elements pane.

To import an already created (and exported) project style

Click the **Import** button. The **Open** dialog box opens. Select the style you wish to import (*.stl).

To save the created style (export) for a later usage or for other users

Click the **Export** button. The **Save** dialog box opens. Select the directory where you wish to export a style.

In the following table you will find all possible options that can be set for the symbols:

| Property (alphabetically sorted) | Function (when selected) |
|-----------------------------------|---|
| Attributes Color | The color of the attribute name. The Color dialog box opens. |
| Attributes Font | The font that is used for the name of an attribute. The Font dialog box opens. |
| Attributes Sort Mode | The mode for sorting attributes. Possible choices: No Sorting , By Name , By Stereotype , or By Visibility . |
| Autosize | Adjusts the size of a symbol to the contained information. Element borders are changed so that it uses minimum space. |
| Background Color | The color of the diagram background. Click the “...” button to open the Color dialog box, select the background color. |
| Constraint Text Mode | Displays constraint name or expression on a symbol. |
| Direction | The direction of a signal. Possible choices: Right or Left . |
| Enable Lolipop notation | If the Enable Lolipop notation check box is selected, notation of the interface becomes as "lolipop". NOTE The Enable Lolipop notation property is included only for the interface symbol properties |
| Enumeration Literals Color | The color of the enumeration literal name. The Color dialog box opens. |
| Enumeration Literals Font | The font that is used for the name of an enumeration literal. The Font dialog box opens. |
| Extension Points Color | The color of the extension point name. The Color dialog box opens. |
| Extension Points Font | The font that is used for the name of an extension point. The Font dialog box opens. |

| Property (alphabetically sorted) | Function (when selected) |
|------------------------------------|--|
| Fill Color | The fill color of the symbol. The Color dialog box opens. |
| Font | The font that is used for the name and other displayed properties of a model element. The Font dialog box opens. |
| Grid Size | Grid size settings from 2 to 30. |
| Header in Bold | Shows the name of the symbol in bold. |
| Header Position | The package name position on the symbol. Possible choices: Top or In Tab . |
| HTML Text | Activates the HTML editor for the text of a note and a text box. |
| Line Style | A line style for a horizontal separator. Possible choices: Dashed or Solid . |
| Operations Color | The color of the operation name. The Color dialog box opens. |
| Operations Font | The font that is used for the name of an operation. The Font dialog box opens. |
| Operations Sort Mode | The mode for sorting operations. Possible choices: No Sorting , By Name , By Stereotype , or By Visibility . |
| Orientation | Primarily the synchronization bar diagram button is set to the vertical or horizontal position. |
| Path Style | The drawing style of a path. Possible choices: Rectilinear , Oblique , or Bezier . |
| Pen Color | The pen color that is used to draw elements. The Color dialog box opens. |
| Show Attributes Constraints | Shows constraints of attributes. |
| Show Attributes Properties | Shows tagged values of attributes. |
| Show Attributes Stereotypes | Shows stereotypes of attributes. |
| Show Attributes Visibility | Shows attribute visibility signs (+, -, #, ~). |
| Show Base Classes | Shows a base class on the stereotype symbol. |
| Show Classifier | Shows a classifier name near the model element name. |
| Show Constraints | Shows constraints on symbols. |
| Show Diagram Info | Shows diagram information table on the diagram pane. |
| Show Direction Arrow | Shows the Direction Arrow on the association. Default Direction Arrow direction is displayed according path creation direction. It helps to read diagram and explain diagram semantics. For more information about Direction Arrow, see “To show the direction arrow near the association name” on page 652. |
| Show Elements List | Shows model elements that are assigned to a model, package, or subsystem as a list. |
| Show Entire Activation | Shows the entire activation bar on an active classifier role in a sequence diagram. |
| Show Full Classifier Type | Shows all attributes that are defined within a class or assigned classifier. |
| Show Grid | Shows a grid on the diagram. |

| Property (alphabetically sorted) | Function (when selected) |
|--|--|
| Show Initial Attribute Value | Shows the initial attribute value on a class or artifact. |
| Show Message Numbers | Shows the message numbers on a diagram. |
| Show More Sign for Attributes | Shows an additional information sign “...” in the class, artifact attributes list, when omissions are made by editing the class compartment. |
| Show More Sign for Operations | Shows an additional information sign “...” in the class, artifact or enumeration operations list, when omissions are made by editing the class, artifact, or enumeration compartment. |
| Show More Sign for Signal Receptions | Shows additional information sign “...” in the signal reception list, when omissions are made by editing the signal reception compartment. |
| Show Multiplicity | Shows the multiplicity value. |
| Show Name | Shows the name of a relationship, role and message/stimulus. |
| Show Operation Parameters Direction Kind | Shows the direction kind for operation parameters |
| Show Operations Constraints | Shows constraints of operations. |
| Show Operations Properties | Shows tagged values and concurrency of an operation. |
| Show Operations Signature | Shows all of the operation arguments and the return type. |
| Show Operations Stereotypes | Shows stereotypes of operations. |
| Show Operations Visibility | Shows operation visibility signs (+, -, ~ #). |
| Show Owner | Changes the display position of qualified name on the element shape. |
| Show Predecessors | Shows predecessors on the message. |
| Show Qualified Name for Operation | <p>The Show Qualified Name for Operation property shows the operation name and the class of the operation on the activity shape (using style <i>ClassName::OperationName</i>). The default property value is true.</p> <div data-bbox="793 1538 1102 1628" style="border: 1px solid #ccc; padding: 5px; width: fit-content; margin-left: auto; margin-right: 0;"> Get supplementary order (OrdersDB::getOrder) </div> |
| | <p>NOTE For projects that were created before MagicDraw version 15.0, the Show Qualified Name for Operation property is unchecked.</p> <p>For more information about displaying qualified name for operation, see “Call Operation Action” on page 643.</p> |
| Show Signal Receptions Constraints | Shows constraints of signal reception. |
| Show Signal Receptions parameter Direction Kind | Shows the direction kind for signal reception parameters. |
| Show Signal Receptions Properties | Shows tagged values and concurrency of properties. |

6 WORKING WITH MODEL ELEMENTS

Style Engine

| Property (alphabetically sorted) | Function (when selected) |
|---|--|
| Show Signal Receptions Signature | Shows the parameter and related information. |
| Show Signal Receptions Stereotypes | Shows stereotypes of signal receptions. |
| Show Signal Receptions Visibility | Shows signal reception visibility signs (+, -, ~ #). |
| Show Stereotypes | If you want to change a stereotype and its icon visibility on the element shape, use the Show Stereotypes property. For more information about stereotype display mode, see “Changing the stereotype display mode” on page 628. |
| Show Tagged Values | Shows tagged values on symbols. |
| Show Visibility | Shows role visibility signs (+, -, #). |
| Signal Receptions Color | The color of the signal reception name. The Color dialog box opens. |
| Signal Receptions Font | The font that is used for the name of a signal reception. The Font dialog box opens. |
| Signal Receptions Sort Mode | The mode for sorting signal receptions. Possible choices: No Sorting , By Signal Name , By Stereotype , or By Visibility . |
| Snap Paths to Grid | Snap paths to grid. |
| Snap Shapes to Grid | Snap shapes to grid. |
| Stereotype Color | The color that will be used to draw stereotypes. The Color dialog box opens. |
| Stereotype Font | The font that will be used to draw stereotypes. The Font dialog box opens. |
| Suppress Actions | Hides actions associated with the state. |
| Suppress Attributes | Hides the attribute list. |
| Suppress Enumeration Literals | Hides enumeration literals on a enumeration. |
| Suppress Extension Points | Hides use-case extensions on a use case. |
| Suppress Operations | Hides operations compartment section. |
| Suppress Realization Elements | Hides realization elements of a subsystem. |
| Suppress Signal Receptions | Hides attributes list from the shape. |
| Suppress Specification Elements | Hides specification elements of a subsystem. |
| Text Color | The color that is used for text coloring. The Color dialog box opens. |
| Text Position | Changes the text position of a separator. Possible choices: Center , Left , or Right . |
| Use Fill Color | Uses the selected fill color and the symbols color changes on the diagram. |
| Use Fixed Connection Points | The end of the path is connected to the fixed point of the shape. |

| Property (alphabetically sorted) | Function (when selected) |
|--------------------------------------|--|
| Use Advanced Members Coloring | Use different colors for text fragments of stereotypes, names, types, and other members. |
| Wrap Words | Wrap words to a new line when text exceeds the text box width. |

Working with Properties Styles

All symbols in MagicDraw are created according to active properties styles. There may be more than one property style in the same project, and the whole style may be applied for the project.

Every style has its own presentation of Diagram, Shape, Path, and Stereotype that you can modify using the **Project Options** dialog box in the **Symbols Properties Styles** branch. You can set your own options for every model element to the current style.

Path, Shape, and Stereotype branches have the inner structure that helps you find the model element, the representation of which must be changed. The section on the right side of the dialog box contains possible choices and instruments to manage them.

The following properties are defined for the formatting symbols:

- **Shapes.** Set general options for the shapes in the right pane of the **Project Options** dialog box. You can set options for all shapes that appear on the Diagram pane.
- **Paths.** Set general options for the paths in the right pane of the **Project Options** dialog box. You can set options for all paths that appear on the Diagram pane.
- **Diagram.** Set general options about a diagram.
- **Stereotypes.** Set general options for the stereotypes in the right pane of the **Project Options** dialog box. You can set options for all stereotypes that may be applied to elements on the Diagram pane.

Changing properties for multiple element

To change properties for multiple symbols, using Ctrl or Shift key select few elements in the **Project Options** dialog box, **Symbol properties styles** branch.

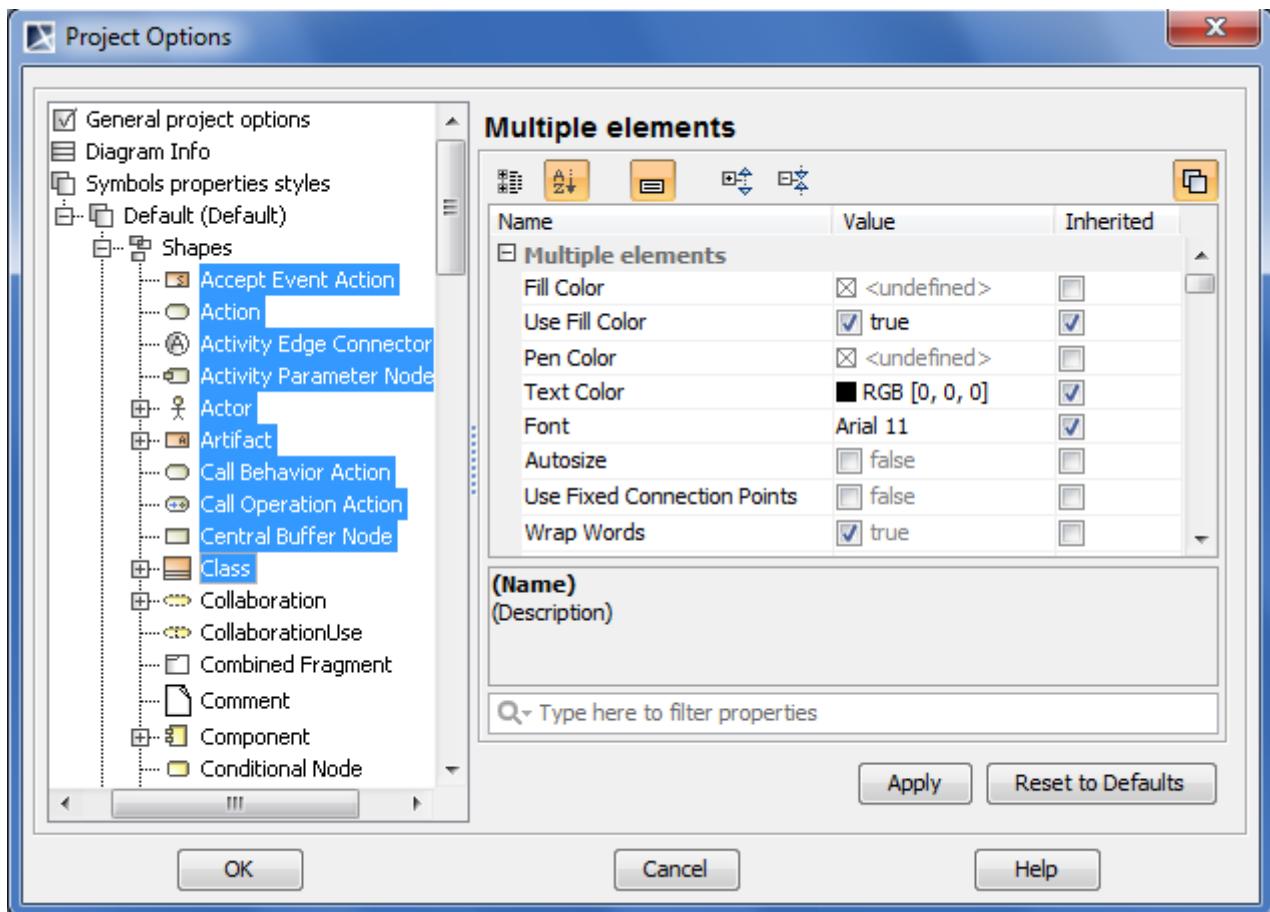


Figure 140 -- The Project Options dialog box, multiple element symbols style is selected

Properties extension by diagram

Diagram, shape, path, and stereotype properties can be extended by the particular diagram type. This means that presentation style options will be applied only for the specified element symbol in the specific diagram.

To extend the element properties by diagram

1. In the **Project Options** dialog box, the **Symbols Properties Styles** tree, expand a branch, select the specific element (shape, path, diagram, or stereotype) and right click the mouse button. The list of diagrams in which the element symbol may be created, opens.

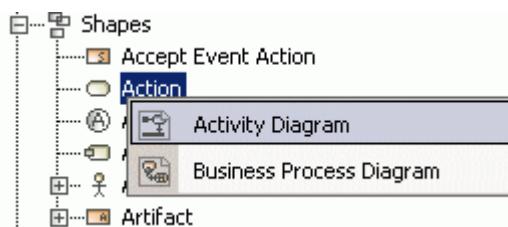


Figure 1 -- Element extension by diagram

6 WORKING WITH MODEL ELEMENTS

Style Engine

2. Select the diagram type. The Diagram is added as an additional branch to the section.

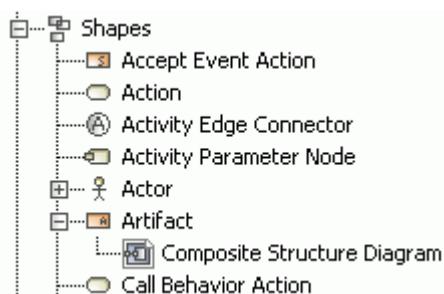


Figure 2 -- Extended diagram addition

3. Set the style properties for the element in the right pane of the **Project Options** dialog box. The properties will be applied only in the specified type of diagram.
- The element can be extended by diagram in the **Project Options** dialog box, specific elements pane, by clicking the **Extend by Diagram** button. The **Extend by Diagram** dialog box opens. Click the **Add Diagram** button and select a diagram from the list.

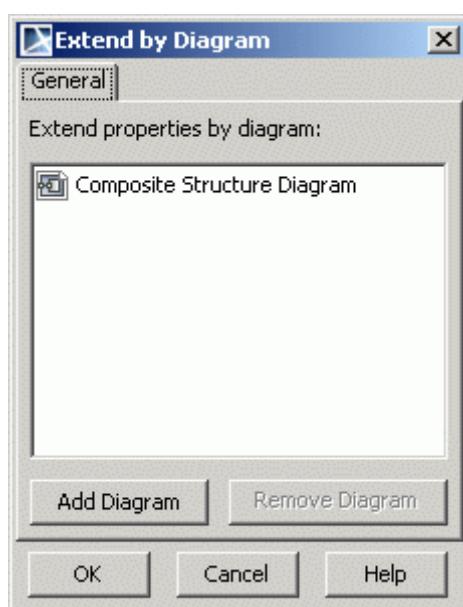


Figure 3 -- Extend by Diagram dialog box

To remove the extended diagram from the tree

- Select the extended diagram and right click on the mouse, then select **Remove**.

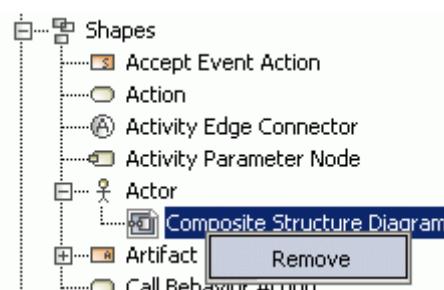


Figure 4 -- Remove extended diagram

- In the **Project Options** dialog box, the extended diagram style properties pane, click the **Remove** button.

Properties Inheritance

All element properties have the “inherited” check box. This check box indicates the property is derived from the base element properties or is it specific.

The **Inherited** column check box value in the elements properties pane specifies if the current property is synchronized with its parent property. When the **Inherited** value is “true”, the element property is changed after changing the parent property.

If the property has no correspondent property in the upper (parent) level, the **Inherited** column check box will be cleared and disabled.

If the property is modified for the specific element and the value differs from the upper level current property value, the **Inherited** column check box is cleared automatically.

General Style Properties

You can define the common properties for the whole style. The style properties are displayed when the properties style is selected in the **Project Options** dialog box styles tree.

Shape, Path and Diagram Properties

All shapes, paths and diagrams that can be created in the project, are listed in the **Project Options** dialog box. If the **Shape**, **Path**, or **Diagram** branches are selected in the tree, the general properties can be set in the right pane of this dialog box.

When expanding any of these branches, the style for a concrete element (diagram) can be created.

To apply a new style to a previously created element symbol

1. In the **Project Options** dialog box, change the element style properties and click the **Apply** button. The **Select Diagrams** dialog box opens. The list of diagrams created in the project is displayed.

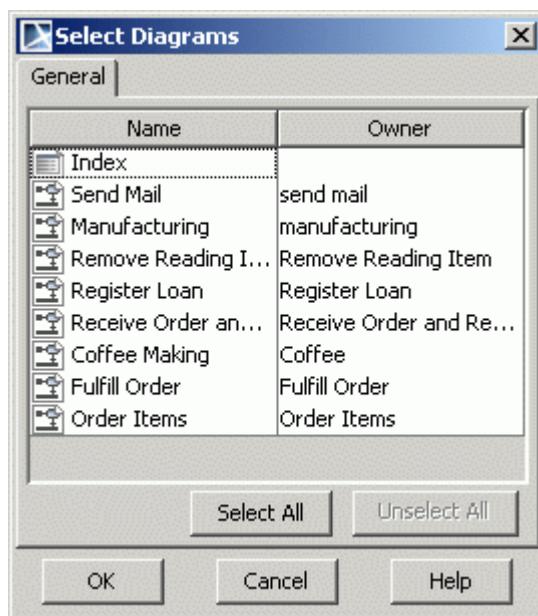


Figure 5 -- Select Diagrams dialog box

2. Select the diagrams to which the element properties will be applied and click **OK**. The **Select Properties to Apply** dialog box opens.

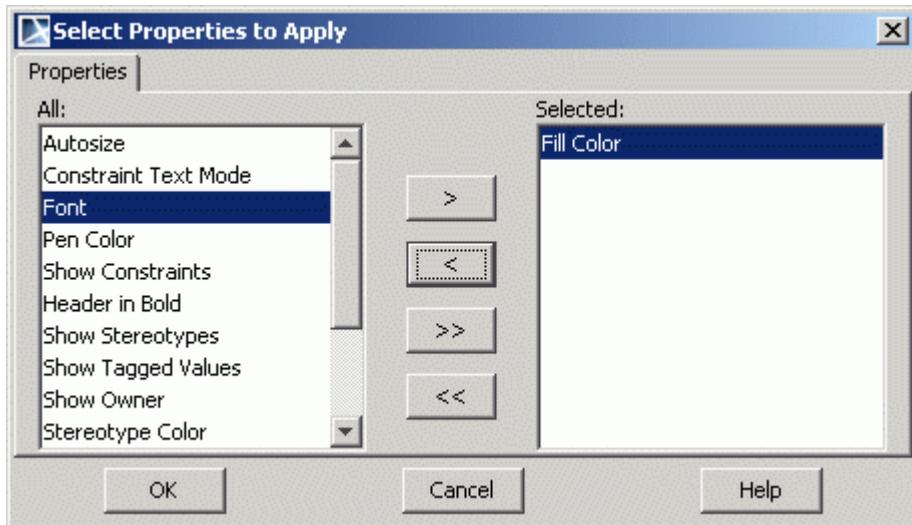


Figure 6 -- Select Properties to Apply dialog box

3. Select the properties to be applied to the element symbol by moving them from the **All** list to the **Selected** list. Click **OK**.

NOTE!

If a new style was set, it will be applied for all newly created elements after drawing them on the diagram pane. You can always set the default symbols style to the element by clicking the **Apply Default Symbol Style** button in the main toolbar.

Stereotype properties

The Stereotype properties can be applied only if the stereotype properties style is created in the **Project Options** dialog box.

The Stereotype properties are derived from their base class. The Stereotype base class is defined in the label of the right pane of the **Project Options** dialog box.

The same element can have several stereotypes assigned. In this case, the style of the first stereotype will be applied to the element symbol. If the stereotype is removed from the element, the next (first) stereotype properties are applied. If the last stereotype is removed from the element, the base class (shape or path) properties are applied to the element symbol.

Stereotypes may be extended by diagram.

All stereotypes that have defined symbol properties are included in the **Stereotypes** branch. By default only *boundary*, *control*, and *entity* stereotypes are added to the tree when expanding the **Stereotypes** branch. The default style is created for these stereotypes.

6 WORKING WITH MODEL ELEMENTS

Style Engine

To add a stereotype to the branch

1. In the **Project Options** dialog box, the **Symbols Properties Styles** tree, right-click the **Stereotypes** branch. The list of stereotypes opens.

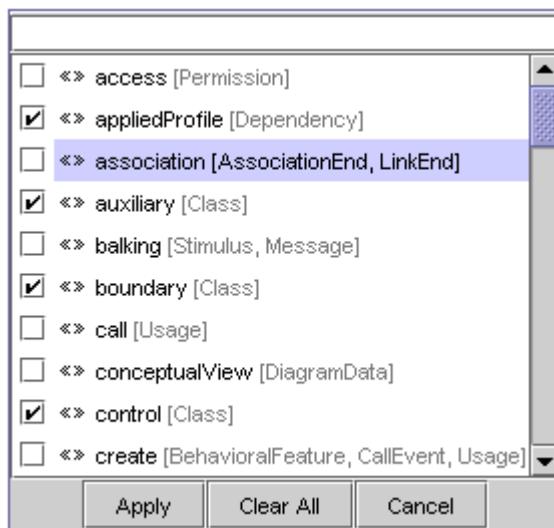


Figure 7 -- List of stereotypes

2. Select the check box near the stereotype and click the **Apply** button. The stereotype will be included into the **Stereotypes** branch. Set the stereotype style properties in the right pane of the **Project Options** dialog box.

To remove a stereotype from the branch

1. In the **Project Options** dialog box, the **Symbols Properties Styles** tree, select the **Stereotypes** branch. The list of stereotypes opens.
2. Clear the check box near the stereotype and click **Apply**. The stereotype is removed from the branch.

To change stereotype properties

1. Expand the **Stereotypes** branch and select a stereotype.
2. Set properties in the right pane of the **Project Options** dialog box.

To apply the stereotype properties to a previously created symbol with assigned stereotype

1. When the stereotype style properties in the right pane of the **Project Options** dialog box are changed, click the **Apply** button. The **Select Diagrams** dialog box opens.
2. Select the diagrams to which the stereotype properties will be applied and click **OK**.
3. In the **Select Properties to Apply** dialog box, select which properties will be applied to the stereotype. Click **OK**.

NOTE!

You can apply stereotype properties to a symbol after changing the style properties and in the **Project Options** dialog box, click **OK**. The style will be applied when selecting symbol on the diagram pane and clicking the **Apply Default Symbol Style** button on the main toolbar.

To apply the stereotype properties when assigning a stereotype to an element

NOTE!

In the **Environment Options** dialog box, **Diagrams** tab, the **Apply Stereotype Style for All Symbols** check box should be selected and in the **Project Options** dialog box, the style properties should be changed for stereotype.

1. In the created diagram, draw an element.

2. From the element shortcut menu, select **Stereotype**. The list of available stereotypes opens.
3. Select the check box near the stereotype you want to assign to the element. Click **Apply**. The stereotype properties are applied automatically when assigning the stereotype to the element.

Defining Hyperlinks Between Elements

You can set text for notes, text boxes, or separators as HTML text. You can also hyperlink to any model element, diagram, external file, or requirement.

Adding a hyperlink to the model element

There are three ways to add a hyperlink to the model element: from diagram **Smart Manipulator**, from **Specification** and from the Browser tree.

To add a hyperlink from the diagram

1. Select the element and click on the Smart Manipulator for Hyperlink.

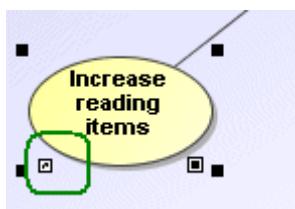


Figure 8 -- Smart manipulator for hyperlinks.

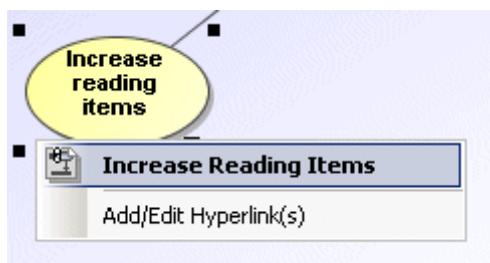


Figure 9 -- Hyperlinks menu.

2. The hyperlink menu opens, listing:

- Previously created hyperlinks with icons corresponding to element type, diagram type, external file, or requirement.
- Menu item for adding and editing existing hyperlinks - **Add/Edit Hyperlink(s)**
- If there are no hyperlinks yet defined, only the menu item to add a hyperlink will be in the hyperlink menu.

6 WORKING WITH MODEL ELEMENTS

Defining Hyperlinks Between Elements

3. Click **Add/Edit Hyperlink(s)** item. The hyperlinks creation editing dialog opens.

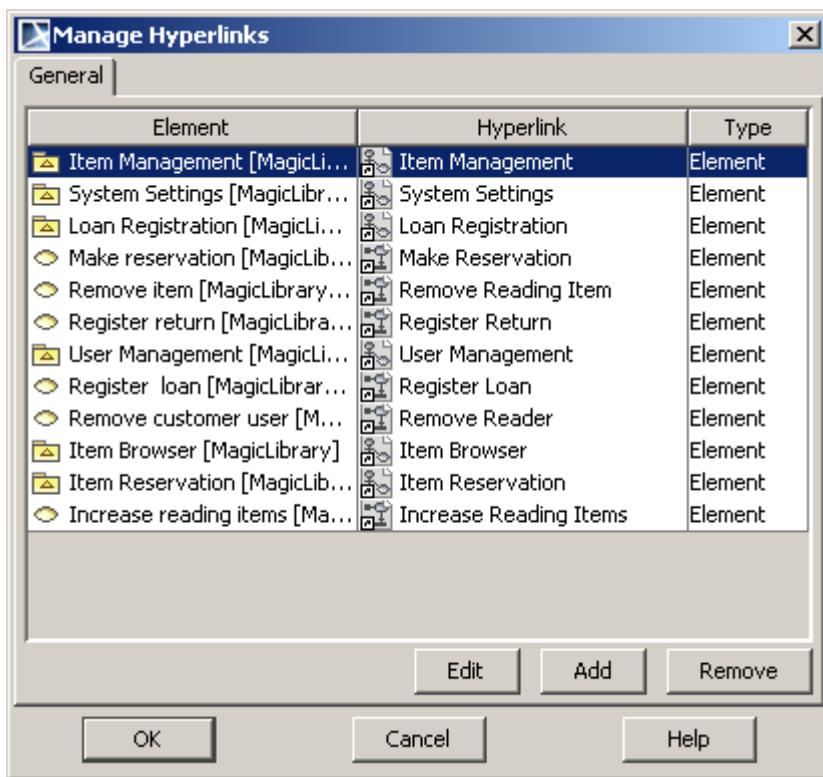


Figure 141 -- Manage Hyperlinks dialog

4. Click **Add** and define the hyperlink to any model element, file, or Web page in the **Edit Hyperlink** dialog box. If you want this hyperlink to be active, select the **Active** check box.
5. Click **OK**.

To add a hyperlink from Specification:

1. Open the model element **Specification** dialog box and select the **Documentation/Hyperlinks** tab.
2. Define the hyperlink to any model element, file, or Web page in the **Edit Hyperlink** dialog box. If you want this hyperlink to be active, select the **Active** check box.
3. Click **OK**.

To add a hyperlink from the Browser tree

Now hyperlinks can be created and edited straight from the element shortcut menu:

- Invoke element shortcut menu from the Browser.
- Choose **Go To** and **Hyperlinks** (see Figure 142 on page 272).

For more detailed description on managing hyperlinks, see “To add a hyperlink from the diagram” on page 270.

6 WORKING WITH MODEL ELEMENTS

Defining Hyperlinks Between Elements

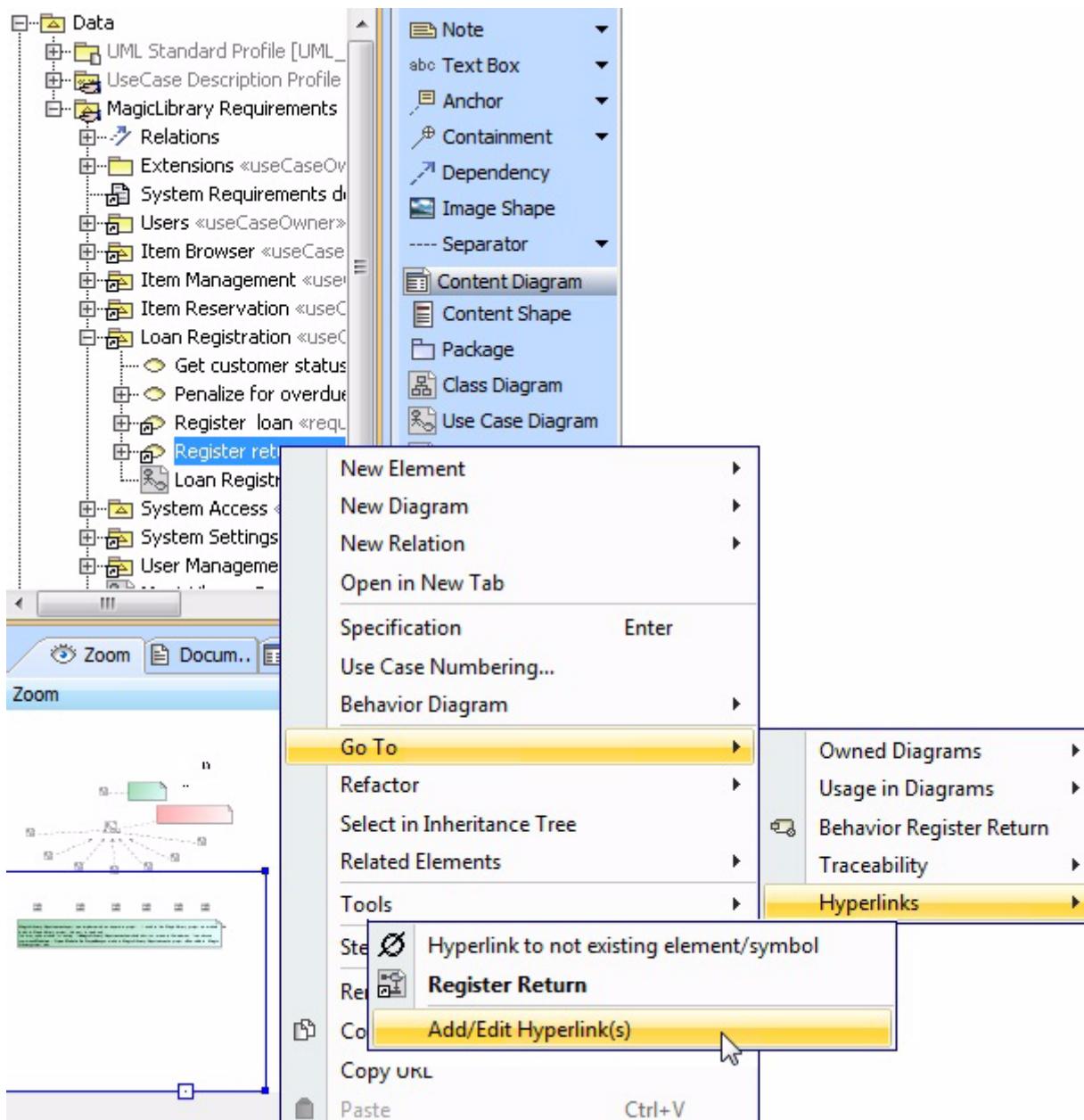


Figure 142 -- Hyperlinks creation from the Browser tree

To add a hyperlink to a note, text box, or separator text

1. Select the text where you want to add a hyperlink and click **Insert Hyperlink** .
2. The **Edit Hyperlink** dialog box opens. Select the hyperlink you want to insert, either to a Web page, another model element, or a file:
 - To link to an existing model element, click the **Element/Symbol** tab. Click the **Select Element/Symbol ...** button and select the model element or symbol you want to link to in the element Selection dialog. More information about this dialog find in the Section “Selecting an Element” on page 278.

6 WORKING WITH MODEL ELEMENTS

Defining Hyperlinks Between Elements

- To link to an existing Web page, click the **Web Page** tab, and in the **Type the Web page name** text box, type the URL of the Web page you want to link to. You can click the “...” button and browse the Web for the desired Web page.

NOTE

Set the path of the HTML viewer in the **Environment Options** dialog box (for a description, see “Customizing Environment Options” on page 89.)

- To link to an existing file, click the **File** tab and enter the path to the file you want to link to. Or, click the **Type the file name** “...” button and, in the **Open** dialog box, select the file you want to link to.

NOTES:

- The selected file opens in the HTML browser.
- Set the path of the HTML viewer in the **Environment Options** dialog box.
- You can only link to an existing file. New files are not created for you.

Using the HTML editor toolbar, you can change the font, color, size, and the alignment of the selected text.

Edit Hyperlink dialog box

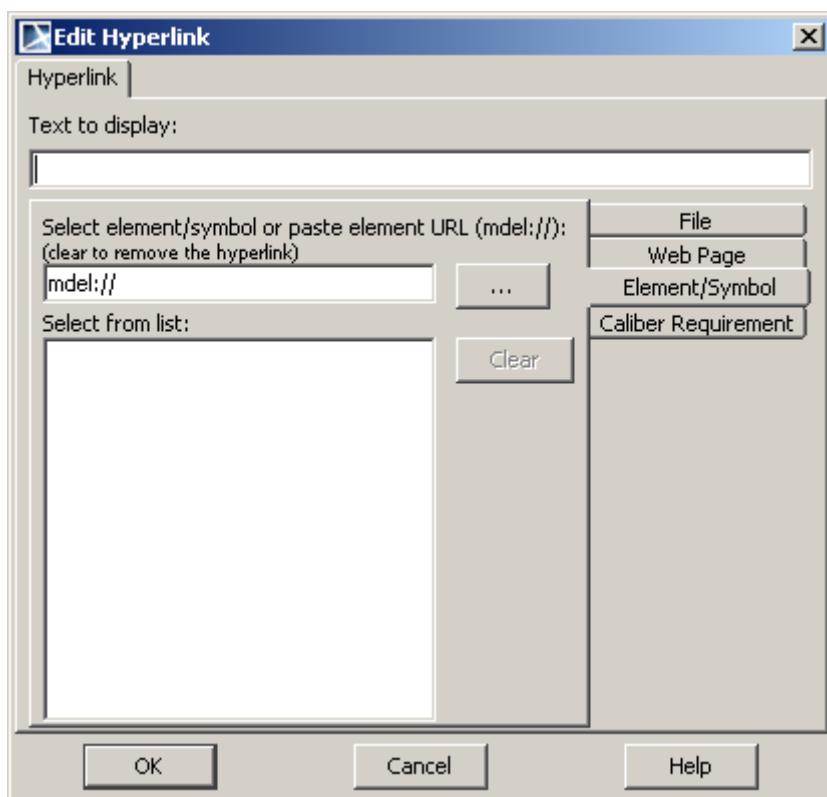


Figure 143 -- The Edit Hyperlink dialog box

| Tab name | Box | Function |
|--|---|--|
| Element/ Symbol Creates a hyperlink that goes to the selected model element. | Text to display | A text that will be displayed as a hyperlink. |
| | Select Element / Symbol or paste element URL "..." | The element Selection dialog opens. Select the model element you want to link to. You can also paste URL to element. For more information about element URL, see “Copying/Opening Element URLs” on page 291. |
| | Select from list | A list of all items that have previously been selected as links. |
| | Clear | Remove all items from the Select from list . |
| Web Page Creates a hyperlink that goes to the specified Web page. | Active | If selected, activates the hyperlink on the diagram. Only one hyperlink can be active. Target referenced by the active hyperlink is accessed after double clicking an element with a hyperlink. By default the last added hyperlink is the active one. |
| | Text to display | The text that will be displayed as a hyperlink. |
| | Type the Web Page name "..." | Type the web page URL. Click the “...” button. The Web browser window opens. Browse the Web and find the web page you want to link to. NOTE Make sure that the path of the HTML viewer is set in the Environment Options dialog box. |
| | Select from list | A list of all items that have previously been selected as links. |
| File Creates a hyperlink that goes to a specified file. | Clear | Remove all items from the Select from list . |
| | Text to display | A text that will be displayed as a hyperlink. |
| | Type the file name "..." | Type the path to the file you want to be opened or click the “...” button. The Open dialog box opens. Select the file you want to link to. |
| | Select from list | A list of items that have previously been selected as links. |
| | Clear | Remove all items from the Select from list . |

Owner of the Model Element

Model elements and diagrams belong to a package, model (system boundary), subsystem or other appropriated model element, which is called *owner*.

The name of the owner is displayed in the model element name compartment in parentheses.

6 WORKING WITH MODEL ELEMENTS

Owner of the Model Element

To add a model element to a package, model (system boundary), or subsystem

- Drag a model element to the desired package on the Diagram pane or in the Browser tree.
- Open the **Inner Elements** tab, located in the **Package**, **Model**, or **Subsystem Specification** dialog box. Click **Add** and select a model element or diagram you want to add to a package. Define a model element or diagram in the open **Specification** dialog box and click **OK**.
- From the selected owner shortcut menu in the Browser tree, select **New Element**. From the list, select the desired model element and type its name in the Browser.

To display/hide the package/system boundary/subsystem name (the owner of an actor) on a model element

- From the symbol shortcut menu, select **Symbol(s) Properties**. The **Properties** dialog box opens. Select/clear the **Show Owner** check box.
- From the **Options** menu, select **Project**. The **Project Options** dialog box opens. Select the desired model element and select/clear the **Show Owner** check box. If you want to apply changes for previously created model elements, click **Apply**.

TIP!

For a class, actor, or interface, you may display/hide the name of the owner from the symbol shortcut menu: select **Presentation Options**, and then select/clear the **Show Owner** check box.

Owner display mode

MagicDraw version 15.0 and later has improved the owner display functionality. Now you can change the display position of the qualified name on the element shape.

To change the qualified name position:

- From the element shape shortcut menu, select the **Presentation Options > Show Owner** command and then select the desired property mode.
- You can change the qualified name position in the symbol **Properties** dialog box, **Show Owner** drop-down list.

Select one of the four property modes for **Show Owner**. The property modes are described in the table below.

| Show Owner Property Mode | Shape | Description | Notation |
|--------------------------|-------|---|--|
| Do Not Display | | Only element name is displayed on the element shape. This is the default value. | - |
| Below Element Name | | Owner is displayed below the element name. | This is MagicDraw style notation. The owner name is constructed from the names of the containing namespaces starting at the root of the hierarchy and ending with the owner of the NamedElement itself. Containing namespaces are separated by dot and owner is displayed in brackets. |

6 WORKING WITH MODEL ELEMENTS

Owner of the Model Element

| Show Owner Property Mode | Shape | Description | Notation |
|--------------------------|---|--|--|
| In Same Line With Name |  | Element owner is displayed in the same line as the element name. | This is a notation from UML specification. The qualified name is constructed from the names of the containing namespaces, starting at the root of the hierarchy and ending with the name of the NamedElement itself. Containing namespaces are separated by double colons. The double colon is shown to separate containing namespaces and element name. |
| Above Element Name |  | Owner is displayed above the element name. | Notation is the same as In Same Line With Name option notation. |

Qualified name starting from model library

MagicDraw version 15.0 and above includes an option to show the owner hierarchy starting from the model library as the root.

Model Library is a package with modelLibrary stereotype.

This option is called **Qualified name display style**. To change its value:

1. From the **Options** menu select the **Project** command. The **Project Option** dialog box opens.
2. Select the **General Project Options** branch. In the right side pane, you can modify the option property.

The **Qualified name display style** property is added to the **Project Options** dialog box, **General Project Options** branch.

If the **Model Library Relative** property value is selected (default value for a new project), then the full qualified name hierarchy is displayed on the shape, starting from the model library as a root. The model library itself is not displayed. The **Qualified name display style** property allows for having the relative path for library items used in the project.

Relations Changes Ownership when Client or Supplier is Moved to Other Owner

Some issues related to relationships have been addressed to improve usability.

Now relationships will not get lost in Containment tree while changing the element ownership. Relationships will also be moved together with the client or supplier (or both) so that all the related elements can be grouped together in one place. This will also prevent unexpected dependencies on model partitioning.

6 WORKING WITH MODEL ELEMENTS

Owner of the Model Element

For example, if you move two Classes which are connected to the Association relationship to another Package in the Containment tree, a question dialog will open, asking if you want to move the relationship as well (Figure 144 on page 277).

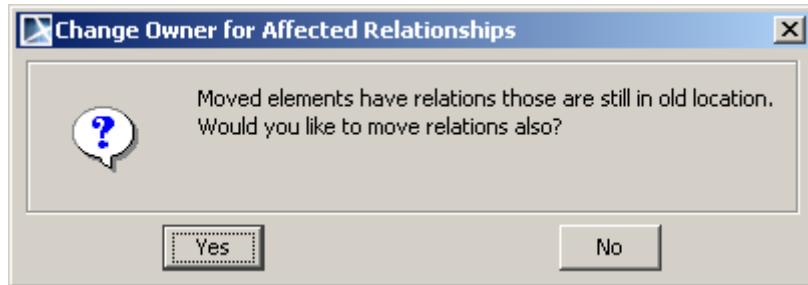


Figure 144 -- The Change Owner for Affected Relationships message

Selecting an Element

The element Selection dialog is used for selecting elements from the available contents and adding them to a certain destination scope. The title of the dialog varies depending on the way it is invoked. Depending on the content, the element Selection dialog is used for selecting an element, diagram, or owner.

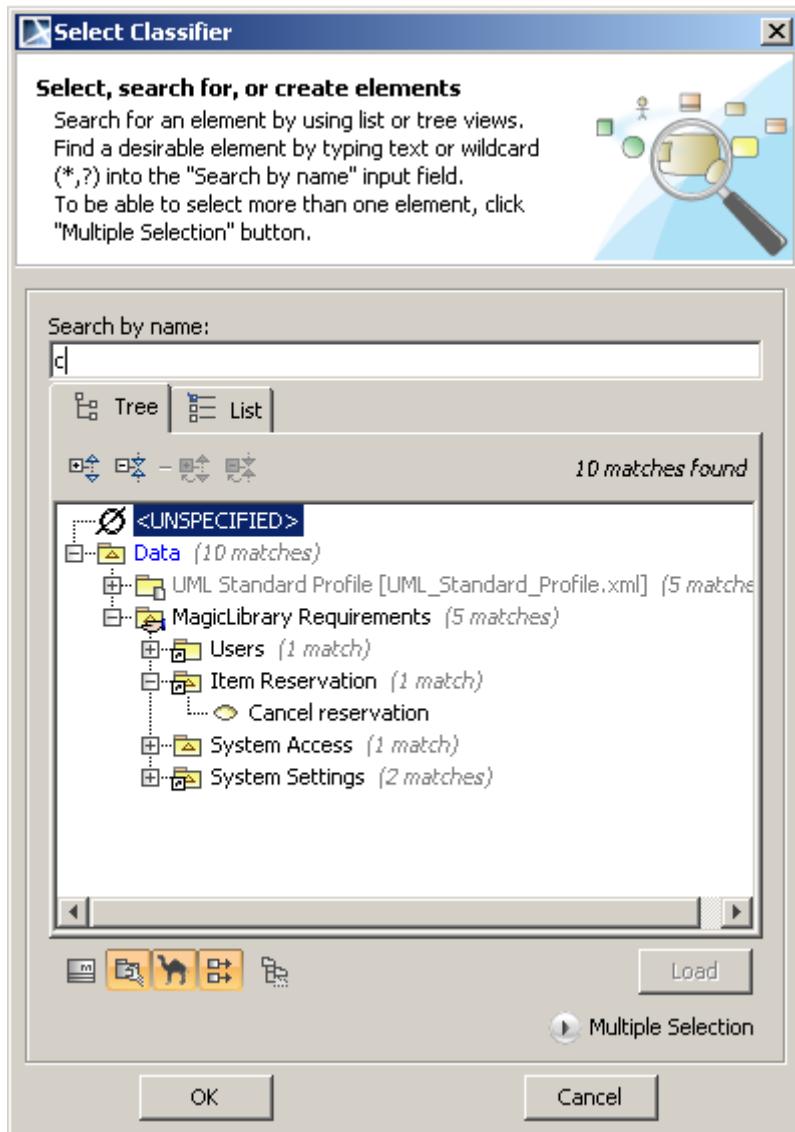


Figure 145 -- Element Selection dialog

You can perform the following actions in the element Selection dialog:

- Search for an element in List or Tree views.
- Search for an element using various search modes.
- Select single or multiple elements.
- Create an element.

Element selection views

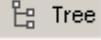
There are two different views for elements selection:

- **List** view displays the elements listed in the alphabetical order.

6 WORKING WITH MODEL ELEMENTS

Selecting an Element

- **Tree** view displays the hierarchical structure of the elements and enables to create a new one.

| Element selection view | Description |
|---|---|
|  List List view | The List view displays all the items that can be selected in a particular case. Elements are sorted in the alphabetical ascending order. The first 30 list items are selectable elements. Other items can be seen after clicking the click here to show the remaining matches at the end of the list. In the event there are over 10000 selectable elements, the list will not sort and display the elements. |
|  Tree Tree view | The Tree view displays all selectable items and their owners. The following commands helps to manipulate in the Tree view:  <ul style="list-style-type: none">• Expand All / Collapse All - all nodes are expanded/collapsed recursively.• Expand selected recursively / Collapse selected recursively - selected node is expanded/collapsed recursively.• Load button - click the Load button to load the selected module which is not loaded. |
| Library view | Available only in UPDM. More information you may find in the “ UPDM Plugin UserGuide.pdf ”, chapter “Common Features”, section “Library Support”. |

Element search criteria and modes

When searching for an element, which you want to select, type the search criteria in the **Search by name** field.

| Search type | Description |
|--|--|
| Simple search in the List and Tree views | In the Search by name text field, type the first letters of the required element name. In the List and Tree views, only the elements matching the search criteria will be displayed. TIP! When switching between the element selection views, the selected elements are remembered e.g., if the element is selected in the Tree view, it will be selected in the List view as well. |
| Search using wild cards | In the Search by name text field, the wildcard symbol (*) substitutes any range of any symbols and the wildcard symbol (?) substitutes any symbol. |

There are four search modes available to help to find the required element in the **Tree** or **List** views.

| Search mode | Description |
|--|--|
|  Search includes meta-classes | If the Search includes meta-classes mode is selected, the available search contents will include the meta-classes that match the search criteria. |

6 WORKING WITH MODEL ELEMENTS

Selecting an Element

| Search mode | Description |
|---|---|
|  | If Search includes elements from modules mode is selected, the elements from the modules will be included in the search contents. |
|  | If the Search uses camel case mode is selected, when searching for the elements, whose names are written in camel case, type only the upper case letters in the type Search by name text field. Lower letters between the upper ones will be skipped when filtering. |
|  | If the Search includes qualified names mode is selected, you can search for matches in all the qualified names of the selectable elements. The search works as a simple search, except if wildcards are used in front. In the event a wildcard is used in front, the output will be all the elements that have the fragment matching search criteria in the qualified names. |

Element creation mode

In the **Tree** view, you can create new elements that can be owned by the selected element in the tree. Element can be created by choosing an element from the list or by cloning the selected element in the tree.

To activate the element creation mode



- Click the **Element creation mode** button . The **Create** and **Clone** buttons will appear.

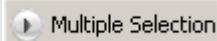
NOTE

The undo of the action done in the **Select Element** dialog will be active only after closing the dialog.

| Creation button | Description |
|-----------------|--|
| Clone | Click the Clone button if you want to create a new element based on the data of the selected element. The element will be cloned with all its internal structure and data. You may specify the new element in the Specification window. |
| Create | Click the Create button if you want to create a new element, whose owner is the selected element. If more than one element type can be created, choose an element type form the list. You can specify the new element in the Specification window. TIP! You may customize the list of appearing element types using DSL categorization. Further information about DSL customization can be found in the "UML Profiling and DSL UserGuide.pdf" |

Elements multiple selection

NOTE



The **Multiple Selection** button is available only when it is allowed to select more than one element.

To select more than one element

1. Click the **Multiple Selection** button. The block for the multiple selection in the element Selection dialog will open.
2. Select one or more elements at a time by holding the Control or Shift keys.

To return to the single selection



- Click the **Single Selection** button. The first element that is in the multiple selection list will be selected in the **Tree** view.

NOTE

In the event more than one element is selected in the tree or in the list in the multi selection mode, the selection will be reduced to one element (the first one based on alphabetical order), when switching to the single selection mode.

The following buttons are available in the Multiple selection mode:

| | |
|------------------------|--|
| Add | Adds the selected element(s) to the elements multiple selection list. TIP! You can also perform this action by double-clicking an element. |
| Add Recursively | Adds all the elements owned by selected element and by the elements owned by the selected element and so forth, to the elements multiple selection list. |
| Remove | Removes selected elements from the elements multiple selection list. TIP! You can also perform this action by double-clicking an element. |
| Remove All | Removes all the elements form the elements multiple selection list. |
| Up | Moves up per one item the selected element in the elements multiple selection list. |
| Down | Moves down per one item the selected element in the elements multiple selection list. |

Refactoring

Converting an Element

Element conversion allows converting one element type to another. Sometimes, during the modeling process, there is a demand to change a class to a component or another type of classifier. The element conversion copies all compatible properties to a converted element (for example Ports of a class will become Ports of a component). If some properties are not compatible, they will be lost.

Element conversion functionality allows UML element conversion from one meta-class to another.

To convert an element

From the element shortcut menu, select **Refactor** and then **Convert To** and then select the element from the open list, to which you want to convert.

When an element is being converted, the converter finds all usages of this element and recreates these references to a new element after conversion. For example, if an instance specification has an element assigned as a classifier, it remains after conversion.

The usages that are not valid after conversion, will be removed. For example, an Interface has an Interface Realization relationship. An interface is converted to a Class. The Interface Realization will be removed from model.

If the converted element and new element have the same symbol properties, then they are reassigned for the new element. For example, if a class is converted to an interface, and the class had the property **Suppress Operations** - *true*, then the Interface property for **Suppress Operations** will be *true* also.

Replacing an Element

You can replace one model element with another of the same metatype type element. Model element replacement is useful when during the modeling process you notice that one model element needs to be replaced with another. All relations and references to former element are updated to point to the newly selected model element.

After the replacement source will be replaced with replacement target:

- All references to replacement source will be replaced by references to replacement target.
- The replacement target will be displayed in all diagrams instead of replacement source.
- The replacement target after replacement will have all paths of replacement target and replacement source.
- Replacement source will be deleted.

To replace one element with another

1. In the element shortcut menu, select **Refactor** and then **Replace With**.
2. In the element Selection dialog, select the element with which you want to replace. More information about element Selection dialog, see section “Selecting an Element” on page 278.

See an example in the Figure 146 on page 282 and Figure 147 on page 283. In the Figure 146 on page 282 you can see two classes, which are similar to each other - *Customer* and *User*. Using Replacement functionality you can replace *User* class with *Customer* class quickly without redrawing relationships (see Figure 147 on page 283).

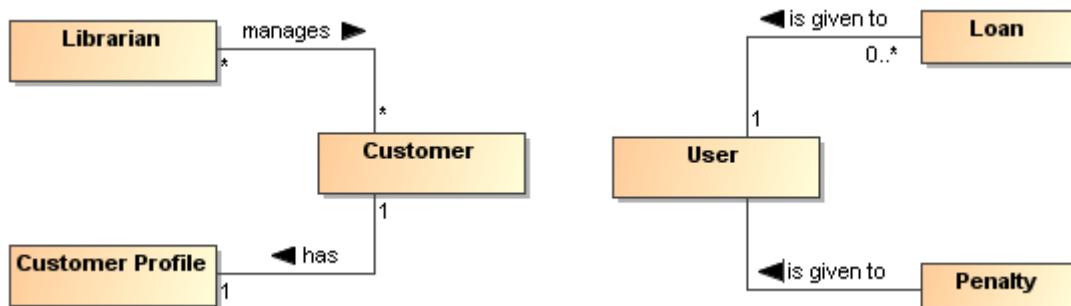


Figure 146 -- Model before the element replacement

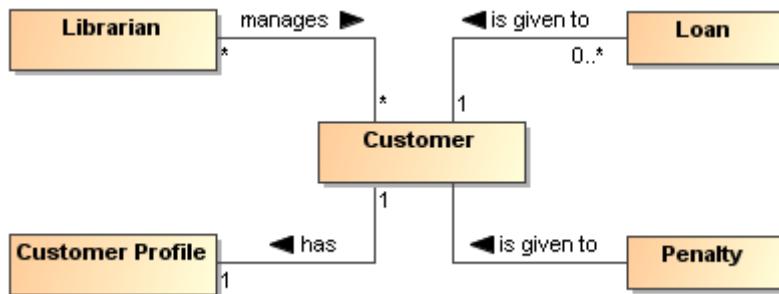


Figure 147 -- Model after the User class has been replaced with the Customer class

HTML Editor

MagicDraw has been enhanced with a new HTML editor to edit HTML text. The new editor improves text editing capability and usability and allows you to preserve the text format when copying formatted text.

To set the note / text box / separator text as HTML

- Select a shape and click Switch To HTML Text that appears on the lower-left corner of the shape.



- From the note / text box shortcut menu, select **HTML Text**.
- Draw the note or text box, using the **Note(HTML text)** or **Text Box(HTML text)** buttons on the diagram pallet.

Click the text area on the selected shape or start typing letters to open the HTML editor toolbar. For more information about the toolbar buttons, see Section ["HTML editor toolbar"](#) on page 289.



Figure 148 -- HTML editor toolbar

You can write HTML text in various dialogs. To start doing this, you have to turn on the HTML mode first.

To turn on the HTML mode in a dialog

- Click to select the **HTML** check box.

These are the samples of the **HTML** check box and HTML editor toolbar in various dialogs:

- Element Specification window, the **Documentation/Hyperlinks** tab (Figure 149 on page 284).
- In the Model Browser, the **Documentation** panel (Figure 150 on page 285).
- Element Specification window, the **Tags** tab, when editing a tagged value (Figure 151 on page 286).

- When specifying the **To Do** property in the element Specification window (Figure 152 on page 287).

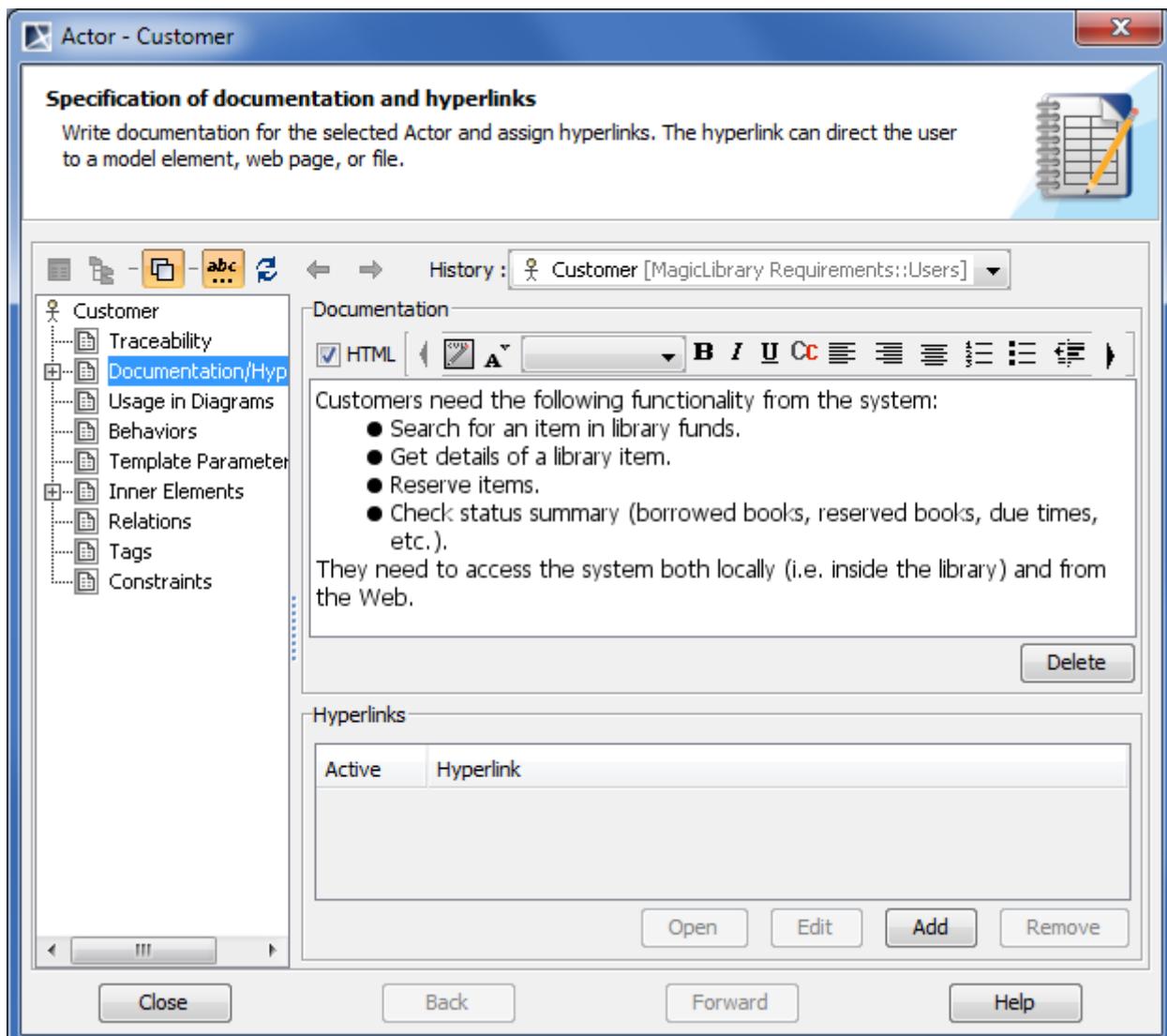


Figure 149 -- HTML editor toolbar in Documentation/Hyperlinks tab of element Specification window

6 WORKING WITH MODEL ELEMENTS

HTML Editor

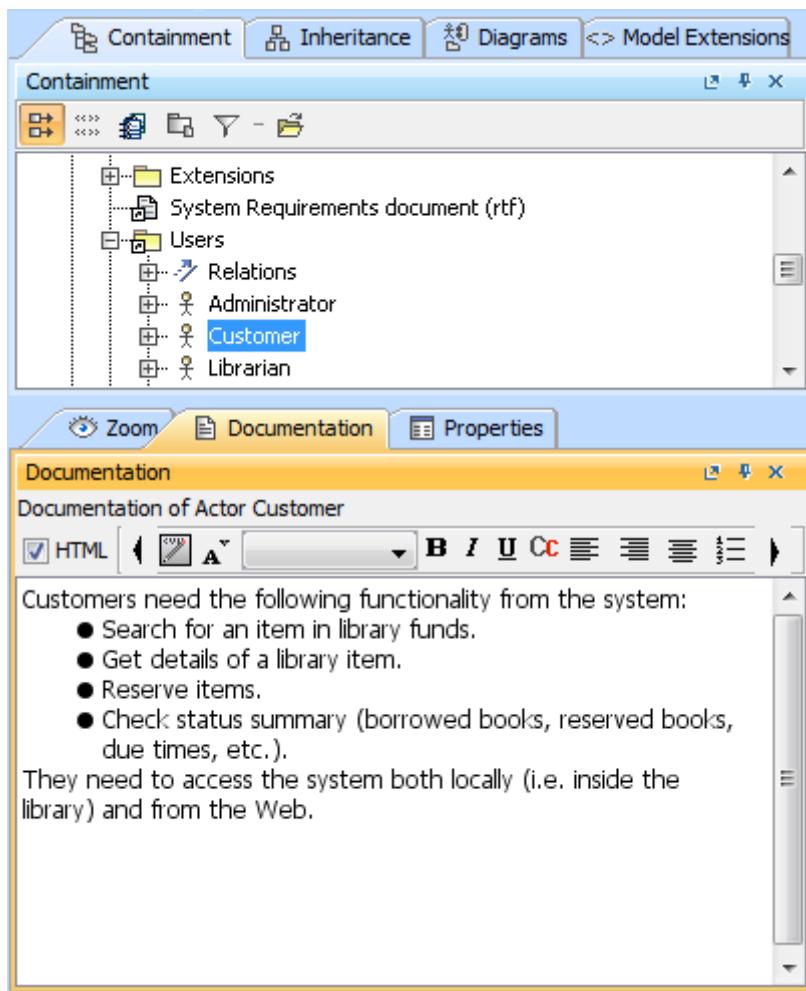


Figure 150 -- HTML editor toolbar in Documentation panel on Model Browser

6 WORKING WITH MODEL ELEMENTS

HTML Editor

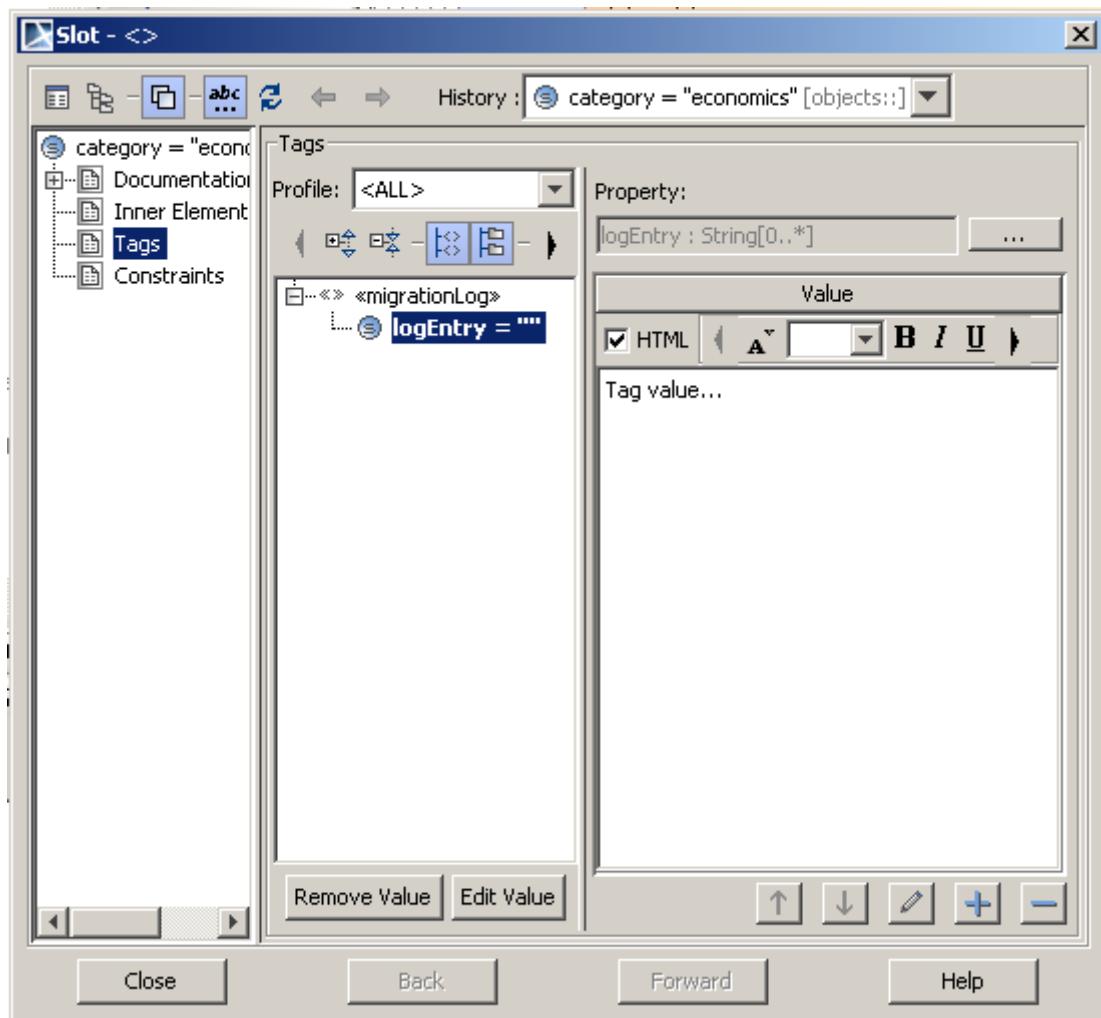


Figure 151 -- HTML editor toolbar in Tags tab of element Specification window when editing tag value

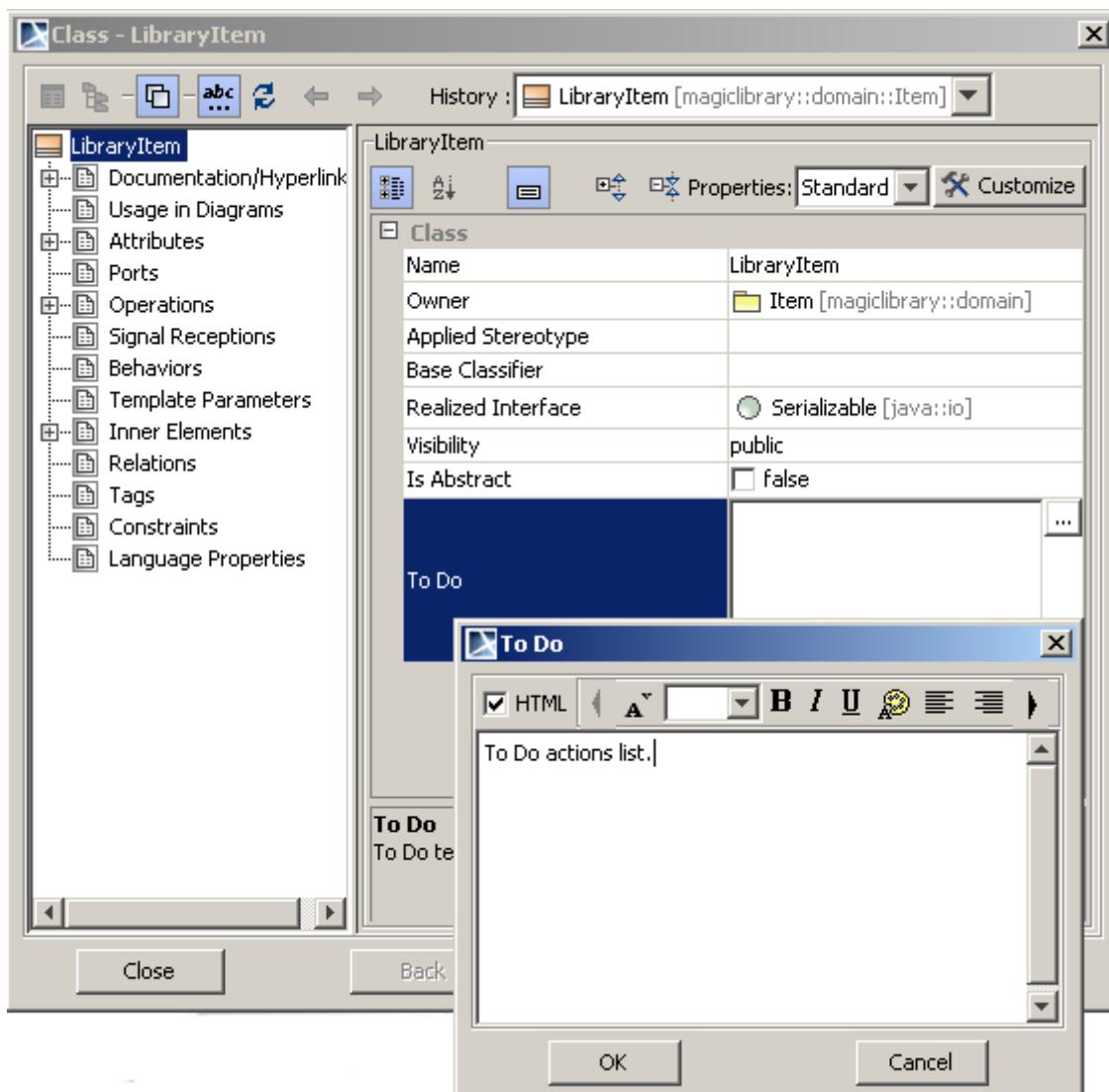


Figure 152 -- HTML editor toolbar in To Do dialog

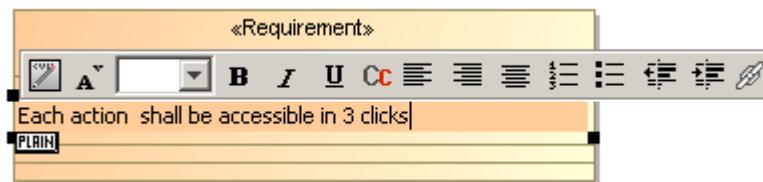
You can also use the HTML editor toolbar, when editing a tagged value directly on the element's shape.

To use the HTML editor toolbar for editing a tagged value

1. Click the tagged value on the element's shape as it is shown in the following picture.



2. Click the tagged value once again. The HTML mode will be turned on and the HTML editor toolbar will open for editing this tagged value.



HTML editor toolbar

| Button | Function |
|---|--|
|  Font properties | Select font style of the text. |
|  Font Size | Select font size of the text. |
|  Bold | Set text as bold. |
|  Italic | Set text as italic. |
|  Underline | Set text as underlined. |
|  Text Color | Select font color of the text. |
|  Align left | Align text to the left side border. |
|  Center | Center the selected text. |
|  Align Right | Align the selected text to the right side border. |
|  Insert Hyperlink | Add a hyperlink to the desired file, model element, or web page for the selected text. The Insert Hyperlink dialog box opens. |
|  Unordered list | Change text style to bullet list. |
|  Ordered list | Change text style to numbered list. |
|  Decrease Indent | Decrease indent by moving text closer to the left border. |
|  Increase Indent | Increase indent by moving text closer to the right border. |
|  Hyperlink | Add hyperlink to file, web page, element or symbol. The Edit Hyperlink dialog box opens. For more information, see Section " Adding a hyperlink to the model element " on page 270. |
|  Advanced HTML Editor | Edit text with advanced HTML editor. The Advanced HTML Editor dialog opens. |

Advanced HTML editorTo open the **Advanced HTML Editor** dialog

- Click the **Advanced HTML Editor** button on the HTML editor toolbar.

6 WORKING WITH MODEL ELEMENTS

HTML Editor

NOTE For the information how to turn on the HTML editor, see the procedure "[To turn on the HTML mode in a dialog](#)" on page 283.

In the **Advanced HTML Editor** dialog you can change the text style, insert symbols, images, and tables, as well as perform other actions using buttons.

Click the **HTML source** tab to view HTML source.

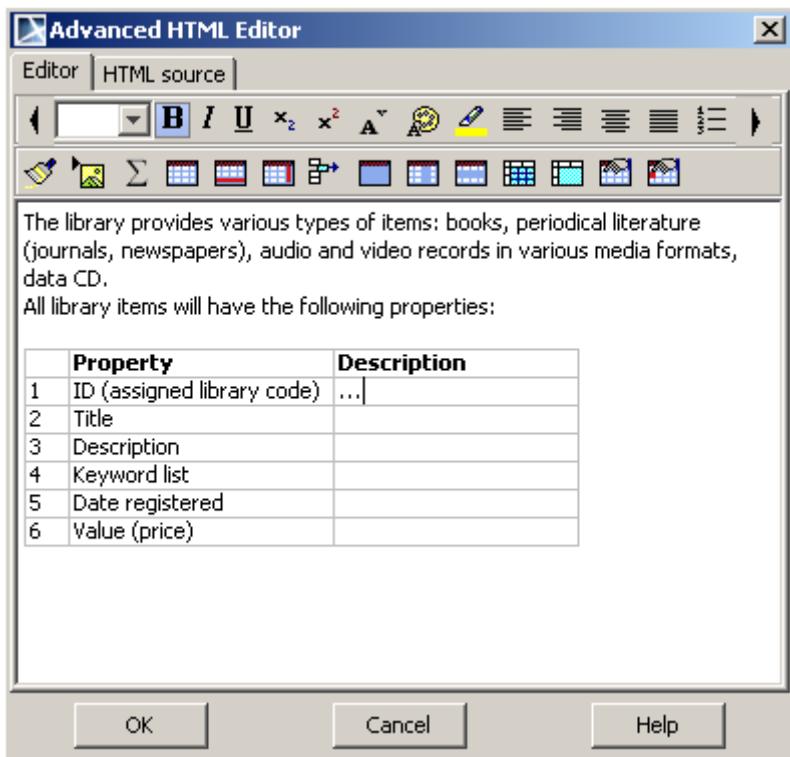
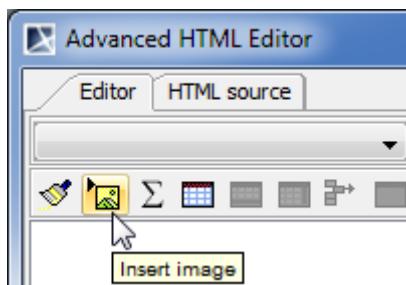


Figure 153 -- Advanced HTML Editor dialog

To insert an image into HTML text

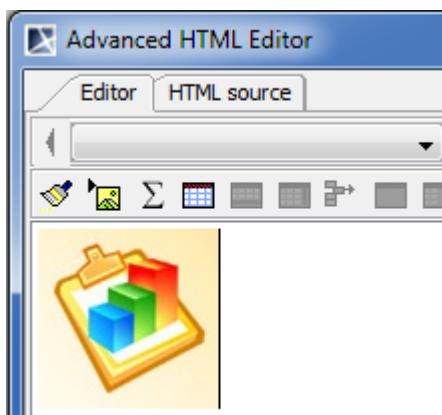
1. Click the Insert image button as it is shown in the following picture.



2. In the **Insert Image Location** dialog, specify the image URL. Do one of the following:
 - Type the path to the image location.
 - Click **Browse** and browse to the image you want to insert.

IMPORTANT! If you are working on a Teamwork project, make sure that the path to the image location is accessible from any computer with the Internet. Otherwise the image will not be displayed, when the project is opened on another computer.

3. Click **OK** when you are done. The image will be inserted into HTML text.



Copying/Opening Element URLs

You can now copy a project element URL to a clipboard and share it with other as a quick reference to model elements

To copy a project element URL, do any of the following

-
- Select **Copy Element URL** from the element shortcut menu in the Containment tree to copy the URL to a model element.
- or
- Select the element symbol in a diagram and click **Edit > Copy Element URL** on the main menu to copy the URL to element symbol.

You can open any elements through their URLs by clicking the **Open Element from URL** command and the element will be highlighted in the Containment tree or in the diagram. Custom URL "mdeI://" is registered into windows registry. Activating the URL in other applications will allow you to start MagicDraw, open the project (if possible), and select any elements. You can paste URLs from the clipboard to any MagicDraw diagrams. Hyperlinks also can hold URLs of any model elements.

7 TOOLS

MagicDraw provides the following tools and wizards to help you quickly and easily perform design tasks.

- "[Model Merge](#)" - Model Merge enables porting changes between different project versions.
- "[Pattern Wizard](#)" - creates various GOF, Java, Junit, CORBA IDL, XML Schema and others design patterns.
- "[Creating Setters / Getters](#)" – creates getting and setting operators for attributes defined in the class.
- "[Implementing or Overriding Operations](#)" – creates defined operations down the inheritance tree.
- "[Model Transformation Wizard](#)" - enables running one of the predefined transformations to convert the given model part into another model according to the rules of that transformation. Transformations are usually used for converting models between different modeling domains, e.g. UML and DDL.
- "[Resource Manager](#)" - MagicDraw Resource Manager functionality allows you to manage resources (Profiles, Plugins, Templates, Language resources, Case studies/examples, Custom diagrams, and others).
- "[Spelling Checker](#)" - Spell Checker will check spelling as you type. Select what you want to be spell checked (the whole project or some specific parts).
- "[Import Data to MagicDraw](#)" - Data Import to MagicDraw using RConverter and data import from other tools to MagicDraw.

Model Merge

[View Online Demo](#)

Model Merge

NOTE

Model Merge functionality is available in Standard Edition and above for an additional fee.

Definitions

| Name | Definition |
|-----------------------|---|
| Model Merge | Model Merge enables copying changes between different project versions. This functionality is usually needed when there are several branches that reflect different releases or versions of the product, e.g. when certain fixes have to be copied from a release branch to the mainstream development. |
| 2-way merge | 2-way merge is a simple merge, which compares two projects and joins them into one. |
| 3-way merge | 3-way merge does not only compare and merge two projects into one project, but it also considers the common ancestor of both projects. |
| Contributor | Participants of 2-way and 3-way merges are called contributors. |
| Ancestor | Ancestor is the common parent project for two projects. |
| Target, Source | In 3-way merge, the target is the contributor into which changes are copied, and the source is the contributor from which changes are copied. |
| Conflict | A conflict is two changes that are incompatible, i.e. changes that cannot be accepted together. For more information about conflicting changes, see “Conflicting changes” on page 317. |

Introduction to Merging

Model merge enables copying changes between different project versions. This functionality is usually needed when there are several branches that reflect different releases or versions of the product, e.g. when certain fixes have to be copied from a release branch to the mainstream development.

Merge functionality in MagicDraw works both on projects stored in MagicDraw Teamwork Server or files.

1. Select **Project Merge** from the **Tools** main menu. The **Merge Projects** dialog box opens.
2. Select the source, target, and ancestor projects (changes are going to be copied from the source to the target project). The source and target projects can be stored in the system file or the Teamwork Server. If both projects are stored in the Teamwork Server, the ancestor is determined automatically. For more information about the **Optimize for** option, see “Controlling Merge memory usage” on page 341.
3. The **Merge** window appears, to let the user accept or reject the changes and resolve the conflicts that occurred in both contributors (e.g. when the same class is edited in both contributors).

4. Confirm the changes made to the target.

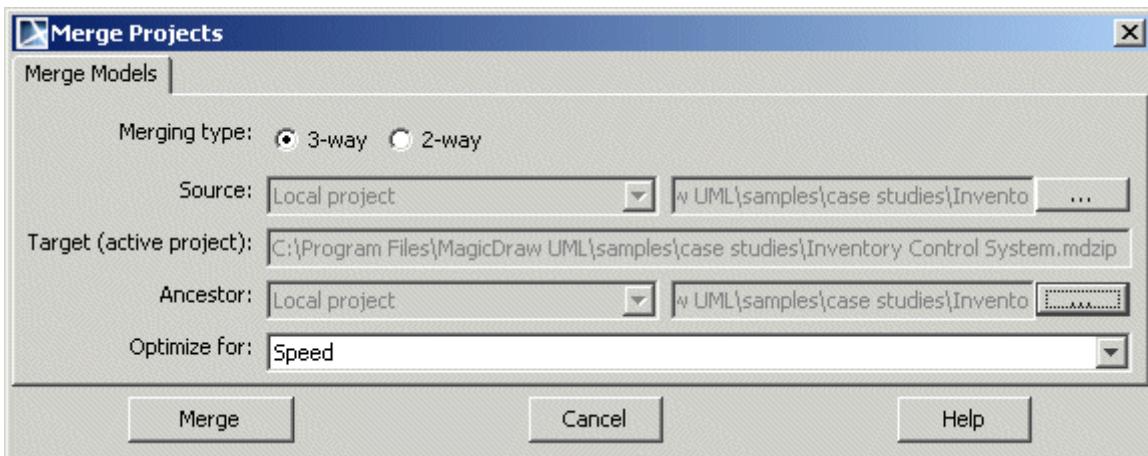


Figure 154 -- The **Merge Projects** dialog box

Two merge types are implemented:

- “3-way merge” on page 294.
- “2-way merge” on page 294.

3-way merge

Conceptually 3-way merge is a reconciliation of 2 difference sets. To merge projects v2 and v3, which have common ancestor v1, difference sets of projects v1-v2 and v1-v3 must be reconciled. 3-way merge can be used for merging changes from one branch into another.

See Figure 155 on page 294, where the merging changes from one branch into another is presented. Branch *b*.1 is created from the project version *i*. Team members work in parallel on project version *i* and make changes in project branch *b*.1. The merge is performed and changes made to branch are copied to the trunk.

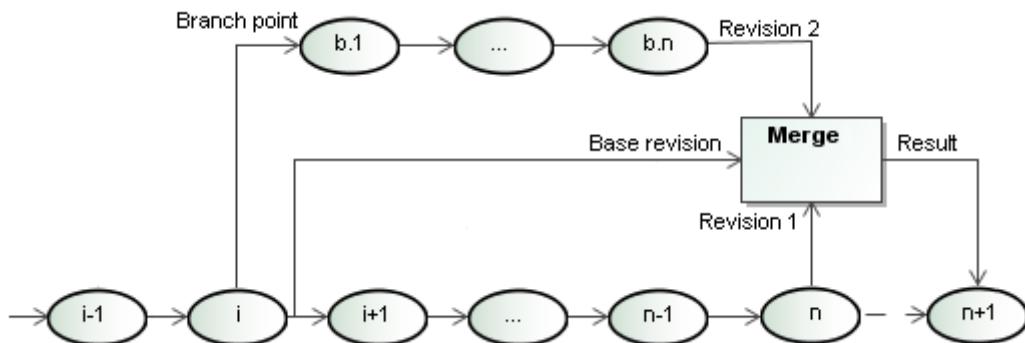


Figure 155 -- 3-way merge

2-way merge

2-way merging is a specific case of 3-way merging. This type of merging is usually used in a non-teamwork environment. In the 2-way merging there is no explicit ancestor available, thus the target version or file is taken as an ancestor.

See Figure 156 on page 295, where the first user works with project version *i* and creates project versions *i*+1, ..., *n*-1, *n* and the other user creates branch *b*.1 of the project *i* and later creates project versions *b*.1, ..., *b*.*n*.

Merging of the following projects is performed: project of the first user (n) and project of the second user ($b.n.$) are merged.

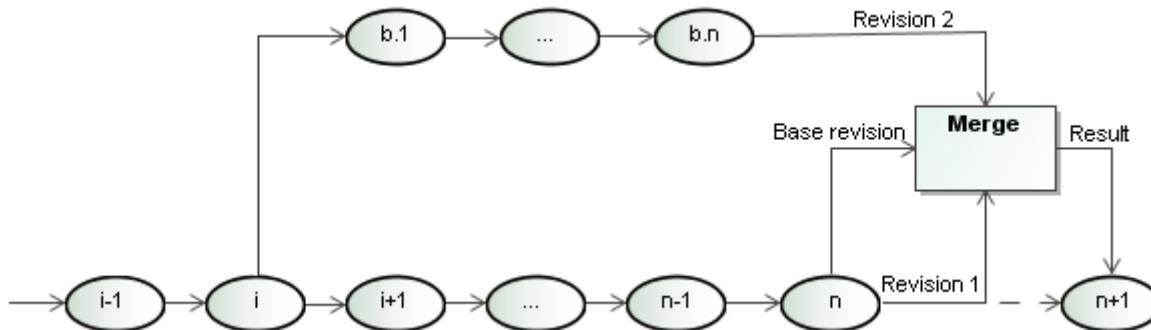


Figure 156 -- 2-way merge

Model Merge Concepts

A change is a delta between the ancestor and participants of a 2-way or 3-way merge. The participants of 2-way and 3-way merges are called contributors. Changes in Model Merge are classified as addition, deletion, modification, move, and order changes. Every change can be accepted or rejected and has dependent changes. If conflicts are detected, the user will be informed which change created the conflict.

Merging begins with building a composite change tree, which consists of model, diagramming, and non-model changes.

Model, diagramming, and non-model changes depend on each other. This means that accepting or rejecting one change not only will accept or reject other changes in the same change tree, but also in other change trees.

For more information about model merge concept, see “Merging concepts in details” on page 315.

Model Merge demonstration

This section demonstrates the following usage scenarios case studies:

- “Usage scenario 1 - 3-way Merge and Analysis” on page 295.
- “Usage scenario 2 - Merging in Teamwork System” on page 299.
- “Usage scenario 3 - Copying changes from the branch to Trunk (in Teamwork) with conflict resolution” on page 305.
- “Usage scenario 4 - Representing DSL elements in the Merged results” on page 313

These case studies show how merging is performed by accepting or rejecting changes and resolving conflicts. Inventory Control System.mdzip project will be used to show basic merge functionality. You can find this project file in the <MagicDraw installation folder>/samples/case studies.

Usage scenario 1 - 3-way Merge and Analysis

The usage scenario 1 presents step-by-step instructions for merging projects and analyzing results.

The model merge functionality in MagicDraw works for both - local projects and projects stored in MagicDraw Teamwork Server. This usage scenario presents model merge of the locally stored projects. For information

about how merging is performed in Teamwork, see “Usage scenario 2 - Merging in Teamwork System” on page 299.

In this usage scenario User1 and User2 will make changes in parallel in the same project and later User1 will copy the changes from User2 file to his/her own file.

User 1 actions before merging

1. Open the Inventory Control System.mdzip project.
2. Create a new *Customer Group* class in the *Top Level class diagram* class diagram, in the *Static View* package (see Figure 157 on page 296).
3. Save the project.

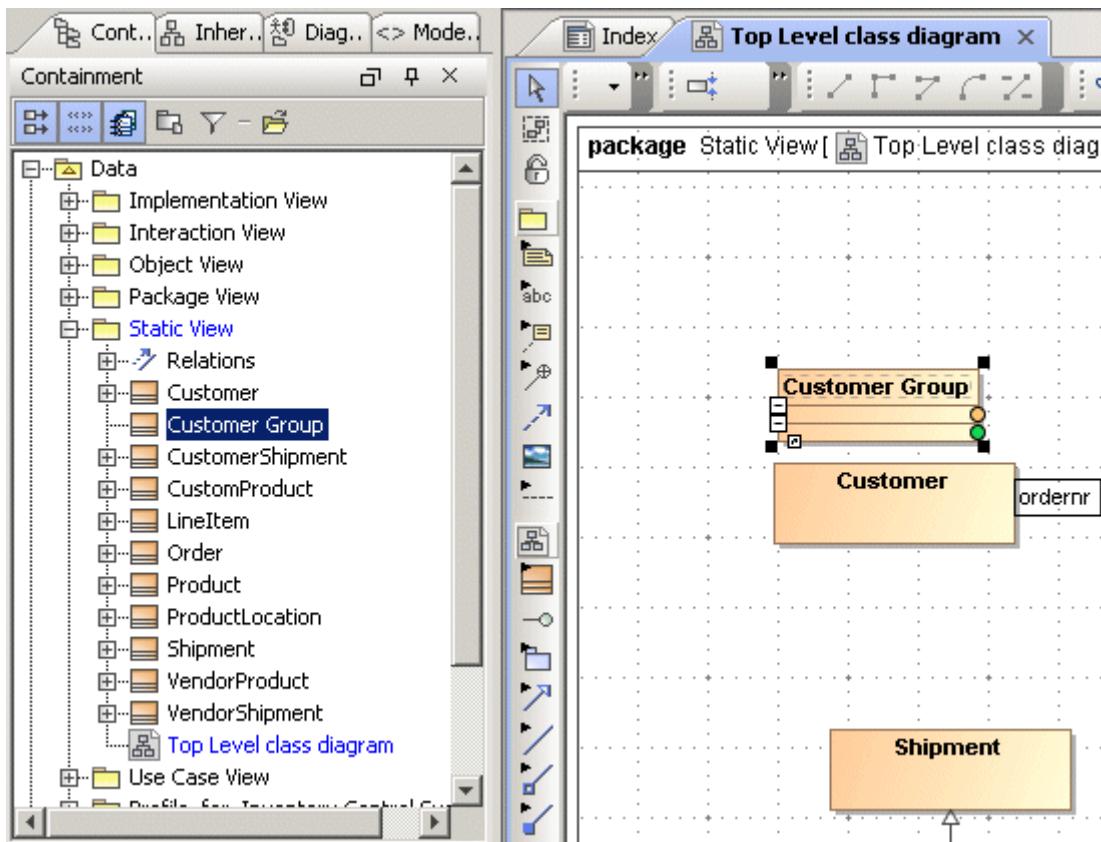


Figure 157 -- Changes made by User1 in the Inventory Control System.mdzip project

User 2 actions before merging

1. Open the original *Inventory Control System.mdzip* project.
2. Rename the *Shipment* class to *Delivery* in the *Product Shipment object diagram* class diagram (see Figure 158 on page 297).

3. Save this project.

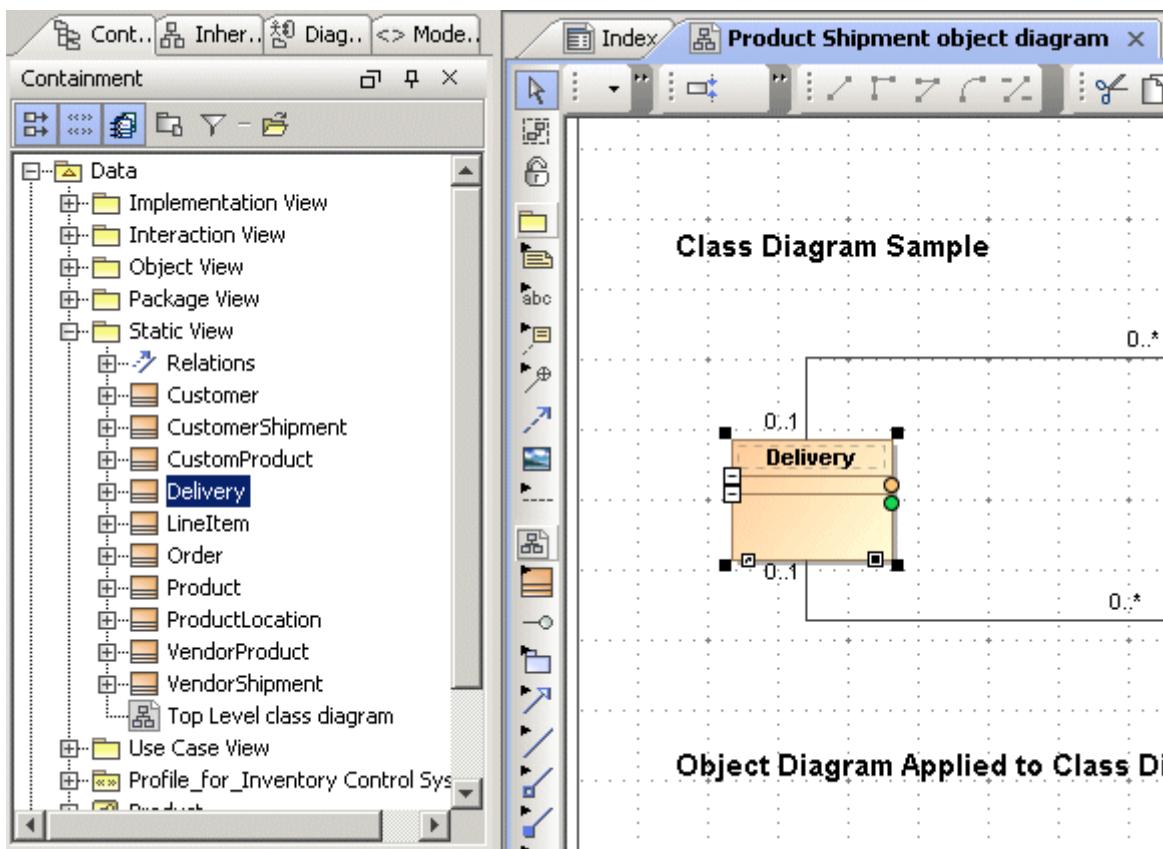


Figure 158 -- Changes made by User2 in the Inventory Control System.mdzip project

User1 actions for merging the projects

1. Open the target Inventory Control System.mdzip project, which includes changes made by him/herself.
2. In the **Tools** menu, click **Project Merge**. The **Merge Projects** dialog box opens .

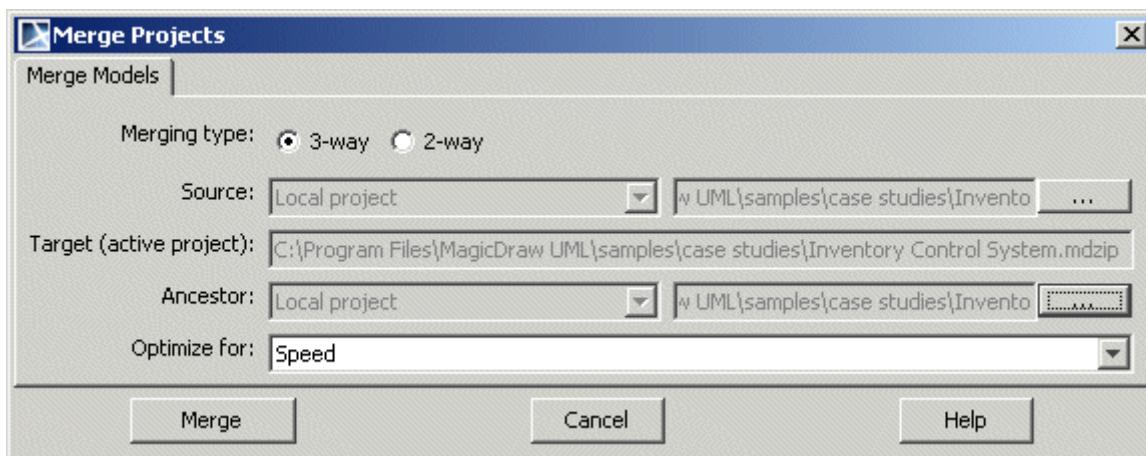


Figure 159 -- The Merge Projects dialog box

3. In the **Source** group, click the “...” button and select the Inventory Control System.mdzip project, which includes the changes made by User2.
4. In the **Ancestor** group, click the “...” button and select the original Inventory Control System.mdzip project that contains no changes.

5. Click **Merge**.

NOTE

A 3-way merging needs an ancestor project. If there is no ancestor project, use 2-way merge. For more information about 2-way and 3-way merge, see the “Introduction to Merging” on page 293.

Analyzing merging results

The **Merge** window allows reviewing differences. The merger automatically applies changes made in both source and target projects and highlights those changes in the Merged Result tree.

Grey lines underneath the *Static View* means that there are changes inside the package. Expand the *Static View* node and see that *Customer Group* class is added in the package, the *Shipment* class and the *Top Level class diagram* are modified.

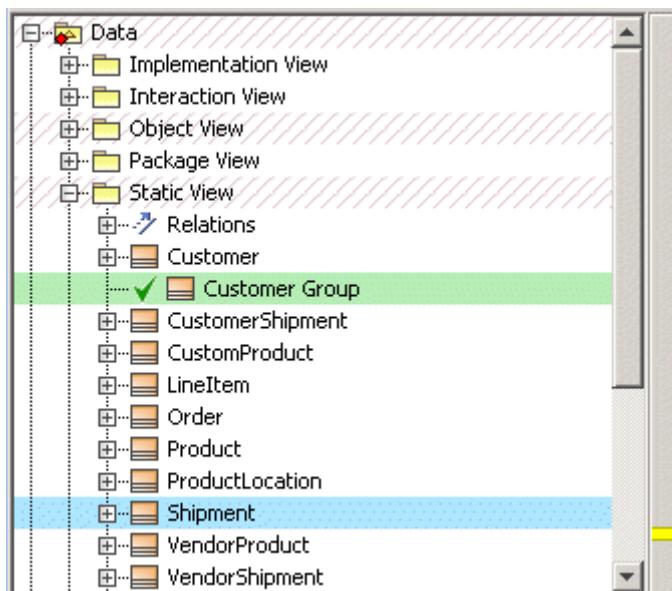


Figure 160 -- The Merged Result tree - the Customer Group class is added in the Static View package

Completing the merging procedure

Confirm the changes made to the target by clicking the **Finish Merging** button. The changes in the source project will be copied to the target project. In this example, the *Customer Group* class is added and the *Shipment* class is renamed as *Delivery* (see Figure 161 on page 299).

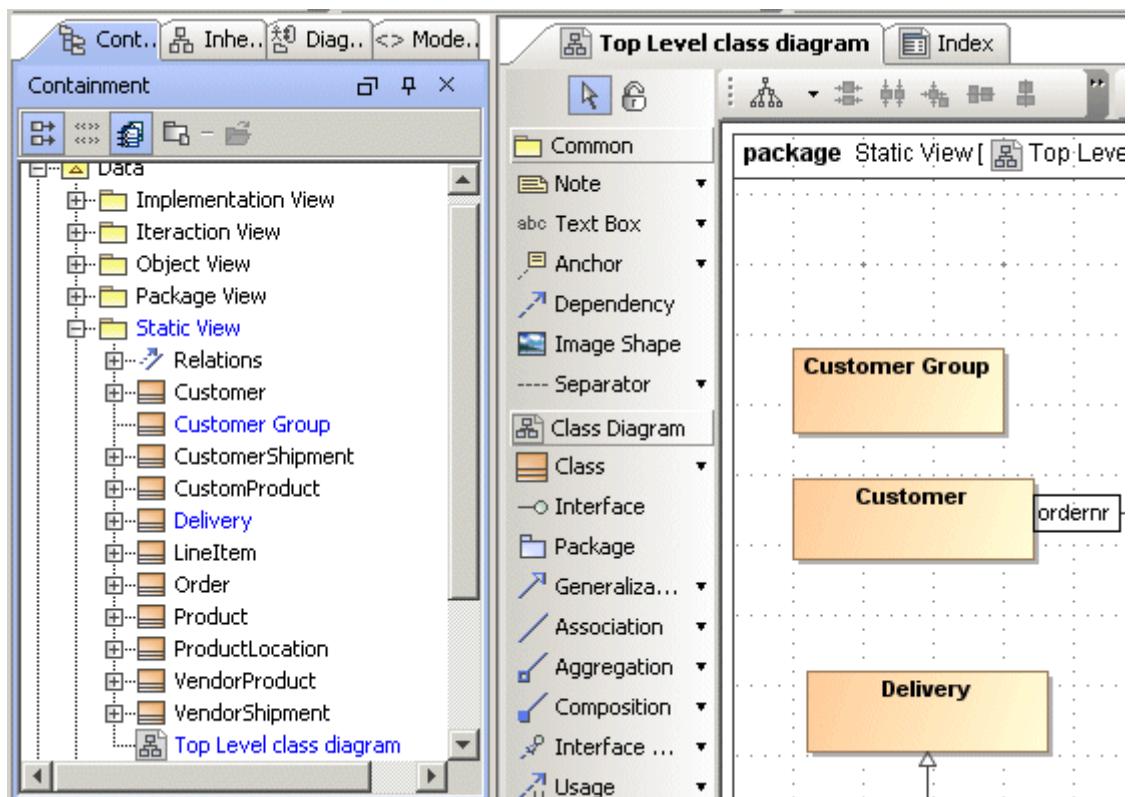


Figure 161 -- The Inventory Control System.mdzip project after merge

Usage scenario 2 - Merging in Teamwork System

This usage scenario shows how simple changes can be copied from the release branch to the main development branch when *User1* and *User2* are working on the same project in Teamwork Environment.

User1 action before merging projects in the Teamwork System

Add the Inventory Control System.mdzip project to the Teamwork Server.

User2 actions before merging: create a branch

Create a branch for the Inventory Control System.mdzip project.

1. In the **Teamwork** menu, click **Projects**. The **Edit Projects** dialog box opens.

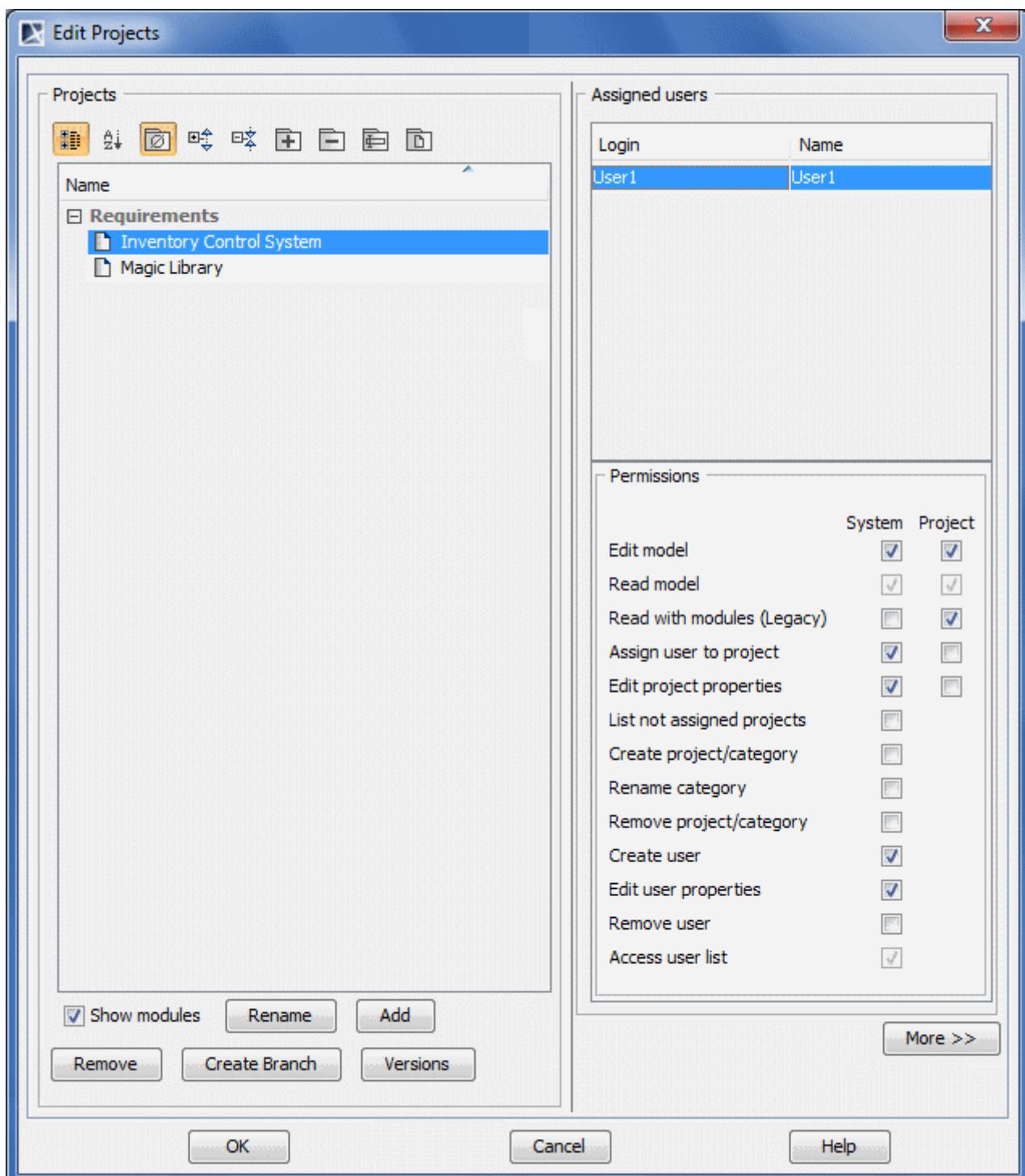


Figure 162 -- The Edit Projects dialog box

2. Select the Inventory Control System project and click the **Versions** button. The **Versions** dialog box opens.

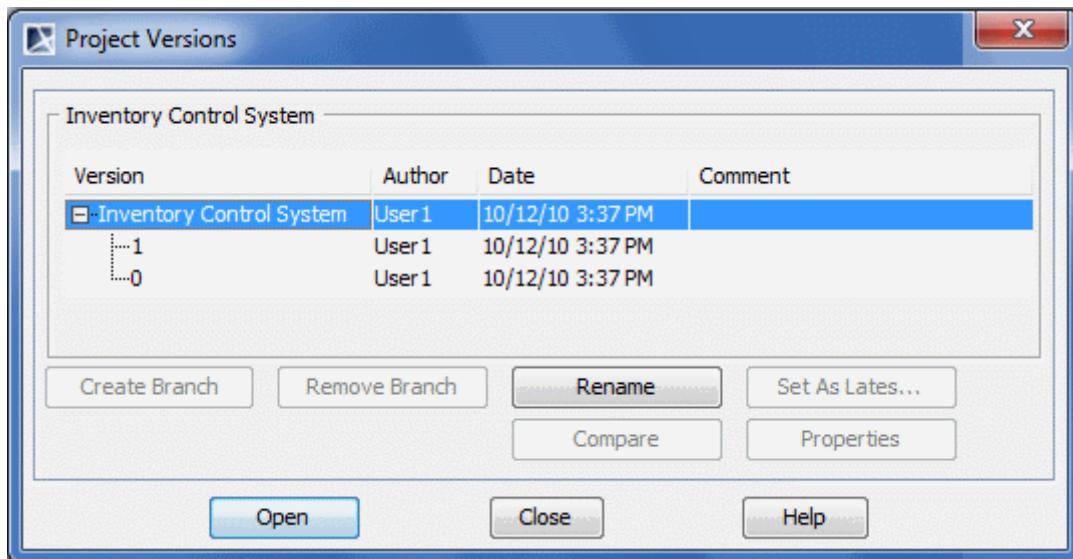


Figure 163 -- The Versions dialog box

3. Select the first version of the project and click **Create Branch**. Now User2 has created a branch, which is derived from the project version 1 from trunk.

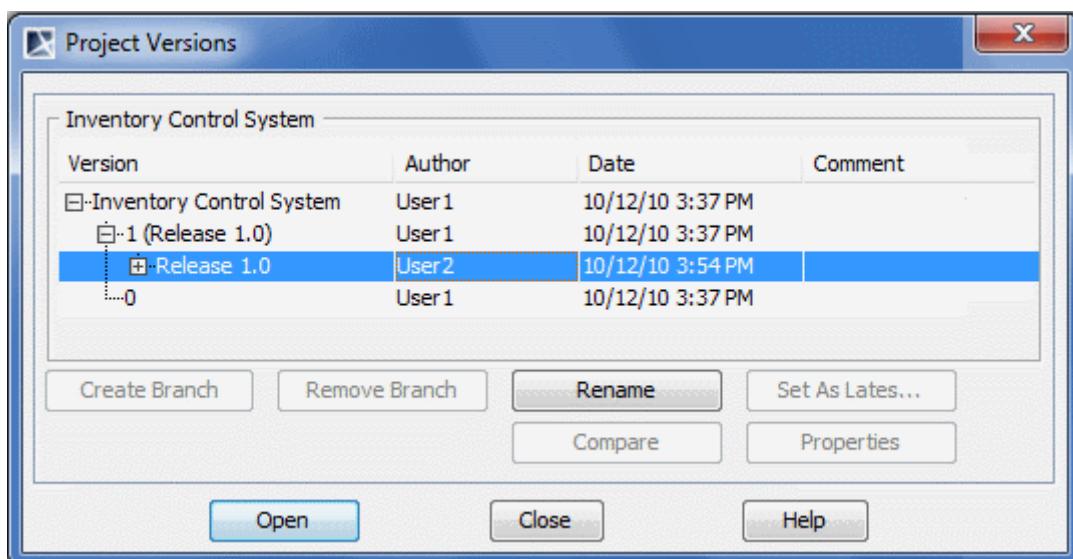


Figure 164 -- The Inventory Control System project versions with created Release 1.0 branch

User2 actions before merging: modify the project.

- In the **Versions** dialog box, select *Release 1.0* and click **Open**. The *Release 1.0* branch of the *Inventory Control System* project opens.
- Create the *Customer Group* class in *Top Level class diagram*, in the *Static View* package (see Figure 165 on page 302).
- Commit changes.

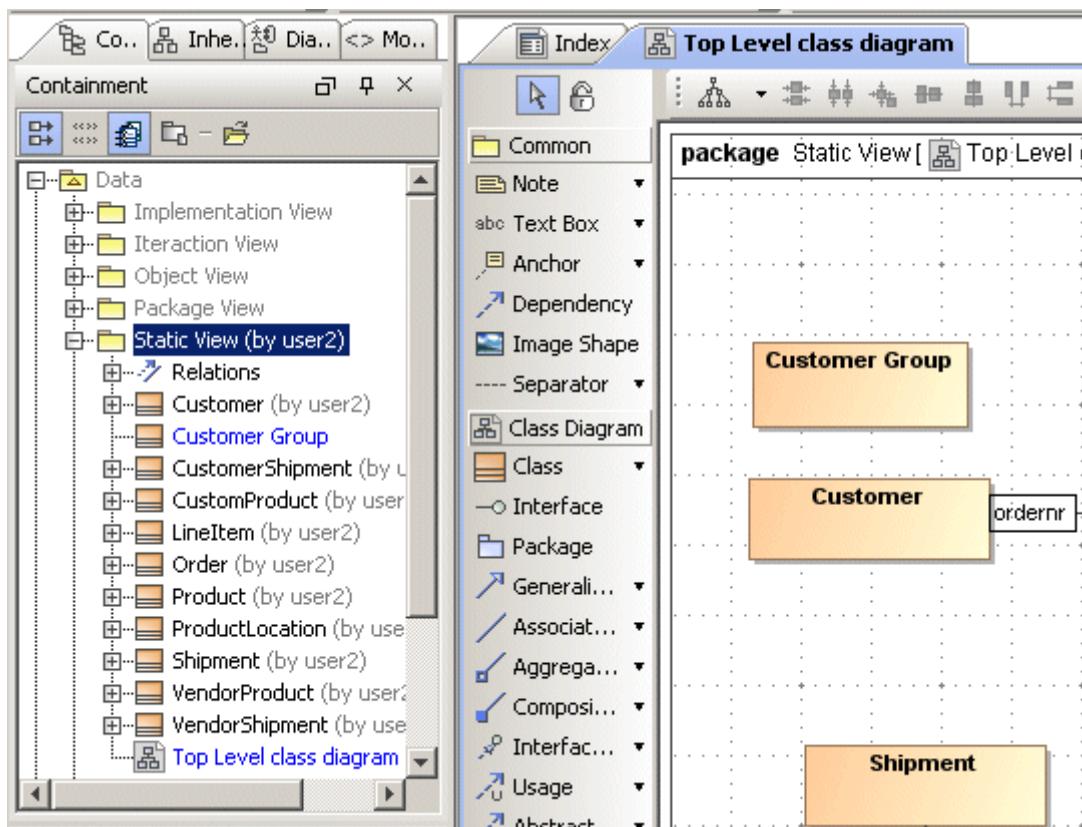


Figure 165 -- Changes made by User2

For more information about branching in Teamwork, see “Project branching in Teamwork” section in MagicDraw Teamwork UserGuide.pdf.

User1 actions: opening the latest version of trunk.

- From the **Teamwork** menu, select **Open Teamwork Project**. The **Open Teamwork Project** dialog opens. Select the *Inventory Control System* project and click **Open**. The latest version of trunk is opened.

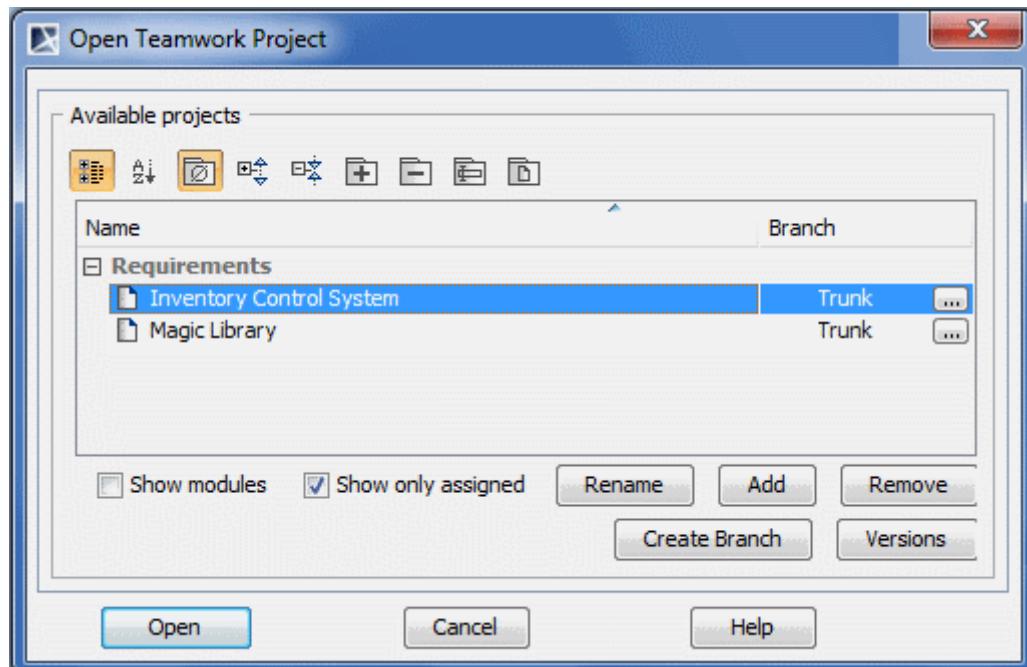


Figure 166 -- The Open Teamwork Project dialog box

User1 actions: merge changes from branch into the trunk

- From the **Teamwork** menu, select **Merge From**. The **Select Teamwork Project** dialog opens.

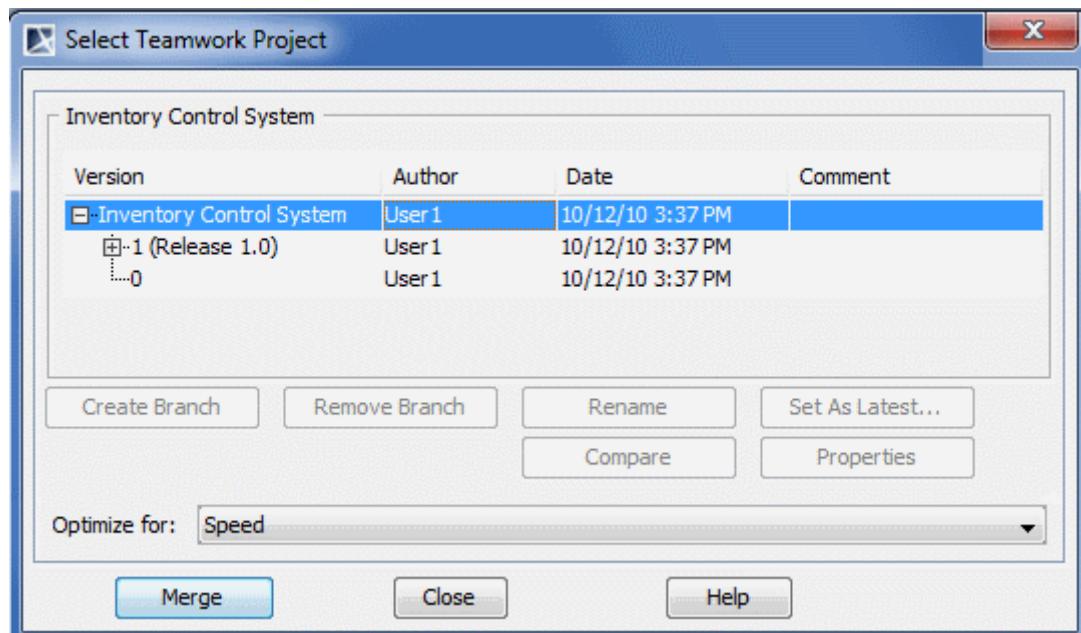


Figure 167 -- The Select Teamwork Project dialog box

3. Expand the 1st node and select the *Release 1.0* branch

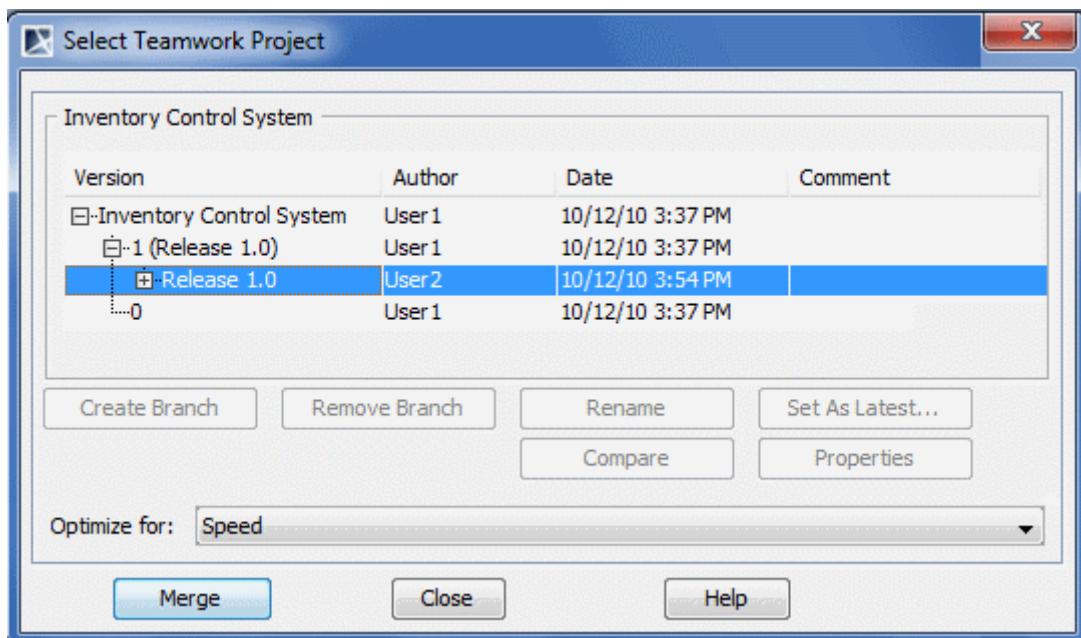


Figure 168 -- The Select Teamwork Project dialog box with selected branch from which we are going to copy changes

4. Click **Merge**. The **Merge** window opens in which you may review differences and complete the merge procedure.

Analyzing the merge results

In the **Merge Result** tree, expand the *Static View* node and you can see that *Customer Group* class has been added to the package and the *Top Level class diagram* has modification changes (see Figure 169 on page 304).

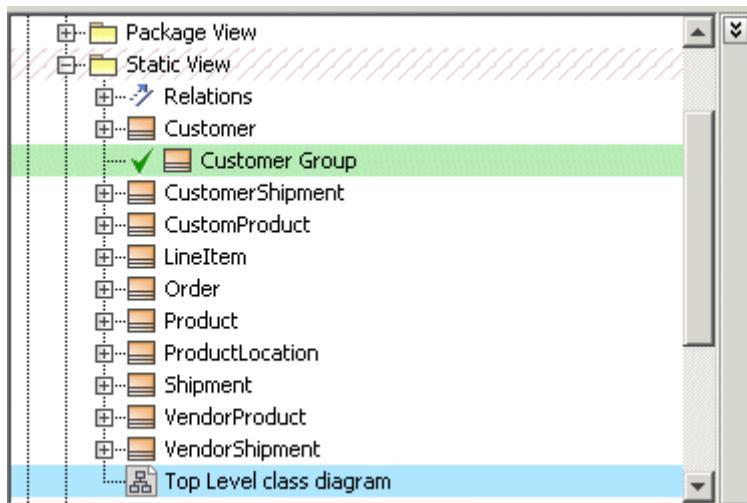


Figure 169 -- Usage scenario 2 - the Merge window, Merge Result tree

User1 actions: confirm changes made to the target project

1. In the **Merge** window, click the **Finish Merging** button.. The changes from the source project are copied to the target project. In this sample the *Customer Group* class is added to the trunk project (see Figure 170 on page 305).
2. Commit the changes to the Teamwork Server.

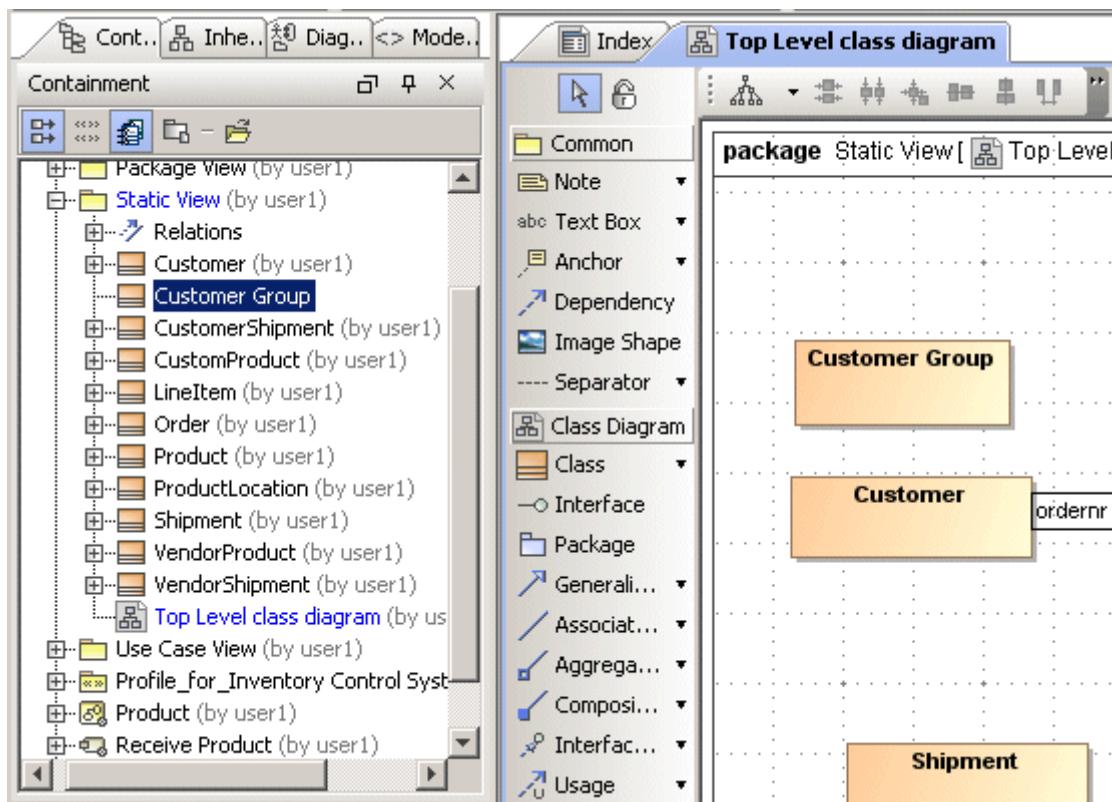


Figure 170 -- Changes are copied from the branch to the Trunk - the Customer Group class is added to the Trunk project

For more information about analyzing merging results, see “Analyzing Merging Results” on page 319.

Usage scenario 3 - Copying changes from the branch to Trunk (in Teamwork) with conflict resolution

This usage scenario 3 analyzes how conflicts are resolved during the merging process.

User 1 actions before merging: add the *Inventory Control System.mdzip* project to the Teamwork Server and modify it

1. Rename the *Customer* class as *Client* class (see Figure 171 on page 306). The *Customer* class is located on *Top Level class diagram*, in the *Static View* package.

2. Commit changes to the Teamwork Server.

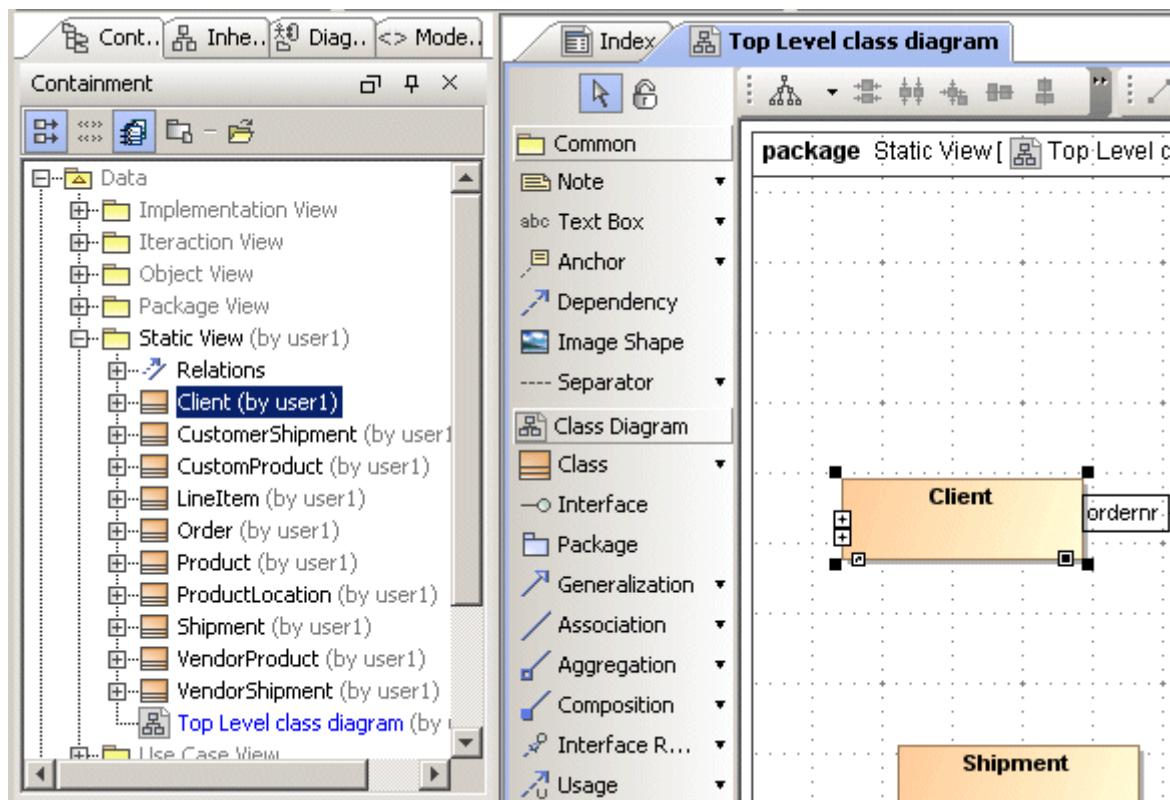


Figure 171 -- Usage scenario 3 - User1 changes in the project

User 2 actions before merging: create a branch of the *Inventory Control System* project

3. From the Teamwork menu, select **Projects**. The **Edit Projects** dialog box opens.

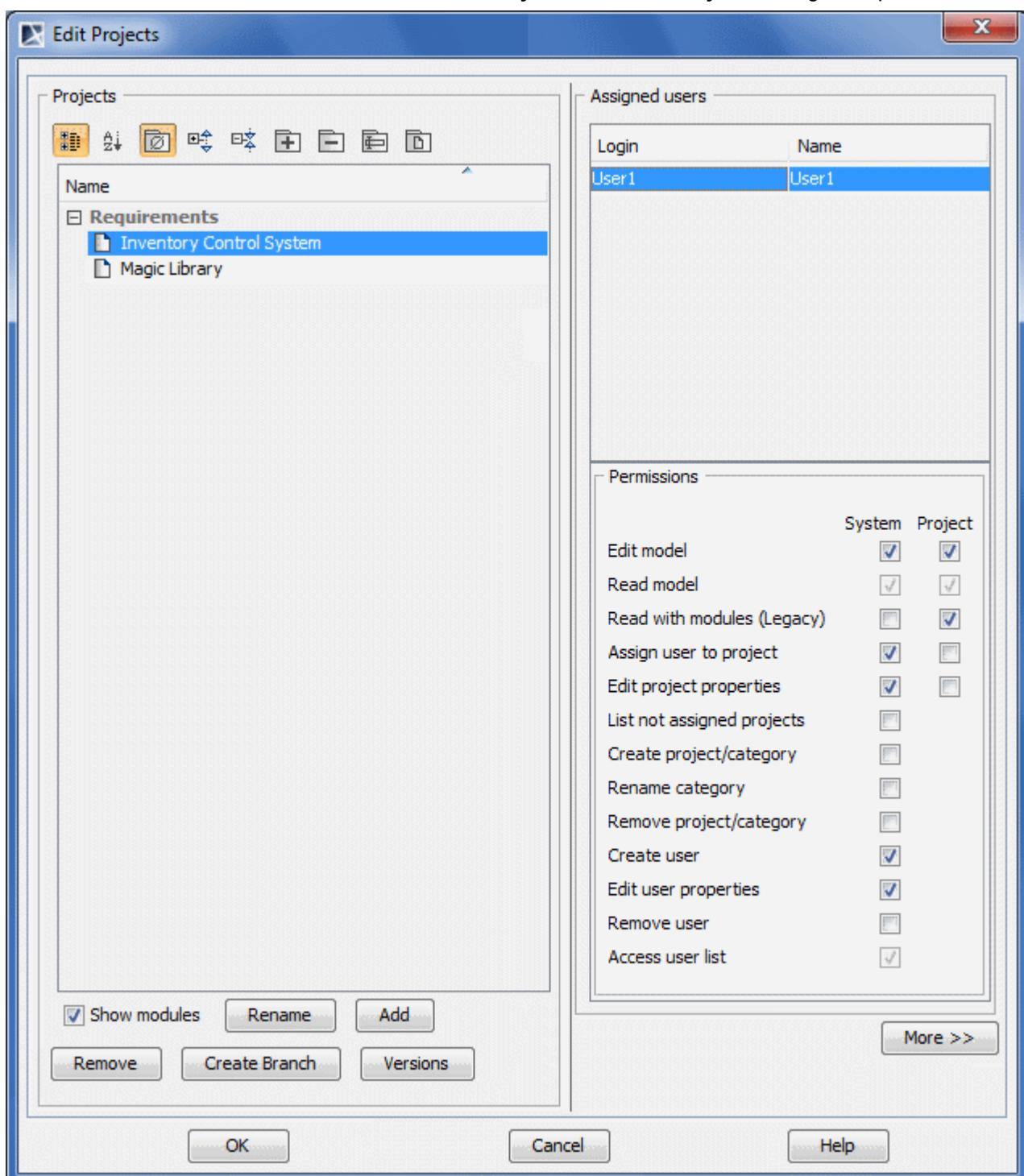


Figure 172 -- Usage scenario 3 - the *Edit Projects* dialog box

4. Select the *Inventory Control System* project and click the **Versions** button. The **Project Versions** dialog box opens.

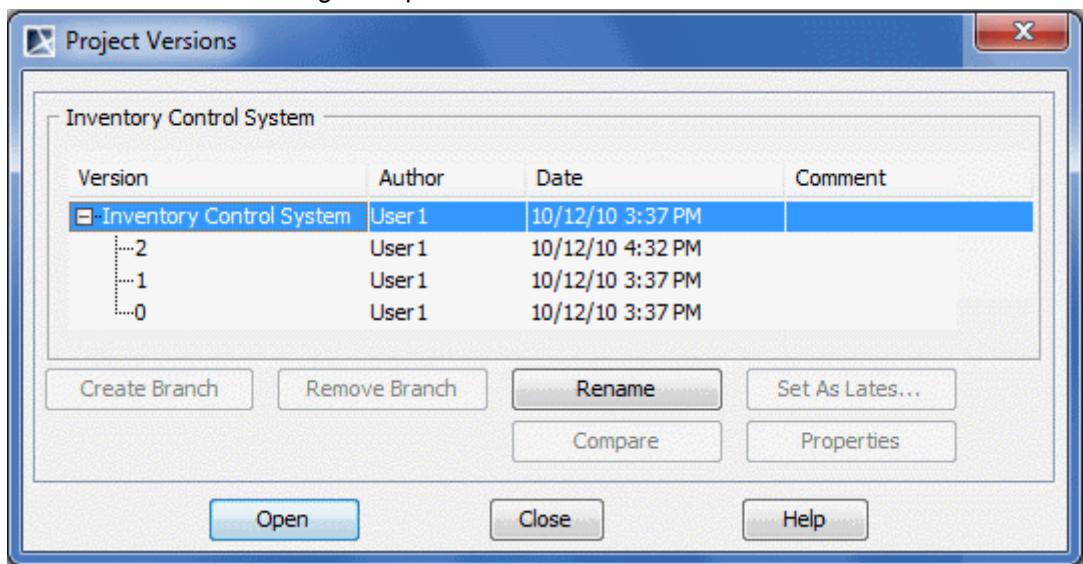


Figure 173 -- Usage scenario 3 - the Versions dialog box

5. Select the first version of the project and click **Create Branch**. Now User2 has created a branch, which is derived from the project version 1 from trunk.

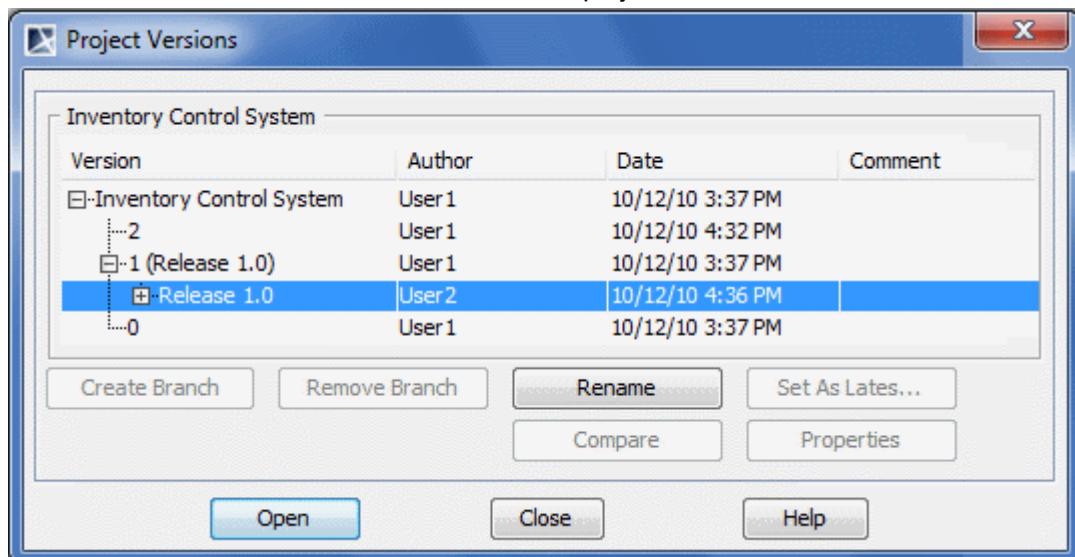


Figure 174 -- Usage scenario 3 - the branch *Release 1.0* is created

User 2 actions before merging: modify a project branch

1. To open the project, in the **Project Versions** dialog box, select *Release 1.0* and click **Open**. The *Release 1.0* branch of the *Inventory Control System* project opens.
2. Rename the *Customer* class to *Customer Group* class (see Figure 175 on page 309). The *Customer* class is located on *Top Level class diagram*, in the *Static View* package.

3. Commit changes.

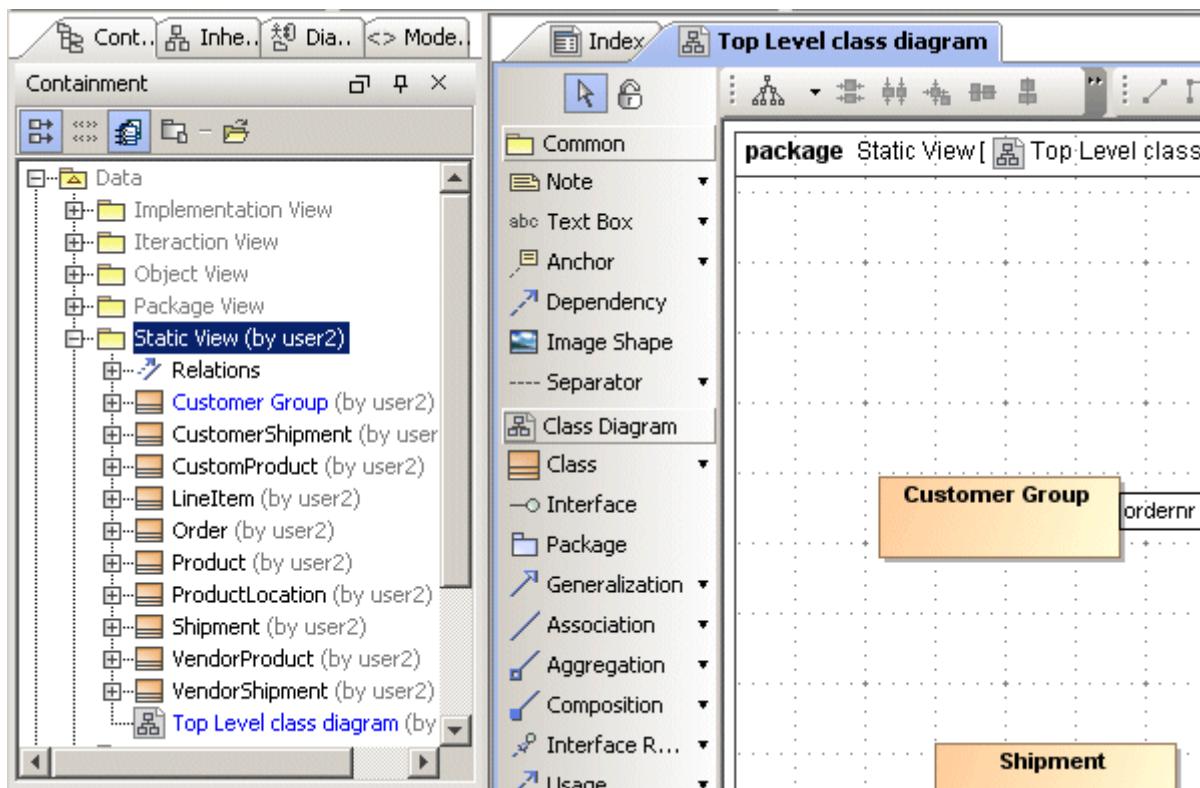


Figure 175 -- Usage scenario 3 - User2 changes made to the project

User1 actions: open the latest version of the trunk

1. From the Teamwork menu, select **Open Teamwork Project**. The **Select Teamwork Project** dialog opens.
2. Select the *Inventory Control System* project and click **Open**. The latest version of trunk opens.

User1 actions: merge changes from the latest version of the project branch to the current active project

1. From the Teamwork menu, choose **Merge From**. The **Select Teamwork Project** dialog opens.

2. Expand the 1 node and select the *Release 1.0* branch.

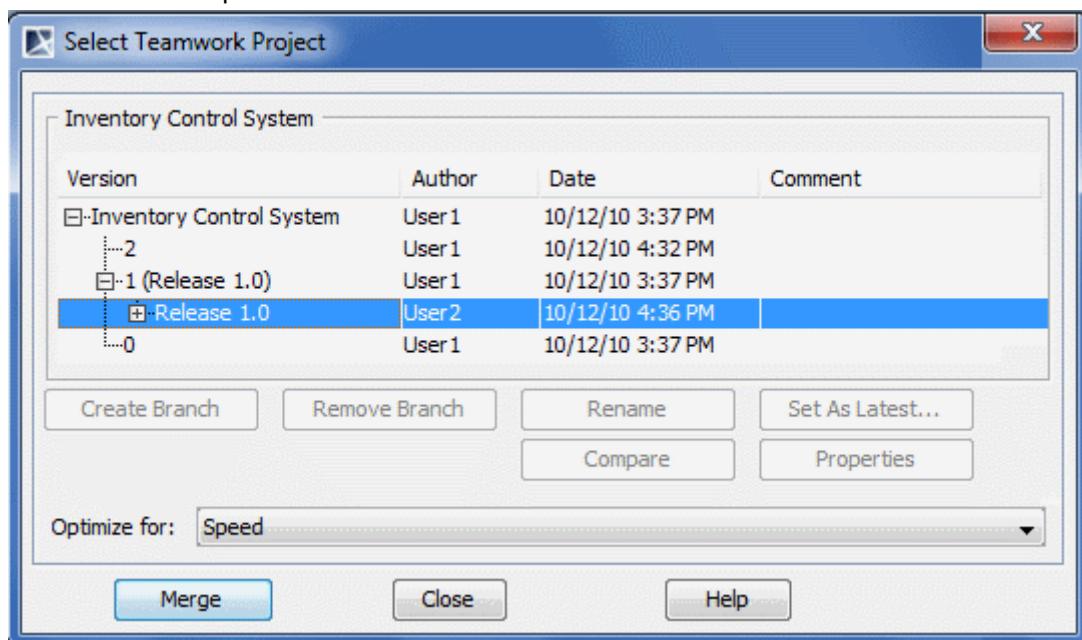


Figure 176 -- Usage scenario 3 - the Select Teamwork Project dialog box

3. Click the **Merge** button. The **Merge** window opens.

Analyzing conflicting changes

1. Figure on page 310 shows the **Merge Results** tree in which there are conflicting changes that cannot be accepted. They are distinguished by the red diamond shapes. The *Data* package has inner changes and conflicts and the *Customer* class has conflicts (the name was changed in the source and target projects).

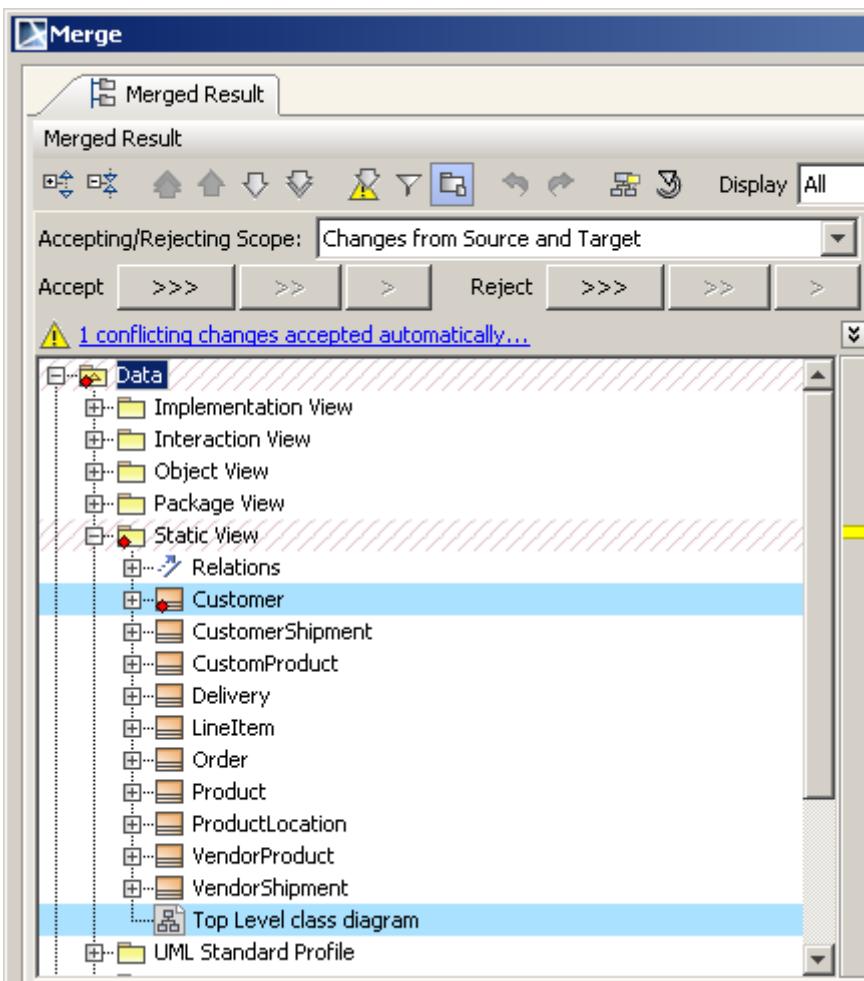


Figure 177 -- The conflicting change

2. The merge engine automatically accepts all non-conflicting changes from the source and target projects. Then it accepts all conflicting changes from the target project and rejects all the conflicting changes from the source. The user can review automatically accepted conflicting changes from the target and select to apply change from the other contributor.
3. Navigate to the automatically accepted conflicting changes from the target project:
 - 3.1 In the quick navigation toolbar, click the **Go To Next Conflict** button.



Figure 178 -- Usage scenario 3 - the Merge window, the Go To Next Conflict button

- 3.2 The element that has conflicts (the *Customer* class) is selected (see Figure 179 on page 312). Its specification is also opened. The name of the *Customer* class is changed in the source and in the target project. Only the change in the target is automatically accepted.

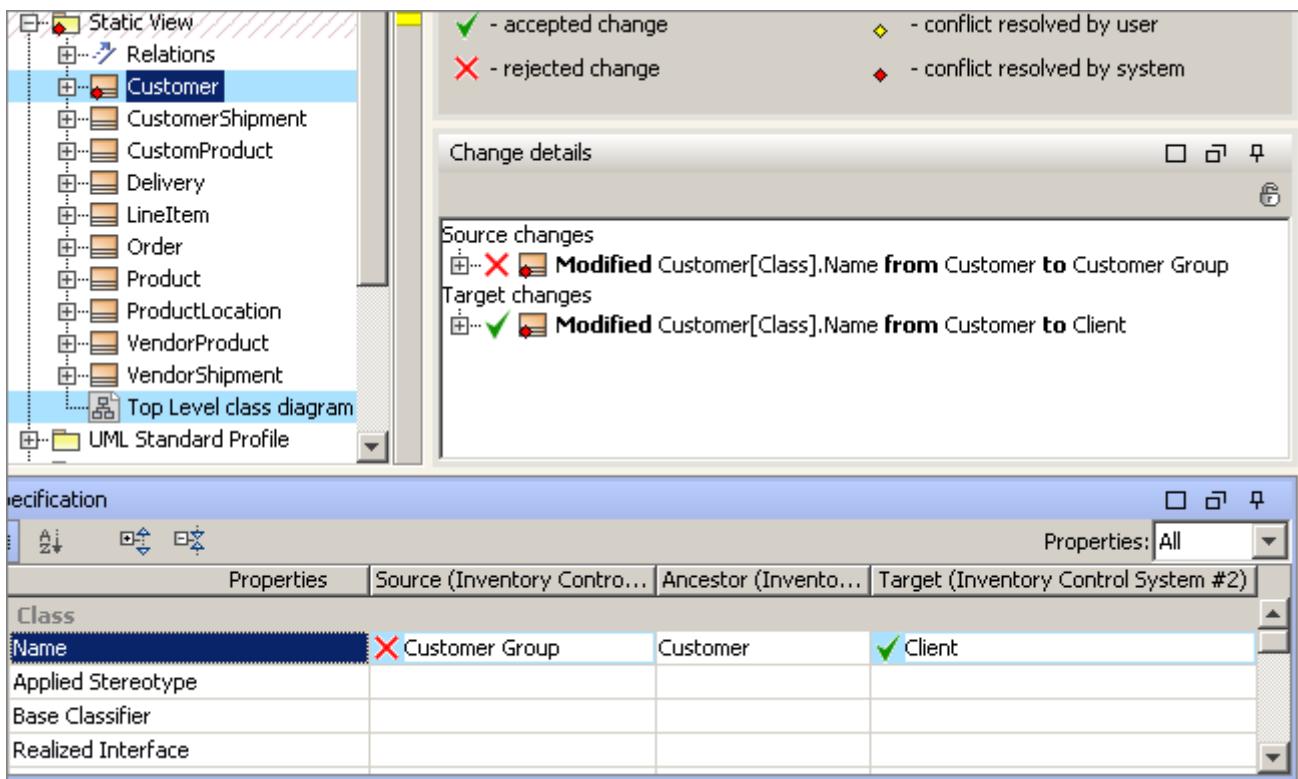


Figure 179 -- Usage scenario 3 - the Merge window, Customer class is selected in the Merged Results tree

Changing the conflict resolution and finishing the merge

1. Change the automatic acceptance setting of the conflict resolution and accept the name change from the source project:
2. In the **Change details** panel of the *Customer* class, right-click the not accepted change. The shortcut menu opens..

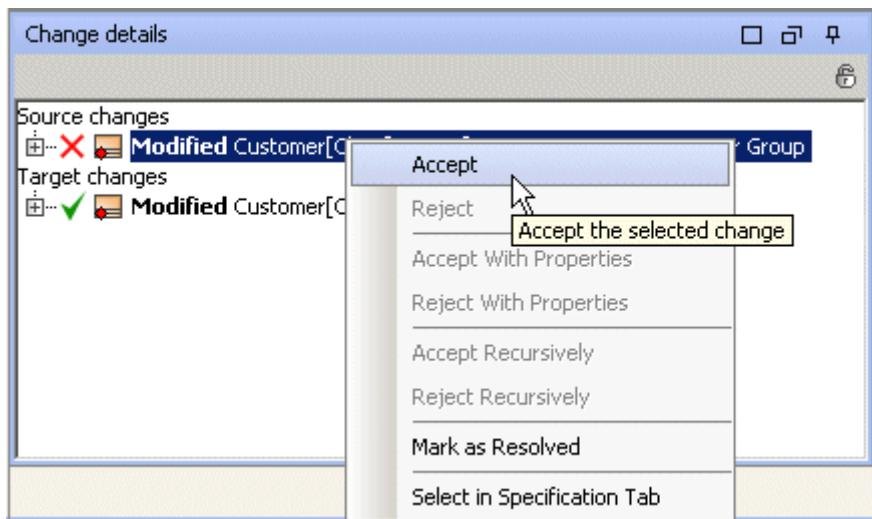


Figure 180 -- The Merge window, Specification panel with invoked shortcut menuAccept the class name modification from the source project by selecting the **Accept** shortcut menu item. The name modification from the source project is accepted.

3. Click the **Finish Merging** button and commit changes to Teamwork.

User1 has successfully resolved conflicting changes and merged them into the target project. The *Customer* class is renamed to *Customer Group* (see Figure 181 on page 313)

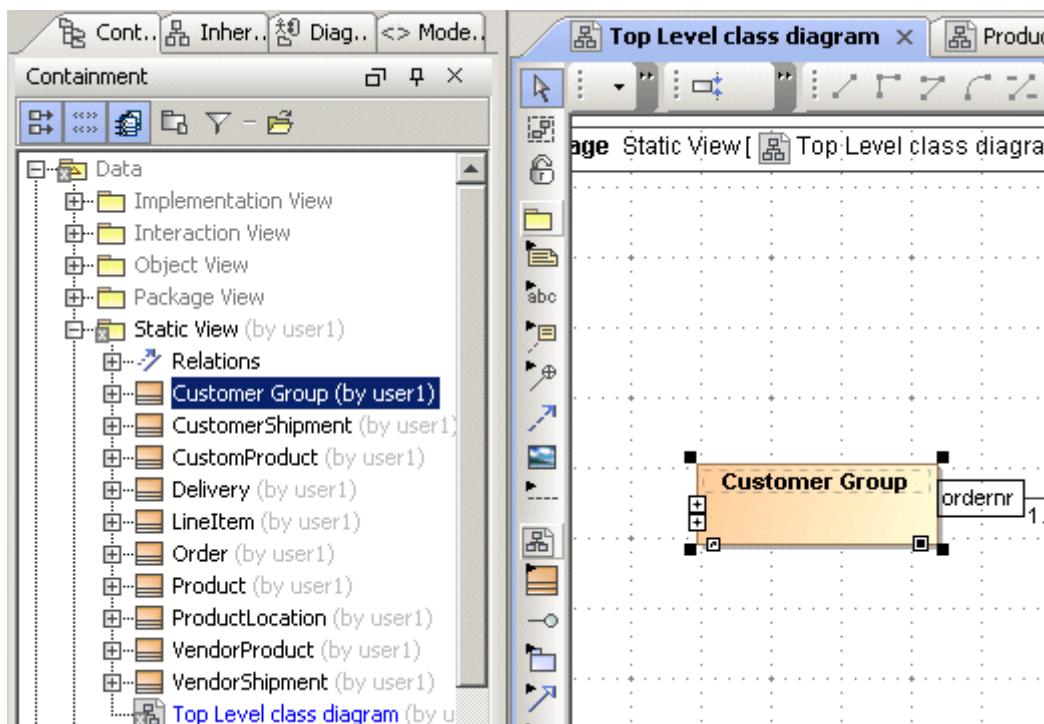


Figure 181 -- Project after merge

Usage scenario 4 - Representing DSL elements in the Merged results

Model elements and their properties are displayed and merged using DSL rules, specified as MagicDraw DSL customization artifacts.

This usage scenario represents how User Interface Modeling diagram (a good DSL example) elements are represented in the **Merge** window.

Actions before merging

The user changes the source project, located in the <MagicDraw installation directory>\samples\diagrams\User Interface Modeling\UI Modeling in System Development.mdzip in the following way:

1. In the model Browser, select the following element: Data::Implementation::GUI Prototypes::StudentProfile::Profile::Group Box::Available Tests:<><<List>>> (see Figure 182 on page 314). Open the <><<List>>> Specification window and change the value for **Horizontal Scroll Bar** property to **Always** (see Figure 183 on page 314).

2. Merge the original and modified projects and .

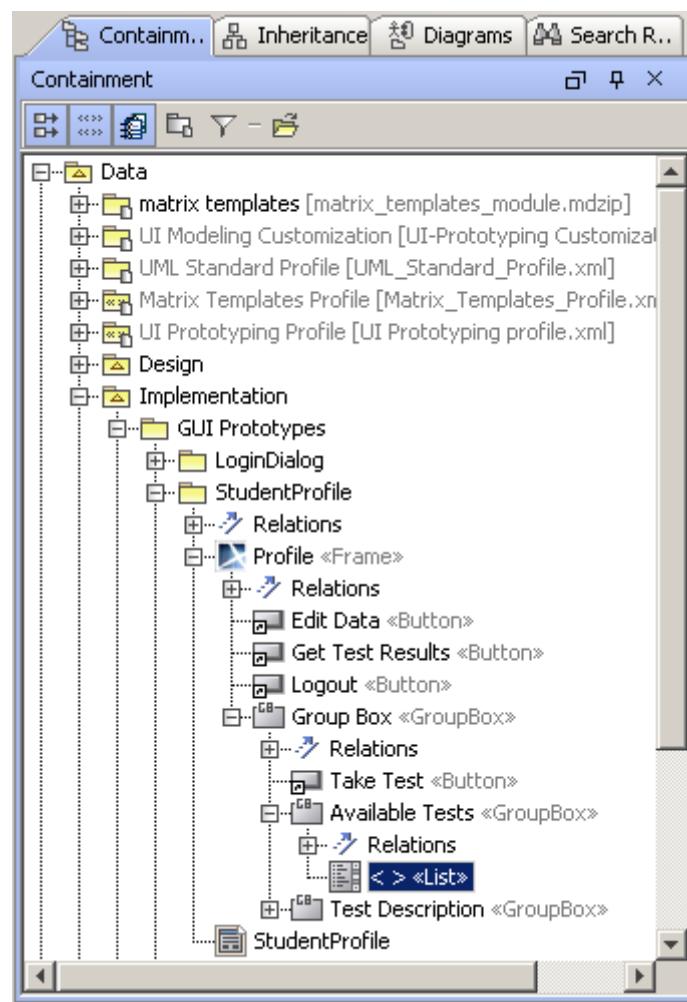


Figure 182 -- Selecting an element in User Interface Modeling diagram

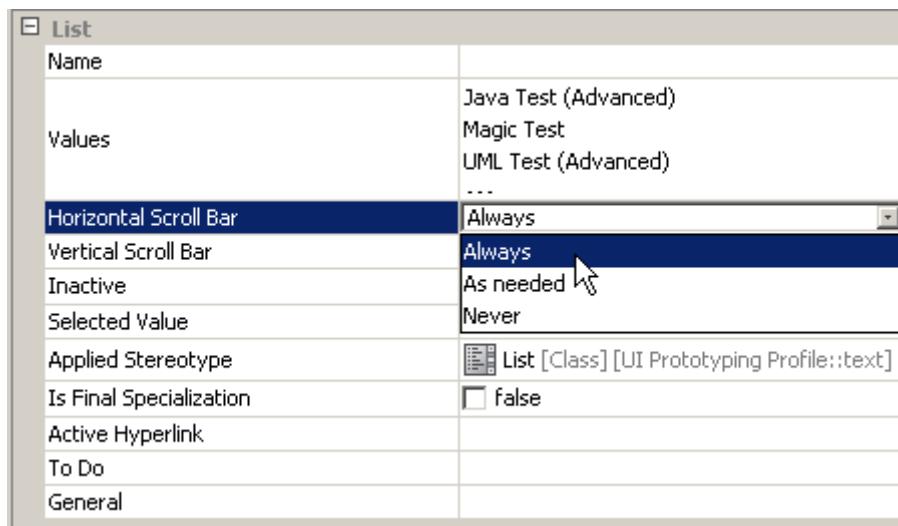


Figure 183 -- Changing the value for the element with the <>List>> stereotype

Analyzing the merge results

The merge results are displayed in the Merge window (see DSL elements representation in the Merge window).

You can see that DSL elements in the Merge window are represented in the same way as ordinary UML elements (and they are treated as first-class UML elements):

- In the **Merged Results** tree, elements are displayed with icons assigned to the corresponding stereotype. In this example, the icon of the <<List>> stereotype is displayed in the Containment tree for the elements stereotyped by it.
- In the **Specification** pane, title of the element matches title of the DSL element

In the **Change details** pane, element changes together with dependencies and conflicts with other changes are represented as a change tree.

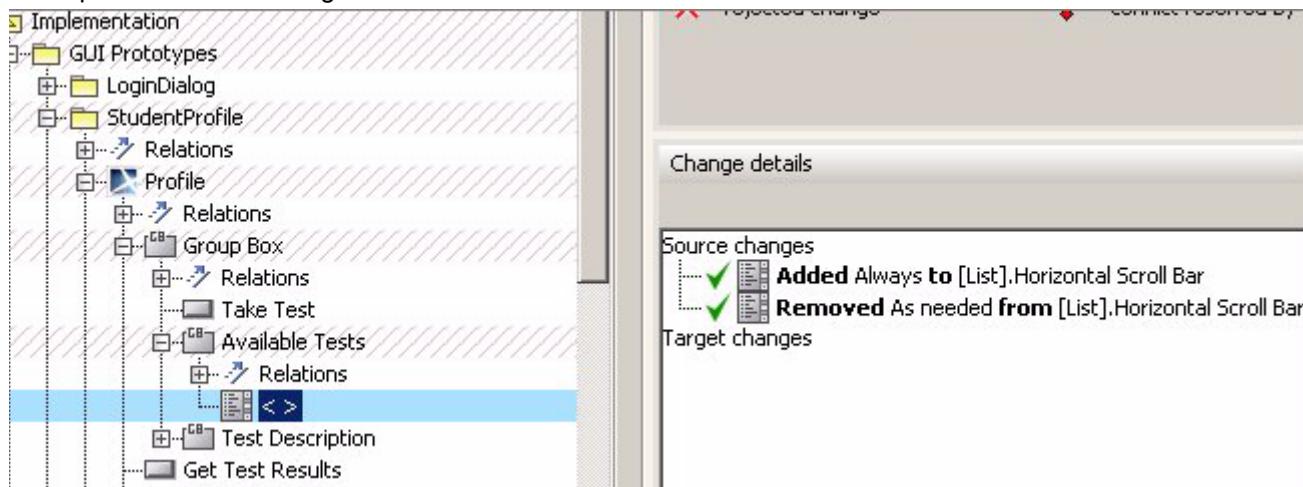


Figure 184 -- DSL elements representation in the Merge window

Merging concepts in details

This section will introduce: the change concept, change the type to accept and reject changes and conflict concept.

A change is a difference, found between the ancestor and participant of the 2-way or 3-way merge. For example, a merger compares elements of the ancestor project to those of the source or target project. Changes are classified as follows:

- addition
- deletion
- modification
- order

For a brief information about changes, see “Change types” on page 315.

Each change can be accepted or rejected and have dependent changes. See “Accepting or Rejecting changes” on page 316 for the details. Dependent changes are described in “Dependent changes” on page 316.

Changes can conflict with each other. See “Conflicting changes” on page 317.

Change types

Addition change

If an element is added to the contributor then an addition change occurs. If the addition change is accepted a new element will be created.

See the example of addition change in the “Addition changes” on page 324.

Modification change

If an element property in the contributor is modified then a modification change occurs (Note: element property means the element specification property, such as *Is Abstract*, *Multiplicity*, and others). For example, if the *Is Abstract* class property in the ancestor has a default value (which is false) and the *Is Abstract* property in the contributor is changed to true, a modification change would occur.

There are three types of modification changes:

- Addition modification change adds a value to the property.
- Deletion modification change removes a value from a property.
- Replacement modification change replaces one value with another. This type of modification change occurs only for properties that have multiplicity less or equal to 1.

For more information about modification change, see “Elements with modified properties” on page 326.

Deletion change

If an element is removed from the contributor, then a deletion change occurs in the removed element. For more information about deletion change, see “Deletion changes” on page 324.

Change order

If the order of elements in the ancestor and contributor differs, then an order change occurs.

For more information about order change, see “Order changes” on page 325.

Accepting or Rejecting changes

Every change, whether it is addition, modification, deletion, or order change, can be accepted or rejected.

Accepted changes are incorporated into the final project. Alternatively, they can be rejected and will not be applied to into the target project.

For more information about accepted or rejected changes in the **Merge** window, see “Analyzing Merging Results” on page 319.

Dependent changes

In some cases, other changes have to be accepted or rejected before accepting or rejecting the selected change. In other words, the selected change sometimes depends on other changes, then it is called a dependent change or dependant.

An example of a dependent change is a type change. If a class attribute type is changed to a type that is created by another change, then the attribute type change is dependent on the change that created the type. This means that type creation has to be accepted before accepting type modification change.

Another example is the type deletion, modification, and addition. Suppose there is an attribute type change in a contributor. The old type is deleted and a new type is added to the contributor. In this case, three changes are formed:

- deletion change (for the old type),

- addition change (for the new type),
- and modification change (for the property type).

These are also ownership changes, but they are accepted together with deletion and addition changes.

A modification change depends on an addition change and a deletion change depends on a modification change. Accepting the addition change does not mean accepting any other changes in this case. Accepting the modification change means accepting the addition change and then accepting itself. Accepting the deletion change means accepting the addition change, the modification change and the deletion change itself. The change dependencies for the described case is depicted in Figure 185 on page 317.

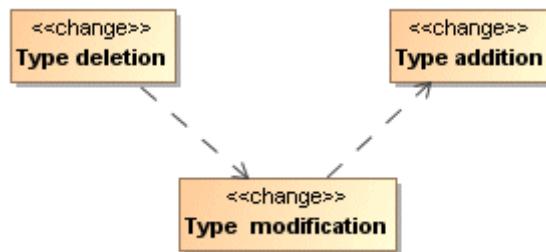


Figure 185 -- Sample of the dependent changes - type deletion, modification, and addition

Conflicting changes

A conflict is two differences that are incompatible with each other, i.e. changes that can not be accepted together. Every change can have several conflicting changes.

NOTE Conflicting changes occur only in a 3-way merge.

Some examples of the conflicting changes:

- Each contributor changed a class' name or any other element properties (metaproPERTY).
- One contributor added an operation to a class and the other contributor deleted the class.
- One contributor moved a class into one package while the other contributor moved it to another package.

For more information about the representation of the conflicting changes, see “Analyzing Merging Results” on page 319.

Building change tree

There are two groups of elements that are compared and merged:

1. Model/diagram elements
2. Non-model elements.

The Model/diagram tree is represented in the Merged Result tree as it is done in the Browser in the main MagicDraw window.

The Non-model tree is represented in the Merged Result tree with two branches:

- Module mount table. Module mount table node lists the in the project used modules. See the sample of changes in the module mount table.
- Project options. Project options node shows if any project options is changed or not.

Sample of changes in the module mount table

Three changes occurs in this sample: Profile A module is updated to a newer version (change c1), which has type A removed (change c2), which implies type unsetting for the typed element P2 (change c3) (see Figure 186 on page 318).

Change c1 and c2 are dependent on each other and change c2 depends on change c3. This situation is depicted in Figure 187 on page 318.

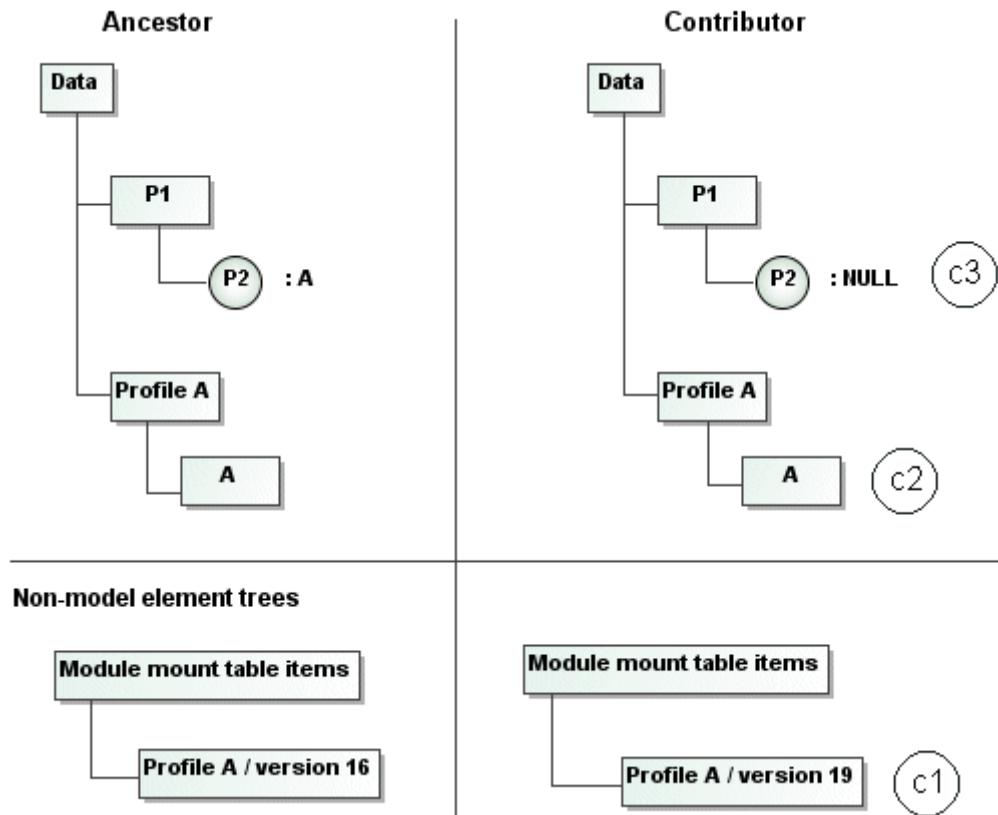


Figure 186 -- Sample of the changes in the module mount table

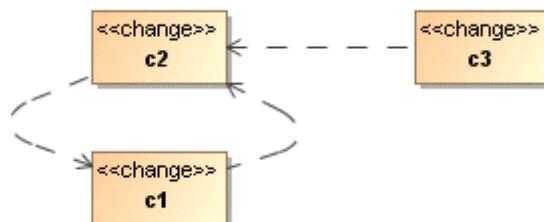


Figure 187 -- Sample of the changes in the module mount table - dependencies

Analyzing Merging Results

Merging results are presented in the **Merge** window.

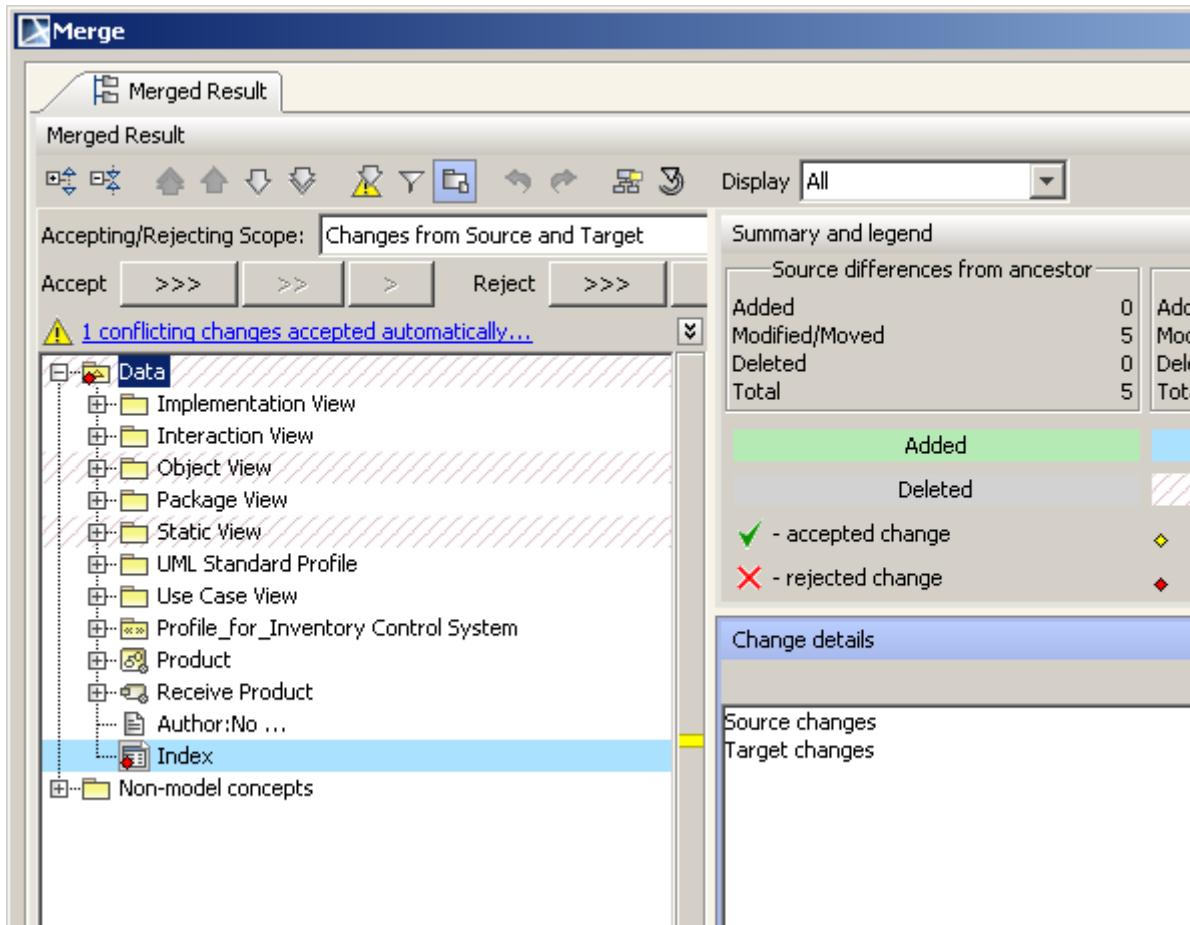


Figure 188 -- The Merge window

The Merge window description

Look at each part of the **Merge** window separately. The **Merge** window consists of the following parts:

- "The Merged Result tree"
- "Toolbar for displaying and navigating through changes"
- "Toolbar for accepting and rejecting changes"
- "Change legend and summary"
- "Buttons for quick navigation through conflicting changes"
- "Element Specification panel"
- "Change details panel"

Reorganizing panels of the **Merge** window

Panels of the **Merge** window can be organized to your desired positions. The following changes can be done to the **Merge** window:

- The **Summary/Legend** panel can be turned on or off using the **Summary/Legend** button, which is on the left side of the **Summary/Legend** panel.

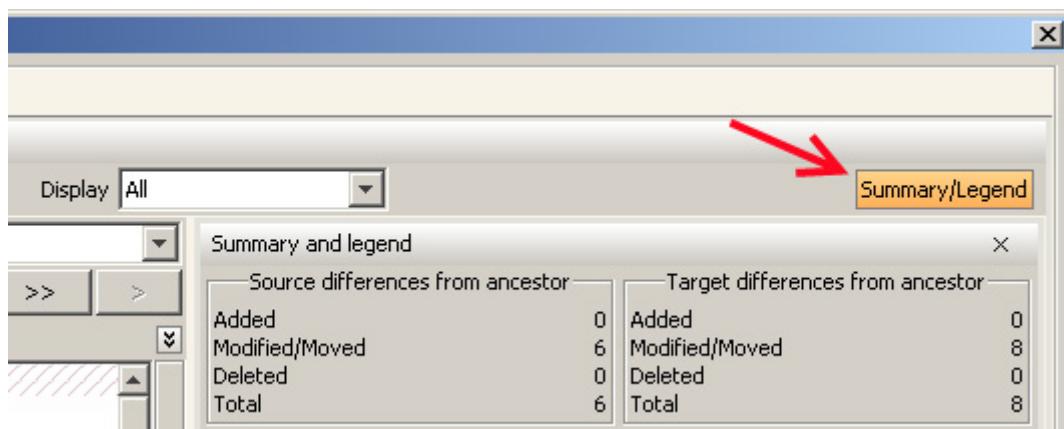


Figure 189 -- The Summary/Legend button in the Merge window

- The **Change details** and **Specification** panels can be maximized, toggled to floating or auto-hide windows using buttons, located at the top-right side of the panel.

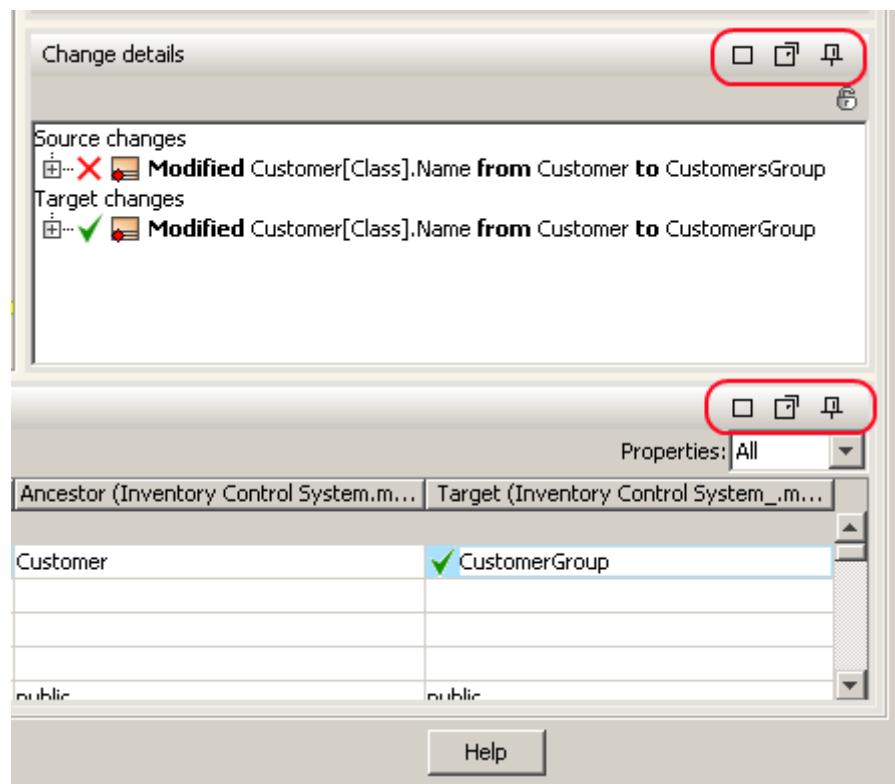


Figure 190 -- Buttons to manage the Change details and the Specification panels in the Merge window

- You can also manage panels of the **Merge** window, using shortcut menu. To invoke it, right-click on the title bar of the window. See the location of the shortcut menu invocation in the following screenshots.

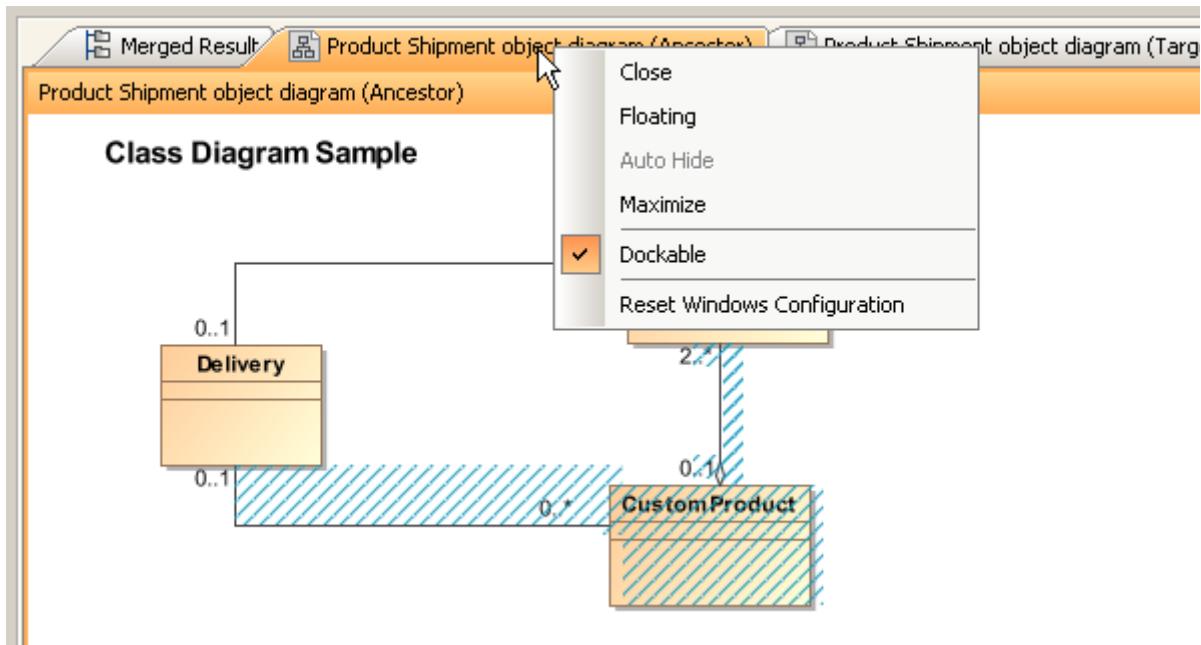


Figure 191 -- Shortcut menu of the Merge panels

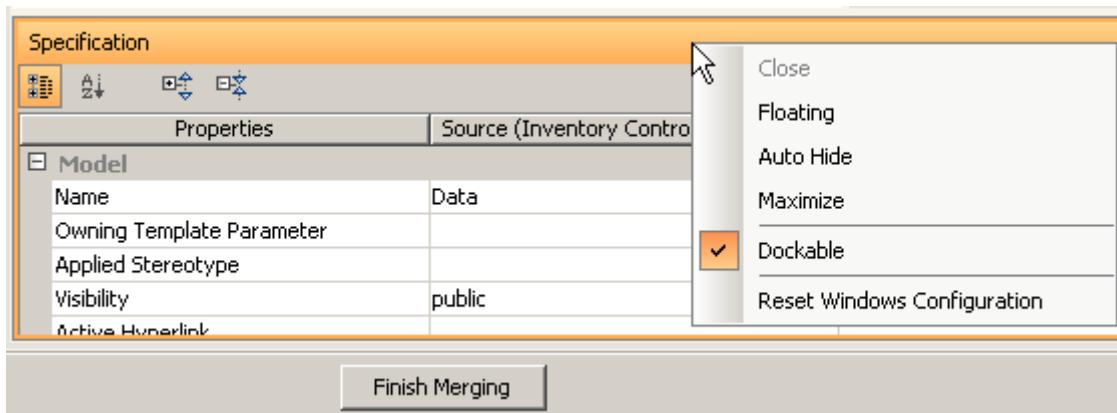


Figure 192 -- Shortcut menu of the Specification window

- To reset the position of the panels of the **Merge** window panel, from the panel shortcut menu select the **Reset Windows Configuration** command or press the **Reset Windows Configuration** button at the top of the **Merge** window.

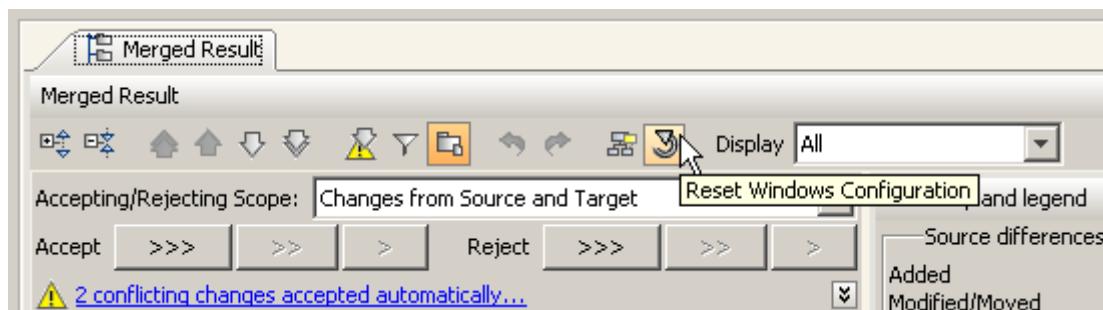


Figure 193 -- The Merge window, Reset Windows Configuration button

For information about managing diagram tabs, see “Viewing changes in diagrams” on page 333.

The Merged Result tree

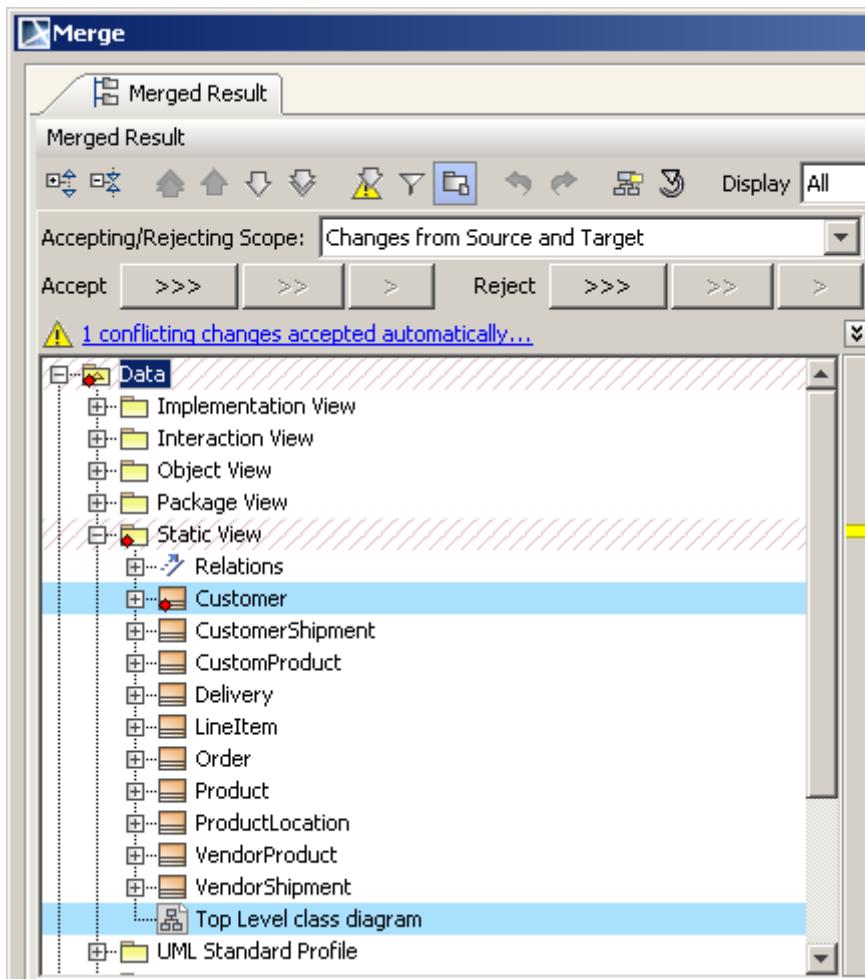


Figure 194 -- The Merged result tree

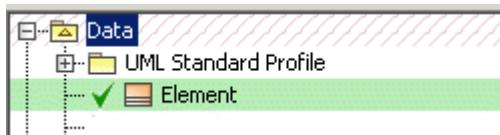
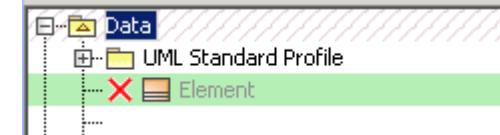
The merged result tree combines both containment and change trees. The following types of changes are displayed in the merged result tree:

1. “Addition changes” on page 324
2. “Deletion changes” on page 324
3. “Move changes” on page 324
4. “Order changes” on page 325
5. “Elements with modified properties” on page 326
6. “Elements with changed inner elements” on page 326.

Modification changes are displayed in the element properties panel other changes are displayed in the **Merged Result** tree.

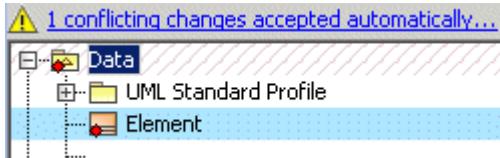
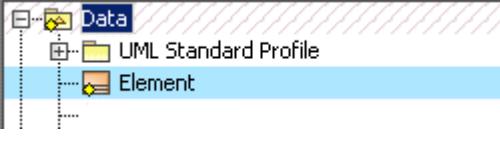
Element decoration in the Merged Result tree

A change occurs on a single element and that element has an icon indicating the state of the change (see the table below).

| Decoration | Change state | Example |
|--|--------------|---|
| Element (green tick before the element) | Accepted |  <p>New element is created in one of the contributors and its addition is accepted.</p> |
| Element (red cross before element) | Rejected |  <p>New element is created in one of the contributors and its addition is rejected.</p> |

For more information about accepted/rejected changes, see “Accepting or Rejecting changes” on page 316.

Changes that have conflicts are additionally decorated with icons showed in the table bellow. Parents of conflicting elements have conflict decorations too.

| Decoration | Change applied by | Example |
|--|-------------------|--|
| Element (red diamond on the left-bottom corner of element) | System |  <p>Element has conflicting modification change.</p> |
| Element (yellow diamond on the left-bottom corner of element) | User |  <p>Element has conflicting modification change which is resolved by the user.</p> |

Addition changes

Addition changes occur when elements are created in the contributors. In the merged result tree added elements are highlighted with green color (see Figure 195 on page 324).

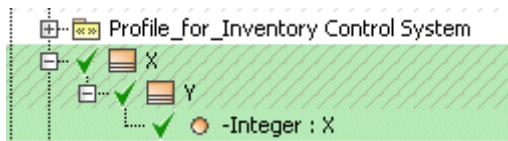


Figure 195 -- Addition change

Deletion changes

Deletion changes occur when elements are deleted in the contributors. Deletion changes are highlighted with grey color (see Figure 196 on page 324).

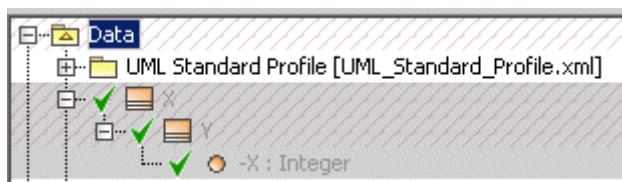


Figure 196 -- Deletion changes

Move changes

Move changes are highlighted using the same blue color that is used to highlight modification changes, because move change is just a kind of modification change. Additionally, move changes are displayed with arrows before the moved element icon. In the following example, element X is moved from package A to package B. The illustration shows the initial situation when where the move change is accepted and element X is owned by package B.



Figure 197 -- Move changes

Navigation from the original to the new location (and vice versa) for elements that have been moved is possible from the moved element shortcut menu (see Figure 198 on page 325 and Figure 199 on page 325).

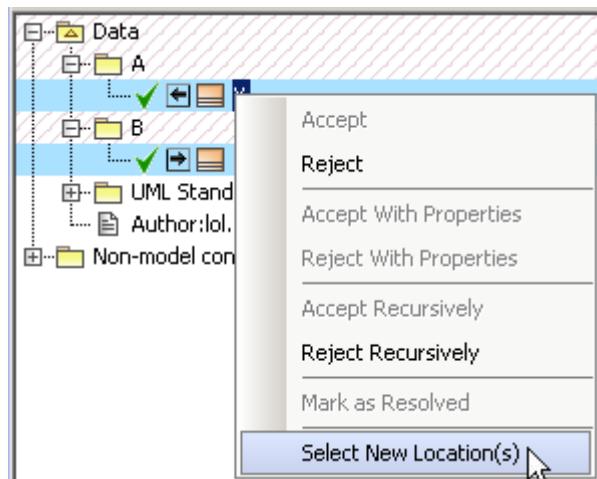


Figure 198 -- Navigation to the new element location

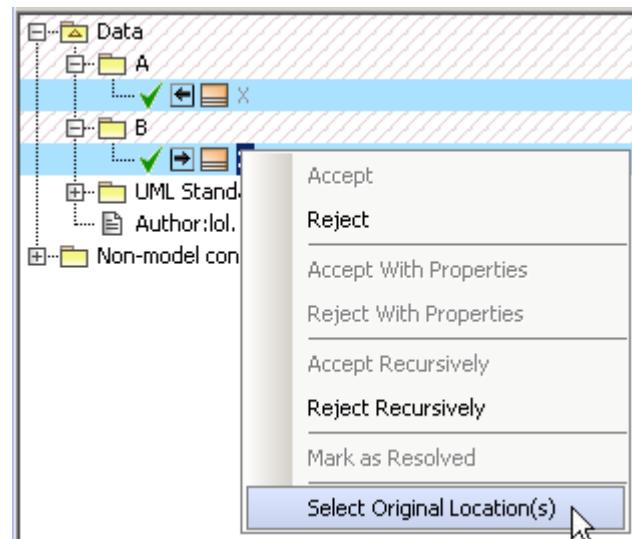


Figure 199 -- Navigation to the original element location

Order changes

Order changes occur on elements such as attributes, operations, and other ordered elements. Even if a single element in a collection has changed its place, the order change is applied to the entire collection. After accepting the change, all elements in the collection is reordered so that it would be the same as in one of the contributors the change occurred.

An element can have several ordered collections. This means several order changes can occur for a single element. Changes in the element will be highlighted using the same blue color that is used to highlight modification changes.

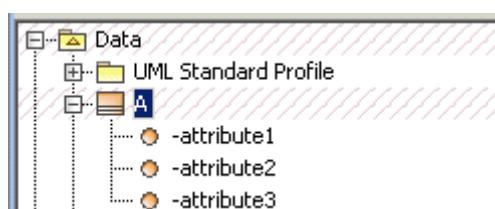


Figure 200 -- Order changes

Elements with modified properties

Modification changes occur when element properties changed in the contributors. Elements whose properties changed are highlighted in blue.

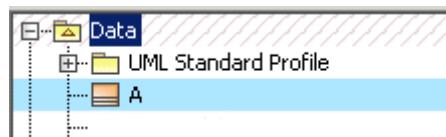


Figure 201 -- Elements with modified properties

Elements with changed inner elements

Elements whose inner elements (in any nesting level) have pending changes are highlighted using red dashes as shown below. If the element has pending property modification changes, then the modification highlight color is mixed with changed inner element highlight, i.e. the element is highlighted using blue color with red dashes.

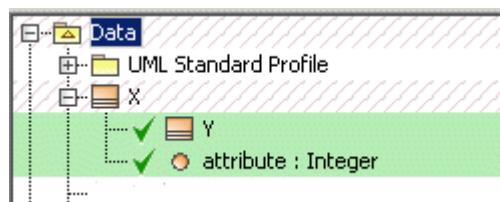


Figure 202 -- Element "X" with changed inner elements "Y" and "attribute"

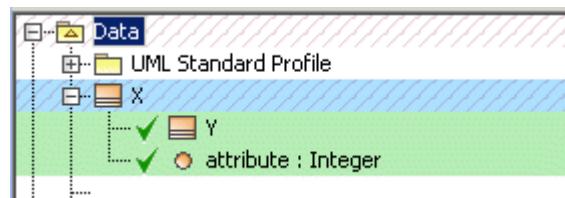


Figure 203 -- Element "X" with changed inner elements "Y" and "attribute" and modification change

Toolbar for displaying and navigating through changes

The toolbar for displaying and navigating through changes is located at the top of the **Merge** window. See the highlighted area in Figure 204 on page 326.

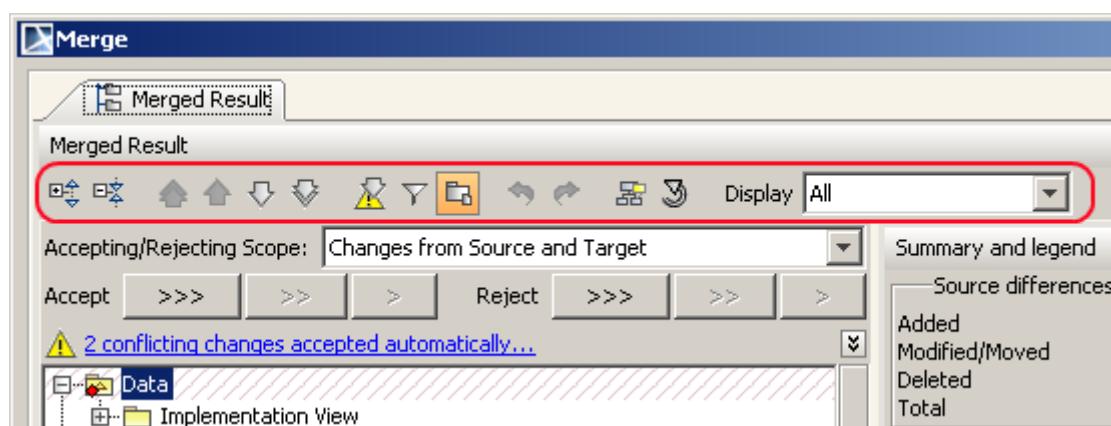


Figure 204 -- Toolbar for displaying and navigating through changes

The toolbar buttons are used to navigate through the **Merged Result** tree and the **Specification** panel:

| Button | Title | Function |
|--------|---------------------------------------|---|
| | Expand | Expand all nodes in the merged result tree. |
| | Collapse | Collapse all nodes in the merged result tree. |
| | Go To First Change Alt+Home | Select the first model change in the merged result tree. Note: The Go To First Change button is disabled if the first change is already selected in the merged result tree. |
| | Go To Previous Change Alt+up arrow | Select previous change in the merged result tree. |
| | Go To Next Change Alt + down arrow | Select the next change in the merged result tree. |
| | Go To Last Change Alt+End | Select the last difference in the merged result tree. |
| | Go To Next Conflict Alt+Page Down | Select the next conflict in the merged result tree. Note: The Go To Next Conflict button is disabled if there are no conflicts. For more information about navigation through conflicting changes, see “Buttons for quick navigation through conflicting changes” on page 330. |
| | Show auxiliary resources | Press the button to show or hide the profiles, modules with applied <<auxiliaryResources>> stereotype in the Merged Result tree (for example, UML Standard Profile). |
| | Undo | Undo the last action. |
| | Redo | Cancel undo command. |
| | Filter | Press the Filter button to invoke the Items Filter dialog box. Clear the checkbox next to the it to hide element in the merged result tree. |
| | Annotate Merged Diagram | Press this button to annotate diagram for the post-merge review. For more information, see “Annotations in the merged diagram” on page 339. |
| | Reset Windows Configuration | Press the Reset Windows button to organize panels of the Merge window to its original location. For more information about resetting windows configuration, see “Reorganizing panels of the Merge window” on page 319. |

| Button | Title | Function |
|---------|-------|--|
| Display | All | <p>List filter options for types of change in the merged result tree:</p> <ul style="list-style-type: none"> • All. This is the default value. Display all elements in the Merged Result tree. • All Changes. Display elements with changes or have inner changes. • All Conflicting Changes. Display elements that have conflicting changes or inner conflicting changes. • Addition Changes. Display elements that are added to the model or have inner addition changes. • Modification Changes. Display elements that have been modified or have inner modification changes. • Deletion Changes. Display elements that have been deleted or have inner deletion changes. • Move Changes. Display elements that have been moved or have inner move changes. • Order Changes. Display elements whose order has been changed or have inner order changes. |

Toolbar for accepting and rejecting changes

The toolbar located above the merged result tree is used for accepting and rejecting changes.

Accepting/Rejecting Scope drop-down box is devoted for choosing the contributors from which changes shall be accepted/rejected when using accepting/rejecting buttons and menu items.

Accept and reject buttons are described in the table bellow and more about accepting and reject you can see "Accepting or Rejecting changes" on page 316.

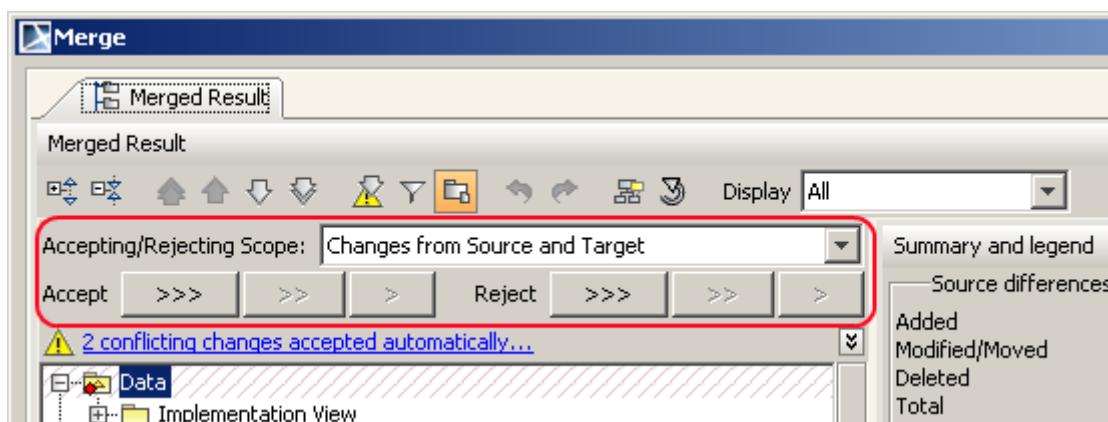


Figure 205 -- The toolbar for accepting and rejecting changes

| Button | Title | Function |
|--|---------------------------------------|---|
| Accepting/Rejecting Scope drop-down box | | |
| | Changes from Source and Target | Accept/reject changes from both - source and target projects. |
| | Changes from Source | Accept/reject changes from a source project. |

| Button | Title | Function |
|--------|----------------------------|--|
| | Changes from Target | Accept/reject changes from a target project. |

Buttons for accepting and rejecting changes

| | | |
|-----|--|---|
| >>> | Accept the selected change, its property changes and all subelement changes | Accept all changes from the left and right contributors starting from the selected element. E.g. if the <i>Data</i> model is selected, then all changes for the whole project will be accepted. |
| >> | Accept the selected change its property change | Perform the same function as the >>> button, except that only the selected change and its metaproPERTY changes are accepted. |
| > | Accept the selected change | Perform the same function as the >>> button, except that only the selected change is accepted. |
| >>> | Reject the selected change, its property changes and all subelement changes | Reject all changes from left and right contributor starting from the selected package. E.g. if the <i>Data</i> model is selected, then all changes for the whole project will be rejected. |
| >> | Reject the selected change its property change | Reject the selected change and its metaproPERTY changes. |
| > | Reject the selected change | Reject the selected element change. |

Change legend and summary

There is a legend on the right of the merged result for counting differences from ancestor and to display the meaning of colors that are used to mark changed elements.

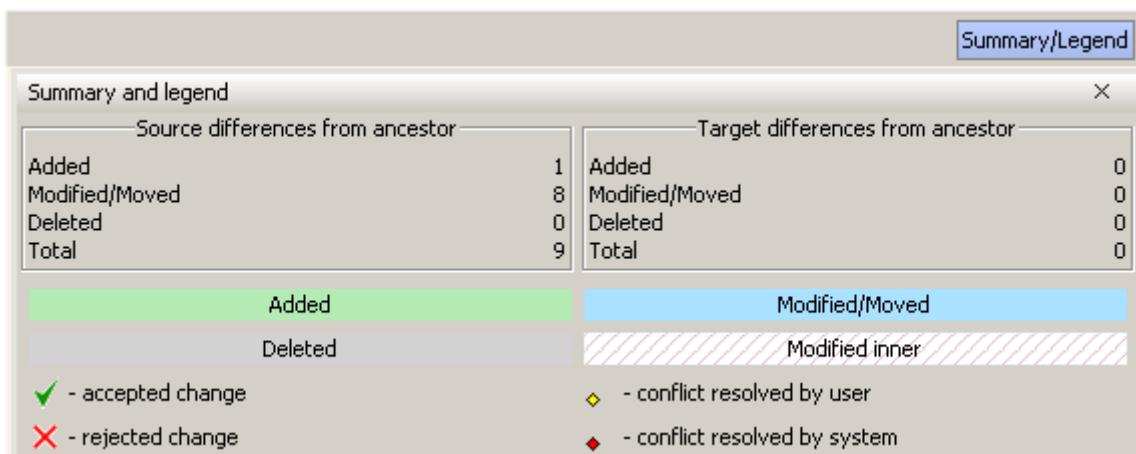


Figure 206 -- Change legend and summary

Press the **Summary/Legend** button at the top of the change legend to hide or show the summary panel.

Buttons for quick navigation through conflicting changes

Let's analyze the quick navigation through the automatically accepted conflicting changes in the **Merge** window (see Figure 207 on page 330).

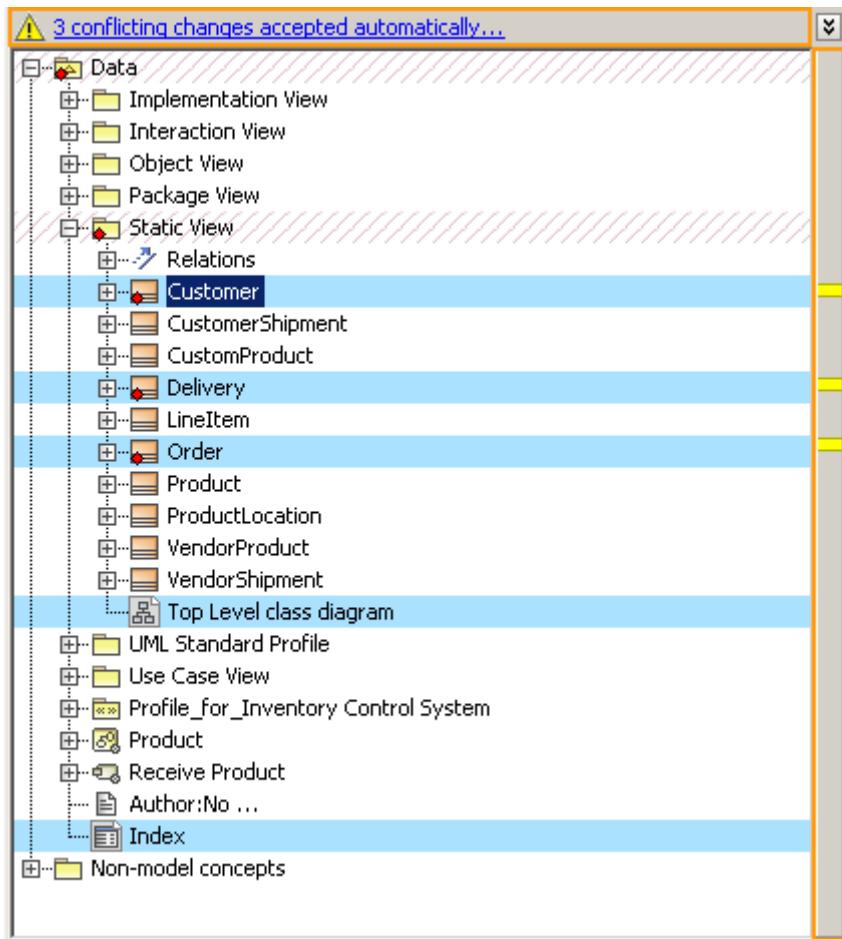


Figure 207 -- Buttons for quick navigation through conflicting changes

The panel to the right of the merged result tree displays yellow buttons representing conflicting changes. Press the yellow button to select the automatically accepted conflicting changes in the merged result tree.

A message with a warning icon is displayed in the **Merge** window stating the number of conflicting changes that are accepted from the target. This message is located above the **Merged Result** tree (see Figure 207 on page 330).

Element Specification panel

Modification changes can only occur in properties. Modification and move changes (which are a certain type of modification changes) are displayed at the bottom of the panel, the **Specification** panel (Figure 208 on page 331).

In the first column of the **Specification** panel, property titles are listed. The *Source*, *Ancestor* and *Target* columns display corresponding change properties from both contributors and the ancestor.

Changed properties have blue background. For example, Figure 208 on page 331 shows a class has different lists of values in source and target. Class is renamed in the source to *B* class and class is renamed to *C* in the target. Green tick in the *Target* column shows that class name change was accepted.

Additions in properties have green background, deletions grey have background (just like in the Merge Result tree) (see Figure 209 on page 331).

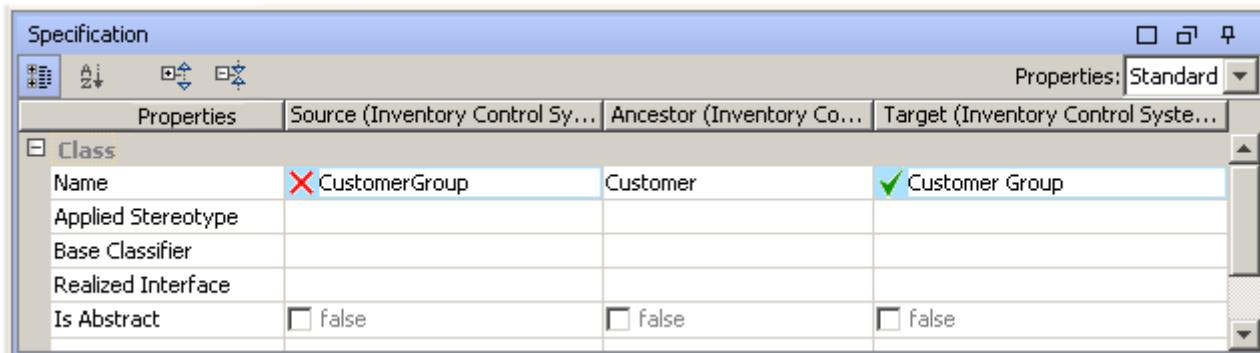


Figure 208 -- Displaying changes in element specification panel

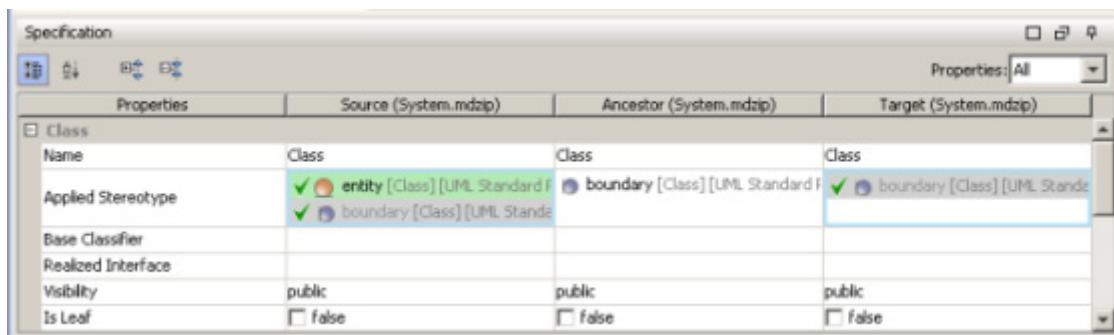


Figure 209 -- Displaying changes in the specification panel - value addition and deletion from a property

To navigate from the **Specification** panel to the **Merged Result** tree:

1. Select a property in the **Properties** column, which references other elements.
2. In the shortcut menu select the **Select in Merged Result Tree** command and then choose element to which you want to navigate (see Figure 210 on page 331).

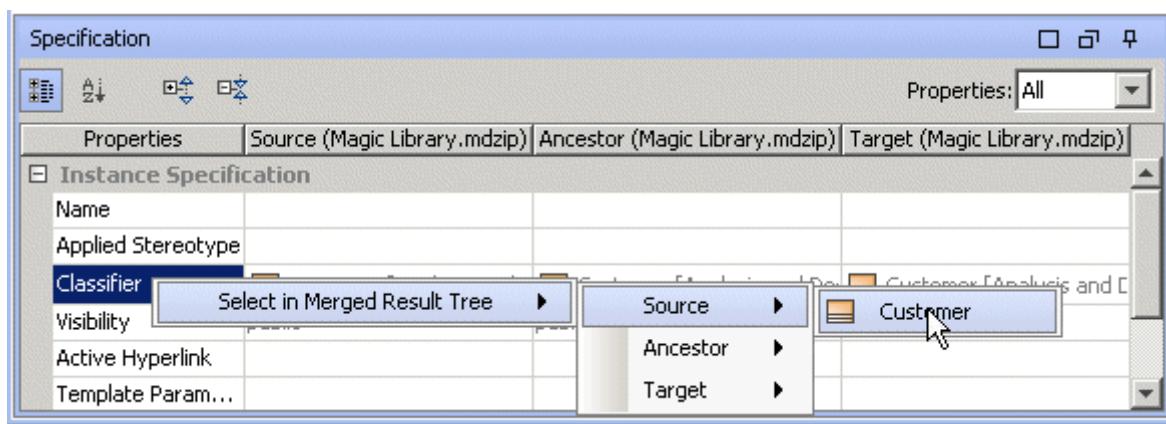


Figure 210 -- Navigating from the Specification panel to the Merged Result tree

You can also navigate to the **Merged Result** tree by invoking shortcut menu from the column cell, which references other elements. The **Select in Merged Result Tree** command has submenu items corresponding to the referenced elements for the selected cell.

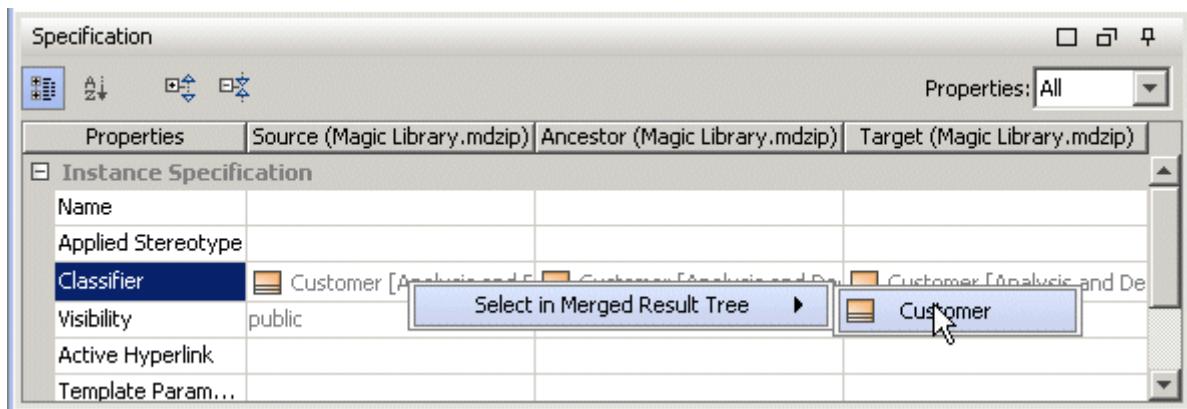


Figure 211 -- Navigating from the Specification panel to the Merged Result tree

Change details panel

The **Change details** panel is located at the bottom of the merge window. The **Change details** panel has a tree reflecting changes occurred on the element selected in the merged result tree or element properties panel. The tree has two root-level nodes:

1. Source changes
2. Target changes

The **Source changes** and **Target changes** nodes display changes occurred in the source or target respectively. If several nodes in the merged result tree are selected, then all changes occurred in the nodes are displayed in the **Change details** panel.

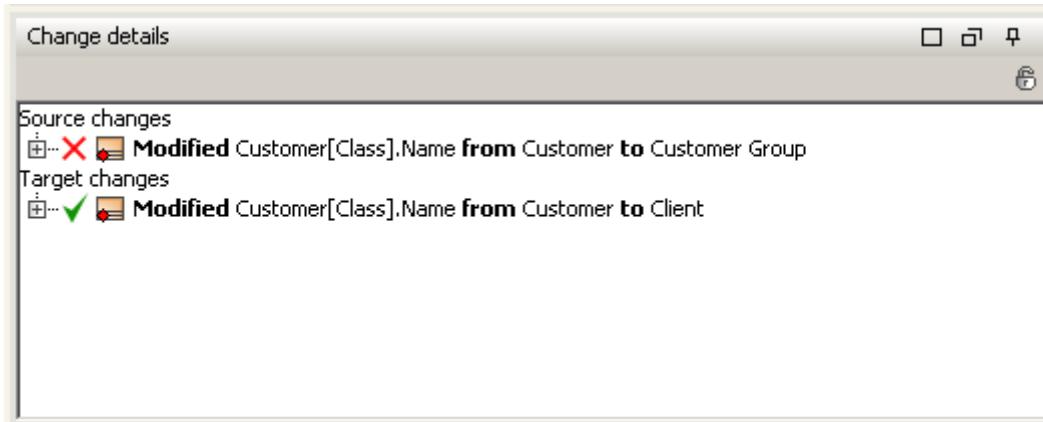


Figure 212 -- Change details panel

The **Change details** panel displays results after click on the element in the **Merged Result** tree or on property in the **Specification** panel.

Press the **Lock contents of this panel** button in the **Change details** panel to freeze the **Change details** results, that is, the last result is displayed and result is not refreshed (see Figure 213 on page 333). To unfreeze the **Change details** panel, press the **Lock contents of this panel** button again.

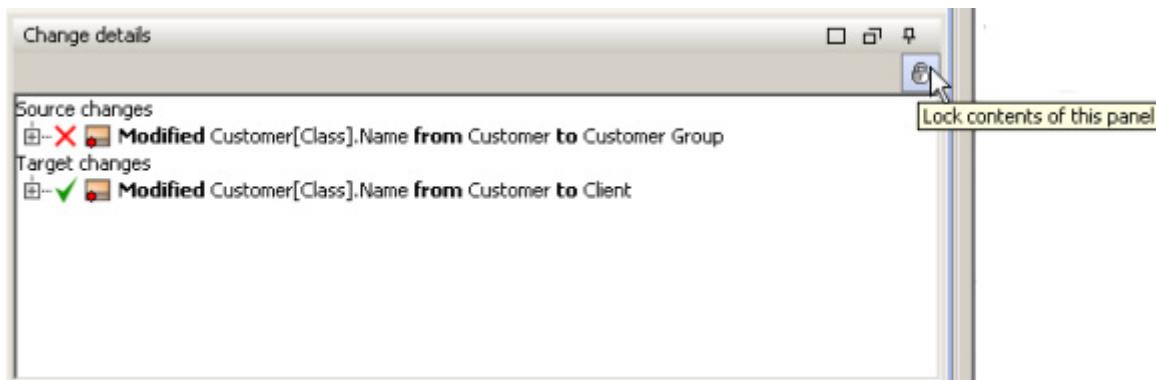


Figure 213 -- The Lock contents of this panel button in the Change details panel

Shortcut menu in the Merge window

The table below lists the commands of the shortcut menu, which is available in the **Merged Result** tree and **Change details** panel.

| Command Name | Description |
|---|--|
| Accept | Accepts the selected change. |
| Reject | Rejects the selected change. |
| Accept With Properties | Accepts the selected change and its property changes. |
| Reject With Properties | Rejects the selecting change, its property changes. |
| Accept Recursively | Accepts the selected change, its property changes, and all subelement changes. |
| Reject Recursively | Rejects the selected change, its property changes, and all subelement changes. |
| Mark as Resolved | The Mark As Resolved command makes the change resolved by user, not the system. The conflicting change is marked as resolved by user too. |
| Select in Merged Result Tree / Select in Specification panel | This menu item changes depending on the type of the change. |

Viewing changes in diagrams

The merge window has functionality to open merged diagrams from the source, ancestor, and target projects in separate tabs.

Opening the diagrams with changes from the Merge window

To open diagrams with changes double click on modified diagram in the **Merged Result** tree. Two or free views of the same diagram are opened. Ancestor view of diagram is opened always and source diagram is opened if

changes were done in diagram in source project, accordingly the target diagram is opened if changes were done in diagram in target project. See the Merge window with the opened diagrams (see Figure 214 on page 334).

You can switch between **Merged Result** window and diagram windows, by pressing tabs at the top of the **Merge** window.

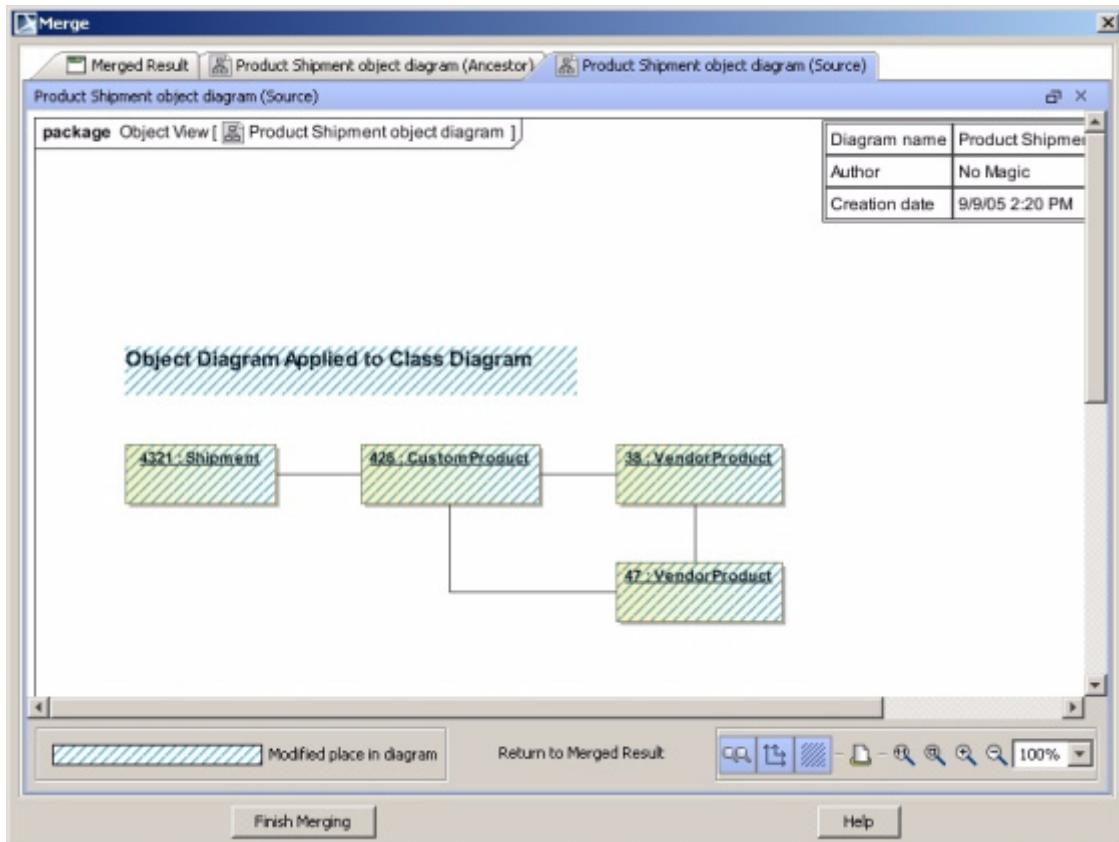


Figure 214 -- The Merge window with opened diagrams

Analyzing differences in diagrams

In opened diagrams changed diagram areas are highlighted with dashed blue background (see Figure 215 on page 335).

The following changes are highlighted in diagrams:

- The symbol or few symbols were moved or resized in diagram
- Then new symbol diagram was added
- The symbol was deleted from diagram

In summary all visual changes in diagram are highlighted.

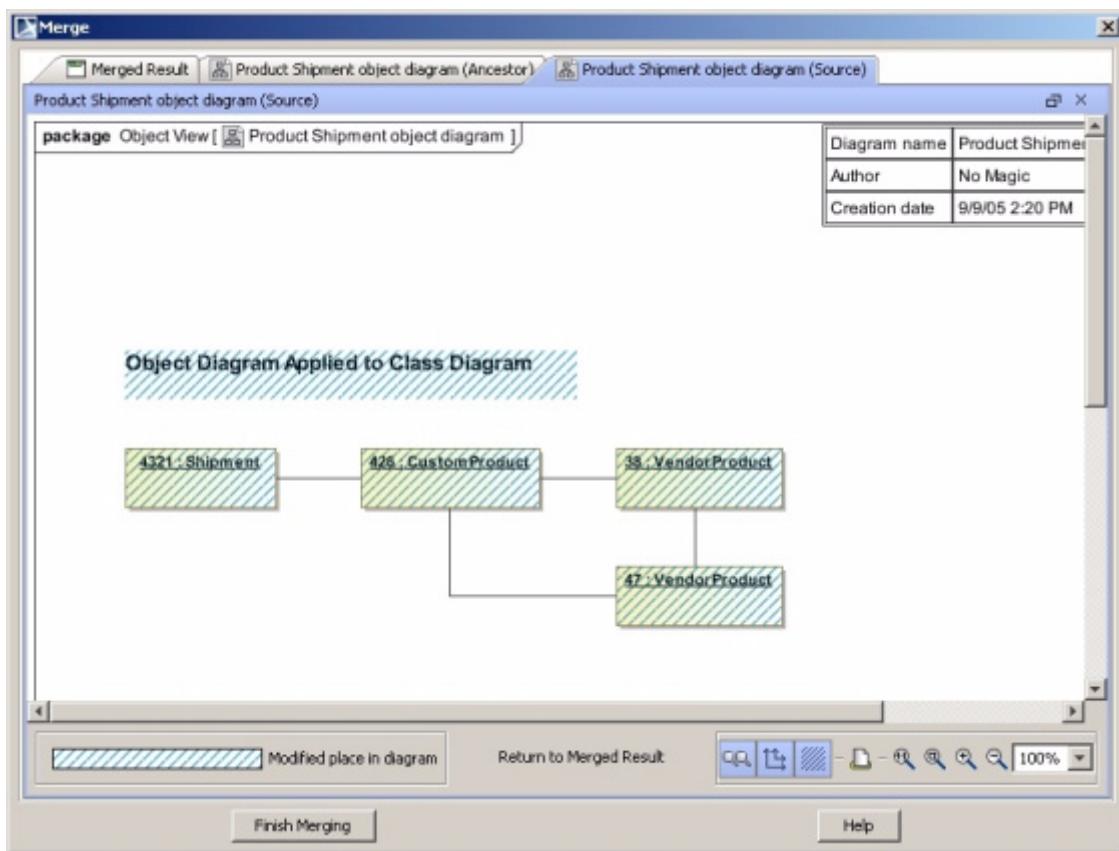


Figure 215 -- Diagram with highlighted differences

Managing the diagram view

Using the buttons located at the bottom of the diagram viewer tab (see Figure 216 on page 336) you can turn on or off difference showing, print a diagram, zoom, synchronize zooming and scrolling in diagram view. See the buttons description in the table bellow.

| Button image and title | Description |
|----------------------------------|---|
| Synchronize Zooming | <p>When button is pressed (default value) source, ancestor, and target diagrams zooming is dependent, i.e. the same zooming operations are performed on diagrams simultaneously.</p> <p>To turn off synchronization of zooming, depress the Synchronize Zooming.</p> <p>NOTE Synchronization is valid for the source, ancestor, and target views of the same diagram.</p> |
| Synchronize Scrolling | <p>When button is pressed (default value) source, ancestor, and target diagrams scrolling is dependent, i.e. the same scrolling operations are performed on diagrams simultaneously.</p> <p>To turn off synchronization of scrolling, depress the Synchronize Zooming.</p> <p>NOTE Synchronization is valid for the source, ancestor, and target views of the same diagram.</p> |

| Button image and title | Description |
|---|---|
| Mark Changes  | <p>Difference showing or hiding is controlled by the Mark Changes button.</p> <p>It is possible to press or depress this button in every diagram independently:</p> <ul style="list-style-type: none"> • If this button is pressed in the source diagram, then differences are shown in the source and ancestor diagrams. • If this button is pressed in the target diagram, then differences are shown in the target and ancestor diagrams. • If this button is pressed in the ancestor diagram, then differences are shown in the source, target, and ancestor diagrams. If buttons in the source and/or target diagrams were depressed previously, they are pressed after pressing this button in the ancestor diagram. <p>If difference showing is turned on for both source and target diagrams or it is turned on for the ancestor diagram, then area that highlights differences in the ancestor is combined from the areas that would be shown in the ancestor if only showing differences for source or target would be enabled.</p> |
| Print Diagram  | <p>To print a diagram, press the Print Diagram button. Diagrams are printed with highlighted area with the Mark Changes button is pressed (default value).</p> |
| Zooming  | <p>Resize the view of diagram by the zooming buttons. To synchronize or desynchronize zooming use the Synchronize Zooming button (see description above).</p> |

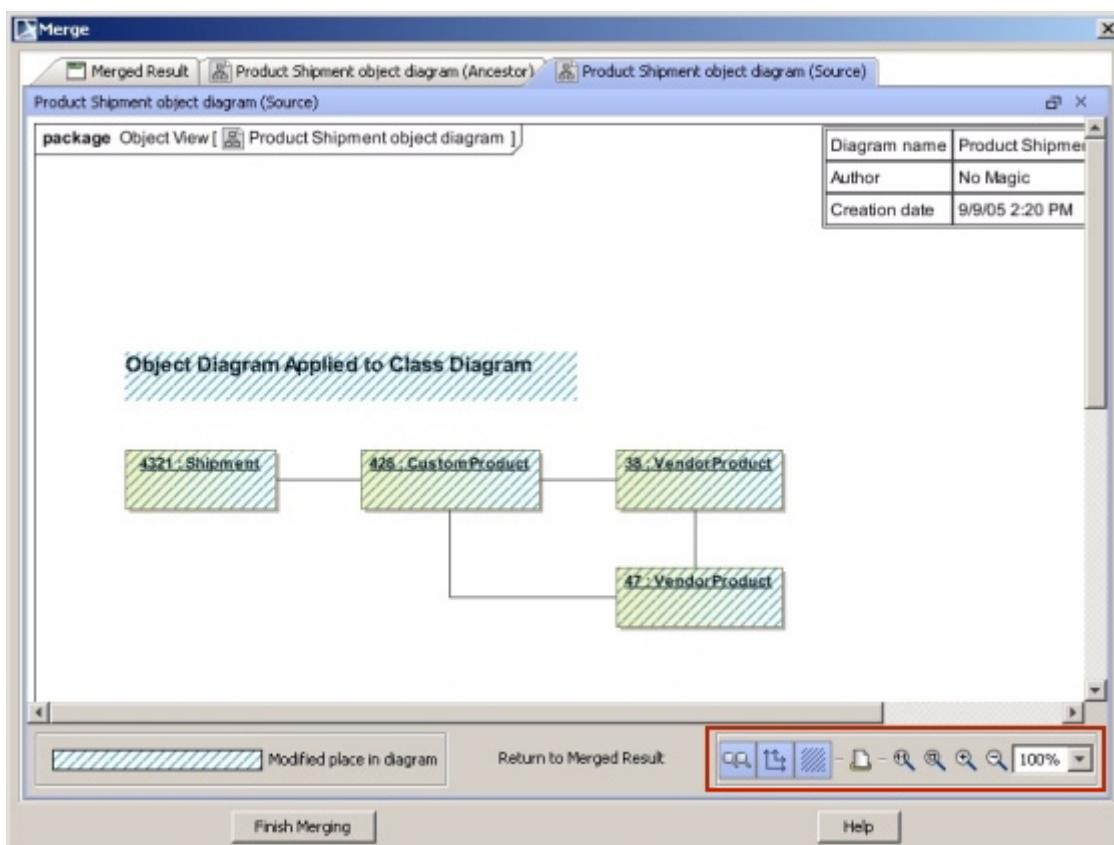


Figure 216 -- Buttons group in the diagram difference viewer

Symbol properties changes marking in the diagram difference viewer

Symbols, which are marked as changed in diagram difference viewer, but has no visual differences, may have symbol properties changes. It means that symbol properties there changes, but no affect on symbol was made.

See an example in Figure 217 on page 337 and in Figure 218 on page 338. In this example the *Shipment* class is displayed as changed in diagram, but it has no visual differences nor in Target neither in Source diagrams. The following symbol properties change was made to the *Shipment* class: in the class **Properties** dialog box, the **Show Stereotypes** option value was changed. Symbol properties changes were made for the class and they are detected in the diagram difference viewer in order to merge them correctly.

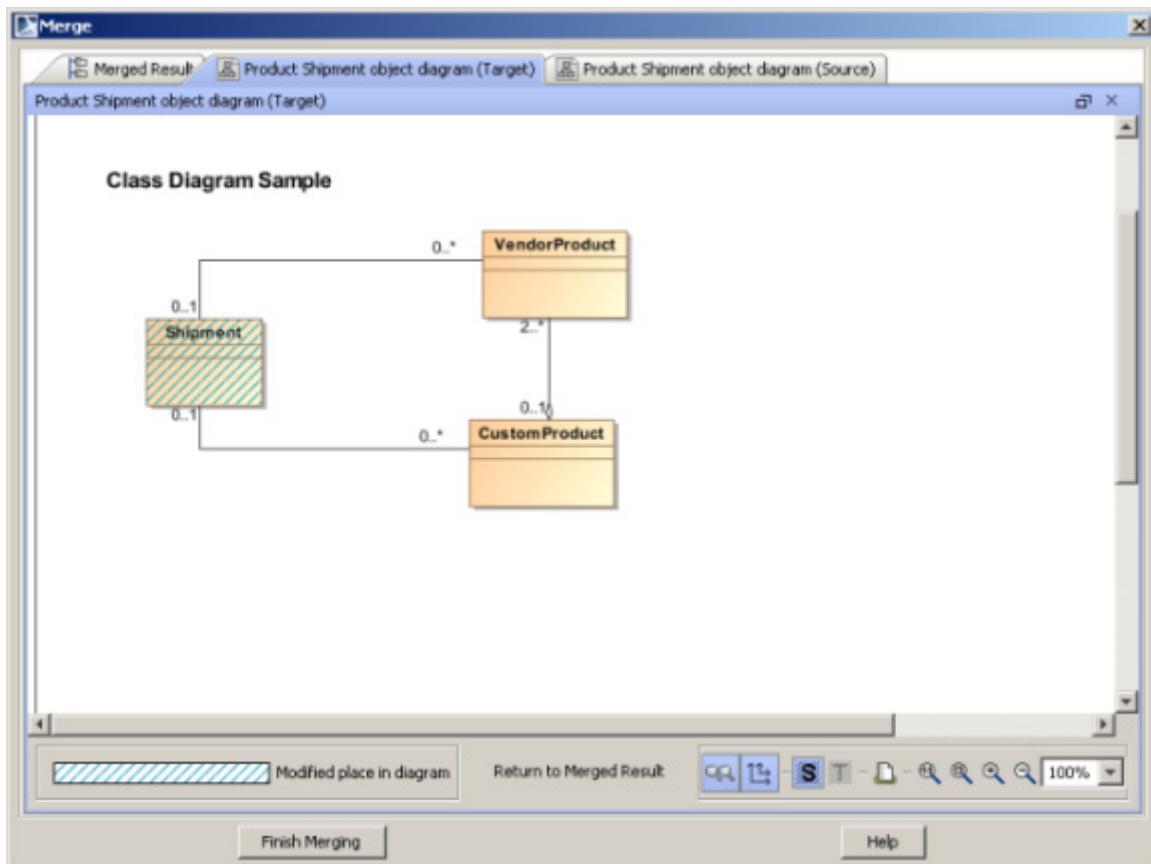


Figure 217 -- The diagram difference viewer window, displaying symbol properties changes in the Target diagram

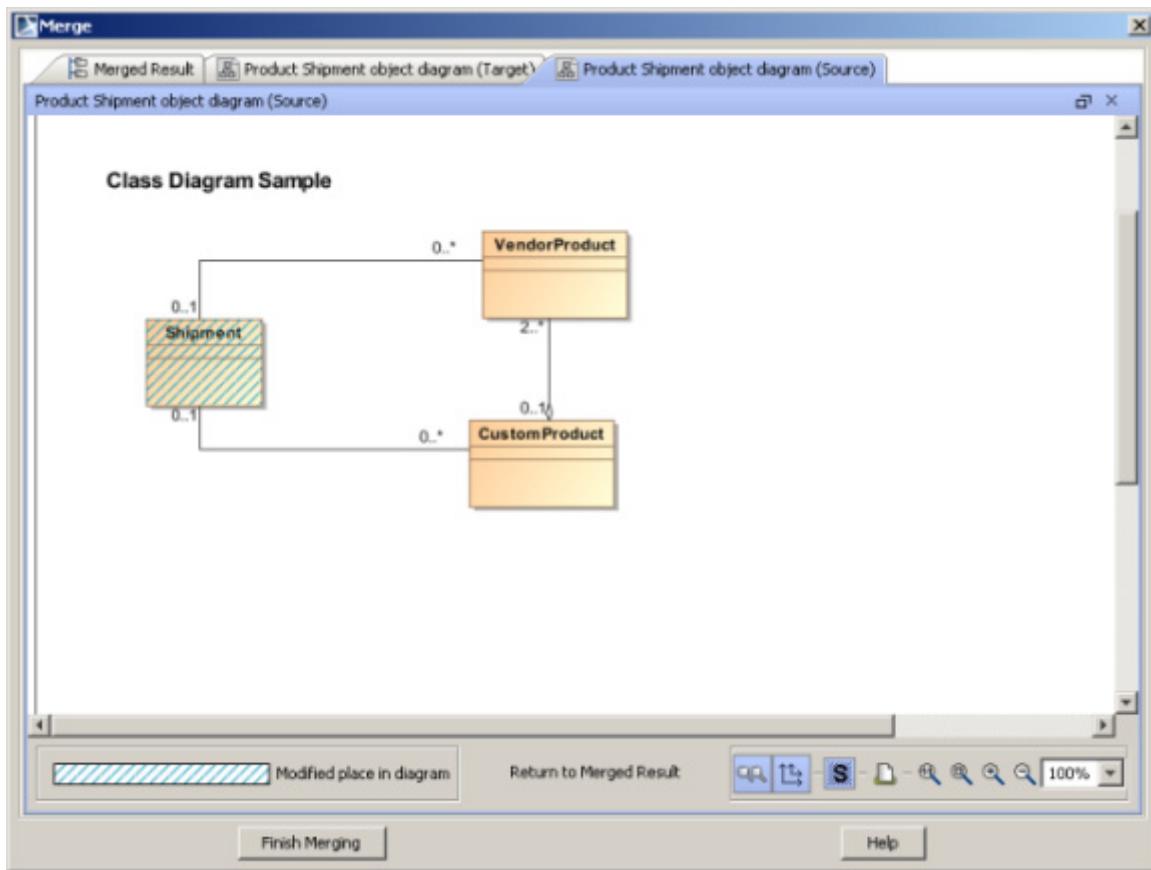


Figure 218 -- The diagram difference viewer window, displaying symbol properties changes in the Source diagram

Finishing projects merge

To finish the merge, in the **Merge** window, click the **Finish Merging** button. The question dialog box appears (see Figure 219 on page 338).

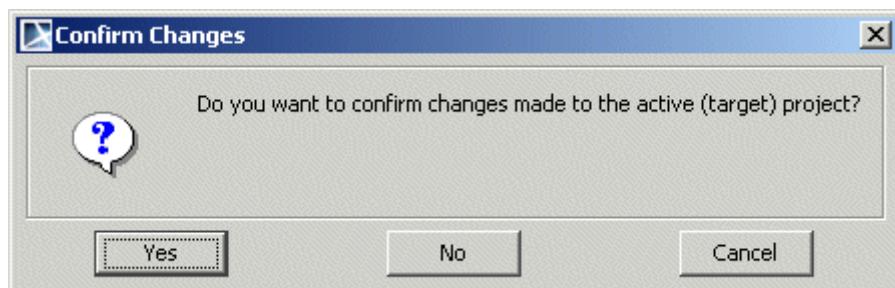


Figure 219 -- The Confirm Changes question dialog box

After pressing **Yes** in the **Confirm Changes** dialog box, project changes to the target project is confirmed.

After pressing **No**, the target project leaves not changed.

Pressing **Cancel** will cancel the **Confirm Changes** dialog box and you will be able to continue merging.

NOTE After the merge results are copied to the project, do not forget to save or to commit project to Teamwork Server.

Annotations in the merged diagram

The **Annotate merged diagram** button specifies whether merged diagrams will be annotated for the post-merge review or not. Press this button to annotate merged diagram. See the location of the button in Figure 220 on page 339.

See an example of the annotated diagram in Figure 221 on page 340.

Click the  warning at the right-bottom corner of the MagicDraw window to invoke the **Active Validation Results** window, there information about merged diagrams is represented (see Figure 222 on page 341). For more information, see “Validation” on page 444.

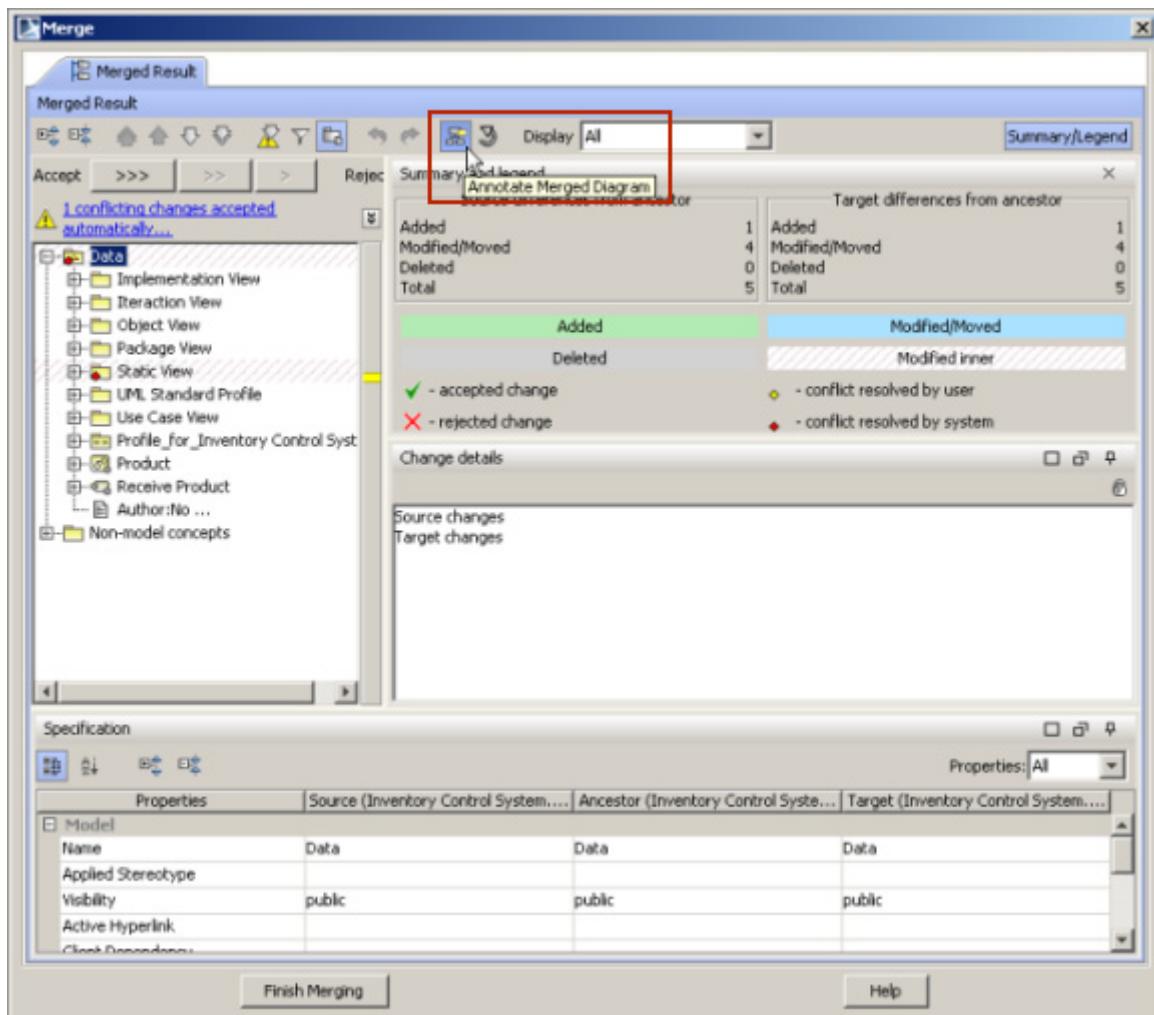


Figure 220 -- The Merge window, the Annotate Merged Diagram button

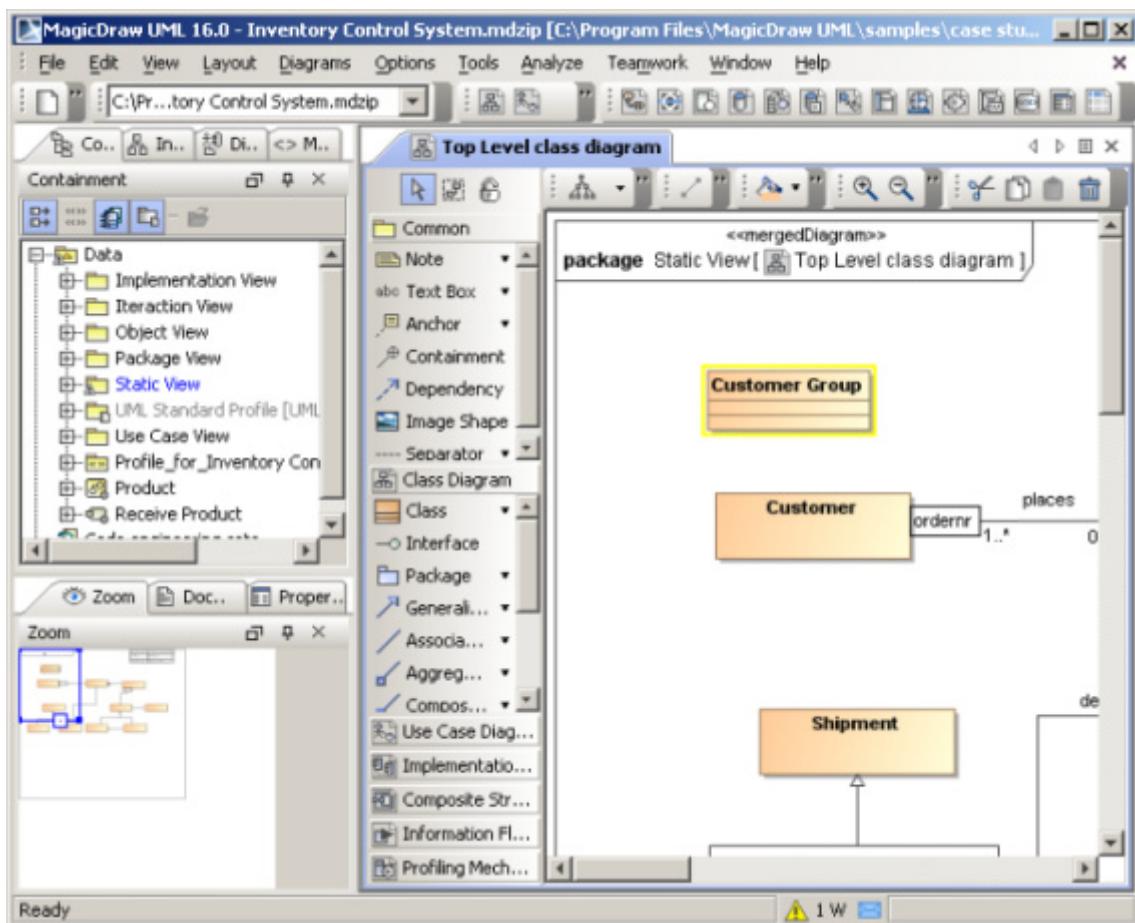


Figure 221 -- Annotations in the merged diagram

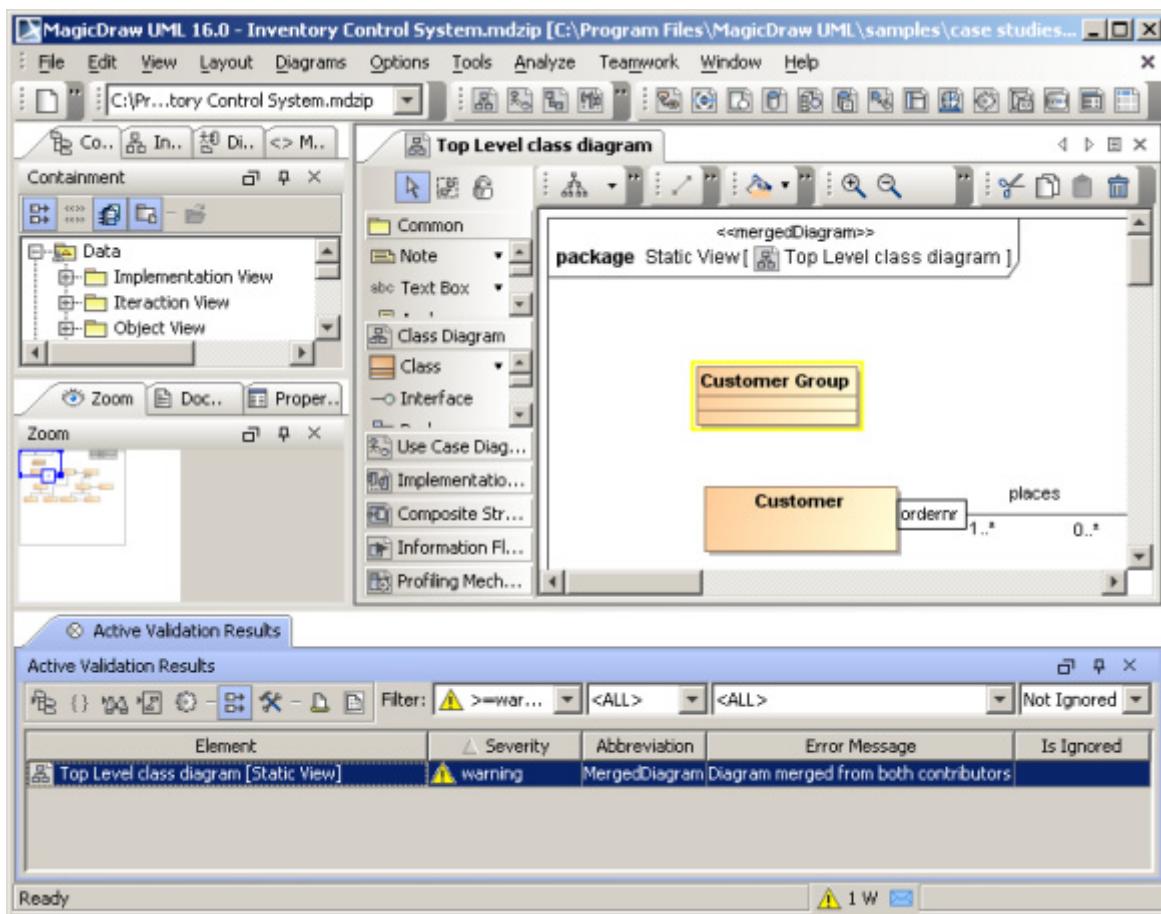


Figure 222 -- Validation results of the merged diagrams

Controlling Merge memory usage

You can customize merge memory usage by selecting the optimization option:

1. From the **Tools** main menu, select the **Project Merge** command. The **Projects Merge** dialog box opens (see Figure 223 on page 342).
2. Select the **Optimize for** option.

You can also specify the **Optimize for** option in the **Environment Options** dialog box, **General** pane, **Merge** group.

Select the **Speed** property to merge the projects faster, but it would require more memory.

If your PC doesn't have memory enough, in the **Optimize for** option, select the **Memory** property. Merge time will be slower, but memory usage will decrease.

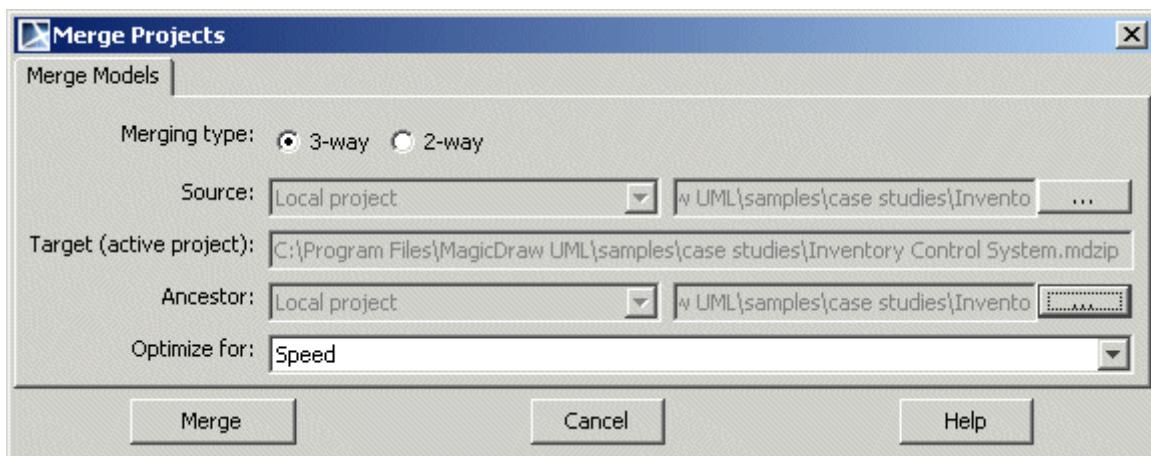


Figure 223 -- The Merge Projects dialog box

Pattern Wizard

In MagicDraw, you can find various GOF, Java, Junit, CORBA IDL, and XML Schema design patterns.

NOTE This functionality is available in Standard, Professional, and Enterprise editions only.

You can also create new patterns and edit existing ones using Java code or JPython scripts. For a detailed description, see [MagicDraw open API user's guide](#).

To open the **Pattern Wizard**

- Select **Tools** from the class shortcut menu and then select the **Apply Pattern** subcommand.

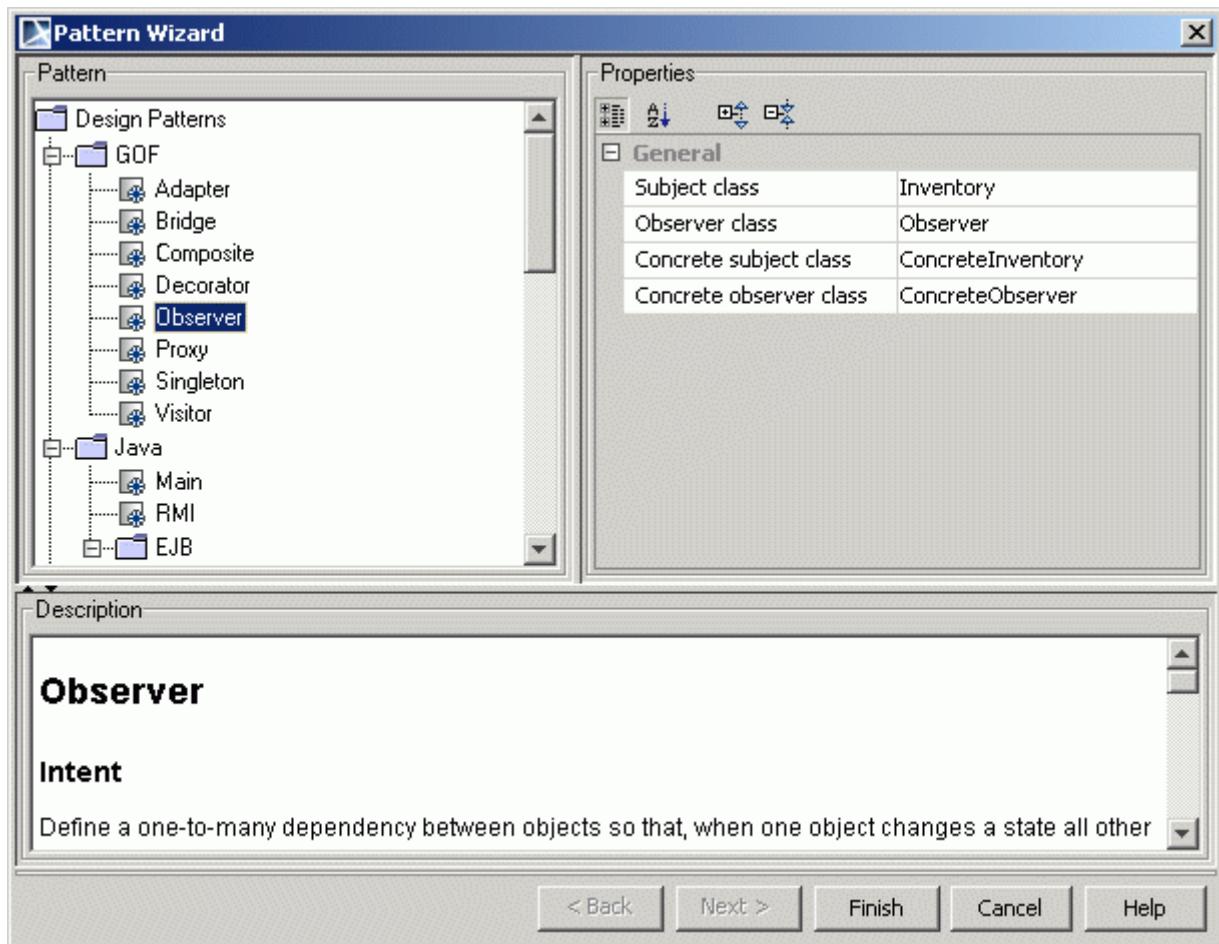


Figure 224 -- Pattern Wizard

The **Pattern Wizard** has three main collections of customizable options, which are represented by the hierarchy tree on the left side of the dialog box:

| Category | Design Pattern | Properties |
|---|------------------|--|
| GOF Templates described in the Design Patterns of Reusable Object-Oriented Software | Adapter | Interface Class Adapter Class Adaptee Class NOTE The Next > button is activated. In the Adapter Operations screen, add or remove operations you want to use. |
| | Bridge | Abstraction Implementor Implementor is: Abstract Class or Interface Name of Reference Suffix of the Concrete Implementor NOTE The Next > button is activated. In the Deriver Classes screen, add or remove classes you want to use. |
| | Composite | Component Class Composite Class NOTE The Next > button is activated. In the Composite Operations screen, add or remove operations you want to use. |
| | Decorator | Component Class Decorator Class Concrete Decorator Class NOTE The Next > button is activated. In the Decorator Operations screen, add or remove operations you want to use. |
| | Observer | Subject Class Observer Class Concrete Subject Class Concrete Observer Class |
| | Proxy | Subject Class Proxy Class Real Subject Class NOTE The Next > button is activated. In the Proxy Operations screen, add or remove operations you want to use. |
| | Singleton | Singleton class |
| | Visitor | Visitor Class Implementation Style: visit (MyClass), visit MyClass (MyClass) NOTE The Next > button is activated. In the Elements to Visit screen, select classes and interfaces you want to visit. |
| Java Specific Java design patterns | Main | Main class |

| Category | Design Pattern | Properties |
|----------|-----------------------|--|
| | RMI | Java RMI classes Remote Interface NOTE The Next > button is activated. In the Remote methods screen, select methods from the All list to the Selected list. |
| | EJB | |
| | Entity | Synchronize Names EJB Name EJB Class Remote Interface Home Interface Local Interface Local Home Interface Display Name Large Icon Small Icon Abstract Schema Name Cmp Version Persistence Type Reentrant Primary Key Class |
| | Message Driven | EJB Name EJB Class Display Name Large Icon Small Icon Acknowledge Mode Destination Type Subscribing Durability Message Selector Transaction Type |
| | Session | Synchronize Names EJB Name EJB Class Remote Interface Home Interface Local Interface Local Home Interface Display Name Large Icon Small Icon Session Type Transaction Type |
| | | NOTE The Next > button is activated. In the Remote Methods screen, add or remove operations you want to use. |

| Category | Design Pattern | Properties |
|--|----------------------------------|--|
| JUnit JUnit is a regression testing framework. It is used by the developer who implements unit tests in Java. JUnit is Open Source Software. The provided templates allow the user to create the constructions implemented in the JUnit framework. For more information, go to http://www.junit.org . | TestCase | TestCase Class Create suite() Create Constructor TestCase(String) Create runTest() Create setUp() Create tearDown() |
| | Tested Class | Tested Class TestCase Class Create suite() Create Constructor TestCase(String) Create runTest() Create setUp() Create tearDown() NOTE: The Next > button is activated. In the Tested Operations pane, add or remove operations you want to use. |
| XML Schema Specific XML design patterns | XSD complex Type | Target Class Content: XSDcomplex content, XSDsimple content. |
| | XSD compositor | Target Class Compositor: XSDall, XSDchoice, XSDsequence Particle |
| | XSD simpleType | Target Class Content: XSDrestriction, XSDlist, XSDunion |
| | XSD simpleType (XSDlist) | Target Class Item Type |
| | XSD simpleType (XSDunion) | Target Class |

| Category | Design Pattern | Properties |
|---|-------------------------------|--|
| | Simple XSD restriction | Target Class Stereotype: XSDsimpleType, XSDsimpleContent Base Min Exclusive Max Exclusive Max Inclusive Min Inclusive Total Digits Fraction Digits Length Min Length Max Length White Space Pattern |
| WSDL Specific WSDL design pattern | Binding | Use the Binding pattern when you want to create binding of some PortType. |
| CORBA IDL Specific CORBA IDL design patterns | Interface | Name Abstract Local Interface |
| | Value Type | Name Abstract Custom Value Type |
| | Type Definition | Name Type Definition Specifier: typedef, boxed value Base Type |
| | Sequence | Name Base Type Sequence Size Anonymous |
| | Array | Name Base Type Array Size Anonymous |
| | Fixed | Name Digits Scale Anonymous |
| | Union | Name Discriminator Type |
| | Enumeration | Name |
| | Struct | Name |
| | Exception | Name |

| Category | Design Pattern | Properties |
|------------------|----------------|---|
| < Back | | Go back to the Pattern screen. |
| Next > | | Go to the other appropriated screen. |
| Finish | | Finish and implement the wizard. The appropriated classes and interfaces are created. |
| Cancel | | Cancel the wizard without implementing your actions. |
| Help | | The MagicDraw Help is displayed. |

Creating Setters / Getters

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

Setters and getters are common operations that contain almost every class. With the help of MagicDraw UML, set and get operations for class attributes can be generated automatically.

To create a setter or getter

- From the shortcut menu of the selected class, select **Tools**, and then select **Create Setters/Getters**. The **Select Attributes/Association Ends** dialog box opens.
- Add a tagged value "getter/setter for attribute=attribute_name" to the selected class.

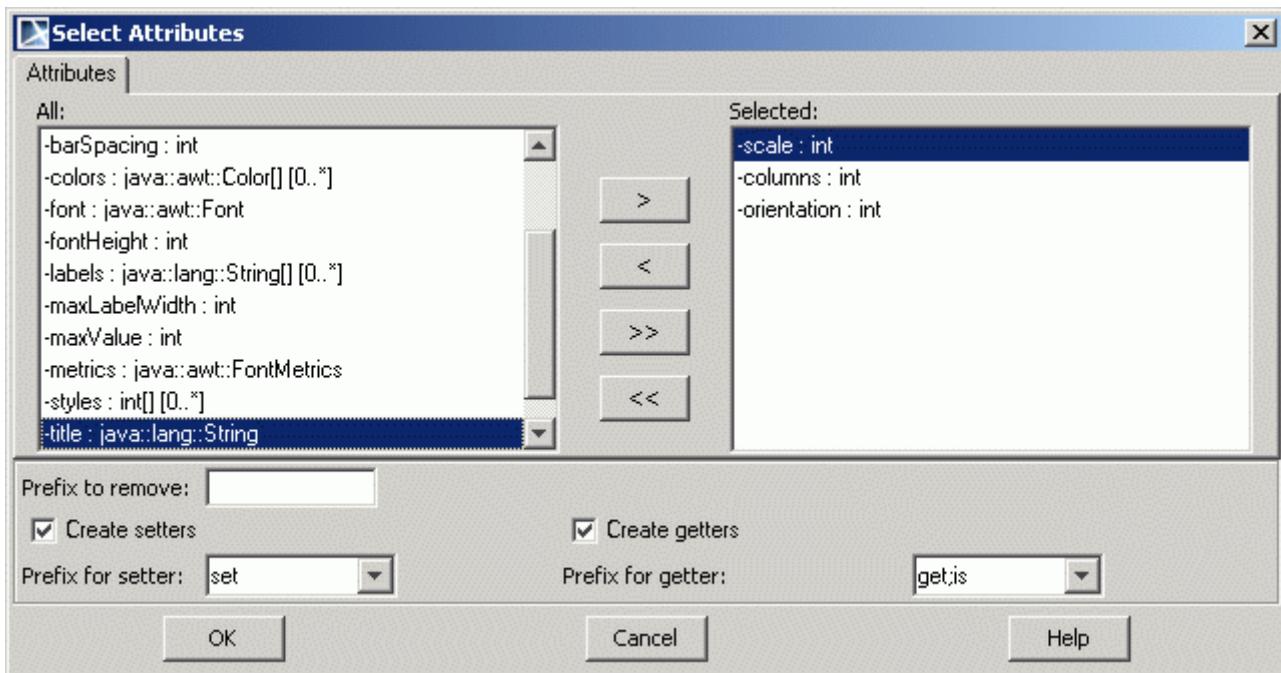


Figure 225 -- Select Attributes dialog box

| Box | Function |
|----------|---|
| All | Contains names of all attributes defined within the selected class. |
| Selected | Contains the selected attributes. |

7 TOOLS

Implementing or Overriding Operations

| Box | Function |
|--------------------------|---|
| > | Moves the selected attribute from the All list to the Selected list. Setter for that attribute will be generated. |
| < | Moves selected attribute from the Selected list to the All list. |
| >> | Moves all attributes from the All list to the Selected list. Setters for all attributes will be generated. |
| << | Moves all attributes from the Selected to the All list. |
| Prefix to Remove | Type a prefix of an attribute (-,) you want to remove while generating setters or getters. |
| Create Setters | Generates setters for the selected attributes. |
| Create Getters | Generates getters for the selected attributes. |
| Prefix for Setter | Select a prefix for the generated setter (operation). Possible choices: set or Set. |
| Prefix for Getter | Select a prefix for the generated getter (operation). Possible choices: get;is, Get;Is; get, or Get. NOTE "Get" is used for every getter, "is" is used if the type of an attribute is set as Boolean. |
| OK | Generates setters and/or getters for attributes that are in the Selected Items list. |
| Cancel | Exits the dialog box without any changes. |
| Help | Displays the MagicDraw Help. |

The names of created operations (setters) are combined according to the following format:

```
public void set + <attribute name> (<attribute type> <attribute name>)
```

For example, if you have an attribute called x of type int, then the generated setter will look this way:

```
public void setx (int x)
```

The names of created operations (getters) are combined according to the following format:

```
public <attribute type> get + <attribute name> ( )
```

For example, if you have an attribute called x of type int, then the generated setter will look this way:

```
public int getx ();
```

Implementing or Overriding Operations

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

When you inherit classes from the base class that has abstract functions, you have to redefine them in the inherited classes. The implement/override operations tool will help you generate operations that are defined as abstract in the base class.

The **Implement/Override Operations** command can be invoked in 2 cases:

- When one classifier inherits operations from the base classifier (Generalization relationship).
- When some classifiers implement an Interface (Realization relationship).

To start the **Implement/Override Operations** tool

From the shortcut menu of the selected class, select **Tools**. Then, select **Implement/Override Operations**. The **Select Operations to Implement/Override** dialog box opens.

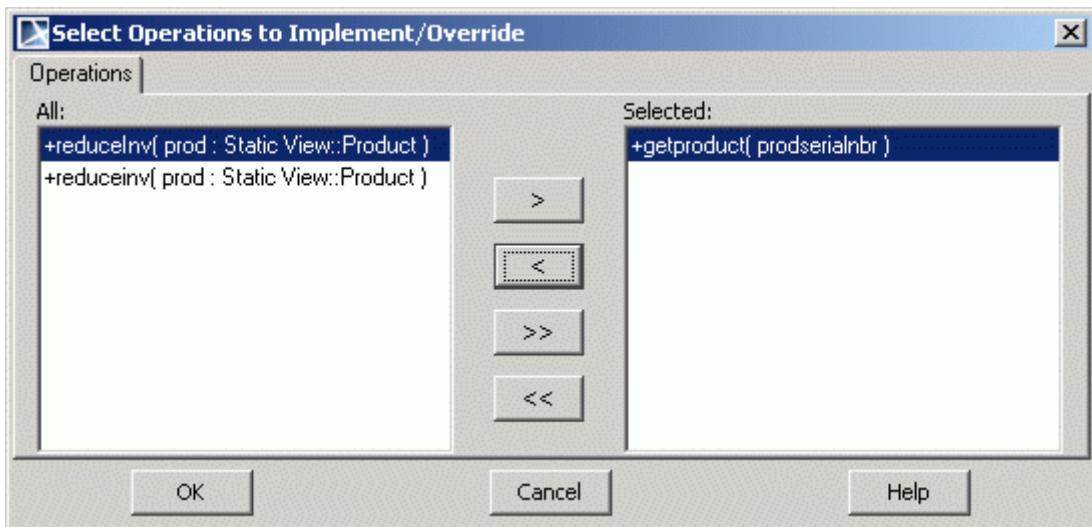


Figure 226 -- Select Operations to Implement/Override dialog box

| Box | Function |
|-----------------|--|
| All | Contains names of all operations defined within the selected class. |
| Selected | Contains the selected operations. |
| > | Moves the selected operation from the All list to the Selected list. |
| < | Moves selected operation from the Selected list to the All list. |
| >> | Moves all operations from the All list to the Selected list. |
| << | Moves all operations from the Selected to the All list. |
| OK | Generates operations that are in the Selected list. |
| Cancel | Exits the dialog box without any changes. |
| Help | Displays the MagicDraw Help. |

| | |
|-------------|---|
| TIP! | Double-click the item name and it will be moved to the opposite list. |
|-------------|---|

Model Transformation Wizard

| | |
|-------------------------|--|
| View Online Demo | Transformations |
| NOTE | This functionality is available in the Architect and Enterprise editions only. Users, who have the Cameo Data Modeler plugin, can use the transformation engine to run ER to Generic/ Oracle DDL transformations, even if they do not have the MagicDraw editions mentioned above. |

The Model Transformation Wizard enables running one of the predefined transformations. When using this wizard to run a chosen transformation, you have to perform the following steps:

1. Choose a transformation type.
2. Specify both the transformation source model (or a part of it) and destination package.

3. Select a type map.
4. Set custom transformation properties.

Each transformation converts the chosen model part into another model according to the rules of this transformation. Transformations are usually used for converting models between different modeling domains, e.g., UML and DDL.

All transformations follow a similar approach. They take a part of a model as the transformation source and copy it to the destination model, establishing traces between the transformation source and target elements. Then each transformation performs the specific model reorganizations, which are necessary for each transformation type according to the transformation options specified by the user in the transformation wizard. Transformation can also be performed in-place, i.e., the source model is not copied to the destination model, but transformation works directly on it instead.

Transformations also perform the so-called type remapping. During the transformation between the different modeling domains, such as UML and DDL, it is necessary to go through the data types used in the source model and change the types from the source domain into the equivalent types in the target domain, for example, changing String type usages in the UML model into the varchar type usages in the DDL model.

Available Transformations

Transformations are usually used for converting models between different modeling domains. Transformations are named by the types of their source models and their destination models. These are the available transformations:

- **Any to Any.** This transformation copies all your model or part of it to another package without making any changes. You can also remap types in the destination model by applying some type mapping rules.
- **UML to XML Schema.** This transformation creates XML Schema models from pure UML models. Once the source model is copied, the necessary stereotypes are applied according to the XML schema modeling rules.
- **XML Schema to UML.** This creates pure UML models from the XML Schema models. The source model is copied but stripped of all unnecessary stereotypes. As this transformation derives the more abstract, platform-independent models from the platform-specific ones, reverse transformation retains less information in the destination model than is available in the source.
- **UML to Generic DDL.** This creates generic DDL models from UML models. Once the source elements are copied to the destination, the appropriate stereotypes are applied and the model is reorganized. Type mapping is performed using the appropriate Generic DDL types.
- **UML to Oracle DDL.** This creates Oracle DDL models from UML models. Type mapping is performed using the appropriate Oracle DDL types. This transformation is very similar to the UML to Generic DDL transformation.
- **Generic DDL to Oracle DDL.** Performs transformations between two flavors of DDL models.
- **DDL to UML.** Derives UML models from DDL models (both Generic and Oracle).
- **Profile Migration.** Helper transformation for migrating models using one profile to models using another profile (usually - between different versions of the same profile - old and new)..

NOTE

Additional custom types of transformations can be defined by MagicDraw plugins.

Working with Model Transformation Wizard

The wizard can be opened from several places.

To start the **Model Transformation Wizard**

Do either:

- From the **Tools** menu, choose **Model Transformations**.
- Right-click one or more packages and select **Tools > Transform**.

Selecting transformation type

In the first step of the Model Transformation Wizard, a list of the available transformation types is displayed.

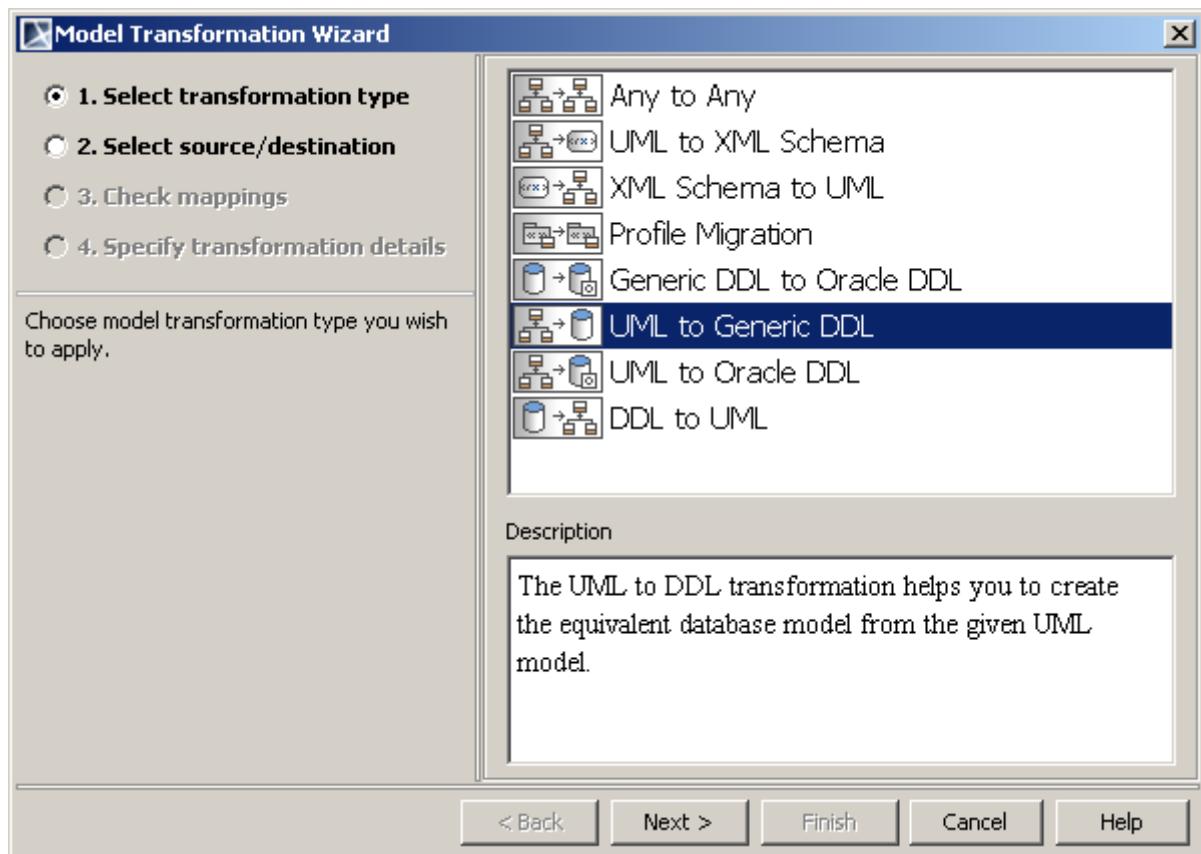


Figure 227 -- Model Transformation Wizard. Select transformation type

Transformation types are displayed in the list window.

The following operations are available in the **Select transformation type** window:

| Button | Function |
|------------------|---|
| Next > | Proceed to the next step (in this case, Select source/destination). |
| Cancel | Cancel the wizard. |
| Help | Display the MagicDraw Help. |

Selecting source and destination models

In the second step of the wizard, the **Transformation Source** tree displays all project data, i.e. the packages and their inner elements, that can be selected as a transformation source. Transformation will take the selected elements as input data.

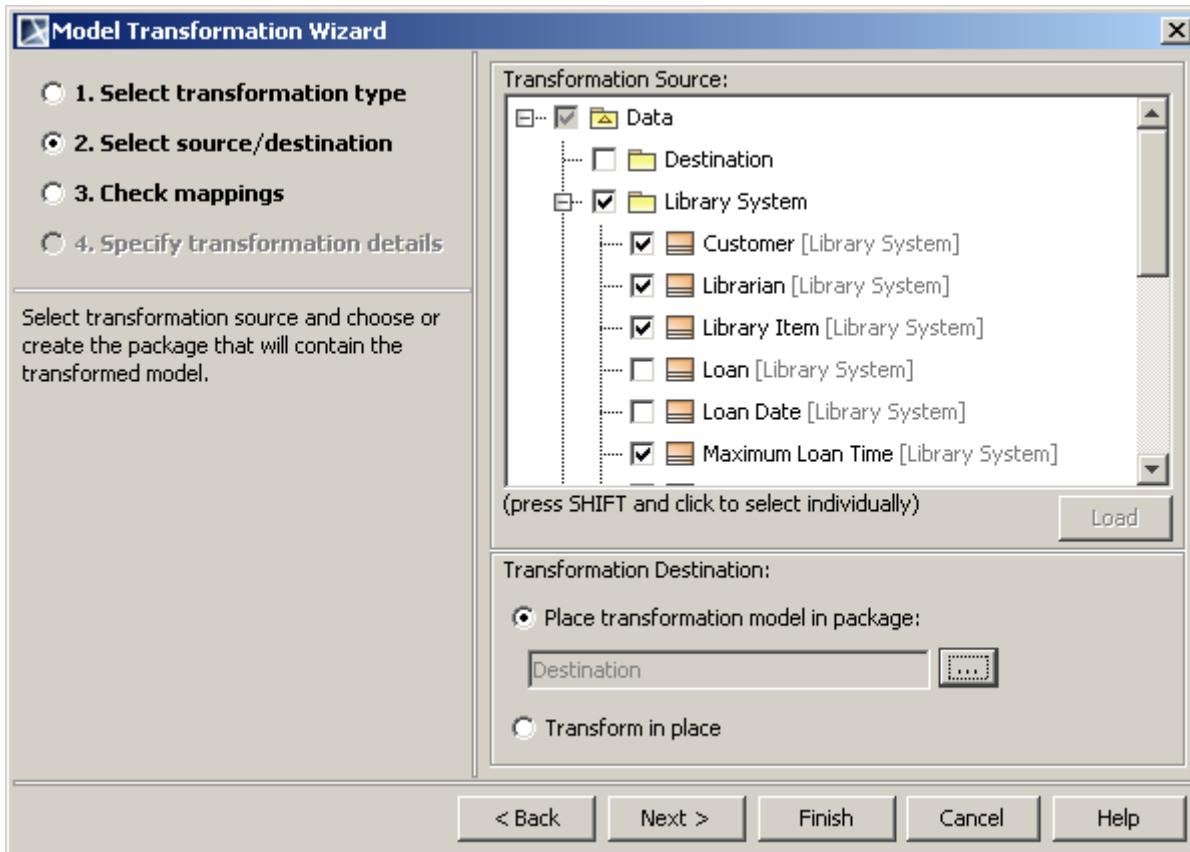


Figure 228 -- Model Transformation Wizard. Select source/destination

Select the **Place transformation model in package** option button to specify the package into which the source will be transformed. Click the “...” button to display the **Destination Package** dialog. Select an existing package from the **Packages** tree or create a new one.

Select the **Transform in place** option button, if you want the source model to be edited.

IMPORTANT! If you choose the in-place transformation, the model part selected as the transformation source, will be edited directly, and you will not retain your original model. So, please, be careful with this option.

If you choose the destination package, the source model will be copied to it and the transformation will be performed on this copy. Hence you will retain your source model and get a resulting model and traces will be established between elements in these model parts.

The following operations are available in the **Select source/destination** window:

| Button | Function |
|--------|---|
| < Back | Return to the previous dialog box. |
| Next > | Proceed to the next step (in this case, Select type mappings). |
| Finish | Finish the transformation configuration. All other options will be set by default. The Model Transformations Wizard exits and transformation results appear in the project. |

| Button | Function |
|---------------|-----------------------------|
| Cancel | Cancel the wizard. |
| Help | Display the MagicDraw Help. |

Selecting type mappings

The third step in the wizard allows for selecting a type map that will be applied during the transformation. Usually transformation has and brings in some predefined type map, but if you want, you can specify another type map.

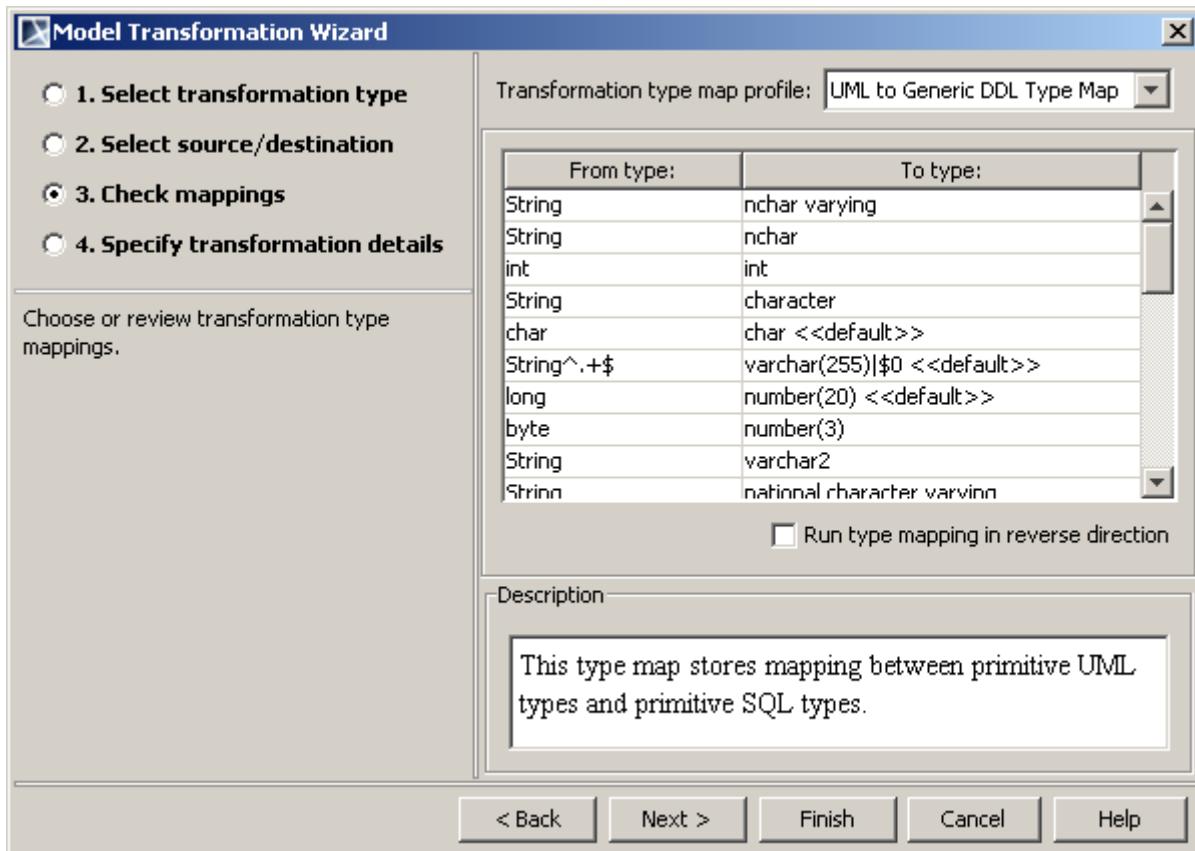


Figure 229 -- Model Transformation Wizard. Select type mappings

A type map can be regarded as a collection of rules of the form “Replace the usage of type X in the module with the usage of type Y”.

A type map is a model object, i.e. a package with a collection of dependencies (for the details about modeling type maps, see section “Transformation Type Mapping” on page 356), hence all model manipulation operations can be performed on it. In particular, it can be refactored into a module and mounted into any project, which needs it. It can be a simple package in your project as well, if you need a custom, one-off type map. A pre-defined type map can be taken from the MagicDraw module and edited.

To see a list of the type maps available in your model, click the down arrow in the **Transformation type map** combo box. These type maps specify the mapping rules that will be applied to the model during the transformation.

When you select a particular map, its contents are displayed in a table below. Each row in the table is a rule to remap one particular type to another. The **From type** and **To type** columns in the table show the source and target types.

The **Run type mapping in reverse order** check box creates the opposite type mapping. Type maps can be bidirectional, e.g., the same type map is reused both in the UML to XML schema and XML schema to UML transformations. This checkbox governs the direction in which the type map should be used.

The following operations are available in the **Select type mapping** window:

| Button | Function |
|------------------|--|
| < Back | Return to the previous dialog box. |
| Next > | Proceed to the next step (in this case, Specify transformation details). NOTE: This button is disabled during the Any to Any transformations. |
| Finish | Finish the configuration of the transformation. The Model Transformations Wizard exits and the transformation results appear in the project. |

Setting custom transformation properties

The **Transformation Details** table displays the various properties of a specific transformation, selected in the first step of the **Model Transformation Wizard**. Each transformation type has its own set of options, which govern functionality of that transformation.

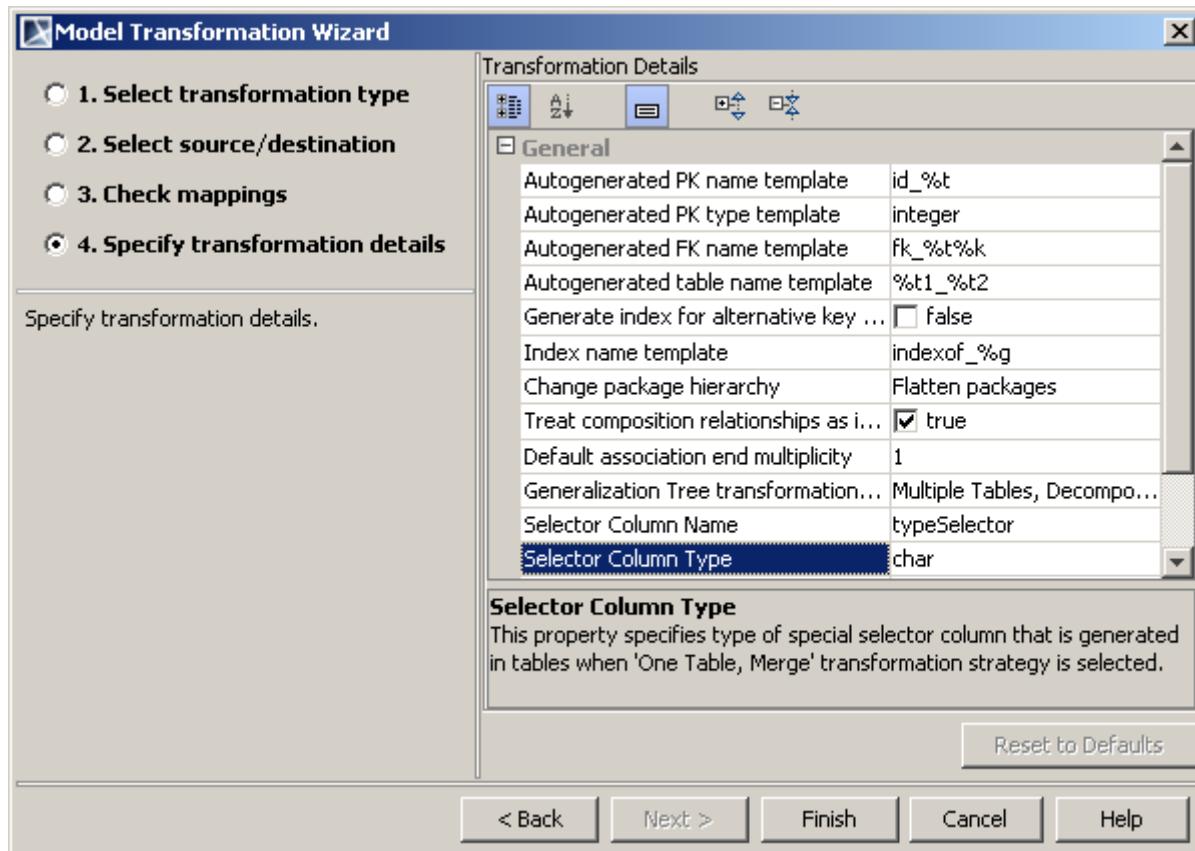


Figure 230 -- Model Transformation Wizard. Specify transformation details

To change the transformation properties to the default values, click **Reset to Defaults**.

The following operations are available in the **Specify transformation details** window:

| Button | Function |
|--------|------------------------------------|
| < Back | Return to the previous dialog box. |

| Button | Function |
|---------------|--|
| Finish | Finish the transformation configuration. The Model Transformations Wizard exits and the transformation results appear in the project. |

Transformation Type Mapping

During the transformation between the different modeling domains, such as UML and DDL, it is necessary to go through data types used in the source model and change the types from the source domain into the equivalent types in the target domain, for example, changing String type usages in the UML model into varchar type usages in the DDL model. This is achieved by establishing a type map and then supplying it for the transformation (many transformations have default, predefined type maps).

A type map can be regarded as a collection of rules of the form “Replace the usage of type X with the usage of type Y”. A type map is created by modeling means and is a model object, hence all model manipulation operations can be performed on it. In particular - it can be refactored into a module and mounted into any project, that needs it. It can be a simple package in your project as well, if you need a custom, one-off type map. Predefined type map can be taken from the MagicDraw module and edited.

A type map is a stereotyped package, holding a collection of stereotyped dependencies. Stereotypes for building type maps are stored in the *Model Transformation Profile*.

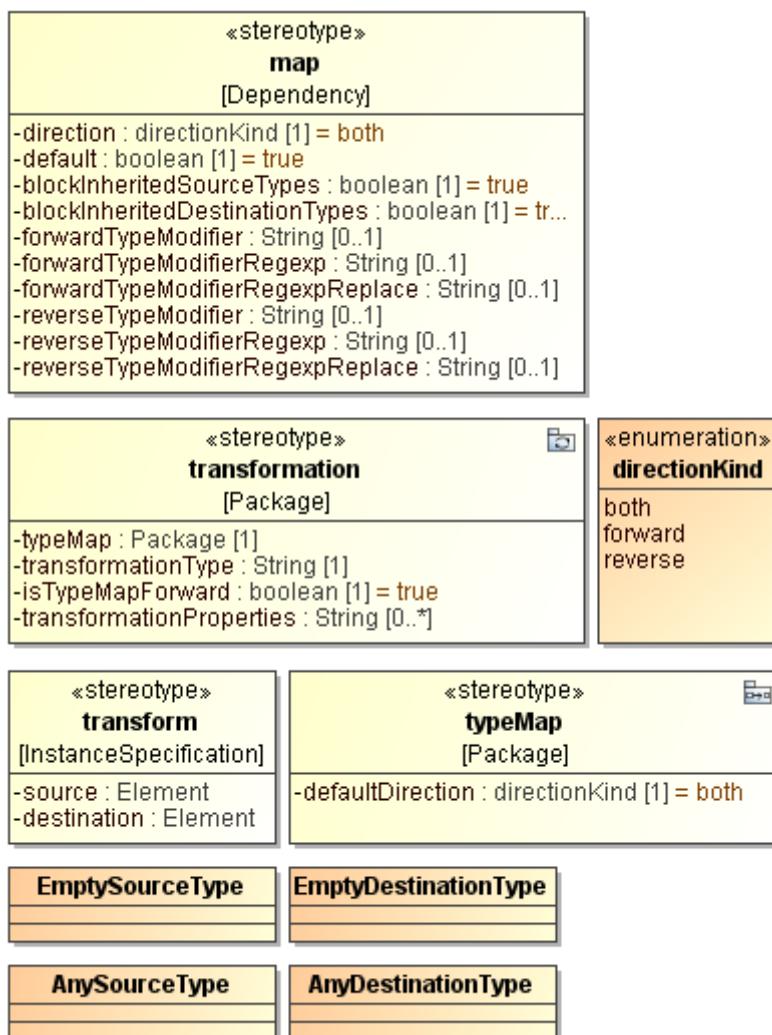
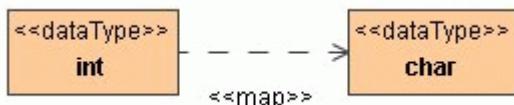


Figure 231 -- Stereotypes used for creating mapping rules

To create a transformation type map

1. Use or import *Model_Transformation_Profile.xml.zip*.
2. Create a package, which will represent your type map. Apply a stereotype <<typeMap>> to it.
3. Choose types (data types, classes, enumerations) in your source domain and their corresponding types in your target domain. Create the desired dependency relationships between the corresponding types. Apply a stereotype named <<map>> to these dependencies.



IMPORTANT! Be sure to place dependencies in the type map package (MagicDraw is prone to placing dependencies in or near the dependent model element, so you may need to relocate them)

In the example above, after the transformation, all *int* types will be transformed to *char*.

Each of the thus created dependencies represents one type remapping rule. The package represents the complete type map.

Type mapping rule behavior can be further customized by setting various tags on the rules (see “Controlling Type Mapping Rule Behavior” on page 357).

NOTE Transitive type mapping (of the form type1->type2->type3) is not supported.

Controlling Type Mapping Rule Behavior

There are several different tags that can be set on a type map or an individual rule to change its behavior slightly.

Controlling direction

By default, the same type map can be applied in two directions: forward and backward. The backward direction can be set by selecting the **Run type mapping in reverse order** check box in the third step of the Model Transformation Wizard. This is useful, when there are two related opposite transformations for some domain; for example, the same type map is used for both UML to XML schema and XML schema to UML transformations.

If you want to limit the directions, in which type map can be used, you can set the **defaultDirection** tag for your type map package. Possible values are **forward**, **reverse**, and **both** (default).

The direction can also be limited on a per-rule basis. This is controlled by setting the **direction** tag on the type map dependency. Then the mapping rule is excluded from the rule set when the type map is run in a different direction than specified for this rule.

Multiple rules for the same source type

There can be multiple mapping rules for the same source type. For example, String -> varchar and String -> nvarchar. In this case, one of the rules must be marked as default by setting the **default** tag value on it to true.

IMPORTANT! The type map having several rules for the same type and without any one set as default cannot be used.

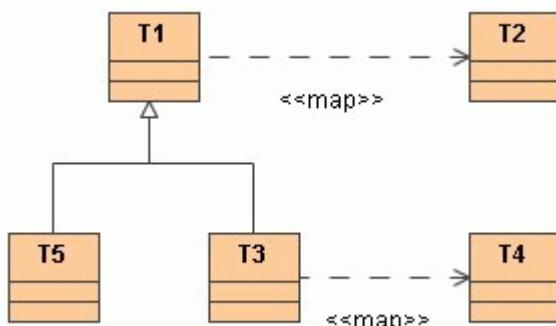
During the initial transformation, only the default rules for each source type come into play. E.g., if the user has a property with the String type, this will be transformed to property having the varchar type set.

However, during the transformation update, all rules come into play. If the destination type is one of the acceptable types according to the map, it is not changed. Otherwise it is replaced with the default mapping.

Regarding the example above, let's say that after the initial transformation, the user changes the type of the property in the destination model from varchar to nvarchar (as a post-transformation refinement process). If the user now runs a transformation update, this change will not be overwritten, since nvarchar is an acceptable type as there is a String → nvarchar mapping in the type map as well. If on the other hand the user sets the type of this property to number, this would be reset during transformation update, and the type will be forced back to varchar, as there is no String → number mapping.

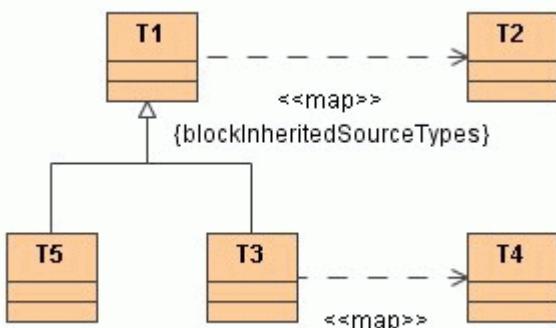
Controlling type inheritance, any, and empty types

You can also control mapping behavior for the type inheritance. By default, derived subtypes are also mapped by the rule governing the parent type (unless, of course, they have their own rules for mapping). If the **blockInheritedSourceTypes** tagged value is set, derived types are not affected by this rule. Let's review the following example:



Here T1, T2, and T5 are types in the source domain, while T2 and T4 are types in the destination domain. Given these two mappings ($T1 \rightarrow T2$ and $T3 \rightarrow T4$), the following statement is true: T1, and all types derived from it (such as T5), are mapped to the T2 type, except T3 and any of the types derived from it. These types are mapped to T4.

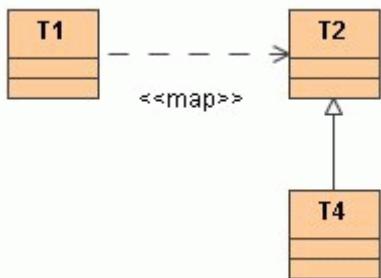
Now consider an example where **blockInheritedSourceTypes** is set:



In this case, T3, along with the types derived from it, are still mapped to T4. T1 is still mapped to T2. However, unlike the previous example, T5 and all the types inherited from T1 are NOT mapped to T2.

You can also control the mapping behavior of the type inheritance in the destination model. This is only effective on the transformation updates, the second (and successive) reapplications of the transformation. By

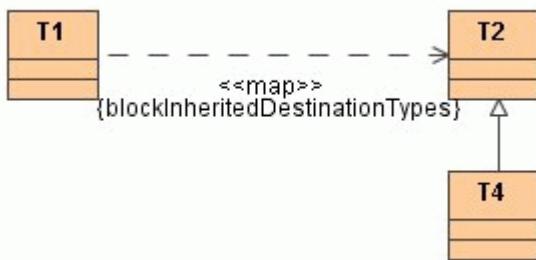
default, derived subtypes in the destination model are not overwritten, since they are considered suitable substitutes of their parent. Let's review the following example:



Here T1 is a type in the source domain, while T2 and T4 are types in the destination domain. Given this mapping ($T1 \rightarrow T2$), on the first application of the transformation, type T1 residing in the source model will be mapped to type T2 in the destination model.

Now let us look at a case, where the user refines the destination model by changing the type on the destination model attribute from T2 to T4. This situation is quite common, for example, the user refines an attribute type from string to basic URI in the XML schema, or from Integer to nonNegativeInteger, and so forth. The essence is that the mapping for inherited types of T2 is performed as if there was a mapping $T1 \rightarrow T2$ (default), $T1 \rightarrow T4$, $T1 \rightarrow \langle\text{any_other_type_inherited_from_T2}\rangle$.

Now consider what happens, when we apply the **blockInheritedDestinationTypes** tagged value:



In this case, type T4 has no special treatment. If the user applies the transformation, T1 is mapped to T2. Afterwards the user refines the destination model, changing the attribute type from T2 to T4. If the user now updates the transformation, the attribute type is overwritten: T4 is reset back to T2.

When the user loads the type map in the reverse direction, the roles of the **blockInheritedSourceType** and **blockInheritedDestinationTypes** are transposed (unless of course the **direction** tag mandates that this mapping is not used in the reverse direction).

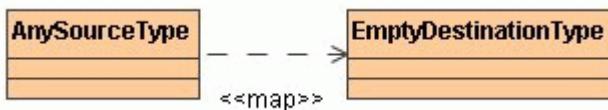
The special type **EmptySourceType** (residing in the *Model_Transformation_Profile.xml.zip*) is used in type maps to indicate that the attributes with no type should be mapped with this dependency.

The special type **EmptyDestinationType** (residing in the *Model_Transformation_Profile.xml.zip*) is used to indicate that the attributes in the destination classes should have no type after remapping (type removal).

The special type **AnySourceType** is a template that matches any type in the source model (see mapping rules for type inheritance). By using this type, together with the inheritance mapping rules, the user can specify that any other types not defined by the mapping should be interpreted by this mapping.

The special type **AnyDestinationType** is a template that matches any type in the destination model (see mapping rules for type inheritance).

Here is an example of template type usage:



According to this rule, any types in the source model for which there are no other mapping rules should be stripped in the destination model.

Type modifiers

Type mapping rules can also affect type modifiers during the type replacement. Type modifier is a small string, which modifies type usage in the typed element. They are used, for example, for specifying arrays during the modeling (e.g., property type = char and type modifier = [30] gives property:char[30]). Type modifiers are extensively used in DDL models for specifying number field widths and varchar field lengths. For example, phone:varchar"(100)", where varchar is a type of phone property and "(100)" is a type modifier.

Each type mapping rule can carry a triple <modifier, regexp, replacementregexp> for setting type modifiers during the type replacement. These are specified in the tags on the mapping rule <**forwardTypeModifier**, **forwardTypeModifierRegexp**, **forwardTypeModifierRegexpReplace**> triple for controlling modifiers during the forward application of type map and correspondingly the <**reverseTypeModifier**, **reverseTypeModifierRegexp**, **reverseTypeModifierRegexpReplace**> triple for controlling modifiers during reverse application of type map.

Any of the components of the triple can be missing, i.e., not specified.

If no tags are specified, then type modifiers are not changed during the type remapping operation (whatever modifier was in the source model, it will be copied into the target model)

If just the modifier is specified for the mapping rule, then modifiers are set during the application of this type rule. This can be used for setting the fixed type modifiers. For example, mapping boolean in the UML model to number(1) in the DDL models (in this case the modifier="(1)" is used in the type map).

If all three are specified, a modifier, regexp, and regexp replacement, modifier remapping is performed as follows: during the transformation, the existing type modifier is matched against the given regexp. If it does not match, the type modifier is overwritten with the value, specified in the modifier field of the rule. If it does match regexp, the replacement is run on the match result and produces a type modifier to be set as a result. This allows quite complex rules to be written and executed, however this mandates good knowledge of regexp.

Let's review the following live example: in the char -> varchar type mapping rule for the UML to DDL transformation, the following triple can be used: modifier="(255)", modifierRegexp="^[\(\)[\(\)(0-9*)[\)\]]\$", and modifierRegexpReplace="(\$1)". This causes the char[20] type usages (type=char, modifier="[20]") in the source be changed to varchar(20); char (without modifier) would be remapped to varchar(255).

If regexp replacement is not specified, it is treated as if "\$0" was specified: the type modifier is copied from the source, if it does match the regexp.

Transformation Traces and Update

When a transformation is performed, it establishes traces between the transformation source model elements and the transformation result model elements. These traces carry information of what was transformed into what. Traces are stored in an auxiliary package under the transformation destination package carrying the <<transformation>> stereotype and containing a lot of instance specifications (since this is implementation specific data, please, do not edit the internals). If you want to remove trace information, simply delete this package.

Trace information can be used for navigating between the model layers. This is done with the traceability features of MagicDraw. To navigate in the forward direction, i.e., from the transformation source model element to the destination model element, right-click that element and choose **Go To > Traceability > Model Transformations > Transformed To > <element>**. To navigate in the backward direction, i.e., from the transformation destination model element to the source model element, right-click that element and choose **Go To > Traceability > Model Transformations > Transformed From > <element>**.

Traceability information is also visible in the element's Specification window, the **Traceability** tab; in the Properties panel, **Traceability** tab; it can also be depicted in the Relation Map diagram or in the custom dependency matrix.

Traces can be used for running the transformation update. The transformation update reapplies the transformation with the same source and target for the purpose of carrying additional changes from the source (which occurred after transformation was made) into the destination.

During the transformation update, presence of unmapped model elements in the source model indicates that these are newly added elements. Usual rules and the same behavior for the transformation are used for these elements as if this were the first application of the transformation.

During the transformation update, if model elements already contain mapping, and the source and destination does not match, the question arises - which properties to use. This is fundamental problem for all updating operations. The general solution is to have some kind of merge between the source and target. But merge is clumsy and expensive. MagicDraw implements a simplistic approach meaning that during the transformation update a user can choose, wherever he/she wants the source model element properties to win (destination model element properties are overwritten) or wherever he/she wants the destination model properties to win (destination model properties are not changed).

To update a transformed model

- Right-click the destination package and select **Tools > Update Transformed Model**.

The **Model Refresh Options** dialog box opens.

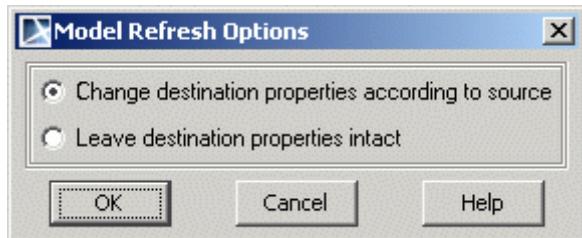


Figure 232 -- Model refresh options dialog box

The **Change destination properties according to source** option causes overwriting of element properties in the destination model with properties from the source model (only for the elements connected with mapping dependencies).

The **Leave destination properties intact** option leaves the destination model properties unchanged but different from the source model, in other words retains changes made to the destination model while it ignores changes made to the source model.

NOTE

If new elements are added to the source, once the update occurs, copies of the new elements will be created in the destination model. If an element is deleted from the source, it will not be removed from the destination after the update.

Profile Migration Transformation

Transformation provides a way to migrate your model according specified mapping rules.

Profile migration transformation search-and-replaces the following elements:

- Applied stereotypes
- Tagged values
- Usages of one type with another type

Element symbols are updated according to model changes.

Before starting transformation you have to define transformation mapping.

Profile Migration Transformation mapping

Before starting transformation you have to create transformation mapping rules.

To create mapping rules you need to create Dependency relationship between elements you want to transform. Mapping rules can be created (dependency relationship can be created) between the following elements:

1. Stereotypes
2. Tags
3. Types

Creating mapping rules for Stereotype transformation

This type of transformation is used to replace Stereotype. To create Stereotype transformation mapping rule:

1. Create Dependency relationship between Stereotypes which you want to transform.
2. Apply *ReplaceStereotype* stereotype to Dependency.
3. Perform transformation.

Tag values of old stereotype are preserved when tag name and type of tag value is the same. For tag values with different names create mapping rule for tag transformation.

ReplaceStereotype stereotype has the following tags:

- **disableNewTypeCreation** tag. By default false. Set this tag to true if you do not want to perform transformation when target and source metaclasses are not compatible. For example, if you do not want that Class would be changed to Use Case.
- **disableReplaceWhereSaveAsElementValue** tag. By default false. Set this tag to true if you do not want that stereotype would be changed where it is used as Tag value (tag value is stereotype itself, for which you perform transformation).

Creating mapping rules for Tag transformation

This type of transformation is used to replace Tag (when tag names differs). For example, source stereotype has *author* Tag and target stereotype has *name* Tag.

To create Tag transformation mapping rule:

1. Create Dependency relationship between Tags which you want to transform.
2. Apply *ReplaceTaggedValue* stereotype to Dependency.

3. Perform transformation.

NOTE

To create mapping rule correctly, you have to create Dependency relationship not only between Tags, but also between Stereotypes of these tags (see mapping between stereotypes, which is described above).

Creating mapping rules for types transformation

This type of transformation is used to replace type. For example, to replace type of Attribute.

To create types transformation mapping rule:

1. Create Dependency relationship between Types which you want to transform.
2. Apply *ReplaceType* stereotype to Dependency.
3. Perform transformation.

Starting Profile Migration Transformation

To start the **Model Transformation Wizard**:

Do either:

- From the **Tools** menu, choose **Model Transformations**.
- Right-click one or more packages and select **Tools > Transform**.

To start the Profile Migration Transformation

- In the Model Transformation Wizard select the **Profile Migration** transformation.

NOTE

The **Next** step in wizard will be disabled if there will be no defined mapping.

Sample of the Profile Migration Transformation

This sample describes step-by-step instructions how to create profile migration mapping rules and perform transformation. In this sample we will change one stereotype to other.

1. Create *Book* stereotype with Class metaclass.
2. Create your model, for example, create *Source* package with *Source Class* diagram. Draw *Sample Class* and apply *Book* stereotype.
- To change *Book* stereotype to other, for example to *Magazine* stereotype, you have to create profile migration transformation mapping rules. Follow next steps for creating mapping.
3. Create stereotype *Magazine* with Class metaclass.
4. From the *Magazine* stereotype to *Book* stereotype draw Dependency relationship.
5. To the Dependency relationship apply *ReplaceStereotype* stereotype (see Figure 1). Profile Migration transformation mapping rule is created. Now you can start transformation.
6. To start transformation select **Model Transformations** from the **Tools** menu. The **Model Transformation Wizard** will open.
7. Select the **Profile Migration** transformation and click **Next**.
8. Select the *Source* package in the *Select source/destination* step (Figure 3). Click **Next**.
9. The **From** and **To** fields display the mappings of the selected transformation profile in the *Check mappings* step (Figure 4). Click **Finish**.

After this transformation stereotype of *Sample Class* will be changed to *Magazine* stereotype.

7 TOOLS

Model Transformation Wizard

For more information about Model Transformation Wizard, see “Working with Model Transformation Wizard” on page 351.

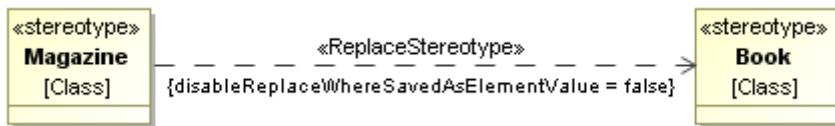


Figure 233 -- Profile Migration Transformation mapping

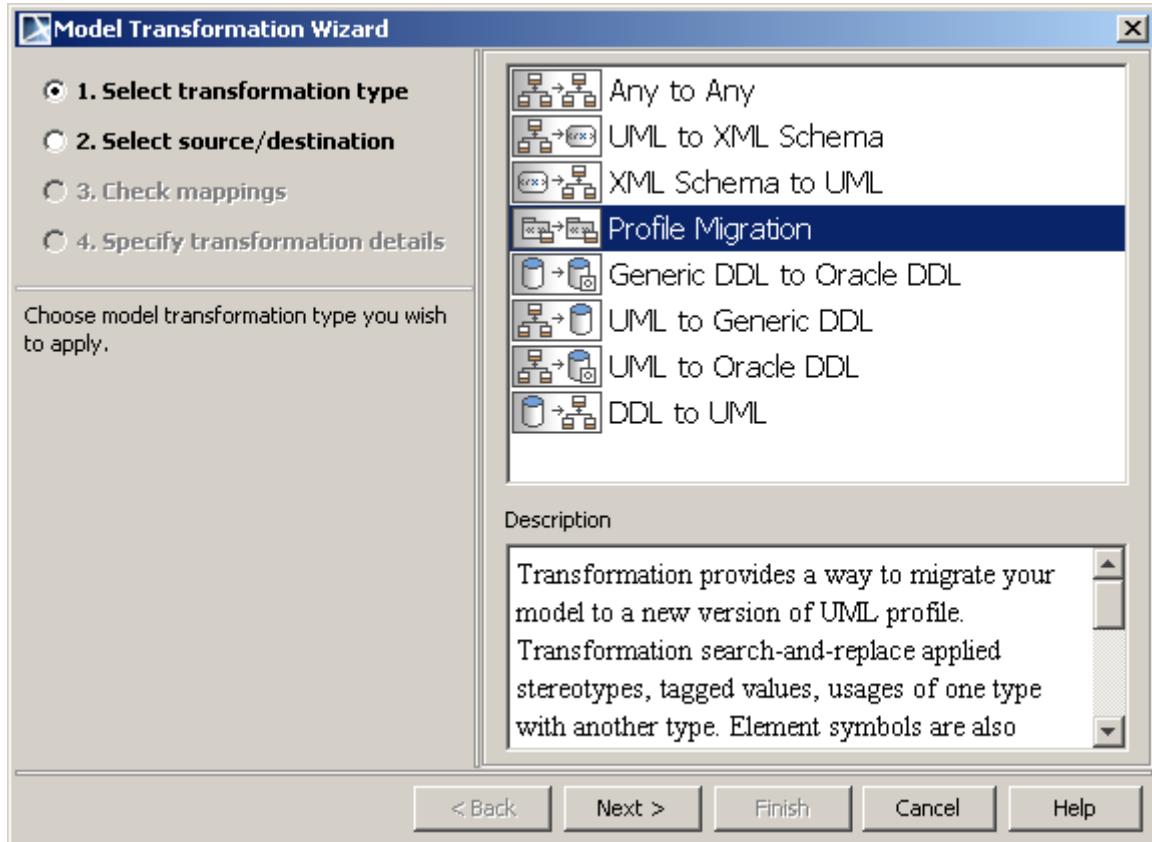


Figure 234 -- The Model Transformation Wizard, the Select transformation type step

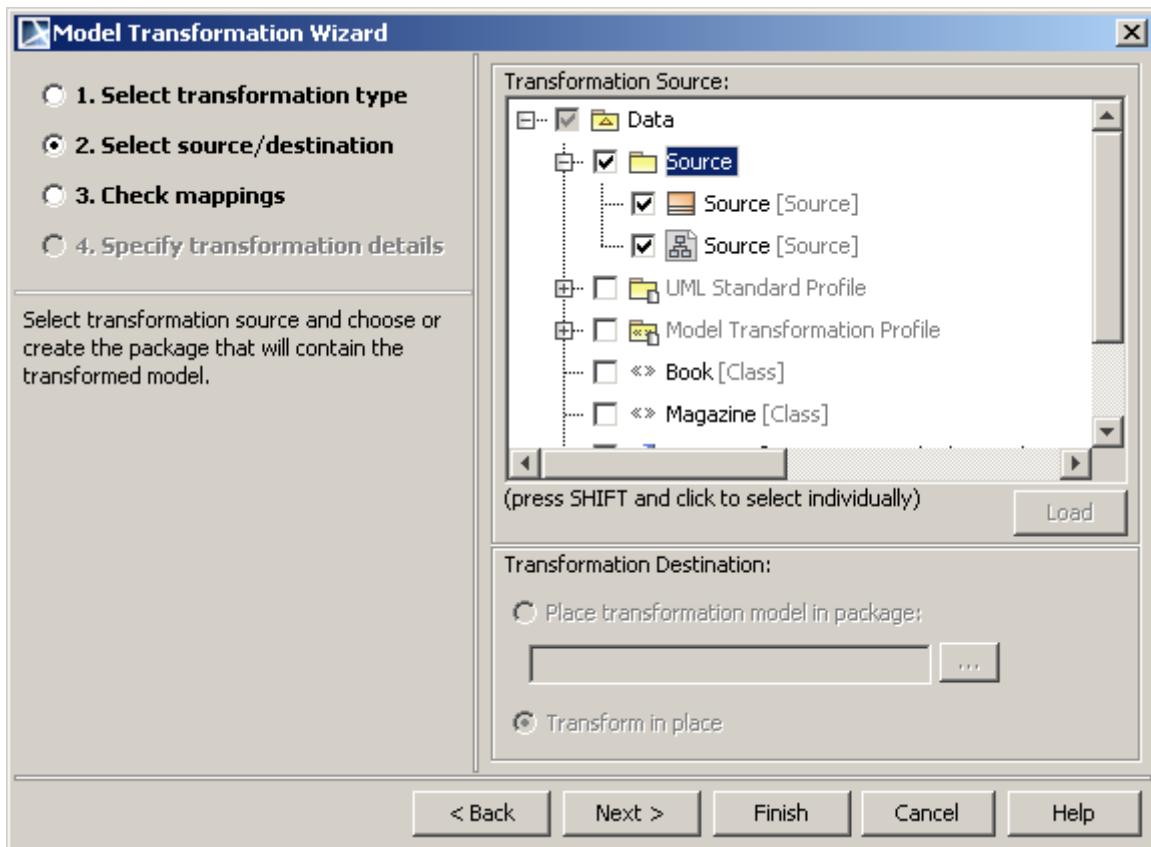


Figure 235 -- The Model Transformation Wizard, the Select source/destination step

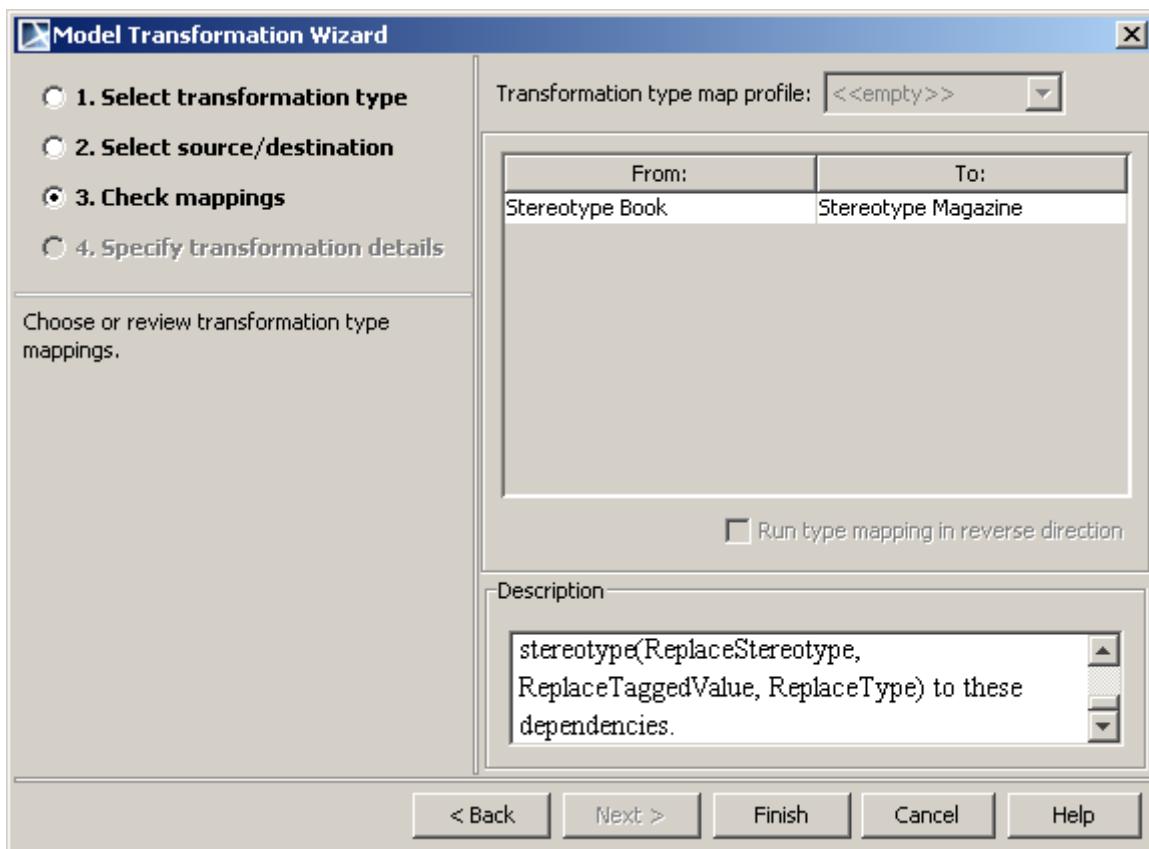


Figure 236 -- The Model Transformation Wizard, the Check mappings step

Resource Manager

MagicDraw Resource Manager functionality allows you to manage local resources (installed with MagicDraw, downloaded) and resources available on the web.

With RM (Resource Manager) you can manage different types of resources (Profiles, Plugins, Templates, Language resources, Case studies/examples, Custom diagrams, and others).

RM allows:

- Review
- Remove (delete)
- Import, download
- Update resources.

The following are benefits of the Resource Manager functionality:

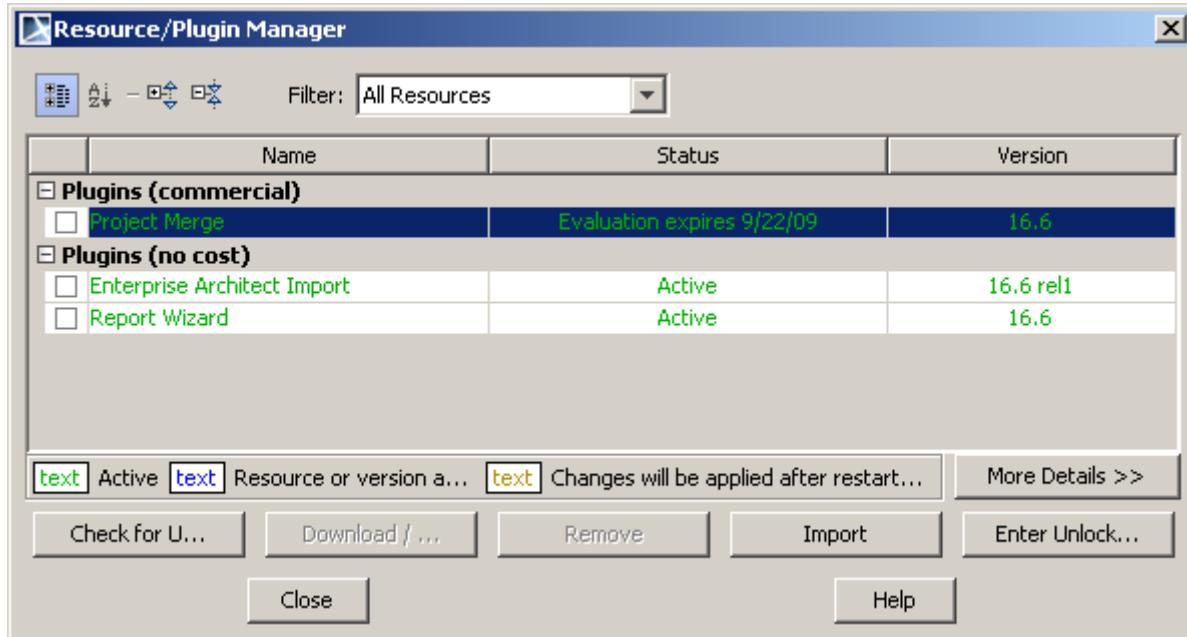
Benefits:

- Easier to find needed resources and download them.
- More informative resource descriptions.
- The possibility to create your own resources and share them.
- The possibility to check resource dependency in the required table.

The Resource Manager functionality is included in all MagicDraw editions, except, Reader.

To open the Resource Manager:

From the [Help main menu](#), select the **Resource/Plugin Manager** command. The [Resource / Plugin Manager window](#) opens.



The table below lists items of the Resource Manager window.

| Element Name | Function |
|--------------|--|
| Name | Different types of resources are listed in separate nodes. There are the following resources types: examples, templates, languages, and profiles. In front of each resource is a check box. Select this check box to manage this resource. If the resource is unavailable, it is impossible to select the check box in front of it (checkbox is grey). The required table rows contain information about why it is not available. |
| ID | Resource ID. |
| Category | Resource type. |
| State | The following states of resources are possible: <ul style="list-style-type: none"> ● Not installed (downloaded) ● Will be installed after restart ● Installed ● Will be removed after restart ● Not downloaded ● removed (if resource does not exist on the web). |
| Ver Obtained | The Version Obtained column indicates the version for a resources and the state: installed, not installed, and removed. |

| | |
|----------------------------------|--|
| Ver Available | The Version Available Column indicates the newest resource version on the web. <ul style="list-style-type: none"> ● Does not exist on the web. If the resource does not exist on the web. ● Check for updates. If the resource list has not been downloaded from the web. |
| Date | Resource creation date. |
| Size | Size of the particular resource. |
| Meaning of text colors | One row lists the meanings of the text colors: <ul style="list-style-type: none"> ● Green - installed resources are marked green. ● Blue - not downloaded resources are marked blue. If the resources are already installed or downloaded and the newer version is available, the resources are marked blue. ● Black - changes will be fully applied after restarting MagicDraw. |
| More>>/<<Less | Expands / Collapses the main Resource Manager window with additional fields. The following fields list all data of the selection in the resource list: <ul style="list-style-type: none"> ● Name, Resource home page, Provider, Description ● Required table (with Name, Required, Status columns). If a resource is installed but a newer version is available on the web, the required table shows the newest version requirements. |
| Check for Updates | The Check for Updates button is inactive if the resource list has already been successfully downloaded. |
| Download/Install | Downloads the selected resources and installs. The Download/Install button is inactive if no resources have been selected. |
| Remove | Button is active if at least one resource is selected with states “not installed” or “installed”. The resources of other states may not be removed from the RM list. The remove functionality is not available for required resources. |
| Import | The Open dialog is opened. Import the resource you need. |
| NOTE | If the resource requires payment, the Price of the particular resource is displayed in the Resource Manager window under the Price column. |

Spelling Checker

Spelling Checker enables you to:

- Check spelling as you type. A shortcut menu provides spelling options. Right click the word underlined in red to enter the shortcut menu. Spelling options will be displayed. Words can also be entered into a customized dictionary using **Add to Dictionary** (see “Spell checking as you type” on page 369).
- Check the spelling of a whole project or of a selected part. You can list all the spelling errors found in a project and correct them easily (see “Spell checking for the whole project or the selected scope” on page 371).
- Set Spelling Checker options. You can set spelling checker options, such as skipping numbers, upper case words in the **Environment Options** dialog box (**Spell Check** option) (see “Setting the spell checking options” on page 375).
- Add a spell checking dictionaries. All “Open Office” supportive spelling languages can be added additionally to the existing ones (see “Spell checking dictionaries” on page 377).

Spell checking as you type

On typing spell checker checks if typed word is correct. In case the word is incorrect it is underlined with red winding line. Right-click on underlined word invokes context menu with suggested editions and capability to add word to dictionary.

Spell checking as you type is performed in the following locations in MagicDraw:

- symbol names on diagram pane,
- for properties in the specification dialog boxes,
- in the documentation pane,
- in the Containment tree,
- In various location on log messages, names, typing and comment fields.

Right-click on underlined word to invoke the shortcut menu in the following MagicDraw window locations:

1. Diagram pane (see Figure 237 on page 369).
2. Containment tree (see Figure 238 on page 370).
3. Dialog boxes (see Figure 239 on page 371).

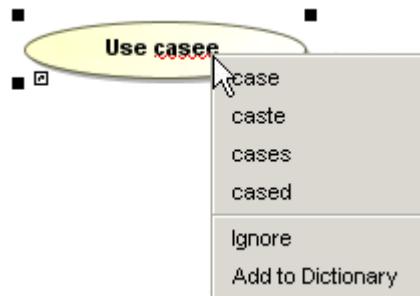


Figure 237 -- Shortcut menu of the spelling error on the diagram pane

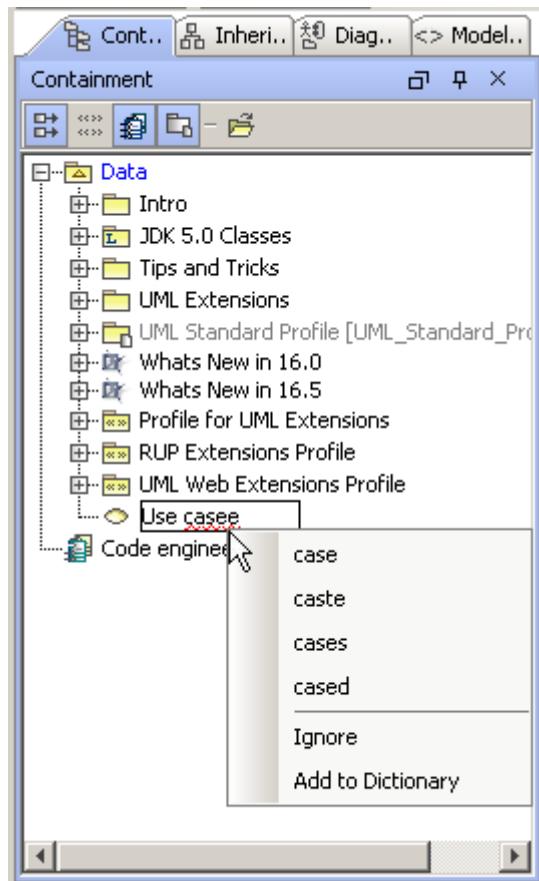


Figure 238 -- Shortcut menu of the spelling error in the Containment tree

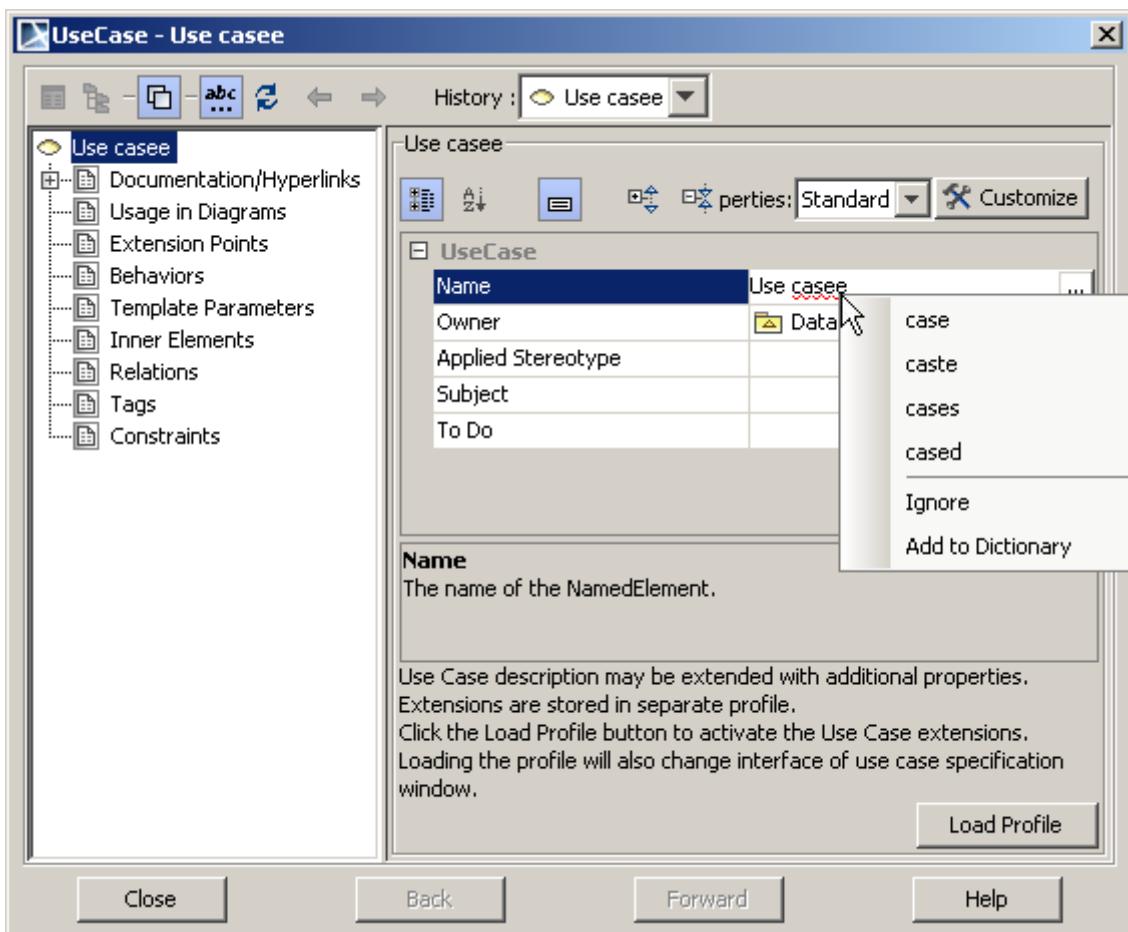


Figure 239 -- Shortcut menu of the spelling error in the dialog box

There are three ways to correct the spelling error:

1. Wrongly typed word can be changed by typing or by selecting provided suggestion that is always correct syntactically, but not always correct semantically.
2. The **Ignore** command. Select the **Ignore** menu item and the wrongly spelled word is treated as correct for this time, but will be discovered as wrongly spelled in the other case.
3. The **Add to dictionary** command. Select the **Add to dictionary** menu item in pop-up menu. By pressing this item the wrongly spelled word will be added to custom dictionary. Next time it will not be treated as wrongly typed.

NOTE: Wrongly spelled words are underlined only in edit mode of particular component. If edit mode has been left, underline disappears and vice versa.

Spell checking for the whole project or the selected scope

Checking spelling for the whole project

1. From the **Tools** main menu, select the **Check Spelling** command or press the **Check Spelling**  button in the main toolbar. The **Check Spelling** dialog box opens.

2. Click the **Check** button. The **Validation Results** dialog box opens.



Figure 240 -- The Check Spelling dialog box

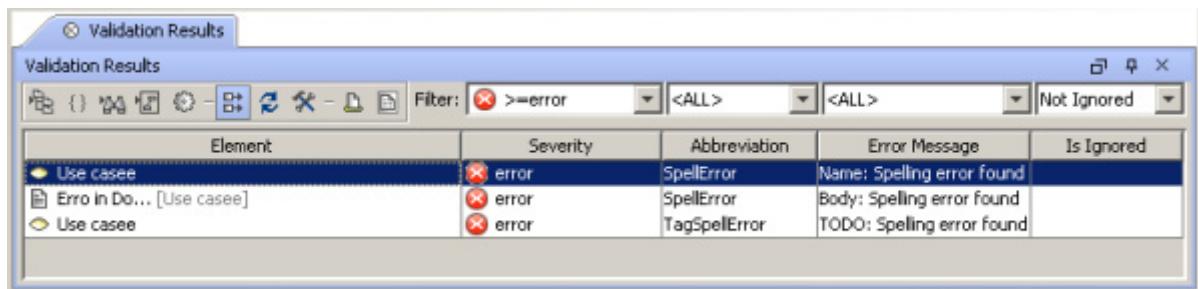


Figure 241 -- The Validation Results dialog box

Checking spelling for the selected scope

1. From the **Tools** main menu, select the **Check Spelling** command. The **Check Spelling** dialog box opens.
2. In the **Check Spelling For** combo box select the **Validation Selection** option.
3. Click the "..." button. In the element Selection dialog, select the scope to check spelling.
4. Click **OK**.
5. In the **Check Spelling** dialog box, click **Check**. The **Validation Results** dialog box opens.

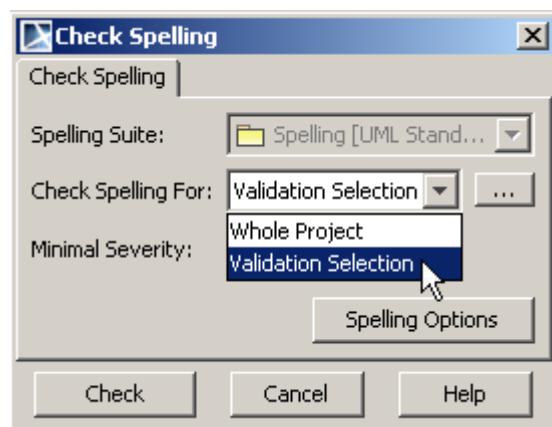


Figure 242 -- The Check Spelling dialog box, the Check Spelling For option

Analyzing the Check Spelling (the Validation Results window)

The **Validation Results** window provides with all spelling errors. You can choose either to correct particular error or to ignore it. For more information about correcting the spell checking error, “Solving the spell checking errors” on page 373.

Spelling error in the **Validation Results** window has description of construct as follows:

| Column title | Description |
|----------------------|---|
| Element | Element with contains spelling error. Spelling error can be found in the element name, or inner properties, such as documentation, the To Do property and others. |
| Error Message | <Element property name>: <Spelling error found> |

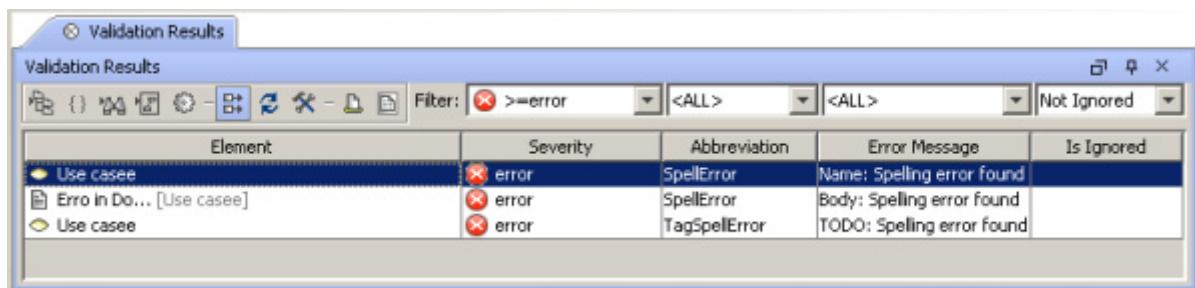


Figure 243 -- The Validation Results window

For more information about validation functionality, see “Validation” on page 444. Here the **Validation Results** window is described in more details.

Solving the spell checking errors

To solve the error:

1. In the **Validation Results** window select the error and with right click invoke its shortcut menu.
2. In the Containment Tree select the element with error (marked with red circle and white cross inside) and with right-click invoke its shortcut menu.
3. On the diagram pane, select the element with error (highlighted with red border) and with right-click invoke its shortcut menu.

In the shortcut menu of element with error, select the **Correct** command to invoke the **Spell Checker** dialog box.

In the **Spell Checker** dialog box the **Element** property shows the name of the element. The **Property** shows the name of the element's property that has spelling error in its value. In the **Value** property all wrongly spelled words are underlined.

Press the **Next** button to go to the next spelling error found during validation. To close spelling dialog and save changes press **OK**. To close dialog without saving changes press **Cancel**.

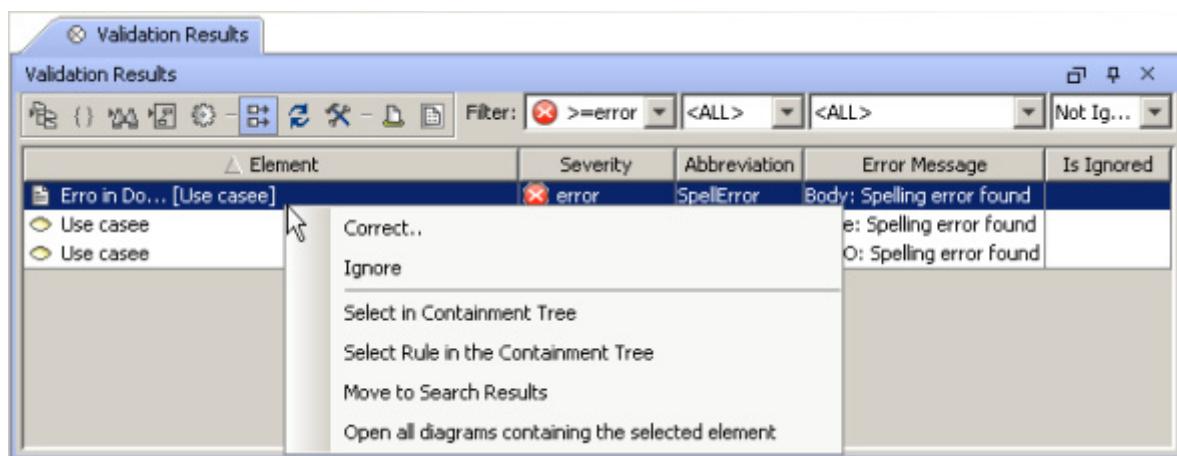


Figure 244 -- Solving the spell checking errors in the Validation Results window

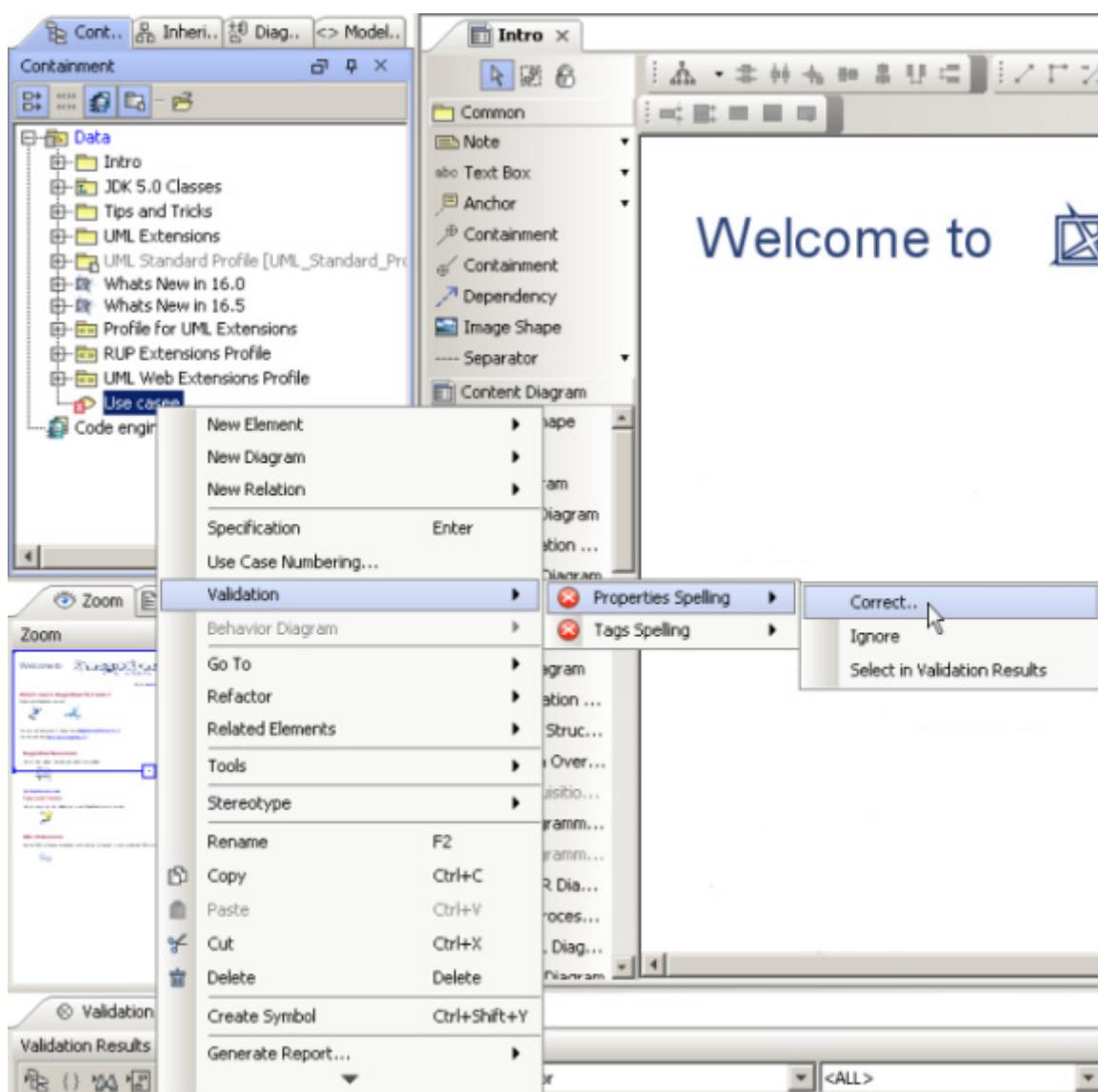


Figure 245 -- Solving spell checking errors in the Containment tree

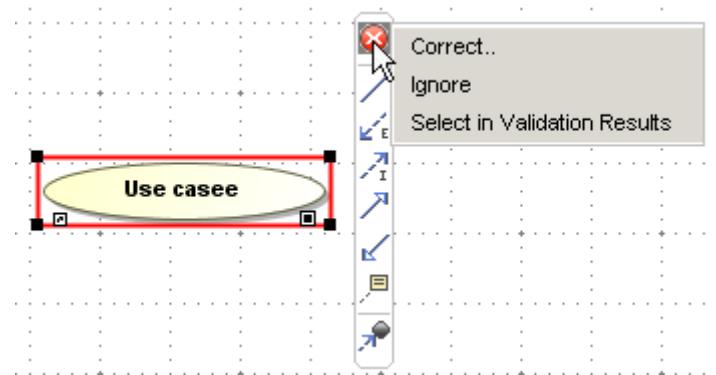


Figure 246 -- Solving spell checking errors in the diagram pane

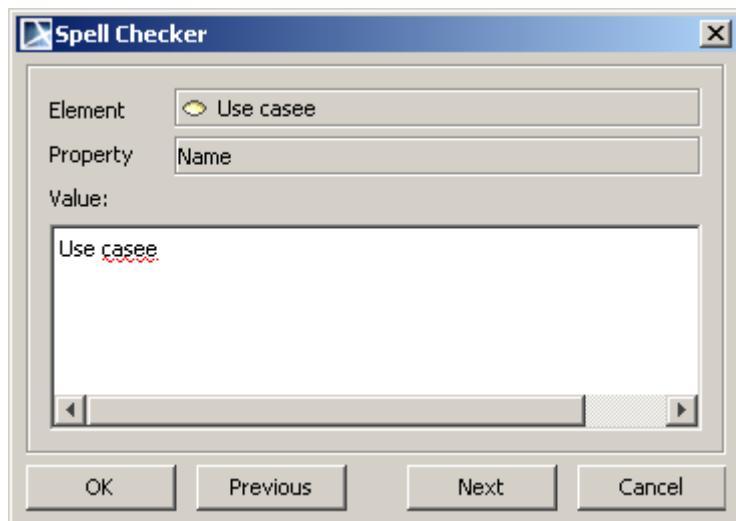


Figure 247 -- The Spell Checker dialog box

Setting the spell checking options

1. From the Options main menu, select Environment. The Environment Options dialog box opens.
2. Select the Spelling branch. Define the spelling options.

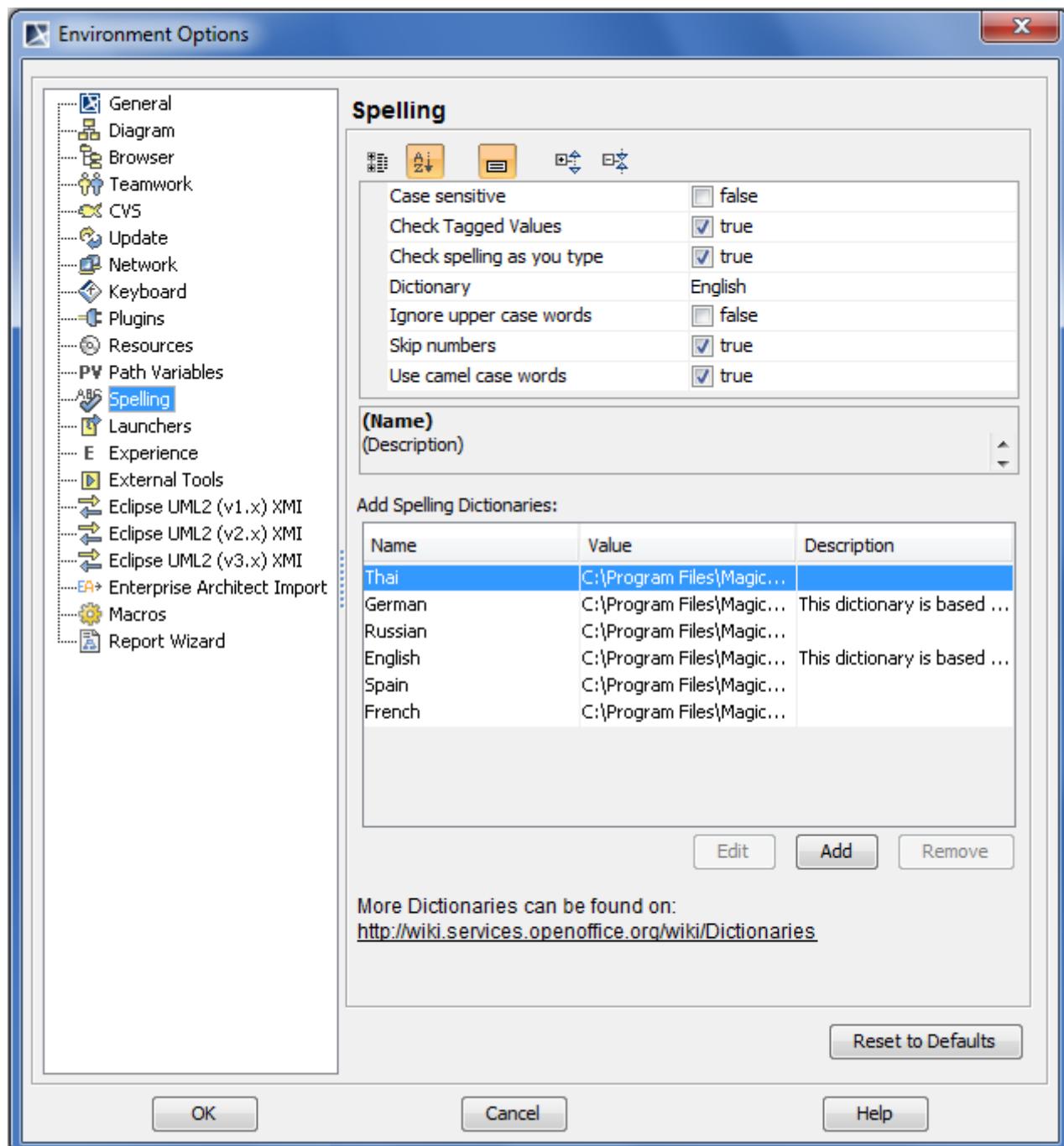


Figure 248 -- The Environment Options dialog box, the Spelling branch

Spell checking options

See the spelling options in the table below.

| Property name | Function |
|-----------------------------------|--|
| Check Spelling as you type | Underlines wrongly spelled words and provide the suggestions list of possible corrections in context menu. For more information about spell check on type, see "Spell checking as you type" on page 369. |

| | |
|--------------------------------|--|
| Check Tagged Values | Checks all tagged values that is Type of string. |
| Dictionary | Select language for spelling. All "Open Office" supportive spelling languages can be added additionally to the existing ones. For more information about spelling dictionaries, see "Spell checking dictionaries" on page 377. |
| Case Sensitive | If true, words will differ in meaning based on differing use of uppercase and lowercase letters. |
| Use camel case words | If true, compound words or phrases in which the words are joined without spaces and are capitalized within the compound-as in BackColor, will be spelled as separate words. |
| Ignore upper case words | If true, words with uppercase words only are not to be spell checked. |
| Skip numbers | If true, numbers are not to be spell checked. |

Spell checking dictionaries

All "Open office" supportive spelling languages are available.

To import spelling dictionaries:

1. Click the **Add** button in the **Environment Options** dialog box > the **Spelling** branch > the **Add Spelling Dictionaries** group. The **Dictionary** dialog box opens.
2. Type the name of a new spelling dictionary in the **Name** text box.
3. Click the "... " button and select the OpenOffice zip file location.
4. Type the description of a new spelling dictionary in the **Description** text box.

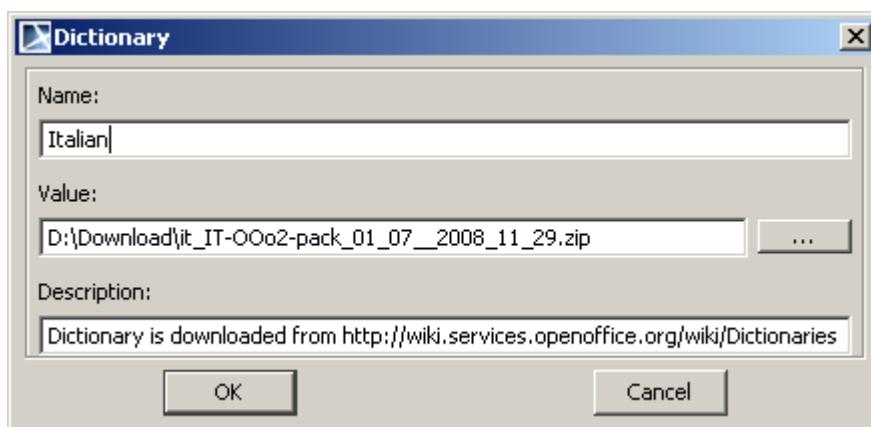


Figure 249 -- The Dictionary dialog box

NOTE

More dictionaries can be found on: <http://wiki.services.openoffice.org/wiki/Dictionaries>

Defining properties of the customized element to be spell checked

DSL customization classes and their properties can be checked either. You can choose what properties of the customized element (Class with <<Customization>> stereotype) you want to be spelling checked.

Defining customization class tag "checkSpelling" value can do it. "checkSpelling" tag can be found in properties tag group. By creating value for this tag you can choose String properties to check spelling for. By default there are no properties marked as checkable.

For more information about DSL, see UML Profiling and DSL UserGuide.pdf.

Import Data to MagicDraw

Import data from Rational Software Architect/Modeler using MagicDraw RSXConverter

MagicDraw RSXConverter provides a seamless way to convert IBM® Rational® Software Architect (RSA) or IBM® Rational® Software Modeler (RSM) file format (*.emx, epx and efx) to MagicDraw-supported file format (*.mdxml).

For more information about MagicDraw RSXConverter see:

[MagicDraw RSXConverter](#)

Import data from Rational Rose using MagicDraw RConverter

MagicDraw RConverter provides a seamless way to convert Rational Rose Model file format (*.mdl) to MagicDraw-supported file format (*.xmi).

For more information about MagicDraw RConverter, see:

[MagicDraw RConverter plugin](#)

Import data from other tools

1. MagicDraw can import most of the model data from the latest Enterprise Architect version. Enterprise Architect does not export 100% standard UML 2.1.1 XMI and this causes some data loss on import. Diagramming information is not imported.
2. XDE model import is not available.
3. Rational Software Architect/Modeler 6.x model files (not diagrams) can be imported with Eclipse UML2 1.x import.
4. Rational Software Architect/Modeler 7.x model files (not diagrams) can be imported with MagicDraw RSXConverter (see “Import data from Rational Software Architect/Modeler using MagicDraw RSXConverter” on page 378) or Eclipse UML2 2.x import could be used.
5. Rational Rose model files can be imported with MagicDraw RConverter (see “Import data from Rational Rose using MagicDraw RConverter” on page 378).
6. Together 2006 model files (not diagrams) can be imported with Eclipse UML2 1.x import.
7. Visio model files can be imported using a XMI exporter. Most of the static structure elements (not diagrams) can be imported. XMI plugin for Visio can be found:

Executable file of the XMI plugin

Visio versions

Link

| | | |
|---------------|--|---|
| XMIExport.exe | Microsoft Visio 2002 Professional, Microsoft Visual Studio .NET Enterprise Architect (Visio for Enterprise Architects) | http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=BE6D20EF-36BA-4ABF-A26F-91434C7E7B7 |
| XMIExprt.exe | Microsoft Visio 2003 | http://www.microsoft.com/downloads/details.aspx?familyid=3DD3F3BE-656D-4830-A868-D0044406F57D&displaylang=en#Overview |

8 MODEL ANALYSIS

MagicDraw provides the following capabilities to help you analyze your model:

- "[Displaying Related Elements](#)" - displays paths among shapes that have already been created in the model data, use the quick and simple Display related elements functionality.
- "[Analyzing Usages and Dependencies](#)" - the MagicDraw Usages and Dependencies feature enables you to track and view element dependencies in UML models, explore how model elements are used by other elements, and understand the relationships between used and dependent elements.
- "[Traceability](#)" - allows you to track, visualize, navigate, and analyze the elements involved in traceability relations.
- "[Relation Map](#)" - the Relation Map diagram enables you to rapidly review and analyze relations between the elements of the whole model.
- "[Symbol Usage in Diagrams](#)" - displaying the list of diagrams in which symbol of current element is represented.
- "[Projects Comparison](#)" - compares two MagicDraw UML local projects or teamwork project versions, as well as diagrams.
- "[Metrics](#)" - allows measuring a project by different viewpoints.
- "[Dependency Matrix](#)" - is a method of visualizing and representing dependency criteria.
- "[Analyzing Package Dependencies](#)" - checks and analyzes package dependencies of the whole project or when exporting and sharing packages.
- "[Validation](#)" - a facility for evaluating completeness and correctness of the models created by the user.
- "[Active Validation](#)" - instantly checks model for correctness and completeness, displays errors in the model and suggests solutions.
- "[Model Visualizer](#)":
 - "[Class Diagram Wizard](#)" – helps to create and customize new class diagrams.
 - "[Package Overview Diagram Wizard](#)" – generates the package dependency diagram for packages in your project.
 - "[Package Dependency Wizard](#)" - generates diagrams containing packages (created within a project) and shows the relationships between them.
 - "[Hierarchy Diagram Wizard](#)" and "[Realization Diagram Wizard](#)" – prepares diagrams and report documents of the relationships between classes in the UML model.
 - "[Activity Decomposition Hierarchy Wizard](#)" - converts activity into class and/or SysML Block Definition Diagram.
 - "[Content Diagram Wizard](#)" - generates content of diagrams that are used in the project.
 - "[Sequence Diagram from Java Source Wizard](#)" - creates a sequence diagram of Java method implementation.

Displaying Related Elements

To display elements related to the selected element

1. From the **Edit** menu, **Symbol** submenu or from the shape shortcut menu, select **Related Elements** and then select **Display Related Elements**.
2. The **Display Related Elements** dialog box opens.

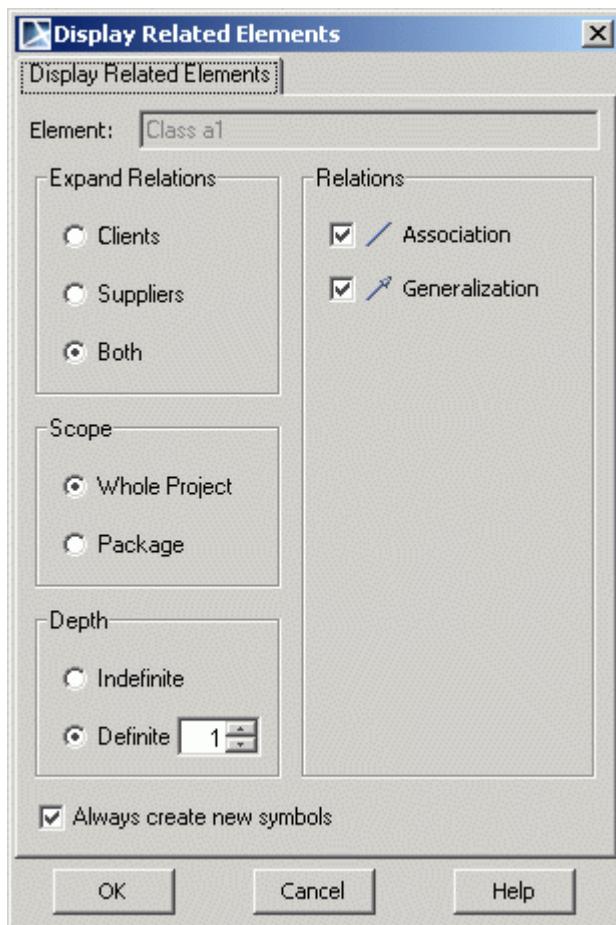


Figure 10 -- Display Related Elements dialog box

3. Select the desired options for displaying the related elements:

| Option name | Function |
|-------------------------|--|
| Expand Relations | Select what kind of elements should be displayed on the diagram: <ul style="list-style-type: none"> • Clients • Suppliers • Both |
| Scope | Select in what scope the elements related to the selected model element will be found: <ul style="list-style-type: none"> • Whole Project • Package |
| Depth | Specify the range for searching for selected relationships: <ul style="list-style-type: none"> • Indefinite. All possible relationships are involved. • Definite. Specify the level of hierarchy for the relationships involved. |

| Option name | Function |
|----------------------------------|---|
| Relations | Select the relationships you want to display on the diagram. |
| Always create new symbols | When this check box is selected, a new symbol is created even if a related element already has a symbol in the diagram. |

To display paths among shapes that already exist in the model data

- From the shape shortcut menu, select **Related Elements** and then **Display Paths**.
- Select a symbol and from the **Edit** menu, select **Symbol**, then **Related Elements**, and then select **Display Paths**.

Analyzing Usages and Dependencies

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The MagicDraw Used By and Depends On features enables you to track and view element dependencies in UML models, explore how model elements are used by other elements, and understand the relationships between used and dependent elements.

It is also useful for analyzing associations between elements or searching for diagrams where these elements are represented.

Understanding Used By

If you want to find all the elements that reference the current element, use the Used By functionality.

For example: If element1 references element2, this means that element1 uses element2. Conversely, element2 is also used by element1. In the **Attribute Specification** dialog box, add class2 in the **Type** box. This means that class2 is used by that attribute.

Containing other elements is not considered usage. For example, when a package contains an inner element class, this does not mean that the package uses the class. The class is categorized as only a container of the package.

Understanding Depends On

If you want to find all elements that current element depends on, use the Depends On functionality.

For example: If element1 contains a reference to element2, this means that element1 depends on element2.

Searching for Usages/Dependent Elements

To search for usages / dependent elements:

1. Select model element in the Browser or in the Diagram pane.
2. From the element shortcut menu, select the **Related Elements** command and then select either the **Used By** or **Depends On** command.

3. The **Usages/Dependencies Search Options** dialog box appears.

The **Usages/Dependencies Search Options** dialog box covers the spectrum of usages and dependent element functionality. This means that if you clear or select any check boxes in the **Usages/Dependencies Search Options** dialog box, the next time you search for dependencies, the values for these check boxes remain the same.

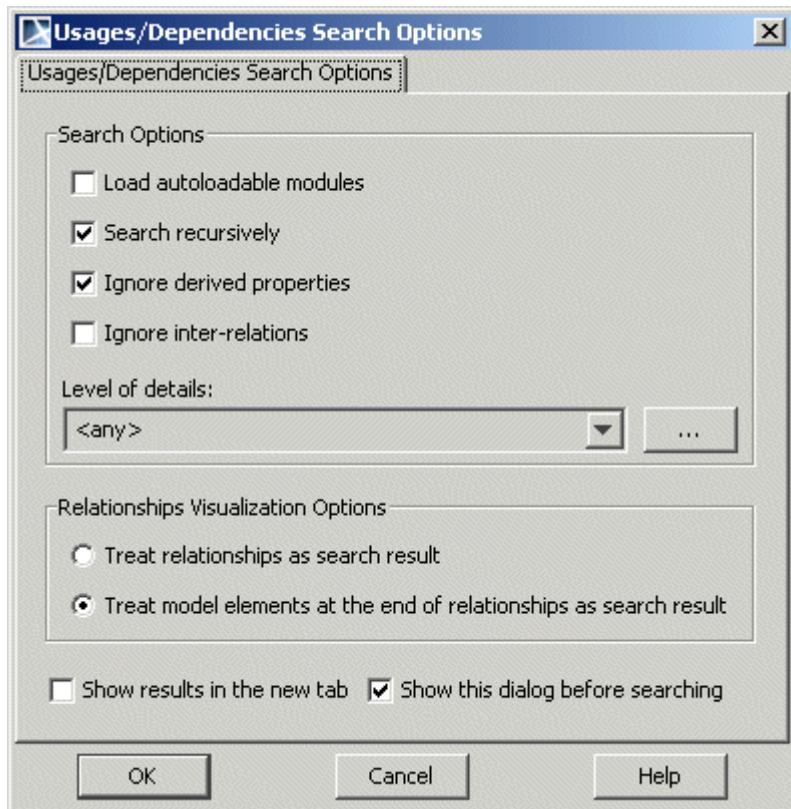


Figure 11 -- Find Usages/Dependent Elements Options

| Element Name | Function |
|---|--|
| Load diagrams and autoloadable modules | If the model has unloaded diagrams or modules, select this check box to load all elements to be included in performing the usages/dependencies search. |
| Search recursively | Usages/dependencies of inner elements (beneath the level of the current element) are listed in the search results list. When the Search recursively check box is cleared (default value), the usages/dependencies table lists those elements that are using the current element. For example: element1 contains element2. When you search non-recursively, those elements that use element1 are listed. When you search recursively, elements using element1 are listed, while the other branch lists those elements that use element2. |
| Ignore derived properties | Derived properties of the element are not included when searching for elements usages/dependencies. NOTE: A derived property is the one that is automatically calculated from the other properties. |

| Element Name | Function |
|---|--|
| Ignore inter-relations | Only usages/dependencies outside the current element (above the level of current element and in the same level) are listed. You may use this option when you export the package as modules. You may analyze the dependencies of a package inner element to its outer elements (note that you cannot export a package that contains inner elements with dependencies to the outer elements according to this package). For example: create package p1 and package p2. Connect these packages with a dependency relationship. In the Browser, drag and drop this dependency to p1. In the Usages/Dependencies Search Options dialog box, select the Ignore inter-relations check box. Search for the p1 dependency. The dependency to the dependency relationship will not be found because this element is a child of p1. |
| Level of details | Select predefined types (Classifiers, Packages) from the combo box or click the "..." button, which opens Select Element/Symbol Type dialog box and allows the selection of custom types. Only elements of these types will be displayed in the search results. For example, <i>Package</i> selected as "level of details". Class B depends on Class A (owned in package p2), because it uses A as type of one operation parameter. In this case package p2 will be displayed as a result, class A will be added under p2 node. |
| Treat relationships as search result | Search results provide a complete list of used or dependent relationships together with other results. |
| Treat model elements at the end of relationships as search results | Relationships are skipped and only usages/dependent elements between the model elements connected with these relationships are displayed. |
| Show results in the new tab | If you want new Elements Using/Dependencies windows displayed for every new search, select Show results in the new tab check box. |
| Show this dialog before searching | If the Show this dialog before searching check box is selected, the Find Usages/Dependent Elements Options dialog box opens when you search for usages or dependencies. If the Show this dialog before searching check box is cleared, you may access the Find Usages/Dependent Elements Options dialog box from the Elements Using/Dependencies window by pressing the Show Usages/Dependent Elements Options button. |

4. Depending on which command you selected on the shortcut menu, either the **Element used by** or **depends on** window appears.

The Elements Using/Dependencies window lists the results of the usages/dependencies. In the Elements Using/Dependencies window you can analyze results, search for an element location (in a diagram, for example, or in a browser), and filter results.

Because the Usages/Dependent Elements Results window is not synchronized with the model,

8 MODEL ANALYSIS

Analyzing Usages and Dependencies

any changes made to the model elements will not show in the results window until you click Refresh.

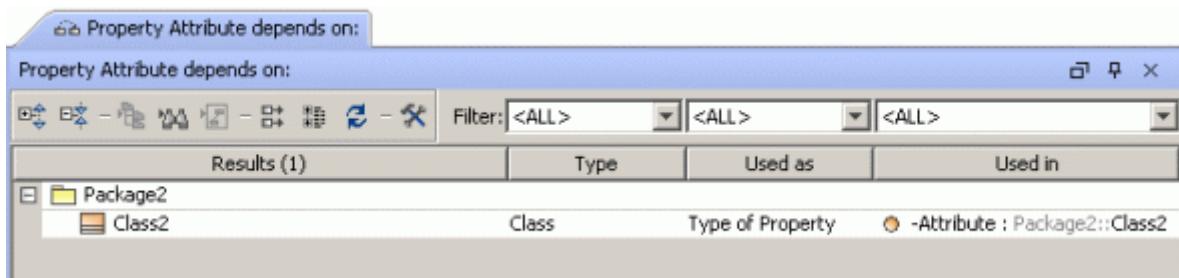


Figure 250 -- Used By/ Depends On window

| Element Name | Function |
|--|---|
| Expand | Expands records listed in groups. Click the plus sign next to the group name to display the contents. |
| Collapse | Collapses records listed in groups. |
| Select in Containment Tree | In the usages/dependencies table, select the record. Click the Select in Containment Tree button. The Element is selected in the Browser, and the Containment tree , and the symbol of this element is selected in the diagram pane. Click the Select in Containment Tree button to open any closed and previously loaded diagrams. You can also select the element in the Containment Tree by double clicking it. NOTE: The Select in Containment Tree button is not available for multiple element selections. |
| Move to Search Results | In the usages/dependencies table, select the record. Click the Move to Search Results button. In the usages/dependencies table results are moved to the Browser, Search Results tree . In the Search Results tree, you will see all the results in one list and organized in two groups: From diagram and From model. |
| Open all diagrams that contain current Usages/ Dependencies | Diagrams, which are referenced in the usages/dependencies table, are opened. In the open diagrams, the view is focused on used/dependent elements. NOTE: The Open all diagrams that contain current Usages/ Dependencies button is inactive when there are no elements that are used in diagrams. |
| Show/hide the Full Path Names | The full path is displayed next to the element name. NOTE: For a symbol this button is not valid. |
| Refresh | The usages/dependencies table should be refreshed after: <ul style="list-style-type: none">• Elements or symbols are deleted.• New dependencies/usages are created for the particular model element. |
| Show Usages/ Dependencies Search Options | The Find Usages/Dependent Elements Options dialog box opens. |

| Element Name | Function |
|--------------|---|
| Filter | <p>Filter boxes in the Elements used by and depends on windows are placed above the column, which will be filtered.</p> <p>In the used by window, you can filter by element type or usage type. In the depends on window, you can filter according to element type and dependency type. When you expand the Filter box, you will see items listed in the usages/dependencies columns.</p> <p>NOTE: In the Element Type Filter box within the used by window, combo box, additional Show All but Symbols filtering options is added.</p> |

Traceability

[View Online Demo](#)

NEW! Traceability

NOTE The traceability solution is available in Architect and Enterprise editions.

TIP! You can also read about the traceability feature and analyze the examples in the Traceability sample.

To open the sample, do any of the following:

- On the Welcome screen, select **Samples > Product Features > Traceability**.
- Go to the folder <MagicDraw installation directory>\samples\product features and open the *Traceability.mdzip* file.

All the examples, given in this section, are based on the data from this sample.

As of version 16.8, MagicDraw supports the traceability feature that allows you to track, visualize, navigate, and analyze the elements involved in the traceability relations.

The traceability relations are used to relate the elements representing the same concept in different UML models at different levels of abstraction or from different viewpoints. Different levels of abstraction may contain, include, or even correspond to different stages of system development process (starting from requirements analysis and finishing with implementation). The higher level of abstraction (e.g., requirements analysis), contains models with specifying elements, and the lower level of abstraction (e.g., implementation) includes models with realizing ones.

The traceability relations help to determine how your requirements or the other model artifacts are satisfied. As they may change, you can use traceability relations to monitor the impact of these changes.

Multiple types of custom and extended UML relationships (e.g., realization, abstraction), tags (e.g., **Alternative Flow of Events Diagrams**), and properties (e.g., **Owned Behavior**) are used to represent traceability relations between the specifying and realizing elements for showing traceability from requirements analysis to implementation and deployment models.

Relation Map and Dependency Matrix

You can visualize the traceability relations of your project in order to analyze them using the MagicDraw features such as Relation Map (for the analysis of traces among multiple levels of abstraction) and Dependency Matrix (for the analysis of traces between any two levels of abstraction).

Element's Specification window, Properties panel, Go To, and Notes

You may track and navigate the elements, that are directly or indirectly related to a specific element through the traceability properties that will be represented in a special tab of the element's Specification window and **Properties** panel, or on the element's shortcut menu under the **Go To** menu. The traceability properties can also be visualized on a diagram using the standard MagicDraw mechanism for displaying property values in notes.

Traceability Report

A Traceability Report is particularly useful when there is a need to visualize and verify that requirement analysis, design, and implementation model elements are all covered in higher or lower levels of abstraction, for example, all use cases should be covered with design classes realizing them.

For more detailed information about the MagicDraw traceability solution, read the following subsections:

1. "Creating Traceability Relations" on page 387.
2. "Traceability Relations Representation" on page 388.
3. "Navigating between Different Levels of Abstraction" on page 394.
4. "Analyzing Traceability Relations" on page 395.
5. "Predefined Traceability Rules" on page 397.

Creating Traceability Relations

Multiple relations can be used for traceability representation, depending on the element type and method used.

The traceability relations can be represented by the following types of element dependencies:

- UML relationships (such as abstraction, realization, derivation)
- UML properties (such as **Owned Behavior**)
- UML tags (such as **Alternative Flow of Events Diagrams**)

The traceability relations can be **single level** and **multilevel**.

- When elements are related directly by using a custom UML relationship, property, or tag, the traceability relation is considered to be a single level relation.
- When elements are related indirectly by usually using multiple types of the above mentioned dependencies, the traceability relation is considered to be a multilevel relation.

The traceability relations can be **specifying**, **realizing**, or **other**. This depends on which direction a relation is analyzed and which element is considered as a basis.

- A relation between a specific element and an element that is the realization of this specific element, from the point of view of the specific element, is considered to be **realizing traceability** (for forward traceability).
- A relation between a specific element and an element that is the specification of this specific element, from the point of view of the specific element, is considered to be **specifying traceability** (for backward traceability).
- All other traceability relations.

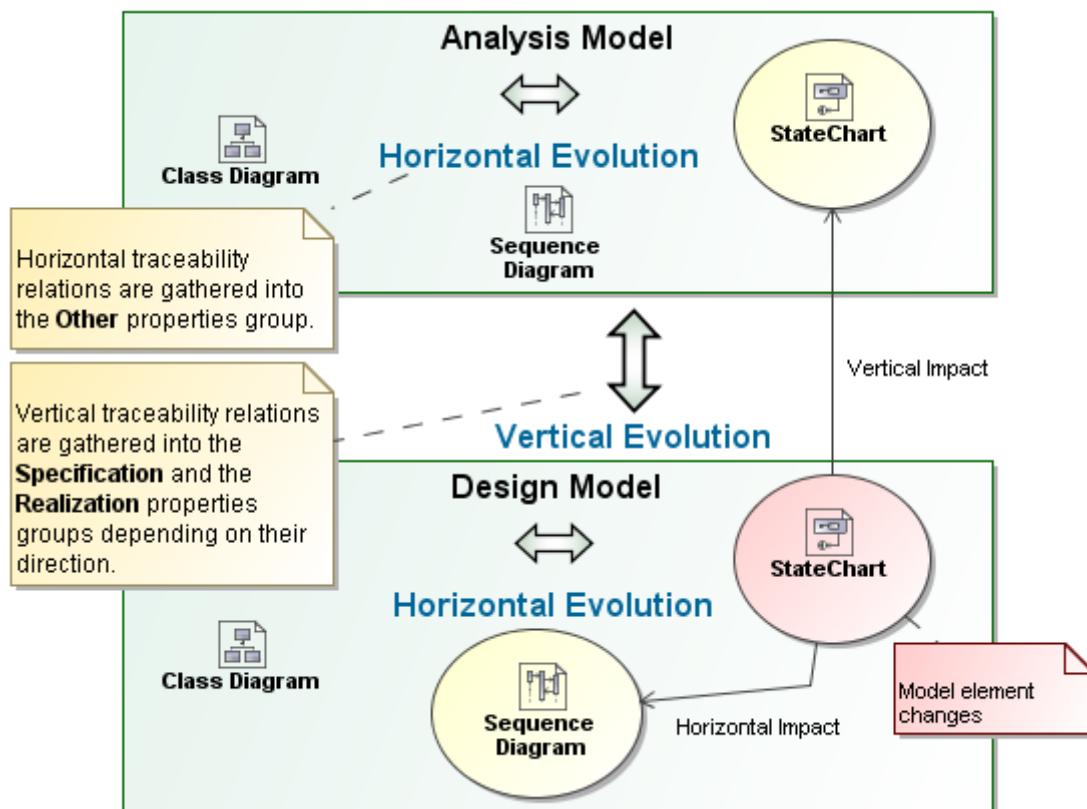


Figure 251 -- Horizontal and vertical traceability representation through the realizing, specifying, and other traceability relations

For more information about the traceability feature, see section “Traceability” on page 386.

Traceability Relations Representation

The traceability relations are represented through the so-called traceability properties that have been pre-defined for each element type according to the traceability method supported in MagicDraw.

A single traceability property shows an element or a set of elements that are related to a particular element through the relations which are specified by some traceability rule. The predefined traceability rules are stored in the Traceability customization profile used by the MagicDraw Profile.

The properties for the traceability relations are grouped into the realization, specification, and other groups, owning both direct (single level) and indirect (multilevel) traceability relations. The indirect traceability relations are represented by the so-called **All** properties, for example, **All Specifying Elements**, **All Realizing Elements**, and **All Specific Classifiers**.

The relations for traceability creation, visualization and navigation is also possible without using traceability properties. However, the customizable model driven traceability properties, which represent the traceability relations in a single place, can greatly help in traceability information visualization and access.

TIP!

You can customize the predefined traceability properties according to your needs or create your own traceability properties and specify your own rules.

You can also customize the grouping of the traceability properties according to your needs.

For more information, please, refer to the sections “Extending metamodel with derived properties” and “Creating your own property groups and sub-groups” from the chapter “DSL Customization engine” in “[UML Profiling and DSL UserGuide.pdf](#)”.

Element's traceability properties are represented in the following places:

- Specification window See section “Traceability properties in Specification window” on page 389.
- **Properties** panel See section “Traceability properties in Properties panel” on page 392.
- Note on a diagram See section “Traceability properties in notes” on page 392.
- **Go To** submenu See section “Traceability properties in Go To submenu” on page 393.
- Traceability Report See section “Traceability properties in Traceability Report” on page 393.

For a list of the predefined traceability rules, see section “Predefined Traceability Rules” on page 397.

For more information about traceability feature, see section “Traceability” on page 386.

Traceability properties in Specification window

The Traceability tab in element's Specification window is one of the places, wherein the element's traceability properties, showing its realizing and/ or more specific elements, are represented.

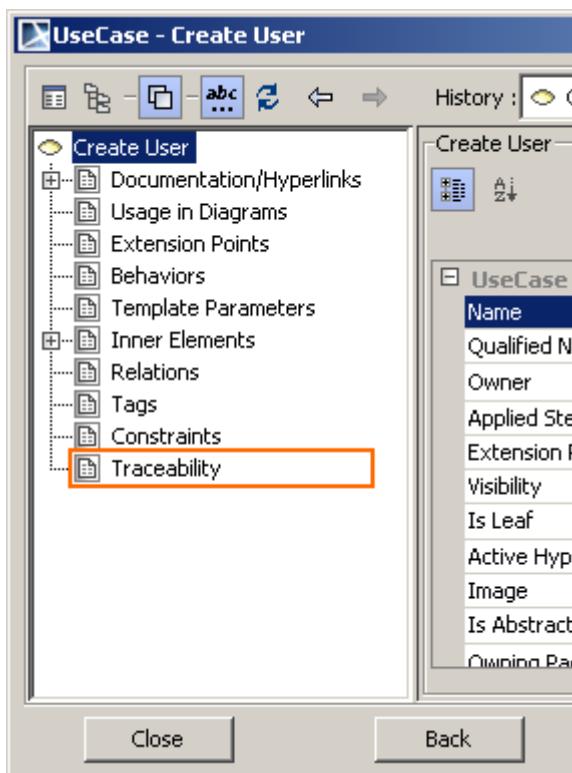


Figure 252 -- Traceability tab in the tab tree of Specification window

To view an element's traceability properties in its Specification window

1. Select an element and open the element's Specification window by using one of the ways given in section "Specification Window" on page 237.
2. Click the **Traceability** tab. Now you can view element's traceability properties.

The following picture gives an example of the traceability relations between the elements from different abstraction levels of the same project.

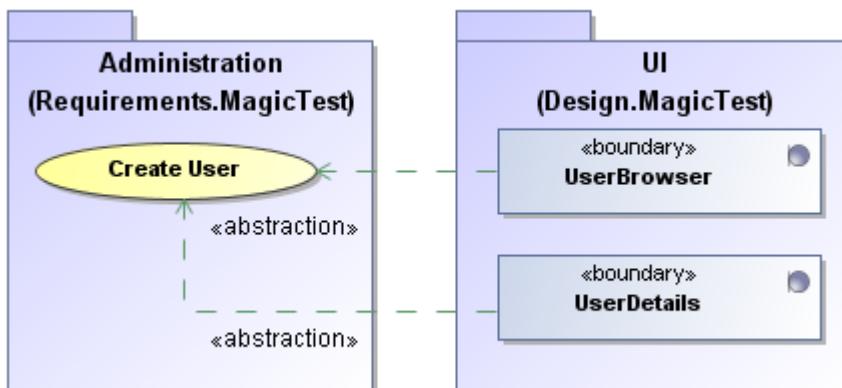


Figure 253 -- An example of single level traceability relations

As you see, the "Create User" use case from the "Requirements" package is covered by two design classes, and these classes are accordingly specified by this particular use case.

Therefore, the traceability properties such as **Realizing Class** and **Realizing Element** represent both design classes as realizing elements of this use case (see Figure 254 on page 391). Accordingly the traceability prop-

erties such as **Specifying Use Case** of each design class represent the “Create User” use case as a more specific element (see Figure 255 on page 391).

Note that the same related element can be represented through different traceability properties.

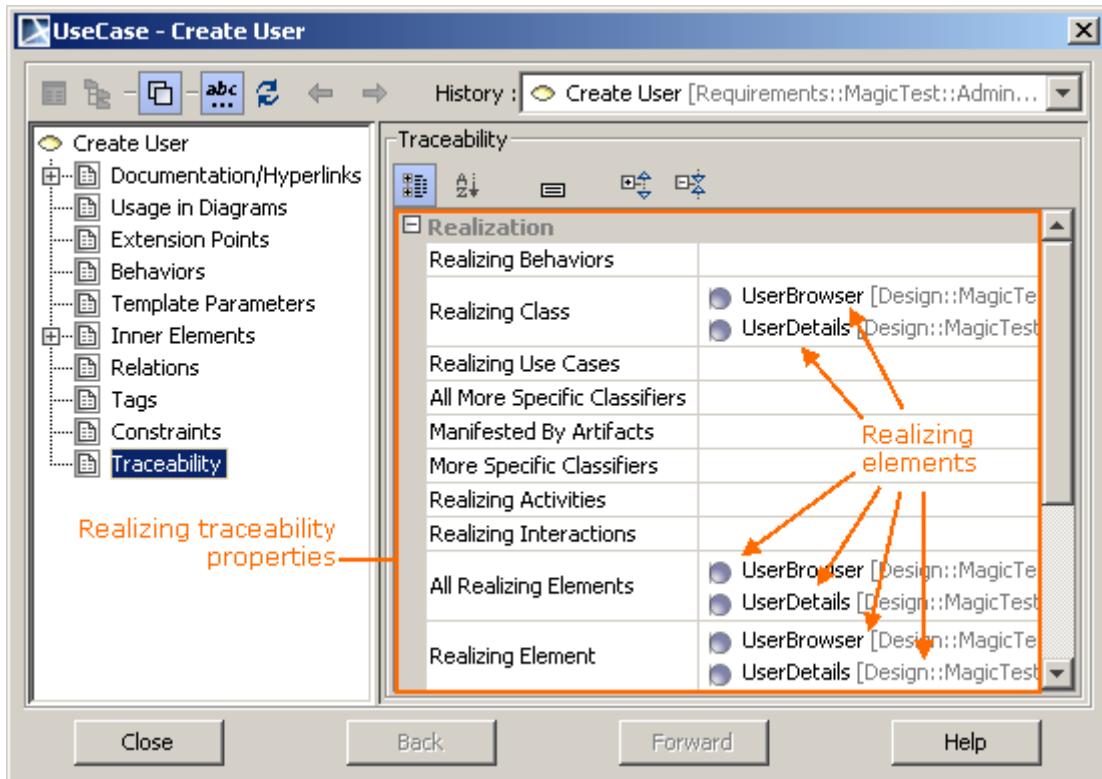


Figure 254 -- Realizing traceability properties in Specification window

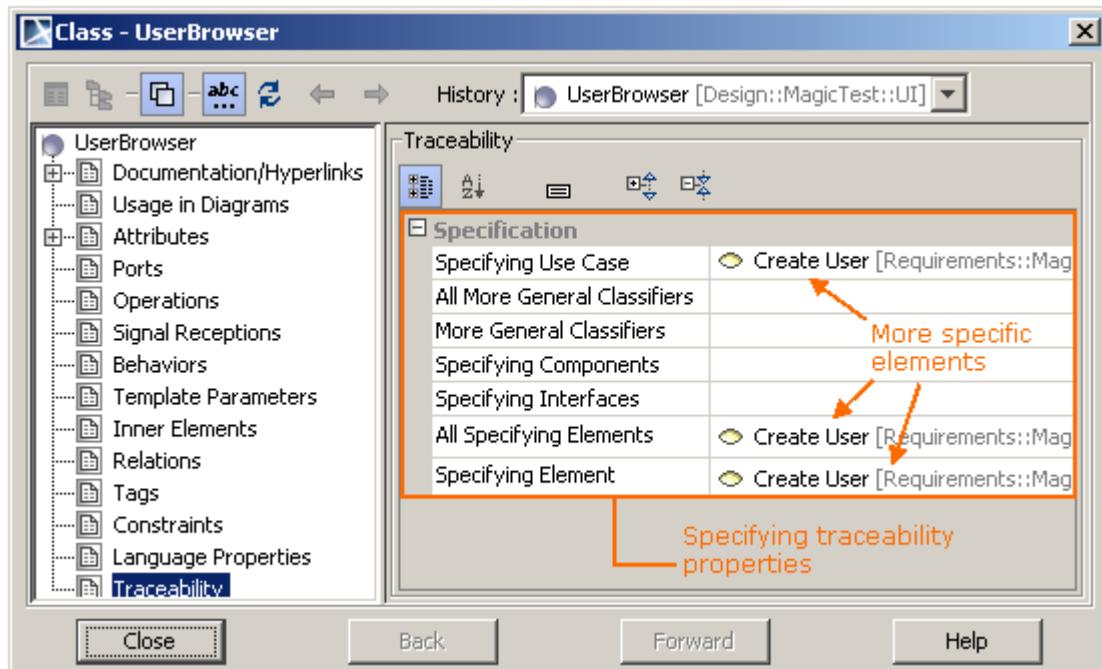


Figure 255 -- Specifying traceability properties in Specification window

For more information about element's Specification window, see “Specification Window” on page 237.

For more information about traceability properties representation, see “Traceability Relations Representation” on page 388.

Traceability properties in Properties panel

The Traceability tab in the element's **Properties** panel (at the bottom of the Browser window) is one of the places, wherein the element's traceability properties, showing its realizing and/ or more specific elements, are represented.

To view an element's traceability properties in its **Properties** panel

1. Select the element in the Containment tree or its shape on the diagram.
2. In the Model Browser, click on the **Properties** panel > **Traceability** tab. Now you can view element's traceability properties.

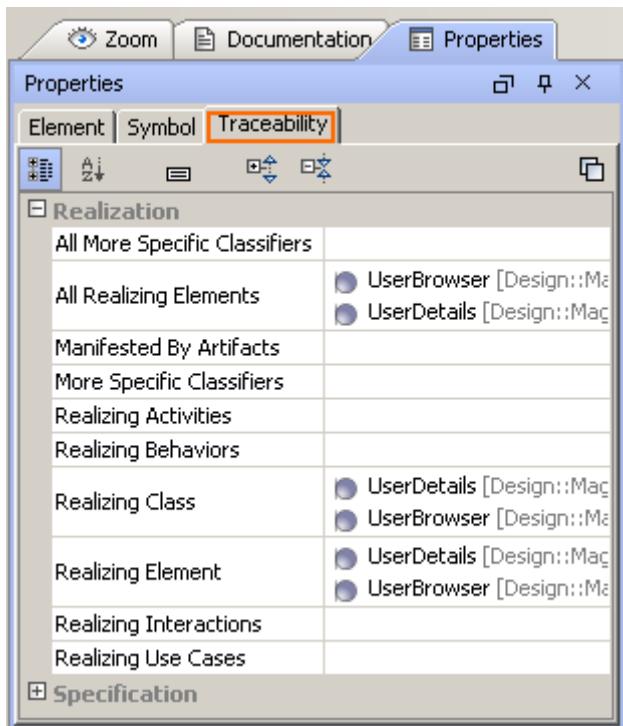


Figure 256 -- Traceability tab in Properties panel

For more information about element's **Properties** panel, see "Properties panel" on page 87.

For more information about traceability properties representation, see "Traceability Relations Representation" on page 388.

Traceability properties in notes

The traceability properties can be visualized on a diagram using the standard MagicDraw mechanism displaying the property values in notes.

To visualize an element's traceability with other elements in the note on a diagram

1. Create a note for the element, whose traceability properties you want to visualize.
2. From the note's shortcut menu, select **Edit Compartment** > **Element Properties**.
3. In the list on the left-hand side, click the traceability property of your choice (e.g., **Realizing Class** or **Specifying Use Cas**) and then click the > button.

4. Click **OK**. Now you can view the element's traceability with other elements.

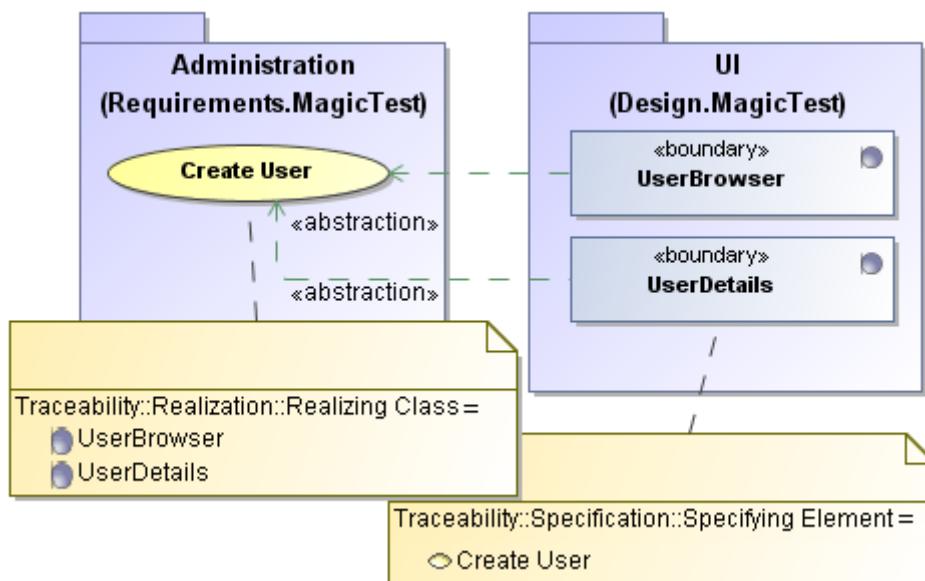


Figure 257 -- Traceability properties in the notes on a diagram

For more information about notes and comments on a diagrams, see “Note, Comment” on page 621.

For more information about traceability properties representation, see “Traceability Relations Representation” on page 388.

Traceability properties in Go To submenu

The submenu of the **Go To** menu on the element’s shortcut menu enables you to easily find and navigate to the related elements through the traceability relations.

To select an element from the higher/ lower level of abstraction in the Containment tree

1. Select the element in the Containment tree or its shape on the diagram.
2. From the element’s shortcut menu, select **Go To > Traceability > Specification/ Realization/ Other**, choose a property, and then select an element. The element will be selected in the Containment tree.

TIP!

If there are more than 10 values, the scroll bar is shown and the text box for typing the keyword is available.

For more information about traceability properties representation, see “Traceability Relations Representation” on page 388.

Traceability properties in Traceability Report

The Traceability Report feature supports the coverage analysis and publishes elements that are related to the selected elements through the traceability relations. You can generate a report either for the whole project or for a selected part of the project. This feature is the output of the coverage analysis.

Coverage analysis provides the visibility of each element’s related artifacts, indicated as realizing (lower level of abstraction) and/ or specifying (higher level of abstraction) ones.

The main objective of the Traceability Report is to visualize and verify that Analysis, Design, and Implementation model elements are all covered.

The Traceability Report provides the ability to:

- Find the areas of uncovered parts.
 - According to the report information, create additional artifacts to increase coverage.
 - Measure the coverage quantitatively.
 - Identify the redundant artifacts.

To generate a traceability report of the selected scope

1. On the **Tools** menu, click **Report Wizard**.
 2. In the **Select Template** area, select **Traceability > Traceability** and then click **Next >**.
 3. Click **Next >**.
 4. In the **Select Element Scope** area, define the scope of the report by using the buttons placed between two lists, and then click **Next >**.
 5. In the **Output Options** area, define the appropriate options.
 6. Click **Generate**. Your traceability report will be generated (generation time depends on the selected scope).

| Traceability | Traceability Report |
|----------------------|---------------------|
| Date: April 08, 2010 | Revision: 1.0 |

6. Use Case Realizing Classes

Shows the class representing the use case realization in the Design model.

| Use Case | Class |
|--|---|
|  Create User [Requirements::MagicTest::Administration] | <ul style="list-style-type: none"><li data-bbox="761 1201 1309 1217"> UserDetails [Design::magictest::ui]<li data-bbox="761 1221 1309 1237"> UserBrowser [Design::magictest::ui] |
| Coverage | 100 % |

Figure 258 -- A fragment of Traceability Report

For more information about traceability properties representation, see “Traceability Relations Representation” on page 388.

Navigating between Different Levels of Abstraction

During the development process and in order to understand the system fast, navigating between elements from different levels of abstraction is necessary. Navigating from one element to another is easy using the MagicDraw GUI capabilities of the traceability feature, if there is at least one traceability relation between them.

To navigate between different levels of abstraction, use the following features

- Element's Specification window Open the element's Specification window and select the **Traceability** tab, and then right-click on the property. Choose **Open Specification** and select an element (if there is more than one related element) as it is depicted in Figure 259 on page 395. The Specification window of this element will be opened.
- Element's **Properties** panel Open the element's **Properties** panel and select the **Traceability** tab, and then right-click on the property. Choose **Open Specification** and select an element (if there is more than one related element) as it is depicted in Figure 259 on page 395. The Specification window of this element will be opened.
- **Go To** submenu on element's shortcut menu From the element's shortcut menu, select **Go To > Traceability > Specification/ Realization/Other**. Choose a property and then select an element. The element will be selected in the Containment tree.

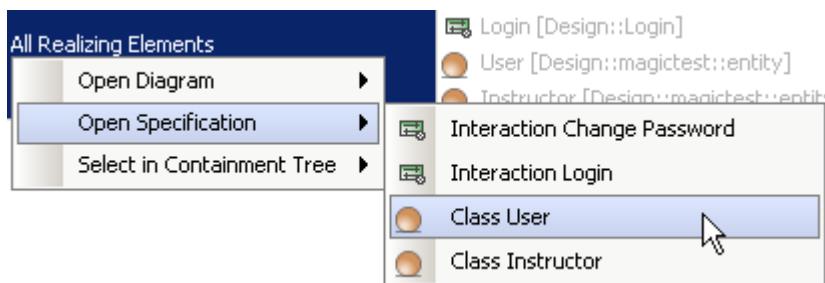


Figure 259 -- Selecting an element to open specification

Analyzing Traceability Relations

You can visualize the traceability relations among the elements in your project in order to analyze them, using the following MagicDraw features:

- **Relation Map** (for the analysis of traces among multiple levels of abstraction).
See section “Analysis using Relation Map” on page 395.
- **Dependency Matrix** (for the analysis of traces between any two levels of abstraction).
See section “Analysis using Dependency Matrix” on page 395.

Analysis using Relation Map

The Relation Map feature enables you to rapidly review and analyze multilevel relations among elements starting from the requirements to implementation all the way through different levels of abstraction (analysis, design, and so on).

For more information about the Relation Map feature, see “Relation Map” on page 398.

Analysis using Dependency Matrix

You can perform the impact and/ or gap analysis in your project using the Dependency Matrix feature, which is a powerful way for representing traceability relations between multiple elements from different packages, levels of abstraction, views, or other relations that cannot be represented on diagrams, for example, relations through UML tags.

You can create your own dependency matrices or use the special **matrix templates**, each customized for one predefined traceability property. These templates allow the appropriate realizing/ specifying traceability relations be represented in the Dependency Matrix diagram.

For a list of the predefined traceability rules, see section “Predefined Traceability Rules” on page 397.

For the detailed information about loading and using matrix templates, see “Working with a Dependency Matrix Template” on page 431.

When creating your own dependency matrices to represent traceability relations, you can define the traceability properties as **dependency criteria**.

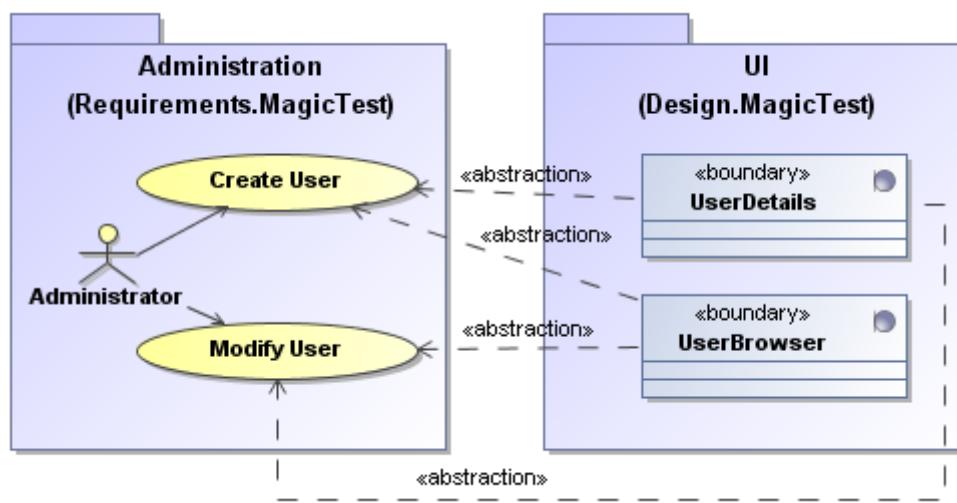


Figure 260 -- An example of traceability relations

The figure above depicts an example of traceability relations between the elements. These relations can be represented by the following traceability properties:

1. **Realizing relations (use cases -> classes):**
 - Realizing Class
 - Realizing Element
 - All Realizing Elements
2. **Specifying relations (classes <- use cases):**
 - Specifying Use Case
 - Specifying Element
 - All Specifying Elements

You can create a dependency matrix using a template suitable for each of these traceability properties.

The following steps will show you how to create a dependency matrix for the **Realizing Class** property using a predefined matrix template.

To create a dependency matrix on the template, predefined for the **Realizing Class** property

1. Create a Dependency Matrix diagram.
2. Apply the **Traceability Realization Use Case Realizing Classes** template.
3. Define new row and column scopes:
 - 3.1 For the row scope, expand **Data > Requirements > MagicTest** and select **Administration**.
 - 3.1 For the column scope, expand **Data > Design > MagicTest** and select **UI**.

4. Rebuild the matrix.

NOTE See the related sections:

- “Creating the Dependency Matrix” on page 426.
- “Working with a Dependency Matrix Template” on page 431.
- “Dependency Matrix View” on page 427.

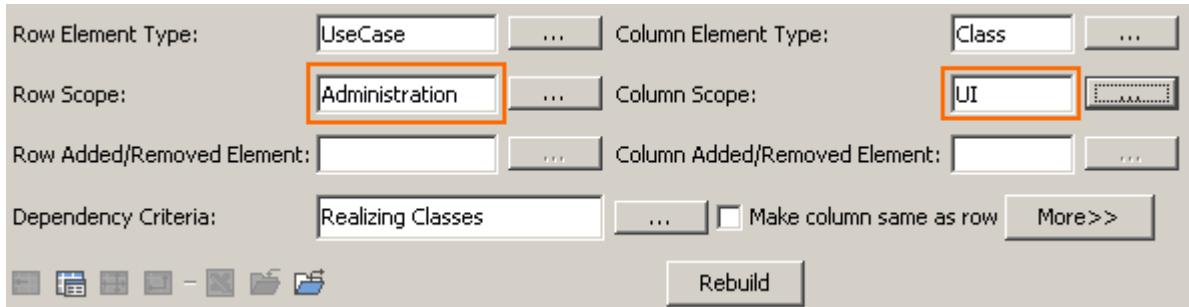


Figure 261 -- Defining row and column scopes

The picture below depicts the created dependency matrix showing the traceability relations between use cases and the realizing design classes (the highlighted intersections mark the traceability relations depicted in Figure 260 on page 396).

| | ClassDetails [Des...] | CourseBrowser [...] | CourseDetails [De...] | InstructorProfile [L...] | LoginDialog [Des...] | QuestionDetails [...] | StudentProfile [D...] | TestAssessment... | ControlButtonPa... | InfoPanel [Desig...] | PauseOptionPan... | PauseTimer [Desi... | TimeInfoPanel [D... | TestDetails [Desig...] | UserBrowser [Des... | UserDetails [Desi... |
|-------------------------|-----------------------|---------------------|-----------------------|--------------------------|----------------------|-----------------------|-----------------------|-------------------|--------------------|----------------------|-------------------|---------------------|---------------------|------------------------|---------------------|----------------------|
| Administration | 3 | 3 | 3 | 1 | | | 1 | | | | | | | 3 | 3 | |
| Create Class | | | | | | | | | | | | | | | | |
| Create Course | | | | | | | | | | | | | | | | |
| Create User | | | | | | | | | | | | | | | | |
| Modify Class | | | | | | | | | | | | | | | | |
| Modify Course | | | | | | | | | | | | | | | | |
| Modify User | | | | | | | | | | | | | | | | |
| Remove Class | | | | | | | | | | | | | | | | |
| Remove Course | | | | | | | | | | | | | | | | |
| Remove User | | | | | | | | | | | | | | | | |
| View Instructor Profile | | | | | | | | | | | | | | | | |
| View Student Profile | | | | | | | | | | | | | | | | |

Figure 262 -- An example of dependency matrix created on traceability template

For more information about the Dependency Matrix feature, see “Dependency Matrix” on page 425.

Predefined Traceability Rules

Besides allowing you to create your own traceability rules, MagicDraw enables you to use a set of predefined element properties for traceability purposes.

For detailed information about the up-to-date traceability rules, visit the following link:

http://www.magicdraw.com/show_content/new_and_noteworthy/?content=traceability_rules_16.9

Relation Map

NOTE

The Relation Map feature is available in the Standard, Professional, Architect, and Enterprise editions.

A Relation Map diagram enables you to rapidly review and analyze relations between the elements of the whole model.

The model structure in the diagram can be overviewed in two different layouts: tree (see Figure 263 on page 398) or radial (see Figure 264 on page 398).

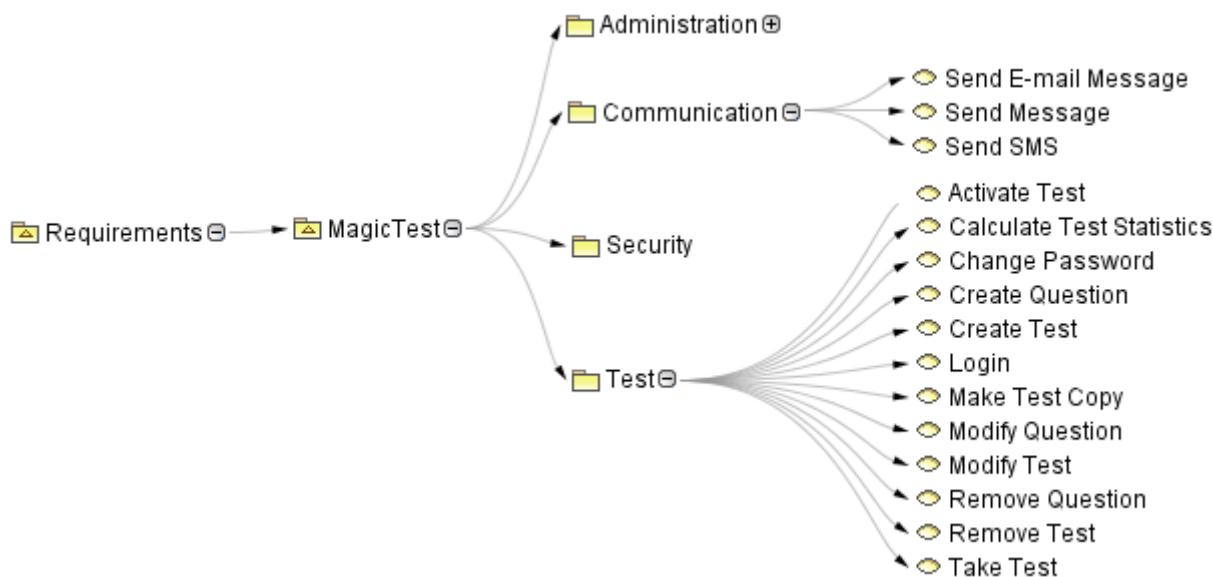


Figure 263 -- The Tree Layout of the Relation Map

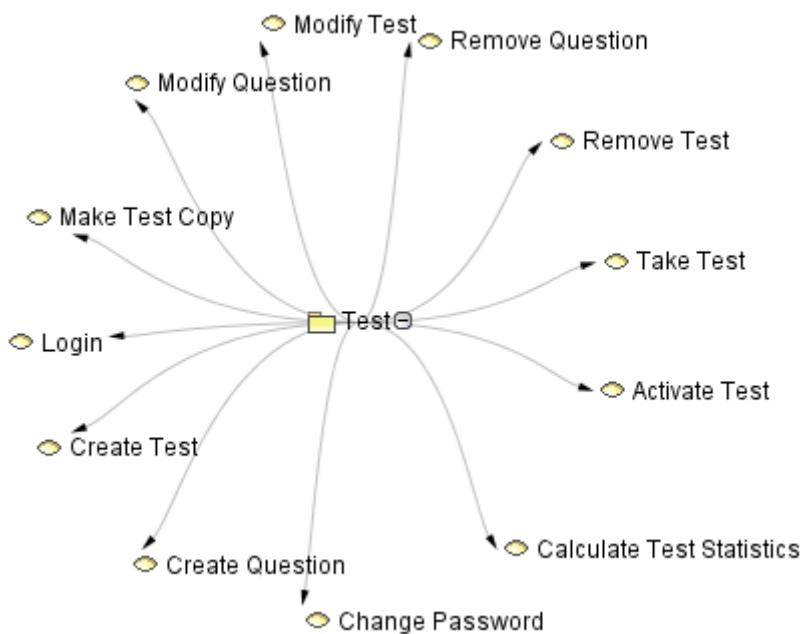


Figure 264 -- The Radial Layout of the Relation Map

Relation Map makes it easier for you to:

- Discover the existing relations of the nodes (elements) in a unique and highly usable feature giving a fast project overview in two different rendering types:
 - Dynamic – holding only needed information at a time by centering the diagram to the selected node and visualizing related elements to the set depth.
 - Static – representing and holding all the step-by-step discovered structure in a diagram. This rendering type is good for publishing, viewing structure and presentation purposes.
- Have a radial or tree layouts for visualizing multilevel relations that are suitable for hierarchical and compact representation of structures.
- Observe traceability from requirements to implementation all the way through different levels of abstraction (analysis, design, and so on).
- Have a visualized representation of relations between elements from different views.
- Have a usable and good-looking project context map.
- Make fast analysis of UML model.

Creating the Relation Map

In MagicDraw, a Relation Map is similar to a diagram element. A newly created Relation Map will appear in the Browser as a model element and you can open the Relation Map pane by double-clicking the Relation Map name. The same actions, which are relevant for diagrams, are valid for Relation Maps.

To create a relation map for a selected element

- Drag and drop an element from the Containment Tree on the created Relation Map (the structure will be created according to filters configuration).
- On the element shortcut menu in the Browser, click **Related Elements> Create Relation Map**

To create a relation map for any element

- On the **Analyze** menu, click **Create Relation Map**. Search for a context element for which you want to create a relation map and click **OK**.

NOTE

The same element will be the owner of the created relation map diagram.

- From the package shortcut menu in the Browser, select **New Diagram > Custom Diagrams > Relation Map Diagram**. Type a name and select a package where you want to save the project and then click **OK**.
- On the custom diagrams toolbar, click the **Relation Map Diagram** button. Type a name and select the package where you want to save the project and then click **OK**.
- In the **Diagrams** menu, click **Custom Diagrams > Relation Map Diagram**. The **Relation Map Diagrams** dialog box opens. Click **Add** to create a new relation map and then click **Open**.

Defining Relation Map filters, layout, and depth

After creating a relation map, you need to specify filters, layout, and depth of the diagram.

To change the relation map properties

- On the diagram shortcut menu, choose **Specification** or click the **Relation map**

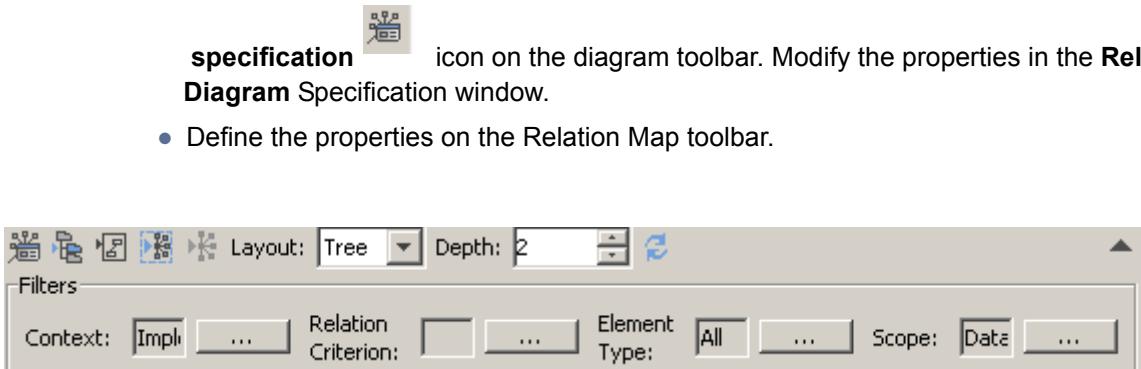


Figure 265 -- Relation Map Toolbar

| Element Name | Function |
|-------------------------------------|---|
| Context | To select the main element from which the structure is started. For more information about the element Selection dialog, see section “Selecting an Element” on page 278. |
| Relation Criterion | To select relations that will be displayed on the relation map. Relation criteria can be relationships from diagrams, properties, custom Tag definitions, OCL expressions, Meta chain expressions, or references to the code classes. For more information, see Section “Choosing the relation criterion” on page 401. |
| Element Type | To select element types that will be shown on the relation map. |
| Scope | To select packages from which the relation map structure will be shown. |
| Suppress/Expand filters area | To suppress or expand the filters area on the toolbar. |
| Layout | To select a the relation map layout: <ul style="list-style-type: none"> • Tree - see the example in Figure 263 on page 398. • Radial - see the example in Figure 264 on page 398. |
| Depth | To select the level of the relation map branches that will be automatically expanded. |

Choosing and changing context element

A context is an element from which a relation map is drawn and analysis is started. You can set any element as a context.

To set an element as a context

- Drag and drop the selected element from the Browser to the relation map diagram.
- Use the following diagram Toolbar commands:

| | | |
|--|---|--|
| | Set selected element as context | Set the selected element as a context element from which the relation map structure will be started. |
| | Make element as context on selection | If selected, the context element will be changed dynamically on the new Node selection. |

- From the Node shortcut menu on the diagram pane, choose **Set Element as Context**.

Choosing the relation criterion

The **Criterion Editor** dialog contains four tabs, each dedicated for one expression type:

- **Simple** (for the detailed information, see section “Simple expressions” on page 401)
- **OCL** (for the detailed information, see section “OCL expressions” on page 403)
- **Reference to code class** (for the detailed information, see section “Binary expressions” on page 404)
- **Meta Chain** (for the detailed information, see section “Meta Chain expressions” on page 405)

Simple expressions

Simple expressions editor, depicted in the figure below, allows expressing direct dependencies between the elements through the custom UML relations, properties, and tags, with some extra options to select the result element type, link direction, and additional filters (for relations only).

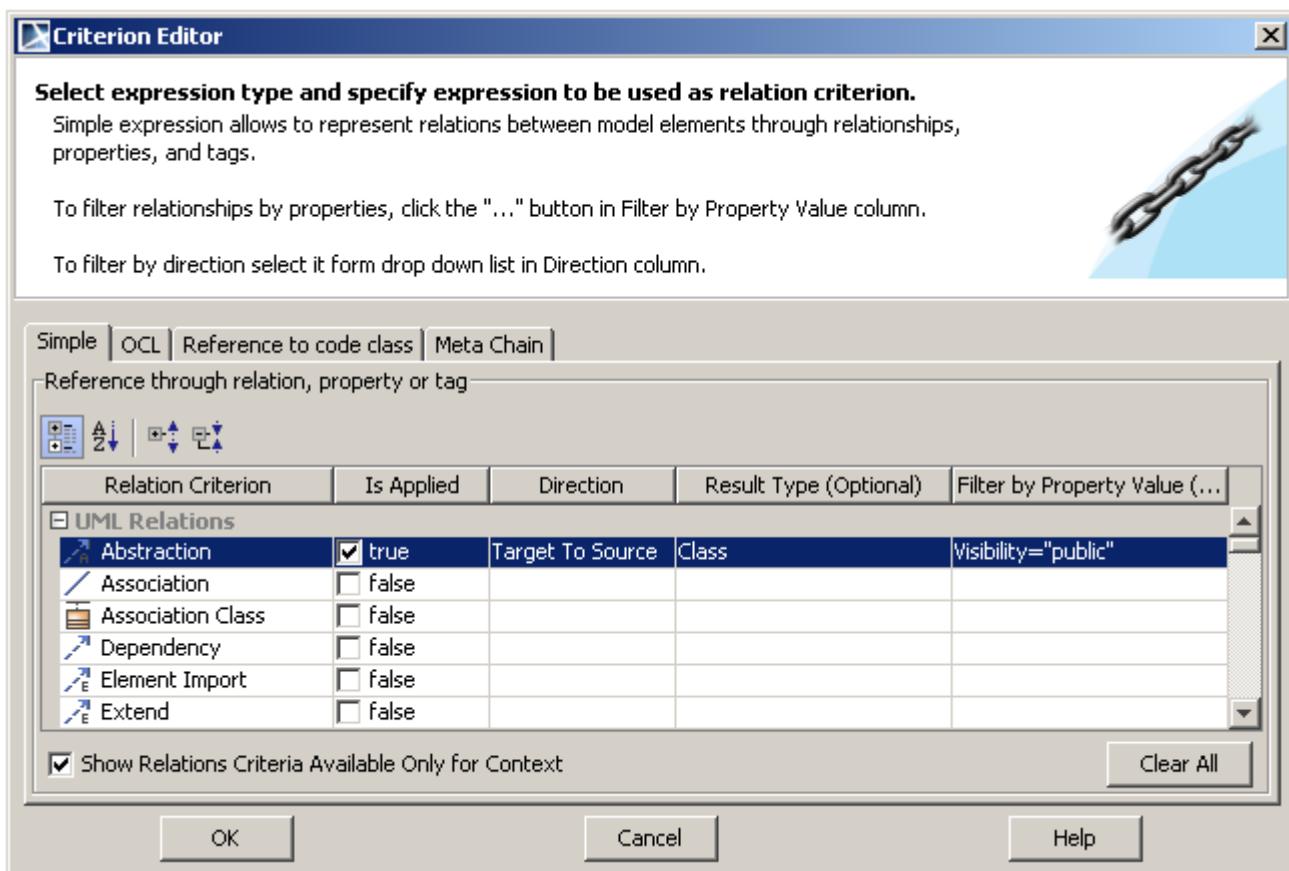


Figure 266 -- Criterion Editor dialog for defining simple expressions

| Column | Description |
|---------------------------|---|
| Relation Criterion | A list of UML relations, properties and tags. To manipulate the values display mode in the list, you can use the Show Relations Criteria Available Only for Context checkbox. |
| Is Applied | If selected, an appropriate relation criterion will be applied. The multiple selection is allowed. To clear the selections in this column, click the Clear All button. NOTE: Clicking this button will not clear the other settings (direction, result type, and filter by property) for an appropriate relation criterion. |

8 MODEL ANALYSIS

Relation Map

| Column | Description |
|---------------------------|--|
| Direction | <p>A direction for the expression analysis applied for relations, properties, and tags.</p> <p>Select one of the following values:</p> <ul style="list-style-type: none"> ● Source To Target (default) ● Target To Source ● Any <p>A source is an element, from which the expression calculations are started (a context element).</p> <p>A target is an element, which is a result of the defined expression.</p> <p>If the direction is defined for relations:</p> <ul style="list-style-type: none"> ● The Source To Target direction means that only the outgoing relations, which are pointing from the source element to the target element, will be treated as a result of this criterion. ● The Target To Source direction means that only the incoming relations, which are pointing from the target element to the source element, will be treated as a result of this criterion. ● If the Any direction is chosen, the both above described cases will be treated as a valid result. <p>If the direction is defined for properties or tags:</p> <ul style="list-style-type: none"> ● The Source To Target direction means that only the properties, which exist in the source element, will be treated as a result of this criterion. ● The Target To Source direction means that only the properties, which exist in the target element, will be treated as a result of this criterion. ● If the Any direction is chosen, the both above described cases will be treated as a valid result. |
| Result Type | <p>A result elements type. It is allowed to select more than one result type for a relation criterion.</p> <p>To open the dialog for selecting the result element types, click the “...” button.</p> |
| Filter by Property | <p>A property of the selected relation criterion used as a more specific filter. It is allowed to select more than one property.</p> <p>To open the dialog for selecting the properties as filters, click the “...” button, which is available for relations criteria only.</p> <p>NOTE: Only primitive enumeration properties and the Applies Stereotype property can be used as more specific filters for criteria.</p> |

The description of the check box in the **Criterion Editor** dialog for the simple expressions is as follows:

| | |
|---|--|
| Show Relations Criteria Available Only for Context | <p>If selected, there will be shown only these relation criteria that are available for the UML element, which is set as a customization target.</p> <p>If cleared, all the properties' tags and relationships will be shown as possible criteria.</p> |
|---|--|

OCL expressions

OCL expressions editor, depicted in the figure below, allows defining an OCL expression for gathering the collections of the result elements.

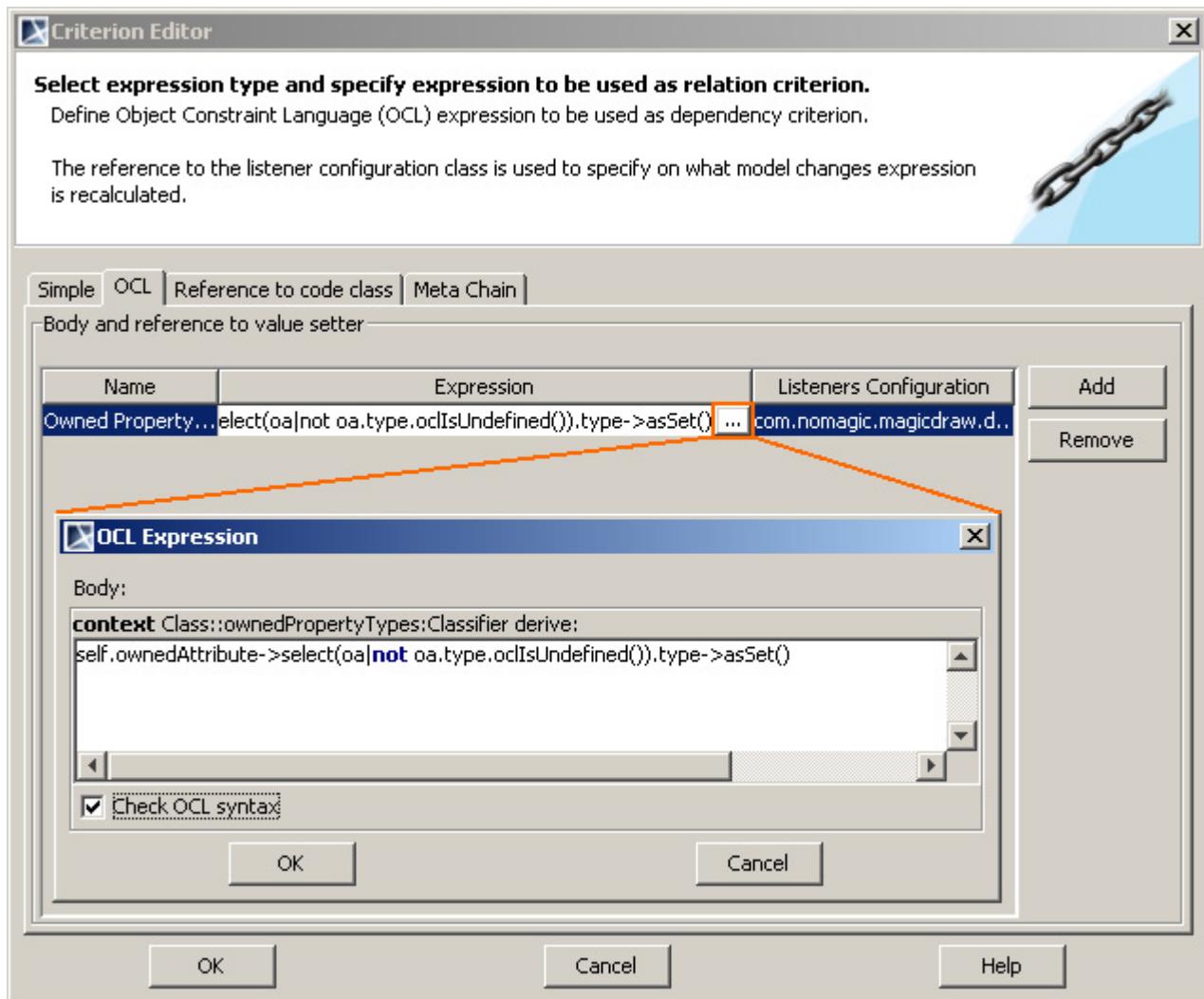


Figure 267 -- Criterion Editor dialog for defining OCL expressions

| Column | Description |
|-------------------|---|
| Name | <p>A name of an OCL expression. It is mandatory and has a default value “Untitledn”, where “n” stands for the expression sequence number.</p> <p>You can edit the name either directly in a cell, or in the Name dialog, opened by clicking the “...” button in this cell.</p> <p>To create an expression, click Add.</p> <p>To remove an expression, click Remove.</p> |
| Expression | <p>An OCL expression.</p> <p>You can edit the expression either directly in a cell, or in the OCL Expression dialog, opened by clicking the “...” button in this cell.</p> <p>NOTE: You can check the OCL syntax of your expression, when writing the expression in the OCL Expression dialog. To enable the OCL syntax checking mode in the dialog, select the Check OCL syntax box.</p> |

| Column | Description |
|-------------------------|--|
| Listeners Configuration | A reference to a java class, the so-called diagram event listener, which specifies when the OCL expression results must be recalculated. NOTE: For the instructions how to create your own listener using MagicDraw API, please, refer to the chapter "Diagram Events" in " MagicDraw OpenAPI UserGuide.pdf ". |

Binary expressions

Binary expressions editor, depicted in the figure below, allows defining a string reference to a java class, which searches for the result elements according to the given parameters.

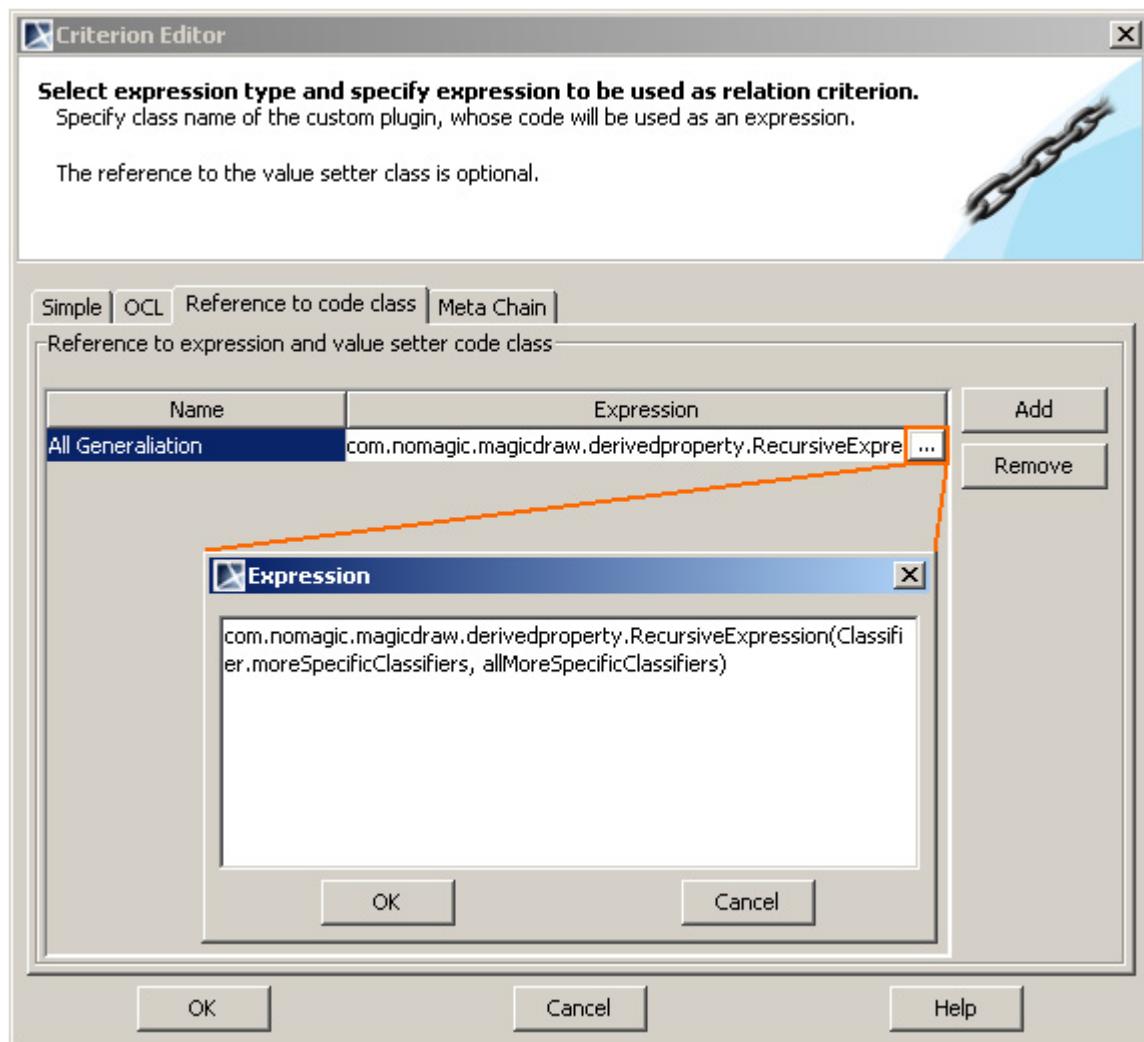


Figure 268 -- Criterion Editor dialog for defining binary expressions

| Column | Description |
|-------------------|---|
| Name | A name of a binary expression. It is mandatory and has a default value "Untitledn", where "n" stands for the expression sequence number. You can edit the name either directly in a cell, or in the Name dialog, opened by clicking the "..." button in this cell. To create an expression, click Add . To remove an expression, click Remove . |
| Expression | A reference to a java class, which specifies the element search results. |

Meta Chain expressions

Meta Chain expressions editor, depicted in the figure below, allows defining a multi properties chain navigating from a context element to a final link property for gathering the result elements.

IMPORTANT! Though the meta chain expressions allow searching for the indirectly related elements, they do not support loops and recursions.

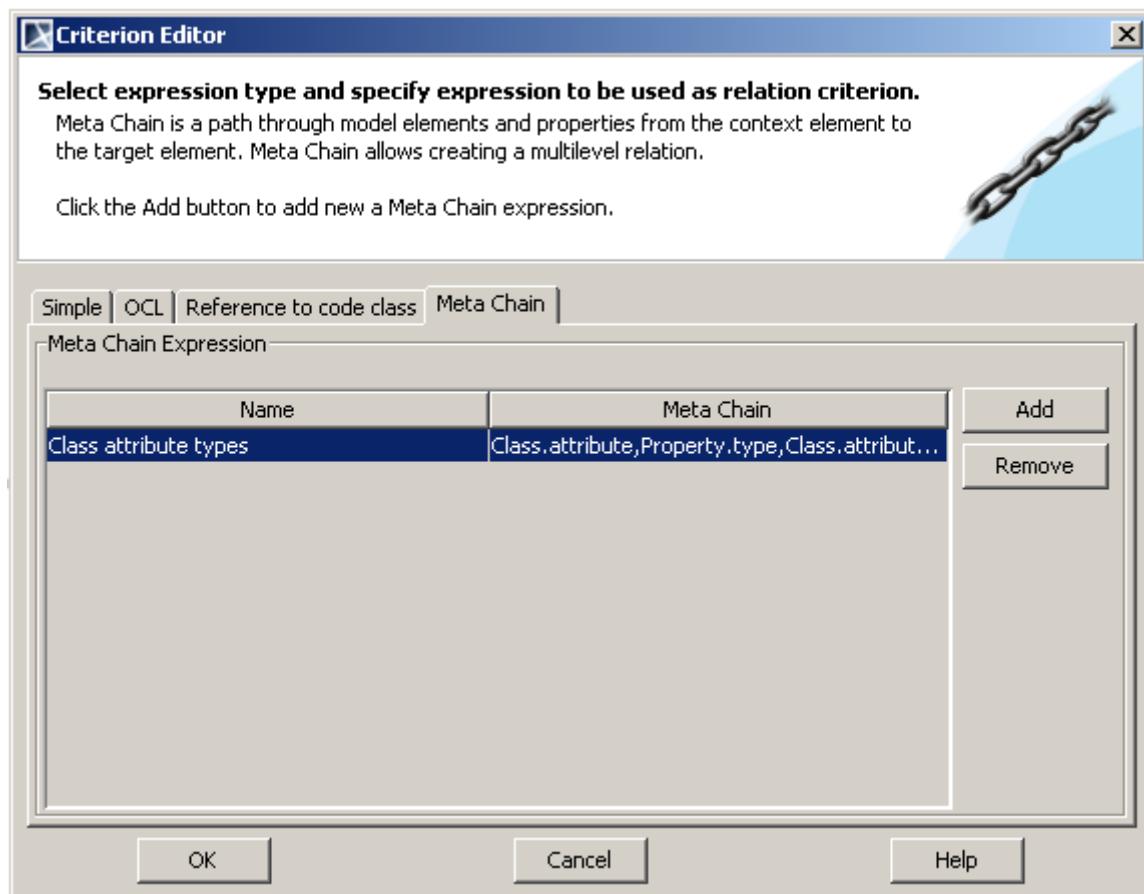


Figure 269 -- Criterion Editor dialog for defining multi properties chain expressions

| Column | Description |
|-------------------|---|
| Name | A name of a meta chain expression. It is mandatory and has a default value "Untitledn", where "n" stands for the expression sequence number. You can edit the name either directly in a cell, or in the Name dialog, opened by clicking the "..." button in this cell. To create an expression, click Add . To remove an expression, click Remove . |
| Meta Chain | A meta chain expression, containing the so-called links, i.e., pairs of a metaclass/ stereotype and a property/ tag. You can edit the expression either directly in a cell, or in the Meta Chain Expression dialog (see Figure 270 on page 406), opened by clicking the "..." button in this cell. |

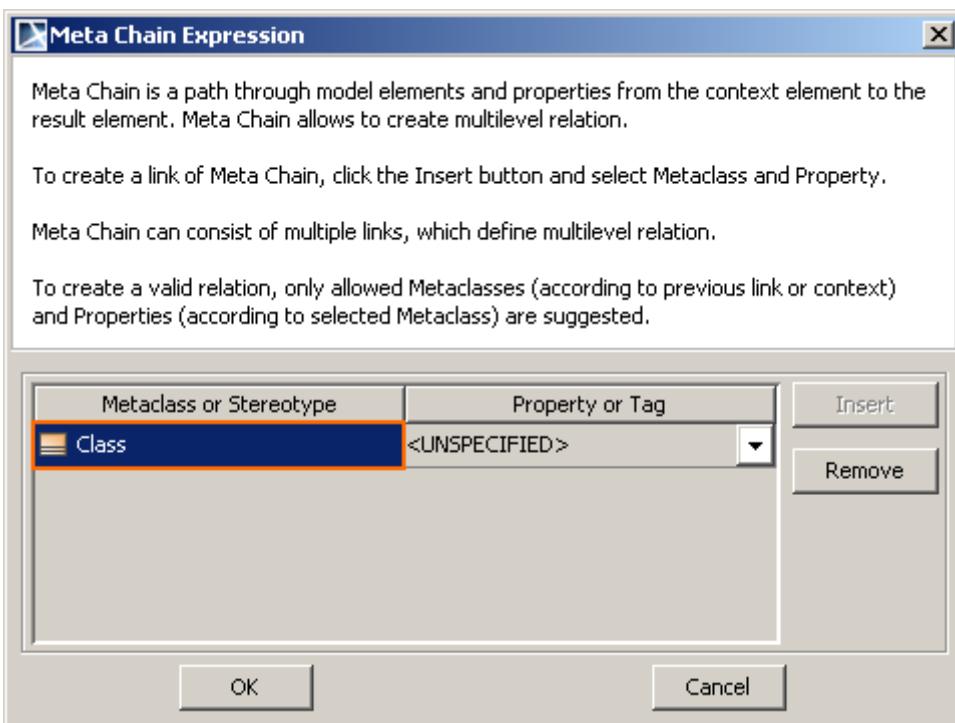


Figure 270 -- Meta Chain Expression dialog for defining links of a meta chain expression

| Column | Description |
|-------------------------|---|
| Metaclass or Stereotype | <p>A metaclass or a stereotype of a link, contained in a meta chain expression.</p> <p>To create a link in a meta chain expression, click Add.</p> <p>To remove a link from a meta chain expression, click Remove.</p> <p>The first link, when created, already has a metaclass or a stereotype selected by default as it is shown in Figure 270 on page 406. This value is the context element's type.</p> <p>The default metaclass or stereotype of the first link can be changed.</p> <p>The second link is more specific: the values suggested for selection are limited according to the values selected in the first link, since you will not be able to add a new link until the current link is not specified. The same rule is valid for the subsequent links.</p> |
| Property or Tag | <p>A property or a tag of a link, contained in a meta chain expression.</p> <p>You can choose a value from the list, which contains properties (including the derived ones) and tags specified for the link's metaclass or stereotype.</p> |

Manipulation of Relation Map

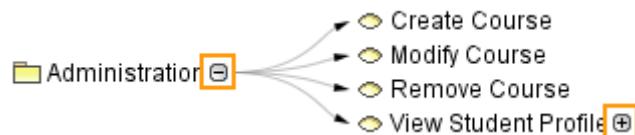
You can manipulate and analyze your relation map diagram in the following ways:

- Expand/suppress branches according to the current filters configuration.
- Restore the layout of manual suppressed/expanded branches, moved, removed and hidden symbols.
- See invalid elements marked by on demand or active validation (more information about validation you may find in Sections "Validation" on page 444 and "Active Validation" on page 462).
- Move the whole structure.

- Move the selected Node.
- Zoom in/ zoom out.
- Select an element in the Containment tree.
- Open all diagrams containing the selected element.
- Remove an element from the model.
- Hide an element in the diagram.

To expand/suppress branches

- Click the smart manipulation button near the Node.



To move the relation map structure

- Click on an empty place in the Relation Map and drag it. The whole structure will be moved.
- Click a Node and drag it. The selected Node will be moved.

To zoom in or zoom out the relation map structure

- Press Ctrl + mouse wheel scroll up to zoom in.
- Press Ctrl + mouse wheel scroll down to zoom out.

NOTE The traditional Zoom In, Zoom Out, Fit in Window, Zoom 1:1 actions are available. For more information on zooming, see Section “Zooming” on page 214.

To remove/hide an existing element

- Element removing and hiding will be initiated and work in same way as for other diagrams. Press Delete, to hide an element. Press Ctrl+D, to delete an element from the model.

To restore the layout of the manually suppressed/expanded branches, moved, removed and hidden symbols

- Click the **Restore Layout** button.

NOTE If the **Preserve manually suppressed/expanded branches and hidden elements** option in the **Environment Options** dialog > **Diagrams** > **Relation Map**, is selected, the layout of moved elements will be restored.

To see invalid elements marked by on demand or active validation

1. Run validation against model. On the **Analyze** menu, click **Validation > Validation**.
2. In the **Validation** dialog, select **Validation Suite** and click **Validate**.
3. Invalid elements will be marked on the Relation Map.

To select an element in the Browser, Containment tree

- Select an element and click the **Select in containment tree** icon on the relation map toolbar.
- On the selected element shortcut menu, click **Select in Containment Tree**.



To open all diagrams containing the selected element

- Select an element and click the **Open all diagrams containing the selected element** icon on the relation map toolbar



Symbol Usage in Diagrams

Definition The term “symbol” means a visual representation of some model element in the diagram. Symbols are further subdivided into shapes and paths (paths are lines used in the model for representing various relationships).

The symbol usage in diagrams functionality allows the user to see the usage of a symbol throughout the diagrams of a project. You may search for symbol usage in diagrams from the element Specification window and from element shortcut menu.

To search for diagrams in which symbol is used from the element Specification window

1. Open the element specification dialog box and select **Usage in Diagrams**. The table shows all diagrams in which the symbol is represented.
2. Select the diagram you want to open.
3. Press the **Open** button. The diagram is opened and the symbols of the element are selected. If the diagram includes more than one of the same element symbol, all symbols of the same element are selected in the diagram.

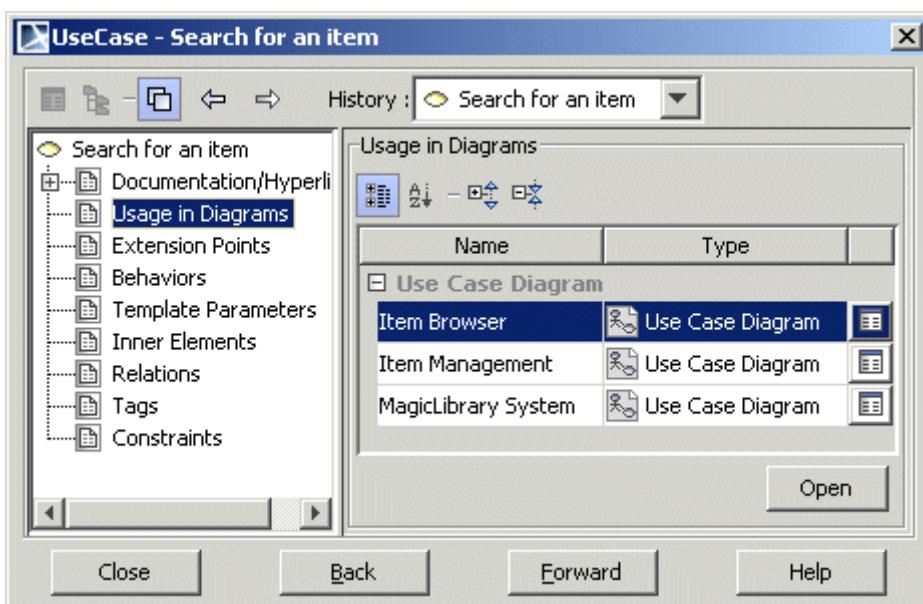


Figure 271 -- The element specification dialog box, the *Usage in Diagrams* branch

To open the diagram specification dialog box, press the  button near the diagram in the list.

For more information about working with the element's Specification window, see "Specification Window" on page 237.

To search for diagrams in which symbol is used from the element shortcut menu

1. Select the element in the Browser or select the symbol in the diagram.
2. From the element shortcut menu, select **Go To** and then **Usage in Diagrams**.
3. Select the diagram that you want to see the particular element symbol. The diagram is opened and symbols of the current element are selected on the diagram.

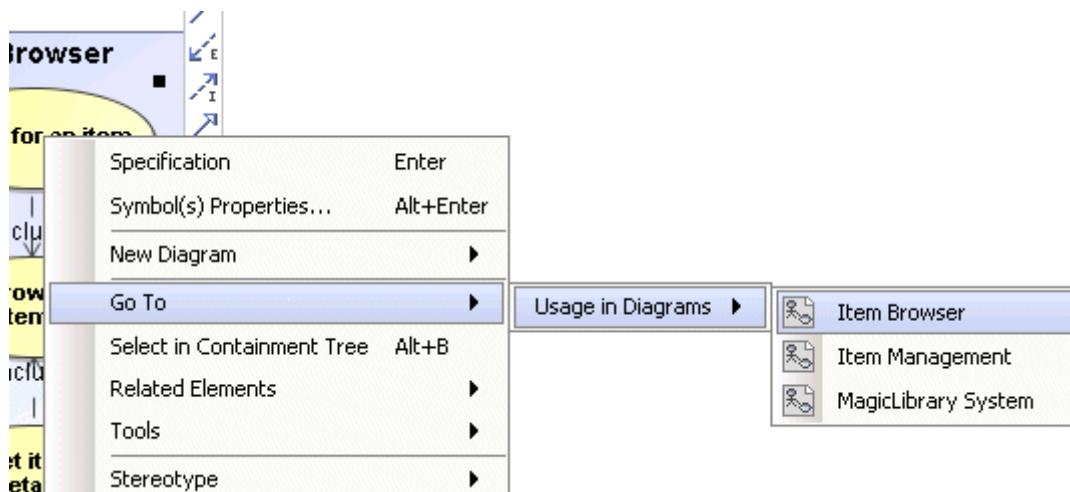


Figure 272 -- Searching for symbol usage in diagrams from the element shortcut menu

TIP!

You can also search for symbols usages in diagrams in the following way:

1. Select the symbol on the diagram.
2. From the **Analyze** main menu, select the **Go To** command and then **Usage in Diagrams**.

NOTES

- All symbols of the same element are selected in the opened diagram.
- If more than one symbol exists in the same diagram, then the diagram is zoomed out, to fit the view in screen.

For information about the Usages /Dependencies functionality for analyzing associations between elements, see "Analyzing Usages and Dependencies" on page 382.

Projects Comparison

[View Online Demo](#) Visual Model Differencing

NOTE This functionality is available in Architect, and Enterprise editions only.

The use of projects comparison functionality allows you to compare two MagicDraw UML local projects or teamwork project versions, as well as diagrams. Model elements are compared by the element ID.

You can compare:

- current project with locally saved project.
- current project with open project.
- two teamwork project versions.
- current project with teamwork project version.
- local project and teamwork project version.
- diagrams.

Comparing Projects

First of all you have to select the projects you want to compare. Differences of the compared projects will be displayed as two model trees. Differences are marked using colors and highlighting.

When comparing two projects the following data changes will be reflected:

- New model elements.
- Deleted model elements.
- Model elements with modified data.
- Model elements that changed location.
- Inner elements changes.

To select the projects you want to compare

1. Open a project you want to compare and from the **Analyze** menu, select **Compare Projects**.
2. The **Compare Projects** dialog box appears.



Figure 273 -- Select Projects to Compare dialog box

3. From the **Active Project** box, select a project you want to compare with another project. The list contains names of all open projects in MagicDraw.
4. From the **Compare With** box, select a project with which you want to compare the first project. The following options are available:
 - open project names are listed, except the project, which is selected in the **Active Project** box.
 - **Local Project**. Click the "..." button and select a project version you want to compare the current project with.
 - **Teamwork Project**. Click the "..." button and from the Open Teamwork Project dialog box, select a project and its version you want to compare with.
 - <current project>Disk version. By selecting this option you will be able to compare currently changed project version with the unchanged project version on the disc without current changes.

By default the first opened project should be selected. If there are no more projects open, Local Project is selected as the value.

Understanding model differences

All differences of the compared projects are displayed in the **Difference Viewer** dialog box.

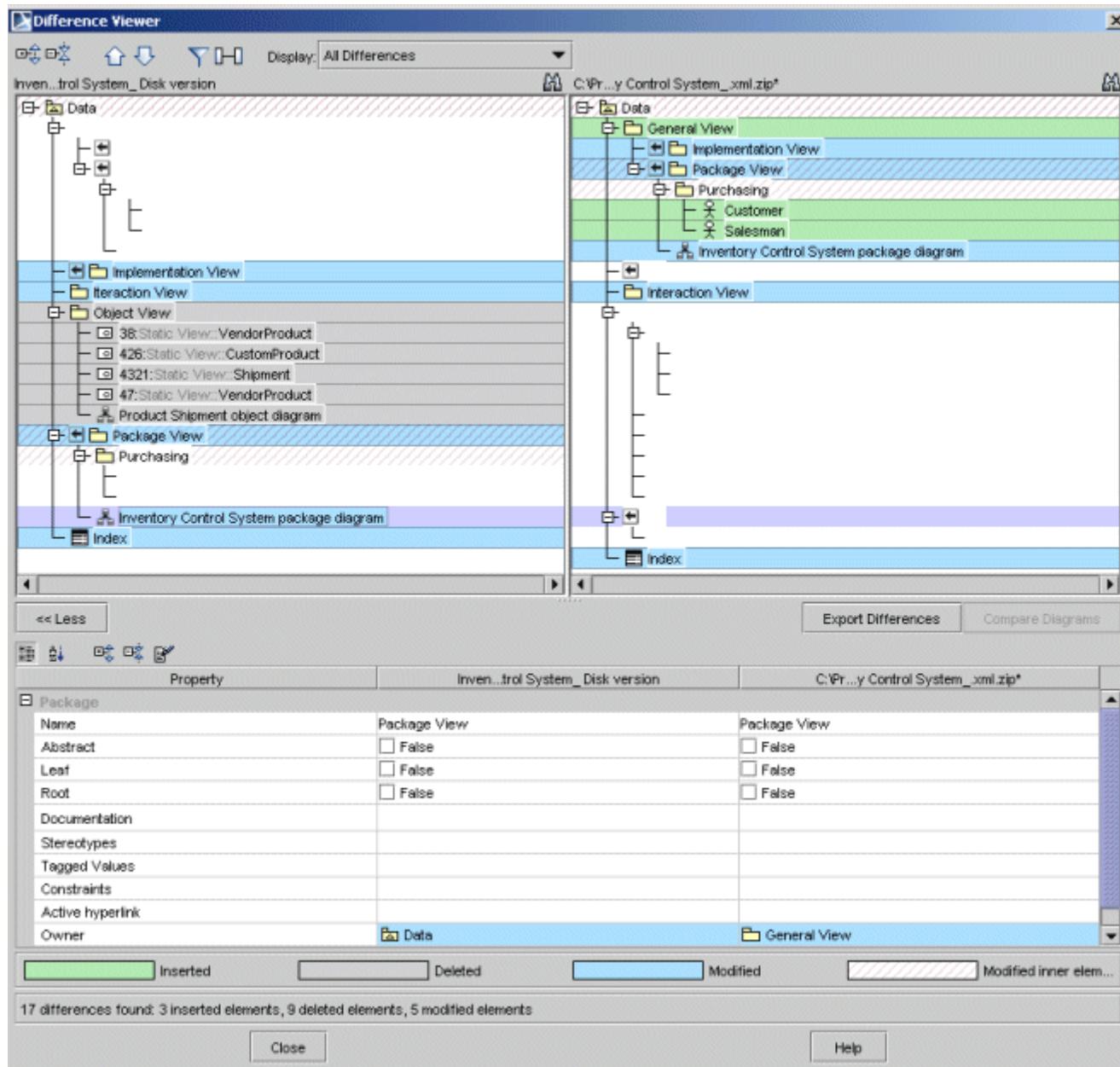


Figure 274 -- Difference Viewer dialog box

In the **Difference Viewer** dialog box, two compared projects are displayed. The number of differences is displayed at the bottom of the dialog box. The number of differences number includes inserted, deleted, and modified elements. The number does not include elements with changes to inner elements. Differences number is displayed in following order:

To mark changes in the model elements several colors are used:

- *Elements that do not exist in the other model (inserted elements)*. Element is displayed only in the right-hand tree. New element is highlighted in light green.

- *Elements that exist in the other model, but do not exist in current model (deleted elements).* Elements are always displayed in the left-hand tree. The deleted element is highlighted in grey.
- *Elements with modified element data (modified elements).* The modified element is highlighted as changed in both trees. The modified element is highlighted in light blue.
- *Elements that changed location (parent has changed).* Element is marked as a modified element. Empty nodes are displayed in the opposite tree where the element does not exist. On the moved element and on its former position, a button is displayed. Pressing the button on the former position, selects the place where the element has moved to. Pressing the button on the moved element position, selects the former element place. Also, you can perform these operations using the shortcut menu commands **Go to former position** or **Go to moved element**.
- *Elements that have inner elements that changed.* The element is marked in both trees and is highlighted in light grey dashes. An element with modified element data and changed inner elements is marked as modified and as element with changed inner elements. Element is marked as modified when:
 - Element specification properties have changed. Element specification properties include all properties, which are not displayed in the Browser as separate elements. Model element specification properties are treated as changed only if the element property can be changed from the element specification. If an element specification has changed because of changes made to other elements, the element should not be treated as changed. Example: typed values in the tagged values specification, attribute links in an object and instance specification, etc.
 - Element parent has changed.
 - Path is drawn from/to element.

Buttons available in the **Difference Viewer** dialog box:

| Button | Function |
|--|---|
|  Expand All | Both trees are expanded. |
|  Collapse All | Both trees are collapsed. |
|  Go to Previous Difference | Select the difference listed prior to the current one. |
|  Go to Next Difference | Select the difference listed next from the current one. |
|  Filter | The Filter dialog box opens. Show/hide the elements you want to analyze. |
|  Include Relation Ends | Relations are displayed in the elements specifications. A relation added to the element means that the element is marked as modified. |

| Button | Function |
|--|---|
| Display: | <p>All - Shows all elements of the projects.</p> <p>All Differences - Shows only differences made between the projects.</p> <p>Deleted Elements - Shows only elements that were deleted from the projects.</p> <p>Inserted Elements - Shows only elements that were inserted in the projects.</p> <p>Modified Elements - Shows only elements that were modified.</p> |
|  Find Next Element | The Find dialog box opens. Search for elements within the corresponding project. |
| More>> <<Less | More/Less button shows/hides the element properties table. By default, the properties table is hidden. |
| Export Differences | <p>Creates .html or .txt differences report. In the Save Difference Report As dialog box, select the directory where you want to save this report.</p> <p>NOTE: *.html format is suitable for viewing the difference report. If you want to copy this report to another program, use of the *.txt format is recommended.</p> |

The first column of the Property window contains the same properties as the Quick Properties tab in the Browser. The second column title is the left-hand project name (with path) for local projects, and teamwork project name and version number for teamwork projects. The third column title is the right-hand project name (with path) for local projects, and teamwork project name and version number for teamwork projects. Modified properties are marked with the same color as in the model element tree.

Diagrams Comparison

To compare diagrams

1. In the **Difference Viewer** dialog box, select the diagram you want to compare and click the **Compare Diagrams** button.

- The **Diagrams Difference Viewer** dialog box opens, which displays both diagrams with changes made in them.

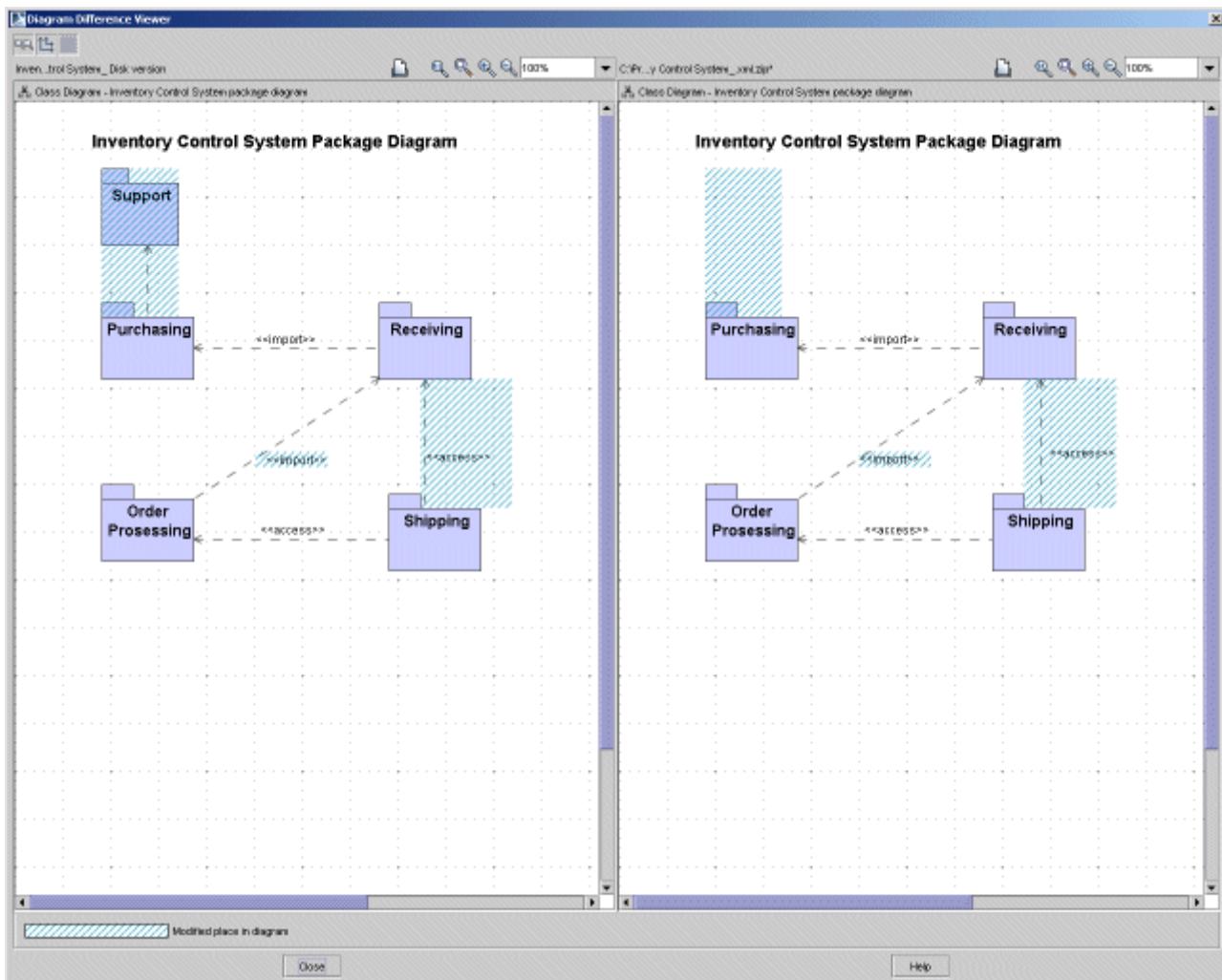


Figure 275 -- Diagram Difference Viewer dialog box

Diagrams Difference Viewer displays two diagrams:

- Current (or first opened) project diagram is displayed at the right-hand side.
- Diagram that is compared with is displayed at the left-hand side.

Symbol changes that are reflected in the diagram:

- Modified symbol properties
- New symbol creation
- Symbol deletion
- Symbol bound changes (resize, bound changes because of element properties changes)

All changes are highlighted in light blue dashes.

Buttons available in the **Diagrams Difference Viewer** dialog box:

| Button | Function |
|---|---|
|  | If pressed, zooming affects both diagrams: zooming one diagram causes zooming of the other diagram. |
|  | If pressed, scrolling one diagram causes scrolling of the other diagram. |
|  | If pressed, places where diagram has changed are marked on both diagrams. |
|  | Prints the corresponding diagram together with marked changes. |
|  | Zooms the corresponding diagram(s) to the original size. |
|  | Zooms the corresponding diagram(s) to the size that fits the window. |
|  | Zooms the corresponding diagram(s) in. |
|  | Zooms the corresponding diagram(s) out. |
| <input type="text" value="100%"/>  | Select the percentage for zooming the corresponding diagram(s). |

Metrics

NOTE This functionality is available in Architect and Enterprise editions only.

MagicDraw metrics functionality, which is implemented as a plugin and accessible through the Open API, enables you to measure your project from three different viewpoints:

- UML model metrics
- System metrics
- Requirements metrics.

Using UML model metrics, you can measure your project using package, class, and diagram measurements (for example, measuring the number of classes, inheritance tree depth, and so on).

System metrics analyze models using the most popular object oriented project metrics: Halstead, McCabe, Chidamber, and Kemerer defined metrics (for example, cyclomatic complexity and weighted methods per class).

Requirement metrics consist of function points and use case metrics. These two metrics groups are so structurally similar that use case metrics are regarded as a subset of function point metrics. Use case metrics measure both the number of use cases in a project and the user case analysis through selected tagged values (priority, for example).

The results of these analyses are displayed in a table, where you can select which metric you would like displayed. You can also export the metrics to a separate file.

Metrics are implemented as a plugin and are accessible through the Open API.

A metric is a numeric value that measures a model or is counted according to model measuring. Each metric has both a lowest and highest limit specified. Metrics that fall outside of this range are marked:

- Values that are too low are displayed in a blue font.
- Values that are too high are displayed in a red font. **Note:** if the highest limit equals zero, the metric is never marked as too high.

Metric Suites

In MagicDraw, you can create your own metric suites or use one of the three predefined metric suites: *System Metrics*, *UML Model Metrics*, or *Requirements Metrics*.

The metric suites contain a list of metrics that will be counted and the properties specified for each selected metric.

To create your own metric suites, clone an existing suite and specify the suite properties. You can edit the pre-defined metrics suites, and all metric suites can be imported or exported, facilitating the exchange of ideas with other users.

Displaying Metrics

Metrics are counted according to properties defined in a selected metric suite and can be counted for an entire project or just the selected packages, classes, interfaces, or diagrams. The results are displayed in the Metrics window, which opens at the bottom of the MagicDraw application window.

The Metrics window contains two tabs:

- **Data** tab. The counted metrics of the selected suite are displayed in a metrics table, which includes counted metrics only.
- **Graphics** tab. The selected metric is displayed as a graphic.

Metrics tables display packages, classes, interfaces, and diagrams. Additionally, elements that contain packages, classes, interfaces, and diagrams, which are displayed using a tree structure, are not counted for these elements.

The following is an example of a metrics table structure:

| Model Element | Metric1 | Metric2 | Metric3 | ... | | MetricN |
|---------------|---------|---------|---------|-------|-------|---------|
| PackageA | value | value | value | value | value | value |

The following is an example of a metrics table structure:

| | | | | | | |
|--------------|-------|-------|-------|-------|-------|-------|
| Inner class1 | value | value | value | value | value | value |
| Inner class2 | value | value | value | value | value | value |

If a value is not counted for a class, interface, package, or diagram, the cell is left empty.

You can apply the following filters to the metrics table:

- All
- Packages
- Classes (classes and interfaces are displayed)
- Diagrams
- Package Violations (only rows that contain package violations are displayed)
- Class Violations (only rows that contain class or interface violations are displayed)

When the Classes, Diagrams, or Class Violations filters are selected, the owner is displayed next to the following element: c1 (Classes::Package1)

Starting Metrics

To start Metrics:

Click **Metrics** on the **Analyze** menu, or on the class/package/interface/diagram shortcut menu, click **Tools** and then click **Metrics**.

The **Metrics** dialog box opens.

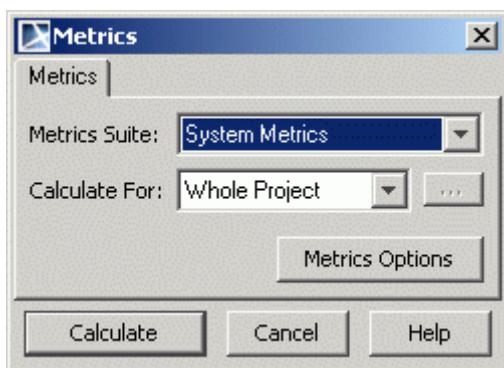


Figure 276 -- Metrics dialog box

| Element | Description |
|------------------------|---|
| Metrics Suite | Lists all the available metrics suites. |
| Calculate For | Lists two values: <ul style="list-style-type: none"> • Whole Project – calculates metrics for the entire project. • Selection – calculates metrics for selected items only. Click the "..." button to open the Select Elements dialog box. |
| Metrics Options | Opens the Metrics Options dialog box. |
| Calculate | Opens the Metrics window. |

To set element options for Metrics, click the "..." button to open the **Select Elements** dialog box:

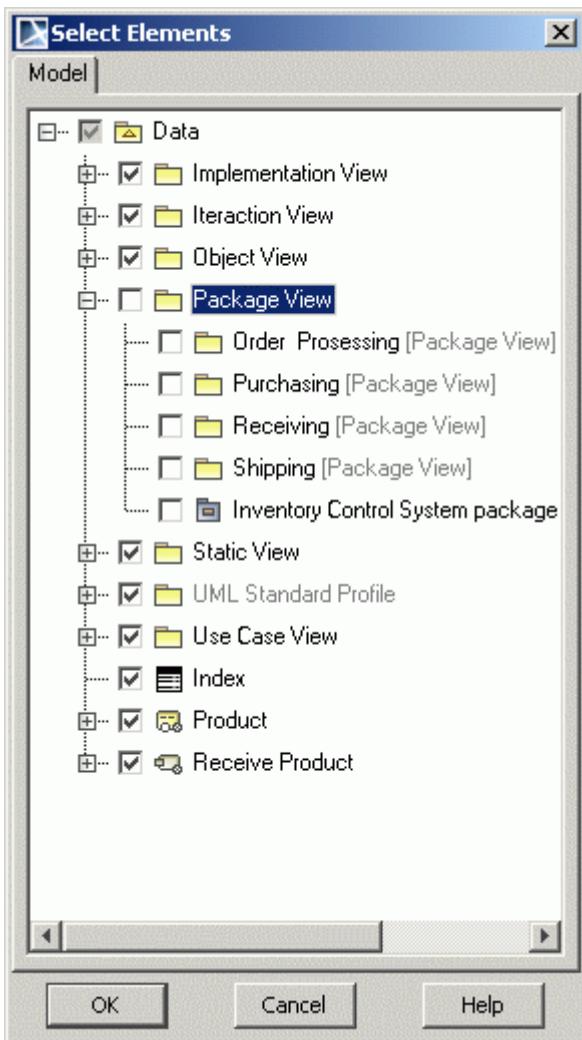


Figure 277 -- Select Elements dialog box

Packages, classes and diagrams are displayed in the **Select Elements** dialog box. If you select the box next to a parent element (for example, the *Data* check box in the image above), all its related child elements are automatically selected. Conversely, clearing the box next to a parent element clears all its related child elements.

If you clear the box next to a child element, the parent box is also cleared. For example, if the *Data* box is selected, all its related child elements are selected. If you then clear the *Package View* box, its child elements are also cleared, as is the box next to *Data*, but all the other boxes remain selected.

Metrics window

The **Metrics** window is implemented as a JIDE GUI window. Like the **Messages** window, it is available at the bottom of the MagicDraw application window.

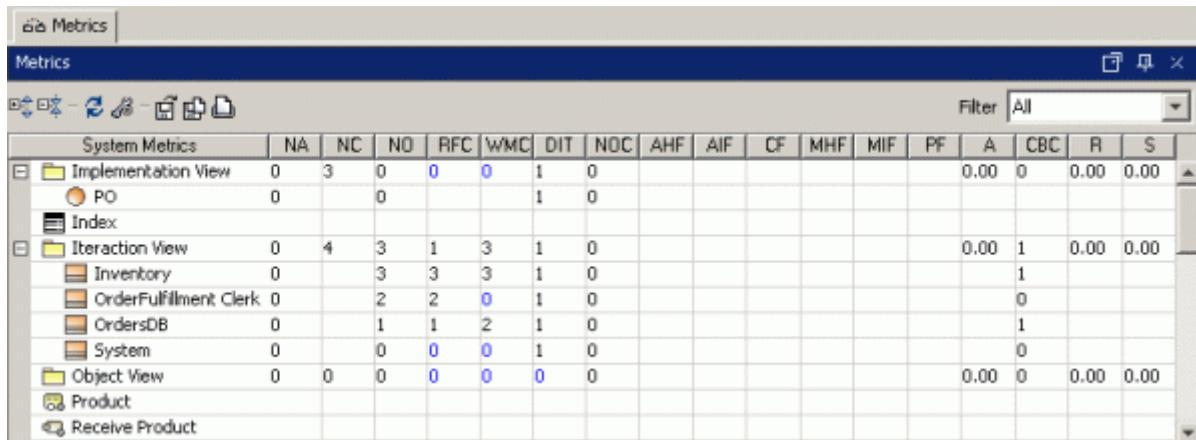


Figure 278 -- Metrics window

| Element name | Description |
|--------------------------------|--|
| Expand Current Branch | Expands all elements in the selected branch within the results table. |
| Collapse Current Branch | Collapses all elements in the selected branch within the results table. |
| Refresh | Recalculates metrics results according to the current model. |
| Metrics Options | Opens the Metrics Options dialog box. |
| Export | Opens the Export Metrics dialog box. |
| Compare Metrics With... | Opens the Open dialog box, where you can select a text file to compare with the currently open metric set. |
| Print | Prints the metrics table. The Print dialog opens. |
| Filter | Contains these values: <ul style="list-style-type: none"> • All • Packages • Classes • Diagrams • Package Violations • Class Violations |

The selected metrics rows or cells can be copied to the clipboard by clicking **Copy** on the shortcut menu or by **Ctrl+C** on your keyboard.

Exporting Metrics

You can export the selected metrics rows and columns, or the entire metrics table, to a metrics results file. Metrics results can be exported using *.txt and *.html formats.

In the following example, metrics are presented in *.txt format and are separated by tabs:

| Element | Metric1 | Metric2 | Metric3 | | MetricN |
|---------|---------|---------|---------|------|---------|
|---------|---------|---------|---------|------|---------|

8 MODEL ANALYSIS

Metrics

| | | | | | |
|---------------------------------|-------|-------|-------|------|-------|
| Package Package1 | value | value | value | | Value |
| Class class1 (Package1::class1) | value | value | value | | Value |
| Class class2 (Package2::class2) | value | value | value | | Value |

Here *Metric1 .. MetricN* – the metric name abbreviation.

Technical information is displayed at the bottom of the file. Text "Element IDs" are added after the metrics of an element and are also printed. This information is needed for metrics comparison.

In the following example, information is presented in *.html format:

Metrics Report

| Element | <u>Metric1</u> | <u>Metric2</u> | <u>Metric3</u> | | <u>Metric N</u> |
|--|----------------|----------------|----------------|------|-----------------|
| Package Package1 (<i>Package1</i>) | value | value | value | | Value |
| Class class1 (<i>Package1::class1</i>) | value | value | value | | value |

Here *Metric1 .. MetricN* – the metric name abbreviation.

Each metric name is hyperlinked with its metric description. Metric descriptions can be opened in a separate window after clicking the hyperlink.

NOTE *.html format is best suited for viewing metrics. If you want to copy the metrics table to another program, use of the *.txt format.

To export Metrics

Click the **Export Metrics** button in the **Metrics** window. The **Export Metrics** dialog box opens:

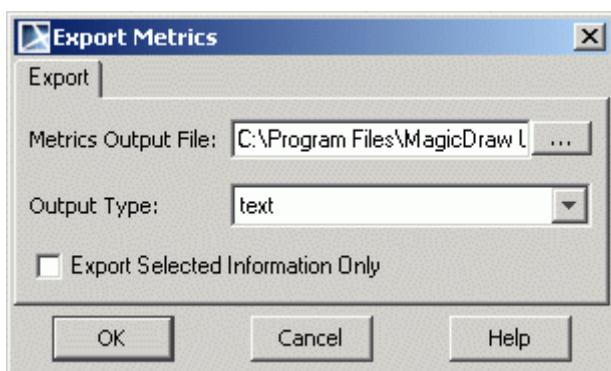


Figure 279 -- Export Metrics dialog box

| Element | Description |
|-----------------------------|---|
| Metrics Output File: | Displays the path and file names of the metrics results output file. Click the "..." button to select the location and file. |
| Output Type: | Contains these values: <ul style="list-style-type: none">• Text (*.txt)• HTML (*.html) |

| Element | Description |
|----------------------------------|---|
| Export Selected Rows Only | When selected, only the selected table rows and the header row are exported. When cleared, the entire metrics table is exported. |

Comparing metrics

Counted metrics can be compared with metrics that are saved in a *.txt file. Metrics can be compared only when the metrics window is open.

Comparison results are displayed in the same metrics table. If a cell contains a metric that has increased, it has a red fill color. If the metric has decreased, a blue fill color is used. Metrics that are not found in other file cells have a grey fill color.

The metrics comparison can be canceled using the ESC key.

Metrics Options

Metrics suites are managed in the **Metrics Options** dialog box.

To open the **Metrics Options** dialog box:

- From the **Analyze** menu, select **Metrics** and then **Metrics Options**
- In the Metrics dialog box, click **Metrics Options**.

The left pane of the **Metrics Options** dialog box displays the defined metrics suites. Using the buttons or short-cut menu, metrics suites can be cloned, renamed, removed, exported, and imported. Predefined metrics sets cannot be renamed or removed.

The suite properties are displayed in the right pane:

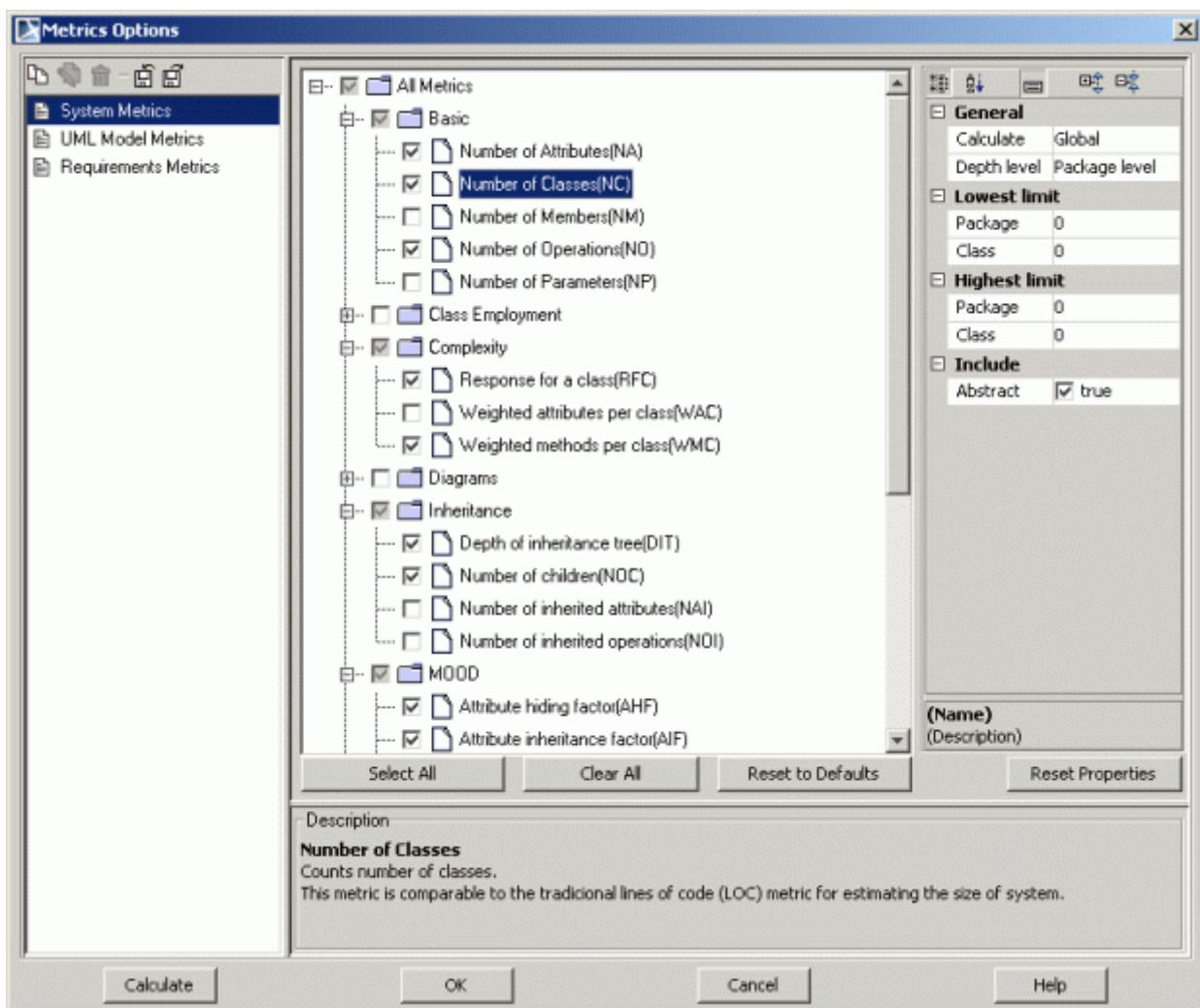
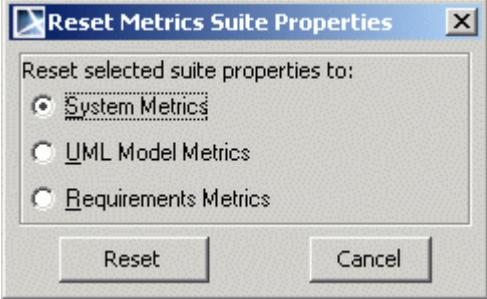


Figure 280 -- Metrics Options dialog box

| Element | Description |
|--|---|
| Metrics suites list | Displays all created metrics suites in a list. |
| Metrics suites list buttons: Clone Rename Remove Import Export | <ul style="list-style-type: none"> ● Clone – clone the selected suite. ● Rename – rename the selected suite. ● Remove – remove the selected suite. ● Import – import a new suite. The Open dialog box opens. ● Export – export the selected suite. The Save dialog box opens. <p>All these commands are available from each metric suite shortcut menu.</p> |
| Metrics tree | Use this tree to select the metrics you want to include in your metrics suite. All metrics are displayed in the metrics tree. |
| Properties list | Metrics properties are displayed individually for each property. |
| Select All | Selects all metrics. |
| Clear All | Clears all metrics. |

| Element | Description |
|-------------------------|--|
| Reset to Default | Predefined MagicDraw metrics are reset to the default metrics suite. User-created metrics suites are reset to the selected predefined metrics suite. The Reset Metrics Suite Properties dialog box opens:  |
| Description | Displays the selected metric description. |
| OK | Saves all changes and closes the dialog box. |

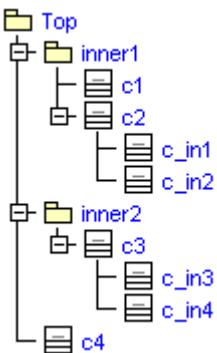
Metrics Properties

| Properties Group | Property | Description |
|----------------------|-----------|---|
| General | Calculate | <p>Defines what will be counted:</p> <ul style="list-style-type: none"> • Local – inside package (class). • Global - inside package (class) recursively. • Average – metrics will be counted from the lowest level of the elements tree. Each upper level metric will be counted as an average of the current object metric and all lower level metrics: $\text{Average_element_metric} = (\text{Element_metric_value} \text{ (if counted separately)} + \sum(\text{inner_elements_metrics_values})) / (1 \text{ (if element_metric_value was counted)} + \text{count_of_inner_elements_that_have_metrics_counted})$ <p>Average metric value should be rounded down to the lower value (for example, 1.5 = 1, 1.6 = 2)</p> • Min – lowest level metrics will be counted. Each upper level metric will be set to the minimum of the current object metric and all lower level metrics (except the metrics that are equal 0). $\text{Min_element_metric} = \min(\text{Element_metric_value}, \min(\text{inner_elements_metrics_values}))$ <p>Here metric_value > 0</p> • Max – lowest level metrics will be counted. Each upper level metric will be set to the maximum of the current object metric and all lower level metrics. $\text{Max_element_metric} = \max(\text{Element_metric_value}, \max(\text{inner_elements_metrics_values}))$ <p>Here metric_value > 0</p> |
| Lowest limit | Package | Recommended lowest metric value for the package. Editable. |
| | Class | Recommended lowest metric value for class and interface. Editable. |
| | Diagram | Recommended lowest metric value for the diagram. Editable. |
| Highest limit | Package | Recommended highest metric value for package. Editable. Note: if the highest limit is equal to 0, the metric is never marked as too high (in red font color). |
| | Class | Recommended highest metric value for class and interface. Editable. Note: if the highest limit is equal to 0, the metric is never marked as too high (in red font color). |
| | Diagram | Recommended highest metric value for diagram. Editable. Note: if the highest limit is equal to 0, the metric is never marked as too high (in red font color). |
| Include | | This properties group specifies whether the information is included when counting metrics. |
| Weight | | This properties group specifies whether the information is included when counting metrics. |

The following is an example of a metrics calculation used for calculating the number of classes (NC) in this tree:

8 MODEL ANALYSIS

Dependency Matrix



Calculated metric values with a different aggregation:

| Element | Local | Global | Average | Min | Max |
|---------|-------|--------|---------|-----|-----|
| Top | 1 | 8 | 1 | 1 | 2 |
| Inner1 | 2 | 4 | 1 | 2 | 2 |
| C1 | 0 | 0 | 0 | 0 | 0 |
| C2 | 2 | 2 | 1 | 2 | 2 |
| c_in1 | 0 | 0 | 0 | 0 | 0 |
| c_in2 | 0 | 0 | 0 | 0 | 0 |
| Inner2 | 1 | 3 | 1 | 1 | 2 |
| C3 | 2 | 2 | 1 | 2 | 2 |
| c_in3 | 0 | 0 | 0 | 0 | 0 |
| c_in4 | 0 | 0 | 0 | 0 | 0 |
| C4 | 0 | 0 | 0 | 0 | 0 |

Dependency Matrix

[View Online Demo](#)

Dependency Matrix

NOTE

This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The Dependency Matrix is a method of visualizing and representing dependency criteria. Diagrams, UML, and extended UML elements serve as row and columns entries. The cells in the matrix show where these elements are associated - related.

Dependency matrixes include different dependency criteria: UML relations, extended UML relations, semantic dependencies (dependency through property), and relationships through tags.

Relationship through tags is a relationship where a cell represents a relation that is implemented as a tag added to the element with a reference to another element. A relation through tags allows relate UML element of any type. Tags are one of the methods for relating elements that cannot be represented on the same diagram.

The Dependency Matrix fulfills the feature, which helps visualize the many-to-many traceability from elements not from the same diagrams. The Dependency Matrix also provides the visualization of many-to-many for large interconnected system elements. The most usefulness is complete the lack of functionality to support different domains - DoDAF.

Dependency Matrix functionality is useful for:

- Quickly visualizing dependency criteria.
- Compactly visualizing relations of a big system. Such system relations cannot be represented by a diagram on a single sheet of paper, as the diagram is very big.
- Creating a matrix template for domains and supporting domain specific element relation visualizations.
- Studying relations from a particular scope and type of element by filtering the unimportant.
- Showing relations that cannot be represented in diagrams: representation (class by lifeline); behavior representation in other diagrams, operation representation by Call Behavior Action, Use Case relations with describing activities through property Owned Behavior, etc. The Semantic dependency matrix is needed for deeper model analysis. The Matrix allows the representation of any kind of relations through the element property.
- Another method of showing custom relations - through tags.

Creating the Dependency Matrix

The matrix element in the model is similar to the diagram element. After creating a new matrix, it appears in the Browser as a model element and the matrix pane can be opened by double-clicking on the matrix name. The same actions, which can be performed with diagrams, are valid for matrixes.

To open the Matrix Dependency View

- From the **Diagrams** main menu, select **Custom Diagrams > Dependency Matrixes**. Click **Add** to create a new dependency matrix and then click **Open**.
- From the **Analyze** main menu, select **Dependency Matrix** and then **Create Blank Matrix**. Type a name and select the package where you want to save it in the project and click **OK**.
- From the package shortcut menu in the Browser, select **New Diagram** and then **Dependency Matrix**.
- In the custom diagrams toolbar, click the **Dependency Matrix** button. Type a name and select the package where you want to save it in the project and click **OK**.

To change the dependency matrix properties

1. In the **Dependency Matrix Specification** dialog box (see “The Dependency Matrix Specification dialog box, tags group” on page 435), open the **Tags** group.
2. In the <<matrix template>> subgroup, select bolded tag, marked with icon  from the list and edit its value in the **Value** field.

Dependency Matrix View

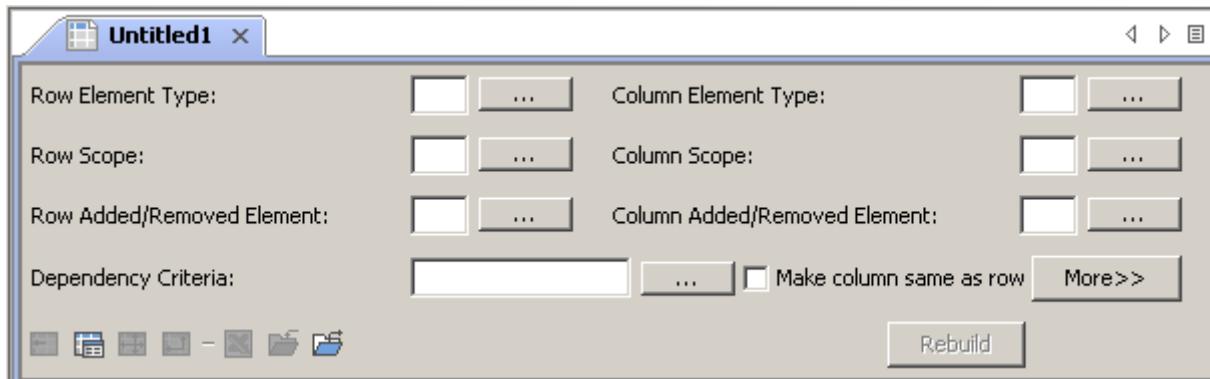


Figure 281 -- Dependency Matrix View

| | |
|-------------------------------------|---|
| Row Element Type | Click the “...” button to select the element or multiple element types to show in rows of the dependency matrix. |
| Column Element Type | Click the “...” button to select the element or multiple element types to show in columns of the dependency matrix. |
| Row Scope | Click the “...” button to define the scope of the model (packages/profiles), from which elements should be displayed in rows of the dependency matrix. |
| Column Scope | Click the “...” button to define the scope of the model (packages/profiles), from which elements should be displayed in columns of the dependency matrix. |
| Row Added/Removed Element | +<owner>::<element> value shows the row that is added to the matrix. -<owner>::<element> value shows the row that is removed from the matrix. Click "... " button to add or remove row from the dependency matrix table. The Add/Remove Elements dialog box opens. For more information about this dialog box, see “The Add / Remove Elements dialog box” on page 434. |
| Column Added/Removed Element | +<owner>::<element> value shows the column that is added to the matrix. -<owner>::<element> value shows the column that is removed from the matrix. Click "... " button to add or remove column from the dependency matrix table. The Add/Remove Elements dialog box opens. For more information about this dialog box, see “The Add / Remove Elements dialog box” on page 434. |
| Dependency Criteria | Click the “...” button to define what relations between row and column elements to display in the matrix cells. <ul style="list-style-type: none"> ● UML Relations. The matrix cells will display the existence of relationships between row elements and column elements. ● Tagged Value. The matrix cells will display relations through tags if matrix row elements have tagged values pointing to matrix column elements or vice versa. ● Properties. The matrix rows will display relations through properties if matrix row elements have properties pointing to column elements or vice versa. |

| | |
|---|--|
| Make column same as row | If selected, sets the column filter setting to the same as row. |
| More>> | Click More>> button to expand the toolbar and find advanced matrix filters areas. |
| Row Property/ Column Property | Select item from the combo box to select row/column elements, filtered by some property. Only entries that meet selected property values will be displayed in the rows/columns. Possible choices: <ul style="list-style-type: none"> • Applied stereotype; • Visibility; • Tagged Value; • To Do. |
| Row Property Value Column Property Value | The “...” button is enabled only when some Property is selected. The corresponding dialog box with available property values opens. Multiple property value selections are available. |
| Rebuild | Columns and rows rebuild action, which rebuilds element lists according to the filters configuration. |
| Add Remove Elements | Add/remove elements and diagrams from/to Matrix rows or columns independently from the filter settings. Click the  button to open the Add Remove Elements dialog box. For more information about this dialog box, see “The Add / Remove Elements dialog box” on page 434. |
| Matrix Properties | Click the  button to open the Dependency Matrix Specification dialog box. For more information about this dialog box, see “The Dependency Matrix Specification dialog box, tags group” on page 435 |
| Quick Diagram Layout | Click the  button to automatically layout the matrix cells to default width and height. |
| Change Axes | Click the  button. Columns and rows will be exchanged with each other. |
| Save as *.csv | Click the  button. The Save dialog box opens. Select location for a file and enter a file name (default name will be set the same as Matrix Name). The file will be saved in Comma Separated Values (*.csv) format. The file can be opened with MS Excel, Open Office, MC Excel imported into databases. |
| Safe Configuration As Template | Click the  button to save the filters configuration as a matrix template. The saved template can be imported into the Dependency Matrix Template dialog box and used in other projects. |

8 MODEL ANALYSIS

Dependency Matrix

Load Matrix Template

Click the  button to open the **Load Matrix Template** dialog box. Select a template from the list and click **OK**.

Dependency Matrix pane

After the dependency criteria (filter options) are specified, click the **Build** button. The Dependency Matrix pane will be opened.

Dependencies between elements are displayed in cells. Rows and columns display elements, which were specified in the Matrix View fields.

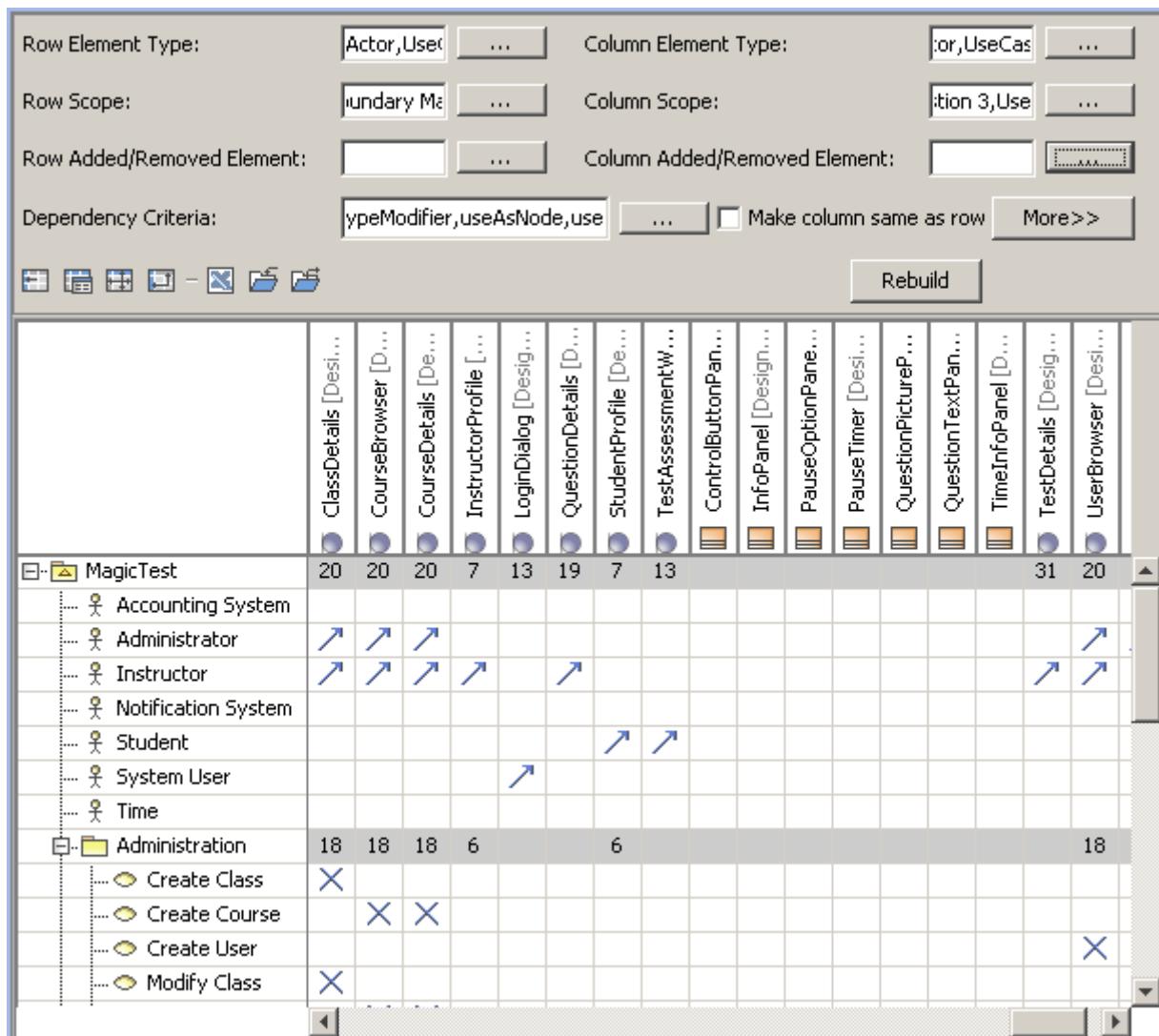


Figure 282 -- Dependency Matrix pane

An Icon with dependency direction representation is displayed in a cell if the single dependency is presented in this cell. Arrows can be:

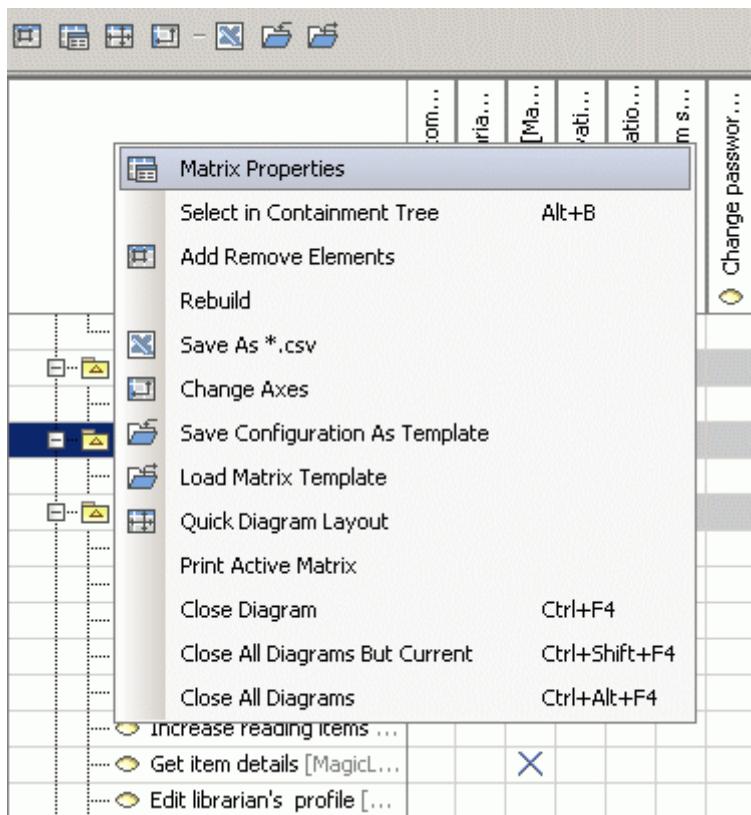
- One directional arrow - direction shows that element is dependent on the element, to which direction it points;
- Line - shows that elements depend on each other;
- "X" icon - appears when multiple dependencies are presented in the cell.

8 MODEL ANALYSIS

Dependency Matrix

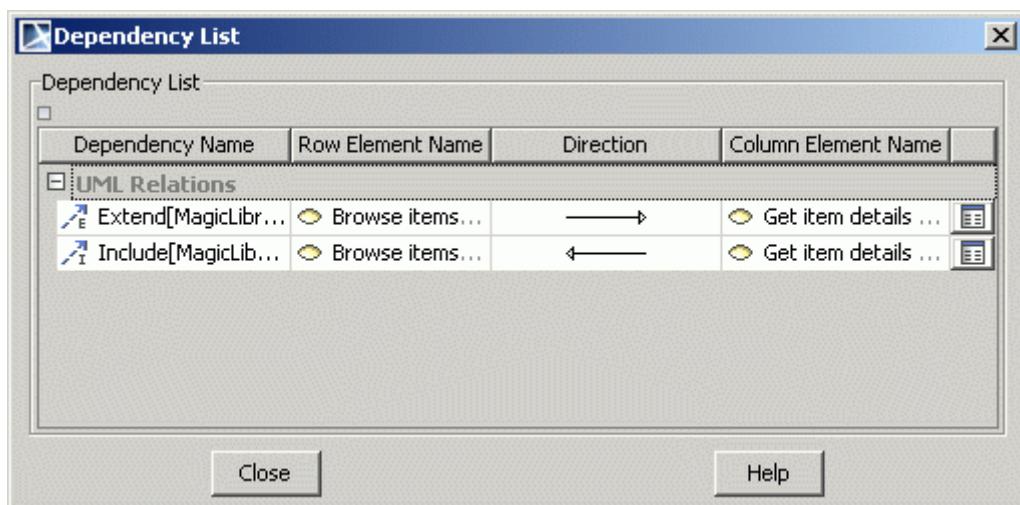
The number of dependencies between package elements is displayed in a cell where packages are intersecting with any element.

Right-click on the empty space in the Dependency Matrix pane to open the shortcut menu:



To open the relation specification dialog box of the dependent element

1. Double click a cell that is not empty or select it and from the shortcut menu, select **Dependency List**. The **Dependency List** dialog box opens.



2. Click the button near the selected dependency to open the corresponding dependency Specification dialog box.

Working with a Dependency Matrix Template

Matrix properties and filter configurations are stored in MagicDraw. The matrix configuration is called the matrix template. The matrix template is used to save a matrix independently of a project and to share them with other users. The user can create a matrix based on a predefined template.

The matrix template can be imported and exported as a file. Filters and Matrix properties will be saved as a matrix template.

To create a Dependency Matrix from a Template

1. From the **Analyze** main menu, select **Dependency Matrix** and then **Matrix Templates**. The **Dependency Matrix Templates** dialog box opens.
2. Select the template from the list and click **Create Matrix**, or click **OK**. After closing the dialog box, click **Rebuild** in the Matrix View.

To load a Dependency Matrix Template

1. In the Matrix View, click the **Load Matrix Template** button  . The **Load Matrix Template** dialog box opens.
2. Select a template from the list and click **OK**. The dialog box is closed and template values appear in the rows/columns filter fields in the Matrix View.
3. Click **Rebuild** in order to apply the template filter for the dependency matrix.

To save a configuration as a Dependency Matrix Template

1. Specify the desired filter values for rows and columns.
2. In the Matrix View, click the **Save Configuration As Template** button  . The **Save** dialog box opens.
3. Type a name for the template, select a location and click **Save**.

To export a Dependency Matrix Template

1. In the **Dependency Matrix Templates** dialog box, click the **Export Selected Matrix template** button  . The **Save** dialog box opens.
2. Type a name for the template, select a location and click **Save**.

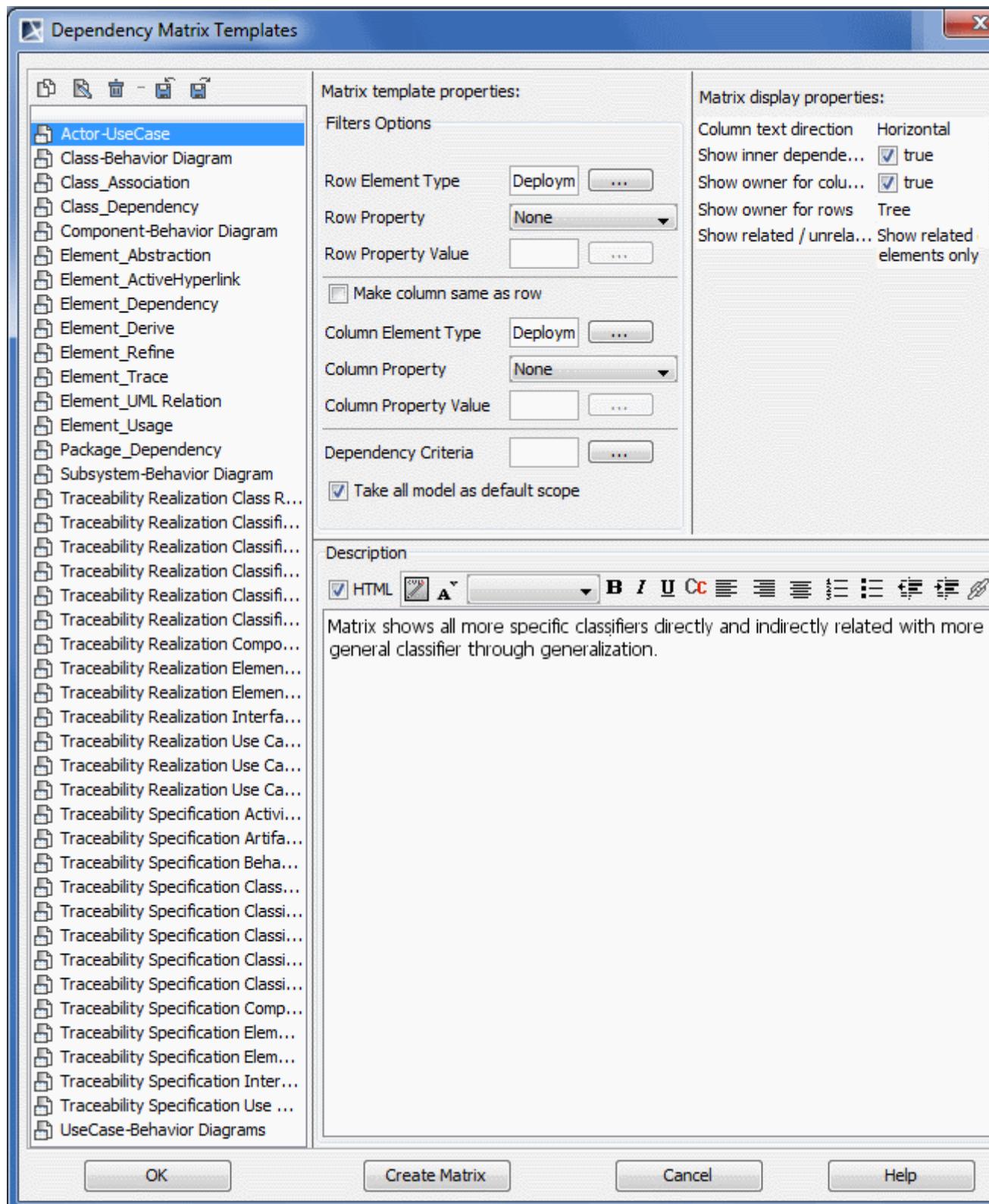
To import a Dependency Matrix Template

1. In the **Dependency Matrix Templates** dialog box, click the **Import New Matrix template** button  . The **Open** dialog box opens.
2. Select a matrix template and click **Open**. The template with defined properties will be added to the templates list.

Dialog boxes in Dependency Matrix functionality

Dependency Matrix Templates dialog box

From the **Analyze** main menu, select **Dependency Matrix** and then **Matrix Templates**. The **Dependency Matrix Templates** dialog box opens.



8 MODEL ANALYSIS

Dependency Matrix

Figure 283 -- The Package Dependencies dialog box

| Box name | Function |
|--|--|
| Clone Selected Matrix template | Select the template and click the  button. A copy of the existing matrix template will be created with the same properties and filter options. |
| Rename Selected Matrix template | Select the template and click the  button. The template name will be set to editable mode for renaming. |
| Remove Selected Matrix template | Select the template and click the  button. The template will be removed from the templates list. |
| Import New Matrix template | Click the  button to import a template. The Open dialog box opens. |
| Export Selected Matrix template | Click the  button to export a template. The Save dialog box opens. |
| Filters Options | For more information, see “Dependency Matrix View” on page 427. |
| Take all model as default scope | If selected, upon matrix creation, the whole model will be taken as the scope (later scope can be changed in Dependency Matrix View). If cleared, matrix will be empty until specifying scope. |
| Matrix Properties | |
| Columns text direction | Changes text direction in column headers. Possible choices: <ul style="list-style-type: none">● Horizontal● Vertical |
| Show inner dependencies | If selected, the number of dependencies among container elements will be shown in the cell where the container is intersecting with other elements. |
| Show owner for columns | If selected, displays the element owners in the column headers. |
| Show owner for rows | Displays the element owner in the row headers. Possible choices: <ul style="list-style-type: none">● Full qualified name● Hide● Tree |

| Box name | Function |
|---------------------------------|---|
| Show related/unrelated elements | Allows showing elements with or without their dependency criteria. Possible choices: <ul style="list-style-type: none">• Show all;• Show related elements only;• Show unrelated elements only. |
| Description | Displays the matrix template description. Select the HTML check box to edit text using the advanced style. |
| Change Exports | The Export Module dialog box opens. You can change the package set, selected for export. |
| OK | Closes the dialog box and adds changes to the Matrix View fields according to the selected template. |
| Create Matrix | Closes the dialog box and creates the dependency matrix, applying filters and options from the selected template. |
| Cancel | Exits the dialog box without saving changes. |
| Help | Displays MagicDraw Help. |

The Add / Remove Elements dialog box

In the Dependency Matrix View, click the  button to open the **Add Remove Elements** dialog box.

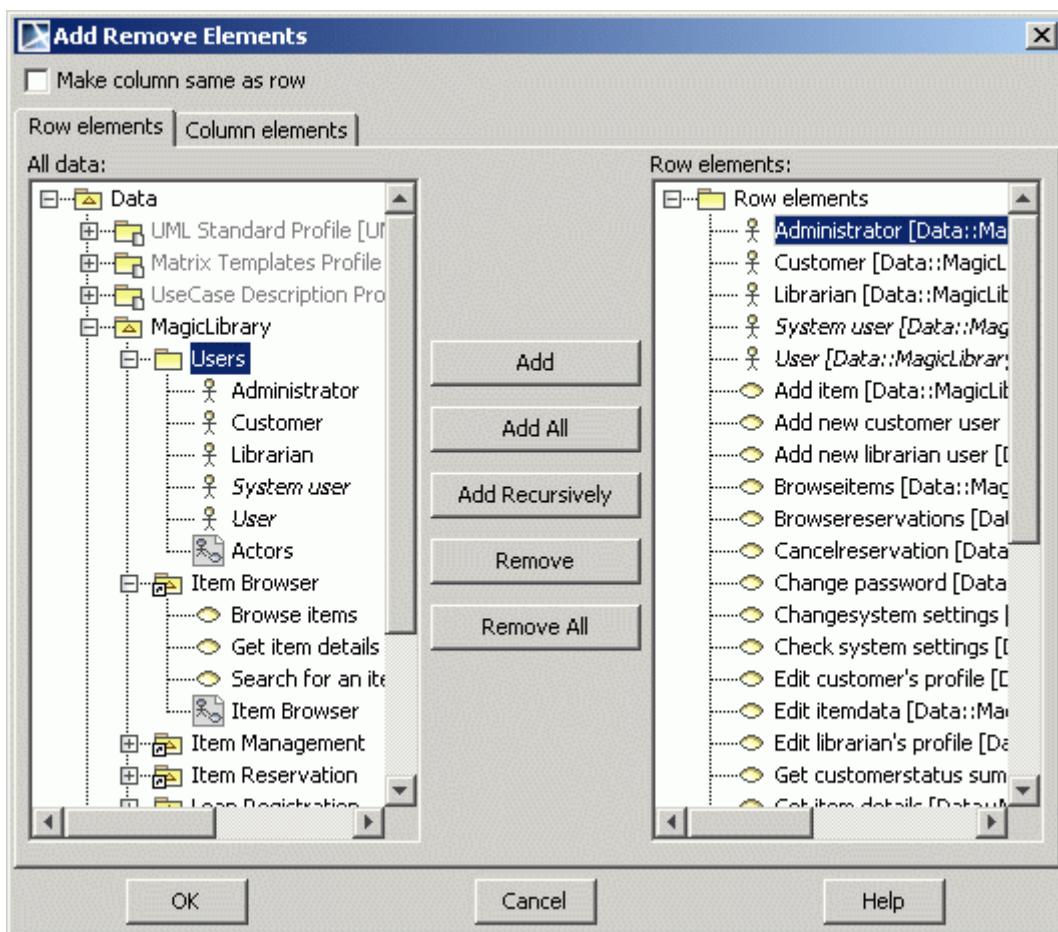


Figure 284 -- The Add / Remove Elements dialog box

| Box name | Function |
|--------------------------------|---|
| Make column same as row | If selected, applies the same element combination that was set for row, to column. |
| Row elements | Displays a list of all model elements and chosen elements to display in the matrix rows. |
| Column elements | Displays a list of all model elements and chosen elements to display in the matrix columns. |
| Add | Adds the selected element from the All data list to Row (Column) elements list without adding its inner elements. |
| Add All | Adds all model elements from the All data list to Row (Column) elements list. |
| Add Recursively | Adds the selected element from the All data list to Row (Column) elements list together with its inner elements. |
| Remove | Removes the selected element from the Row (Column) elements list. |
| Remove All | Removes all elements from the Row (Column) elements list. |
| OK | Saves changes and closes dialog box. |
| Cancel | Closes dialog box without saving changes. |
| Help | Displays the MagicDraw help. |

The Dependency Matrix Specification dialog box, tags group

- In the Browser, select the matrix element and from the shortcut menu, choose **Specification**.
- From the Dependency Matrix pane shortcut menu, click **Matrix Properties**.

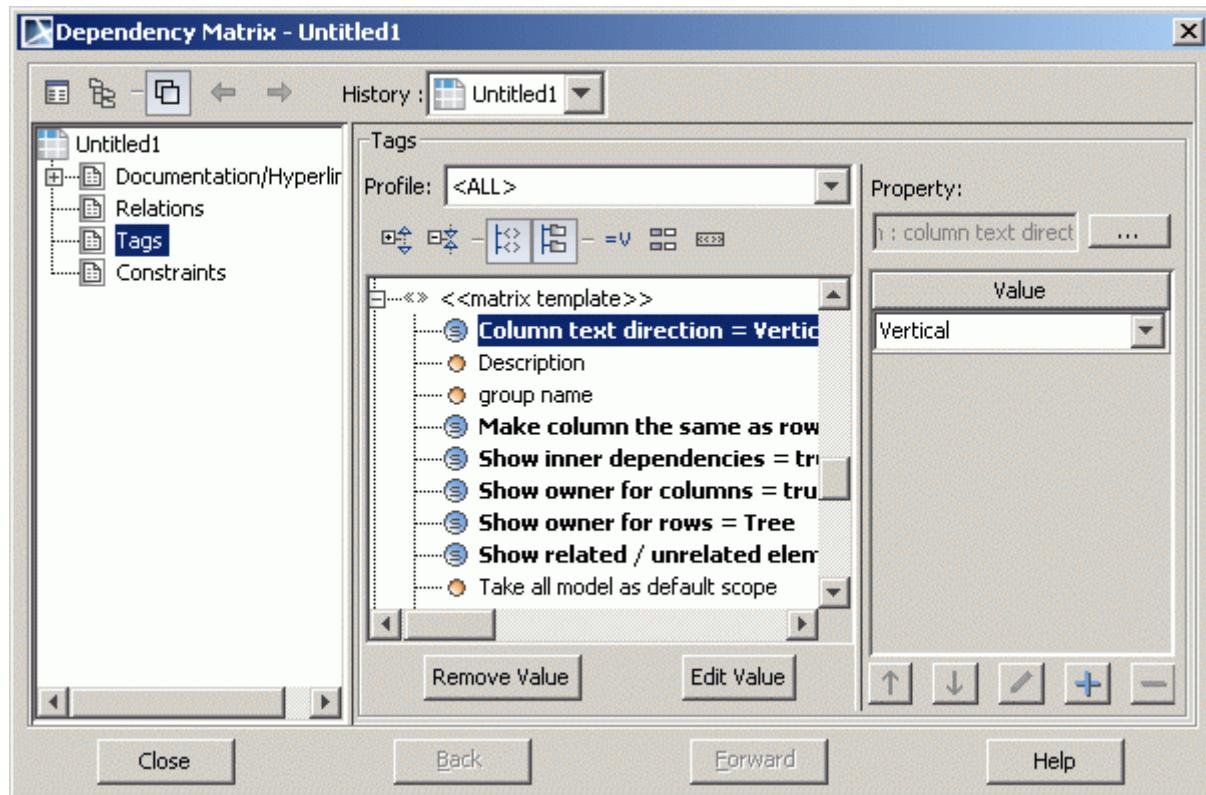


Figure 285 -- The Dependency Matrix specification dialog box, Tags group

This dialog box can be used to change matrix property values. In the <<matrix template>> tags group, select bolded tags and in the right open window, change the default value to the desired one.

Analyzing Package Dependencies

An element depends on the module when at least one of its metaproPERTIES reference elements from at least one share of that module. In such a case the element has a dependency from the module.

NOTE You may define options for the dependency checking in the in **Project Options** dialog box -> **General Project Options** -> **Dependency Checker** options group.

To analyze package dependencies of the whole project or between the selected package/model and shared packages

1. Choose one of the following:

| | |
|--|--|
| To analyse package dependencies of the whole project, | On the Analyze menu, click Dependency Checker . |
| To analyze dependencies between the selected package/model and shared packages | Right click the package or model in Browser or in diagrams, point to Tools and then click Dependency Checker . |

2. The **Dependency Checker** dialog box appears. Click **OK**.

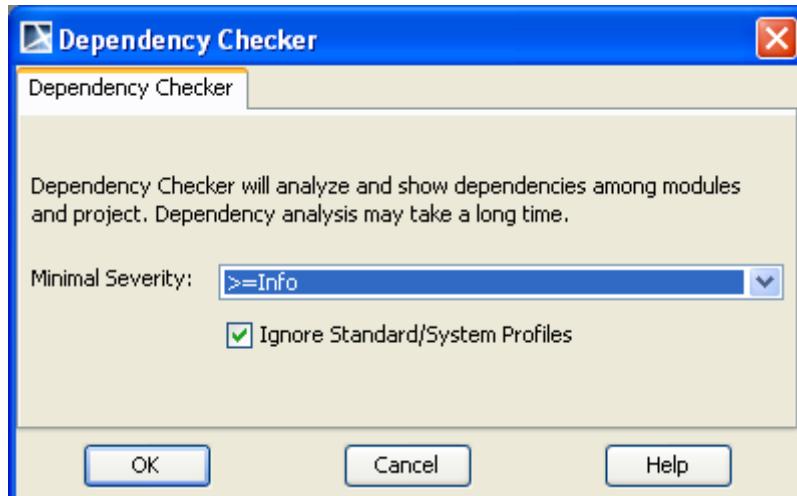


Figure 12 -- The Options window for the dependency checker

3. Dependencies between the selected package/model and shared packages will be analyzed and shown in the opened **Package Dependency** panel. The detailed description of the **Package Dependency** panel, see in Section "Package Dependencies panel" on page 438.

To analyze package dependencies while exporting modules or sharing packages

1. Choose one of the following:

| | |
|---|--|
| To analyze package dependencies while exporting modules | From the package/model shortcut menu, choose Modules > Export Module . The Export Module dialog box appears. Select the modules you want to export (more information about exporting modules you may find in section “Exporting the module of a project” on page 142). |
| To analyze package dependencies while sharing packages | From the package/model shortcut menu, choose Modules > Share Packages . The Shared Packages dialog box appears. Select the packages you want to share (more information about sharing packages you may find in section “Sharing the module of a project” on page 143). |

2. The question dialog box appears asking for your confirmation to start dependency checking between the exported package/model and the rest of the project (including shared packages that belong to the project and used modules).

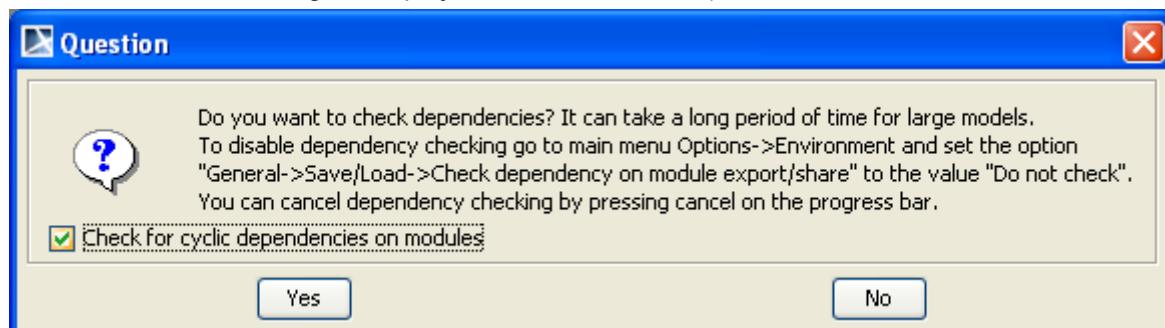


Figure 286 -- A confirmation window for starting up the dependency checker

3. If you want to discover cyclic dependencies*, select the **Check for cyclic dependencies on modules** check box.. Press **Yes** to start dependency analysis. If the **Check for cyclic dependencies on modules** check box is selected, only dependencies which have **Error** and **Warning** severity levels are displayed..

NOTE:

If there is a chain of dependencies such that:

A->M(1), M(1)->M(2), M(2)->M(3), ..., M(X)->M(A), where:

- A is an element from the module M(A)
- M(1..X) are other modules
- A->M(x) is element A dependency on module M(x)
- M(y)->M(x) is a dependency of at least one element in module M(y) on module M(x),

then this chain is called a cyclic dependency and every atomic dependency in this chain is considered as part of cyclic dependency.

4. Dependencies between the selected package/model and shared packages will be analyzed and shown in the opened **Package Dependency** panel. The detailed description of the **Package Dependency** panel, see in Section “Package Dependencies panel” on page 438.

Package Dependencies panel

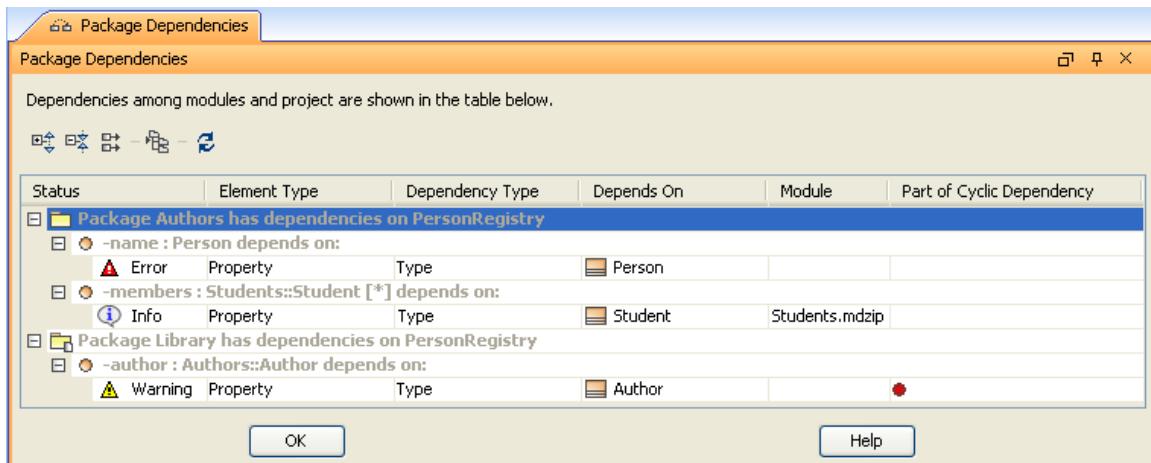


Figure 287 -- The Package Dependencies panel, opened independently

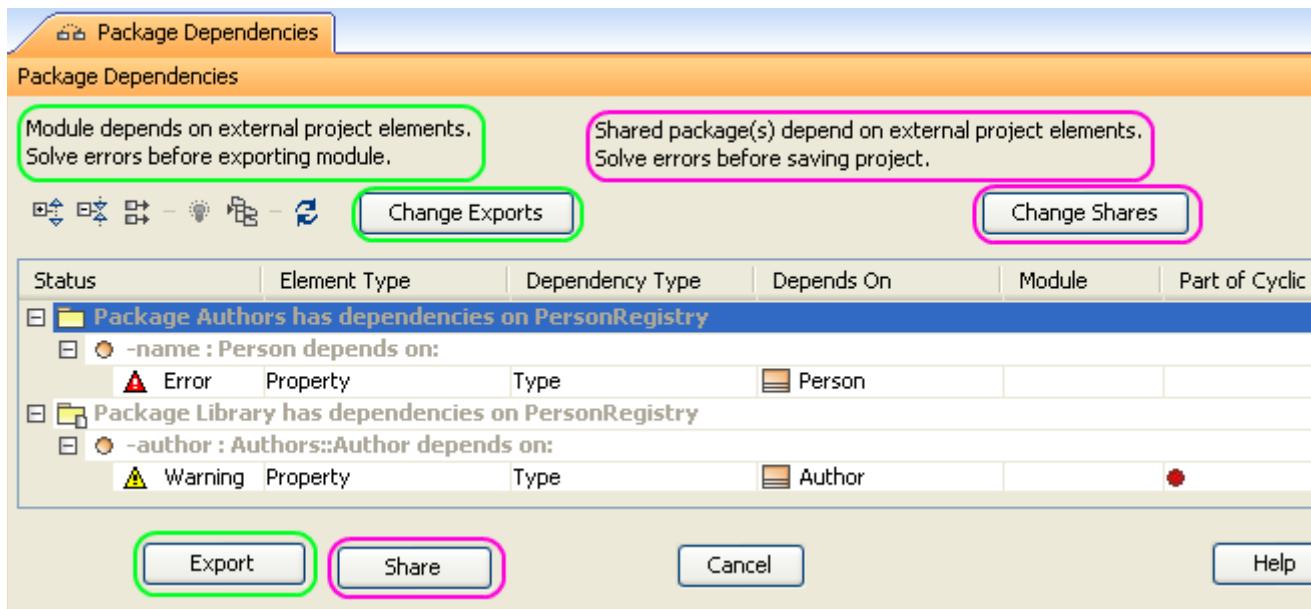


Figure 288 -- The Package Dependencies panel, opened when exporting (GUI elements, highlighted in green) or sharing a package (GUI elements, highlighted in pink)

The **Package Dependencies** panel has a table which shows the list of dependencies and buttons for managing data displayed in this table.

| Button | Description |
|-------------------------------|--|
| Expand All Tree Branches | Expands all nodes in the package dependencies tree. |
| Collapse All Tree Branches | Collapses all nodes in the package dependencies tree. |
| Show/hide the Full Path Names | Displays the element full path next to the element name. |

| | |
|-----------------------------------|---|
| Solve | The button is enabled, when a dependency whose status is Error is selected in the table and a solution for the problem can be found. Clicking the button opens the dialog box for choosing the solution for a specific dependency problem. This button is only visible when exporting or sharing a package. |
| Select in Containment Tree | Shows the selected element in the Browser. The button is enabled, when a dependency is selected in the table. |
| Refresh | Performs dependency analysis and refreshes the dependency table with the new analysis results. |
| Change Shares | Opens the Shared Packages dialog window for reselecting the packages to be shared. This button is visible only when sharing a module. |
| Change Exports | Opens the Export Module dialog window for reselecting the packages to be exported. This button is visible only when exporting a module. |
| OK | Closes the Package Dependencies panel. This button is available when the dependency checker is opened independently by selecting Analyze > Dependency Checker from the main menu or Tools > Dependency Checker from the package or model shortcut menu. |
| Share | Closes the Package Dependencies panel and makes the package shared. The button is enabled, when the environment option Check dependency on module export/share is set to Allow dependencies (Options > Environment) from the main menu, General section). Environment options are detailed in section “Customizing Environment Options” on page 89. This button is visible only when sharing a module. |
| Export | Closes the Package Dependencies panel and opens the Save as.../Commit Settings dialog window for saving/committing the package as separate module. The button is enabled, when the environment option Check dependency on module export/share is set to Allow dependencies (Options > Environment) from the main menu, General section). Environment options are detailed in section “Customizing Environment Options” on page 89. This button is visible only when exporting a module. |
| Cancel | Cancels package sharing or exporting. This button is visible only when exporting or sharing a module. |
| Help | Displays MagicDraw Help. |

| Column | Description |
|--------|-------------|
|--------|-------------|

| | |
|-----------------------------------|--|
| Status | Shows severity of the element dependency problem. The status can be Error , Warning , or Info . Dependencies that have Error status: <ul style="list-style-type: none">• module dependencies on the project Dependencies that have Warning status: <ul style="list-style-type: none">• cyclic module dependencies on other modules Dependencies that have Info status: <ul style="list-style-type: none">• project element dependencies on elements from shared packages (shared packages can belong both to the project and come from an external project) |
| Element Type | Displays element type. |
| Dependency Type | Displays dependency type. |
| Depends On | Displays the model element, on which the package/model element is dependent. |
| Module | Displays the name of the module file that owns the model element, on which the package/model element depends. |
| Part of Cycling Dependency | If a symbol “  ” is depicted, that indicates that the element is a part of the cycling dependency. |

Analyzing Dependencies Among Elements

A package can be exported to an independent module only if it does not depend on external elements (except other modules). Cyclical dependencies between several modules are not allowed.

There are three types of dependencies:

- Dependency by relationship
- Dependency by reference
- Diagram dependencies

Package dependencies by relationship

The module depends on external elements

If a module element has a relationship with an external element and this relationship is contained in the module package, an error message appears when exporting the module.

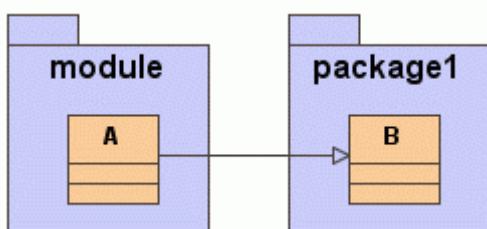


Figure 289 -- Example of a module dependency on an external element

8 MODEL ANALYSIS

Analyzing Package Dependencies

Such dependencies on external elements are displayed in the Browser tree:



Figure 290 -- Package has a dependency on an external element

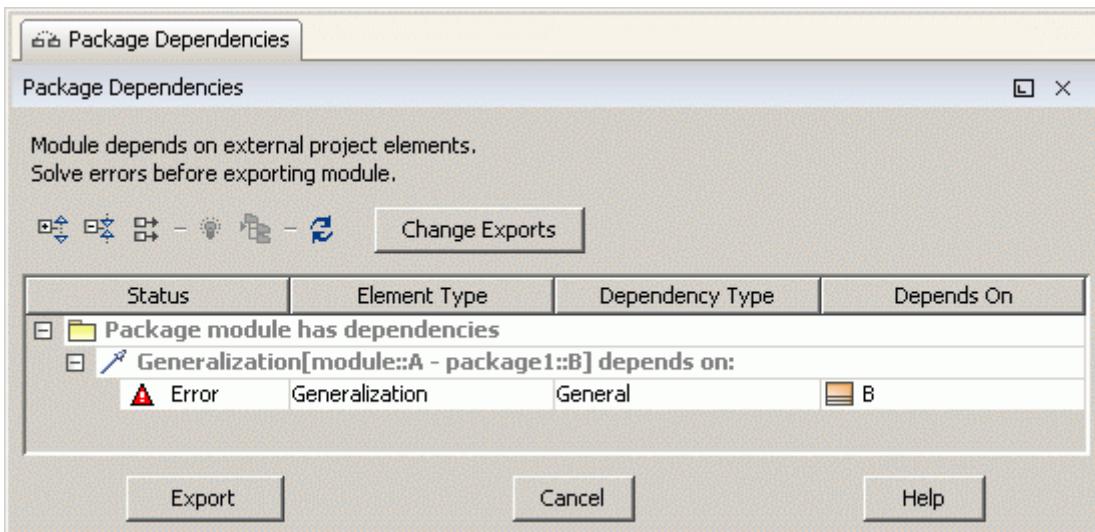


Figure 291 -- Error in the Package Dependencies dialog box

In this situation, MagicDraw can suggest moving the relationship into the parent package of this external element. For example, **package1** is a parent of class **B**, so the relationship can be moved from the **module** into **package1**:



Figure 292 -- Resolved package dependency on an external element

Some movements can be achieved by clicking **Solve** in the **Package Dependencies** dialog box. For a detailed description of this dialog box, see “The module package can now be exported into an independent module.” on page 443.

You can also drag-and-drop the relationship from one package to another in the Browser tree.

The module depends on an external element, but can be exported (with warning)

Though the module element has a relationship with an external element, this relationship is contained in an external package:

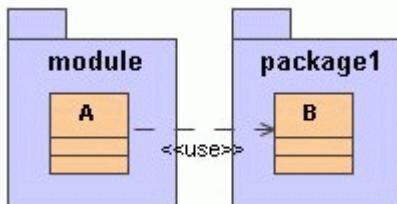


Figure 293 -- Example of a “legal” module dependency on an external element

In this case, the dependency on an external element is displayed in the Browser tree:

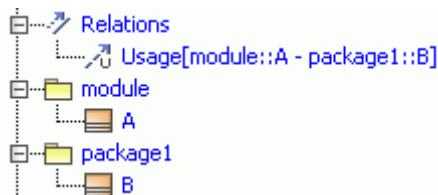


Figure 294 -- “Legal” module dependency on an external element in the data model browser

The package can be exported as a module because the relationship is contained in an external package.

The module does not depend on an external element

If the module element has a relationship with an external element is irrelevant in the context of UML (for instance, the external model uses the module, but not vice versa) and this relationship is contained in an external model, the package can be exported into an independent module:

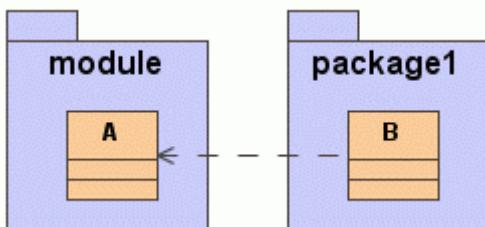


Figure 295 -- Example of a relationship when the module does not depend on an external element

Dependencies by reference

The module depends on external elements when the model elements from the module packages have references to external elements.

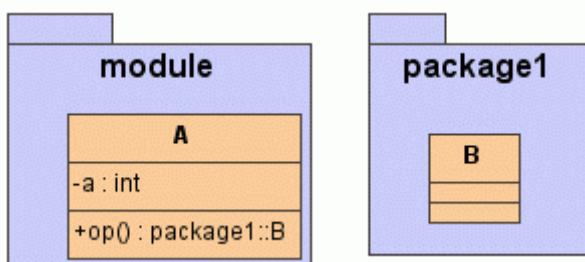


Figure 296 -- Example of a dependency by reference

In this case, the **module** package cannot be exported to an independent module.

Diagram dependencies

The diagram depends on all model elements displayed within it.

If the diagram is contained in a **module** package and depends on external elements, this package cannot be exported to a module.

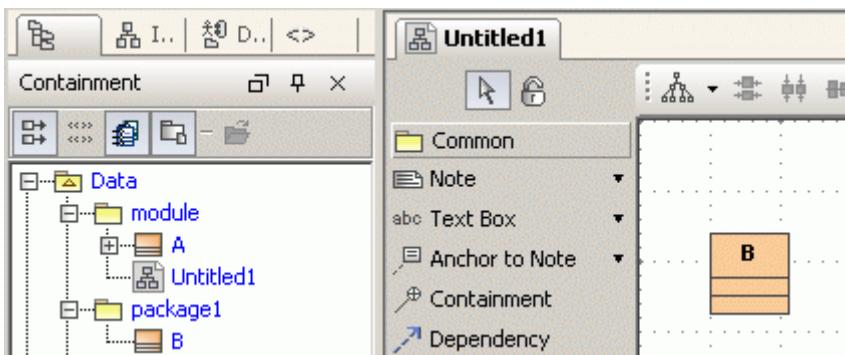


Figure 297 -- Example of the relationship when a diagram depends on an external element

For more information about the package dependencies on external elements, see “The module depends on external elements” on page 440.

In this case, if the diagram is not important to the module, it can be moved from the **module** package into any external package by dragging and dropping it within the Browser tree:

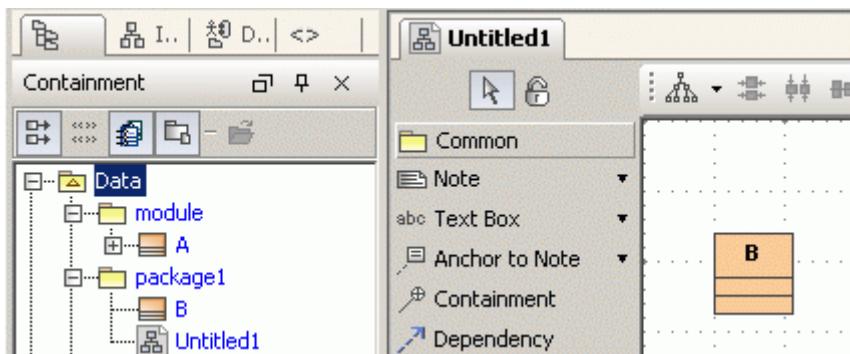


Figure 298 -- Diagram is moved from the module package to the package1 package.

The **module** package can now be exported into an independent module.

MagicDraw Teamwork Server is an ideal solution for group work on the same project. For more information about the Teamwork Server, see the [MagicDraw Teamwork System User’s Guide](#).

Unresolved dependencies

When a model part is exported to a separate module, if there are dependencies from the module back to the project, you are asked to resolve them (dependencies in the opposite direction - elements in project depending on elements in module - are OK).

The same situation occurs when you edit the module inside the project (when the module is mounted read-write on the project) and introduce dependencies from the module back to the project. In this case, you will be asked to resolve these dependencies on module save.

However, it might be inconvenient to resolve these dependencies at that moment (perhaps you have finished work for today and you will resolve dependencies tomorrow, and now you just want to save the project and leave; perhaps the particular dependency resolution is not a trivial task, which will take some time).

MagicDraw allows you to continue without resolving these dependencies. The elements, which were referenced, but are missing in the module will be shown as missing proxy elements (see “Missing elements for the proxies (orphaned proxies)” on page 151).

This is one more improvement - in previous versions MagicDraw was strict in checking dependencies and did not allow dangling references. Now more flexibility is allowed.

This behavior is controlled by the **Check dependency on module export/share** environment option (from the **Options** menu, select **Environment, General** section).

There are three choices:

- **Do not allow dependencies** setting restores previous, strict checking.
- **Allow dependencies** is the default setting, described above.

Do not check setting is an even more lax setting; it does not prompt the user to resolve dependencies at all. If you are not careful, this can lead to the proliferation of missing proxy elements, hence proceed with care.

Validation

[View Online Demo](#) Validation

NOTE This functionality is available in Architect and Enterprise editions only.

Introduction

MagicDraw has the functionality to check the created models. It consists of:

- A set of validation rules. Each validation rule captures some imperative conditions, which must be checked against the model. Validation rules are specified as invariant constraints in the model.
- One or more validation suites (modeled as packages). A validation suite is a simple concept of grouping the validation rules into meaningful groups, so that the collection of rules can be applied.

To run the validation, select some suites and validation scope - either the entire model or some part of it. When the validation is run, each rule from the suite is evaluated for each suitable element in the validation scope. Each element that violates the rule (constraint evaluates to false) is reported in the results table.

Since rules and suites are model elements, they can be manipulated using the standard MagicDraw modeling means - they can be copied, moved, and edited in the model; they can be refactored into modules, to facilitate reuse in other projects, placed in the Teamwork Server for exchange, etc. And of course this approach allows editing predefined rules and defining new, custom rules for models and profiles.

Constraint Types

Each validation rule, modeled as a constraint has a target classifier property. This property determines on what type of element this rule applies. Thus the usual level - metalevel separation appears. Constraints that are

defined on some particular classifiers are evaluated on the instances of these particular classifiers when validating. Inheritance is taken into account - instances of the subclasses of the class are also validated.

Thus there are 3 types of constraints that MagicDraw can evaluate:

- **Classifier level constraints.** Constraints that are placed on the classes, datatypes and other classifiers of the model are evaluated on all the instances of these classifiers - i.e. those InstanceSpecifications that have the particular classifier set as their type.
- **Constraints on metaclasses.** When a constraint is placed on a metaclass (one of the classes in the *UML Standard Profile::UML2 Metamodel*), this constraint is evaluated on all the model elements of that kind. E.g. if the constraint is placed on Actor metaclass, then this constraint applies to all the actor elements in the model. The following is an example of rule (specified in OCL2.0), which mandates that all actor names in the model must be capitalized:

context Actor inv capitalize:

```
let startswith:String = name.substring(1,1) in  
startswith.toUpperCase() = startswith
```

These constraints are useful for specifying generic rules, which must apply on all the model elements of particular kind.

- **Constraints on stereotypes.** When a constraint is placed on some stereotypes of the profile, that constraint applies to all the model elements that have these stereotypes applied to them. These constraints are useful when creating domain specific profiles. When adapting UML to some specific modeling domain, a profile is usually created with extensions for that domain - stereotypes, tags etc. The constraints on these stereotypes allow enforcing the rules of that domain.

It is advisable not to mix the constraints from different metalevels into one suite (constraints on classifiers versus constraints on stereotypes and metaclasses).

Predefined Validation Suites

There are several validation suites (collections of validation rules) predefined in the profiles that come with MagicDraw. Since validation rules and validation suites are concepts, stored in the model, the availability of list of validation suites for validating depends on what profiles the model includes.

UML Standard Profile brings two predefined suites with it. These two suites are present in all models:

- UML completeness constraints;
- UML correctness constraints.

Completeness suite has a collection of rules, which check if a model is complete, that there are no gaps, and the essential information fields in the elements have been filled in (e.g. checks that all the properties have type specified etc.).

Correctness suite has a collection of rules, which check common mistakes while modeling in UML2 (NOTE: this collection is not exhaustive).

Additionally, there are validation suites for each of these modeling domains - XML schemas, DDL, Java, C++ plus DoDAF and SysML, if any. These validation suites are defined in the corresponding profiles of these modeling domains, hence they are included automatically when you start modeling in that domain.

For example - if you create a new XML schema diagram, XML schema profile will be automatically included in your model and this profile brings in XML schema validation suite with it. So, from that moment, XML schema validation suite is available in the project.

Validating

To run the validation, you have to select a group of rules to be validated (validation suite) and indicate which part of the model to check (validation scope - either the entire model or some part of it).

To validate UML model for correctness

1. Open the **Validation** dialog box. Do one of the following:
 - From the **Analyze** menu, select **Validation** and then click **Validate**.
 - On the **Validation** toolbar, click **Validate** .

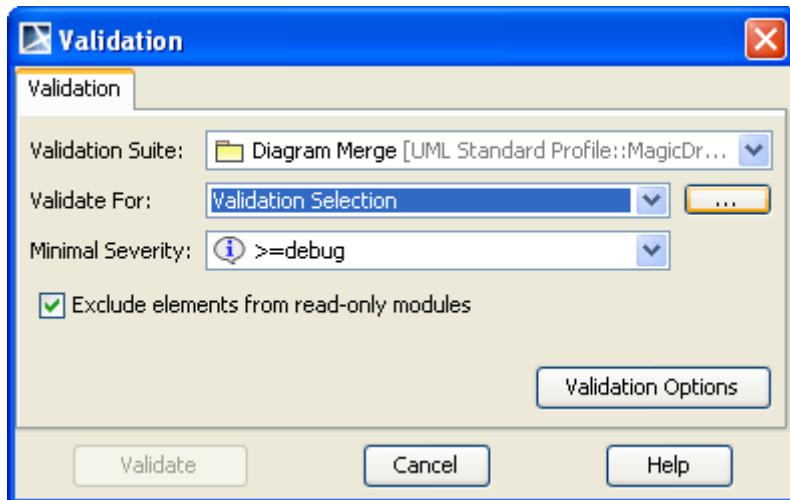
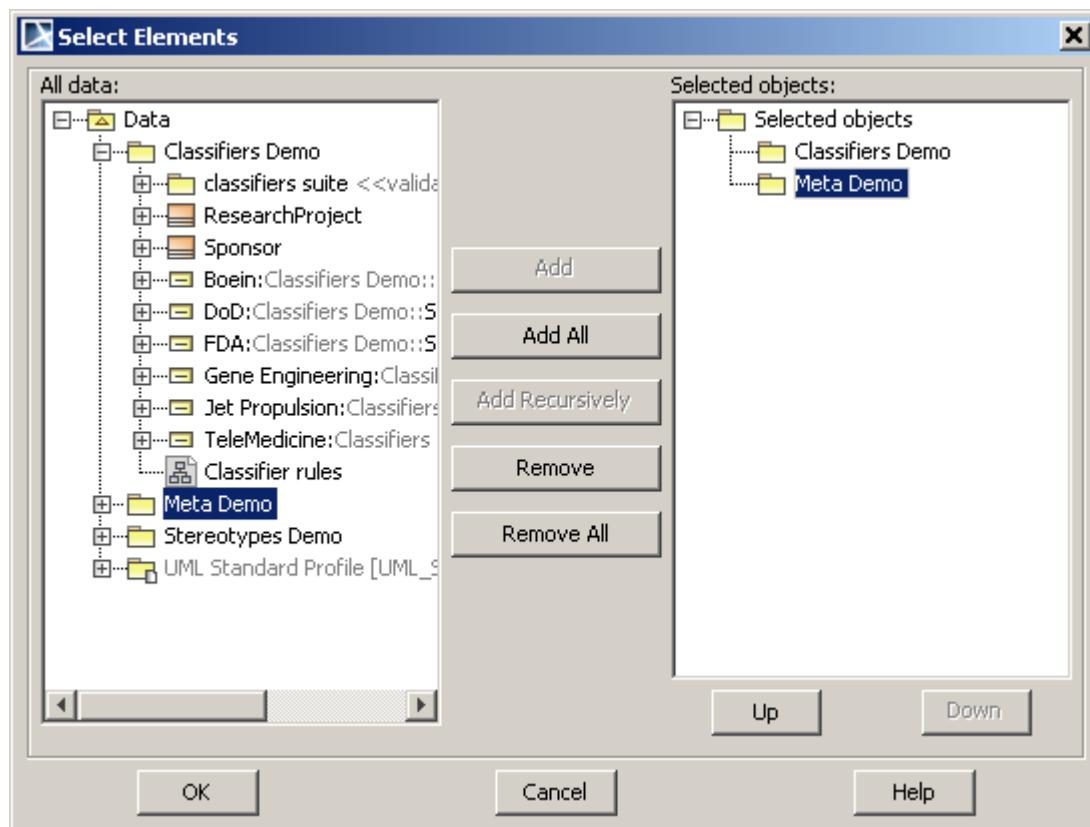


Figure 299 -- Validation dialog box

2. In the **Validation Suite** box, select the UML correctness constraints. All available validation suites are listed here.
The list of available validation suites depends on the opened project - the validation suites and validation rules are stored in the model as normal model elements. By default, a project has two suites - UML completeness constraints and UML correctness constraints - defined in the Standard profile.
If a project uses other profiles/modules - such as Java/XML schema/DDL profile, these profiles bring in their own predefined suites. You can also create your own validation rules and group them into a suite and this suite will be available in this box.
3. In the **Validate For** box, choose the validation target:
 - To run validation on the entire model, choose the **Whole Project**.
 - To run validation on the special packages and/or elements, choose the **Validation Selection**, then click the “...” button and select the scope for validation. The **Select Elements** dialog box appears.



4. **Select Elements dialog** In the **Select Elements** dialog box, select packages and/or elements for validation. Select the **Minimal Severity** level. Debug is the lowest possible severity level, all validation rules will be run.

NOTE

Validation is always recursive, hence if you select a package for validation, you do not need to select its inner elements (no need of the **Add Recursively** button). In the case depicted above, all model elements in the *Classifiers Demo* and *Meta Demo* packages will be validated. Adding *Data* package is equivalent to validating the entire model.

5. If you want to run validation on the read-only modules and the elements that exist in those modules, clear the **Exclude elements from read-only modules** check box (by default it is selected).
6. Click **Validate**. Validation results are displayed in the **Validation Results** window.

Validation Results Window

Validation results are displayed in the **Validation Results** window.

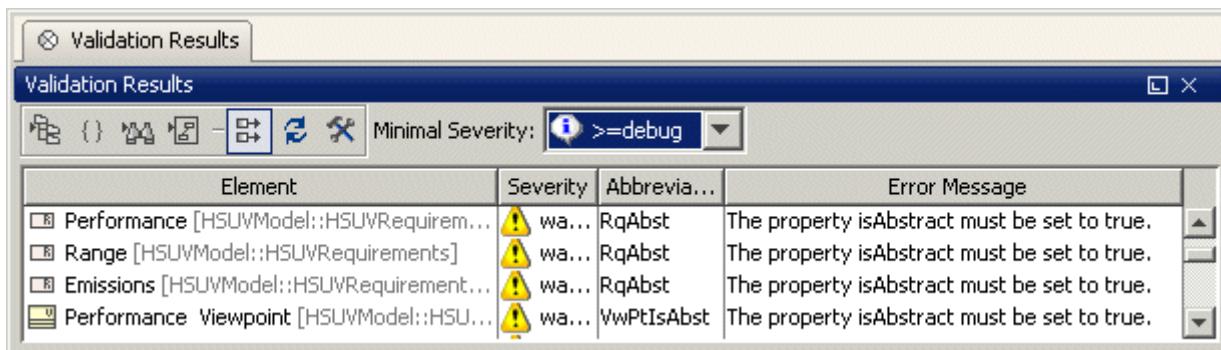


Figure 300 -- Validations Results window.

Window opens automatically after the validation has ended. The **Validation Results** window has the following columns:

| Column name | Description |
|---------------|---|
| Element | Elements, which violate the constraint rule are shown here. |
| Severity | Rule severity violations. |
| Abbreviation | Simple short strings showing the abbreviation of the violated constraint. Mostly used for sorting/grouping. |
| Error Message | Error texts of the violated constraints. |

If the validation rule is incorrect, **Validations Results** view will show the reason. This rule is then excluded from checking models.

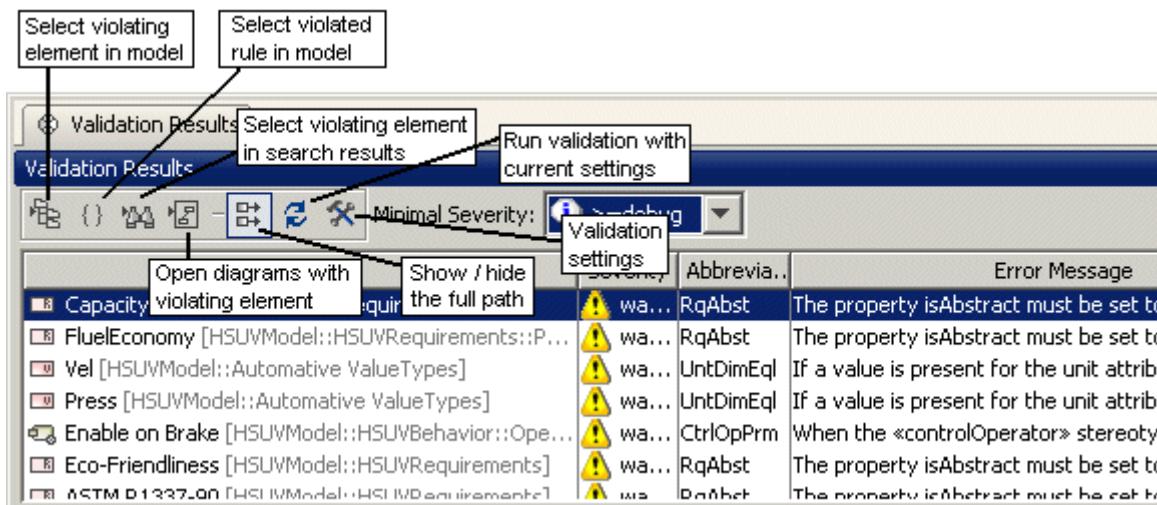


Figure 301 -- Buttons of Validation Results view.

Model Validation Example

We will validate SysML model - SysML.mdzip for correctness and completeness. The model is located in MagicDraw installation directory / samples. This model requires that SysML plugin would be installed. This can be done from **Help** menu **Resource / Plugin Manager**.

1. Open model. From the **Analyze** menu, select **Validation** command, and then **Validation**. The **Validation** dialog box opens.
2. Select **SysML Validation Constraints suite** and click **Validate**. The **Validations Results** view opens.
3. Narrow the validation scope. Click the **Run validation with a new**



settings button, the **Validation** dialog box opens. Change the scope to validation selection and select **SI Value Types** package, click **Validate**. The **Validations Results** view is refreshed with new options.

4. Double click the violating element **kg** in the results view or click **Select in**



Containment Tree button. The element is selected in the model.



5. Fix the problem and click the **Refresh** button to rerun the validation suite with the same options and refresh validation results. Refreshed results do not include element **kg**.
6. Select other violating element **N** in the results view and click **Open all**



Diagrams Containing the Selected Elements button to open all the diagrams, containing the selected model elements.

When the validation result view is opened and diagrams are shown on the screen, elements and links of the diagram, which have at least one result in the validation result view, are highlighted:

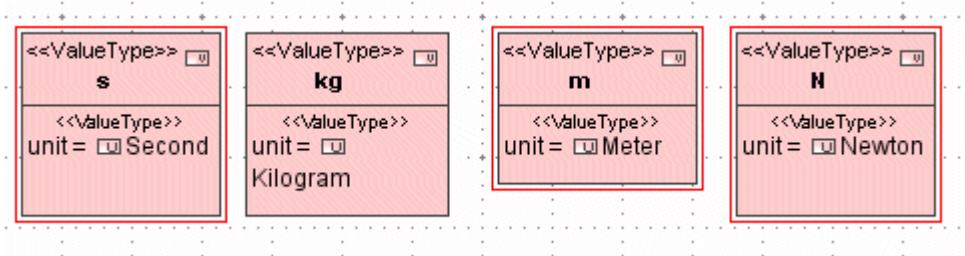


Figure 302 -- "N" element violating validation rule is highlighted.

In general, highlighting depends on the validation rule severity:

- Error - red
- Warning - yellow
- Debug - gray

- Info - black

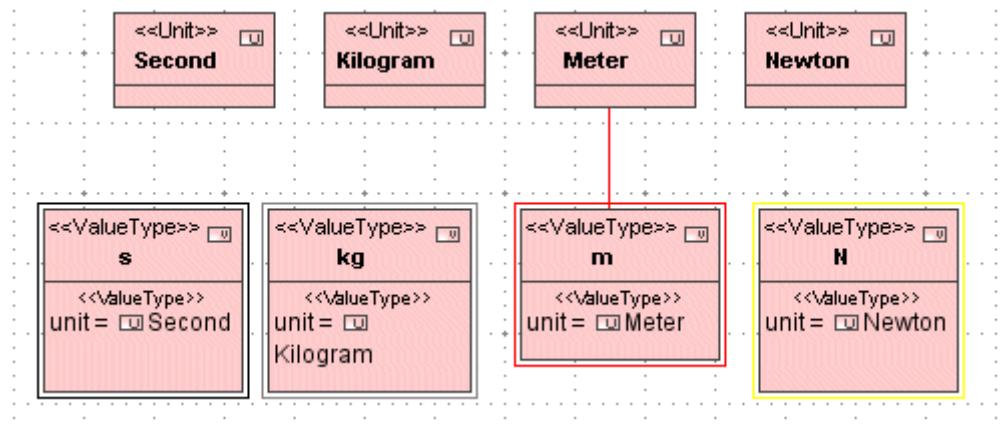


Figure 303 -- Elements and links violating constraints are highlighted.

If the element violates multiple rules, the color of the most severe rule is used.

Highlighting is shown until the rule is violated or the Validations Results view is not closed.

Other features available for the Validations Results view:

- By clicking the **Select Rule in Containment Tree** { } button, you may select rule in the model.
- You may move the selected results of the validation to the search result window by clicking the **Move to Search Results** M button.
- The **Open all Diagrams Containing the Selected Elements** D button opens all the diagrams, containing the selected model element.
- The Show / hide the Full Path names F button shows element location in the model path.

Validation Rules

The validation rules are modeled as UML2 constraints. This approach allows treating the validation rules as simple model elements. They can be handled using usual modeling mechanisms. They can be copied, moved around in the model, refactored into a separate module, stored in the teamwork server for easy information exchange, etc.

Since constraints can have different semantic meanings in UML2, a special type of the constraint – invariant constraint is used for modeling validation rules.

To distinguish these constraints from the other types of constraints, <<invariant>> stereotype should be applied to them.

Additionally, validation rules require other pieces of information – severity level (for sorting/filtering), abbreviation string (a short string, for easy recognition) and error message (complete description of error explanation). This information is displayed in the validation result view.

For storing this information, a special stereotype <<validationRule>>, derived from the <<invariant>>, is used. If you want to run this constraint as validation rule, use the former stereotype. If you have created the constraint just for documentation purposes and do not intend to run it, the latter constraint is sufficient.

Validation rules can be placed anywhere in the model (where UML2 constraint can be placed), however, usually they are stored in the classifier, which is constrained – classes, datatypes, etc. (for classifier level constraints), stereotypes (for meta-classifier level constraints). This convention breaks down for the constraints, placed on metaclasses (since these classes are stored in read only profile). In this case, place constraints wherever you like (e.g. group them into a package).

To create a validation rule on a metaclass

1. Select a package where you want to place the rule.
2. Right-click this package, select *New Element*, and then *Constraint*.

To create a validation rule on a classifier or stereotype

1. Open their specification, select *Constraints* section, and click **Create**.

Example:

Let's say we have 2 stereotypes - <<product>> and <<part>>. We want to place a validation rule, that products must have at least one part in them.

1. Open the product stereotype specification, select *Constraints* section, and click **Create**.

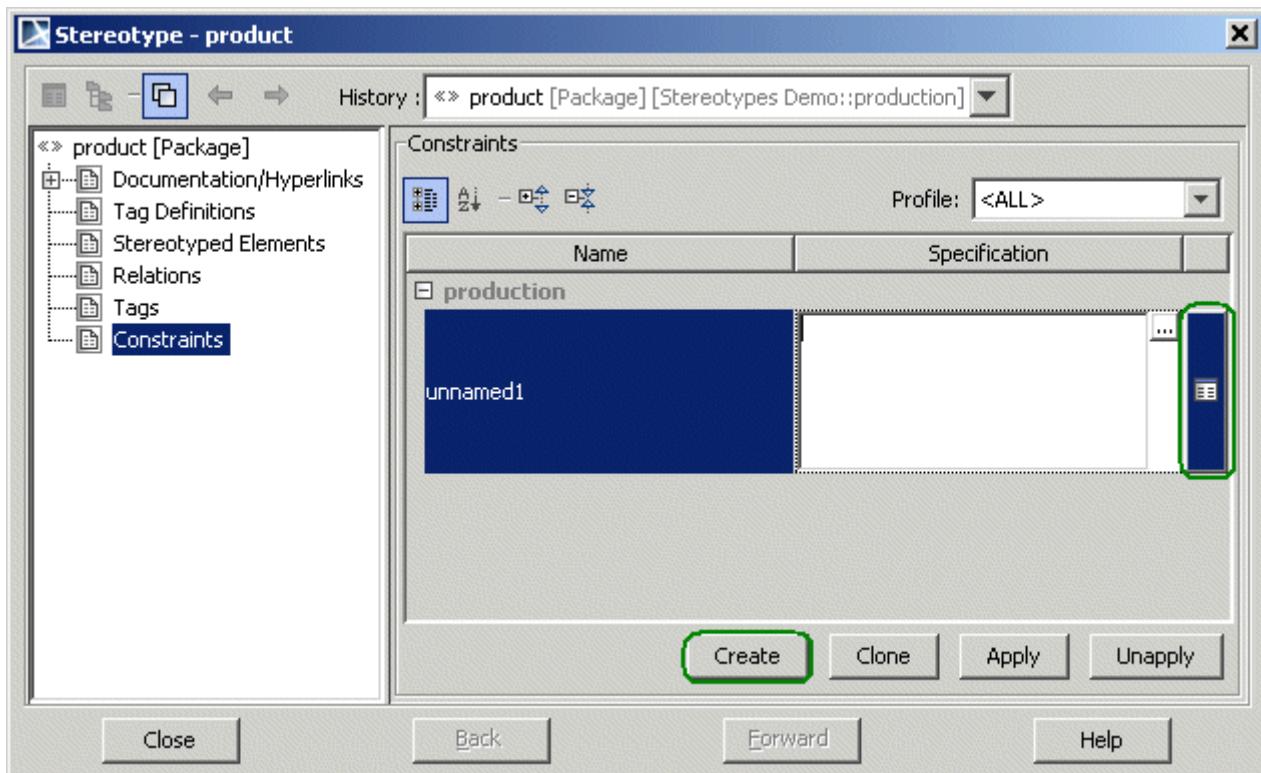


Figure 304 -- Creating constraints

2. The name of the constraint and the expression can be specified right away, but since we want to specify more information, we need to open the specification of a newly created constraint (press the button on the right of the constraint).

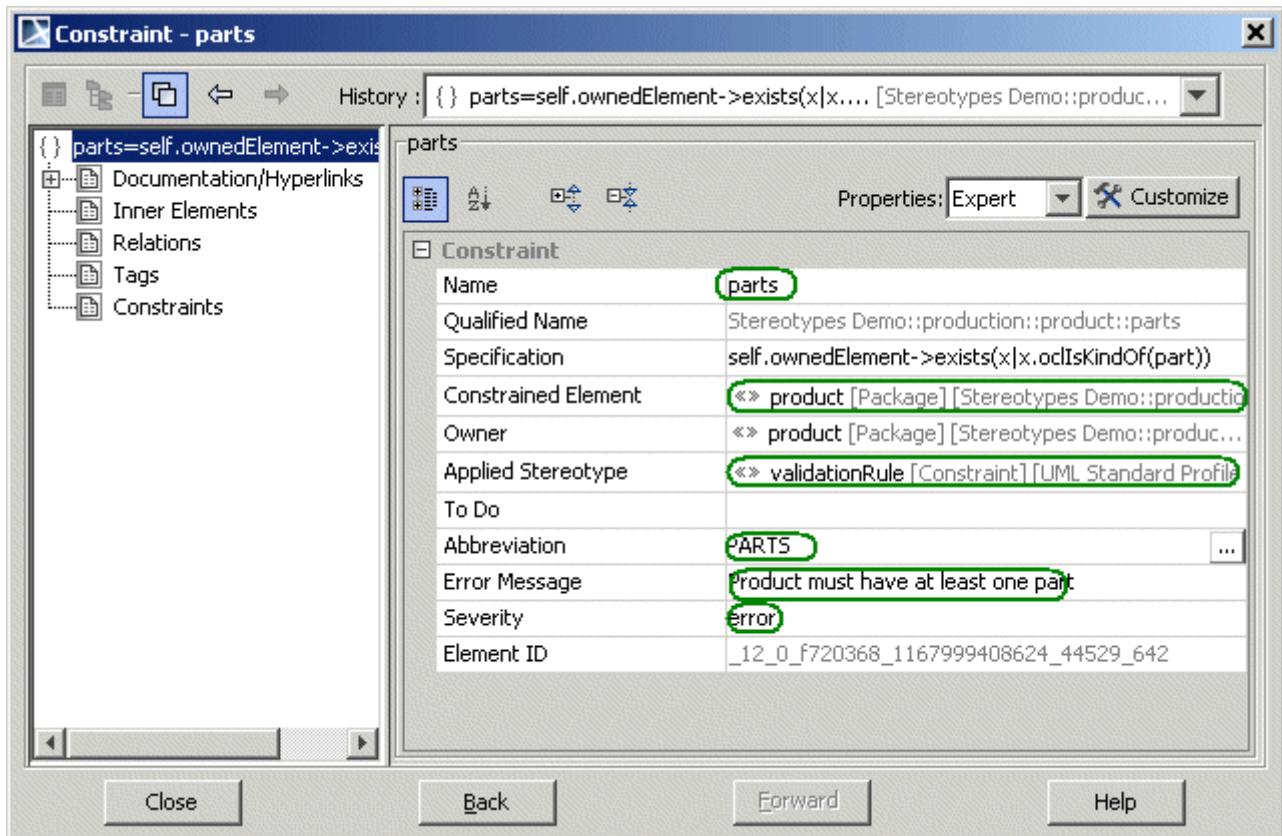


Figure 305 -- Specifying details of the constraint

3. In the specification panel of the constraint, specify the constraint name. Then ensure that the *Constrained Element* field points to the necessary classifier (*product* stereotype in our case). If we have created the constraint as described here, this field will be filled automatically. If we have created the constraint through the right-click, *Create Element, Constraint* route (e.g. constraint for metaclasses), we will need to specify the constrained element manually. For constraints on metaclasses select the appropriate element from the *UML Standard Profile::UML2 Metamodel*. In UML2 the constrained element field is multivalued, but only single value is supported for validation rules.
4. Now apply the `<>validationRule>>` stereotype on this constraint. Additional fields will open on the pane (*Abbreviation*, *Error Message* and *Severity*). If these fields do not open automatically, click **Customize** and then **Reset to Defaults** in the open customization dialog box (you can also access these fields in the *Tags* section of the specification).
5. Fill in the values for those fields.

Usage of the Severity levels (approximate guidelines):

- **DEBUG.** This severity level should be assigned only to those validation rules, which fit the description of INFO, but are too numerous and annoying to constantly bother the user.
- **INFO.** Situations, which might be interesting to the user.
- **WARNING.** Used for less severe situations than ERROR, which are not errors per se, but have a high probability of causing errors. A good example would be – In Java model user redefines `equals()` method of the class, but does not redefine `hashCode()`. This is a dangerous coding situation.
- **ERROR.** Normal error message. For ordinary, run-of-the-mill errors.

- **FATAL.** Used for the errors, which lead to model corruption or are not valid from the UML metamodel structure viewpoint. There should be few or no validation rules of this level since MagicDraw automatically precludes such situations. This level is mostly reserved for future use.

Abbreviation is a simple (and preferably short) string, for quickly distinguishing the validation rules among other rules and sorting. Acronyms and short forms, used in the domain of this validation rule can be used here (e.g. NPE for hypothetical NullPointerException check).

Error Message is a longer string, fully describing the invalid situation.

Now that we have all the peripheral information about the validation rule, let's specify the actual validation rule expression. Validating expression is stored in the *Specification* field. UML2 expression has 2 fields – *Language* and *Body*.

MagicDraw supports 2 languages for expressions, that can be evaluated:

- **OCL2.0** is used for validation rules, specified in OCL language (version 2.0 of the spec - 06-05-01 specification document from OMG).
- **Binary** is used for more advanced expressions, which are not easily expressed in OCL.

These expressions are written in Java, compiled, specified in the MagicDraw classpath. Then these expressions can be specified as validation rule expressions

Other languages are not evaluable (OCL1.5, English and others). They can be used for documentation purposes.

OCL Constraints

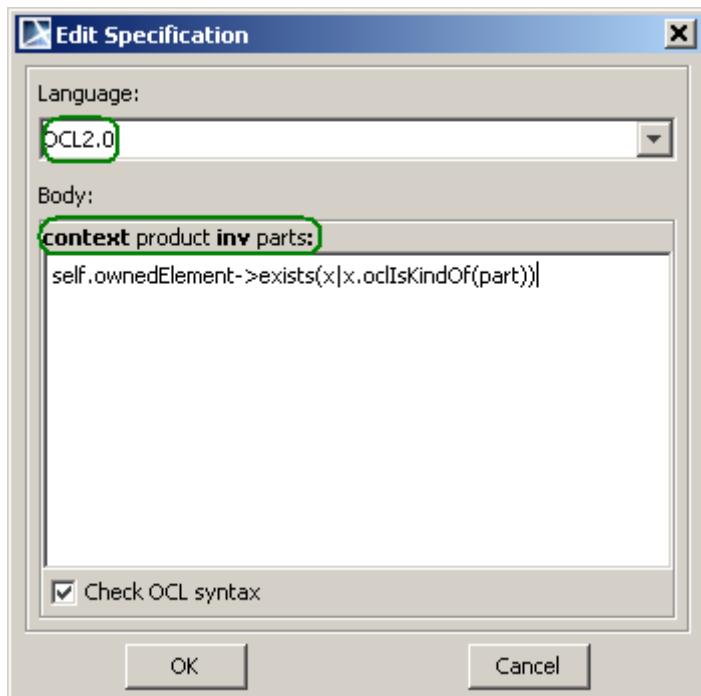


Figure 306 -- validation rule in OCL

Continuing our example, in the constraint **Specification** dialog box, click the button near the *Specification* field and open the **Edit Specification** dialog box. Select **OCL2.0** language. Observe that MagicDraw has automatically generated the header of the expression from the constraint information, and we only need to specify the body of expression. The expression header is generated according to the following rules:

context <constrained element> <constraint type> <constraint name if any>:

Constraint type is one of the types, defined in the OCL2.0 spec:

- **inv** - when the expression is placed in the constraint with <<invariant>> stereotype applied.
- **def** – when the expression is placed in the constraint with <<definition>> stereotype applied.
- **init, derive** when the expression is placed in the default value of the property.
- **pre, post, body** when the expression is placed in the appropriate fields of operation.

Since our constraint is stereotyped with <<validationRule>> stereotype (derived from *invariant* stereotype), **inv** is shown in the header. Only invariant stereotype is used for the validation rules and are executed (plus derive expressions, when referenced from invariants – see “Advanced Topics” on page 458), other types of constraints can be used for documentation purposes.

MagicDraw checks the syntax of expression as you type. However this syntax check is not enough to catch all the errors. When the validation rule is run, additional checks are performed (semantic checks – such as checks for the existence of appropriate properties, type checks, multiplicity checks, etc.) to ensure that the expression can be evaluated correctly (internally, MagicDraw generates Java code from the expressions and then compiles it for execution).

Binary Constraints

For more information about Binary Constraints, see MagicDraw OpenAPI UserGuide.pdf.

Validation Suites

In MagicDraw, you can create your own validation suites or use one of the predefined ones as: *UML completeness constraints* and *UML correctness constraints*.

The validation suite defines the set of validation rules, which will be applied when validating. The purpose of the validation suites is to group constraints without duplicating them.

To create a new validation suite

We need to check the Oracle model for correctness, but not all constraints in *Generic DDL constraints* suite are suitable for our Oracle model. We will create a new suite with a narrow constraint collection.

1. From the **Analyze** menu, choose **Validation** command, and then **Validation Options**. The **Validation Options** dialog box opens.
2. Click the **Create New Validation Suite** button and name it *Oracle specific constraints*.

3. Define the validation rules in the **Validation Rules** pane.

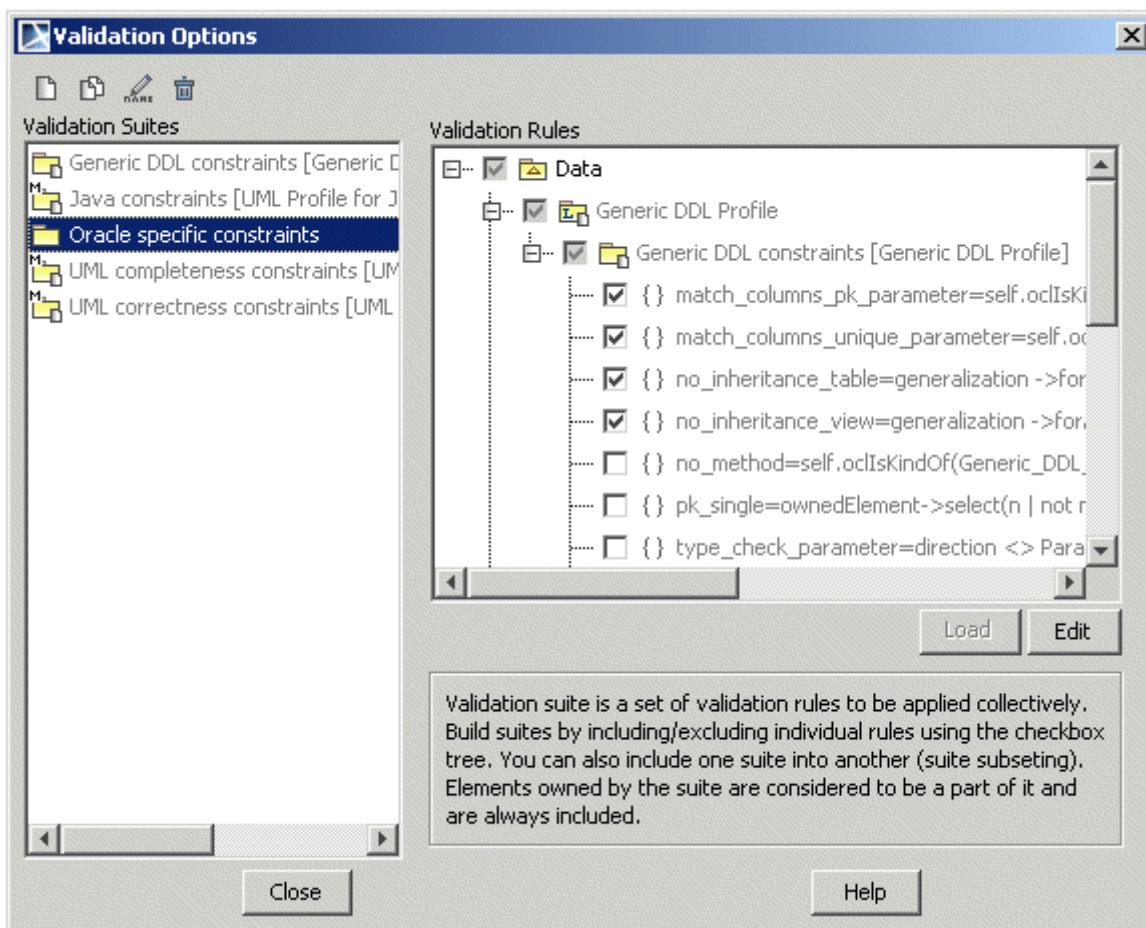


Figure 307 -- Parts of Generic DDL constraints suite are included into Oracle specific constraints suite.

The validation suite is stored in a model as a package, to which <<ValidationSuite>> stereotype is applied. The **Validation Suites** pane lists all such packages of the entire model as suites. So, the alternative method to create the validation suite is to apply <<ValidationSuite>> stereotype for a package.

When the user includes / excludes the constraint, the appropriate element import link is created/deleted in the model.

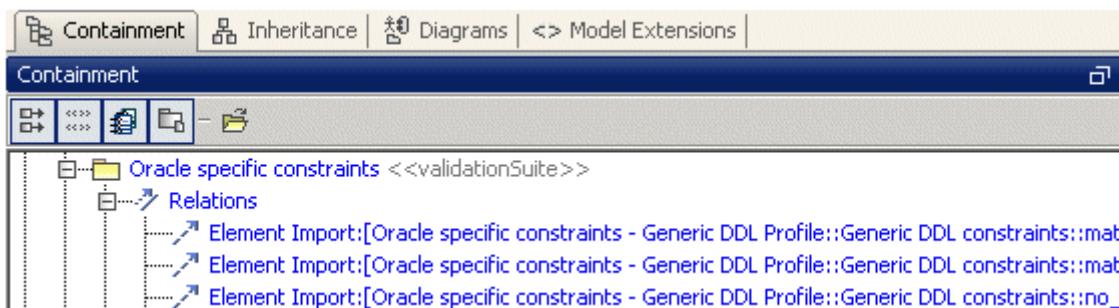


Figure 308 -- Element import relation showing in model that Oracle specific constraints suite includes other constraints.

Also, there can be constraints that are stored directly in the suite package - they are also considered as contained in this suite, and because they are physical in package those constraints can not be excluded from the suite through **Validation Options** dialog box. Typically, validation rules should be stored in a constrained element, but in cases when the constrained element is read-only, for example it is stored in a read-only profile, adding constraints to it requires profile editing and a separate constraints grouping is easier.

Constraint Tree

The constraint tree is shown in the right **Validation Options** pane. This tree shows all the constraints with `<<invariant>>` or `<<validationRule>>` stereotype applied, presented in the model, together with the appropriate grouping elements. Each item has a check box, indicating inclusion or exclusion of the constraint in the selected validation suite.

The constraint tree contains packages and other model elements. If it contains constraints, they are arranged according to their containment in a model. Additionally, this tree contains other validation suites. The user can include / exclude rules and these rules must suit the selected validation suite by selecting / unselecting these check boxes in the tree.

To group two or more suites into one

We have created an abstract system model, and modeled its implementation with Java specific classes. To check this model completeness, correctness, and conformity to Java language by using three suites. We will combine all these suites to one in order to simplify the model checking.

1. From the **Analyze** menu, choose **Validation** command, and then **Validation Options**. The **Validation Options** dialog box appears.
2. Create a new suite, name it *General*, and select it.
3. Include the existing validation suites: *Java constraints*, *UML completeness constraints* and *UML correctness constraints* in the suite by selecting the check box in front of the packages in the **Validation Rules** panel.

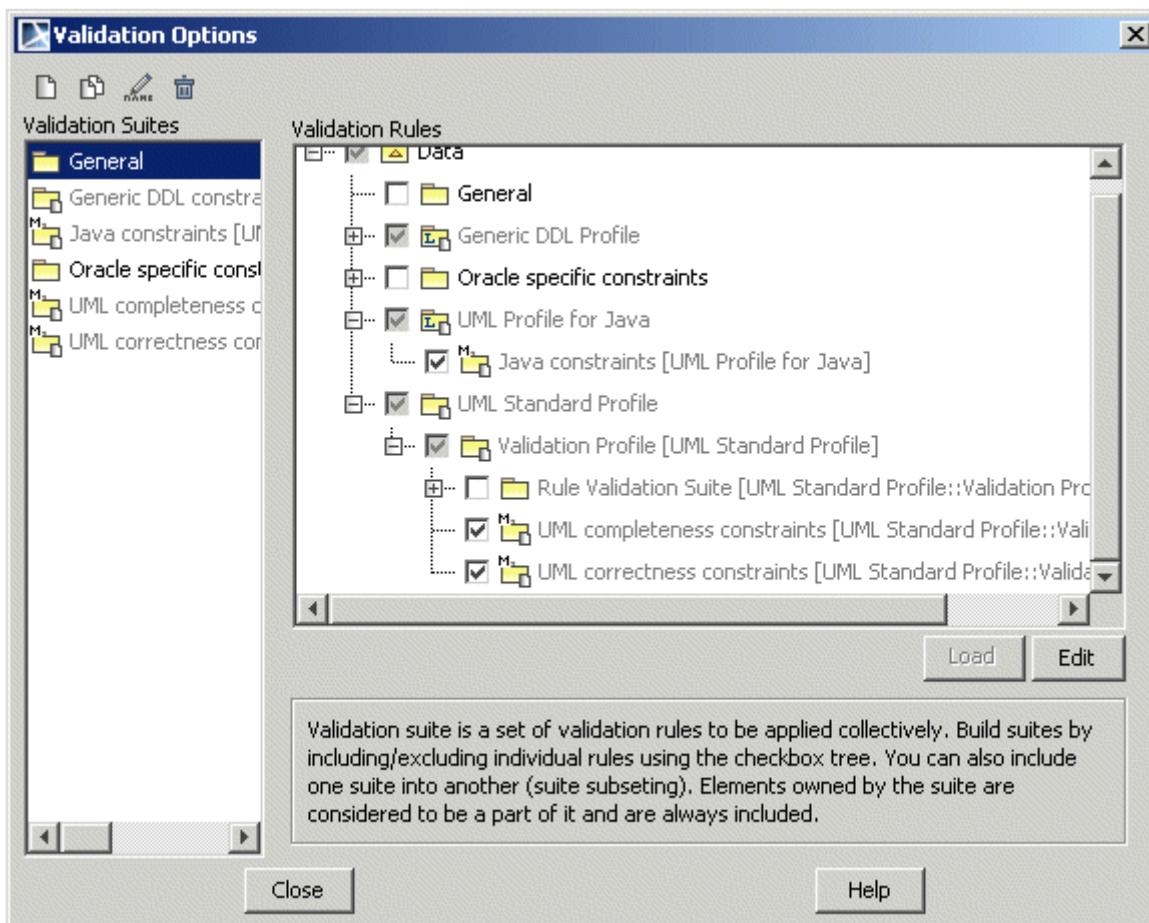


Figure 309 -- UML correctness, UML completeness, and Java validation suites included in the General validation suite

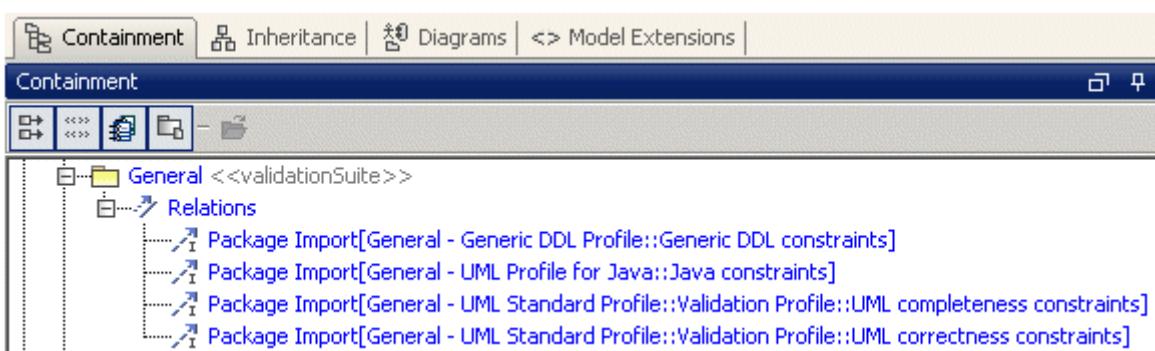


Figure 310 -- Package import is signifying in the model that one General suite includes the other as a subset.

To share constraints

Let's say we have created a validation suite with constraints and need to share it for other group members for their models validation. Validation rules/suites sharing is available through standard MagicDraw module mechanism. Package with constraints might be exported as module and used by any other project.

Another way of sharing constraints is copying them between projects. Since validation rules are simple model elements, any mechanism can be used on them.

1. From the model select *Java constraints* package to which `<<ValidationSuite>>` stereotype is applied.
2. From the package context menu select **Modules** and select **Export module**.
3. Save the exported package as *Java constraints.mdzip*.

Only constraints that are stored physically in the *Java constraints* package are exported together with the package.

Now the exported package can be used by other users and projects.

1. From the **File** menu, select **Use Module**. The **Use Module** dialog appears.
2. Select path to *Java constraints.mdzip* and select it to use.
3. Specify module import options.
4. Module is added into a project and the constraints can be used for validation.

The validation suite can be defined in the module, which is mounted as *read-only* in the project. **Read-only** and **Autoload** module will not be loaded into the project, but will be visible through validations dialog box if `<<ValidationSuite>>` stereotype was applied to the exported package.

Figure 311 -- In this way the model and the profile will be smaller. And the validation suite is still visible through validation dialog box. Java constraint validation suite module is mounted onto the project as Read-only and Autoload. In this way constraints are not added into project by default, until the validation suite is used.

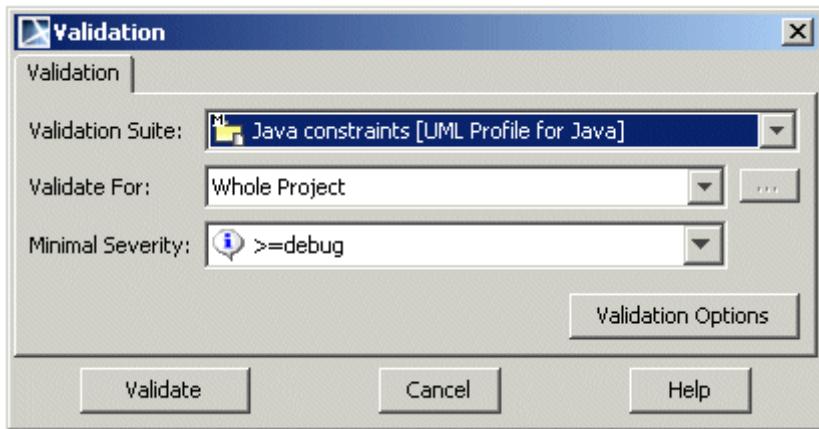


Figure 312 -- Unloaded module with Java constraints validation suite is available through Validation dialog.

Advanced Topics

Global validation rules

Some of the validation rules, specified in OCL, do not refer to the current element (*self*). Such rules are often encountered when using `allInstances()` method to refer to all instances of a particular classifier. Such validation rules are called global validation rules in MagicDraw terminology. Evaluating these rules for each model element is pointless, hence they are evaluated only once per entire validation run. When reporting violations, such rules have a string `<model>` in the column of violating model elements. This means that it is not the concrete element that violates the rule, but the entire model itself.

Here is an example of such a rule (always fails):

```
context anything inv:  
false
```

Such a rule is not very useful indeed. The following is another example:

```
context SomeSingletonClass inv:  
SomeSingletonClass::allInstances()->size() <= 1
```

This rule checks that there is at most one instance of the `SomeSingletonClass` in the model. The following is a more complex example:

8 MODEL ANALYSIS

Validation

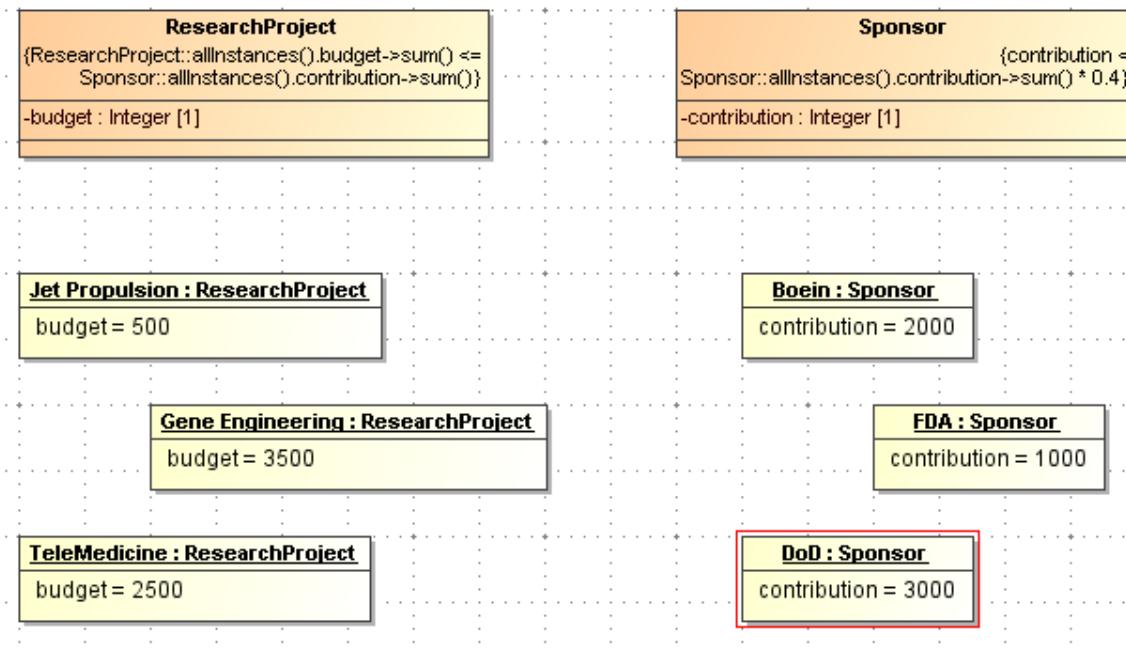


Figure 313 -- example of global validation rule

Here, *ReseachProject* class has a following validation rule (budget must be balanced – sum of expenses of all projects must be less than sum of all sponsor contributions):

```
context ResearchProject inv balanced_budget:
ResearchProject::allInstances().budget->sum() <=
Sponsor::allInstances().contribution->sum()
```

Sponsor class has a following rule (anticorruption rule - each sponsor can not contribute more than 40% of the funds):

```
context Sponsor inv anticorruption_law:
contribution < Sponsor::allInstances().contribution->sum() * 0.4
```

Now look at the results of applying these rules:

| Validation Results | | | |
|--|----------|--------------|---|
| Validation Results | | | |
| Element | Severity | Abbreviation | Error Message |
| <model> | error | BDG | Budget not balanced - overbudget by 500\$ |
| DoD:Classifiers Demo::Sponsor [Classifiers Demo] | error | Sponsor | Each sponsor contribution must be <40% of all contributions |

Figure 314 -- results of running the global validation rule

We see that budget balancing rule is a global rule – it is not the concrete instance of *ResearchProject* that violates the rule, but the entirety of instances in the model. Hence the string <model> in the column of that offending elements.

However, note that anticorruption rule is not a global rule - it refers to *contribution* field, which is really a short-hand for *self.contribution*, hence this rule refers to self variable and therefore, is not global and is evaluated for each instance of the *Sponsor* class separately.

MagicDraw has no means to determine if the binary validation rules are global, hence all binary rules are treated as local.

Expressions in error messages

When specifying error messages, more than a simple error string can be entered. Error messages can have template areas, which hold expressions, that will be evaluated and expanded when displaying validation results. Refer to the *ResearchProject/Sponsor* example above. The validation result shows:

Budget not balanced – overbudget by 500\$

Where does the number 500 come from? It is not directly specified in the error message string (since it is different for different models) but a calculated value of the expression, embedded in an error string. This error string in this case is:

```
Budget not balanced - overbudget by {  
ResearchProject::allInstances().budget->sum() -  
Sponsor::allInstances().contribution->sum()}$
```

Expressions are embedded in the error messages by using curly brackets - {}. Everything between them is treated as an expression and evaluated for each validation result. Expressions are treated as OCL2.0 expression by default, however you can also use binary expressions. In this case use {bin: <binary expression>} syntax.

Modeling other types OCL2.0 constraints/expressions

Only **inv** constraints can be evaluated in MagicDraw. However, there are more constraint types defined in the OCL2.0 specification. There are also **def**, **init**, **derive**, **pre**, **post**, **body** constraints. These constraints are not evaluated, but can be modeled for documentation purposes. Here is how to model them:

- **def** – create a usual constraint, but apply <<*definition*>> stereotype, instead of <<*invariant*>> or <<*validationRule*>>.
- **init** – place an opaque expression in the defaultValue field of the property.

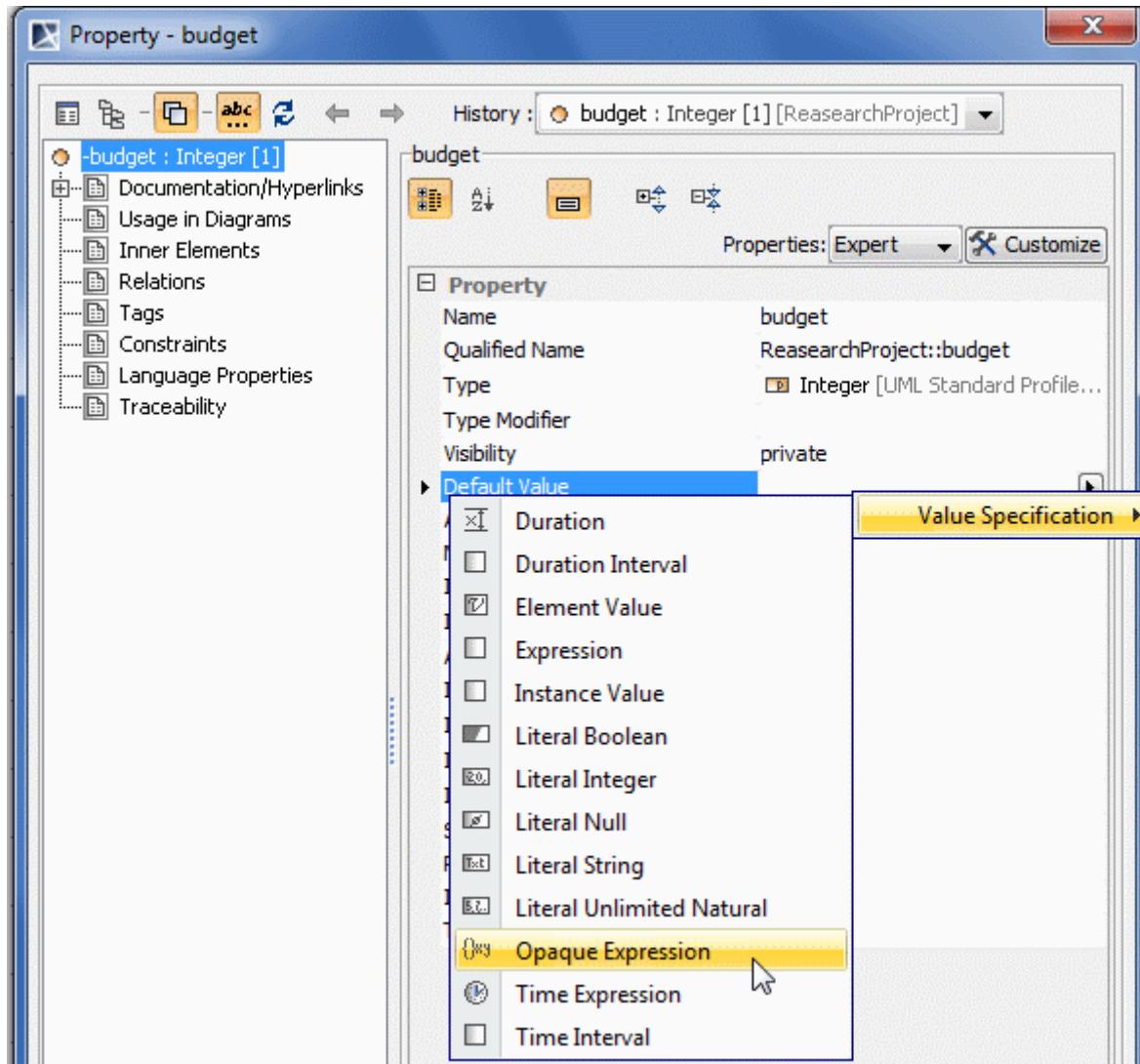


Figure 315 -- setting the expression as a default value of property

To place an opaque expression in the default value of the property, click the button with a small black arrow pointed to the right or right click on the default value field in the specification of the property and from the shortcut menu select **Value Specification > Opaque Expression**.

- **derive** when the expression is placed in the default value of the property (the same as for **init** expressions) but the property is marked as derived in the specification.
- **pre, post, body** when constraints are placed in the appropriate fields of operation (precondition, postcondition and body condition respectively).

Note that **derive** expressions can be evaluated indirectly, when the validation rule (**inv** constraint) is referencing the property and the validation rule is evaluated.

Unsupported OCL2.0 features

Not all OCL2.0 features are supported in the current release of MagicDraw.

In particular these features are not supported:

- Distinction between null values and undefined values.
- Tuples.

- All the operations, defined in the UML2 superstructure specification on the metaclasses are not present and are not callable.
- Defining and calling operations on classifiers.

There may be some other features that are not working properly. These issues can be reported to <https://support.nomagic.com>. MagicDraw uses external library – Dresden OCL Toolkit for constraint evaluation.

Adding/customizing severity levels

If the default severity level choice, provided by MagicDraw is not enough for you, new severity levels can be added. This can be done by editing the *SeverityKind* enumeration in the *UML Standard Profile::Validation Profile* package. Each enumeration literal in this enumeration corresponds to available severity levels. Severity levels in this enumeration should be sorted in ascending order.

If you need to specify a new icon for your custom severity level:

1. Create stereotype, derived from the *imed* stereotype, with the *EnumerationLiteral* as base class.
2. Set the necessary icon on this stereotype.
3. Apply this stereotype on your custom severity level enumeration literal.
4. Additionally, specify the *highlightColor* tagged value on the literal. This field (of the *String* type) determines how the offending elements will be highlighted in the diagrams. The string format is the same as for specifying colors in HTML pages (as described in <http://www.w3.org/TR/html4/types.html#h-6.5>). Simple string constraints (such as *highlightColor="red"*) or numeric values (such as *highlightColor="#FF0000"*) can be used here.

Performance Issues

When validation rules, written in OCL are evaluated, MagicDraw generates Java source for them and invokes Java compiler to compile them into an executable form. Hence, on the first run of validation there is usually a delay of 20-30 seconds (depending on the computer performance) while Java compiler is loading. Subsequent runs will be faster than the first one.

Also, this process consumes some amount of RAM. If the validation process is run heavily on medium-large projects, increasing the default Java VM size is advisable. By default, VM size is set to 400MB in MagicDraw; increasing this to 600 (or 800 if the computer has sufficient RAM) might improve the performance.

Active Validation

[View Online Demo](#)

Active Validation

NOTE

This functionality is available in all MagicDraw editions.

Active Validation instantly checks the accuracy, completeness, and correctness of a model, displays errors in the model, and suggests solutions.

The following modeling cases are validated with the Active Validation:

- Parameters and arguments synchronization validation
- Correct ownership validation

- Orphaned proxies' validation.

Active Validation is an extendable mechanism that identifies common problems and solutions. Custom validation suites and constraints can be created using binary or OCL constraint. Invalid symbols are marked on diagrams and elements in the model.

Errors in the model are represented in the following ways:

- If an invalid or incomplete model is created, an error indicator will appear in the bottom right corner of MagicDraw.
- Invalid elements are marked in the Browser and Diagram.

From the invalid element/symbol shortcut menu, you can analyze incorrect elements, and solve problems, resulting from the errors, through the **Active Validation Results** window.

Model is validated with three predefined validation suites and the Validation is performed for the parameters and arguments synchronization, correct ownership, and orphaned proxies. You can modify these suites and create your own through the **Validation Options** dialog box.

Detecting errors in the model

When Active Validation detects errors in the model, errors will be revealed in the following ways:

- the failure indicator
- in the Browser, an invalid element is marked with little cross an x symbol.
- on the diagram, invalid symbol is highlighted.

Failure indicator

If an error occurs, the failure indicator  will appear in the bottom right corner of MagicDraw (see Figure 316 on page 464). Click this indicator and the **Active Validation Results** window will open.

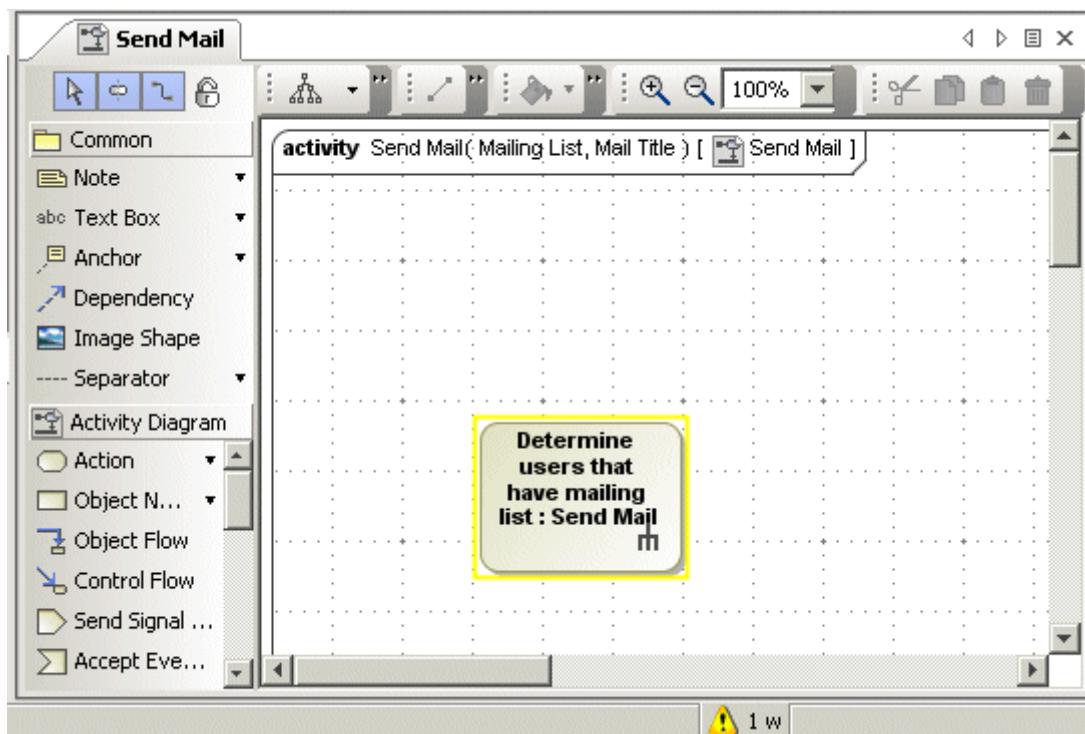


Figure 316 -- General validation notification in MagicDraw

See the parts of the failure indicator in the figure below.

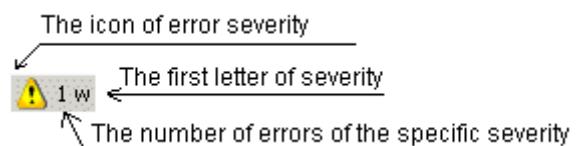


Figure 317 -- The failure indicator

| Example of the failure indication | Explanation | Possible values |
|-----------------------------------|---|---|
| ! | Error symbol showing the level of severity. | - indicating warning - indicating error or fatal error. - indicating debug error or info. |
| 1 | Number of errors of that specific severity | 1, 2, 3 ... |
| w | First letter of the error severity | F - fatal error E - error W - warning D - debug I - info |

TABLE 1. Parts of the failure indicator

Marking errors in the Browser

An invalid model element is marked in the Browser with a small x symbol (see Figure 318 on page 465). The owner of this element is marked with a small grey symbol.

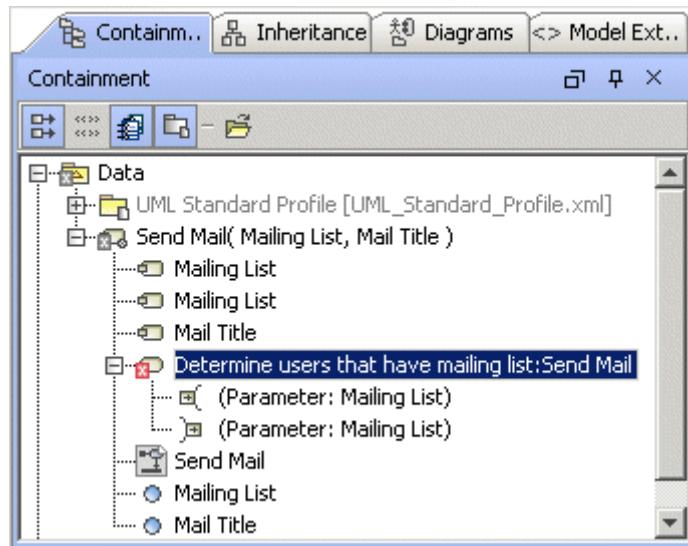


Figure 318 -- Invalid elements marking in browser

Highlighting errors on the diagram

The element symbol is colored according to the severity of the error on the diagram.

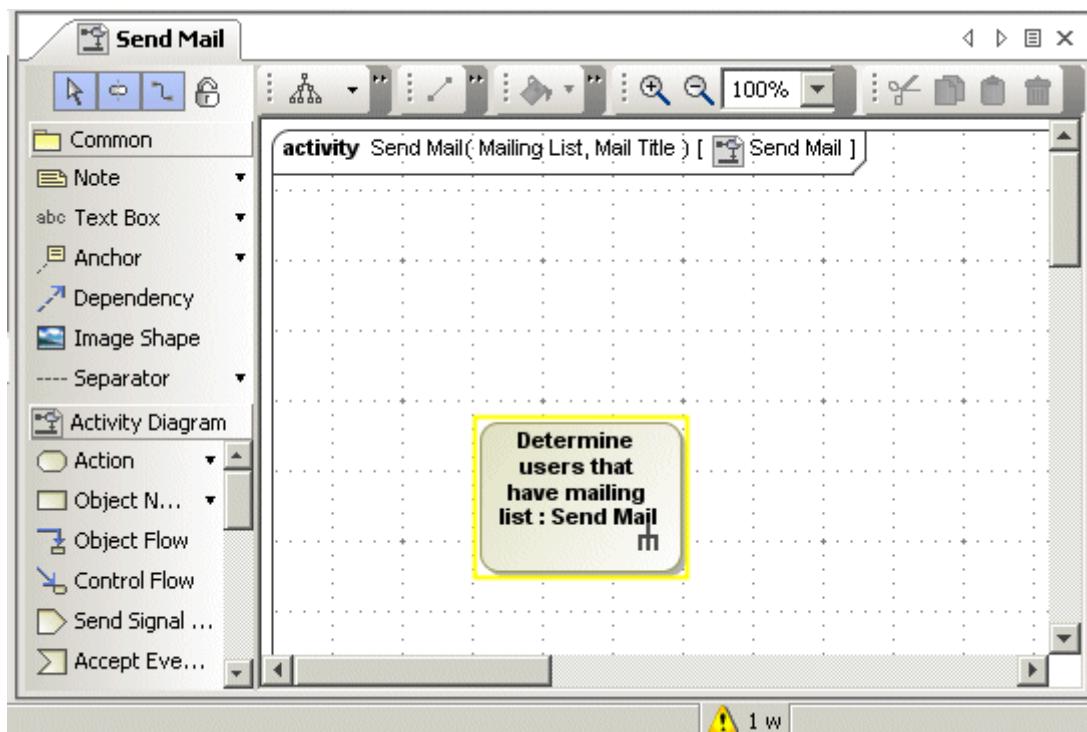


Figure 319 -- Invalid symbols marking in diagram

Handling incorrect model

The **Active Validation Results** window lists the active validation results. It can navigate users to have ability not only to navigate to constraints and invalid elements or symbols, correct errors, filter, and ignore problems.

| Active Validation Results | | | |
|---|----------|--------------|--------------------------------------|
| Element | Severity | Abbreviation | Error Message |
| Determine users that have mailing list:Sen... | warning | OutSyncBe... | Broken parameters synchronization of |

Figure 320 -- The Active Validation Results window

Changing the Active Validation Options

To change the active validation options:

1. From the **Options** main menu, select **Project**.
2. In the **Project Options** dialog box, select the **General project options** group.
3. In the **Active Validation** group, specify the active validation options.

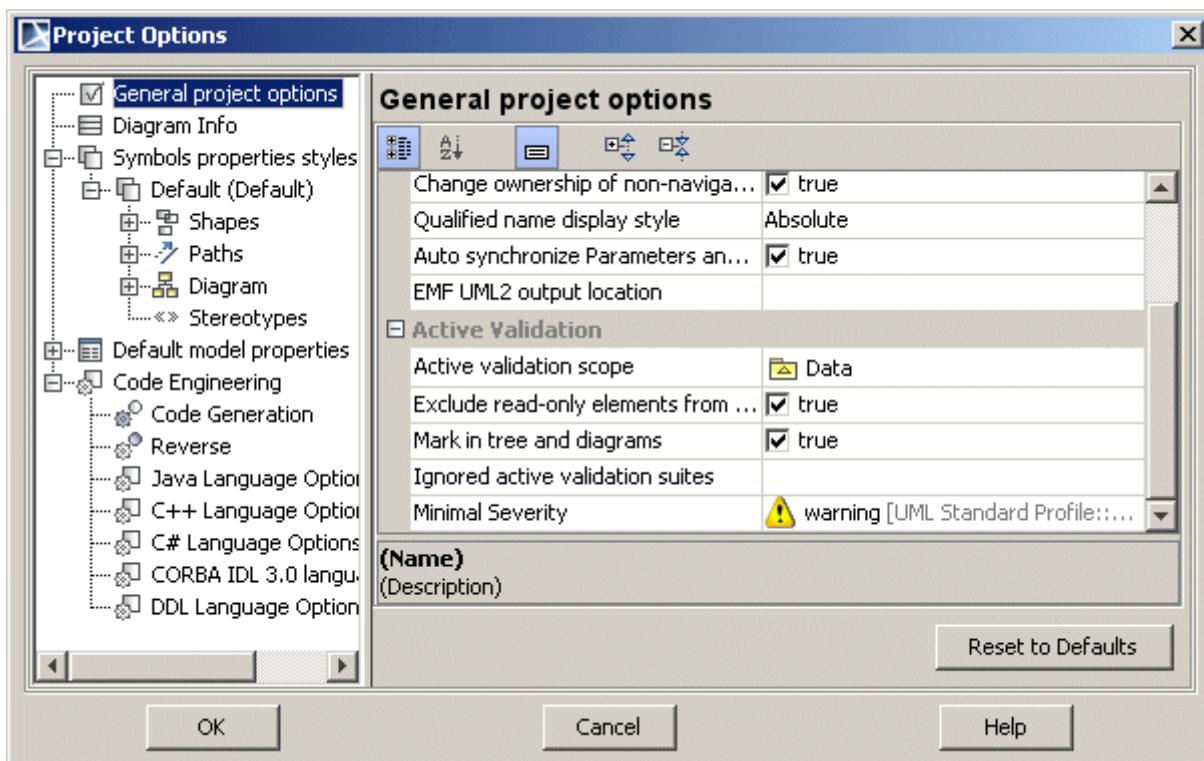


Figure 321 -- The Project Options dialog box, Active Validation group

The Active Validation Suites

Validating Parameters and Arguments Synchronization

More information about Parameter Synchronization can be found here “Parameters synchronization with Arguments” on page 736.

In most cases the parameters and arguments synchronization is not visible. Automation and synchronization between Parameters and Arguments (for example Operation parameters and Call Behavior Action Pins) increase modeling speed and helps avoid modelling errors.

Arguments should always be in sync with Parameters. But in cases when synchronization is not possible or corrupted, the active validation will notify the user by highlighting the symbols on the diagrams.

Shape Ownership

The term “symbol” means a visual representation of some model elements in the diagram. Symbols are further subdivided into shapes and paths (lines in the model, for displaying various relationships).

When drawing UML diagrams, the element ownership is not easily visible. One diagram can contain symbols for elements from several different packages. Element rearrangements in the model may lead to situations where the element ownership in the model does not match the symbol ownership as displayed in the diagram. Such situations are not easy to detect from diagram view.

MagicDraw version 15.0 has a built-in validation code to detect this mismatch. This feature is enabled by default and runs unnoticed without requiring any additional input from the user.

When the symbol ownership on the diagram pane does not match the actual element ownership in the model, the symbol is highlighted with red. So, you will easily see it on the diagram and will be able to correct the problem (problem correction hints are also suggested).



Figure 322 -- Class symbol highlighted with red border

When the mismatch is resolved, the highlighted symbol will return to normal.

The symbol ownership validation feature uses the same mechanism for problem highlighting as the generic validation feature available in MagicDraw Enterprise edition. If you run some validation suites against the model, the element can be highlighted due to any of the validations failures:

- either validation rule(s) from that suite,
- or this automatic symbol ownership checking rule(s).

For more information about validation, see “Validation” on page 444.

Symbol ownership validation covers two cases:

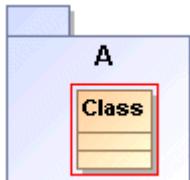
- Class diagram and its derivatives (use case, implementation, composite structure diagrams etc.) can display elements from many model locations and show their ownership. If shape owner is incorrect (not the same as the element owner in the model), this shape will be highlighted.
- Dynamic diagrams (state machine, activity and interaction diagrams) have a restriction that only elements from one definite state machine, activity or interaction can occur in each concrete diagram. Validation rule checks for these diagrams that only elements from correct state machine, activity, or interaction appear in the diagram. All foreign elements are highlighted as erroneous.

These cases are summarized in the table below

| | Validation of ownership on the diagram pane | Validation of diagram owner, on which element is drawn |
|------------------------|---|--|
| Validation rule | Check if the symbol owner on the diagram correctly reflects the element owner in the model. | Check if the element and the diagram on which the symbol is drawn belong to the same owner. This is valid for dynamic diagrams. The checking is performed according to the context of the diagram. |
| What is checked | Shapes are checked, excluding paths. | Validation works for shapes, including paths. |

8 MODEL ANALYSIS

Active Validation

| | Validation of ownership on the diagram pane | Validation of diagram owner, on which element is drawn |
|-----------------|---|---|
| Cases | <p>Check is performed:</p> <ul style="list-style-type: none"> • if the element symbol is draw on the same package/model/profile to which it actually belongs; • if the element symbol is drawn on the same component/node to which it actually belongs; • if the element symbol is drawn on the same state to which it actually belongs; <p>and other similar cases.</p> | <p>Check is performed:</p> <ul style="list-style-type: none"> • if the communication diagram and elements from the communication diagram are in the same interaction; • if the sequence diagram and its elements are in the same interaction; • if the state diagram and its elements are in the same state machine; • if the protocol state machine diagram and its elements are in the same protocol state machine. • if the activity diagram and its elements are in the same activity. |
| Examples | <p>For example, on the diagram pane, an element shape is nested in package A, but actually the element is in package B.</p>  | <p>For example, a diagram belongs to activity A, but elements of this diagram belong to activity B.</p>  |

Important Ownership validation is not available in the Business Process diagram.

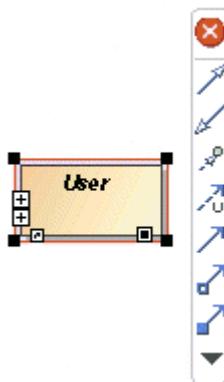
TIP To locate the actual owner of the element:

- Select the element on the diagram pane and right click. Select the **Select in Containment Tree** command. In the Browser you can see the actual element. The parent in the tree is the actual owner.
- In the element **Specification** dialog box, look up the **Owner** property.

Solving the detected symbol ownership problems

In this section you will find out the reasons why an element is highlighted in red and how to solve symbol ownership problems quickly.

Select the highlighted element on the diagram pane. The Smart Manipulator toolbar opens. Notice the red button on the top of Smart Manipulator toolbar.



To see the reason why a shape is highlighted, move the mouse pointer over the red button in the Smart Manipulator toolbar. You will see a tool tip that explains why the element is highlighted. For example: "Shape ownership in the diagram does not correspond to the element ownership in the model".

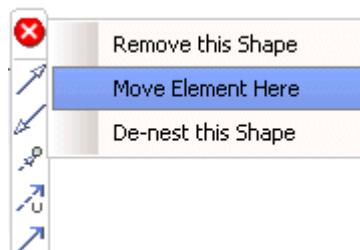
You may either solve the problem yourself or choose one of the suggested solutions. To select an available solution, click the red button in the Smart Manipulator toolbar. A menu with the following commands will open (note that some items might not be included depending on the situation in the model):

- **Remove this Shape.** Symbol is deleted from the diagram pane.
- **Move Element Here.** Element is moved to the new owner in the model, so that the element ownership in the model corresponds to the present shape ownership in the diagram.

NOTE

When a symbol is deleted, the element is not deleted from the project.

- **De-nest this Shape.** Shape (not element!) will be extracted from the current owner and placed directly on the diagram. This solution is applicable only for some cases (e.g. De-nest, this Shape command is not available for shape, placed in the incorrect state machine, activity, interaction diagrams).



To turn the ownership validation on/off

- Select/clear the **Validate Shape Ownership** check box in the **Analyze** main menu.
- Select/clear the **Validate Shape Ownership** check box in the **Environment Options** dialog box, **Diagram** branch, **Display** options group.

To define the ownership checking period

In the **Environment Options** dialog box, **Diagram** branch, **Display** group, **Diagram Checking Period** text box, type the diagram checking period in seconds.

Related Topics

- "Customizing Environment Options" on page 89.

- “Smart Manipulation” on page 202.

UML model correctness

Active validation instantly check the most important correctness rules of UML model. The following validation rules are checked: Ports compatibility, Pin types compatibility, Slot and Tags multiplicity correctness and others. Automated solutions are suggested for solving the model errors.

Validating the Orphaned Proxies (OP)

Checking for OP moved from a separate feature into one unified validation mechanism that identifies problems and solutions.

During model editing there is possibility can happen that an element is referenced using proxy, but the real element is not shared or does not exist in the module. After the modules that load such proxies are detected, they are called Orphaned Proxies. These proxies are marked by a validation sign in the browser. Actions to clean such proxies are added.

Customizing the Active Validation

Any constraint (binary or OCL), additionally optimized, can be used to validate models in real-time. MagicDraw provides predefined suites for validation of: Correct ownership in the model and on the diagram, Parameters and arguments synchronization, missing referenced elements (Orphaned Proxies) in the modules, and others.

A model is validated automatically without any additional input. Additional constraints can be added or properties can be customized from **Analyze** (main menu)-> **Validation**.

Validate element that has no representation in diagram

Element or its symbol is highlighted in diagram pane and in Browser if it is owner of element that can not be represented in diagram and has validation error.

See the examples of the spelling error in the table below:

| Sample description | Representation in Diagram | Representation in Containment Tree | Notes |
|---|---|---|--|
| Use Case documentation (comment owned by Use Case and annotating Use Case) has a spelling error | Use Case symbol is highlighted with red dashed border (see Figure 323 on page 472). | The owner of the hidden element with error is marked with white x in red quadrant in the Containment tree as itself would have error (see Figure on page 472). | To the symbol smart manipulator toolbar there is added additional button (grey circle with cross inside) that presents the errors solving solutions. |
| Use Case extension point name has a spelling error | Use Case symbol is highlighted with red dashed border (see Figure 324 on page 472). | The owner of the corrupted and not represented in diagram element is marked with white X in grey quadrant in the Containment tree (see Figure 325 on page 472). | |

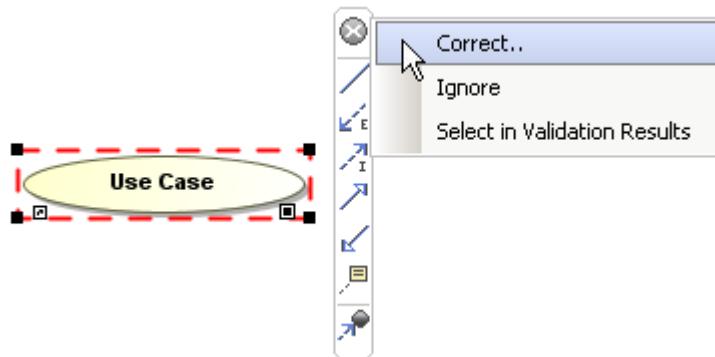


Figure 323 -- The Use Case is highlighted with red dashed border, because it has documentation with spelling error



The Use Case is highlighted with white X in red quadrant in the Containment tree, because it has documentation has a spelling error

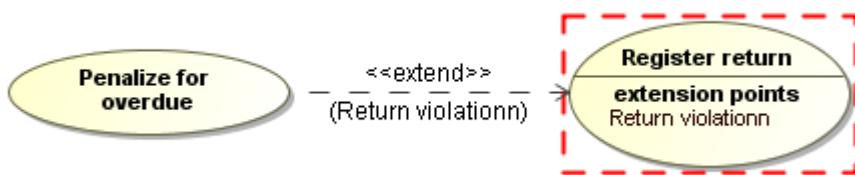


Figure 324 -- The "Register return" Use Case is highlighted with red dashed border, because the "Return violationnn" Extension Point has a spelling error

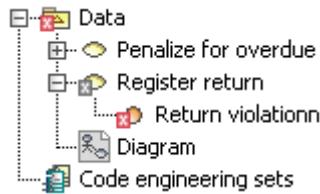


Figure 325 -- The "Register return" Use Case is highlighted with grey X in grey quadrant in the Containment tree, because the "Return violationnn" Extension Point has a spelling error

Model Visualizer

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

MagicDraw contains tools that help to create elements from existing data and analyze the relationships between elements created in the UML model. It is also possible to analyze the inheritance and dependency relationships between classes.

All model visualizing and analyzing tools are presented in MagicDraw as Wizards with several steps that should be followed in order to accomplish the desired operation.

All Wizards have several common buttons:

- **Add** – add selected model elements from the **All** list to the **Selected** list.
- **Add All** – add all elements to the **Selected** list that are located in the same hierarchy level as the selected element.
- **Add Recursively** – add all elements in the selected packages and all elements from nested packages to the **Selected** list.
- **Remove** – remove the selected element from the **Selected** list.
- **Remove All** – remove all selected elements.
- **< Back** – return to the previous dialog box.
- **Next >** – proceed to the next step.
- **Finish** – finish the configuration. All other options will be set by default. The **Wizard** exits and results are displayed.
- **Cancel** – cancel the wizard.
- **Reset To Defaults** - if changes were made to the element properties, values will be set to default.

Class Diagram Wizard

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The **Class Diagram Wizard** helps you create a new class diagram when all classes and their relationships are already created and specified. You can select which classes, packages, and relationships will be included in a new class diagram and the details of the class representation to be configured (attributes, operations, accessibility). The **Class Diagram Wizard** frees you from creating the class diagram manually. The **Class Diagram Wizard** guides you through several steps and collects information along the way. It will automatically create a new class diagram and all the necessary elements.

To start the **Class Diagram Wizard**

1. From the **Analyze** menu, select **Model Visualizer**. The **Model Visualizer** dialog box opens.
2. From the wizards list, select the **Class Diagram Wizard**.

3. Click **Start**. The **Class Diagram Wizard** opens.

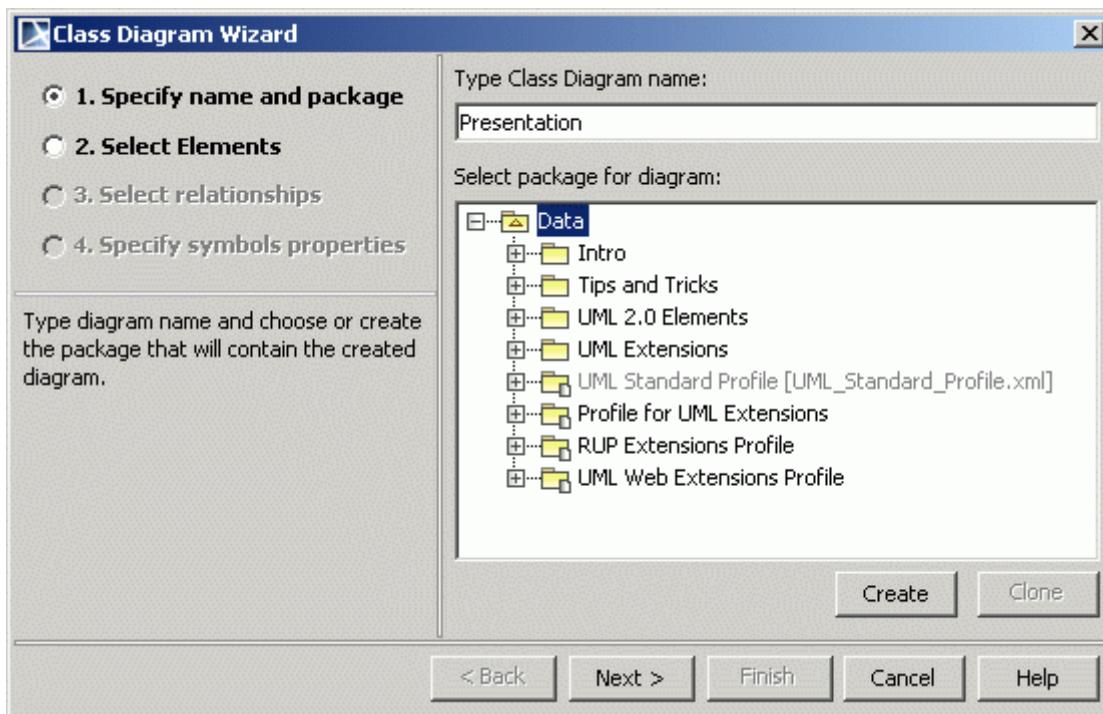


Figure 326 -- Class Diagram Wizard. Diagram name

Type the name of the new diagram in the **Type Class Diagram name** text box or leave the default name.

Select the package that will contain the created diagram. The hierarchy of UML model packages is displayed in the **Select package for diagram** list window. Select the package from the data tree that will be the parent for the newly created diagram or create a new package by clicking the **New** button.

When you select the **Specify name and package** option, the following functions are available:

- **Create** - create a new package.

8 MODEL ANALYSIS

Model Visualizer

- **Clone** - copy an existing package to a new one.

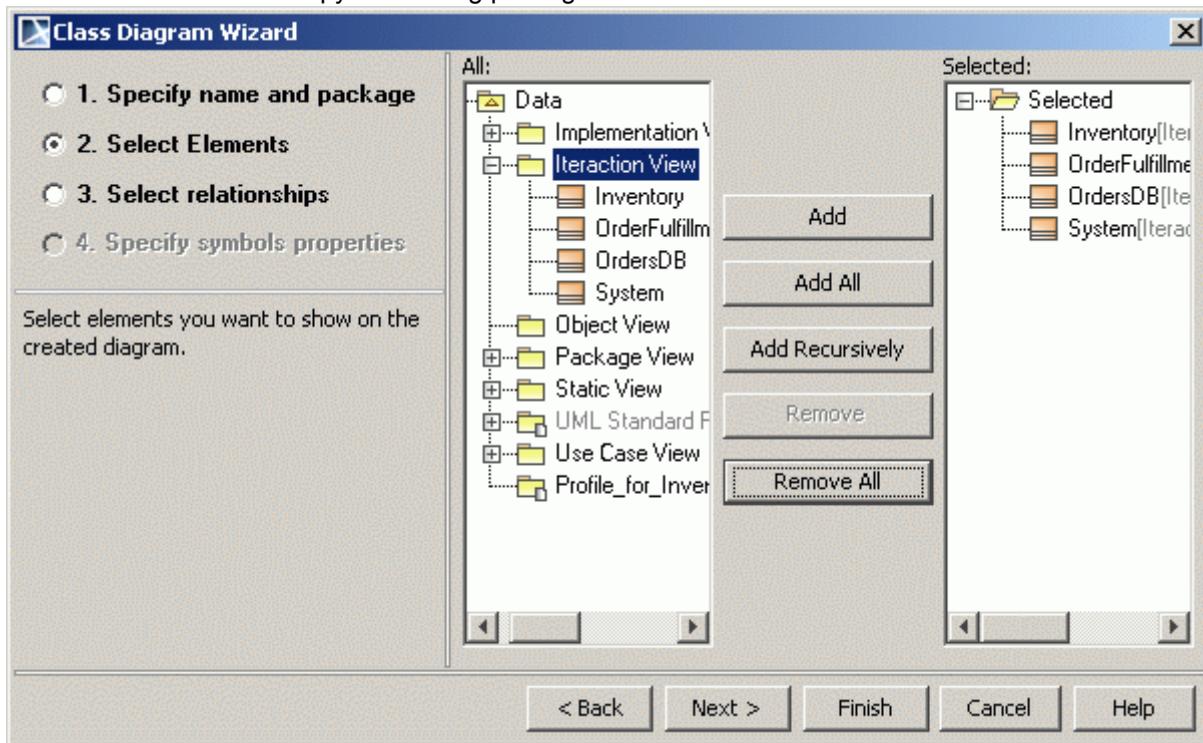


Figure 327 -- Class Diagram Wizard. Select Elements

- **All** list – contains all model elements.
- **Selected** list – contains the elements that are added to the class diagram.

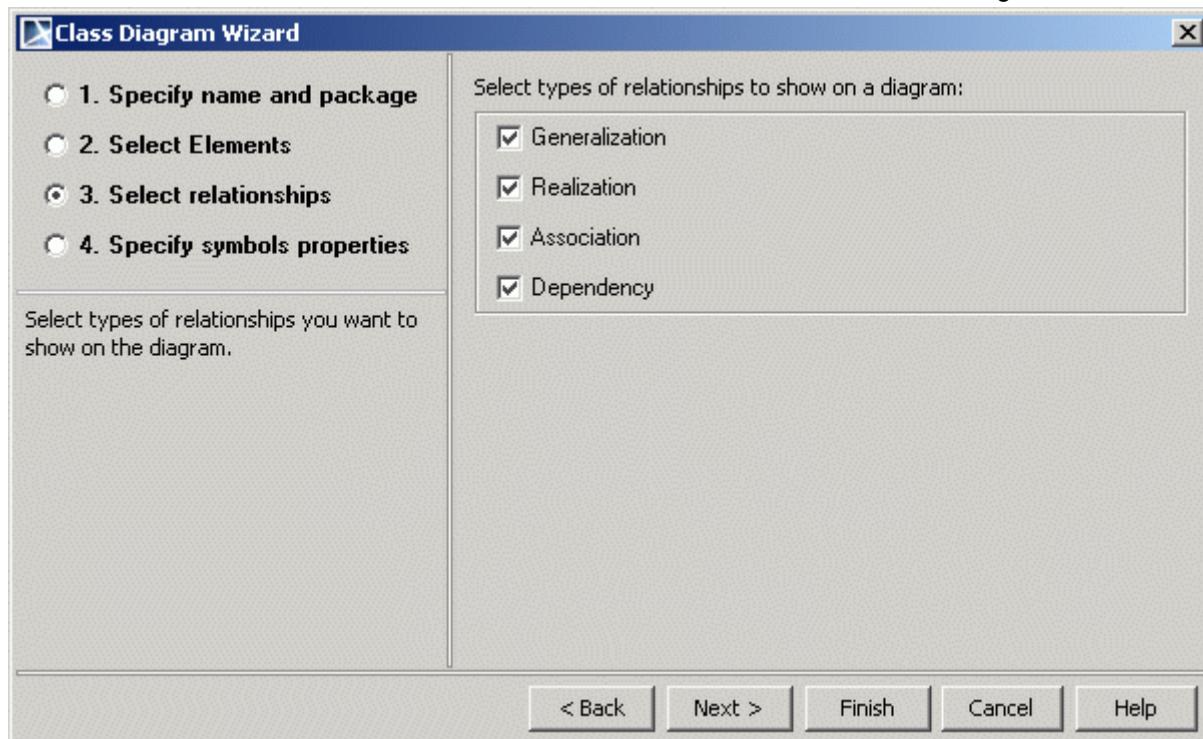


Figure 328 -- Class Diagram Wizard. Select Relationships

Select relationships to include in the class diagram:

- **Generalization** – relationship between a general element and a more specific element (inheritance, extension).

- **Realization** – relationship between model elements where one of the elements implements the interface defined by the other model element.
- **Association** – semantic relationship between classes.
- **Dependency** – usage relationship between UML model elements.

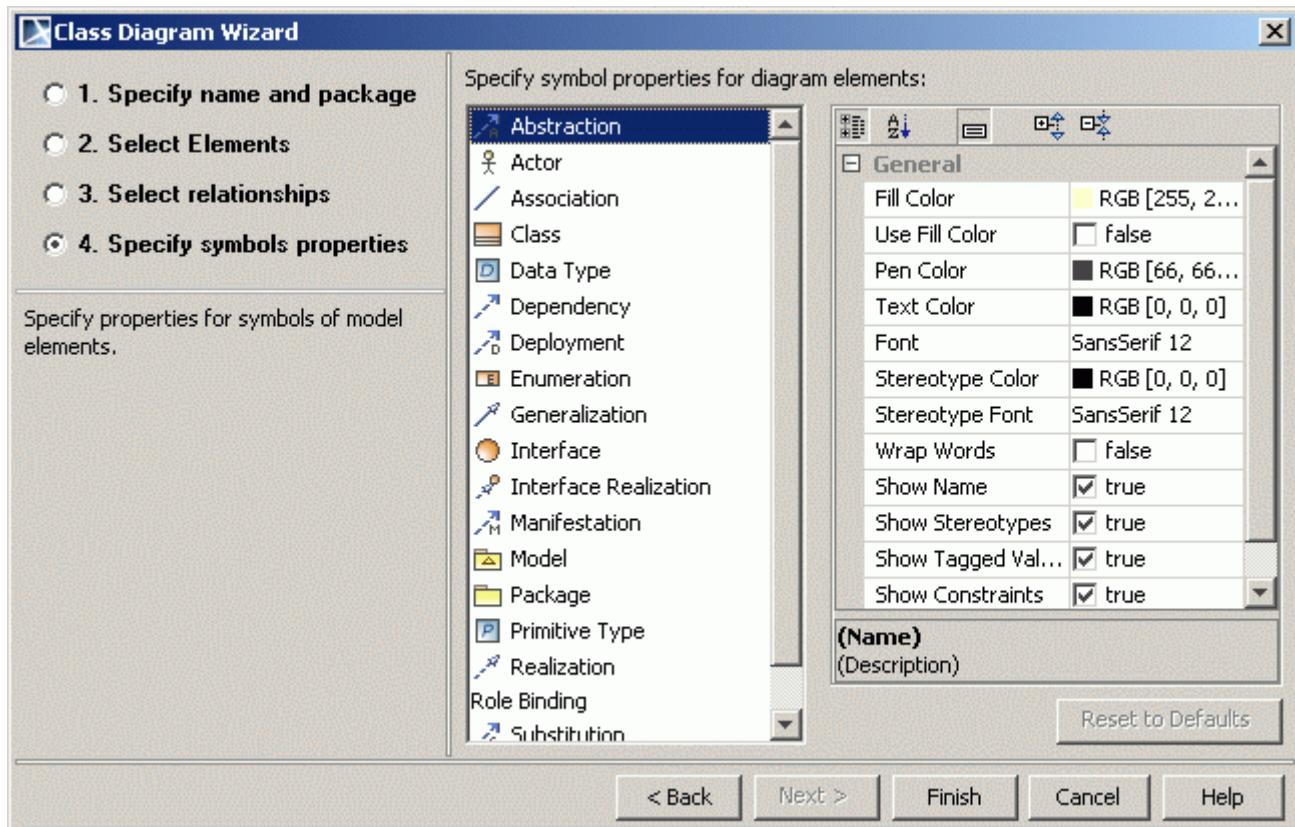


Figure 329 -- Class Diagram Wizard. Specify Symbols Properties

Select options for representing elements in the class diagram.

NOTE If the **Suppress Attributes** and the **Suppress Operations** check boxes are selected, the class is displayed only as a rectangle with the class name in it.

Package Dependency Wizard

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The **Package Dependency Diagram Wizard** generates diagrams containing packages (created within a project) and shows the relationships between them. The diagram may reflect all packages in the project, or just those selected. The **Package Dependency Diagram Wizard** collects all the information needed to both analyze dependencies and generate a new diagram.

To start the Package Dependency Diagram Wizard

1. From the **Analyze** menu, select **Model Visualizer**. The **Model Visualizer** dialog box opens.
2. From the wizards list, select the **Package Dependency Diagram Wizard**.

3. Click **Start**. The **Package Dependency Diagram Wizard** opens.

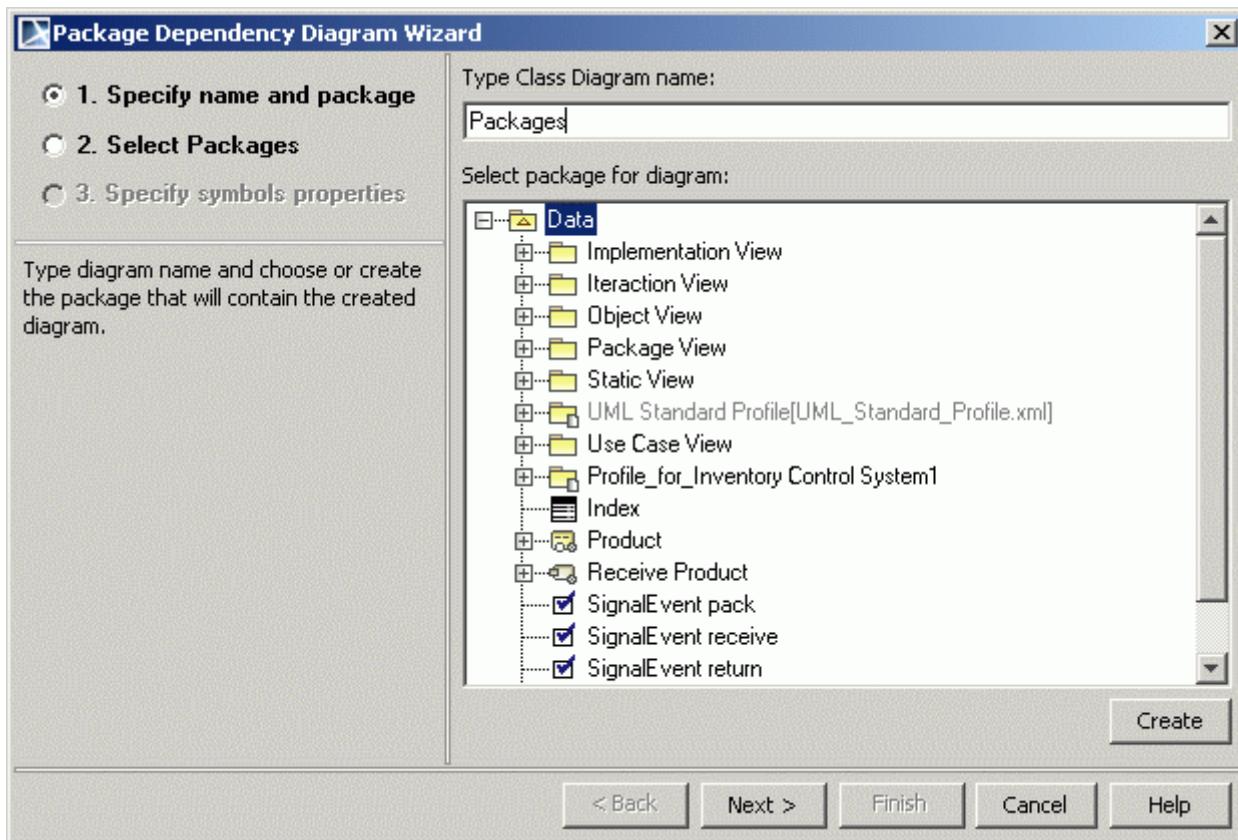


Figure 330 -- Package Dependency Diagram Wizard. Diagram name

Type a name for the newly created diagram in the **Type Class Diagram name** text box.

8 MODEL ANALYSIS

Model Visualizer

Select the package that will contain the created diagram. The hierarchy of UML model packages is displayed in the **Select package for diagram** list window. Select the package from the data tree that will be the parent for the newly created diagram or create a new package by clicking the **Create** button.

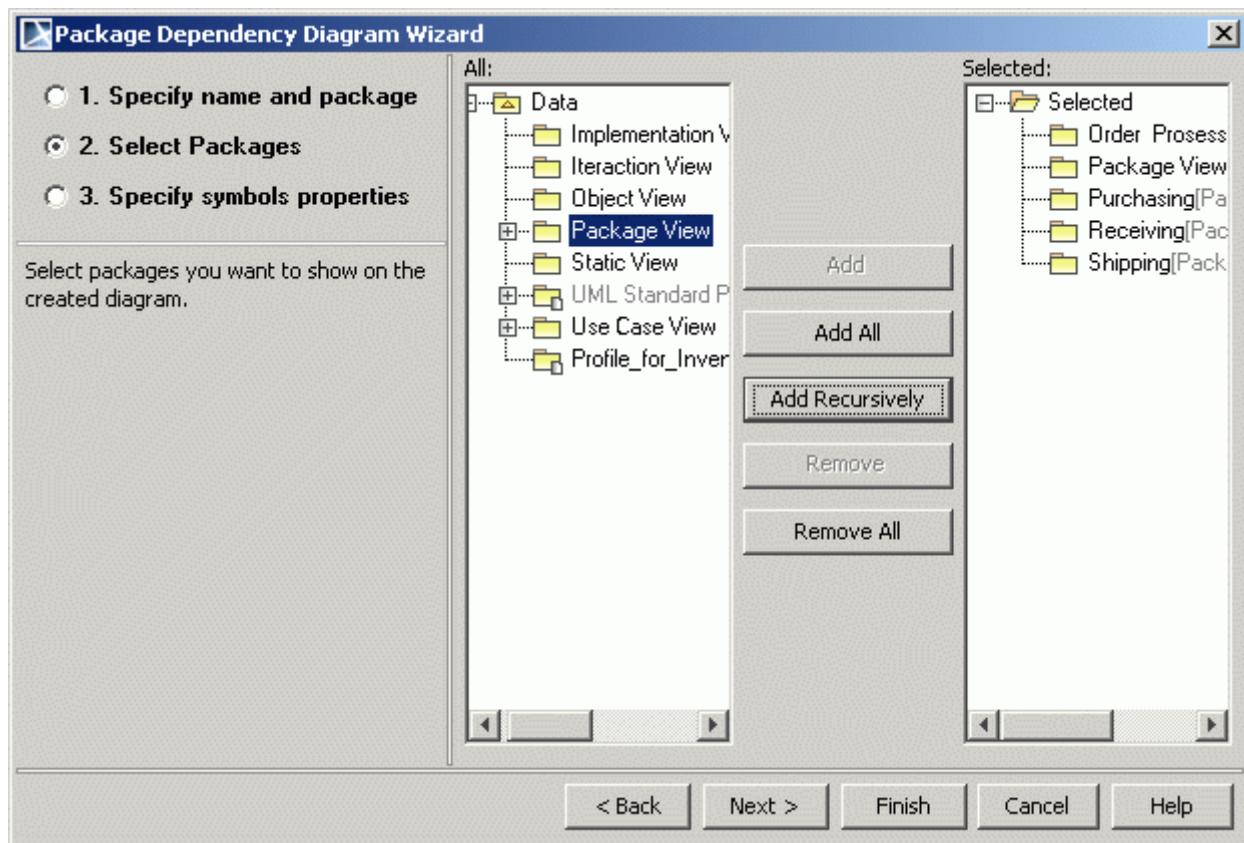


Figure 331 -- Package Dependency Diagram Wizard. Select Package.

- **All** list – contains all packages.

- **Selected list** – contains all packages that are added to the diagram.

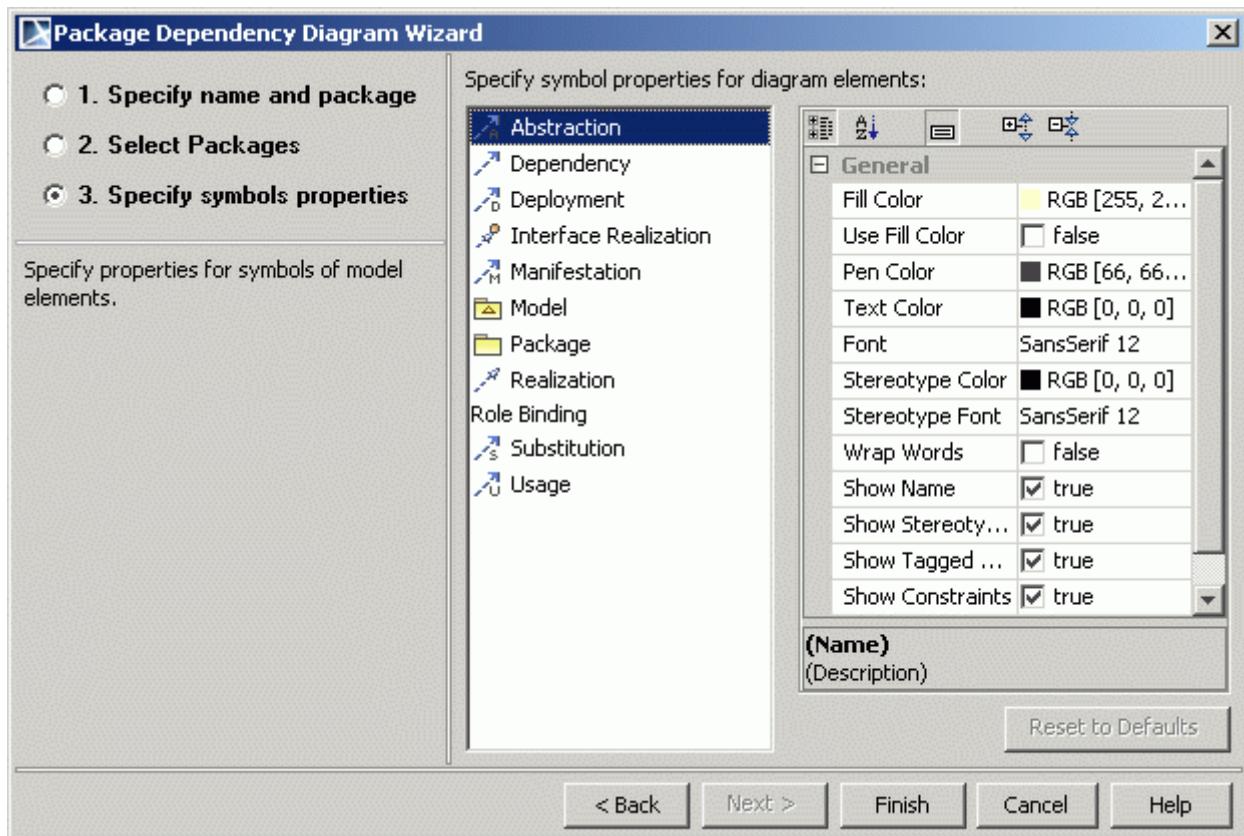


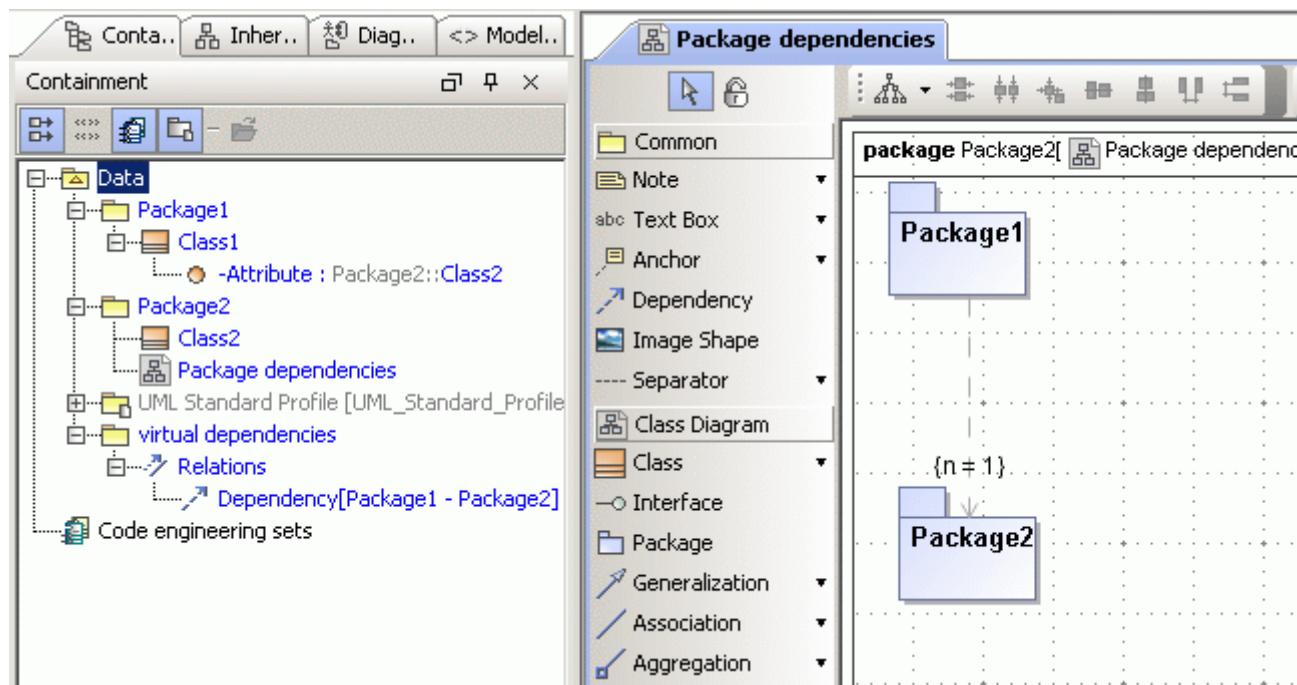
Figure 332 -- Package Dependency Diagram Wizard. Specify Symbols Properties

Select options for representing elements in the class diagram.

Displaying package dependencies

If a package has inner elements used by or dependent on other package elements, the Package Dependencies Wizard will analyze dependencies and create “virtual” relations between dependent packages.

For example:



The **Virtual dependencies** package containing dependency links is created after finishing the wizard. If there is more than one dependency between the package inner elements, then the tagged value number $\{n \neq 1\}$ on the virtual dependency will be changed according to the number of dependencies.

Package Overview Diagram Wizard

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The **Package Overview Diagram Wizard** allows the creation of a diagram for every package from the selected scope (reversed packages). The created diagram displays the content of the packages – inner packages with inner elements connected with available relations.

To start the **Package Overview Diagram Wizard**

- Select the **Package Overview Diagram Wizard** from the **Diagrams** menu, **Diagram Wizards** submenu.
- Open the package shortcut menu, select **Tools** and then **Package Overview Diagram Wizard**.

- From the **Analyze** menu, select the **Model Visualizer** command. The **Model Visualizer** dialog box opens. In the wizards list, select the **Package Overview Diagram Wizard**. Click the **Start** button.

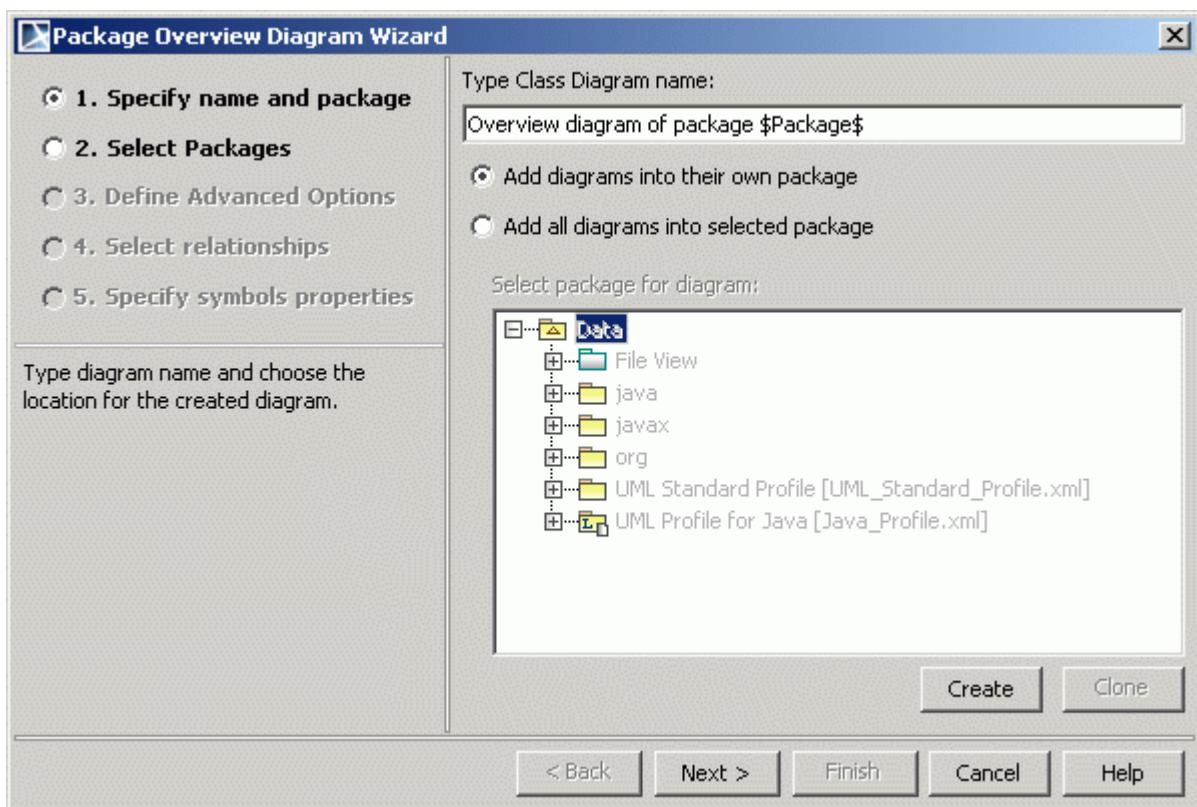


Figure 333 -- Package Dependency Diagram Wizard. Diagram name

Type a name for the newly created diagram in the **Type Class Diagram name** text box.

The **Add diagrams into their own package** option button - adds diagrams in the same package they are created.

The **Add all diagrams into selected package** - while creating new package overview diagrams, adds diagrams in the selected package. Select the package that will contain the created diagram. The hierarchy of UML model packages is displayed in the **Select package for diagram** list window. Select the package that will be

8 MODEL ANALYSIS

Model Visualizer

the parent for the newly created diagram from the Data tree or create a new package by clicking the **New** button.

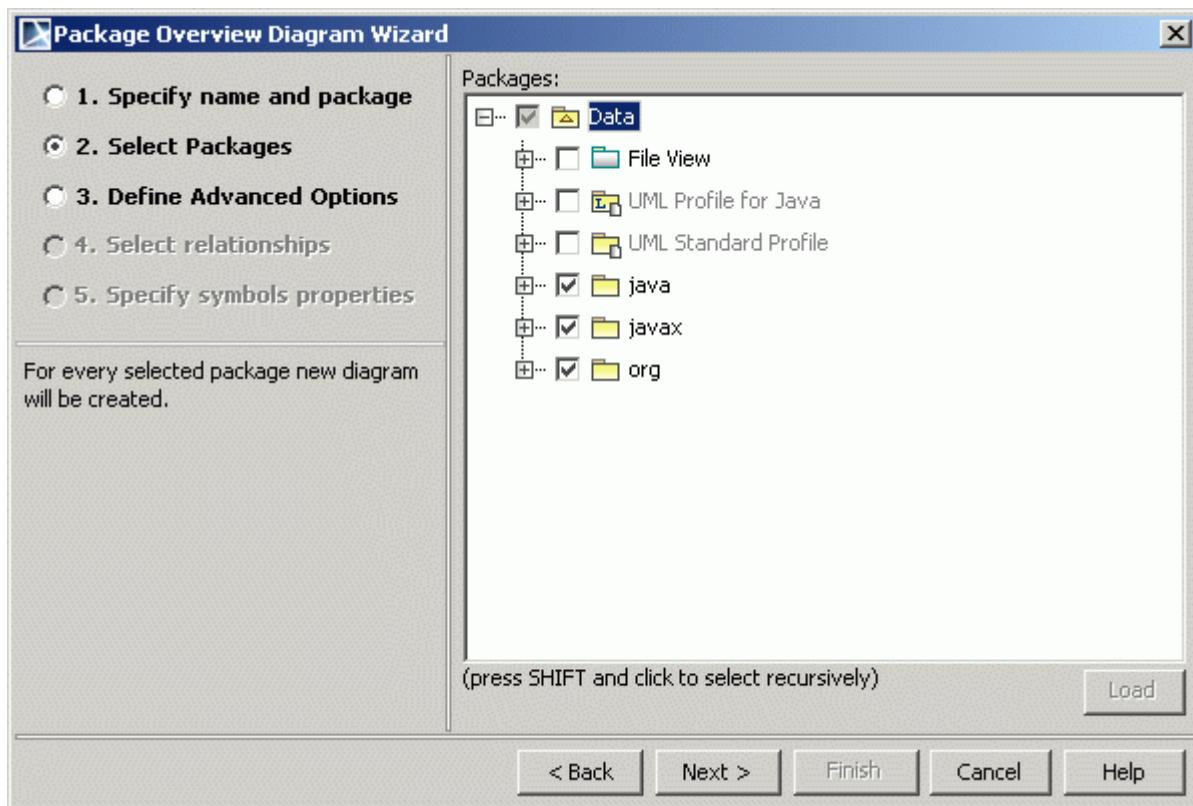


Figure 334 -- Package Overview Diagram Wizard. Select Package.

Select the packages, which will be represented in the new diagram. For every selection, a package diagram will be created.

If the selected package is read-only and the package for the diagrams is not specified, a warning will be displayed when the **Next** button is pressed.

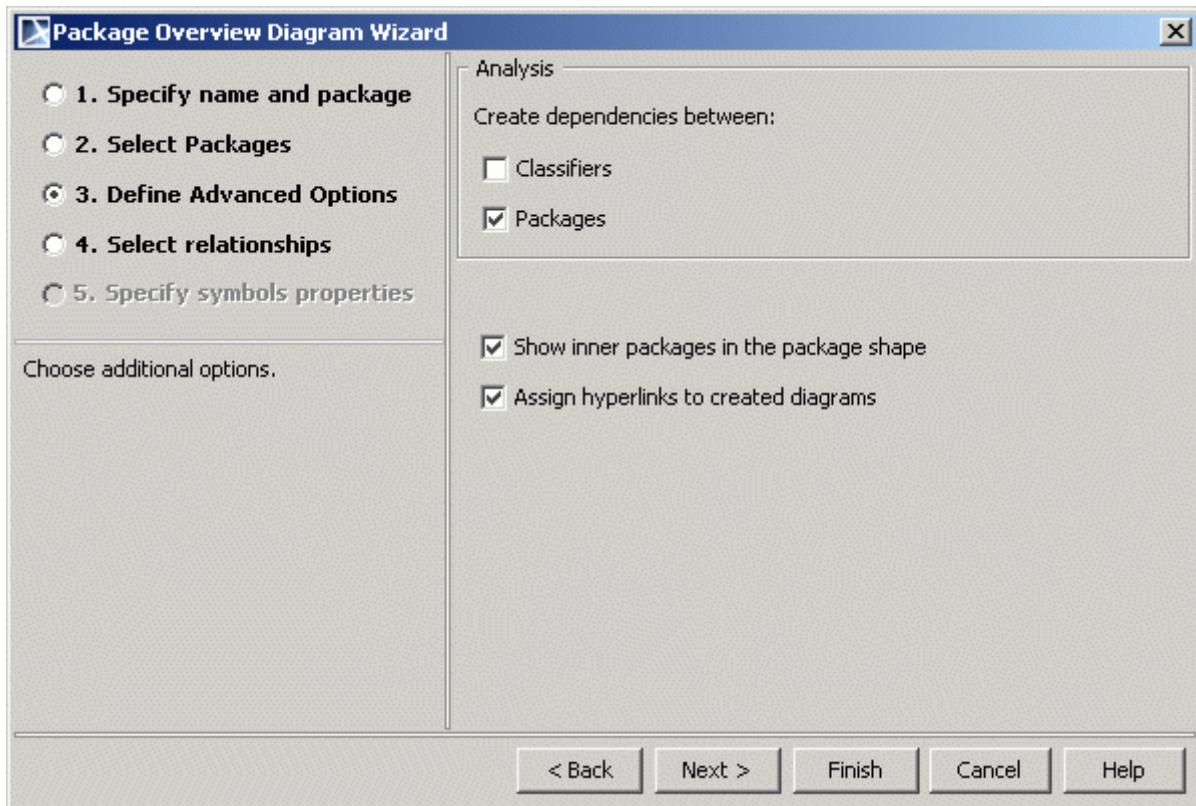


Figure 335 -- Package Overview Diagram Wizard. Define Advanced Options

Set the advanced properties for elements to be represented in the diagrams.

If you want to see the classifiers structure in the created class diagram, then select the create dependencies between **Classifiers** check box in the Analysis options group.

If create dependencies between **Packages** check box is selected, then only the package content class diagram will be created. Analyzes are performed of all inner elements, recursively by all criteria.

The **Show inner packages in the package shape** check box - displays one level of inner packages in every package shape, connected with dependencies.

8 MODEL ANALYSIS

Model Visualizer

The **Assign hyperlinks to created diagrams** check box - adds an active hyperlink to every package, referenced to the inner diagram of this package.

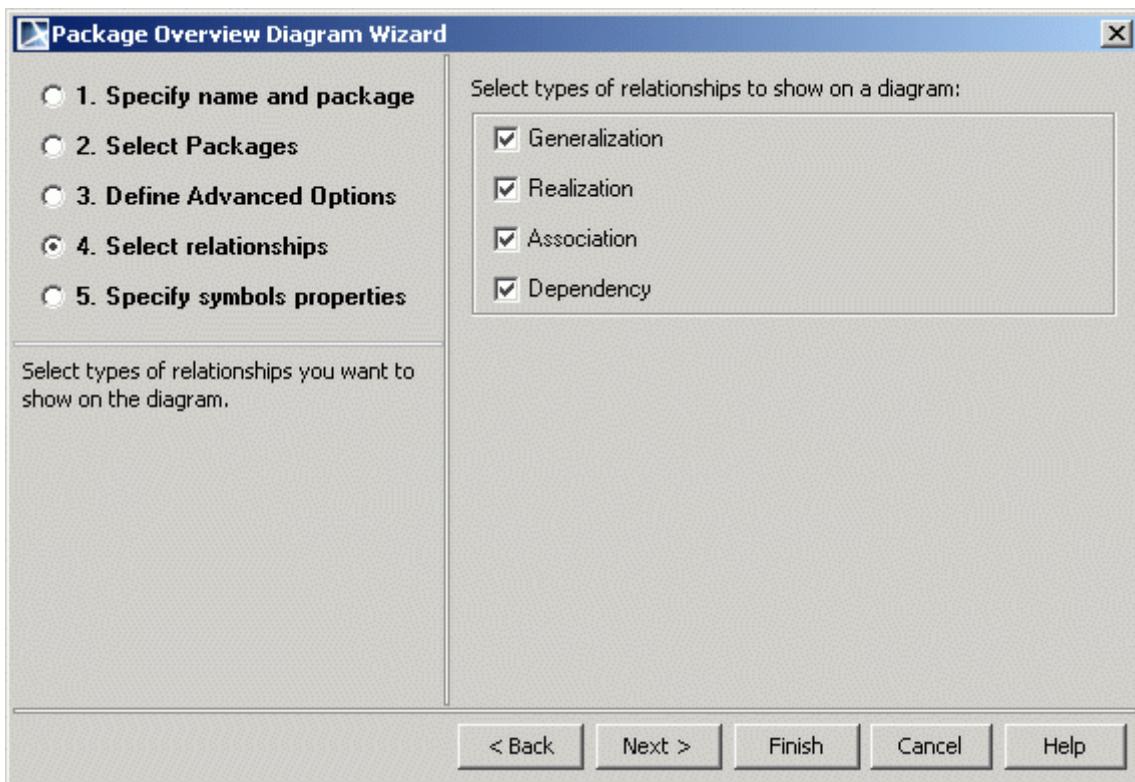


Figure 336 -- Package Overview Diagram Wizard. Select Relationships

Select the relationships you wish to include in the class diagram:

- **Generalization** – relationship between a general element and a more specific element (inheritance, extension).
- **Realization** – relationship between model elements where one of the elements implements the interface defined by the other model element.
- **Association** – semantic relationship between classes.

- **Dependency** – usage relationship between UML model elements.

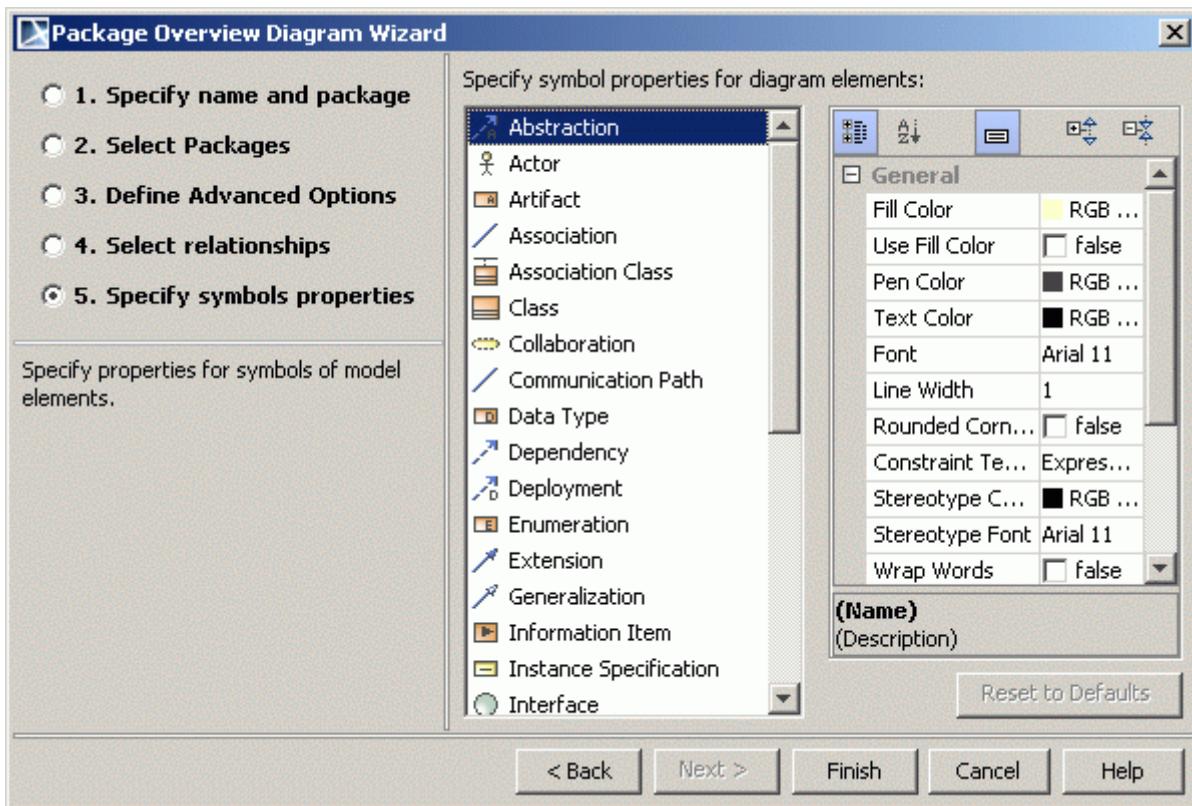


Figure 337 -- Package Overview Diagram Wizard. Specify symbols properties

Select options for representing elements in the diagram.

Hierarchy Diagram Wizard

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The **Hierarchy Diagram Wizard** collects the largest hierarchies and allows every of them to be displayed as separate diagrams or all in one diagram.

To start the **Hierarchy Diagram Wizard**

- From the **Diagrams** main menu, select the **Diagram Wizards** command and then **Hierarchy Diagram Wizard**.
- From the **Analyze** menu, select the **Model Visualizer** command. The **Model Visualizer** dialog box opens. From the wizards list, select the **Hierarchy Diagram Wizard**.

- From the model element shortcut menu, select **Tools** and then **Hierarchy Diagram Wizard**.

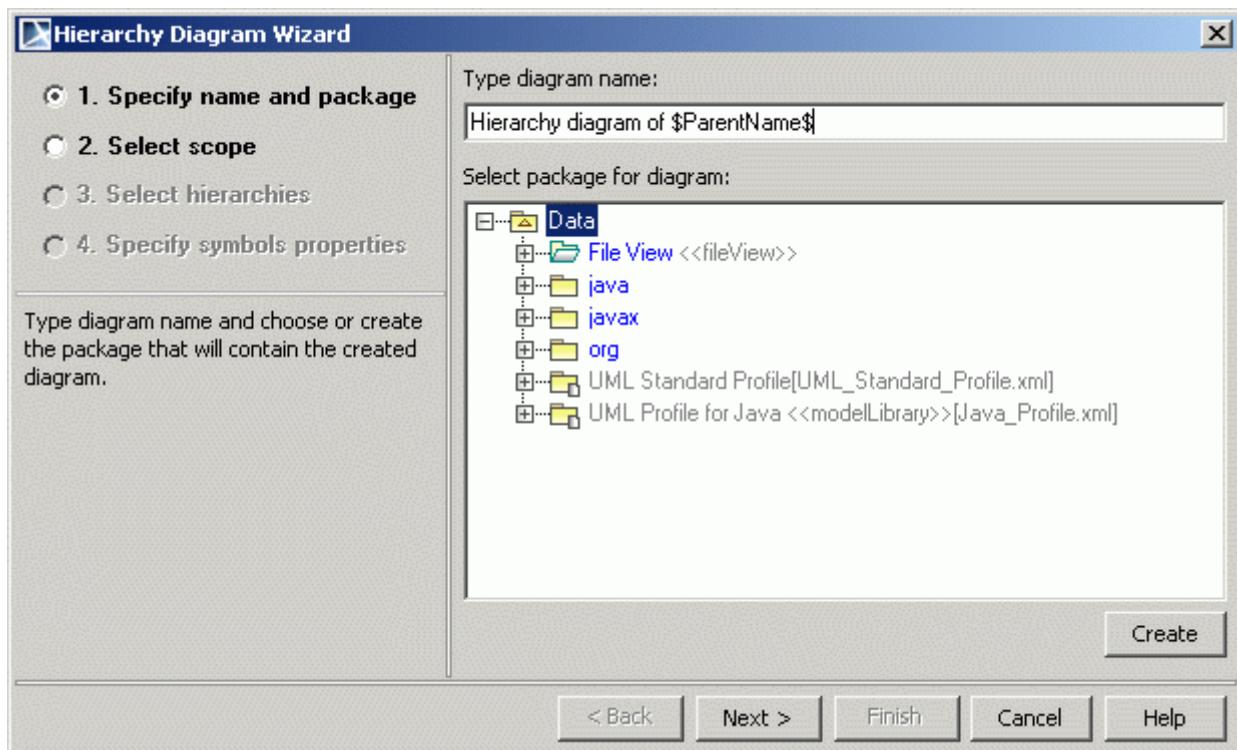


Figure 338 -- *Hierarchy Diagram Wizard. Specify Name and Package*

Type a name for the new diagram in the **Type diagram name** field.

Select the package that will contain the created diagram. The hierarchy of UML model packages is displayed in the **Select package for diagram** list window. Select the package that will be the parent for the newly created diagram from the Data tree or create a new package by clicking the **Create** button.

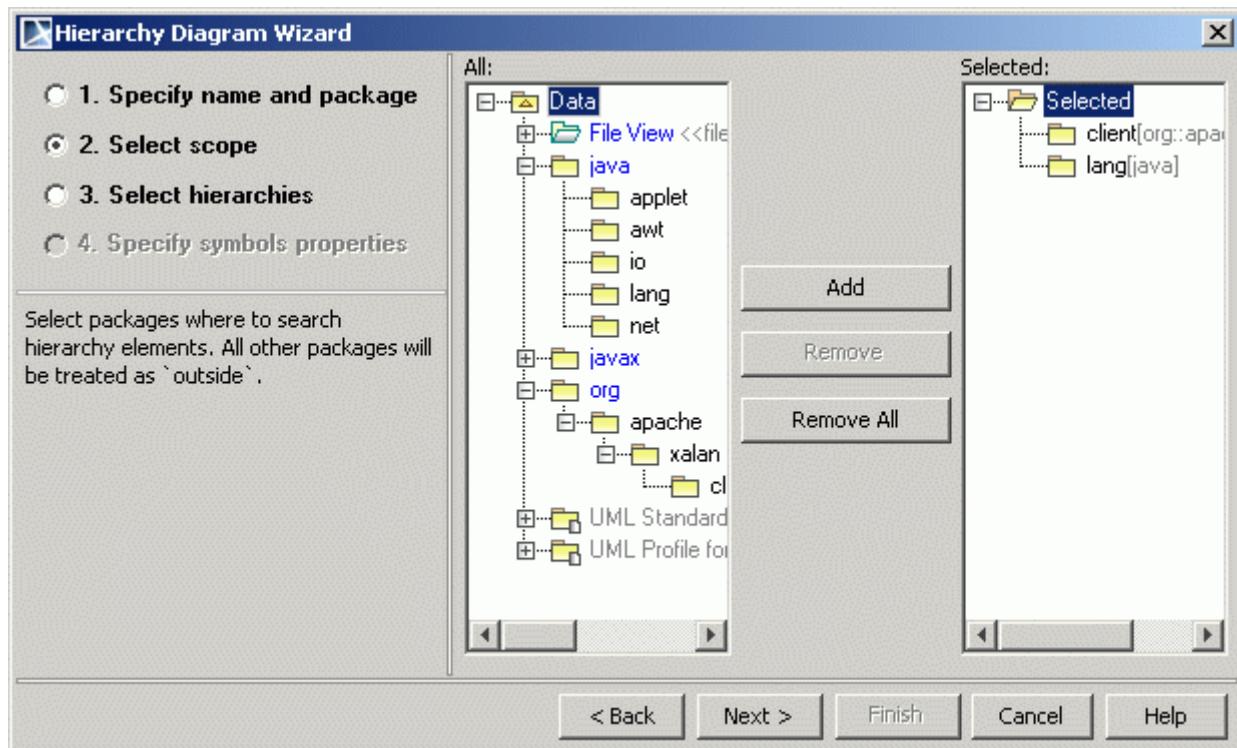


Figure 339 -- *Hierarchy Diagram Wizard. Select Scope*

Select packages from the **All** list to the **Selected** list.

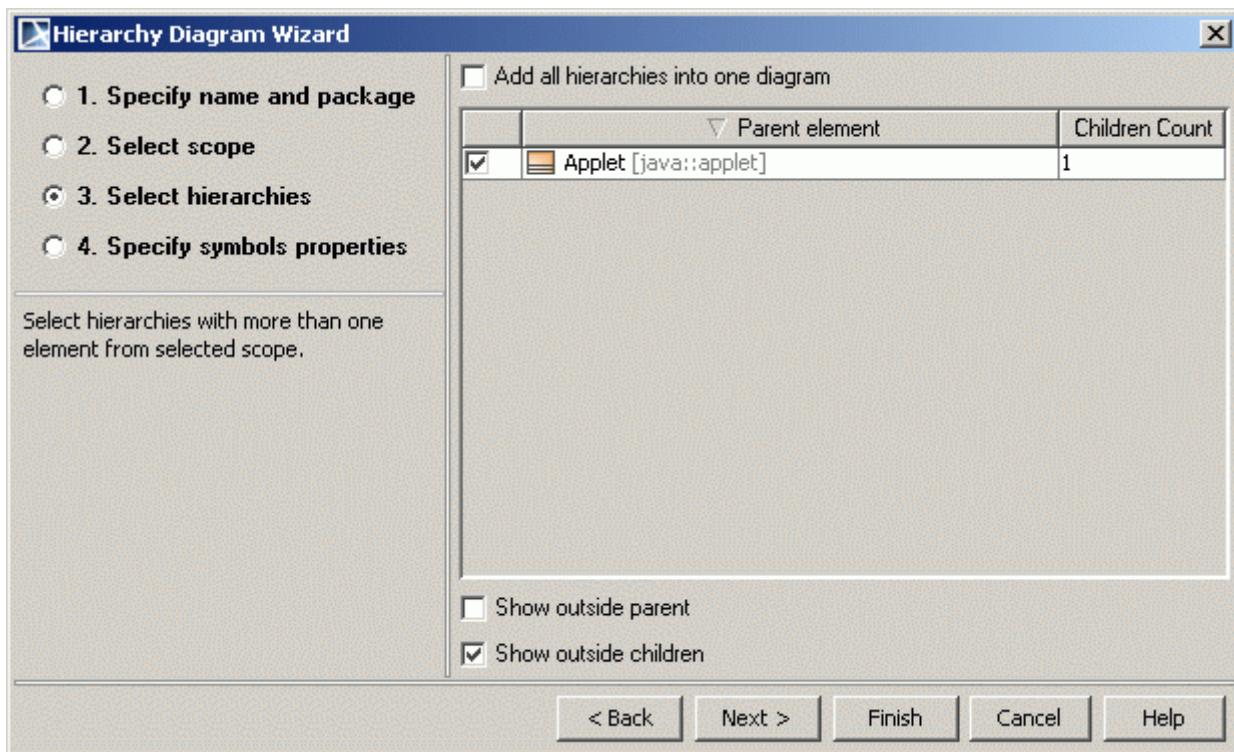


Figure 340 -- Hierarchy Diagram Wizard. Select Hierarchies.

The **Add all hierarchies into one diagram** check box creates only one diagram for all selected hierarchies. This option is enabled only if the selected hierarchies can be added into one diagram (the same diagram type).

All available hierarchies are listed in the **Parent Element** column. In the **Children Count** column, the number of model elements is presented.

The **Show outside parent** check box shows hierarchies, when derived packages are in the scope, but specializations are from outside the scope.

The **Show outside children** check box counts outside derived elements from the displayed hierarchies. Otherwise the hierarchy will not be fully displayed and the diagram may be not valid.

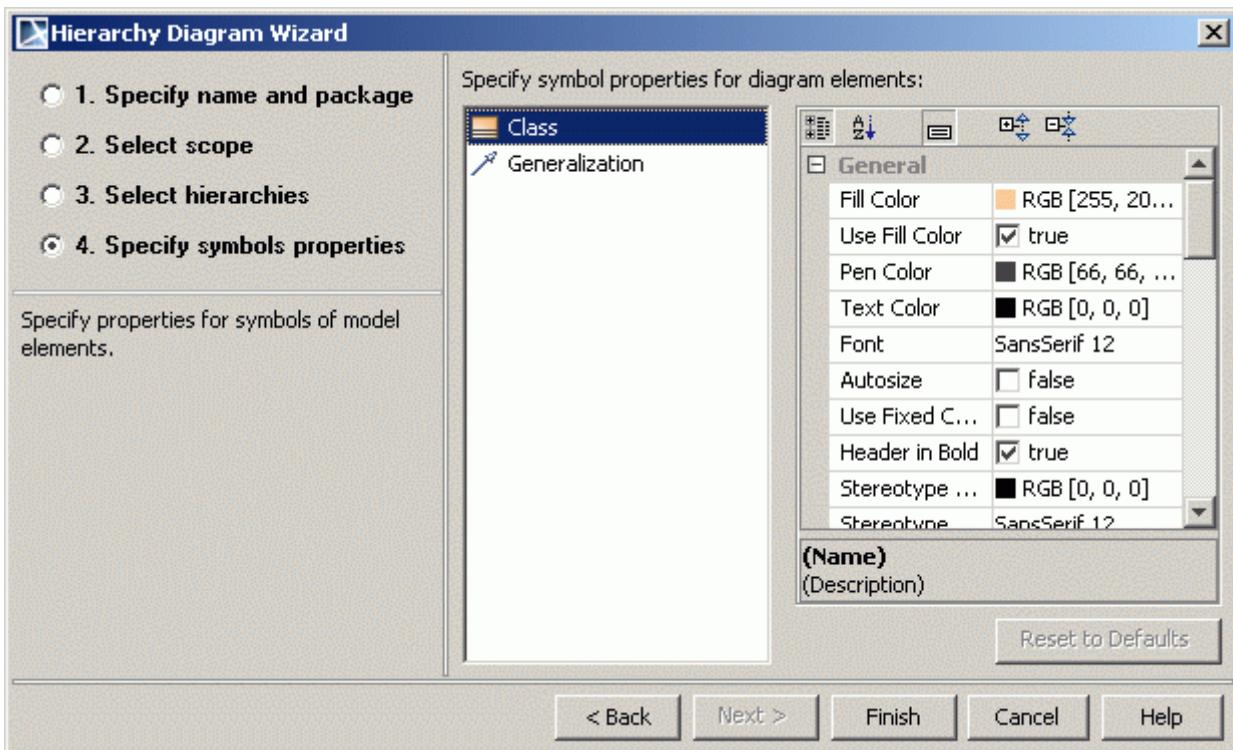


Figure 341 -- Hierarchy Diagram Wizard. Specify Symbols Properties

Select options for representing elements in the diagram.

Realization Diagram Wizard

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The **Realization Diagram Wizard** shows a table of the largest element groups that realize some interface.

To start the **Realization Diagram Wizard**

- From the **Diagrams** main menu, select the **Diagram Wizards** command and then **Realization Diagram Wizard**.
- From the **Analyze** menu, select the **Model Visualizer** command. The **Model Visualizer** dialog box opens. From the wizards list, select the **Realization Diagram Wizard**.

- From the model element shortcut menu, select **Tools** and then **Realization Diagram Wizard**.

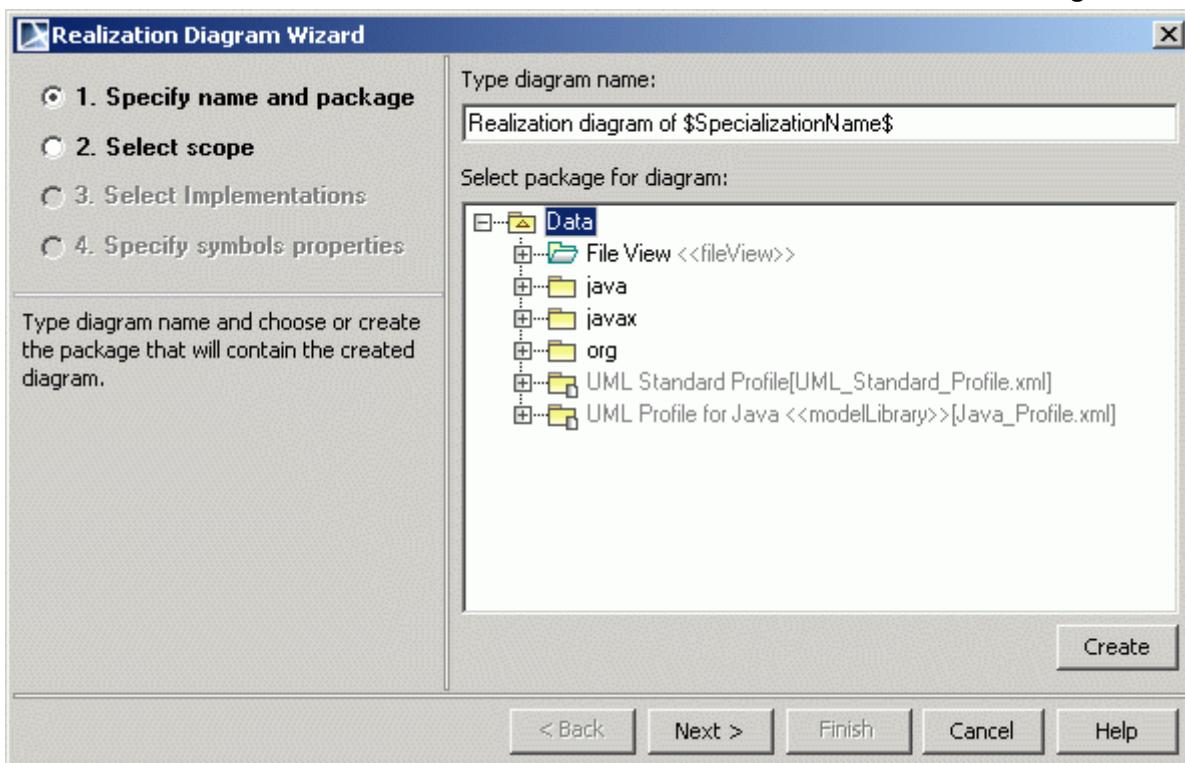


Figure 342 -- Realization Diagram Wizard. Specify Name and Package

Type a name for the new diagram in the **Type diagram name** field.

Select the package that will contain the created diagram. The hierarchy of UML model packages is displayed in the **Select package for diagram** list window. Select the package that will be the parent for the newly created diagram from the Data tree or create a new package by clicking the **Create** button.

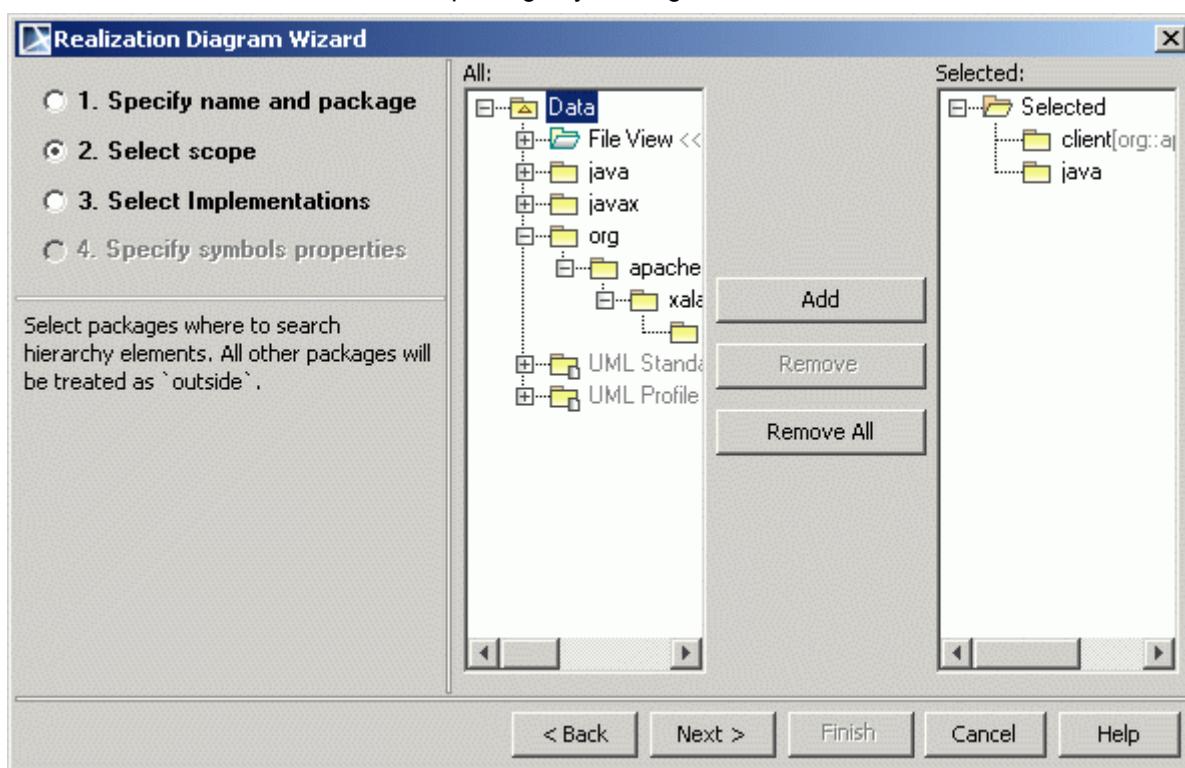


Figure 343 -- Realization Diagram Wizard. Select Scope

Select packages from the **All** list and add them to the **Selected** list to search for hierarchy elements.

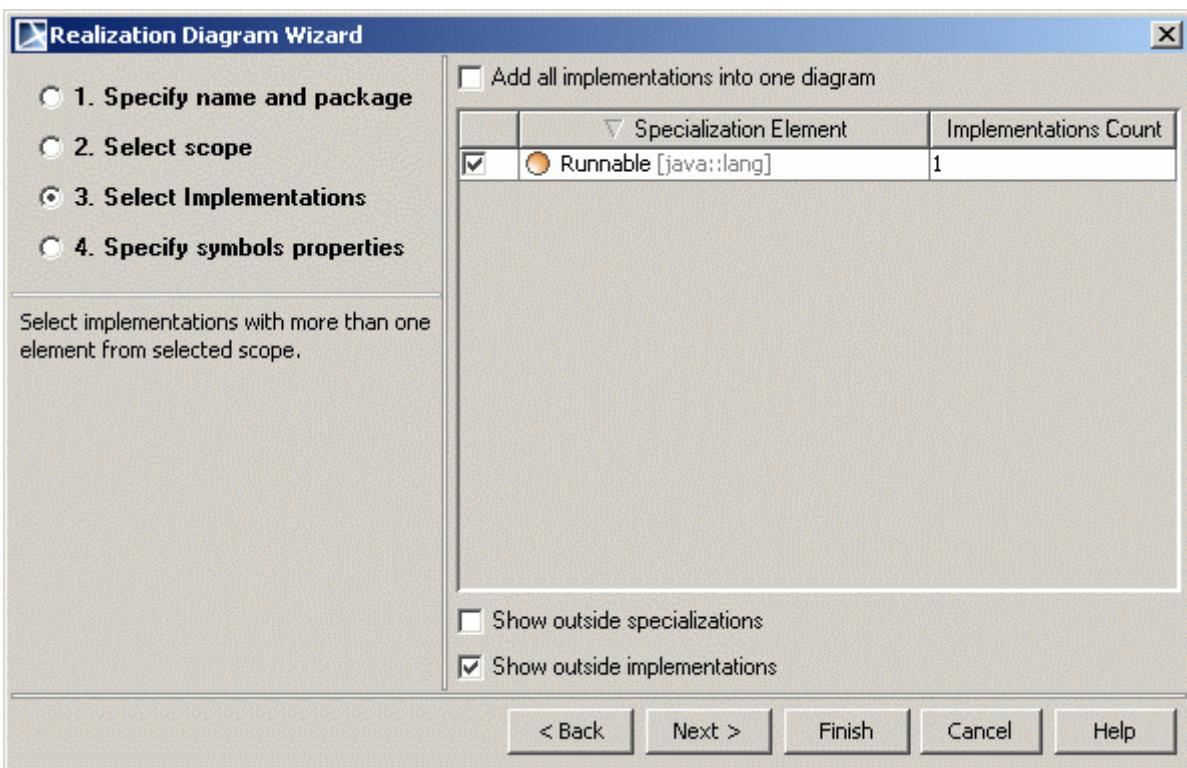


Figure 344 -- Realization Diagram Wizard. Select Implementations

The **Add all implementations into one diagram** check box creates only one diagram for all selected realizations. This option is enabled only if the selected realizations can be added into one diagram (the same diagram type).

All available implementations are listed in the **Specialization Element** column. In the **Implementations Count** column, the number of model elements is presented.

The **Show outside specializations** check box shows realizations, when derived interfaces are in the scope, but specializations are from outside the scope.

The **Show outside implementations** check box count outside derived elements from displayed realizations. Otherwise realization will not be fully displayed and the diagram cannot be valid.

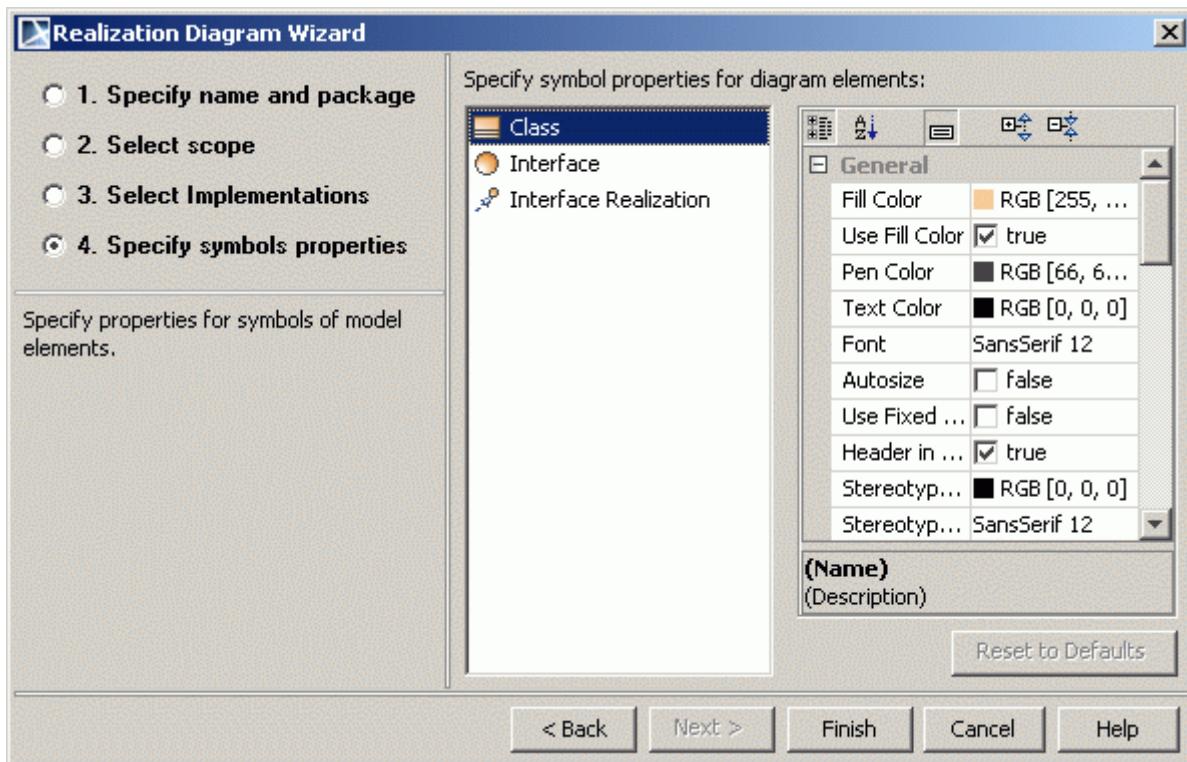


Figure 345 -- Realization Diagram Wizard. Specify Symbols Properties

Select options for representing elements in the diagram.

Activity Decomposition Hierarchy Wizard

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

Activity Decomposition Hierarchy Wizard allows converting activity into class and SysML Block Definition Diagram. This gives the capability to represent, analyze, and document activity hierarchies in the structure diagrams.

Diagram generation rules:

- Behaviors will be connected with contained object node types by compositions. The name of the object node that corresponds to the composition will be used as the end name of the association on the end towards the object node type.
- Pins are not included in the calculation.
- CallBehaviorActions that are not directly in the Activity, but are in the Structured Activity Nodes contained by the Activity, for example, are also included in the calculation.
- Activity will be connected by composition association with other behaviors that are called by CallBehaviorActions. The part end name must be the same as the name of a CallBehaviorAction in the composing activity. If the action has no name, then the end name is as same as that of the invoked activity.
- Hierarchical layout - Top to Bottom is used to arrange the generated diagram.
- If CallBehaviorAction calls the same activity, the composition to self will be displayed on the generated diagram.

Recursive structure analysis will be stopped after reaching the same behavior, which has already been analyzed. This requirement prevents an endless cycle.

- In such a case, the composition will be created in a previously analyzed activity in the diagram. The new behavior symbol will not be created.
- There will be as many compositions from one activity into another as different CallBehaviorActions call this activity.

To start the **Activity Decomposition Hierarchy Wizard**

- From the **Diagrams** main menu, select the **Diagram Wizards** command and then **Activity Decomposition Hierarchy Wizard**.
- From the **Analyze** menu, select the **Model Visualizer** command. The **Model Visualizer** dialog box opens. From the wizards list, select the **Activity Decomposition Hierarchy Wizard**.

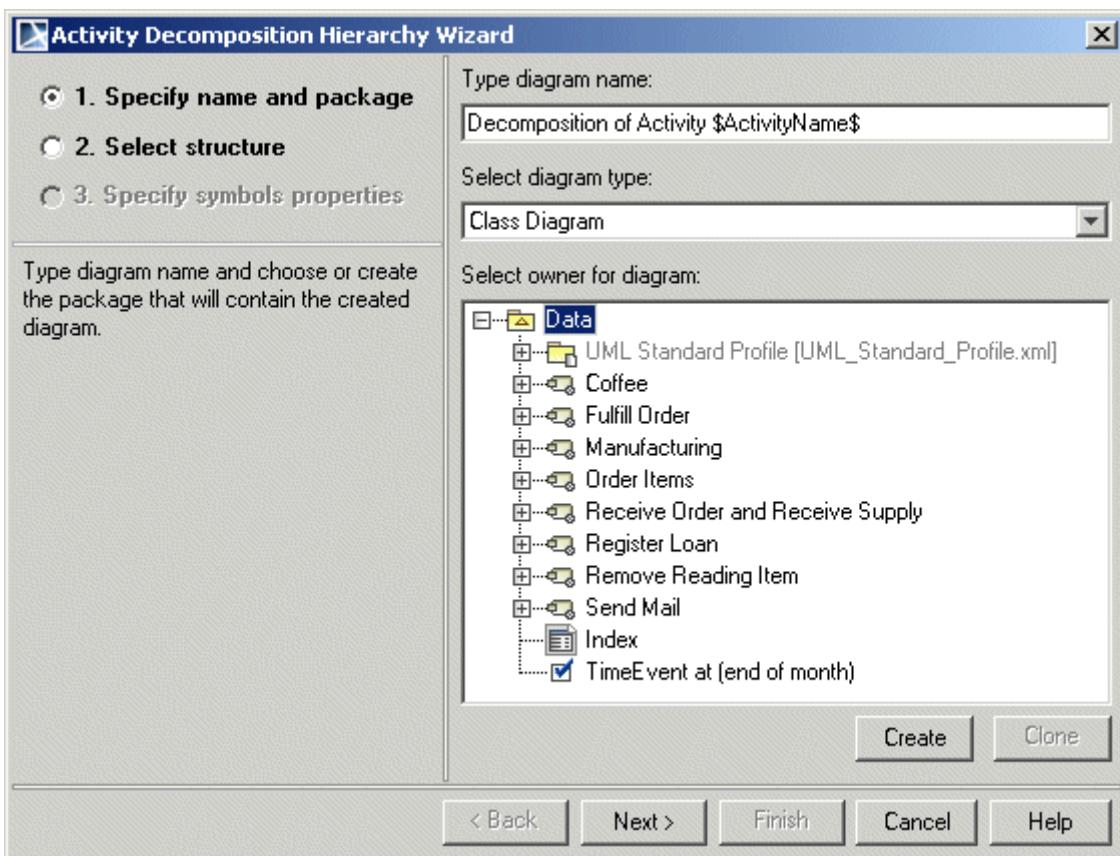


Figure 346 -- Activity Decomposition Hierarchy Wizard. Specify name and package tab

In the **Specify name and package** tab, type the diagram name, select the diagram type the activity will be converted and select or create a package that will contain the desired activity diagram.

NOTE

If you are using the SysML plugin, SysML Block Definition and Class diagrams are available as diagram types. For other domains this list depends on a plug-in of those domains.

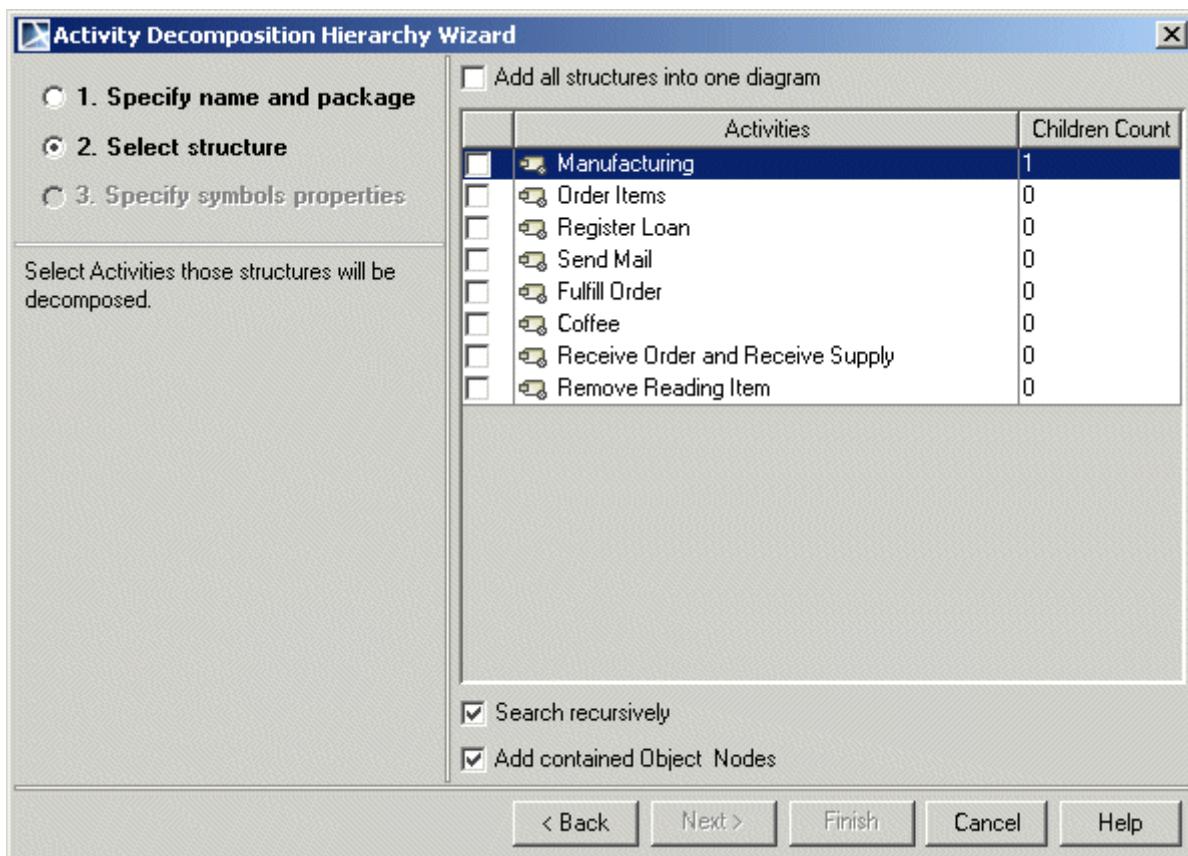


Figure 347 -- Activity Decomposition Hierarchy Wizard. Select structure tab

In the **Select structure** tab, select Activities structures that will be decomposed.

To add all activity structures into one diagram, select **Add all structures into one diagram** check box.

Select check boxes of the desired activity structures. The **Children Count** column shows the number of included behaviors (also owned object nodes if the **Add contained Object Nodes** check box is selected). The number also depends on the option **Search recursively**.

- The **Add contained Object Nodes** check box is selected by default. If selected, types of object nodes are displayed and connected to the composition with activities containing object nodes.
- The **Search recursively** check box is selected by default:
 - If not selected, the search will be conducted in only one level of the selected activity.
 - If selected, the search will be conducted in the selected activity and those activities that are invoked by CallBehaviorActions that are in the selected activity. This search is recursive.

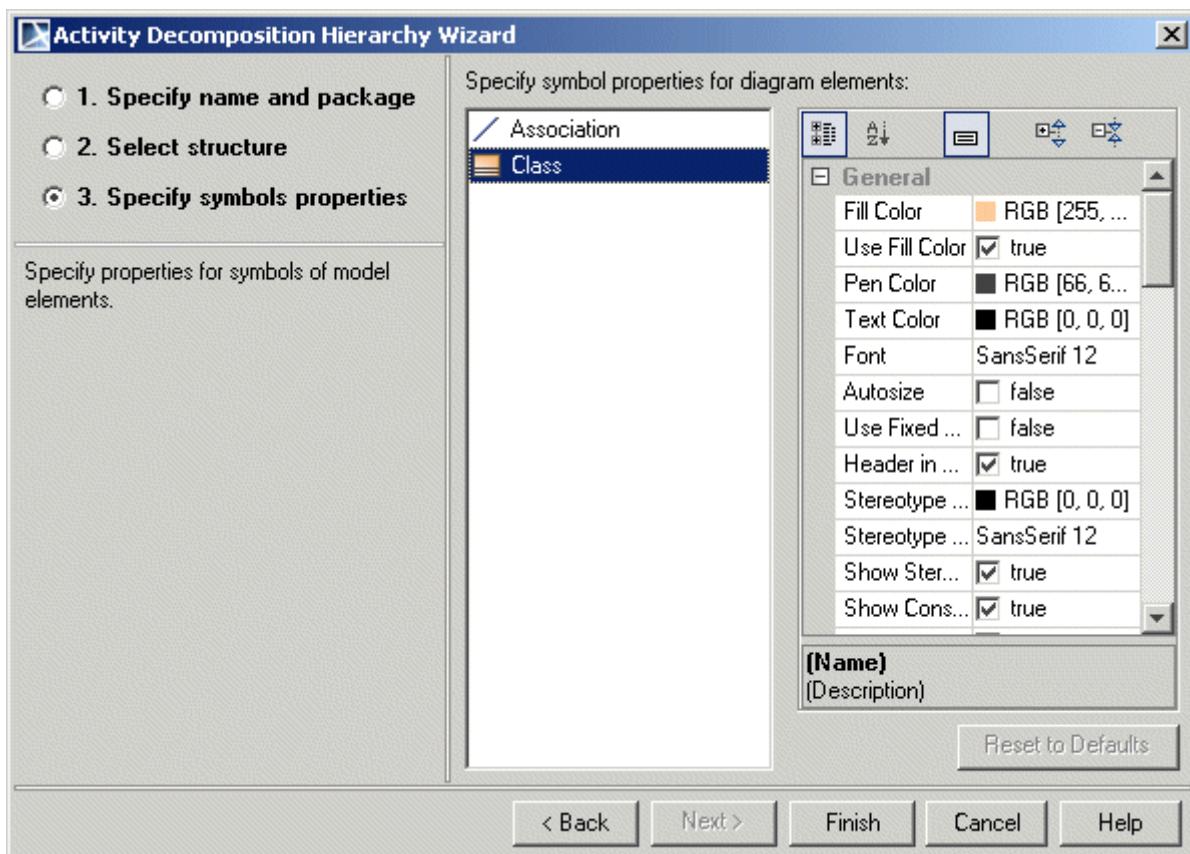


Figure 348 -- Activity Decomposition Hierarchy Wizard. Specify symbols properties tab

Specify properties for symbols for model elements.

Content Diagram Wizard

NOTE This feature is available in Standard, Professional, Architect, and Enterprise editions.

The **Content Diagram Wizard** generates content of diagrams that are used in the project.

To start the Content Diagram Wizard

Do one of the following:

- On the main menu select **Diagrams > Diagram Wizards > Content Diagram Wizard**.

- On the main menu select **Analyze > Model Visualizer**. The **Model Visualizer** dialog opens. Click **Content Diagram Wizard** in the wizards list and then click the **Start** button.

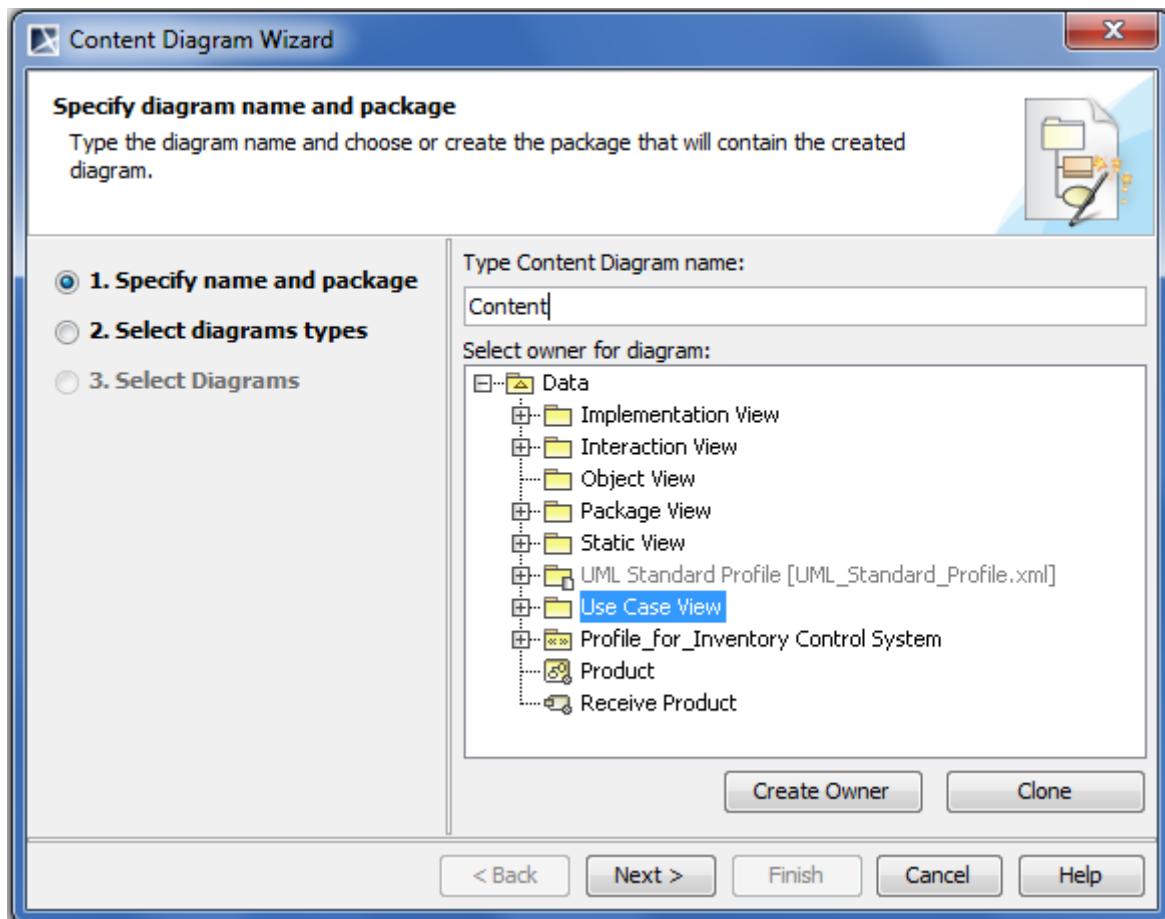


Figure 349 -- Content Diagram Wizard. Specifying name and package

Type a name for the new diagram in the **Type Content Diagram name** box.

Select the package that will contain the created diagram. The hierarchy of UML model packages is displayed in the **Select owner for diagram** tree. You can select the existing package or create a new one by clicking the **Create Owner** button, or copy the selected package with all its content by clicking the **Clone** button.

8 MODEL ANALYSIS

Model Visualizer

Click **Next** or choose the **Select diagram types** option for further steps.

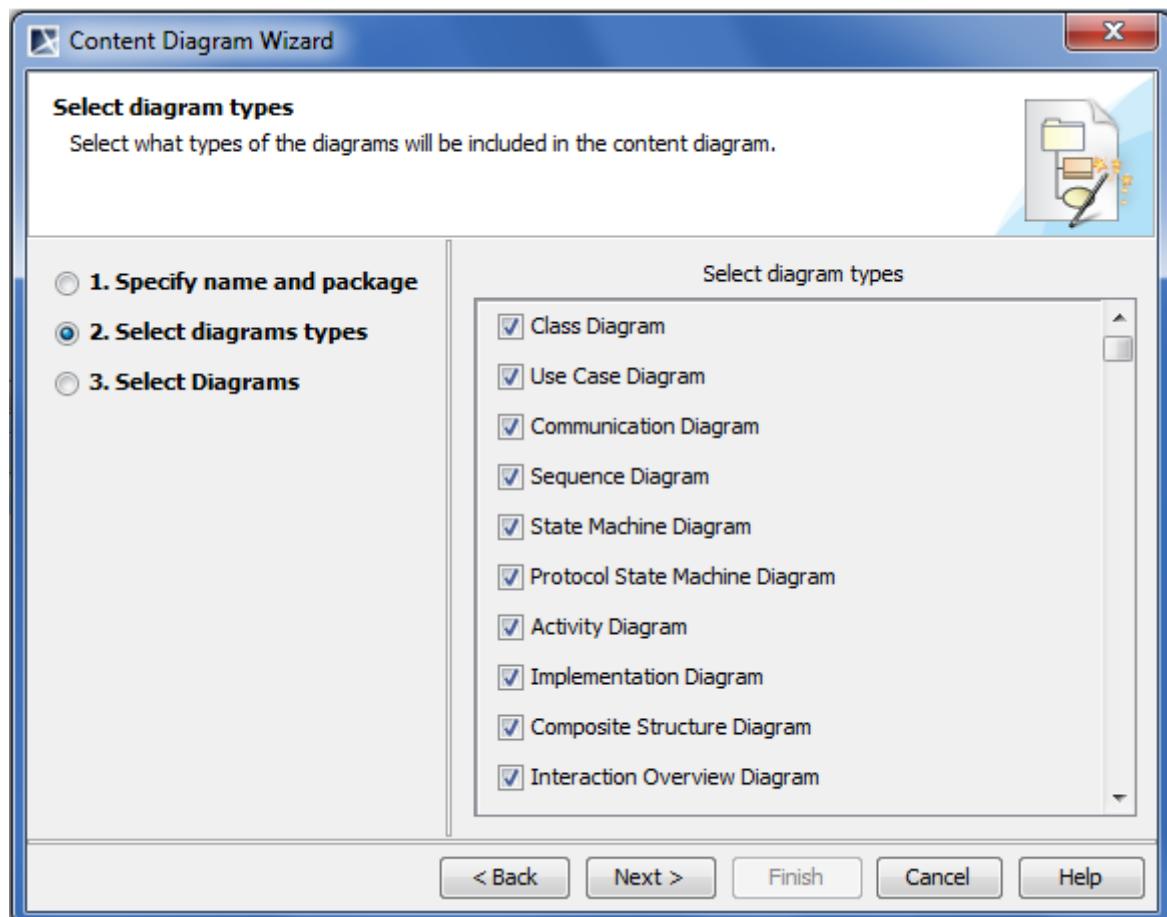


Figure 350 -- Content Diagram Wizard. Selecting diagram types

8 MODEL ANALYSIS

Model Visualizer

Select the types of diagrams to be included in the content diagram. Then click **Next** or choose the **Select Diagrams** option.

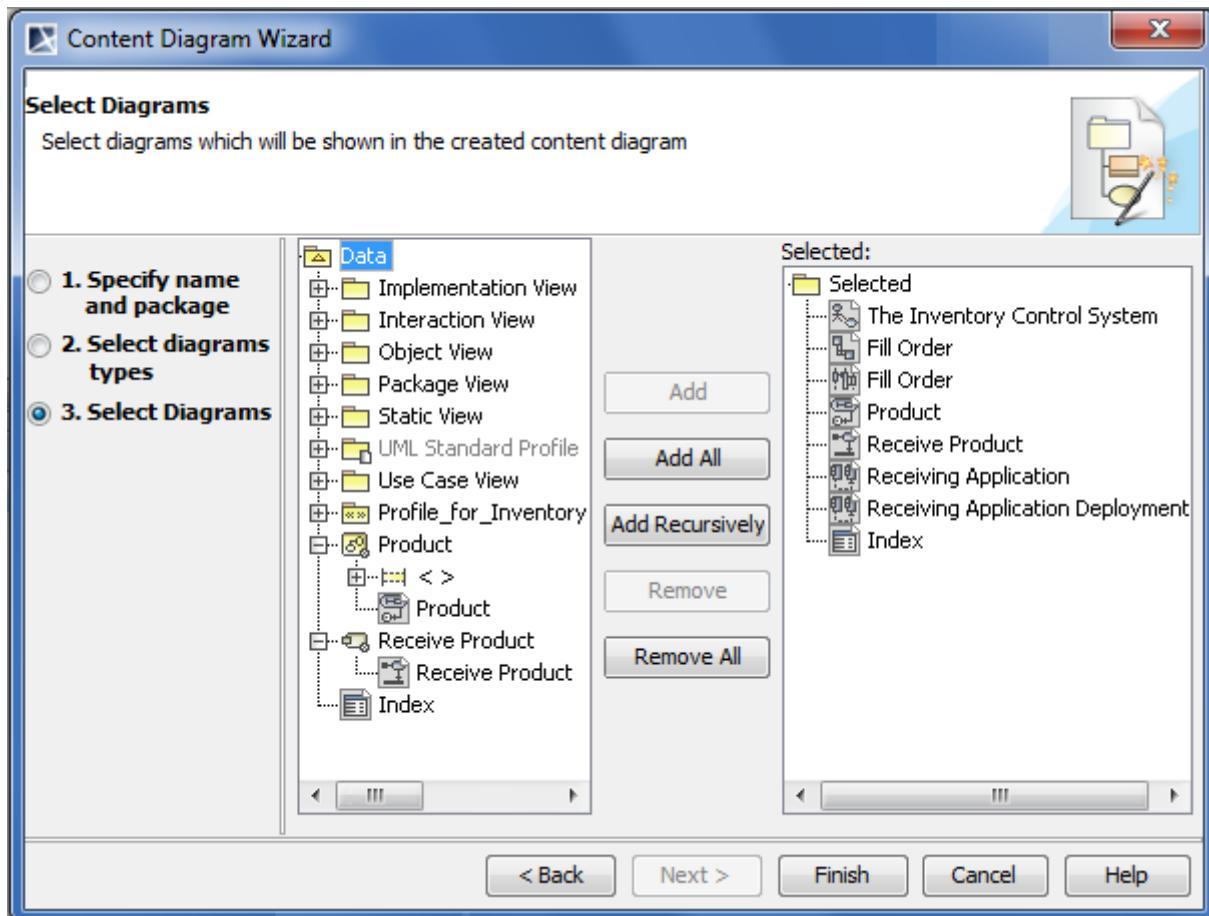


Figure 351 -- Content Diagram Wizard. Selecting diagrams

Select diagrams to be included in the content diagram and click **Finish**. The content diagram is now created.

Sequence Diagram from Java Source Wizard

NOTE This functionality is available in Enterprise edition only.

The **Sequence Diagram from Java Source Wizard** enables the visualization of Java method implementation with UML Sequence diagrams. Because UML Sequence diagrams cannot yet show Java code with 100% accuracy, MagicDraw provides a mechanism to generate a diagram that reflects essential Java method content.

To start the **Sequence Diagram from Java Source Wizard**

1. From the **Analyze** menu, select **Model Visualizer**. The **Model Visualizer** dialog box opens.
2. From the wizards list, select the **Sequence Diagram from Java Source Wizard**.
3. Click **Start**. The **Sequence Diagram from Java Source Wizard** opens.
 - From the **Diagrams** menu, select **Diagram Wizards**, then **Sequence Diagram from Java Source Wizard**.
 - Create a class with operation. Create a new code engineering Java set. Add a class to this set. In the Browser, Code engineering sets tree, select the RTMethod node. From its shortcut menu, select **Reverse Implementation**. The **Sequence Diagram from Java Source Wizard** opens (method is selected in the wizard).

- Create a class with operation. Select the operation in the Browser. From the operation shortcut menu, select the **Reverse Implementation** command. The **Sequence Diagram from Java Source Wizard** opens (method is selected in the wizard).
- In the Sequence diagram, create *call* message. Select it on the diagram pane. From the message shortcut menu, select **Reverse Implementation**. The **Sequence Diagram from Java Source Wizard** opens (method is selected in the wizard).

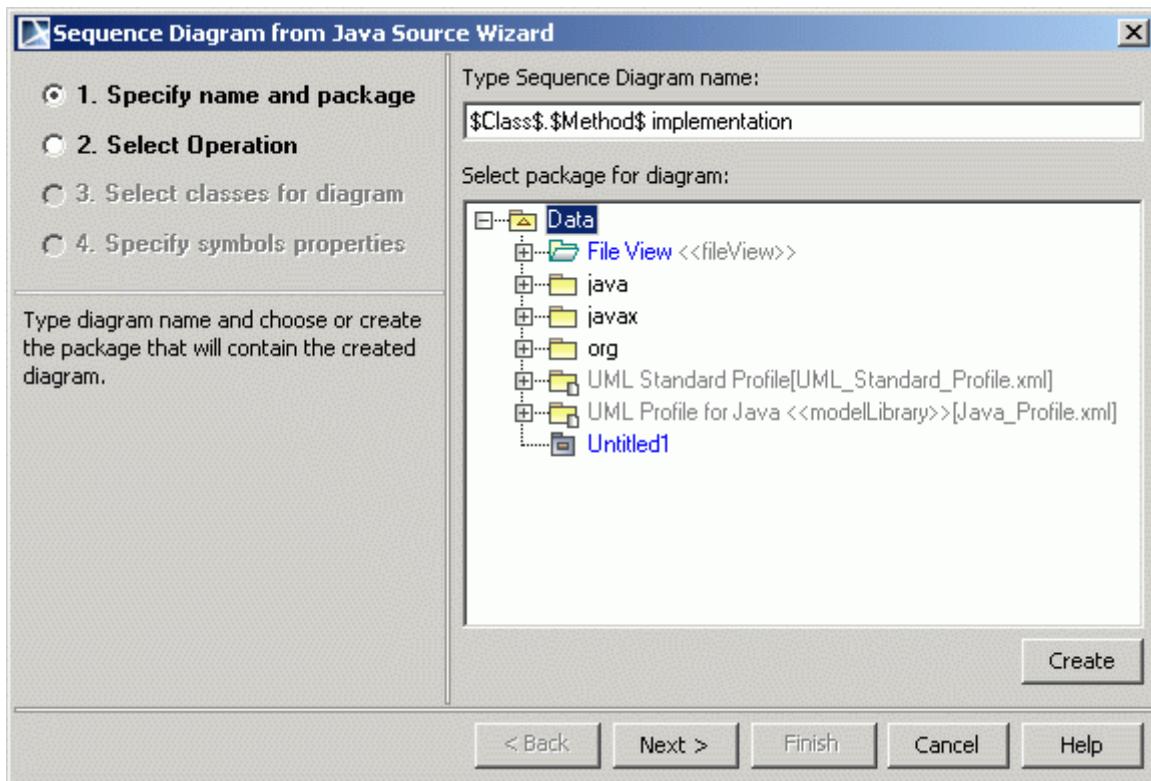


Figure 352 -- Sequence Diagram from Java Source Wizard. Specify Name and Package

In the **Type diagram name** text box, type a name for the new sequence diagram. By default, the name of the class and method is assigned to the sequence diagram.

Select the package that will contain the created diagram. The hierarchy of UML model packages is displayed in the **Select package for diagram** list window. From the **Data** tree, select the package that will be the parent for the newly created diagram, or create a new package by clicking **New**.

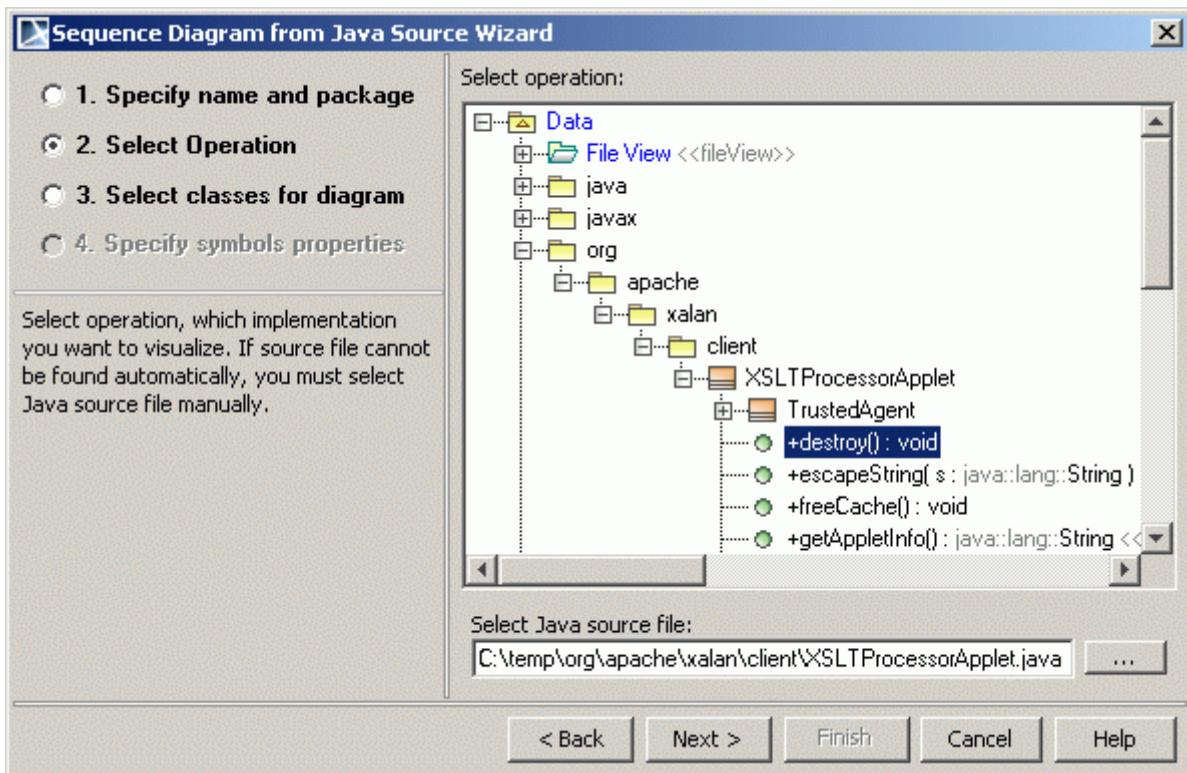


Figure 353 -- Sequence Diagram from Java Source Wizard. Select Operations.

Select the operation, which will be displayed in the sequence diagram. When you start the wizard from either the operation shortcut menu or the Sequence diagram, both the operation and Java source file are selected by default.

In the **Select Java source file field**, specify the Java source file (if this file cannot be found automatically).

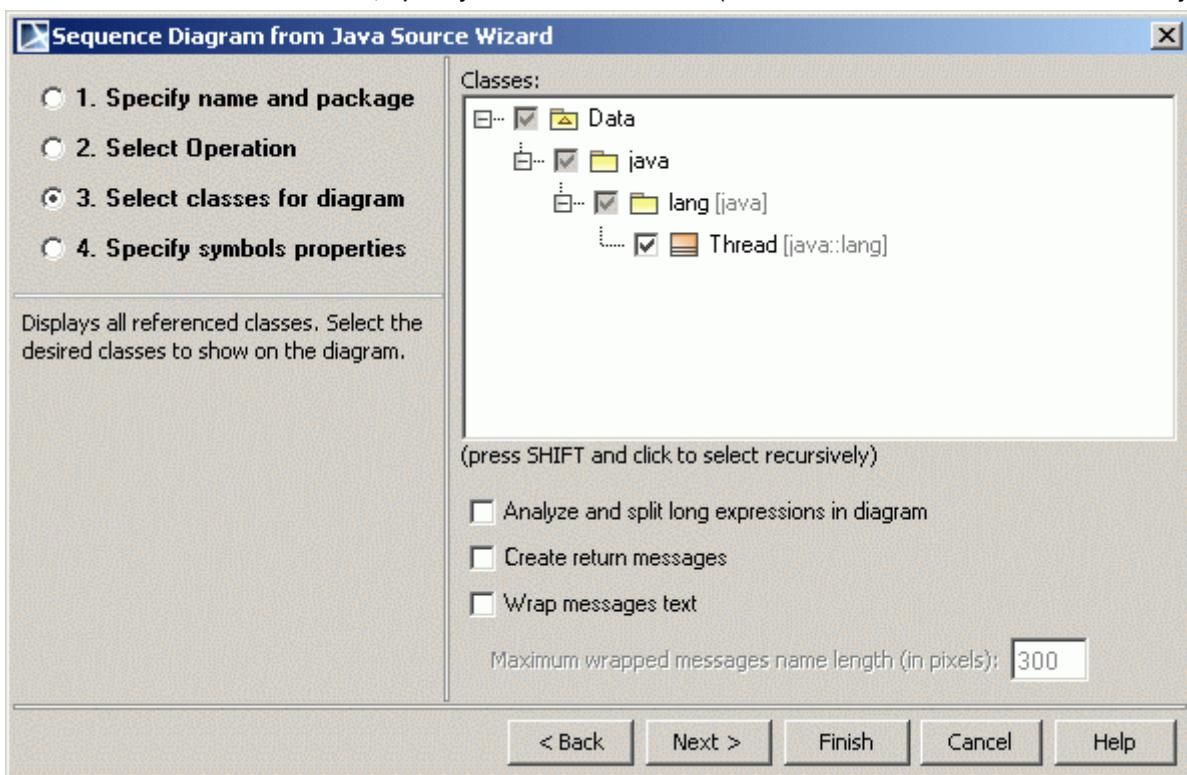


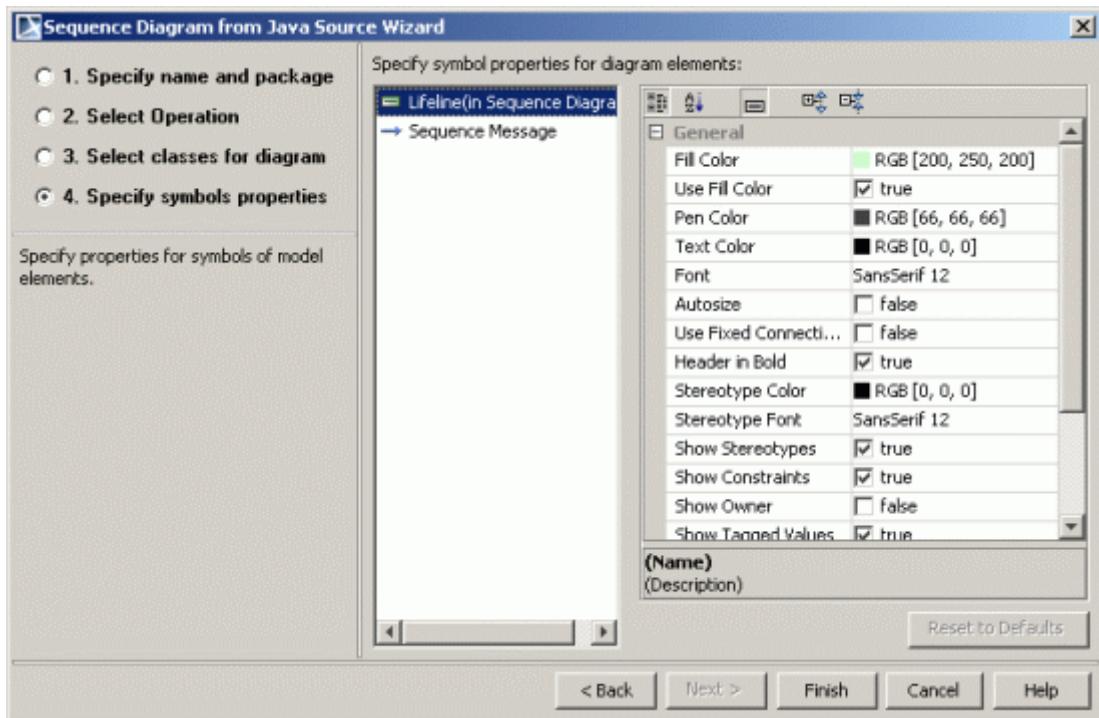
Figure 354 -- Sequence Diagram from Java Source Wizard. Select Classes for Diagram.

Select Classes for Diagram includes the following check boxes:

Analyze and split long expressions in diagram - if the expression contains calls and cannot be displayed as a call message, every call is displayed as a separate call message with temporary variable initialization. In the final expression message, these calls are replaced with appropriate temporary variable names.

Create return message - displays a return message for every call message.

Wrap message text - specifies the maximum message text length in pixels for wrapping longer messages.



Using **Specify Symbol Properties**, you can select options for the elements to be represented in your sequence diagram.

NOTE

You can extend the created sequence diagram by method: Select the method (message), which you want to be displayed with additional details (this can include referenced classes and other defining methods). Then, from the message shortcut menu, select **Reverse Implementation**. The **Sequence Diagram from Java Source Wizard** opens.

9 UML DIAGRAMS

In software development, the diagram is the equivalent of a blueprint. To meet the various needs of many parties, we often need several different “blueprints” of the same system. Furthermore, every system is described by many different aspects. For example:

- Functional (static structure and dynamic interactions)
- Nonfunctional (timing requirements, reliability, and deployment)
- Organizational (work organization and mapping to code modules)

MagicDraw supports the following diagrams that are defined in UML 2:

| UML Diagrams | MagicDraw UML Diagrams |
|--------------------------------|-------------------------------------|
| Use Case Diagram | Use Case Diagram ^a |
| Class Diagram | Class Diagram ^a |
| Object Diagram | |
| Composite Structure Diagram | Composite Structure Diagram |
| State Machine Diagram | State Machine Diagram |
| Protocol State Machine Diagram | Protocol State Machine Diagram |
| Activity Diagram | Activity Diagram |
| Interaction Overview Diagram | Interaction Overview Diagram |
| Sequence Diagram | Sequence Diagram |
| Communication Diagram | Communication Diagram |
| Component Diagram | |
| Deployment Diagram | Implementation Diagram ^a |

a. The Package Diagram is provided for use with Use Case, Class, and Implementation Diagrams

Architectural Views

UML defines 13 diagrams that describe 4+1 architectural views:

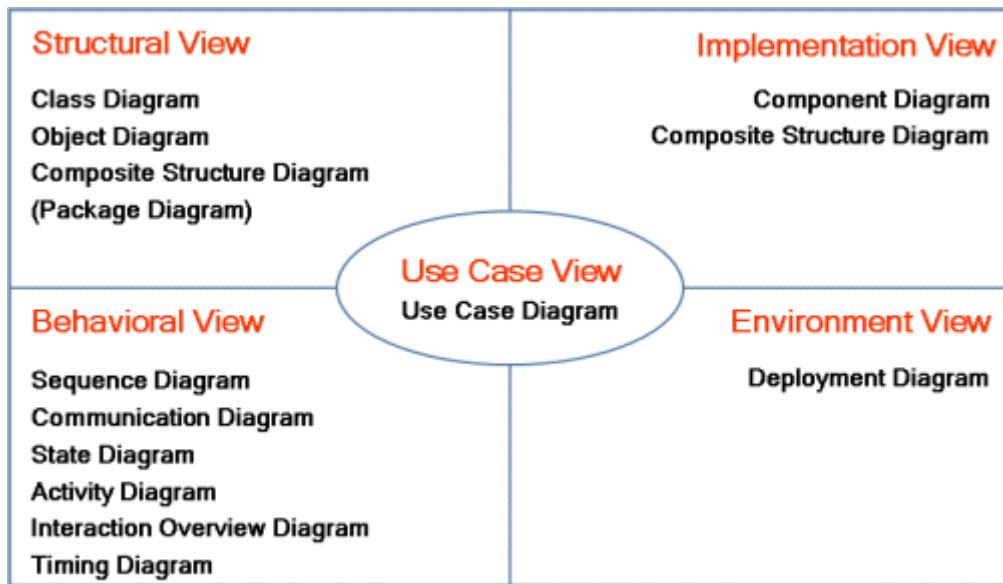


Figure 355 -- Architectural Views

Several kinds of diagrams provide a visual notation for the concepts in each view.

Use Case View

The use case view represents the functionality and behavior of a system or subsystem as it is perceived by external users. This view is targeted mainly at customers, designers, developers, and testers.

The use case view usually is presented as a number of use cases and actors in Use Case diagrams. Occasionally it is used in Activity and Sequence diagrams.

The use case view is central because the contents drive the development of the other views. It is also used for project planning. Every single use case unit is deemed as a manageable unit during the project execution.

Structural View

The structural view represents structural elements for implementing a solution for defined requirements. It identifies all of the business entities and how these entities are related to each other. Usually entities are represented as classifiers and their instances in class and object diagrams in multiple abstraction levels. System decomposition to different layers can be displayed using Package diagrams. A Composite structure diagram can be used to represent the classifier inner structure. The system structural view artifacts are created by software architects and represent the system implementation design solutions.

Behavioral View

The dynamic behavior of the system is displayed on the Interaction (sequence and collaboration), State, Activity, Interaction overview, and Timing diagrams. It focuses mainly on the interactions that occur between objects inside a system, activities and work performed by the various parts of a system, and state changes within a particular object or collaboration. Rather than defining the participants of the system, it defines how particular use cases are executed, which provides value for the external user. The dynamic view is concerned about what is happening inside the system and how those actions impact other participants.

Implementation view

The implementation view describes the implementation artifacts of logical subsystems defined in the structural view. It may include the intermediate artifacts used in a system construction (code files, libraries, data files, etc.) This view defines dependencies between the implementation components and their connections by the required and provided interfaces. Components and their relationships are displayed on the Component diagram. Inner parts of the component can be represented with the Composite structure diagrams. The implementation view helps analyze system parts and their dependencies in a higher component level.

Environment view

The environment view represents the physical arrangement of a system, such as computers and devices (nodes) and how they are connected to each other. In contrast to the component view, the deployment view is concerned with the physical structure of the system and the location of the software modules (components) manifested by artifacts within the system.

The environment view is displayed on the deployment diagram that is called the implementation diagram in MagicDraw™ UML.

Class Diagram

A class diagram is a graphic representation of the static structural model. It shows classes and interfaces, along with their internal structure and relationships. The classes represent types of objects that are handled in a system. A class diagram does not show temporal information, it describes only the classification. The instances of those types (objects) are instantiated only on the runtime and are represented by an object and the interaction diagrams.

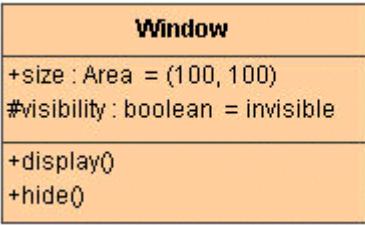
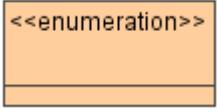
The classes can be related to each other in a number of ways: associated (connected to each other), dependent (one class depends/uses another class), specialized (one class is a subtype of another class), or packaged (grouped together as a unit – package). A class diagram does not express anything specific about the relationships of a given object, but it does abstractly describe the potential relationships of one object with other objects.

A system typically has a number of class diagrams – not all classes are inserted into a single class diagram. A class may have multiple levels of meaning and participate in several class diagrams.

A class diagram is the logical map of an existing or future source code.

The classes can be grouped into packages. The packages can be nested within other packages. A package, as an entity, may have all the relationships that can be drawn for a class. Those relationships are derived from the classes or packages that are nested within two particular packages (i.e., the relationship between packages reflects a set of relationships between classes placed in those packages).

Class diagram elements

| Model element | Button (hot key) | Notation |
|--|------------------|---|
| Class A descriptor for a set of objects with similar structures, behaviors, and relationships. | (C) |  <p>class name</p> <p>+size : Area = (100, 100) #visibility : boolean = invisible</p> <p>+display() +hide()</p> <p>operations</p> |
| Enumeration A user-defined data type whose instances are a set of user-specified named enumeration literals. The literals have a relative order but no algebra is defined on them. | (K) |  |
| Class by Pattern | (SHIFT+P) | |
| Signal | | |
| Data Type | | |
| Primitive Type | | |
| Interface The description of a visible behavior of a class, a component or a package. Attributes and operations inside the Interface can be suppressed. | (I) |   |

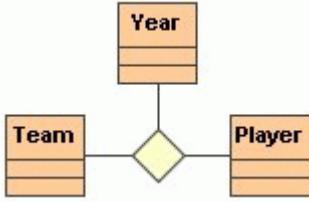
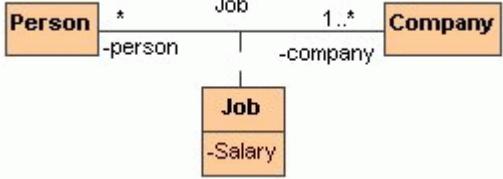
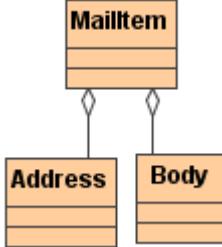
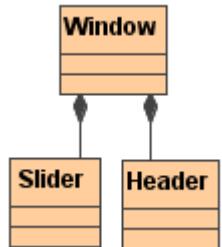
9 UML DIAGRAMS

Class Diagram

| Model element | Button (hot key) | Notation |
|---|------------------|----------|
| Package A group of classes and other model elements. | (P) | |
| Model A model is an abstraction of a physical system from a particular point of view. A model contains a hierarchy of packages/sub-systems and other model elements that describe the system. | (M) | |
| Generalization A relationship between a more general and a more specific element. Note: Choose a different Generalization direction from the toolbar to draw a line with an opposite arrow end. | (G) | |
| Association A connection among classes, which also means a connection among objects of those classes. | (S) | |
| Directed Association | | |
| Non-navigable Association | | |

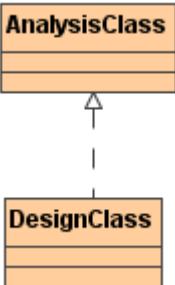
9 UML DIAGRAMS

Class Diagram

| Model element | Button (hot key) | Notation |
|---|---|--|
| N-ary association An association among two or more classes (a single class may appear more than once). | (O)  |  |
| Association Class The Association Class is a declaration of a semantic relationship between Classifiers. The Association Class, which has a set of features of its own, is both an Association and a Class. |  |  |
| Aggregation An aggregation is an association that represents a whole-part relationship. | (A)  |  |
| Directed Aggregation |  | |
| Composition A composition is a form of aggregation with a stronger ownership and coincident lifetime of part with the whole. | (F)  |  |
| Directed Composition |  | |

9 UML DIAGRAMS

Class Diagram

| Model element | Button (hot key) | Notation |
|---|--|--|
| Interface Realization A relationship is usually used between an interface and an implementation class. Note: Choose a different Interface Realization direction from the toolbar to draw a line with an opposite arrow end. | (R)  |  |
| Realization A relationship between a specification and its implementation. | (E)  |  |
| Substitution A substitution is a relationship between two classifiers. |  | |
| Usage A usage is a relationship in which one element requires another element (or set of elements) for its full implementation or operation. Note: Choose a different Usage direction from the toolbar to draw a line with an opposite arrow end. |  | |
| Abstraction An abstraction is a dependency relationship that relates two elements or sets of elements that represent the same concept at different levels of abstraction or from different viewpoints. |  | |
| Template Binding A binding is a relationship between a template and a model element generated from the template. | (B)  | |

9 UML DIAGRAMS

Class Diagram

| Model element | Button (hot key) | Notation |
|---|------------------|-------------------------------------|
| Package Merge A package merge is a directed relationship between two packages that indicates that the contents of the two packages are to be combined. | | |
| Package Import A package import is defined as a directed relationship that identifies a package whose members are to be imported by a namespace. | | |
| Element Import An element import is defined as a directed relationship between an importing namespace and a packageable element. | | |
| Instance | | (SHIFT+O) |
| Link A connection between two or more objects. | | :MainWindow ————— :ClientWindow |
| Profiling Mechanism Toolbar | | |
| Stereotype A stereotype is an extension mechanism that defines a new and more specialized element of the model based on an existing element. | | (SHIFT+S) |
| MetaClass A class whose instances are classes. Metaclasses are typically used to construct metamodels. | | |
| Extension An extension is used to indicate that the properties of a metaclass are extended through a stereotype, and gives the ability to flexibly add (and later remove) stereotypes to classes. | | |

| Model element | Button (hot key) | Notation |
|--|------------------|----------|
| Profile A Profile is a kind of Package that extends a reference metamodel. | | |
| Profile Application A profile application is used to show which profiles have been applied to a package. | | |

Use Case Diagram

A use case is a description of the functionality (a specific usage of a system) that a system provides. The use case descriptions may exist in a textual form (a simple table), where the use case diagram provides additional information about the relationship between the use cases and the external users. The diagram also allows a definition of the system's boundary.

The Use cases are described only in terms of how they appear when viewed externally by the user (a system's behavior as the user perceives it), and do not describe how the functionality is provided inside the system. The Use cases are not object-oriented, but they are included in the UML to simplify the approach of the project's life-cycle -- from the specification to the implementation.

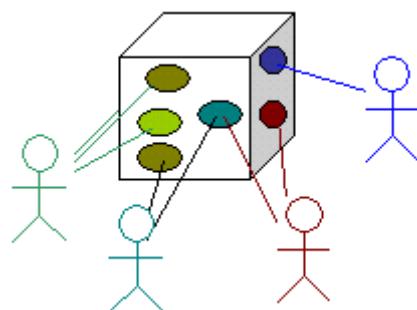
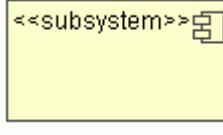


Figure 356 -- The schematic view of the use cases in the system.

Use Case diagram elements

| Element | Button (hot key) | Notation |
|--|--|--|
| Actor Actors represent roles played by human users, external hardware, and other subjects. An actor does not necessarily represent a specific physical entity but merely a particular facet (that is, "role") of some entities that is relevant to the specification of its associated use cases. | (A)  |  |
| Use Case A use case is a kind of behavior-related classifier that represents a declaration of an offered behavior. Each use case specifies a particular behavior, possibly including the variants that the subject can perform in collaboration with one or more actors. The subject of a use case could be a physical system or any other element that may initiate a behavior, such as a component, a subsystem, or a class. | (U)  |  |
| Package A group of classes and other model elements. A package may contain other packages. | (P)  |  |
| Subsystem A subsystem is treated as an abstract single unit. It groups model elements by representing the behavioral unit in a physical system. | (Y)  |  |
| System Boundary Another representation of a package. A system boundary element consists of use cases related by Exclude or Include (uses) relationships, which are visually located inside the system boundary rectangle. | (B)  |  |

9 UML DIAGRAMS

Communication Diagram

| Element | Button (hot key) | Notation |
|--|---------------------|---|
| Include An include (uses) relationship from use case A to use case B indicates that an instance of the use case A will also contain the behavior as specified by B. | (C) | <p>The diagram shows two yellow rounded rectangles representing use cases: "Place Order" at the top and "Order Product" below it. A dashed arrow points from "Place Order" to "Order Product" with the label "<<include>>" written below the arrow.</p> |
| Extend A relationship from an extending use case to an extended use case that specifies how and when the behavior defined in the extending use case can be inserted into the behavior defined in the extended use case. The extension takes place at one or more specific extension points defined in the extended use case. Note: Choose a different Extend direction from the toolbar to draw a line with an opposite arrow end. | (E) | <p>The diagram shows two yellow rounded rectangles: "Receive Product" at the top and "Update Inventory" below it. A dashed arrow points from "Receive Product" to "Update Inventory" with the label "<<extend>>" written below the arrow.</p> |
| Association The participation of an actor in a use case, i.e. instances of the actor and instances of the use case communicate with each other. This is the only relationship between actors and use cases. | (S) | <p>The diagram shows a yellow rounded rectangle "Place Order" on the right and an actor symbol "Salesperson" on the left. A solid line connects them, representing an association.</p> |
| Generalization A relationship between a more general and a more specific element. Note: Choose a different Generalization direction from the toolbar to draw a line with an opposite arrow end. | (G) | <p>The diagram shows three elements: "Administrator" and "Salesperson" at the bottom, and "Person" above them. "Administrator" and "Salesperson" are connected to "Person" by lines with hollow arrowheads, indicating they inherit from "Person".</p> |

Communication Diagram

The Communication diagram illustrates the various static connections between objects, and models their interactions. It also presents a collaboration that contains a set of instances as well as their required relationships given in a particular context, and includes an interaction that defines a set of messages. These messages specify the interaction between the classifier roles within a collaboration that will serve to achieve the desired result.

A Communication diagram is given in two different forms: at the instance level or at the specification level.

Communication Diagram elements

| Element | Button (hot key) | Notation |
|--|--|----------|
| Lifeline A lifeline represents an individual participant in the Interaction. The Lifelines represent only one interacting entity. | | |
| Connector Specifies a link that enables communication between two or more lifelines. Each connector may be attached to two or more connectable elements, each representing a set of lifelines. | | |
| Connector to Self Self connector for self-calls. It begins and ends on the same lifeline. | | |
| Message to Right Message to Left A Message defines a particular communication between the Lifelines of an Interaction. It implies that one object uses the services of another object, or sends a message to that object. A communication can be formed by e.g. raising a signal, invoking an Operation, creating or destroying an Instance. Click on the arrow to expand the toolbar and choose an appropriate message type. |   | |
| Call Message to Right |  | |
| Call Message to Left |  | |
| Send Message to Right |  | |
| Send Message to Left |  | |

9 UML DIAGRAMS

Sequence Diagram

| Element | Button (hot key) | Notation |
|----------------------------------|---------------------|----------|
| Reply Message to Right | | |
| Reply Message to Left | | |
| Create Message to Right | | |
| Create Message to Left | | |
| Delete a Message to Right | | |
| Delete Message to Left | | |

Sequence Diagram

The Sequence diagram focuses on the Message interchange between a number of Lifelines.

A sequence diagram shows the interaction information with an emphasis on the time sequence. The diagram has two dimensions: the vertical axis that represents time and the horizontal axis that represents the participating objects. The time axis could be an actual reference point (by placing the time labels as text boxes). The horizontal ordering of the objects is not significant to the operation, and you may rearrange them as necessary.

In the tables below you will find the description and notation of all the elements available in the Sequence diagram.

Related sections

[Message](#)

[Message in Sequence Diagram](#)

Sequence diagram elements

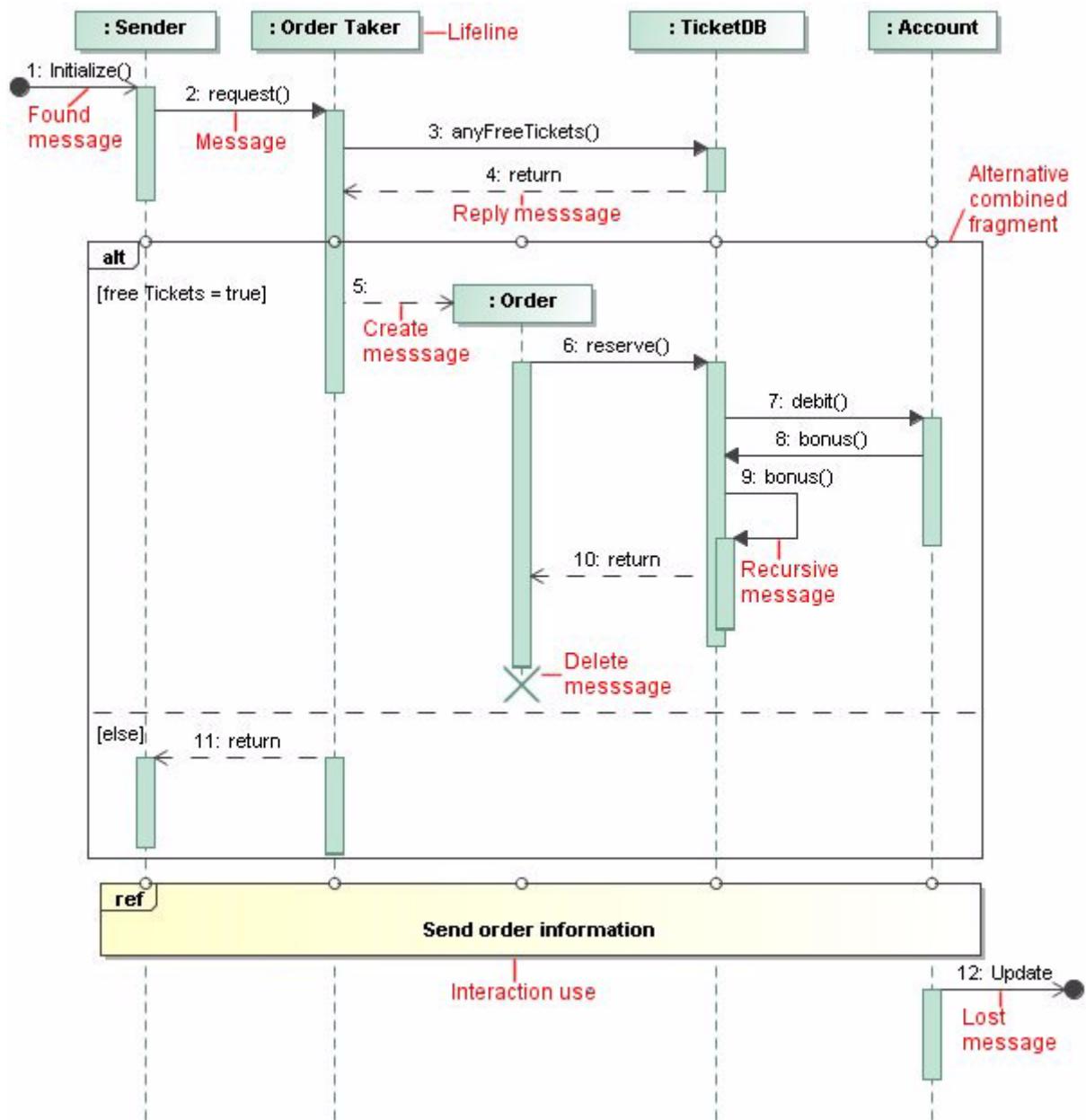
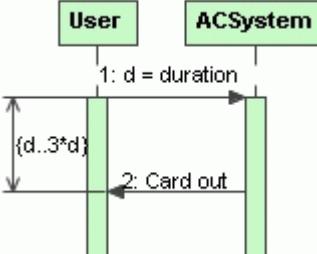
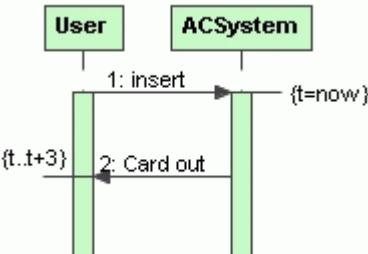


Figure 357 -- Elements in sequence diagram

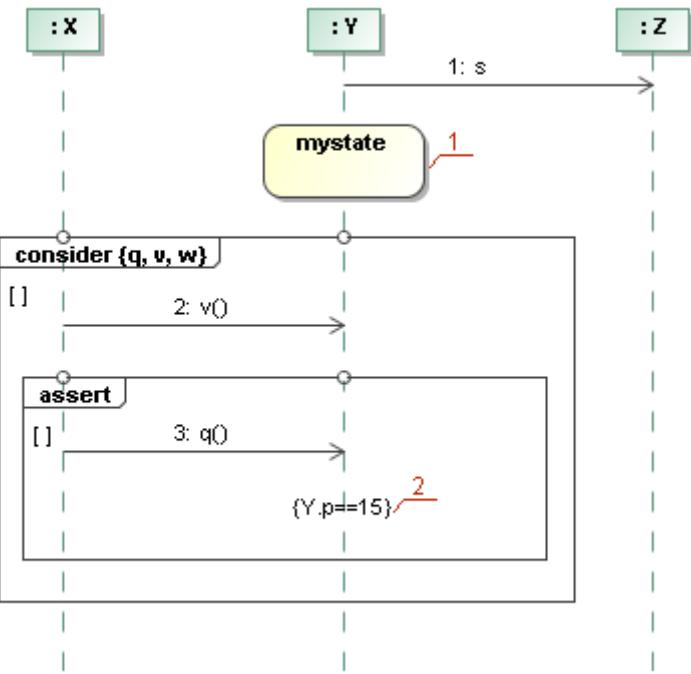
9 UML DIAGRAMS

Sequence Diagram

| Model elements | Button (hot key) | Notation |
|---|--|--|
| Lifeline Represents the existence of an object at a particular time. |  (O) |  |
| Activation Bar Focus of control. Shows the period during which an object is performing an action either directly or through a subordinated procedure. | |  |
| Interaction Use A reference to interactions, communication diagram, sequence diagram, and time diagram can be created. |  (SHIFT+T) | See Figure 357 on page 515. |
| Duration Constraint A duration defines a value specification that specifies the temporal distance between two time instants. |  |  |
| Time Constraint Specifies the combination of min and max timing interval values. |  |  |

9 UML DIAGRAMS

Sequence Diagram

| Model elements | Button (hot key) | Notation |
|--|--|---|
| State Invariant A StateInvariant is a runtime constraint on the participants of the interaction. It may be used to specify a variety of different kinds of constraints, such as values of attributes or variables, internal or external states, and so on. |  (SHIFT+V) | <p>1. State Invariant with a assigned mystate State. 2. State Invariant with a defined constraint $Y.p=15$.</p>  <pre> sequenceDiagram actor X actor Y actor Z state mystate X->>mystate: Y->>Z: 1:s activate mystate assert {Y.p==15} deactivate mystate </pre> |

Combined fragments

A combined fragment defines an expression of interaction fragments. A combined fragment is defined by an interaction operator and corresponding interaction operands. Through the use of combined fragments you will be able to describe a number of traces in a compact and concise manner.

Interaction operands can be of the following types and are described in the table below.

| Interaction operand name and description | Button (hot key) |
|--|--|
| Alternatives The alternative combined fragment alt represents a choice of behavior. Alternative combined fragment has several operands. At most one of the operands has to be chosen. Using alternative combined fragment you can model if-then-else statement. |  (SHIFT+A) |
| Loop The loop combined fragment represents that the loop operand will be repeated a number of times. If the loop contains a separate interaction constraint with a specification, the loop will only continue if that specification evaluates to true during execution regardless of the mini-mum number of iterations specified in the loop. |  (SHIFT+L) |
| Option The option combined fragment opt represents a choice of behavior where either the (sole) ope-rand happens or nothing happens. An option combined fragment is used to model "if-then" construct. |  (SHIFT+O) |

9 UML DIAGRAMS

Sequence Diagram

| | |
|---|--|
| Parallel The interaction operator par designates that the combined fragment represents a parallel merge between the behaviors of the operands. A parallel merge defines a set of traces that describes all the ways that occurrence specifications of the operands may be interleaved without ob-structing the order of the occurrence specifications within the operand. |  (SHIFT+P) |
| Break The interaction operator brk designates that the combined fragment represents a breaking scenario in the sense that the operand is a scenario that is performed instead of the remainder of the enclosing interaction fragment. A break operator with a guard is chosen when the guard is true and the rest of the enclosing interaction fragment is ignored. When the guard of the break operand is false, the break operand is ignored and the rest of the enclosing interaction fragment is chosen. A combined fragment with interaction operator break should cover all Lifelines of the enclosing interaction fragment. |  (SHIFT+B) |
| Negative The interaction operator neg designates that the combined fragment represents traces that are defined to be invalid. The set of traces that defined a combined fragment with interaction operator negative is equal to the set of traces given by its (sole) operand, only that this set is a set of invalid rather than valid traces. All interaction fragments that are different from Negative are considered positive meaning that they describe traces that are valid and should be possible. |  (SHIFT+G) |
| Critical Region The interaction operator crt designates that the combined fragment represents a critical re-gion. A critical region means that the traces of the region cannot be interleaved by other occurrence specifications (on those Lifelines covered by the region). This means that the region is treated atomically by the enclosing fragment when determining the set of valid traces. |  (SHIFT+R) |
| Consider The interaction operator con designates which messages should be considered within this combined fragment. This is equivalent to defining every other message to be ignored. |  (SHIFT+C) |
| Ignore The interaction operator ign designates that there are some message types that are not shown within this combined fragment. These message types can be considered insignif-icant and are implicitly ignored if they appear in a corresponding execution. Alternatively, one can understand ignore to mean that the message types that are ignored can appear anywhere in the traces. |  (SHIFT+I) |
| Weak Sequencing The interaction operator seq designates that the combined fragment represents a weak sequencing between the behaviors of the operands. It is the same as parallel execution, except that event on the same lifeline from different subfragments are ordered in the same order as the subfragments within the enclosing weak sequencing fragment. |  (SHIFT+W) |
| Strict Sequencing The interaction operator str designates that the combined fragment represents a strict se-quencing between the behaviors of the operands. The semantics of strict sequencing de-fines a strict ordering of the operands on the first level within the combined fragment with interaction operator strict. |  (SHIFT+S) |
| Assertion The interaction operator asr designates that the combined fragment represents an asser-tion. The sequences of the operand of the assertion are the only valid continuations. All other continuations result in an invalid trace. |  (SHIFT+R) |

Messages

| Model elements | Button (hot key) | Notation |
|---|------------------|---|
| Message A communication between objects that conveys information with the expectation that an action will ensue. The receipt of a message is one type of event. | (M) | See Figure 357 on page 515. |
| Call Message A call message represents the request to invoke a specific operation. | (A) | See Figure 357 on page 515. |
| Send Message A send message specifies the sending of a request to invoke a specific operation. | (E) | See Figure 357 on page 515. |
| Reply Message The reply message returns the values to the caller of the previous call, completing the execution of the call. | (R) | See Figure 357 on page 515. |
| Create Message A create message specifies the creation of a specific operation. | (C) | The message is connected directly to an object (not lifeline). |
| Delete Message Destroy message represents the destruction of the instance described by the lifeline. | (T) | A large X mark is displayed on the object's lifeline in the message's destination. |
| Diagonal Message Requires some time to arrive, during which another action occurs. | (D) | <pre> sequenceDiagram participant Sender as :Sender participant OrderTaker as :Order Taker Note over Sender: Note over OrderTaker: Sender->>OrderTaker: 1: request() activate OrderTaker Note over OrderTaker: 9: bonus() deactivate OrderTaker </pre> <p>The sequence diagram illustrates a diagonal message. It starts with a synchronous message '1: request()' from the 'Sender' lifeline to the 'Order Taker' lifeline. While the 'Order Taker' is active, an asynchronous return message '9: bonus()' is sent back to the 'Sender' lifeline.</p> |
| Message to Self | (S) | <pre> sequenceDiagram participant Self as Note over Self: Self->>Self: 9: bonus() </pre> <p>The sequence diagram shows a self-loop message '9: bonus()' sent from the 'Self' lifeline back to itself.</p> |
| Recursive message A connected set of messages can be enclosed and marked as iteration. | (U) | See Figure 357 on page 515. |

| | | |
|-----------------------|--|-----------------------------|
| Lost Messages |  (L) | See Figure 357 on page 515. |
| Found Messages |  (F) | See Figure 357 on page 515. |

State Machine Diagram

The behavior of objects of a class can be described in terms of states and events, using a state machine connected to the class under construction.

The state machine is a specification of the sequence of states through which an object or an interaction goes in response to events during its life, together with its responsive actions. The state machine may represent the sequence of states of a particular collaboration (e.g., collection of objects) or even the whole system (which is also considered as a collaboration). The abstraction of all possible states defined in a state machine is similar to the way class diagrams are abstracted: all possible object types (classes) of a particular system are described.

Objects that do not present a very pronounced reactive behavior may always be considered to stay in the same state. In such a case, their classes do not possess a state machine.

State diagrams (also called Statechart diagrams) represent the behavior of entities capable of dynamic behavior by specifying its response to the receipt of event instances. Typically, the state diagrams describe the behavior of classes, but the statecharts may also describe the behavior of other model entities such as use-cases, actors, subsystems, operations, or methods.

A state diagram is a graph that represents a state machine. States and various other types of vertices (pseudostates) in the state machine graph are rendered by the appropriate state and pseudostate symbols, while transitions are generally rendered by directed arcs that inter-connect them. The states may also contain subdiagrams by a physical containment or tiling. Note that every state machine has a top state, which contains all the other elements of the entire state machine. The graphical rendering of this top state is optional.

The states are represented by the state symbols, while the transitions are represented by arrows connecting the state symbols.

The state diagram concerns with an internal object changes (as opposed to the external object interaction in a collaboration). Do not attempt to draw them for all classes in the system, because they are used only for modeling a complex behavior. The state diagram shows all the possible states that objects or collaborations may

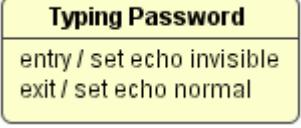
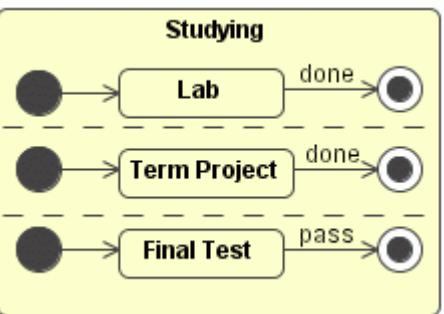
9 UML DIAGRAMS

State Machine Diagram

have, and the events that cause the state to change. An event can be another object that sends a message to it announcing for example that a specified time has elapsed or that some conditions have been fulfilled. A change of a state is called a transition. A transition may also have a connected to it action that specifies what should be done in connection with the state transition.

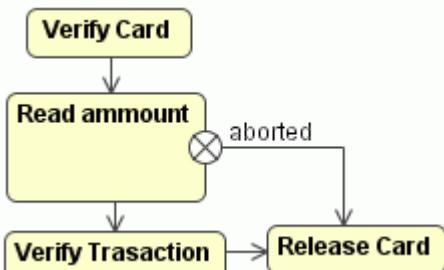
State Machine Diagram elements

NOTE If the black arrow button is placed near a button, either right-click the button or click the black arrow button to open other available buttons.

| Model elements | Button (hot key) | Notation |
|--|--|--|
| State A state models a situation during which some (usually implicit) invariant conditions holds. The invariant may represent a static situation such as an object waiting for some external events to occur. However, it can also model dynamic conditions such as the process of performing some behavior. |  (SHIFT+S) |  |
| Composite State A composite state either contains one region or is decomposed into two or more orthogonal regions. Each region has a set of mutually exclusive disjoint subvertices and a set of transitions. |  (SHIFT+C) |  |
| Orthogonal State An orthogonal state is a composite state with at least 2 regions. |  (C) |  |

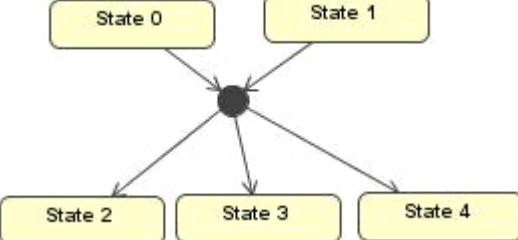
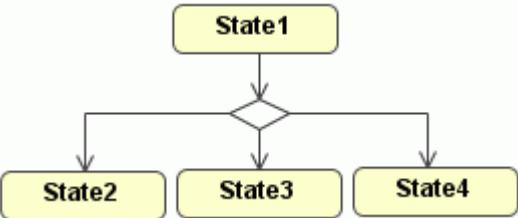
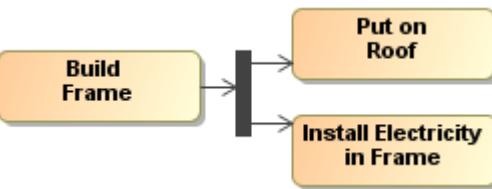
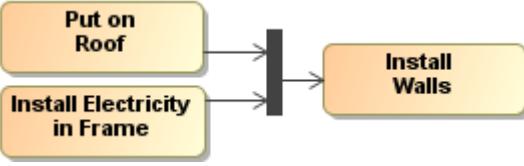
9 UML DIAGRAMS

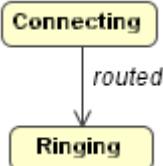
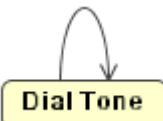
State Machine Diagram

| Model elements | Button (hot key) | Notation |
|--|---------------------|---|
| Submachine State The submachine state specifies the insertion of the specification of a submachine state machine. The submachine state is a decomposition mechanism that allows factoring of common behaviors and their reuse. | (A) |  |
| Initial Represents a default vertex that is the source for a single transition to the <i>default</i> state of a composite state. There can be at most one initial vertex in a region. | (I) |  |
| Final State A special kind of state signifying that the enclosing region is completed. If the enclosing region is directly contained in a state machine and all other regions in the state machine also are completed, then it means that the entire state machine is completed. | (F) |  |
| Terminate Implies that the execution of this state machine by means of its context object is terminated. | (R) |  |
| Entry Point The entry point connection points a reference as the target of a transition. This implies that the target of the transition is the entry point pseudostate as defined in the submachine of the submachine state. | (Y) |  |
| Exit Point The exit point connection points a reference as the source of a transition. This implies that the source of the transition is the exit point pseudostate as defined in the submachine of the submachine state that has the exit point connection point defined. | (U) |  |
| Connection Point Reference The Connection point references of a submachine state can be used as the sources/targets of the transitions. They represent entries into or exits out of the submachine state machine referred by the submachine state. | (Z) | |

9 UML DIAGRAMS

State Machine Diagram

| Model elements | Button (hot key) | Notation |
|--|--|--|
| Deep History Represents the most recent active configuration of the composite state that directly contains the pseudostate (e.g., the state configuration that was active when the composite state was last exited). |  (P) |  |
| Shallow History Represents the most recent active substate of its containing state (but not the substates of that substate). A composite state can have at most one shallow history vertex. |  (SHIFT+R) |  |
| Junction The junction vertices are semantic-free vertices that are used to chain together multiple transitions. They are used to construct the compound transition paths between states. |  (J) |  |
| Choice The choice points are used to split transition paths. In the dynamic choice point, a decision concerning which branch to take is only made after the transition from State1 is taken and the choice point is reached. |  (O) |  |
| Fork Vertical/Horizontal Serves to split an incoming transition into two or more transitions terminating on orthogonal target vertices (i.e., vertices in different regions of a composite state). |  (G)  (D) |  |
| Join Vertical/Horizontal Serves to merge several transitions emanating from source vertices in different orthogonal regions. |  (G)  (D) |  |

| Model elements | Button (hot key) | Notation |
|--|--|---|
| Transition A directed relationship between a source vertex and a target vertex. It may be part of a compound transition, which takes the state machine from one state configuration to another, representing the complete response of the state machine to an occurrence of an event of a particular type. |  (T) |  |
| Transition to Self When an object returns to the same state after the specified event occurs. |  (E) |  |

Protocol State Machine Diagram

A protocol state machine is always defined in the context of a classifier. It specifies which operations of the classifier can be called, in which state, and under which condition, thus specifying the allowable call sequences on the classifier's operations.

The protocol state machine presents the possible and permitted transitions on the instances of its context classifier, together with the operations that carry the transitions.

In this manner, an instance lifecycle can be created for a classifier, by specifying the order in which the operations can be activated and the states through which the instance progresses during its existence.

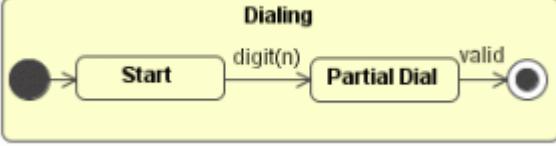
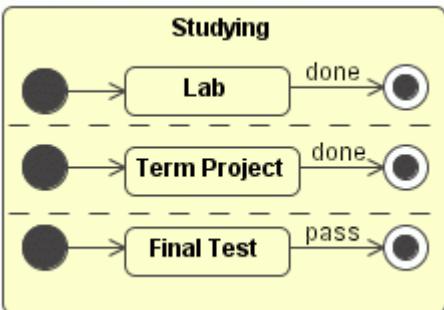
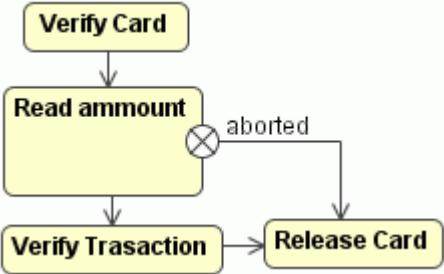
The Protocol State Machine Diagram is created for use with the Protocol State Machine and the Protocol Transitions.

Protocol State Machine Diagram elements

NOTE If the black arrow button is placed near a button, either right-click the button or click the black arrow button to open other available buttons.

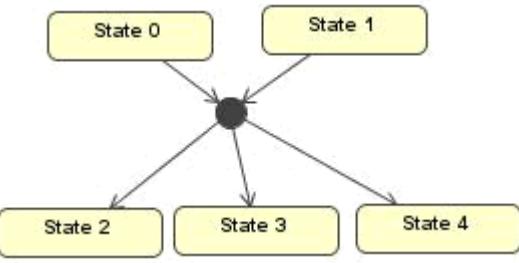
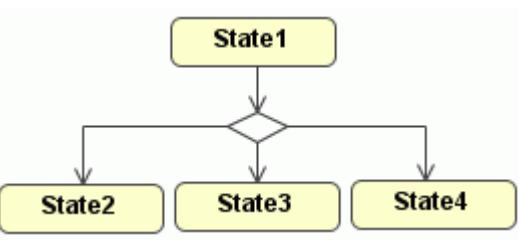
9 UML DIAGRAMS

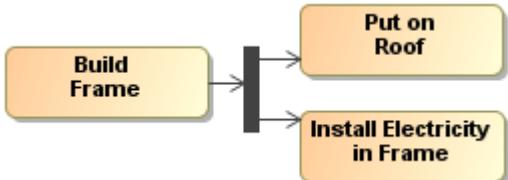
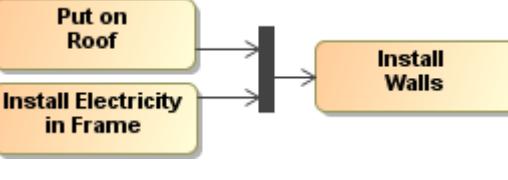
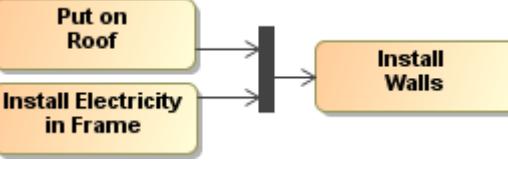
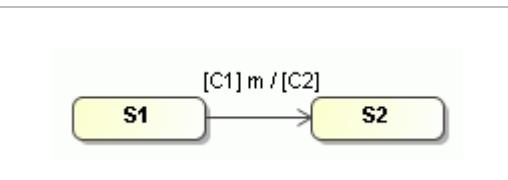
Protocol State Machine Diagram

| Model elements | Button (hot key) | Notation |
|--|--|---|
| State The states of the protocol state machines are exposed to the users of their context classifiers. A protocol state represents an exposed stable situation of its context classifier: When an instance of the classifier is not processing any operation, the user of this instance can always know its configuration state. |  (SHIFT+S) |  |
| Composite State A composite state either contains one region or is decomposed into two or more orthogonal regions. Each region has a set of mutually exclusive disjoint subvertices and a set of transitions. |  (SHIFT+C) |  |
| Orthogonal State An orthogonal state is a composite state with at least 2 regions. |  (C) |  |
| Submachine State A submachine state specifies the insertion of the specification of a submachine state machine. The submachine state is a decomposition mechanism that allows factoring of common behaviors and their reuse. |  (A) |  |
| Initial Presents a default vertex that is the source for a single transition to the <i>default</i> state of a composite state. There can be at most one initial vertex in a region. |  (I) |  |

9 UML DIAGRAMS

Protocol State Machine Diagram

| Model elements | Button (hot key) | Notation |
|--|--|--|
| Final State A special kind of state signifying that the enclosing region is completed. If the enclosing region is directly contained in a state machine and all other regions in the state machine also are completed, then it means that the entire state machine is completed. |  (F) |  |
| Terminate Implies that the execution of this state machine by means of its context object is terminated. |  (R) |  |
| Entry Point The entry point connection points a reference as the target of a transition. This implies that the target of the transition is the entry point pseudostate as defined in the submachine of the submachine state. |  (Y) |  |
| Exit Point The exit point connection points a reference as the source of a transition. This implies that the source of the transition is the exit point pseudostate as defined in the submachine of the submachine state that has the exit point connection point defined. |  (U) |  |
| Connection Point Reference The connection point references of a submachine state that can be used as the sources/targets of the transitions. They represent entries into or exits out of the submachine state machine referenced by the submachine state. |  (Z) | |
| Junction The junction vertices are semantic-free vertices that are used to chain together multiple transitions. They are used to construct the compound transition paths between states. |  (J) |  |
| Choice The choice points are used to split transition paths. In the dynamic choice point, a decision on which branch to take is only made after the transition from State1 is taken and the choice point is reached. |  (O) |  |

| Model elements | Button (hot key) | Notation |
|---|---------------------|---|
| Fork Vertical/Horizontal Serves to split an incoming transition into two or more transitions terminating on orthogonal target vertices (i.e., vertices in different regions of a composite state). | (G) |  |
| | (D) |  |
| Join Vertical/Horizontal Serves to merge several transitions emanating from source vertices in different orthogonal regions. | (G) |  |
| | (D) |  |
| Protocol Transition A protocol transition (transition as specialized in the ProtocolStateMachines package) specifies a legal transition for an operation. Transitions of the protocol state machines have the following information: a pre condition (guard), an on trigger, and a post condition. Every protocol transition is associated to zero or one operation that belongs to the context classifier of the protocol state machine. | (T) | [C1] m / [C2] |
| Protocol Transition to Self When an object returns to the same state after the specified event occurs. | (E) | |

Activity Diagram

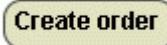
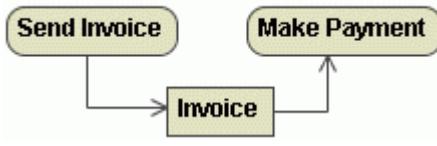
An activity graph is a variation of a state machine. In the state machine, the states represent the performance of actions or subactivities, while the transitions are triggered by the completion of the actions or subactivities. It represents a state machine of a procedure itself. The entire activity diagram is attached (through the model) to a class, such as a use case, or to a package, or to the implementation of an operation. The purpose of this diagram is to focus on flows driven by the internal processing (as opposed to external events). You should use the activity diagrams in situations where all or most of the events represent the completion of internally-generated actions (that is, procedural flow of control). You should use the ordinary state diagrams in situations where asynchronous events occur. An activity diagram is a variant of a state diagram. Organized according to actions, the activity diagrams are mainly targeted towards the representation of the internal behavior of a method (the implementation of an operation) or a use case.

Though activity diagrams are often classified alongside the interaction diagrams, they actually focus on the work performed by a system instead of an object interaction. An activity diagram captures actions and displays their results.

A state diagram may also represent this sequencing of steps. However, given the procedural nature of the implementation of the operations – in which most events simply correspond to the end of the preceding activity – it is not necessary to distinguish states, activities, and events systematically (i.e. state changes and external events have less importance inside the method). It is therefore beneficial to have a simplified representation for directly displaying activities.

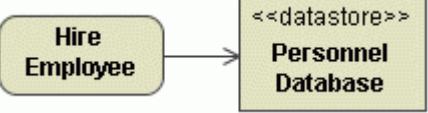
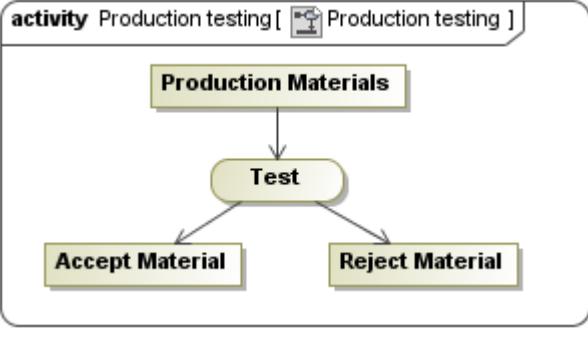
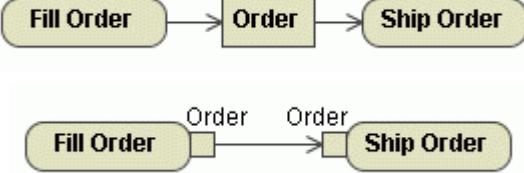
The activity diagram provides a convenient way to describe complex algorithms, parallel operations, and business processes. Together with the collaboration and sequence diagrams, they are used to relate use cases.

Activity Diagram elements

| Element | Button (hot key) | Notation |
|--|------------------|--|
| Action An action is a named element that is the fundamental unit of an executable functionality. The execution of an action represents some transformations or processing in the modeled system. When the action is to be executed or what its actual inputs are is determined by the concrete action and the behaviors in which it is used. | (B) |   |
| Call Operation Action An action that transmits an operation call request to the target object, where it may cause the invocation of the associated behavior. The argument values of the action are available to the execution of the invoked behavior. | (O) |   |
| Opaque Action An opaque action is introduced for implementing specific actions or for use as a temporary placeholder before some other actions are chosen. | |  |
| Any Action This element is introduced in order to maintain any other desirable action element with an appropriate metaclass stereotype applied. | |  |
| Object Node The Activity nodes are introduced to provide a general class for the nodes connected by activity edges. The ActivityNode replaces the use of the StateVertex and its children for activity modeling in UML. | (SHIFT+B) |  |

9 UML DIAGRAMS

Activity Diagram

| Element | Button (hot key) | Notation |
|---|--|--|
| Data Store A data store node is a central buffer node for a non-transient information. A data store keeps all tokens that enter it, copies them when they are chosen to move downstream. Incoming tokens containing a particular object replace any tokens in the object node containing that object. |  (SHIFT+D) |  |
| Activity Parameter Node It is an object node for inputs and outputs to the activities. The Activity parameters are object nodes at the beginning and end of the flows, to accept inputs to an activity and provide outputs from it. |  |  |
| Input Expansion Node An expansion node is an object node used to indicate a flow across the boundary of an expansion region. A flow into a region contains a collection that is broken into its individual elements inside the region, which is executed once per element. |  | |
| Output Expansion Node A flow out of a region combines individual elements into a collection for use outside the region. |  | |
| Object Flow Is an activity edge that can have objects or data passing along it. An object flow models the flow of values to or from the object nodes. |  (SHIFT+F) |  |

9 UML DIAGRAMS

Activity Diagram

| Element | Button (hot key) | Notation |
|---|------------------|--|
| Control Flow Is an edge that starts an activity node after the previous one is finished. Objects and data cannot pass along the control flow edge. | (F) | <pre>graph LR; FillOrder[Fill Order] --> ShipOrder[Ship Order]</pre> |
| Send Signal Action Is an action that creates a signal instance from its inputs, and transmits it to the target object, where it may trigger the state machine transition or the execution of an activity. | (SHIFT+S) | <pre>graph LR; CreateOrder[Create Order] --> FillOrderRequest[Fill Order Request]; FillOrderRequest --> CreateInvoice[Create Invoice]</pre> |
| Accept Event Action Is an action that waits for the occurrence of an event that meets the specified conditions. The Accept event actions handle event occurrences detected by the object owning the behavior. | (E) | <pre>graph LR; CancelOrderRequest[Cancel Order Request] --> CancelOrder[Cancel Order]</pre> |
| Initial Node An initial node is a starting point for executing an activity. It has no incoming edges. | (T) | <pre>graph TD; Start(()) --> ReceiveOrder[Receive Order]</pre> |
| Activity Final An activity final node is a final node that stops all flows in an activity. | (D) | <pre>graph TD; CloseOrder[Close Order] --> End(())</pre> |
| Flow Final The Final node that terminates a flow and destroys all tokens that arrive at it. It has no impact on other flows in the activity. | (L) | <pre>graph LR; BuildComponent[Build Component] --> FlowFinal[Flow Final]; FlowFinal --> InstallComponent[Install Component]; Decision{ } --> FlowFinal</pre> <p>[more components to be built] [no more components to be built]</p> |

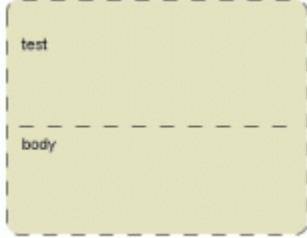
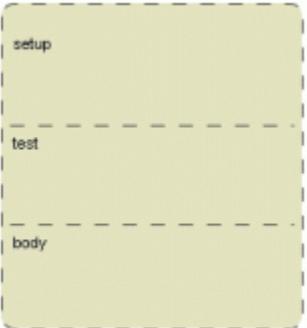
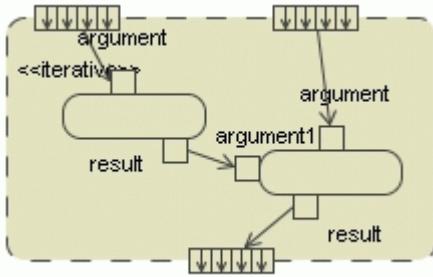
9 UML DIAGRAMS

Activity Diagram

| Element | Button (hot key) | Notation |
|---|------------------|--|
| Decision Decision is a control node that chooses between outgoing flows. A decision node has one incoming edge and multiple outgoing activity edges. | (G) | <pre> graph LR A[Receive Order] -- "[Order accepted]" --> B[Fill Order] A -- "[Order rejected]" --> C[Close Order] B --> D C --> D </pre> |
| Merge A merge node is a control node that brings together multiple alternate flows. It is not used to synchronize concurrent flows but it is used to accept one among several alternate flows. | (G) | <pre> graph LR A[Buy Item] --> B(()) C[Make Item] --> B B --> D[Ship Item] </pre> |
| Fork/Join Horizontal Helps to control parallel actions. | (K) | <pre> graph LR A[Fill Order] --> B(()) B --> C[Fill Order] B --> D[Send Invoice] </pre> |
| Fork/Join Vertical Helps to control parallel actions. | (SHIFT+K) | <pre> graph LR A[Fill Order] --> B(()) B --> C[Fill Order] B --> D[Send Invoice] </pre> |
| Exception Handler An exception handler is an element that specifies a body to execute in case the specified exception occurs during the execution of the protected node. | (P) | <pre> graph LR A[Receive Order] -- "[Order accepted]" --> B[Fill Order] A -- "[Order rejected]" --> C[Fill Order] C --> D[Close Order] B --> E{<<structured>>} E --> F </pre> |
| Structured Activity Node A structured activity node is an executable activity node that may have an expansion into the subordinate nodes. The structured activity node represents a structured portion of the activity that is not shared with any other structured node, except for nesting. | | <pre> graph LR A{<<structured>>} --> B </pre> |

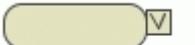
9 UML DIAGRAMS

Activity Diagram

| Element | Button (hot key) | Notation |
|---|------------------|--|
| Conditional Node A conditional node is a structured activity node that represents an exclusive choice among alternatives. | if |  |
| Sequence Node A sequence node is a structured activity node that executes its actions in order. | sequence |  |
| Loop Node A loop node is a structured activity node that represents a loop with the setup, test, and body sections. | loop |  |
| Expansion Region An expansion region is a structured activity region that executes multiple times corresponding to the elements of an input collection. | iteration |  |
| Time Event A time event specifies a point of time by an expression. The expression might be absolute or might be relative to some other points of time. | time |  |

9 UML DIAGRAMS

Activity Diagram

| Element | Button (hot key) | Notation |
|---|--|--|
| Input Pin An input pin is a pin that holds input values to be consumed by an action. They are object nodes that receive values from other actions through object flows. |  (SHIFT+I) |   Output pin Input pin |
| Output Pin A pin that holds output values produced by an action. Output pins are object nodes that deliver values to other actions through object flows. |  (SHIFT+O) | |
| Value Pin A value pin is an input pin that provides a value to an action that does not come from an incoming object flow edge. |  |  argument |
| Swimlanes Swimlanes are used to organize responsibility for actions and subactivities according to the class, dividing an activity diagram. |  (SHIFT+V)  (SHIFT+W) | |

Smart Activity Diagram layout

Dynamic centerlines

The centerlines are displayed only when a center of the shape that was moved or newly drawn is located near the center of another shape that already exists in the diagram. These lines help to draw diagram with aligned shapes easily.

9 UML DIAGRAMS

Activity Diagram

When the center of the shape that was moved coincides with a center of any shape that is placed to its right or left, a horizontal centerline is displayed. When the center of the shape is close to any center of a shape that is located above or below it, a vertical centerline is displayed.

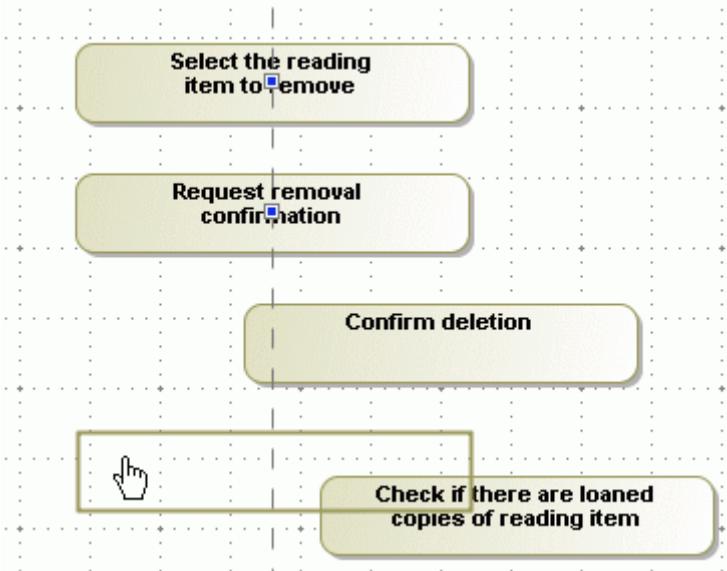


Figure 358 -- Dynamic vertical centerline

To switch off the dynamic centerlines

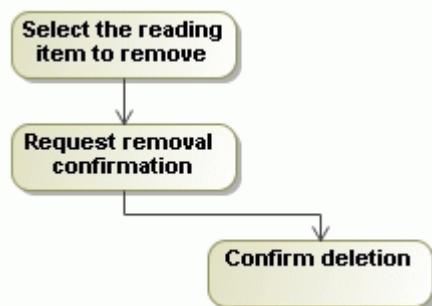
- Click the **Show Centerlines** button  in the diagram toolbar or press **C**.
- From the **Options** main menu, select **Environment**. In the open dialog box, click the **Diagram** node and clear the **Show centerlines in flow diagrams** check box in the **Display** properties group.

Diagram orientation

The diagram orientation is used to assign the correct rectilinear path breaks and draw paths between the activity diagram shapes. The paths can be drawn from side to side, or from the lower to the upper shape borders.

Example:

For a vertical diagram orientation - in this case if two shapes are not in the same centerline, the paths will be connected from the lower border of the first shape to the upper border of the next shape, adding break points:



For a horizontal diagram orientation - in this case the paths will be connected from the side border of the first shape to the next side border of the second shape, adding break points:



To change the diagram orientation

- From the **Options** main menu, select **Project**. In the open dialog box, expand **Diagram** group and in the right side properties list, change the value for the **Diagram Orientation** property.
- From the diagram pane shortcut menu, select **Diagram Properties** and change the value for the **Diagram Orientation** property.

For applying the Activity diagram layout tool, see “Activity Diagram Layout Tool” on page 224.

Implementation Diagram

The Implementation diagrams help developers describe the logical software structure inside a computer system or across a large distributed system. They show aspects of the physical implementation, including the structure of components and the run-time deployment system.

They come in two forms:

- Component diagrams** that show the structure of the components, including the classifiers that specify them and the artifacts that implement them; and
- Deployment diagrams** that show the structure of the nodes on which the components are deployed.

These diagrams can also be applied in a broader way on the business modeling where the components represent the business procedures and artifacts, and the deployment nodes represent the organization units and resources (human and otherwise) of the business.

Component diagram overview

A component diagram represents a physical structure of a code (as opposed to the class diagram, which portrays the logical structure) in terms of code components and their relationships within the implementation environment. A component can be a source code component, a binary component, or an executable component.

A component contains information about the logical class or classes that it implements, thus creating a mapping from a logical view to a component view. Dependencies between the components are shown, making it easy to analyze how a change in one component affects the others. The components may also be shown with any of the interfaces that they expose. They, as with almost any other model elements, can be grouped into packages, much like classes or use cases.

The component diagrams are used in the later phases of the software development, when there is a need to divide up classes among different components. When working with the CASE facilities, the components are used for file-class mapping during code generation, reverse engineering, and round-trip engineering operations.

Deployment diagram overview

The Deployment diagrams show the physical layout of the various hardware components (nodes) that compose a system as well as the distribution of executable programs (software components) on this hardware.

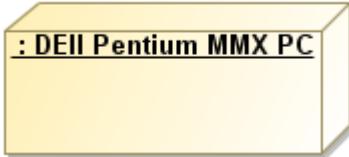
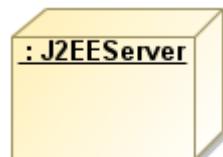
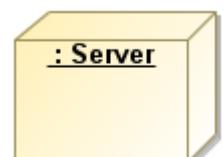
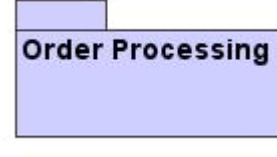
The Deployment diagrams are crucial when dealing with distributed systems. You may show the actual computers and devices (nodes), along with the connections they have to each other, thus specifying a system topology. Inside the nodes, the executable components and objects are located in a way that it shows where the software units are residing and on which nodes they are executed. You may also show dependencies between the components.

Implementation Diagram elements

| Element | Button (hot key) | Notation |
|--|------------------|----------|
| Component The Component may be used to define the requirements for each physical software element. | (K) | |
| Component Instance An instance of a component. | (A) | |
| Port Ports represent interaction points between a classifier and its environment. A port has the ability to specify that any requests arriving at this port are handled. | (SHIFT+R) | |
| Node A Node is a physical object that represents a processing resource, generally having at least a memory and often the processing capability as well. | (O) | |
| Execution Environment The element that is used to indicate that a node is the execution environment. | | |
| Device A physical computational resource with the processing capability upon which artifacts may be deployed for an execution. | | |

9 UML DIAGRAMS

Implementation Diagram

| Element | Button (hot key) | Notation |
|--|---|--|
| Node Instance An instance of a node. | (T)  |  |
| Execution Environment Instance The element that is used to indicate that a node is an instance of the execution environment. | (I)  |  |
| Device Instance | (D)  |  |
| Package A group of classes and other model elements. | (P)  |  |
| Artifact An artifact represents a physical piece of information that is used or produced by a software development process. Examples of Artifacts include models, source files, scripts, and binary executable files. An Artifact may constitute the implementation of a deployable component. | (B)  |  |
| Deployment Specification It indicates a set of properties that defines how a component should be deployed. |  |  |
| Artifact Instance |  |  |

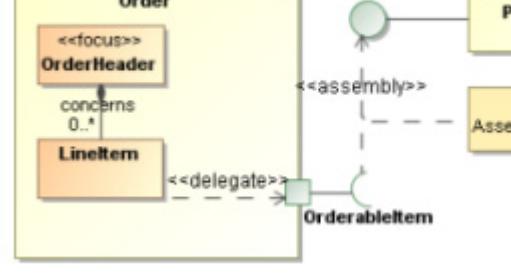
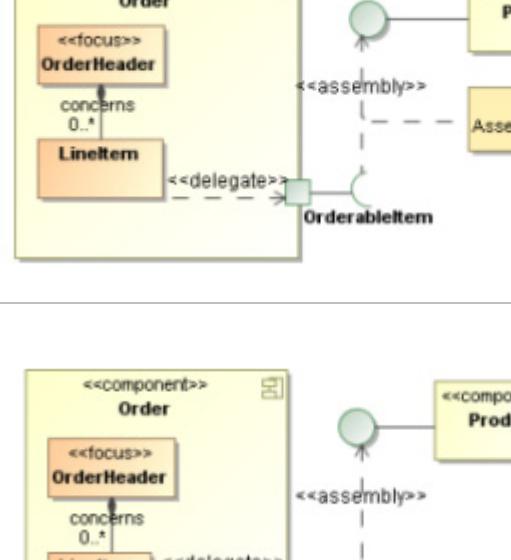
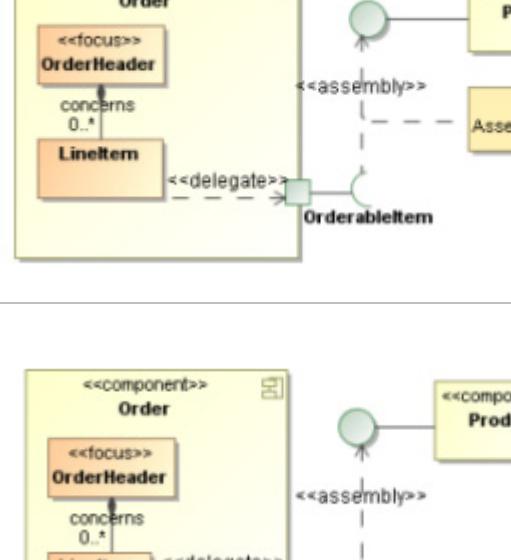
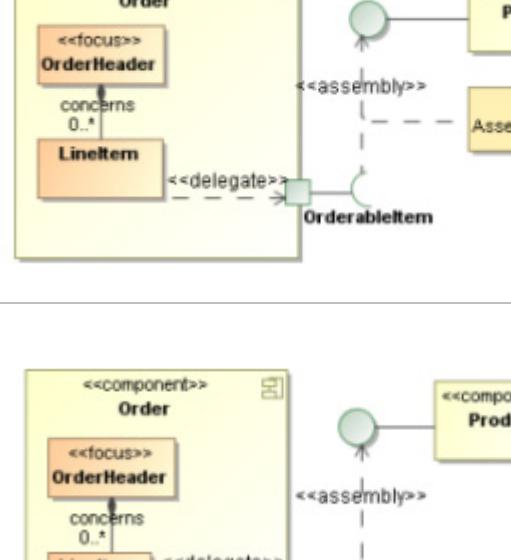
9 UML DIAGRAMS

Implementation Diagram

| Element | Button (hot key) | Notation |
|--|------------------|----------|
| Deployment Specification Instance | | |
| Interface All functionalities implemented by a particular component. | (F) | |
| Interface Realization The classifier at the tail of the arrow implements the interface that is located at the arrow head or uses that interface. Note: Choose a different Interface Realization direction from the toolbar to draw a line with an opposite arrow end. | (R) | |
| Realization A realization signifies that the client set of elements are an implementation of the supplier set, which serves as the specification. | (E) | |
| Usage A usage is a relationship in which one element requires another element (or set of elements) for its full implementation or operation. Usage is also used to create required interfaces. Note: Choose a different Usage direction from the toolbar to draw a line with an opposite arrow end. | (U) | |
| Instance An instance specifies the existence of an entity in a modeled system and completely or partially describes the entity. It has a unique name, state, and classifier as the type of the identifier. | | |

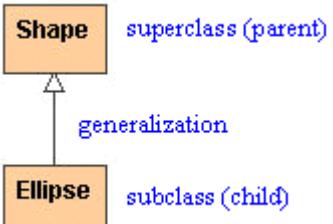
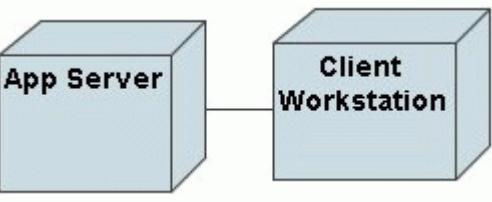
9 UML DIAGRAMS

Implementation Diagram

| Element | Button (hot key) | Notation |
|---|---|--|
| Assembly Connector An assembly connector is a connector between two components that defines that one component provides the services that another component requires. |  |  <p>The diagram illustrates an Assembly Connector (AC) between the Order component and the Product component. The Order component contains a LineItem class, which concerns the OrderHeader class. A dashed line labeled '<<assembly>>' connects the LineItem class to the OrderableItem interface, which is implemented by the OrderableItem class. The Product component is shown as a separate box.</p> |
| Delegation Connector A delegation connector is a connector that links the external contract of a component (as specified by its ports) to the internal realization of that behavior by the component's parts. |  |  <p>The diagram illustrates a Delegation Connector (DC) between the Order component and the Product component. The Order component contains a LineItem class, which delegates to the OrderableItem interface. A dashed line labeled '<<delegate>>' connects the LineItem class to the OrderableItem interface. The Product component is shown as a separate box. A yellow box labeled 'Delegation path' points to the delegation relationship.</p> |
| Deployment A deployment is the allocation of a deployment target to an artifact or artifact instance. |  |  <p>The diagram shows a deployment target named ':AppServer' represented as a 3D cube. It has two deployment relationships: one to an artifact named 'ShoppingCart.jar' and another to an artifact named 'Order.jar'. Each relationship is labeled with '<<deploy>>'.</p> |
| Manifestation A manifestation is the concrete physical rendering of one or more model elements by an artifact. |  |  <p>The diagram shows a manifestation relationship. A circle representing a deployment target is connected to a artifact named 'Order.jar' via a line labeled '<<manifest>>'. The artifact 'Order.jar' is shown as a box with a file icon.</p> |

9 UML DIAGRAMS

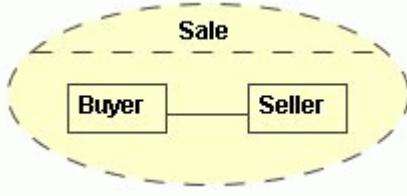
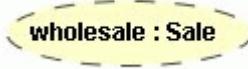
Composite Structure Diagram

| Element | Button (hot key) | Notation |
|--|---|---|
| Generalization | (G)  |  Shape superclass (parent) generalization Ellipse subclass (child) |
| Link A relationship between a specification element and an implementation element. | (L)  |  :MainWindow — :ClientWindow |
| Communication Path A communication path is an association between two DeploymentTargets, through which they are able to exchange signals and messages. |  |  App Server — Client Workstation |

Composite Structure Diagram

The Composite Structure diagram allows a decomposition and modeling of the internal structure of the classifiers.

Composite Structure Diagram elements

| Element | Button (hot key) | Notation |
|--|--|--|
| Classifier Is extended with the capability to own collaboration uses. These collaboration uses link a collaboration with the classifier to give a description of the works of the classifier. |  (SHIFT+G) |  |
| Part |  (P) |  |
| Port A port may appear either on a contained part representing a port on that part, or on the boundary of the class diagram, representing a port on the represented classifier itself. |  (SHIFT+R) |  |
| Collaboration A collaboration describes a structure of collaborating elements (roles), each performing a specialized function, which collectively accomplishes some desired functionalities. Its primary purpose is to explain how a system works and, therefore, it typically only incorporates those aspects of reality that are deemed relevant to the explanation. |  (Q) |  |
| Collaboration Use A collaboration use represents one particular use of a collaboration to explain the relationships between the properties of a classifier. The collaboration use shows how the pattern described by a collaboration is applied in a given context, by binding specific entities from that context to the roles of the collaboration. |  (U) |  |
| Connector Each connector may be attached to two or more connectable elements, each representing a set of instances. Each connector end is distinct in the sense that it plays a distinct role in the communication realized over the connector. |  (C) |  |

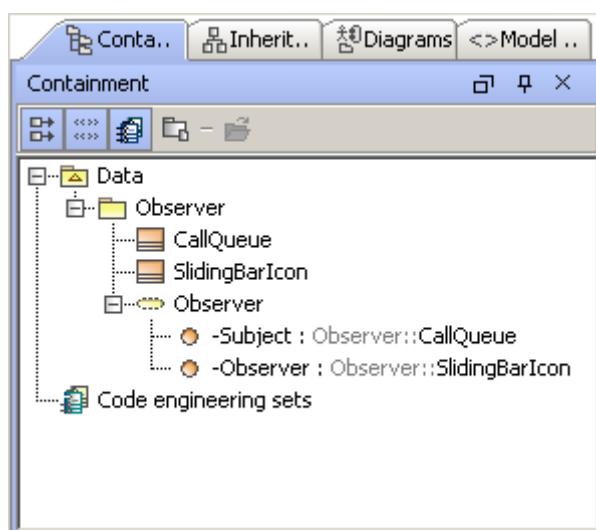
| Element | Button (hot key) | Notation |
|--|------------------|----------|
| Role Binding Is a relationship from parts to the Collaboration Use (in diagram). | (B) | |

Displaying existing Parts on the Composite Structure diagram creation

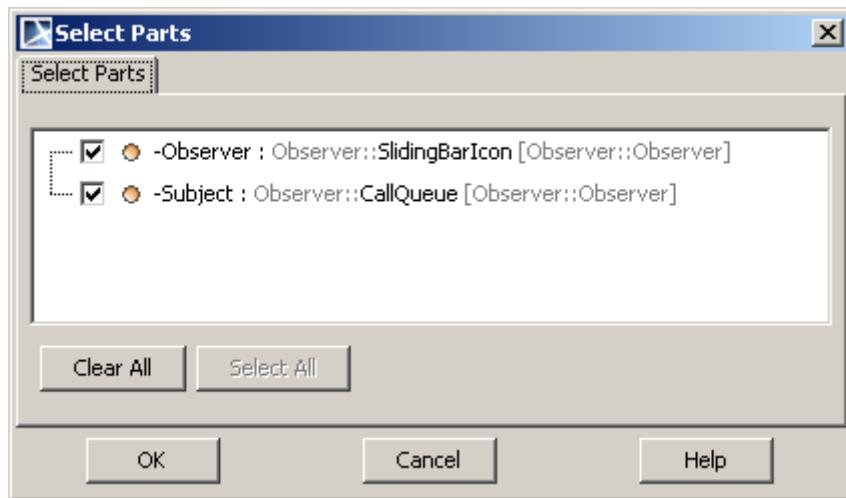
On a new Composite Structure diagram creation, if context classifier has properties, the **Select Part** dialog will open. Select Parts to be drawn automatically in the Composite Structure diagram.

Case study to display existing parts on the Composite Structure diagram [Example is taken from UML 2 Superstructure Specification]:

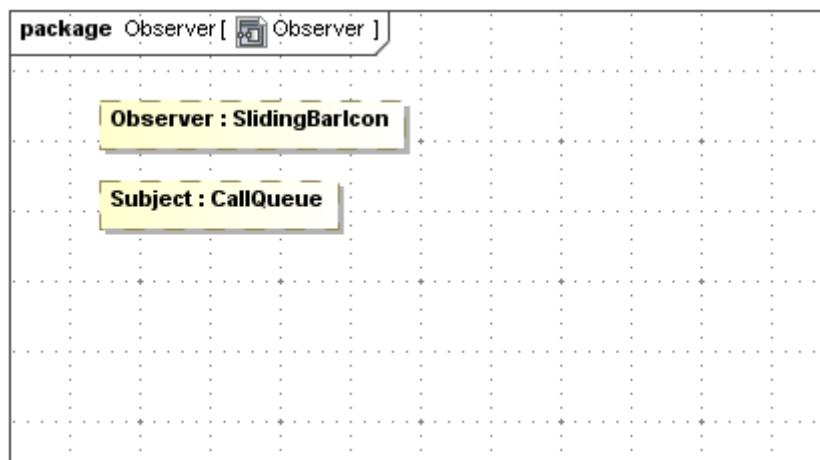
1. Create the *Observer* package with *CallQueue*, *SlidingBarIcon* classes inside and *Observer* collaboration.
2. To the *Observer* collaboration add *Subject* property (*CallQueue* type) and *Observer* property (*SlidingBarIcon* type)..



Create a composite structure diagram inside the collaboration. The **Select Parts** dialog will open.



3. Select properties (parts) to display them in a diagram. Click OK.
4. Parts are displayed in a diagram.



Interaction Overview Diagram

The Interaction Overview diagram focuses on the overview of the flow of control between Interactions. It is based on the Activity diagram notation. Interactions in the Interaction Use diagram are represented using the Interaction Use element. See the sample of the Interaction Overview diagram in Figure 359 on page 546.

9 UML DIAGRAMS

Interaction Overview Diagram

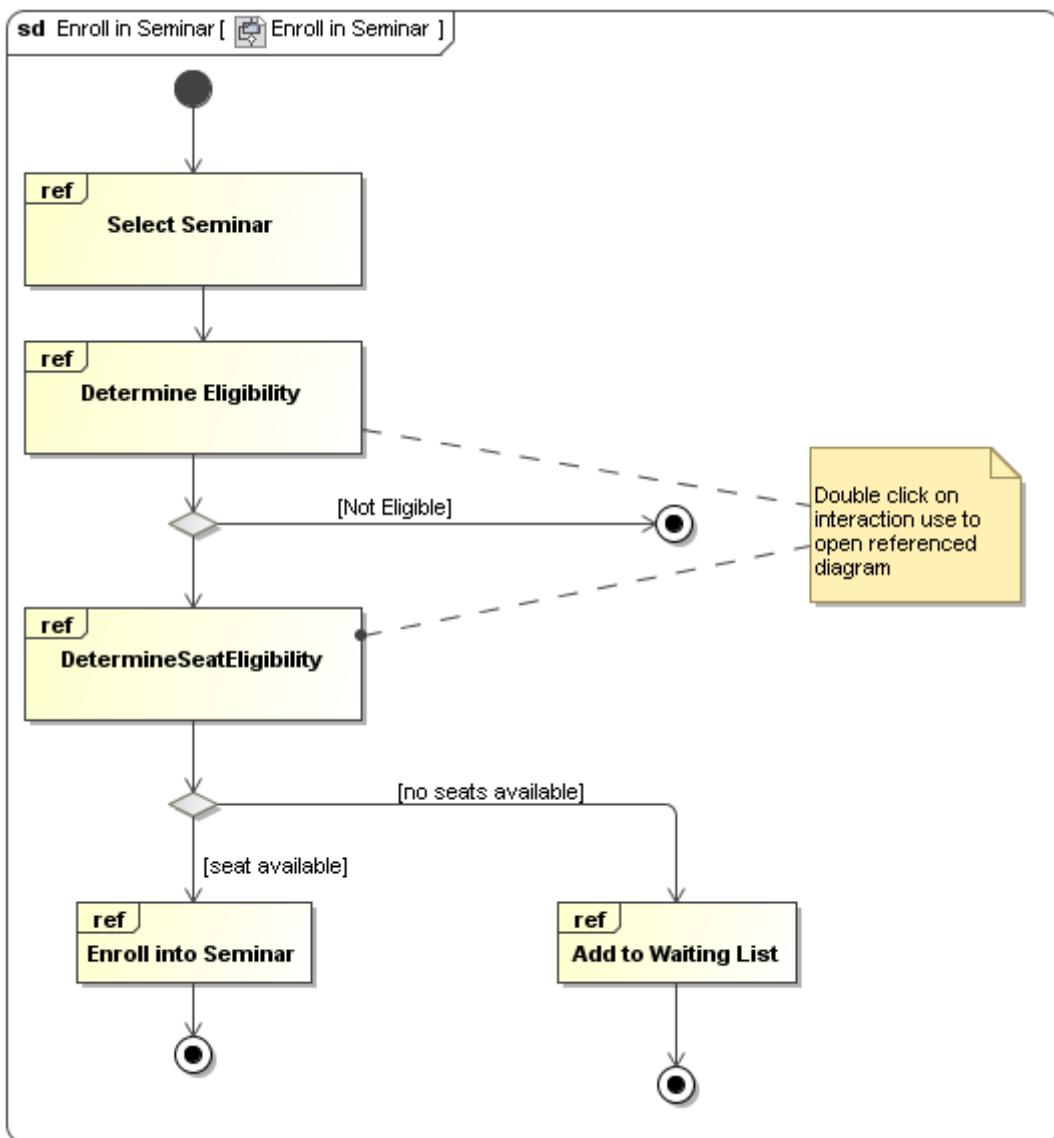


Figure 359 -- Sample of the Interaction Overview diagram

| Element | Button (hot key) | Notation |
|--|---------------------|----------|
| Interaction Use Represents interactions. | (Shift-T) | |

Other elements of the Interaction Overview diagram are the same as in the Activity diagram. For more information about the Activity diagram, see “Activity Diagram elements” on page 529.

10 EXTENSION DIAGRAMS

MagicDraw supports these extensions of UML diagrams:

- "[Content Diagram](#)" (Standard, Professional, Architect, and Enterprise editions), see page 572.
- "[Robustness Diagram](#)" (Standard, Professional, Architect, and Enterprise editions), see page 577.
- "[User Interface Modeling Diagram](#)" (Standard, Professional, Architect, and Enterprise editions), see page 550.

Moreover MagicDraw provides various types of other diagrams:

- "[Web Diagram](#)" (Standard, Professional, Architect, and Enterprise editions), see page 579.
- "[CORBA IDL Diagram](#)" (Architect and Enterprise editions), see page 583.
- "[Generic DDL Diagram](#)" (Architect and Enterprise editions), see page 584.
- "[Oracle DDL Diagram](#)" (Architect and Enterprise editions), see page 585.
- "[WSDL Diagram](#)" (Architect and Enterprise editions), see page 587.
- "[XML Schema Diagram](#)" (Architect and Enterprise editions), see page 589.
- "[Time Diagram](#)" (Standard, Professional, Architect, and Enterprise editions), see page 591.
- "[Struts Diagram](#)" (Professional, Architect, and Enterprise editions), see page 592.
- "[Networking Diagram](#)" (Standard, Professional, Architect, and Enterprise editions), see page 594.
- "[Business Process Diagram](#)" (Standard, Professional, Architect and Enterprise editions), see page 597.
- "Free Form Diagram" (Standard, Professional, Architect and Enterprise editions).
- "[Relation Map Diagram](#)" (Standard, Professional, Architect and Enterprise editions), see page 605.
- "[Dependency Matrix](#)" (Standard, Professional, Architect and Enterprise editions), see page 605.
- "[NEW! Generic Table](#)" (Standard, Professional, Architect and Enterprise editions), see page 605.

Working with Diagrams

For the general information about working with diagrams see chapter "Diagramming" on page 167.

If there is a need, you can enable or disable a plugin that allows to create the appropriate extension diagram in the **Plugins** tab of the **Environment Options** dialog. For more information see section "Plugins Options Tab" on page 104.

Patterns

Various types of classes can be created in every class diagram using a **Pattern Wizard**. It contains GOF, Java, Junit, CORBA IDL, XML Schema, and WSDL design patterns.

New patterns and modifications of the existing ones can also be created using Java code or JPython scripts. For the detailed description see "[MagicDraw OpenAPI UserGuide.pdf](#)".

For a detailed description of the **Pattern Wizard**, see section "Controlling Merge memory usage" on page 341.

10 EXTENSION DIAGRAMS

To open the Pattern Wizard

- In the class diagram, click the **Class by Pattern** button.
- From the class shortcut menu, select **Tools** and then select the **Apply Pattern** subcommand.
- Select a class and select **Apply Pattern** from the **Tools** main menu.

Common Elements

| | | |
|----------------------|--|--|
| Toolbar Button | Button (Hot key) | |
| Selection |  (Escape) | |
| Sticky Button |  (Z) | |
| Text Box | abc (X) | |
| Text Box (HTML Text) | abc HTML (SHIFT+X) | |
| Note |  (N) | |
| Note (HTML Text) |  (SHIFT+N) | |
| Comment |  | |
| Comment (HTML Text) |  | |
| Anchor |  (H) | |
| Constraint |  (SHIFT+H) | |
| Containment |  (SHIFT+C) | |

10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

| Toolbar Button | Button (Hot key) | |
|-------------------|--|--|
| Dependency |  (SHIFT+D) | |
| Separator | ---- | |
| Rectangular Shape |  (W) | |

User Interface Modeling Diagram

[View Online Demo](#)

NEW! User Interface Modeling

TIP!

Web UI sample project, which represents the UI Modelling Diagram usage for the web based interface modelling is available at <MagicDraw installation directory>\samples\diagrams\User Interface Modeling.

User Interface (UI) Modeling diagram makes it possible to build prototypes of user interfaces, connect UI mock-ups with whole Architectural model, export them as images, and create browsable reports for presentations. In short, they help gather information faster and thus save time and money.

The merits of User Interface Modeling lies in its ability to:

- Create WYSIWYG User Interface prototypes rapidly.
- Integrate User Interface development with UML specifications.
- Get immediate feedback from prospective users on real situations and reuse it for next designs.
- Create browsable reports with the **MagicDraw Report Wizard** (for more information about report generation see "[MagicDraw ReportWizard UserGuide.pdf](#)").

The Reasons Why Prototyping is so Important?

Various versions of user interfaces need to be tested in order to see how clients respond. This is especially true if you work on key dialogs like for example a sign up screen for a website or an e-commerce application. Working with User Interface prototypes instead of "real" user interfaces will thus enable you to work with small details such as color and the position of a button and substantially reduce designer and programmer overhead in the design phase.

You can build up mock-ups or prototypes, get quick feedback from prospective users, and take and reuse the feedback for future designs. All this is possible with User Interface prototyping. The feedback loop makes for quicker mature designs that work for everybody, which is what really matters.

You can build mock-ups or prototypes to meet following objectives:

- **Integrate GUI development with UML specifications** since this is a new field with increased design capability and since it is easier to see missing parts in UML or User Interface modeling.
- **Help business analysts gather requirements** because of the permanent feedback from prospective users, which makes it easier for them to get all the information needed.
- **Create browsable User Interface prototypes / GUI simulation / Story boarding** since several prototypes can be connected via hyperlinks with one another and then be presented together in a report and simulate the workflow of an application.
- **Resolve flow issues** as it is easier to think through a problem when a user-related interface can be changed quickly.
- **Get "buy-in" from stakeholders** as it can be shown more rapidly how a particular user interface will look like.
- **Run a usability test before full production** so that potential errors in usability like overfilled screens or a usage too complicated can be avoided.
- **Test a series of interaction widgets.** You may think, for example, that another text field or button would be good on a screen. Since modifying a prototype is so easy, it is not a problem to present these suggestions to others.

For more information about prototyping, go to:

<http://today.java.net/pub/a/today/2005/08/23/prototyping.html>

<http://www.scottberkun.com/essays/12-the-art-of-ui-prototyping/>

Working with User Interface Modeling Diagrams

To start working with a User Interface Modeling diagram

Do either one of the following:

- Click the **User Interface Diagram**  button in the Diagram toolbar.
- Select **New Diagram > Custom Diagrams > User Interface Modeling Diagram** from the Package or Model shortcut menu in Browser.

After a project is loaded and a diagram is created, UI modeling elements can be added to the diagram pane.

To add new elements to the User Interface Modeling diagram

- Simply drag and drop them out of the Diagrams modeling elements toolbar.

For further information, see “User Interface Modeling” on page 552 and “Case Studies for User Interface Modeling” on page 564.

To load a sample with an already created User Interface model

1. Open either the **UI Modeling Samples.mdzip** or **UI Modeling in System Development.mdzip** projects which you will find in the <MagicDraw installation folder>/samples/diagrams/User Interface Modeling directory.
2. After loading the sample, select the models from the **Index-Diagram** by clicking the hyperlinks.

User Interface Modeling

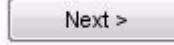
This section gives the basic information you need to know about user interface modeling in MagicDraw. It includes the following sub-sections:

- “User Interface Modeling diagram elements” on page 552.
- “Modifying a table” on page 557
- “Modifying a tree” on page 559.
- “Nesting” on page 559.
- “Reusability” on page 560.
- “Specifying elements” on page 560.
- “Icon usage” on page 562.
- “Using symbol properties” on page 563.
- “User interface prototyping” on page 564.

For User Interface Modeling, see “Case Studies for User Interface Modeling” on page 564. Three case studies will provide step-by-step instructions how to build user interfaces and create browsable reports.

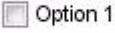
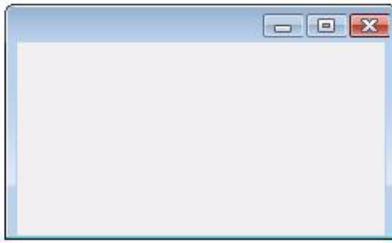
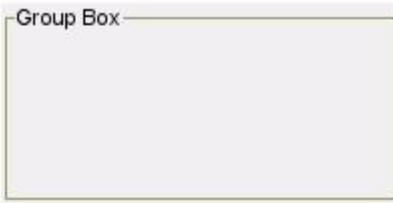
User Interface Modeling diagram elements

The following table lists all available User Interface modeling elements and their properties.

| Element | Description | Properties | Example |
|--|--|---|---|
|  Button | A regular button with the possibility to add text, icon or both. | <ul style="list-style-type: none"> - Icon: select one of the available icons or choose your own. - Inactive: activates/deactivates the button. - Text: displays text on the button on the diagram pane. - Selected: sets the button as selected or not. |  |
|  Toolbar button with Icon | Small sized button with pre-defined icon (close, copy, cut, delete, new, open, paste, print, redo, save, search, undo, zoom in, zoom out). | - same as for regular buttons. |  |
|  Button with Text | Button with pre-defined text (Back, Cancel, Close, Finish, Next, OK). | - same as for regular buttons. |  Next > |

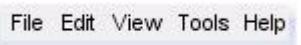
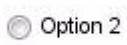
10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

| Element | Description | Properties | Example |
|--|---|---|---|
|  Check Box | Check box with text. | <ul style="list-style-type: none"> - Inactive: activates/deactivates the box. - Text: displays text next to the check box on the diagram pane. - Selected: marks/unmarks the check box. |  |
|  Combo Box | Regular combo box in non-editable style with possibility to show a value. | <ul style="list-style-type: none"> - Inactive: activates/deactivates the box. - Text: displays text in the combo box. |  |
|  Frame* | Main container component represented by a regular Internal Frame. Any other component can be nested inside. | <ul style="list-style-type: none"> - Icon: select one of the available icons or choose your own. - Inactive: activates/deactivates the frame. - Maximize: defines if the maximize-icon should be visible or hidden. - Minimize: defines if the minimize-icon should be visible or hidden. - Title: displays the title of the frame. |  |
|  Group Box* | A panel with a titled border. Any other component can be nested inside. | <ul style="list-style-type: none"> - Border Type: defines the style of the border. - Title: displays the title of the border. - Titled: defines whether border text should be shown or not. |  |
|  Hyperlink | A blue colored and underlined item for showing some text or an icon or both. | <ul style="list-style-type: none"> - Icon: select one of the available icons or choose your own. - Inactive: activates/deactivates the link. - Text: displays the text of the hyperlink. |  |
|  Label | Item with ability to present a text, an icon or both. | <ul style="list-style-type: none"> - Icon: select one of available icons or choose your own. - Inactive: activates/deactivates the label. - Text: displays the text in the label. | This is a Label |

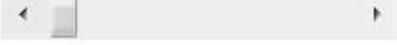
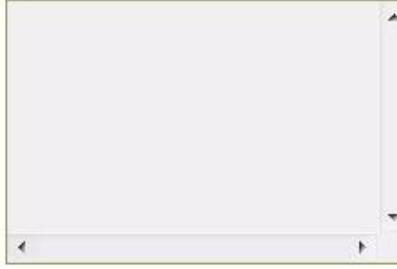
10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

| Element | Description | Properties | Example |
|--|--|---|---|
|  List | Bordered item which stores and shows numerous values. Values can be shown as selected. | <ul style="list-style-type: none"> - Horizontal Scroll Bar: defines the visibility of the horizontal bar. - Inactive: activates/deactivates the list. - Selected Value: defines which value should be shown as selected. - Values: add/remove values to/from the list. - Vertical Scroll Bar: defines the visibility of the vertical bar. |  |
|  Menu Bar | A bar that can present several text items. | - Values: add/remove menus to/from the bar. |  |
|  Panel* | A bordered panel. | |  |
|  Password Field | A bordered field with a text which is hidden through symbols. | <ul style="list-style-type: none"> - Hidden: decides whether the text should be shown as text or symbolized. - Inactive: activates/deactivates the field. - Text: displays the text of the password field. |  |
|  Progress Bar | A bar which presents the state of progress. | <ul style="list-style-type: none"> - Maximum Value: sets the maximum value of the bar. - Minimum Value: sets the minimum value of the bar. - Value: sets the display text on the bar. - Vertical: switches between horizontal and vertical orientation. |  |
|  Radio Button | A radio button with possibility to present some text. | <ul style="list-style-type: none"> - Inactive: activates/deactivates the button. - Text: displays text next to the radio button. - Selected: marks/unmarks the button. |  |

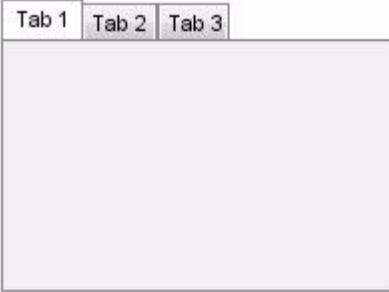
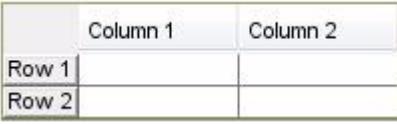
10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

| Element | Description | Properties | Example |
|--|---|---|---|
|  Scroll Bar | Item which represents regular scroll bar. | <ul style="list-style-type: none"> - Inactive: activates/deactivates the bar. - Vertical: switches between horizontal and vertical orientation. |  |
|  Scroll Pane* | A panel which can contain a vertical and/or a horizontal scroll bar. | <ul style="list-style-type: none"> - Horizontal Scroll Bar: defines the visibility of the horizontal bar. - Vertical Scroll Bar: defines the visibility of the vertical bar. |  |
|  Separator | A line with ability to split up a component. | <ul style="list-style-type: none"> - Vertical: switches between horizontal and vertical orientation. |  |
|  Slider | An item for presenting a range of several values. The knob of this item can be moved to these values. | <ul style="list-style-type: none"> - Inactive: activates/deactivates the slider. - Invert: switches maximum and minimum values. - Knob Position: defines the location of the knob. - Maximum Value: sets the maximum value of the slider. - Minimum Value: sets the minimum value of the slider. - Spacing: sets the space between two markings. - Values: defines at which position the slider should be a text. - Vertical: switches between horizontal and vertical orientation. |  |
|  Spinner | Regular spinner that can show a value. | <ul style="list-style-type: none"> - Inactive: activates/deactivates the spinner. - Text: displays value on the spinner. |  |

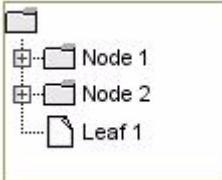
10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

| Element | Description | Properties | Example |
|--|---|--|--|
|  Tabbed Pane* | Item which represents a panel with tabs. | <ul style="list-style-type: none"> - Active Tab: sets which tab should be shown as selected. - Tab Position: selects the placement of the tabs. - Tabs: add/remove tabs to/from the pane. |  |
|  Table | A table item with the ability to add and/or remove columns and/or rows. See section "Modifying a table" on page 557 for more information. | <ul style="list-style-type: none"> - Column Header: defines whether the column header should be visible or not. - Row Header: defines whether the row header should be visible or not. |  |
|  Text Area | A multi-line bordered item to present long text. | <ul style="list-style-type: none"> - Horizontal Scroll Bar: defines the visibility of the horizontal bar. - Inactive: activates/deactivates the area. - Text: displays text in the text area. - Vertical Scroll Bar: defines the visibility of the vertical bar. |  |
|  Text Field | A single-line bordered item to present some text. | <ul style="list-style-type: none"> - Inactive: activates/deactivates the field. - Text: displays text in the text field. |  |
|  Tool Bar* | A bar which can consist of any other component. Usually used to nest buttons or labels with icons. | |  |

10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

| Element | Description | Properties | Example |
|--|--|--|---|
|  Tree | An item to present a tree structure. It can add numerous nodes and/or leaves to it. For information on how to do this, see section "Modifying a tree" on page 559. | <ul style="list-style-type: none">- Expand: defines whether the whole tree should be expanded or not.- Horizontal Scroll Bar: defines the visibility of the horizontal bar.- Icon: select one of available icons or choose your own.- Text: displays the text of the root node.- Vertical Scroll Bar: defines the visibility of the vertical bar. |  |

* NOTE

The elements marked with an asterisk (*) are the **Container** elements and can nest other components.

Modifying a table

You can add columns and rows to a table, order and name them. When expanding a table in the Browser tree, the structure of the table will be displayed. Rows are added directly to the model element of the table. Columns are stored in the Columns Enumeration. For every new column a new Cell will automatically be added to every row. To add a value to the table, you need to edit the name of the Cell.

You can edit a table in the following ways:

- Add a row to the table.
- Change the order of the rows.
- Add a column to the table.
- Change the order of the columns.
- Add a value to the table.

To add a row to a table

1. Right-click on the table element in the Browser tree to open the shortcut menu.
2. Select **New Element** and then **Row**. New row will be added to the table.
3. Type in a title for the new row.

To change the order of the rows

1. Select the table in the diagram.
2. Right-click on it and select **Specification** in the shortcut menu.
3. Go to the Rows node - all rows are listed there.
4. Select one row. You can move it to another position in the list by clicking **Up** or **Down** buttons.

10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

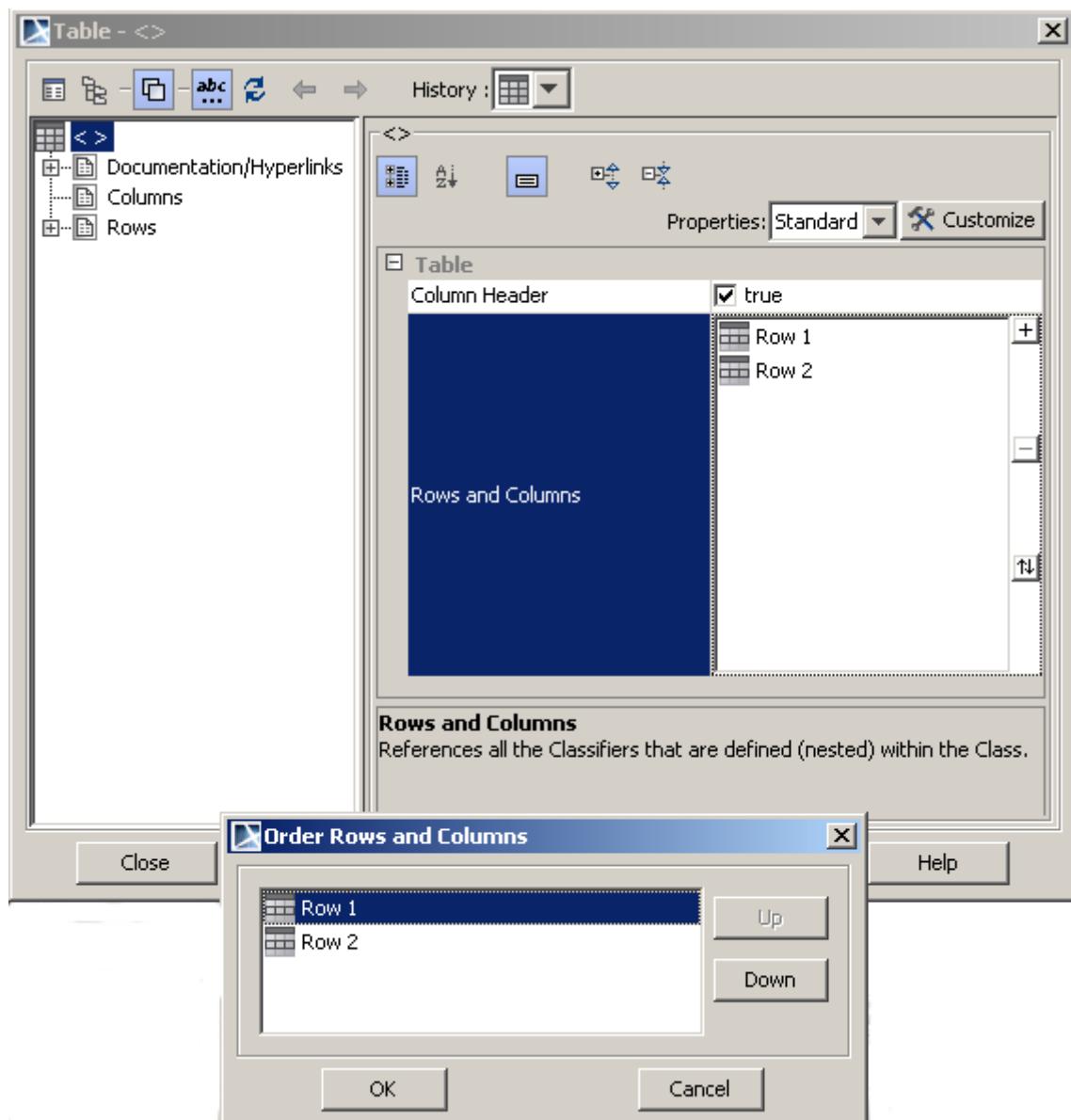


Figure 360 -- Example of row ordering in a table

To add a new column to a table

1. Select the table in the diagram.
2. Right-click on it and select **Specification** in the shortcut menu.
3. Go to **Columns** node - all columns are listed there.
4. You can add new column to the table and also one new cell to every row by clicking **Create** button.
5. Type in a title for the new column.

To change the order of the columns

1. Select the table in the diagram.
2. Right-click on it and select **Specification** in the shortcut menu.
3. Go to **Columns** node - all columns are listed there.
4. Select one column. By clicking **Up** or **Down** buttons, you can move it to another position in the list.

To change the cell name

1. Expand the Table element in Browser. You will see the Cell elements listed under the Table elements.
2. Change the name of the Cell:
 - 2.1 Select a Cell element in Browser and press **F2**. The Cell text area will turn to editable.
 - 2.2 Slowly double-click on the Cell in Browser. The Cell text area will turn to editable.
 - 2.3 Right-click on the Cell element and select **Specification**. The **Cell** specification dialog box will open. Specify the **Text** property and click **Close**.

To add a value to the table, you need to edit the text property of the cell.

Modifying a tree

You can add as many nodes and leaves as you need to a tree and to any node in that tree and also change the order of the elements of a tree.

To add a node or a leaf to the tree or to another node

1. Right click on the tree or a node element in the model browser.
2. Select **New Element** and then **Node or Leaf**.
3. Type in a text property for the element. This will define the text of the node or the leaf in the diagram.

To change the order of the elements of a tree

1. Select the table in the diagram.
2. Right-click on it and select **Specification** in the shortcut menu.
3. Go to Nodes and Leafs node - all rows are listed there.
4. Click the small button for Alphabetical View.
5. Select one row. You can move it to another position in the list by clicking **Up** or **Down** buttons. Click **OK**. The order will then be changed.

Nesting

You can move and arrange user interface modeling elements, since they nest each other, and thus create deep structured User Interface models. However, not all elements have the nesting ability: only **Container** elements can nest other elements (these elements are marked with an asterisk (*) in the section "User Interface Modeling diagram elements" on page 552).

10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

As Figure 361 on page 560 shows, all frames nest in each others and, as a result, if you were to move the element with title *Frame 1*, all other elements would stay in position within this element.

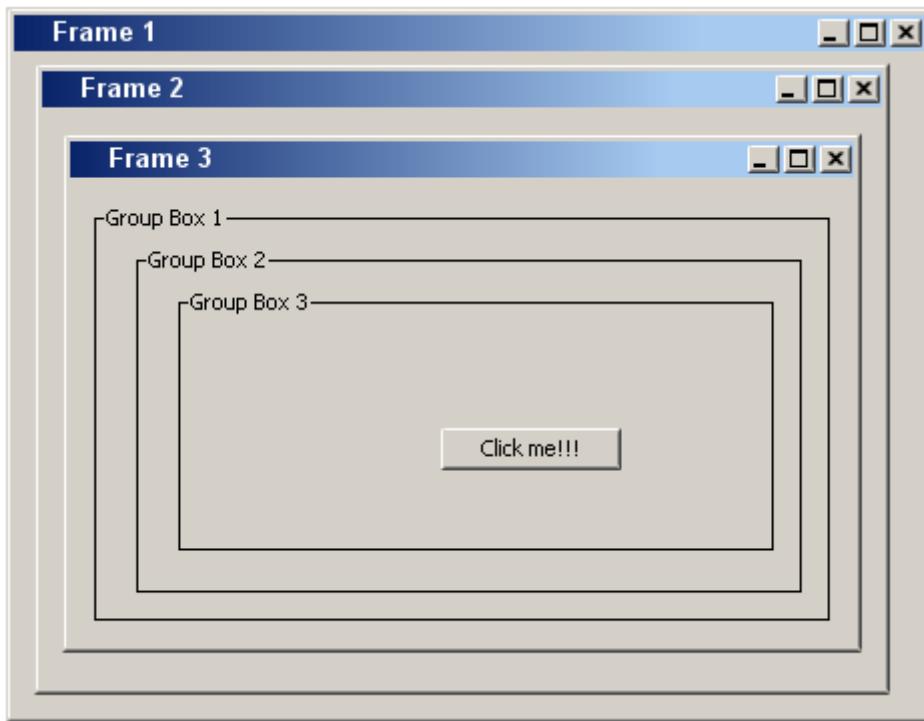


Figure 361 -- Example of a deep structure that was created using containers

Reusability

If you have created a complex model and need to use it again, you do not have to create a new one. All you need to do is select the elements, copy them and then paste them into the same diagram or in any other diagram. You can also reuse just a single element by copying and pasting it or by dragging it from the Browser onto the diagram pane.

Specifying elements

You can modify UI modeling elements by editing their properties in the following three ways:

- Via the **Specification** dialog box.
- Via the shortcut menu.
- On a diagram pane.

To edit properties via the **Specification** dialog box

1. Select an element on the diagram.
2. Right-click on it and select **Specification** on the shortcut menu (Figure 362 on page 561).
3. Edit one of the properties to modify the element (for the effects of editing properties, see section “User Interface Modeling diagram elements” on page 552).

10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

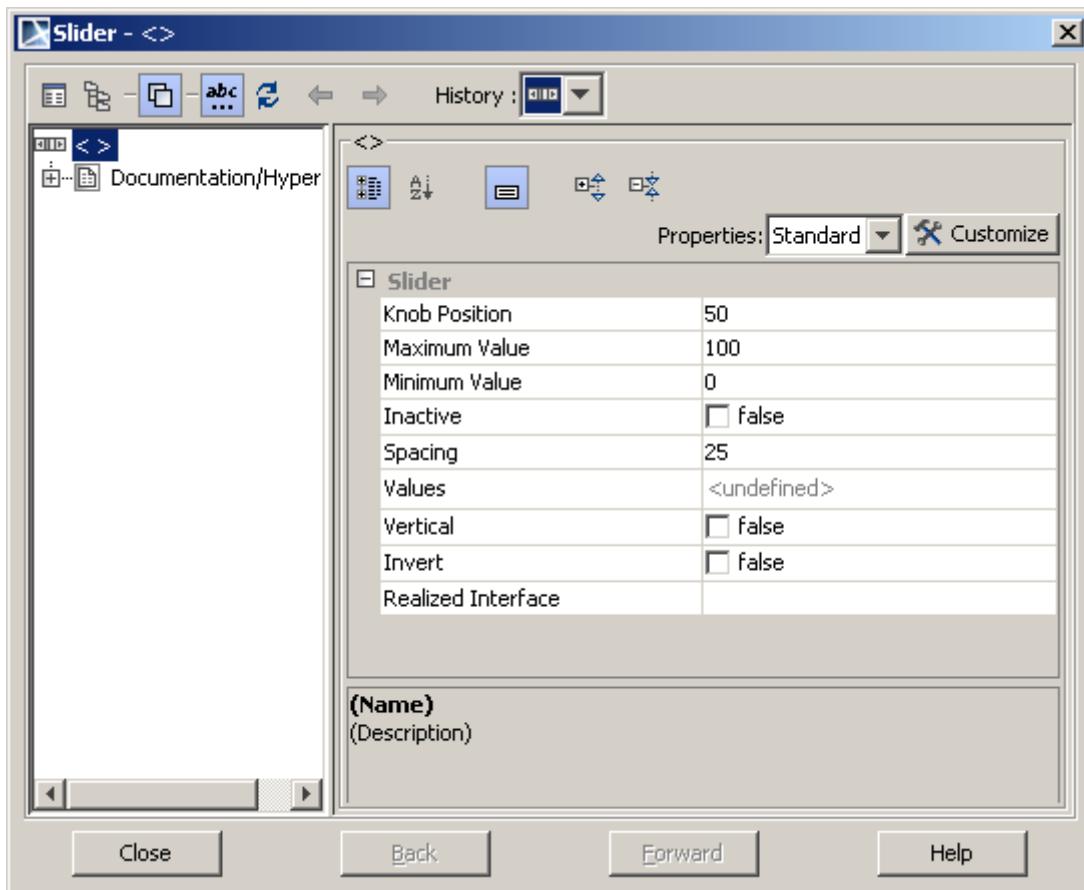


Figure 362 -- Specification dialog box of the Slider element

The so called boolean properties that have only two values can also be edited via the shortcut menu.

To edit properties via the shortcut menu

NOTE This is possible for the boolean properties only.

1. Select an element on the diagram.
2. Right-click on it. The shortcut menu will appear and the properties which can be edited will be displayed at its bottom (Figure 363 on page 562).
3. Select the property you want to change.

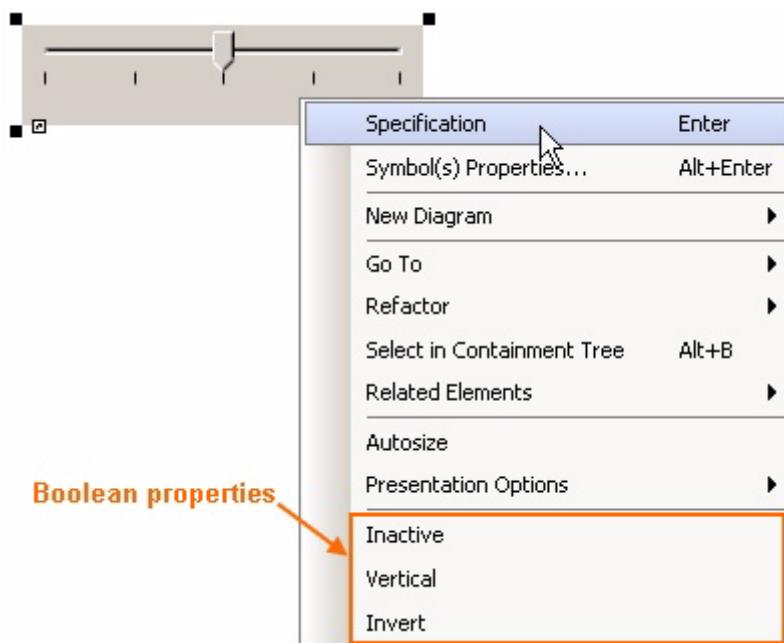


Figure 363 -- Shortcut menu of the Slider element with boolean properties

For the elements that own a text or a title, for example a button or a frame, it is also possible to edit the appropriate properties straight on a diagram.

To edit properties on a diagram

NOTE This is possible for the text or title properties only.

1. Select an element on the diagram.
2. Click on the selected element.
3. Type the text on the element (Figure 364 on page 562).
4. Click on the free space of the diagram pane. The text you have typed will appear on the element and in the **Text/Title** property of the element as well.

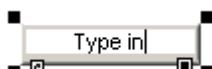


Figure 364 -- Editing a title of an element

Icon usage

Some elements are capable of owning an icon. User Interface Modeling provides a number of frequently-used icons such as cut, delete, undo/redo, etc. Of course, it is also possible to use any images from your computer as icons.

To add an icon

1. Select an element which can own an icon, for example a button..

NOTE To find out which elements can have an icon, see “User Interface Modeling diagram elements” on page 552.

2. Right click on the element and select **Specification** in the shortcut menu.

10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

3. In the **Icon** list, select an icon you want to appear on the element (Figure 365 on page 563).

NOTE If you want to use any image from your computer as an icon, click **custom**.

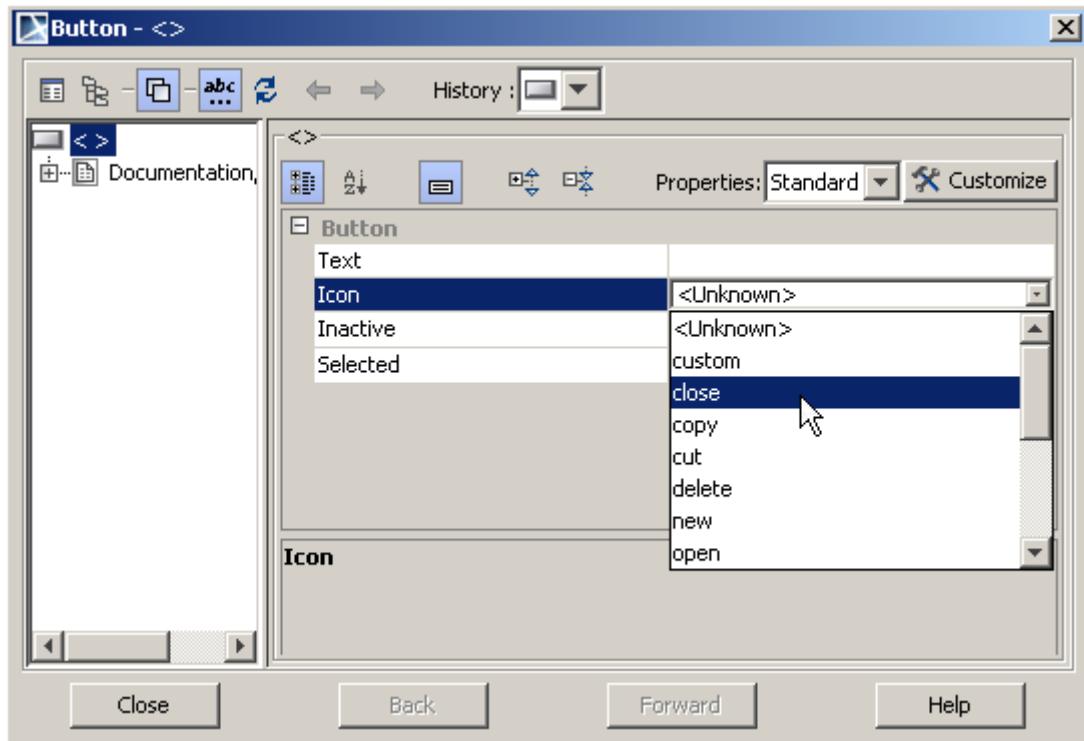


Figure 365 -- Setting an icon for a button

Using symbol properties

User Interface Modeling also enables you to modify user interface elements via symbol properties.

To edit an element via symbol properties:

1. Select an element on the diagram.
2. Right-click and select **Symbol(s) Properties** on the shortcut menu.
3. Change one of the properties listed in the table below.

TIP!

If you want to set, for example, the same background color to more than one UI modeling element, you can select all these elements in the diagram and then change the required property for all of them in the same way.

The following table lists the supported symbol properties:

| Symbol Property | Effect |
|-----------------|--|
| Fill Color | Sets the background color of the component to the chosen one, property Use Fill Color must be marked. |
| Font | Sets the font if there is any text in the component. |
| Text Color | Sets the text color of the component to the chosen one. |
| Pen Color | Sets the color of the border for components that have one. |

User interface prototyping

User Interface Modeling enables you to create browsable reports. All you need to do, when creating a breathable report, is add a hyperlink to the UI model element, is to and link it to the other one. Once a report of these models has been created, click on any element in the report. You will then be directed to the other diagram.

As Figure 367 on page 566 shows, when you click **OK** in the upper left-hand frame you will be directed to the next frame. And using the **OK** of this frame you can get to a couple of frames in which you can step forward and backward as you want since they are all linked to one another.

"Case Study 3 - User Interface Prototyping Example" on page 570 provides an example of how this feature works and explains how to add hyperlinks and create browsable reports.

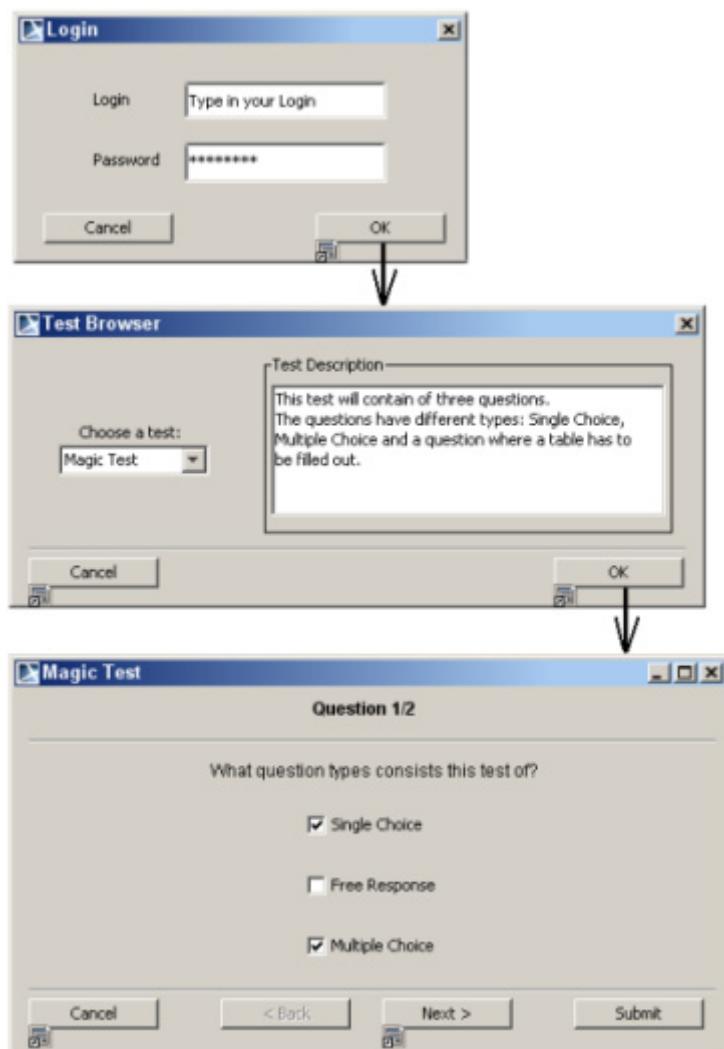


Figure 366 -- Example of User Interface Models and their Workflow

Case Studies for User Interface Modeling

This section includes three case studies on how to create User Interface models:

- "Case Study 1 - Modelling User Interface for the Report Wizard Window" on page 565.
- "Case Study 2 - Slider Example" on page 568.
- "Case Study 3 - User Interface Prototyping Example" on page 570.

10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

Full detailed samples can be found in “<MagicDraw installation folder>\samples\diagrams\User Interface Modeling\UI Modeling Samples.mdzip”.

Case Study 1 - Modelling User Interface for the Report Wizard Window

This case study provides step-by-step instructions for modeling the MagicDraw Report Wizard window.

Step 1# Create a new Project

1. Choose **New Project** from the **File** menu.
2. Name it *Report Wizard*.

Step 2# Create a new Diagram

1. Click the **User Interface Diagram**  button in the main diagram toolbar.
2. Name the new diagram *Report Wizard*.

Step 3# Create Container components

1. Click the **Frame** button in the User Interface diagram toolbar and drag-drop it on the diagram pane. A Frame component will be created.
2. Name the Frame component.
 - Select Frame on the diagram and type its name *Report Wizard*.
 - Or you can define the Frame component name in the **Frame** specification dialog box:
 - 2.1.1 Double-click the Frame component on diagram to open the Frame specification dialog box.
 - 2.1.2 Type *Report Wizard* in the **Title** field.
3. Uncheck properties **Maximize** and **Minimize** of the Frame in the **Frame** specification dialog box.
4. Define Icon for the Frame component:
 - 4.1.1 Select the **custom** option in the **Icon** property in the **Frame** dialog box. The **Open** dialog box will open.
 - 4.1.2 Browse to the "<MagicDraw installation>\plugins\com.nomagic.magicdraw.uimodeling" folder, and select the *nomagic.png* image to set the frame icon.
5. Create two **Group Boxes** and one **Separator**.
 - 5.1.1 Name one **Group Box** as *Select Template*
 - 5.1.2 For the other Group Box uncheck property **Titled**.
 - 5.1.3 Drag all three components to the **Frame** and arrange everything like in Figure 368 on page 567.

10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

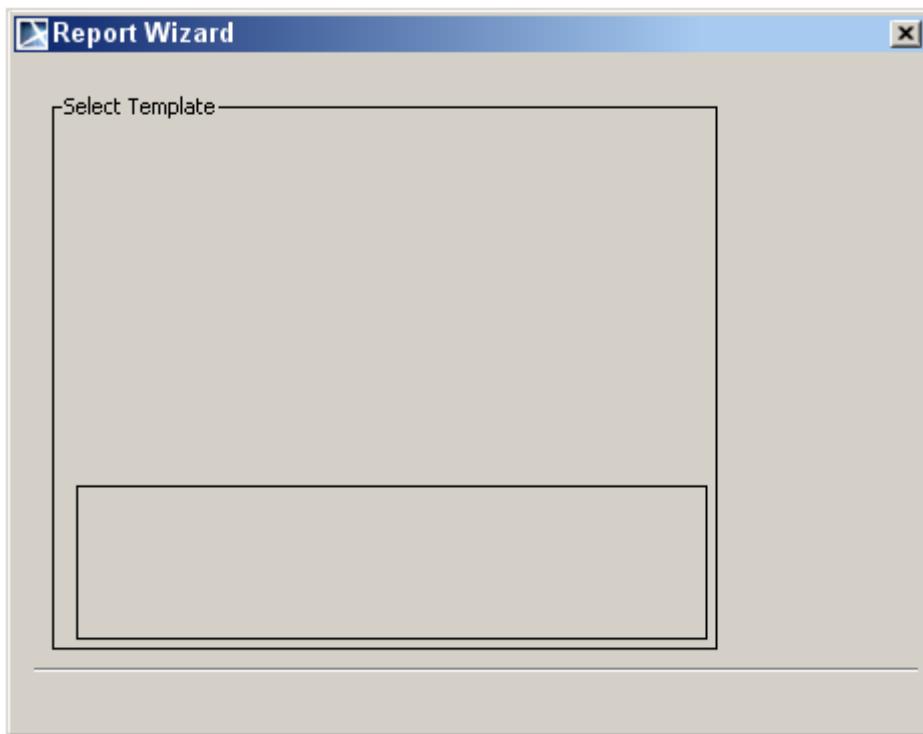


Figure 367 -- Report Wizard Frame with Group Boxes and Separator

Step 4# Create a Tree

1. Create a new Tree component:
 - 1.1.1 Click the **Other** group in the diagram toolbar to expand the toolbar.
 - 1.1.2 Click the **Tree** button in the diagram toolbar and drag-drop it on the *Select Template* Group Box on diagram pane.
2. Delete old nodes from the Tree component.
3. Create new nodes to the Tree component.
 - 3.1.1 Right-click on the Tree in the Containment tree to invoke its shortcut menu and choose **New Element > Node**.
 - 3.1.2 Create five new **Nodes** in this way and name them as shown in Figure 369 on page 568.
4. Add Leaves to Nodes.
 - 4.1.1 You can add a **Leaf** by right-clicking on a **Node** in the containment tree and then choose **New Element > Leaf**.
 - 4.1.2 Add at least one **Leaf** to every **Node** to indicate that the nodes have internal elements.

10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

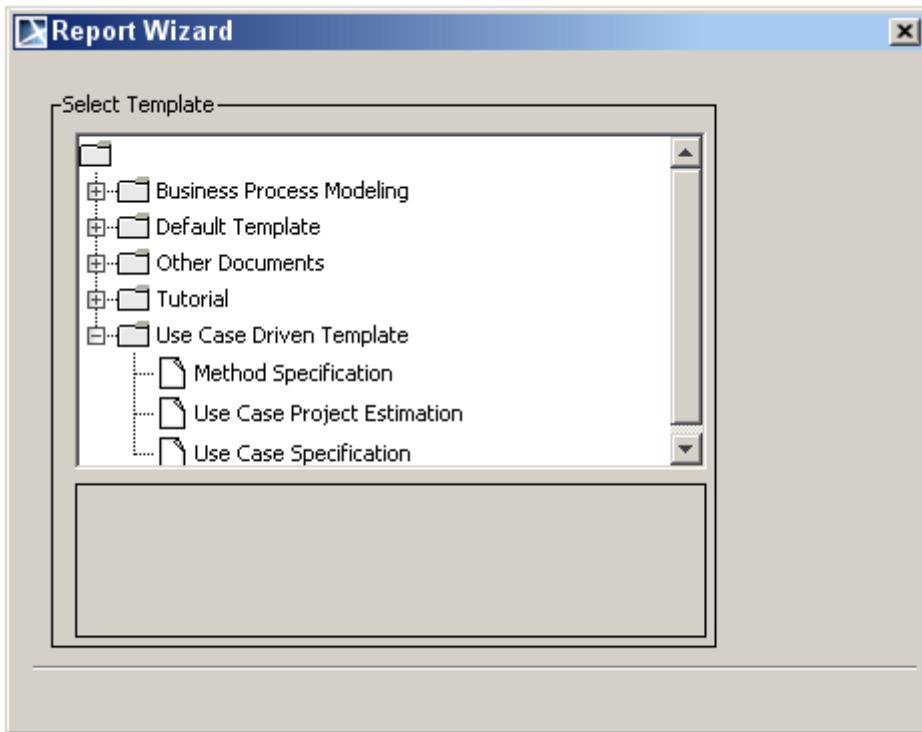


Figure 368 -- The Report Wizard Window with Added Tree

Step 5# Add Buttons

1. Create predefined text buttons.
 - 1.1.1 Click the **Buttons** group in the diagram toolbar to expand the toolbar.
 - 1.1.2 Right-click the **Back** button to expand the text buttons group. Create **Back**, **Next** and **Cancel** buttons (Figure 370 on page 569).
2. Create regular buttons.
 - 2.1.1 Click the **Button** button in the toolbar and drag it to the pane.
 - 2.1.2 Select the created button in the diagram and type in *New*.
 - 2.1.3 Create remaining buttons by repeating steps 2.1 and 2.2.
3. Check property **Inactive** for all buttons, except **Next>**, **Cancel**, **New** and **Import**.

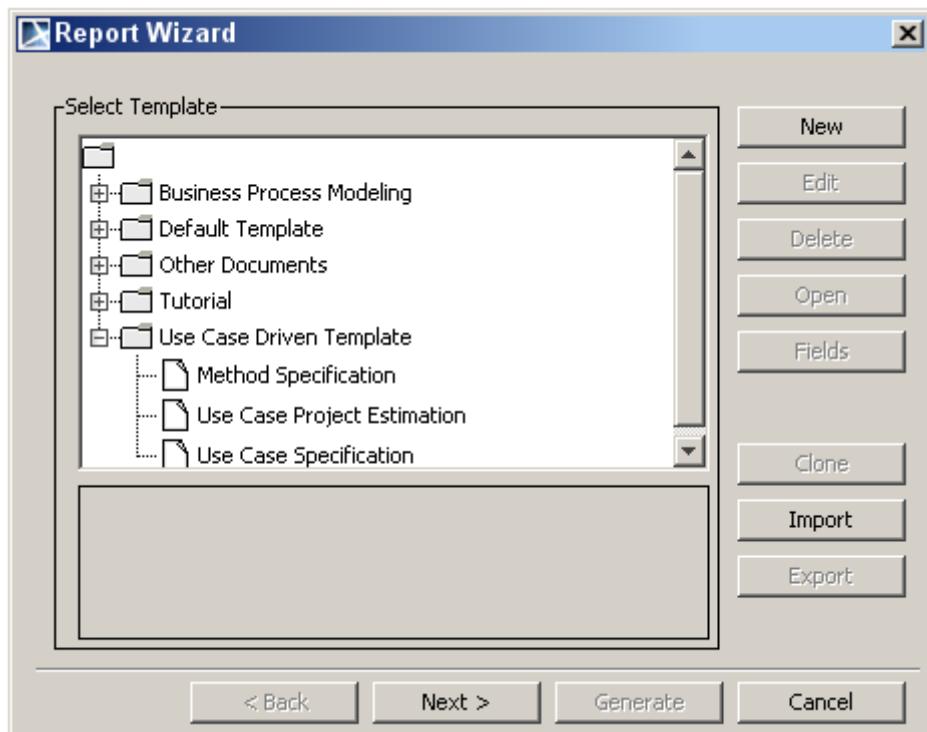


Figure 369 -- Sample of the Report Wizard window

Step 6# Using the Report Wizard window

For example this user interface model could now be exported as an image. The steps to do this are as follows:

1. Select **Save As Image** from the **File** menu.
2. In a new dialog mark **Active Diagram**.
3. In **Image File** define the location where the image should be placed.
4. Select *Joint Photographic Experts Group (*.jpg)* in **Image Format** and then press the **Save-Button**. Of course you can also take another format if you want to.

Case Study 2 - Slider Example

This case study contains step-by-step instructions showing how to create a User Interface model with Sliders. It also shows how to customize the symbol properties of User Interface components. It does not explain, however, how to create a new project and a new diagram since those are explained in “Case Study 1 - Modelling User Interface for the Report Wizard Window” on page 565.

Step 1# Create a Frame, Labels and Sliders

1. Create a new **Frame** and add title *Symbol Properties Customization*.
2. Add new **Label** to the **Frame**.
 - 2.1.1 Click the **Text** group in the diagram toolbar to expand the toolbar.
 - 2.1.2 Click the **Label** button and place the label on the diagram. Name it *Fill Color*.
 - 2.1.3 Create the remaining two labels.
3. Add **Sliders** to the **Frame**.
 - 3.1.1 Click the **Slider** button in the **Other** toolbar.
 - 3.1.2 Set property **Spacing** to 50 so as to just have three values marked in the slider.
 - 3.1.3 Set property **Knob Position** to 0 to move the knob to the left position.

10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

3.1.4 Select given value *0 0* out of property **Values** and rename it to *0 Red*.
Rename the other two given values to *50 Blue* and *100 White*.*

3.1.5 Draw two more sliders and repeat last steps setting values as shown in Figure 370 on page 569.

*** NOTE**

With regard to setting the values of a slider it is important to know that there is no empty space between *0 0*, for example, represents a new line. So, in this case, when entering a value, it should look like this:

0
0

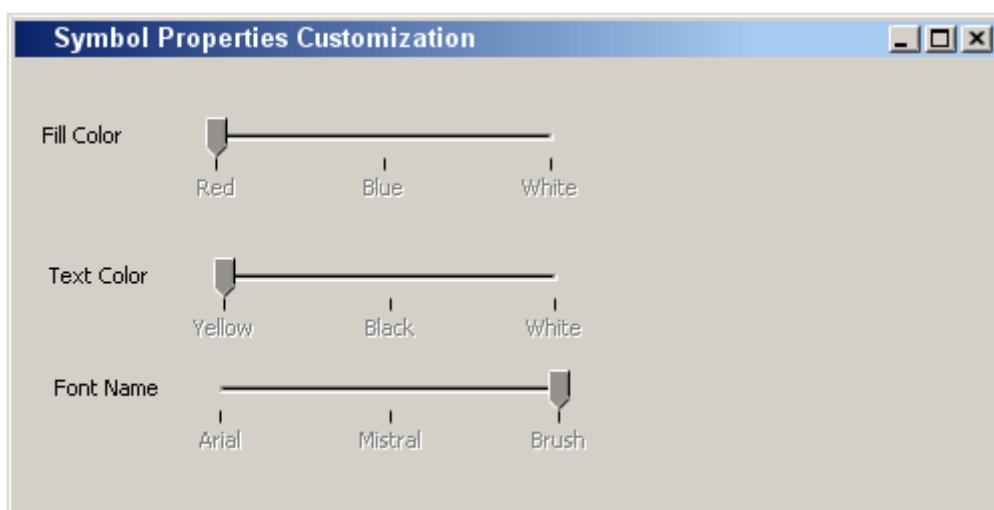


Figure 370 -- Symbol Properties Customization Frame with Added Labels and Sliders

Step 2# Add Text Fields

1. Create four **Text Fields**.

- 1.1.1 Click the **Text Field** button in the **Text** toolbar.
- 1.1.2 Name it *Red Background*.
- 1.1.3 Repeat the previous two steps for the remaining two text fields (Figure 371 on page 570).

10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

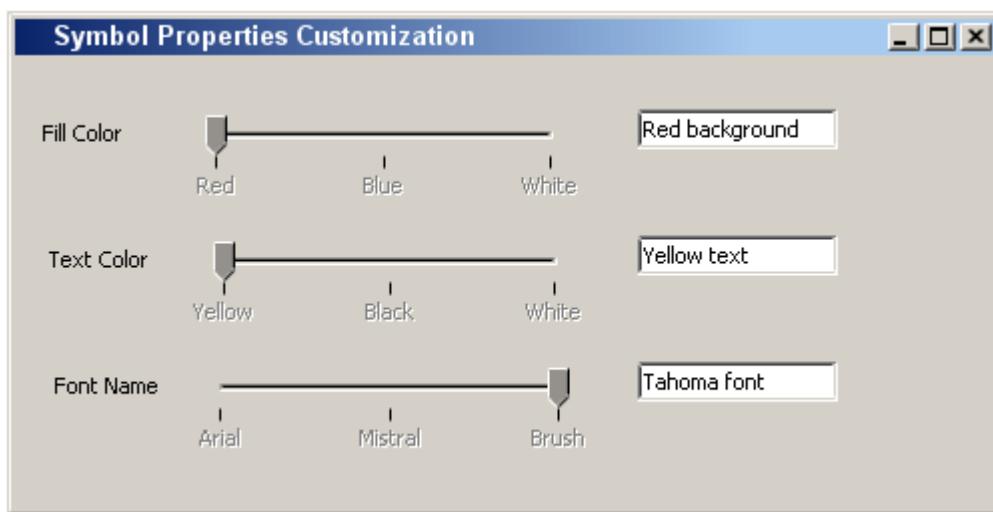


Figure 371 -- Added Text Fields to the Frame

Step 3# Edit Symbol Properties for the Text Fields

1. Edit Symbol Properties for the first text field.
 - 1.1.1 Select the first text field on the diagram surface.
 - 1.1.2 Right click and select **Symbol(s) Properties** in the shortcut menu.
 - 1.1.3 Check property **Use Fill Color** and change **Fill Color** to red.
2. Edit Symbol Properties for the remaining fields.
 - 2.1.1 For the second text field, select yellow in **Text Color**.
 - 2.1.2 For the third one, select the font name Tahoma in **Font** (Figure 372 on page 570).

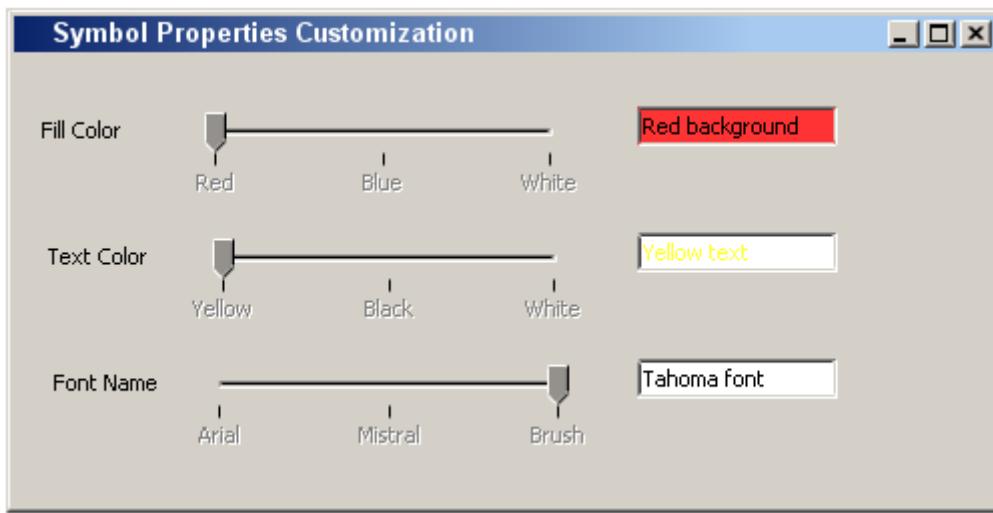


Figure 372 -- Finalized Slider Example

Case Study 3 - User Interface Prototyping Example

This case study shows how to connect several user interface models with one another and create a browsable report out of them to display the wildfowl of an application. The models should represent a test application with a Login Dialog, a Test Browser and a test with several questions. Here are step-by-step instructions for adding hyperlinks and creating browsable reports.

10 EXTENSION DIAGRAMS

User Interface Modeling Diagram

Step 1# Create first Model

1. Create a new **Package** for the model.
2. Create a model similar to the one in Figure 373 on page 571.

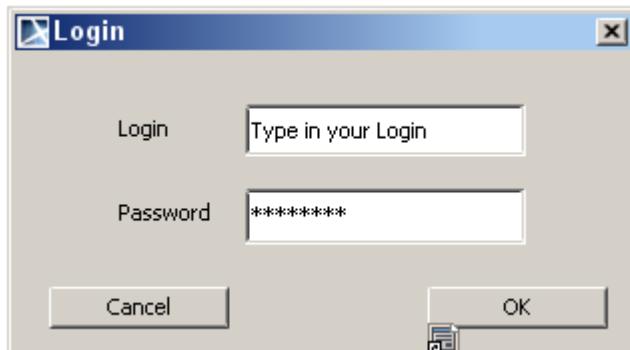


Figure 373 -- First Model of Prototyping Example - Login Dialog

Step 2# Create second Model

1. Create again a new **Package** for this model.
2. Create a model similar to the one in Figure 374 on page 571.

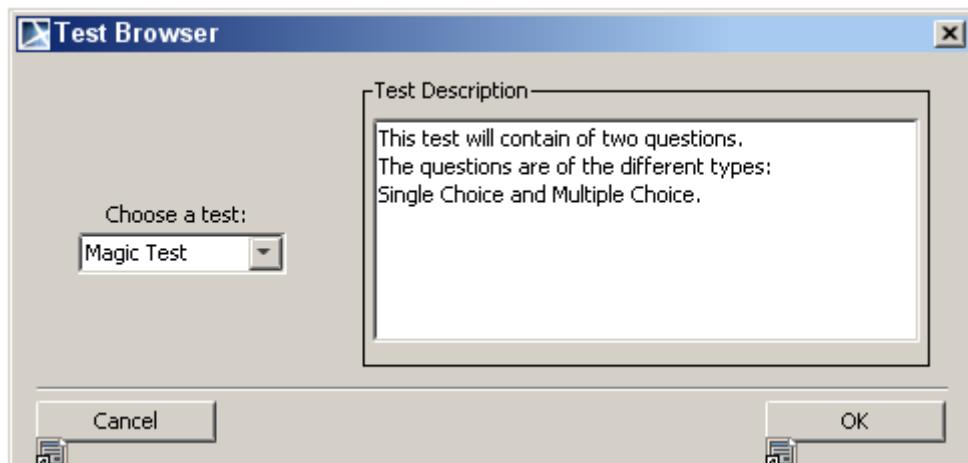


Figure 374 -- Second Model of Prototyping Example - Test Browser

Step 3# Create remaining Models

1. Create a separate **Package** for every model.
2. Build models similar to the ones shown in the full detailed sample which was mentioned in the beginning of this chapter. Or create models for questions on your own.

Step 4# Add Hyperlinks

1. Add a hyperlink.
 - 1.1.1 Open model with **Login Dialog**.
 - 1.1.2 Select the **OK** button.
 - 1.1.3 Click on smart manipulator **Hyperlinks/Go To** and select **Add Hyperlink** in the popup menu.
 - 1.1.4 Select **Element/Symbol** and click on "..." button. Browse to the **Package Test Browser**, select the **User Interface Diagram** in it and confirm

two times with OK - a diagram symbol will appear next to the **OK** button.

Double clicking on the **OK** button will lead to the other diagram.

2. Connect now all the buttons in the other diagrams with hyperlinks - you will see which component has an hyperlink because of the diagram symbol next to it.

Step 5# Create a browsable Report

1. Create a new report.

1.1.1 Select **Report Wizard** from the **Tools** menu.

1.1.2 Open the node **Default Template** in the tree and select **Web Publisher 2.0**.

1.1.3 Confirm three times by clicking **Next** until dialog appears where to add data to the report. Add then all **Packages** that contain the created diagrams and click **Next**.

1.1.4 Give a name to the output file by entering **Report file** and check the box **Display in viewer after generating report**. After clicking **Generate**, report will be built and shown in your default browser.

Content Diagram

NOTE

This feature is available in Standard, Professional, Architect, and Enterprise editions.

The content diagram is an extension of UML notation. The purpose of the content diagram is to generate or represent a project structure (diagrams) and relations between them. The content table works as a table of contents for a project.

All content diagrams have their own specifications, wherein you can specify their name, documentation, and view the relationships in which they participate. You can also add stereotypes, tagged values, and constraints.

10 EXTENSION DIAGRAMS

Content Diagram

NEW! Buttons for creating diagram shapes are grouped into categories in the content diagram pallet.

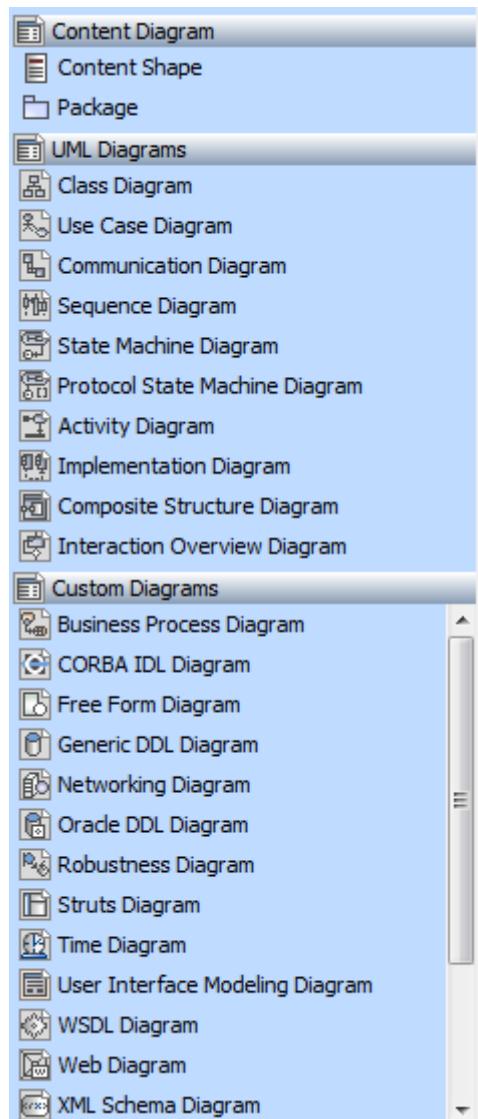


Figure 375 -- Content diagram pallet. Buttons grouped into categories

Content Diagram Elements

| Diagram Pallet Button | Button (Hot key) | Notation |
|---|--|---|
| Content Shape Creates a table of contents of all diagrams of the project. |  (C) | <p>Content of Model Data</p> <p>State Machine Diagrams</p> <ul style="list-style-type: none">  Access Control  Build House  Library Item  Pawn  Phone |
| Package |  (P) |  Order Processing |
| UML Diagrams | | |
| Class Diagram Creates a class diagram. |  |  Class Diagram |
| Use Case Diagram Creates a use case diagram. |  |  Use Case Diagram |
| Communication Diagram Creates a communication diagram. |  |  Communication Diagram |
| Sequence Diagram Creates a sequence diagram. |  |  Sequence Diagram |

10 EXTENSION DIAGRAMS

Content Diagram

| Diagram Pallet Button | Button (Hot key) | Notation |
|--|---|--|
| State Machine Diagram Creates a state diagram. |  |  State Machine Diagram |
| Protocol State Machine Diagram Creates a protocol state machine diagram. |  |  Protocol State Machine Diagram |
| Activity Diagram Creates an activity diagram. |  |  Activity Diagram |
| Implementation Diagram Creates an implementation diagram. |  |  Implementation Diagram |
| Composite Structure Diagram Creates a composite structure diagram. |  |  Composite Structure Diagram |
| Interaction Overview Diagram Creates an interaction overview diagram. |  |  Interaction Overview Diagram |
| Custom Diagrams | | |
| Business Process Diagram Creates a business process diagram. |  |  Business Process Diagram |
| CORBA IDL Diagram Creates a CORBA IDL diagram. |  |  CORBA IDL Diagram |

10 EXTENSION DIAGRAMS

Content Diagram

| Diagram Pallet Button | Button (Hot key) | Notation |
|---|---|---|
| Free Form Diagram Creates a free form diagram. |  |  Free Form Diagram |
| Generic DDL Diagram Creates a DDL diagram. |  |  Generic DDL Diagram |
| Networking Diagram Creates a networking diagram |  |  Networking Diagram |
| Oracle DDL Diagram Creates an Oracle DDL diagram. |  |  Oracle DDL Diagram |
| Robustness Diagram Creates a robustness diagram. |  |  Robustness Diagram |
| Struts Diagram Creates a struts diagram. |  |  Struts Diagram |
| Time Diagram Creates a time diagram. |  |  Time Diagram |
| User Interface Modeling Diagram Creates an user interface modeling diagram. |  |  User Interface Modeling Diagram |

10 EXTENSION DIAGRAMS

Robustness Diagram

| Diagram Pallet Button | Button (Hot key) | Notation |
|--|---|--|
| WSDL Diagram Creates a WSDL diagram. |  |  WSDL Diagram |
| Web Diagram Creates a web diagram. |  |  Web Diagram |
| XML Schema Diagram Creates an xml schema diagram. |  |  XML Schema Diagram |
| Relation Map Diagram Creates a relation map diagram. |  |  Relation Map Diagram |
| Dependency Matrix Creates a dependency matrix. |  |  Dependency Matrix |
| NEW! Generic Table Creates a generic table. |  |  Generic Table |
| Content Diagram Creates a content diagram. |  |  Content Diagram |

Robustness Diagram

NOTE This feature is available in Standard, Professional, Architect, and Enterprise editions.

The Robustness Analysis involves analyzing the narrative text of use cases, identifying a first-guess set of objects that will participate in those use cases, and classifying these objects based on the roles they play.

- Boundary or Interface objects are what actors use in communicating with the system.

10 EXTENSION DIAGRAMS

Robustness Diagram

- Entity objects are usually objects from the domain model.
- Control objects (are usually called controllers because they often are not real objects), serve as the “glue” between boundary objects and entity objects.

The Robustness analysis serves as a preliminary design within the project life cycle and provides the missing link between an analysis and a detailed design.

Four basic rules apply:

1. Actors can only talk to the boundary objects.
2. The boundary objects can only talk to the controllers and actors.
3. The entity objects can only talk to the controllers.
4. The controllers can talk to the boundary objects and entity objects, and to other controllers, but not to the actors.

Both the boundary objects and entity objects are nouns, and the controllers are verbs. Nouns cannot talk to other nouns, but verbs can talk to either nouns or verbs.

Robustness Diagram Elements

| Diagram Pallet Button/Model Element | Button (Hot key) | Notation |
|---|------------------|---------------------------------------|
| Actor An actor represents a role played by an external person, a process or a thing interacting with a system. One physical object may play several roles. | (A) | Customer |
| Boundary Actors use the boundary objects to communicate with the system (sometimes called the interface objects). A class with a stereotype "boundary" | (B) | Login Page |
| Control Serves as the "glue" between the boundary objects and the entity objects. A class with a stereotype "control" | (C) | Display Error Message |
| Entity The entity objects usually are objects from the domain model. A class with a stereotype "entity" | (E) | Account Table |
| Robustness Association The Association with a default Association End A navigability = false and Association End B navigability = true values. | (S) | Customer Login Page |

Web Diagram

NOTE This feature is available in Standard, Professional, Architect, and Enterprise editions.

The web system consists of server applications, network, communicating protocol, and the browser. Basically, a user's requests begin from starting the browser and requesting a document through a network from the server (host computer). The web server running on the host computer, catches the user's request, locates the document and delivers it to the user.

UML is a standard language for modeling software. However, modeling specific web components cannot be done by using just a standard UML. The Web-UML diagram provides an extension to the UML model, which enables developers to model web applications.

10 EXTENSION DIAGRAMS

Web Diagram

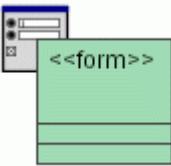
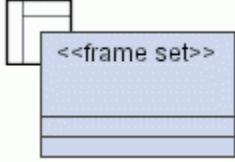
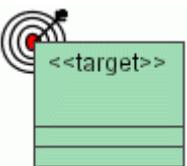
The MagicDraw Web-UML diagram includes Web-UML elements (stereotyped UML elements) for modeling: client, server pages, web form, frame classes, java script class representation and target class, and web page component.

Reference: [Building Web Applications with UML](#) by Jim Conallen Copyright ©2000 by Addison Wesley Longman, Inc.

10 EXTENSION DIAGRAMS

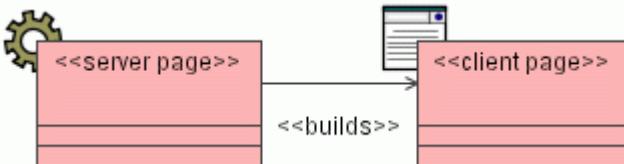
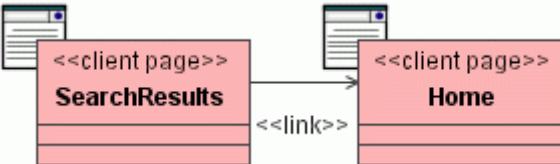
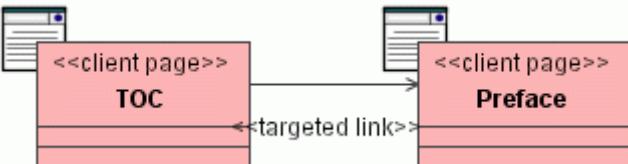
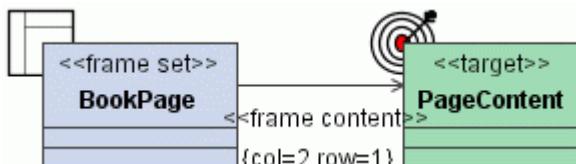
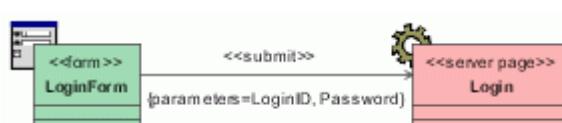
Web Diagram

Web Diagram Elements

| Diagram Pallet Button/Model Element | Button (Hot key) | Some examples |
|---|--|--|
| Client Page |  (SHIFT+G) |  |
| Server Page |  (SHIFT+S) |  |
| Form |  (SHIFT+F) |  |
| Frame |  (SHIFT+E) |  |
| Target |  (SHIFT+T) |  |
| Java Script |  (SHIFT+J) |  |

10 EXTENSION DIAGRAMS

Web Diagram

| Diagram Pallet Button/Model Element | Button (Hot key) | Some examples |
|---|--|--|
| Page |  (SHIFT+P) |  |
| Builds |  (B) |  |
| Link |  (L) |  |
| Redirect |  (T) | |
| Targeted Link |  (SHIFT+L) |  |
| Frame Content |  (SHIFT+O) |  |
| Submit |  (U) |  |
| Object |  (O) | |

| Diagram Pallet Button/Model Element | Button (Hot key) | Some examples |
|---|--|---------------|
| RMI |  (SHIFT+R) | |
| IIOP |  (SHIFT+I) | |

CORBA IDL Diagram

NOTE This feature is available in Architect and Enterprise editions.

The CORBA IDL diagram facilitates the creation of CORBA IDL elements. The following patterns are also available for CORBA IDL: Interface, Value Type, Type Definition, Sequence, Array, Fixed, Union, Enumeration, Struct, and Exception.

For more information about CORBA IDL usage in MagicDraw, see "[MagicDraw Code&DatabaseEngineering UserGuide.pdf](#)", chapter "CORBA IDL Mapping to UML".

Reference: UML™ Profile for CORBA™ Specification, Version 1.0, April 2002. http://www.omg.org/technology/documents/formal/profile_corba.htm

CORBA IDL Diagram Elements

| Diagram Pallet Button/ Model Element | Button (Hot key) |
|--|---------------------|
| CORBAModule | |
| | (M) |
| CORBA IDL Interface | |
| | (U) |
| CORBA IDL Value | |
| | (V) |
| Generalization | |
| | (G) |
| Truncatable Generalization | |
| | |
| Value Supports Generalization | |
| | |
| CORBA IDL Association | |
| | |
| Interface | |
| | (I) |

Generic DDL Diagram

NOTE This feature is available in Architect and Enterprise editions.

The DDL diagram is used to draw database definition elements. It simplifies the creation of primary key, foreign key, triggers, etc.

For the detailed description about DDL usage in MagicDraw, see ["MagicDraw Code&DatabaseEngineering UserGuide.pdf"](#), chapter "DDL Script Engineering".

Generic DDL Diagram Elements

| Diagram Pallet Button/ Model Element | Button (Hot key) |
|--|---|
| Table |  |
| | (T) |
| View |  |
| | (V) |
| Relationship |  |
| | (R) |
| Foreign Key |  |
| | (F) |
| Database |  |
| | (D) |
| Schema |  |
| | (S) |

Oracle DDL Diagram

NOTE This feature is available in Architect and Enterprise editions.

The DDL diagram is used to draw database definition elements. It simplifies the creation of primary key, foreign key, triggers and etc.

For the detailed description about DDL usage in MagicDraw, see ["MagicDraw Code&DatabaseEngineering UserGuide.pdf"](#), chapter "DDL Script Engineering".

Oracle DDL Diagram Elements

| Diagram Pallet Button/ Model Element | Button (Hot key) |
|--|---|
| Table |  |
| | (T) |
| View |  |
| | (V) |
| Materialized View |  |
| | (M) |
| Package |  |
| | (P) |
| Type |  |
| | |
| Relationship |  |
| | (R) |
| Foreign Key |  |
| | (F) |
| Scope |  |
| | |
| Database |  |
| | (D) |
| Schema |  |
| | (S) |

WSDL Diagram

NOTE This feature is available in Architect and Enterprise editions.

The WSDL diagram is used to draw WSDL elements. It allows the creation of all elements used in the wsdl file, except schema. The schema elements can be created using the XMLSchema diagram. WSDL plugin provides patterns to create binding elements. The WSDL plugin requires XMLSchema plugin.

WSDL Diagram Elements

| Diagram Pallet Button/Model Element | Button (Hot key) |
|---|---------------------|
| WSDLmessage | |
| | (M) |
| WSDLporttype | |
| | (T) |
| WSDLbinding | |
| | (B) |
| WSDLport | |
| | (SHIFT+P) |
| WSDLservice | |
| | (S) |
| WSDLdefinitions | |
| | (D) |
| WSDLtypes | |
| | (Y) |
| WSDLimport | |
| | (I) |
| Xmns | |
| | (SHIFT+L) |
| XSDnamespace | |
| | (P) |

XML Schema Diagram

NOTE This feature is available in Architect and Enterprise editions.

The purpose of the diagram is to create the structure of the xml schema file. It helps to draw the xml schema elements quickly. These elements are the stereotyped UML elements from the class and implementation diagrams.

The following patterns are also available for the XML Schema: XSDcomplexType, XSDcomposition, XSDsimpleType, XSDsimpleType(XSDlist), XSDsimpleType(XSDunion), and Simple XSDrestriction.

For the detailed description about XML Schema usage in MagicDraw, see “MagicDraw Code&DatabaseEngineering UserGuide.pdf”, chapter “XML Schema”.

Reference: <http://www.w3.org/TR/xmlschema-2/>

XML Schema Diagram Elements

| Diagram Pallet Button/ Model Element | Button (Hot key) |
|---|---|
| XSDcomplexType |  (C) |
| XSDsimpleType |  (S) |
| XSDsimpleType(XSDlist) |  (SHIFT+S) |
| XSDsimpleType(XSDunion) |  (U) |
| XSDall |  (L) |
| XSDchoice |  (O) |
| XSDsequence |  (Q) |
| XSDgroup |  (G) |
| XSDattributeGroup |  (T) |
| XSDschema |  (SHIFT+M) |
| XSDnamespace |  (P) |

10 EXTENSION DIAGRAMS

Time Diagram

| Diagram Pallet Button/ Model Element | Button (Hot key) |
|---|---------------------|
| Aggregation | (A) |
| Composition | (F) |
| Template Binding | (B) |
| XSDrestriction | (R) |
| XSDextension | (E) |
| XSDimport | (M) |
| XSDinclude | (D) |
| XSD redefine | (SHIFT+F) |
| xmns | (SHIFT+L) |

Time Diagram

NOTE This feature is available in Standard, Professional, Architect, and Enterprise editions.

A Time Diagram is an extension of UML notation. The time diagram is similar to a sequence diagram, but the model elements of the time diagram have the predefined stereotypes.

Time Diagram Elements

| Diagram Pallet Button/ Model Element | Button (Hot key) |
|---|---------------------|
| Lifeline <<CRconcurrent>> | |
| Lifeline <<SAshedList>> | |
| Message <<RTevent>>, <<CRimmediate>> | |
| Message <<CRimmediate>> | |
| Message <<SAtrigger>> | |
| |) |

Struts Diagram

NOTE This feature is available in Professional, Architect, and Enterprise editions.

A Struts Diagram is an extension of UML notation.

Struts Diagram Elements

| Diagram Pallet Button/ Model Element | Button (Hot key) |
|--|---------------------|
| Class | |
| | (C) |
| Class by Pattern | |
| | (SHIFT+P) |
| Interface | |
| | (I) |
| ActionForm class for struts | |
| | (G) |
| Action class for struts | |
| JavaServer Page for use with struts | |
| Package | |
| | (P) |
| Model | |
| | (M) |
| Interface Realization | |
| | (R) |
| Realization | |
| | (E) |
| Abstraction | |
| | (T) |

10 EXTENSION DIAGRAMS

Networking Diagram

| Diagram Pallet Button/ Model Element | Button (Hot key) |
|--|---|
| Usage |  (U) |
| Generalization |  (G) |
| Association |  (S) |
| Aggregation |  (A) |
| Composition |  (F) |

Networking Diagram

NOTE This feature is available in Standard, Professional, Architect, and Enterprise editions.

This diagram allows a visual display of the network topology. The *Networking Profile* contains stereotypes for the network description. Elements with icons can be drawn on the Diagram pane.

The Networking Diagram is commonly used to depict hardware nodes as well as the connections between them.

Networking Diagram Elements

| Diagram Pallet Button/ Model Element | Button (Hot key) |
|--|---|
| Server |  |
| Application Server |  |
| DB Server |  |
| File Server |  |
| Proxy Server |  |
| Web Server |  |
| PC |  |
| Laptop |  |
| Monitor |  |
| Fax |  |
| Plotter |  |
| Printer |  |
| Scanner |  |
| Modem |  |
| Router |  |

10 EXTENSION DIAGRAMS

Networking Diagram

| Diagram Pallet Button/ Model Element | Button (Hot key) |
|--|---------------------|
| Switch | |
| Firewall | |
| Database | |
| Program | |
| Internet Browser | |
| Document | |
| File | |
| Wireless Network | |
| Internet | |
| Building | |
| City | |
| Actor | |
| User | |
| HTTP | |
| Ethernet | |
| RMI | |

| | |
|--|---|
| Diagram Pallet Button/ Model Element | Button (Hot key) |
| Communication Path |  |

Business Process Diagram

NOTE This feature is available in Standard, Professional, Architect, and Enterprise editions.

BPMN (Business Process Modeling Notation) is the most recognized standard used by business users for an end-to-end business process modeling. MagicDraw with its business-oriented notation support has become one useful tool for both business and IT users. Working with business and IT models in one tool helps save time and solve model interchange problems.

The Business Process Diagrams convey the business processes of different participants in a modeled system. In a MagicDraw model, Process contains the Business Process Diagram, depicting it as a graph of the Flow Objects.

The BP diagrams can be used for three basic BPMN sub-models creation:

- Private (internal) business processes
- Abstract (public) processes
- Collaboration (global) processes

Within and between these three BPMN sub-models, many types of Diagrams can be created (High-level private process activities, detailed private business processes, Collaboration Processes, etc.). Since a BPMN Diagram may depict the Processes of different Participants, each Participant may view the diagram differently.

Business Process Diagram Elements

| Diagram Pallet Button/ Model Element | Button | Element Description |
|---|--------|--|
| Text Annotation | | |
| Association of Text Annotation | | |
| Start Event None | | The Start Event indicates where a particular process will start. |
| Start Event Message | | |
| Start Event Timer | | |
| Start Event Rule | | |
| Start Event Link | | |
| Start Event Multiple | | |

10 EXTENSION DIAGRAMS

Business Process Diagram

| Diagram Pallet Button/ Model Element | Button | Element Description |
|--|--------|---|
| Text Annotation | | |
| Association of Text Annotation | | |
| Intermediate Event None | | The Intermediate Events occur between a Start Event and an End Event. It will affect the flow of the process, but will not start or (directly) terminate the process. |
| Intermediate Event Message | | |
| Intermediate Event Timer | | |
| Intermediate Event Error | | |
| Intermediate Event Cancel | | |
| Intermediate Event Compensation | | |
| Intermediate Event Rule | | |
| Intermediate Event Link | | |
| Intermediate Event Multiple | | |

10 EXTENSION DIAGRAMS

Business Process Diagram

| Diagram Pallet Button/ Model Element | Button | Element Description |
|---|--------|--|
| Text Annotation | | |
| Association of Text Annotation | | |
| End Event None | | The End Event indicates where a process will end. |
| End Event Message | | |
| End Event Error | | |
| End Event Cancel | | |
| End Event Compensation | | |
| End Event Link | | |
| End Event Terminate | | |
| End Event Multiple | | |
| Task | | The Task is used when the work in the Process is not broken down to a finer level of Process Model detail. |

10 EXTENSION DIAGRAMS

Business Process Diagram

| Diagram Pallet Button/ Model Element | Button | Element Description |
|---|--------|--|
| Text Annotation | | |
| Association of Text Annotation | | |
| Loop Task | | |
| Multiple-Instance Task | | |
| Compensation Task | | |
| Collapsed Sub-Process | | A Sub-Process is a compound activity that is included within a Process. It can be broken down into a finer level of detail (a Process) through a set of subactivities. |
| Collapsed Transaction | | |
| Collapsed Loop Sub- Process | | |
| Collapsed Multiple- Instance Sub-Process | | |
| Collapsed Compensation Sub-Process | | |
| Collapsed Ad-Hoc Sub- Process | | |

10 EXTENSION DIAGRAMS

Business Process Diagram

| Diagram Pallet Button/ Model Element | Button | Element Description |
|--|--------|---------------------|
| Text Annotation | | |
| Association of Text Annotation | | |
| Expanded Sub-Process | | |
| Expanded Transaction | | |
| Expanded Loop Sub- Process | | |
| Expanded Multiple- Instance Sub-Process | | |
| Expanded Compensation Sub-Process | | |
| Expanded Ad-Hoc Sub- Process | | |

10 EXTENSION DIAGRAMS

Business Process Diagram

| Diagram Pallet Button/ Model Element | Button | Element Description |
|---|--------|---|
| Text Annotation | | |
| Association of Text Annotation | | |
| Gateway | | A Gateway is used to control the divergence and convergence of multiple Sequence Flows. |
| Data-Based XOR | | |
| Event-Based XOR | | |
| OR | | |
| Complex | | |
| AND | | |
| Sequence Flow | | A Sequence Flow is used to show the order of how activities will be performed in a Process. |
| Conditional Sequence Flow | | |
| Default Sequence Flow | | |

10 EXTENSION DIAGRAMS

Business Process Diagram

| Diagram Pallet Button/ Model Element | Button | Element Description |
|---|--------|---|
| Text Annotation | | |
| Association of Text Annotation | | |
| Compensation Association | | The Compensation Association occurs outside the Normal Flow of the Process and is based on an event (a Cancel Intermediate Event) that is triggered through the failure of a Transaction or a Compensate Event. |
| Message Flow | | A Message Flow is used to show the flow of messages between two entities that are prepared to send and receive them. |
| Horizontal/Vertical Pool | | A Pool represents a Participant in a Process. |
| Pool with Lanes | | |
| Data Object | | The Data Objects provide information about what activities are required to be performed and/or what they produce. |
| Association | | An Association is used to associate information with the Flow Objects. |
| Directional Association | | |
| Group | | A grouping of activities that does not affect the Sequence Flow. |

The Business Process diagram possesses smart layout features. For more information about these features, see “Smart Activity Diagram layout” on page 534.

Free Form Diagram

Free form diagram allows to draw various geometrical shapes. The diagram also includes shapes for drawing business flowcharts.

Relation Map Diagram

For more information about the relation map diagram, see section “Relation Map” on page 398.

Dependency Matrix

For more information about the dependency matrix, see section “Dependency Matrix” on page 425.

NEW! Generic Table

NOTE This feature is available in Standard, Professional, Architect, and Enterprise editions.

As of version 17.0 a generic table feature is introduced. The purpose of generic table is enabling the user to review the group of elements as one set in the same place. With the help of generic table you can do the following:

- Review and edit specifications of model elements in a tabular form.
- Modify selected group of element properties at once.
- Create elements of the selected types in the one table.

10 EXTENSION DIAGRAMS

NEW! Generic Table

| # | Name | PreCondition | Goal | Basic Flow of Events |
|---|----------------|--|---|--|
| 1 | >Create Course | The user has permissions to create/modify. | Create a new course | 1. Type the course information 2. Select the class where the course take place 3. Specify the schedule of the course |
| 2 | >Create Class | The user has permissions to create/modify. | Create a new class | 1. New class creation form is opened 2. Enter all class information (class name, schedule) |
| 3 | >Create User | The user has permissions to create/modify. | Create a new user | 1. Type the information of the user 2. Select the role type 3. Specify the permissions for the new user |
| 4 | Modify User | The user has permissions to create/modify. | Modify selected user | 1. Open the user information form 2. Modify the information of the selected user |
| 5 | Modify Course | The user has permissions to create/modify. | Modify information of the selected course | 1. Open course list 2. Select course 3. Edit information of the selected course |
| 6 | Modify Class | The user has permissions to create/modify. | Modify selected class | 1. Open class list 2. Select class 3. Edit information of the selected class |

Figure 376 -- An example of generic table

TIP!

You can also read about the generic table feature and analyze given examples within the Generic table project sample.

To open the sample, do either:

- On the Welcome screen, select **Samples > Diagrams > Generic table**.
- Go to <MagicDraw installation directory>\samples\diagrams and open the *generic table.mdzip* file.

All the examples given in this section are based on the data from this sample.

Creating Generic Tables

You could create a generic table in one of the ways that are suitable for creating any diagram in MagicDraw (see the procedure "To create a new diagram" that is described in the section "Diagram Basics" on page 167).

Moreover, this section describes several ways for creating a generic table in more detail. They are as follows:

- **Creating an empty generic table without using any wizard.** Once created the table can be filled with data using the generic table environment capabilities. For more information see procedure "To create an empty generic table" on page 607.
- **Creating a generic table using the diagram creation wizard.** Once created the table is already filled with data. For more information see "To create a generic table using the diagram creation wizard" on page 609.
- **Creating a generic table for a set of selected elements using the diagram creation wizard.** Note that element types, elements, and columns for the generic table will be selected automatically according to the set of selected elements. For more information see "To create a new generic table for a set of selected elements" on page 611.

When creating a generic table you should specify the following:

1. The name of a table and the owner package should be denoted first of all. This is mandatory.
2. Element types that will be used in the table should be defined. This is optional.
3. Elements corresponding to the element types should be selected. This is optional.

10 EXTENSION DIAGRAMS

NEW! Generic Table

4. Table columns should be specified. Columns represent element properties. This action is also optional. The creation of the table could be finished without specifying columns.

You can modify the table after it is created. For more information see “Modifying generic table” on page 612.

To create an empty generic table

1. Open the **Create Diagram** dialog. It could be done in one of the following ways:
 - On the **Custom diagrams** toolbar, click the  button.
 - From the main menu, select **Diagrams > Custom Diagram > Generic Table** and click **Add**.
 - On the content diagram pallet, click the **Generic Table** button.
 - Right-click the element or its symbol, that is a possible diagram owner, and from the shortcut menu select **New Diagram > Custom Diagram > Generic Table**.
2. Type the name of the table and select the owner for it.

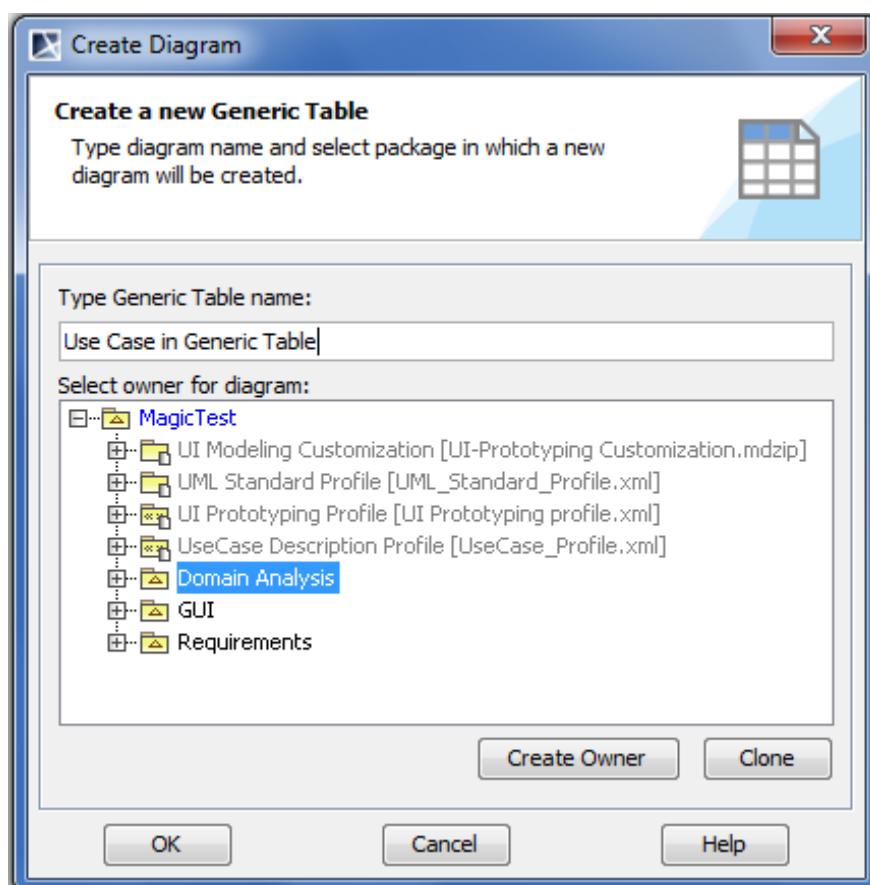


Figure 377 -- Specifying name for a new generic table and selecting the owner

The owner for the generic table could be one of the following:

- Selected in the presented model tree.
- Created as a new owner by clicking the **Create Owner** button on the **Create Diagram** dialog (the list of available elements will be provided).
- Cloned as a copy of the possible diagram owner with all its content by clicking the **Clone** button on the **Create Diagram** dialog.

NOTE If a generic table is created using the element’s shortcut menu in the Model Browser, the element in which it has been created is automatically assigned as the owner of this table.

10 EXTENSION DIAGRAMS

NEW! Generic Table

3. Select element types.

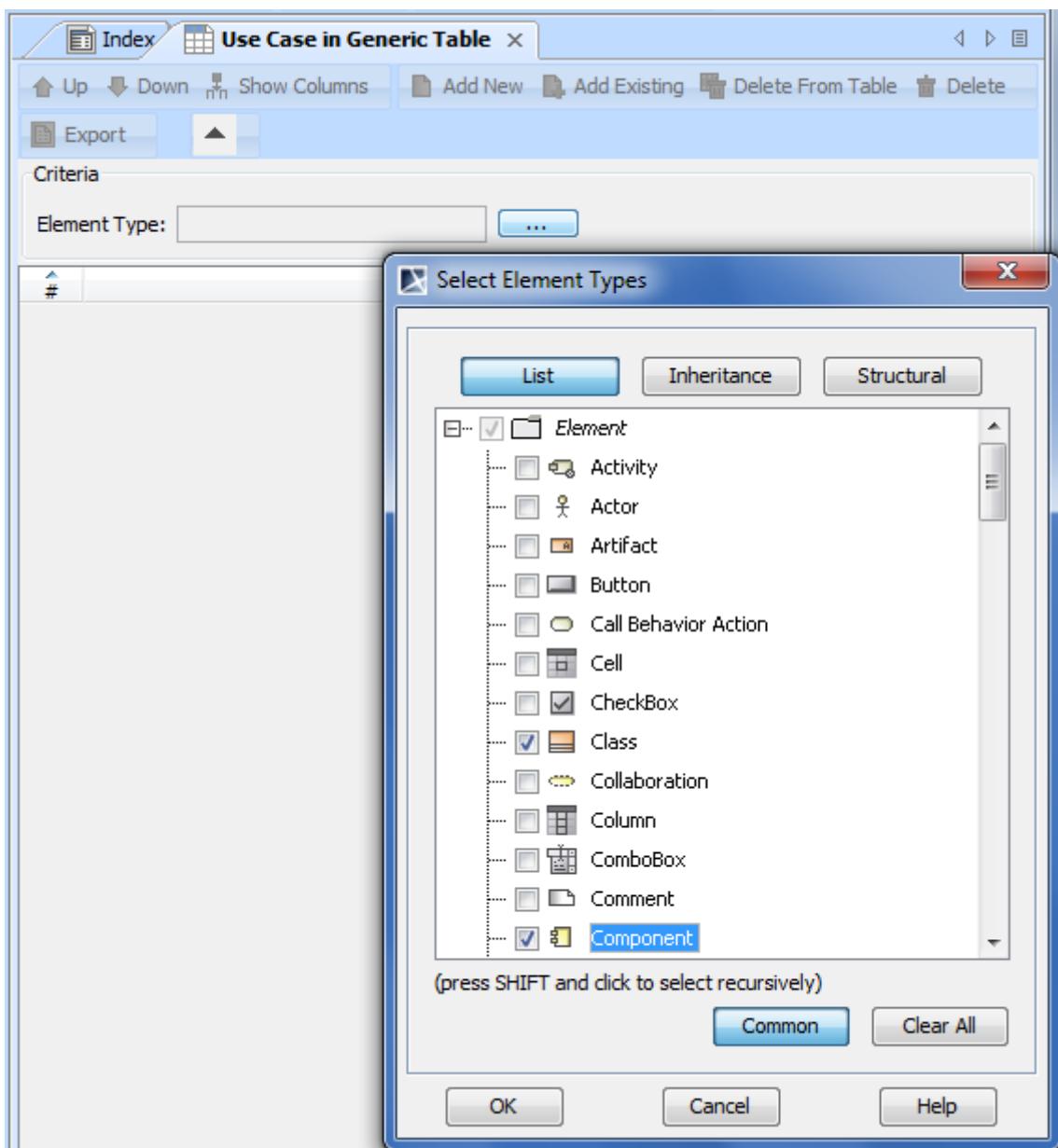


Figure 378 -- Selecting element types for the general table

The list of elements and stereotypes is presented in the **Select Element Types** dialog. Select desired element types and/or stereotypes. This action is optional, thus you can continue the creation without selecting any type or stereotype. You will be able to modify selected element types when the generic table is created.

Selected element types will be displayed in the **Element Type** box in the diagram pane.

The **Select Element Types** dialog contains the following buttons:

| Button name | Function |
|--------------------|---------------------------------------|
| List | Lists types in alphabetical order. |
| Inheritance | Lists types according to inheritance. |
| Structural | Lists types in structural way. |
| Common | Lists only common types. |
| Clear All | Clears all selections. |

10 EXTENSION DIAGRAMS

NEW! Generic Table

4. Add elements to the table using the **Add New** or **Add Existing** buttons. For more information about adding elements to a generic table, refer to the procedure “To add a new element” on page 613.
5. Specify columns for the table using the **Show Columns** button. For more information about specifying columns, refer to the procedure “To add or remove columns” on page 614.

To create a generic table using the diagram creation wizard

1. Open the **Generic Table Wizard** dialog. It could be done in one of the following ways:
 - On the **Diagrams** menu, click **Diagram Wizards > Generic Table Wizard**.
 - On the **Analyze** menu, click **Model Visualizer**.
2. Type the name of the table and select the owner for it.

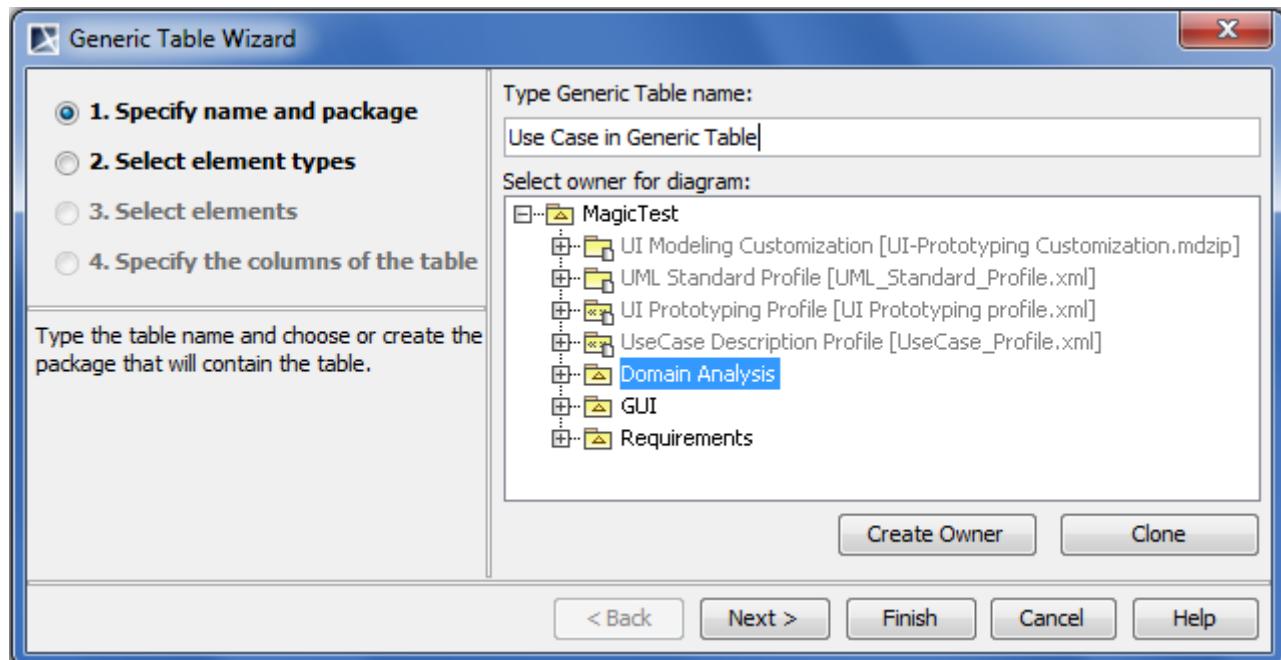


Figure 379 -- Generic table creation wizard. Specifying name for a new generic table and selecting its owner

For the instructions on how to select the generic table owner see [step #2](#) of the procedure “To create an empty generic table” on page 607.

10 EXTENSION DIAGRAMS

NEW! Generic Table

3. Select element types.

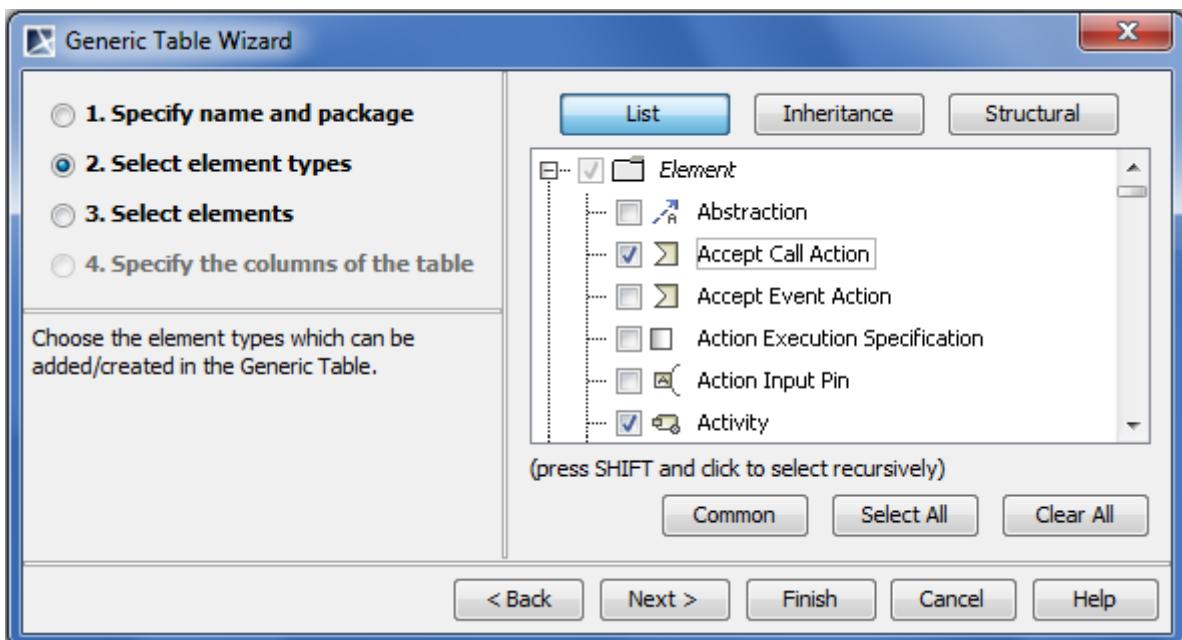


Figure 380 -- Generic table creation wizard. Selecting element types

For the instruction on how to select element types see [step #3](#) of the procedure "To create an empty generic table" on page 607.

4. Select elements.

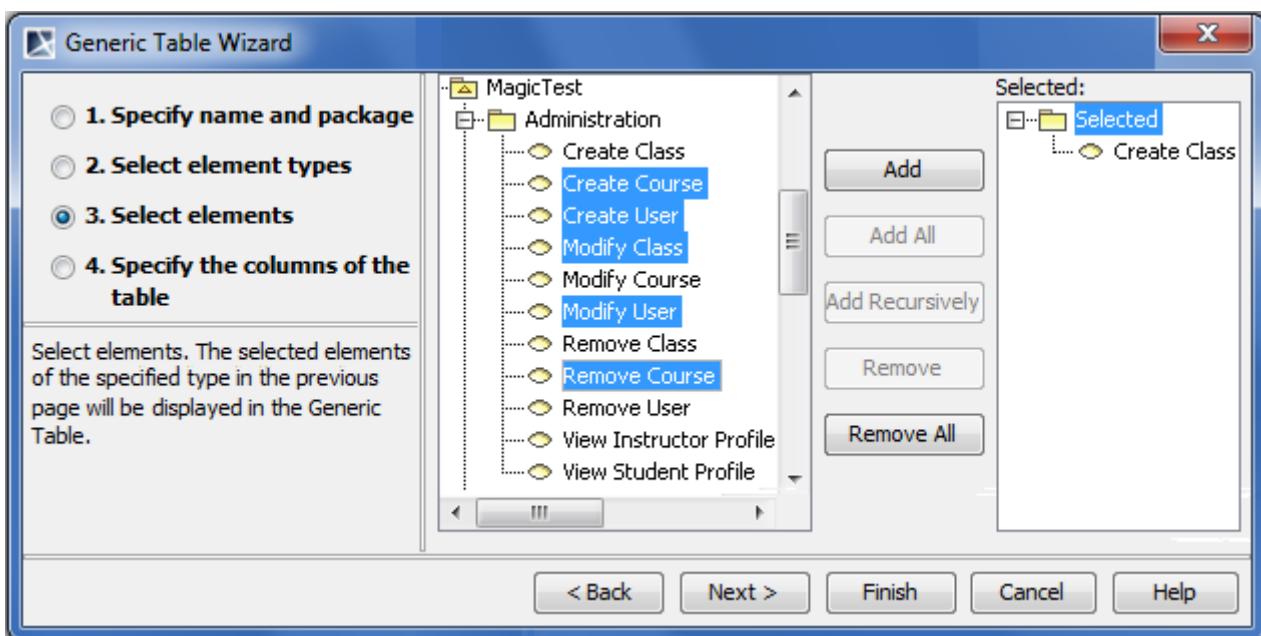


Figure 381 -- Generic table creation wizard. Selecting elements

Select elements from the model tree and add them into the **Selected** list. For more information about the element selection and dialog buttons see the section "Elements multiple selection" on page 281.

NOTES

You may select any element from the model tree, though only elements of selected types will be added into the generic table.

If the generic table creating wizard is opened from one or more elements' shortcut menu, the element or the set of elements will be added to the **Selected** list automatically.

10 EXTENSION DIAGRAMS

NEW! Generic Table

5. Specify the columns of the table.

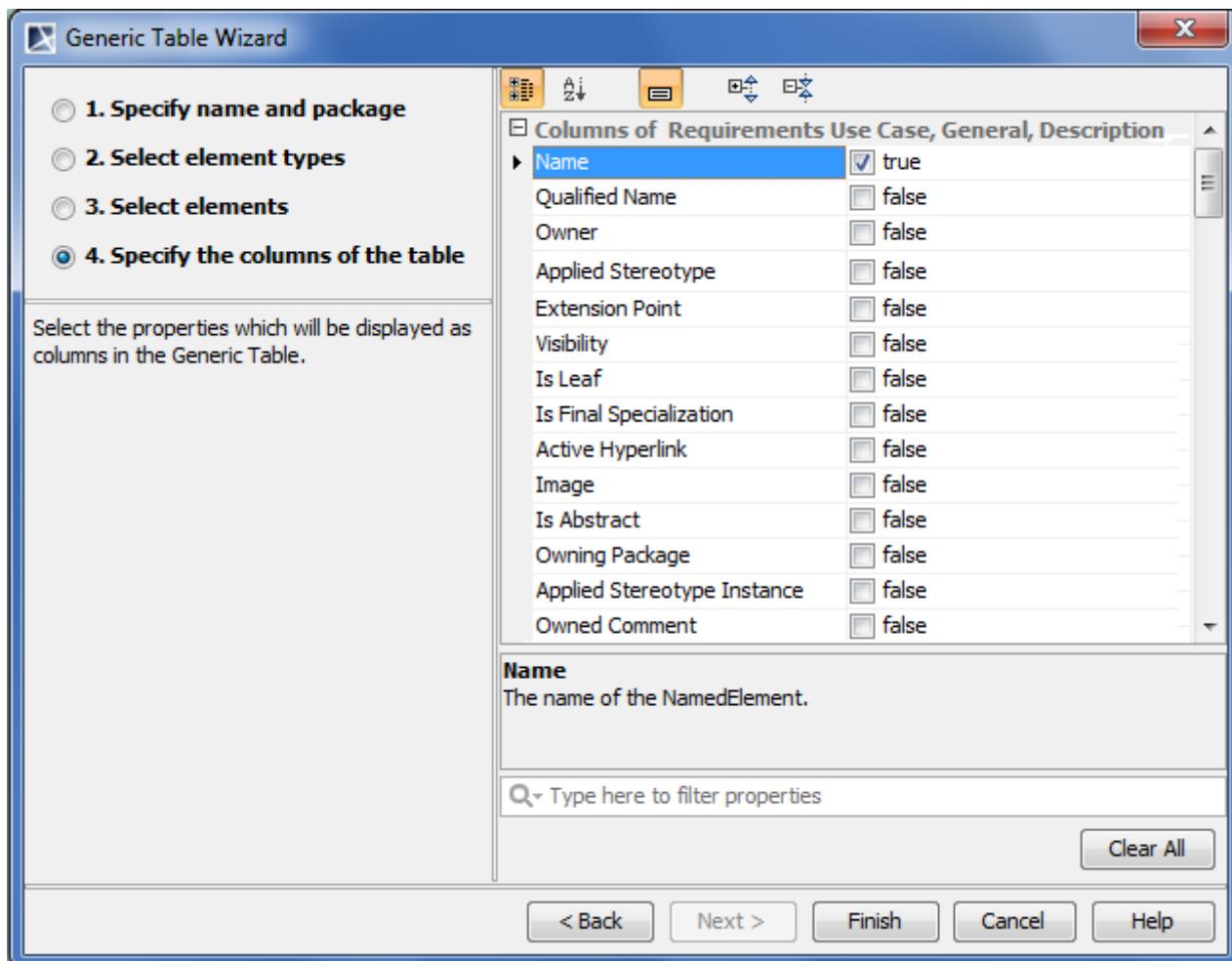


Figure 382 -- Generic table creation wizard. Selecting columns

The list of all available properties corresponding to selected element types is presented in the dialog. If selected element types have tags, they are also displayed in the list. All selected properties will be displayed as columns in the generic table. The **Name** property is selected automatically, all other properties should be selected by the user.

Select properties you need to see as columns to finish the table.

To create a new generic table for a set of selected elements

1. Select a set of elements you want to add to the generic table.
2. From the shortcut menu select **Tools > Generic Table Wizard**.
3. Follow steps described in section "To create a generic table using the diagram creation wizard" on page 609 but note that element types, elements, and columns will be selected automatically according to the set of selected elements.

Using Generic Tables

There is an ability to modify a generic table after it is created. Features for a table modification and working with a generic table are described in the following subsections:

- ["Setting detailed column names" on page 612](#)
- ["Modifying generic table" on page 612](#)
- ["Manipulations in generic table" on page 615](#)

Setting detailed column names

Column names in the generic table heading are set automatically, and they are element property names. You can not change column names.

If an element has some properties with the same names in a heading (for example, in associations or if a tag of stereotypes is selected as a column), the detailed column names could provide more precise information.

To see detailed column names

- In the generic table from the table shortcut menu, select **Table Options > Show Detailed Column Name**. The group name (between brackets), to which the selected property belongs, and/or the stereotype name (just before the property name) will be displayed in the column heading.

The following picture gives the example of three different columns with the same name for the association element: **Name**, **Name (Role of A)**, and **Name (Role of B)** and a column for the class element with the stereotype «Teacher». **Name (Tags)**. If the command **Show Detailed Column Name** were not selected, there would be four columns with same headings, i.e., **Name**.



| Criteria | | Element Type: | Documentation | Name (Role A) | Name (Role B) | «Teacher»,Name (Tags) |
|----------|----------------|---------------------------------------|---------------|---------------|---------------|-----------------------|
| # | Name | | | | | |
| 1 | / instantiates | | hasCourses | takePlace | | |
| 2 | Course | The unit of the teaching information. | | | | Brigit Smith |
| 3 | Class | The room where courses take place. | | | | |

Figure 383 -- An example of detailed column names in a generic table

Modifying generic table

To sort data

- Click the header of the column by which you want to sort table data. A small arrow appears on that column header. This arrow shows, how records are sorted: ascending or descending.



Figure 384 -- Table sorting

Rows are renumbered automatically after the sorting.

10 EXTENSION DIAGRAMS

NEW! Generic Table

To add or remove element types

1. Click the “...” button in the **Criteria** area (see the picture below). The dialog with the element and stereotype list will open.

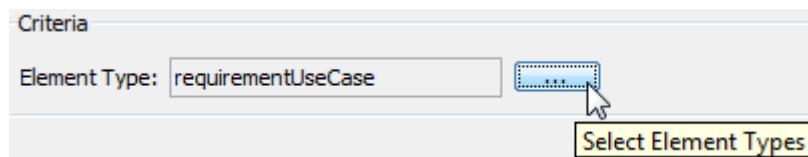


Figure 385 -- Selecting element types

2. Do one of the following:

- To add element types select the appropriate check boxes in the list.

IMPORTANT! Properties corresponding to the selected types will be added to the list of available columns.

- To remove element types click to clear appropriate check boxes in the list.

IMPORTANT! Properties corresponding to the unselected types will be removed from the list of available columns.

To add a new element

IMPORTANT There should be at least one element type selected in the **Element Type** box to add a new element.

1. In the generic table toolbars, click the **Add New** button. If there is more than one element type selected, a submenu with the list of available element types will open.

NOTE Only element types available to create in a possible diagram owner which contains the generic table will be displayed in the submenu.

2. Select an element type. The element of the selected type will be added to the last row of the table and to the model.

3. Name the new element in the table.

10 EXTENSION DIAGRAMS

NEW! Generic Table

To add an existing element from the model

1. In the generic table toolbars, click the **Add Existing** button. The **Select Element** dialog will open.

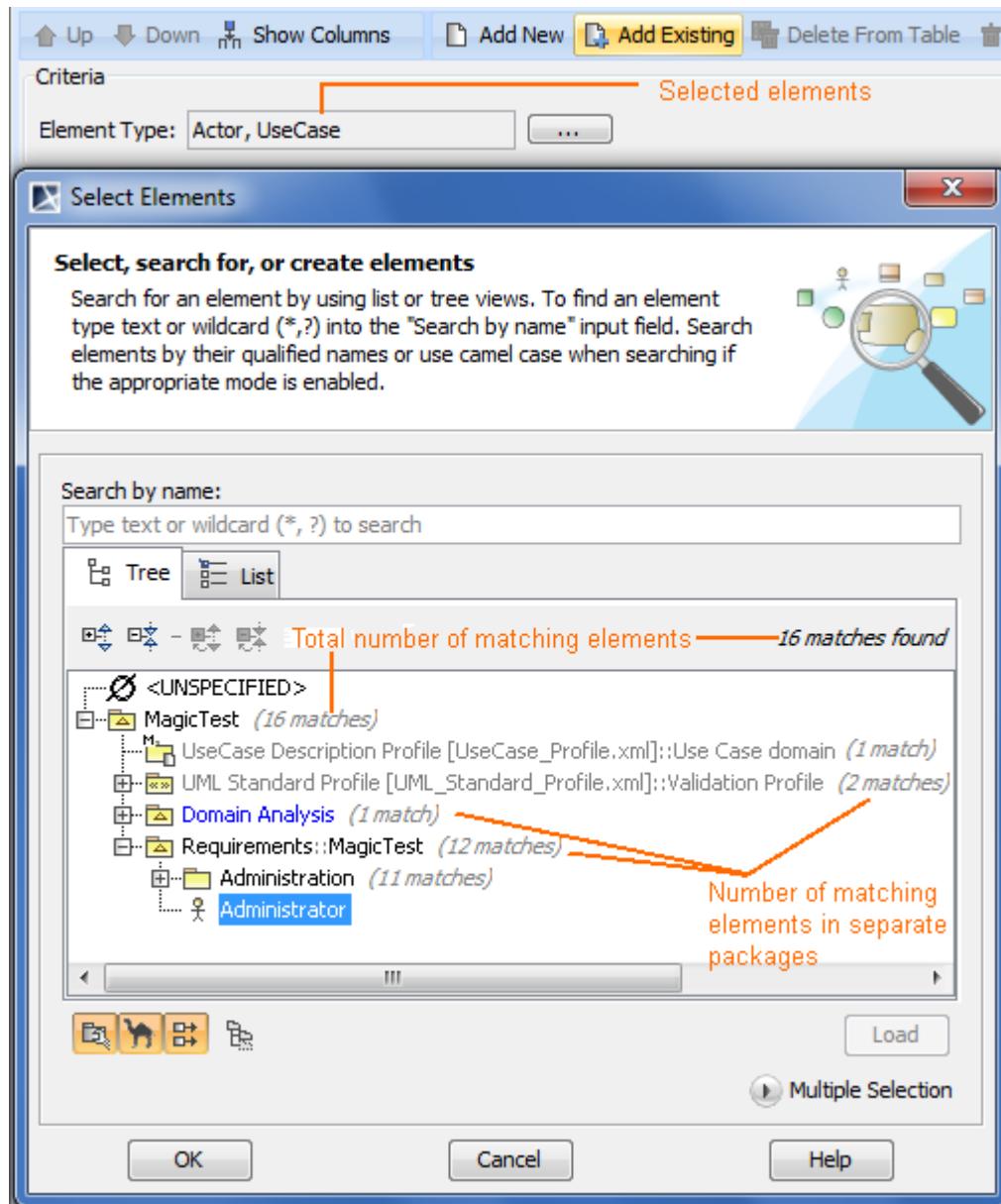


Figure 386 -- Adding existing elements to generic table

2. Select the element you want to add to the generic table. This element and its name will be added to the table as the last row.

TIP! Use the **Multiple Selection** mode to add more than one element at a time.

To add or remove columns

1. To open the available columns list:

- On the generic table toolbars, click **Show Columns** to open the submenu with common properties corresponding to element types that have been selected for the generic table.

10 EXTENSION DIAGRAMS

NEW! Generic Table

- On the generic table toolbars, click **Show Columns > Select Columns** to open the dialog with all properties, including tags of stereotypes corresponding to selected element types.

2. Do either:

- Select the properties you want to see as columns in the table.
- Unselect the properties you do not want to see as columns in the table.

To edit element's property in a cell

NOTE

The property can be editable if it is editable in the element specification window and if it is not locked in teamwork project.

Click the cell you want to edit. The edit mode will be turned on in the cell.

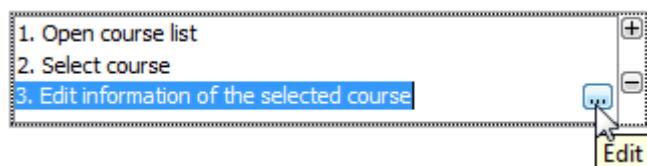


Figure 387 -- Editing element properties in a cell

- Click the “...” button to edit a property.
- Click the “+” button to add a new item. This button is displayed if adding items is available.
- Click the “-” button to remove the selected item. This button is displayed if removing items is available.

IMPORTANT! All changes made in the generic table are saved in the model.

Manipulations in generic table

To open the element's Specification window, do either

- Double-click a non-editable cell in the element's row, if there is such cell.
- Right-click on a cell and from the shortcut menu select **Open Specification**. Then:
 - If the selected cell does not refer to any element, the Specification window of the element in the row will open.

10 EXTENSION DIAGRAMS

NEW! Generic Table

- If the selected cell refers to some elements, the submenu with the list of appropriate elements will open. Choose the element whose Specification window you want to open,

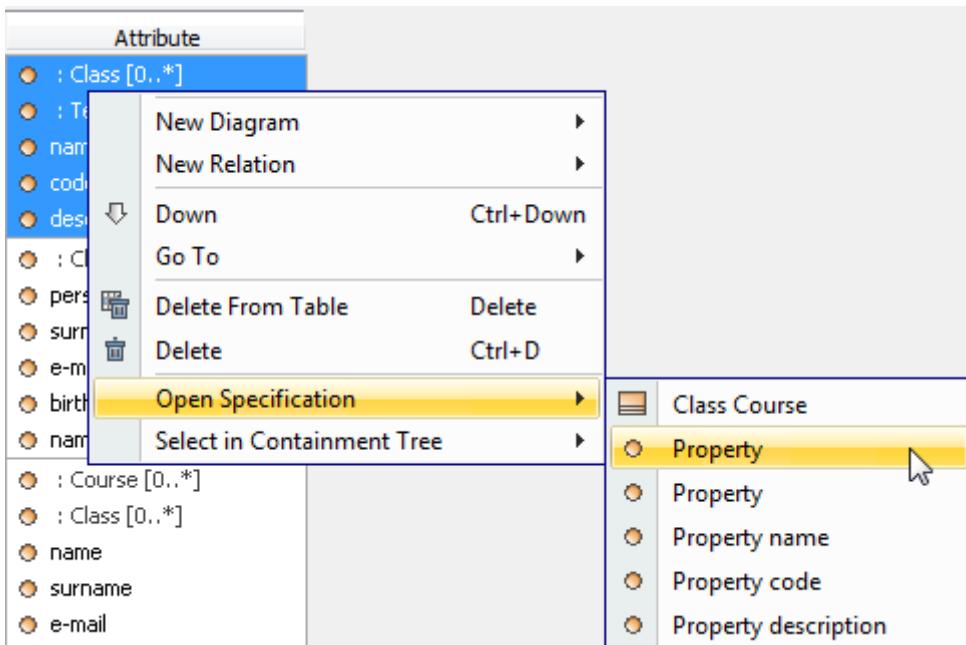


Figure 388 -- The Open Specification submenu

To add an element using the drag-and-drop feature

IMPORTANT! Elements you want to add should correspond to selected element types.

- Select an element in the Model Browser and drag it to the generic table. The element will be added to the generic table recursively.

NOTE If you drag a package, all its content will be added to the generic table.

To order columns using the drag-and-drop feature

- Click the column header and drag it to a desired place. Only the first column can not be moved.

Generic Table Environment

The generic table environment elements such as toolbars and shortcut menus are described in this section. There are the following subsections:

- ["Generic table toolbars" on page 616](#)
- ["Generic table shortcut menu" on page 617](#)
- ["Element row shortcut menu" on page 619](#)

Generic table toolbars

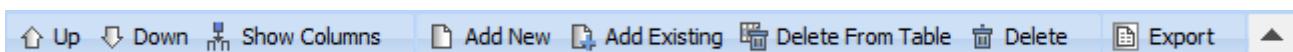


Figure 389 -- Generic table toolbars

10 EXTENSION DIAGRAMS

NEW! Generic Table

Buttons displayed in Figure 389 on page 616 are intended to be used for editing the generic table.

| | |
|-------------|--|
| NOTE | If generic table toolbars are inactive within a teamwork project, try to lock the table for edit (make sure you have the right to edit model of this project). |
|-------------|--|

| Icon (Hot key) | Name | Function |
|---|-----------------------------|---|
|  (INSERT) | Add New | Creates a new element in the table. This element will be added to the model also. If there are available several element types, a shortcut menu with available types will open. You will be able to select an element of the type you need. |
|  (CTRL+INSERT) | Add Existing | Adds an element from a model. The Select Element dialog will open. Only elements of the type defined in a generic table will be listed. |
|  (DELETE) | Delete From Table | Removes the selected element from the table. |
|  (CTRL+D) | Delete | Removes the selected element from both the table and the model. |
|  (CTRL+UP) | Up | Moves the selected element up one row. All elements are renumbered automatically after the moving is done. |
|  (CTRL+DOWN) | Down | Moves the selected element one row. All elements are renumbered automatically after the moving is done. |
|  n | Show Columns | Opens the list of available columns. You can select columns either to show or to hide in the table. |
|  e | Export | Exports the content of the table in the *.html or *.csv formats. |
|  ▲ ▼ | Suppress/expand filter area | Hides or shows the Criteria area. |

Generic table shortcut menu

To open the table shortcut menu, do either

- Right-click the empty space of the table.
- Right-click the empty space of the generic table toolbar.

10 EXTENSION DIAGRAMS

NEW! Generic Table

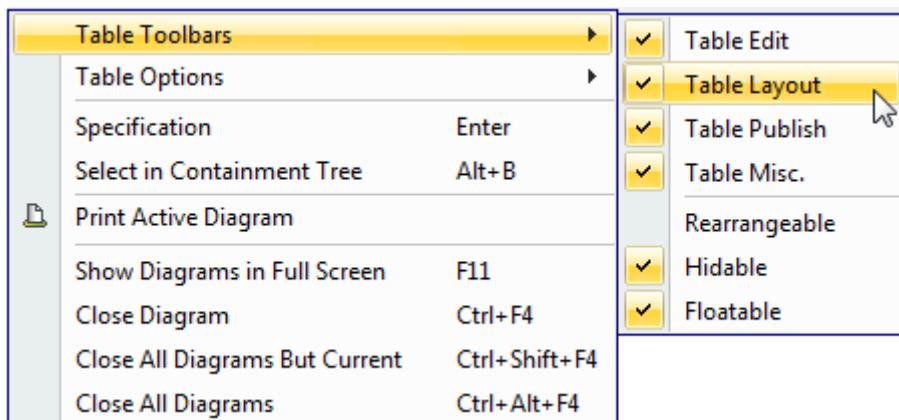


Figure 390 -- Generic table shortcut menu

| Command (Hot Keys) | Description |
|----------------------------------|--|
| Table Toolbars | |
| Table Edit | Select to display or clear to hide these generic table toolbar buttons: <ul style="list-style-type: none"> ● Add New ● Add Existing ● Delete From Table ● Delete |
| Table Layout | Select to display or clear to hide these generic table toolbar buttons: <ul style="list-style-type: none"> ● Up ● Down ● Show Columns |
| Table Publish | Select to display or clear to hide the Export button in the generic table toolbar. |
| Table Misc. | Select to display or clear to hide the ▲ Suppress/ ▾ Expand filters area button in the generic table toolbar. |
| Rerrangeable | Select to make generic table toolbars rearrangeable or clear the selection to lock toolbars in their current position. |
| Hidable | Select to make available or clear the selection to make unavailable the following commands in the generic table shortcut menu: <ul style="list-style-type: none"> ● Table Edit ● Table Layout ● Table Publish ● Table Misc. |
| Floatable | Select to allow toolbar dragging to any desirable position within the application window. Clear the selection to forbid toolbar dragging outside the generic table toolbars area. |
| Table Options | |
| Show Detailed Column Name | Select to show the detailed information such as stereotype or group name in a column heading. Clear the selection to hide the detailed information. |

10 EXTENSION DIAGRAMS

NEW! Generic Table

| Command (Hot Keys) | Description |
|---|---|
| Lock Diagram for Edit/Unlock Diagram | Locks/unlocks the table for editing. NOTE: This command is available when working on a teamwork project. For more information about locking elements please refer to the section "Locking Model Elements for Editing" in " MagicDraw Teamwork User-Guide.pdf ". |
| Specification (ENTER) | Opens the table's Specification window. |
| Select in Containment Tree (ALT+B) | Selects the table in the Containment tree of the Model Browser. |
| Print Active Diagram | Prints a table. |
| Show Diagrams in Full Screen (F11) | Shows a table in the full screen mode. Click Close Full Screen to return to the previous view. |
| Close Diagram (CTRL+F4) | Closes an active table. |
| Close All Diagrams But Current (CTRL+SHIFT+F4) | Closes all diagrams except an active one. |
| Close All Diagrams (CTRL+ALT+F4) | Closes all opened diagrams. |

Element row shortcut menu

To open the element shortcut menu

- Right-click the element row in the generic table.

| Command (Hot Keys) | Description |
|---|---|
| New Diagram | Opens the list of available diagrams to create a new diagram for the selected element. The new diagram will be created in the selected element. |
| New Relation | Opens the list of available relations to create a new relation for the selected element. |
| Up (CTRL+UP) | Moves the selected element up one row. All elements are renumbered automatically after moving is done. |
| Down (CTRL+DOWN) | Moves the selected element down one row. All elements are renumbered automatically after moving is done. |
| Go To | Finds and navigates either to a diagram, wherein the element shape is drawn, or to a hyperlinked object. |
| Delete From Table (DELETE) | Removes the selected element from the table. |
| Delete (CTRL+D) | Removes the selected element from both the table and the model. |
| Open Specification | Opens the element's Specification window. |
| Select in Containment Tree (ALT+B) | Selects the element in the Containment tree of the Model Browser. |

11 MODEL ELEMENTS

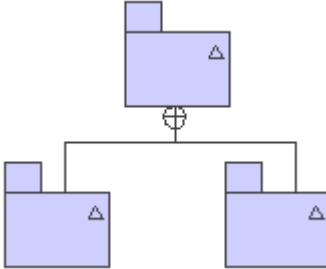
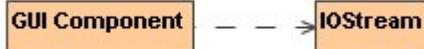
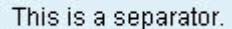
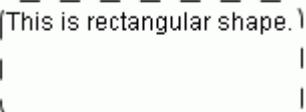
Common Model Elements in the Diagrams

The following symbols are only graphical symbols. They do not possess any data.

| Model elements | Button | Function | Notation |
|-----------------------------|-----------|--|---|
| Text Box | | Type texts directly on the diagram. | This is a text box . |
| Text Box (HTML text) | (X) | | |
| | | | |
| | (SHIFT+X) | | |
| Note | | A graphical symbol containing a textual information. Use a note to add any information needed for your diagram. Usually the note is connected to another symbol using an anchor line. | |
| Note (HTML text) | (N) | | |
| | | | |
| | (SHIFT+N) | | |
| Comment | | A graphical symbol, which gives an ability to display different remarks on the diagrams. | |
| Comment (HTML text) | | | |
| Anchor | | Relates symbols and notes. Use an anchor to relate any symbol to a note or comment. The style of the anchor can be changed to rectilinear, oblique, or bezier. For a detailed description on editing the appearance of paths, see "Working with Paths and Relationships" on page 198. | <p>The diagram illustrates the use of an anchor. A rectangular note box contains the text: "Only one solver of each kind may exist at one time". An anchor line originates from the bottom right corner of the note box and points to a rectangular symbol labeled "Solver". The anchor line is labeled "anchor to note". Above the note box, the word "note" is written in blue.</p> |

11 MODEL ELEMENTS

Common Model Elements in the Diagrams

| Model elements | Button | Function | Notation |
|--|---------------|---|--|
| Constraint NOTE To select a constraint, right-click the Note Anchor button. | (SHIFT+H) | The presentation of a constraint between two graphical symbols. |  |
| Containment | SHIFT+C | Shows a class, package or other model elements declared within another model element. |  |
| Dependency | (SHIFT+D) | Indicates a semantic relationship between two or more model elements. |  |
| Image Shape | (I) | Provides a simple and quick way to insert pictures into a diagram. They can be logo, graph, table, etc. | |
| Separator | (W) | Separates different parts of a diagram with a horizontal line. |  |
| Rectangular Shape | | Separates different parts of a diagram with a rectangular shape. |  |

Note, Comment

A note is a graphical symbol containing a textual information. It is used to add any information needed for your diagram. A note is usually connected to another element symbol using an anchor line.

A comment is a graphical symbol used to display different remarks on the diagrams.

To change the text display mode

1. Connect a note to an element symbol with an anchorlink.
2. From the note shortcut menu, select **Text Display Mode** and then **Show Text**, to display the text that is added to the note. Select **Show Documentation**, to display a documentation of the element on the note.

11 MODEL ELEMENTS

Common Model Elements in the Diagrams

To show the constraints and/or tagged values on the note

1. Connect a note to an element, the constraint and/or tagged value of which is specified, by an anchor link.
2. From the note shortcut menu, select **Show Constraint** and/or **Show Tagged Value**. Separate compartments with the constraint and tagged values will be displayed on the note.

To remove a line between the compartments on the note

1. From the note shortcut menu, select **Symbol(s) Properties**.
2. Clear the **Show Line Between Compartments** check box in the open dialog.

Anchor

Use anchors to relate symbol to a note or comment. You can change the anchor style to rectilinear, oblique, or bezier.

To delete an anchor to the comment together with the annotated element

- Select the anchor and press CTRL+D. The attached model element will be removed from Annotated Element property.

Constraint path

The constraint for two graphical symbols (such as two classes or two associations) is shown as a dashed arrow from one element to the other. The constraint is labeled by the constraint string (in braces {}). The direction of the arrow represents a relevant information within the constraint. The client (the tail of the arrow) is mapped to the first position and the supplier (arrowhead) is mapped to the second position in the constraint.

To add a constraint expression to the constraint path

From the constraint path shortcut menu, choose **Select Constraint** and select one of the following:

- Select <new>. The **Select Extension Element Storage** dialog box opens. Select the folder where the constraint will be stored and click **OK**. The **Constraint Specification** dialog box opens. Fill in the dialog box fields.
- Select an already created constraint from the list.

Image Shape

The Image Shape provides a simple and quick way to insert a picture into a diagram. This can be logo, graph, table, etc. The preferred shape size after the insertion is the actual image size.

To insert a picture on the Diagram pane

1. Click the **Image Shape** button on the Common toolbar. The **Open** dialog box opens.
2. Select an image in *.gif, *.jpg, *.jpeg, *.svg, or *.png format and click **Open**.

Separator

You may use a horizontal separator to rule off different parts of a diagram.

11 MODEL ELEMENTS

UML Extension Elements

To set the text position of a separator name

- From the separator shortcut menu, select **Text Position**, and then select the text position you need: **Left** (default), **Center**, or **Right**.
- Set the text position in the **Project Options** dialog box.

To set a separator line style (dashed or solid)

- From the separator shortcut menu, select **Line Style**, and then select the style you need: **Dashed** (default) or **Solid**.

Documentation

Define a documentation for various documents in the **Comment Specification** dialog box. You can also define a comment stereotypes, tagged values, and constraints.

To open the **Comment Specification** dialog box

- In the element **Specification** dialog box, click **Documentation/Hyperlinks**.
- Expand the **Documentation/Hyperlinks** branch in the **Specification** tree and double click **Comment**.

To display the documentation on the Diagram pane

- Draw a note on the Diagram pane.
- Using an Anchor to Note, connect the note to the model element containing the documentation you want to display.
- From the note shortcut menu, select **Text Display Node** and then **Show Documentation**.

TIP!

If a comment contains stereotypes, tagged values, and/or constraints, you can choose to display them in the content of the note.

UML Extension Elements

UML is a general purpose visual modeling language for specifying, constructing and documenting the artifacts of systems that can be used with all major application domains and implementation platforms. It has been widely adopted by both industry and academia as the standard language for describing software systems. However, the fact that UML is a general purpose notation may limit its suitability for modeling some particular specific domains (e.g. web applications or business processes), for which specialized languages and tools may be more appropriate.

UML provides a set of extension mechanisms (stereotypes, tag definitions, and constraints) for specializing its elements, allowing customized extensions of UML for particular application domains. These customizations are sets of UML extensions grouped into UML profiles.

There are several reasons why you may want to extend a metamodel:

- Assign a terminology that is adapted to a particular platform or domain (such as capturing EJB terminology like home interfaces, enterprise java beans, and archives).
- Assign a syntax for constructs that do not have a notation (such as in the case of actions).
- Assign a different notation for the existing symbols (such as being able to use a picture of a computer instead of the ordinary node symbol to represent a computer in a network).

11 MODEL ELEMENTS

UML Extension Elements

- Add semantics that is left unspecified in the metamodel (such as how to deal with priority when receiving signals in a statemachine).
- Add semantics that does not exist in the metamodel (such as defining a timer, clock, or continuous time).
- Add constraints that restrict the way you may use the metamodel and its constructs (such as disallowing actions from being able to execute in parallel within a single transition).
- Add information that can be used when transforming a model to another model or code (such as defining mapping rules between a model and Java code).

MagicDraw goes beyond the traditional profiling. It allows developers to create a new modeling tool based on their UML profile. In the following sections, you will find information about how to work with profiles, stereotypes, and tags.

For a detailed description of UML Profiling and DSL in MagicDraw, see the [UML Profiling and DSL User's Guide](#).

Stereotype

A stereotype defines how an existing metaclass may be extended. It enables the use of platform or domain specific terminology or notation in place of, or in addition to, the ones used for the extended metaclass.

Just like a class, a stereotype may have properties, which may be referred to as tag definitions. When a stereotype is applied to a model element, the values of the properties may be referred to as tagged values.

Any model element from the reference metamodel (any UML model element) can be extended by a stereotype. For example in UML, States, Transitions, Activities, Use cases, Components, Attributes, Dependencies, etc. can all be extended with the stereotypes.

The stereotypes are created as separate model elements and can be drawn in the Class diagrams. According to UML, the stereotypes have to be stored in the Profiles (a subtype of Package), so the Class diagram for the stereotypes should be created in the Profile instead of the Package.

For more information about working with symbols, see Chapter "Diagramming" on page 167.

To create a new stereotype

To create a new stereotype it is recommended to first create a profile (see the Section above).

- Use the **Stereotype** button on the class diagram toolbar, **Profiling Mechanism** group. The **Select Metaclass** dialog box opens. Specify the metaclass that you want to extend and click **OK**. Double-click to open the **Stereotype Specification** dialog box. For a detailed description of this dialog box, see Section "Stereotype Specification dialog box" on page 627.
- From the Profile or Package shortcut menu in the Browser, select **New**, and then select **Stereotype**.

To create a stereotype with an image

1. Open the **Stereotype Specification** dialog box.
2. Click the **Icon ...** button and from the **Open** dialog box, select an image you want to place for the stereotype. Click the **Open** button.

NOTE

For more information about displaying stereotype icon on shape, see "Displaying icon or image" on page 256.

11 MODEL ELEMENTS

UML Extension Elements

To view assigned image properties

1. In the **Stereotype** specification dialog box, invoke the **Icon** text box shortcut menu.
2. Select the **Open Specification** dialog box. The **Image** specification dialog box opens. You can see the format and location of the image, and specify the other image properties.

To display a stereotype icon as a main shape on the diagram pane

From the shape that has an assigned stereotype with an icon, select **Presentation Options** and then **Shape Image** from the **Shape Shortcut menu**. Note that all compartments must have been suppressed before.

To order stereotypes

Stereotypes can be ordered. Symbol style of the first in the list stereotype will be applied to the symbol on diagram and in Browser.

To order stereotypes:

1. Invoke symbol shortcut menu, select the **Stereotype** command. The list of stereotypes opens (see Figure 391 on page 626).
2. Click the **Order** button. The **Order Stereotypes** dialog box opens (see Figure 392 on page 626).
3. Click **Up** or **Down** buttons to order stereotypes.

11 MODEL ELEMENTS

UML Extension Elements

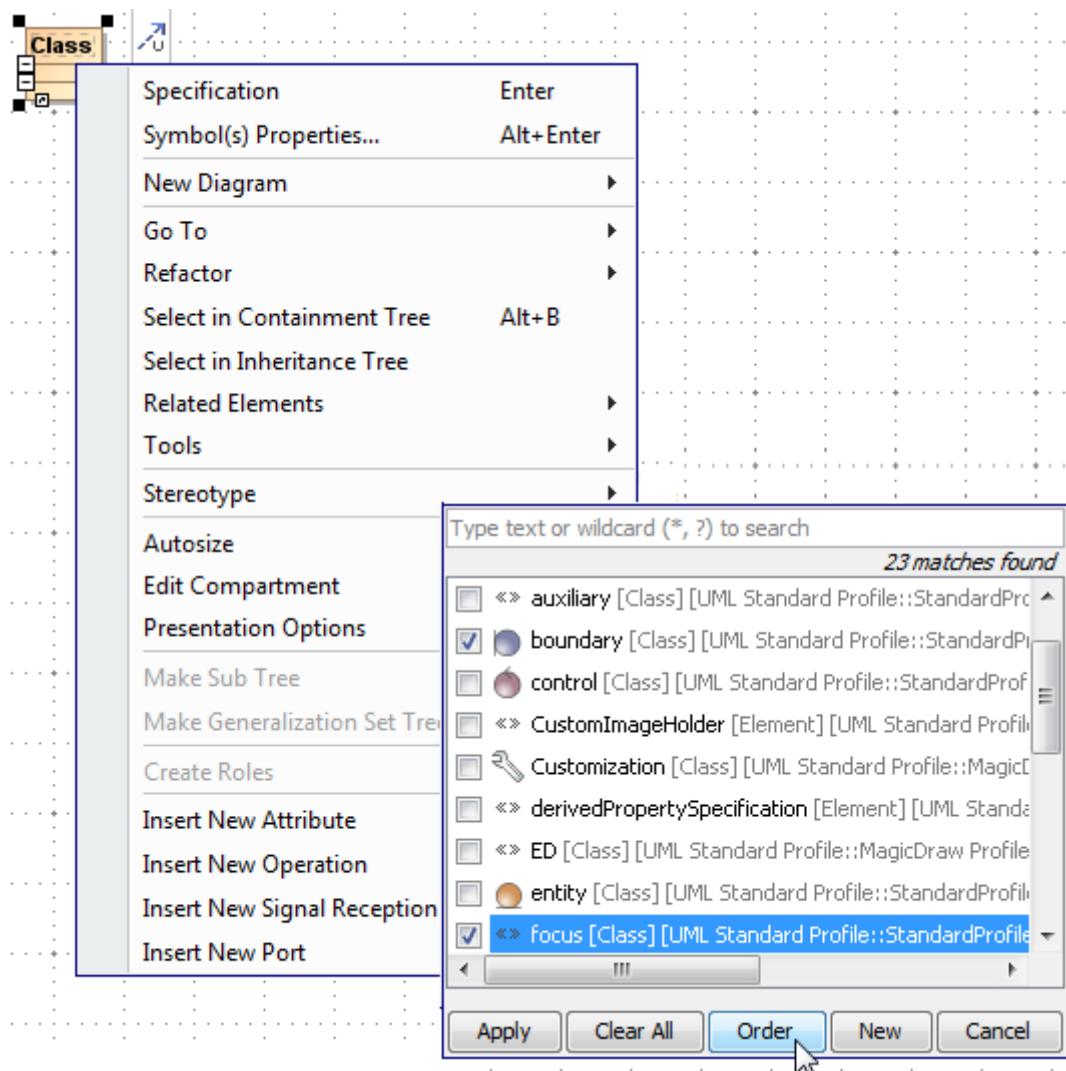


Figure 391 -- The Order button in the Stereotypes list, in the Class symbol shortcut menu

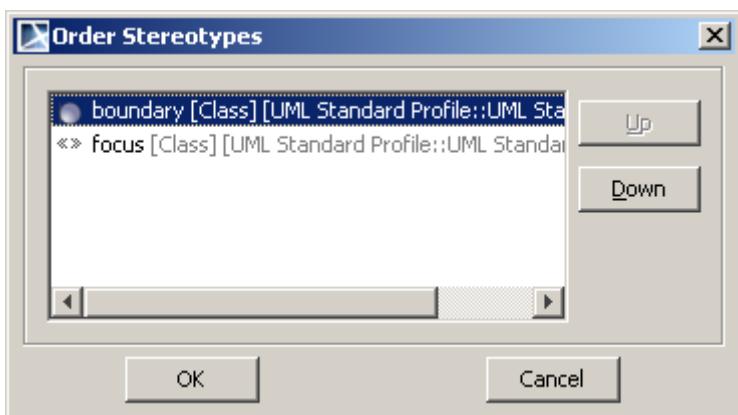


Figure 392 -- The Order Stereotypes dialog box

Saving of stereotype information in XMI

Ability to choose options where to save applied stereotype information in XMI file. Information can be saved at the end of the file or inside the element information.

11 MODEL ELEMENTS

UML Extension Elements

By default stereotype information is stored at the end of XMI file. To store information inside element, open **Environment Options** dialog (choose **Options - Environment** command from main menu), and in **General - Save/Load** section select property **Save stereotype information within element**.

Stereotype Specification dialog box

Define the selected stereotype in the **Stereotype Specification** dialog box.

To open the **Stereotype** specification dialog box

- Double-click the stereotype shape or select **Specification** from the stereotype shortcut menu.

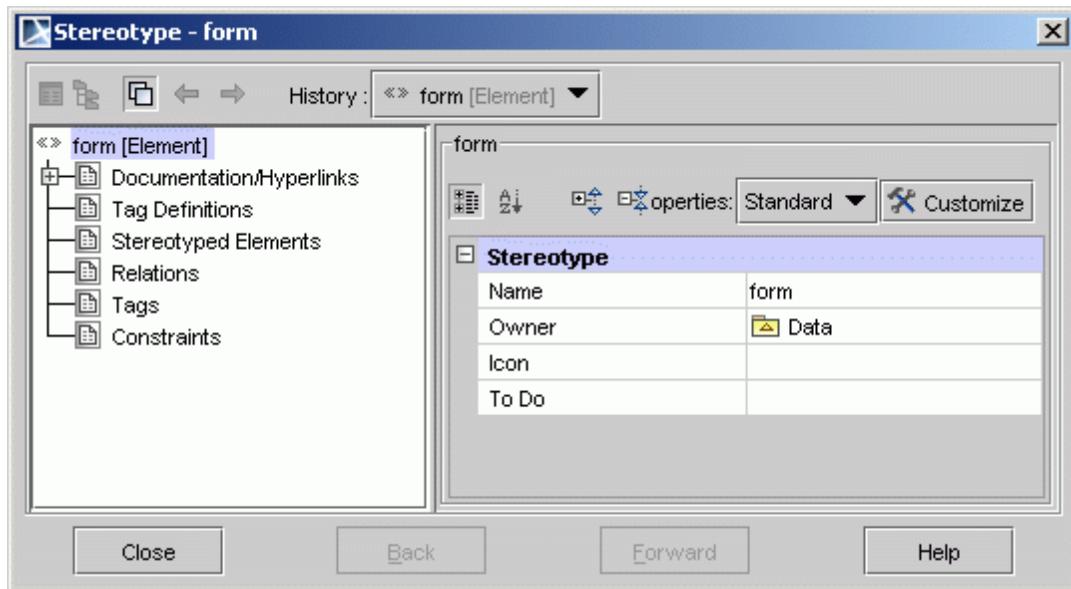


Figure 393 -- Stereotype Specification dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Group name | Box name | Function |
|---|----------------------|---|
| General Set a general information about the stereotype. | Icon | An image file icon. Refer to the UML Profiling and DSL manual for the description of the advanced icons for relationships. |
| Tag Definitions Specify information for tag definition | Name | The name of a stereotype attribute. |
| | Type | In general case, where the tag type is a data type, this specifies the range of values of the tagged values associated with the tag definition. In a special case, where the tag type refers to a metaclass that is not a data type, the tag value references model elements that are instances of the metaclass. |
| | Classifier | The classifier element of a stereotype attribute. |
| | Default value | The default value for tag definition. |
| | Create | Creates a new tag definition. |

11 MODEL ELEMENTS

UML Extension Elements

| | | |
|---|----------------|---|
| | Delete | Deletes an existing tag definition. |
| | Clone | |
| | Up/Down | |
| Stereotyped Elements Shows the model elements affected by the stereotype. | Name | The name of the stereotyped model element. |
| | Type | The type (class, use case, etc.) of a model element. |
| | Add | Applies a stereotype to the selected model element. |
| | Remove | Unapplies a stereotype from the selected model element. |

Assigning a stereotype

You may assign a stereotype to an element in the following ways:

- Open the corresponding element **Specification** dialog box. Click the "..." button in the general pane, next to the **Applied Stereotype** property. Select one or more available stereotypes from the open list and click **Apply**. Or you may create a new stereotype by clicking **New**.
- You may apply a stereotype to a model element easily from the model element shortcut menu. To do this, follow these steps:
 1. Select the model element on the diagram pane or in the Browser. Open its shortcut menu by right-clicking the mouse.
 2. From the model elements shortcut menu, select the **Stereotype** command. The stereotype menu opens.
 3. The list of all stereotypes is seen. You can type the name of the stereotype for which you are searching in the text box, above the list. Or, you can create a new stereotype by clicking the **New** button.
 4. When you find the stereotype that you want, or create a new one, select the check boxes of those stereotypes to add to the model element.
 5. Click the **Apply** button.
- On the diagram pane in the element name area, type two open angle brackets <<, type the stereotype name and type two close angle brackets. Then you may type the element name itself. For example: if you want to name element Books and assign <<table>> stereotype, in the element name area type the following: <<table>>Books. The name completion for the stereotypes works in the name editing mode, press the **Ctrl+Space** or **Ctrl+Backspace** to get a list of possible to apply stereotypes.

Changing the stereotype display mode

As of MagicDraw version 15.0 you can change the stereotype name and its icon visibility on an element shape.

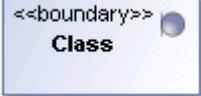
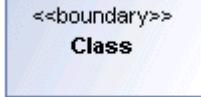
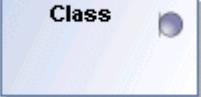
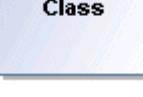
To change the stereotype display mode on the element shape

- On the element shape shortcut menu, point to **Presentation Options > Show Stereotypes**, and then select a desired stereotype display mode.
- In the **Symbol Properties** dialog select a new value for the **Show Stereotypes** property.

11 MODEL ELEMENTS

UML Extension Elements

The stereotype display modes are described in the following table.

| Show Stereotypes Property Value | Shape | Icon of the stereotype on the shape | Name of the stereotype | Image of the stereotype instead of the element shape |
|---------------------------------|---|-------------------------------------|------------------------|--|
| Text and icon |  | displayed | displayed | - |
| Text |  | not displayed | displayed | - |
| Icon |  | displayed | not displayed | - |
| Shape Image and Text |  | - | displayed | displayed* |
| Shape Image |  | - | not displayed | displayed* |
| Do Not Display |  | not displayed | not displayed | - |

* - To display the image of a stereotype instead of the element shape all element compartments should be suppressed.

NOTE **Shape Image and Text** and **Shape Image** properties are not included in the Relationships, Roles and Diagram Frame element property list.

Parent topic: “Stereotype” on page 624.

Related topics:

“Assigning a stereotype” on page 628.

Stereotype notation

Since v 16.6 MagicDraw supports standard stereotype notations. Stereotype notations in diagrams use the guillemets « » instead of symbols << >> (Figure 394 on page 630).

However, when editing elements in a diagram, you can still enter the stereotype names between the << >> symbols.



Figure 394 -- Stereotype Notation

Tags

Just like a class, a stereotype may have properties, which may be referred to as tag definitions. When a stereotype is applied to a model element, the values of the properties may be referred to as tagged values.

The tag definitions are used to define new meta attributes of the extended metaclass, they are used as regular class attributes. For more information about working with class attributes, see Section “Drag and drop the selected Opaque Behavior element from the Browser tree on the Diagram pane.” on page 728.

An actual instance of the tag definition is a tagged value (tag).

A tag holds extra information like:

- additional information that does not come with UML, for example Precondition for Use Cases.
- management data about the state and progress of the project such as author, status, and tested.
- language specific data for code generation tools.

A tagged value consists of two parts: name and value (example: Author = Joe).

To create a new tag definition

1. Create a new stereotype. Open the **Stereotype Specification** dialog box.
2. Click the **Tag Definitions** tab and use the **Create** button to add a new tag definition for stereotype. Select the type of this property. It may be a standard UML data type or another user defined Stereotype. Regular classes should not be used as types of tag definition.
3. Click **Close** in the **Stereotype Specification** dialog box to save changes.

Refer to Section “Stereotype Specification dialog box” on page 627 for more information about **Stereotype Specification** dialog box fields not covered in this section.

Editing Tagged Value

To create a new tagged value

- In the element **Specification** dialog box, **Tags** group, select an available tag definition and click the **Create Value** button.

11 MODEL ELEMENTS

UML Extension Elements

- Set the tag value using drag and drop. Now you can drag an element from the Containment tree and drop in the Specification window tag value area.

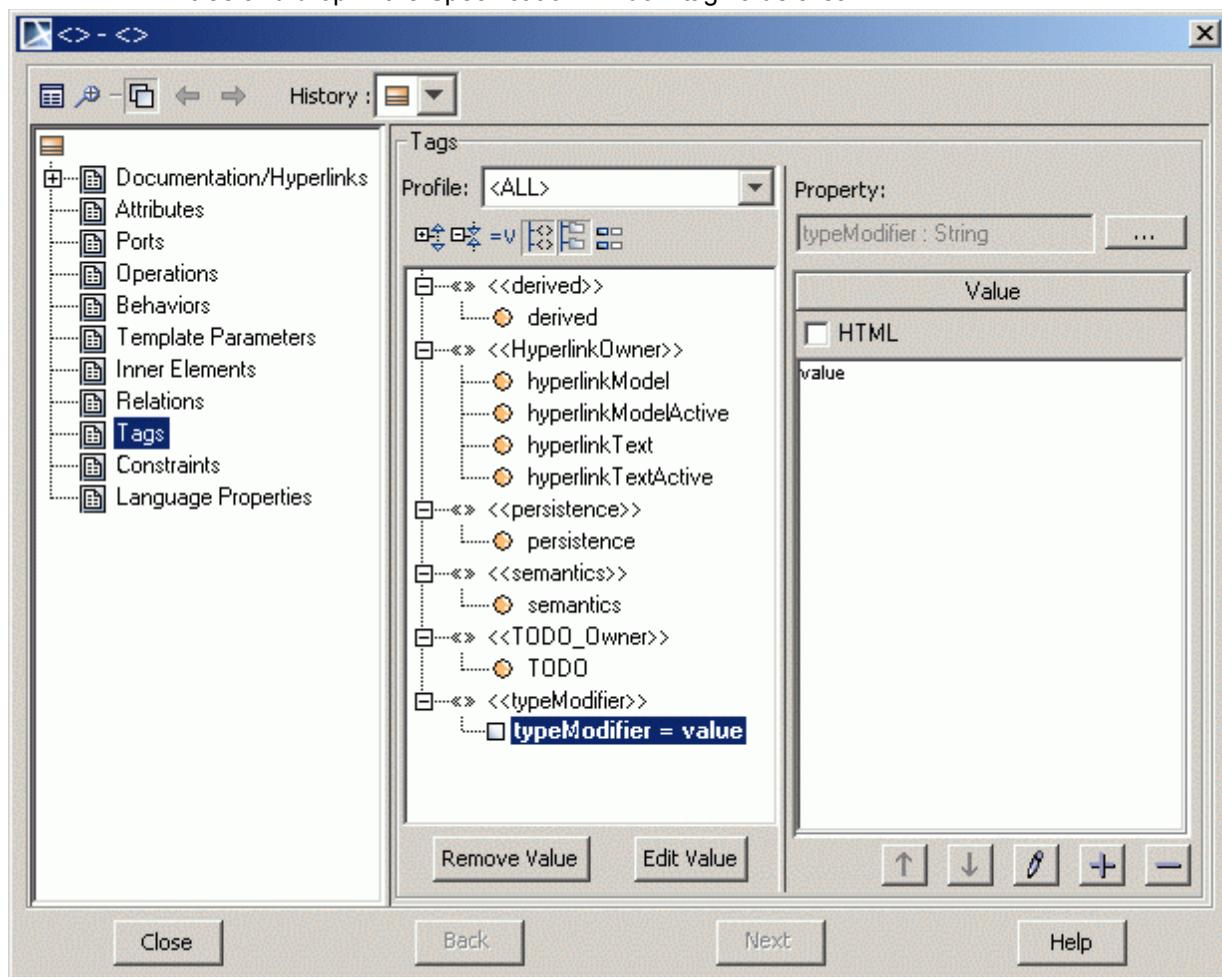


Figure 395 -- Tags dialog box

| Button | Description |
|--------|---|
| Up | Moves the created value to an upper position in the list. [Up arrow icon] |
| Down | Moves the created value to a lower position in the list. [Down arrow icon] |
| Edit | Opens an editable window for value. [Edit icon] |
| Add | Adds a new tagged value to the list. [Add icon] |

11 MODEL ELEMENTS

UML Extension Elements

| Button | Description |
|---|---------------------------------------|
|  | Removes a tagged value from the list. |

To create default tag values

1. In the **Stereotype** specification dialog box, expand the **Tag Definitions** branch and select the tag definition.
2. Specify a default value in the **Default Value** field.
3. Create an element and assign it a stereotype. The element will have tags with assigned default values.

To set a default tag value to tag with empty value

MagicDraw version 15.0 and later allows setting default tag values to tag with empty value. This functionality is needed when a stereotype is already assigned to an element and a new mandatory tag definition with a default value is created for the stereotype. After creating such a tag definition, the model elements that have the modified stereotype applied will have the newly created tags unset.

To set default values instead of empty values:

1. From the **Tools** main menu, select the **Set Empty Tags to Defaults** command. The **Select Package** dialog box opens.
2. Select the scope of elements to which you want to set the default tag values.
Note If you want to assign default values to all project tags, in the *All data tree* select the *Data* model and click the **Add All** button. *Data package* is added to the *Selected objects* list. Click **OK**.
3. The **Question** dialog box opens informing you that this action will set the mandatory tags without values to defaults.
If you do not want to see this message again next time, clear the **Show this message next time** check box.
4. Click **Yes**. Now, the element with stereotype tags has been assigned default values.

The following conditions are required to set the default tag values:

1. The element should have an assigned stereotype with a specified default value property.
2. The tag definition, to which you want to assign a default value, should contain no value.

NOTE The default tag values are not set when the stereotype property (tag definition) multiplicity is equal to 0.

Parent Topic: “UML Extension Elements” on page 623.

Related topics

[Stereotype](#)
[Attribute](#)

Constraint

A Constraint represents additional semantic information attached to the constrained elements. It is an assertion that indicates a restriction that must be satisfied by a correct design of the system. The constrained elements are those elements required to evaluate the constraint specification. In addition, the context of the Constraint may be accessed, and may be used as the namespace for interpreting names used in the specification.

11 MODEL ELEMENTS

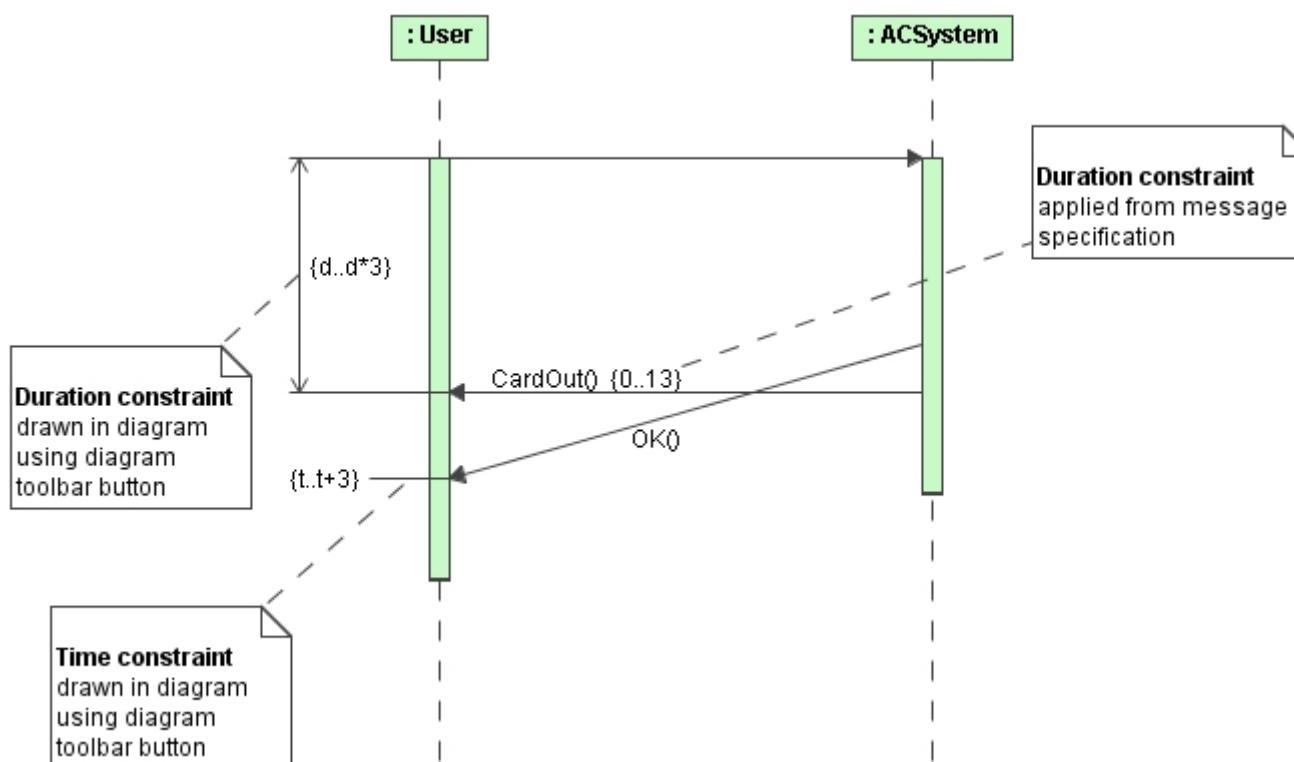
UML Extension Elements

A Constraint is represented as a linguistic, enclosed in braces {}, statement in some formal (OCL, C++, and other), or a natural language. There are 14 standard constraints in UML such as association, global, and parameter. You may also define your own constraints.

The Time and Duration Constraints

The Time Constraint specifies the combination of min and max timing interval values.

The Duration Constraint defines a value specification that specifies the temporal distance between two time instants.



Working with Constraints

Edit constraints in the Constraints tab of the Model Element Specification dialog box.

To define a new constraint

- From the element shortcut menu in the Browser, select **New**, and then select **Constraint**.
- Click the **Inner Elements** tab in the **Specification** dialog box for each model element and click the **Create** button (select **Constraint** from the open list, if needed). Specify the information about the constraint in the open **Constraint Specification** dialog box.
- Click the **Constraints** tab in the **Specification** dialog box for each model element and click the **Create** button. Type the name and specification of the constraint.

NOTE

If a constraint is displayed in the **Constraints** tab of the Element Specification dialog box, it means this constraint is valid and applied for the element. If it is displayed in the **Inner Elements** list, this constraint is only owned by the element.

11 MODEL ELEMENTS

UML Extension Elements

To apply a constraint to an element

1. Click the **Constraints** tab in the **Specification** dialog box for each model element and click the **Apply** button.
2. The **Select Elements** dialog box opens. Select a constraint existing in the model from the **All Data** tree and click the **Add** button to move it to the **Selected Objects** list.
3. Click **OK**.

The Constraint Specification dialog box

To open the **Constraint Specification** dialog box

In the **Edit Constraints** dialog box, click the **Add** or **Edit** button.

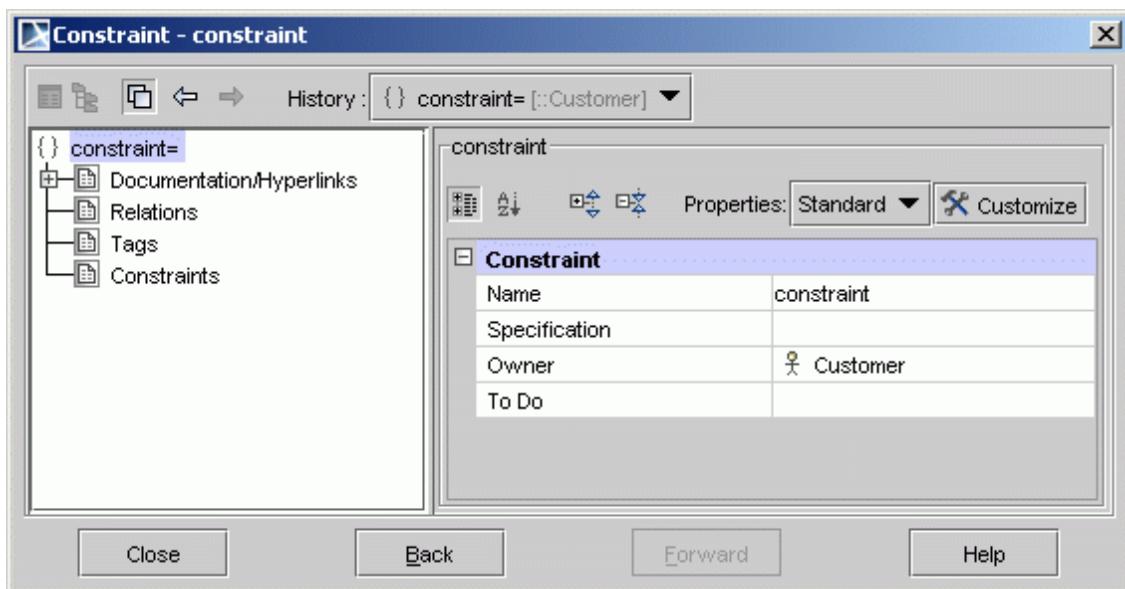


Figure 396 -- Constraint Specification dialog box

| Tab name | Box | Function |
|---|----------------------|--|
| General Set a general information about the constraint. | Name | Enter the name of the constraint. |
| | Owner | The name of the model element to which the current constraint is assigned. |
| | Specification | Enter an expression of the constraint. Click “...” button and in the Edit Specification window, choose OCL. The specification will be checked according to Object Constraint Language (OCL) |
| | ToDo | Additional information can be added. |
| Documentation/ Hyperlinks | Documentation | Set the tagged value documentation. It can be HTML text also. |
| | Hyperlink | Choose a hyperlink for the tagged value. It can be a file, model element, or web site. |

11 MODEL ELEMENTS

UML Extension Elements

| Tab name | Box | Function |
|-------------|-----|---|
| Tags | | Creates, edits, and removes tagged definitions. |
| Constraints | | Creates, applies, and removes a constraint for a tag. |

OCL

Object Constraint Language (OCL) is a formal language used to express constraints. These typically specify the invariant conditions that must hold for the system being modeled.

Expressions can be used in a number of places in a UML model:

- To specify the initial value of an attribute or association end.
- To specify the derivation rule for an attribute or association end.
- To specify the body of an operation.
- To indicate an instance in a dynamic diagram.
- To indicate a condition in a dynamic diagram.
- To indicate the actual parameter values in a dynamic diagram.

There are four types of constraints:

- An *invariant* is a constraint that states a condition that must always be met by all instances of the class, type, or interface. The invariant is described using an expression that evaluates to true if the invariant is met. Invariants must be true all the time.
- A *precondition* to an operation is a restriction that must be true at the moment the operation is going to be executed. The obligations are specified by the postconditions.
- A *postcondition* to an operation is a restriction that must be true at the moment the operation has just been executed.
- A *guard* is a constraint that must be true before a state transition discharged.

Invariants on attributes

The simplest constraint is an invariant on an attribute. Suppose a model contains a class *Customer* with an attribute *age*, then the following constraint restricts the value of the attribute:

```
context Customer inv:  
age >= 18
```

Invariants on associations

One may also put constraints on the associated objects. Suppose a model contains the class *Customer* that has an association to the class *Salesperson* with the role name *salesrep* and multiplicity 1, then the following constraint restricts the value of the attribute *knowledgelevel* of the associated instance of *Salesperson*:

```
context Customer inv:  
salesrep.knowledgelevel >= 5
```

Collections of objects

In most cases the multiplicity of an association is not 1, but more than 1. Evaluating a constraint in these cases will result in a collection of instances of the associated class. Constraints can be put on either the collection

11 MODEL ELEMENTS

UML Extension Elements

itself, e.g. limiting the size, or on the elements of the collection. Suppose in a model the association between *Salesperson* and *Customer* has the role name *clients* and multiplicity 1..* on the side of the *Customer* class, then we might restrict this relationship by the following constraints:

```
context Salesperson inv:  
clients->size() <= 100 and clients->forAll(c: Customer | c.age >= 40)
```

Pre- and postconditions

In the pre- and postconditions the parameters of the operation may be used. Furthermore, there is a special keyword *result* which denotes the return value of the operation. It can be used in the postcondition only. For example an operation *sell* was added to the *Salesperson* class.

```
context Salesperson::sell( item: Thing ): Real  
pre: self.sellableItems->includes( item )  
post: not self.sellableItems->includes( item ) and result = item.price
```

Derivation Rules

Models often define derived attributes and associations. A derived element does not stand alone. The value of a derived element must always be determined from other (base) values in the model. Omitting the way to derive the element value results in an incomplete model. Using OCL, the derivation can be expressed in a derivation rule. In the following example, the value of a derived element *usedServices* is defined to be all services that have generated transactions on the account:

```
context LoyaltyAccount::usedServices : Set(Services)  
derive: transactions.service->asSet()
```

Initial Values

In the model information, the initial value of an attribute or association role can be specified by an OCL expression. In the following examples, the initial value for the attribute *points* is 0, and for the association end *transactions*, it is an empty set:

```
context LoyaltyAccount::points : Integer  
init: 0  
context LoyaltyAccount::transactions : Set(Transaction)  
init: Set{}
```

Body of Query Operations

The class diagram can introduce a number of query operations. The query operations are operations that have no side effects, i.e. do not change the state of any instance in the system. The execution of a query operation results in a value or set of values without any alterations in the state of the system. The query operations can be introduced in the class diagram, but can only be fully defined by specifying the result of the operation. Using OCL, the result can be given in a single expression, called a body expression. In fact, OCL is a full query language, comparable to SQL. The use of body expressions is an illustration thereof.

The next example states that the operation *getCustomerName* will always result in the name of the card owner associated with the loyalty account:

```
context LoyaltyAccount::getCustomerName() : String  
body: Membership.card.owner.name
```

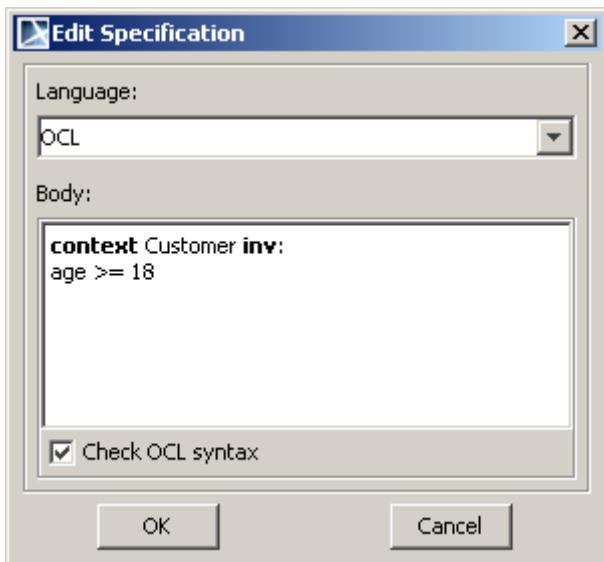
To check OCL syntax according to OCL grammar

1. Click the “...” button near the **Specification** field in the **Constraint Specification** dialog box. The **Edit Specification** dialog box opens.

11 MODEL ELEMENTS

UML Extension Elements

2. Select the **OCL** in **Language** field and check the **Check OCL syntax** check box. Incorrect expression in the **Body** text field will be highlighted in red.



Profiles

A Profile is a kind of Package that extends a reference metamodel. The primary extension construct is the Stereotype, which are defined as part of the Profiles.

A profile introduces several constraints, or restrictions, on ordinary metamodeling through the use of the metaclasses defined in the package. It is a restricted form of a metamodel that must always be related to a reference metamodel, such as UML, as described below. It cannot be used without its reference metamodel, and it defines a limited capability to extend metaclasses of the reference metamodel. The extensions are defined as stereotypes that apply to the existing metaclasses.

The profile is defined as a package – it has package properties – in the **Profile Specification** dialog box. For a detailed description of packages, see “Package” on page 733.

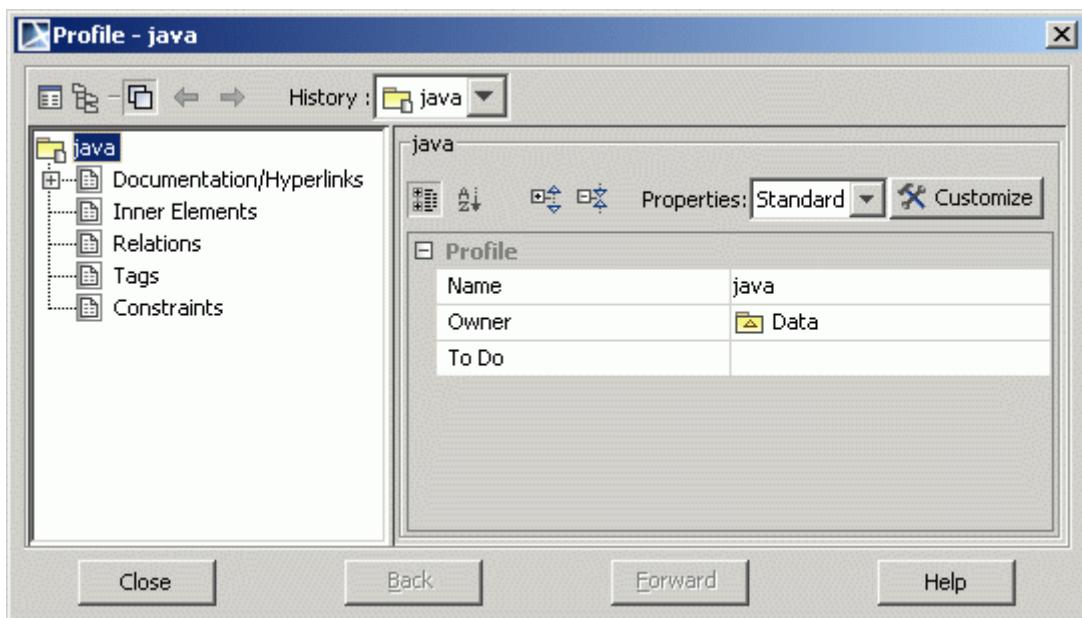


Figure 397 -- Subsystem Specification dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

Profiles

The Profiles are defined using the UML extensibility mechanisms. These allow modelers to customize UML for specific domains such as software development processes. In MagicDraw, the mechanism of the profile is similar to the functionality of the modules, except that the profile is a package containing the stereotype <>profile>>, and its content is visible in both the Profiles dialog box and the UML extensions tabs of the Specification dialog boxes.

Working with profiles

There are several ways to work with profiles. You may:

1. Create your own profile.
2. Use the preexisting one.
3. Import the preexisting one.
4. Export your created profile.

Further details on these functions can be found in the following sections.

Creating Profiles

Usually a profile contains specific stereotypes, constraints, and tagged values that you can use according to your needs. The Profiles often contain other specific model elements, which MagicDraw allows you to include in your profile. You can use your existing profile in other parts of the projects. Additionally, other users may reuse your profile as a module.

To create your own profile

-
1. Create a profile element (in the Browser or using the Class Diagram toolbar, Profiling Mechanism group)
 2. Add the stereotypes, constraints, and/or other model elements you need for extending UML.

Using and Importing Profiles

You may use other profiles in your project, or import them. MagicDraw offers several profiles, found in the <MagicDraw installation directory>/profiles folder.

To use the preexisting profile in your project

-
1. From the **File** menu, select **Use Module**.
 2. The **Use Module** dialog box opens. If needed, select the directory where the desired profile is located. Select the desired profile, and click **OK**.

The selected profile opens in the Browser tree as a read-only profile. You may use it in your project. You can also import and edit it, if necessary.

To import the preexisting profile into your project

-
- From your selected profile, go to the Browser's shortcut menu. Select **Modules** and then **Import Module**.
 - From the **File** menu, select **Import**. The **Import** dialog box opens. Select the profile you want to import and click **Open**.

The imported profile will open as an editable package in the Browser.

Exporting Profiles

If you want to use the existing profile in another project, or enable other users to use it, you need to save it in your selected directory. Exporting profiles is similar to exporting module functionality.

1. From the **File** menu, select **Export**, and then **Module**.

The **Export Module** dialog box opens. Select the profile package you want to export and add it to the **Selected** list. Write the description of the profile in the text box, click **OK**, and save it to the desired directory.

Action

An action is a named element that is the fundamental unit of an executable functionality. The execution of an action represents some transformations or processing in the modeled system, be it a computer system or otherwise. An action execution represents the run-time behavior of executing an action within a specific behavior execution. As action is an abstract class, all action executions will be the executions of a specific kind of action. When the action will be executed and what its actual inputs will be are determined by the concrete action and the behaviors in which it is used.

The following are types of actions:

- ["Accept Event Action"](#) on page 639.
- ["Call Behavior Action"](#) on page 642.
- ["Call Operation Action"](#) on page 643.
- ["Opaque Action"](#) on page 644.
- ["Send Signal Action"](#) on page 645.
- Any other action.

See also: ["Working with actions"](#) on page 648.

Usage in diagrams: ["Activity Diagram"](#) on page 527.

Parent topic: ["Model Elements"](#) on page 620.

Related topics

["Specification Window"](#) on page 237.

["Formatting Symbols"](#) on page 255.

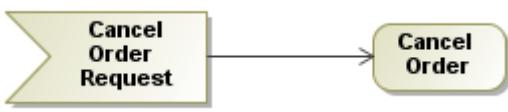
Accept Event Action

The Accept event action is an action that waits for the occurrence of an event that meets specified conditions. If an accept event action has no incoming edges, then the action starts when the containing activity or structured node starts, whichever most immediately contains the action. In addition, an accept event action with no incoming edges remains enabled after it accepts an event. It does not terminate after accepting an event and output-

11 MODEL ELEMENTS

Action

ting a value, but continues to wait for other events. An accept event action with no incoming edges and contained by a structured node is terminated when its container is terminated.

| Notation | Description |
|-----------------------------|---|
| Accept event action |  |
| Rotated accept event action |  |
| Accept time event action |  |

NEW! Assigning signals

To assign a signal to an accept event action you can use any of the following features:

- Drag-and-drop operation
- Accept event action's shortcut menu

To assign a signal using a drag-and-drop operation

IMPORTANT! To assign a signal to an accept event action, at least one signal element should exist in the project.

1. Select the signal in the Containment tree.
2. Drag the signal to the accept event action as it is shown in the following picture.

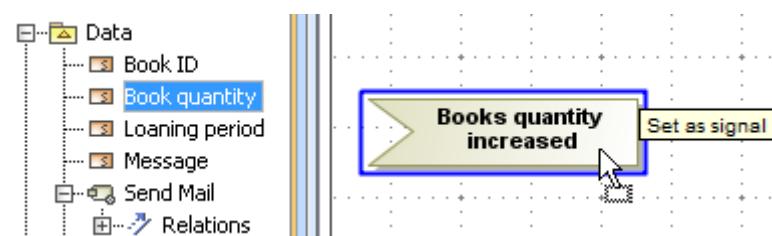


Figure 398 -- Signal dragging to accept event action

To assign a signal using the accept event action's shortcut menu

1. Right-click the accept event action to open the shortcut menu.
2. Click **Signal**. The list of signals available in the project will open. The example of the dialog is shown in the following picture.

11 MODEL ELEMENTS

Action

3. Select the signal from the list or click **New Signal** and create a new one. For more information about creating a new element see Section "[Element creation mode](#)" on page 280.

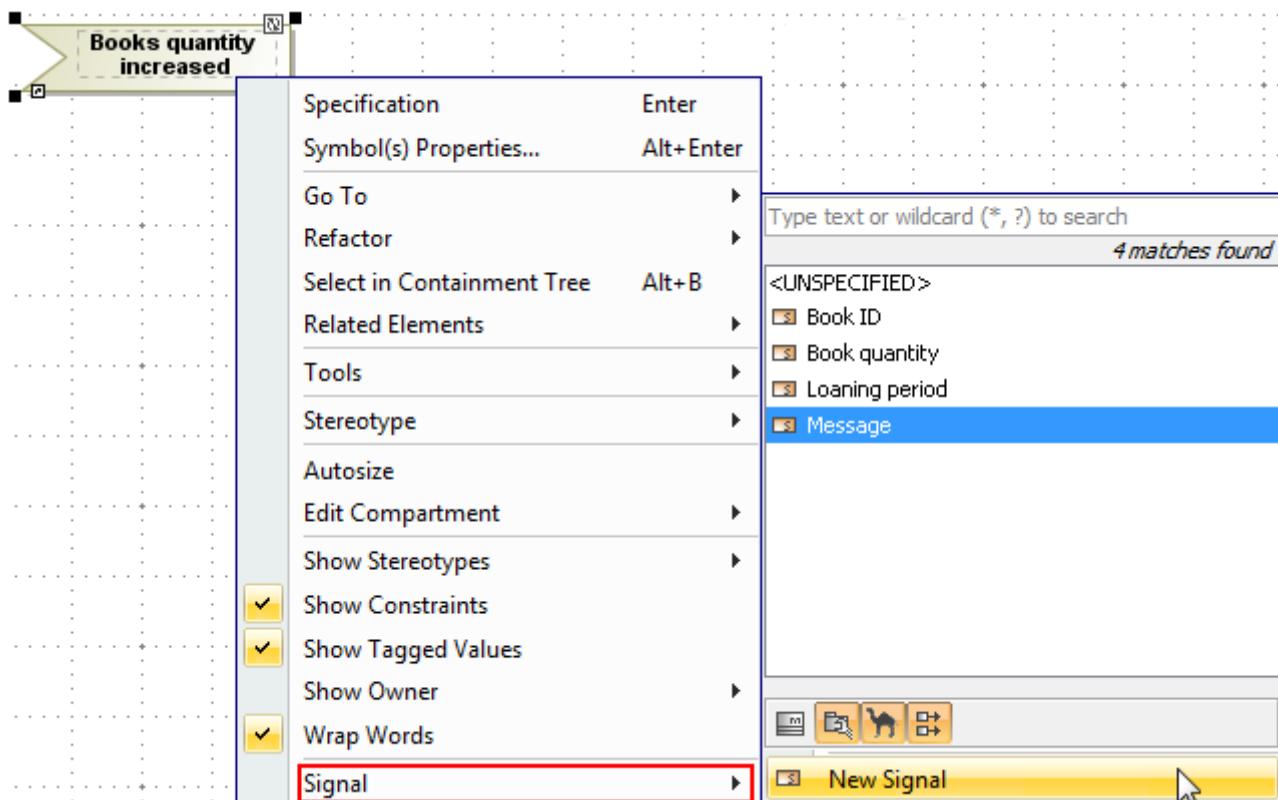


Figure 399 -- Signal selection using action shortcut menu

Once the signal is assigned to the accept event action, a signal event and a trigger for this action are created automatically. If you change the signal, the signal event and the trigger will change accordingly to a new signal. The following figure shows the signal event and the trigger positions in the Containment tree.

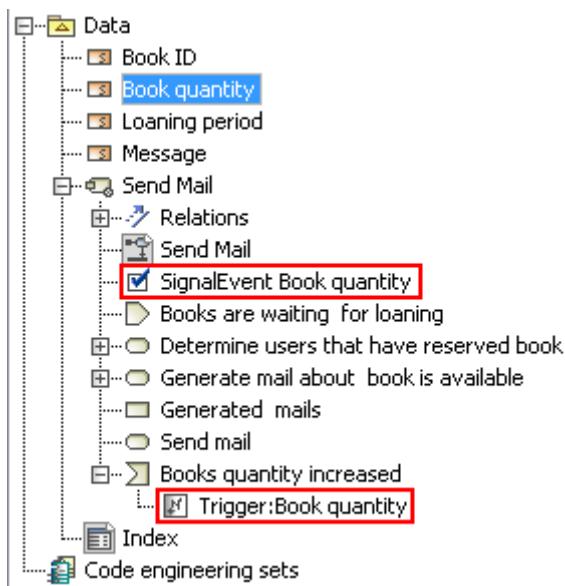


Figure 400 -- Signal event and trigger in the Containment tree

Using Time Event Shape

To draw the accept event action with the time event shape

- In the activity diagram panel select the **Time Event** button and then click the diagram pane.

11 MODEL ELEMENTS

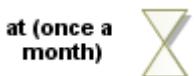
Action

To change the position of the accept event action name

1. Select the accept event action shape on the diagram pane.



2. Click the **Rotate State** button on the upper-right side of the shape. The name of the accept event action will appear on the opposite side of the shape.



You can specify for an accept event action, whether there is a single output pin for the event, or multiple output pins for the attributes of the event.

Specifying the Is Unmarshall property

To specify the **Is Unmarshall** property

1. Select the accept event action and open its Specification window.
2. Select or clear the **Is Unmarshall** check box. If the value is set to:
 - *false*, then there is a single output pin for the event, and a real-world instance of the signal is placed on this output pin.
 - *true*, then there are multiple output pins for the attributes of the event, and feature values of the signal instance are placed on the corresponding output pins.

Call Behavior Action

The call behavior action invokes a behavior directly rather than invoking a behavioral feature that, in turn, causes the behavior. The argument values of the action are available to the execution of the invoked behavior. The execution of the call behavior action waits until the execution of the invoked behavior completes and a result is returned on its output pin. In particular, the invoked behavior may be an activity.



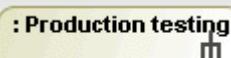
Figure 401 -- The Receive Order and Fill Order call behavior actions

To assign a behavior to the call behavior action

- From the call behavior action shortcut menu select the **Behavior** command.
- In the **Call Behavior Action** specification dialog box, modify the **Behavior** field.

NOTES

- Double click the call behavior action with the assigned behavior - the behavior specification dialog box is opened or if the assigned behavior is a diagram, the diagram will be opened.
- After the behavior is assigned to the call behavior action, a rake symbol is displayed on the action shape.



11 MODEL ELEMENTS

Action

To create a new diagram for the call behavior action

From the call behavior action symbol's shortcut menu, select **New Diagram** and then diagram from the list. This accelerates the creation of behavior diagrams.

To select the name display mode on the call behavior action

Stereotypes from the behavior can be visible on the call behavior action. You can select to show an action name, a behavior name or both by changing the **Name Display Mode** property in the call behavior action symbol **Properties** dialog box.

Call Operation Action

The call operation action transmits an operation call request to the target object, where it may cause the invocation of an associated behavior.

You can display an action name and/or name of the operation on the call operation action shape. For example, if you have two call operation action elements calling the same operation, you may specify their names to distinguish which action means what.

| Notation | Description |
|--|---|
| The call operation action with getOrder operation  | In this example the call operation action has been assigned <i>getOrder</i> operation whose type is <i>OrdersDB</i> class. |
| The call operation action named Get supplementary order  | In this example the call operation action is named <i>Get supplementary order</i> . It has been assigned <i>getOrder</i> operation whose type is <i>OrdersDB</i> class. |
| Call operation action with hidden classified name  | The classified name of operation is hidden from the call operation action shape. |

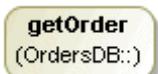
To assign an operation for the call operation action

From the action shortcut menu, select the **Operation** command or set an operation in the **Call Operation Action** specification dialog box, **Operation** field.

MagicDraw version 15.0 and later allows the display of operation name and class of operation on the call operation action shape.

When an operation is assigned to the call operation action, there are three name/operation display options available:

- If the call operation action is not named, the name of the class is displayed under the operation name.



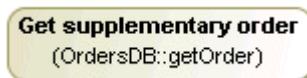
11 MODEL ELEMENTS

Action

- If the name of the call operation action is the same as the assigned operation name, then the name of the class is displayed under the operation name.



- If the call operation action name differs from the assigned operation name, then *<class of the operation>::<operation name>* is displayed under the call operation action name.



To hide the operation name and class of the operation from the call operation action shape

From the call operation action shortcut menu, clear the **Show Qualified Name for Operation** check box. You may also customize this property in the call operation action **Properties** dialog box.

NOTE When loading a project from a version older than MagicDraw version 15.0, by default the class of the operation is not displayed on the diagram pane. The **Show Qualified Name for Operation** option is unchecked.

Opaque Action

The opaque action is introduced for implementation-specific actions or for use as a temporary placeholder before some other actions are chosen.

The opaque action has no specific notation.

11 MODEL ELEMENTS

Action

There are additional **Body** and **Language** text fields in the **Opaque Action Specification window**.

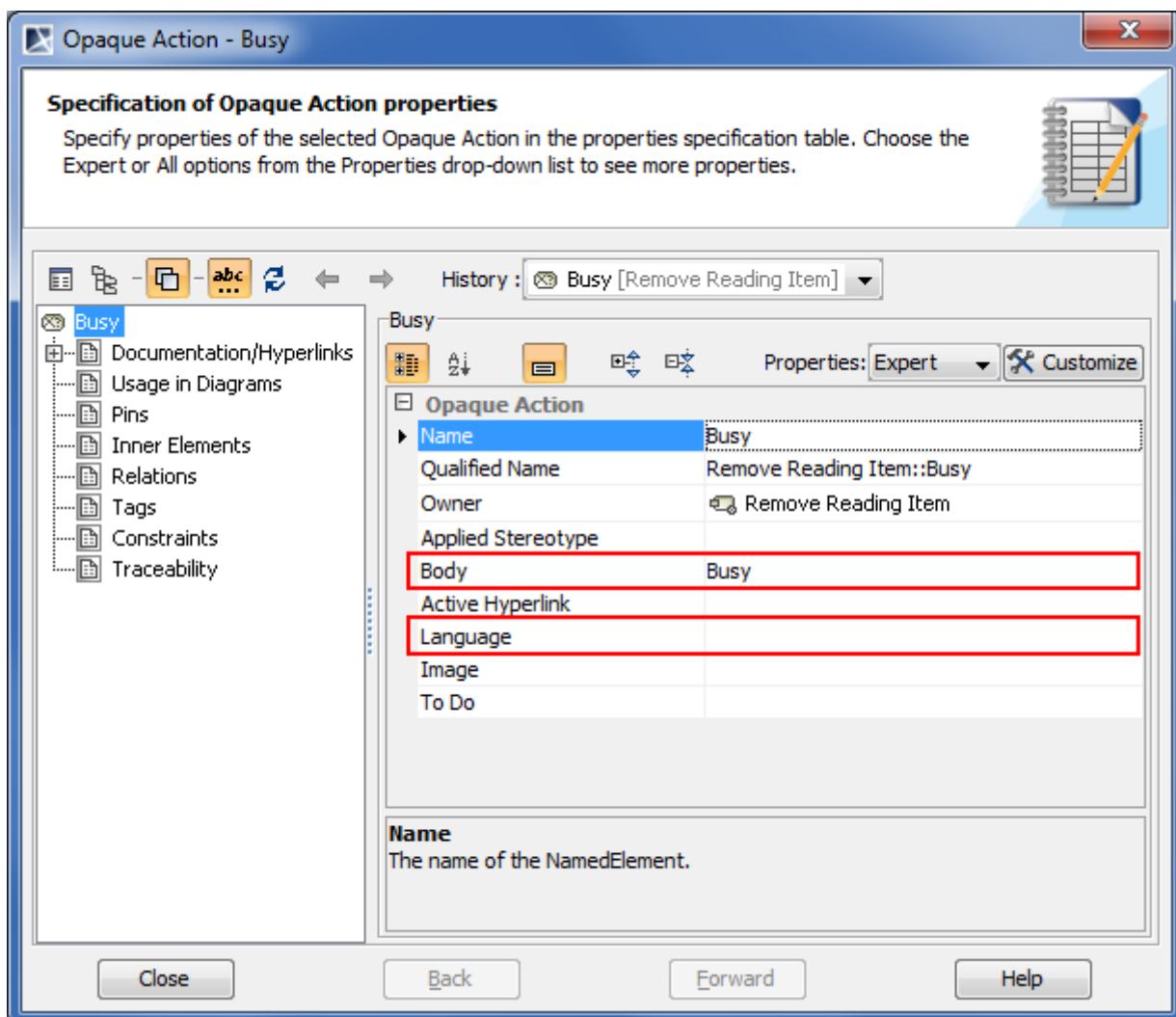


Figure 402 -- Opaque Action's Specification window

Send Signal Action

The send signal actions creates a signal instance from its inputs and transmits it to the target object, where it may cause the start of a state machine transition or the execution of an activity. The argument values are available to the execution of associated behaviors. The requester continues the execution immediately. Any reply message is ignored and is not transmitted to the requester. If the input is already a signal instance, use the Send object action.

| Notation | Description |
|--|---|
| <p>Send signal action</p>  | <p>This example describes an order process.</p> <ol style="list-style-type: none">First, an order is created (<i>Create order</i> call behavior action).Then, a signal to fill the order request is sent to the warehouse (<i>Fill order request</i> send a signal action).Finally, an invoice is created (<i>Create invoice</i> call behavior action). <p>The relationships are represented with control flow paths.</p> |

11 MODEL ELEMENTS

Action

Assigning signals

To assign a signal to a send signal action you can use any of the following features:

- Send signal action's Specification window
- Drag-and-drop operation
- **NEW!** Send signal action's shortcut menu

To assign a signal via the send signal action's Specification window

1. Open the Specification window for the send signal action.
2. In the **Signal** property value cell do any of the following:
 - Click the “...” button . The **Select Signal** dialog will open. Select the signal from the list or create a new one. For more information about creating a new element see Section ["Element creation mode"](#) on page 280.
 - Click the button to open the list of signals available in the project. Select the signal from the list. The example of the signal list is shown in the following picture.
3. Click **Close** once the signal is selected.

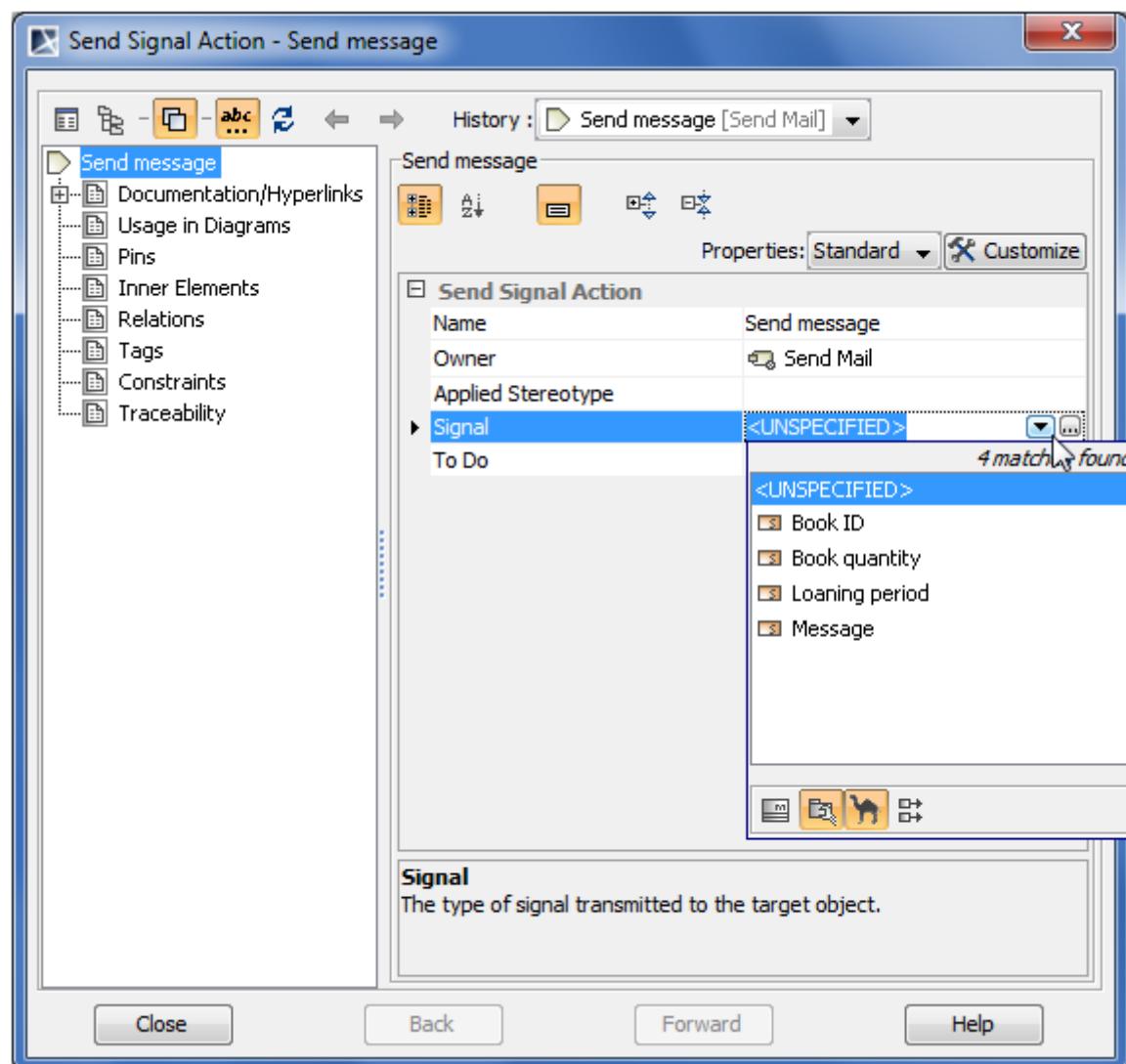


Figure 403 -- Send signal action's Specification window. Signal selection

11 MODEL ELEMENTS

Action

To assign a signal using a drag-and-drop operation

1. Select the signal in the Containment tree.
2. Drag the signal to the send signal action as it is shown in the following picture.

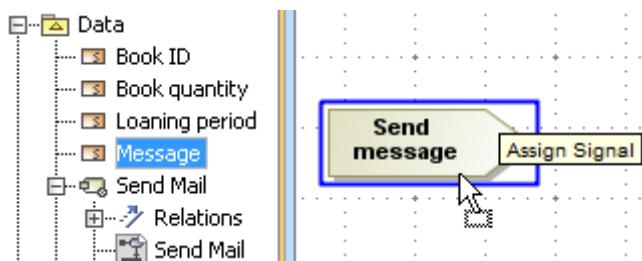


Figure 404 -- Signal dragging to send signal action

NEW! To assign a signal using the send signal action shortcut menu

1. Right-click the send signal action to open the shortcut menu.
2. Click **Signal**. The list of signals available in the project will open. The example of the dialog is shown in the following picture.
3. Select the signal from the list or click **New Signal** and create a new one. For more information about creating a new element see Section "[Element creation mode](#)" on page 280.

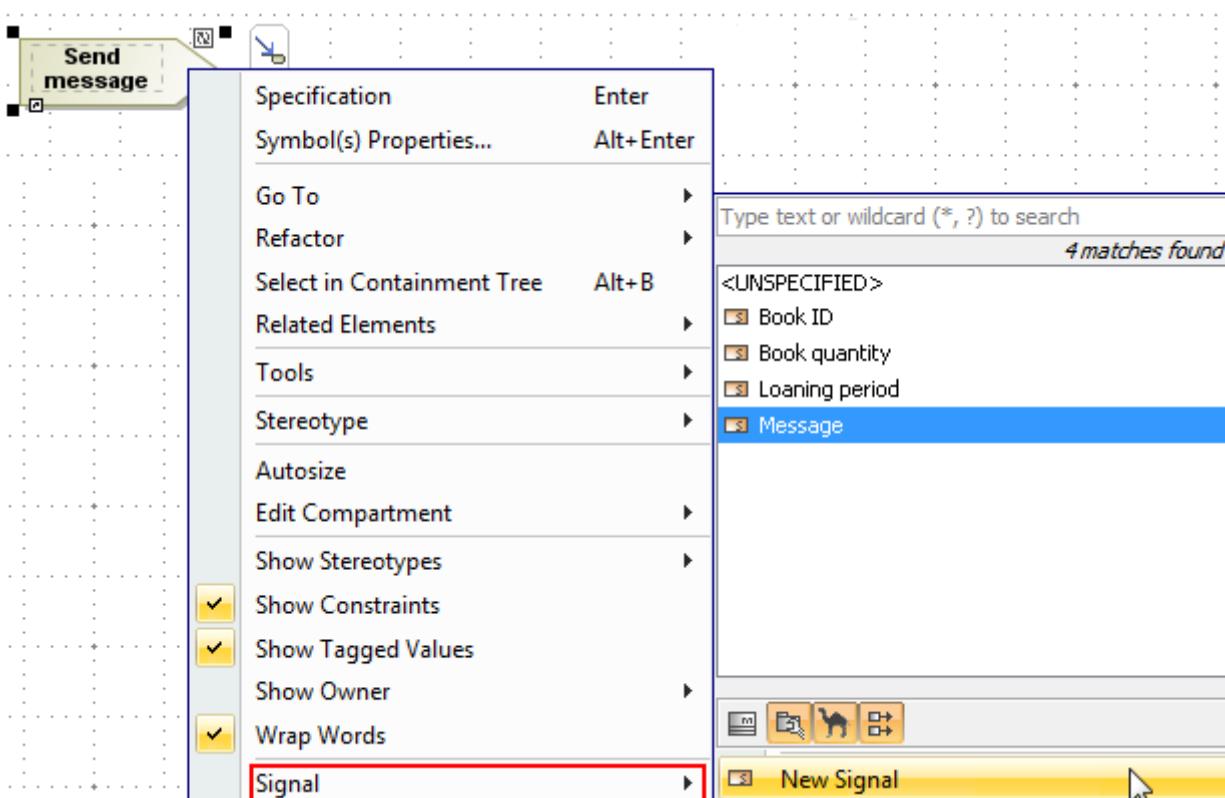


Figure 405 -- Signal selection using shortcut menu

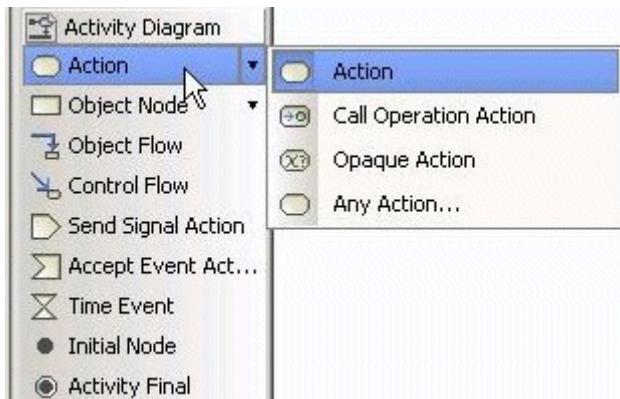
11 MODEL ELEMENTS

Action

Working with actions

To quickly create any action

1. In the activity diagram toolbar, right-click the **Action** button. The menu opens.



2. Select the **Any Action** command. The **Select Action Metaclass** dialog box opens.



3. Select an action metaclass from the list or type the first letter of the metaclass. Click **OK**. The action is created.
4. Click on the diagram pane. An action symbol is drawn.

To assign pins for an action

1. In the action specification dialog box, select the **Pins** branch.
2. In the right window size, in the **Input Pin** field, select the **Argument** box. Click '+' button. A



- menu with input pin, action input pin, or value pin opens.
3. Select the pin from the list. The pin specification dialog box opens.
 - In the **Output Pin** field, select the **Result** box and click '+' button. The **Output Pin** specification dialog box opens.

NOTES

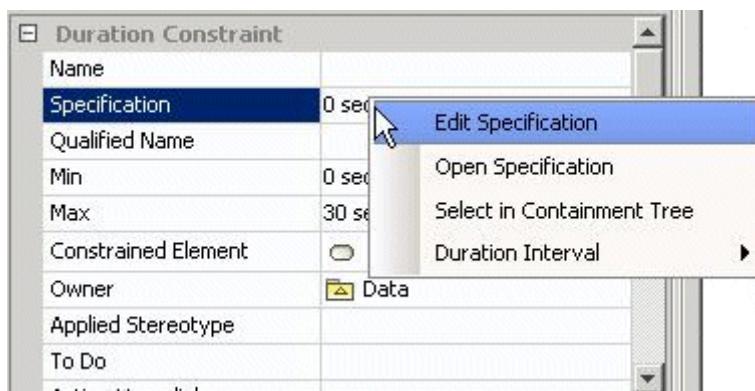
- The Output pin is not included when selecting the send signal action.
- The Input pin is not included when selecting the accept event action.

Advancing actions: applying duration constraint

You can create and apply a time duration constraint on an action that states that the output must occur after delay from the input.

To create and apply a time duration constraint

1. Select an action and create the input and output pins to specify the events.
2. Create a duration constraint for the action:
 - 2.1. In the **Call Behavior Action** specification dialog box, select the **Constraints** branch. At the bottom of the right side pane, click the **Apply** button. The **Select Elements** dialog box opens.
 - 2.2. Select the constraint storage place and click the **Create** button. In the menu that opens, select the **Duration Constraint** command. The **Duration Constraint** specification dialog box opens.
3. Specify a duration interval. In the **Duration Constraint** dialog box, select the **Expert** property display mode. Type the minimum and maximum duration for holding an activity in the **Min** and **Max** fields, e.g. type the following values: **0 sec** and **30 sec**.
4. Assign the events for the input and output pins:
 - 4.1. In the open **Duration Constraint** specification dialog box, right-click the **Specification** field. The following menu opens:



- 4.2. Select the **Open Specification** command. The **Duration Interval** specification dialog box opens.
- 4.3. Right-click the **Min** property and select the **Open Specification** command. The **Duration** specification dialog box opens.
- 4.4. In the **Event** field, click the “...” button. The **Select Element** dialog box opens. Select the activity input pin.
- 4.5. Repeat steps c and d for the **Max** property - select the action output pin as an event.

5. Apply the created duration constraint on the action.

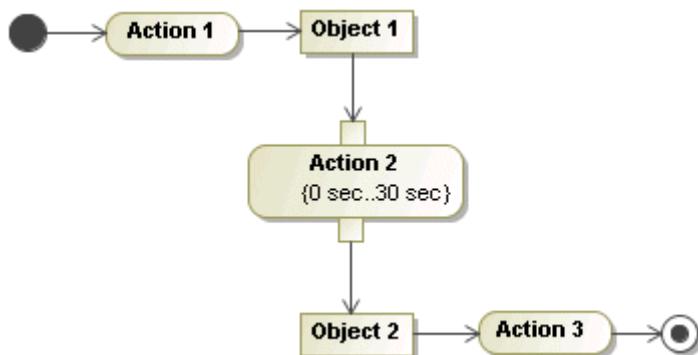
TIP!

To return to the former dialog box in the specification dialog box, click the **Back** button.

11 MODEL ELEMENTS

Actor

The sample below depicts an activity diagram with a duration constraint applied on an action.



Actor

Actors represent roles played by human users, external hardware, and other subjects. An actor does not necessarily represent a specific physical entity but merely a particular facet (that is, "role") of some entities that is relevant to the specification of its associated use cases.

An actor requires task solutions from a system. This task is represented as a use case.

An actor is shown as a “stick man” figure with the name below the figure.

A general information about working with shapes is offered in "Diagramming"

Define an actor in the **Actor Specification** dialog box.

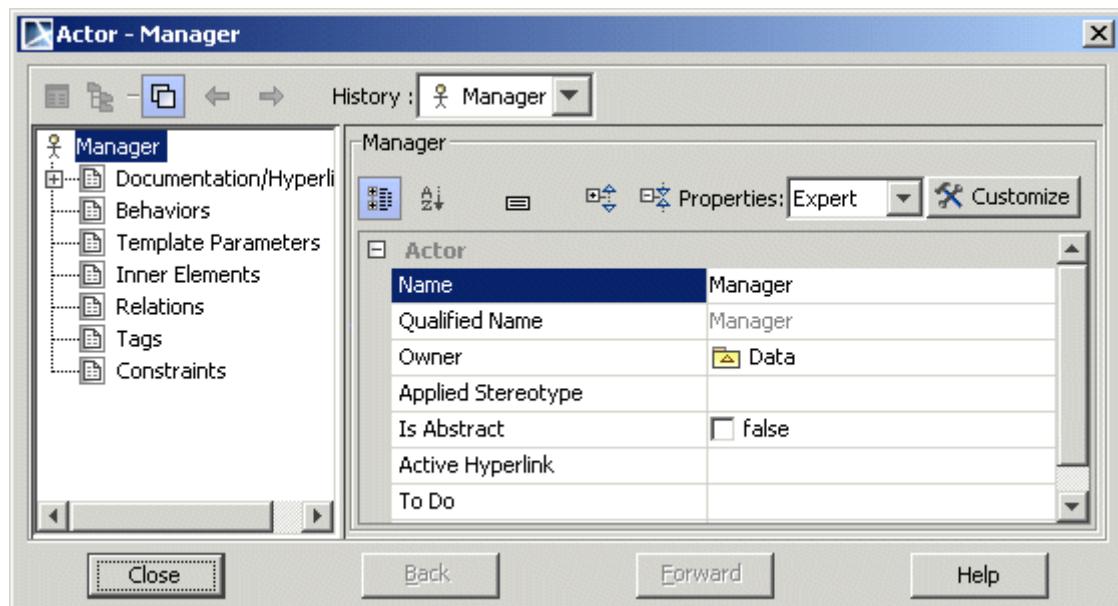


Figure 406 -- the Actor Specification dialog box

Refer to the “Specification Window” on page 237 for more information on the specification elements.

11 MODEL ELEMENTS

Association

| Tab name | Box | Function |
|---|---------------|--|
| Inner Elements Add a constraint to a actor | Name | The constraint name. |
| | Type | The constraint type. |
| | Create | The Constraint Specification dialog box opens. Define the constraint in the dialog box. |
| | Delete | Removes the selected constraint element from the actor. |

To define an actor as abstract

1. Double-click the selected actor or select **Specification** from the actor shortcut menu. The **Actor Specification** dialog box opens.
2. Select the **Is Abstract** check box in the general properties group.

To draw an actor icon in the sequence diagram

1. Drag an actor from the Browser and drop it on the diagram pane.
2. From the classifier shortcut menu, select the **Suppress Content** command.

Association

An association in the class diagrams represents the semantic relationship between two or more classifiers, which specifies connections between their instances. An association relationship is the most general of all relationships and the most semantically weak.

An association in the use case diagrams represents the participation of an actor in a use case, i.e., when instances of the actor and instances of the use case communicate with each other. This is the only relationship between actors and use cases. Sometimes an association relationship is called communication association.

An association is drawn as a solid path connecting two classifier symbols.

For a general information about working with symbols, see "Diagramming"

Specify an association in the Association Specification dialog box.

Refer to the "Specification Window" on page 237 for information about the specification elements not covered in this section.

| Tab name | Box | Function |
|---|--------------------------|--|
| General Set a general information about the association. | Association End A | |
| | Name | The name of the class A. Click the '...' button, to open the Name dialog box. |
| | Navigable | If selected, a navigable owned end of the association indicates that the association is navigable from the opposite ends, otherwise the association is not navigable from the opposite ends. |

11 MODEL ELEMENTS

Association

| Tab name | Box | Function |
|--------------------------|----------------------|---|
| | Owned By | <ul style="list-style-type: none"> If a Classifier is selected, the classifier name from the opposite association end is displayed on the association end. If an Association is selected, the name of the association to which the association end belongs is displayed on the association end. |
| | Multiplicity | The multiplicity value of an association end A. |
| | Default Value | Click the '...' button to open the Select Elements dialog box and select the value for the model elements tree. |
| Association End B | | |
| | Name | The name of the association end B. Click the '...' button, to open the Name dialog box. |
| | Navigable | If selected, a navigable owned end of the association indicates that the association is navigable from the opposite ends, otherwise the association is not navigable from the opposite ends. |
| | Owned By | <ul style="list-style-type: none"> If a Classifier is selected, the classifier name from the opposite association end is displayed on the association end. If an Association is selected, the name of the association to which the association end belongs is displayed on the association end. |
| | Multiplicity | The multiplicity value of an association end B. |
| | Default Value | Click the '...' button to open the Select Elements dialog box and select the value for the model elements tree. |

To show the direction arrow near the association name

From the association shortcut menu, select the **Show Direction Arrow** check box.

Default Direction Arrow direction is displayed according path creation direction. To change the Direction Arrow direction from the association shortcut menu, select the **Reverse Direction Arrow** command.

The Direction Arrow is graphical display most often used in top level domain class diagrams. The Direction Arrow helps to read diagram and explain diagram semantics. The Direction Arrow has no meaning in a model.

Usually Direction Arrow is used in diagram where navigability is not defined yet. Direction Arrows are usually displayed for named associations. When you move on with your modeling and create mode detail diagrams with specified navigability, direction arrows and associations names usually are not displayed in this type of diagrams.

See an example in Figure 407 on page 653 and Figure 408 on page 653. *User* and *Account* classes are connected with association. Navigation arrow may be displayed to either side, depending on the association name. If association name is "belongs to" - Direction Arrow should point from *Account* class to *User* class. If association name is "has" - Direction Arrow should point from *User* class to *Account* class.

11 MODEL ELEMENTS

Association

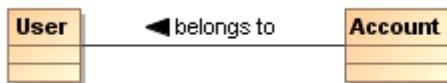


Figure 407 -- The Direction Arrow points from Account class to User class



Figure 408 -- The Direction Arrow points from User class to Account class

To draw an association class

In the class diagram, you may add attributes to an association using an association class. The association class is a simple class that has a dashed line connected to the association.

1. Draw two classes.
2. Click the **Association Class** button on the diagram toolbar.
3. On the diagram pane click the first class shape (path source).
4. Drag the path to the second class (path destination) and drop it there.

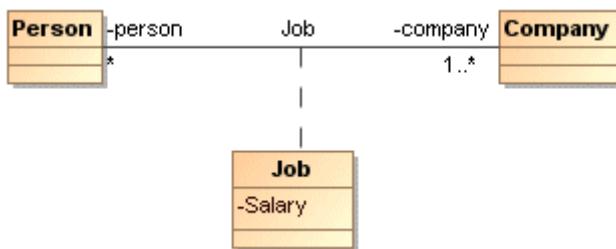


Figure 409 -- Sample of the association class

NOTE If you need to model a relationship among a number of classes, N-ary association is used.

To draw an N-ary association class

The N-ary association is drawn as a big diamond with all the associations attached to its points. Every involved class may have a role name and multiplicity.

1. Draw three classes on the Diagram pane.
2. Draw the N-ary association connector icon.
3. Connect all classes and the N-ary association connector icon using an association path.

Adding Association between Read-only Classifiers

Adding new Association always creates two roles or properties at both ends that are owned by the attached Classifier by default. However, when one or both ends of the Association is or are not editable for some reasons (for example, locked in Teamwork Server or located in a read-only profile/module), the properties will be

11 MODEL ELEMENTS

Association

owned by the Association itself. In this case, MagicDraw will display a warning informing you about the sometimes-unexpected issue of model creation (Figure 410 on page 654, Figure 411 on page 654).

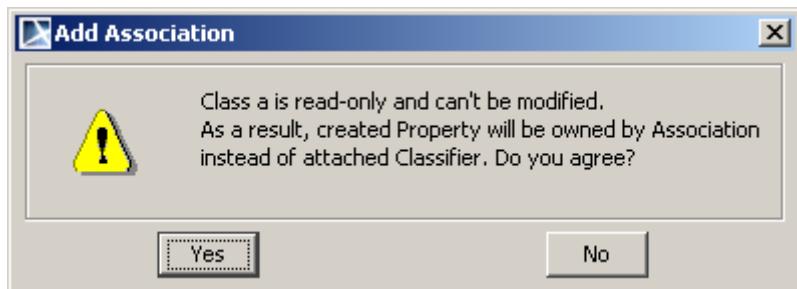


Figure 410 -- The Add Association Dialog for a Read-Only Classifier

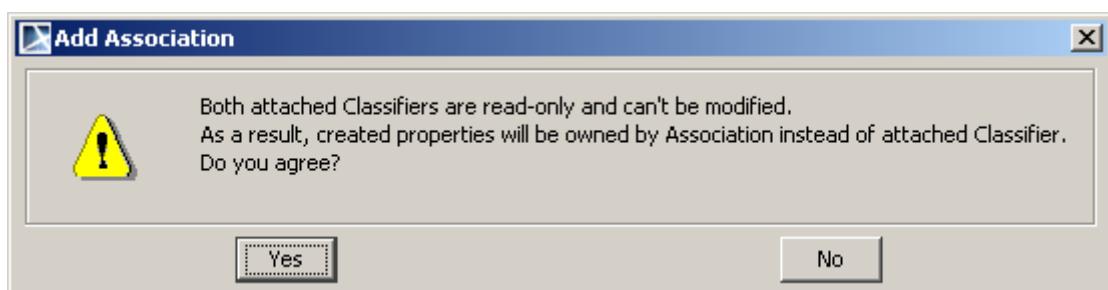


Figure 411 -- The Add Association Dialog for Both Read-Only Classifiers

Association End

The associations ends are represented by properties, each of which is connected to the type of the end. When a property is an association end, the value or values are related to the instance or instances at the other end(s) of the association.

An association end is the connection between the lines depicting an association and the icon (often a box) depicting the connected classifier.

To open the **Association End Specification** dialog box

1. Double-click the association path. The **Association Specification** dialog box appears.
2. Select the **Association Ends** group in the specification tree and expand it by clicking the plus sign.
3. Double click the selected **Association End**. The **Property Specification** dialog box opens.
 - From the association end shortcut menu, select **Specification**.

An association end is defined as a property – it has attribute properties in the **Property Specification** dialog box. For a detailed description of attributes, see “Drag and drop the selected Opaque Behavior element from the Browser tree on the Diagram pane.” on page 728.

Click the **Expert** in the **Properties** field for showing more properties of the association end.

11 MODEL ELEMENTS

Association

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box | Function |
|---|----------------------|---|
| General Set general information about the association end | Type | Shows an association end type - name of the class, to which the association end is related. |
| | Default Value | Click the ‘...’ button to open the Select Elements dialog box and select the value for the model elements tree. |
| | Owner | Shows a class name, containing an association end as the property. |
| | Type Modifier | Additional information about the type. <ul style="list-style-type: none"> • & - one class has a reference to other model elements. • * - one class has a pointer to other model elements. • [] – one class has an array of other model elements. |
| | Navigable | If selected, a navigable owned end of the association indicates that the association is navigable from the opposite ends, otherwise the association is not navigable from the opposite ends. |
| | Multiplicity | The multiplicity value of an association end A. |
| | Is Read Only | If <i>true</i> , the association end may only be read, not written. |
| | Is Static | This property scope means that the values returned by the property have no duplicates. |
| | Aggregation | When placed on a target end, specifies whether the target end is an aggregation with respect to the source end. Only one end can be an aggregation. <ul style="list-style-type: none"> • None. The end is not an aggregate. • Shared. The end is an aggregate; therefore, the other end is a part and must have the aggregation value of none. The part can be contained in other aggregates. • Composite. The end is a composite; therefore, the other end is a part and must have the aggregation value of none. The part is strongly owned by the composite and may not be part of any other composites. |
| | Is Derived | Specifies whether the Property is derived, i.e., whether its value or values can be computed from other information. |
| | Is Ordered | If the multiplicity is greater than one, then the set of related elements can be ordered or unordered. <ul style="list-style-type: none"> • <i>False</i> - the elements form an unordered set. This is the default and need not be shown explicitly. • <i>True</i> - the elements of the set are ordered, but duplicates are still prohibited. This generic specification includes all ordering types. |
| | Is Unique | If <i>true</i> , the collection of values may not contain duplicates. |

11 MODEL ELEMENTS

Association

| Tab name | Box | Function |
|--|---------------|--|
| Qualifiers Define qualifiers of the association end. A qualifier is an attribute of an association end. | Name | The name of the assigned attribute of an association end. |
| | Type | The type of the assigned attribute of an association end. |
| | Property | Shows the owner of the qualifier - to which property it belongs. |
| | Default value | The initial value of an attribute of an association end. |
| | Up | Moves item to an upper position in the list |
| | Down | Moves item to a lower position in the list. |
| | Create | The Property Specification dialog box opens. |
| | Delete | Removes the qualifier. |

To define an association end name

1. Open the **Property Specification** dialog box.
2. Type an association end name in the **Name** box.
 - Select **Role A (<class name>)** or **Role B (<class name>)** from the association shortcut menu, and then select the **Edit Name** subcommand, then type or edit the name directly on the Diagram pane.

To make a special case of an association path (aggregation/composition)

1. Open the **Property Specification** dialog box.
2. Click the desired option button for an aggregation kind (**None**, **Shared** or **Composite**).
 - From the association shortcut menu, select **Role A (<class name>)** or **Role B (<class name>)**, and then select **None**, **Shared**, or **Composite**.
 - Click the **Composition** or **Aggregation** button and draw an appropriate path on the diagram.
 - Right click the association path end and select **Shared** or **Composite** command from the shortcut menu.

To change the association navigability

The association navigability indicates whether it is possible to traverse an association within an expression of a classifier to obtain the object or set of objects associated with the instances. The navigability is shown as an arrow that can be attached to the end of the path to indicate that the navigation is supported toward the classifier attached to the arrow.

- Open the **Association End Specification** dialog box and select/clear the **Navigable** check box.
- Select **Role A (<class name>)** or **Role B (<class name>)** from the association shortcut menu, and then select/clear the **Navigable** check box..

NOTE

By default, an association is navigable on both sides and its navigability is not visible.

To display the association navigability

From both association ends shortcut menu, select the **Show Navigability** check boxes.

11 MODEL ELEMENTS

Association

In the Figure below the association is navigable on both sides and its navigability is visible.



NOTES

- To open the association ends shortcut menu, right-click on the end of the association (on the association end area) on the diagram pane.
- Open the association shortcut menu and select the Role A (<class name>) or Role B (<class name>).

To define the association end visibility

A role indicates a role played by the class in terms of an association. The role name is placed at the association end, near the class playing that role. The role name at the implementation level maps into the reference name to the opposite class. Roles may have visibility (public, package, protected, and private).

1. Open the **Property Specification** dialog box.
2. From the **Visibility** drop-down list box, choose **public**, **package**, **protected**, or **private**.

To place multiplicity values in the association path ends

The association end multiplicity describes how many entities are participating at each association end:

- 0 – zero and only zero.
- 1 – one and only one.
- 0..1 – zero or one.
- 0..* – from zero to any positive integer.
- 1..* – from one to any positive integer.
- * – any positive integer.

1. Open the **Property Specification** dialog box.
2. Type or select from the list multiplicity value (1, *, 0..*, etc.) in the **Multiplicity** box.
- Open the **Association Specification** dialog box and from the **Multiplicity** drop-down list box, select or type the multiplicity value for the desired association end.
- From the association shortcut menu, select **Role A (<class name>)** or **Role B (<class name>)**, then select the multiplicity value (1, *, 0..*, etc.).

To add, edit, or remove a qualifier to/from an association end

A qualifier is an attribute or list of attributes whose values serve to partition the set of instances associated with an instance across an association. The qualifiers are attributes of the association. It is shown as a small rectangle attached to the end of an association path between the final path segment and the symbol of the classifier that it connects to. The qualifier rectangle is part of the association path, not part of the classifier. The qualifier rectangle drags with the path segments. The qualifier is attached to the source end of the association.

1. Open the **Association End Specification** dialog box, **Qualifiers** group.
2. Click the **Create** button. The **Property Specification** dialog box opens. Define a qualifier.
3. To remove the qualifier, click the **Delete** button.

11 MODEL ELEMENTS

Association

To show the association ends as attributes on linked class shapes

If two classes are linked with an association path, both classes have a property created with an opposite class assigned as the property type in them. This property can be displayed on the class shape as well as an association link.

From the class shortcut menu, select **Symbol(s) Properties**. In the open **Properties** dialog box, the **Attribute** group, select one of the possible options near the **Show Association End** property:

- **All** - property and association paths will be displayed on the diagram pane.
- **Without Association Symbol** - if an association symbol is deleted, the property will be displayed on the class shape.
- **Do Not Show** - neither property, nor association path will be displayed on the diagram pane.

Advancing actions: navigable owned association ends

Navigability describes the need for an object to access another object by navigating across the link. According to the UML 2 specification, the association ends owned by the classes and associations navigable. This improved functionality allows a proper management of the `navigableOwnedEnd` property for associations:

1. Able to manually change the ownership of an association end.

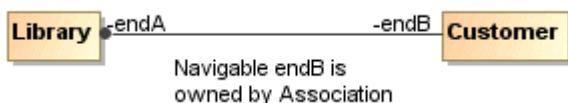


Figure 412 -- Sample of navigable endB, which is owned by association

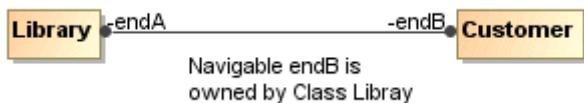


Figure 413 -- Sample of navigable endB, which is owned by Class Library

To change the ownership of an association end, select the **Owned By** command from the **Association End** shortcut menu and then select the desired owner.

2. Set the navigability for the association ends owned by the associations while keeping the ownership.

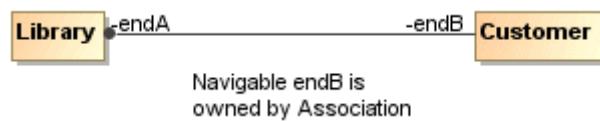


Figure 414 -- Sample of navigable endB, which is owned by Association

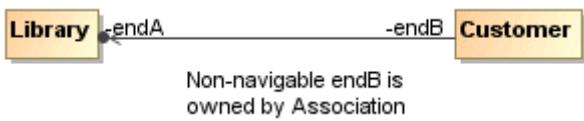


Figure 415 -- Sample of Non-navigable endB, which is owned by Association

3. Support the dot notation.

11 MODEL ELEMENTS

Association in Use Case Diagrams

The ownership of association ends by an associated Classifier is now indicated graphically.

The following example shows that endA is owned by the Customer class and endB is owned by the association.

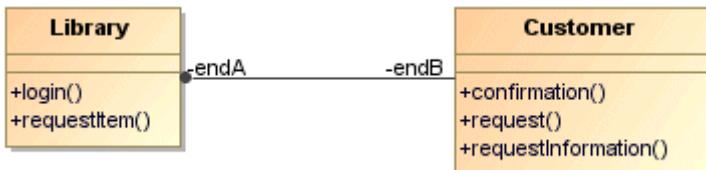


Figure 416 -- Sample of endA owned by class and endB owned by association

To enable the dot notation from the **Options** menu select the **Project** command and then select the **General project options** branch. Next, select the **Enable dot notation for associations** check box.

Association in Use Case Diagrams

The participation of an actor in a use case, i.e. instances where an actor and a use case communicate with each other. This is the only relationship between actors and use cases. The association relationships are also known as communication associations.

For more information on defining associations, see “Realization” on page 757.

Attribute

An attribute is a named property of a class that describes a range of values that can be held by the instances of that class.

To create a new attribute

- Double-click the selected class or select **Specification** from the class shortcut menu. The **Class Specification** dialog box opens. Click the **Attributes** tab and then click the **Create** button. The **Property Specification** dialog box opens. Define a new attribute and click **OK**.
- Select the **Insert New Attribute** from the class shortcut menu. Type the attribute name directly on the class shape.
- In the Browser tree, select an already created class. From the class item shortcut menu, select **New** and then **Property**.
- Press **CTRL+ALT+A** shortcut key and type the attribute name on the Diagram pane.
- Select a class shape and click the small orange **Insert New Attribute** smart manipulation button.

Define an attribute in the **Property Specification** dialog box.

To open the **Property Specification** dialog

1. Open the **Class Specification**, **Actor Specification**, or **Interface Specification** dialog box.
2. In the **Attributes** tab, double-click the desired attribute in the tree, or click the **Create** button.
 - Draw an association path and click on the association end.

11 MODEL ELEMENTS

Attribute

- Double-click the desired attribute directly on the diagram.

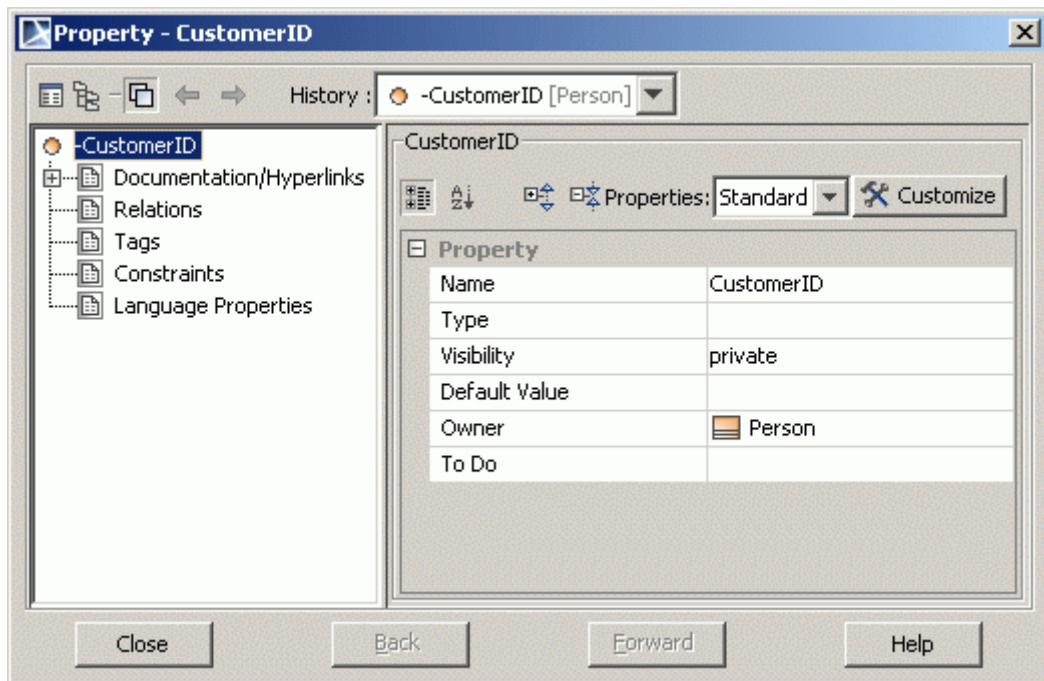


Figure 417 -- Property Specification dialog box

Click the **Expert** in the **Properties** field for showing more properties of the attribute.

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Options | Function |
|--|----------------------|---|
| General Set a general information for the attribute. | Type | Shows an attribute type. It can be another project class or primitive type such as int or double or other. Select a type from drop-down combo box or click “...” button and select the type from the Select Element tree. |
| | Default Value | Type a value for an attribute or click “...” button and enter it in the Default Value dialog box. |
| | Type Modifier | Additional information about the type. <ul style="list-style-type: none">• & - one class has a reference to other model element.• * - one class has a pointer to other model element.• [] – one class has an array of other model elements. |
| | Navigable | Indicates whether it is possible to navigate across the property. |
| | Multiplicity | The multiplicity value of an attribute. |
| | Is Read Only | If <i>true</i> , the property may only be read, and not written. |
| | Is Static | This property scope means that the values returned by the property have no duplicates. |

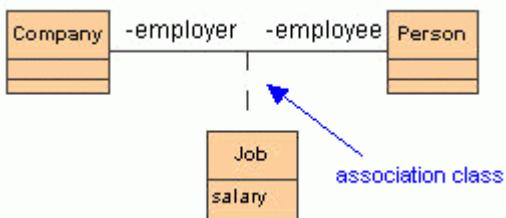
11 MODEL ELEMENTS

Attribute

| Tab name | Options | Function |
|----------|--------------------|--|
| | Aggregation | When placed on a target end, specifies whether the target end is an aggregation with respect to the source end. Only one end can be an aggregation. <ul style="list-style-type: none"> • None. The end is not an aggregate. • Shared. The end is an aggregate; therefore, the other end is a part and must have the aggregation value of none. The part can be contained in other aggregates. • Composite. The end is a composite; therefore, the other end is a part and must have the aggregation value of none. The part is strongly owned by the composite and may not be part of any other composite. |
| | Is Derived | Specifies whether the property is derived, i.e., whether its value or values can be computed from other information. |
| | Is Ordered | If the multiplicity is greater than one, then the set of related elements can be ordered or unordered. <ul style="list-style-type: none"> • <i>False</i> - the elements form an unordered set. This is the default and need not be shown explicitly. • <i>True</i> - the elements of the set are ordered, but duplicates are still prohibited. This generic specification includes all ordering types. |
| | Is Unique | If <i>true</i> , the collection of values may not contain duplicates. |

To create an association class

1. Draw two classes (for example A and B).
2. From the class diagram toolbar, expand the **Association** elements group and select the **Association Class** to draw.
3. Link the previously drawn classes with this path. An additional class with a dashed line will be created on the association between classes.



If an attribute type is another model class, this attribute can be represented as an association with a role (attribute name) between the owner class and the class of attribute type

Select **Create Roles** from the class shortcut menu. A list of available attributes opens. Select all or only one and an association relationship with a role is created.

NOTE

This command is visible only if one or more attribute types are other model classes.

To change an attribute name

The attribute name must be unique in the class scope.

1. Click the attribute in the selected class on the diagram pane or in the Browser tree.
2. Type a new name.

11 MODEL ELEMENTS

Attribute

- Change an attribute name in the **Attribute Specification** dialog box.

To define the type of an attribute

The attribute type can be of the other class, interface, or a primitive class, such as int or double.

- Select the type of an attribute from the **Type** drop-down list box in the **Attribute Specification** dialog box.
- Type a colon ":" and the name of the attribute type just after the attribute name on the diagram pane. If you specify a nonexistent type of an attribute, a new class is created.
- Click the "..." button in the **Type** field and select they type from the **Select Elements** tree.

To add additional information about the type of an attribute

1. Open the **Attribute Specification** dialog box.
2. Click the **Show Expert Properties** button to enlarge a list of general available attribute properties.
3. Select a sign in the **Type Modifier** drop-down list box:
 - & - one class has a reference to other model elements.
 - * - one class has a pointer to other model elements.
 - [] - one class has an array of other model elements.

To set the attribute visibility

| Visibility Name | Function |
|-----------------|---|
| Public '+' | an attribute can be accessed by any other elements. |
| Package '~' | an attribute can be accessed by elements from the same package. |
| Protected '#' | an attribute can be accessed from the inside of the selected class and classes derived from that class. |
| Private '-' | <ul style="list-style-type: none">• Type '+', '~', '−', or '#' visibility marks just before an attribute name directly on a diagram. <ol style="list-style-type: none">1. Open the Attribute Specification dialog box.2. From the Visibility drop-down list box, select the desired item (public, package, protected, and private.). |

NOTE

The attribute visibility is shown at the attribute signature.

To set an attribute scope

1. Open the **Attribute Specification** dialog box.
2. Click the **Show Expert Properties** button to enlarge the list of available attribute properties.
3. Select the **Is Static** check box.

To set the attribute multiplicity

1. Open the **Attribute Specification** dialog box (see above).
2. Click the **Show Expert Properties** button to enlarge the list of available attribute properties.
3. Select or set the multiplicity value in the **Multiplicity** drop-down box.

11 MODEL ELEMENTS

Class

To set the attribute changeability

The attribute changeability controls the access by operations on the class on the opposite end.

| Name | Function |
|---------------------|---|
| Is Read Only | When <i>false</i> - no restrictions on modifications. When <i>true</i> - the value may not be altered after the object is instantiated and its values initialized. No additional values can be added to a set. 1. Open the Attribute Specification dialog box (see above). 2. Click the Show Expert Properties button to enlarge the list of available attribute properties. 3. Select/clear the Is Read Only check box. |

Class

A class is drawn as a solid-outline rectangle with three compartments separated by horizontal lines. The top name compartment holds the class name and other general properties of the class (including stereotype); the middle list compartment holds a list of properties; the bottom list compartment holds a list of operations. The property and operation compartments are optional and you may suppress them.

A class is the descriptor for a set of objects with similar structure, behavior, and relationships. The model is concerned with describing the intention of the class, that is, the rules that define it. The run-time execution provides its extension, that is, its instances.

Classes are declared in the class diagrams and used in most of other diagrams. UML provides a graphical notation for declaring and using these classes as well as a textual notation for referencing classes within the descriptions of other model elements.

A class represents a concept within the system being modeled. It has a data structure, behavior, and relationships to other elements. The name of a class has a scope within the package in which it is declared and the name must be unique (among class names) within its package.

Working with classes

A general information about working with shapes is offered in the Section “Diagramming” on page 167.

All options associated with a class can be set in the **Class Specification** dialog box.

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box | Function |
|---|----------------------------|---|
| General Set general information about the class | Base Classifiers | Click “...” and assign an existing class from a model in the Select Elements dialog box or create a new one. |
| | Realized Interfaces | Click “...” and assign an existing interface from a model in the Select Elements dialog box or create a new one. |

11 MODEL ELEMENTS

Class

| Tab name | Box | Function |
|--|-------------------|--|
| Ports | Name | Name of the class port. |
| | Type | Type, assigned to the port. |
| | Provided | Provided classifier is displayed. |
| | Required | Required classifier is displayed. |
| | Classifier | Name of classifier, owning port. |
| | Create | Creates a new port. |
| | Delete | Removes an existing port from the class. |
| Signal Receptions | | Manage the receptions of a class. For more information about receptions, see "Reception" on page 759. |
| Behaviors | Name | Name of the behavior. |
| | Type | Type, assigned to the behavior. |
| | Create | Choose item from the list - activity, interaction, state machine. |
| | Delete | Removes a behavior from the class. |
| Inner Elements Add another element to a . | Name | Model element name. |
| | Type | Model element type. |
| | Create | The corresponding specification dialog box opens. Define the selected model element in the dialog box. |
| | Delete | Removes the selected model element from the class. |
| | | |

To insert an inner element in the selected class

1. Double-click the selected class or select **Specification** from the class shortcut menu. The **Class Specification** dialog box opens.
2. Click the **Inner Elements** tab and then click the **Create** button or press Insert. Select the element you wish to add from the list.
3. Click the selected element.
4. The corresponding **Specification** dialog box opens. Define the class, use case or interface, and click **OK**.

To show an inner element on the diagram

1. Select a created inner element in the Browser.
2. From the element shortcut menu, select **Create Symbol**.

To generate operations for setting or getting private data to the selected class

From the class shortcut menu, select **Tools** and then **Create Setters/Getters**. For a detailed description, see "Creating Setters / Getters" on page 348.

To control a list of operations and attributes that are visible on a diagram

Select **Edit Compartment** from the class shortcut menu. The **Compartment Edit** dialog box opens.

A class can be defined as active (a border to the class shape is added). An active class specifies whether an object of the class maintains its own thread of control.

A class is a generalizable element and can be defined as Abstract.

To define a class as abstract and/or active

1. Double-click the selected class or select **Specification** from the class shortcut menu. The **Class Specification** dialog box opens.
2. Select the **Is Abstract**, and/or **Is Active** check box in the **General** tab.

To show members (attributes and operations) on the classifier shape according to the visibility

From the shape shortcut menu, point to **Presentation Options**, and then select one of the check boxes in the **Show Members** subcommand. Possible choices:

- All;
- Only Public;
- Not Private.

Creating A Structured Class

There is a way to create a piece of model with a single click: class, with a composite structure diagram inside it and hyperlink them. It may be useful for architects and system engineers.

The same applies to SysML Block and IBD.

To create a structured class:

1. Create a class diagram.
2. In the class diagram toolbar, right-click the **Class** button. A menu with available options opens. Select the **Structured Class** element.
3. Click on the diagram. The class element with a hyperlink to a composite structure diagram is created.
4. Select the class in the Browser to see its structure. To do this, choose **Select in Containment Tree** on the diagram from the class shortcut menu.

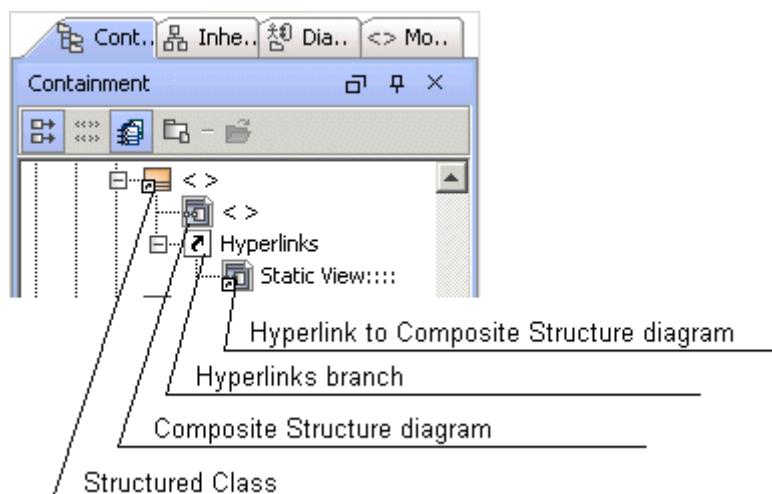


Figure 418 -- Example of structured class

See the above example of a structured class. In the Browser you may see the created class with the composite structure diagram inside it. The class is linked with the diagram. It means that after double click the diagram, or the Browser, the composite structure diagram will be opened.

For more information about hyperlink,s see “Defining Hyperlinks Between Elements” on page 270.

The names of the class and the composite structure diagram are synchronized. Type a name for the class, for example, Order, and the name of the diagram is automatically changed to Order and vice versa. This is synchronization of a diagram name and its context name.

For more information about diagram name and its context synchronization, see “Diagram Name and its Context Name Synchronization” on page 173.

Design Patterns

In MagicDraw, you may create and edit the design patterns for the selected class. A detailed description of templates can be found in the Design Patterns of Reusable Object-Oriented Software.

To create the design pattern for the selected class

1. From the class shortcut menu, select **Tools**, and then **Apply Pattern**. The **Pattern Wizard** dialog box opens.
2. Select the design pattern you want to apply and select the desired options. Click **OK**.
 - Select the class and then **Apply Pattern** from the **Tools** menu.

For a detailed description of this dialog box, see Section “Controlling Merge memory usage” on page -341.

Class presentation options

To organize a class data on the class shape

- Select **Presentation Options** from the class shortcut menu. The following choices are available in the **Presentation Options** submenu.
- The class presentation options can also be defined in the **Project Options** dialog box. For a detailed description of this dialog box, see Section “Project Options” on page -132.

11 MODEL ELEMENTS

Class

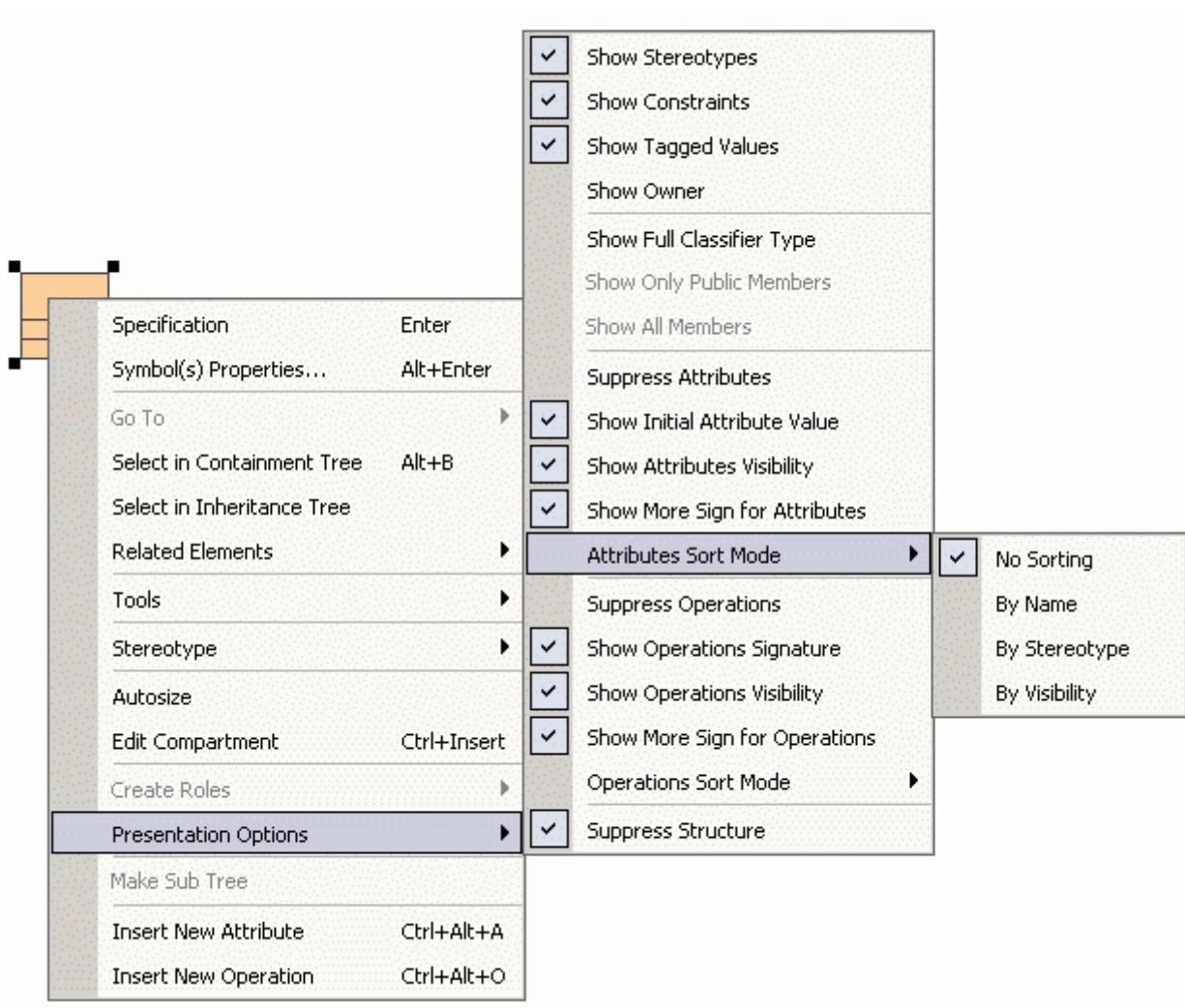


Figure 419 -- Presentation Options submenu

| Command | Function (when selected) |
|-------------------------------------|--|
| Show Operations Signature | Shows the operation signature (arguments, return value, etc.). |
| Show Full Classifier Type | Shows the full track of the type of an attribute from the root package. |
| Show Initial Attribute Value | Shows the initial attribute value. |
| Show Attributes Visibility | Shows the attribute visibility (public, package, private, or protected). |
| Show Operations Visibility | Shows the operation visibility (public, package, private, or protected). |
| Suppress Attributes | Attributes compartment is suppressed. |
| Suppress Operations | Operations compartment is suppressed. |
| Show Only Public Members | Shows only the public attributes and operations. |
| Show all Members | The default settings are restored and shows all attributes and operations. |

11 MODEL ELEMENTS

Collaboration

| Command | Function (when selected) |
|---|--|
| Attributes Sort Mode <ul style="list-style-type: none">● By Name● By Stereotype● By Visibility● No Sorting | Choose the sorting parameter: <ul style="list-style-type: none">● Sort attributes by name.● Sort attributes by stereotype● Sort attributes by visibility (public, package, private, or protected)● No sorting is executed. |
| Operations Sort Mode <ul style="list-style-type: none">● By Name● By Stereotypes● By Visibility● No Sorting | Choose the sorting parameter: <ul style="list-style-type: none">● Sort operations by name.● Sort operations by stereotype.● Sort operations by visibility (public, package, private, or protected)● No sorting is executed. |
| Show More Sign For Attributes | Additional information sign '...' in the class attributes list, when a portion of attributes are omitted by editing a class compartment. |
| Show More Sign For Operations | Additional information sign '...' in the class operations list, when a portion of operations are omitted by editing a class compartment. |
| Show Stereotypes | Shows the stereotypes on a class. |
| Show Constraints | Shows the constraints on a class. |
| Show Tagged Values | Shows the tagged values on a class. |
| Show Owner | Shows the owner's (package, subsystem, or model) name on a class. |

Collaboration

A collaboration is represented as a kind of classifier. It defines a set of cooperating entities to be played by instances (its roles) as well as a set of connectors that define the communication paths between the participating instances. The cooperating entities are the properties of the collaboration.

A collaboration specifies a view (or projection) of a set of cooperating classifiers. It describes the required links between instances that play the roles of the collaboration as well as the required features of the classifiers that specify the participating instances. Several collaborations may describe different projections of the same set of classifiers.

11 MODEL ELEMENTS

Collaboration

Define a collaboration in the **Collaboration Specification** dialog box.

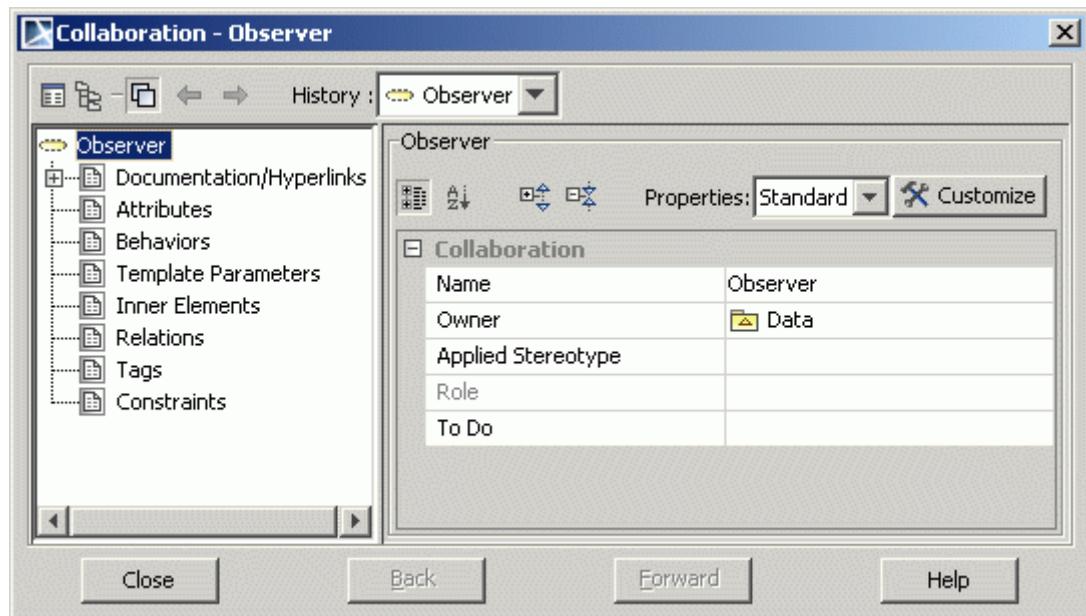


Figure 420 -- Collaboration Specification dialog box

Refer to the “Specification Window” on page 237 for the information of the specification elements not covered in this section.

| Tab name | Box | Function |
|--|---------------|--|
| Behaviors Any behavior attached to the collaboration type applies to the set of roles and connectors bound within a given collaboration use. | Name | Name of the added behavior. |
| | Type | Type of the behavior |
| | Create | Select the kind of behavior you want to create - Activity, Interaction, or State Machine. The corresponding Specification dialog box opens. |
| | Delete | Remove the selected behavior from the collaboration. |
| Inner Elements | Name | Name of the inner collaboration element. |
| | Type | Type of the inner collaboration element. |
| | Create | Select an element from the list. The corresponding Specification dialog box opens. Define the element and click Back to return to Collaboration Specification . |
| | Delete | Remove the inner element from the collaboration. |

To assign a new behavior to the collaboration

- In the **Collaboration Specification** dialog box, **Behaviors** tab, click the **Create** button. Define a behavior in the open **Specification** dialog box.
- From the collaboration shortcut menu in the Browser, select **New**, and then **Activity**, **Interaction**, or **State Machine**. Type the name of the behavior and modify it in the open **Specification** dialog box.

NOTE

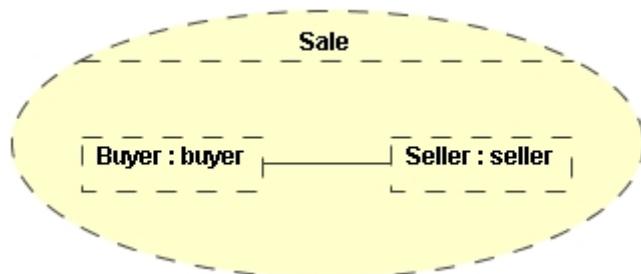
You can edit the assigned classifier role in the **Collaboration Specification** dialog box, **Behaviors** tab.

11 MODEL ELEMENTS

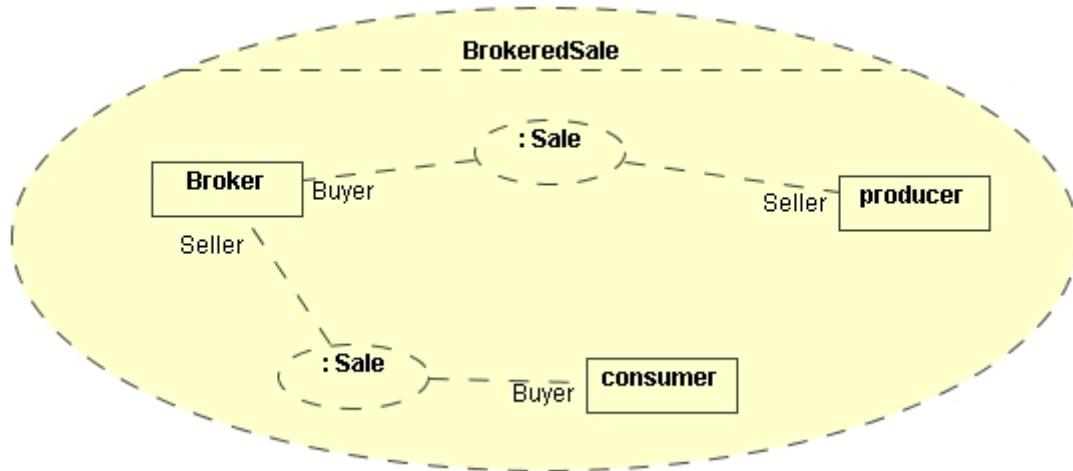
Combined Fragment

To draw the Collaboration

1. From the Composite Structure diagram toolbar, select the Collaboration to draw on the diagram pane and type the collaboration name.
2. Draw Part from the Composite Structure diagram toolbar and from the shortcut menu, **Type** submenu, select an already existing (or create a new) classifier as the part type.
3. From the Composite Structure diagram toolbar, select the Connector to connect the parts.



4. From the Composite Structure diagram toolbar, select the Collaboration Use element to draw on the collaboration. Select an already existing, or create new collaboration for the Collaboration Use.
5. From the Composite Structure diagram toolbar, select the Role Binding path to connect the parts and Collaboration Use role, which can be selected in the **Select Role** dialog box.



Combined Fragment

A fragment is an abstract notion of the most general interaction unit. It is a piece of an interaction. Each interaction fragment is conceptually like an interaction by itself. Using the Combined Fragment, a fragment of a sequence diagram can be separated.

MagicDraw represents twelve kinds of fragments: Alternatives, Loop, Option, Parallel, Break, Negative, Critical Region, Consider, Ignore, Weak Sequencing, Strict Sequencing, and Assertion.

| Tab name | Property | Function |
|----------|----------------------|---|
| General | Interaction Operator | Select an operator for the fragment. Available options: seq, alt, opt, break, par, strict, loop, critical, neg, assert, ignore, consider. |

11 MODEL ELEMENTS

Component

| Tab name | Property | Function |
|--|---------------|--|
| Formal Gates Actual Gates | | In the Formal Gates and Actual Gates panes are listed inside or outside combined fragment created gates. For more information about gates, see “Gate” on page 691. |
| Operands | Guard | Separated alternative parts of the interaction fragment. |
| | Create | Creates a new guard value. |
| | Delete | Removes a value from the list. |
| | Up | Moves an item to an upper position in the list. |
| | Down | Moves an item to a lower position in the list. |

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

Component

A component represents all kinds of elements pertaining to piecing together software applications. A component can always be considered as an autonomous unit within a system or subsystem.

According to UML it is possible to list component properties. The following compartments are available for the component:

- Provided/required interfaces.
- Realizations.

Artifacts.

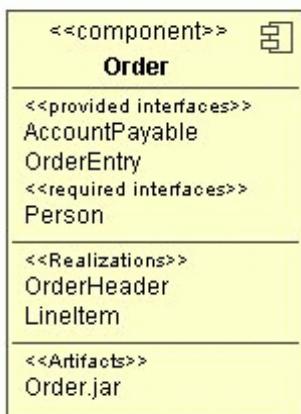


Figure 421 -- An example of a component

For more information about working with symbols, see “Diagramming” on page 167.

11 MODEL ELEMENTS

Component

Define the selected component in the component's Specification window.

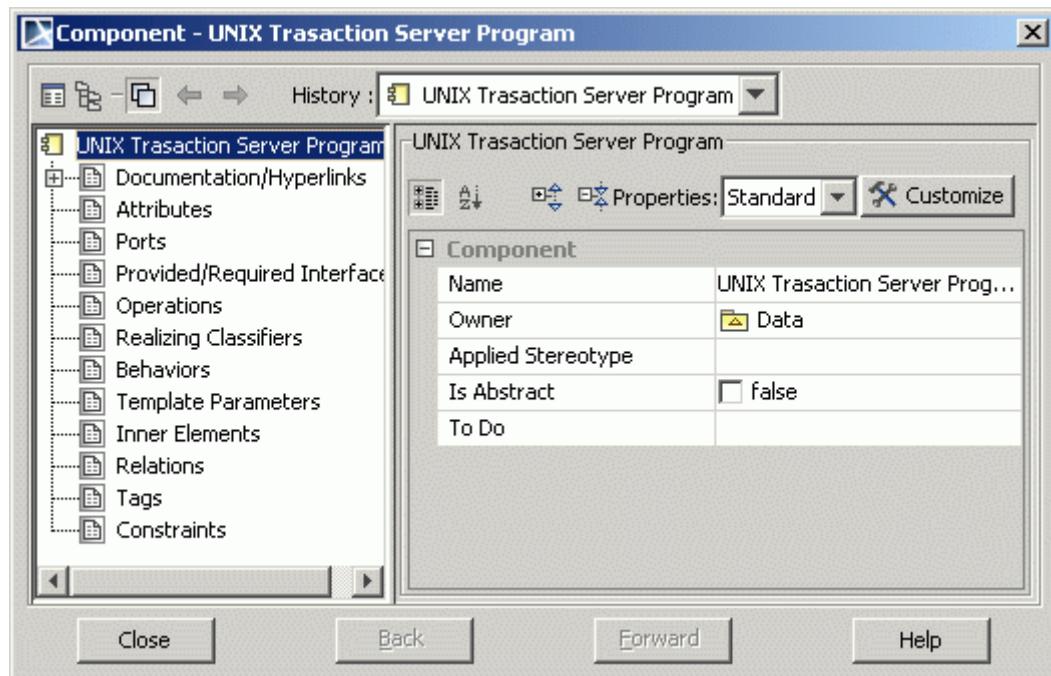


Figure 422 -- Component's Specification window[UPDATE]

For information about the Specification window elements, which are not covered in this section, refer to the "Specification Window" on page 237 .

| Tab name | Element | Description |
|------------------------------|------------|---|
| Ports | Name | Name of the port. |
| | Type | Type, assigned to the port. |
| | Provided | The provided classifier is displayed. |
| | Required | The required classifier is displayed. |
| | Classifier | Name of the classifier, owning a port. |
| | Create | Creates a new port. |
| | Delete | Removes an existing port from the component. |
| Provided/Required Interfaces | Name | Name of the provided/required interface. |
| | Type | Possible values - provided or required. |
| | Add | Adds the provided/required interface. |
| | Remove | Removes the provided/required interface from the list. |
| Realizing Classifiers | Name | Name of the realizing classifier |
| | Type | Type of the classifier (class, interface, etc.) |
| | Owner | The model element name, owning a classifier. |
| | | Click button to open the classifier's Specification window. |
| | Add | The Select Elements dialog box opens. Select an element from the model element tree. |
| | Remove | Removes a realizing classifier from the list. |

11 MODEL ELEMENTS

Connector

| Tab name | Element | Description |
|--|---------------|---|
| Behaviors Add another element to a component. | Name | Name of the behavior. |
| | Type | Type, assigned to the behavior. |
| | Create | Select an item from the list - activity, interaction, state machine. |
| | Delete | Removes a behavior from the class. |
| Inner Elements Add another element to a component. | Name | The model element name. |
| | Type | The model element type. |
| | Create | Select an element from the list. The corresponding specification dialog box opens. Define the selected model element in the dialog box. |
| | Delete | Removes the selected model element from the component. |

To show/hide the interfaces, realizations, and artifacts on the component's shape

- From the component's shape shortcut menu, select **Presentation Options** and then clear/ select **Suppress Interfaces**, **Suppress Realizations**, or **Suppress Artifacts** check box.
- From the component's shape shortcut menu, select **Symbol(s) Properties**. In the opened component's **Symbol Properties** dialog, change the same check box values.

Connector

Specifies a link that enables communication between two or more instances. This link may be an instance of an association, or it may represent the possibility of the instances being able to communicate because their identities are known by virtue of being passed on as parameters, held in variables or slots, or because the communicating instances are the same instance.

The link may be realized by something as simple as a pointer or by something as complex as a network connection. In contrast to the associations, which specify the links between any instance of the associated classifiers, the connectors specify the links between instances playing the connected parts only.

Each connector may be attached to two or more connectable elements, each representing a set of instances.

For more information about working with symbols, see "Diagramming"

11 MODEL ELEMENTS

Connector

Define the selected connector in the **Connector Specification** dialog box.

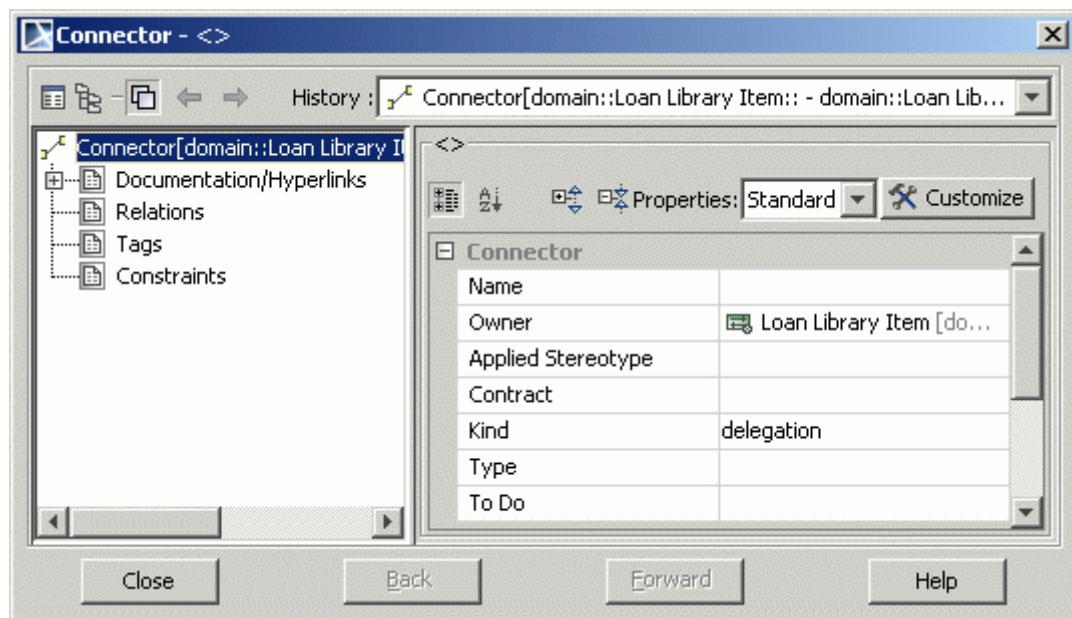


Figure 423 -- the Connector Specification dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box | Function |
|--|-----------------|--|
| General Set a general information about the include relationship. | Owner | The name of the element, which owns a connector. |
| | Contract | Assign an activity, interaction, or state machine to a connector. Click the '...' button and select the elements in the Select Elements dialog box. |
| | Kind | Possible choices: <ul style="list-style-type: none">• delegation (default)• assembly |

11 MODEL ELEMENTS

Connector

To open the **Connector End Specification** dialog box

1. In the Browser tree, expand Interaction, **Relations** branch and in the Connector group, select one of the roles. Select **Specification** from the shortcut menu, or double-click the role. The **Connector End Specification** dialog box opens.

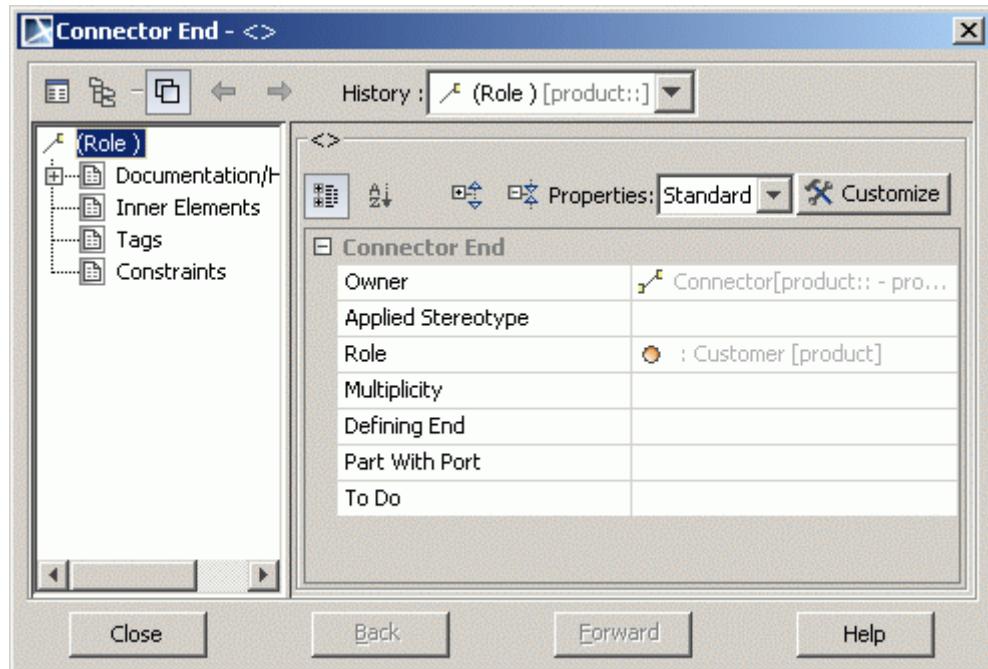


Figure 424 -- the *Connector End Specification* dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

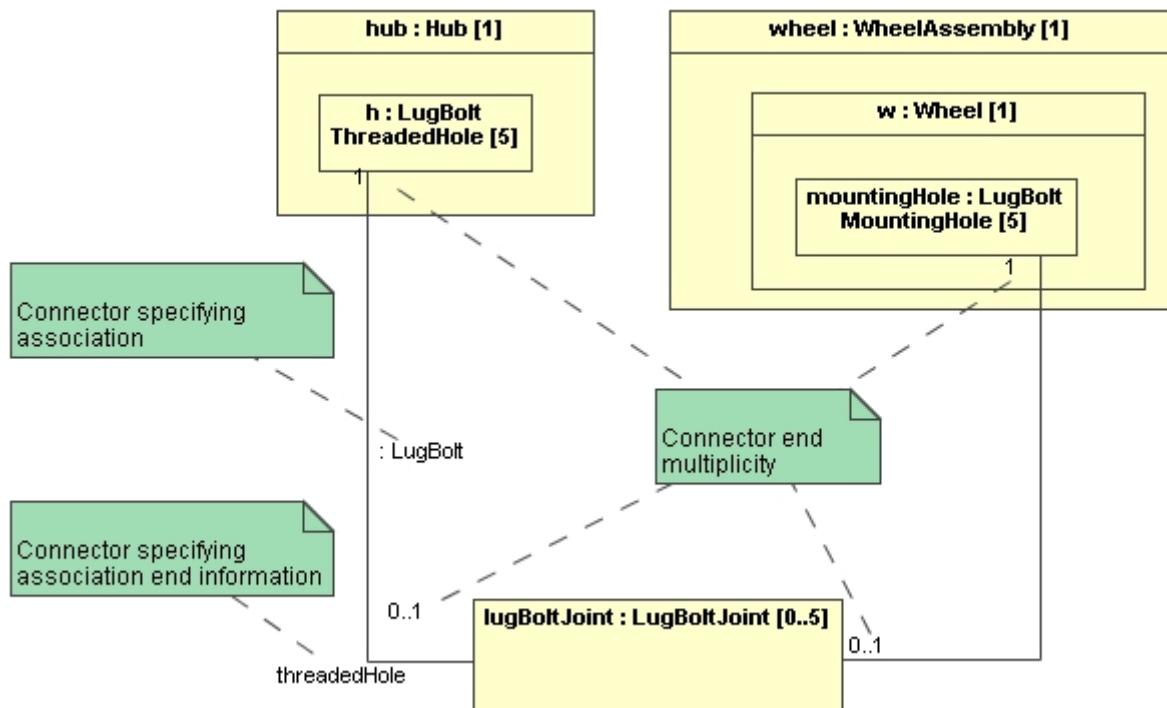
| Tab name | Box name | Function |
|---|-----------------------|---|
| General Set general information about the link end | Owner | The name of the connector, which owns the connector end. |
| | Role | The name of the property, representing a lifeline. |
| | Multiplicity | The multiplicity value of the connector end. |
| | Defining End | When a connector has an association specifying the type of the connector, connector end, this property refers to the appropriate association end. |
| | Part With Port | Refers to the part containing the port that the connector is attached to. |

11 MODEL ELEMENTS

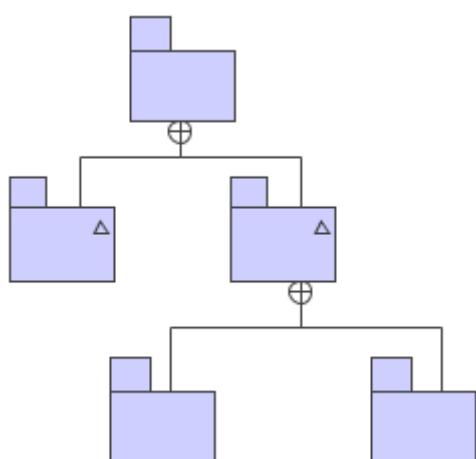
Containment

Drawing a connector end

The connector end information can be displayed on the diagram.



Containment



A containment shows a class, package or other model element declared within another model element. Such a declared class is not a structural part of the enclosing class but merely has scope within the namespace of the enclosing class, which acts like a package toward the inner class.

A data type is a type whose values have no identity; that is, they are pure values. It is a classifier and inherits the general features of the classifier: visibility, generalizable element properties, and operations.

MagicDraw provides the following predefined data types: boolean, byte, char, date, double, float, int, Integer, long, short, void, and String.

You may also create Enumeration or Primitive data types.

To create a new Data Type, including Enumeration or Primitive

- From the Browser, select **New** from the class or **New Element** from the package, subsystem, or model shortcut menu, and then select **Data Type**, **Enumeration**, or **Primitive**.
- In the **Class**, **Package**, **Subsystem**, and **Model Specification** dialog boxes, **Inner Elements** tab, click **Create** and select a data type.

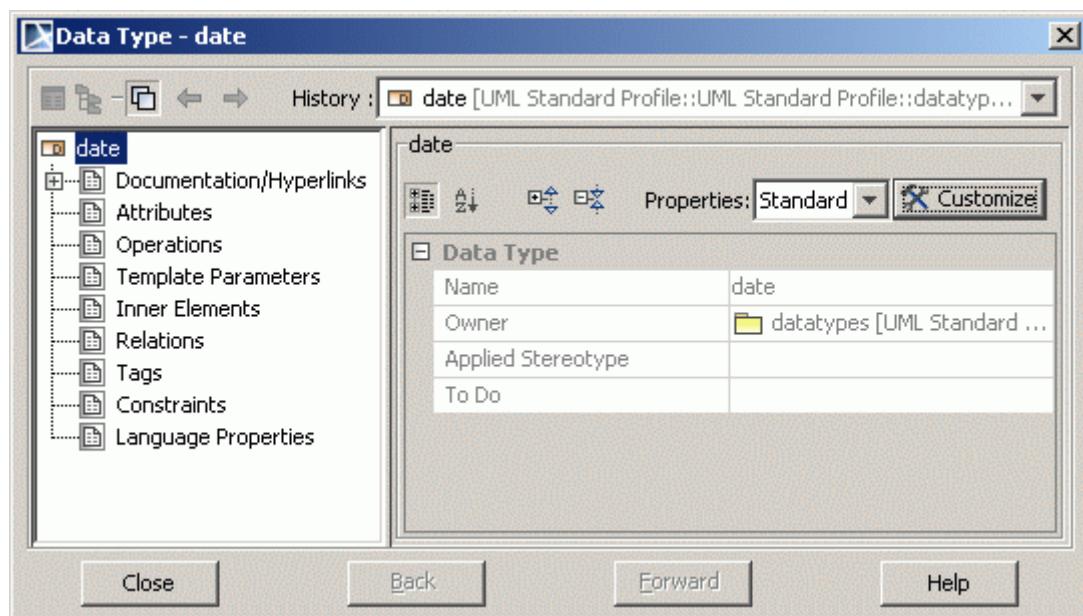
NOTE:

You may create an enumeration by clicking the **Enumeration** button from the class diagram toolbar:



To create a symbol of the created data type

From the created data type shortcut menu in the Browser, select the **Create Symbol** command.



11 MODEL ELEMENTS

Data type

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box | Function |
|---|---------------|--|
| General Set general information about the data type | Owner | Shows an element, which contains the data package. The default data packages contain the <i>UML Standard Profile</i> . |
| Inner Elements Add constraint to data type. | Name | Displays a constraint name. |
| | Type | Shows an item type, in this case Constraint. Click the button to open the Constraint Specification dialog box. |
| | Create | The Constraint Specification dialog box opens. Define the constraint. |
| | Delete | Removes the selected constraint from the inner elements list. |

Enumeration

The enumeration defines a kind of data type whose range is a list of predefined values, called enumeration literals. An Enumeration may contain operations, but they must be pure functions (this is the rule for all data type elements).

Define an enumeration in the **Enumeration Specification** dialog box.

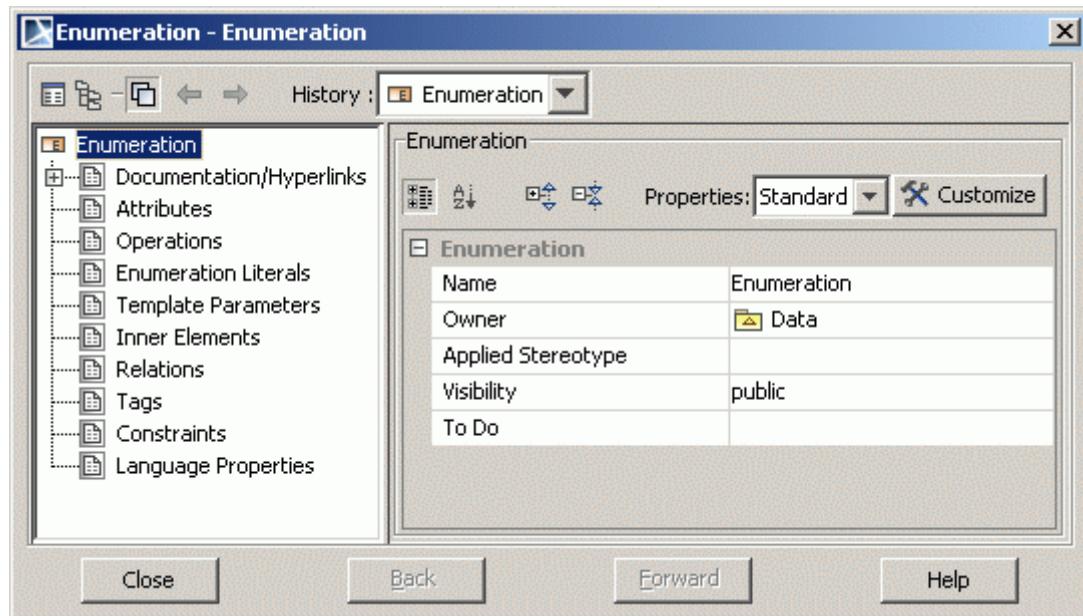


Figure 425 -- Enumeration Specification dialog box

11 MODEL ELEMENTS

Data type

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box | Function |
|---|--------------------|--|
| General Set a general information about the enumeration | Owner | Shows the name of the element, which contains an enumeration. |
| Enumeration Literals An enumeration literal defines an element of the run-time extension of an enumeration data type. | Name | The name of the enumeration literal. |
| | Enumeration | The owner of the enumeration literal. Click the button to open the Enumeration Literal Specification dialog box and edit the information about the enumeration literal. |
| | Create | The Enumeration Literal Specification dialog box opens. Define a new enumeration literal. |
| | Delete | Removes the selected enumeration literal from the enumeration. |
| Inner Elements Add a constraint to an enumeration | Name | Displays a constraint name. |
| | Type | Shows an item type, in this case Constraint. Click the button to open the Constraint Specification dialog box. |
| | Create | The Constraint Specification dialog box opens. Define the constraint. |
| | Delete | Removes the selected constraint from the inner elements list. |

To add an enumeration literal

An enumeration literal defines an element of the run-time extension of an Enumeration data type. It has no relevant substructure, therefore, it is atomic.

1. Open the **Enumeration Specification** dialog box.
2. In the **Enumeration Literals** tab, click **Create** button.
3. The **Enumeration Literal Specification** dialog box opens. Define an enumeration literal. Click **Back** to return to **Enumeration Specification** dialog box.
 - Choose the **Insert New Enumeration Literal** command from the **Enumeration** shortcut menu.

To suppress the enumeration literals

1. From the enumeration shortcut menu, select **Presentation Options**.
2. Select the **SUPPRESS ENUMERATION LITERALS** check box.
 - Use the smart manipulation button with minus sign on the diagram pane, enumeration symbol.

To open the **Enumeration Literal Specification** dialog box

1. Open the **Enumeration Specification** dialog box.
2. In the **Enumeration Literals** tab expand tree, double-click the desired literal, or click the **Edit**, or **Create** button.

Primitive

A primitive defines a predefined data type without possessing any relevant UML substructure; that is, it has no UML parts. A primitive data type may have an algebra as well as operations defined outside of UML (for example, mathematically). The primitive data types used in UML include Integer, Unlimited Integer, and String.

Decision Node

Decisions are made using guard conditions. They help protect transitions that depend on a guarding condition. The symbol used for the decision is a large diamond shape, which may have one or more incoming transitions and two or more outgoing transitions.

A decision in an activity diagram is used much like a choice or junction point in the state diagrams. Decision points allow to separate the transition paths. Merges allow to merge the transition paths back together. The symbol used for the merge is the same as for the decision..

Dependency

A dependency is a relationship signifying that a single or a set of model elements require

other model elements for their specification or implementation. This means that the complete semantics of the depending elements is either semantically or structurally dependent on the definition of the supplier element(s).

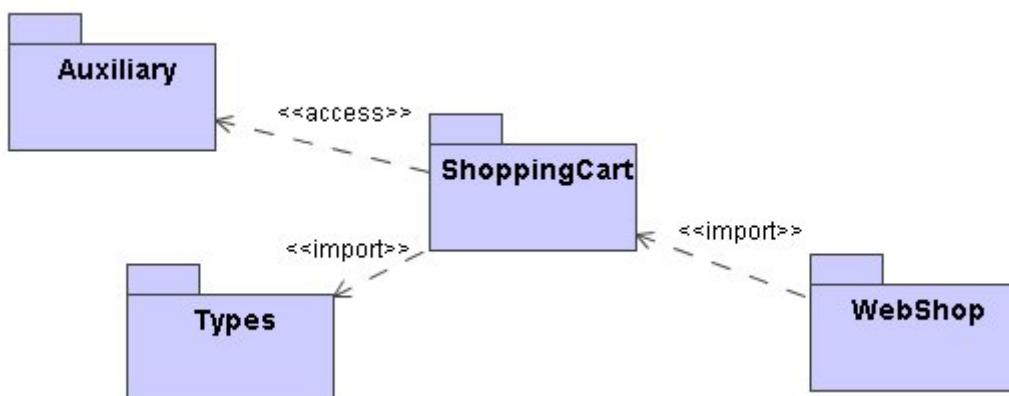
A dependency is shown as a dashed arrow between classes or packages. The model element at the tail of the arrow (the client element) depends on the model element at the arrowhead (the supplier element). The arrow can be labeled with an optional stereotype and an optional individual name.

NOTE You may also draw a dependency between a class and other class elements, such as attributes and operations.

For more information about working with the symbols, see "Diagramming"

Example of the dependency relationships

See the example of dependency relationships in the Figure below.



11 MODEL ELEMENTS

Dependency

The Dependency and Its Kinds Specification dialog boxes

Dependency, abstraction, and usage relationships defined in the dialog box of the same structure. They differ from one another only by the corresponding Specification name.

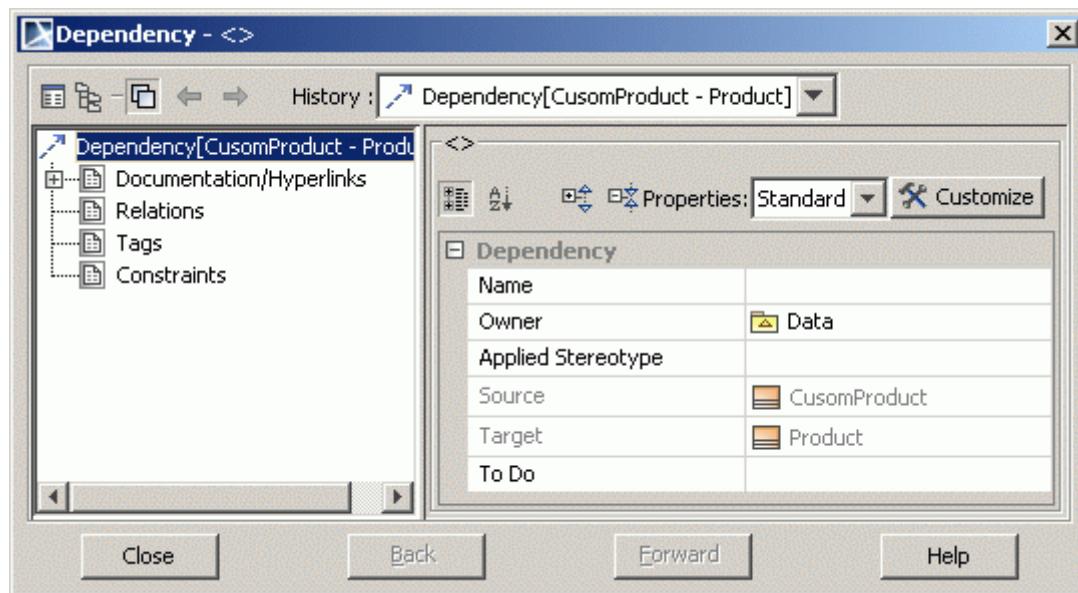


Figure 426 -- Dependency Specification dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box name | Function |
|----------|----------|--|
| General | Source | Shows the name of the dependency's supplier element. |
| | Target | Shows the name of the dependency's client element. |

Template Binding dependency

| | |
|-------------|---|
| Term | The OMG UML specification (UML 2.2: Superstructure) states: “A template binding represents a relationship between a templateable element and a template. A template binding specifies the substitutions of actual parameters for the formal parameters of the template.” |
|-------------|---|

11 MODEL ELEMENTS

Dependency

Define a binding dependency in the **Template Binding Specification** dialog box.

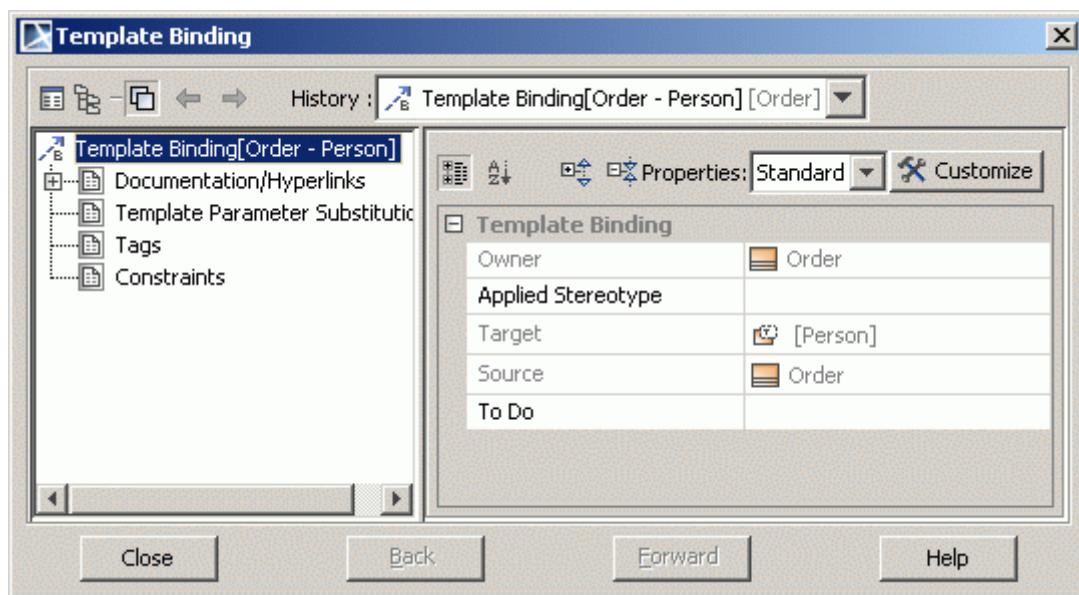


Figure 427 -- Binding Specification dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box name | Function |
|--|---------------|--|
| General | Source | Shows the name of the dependency's supplier element. |
| | Target | Shows the name of the dependency's client element. |
| Template Parameter Substitutions The parameter substitutions owned by this template binding. | Name | The name of the template parameter. |
| | Type | The type of the template parameter. |
| | Actual | A reference to a formal template parameter of the target template signature. |
| | Create | The Select Template Parameter dialog box opens. Select the item from the list and click OK . |
| | Delete | Removes the template parameter substitution. |

Abstraction

An abstraction is a relationship that relates two elements or sets of elements that represent the same concept at different levels of abstraction or from different viewpoints. In the metamodel, an abstraction is a dependency in which there is a mapping between the supplier and the client.

Define an abstraction relationship in the **Abstraction Specification** dialog box. For a detailed description of this dialog box, see “The Dependency and Its Kinds Specification dialog boxes” on page 681.

Usage

A usage is a relationship in which one element requires another element (or set of elements) for its full implementation or operation. In the metamodel, a usage is a dependency in which the client requires the presence of the supplier.

Define a usage relationship in the **Usage Specification** dialog box. For a detailed description of this dialog box, see “The Dependency and Its Kinds Specification dialog boxes” on page 681.

Package Merge

A package merge is a directed relationship between two packages, that indicates that the contents of the two packages are to be combined. It has a dependency relation with the applied stereotype <<merge>>.

Define a merge relationship in the **Dependency Specification** dialog box. For a detailed description of this dialog box, see “The Dependency and Its Kinds Specification dialog boxes” on page 681.

Package Import

A package import is defined as a directed relationship that identifies a package whose members are to be imported by a namespace. It is a relationship between an importing namespace and a package, indicating that the importing namespace adds the names of the members of the package to its own namespace. It is dependency relation with applied stereotype <<import>>.

Define an import relationship in the **Dependency Specification** dialog box. For a detailed description of this dialog box, see “The Dependency and Its Kinds Specification dialog boxes” on page 681.

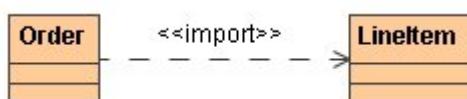
To draw the Package Import link, select the Package Import path to draw in the Class diagram toolbar, from the Abstraction group.

Element Import

An element import is defined as a directed relationship between an importing namespace and a packageable element. The name of the packageable element or its alias is to be added to the namespace of the importing namespace. It has a dependency relation with the applied stereotype <<import>>.

Define an import relationship in the **Dependency Specification** dialog box. For a detailed description of this dialog box, see “The Dependency and Its Kinds Specification dialog boxes” on page 681.

To draw the Element Import link, select the Element Import path to draw in the Class diagram toolbar, from the Abstraction group



Access

An access relationship shows that elements can only be accessed from a package, and it cannot be referenced.

To draw an Access link:

3. In the Class diagram toolbar, from the Abstraction group, select the Package Import path to draw.
4. Open the **Package Import Specification** dialog box and set the **Visibility** property to *private*.

Deployment

To draw a deployment link

1. In the implementation diagram toolbar, click the Deployment button and draw a deployment link from a node to an artifact.
2. From the node shortcut menu, Presentation Options submenu, clear the Suppress Deployment check box to display the deployed artifacts on the node instance shape.

Deployment Specification

The Deployment Specification is a type of Artifact.

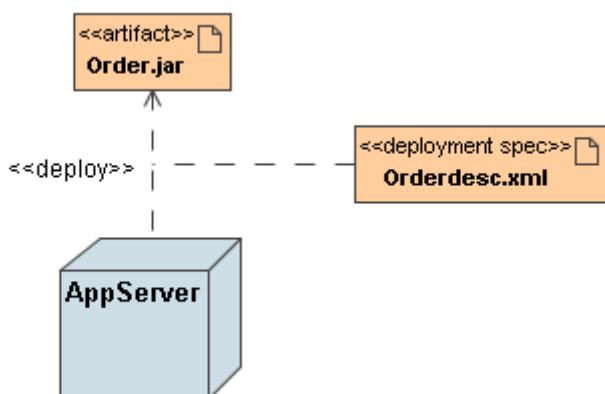
To draw the Deployment Specification on the diagram pane

In the Implementation diagram toolbar, right-click the Artifact button group. In the open list, select the Deployment Specification to draw.

The Deployment Specification is a general mechanism to parameterize a Deployment relationship.

To specify the Deployment relationship

1. Create the Deployment Specification.
2. Between the Deployment relationship and Deployment Specification draw the Dependency relationship. The Dependency is drawn without an arrow.



Events

An event is the specification of some occurrence that may potentially trigger effects by an object, i.e., an event shows what should happen to change a particular state in a system.

Event types

| Name | Description |
|--------------------------------|---|
| Any Receive Event | A trigger for an AnyReceiveEvent is triggered by the receipt of any message that is not explicitly handled by any related trigger. |
| Call Event | A call event represents the <i>reception</i> of a request to invoke a specific operation. A call event is distinct from the call action that caused it. A call event may cause the invocation of a behavior that is the method of the operation referenced by the call request, if that operation is owned or inherited by the classifier that specified the receiver object. |
| Change Event | A change event occurs when a Boolean-valued expression becomes true. For example, as a result of a change in the value held in a slot corresponding to an attribute, or a change in the value referenced by a link corresponding to an association. A change event is raised implicitly and is <i>not</i> the result of an explicit action. |
| Creation Event | A creation event models the creation of an object. |
| Destruction Event | A destruction event models the destruction of an object. |
| Execution Event | An ExecutionEvent models the start or finish of an execution specification. |
| Receive Operation Event | This specifies the event of receiving an operation invocation for a particular operation by the target entity. |
| Receive Signal Event | This specifies the event of receiving signal by the target entity. |
| Send Operation Event | A SendOperationEvent models the invocation of an operation call. |
| Send Signal Event | A SendSignalEvent models the sending of a signal. |
| Signal Event | A signal event represents the <i>receipt</i> of an asynchronous signal. A signal event may cause a response, such as a state machine transition as specified in the classifier behavior of the classifier that specified the receiver object, if the signal referenced by the send request is mentioned in a reception owned or inherited by the classifier that specified the receiver object. |
| Time Event | A time event specifies a point in time by an expression. The expression might be absolute or might be relative to some other point in time. |

Usability

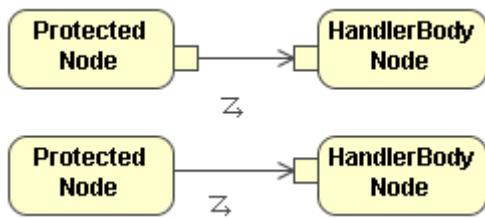
Events may trigger the change of a particular state. Events are important in diagrams which represent a behavior of a system. These diagrams are listed in the following table.

| Diagram name | Usage description |
|--|---|
| Activity Diagram | To define an Accept Event Action . |
| State Machine Diagram | To define a Transition , a transition to self. |
| Protocol State Machine Diagram | To define a protocol transition, a protocol transition to self. |

Exception Handler

An Exception Handler is an element that specifies a body to execute in case the specified exception occurs during the execution of the protected node.

The Exception Handler may be drawn from an Output Pin to an Input Pin or from an Action to an Input Pin:



Extend

A relationship from an extending use case to an extended use case that specifies how and when the behavior defined in the extending use case can be inserted into the behavior defined in the extended use case. The extension takes place at one or more specific extension points defined in the extended use case.

Define the extend relationships in the **Extend Specification** dialog box.

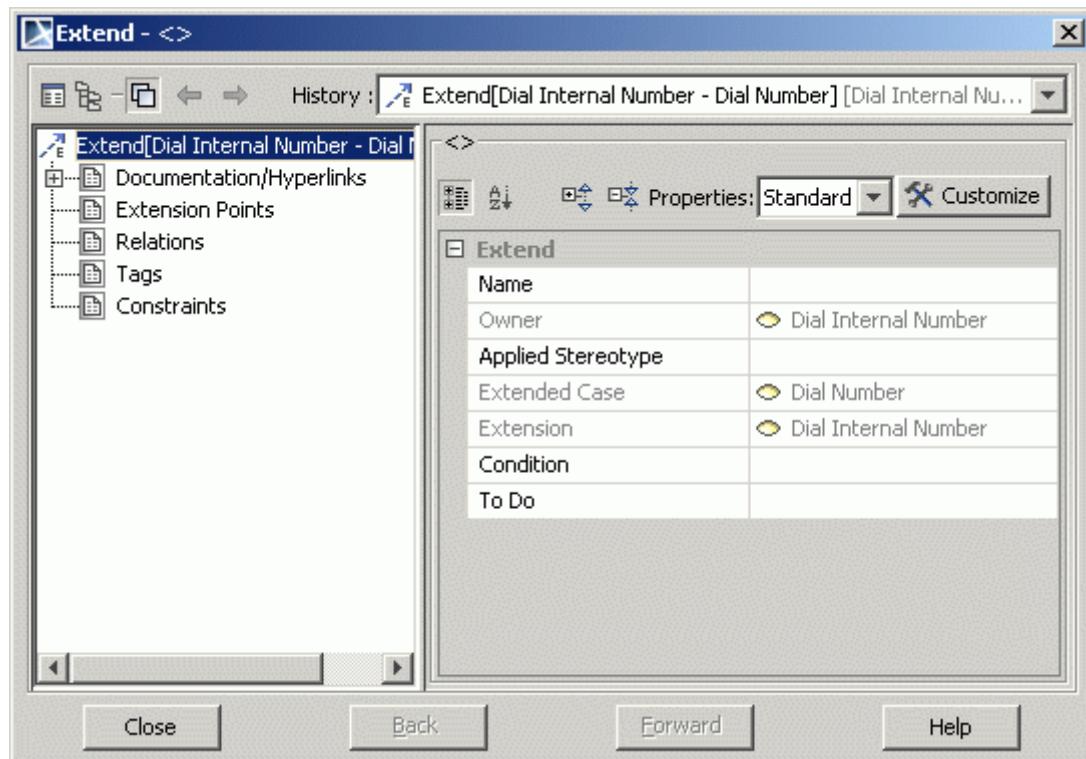


Figure 428 -- the Extend Specification dialog box

11 MODEL ELEMENTS

Extend

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Group name | Box | Function |
|--|------------------|---|
| General Set general information about the extend relationship | Condition | An expression specifying the condition, which must be fulfilled if the extension is to take place. |
| Extension Points Select the extension points to be assigned to the extend relationship | Assign | Select predefined extension points you wish to assign to the extend relationship in the Extension Points dialog box. |
| | Unassign | Remove an already created extension point from the use case. |
| | Up | Move the selected extension point to an upper position. |
| | Down | Move the selected extension point to a lower position. |

To open the **Extension Point Specification** dialog box

1. Select **Specification** from the use case shortcut menu, or double-click the use case shape. The **Use Case Specification** dialog box opens.
2. Expand the **Extension Points** group and then click on the created extension point. (Double-clicking opens an independent **Extension Points Specification** window).

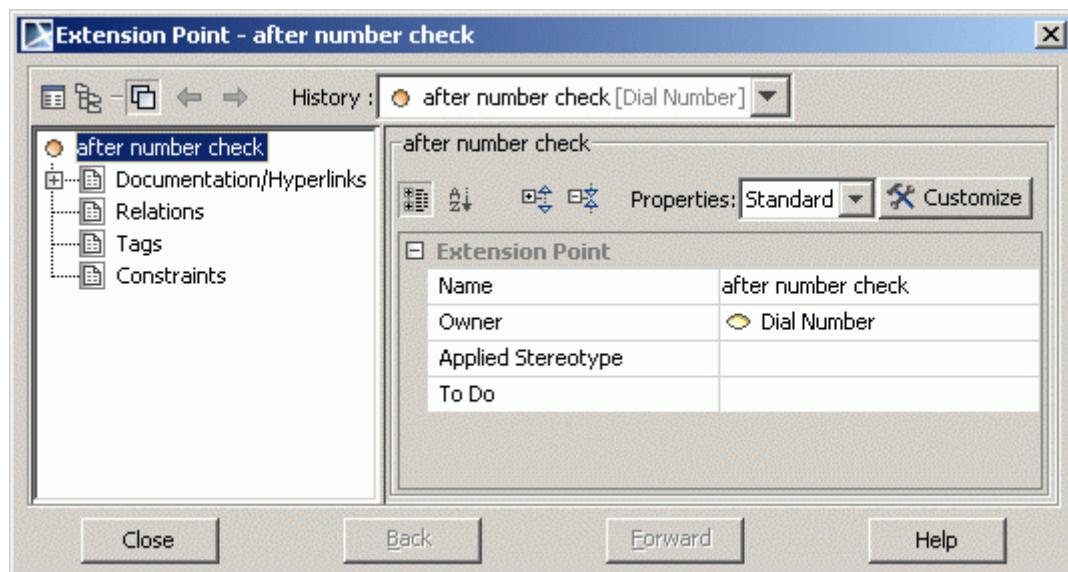


Figure 429 -- Extension Point Specification dialog box

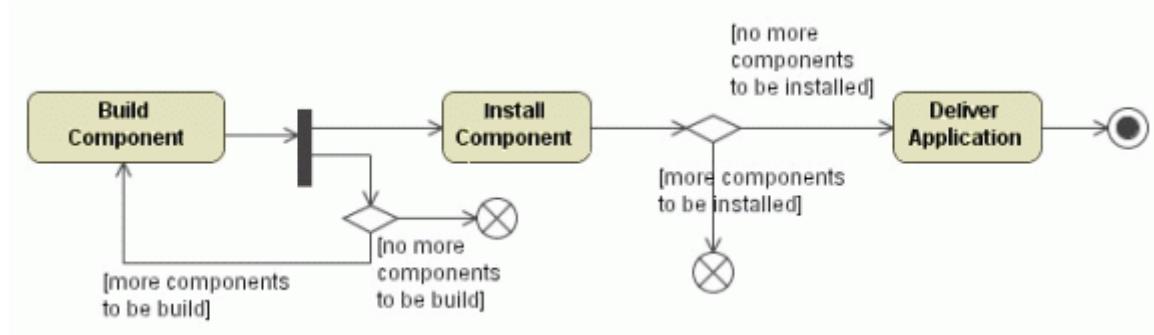
Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box name | Function |
|---|--------------|---|
| General A general information about the extension point is displayed. | Owner | The name of the use case, which owns the extension point. |

Flow Final Node

It is a final node that terminates a flow and destroys all tokens that arrive at it. It has no effect on other flows in the activity.

See the following example for the flow final element notation in a MagicDraw project.



For more information about defining the flow final node, see “Containment” on page 676.

Fragmet

MagicDraw represents twelve kinds of fragments: Alternatives, Loop, Option, Parallel, Break, Negative, Critical Region, Consider, Ignore, Weak Sequencing, Strict Sequencing, and Assertion.

Alternative Fragment

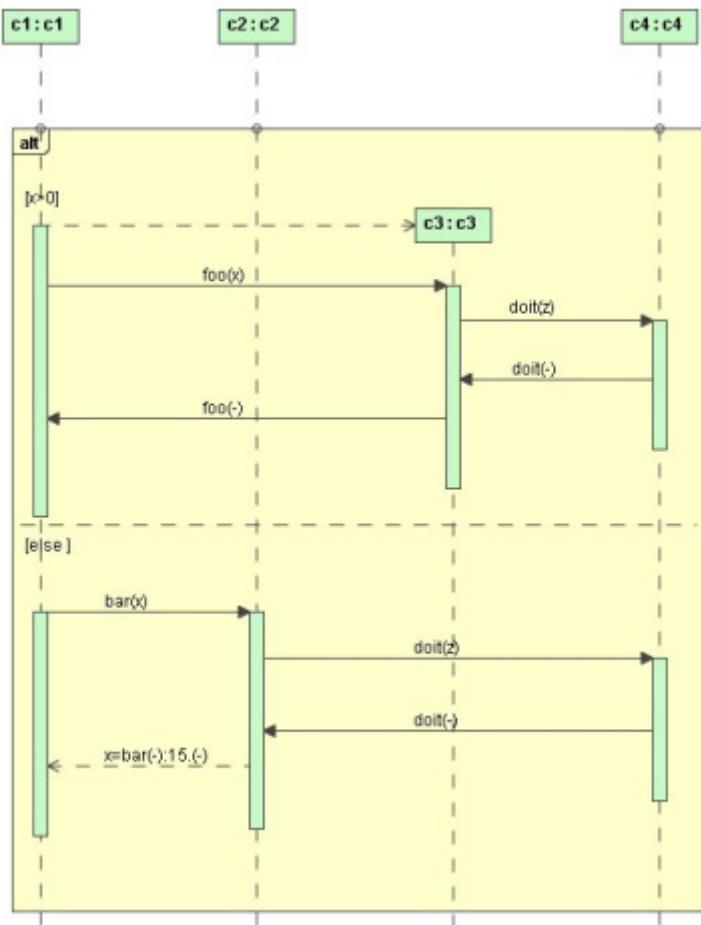
The alternative fragment models *if...then...else* constructions.

11 MODEL ELEMENTS

Fragment

To draw an Alternative Fragment

- In the Sequence diagram toolbar, select the Alternatives element to draw.



Combined Fragment

The UML Combined Fragment element allows the expressions of interaction fragments to be defined in the sequence diagram. The combined fragments provide a means to define special conditions and subprocesses for any sections of lifelines in the sequence diagram by specifying an area where the conditions or subprocesses apply. Using the Combined Fragment, a fragment of the sequence diagram can be separated.

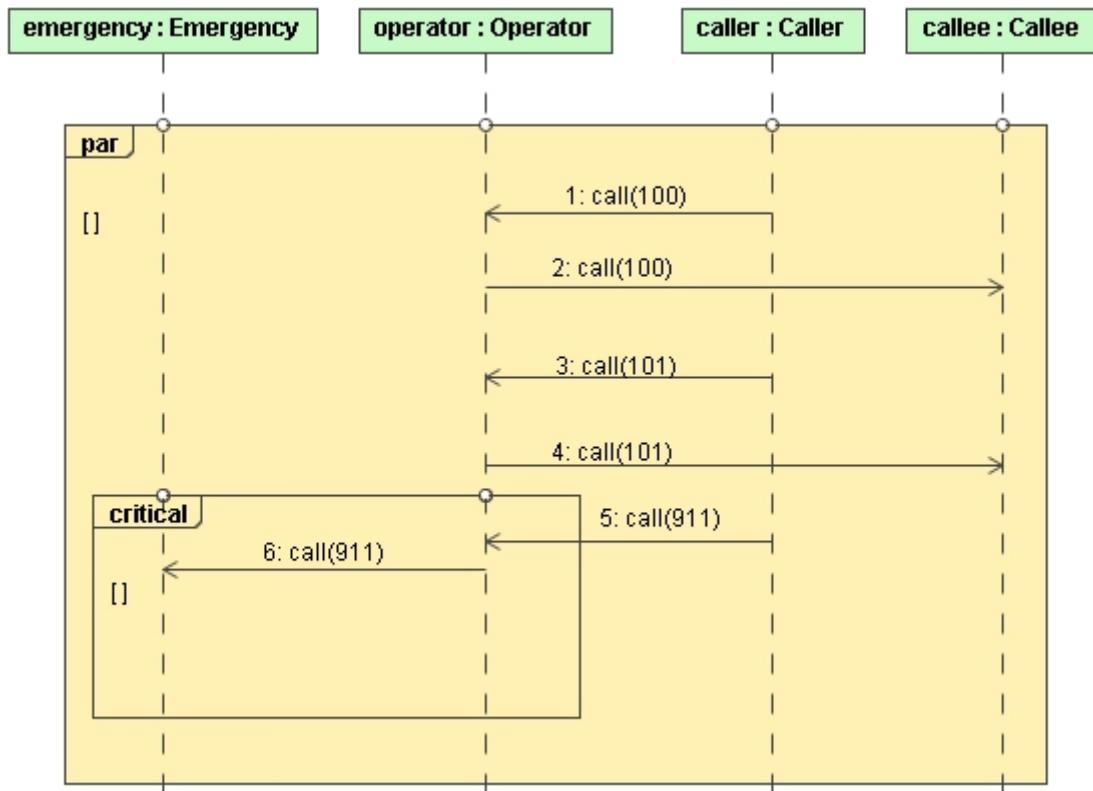
To draw a Combined Fragment

- In the Sequence diagram toolbar, select the Alternatives element to draw.
- From the combined fragment shortcut menu select the **Covered Lifelines** command and in the **Covered Lifelines** dialog box, select the lifelines to display.
- In the Sequence diagram toolbar, **Options** group, select the Parallel combined fragment to draw.

11 MODEL ELEMENTS

Fragment

- Draw the Critical Region combined fragment from the Sequence diagram toolbar, **Option** button group.



Function Behavior

Term The OMG UML specification (UML 2.3: Superstructure) states:

Description

A function behavior is an opaque behavior that does not access or modify any objects or other external data.

Semantics

Primitive functions transform a set of input values to a set of output values by invoking a function. They represent functions from a set of input values to a set of output values. The execution of a primitive function depends only on the input values and has no other effect than to compute output values. A primitive function does not read or write structural feature or link values, nor otherwise interact with object memory or other objects. Its behavior is completely self-contained. Specific primitive functions are not defined in the UML, but would be defined in domain-specific extensions. Typical primitive functions would include arithmetic, Boolean, and string functions.

During the execution of the function, no communication or interaction with the rest of the system is possible. The amount of time to compute the results is undefined. FunctionBehavior may raise exceptions for certain input values, in which case the computation is abandoned.

The function behavior allows modeling external functions that take only inputs and produce outputs. It has no effect on the specified system.

To create the function behavior

In the Browser tree, right-click the Data package. In the shortcut menu, select the **New Element** command and then select Function Behavior  .

Parent topic: “Model Elements” on page 620.

Gate

Term The OMG UML specification (UML 2.2: Superstructure) states:

“A **Gate** is a connection point for relating a Message outside an Interaction-Fragment with a Message inside the InteractionFragment.”
“Gates are connected through Messages. A Gate is actually a representative of an OccurrenceSpecification that is not in the same scope as the Gate. Gates play different roles: we have formal gates on Interactions, actual gates on InteractionUses, expression gates on CombinedFragments.”

MagicDraw version 15.0 and later allows the display of messages leaving or entering a sequence diagram, interaction use, or combined fragment. The Gate is a connection point for representing a message from the outside to the current sequence diagram, interaction use, or combined fragment.

Gates can be used in three cases:

- For passing and returning arguments to InteractionUse, which calls some Interaction.

11 MODEL ELEMENTS

Gate

- For displaying “exceptions” as messages that stops an interaction execution and leaves it.
- For “calling” sequence blocks represented as CombinedFragments.

Gate has no notation. Gates are created as message ends when drawing messages to/from a diagram frame, an interaction use, or a combined fragment.

Parent topic: “Model Elements” on page 620.

Usage in diagrams: “Sequence Diagram” on page 514.

Related topics:

“Model” on page 721.

“Lifeline in the Sequence Diagram” on page 712.

“Combined Fragment” on page 670.

“Interaction Use” on page 706.

To create a formal gate

- Draw a call, send, create, or delete a message from the diagram frame.
- Draw a reply message to the diagram frame.
- Draw a call, send, create, or delete a message from the combined fragment (inside combined fragment).

NOTES

- You can view the created gates of message in the **Message** specification dialog box. In the **Send Event** list box you may see formal gate and the **Receive Event** lists the actual gate.
- The Gate uses text from the message as an identification name. For example, a message name or a message operation.

To draw create an actual gate of the formal gate

- Draw a call, send, create, or delete a message to the interaction use, which refers to the diagram with the formal gates. The **Select Formal Gate** dialog box opens.
- Draw a reply message from the interaction use, which refers to the diagram with the formal gates. The **Select Formal Gate** dialog box opens.
- Draw a call, send, create, or delete a message to the combined fragment (outside combined fragment). The **Select Formal Gate** dialog box appears.

NOTE

You can also view the formal and actual gates in the gates **Interaction** specification dialog box, **Interaction Use** dialog box, **Combined Fragment** dialog box, and the **Actual Gates** and **Formal Gates** panes.

To select a formal gate for the actual gate

1. Draw a message to invoke the **Select Formal Gate** dialog box (see “To draw create an actual gate of the formal gate” on page 692).
2. Select one of the listed formal gates and click OK. An actual gate is created.

11 MODEL ELEMENTS

Gate

- Or from the message shortcut menu, select the **Select Formal Gate** command. The **Select Formal Gate** dialog box opens.

NOTE The **Select Formal Gates** command exists only if there are formal gates.

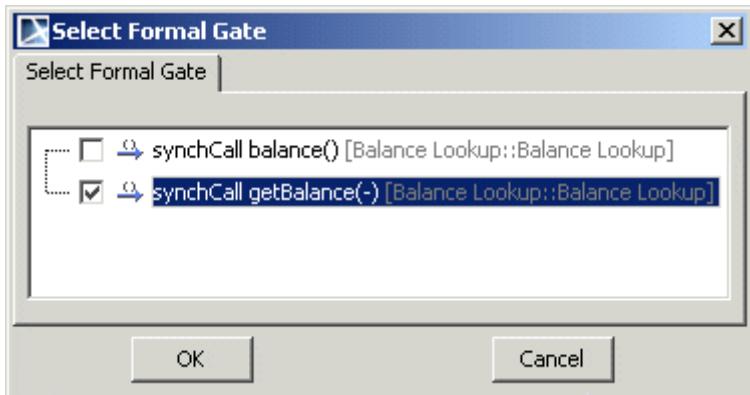


Figure 430 -- The Select Formal Gate dialog box

The formal gate and actual gate usage in the sequence diagram

See Figure 431 on page 693 where the *getBalance* message is drawn from the diagram frame to the *theirBank* lifeline. The *getBalance* message has a gate.

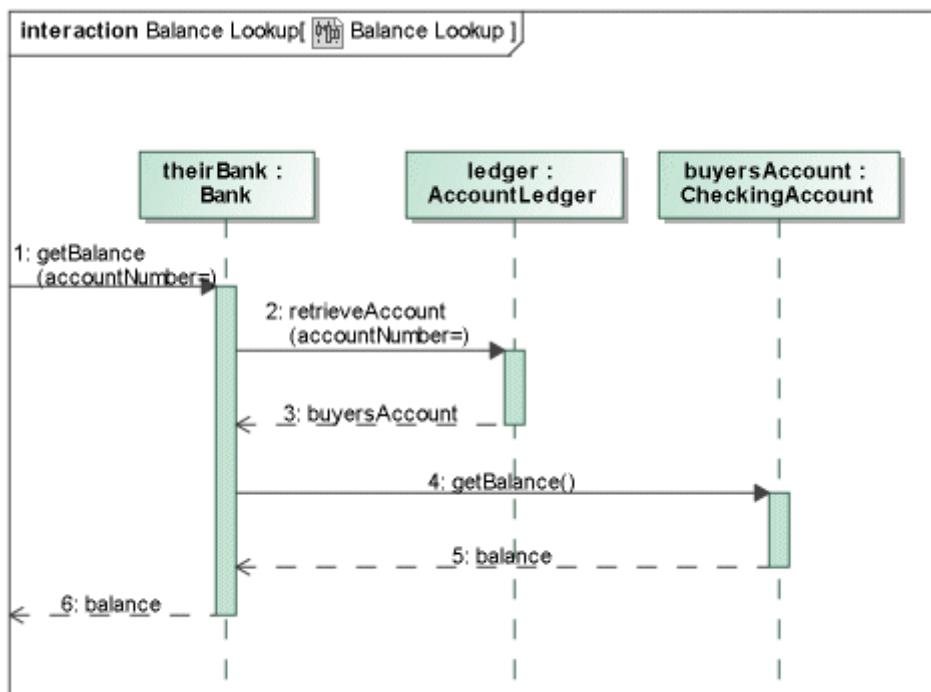


Figure 431 -- Formal gates usage in Sequence diagram

See Figure 432 on page 694 where the actual gate is presented. The *Balance Lookup* interaction use refers to the *Balance Lookup* sequence diagram. The *getBalance* message (see the 2nd message) has selected the formal gate and automatically repeats the data of the *getBalance* message from the *Balance Lookup* diagram.

11 MODEL ELEMENTS

Generalization

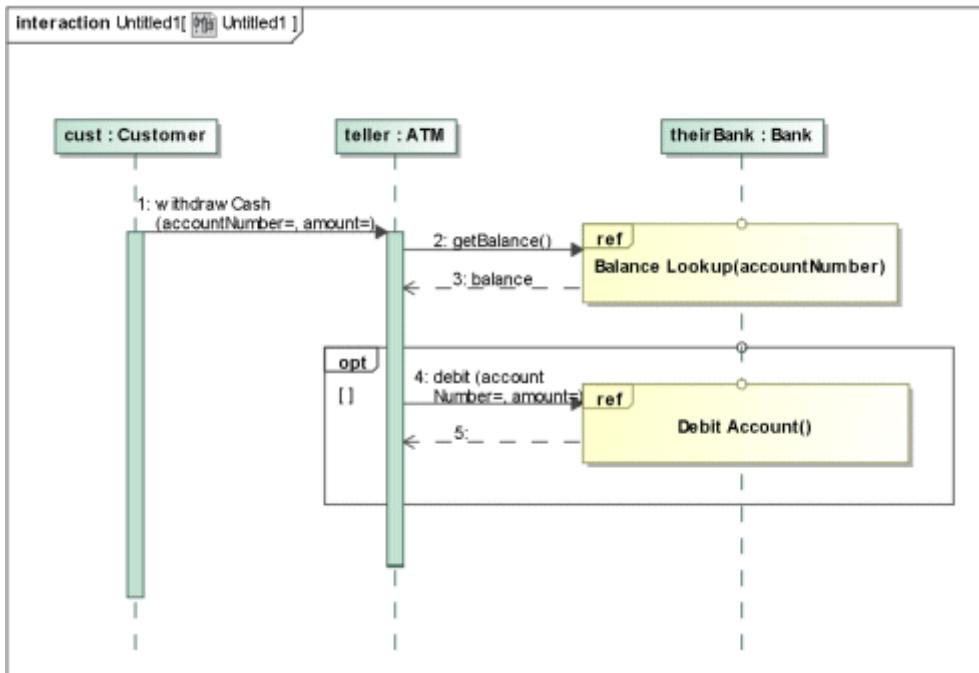


Figure 432 -- Actual gates usage in sequence diagram

Generalization

Term The OMG UML specification (UML 2.2: Superstructure) states:

"A generalization is a taxonomic relationship between a more general classifier and a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier. Thus, the specific classifier inherits the features of the more general classifier."

"Where a generalization relates a specific classifier to a general classifier, each instance of the specific classifier is also an instance of the general classifier. Therefore, features specified for instances of the general classifier are implicitly specified for instances of the specific classifier. Any constraint applying to instances of the general classifier also applies to instances of the specific classifier."

For more information about working with symbols, see "Diagramming"

Define the generalization in the **Generalization Specification** dialog box.

Refer to the "Specification Window" on page 237 for information about the specification elements not covered in this section.

| Tab name | Box | Function |
|--|-----------------|---------------------------------|
| General Set a general information about the generalization relationship. | General | The name of the parent element. |
| | Specific | The name of the child element. |

11 MODEL ELEMENTS

Generalization

To group generalization paths into a tree

1. Draw a generalization path between the child element and the parent element.
2. Click the generalization path button on the diagram toolbar.
3. Click the other child shape.
4. Drag the path to the other generalization path and drop it there.
 - Select **Make Sub Tree** from the parent class shortcut menu.
 - Select **Make Sub Tree** from the parent package shortcut menu.

To ungroup a generalization tree

1. Click the generalization tree's hollow triangle pointing to a parent element.
2. From the tree shortcut menu, select **Ungroup Tree**.

To separate the generalization path from the generalization tree

- From the generalization path shortcut menu, select **Remove From Tree**.
- Drag one generalization path to another class.

NOTE This command is available if the same tree contains the selected generalization path.

Generalizable elements

A generalizable element is a model element that may participate in a generalization relationship.

| Name | Function |
|--------------------|---|
| Is Abstract | Specifies whether the generalizable element may or may not have a direct instance. True indicates that an instance of the generalizable element must be an instance of a child of the generalizable element. False indicates that there may be an instance of the generalizable element that is not an instance of a child. An abstract generalizable element cannot be instantiated since it does not contain all the necessary information. |
| Is Leaf | Specifies whether the generalizable element is with descendants. True indicates that it may not have any descendant. False indicates that it may have some descendants (whether or not it actually has any descendants at the moment.) |

To define a generalizable model element (class, package, use case, etc.) as abstract or leaf

1. Open the corresponding **Specification** dialog box.
2. Select the **Is Abstract** or **Is Leaf** check box(es) in the **General** tab.

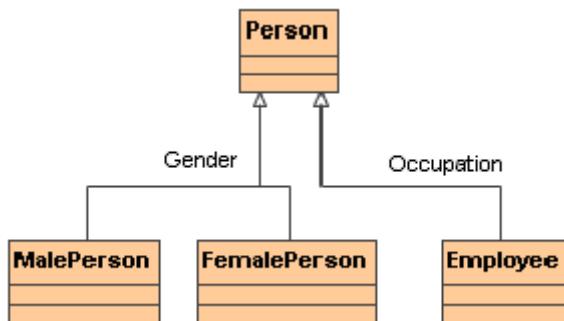
Generalization sets

Each generalization is a binary relationship that relates classifier to a more specific classifier. The Generalization Set defines a particular set of generalization relationships that describes the way in which a general classifier may be divided using specific types.

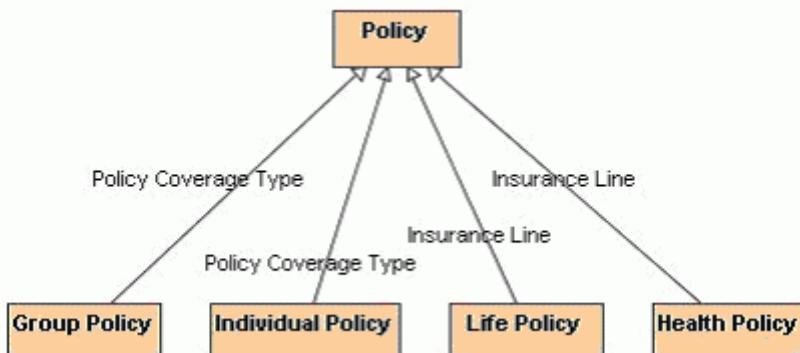
11 MODEL ELEMENTS

Generalization

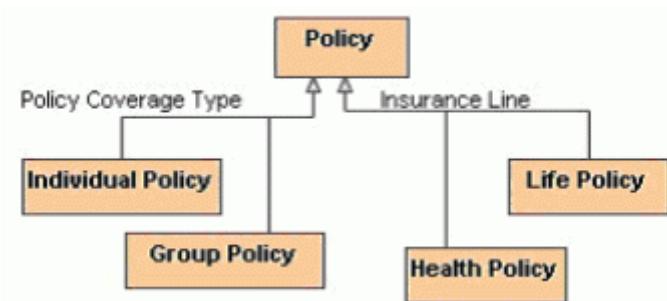
For example: the Person class can be specialized as either a Male Person or a Female Person. Furthermore, the Person may be specialized as an employee. The Female Person and Male Person constitute one Generalization Set and the Employee another:



The label next to the generalization designates the name of the Generalization Set to which the generalization belongs.



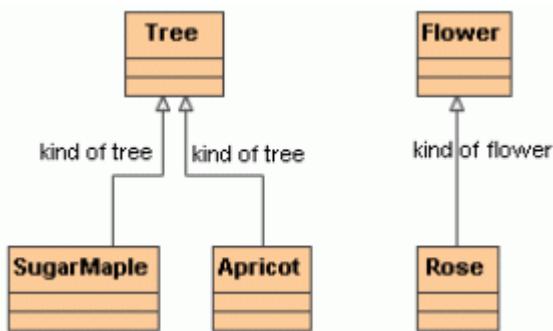
The generalization set may be drawn as two or more generalization lines drawn to the same arrowhead. In this case it is labeled once. Drawing such a tree helps to understand the grouping of generalizations sets quickly .



11 MODEL ELEMENTS

Generalization

The generalization set may contain only generalizations that have the same General element. The following is the example of generalizations, which may not have the same generalization set:



From the above example, the generalization, which is between the *Flower* and *Rose* classes, may not belong to the kind of tree generalization set, because its general element is *Flower* and the general element for the kind of tree generalizations is *Tree*. MagicDraw only lists all generalization sets that are allowed to assign. If a generalization set created in project does not have any generalizations, it may contain any generalization sets.

To create a new or assign an existing generalization set

- In the **Generalization Specification** dialog box, near the **Generalization Set** box click "... button. In the open **Select Elements** dialog box assign an existing generalization set or click the **Create** button to create new generalization set.

NOTE!

- In the Generalization Set list many generalization sets can be listed, but in this case on the diagram pane, near the generalization, only the first in the list is displayed.
- On the diagram pane, select a generalization. From the generalization shortcut menu, select the **Generalization Set** command. In the opened list, select the generalization set or click the **New** button and in the indicated place create a new Generalization Set to which the current generalization is assigned.
 - On the diagram pane select some generalizations, which you want to assign to the generalization set. From the shortcut menu, select the **Generalization Set** command.

NOTE!

- The Generalization Set command is enabled only if all selected generalizations have the same general element.
- Draw or move a generalization line to a generalization set tree or to a generalization that belongs to a generalization set. The newly created generalization is assigned to the same generalization set.

11 MODEL ELEMENTS

Generalization

The Generalization Set dialog box

Specify the Generalization set in the **Generalization Set** dialog box.

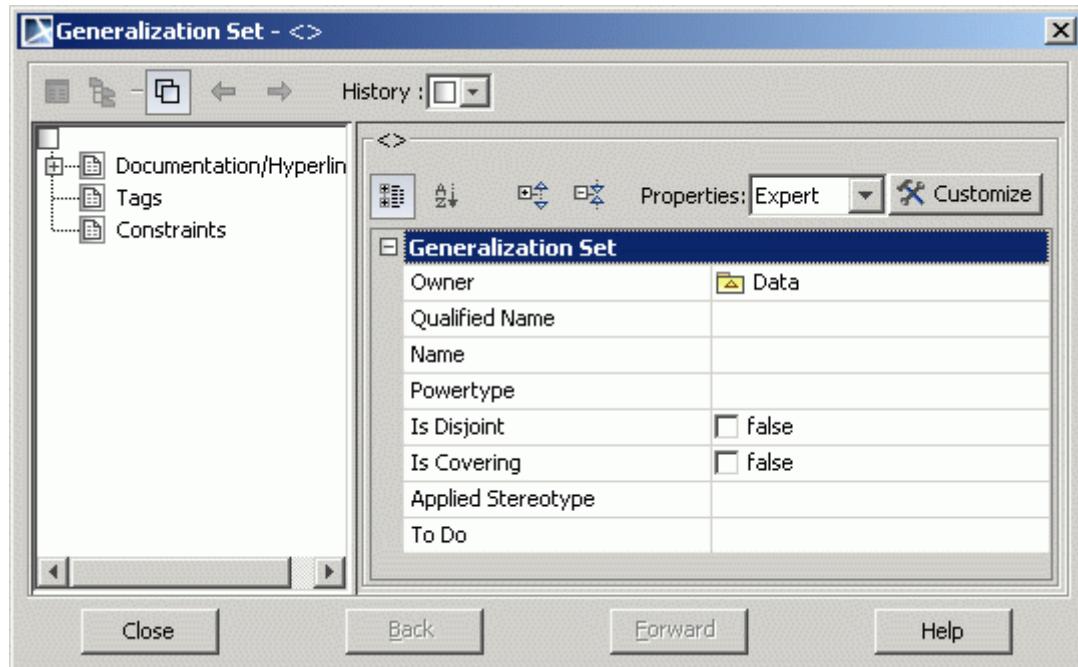


Figure 433 -- the Generalization Set dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box | Function |
|---|--------------------|--|
| General Set a general information about the generalization relationship. | Is Disjoint | Indicates whether or not a set of specific Classifiers in a Generalization relationship has a common instance. If the value is <i>true</i> , the Classifiers for a particular GeneralizationSet have no members in common; that is, their intersection is empty. The default value of the Is Disjoint property is <i>false</i> . |
| | Is Covering | Indicates whether or not a set of specific Classifiers is covering for a particular general classifier. When the value is <i>true</i> , every instance of a particular general Classifier is also an instance of at least one of its specific Classifiers for the Generalization Set. |

In the **Generalization Set** dialog box, the **Generalization** list box lists the generalizations which are assigned to the current generalization set.

To assign a generalization:

1. Near the Generalization list click the “...” button. The list of available generalizations opens.
2. Near the generalization you want to assign to the generalization set, select the check box.
3. Click the **Apply** button.

NOTE!

The generalizations that belong to other sets are allowed to be selected. Selecting such a generalization removes it from the previous set and adds it to the current one.

11 MODEL ELEMENTS

Include

To open the Generalization Set symbol properties

On the diagram pane select the generalization set name and from its shortcut menu, select the **Symbol(s) Properties** command.

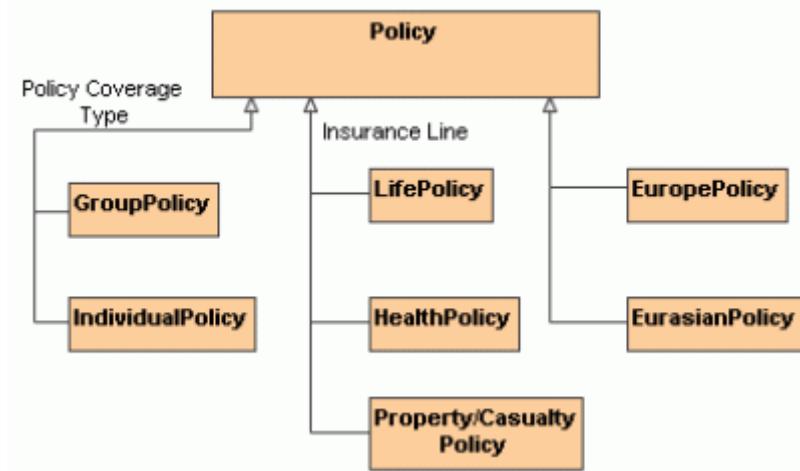
If the **Show Powertype** check box is selected, on the diagram pane, next to the generalization, the label of Powertype of Generalization Set is displayed instead of the Generalization Set name. The default **Show Powertype** property value is *false*.

If the **Show Complete/Disjoint** check box is selected, on the diagram pane, next to the generalization, the values of **Is Covering** and **Is Disjoint** properties are displayed on the diagram. The default **Show Complete/Disjoint** property value is *false*.

To group the generalizations to the generalization set trees

1. Select a general classifier on the diagram pane.
2. From the classifier shortcut menu, select the **Make Generalization Set Tree** command. The generalizations are grouped to trees, according to the generalization sets.

When the **Make Generalization Set Tree** command is selected, a tree is created for each generalization set. The generalizations, which do not belong to any generalization set, are grouped to a separate tree. For example:



Include

An include (uses) relationship from the use case A to the use case B indicates that an instance of the use case A will also contain the behavior as specified by B.

The include relationship is used when there are common parts of the behavior among two or more use cases. Each common part is then extracted to a separate use case, to be included by all base use cases having this part in common. Since the primary use of the include relationship is to reuse the common parts, what is left in the base use case is usually not complete in itself but dependent on the included parts for meaning and context. This is reflected in the direction of the relationship, indicating that the base use case depends on the addition but not vice versa.

11 MODEL ELEMENTS

Information Flow

Define the extend relationships in the **Include Specification** dialog box.

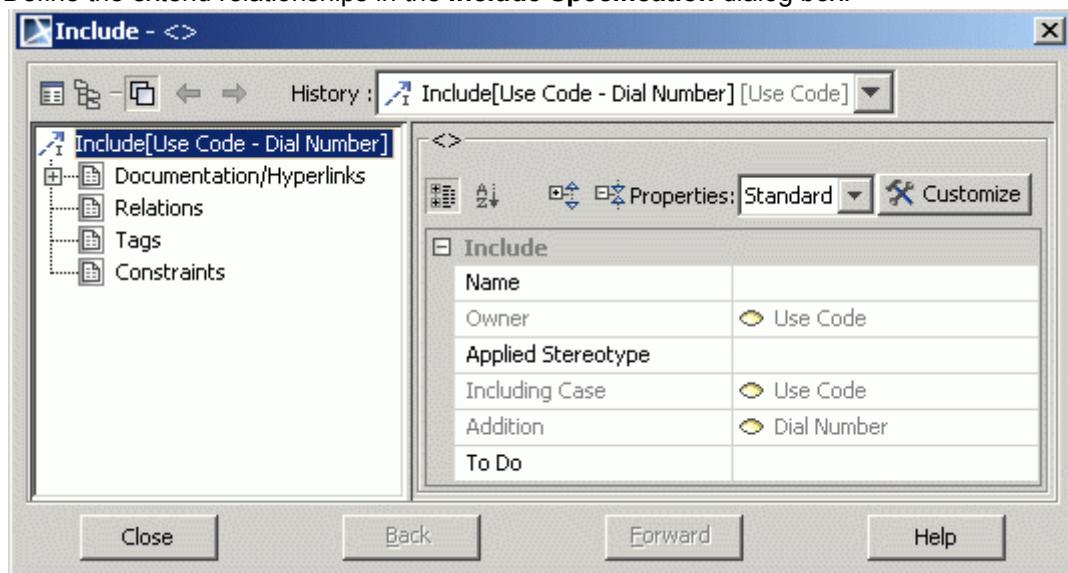


Figure 434 -- *Include Specification* dialog box

Refer to the “Specification Window” on page 237 for information about specification elements not covered in this section.

| Tab name | Box | Function |
|---|-----------------------|--|
| General Set a general information about the include relationship. | Owner | The name of the use case, which owns the include relationship. |
| | Including Case | Name of the use case to where the relationship goes. |
| | Addition | Name of the use case from where the relationship comes. |

Information Flow

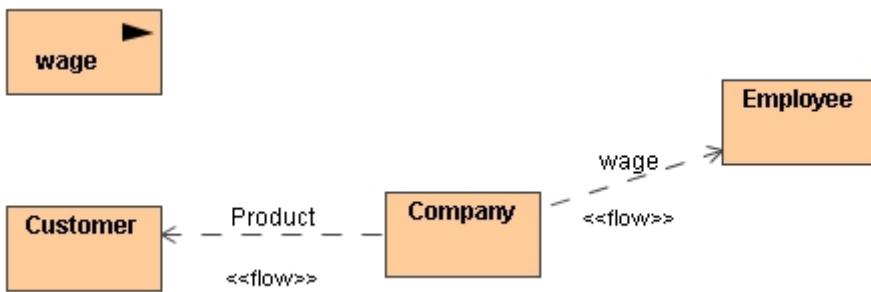
An Information Flow specifies that one or more information items circulates from its sources to its targets. Information flows require some kind of “information channel” for transmitting information items from the source to the destination.

An information channel is represented in various ways depending on the nature of its sources and targets. It may be represented by connectors, links, associations, or even dependencies. For example, if the source and destination are parts in some composite structure diagrams such as a collaboration, then the information channel is likely to be represented by a connector between them. Or, if the source and target are objects (which are a kind of InstanceSpecification), they may be represented by a link that joins the two, and so on.

11 MODEL ELEMENTS

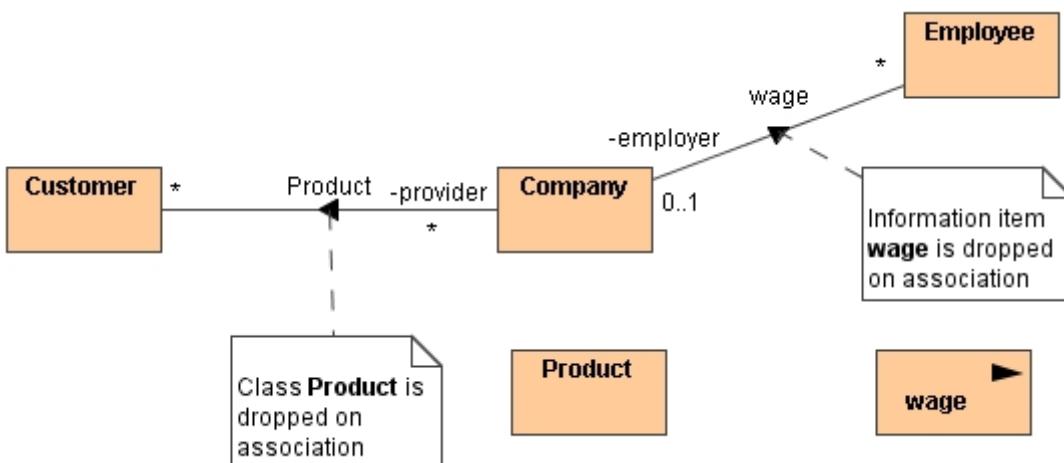
Information Item

The information flow and the information item notation are added. You may draw them using the Information Flows toolbar in the class or composite structure diagram:



You can also create information flows in the associations in the class diagram and on the connectors in the composite structure diagrams:

3. Draw classes and associations.
4. From the association shortcut menu select command **Symbol(s) Properties** and select the **Show conveyed information A** and **Show conveyed information B** checkboxes.
5. Drag the class or information item on an association. An information flow is created.



Information Item

The InformationFlows package provides mechanisms for specifying the exchange of information between entities of a system at a high level of abstraction.

The Information flows describe a circulation of information in a system in a general manner. They do not specify the nature of the information nor the mechanisms by which this information is conveyed (message passing, signal, common data store, parameter of operation, etc.). They also do not specify sequences or any control conditions. It is intended that, while modeling in detail, representation and realization links will be able to specify which model element implements the specified information flow, and how the information will be conveyed.

An information item is an abstraction of all kinds of information that can be exchanged between objects. It is a kind of classifier intended for representing information in a very abstract way, the one which cannot be instantiated.

One purpose of information items is to be able to define preliminary models, before making a detailed modeling decisions on types or structures. Another purpose of information items and information flows is to abstract com-

11 MODEL ELEMENTS

Instance

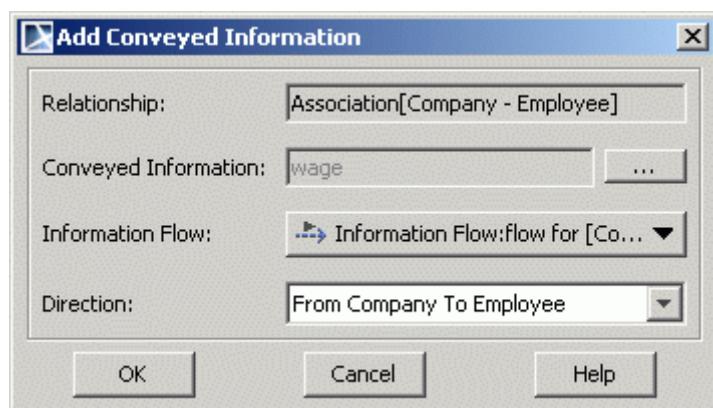
plex models by using a less specific but more general representation of the information exchanged between entities of a system.

In a classifier, the information item can be represented as a name inside a rectangle. The black triangle icon on top of this rectangle indicates that it is an information item.



the information items (or any conveyed classifiers) can be displayed on any relationship:

1. Select the information item on the diagram pane and drag it on the relationship shape. The **Add Conveyed Information** dialog box opens.



2. After specifying information flow and direction arrow, click **OK**.

Instance

An instance specification specifies the existence of an entity in a modeled system and completely or partially describes the entity.

The description may include:

- The classification of an entity by one or more classifiers of which the entity is an instance. If the only classifier specified is abstract, then the instance specification only partially describes the entity.
- A kind of instance based on its classifier or classifiers - for example, an instance specification whose classifier is a class describes an object of that class, while an instance specification whose classifier is an association describes a link of that association.
- A specification of values of structural features of the entity. Not all structural features of all classifiers of the instance specification need to be represented by slots, in which case the instance specification is a partial description.
- A specification of how to compute, derive, or construct the instance (optional).

MagicDraw allows you to create the instances of classifiers – class, interface, enumeration, use case, actor, node, component, artifact, and other classifiers.

The instances are shown using a rectangle by underlining the name string of an instance element. The instance of an actor is shown as an actor “stick man” figure with the actor’s name string below the symbol.

11 MODEL ELEMENTS

Instance

For more information about working with symbols, see "Diagramming"

Define the selected instance in the **Instance Specification** dialog box.

An instance is an individual unit with its own identity and value. Each instance has a descriptor – model element.

To open the **Instance Specification** dialog box

Select **Specification** from the instance shortcut menu or double-click the instance shape.

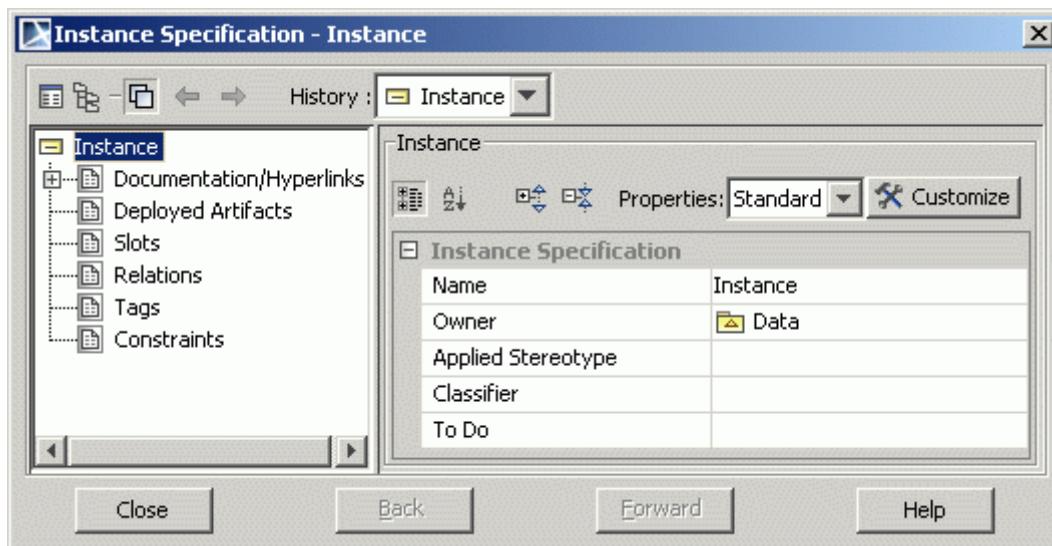


Figure 435 -- *Instance Specification dialog box*

Refer to the "Specification Window" on page 237 for information about the specification elements not covered in this section.

| Tab name | Box name | Function |
|---|-------------------|---|
| General Set general options for the instance. | Owner | The name of the element, containing an instance. |
| | Classifier | The name of the assigned classifier. |
| Deployed Artifacts List of an artifacts or artifact instances that has been deployed to a deployment target. | Name | The name of the artifact or instance. |
| | Type | An element, which appears as a type of deployed artifact. |
| | Owner | The name of the element, containing deployed artifact. |
| Slots A named slot in an instance, which holds the value (the instance that is the value of the AttributeLink) of an attribute (the attribute from which the AttributeLink originates). | Type | The name, which is the owner of the attribute. |
| | Attribute | The name of the attribute. |
| | Instance | The name of an instance, to which the current instance is set as a default value. |

To assign an already existing classifier to an instance

1. Right-click the instance shape and select **Type** from the instance shortcut menu.
2. Select the classifier you wish to assign to an instance.

11 MODEL ELEMENTS

Instance

- Open the **Instance Specification** dialog box and in the **Classifiers** field, click the ‘...’ button. The **Select Elements** dialog box opens. Move the classifier from **All** to the **Selected** list and click **OK**.

To assign/unassign an existing classifier for an instance in the Instance Specification dialog box

1. Double-click an instance shape or select **Specification** from the shape shortcut menu.
2. The **Instance Specification** dialog box appears. Click the ‘...’ button in the **Classifiers** property.
 - To assign a new classifier, select an element in **All** and click the **Add** button to move it to the **Selected** list.
 - To unassign the assigned classifier, select an element in the **Selected** list and click the **Remove** button.

To hide/show an assigned classifier

From the instance shortcut menu, select/clear the **Show Classifier** check box.

To set the initial value to an attribute of the assigned classifier

1. Double-click an instance shape or select **Specification** from the shape shortcut menu.
2. The **Instance Specification** dialog box opens. Click the **Slots** tab.
3. Click the **Edit Value** button and type the name of the value.

To show/hide slots of the assigned classifier

Clear/select **Suppress Slots** check box in the instance shortcut menu.

NOTE By default slots of the classifier are suppressed.

To display slot type on the instance symbol

Slot type can be optionally displayed on Instance or Part shapes.

Property **Show Slot Type** is added to slot symbol properties (select command **Symbol(s) Properties** from instance shortcut menu to invoke **Properties** dialog). Slot type name (see Figure 436 on page 704), slot type qualified name (see Figure 437 on page 704) or no slot type (see Figure 438 on page 705) may be displayed next to slot.

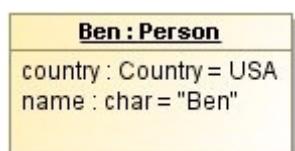


Figure 436 -- Slot type name is displayed next to slot

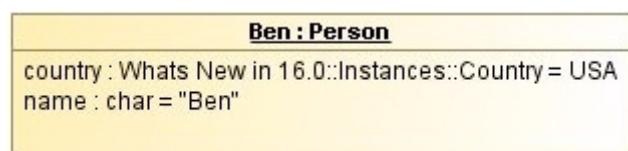


Figure 437 -- Slot type qualified name is displayed next to slot

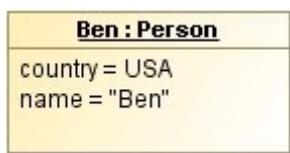


Figure 438 -- No slot type is displayed

To select slot in a diagram

Slot can be selected in a diagram. This allows the deletion of a slot straight from the diagram and to attach a note to a slot.

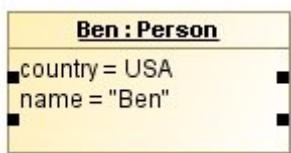


Figure 439 -- Slot selected in a diagram

Instance Specification

An instance specification represents an instance in a modeled system.

In the implementation diagram Node Instance, Component Instance, Artifact Instance elements are the same Instance Specification elements with an assigned component, node or artifact.

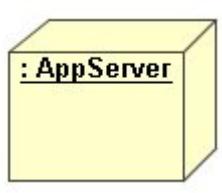
To create a Component Instance

1. In the Implementation diagram toolbar, click the Component Instance button. The **Select Components** dialog box opens.
2. Select a component from the list or click the **Create** button to create a new one. Click **OK**.

The same is valid for the Node Instance and Artifact Instance creation.

TIP!

Use the Node Instance button to create the Instance Specification with assigned Node and the Node Instance will have a Node shape.



To display specification value on the Instance Specification symbol

Specification value can be optionally displayed on the Instance Specification symbol. Check box **Show Specification Value** is added to Instance Specification properties (select command Symbol(s) Properties from instance shortcut menu to invoke **Properties** dialog).

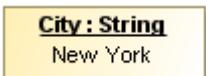


Figure 440 -- Specification value is displayed on the Instance Specification symbol

Assigning Instance Specification as Default Value quickly

You can drag Instance Specification element on Classifier property on a diagram to assign it as default value. Drag and drop is available only if Instance Specification classifiers are compatible with Property type and if Property is editable.

Interaction Use

Interactions are units of behavior of an enclosing Classifier. They focus on the passing of information with Messages between the Connectable Elements of the Classifier.

A reference to the interaction can be created.

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

To add a reference to an element

- In the **Interaction Use** specification dialog box, click the **Refers To** drop down list. A list of interactions, created in the project, opens. Click ‘...’ button, to add the interaction from the **Select Element** tree. Click the **Create** button to create a new interaction.
- From the interaction use shortcut menu select the **Refers To** command. In the appeared list select the interaction or create a new one.

To add an actual gate

1. Add a reference to the diagram, from which the diagram frame formal message is created.
2. To the current interaction use draw an actual message with the selected formal gate.
For more information about working with gates, see “Gate” on page 691.

Interface

An interface is a specifier for the externally-visible operations of a class, component, or other classifiers (including subsystems) without a specification of the internal structure. Each interface often specifies only a limited part of the behavior of an actual class.

The set of interfaces realized by a classifier is its provided interfaces, which represent the obligations that instances of that classifier have to their clients. They describe the services that the instances of that classifier offer to their clients.

All options associated with an interface can be set in the **Interface Specification** dialog box.

11 MODEL ELEMENTS

Interface

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box name | Function |
|---|----------|---|
| Signal Receptions | | Manage the receptions of an interface in the Signal Receptions pane. For more information about signal receptions, see “Reception” on page 759. |
| Inner Elements Add class, use case, interface, enumeration, data type, primitive, collaboration, or constraint to an interface | Name | The model element name. |
| | Type | the model element type. |
| | Create | Select an element from the list. The corresponding (class, use case, interface, enumeration, data type, primitive, collaboration, or constraint) specification dialog box opens. Define the selected model element in the dialog box. |
| | Delete | Remove the selected model element from an interface. |

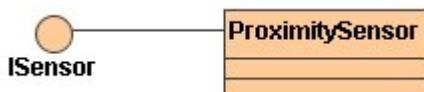
Provided and Required Interfaces

The set of interfaces realized by a classifier is its provided interfaces, which represent the obligations that instances of that classifier have to their clients. They describe the services that the instances of that classifier offer to their clients.

The interfaces may also be used to specify required interfaces, which are specified by a usage dependency between the classifier and the corresponding interfaces. Required interfaces specify services that a classifier needs in order to perform its function and fulfill its own obligations to its clients.

To draw a Provided Interface

1. In the Class diagram toolbar, select the Interface Realization path to draw from a class to an interface.
2. Suppress the attributes and operations of the interface (from the interface shortcut menu, **Presentation Options** submenu, select the **Suppress Attributes** and **Suppress Operations** check boxes.).



To draw a Required Interface

1. In the Class diagram toolbar, select the Usage path to draw from a class to an interface.
2. Suppress the attributes and operations of the interface (from the interface shortcut menu, **Presentation Options** submenu, select the **Suppress Attributes** and **Suppress Operations** check boxes.).



11 MODEL ELEMENTS

Interface

To draw both Provided and Required Interfaces together



Working with *s* is similar to working with classes. For more information, see “Working with classes” on page 663.

A general information about working with shapes is offered in "Diagramming"

Provided and Required Interfaces in the Composite Structure diagram

Term The OMG UML specification (UML 2.2: Superstructure Specification) states:

“Ports represent interaction points between a classifier and its environment. The interfaces associated with a port specify the nature of the interactions that may occur over a port. The required interfaces of a port characterize the requests that may be made from the classifier to its environment through this port. The provided interfaces of a port characterize requests to the classifier that its environment may make through this port.”

Information about provided and required interfaces is crucial in the assembly stage of complex internal structures. It helps to decide where connectors should be attached.

Provided and required interfaces are valuable parts of the UML Composite Structure Diagram and SysML Internal Block Diagram.

A provided interface is shown using the "lollipop" notation attached to the port and required interface is shown using the "fork" notation attached to the port (see Figure 441 on page 708).

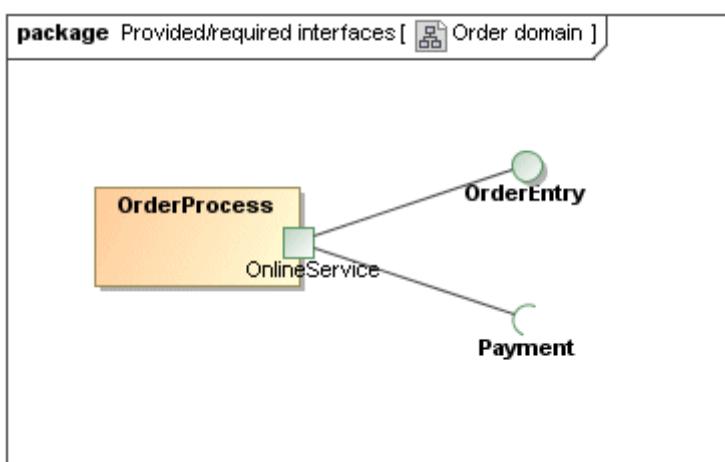


Figure 441 -- Provided and Required interface

11 MODEL ELEMENTS

Interface

In the Composite Structure diagram you cannot draw provided and required interfaces itself, but with the new functionality of MagicDraw you can display preexisting port with the required and provided interfaces as images.

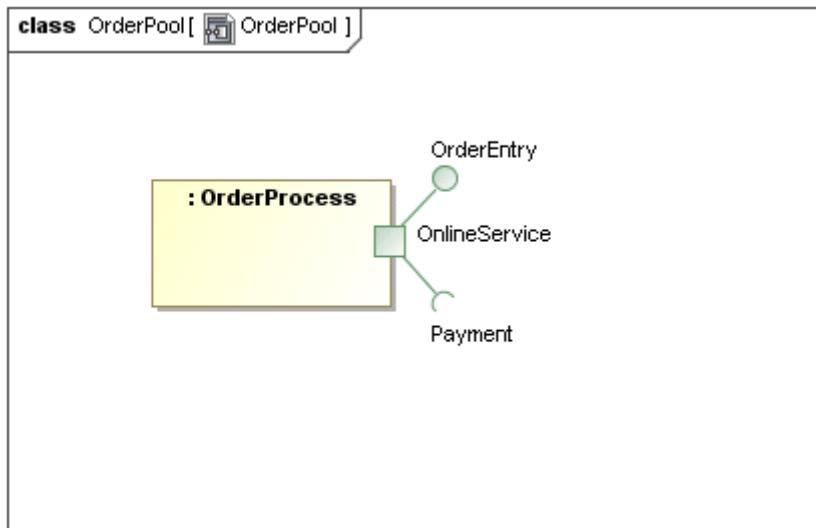


Figure 442 -- Provided and required interface in the Composite Structure diagram

Lollipop and fork symbols in the Composite Structure diagram are implemented as small attachments to a Port symbol (like name label). It is not the same as the independent standalone notation of the interface, it is only part of port symbol. It is important for Composite Structure diagrams where real Interfaces (as Classifiers) can not be used.

It is an optional notation, a port does not display provided or required interfaces by default.

Displaying provided/required interfaces in the Composite Structure diagram

1. Create provided and required interfaces in the Class diagram. See section “To draw a Provided Interface” on page 707 and section “To draw a Required Interface” on page 707.
2. In the Composite Structure diagram, select **Related Elements** from the port shortcut menu and then **Display Provided/Required Interfaces**. Or, in the individual Port symbol **Properties** dialog box, select the **Show Provided Interfaces** and **Show Required Interfaces** check boxes.

As Port can provide or require many interfaces, displayed or hidden interfaces can be managed in the **Edit Compartment** dialog box.

Provided/required interfaces in the Component diagram

A component specifies a formal contract of the services that it provides to its clients and those that it requires from other components or services in the system in terms of its provided and required interfaces.

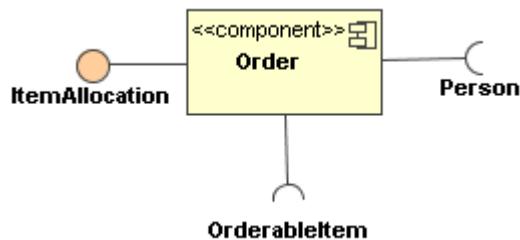
The required and provided interfaces may optionally be organized through ports.

To add and manage the added provided and required interfaces quickly, in the **Component Specification** dialog box, select the **Provided/Required Interfaces** pane.

11 MODEL ELEMENTS

Internal transition

For more information about provided and required interfaces, see “Provided and Required Interfaces” on page 707.



Internal transition

In all other cases, the action label identifies the event that triggers the corresponding action expression. These events are called internal transitions and are semantically equivalent to self transitions except that the state is not exited or re-entered. This means that the corresponding exit and entry actions are not performed.

For more information on defining transitions, see “The join vertices are used to merge several transitions emanating from the source vertices in different orthogonal regions. The transitions entering a join vertex cannot have guards or triggers.” on page 757.

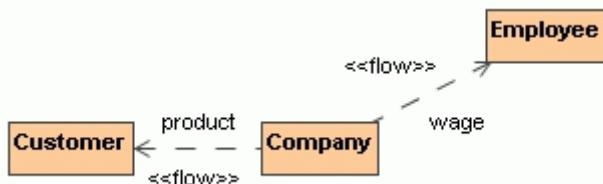
Specify the internal transition in the **Transition Specification** dialog box. For a detailed description of this dialog box, see “Transition’s Specification window” on page 773.

To define an internal transition

1. Double-click the state or select **Specification** from the state shortcut menu. The **State Specification** dialog box opens.
2. Click the **Internal Transitions** group.
3. Click the **Create** button. The **Transition Specification** dialog box opens. Specify an internal transition.

To remove the internal transition, click the **Delete** button.

The Information Flow can be related to any relationship.



Lifeline

A lifeline represents an individual participant in the Interaction. The lifeline represents only one interacting entity. It is shown using a rectangle symbol.

For more general information about working with symbols, see "Diagramming"

11 MODEL ELEMENTS

Lifeline

To assign a type (classifier) to a lifeline

1. Right-click the lifeline shape and select **Type** from the shortcut menu.
2. Select the classifier you wish to assign to a classifier role or click **New** and select the element from the list.

To hide/show a base classifier

Select the **Show Classifier** check box from the lifeline shortcut menu.

To create lifelines for existing data (the interaction properties and parameters) or display all lifelines of the interaction which are not displayed in the diagram

- Drag and drop the selected element from the Browser to the diagram pane.
- On the diagram shortcut menu, click **Related Elements > Display Lifelines** and, in the **Display Lifelines** dialog, choose the required elements you want to display or create.

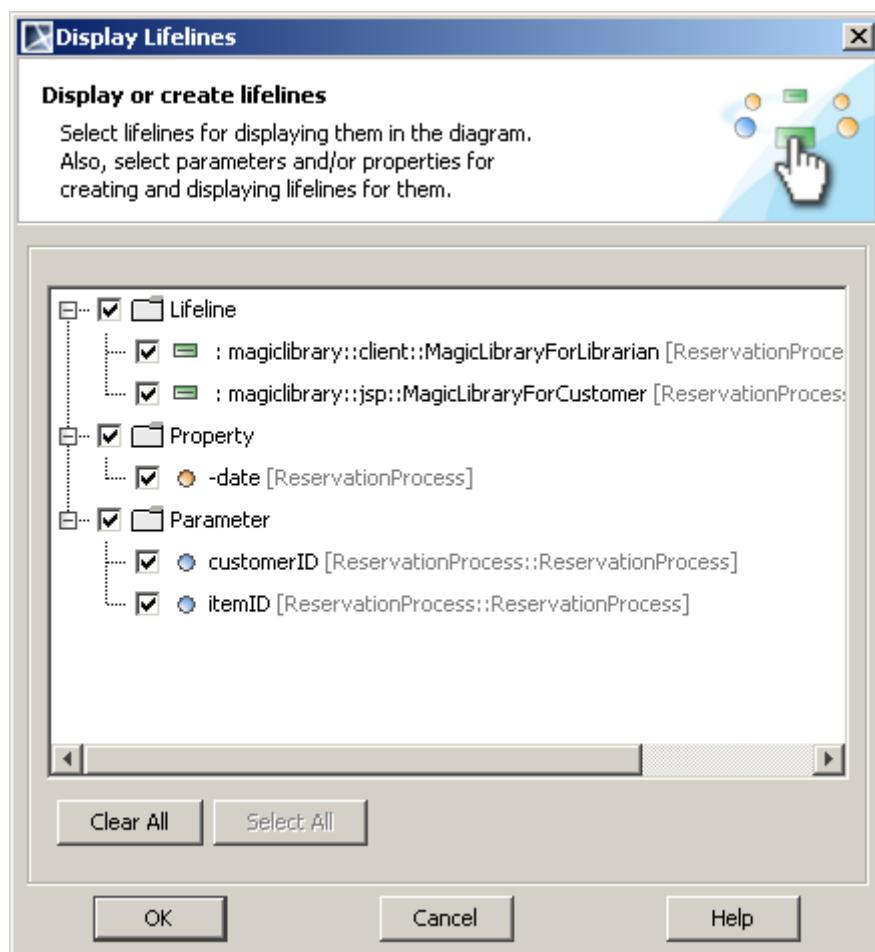


Figure 13 -- The **Dispaly Lifelines** dialog

Related sections

- [Communication Diagram](#)
- [Sequence Diagram](#)
- [Lifeline in the Sequence Diagram](#)
- [Message](#)
- [Message in Sequence Diagram](#)

Lifeline in the Sequence Diagram

A lifeline represents the existence of the object at a particular time. It stretches from the top to the bottom of the diagram. In the sequence diagram, an object lifeline denotes an Object playing a specific role.

To destroy a sequence object - a large “X” at the end of its lifeline marks its destruction

- Select **Delete Mark** from the lifeline shortcut menu.
- Set a message as delete message.

When an object receives a message, an activity starts in that object. An activation (focus of control) shows the period during which an object is performing an action either directly or through a subordinate procedure. The activation bar is used to denote that activity.

To change the activation bar size

1. Click the desired activation bar on the Diagram pane.
2. Drag the activation bar to the desired direction.

NOTE After resizing, the lines on the activation bar are thickened, but the size may not change automatically.

To add a recursive message to a lifeline

Click the **Recursive Message** button on the diagram toolbar and click the lifeline on the desired place where you wish to draw this message.

Related sections

[Lifeline](#)

[Sequence Diagram](#)

[Message in Sequence Diagram](#)

Link

Link is instance specification with assigned classifier - association.

An instance specification whose classifier is an association represents a link and is shown using the same notation as for an association, but the solid path or paths connect instance specifications rather than classifiers.

End names can adorn the ends. Navigation arrows can be shown, but if shown, they must agree with the navigation of the association ends.

You can show role names and navigability on the link symbol (see Figure 443 on page 712). Properties **Show End A**, **Show End B**, and **Show Navigability** are added to link shortcut menu and symbol properties. These properties are displayed according classifier association.

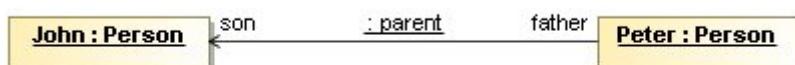


Figure 443 -- Role names displayed on the link symbol

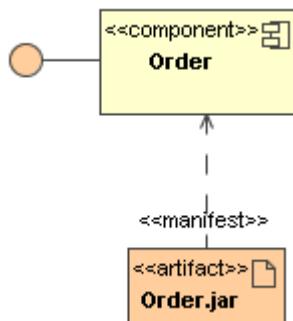
Manifestations

An artifact embodies or manifests a number of model elements. It owns the manifestations, each representing the utilization of a packageable element.

To create the manifestations, simply draw the Manifestation link from an artifact to a component.

To display the manifested artifacts on the component shape

From the component shortcut menu select **Presentation Options** and then clear the **Suppress Artifacts** check box.



Message

A Message is an element that defines one specific kind of communication in an Interaction. A communication can be, for example, raising a signal, invoking an Operation, creating or destroying an Instance. The Message specifies not only the kind of communication given by the dispatching Execution Specification, but also the sender and the receiver.

A message is shown as a line from the sender message end to the receiver message end. The form of the line or arrowhead reflect the properties of the message.

Related sections

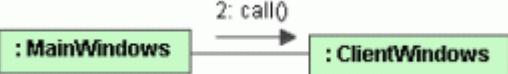
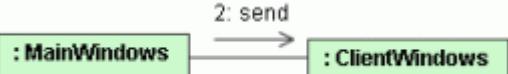
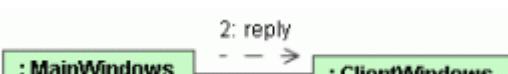
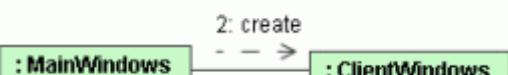
[Communication Diagram](#)

[Sequence Diagram](#)

| Name | Function | Notation in Communication Diagram |
|----------------------|---|--|
| Asynchronous Message | The sender is not waiting for the recipient's acceptance. | <p>The diagram shows two green rounded rectangles labeled ':MainWindows' and ':ClientWindows'. A horizontal line with a small arrowhead at the end of the line connects them. Above the line, the text '2: Layout' is written.</p> |
| Synchronous Message | The operation that should be completed before the caller resumes the execution. | <p>The diagram shows two green rounded rectangles labeled ':MainWindows' and ':ClientWindows'. A horizontal line with a small arrowhead at the end of the line connects them. Above the line, the text '2: Layout' is written.</p> |

11 MODEL ELEMENTS

Message

| Name | Function | Notation in Communication Diagram |
|------------------------|--|--|
| Call Message | A call message represents the request to invoke a specific operation. |  |
| Send Message | A send message specifies the sending of a request to invoke a specific operation. |  |
| Reply Message | The reply message returns the values to the caller of the previous call, completing the execution of the call. |  |
| Create Message | A create message specifies the creation of a specific operation. |  |
| Destroy Message | Destroy message represents the destruction of the instance described by the lifeline. |  |

To open the **Message Specification** dialog box

- Select **Specification** from the message shortcut menu or double-click the message.

11 MODEL ELEMENTS

Message

- Open the **Association Role Specification** dialog box. Open the **Messages** tab and click the **Edit** button.

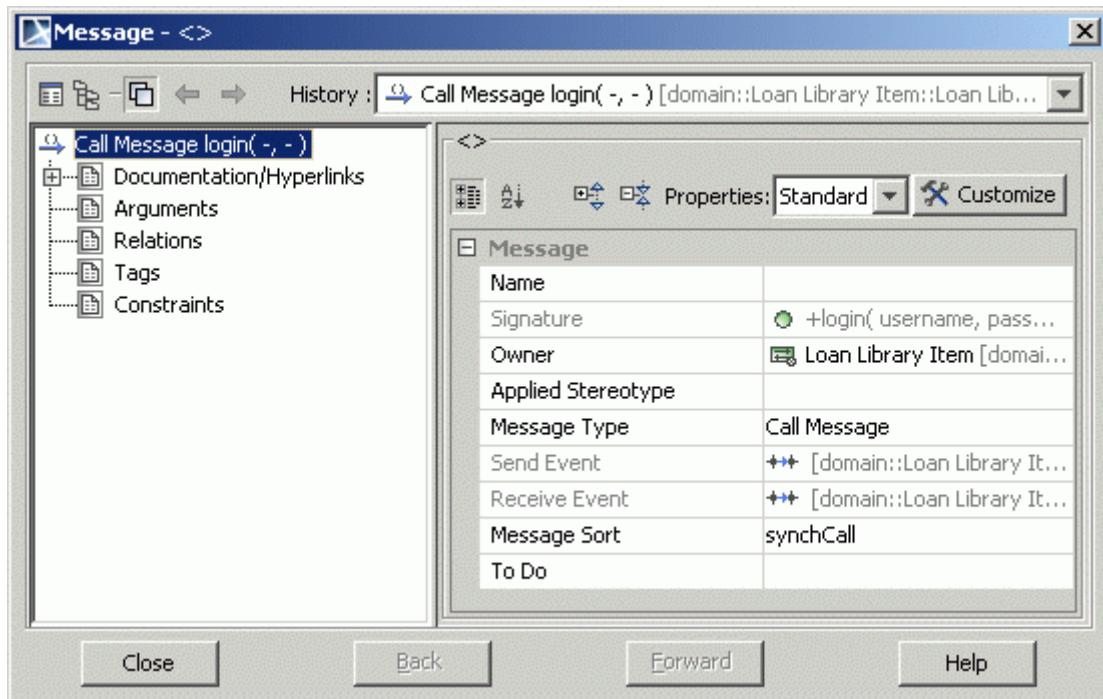


Figure 444 -- Message Specification dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box name | Function |
|--|----------------------|---|
| General Set a general information about the message. | Signature | The definition of the message sort, depending on its type. Note that this field is not editable. |
| | Owner | The name of the interaction, which owns the message. |
| | Send Event | The name of the sender interaction. |
| | Receive Event | The name of the receiver interaction. |
| | Message Sort | The type of the action: synchCall, asynchCall, asynchSignal, . Click the ‘...’ button to select a type from the list: <ul style="list-style-type: none">● synchCall● asynchCall● asynchSignal● createMessage● deleteMessage● reply |

11 MODEL ELEMENTS

Message

| Tab name | Box name | Function |
|---|------------------|---|
| Arguments Specify a message argument value that appears during the execution. | Parameter | Parameter column displays the parameter and the signature (owner's qualified name, type) of the parameter, which is referenced by the argument displayed in the corresponding argument table row. |
| | Argument | Argument column displays the value of the argument. |
| | Create | A list of arguments opens. Select an argument and specify the value in the Argument field. |
| | Delete | Removes an argument from the list. |
| | Up | Move the selected item to an upper position. |
| | Down | Move the selected item to a lower position. |

To draw a message on the connector

1. Click the **Message to Right** or **Left** button on the diagram toolbar.
2. Click the desired connector on the Diagram pane. A message arrow is placed to the connector you click on the diagram.

NOTE A message flow has two directions: right and left. Choose one of them by clicking the associated button on the diagram toolbar.

To set an action type for a message

1. Open the **Message Specification** dialog box.
2. Click the **Message Sort** “...” button and select the action type from the drop-down list. See “Message” on page 713.

To show/hide the numbers of the messages

From the communication diagram shortcut menu, select **Numbering** and then select/clear the **Show Meesage Numbers** check box.

To remove the automatic advanced numbering of messages

From the communication diagram shortcut menu, select **Numbering** and then clear the **Advanced Message Numbering** check box and you will be able to number messages according your needs.

To change the numbering of the messages

1. From the communication diagram shortcut menu, select the **Change Numbering** command.

11 MODEL ELEMENTS

Message

2. Increase, decrease, and/or change the level of numbering in the **Change Communication Numbering** dialog box:

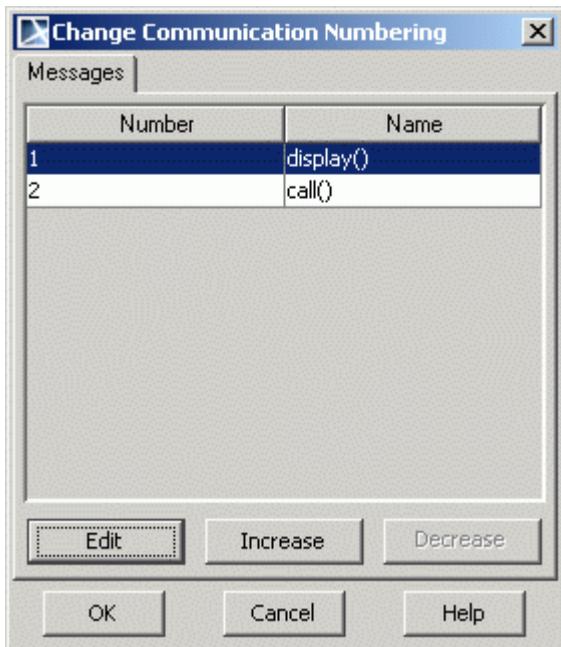


Figure 445 -- Change Communication Numbering dialog box

| Box name | Function |
|-----------------|--|
| Number | The number of the message. |
| Name | The type, name of the action of the corresponding message. |
| Edit | The Type Number dialog box opens. Type the number of the message. |
| Increase | Increases the selected number to one point. |
| Decrease | Decreases the selected number to one point. |

Assigning/Creating operation for message

Usually a model creation order is the following: there is a created class diagram with classes and then the sequence diagram uses the classes and operations to representing the call order.

MagicDraw allows a faster way to assign or create operations than the traditional model creation does. A sequence diagram is created to represent classes and messages, and with single click you can convert a message into a call as well as create operations in the classes according to the message name.

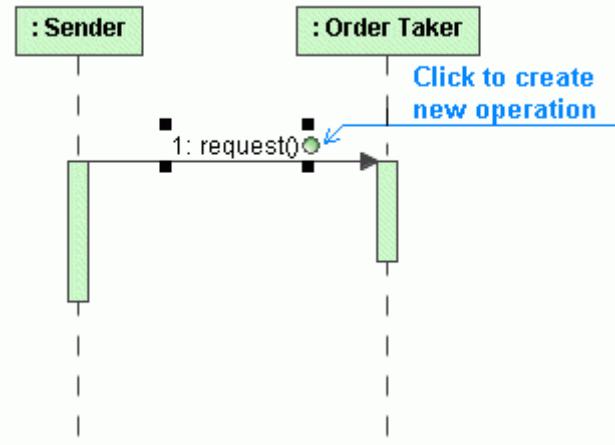
To create a new operation to the message

1. Draw call message between the lifelines.

11 MODEL ELEMENTS

Message

2. Select it on the diagram pane. Click on the smart manipulator at the end of the message name to create a new operation.



3. The new operation, the name of which is as same as that of the message, is created. The parameters, if provided, are added to the operation.
-or-
- From the message shortcut menu, select the **Create New Operation** command.

NOTE: The possibility to create a new operation exists only if the lifeline to which the message is drawn has an assigned type. The type element is not a read only type and it may have operations.
[update]

Assigning/Creating signal reception for message

To create a new signal reception to the message

1. Draw any send message between the lifelines. Lifelines need to have types specified.
2. Assign signal to the message.
3. Select message on the diagram pane. Click on the smart manipulator at the end of the message name to create a new signal reception.
4. The new signal reception, the name of which is as same as that of the message, is created for a type of a lifeline. The parameters, if provided, are added to the signal reception.
-or-
- From the message shortcut menu, select the **Create New Signal Reception** command.

Predecessors and activators

The predecessor is the set of messages of which the completion enables the execution of the current message. It is a comma-separated list of sequence numbers followed by a slash

('/'): sequence-number ',' . . . '/'

The meaning of the predecessor is that the execution of a message is not enabled until all of the communications of which the sequence numbers appeared in the list have occurred. Therefore, the list of predecessors represents a synchronization of threads. The message corresponding to the numerically preceding sequence number is an implicit predecessor and does not need to be explicitly listed.

All of the sequence numbers with the same prefix form a sequence. The numerical predecessor is the one in which the final term is one less. That is, number 3.1.4.5 is the predecessor of 3.1.4.6, where the number "3" is an activator.

11 MODEL ELEMENTS

Message in Sequence Diagram

To show the predecessors on the message

- Select **Show Predecessors** from the current message shortcut menu.
- Select the **Symbol(s) Properties** command from the message shortcut menu and select the **Show Predecessors** check box.

To change the activator number of the messages

1. Select the **Activator** command from the current message shortcut menu.
2. Select the number of activators that you want to assign for the current message and predecessors.

NOTE: If you change the activator number to one of the predecessor message, this number also changes for other predecessors.

Define the selected message in the **Message Specification** dialog box. For a detailed description of this dialog box, see Section “Message Specification dialog box” on page -715.

Message in Sequence Diagram

Messages are the main tool for displaying an interaction between objects, which is usually an operation call or a creation and destruction event. Messages are labeled with the name (operation or signal) and their argument values.

Related sections

- [Message](#)
- [Sequence Diagram](#)

Creating nested activation

Nested activations allow modeling parallel execution of operations that belong to a single class and modeling callback messages.

To create a nested activation, create at least two incoming messages for the activation, select the second (or any subsequent) message and from its shortcut menu select **Create Nested Activation**.

Nested activations can be created for the following message sorts:

1. **synchCall**
2. **asynchCall**
3. **asynchSignal**

Nested activations are used in three cases:

1. Modeling of parallel execution for a non-active object.
 - Create at least two incoming messages for the activation.
 - Select the second (or any subsequent) message and select the **Create Nested Activation** menu item from the shortcut menu.

11 MODEL ELEMENTS

Message in Sequence Diagram

- A nested activation will be created for the selected message.

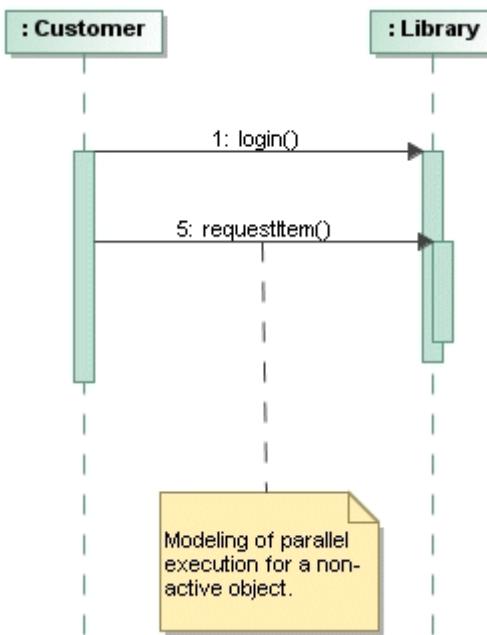


Figure 446 -- Nested activation

2. Modeling of parallel execution for an active object.

- Create an outgoing message for the activation that has the "Show Entire Activation" mode enabled.
- Select the created message and select the **Create Nested Activation** menu item from the shortcut menu.
- A nested activation will be created for the selected message.

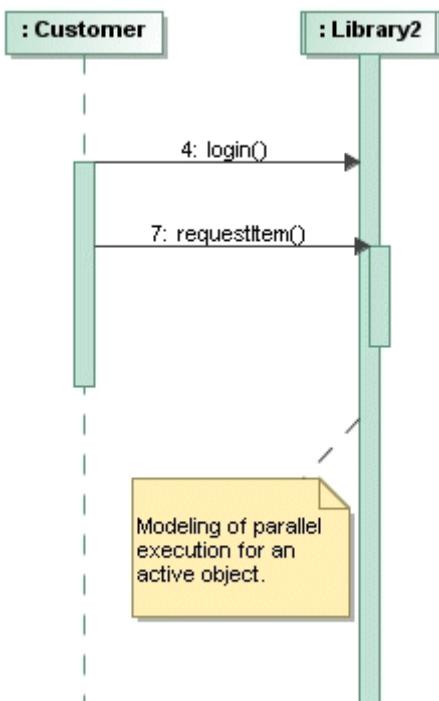


Figure 447 -- Parallel execution for an active object

3. Modeling of callback messages.

- Create an incoming message (the modeled callback message) for the activation that has an outgoing message to the activation for which the modeled message acts as an outgoing message.
- Select the created message and select the **Create Nested Activation** menu item from the shortcut menu.
- A nested activation will be created for the selected message.

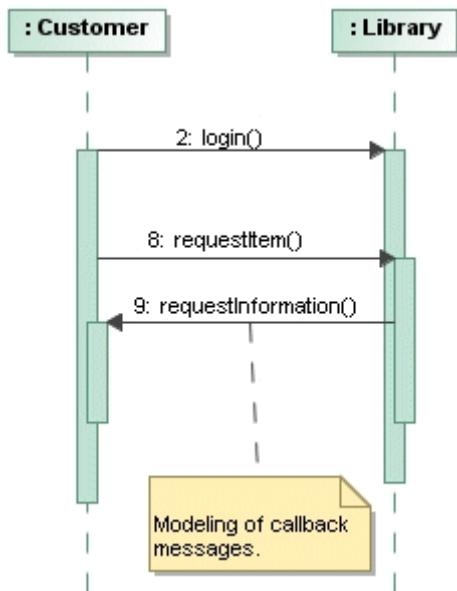


Figure 448 -- Callback messages

To merge the nested activations with parent activation

From the message shortcut menu select the **Reduce Nesting Level** command. The message will be connected to the parent activation.

Model

A model contains a (hierarchical) set of elements that together describe the physical system being modeled. It may also contain a set of elements that represents the environment of the system, typically Actors, together with their interrelationships, such as Associations and Dependencies.

A model is presented as a package with a small triangle in the upper right corner of the large rectangle. The triangle can be shown in the tab.

11 MODEL ELEMENTS

Node

The model is defined as a package – it has package properties - in the **Model Specification** dialog box. For a detailed description of packages, see “Package” on page 733.

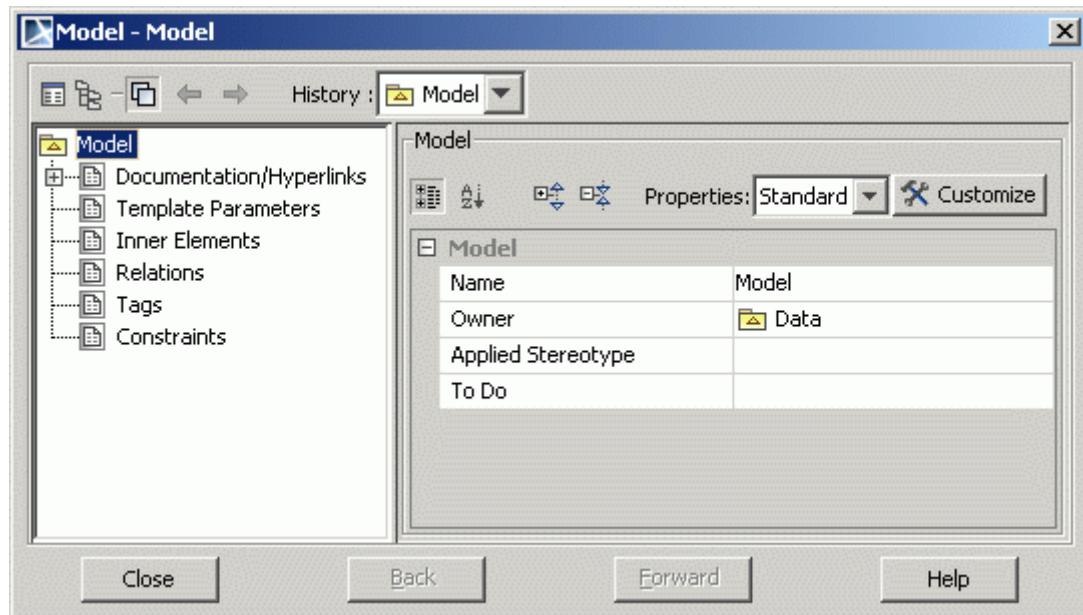


Figure 449 -- Model Specification dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

Node

Any computer or device that is relevant to the implemented system can be shown as a node. The node is drawn as a three-dimensional cube with a name inside it. Devices in a system are typically represented with a stereotype that specifies the device type. The nodes can be represented as types and as instances.

It is shown as a figure that looks like a 3-dimensional view of a cube.

For more information about working with symbols, see "Diagramming"

11 MODEL ELEMENTS

Node

Define the selected node in the **Node Specification** dialog box.

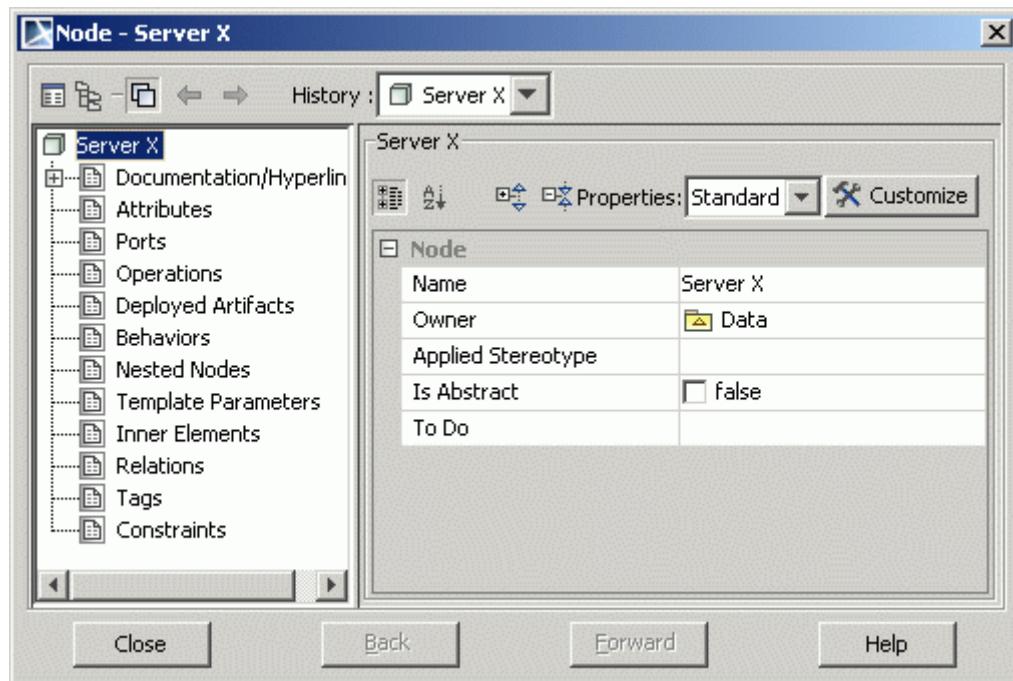


Figure 450 -- Node Specification dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box | Function |
|--------------------|-------------------|--|
| Ports | Name | Name of the port. |
| | Type | Type, assigned to the port. |
| | Provided | The provided classifier is displayed. |
| | Required | The required classifier is displayed. |
| | Classifier | The name of classifier, owning a port. |
| | Create | Create a new port. |
| | Delete | Remove an existing port from the node. |
| Deployed Artifacts | Name | Name of the deployed artifact. |
| | Type | Type of the item - artifact. |
| | Add | Add an artifact to the list. |
| | Remove | Remove an artifact from the list. |
| Behaviors | Name | Name of the behaviors. |
| | Type | Type, assigned to the behavior. |
| | Create | Select an item from the list - activity, interaction, and state machine. |
| | Delete | Remove a behavior from the class. |

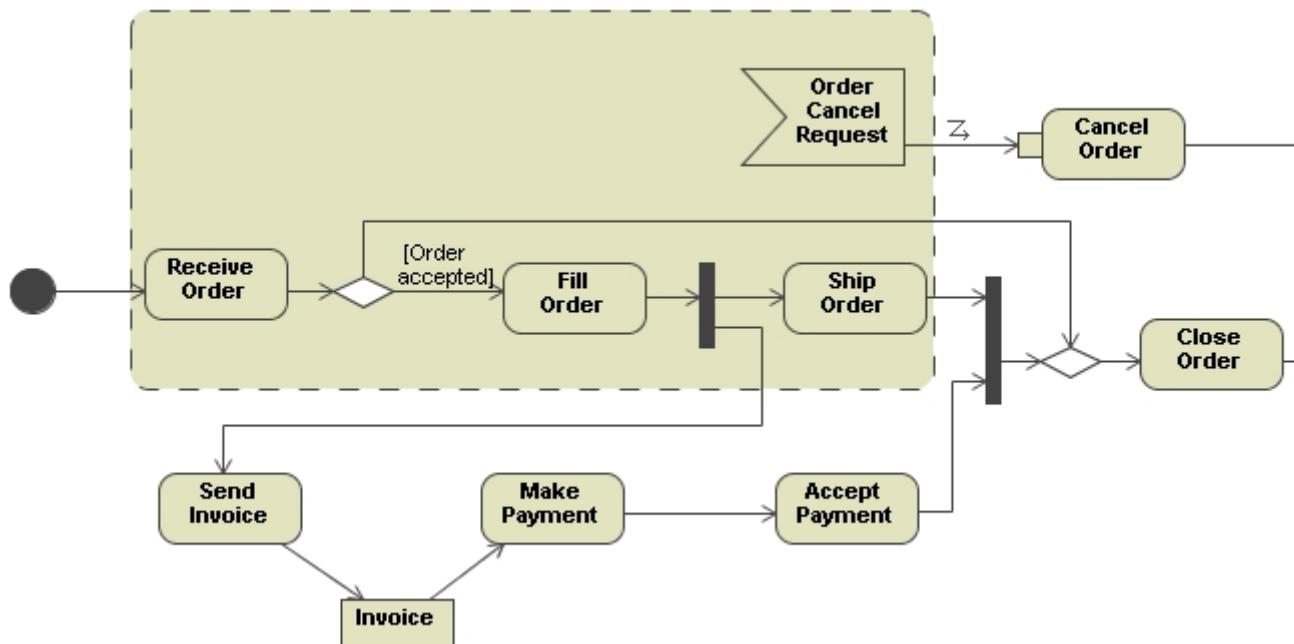
11 MODEL ELEMENTS

Node

| Tab name | Box | Function |
|---|---------------|--|
| Nested Nodes Add another element to a node. | Name | Name of the nested node. |
| | Type | Type of the classifier (class, interface, etc.). |
| | | Click  button to open the classifier Specification dialog box. |
| | Create | Select a node, device, or execution environment from the list. The element Specification dialog box opens. |
| | Remove | Remove a node from the list. |
| Inner Elements Add another element to a node. | Name | The model element name. |
| | Type | The model element type. |
| | Create | Select an element from the list. The corresponding specification dialog box opens. Define the selected model element in the dialog box. |
| | Delete | Remove the selected model element from the node. |

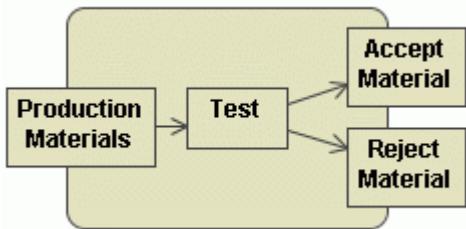
Structured activity node

A structured activity node is an executable activity node that may have an expansion into the subordinate nodes. It represents a structured portion of the activity that is not shared with any other structured node, except for nesting.



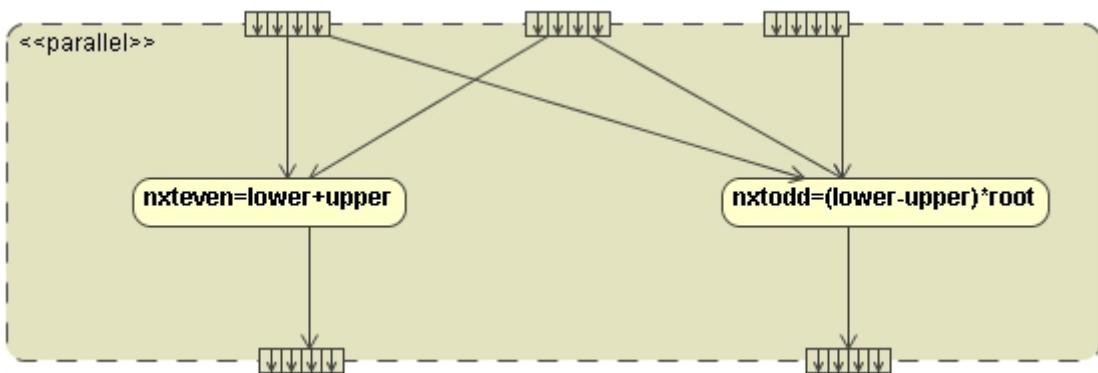
Activity parameter node

It is an object node for inputs and outputs to activities. The activity parameters are object nodes at the beginning and end of the flows, to accept inputs to an activity and provide outputs from it.



Expansion Region and Expansion Nodes

The Expansion Region and Expansion Nodes may be drawn in the activity diagram (the Input and Output Expansion Nodes may be found in the diagram toolbar Object Node button group):



If, Loop and Sequence Conditional Nodes

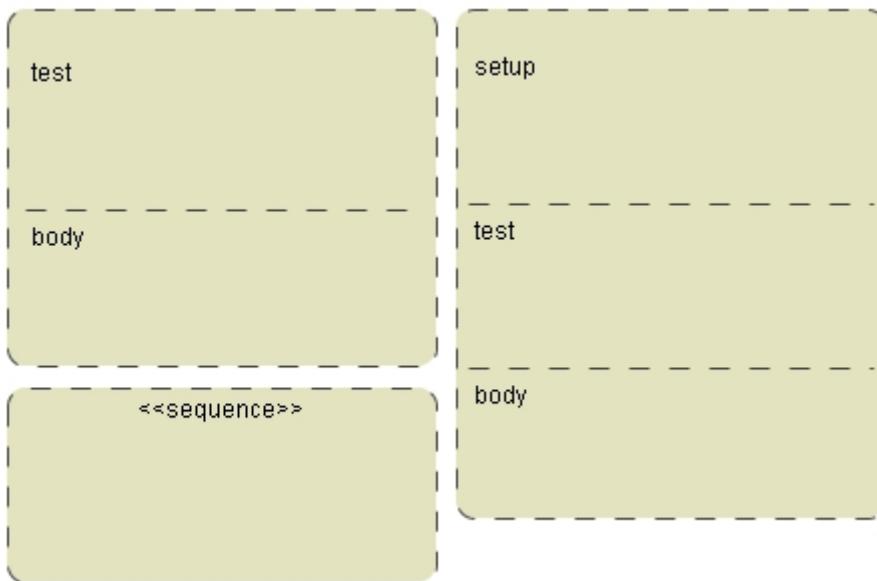
A conditional node is a structured activity node that represents an exclusive choice among some number of alternatives.

A sequence node is a structured activity node that executes its actions in order.

11 MODEL ELEMENTS

Object

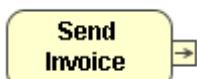
A loop node is a structured activity node that represents a loop with the setup, test, and body sections.



Object

To convert a pin to an object

1. Draw an Input, Output, or Value Pin on the Action:



2. Select an Output, Input or Value Pin on the diagram pane and from its shortcut menu select **Convert to Object**. The Pin is converted to an Object:



Object Flow

To split an Object Flow

1. Draw an Object Flow relation between two Actions.



11 MODEL ELEMENTS

Object Node

- From the Object Flow shortcut menu, select the **Split Object Flow** command.



Object Node

An object node is an activity node that indicates an instance of a particular classifier, possibly in a particular state, may be available at a particular point in the activity. It can be used in a variety of ways, depending on where the objects are flowing from and to, as described in the semantics section.

For more information about working with symbols, see "Diagramming"

Define the selected object node in the **Central Buffer Node Specification** dialog box.

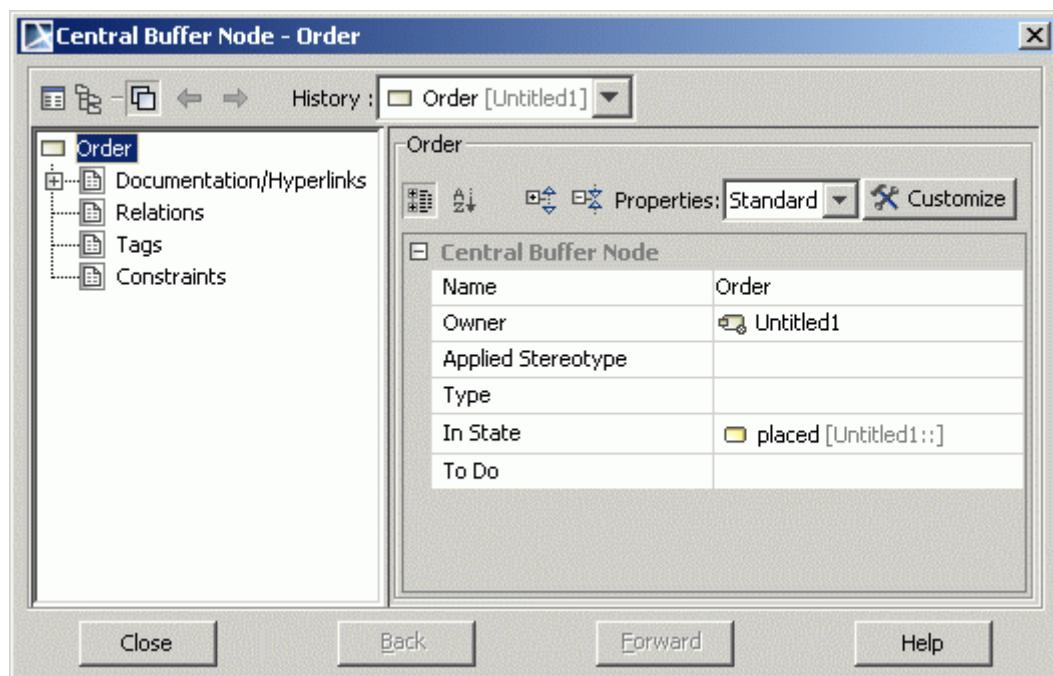


Figure 451 -- Central Buffer Node Specification dialog box

Refer to the "Specification Window" on page 237 for information about the specification elements not covered in this section.

| Tab name | Box name | Function |
|---|-------------|---|
| General Set a general information about the object node | Type | A list of classifiers. Select a classifier you wish to assign to an object node shape. Click the '...' button to open the Select Element dialog box for selecting the type from the elements tree. |

11 MODEL ELEMENTS

Opaque Behavior

| Tab name | Box name | Function |
|----------|----------|---|
| | In State | Click the “...” button to assign an existing final state or state from the model in the Select Elements dialog box, or click Create for defining a new one. |

To set a classifier to an object node

1. Double-click the object node or select **Specification** from the shape shortcut menu. The **Central Buffer Node Specification** dialog box opens.
2. Select a classifier you wish to assign to an object node from the **Type** drop-down list.

To assign a state or final state to an object node

1. Click the “...” button in the **Central Buffer Node Specification** dialog box, **In State** field. The **Select Elements** dialog box opens.
2. Select a state from the existing model elements, or click **Create**. The **State Specification** dialog box opens. Specify a new state, which will be assigned to an object node.

Opaque Behavior

A behavior with implementation-specific semantics. The Opaque Behavior is introduced for implementation-specific behavior or for use as a place-holder before one of the other behaviors is chosen.

To create a new Opaque Behavior:

1. In the Browser, select a package.
2. From the package shortcut menu, select **New Element** and then **Opaque Behavior**. Enter the name for a newly created element.

To create an Opaque Behavior symbol

Drag and drop the selected Opaque Behavior element from the Browser tree on the Diagram pane.

Operation

Entries in the operation compartment are strings that show the operations defined on classes as well as those that are supplied by the classes. An operation is a service that can be requested to perform by an instance of the class. It has a name and a list of arguments.

Usually class attributes are accessed through the operations. The operations are used to perform specific actions, such as system calls, utility functions, and queries. The operation signature provides all information needed to use that operation.

To create a new operation

- Double-click the selected class or select **Specification** from the class shortcut menu. The **Class Specification** dialog box opens. Click the **Operations** tab and then click the **Create** button. The **Operation Specification** dialog box opens. Define a new operation and click **OK**.

11 MODEL ELEMENTS

Operation

- Select **Insert New Operation** from the class shortcut menu. Type an operation name directly on the class shape.
- In the Browser tree, select an already created class. From the class item shortcut menu, choose **New** and then **Operation**.
- Select a class shape, press CTRL+ALT+O shortcut key and type the operation name on the Diagram pane.
- Select the class shape and click the small green **Insert New Operation** smart manipulation button.

Define an operation in the **Operation Specification** dialog box.

To open the **Operation Specification** dialog box

1. Double-click the class /actor or select **Specification** from the class/actor shortcut menu. The **Class Specification / Actor Specification** dialog box opens.
2. Click the **Operations** tab. Double-click the desired operation in the tree, or click the **Create** button. The **Operation Specification** dialog box opens.
- Double-click an operation on the Diagram pane or in the Browser.

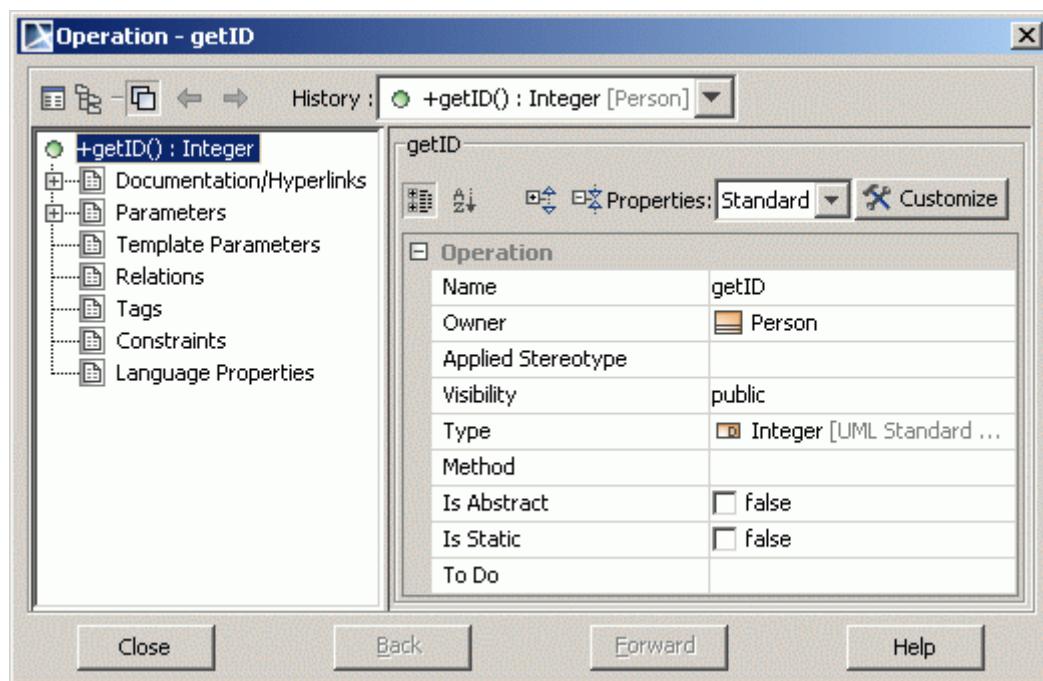


Figure 452 -- Operation Specification dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box | Function |
|--|--------|---|
| General Set a general information for the operation | Owner | Shows a class, which contains the current operation. The value of the Owner field cannot be changed. It is automatically defined when an operation is created. |
| | Type | Shows an operation type. It can be another project class or primitive type such as int or double. Select the type from the list or create a new one by clicking “...” button. |
| | Method | Click the “...” button to open the Select Elements dialog box. The Activity, Interaction, or State Machine elements can be assigned from the model. |

11 MODEL ELEMENTS

Operation

| Tab name | Box | Function |
|---|----------------------|--|
| Parameters A parameter is an unbound variable that can be changed, passed, or returned. | Name | Shows the parameter name. |
| | Type | Shows the parameter type. It can be a classifier or a data type. |
| | Default Value | An expression whose evaluation yields a value to be used when no argument (operation) is supplied for the parameter. |
| | Direction | Specifies what kind of parameter is required: <ul style="list-style-type: none">● return - return parameter.● in - an input parameter (may not be modified).● out - an output parameter (can be modified to communicate information to the caller).● InOut - an input parameter that can be modified. |
| | Up | Move the list up. |
| | Down | Move the list down. |
| | Create | The Parameter Specification dialog box opens. |
| | Delete | Removes the parameter. |

To change an operation name

1. Click the operation in the selected class on the diagram pane or in the Browser tree.
2. Type a new name.
 - Change an operation name in the **Operation Specification** dialog box.

To define the type of an operation

- In the **Operation Specification** dialog box, the **Type** drop-down list box, select the operation type.
- Type a colon ":" and the name of the operation type just after the operation name on the diagram pane. If you specify a nonexistent type of an operation, a new class is created.

To edit / add an operation parameter

1. Open the **Operation Specification** dialog box.
2. Click the **Parameters** tab.
3. Double click on the existing parameter name in the expanded tree or click the **Create** button. The **Parameter Specification** dialog box opens.
 - Type a parameter text (in parenthesis) directly on a diagram.
 - Select an operation in the Browser tree, select **New** from its shortcut menu and select **Parameter**. The **Parameter Specification** dialog box opens.

The Parameter Specification dialog box

The **Parameter Specification** dialog box defines an operation argument.

To open the **Parameter Specification** dialog box

1. Open the **Operation Specification** dialog box.

11 MODEL ELEMENTS

Operation

2. Click the **Parameters** tab, expand tree, and then double-click the desired parameter. Or click the **Create** button.

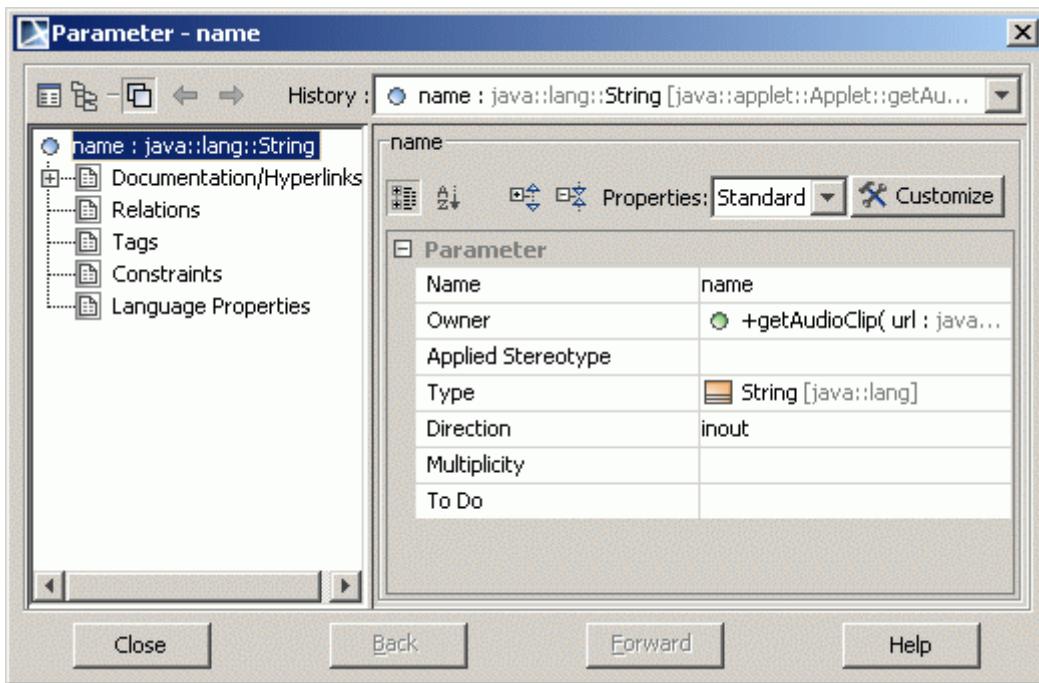


Figure 453 -- Parameter Specification dialog box

Click the **Show Expert Properties** button for showing more properties of the parameter.

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box | Function |
|--|---------------|---|
| General Set general information about the parameter | Owner | The name of the operation, which contains the parameter. |
| | Type | Shows the parameter type. It can be a classifier or a data type. Select a type from the list or create a new one by clicking the “...” button. |
| | Type Modifier | Additional information about the type. <ul style="list-style-type: none">• & - a parameter is a reference to other model element.• * - a parameter is a pointer to other model element.• [] - a parameter is an array of other model elements. |
| | Default Value | An expression whose evaluation yields a value to be used when no argument (operation) is supplied for the parameter. |
| | Direction | Select the direction kind: <ul style="list-style-type: none">• return - return parameter.• in - an input parameter (may not be modified).• out - an output parameter (can be modified to communicate information to the caller).• InOut - an input parameter that can be modified. |

To add additional information about the return type of an operation

1. Open the **Operation Specification** dialog box.

11 MODEL ELEMENTS

Operation

2. Click the **Show Expert Properties** button. More properties for the operation show.
3. Select a sign from the **Type Modifier** drop-down list box:
 - & - one class has a reference to other model element.
 - * - one class has a pointer to other model element.
 - [] - one class has an array of other model elements.

An operation can be defined as:

| Name | Function |
|--------------------|--|
| Is Abstract | The operation does not have an implementation, and one must be supplied by a descendant. |
| Is Static | This operation scope means that the values returned by the parameter have no duplicates. |
| Is Query | The operation does not change the state of the system. |

To define an operation as abstract, static, or query

1. Open the **Operation Specification** dialog box.
2. Select the **Expert** mode from the **Properties** field. More properties for the operation show.
3. Select the **Is Abstract**, **Is Static**, and/or **Is Query** check box(es) in the **General** tab.

To set the operation visibility

| Visibility name | Function |
|----------------------|--|
| Public '+' | The operation can be accessed by any other object from the outside. |
| Package '~' | The operation can be accessed by an element from the same package. |
| Private '-' | The operation can be accessed only from that class. |
| Protected '#' | The operation can be accessed from the inside of that class and the classes derived from that class. |

1. Open the **Operation Specification** dialog box.
2. From the **Visibility** drop-down list box, select **Public**, **Package**, **Private**, or **Protected**.

NOTE The operation visibility is shown in the operation signature.

11 MODEL ELEMENTS

Package

To set an operation Concurrency: sequential, guarded or concurrent

| Name | Function |
|-------------------|---|
| Sequential | The callers must coordinate, so that only one call to an Instance (on any sequential Operation) is made at a time. If simultaneous calls occur, then the semantics and the integrity of the system can not be guaranteed. |
| Guarded | Multiple calls from concurrent threads may occur simultaneously to one Instance (on any guarded Operation), but only one is allowed to commence. The others are blocked until the performance of the first Operation is complete. It is the responsibility of the system designer to ensure that deadlocks do not occur due to simultaneous blocks. The Guarded Operations must perform correctly (or block themselves) in case a simultaneous sequential Operation or guarded semantics cannot be claimed. |
| Concurrent | Multiple calls from concurrent threads may occur simultaneously to one Instance (on any concurrent Operation). All of them may proceed concurrently with correct semantics. The Concurrent Operations must perform correctly in case a simultaneous sequential or the guarded Operation, or concurrent semantics cannot be claimed. |

1. Open the **Operation Specification** dialog box.
2. Select the **Expert** mode from the **Properties** field. More properties for the operation show.
3. Select the concurrency type in the **Concurrency** drop-down list box.

Package

A package groups classes and other model elements together. All types of UML model elements can be organized into packages. Each diagram must be owned by one package and the packages themselves can be nested within other packages. Subsystems and models are special kinds of packages.

The packages may have dependency, generalization, realize, containment, and association relationships. These relationships are usually derived from the relationships between the classes inside those packages.

Working with packages

For more information about working with symbols, see "Diagramming"

11 MODEL ELEMENTS

Package

Define the selected package in the **Package Specification** dialog box.

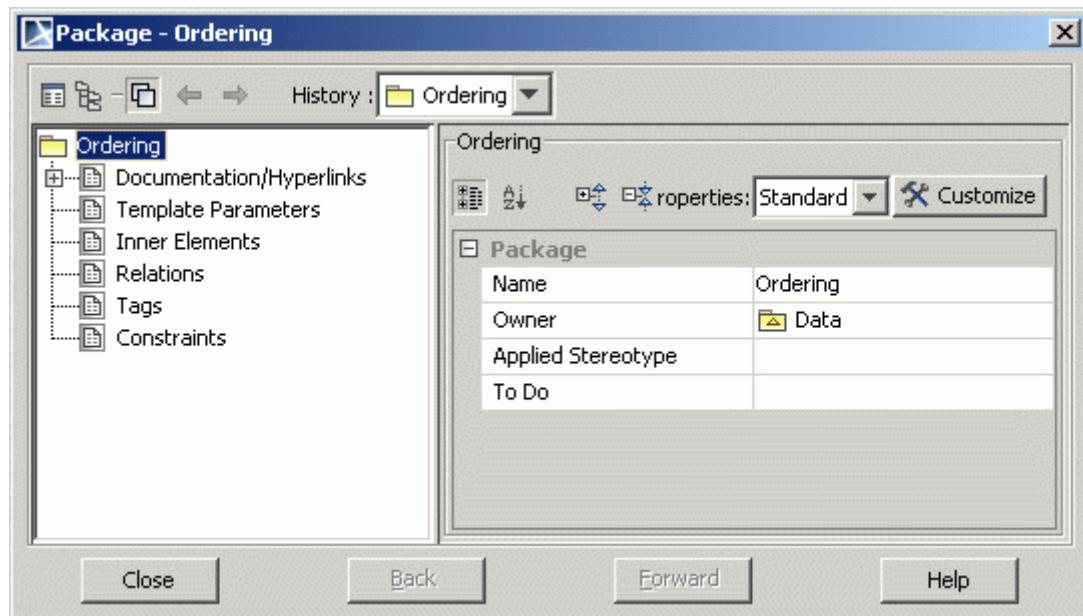


Figure 454 -- Package Specification dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

To add inner elements to the selected package

1. Open the **Package Specification** dialog box.
2. Click the **Inner Elements** tab.
3. Click the **Create** button, and then click an element you wish to add.
4. After selecting the element, the corresponding element **Specification** dialog box opens. If you select a diagram, the **Diagram Specification** dialog box opens.
5. Define the information and click **Back** to return to **Package Specification**.

To change the package header (name, stereotypes, tagged values, and constraints) position

- From the package shortcut menu, select **Header Position** and then **Top** or **In Tab**.
- Select **Symbol(s) Properties** from the package shortcut menu, and then **Header Position Top** or **In Tab** in the **Properties** dialog box.

To show the list of elements assigned to a package on the package shape

Select **Show Elements List** from the package shortcut menu.

Displaying Package Inner Elements in a Diagram

The **Select Inner Elements** dialog allows you to quickly display inner elements of the package in a diagram.

To display inner elements of the package in a diagram:

1. Right-click a package in a diagram.
2. Select **Related Elements > Display Inner Elements** from the package shortcut menu (Figure 455 on page 735). The **Select Inner Elements** dialog will open (Figure 456 on page 735).

11 MODEL ELEMENTS

Package

3. Select the elements to be displayed and click **OK**. The selected elements will be displayed in the diagram's package (Figure 457 on page 736).

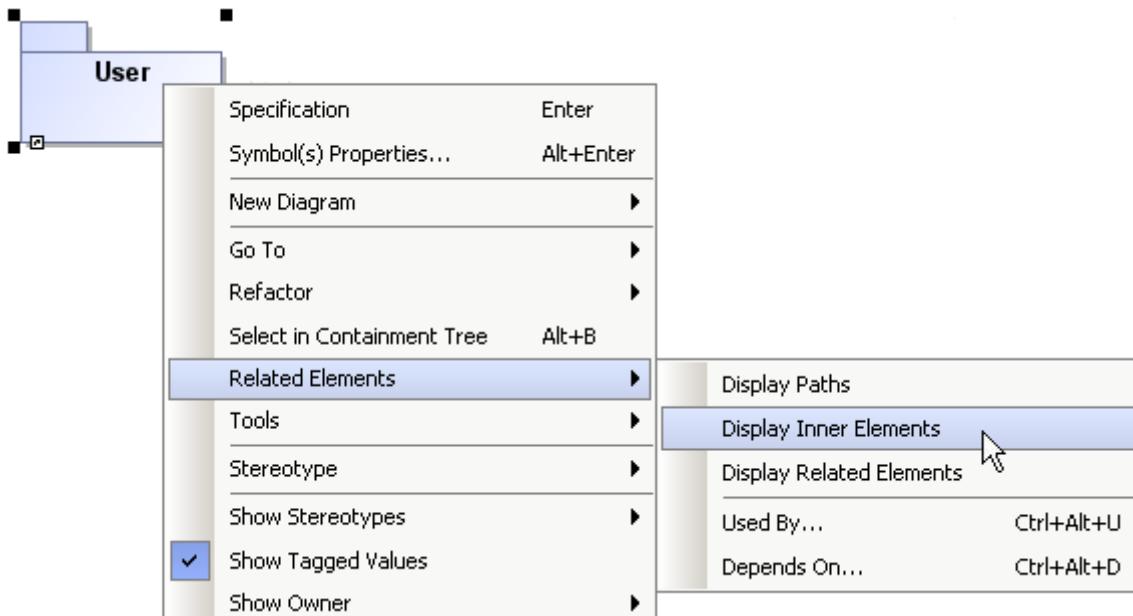


Figure 455 -- Display Inner Elements Menu

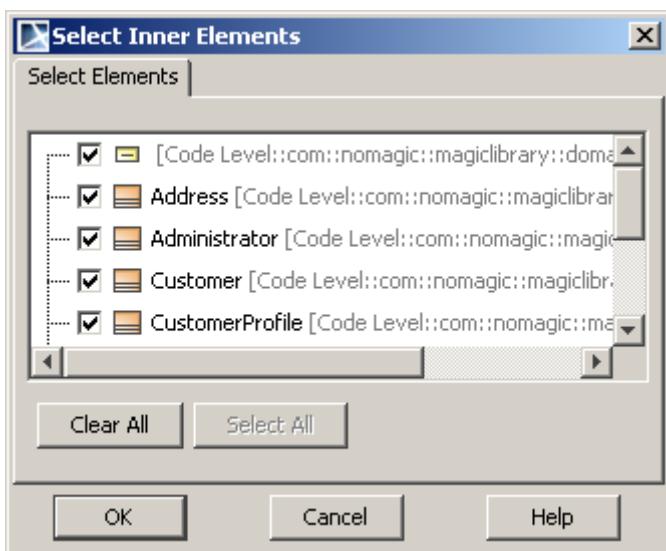


Figure 456 -- The Select Inner Elements Dialog

11 MODEL ELEMENTS

Parameter

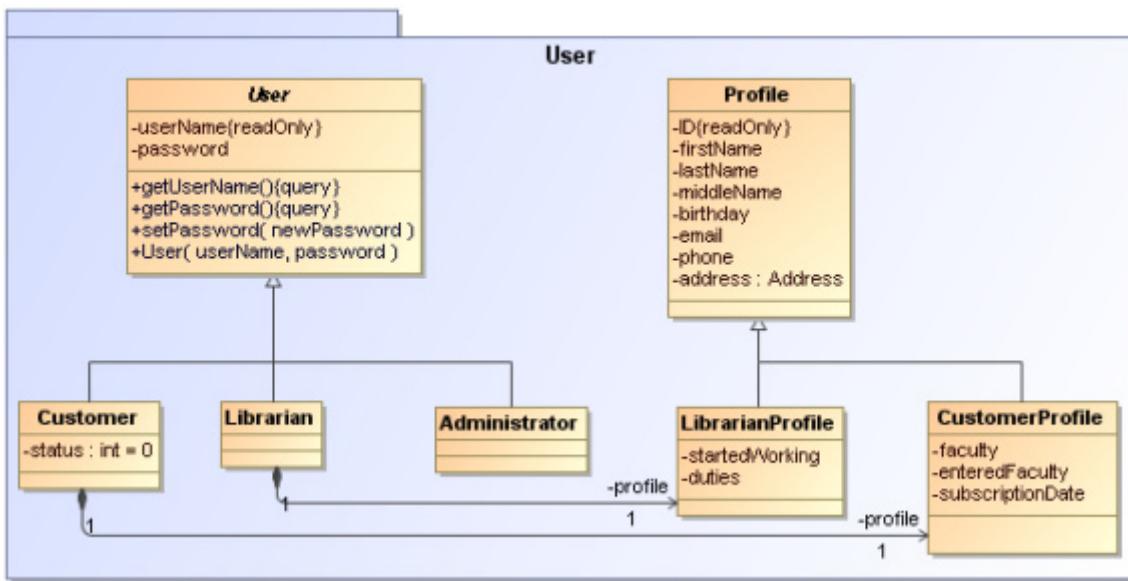


Figure 457 -- User Package with Displayed Inner Elements

Parameter

Parameters synchronization with Arguments

After you have modeled certain references between elements, arguments will be created automatically according to the parameters. Such synchronization increases modeling speed and helps to avoid invalid models.

The table below lists parameter and argument synchronizations.

| Link to synchronization description | Parameter | Argument | Paired elements |
|--|--|---|---------------------------------------|
| 1 "Synchronization between Operation parameters and Behavior parameters" | Operation parameter | Behavior parameter | Parameter and paramemeter |
| 2 "Synchronization between Activity parameters and Activity Parameter Nodes" | Activity parameter | Activity Parameter Node | Parameter and activity parameter node |
| 3 "Synchronization between Operation parameters and pins on Call Operation Action" | -Operation parameter -Behavior parameter | - Pin of Call Operation Action - Pin of Call Behavior Action | Parameter and Pin |
| 4 "Synchronization between Interaction parameters and Interaction Use arguments". | Interaction parameter Operation parameter | - Argument of Interaction Use - Argument of Message | Parameter and Argument |

11 MODEL ELEMENTS

Parameter

| | Link to synchronization description | Parameter | Argument | Paired elements |
|---|--|-----------------------|----------|------------------------|
| 5 | "Synchronization between Interaction Parameters and Lifelines" | Interaction parameter | Lifeline | Parameter and Lifeline |

Created arguments have the same number, order, and name as the parameters. Some properties of the parameters are cloned to argument properties, such as name, type, direction, and multiplicity for a particular argument.

Changes made in parameters are reflected in arguments. Changes to arguments are not reflected in parameters. Exception: synchronization between Activity Parameters and Activity Parameter Node.

When synchronization case is removed, arguments created on synchronization are not removed, but synchronization between parameters and arguments is not working anymore.

For synchronization to work it should match criteria, such as number, order, or other criteria that should be the same for the parameter and argument.

To turn on/off the parameters and arguments synchronization for the whole project, select/clear the **Auto synchronize Parameters and Arguments** check box in the **Project Options** dialog box. After the synchronization is turned off, arguments will not be created and modified on parameters creation and modification. By default the **Auto synchronize Parameters and Arguments** check box is selected.

The new in MagicDraw 15.5 Active Validation functionality improves the parameters and arguments synchronization. Active Validation functionality displays unsynchronized elements on diagram pane and in Browser. You can also use the **Parameters Synchronization** dialog for automatic and manual synchronization solving. For more information about the active validation see Active Validation.

Rules of synchronization between parameters and arguments

In this section you will find information about rules for making the synchronization between parameter and argument function properly. If parameters and arguments do not match rules, they are not synchronized and no changes to arguments will be performed when parameters change.

The general rules of synchronization are:

4. The number of parameters and arguments should be the same.
5. The order of parameters should be the same as order of arguments. If the order of parameters is changed, the order of Arguments is changed too. This is valid if arguments have ordering possibility.
6. The same properties. The parameter properties should be the same as the argument properties. Such as name, type, multiplicity, direction.
7. Properties change. Change the parameter property and the argument property changes. Some of the properties are changed only the first time and after the second change the parameter property is not changed and synchronization is not performed anymore.
8. Each couple of parameters and arguments should be synchronized. If one of them is not synchronized, the other is not synchronized as well. Note that synchronization is checked in element scope.

Synchronization between Operation parameters and Behavior parameters

Term The OMG UML specification (UML 2.2: Superstructure) states:

ownedParameter: Parameter

References a list of parameters to the behavior that describes the order and type of arguments that can be given when the behavior is invoked and of the values that will be returned when the behavior completes its execution.

Definitions *BehavioralFeature* - operation;

Behavior - Activity, Function Behavior, Interaction, Opaque Behavior, Protocol State Machine, State Machine.

Synchronization between operation parameters and behavior parameters works in the following way: after you have assigned a behavior to the operation with parameters, arguments to the behavior will be created automatically.

How synchronization works

After the operation parameters have been modified, arguments change in the following way:

- Create operation parameter - argument is created and properties cloned according to the parameter properties.
- Edit operation parameter - argument properties change. Argument name changes according to parameter properties only on the first parameter name change. For example, for operation create not named parameter. Assign behavior as method to the operation. To behavior not named parameter is created. Now name the operation parameter. Behavior parameter name automatically changes to the operation parameter name. Change the operation parameter the second time. The title of behavior parameter is not changed.
- Remove operation parameter - behavior argument is removed if it does not have links or values. For example, operation parameter is synchronized with activity parameter. Activity parameter is included to other synchronization - activity parameter is synchronized with activity parameter node (see “Synchronization between Activity parameters and Activity Parameter Nodes” on page 740). In this case, after the operation parameter is removed, the activity parameter is not removed, because it has link.

Validation of Synchronization

Synchronization between Parameter and Argument is valid when:

- The type is compatible. The type of parameter and behavior parameter should be compatible. This means that the type of operation parameter and the type of behavior parameter should be the same or inherited.
- The direction is the same. The direction of the operation parameter and the behavior parameter should be the same.
- The multiplicity is the same. The multiplicity of the operation parameter and the behavior parameter should be the same.

If one of the rules is not valid, operation parameters and behavior parameters will not be synchronized anymore.

11 MODEL ELEMENTS

Parameter

Sample

1. Create a class named *Computer*, with an operation called *Collect*, and with parameters *Accepted Computers*, *Production Materials*, and *Rejected Computers*.
2. Create Activity diagram *Collect Computer*.

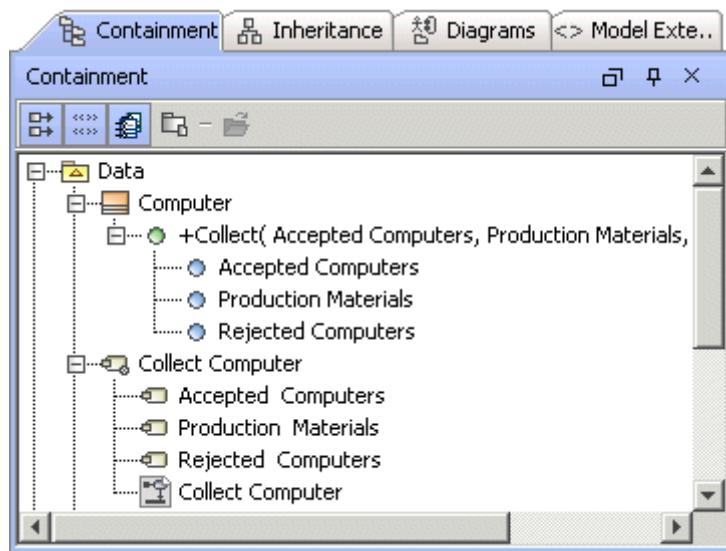


Figure 458 -- Project before synchronization

3. Assign activity diagram *Collect Computer* to the operation *Collect* as behavior. To do this from the operation shortcut menu in the Browser, select the **Behavior Diagram** and then **Assign**. Activity of activity diagram is assigned as a method to the operation automatically. The text after the name of activity will appear in the Browser: *specifies Collect*.

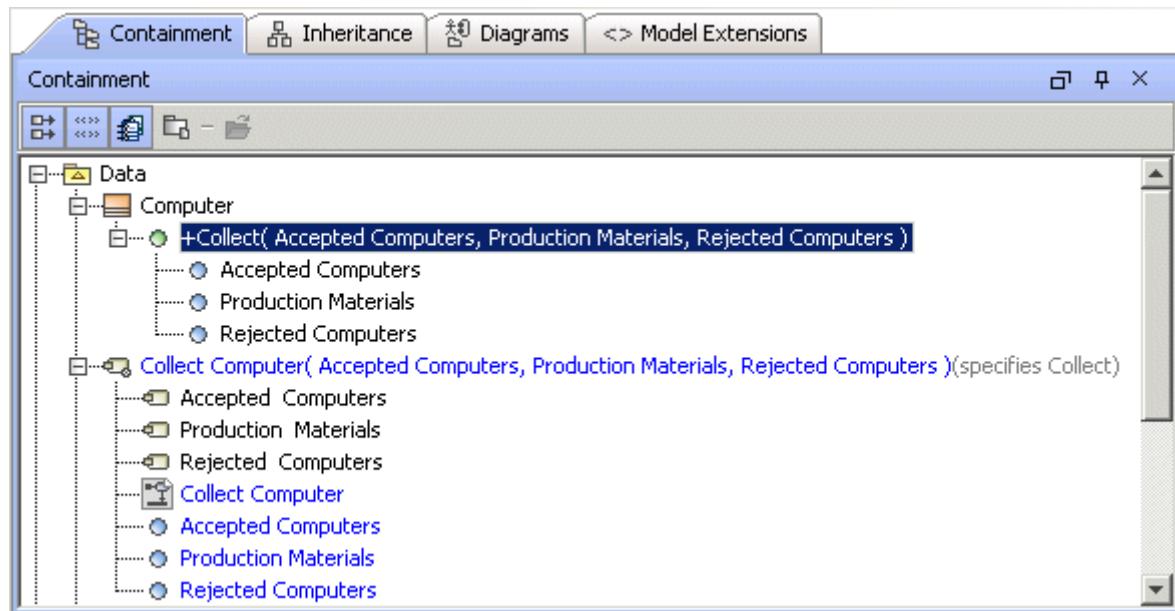


Figure 459 -- Project after synchronization between Operation Parameters and Behavior Parameters

4. The following operation parameters are created to the activity of activity diagram automatically: *Accepted Computers*, *Production Materials*, *Rejected Computers*. The *Collect* operation parameters are synchronized with *Collect Computer* activity parameters.

Synchronization between Activity parameters and Activity Parameter Nodes

Terms The OMG UML specification (UML 2.2: Superstructure) states:

Activity Parameter Node

“Activity parameter nodes are object nodes at the beginning and end of flows that provide a means to accept inputs to an activity and provide outputs from the activity, through the activity parameters.

Activity parameters inherit support for streaming and exceptions from Parameter.”

Constraints

[1] “Activity parameter nodes must have parameters from the containing activity.”

[2] “The type of an activity parameter node is the same as the type of its parameter.”

Definition *Parameter - Parameter of Activity;*
s

Argument - Activity Parameter Node.

Synchronization between activity parameters and activity parameter nodes works in the following way: after you have created a parameter to the activity, an activity parameter node of the activity will be created automatically.

How synchronization works

How synchronization works on parameter edit:

- Create a parameter in the activity. After you have created the parameter, the argument for parameter will be created automatically. If the parameter direction is *inout* - two activity parameter nodes will be created.
- Remove a parameter from the activity. After you have removed the parameter from the activity, the argument will be removed as well. If two arguments were created for one *inout* parameter, both arguments will be removed.

How synchronization works on argument edit:

11 MODEL ELEMENTS

Parameter

- Create an activity parameter node in the activity. The **Select Activity Parameter** dialog box appears. Select the existing activity parameter or click the **Create** button to create a new one.

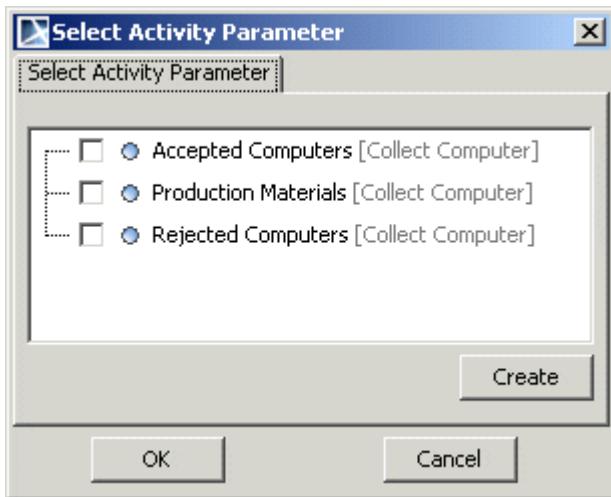


Figure 460 -- Select Activity Parameter dialog box

- Remove synchronized activity parameter node. After you have removed the activity parameter node from the activity, the assigned activity parameter will be removed as well.

NOTES In the **Select Activity Parameters** dialog box, the following activity parameters are not displayed:

- Inherited parameters
- Parameters which already have Activity Parameter Nodes.

Sample

1. In the *Collect Computer* activity create *Accepted Computers* parameter.
2. The *Accepted Computers* activity parameter node is created in activity automatically. Properties of the parameter and activity parameter node are synchronized..

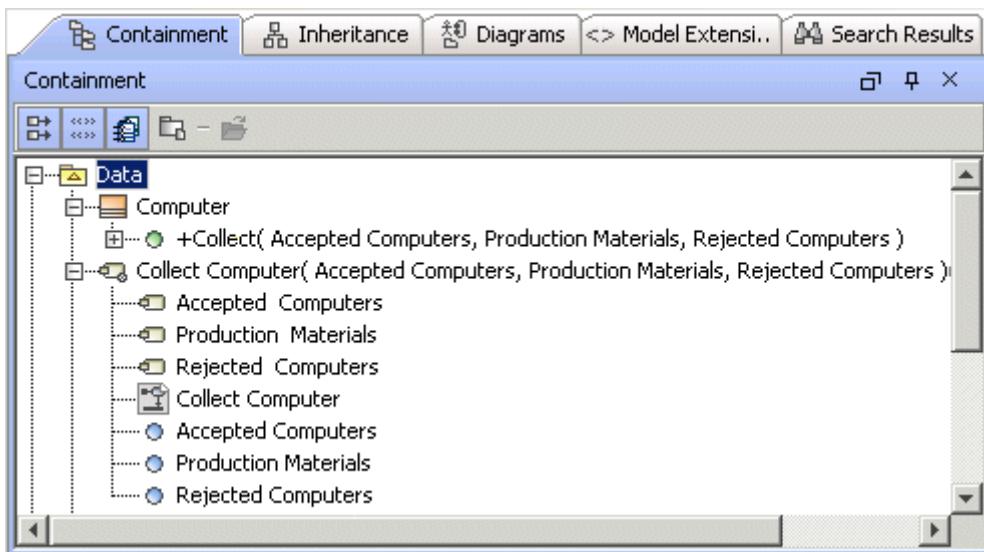


Figure 461 -- Synchronization of Activity Parameters with Activity Parameter Nodes

Synchronization between Operation parameters and pins on Call Operation Action

And

Synchronization between Behavior parameters and pins of Call Behavior Action

Terms The OMG UML specification (UML 2.1.2: Superstructure, p. 244) states:

Call Action

“Constraints

[1] Only synchronous call actions can have result pins.

[2] The number and order of argument pins must be the same as the number and order of parameters of the invoked behavior or behavioral feature. Pins are matched to parameters by order.

[3] The type, ordering, and multiplicity of an argument pin must be the same as the corresponding parameter of the behavior or behavioral feature.”

The OMG UML specification (UML 2.1.2: Superstructure, p. 244) states:

Call Behavior Action

“Constraints

[1] The number of argument pins and the number of parameters of the behavior of type in and in-out must be equal.

[2] The number of result pins and the number of parameters of the behavior of type return, out, and in-out must be equal.

[3] The type, ordering, and multiplicity of an argument or result pin is derived from the corresponding parameter of the behavior.”

Definition **Parameter** - Parameter of Operation, Parameter of Behavior;
s

Argument - Pin of Call Operation Action; Pin of Call Behavior Action.

How synchronization works

How synchronization works on parameter edit:

- Create parameter. After you have created a parameter, pins to the Call Operation Action or to the Call Behavior Action will be created automatically. If parameter direction is *inout*, two Pins to one parameter will be created.
- Remove parameter. After parameter is removed, the argument is removed automatically if it does not have links or values.

How synchronization works on argument edit:

11 MODEL ELEMENTS

Parameter

- Create pin. Create to the Call Operation Action or the Call Behavior Action input or output pin. The **Select Operation Parameter** dialog box appears for the Call Operation Action element and the **Select Behavior Parameter** dialog box appears for the Call Behavior Action.

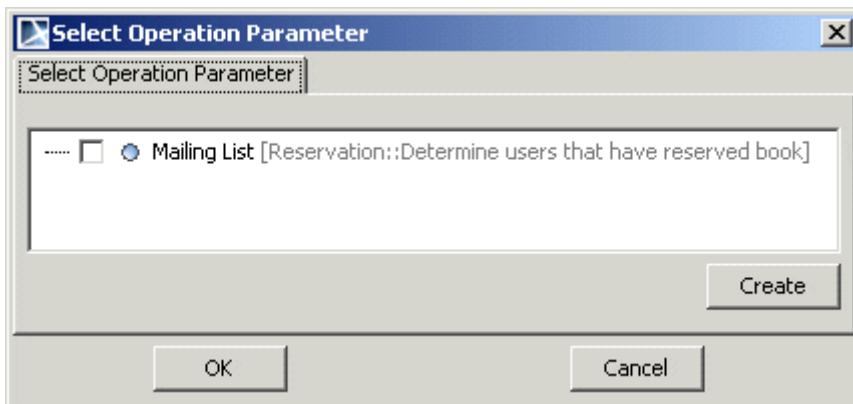


Figure 462 -- Select Operation Parameter dialog box

- Remove pin. After you have removed the pin, the parameter will not be removed.
- The order change and the properties change. After you have changed the order of arguments and after you have changed properties, no changes will be done to the parameters.

Synchronization validation

Synchronization between Parameters and Argument is valid if:

- The direction is compatible. The parameter direction and the pin direction should be compatible.
- The types are compatible. The parameter type and the pin type should be the same or inherited.

Sample

1. Create the *Mailing List* parameter to operation.

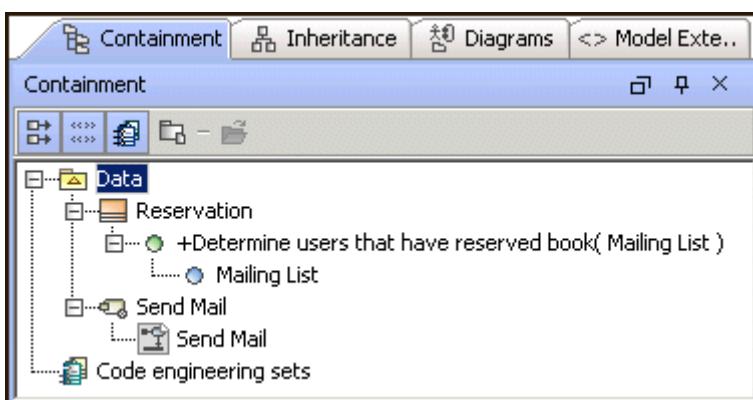


Figure 463 -- Project before synchronization

2. Create the call operation action named *Determine users that have reserved book*.

11 MODEL ELEMENTS

Parameter

3. Assign an operation to the call operation action. Pins to the call operation action will be created automatically.

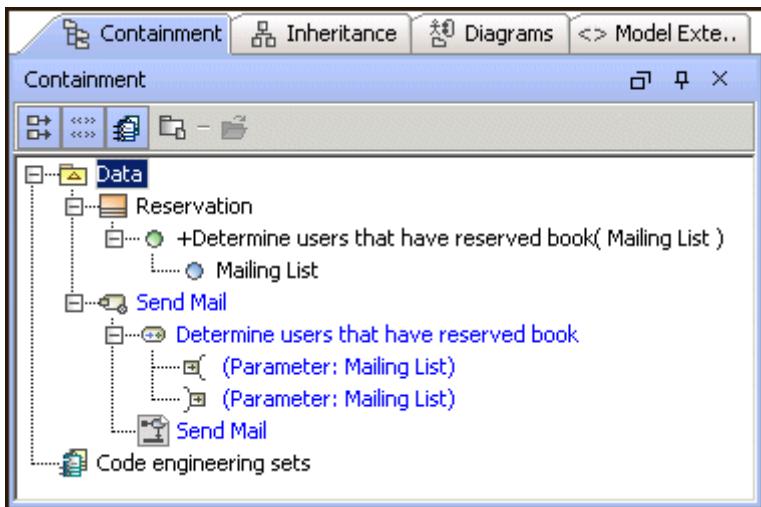


Figure 464 -- Project after synchronization between Operation Parameter and Pin of Call Operation Action

Synchronization between Interaction parameters and Interaction Use arguments

And

Synchronization between Operation parameters and Message arguments

Definition *Parameter* - Interaction Parameter, Operation Parameters;

s

Argument - Interaction Use Argument, Message Arguments.

Synchronization between interaction parameters and interaction use arguments works in the following way: after you have referred an interaction use to the interaction with parameter, an argument to the interaction use will be created automatically. The parameter of interaction is synchronized with interaction use argument.

Synchronization between operation parameter and message arguments works in the following way: after you have created the message with an assigned operation, the arguments to the message will be created automatically.

How synchronization works

How synchronization works on parameter edit:

- Create. After you have created interaction use or message, arguments will be created automatically according to parameters.
- Remove parameter. After you have removed the parameter, the synchronized argument will be removed as well.
- The order change. After you have changed the parameters order, arguments order will be changed automatically.
- The property change. After you have changed parameter properties, this change will not affect the argument.

Important Changes in arguments are not reflected in parameters.

11 MODEL ELEMENTS

Parameter

Sample

1. Create a class with *Collect* operation and with *Production Material* parameter. Create the sequence diagram. Draw a lifeline with assigned type - *Computer* class. To the lifeline, draw a call message.

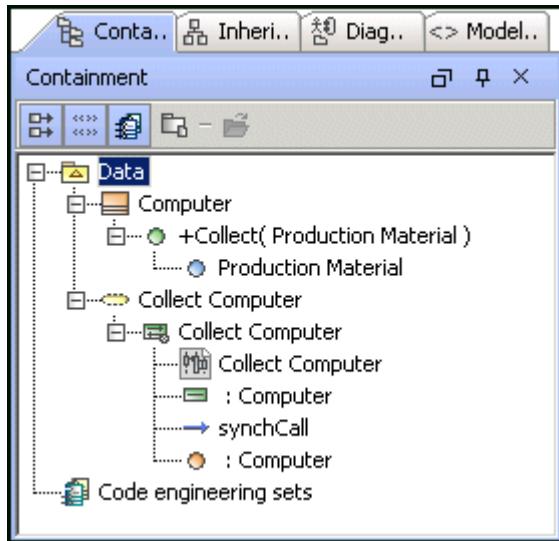


Figure 465 -- Project before synchronization

2. Assign the *Collect* operation to the call message. An argument will be added to the message automatically. To see the created argument, open the **Message specification** dialog box and select the **Arguments** branch. In the sequence diagram on the message, you can see the parameter name with argument name (in this example, the argument is not named).

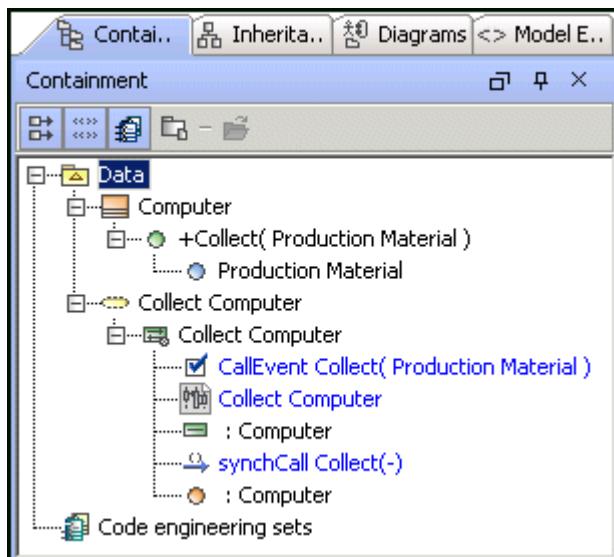


Figure 466 -- Project after synchronization

11 MODEL ELEMENTS

Parameter

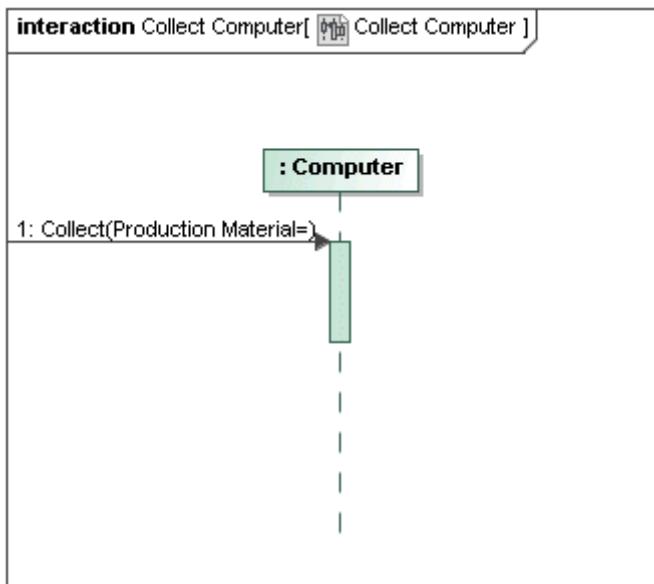


Figure 467 -- Sequence diagram with message with argument

Synchronization between Interaction Parameters and Lifelines

Synchronization between Interaction Parameters and Lifelines works in the following way: after you have created the sequence diagram in interaction with parameters, lifelines for the chosen parameters will be created in the sequence diagram.

This is not the same synchronization as in other cases, because this gives automated lifelines creation from parameters only.

How synchronization works

After you have created, edited, or removed parameters, arguments will be unchanged. Conversely, after you have created, edited, or removed arguments - parameters will be unaffected.

Displaying parameters as lifelines in already existing sequence diagram

You can also display parameters as lifelines in already existing sequence diagram:

1. From the sequence diagram shortcut menu, select **Related Elements** and then select **Display Parameters as Lifelines**. The **Display Parameters as Lifelines** dialog box appears.
2. Select parameters and click **OK**.
3. Lifelines for the selected parameters are created.

11 MODEL ELEMENTS

Parameter

Sample

1. Create an interaction with a parameter.

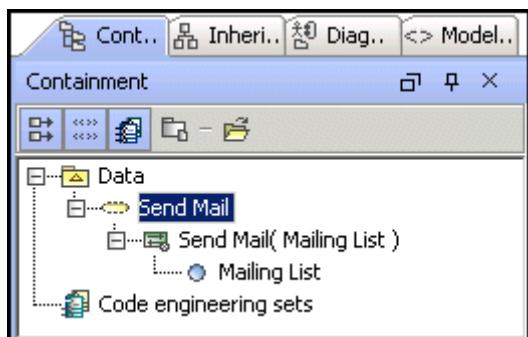


Figure 468 -- Project before synchronization

2. In the Interaction, create the sequence diagram. The **Display Parameters as Lifelines** dialog box appears.

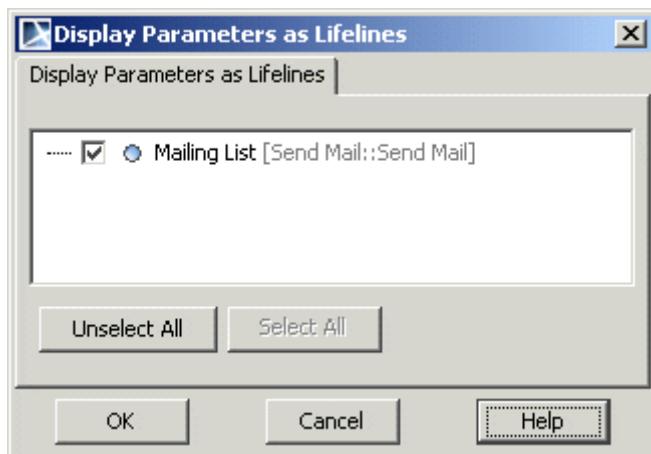


Figure 469 -- The Display Parameters as Lifelines dialog box

3. Select parameters, which will be created as lifelines in the sequence diagram. Click OK.
4. Lifelines are created in the interaction, which is drawn on the sequence diagram.

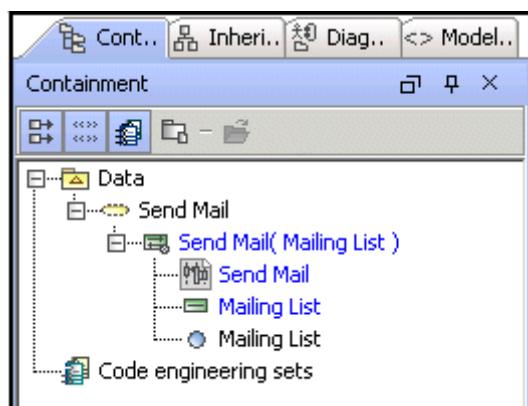


Figure 470 -- Project after synchronization

The Parameters Synchronization dialog box

The **Parameters Synchronization** dialog box (see Figure 471 on page 748) is shown below. It provides useful Parameters and Arguments synchronization options: not synchronized notification, automatic synchronization restoration algorithms, and manual synchronization abilities.

11 MODEL ELEMENTS

Parameter

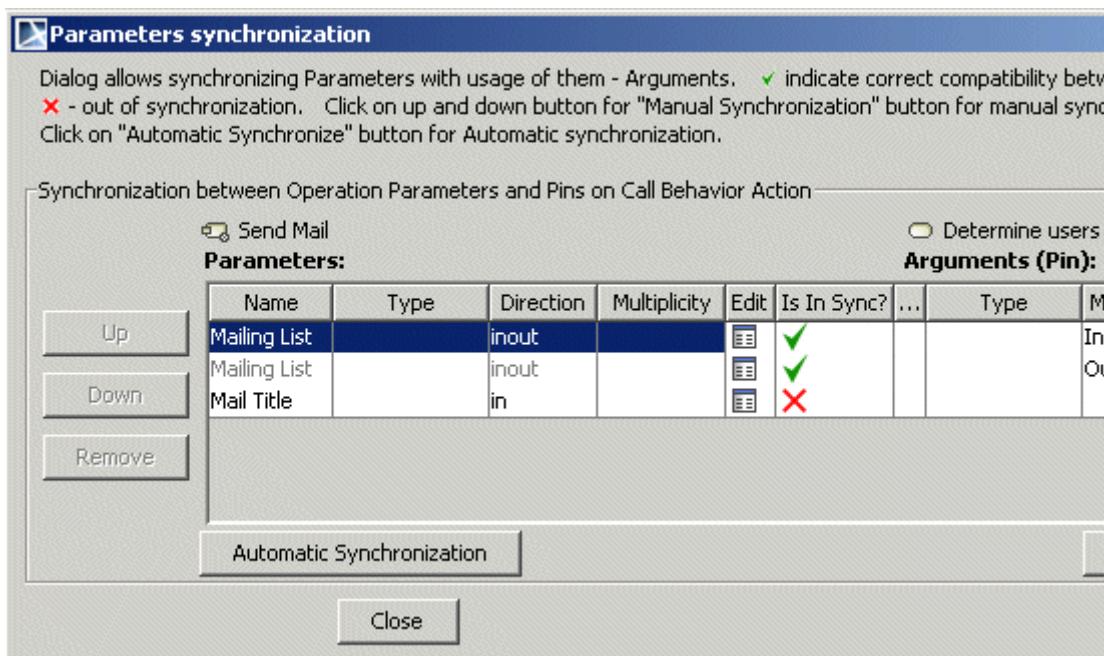


Figure 471 -- The Parameters Synchronization dialog box

There are two ways to open the **Parameter Synchronization** dialog box:

- Select the not synchronized element in the Browser. From the shortcut menu, select the **Validation > validation group > Parameter Synchronization** dialog opens.
- Select the invalid element on the diagram. In the symbol smart manipulator, click the invalid element indicator. From the menu that open, select the **Parameter Synchronization dialog** (see Figure 472 on page 748).

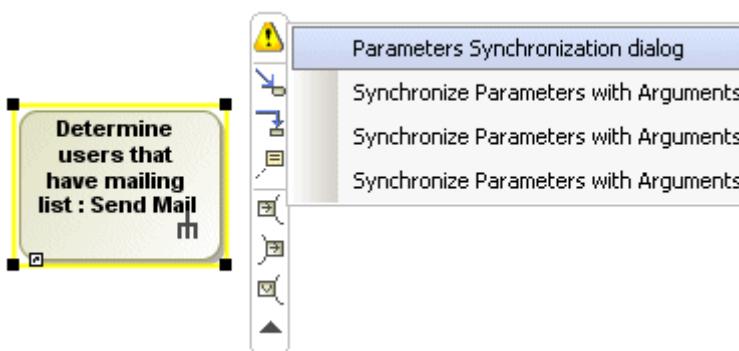


Figure 472 -- Smart Manipulator of the Invalid element

The parameters synchronization dialog box presents the elements those parameters and arguments are not synchronized and provides the possibility to restore the synchronization.

In the **Parameters Synchronization** dialog box, the parameters with arguments are synchronized according to the general synchronization rules, which are described in "Rules of synchronization between parameters and arguments" on page 737.

The **Parameters** group presents information about particular element parameters (see Figure 473 on page 749) and the **Arguments** group presents information about particular element arguments (see Figure 474 on page 749). To edit the parameter or argument, click on the icon in the **Edit** column (see Figure 475 on page 750). The **Is In Sync?** column displays if parameter and argument are synchronized (see Figure 476 on

11 MODEL ELEMENTS

Parameter

page 750). Green tick indicates that parameter and argument are synchronized. Red cross indicates that parameter and argument are not synchronized.

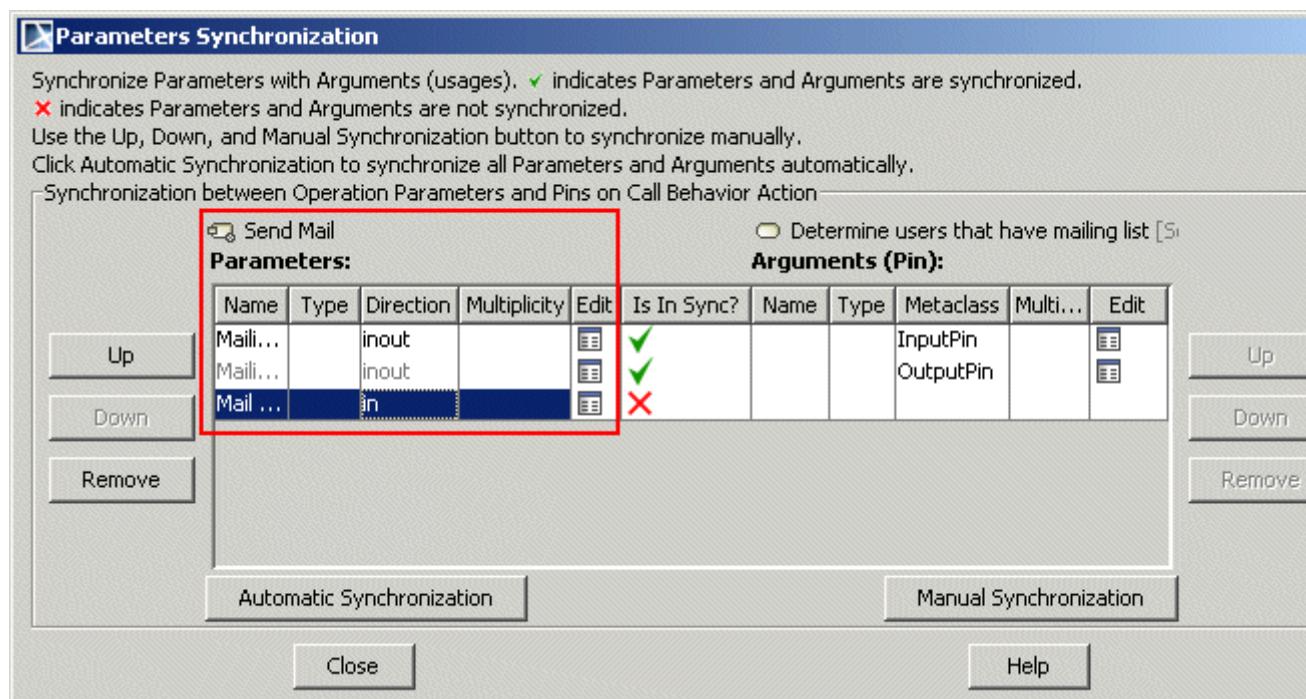


Figure 473 -- The Parameters Synchronization dialog box, Parameters group

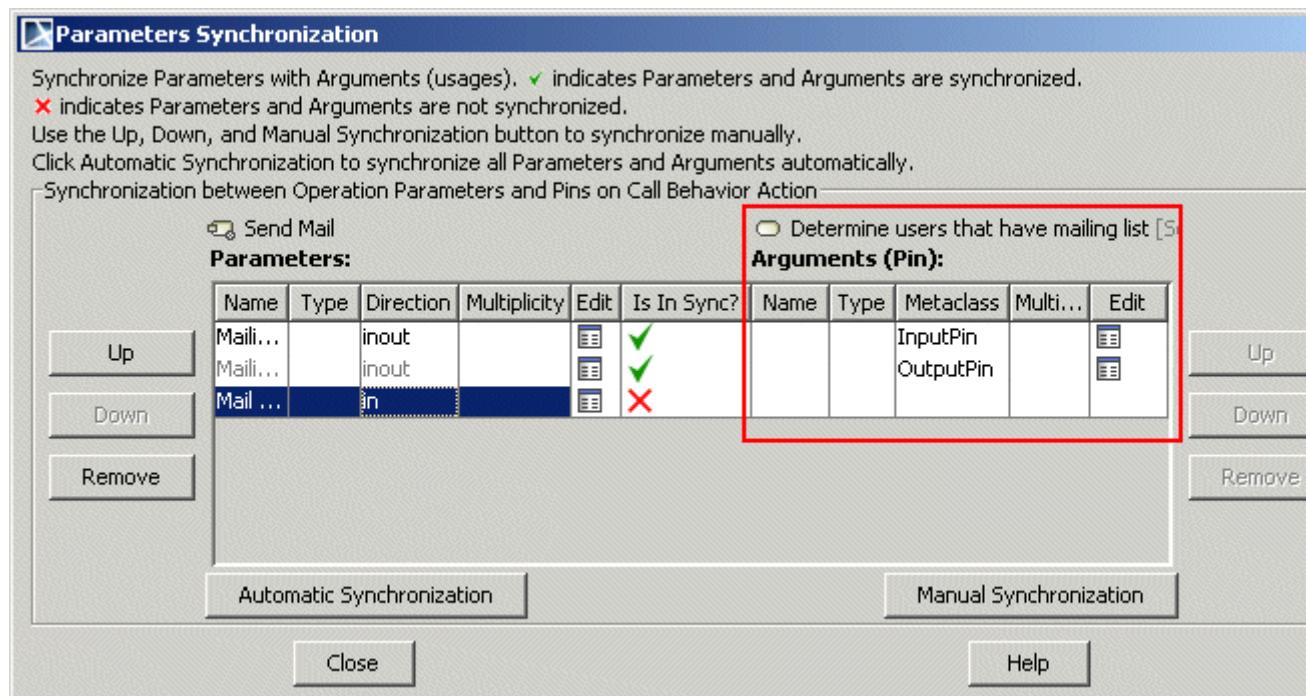


Figure 474 -- The Parameters Synchronization dialog box, Arguments group

11 MODEL ELEMENTS

Parameter

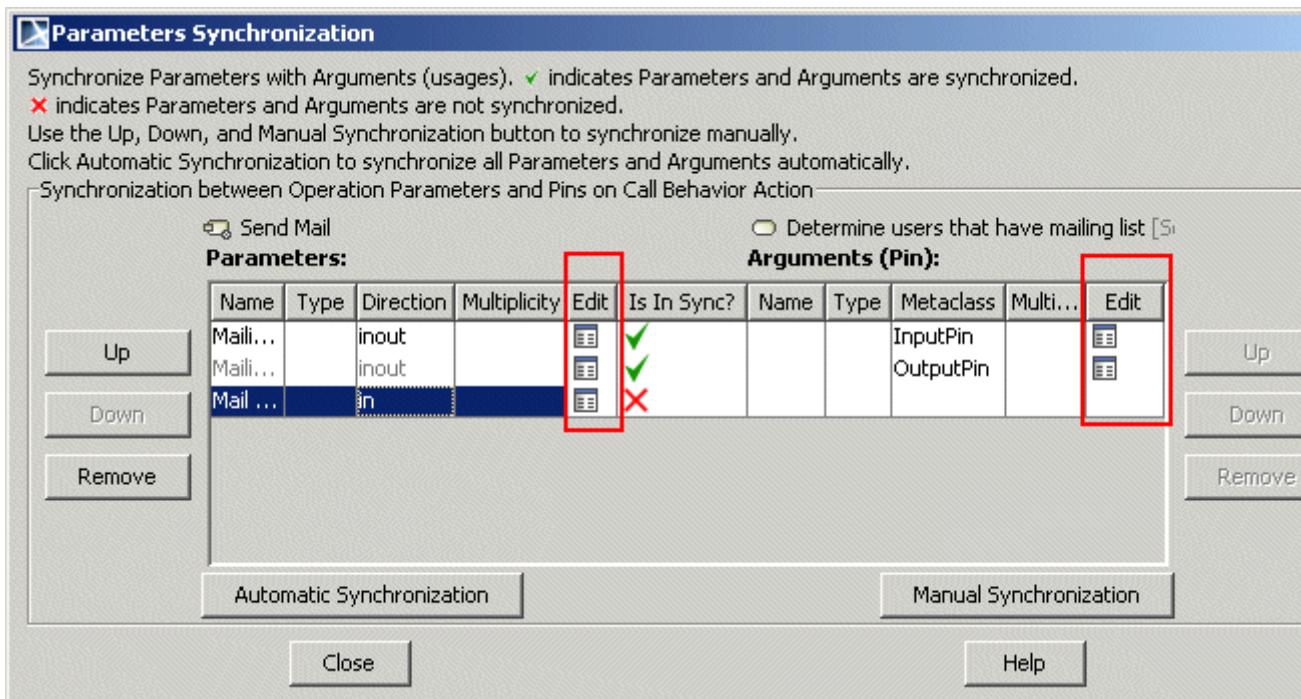


Figure 475 -- The Parameters Synchronization dialog box, Edit column

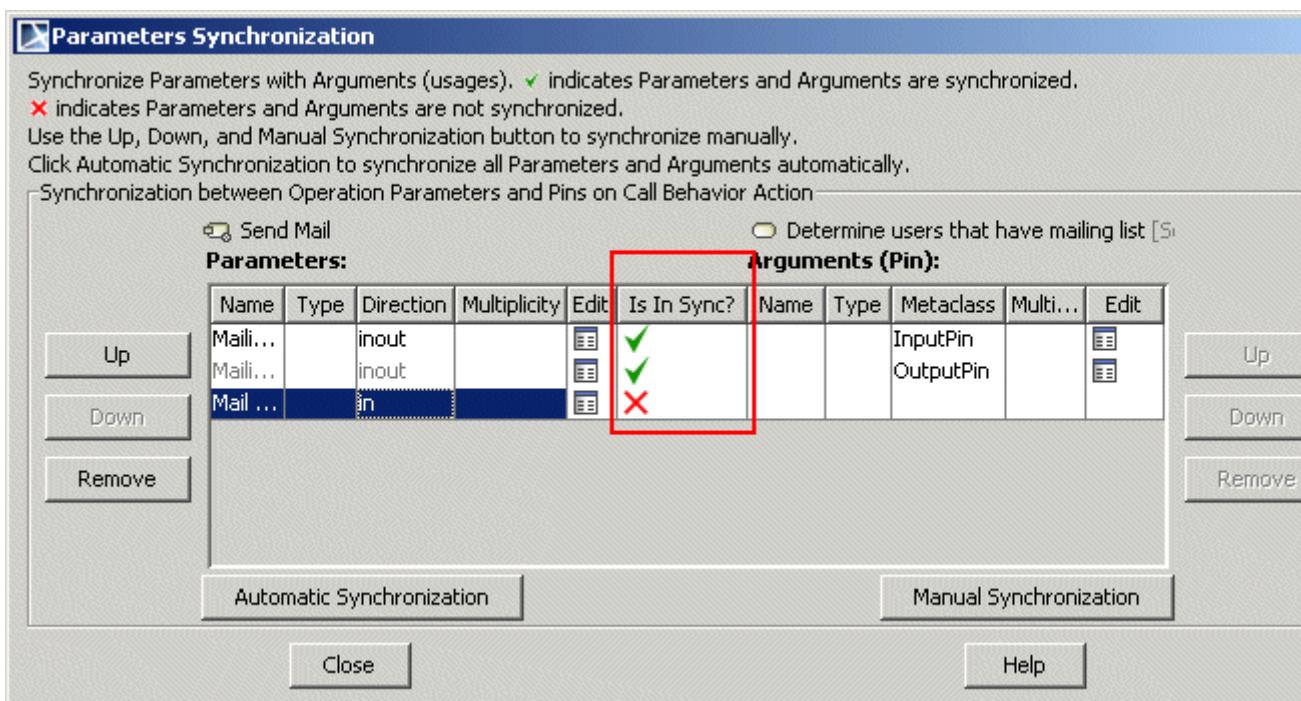


Figure 476 -- The Parameters Synchronization dialog box, Is In Sync? column

Automatic Synchronization

To synchronize parameters with arguments automatically, in the **Parameters Synchronization** dialog box, click the **Automatic Synchronization** button (see Figure 477 on page 751). According to the available synchronization the following available commands appears:

1. Synchronize Parameters with Arguments by restoring initial order and creating missing ones.

11 MODEL ELEMENTS

Parameter

- Missing arguments will be created.
 - Not synchronized Arguments order will be changed and not synchronized properties will be changed according reference.
2. Synchronize Parameters with Arguments by reordering and creating missing ones.
- Not synchronized arguments order will be changed to fit properties.
 - Missing arguments will be created automatically.
 - If reference between parameters and arguments existed, but they are not synchronized or reference will be removed.
3. Synchronize Parameters with Arguments by updating and creating missing ones.
- Not synchronized arguments not synchronized properties according to parameter will be changed and missing arguments will be created.
4. Synchronize Parameters with Arguments by creating missing ones.
- Missing arguments will be created.

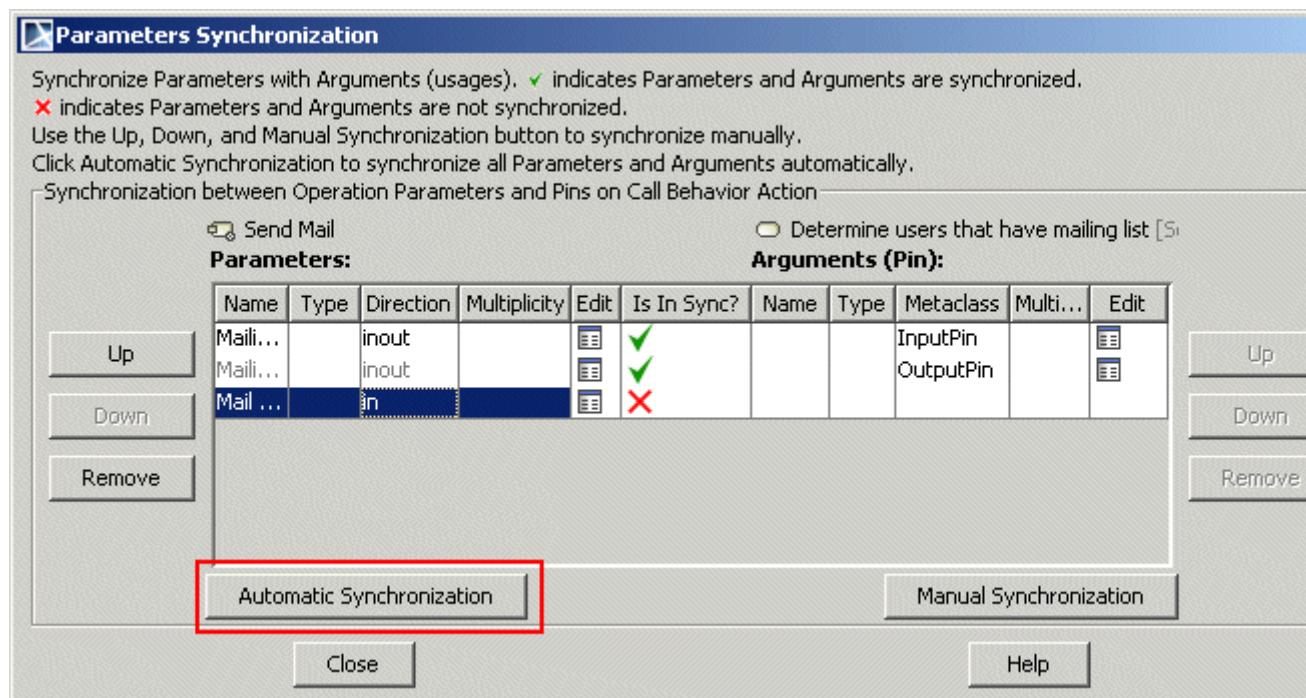


Figure 477 -- The Parameters Synchronization dialog box, Automatic Synchronization

Manual Synchronization

You can manually synchronize parameters with arguments using the **Up/Down/Remove** buttons or click the **Manual Synchronization** button (see Figure 478 on page 752).

Click the **Up/Down/Remove** buttons to move up/down or remove parameters or arguments.

In the **Parameters Synchronization** dialog box table select not synchronized parameter/ argument and click the **Manual Synchronization** button to synchronize manually. See the description of the manual synchronization commands in the table below.

| Command | Description |
|---------|--|
| <-Clone | Select the <-Clone command to create parameter with the same properties. |
| Clone-> | Select the Clone-> command to create parameter with the same properties. |

11 MODEL ELEMENTS

Port

| Command | Description |
|----------|---|
| <-Update | Select the <-Update command to update parameter not synchronized properties in order to be synchronized. The Update item is available if argument exist and there are properties to update. |
| Update-> | Select the <-Update command to update argument not synchronized properties in order to be synchronized. The Update item is available if parameter exist and there are properties to update. |

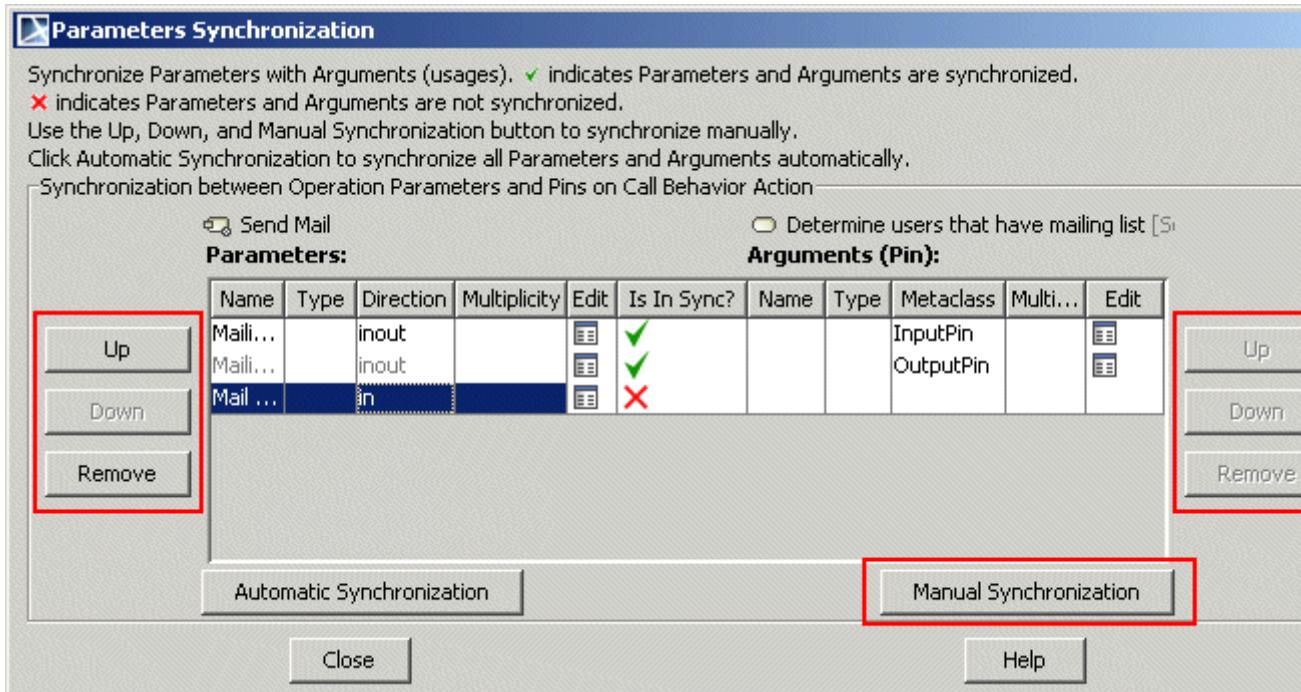


Figure 478 -- The Parameters Synchronization dialog box, Manual Synchronization

Port

A port is a property of a classifier that specifies a distinct interaction point between that classifier and its environment, or between the (behavior of the) classifier and its internal parts. Ports are connected to the properties of the classifier by connectors through which requests can be made to invoke the behavioral features of the classifier.

A Port may specify the services a classifier provides (offers) to its environment as well as the services that a classifier expects (requires) from its environment. It has the ability to specify that any requests arriving at this port are handled.

The Class model element and Component model elements may have any number of Ports.

11 MODEL ELEMENTS

Port

Define a port in the **Port Specification** dialog box.

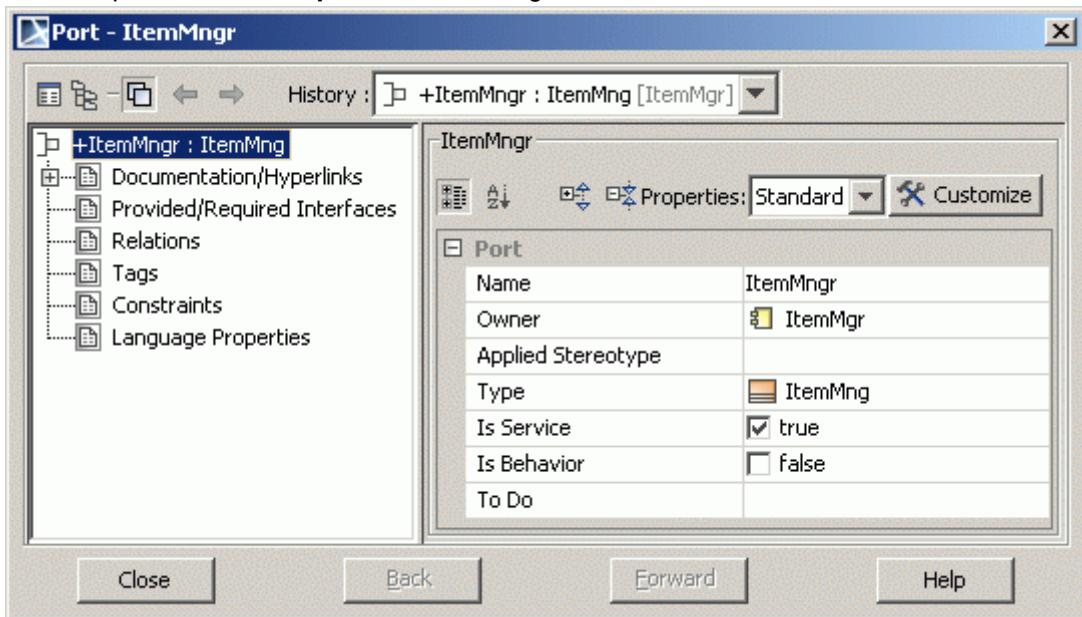


Figure 479 -- Port Specification dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box | Function |
|--|-------------|--|
| General Set a general information for the port. | Owner | The name of a class or component, which contains the port. |
| | Is Behavior | Specifies whether requests arriving at this port are sent to the classifier behavior of this classifier. Such a port are referred to as a behavior port. Any invocation of a behavioral feature targeted at a behavior port will be handled by the instance of the owning classifier itself, rather than by any instances this classifier may contain. |
| | Is Service | If the value of the Is Service check box is true, it indicates that this port is used to provide the published functionality of a classifier. If the value of the Is Service check box is false, it indicates that this port is used to implement the classifier but is not part of the essential externally. |
| Provided/ Required Interfaces | Name | Shows the name of the provided/required interface. |
| | Type | Shows the type of the interface that specifies a port. |
| | Add | Drop down menu opens. Select the <i>Provided</i> or <i>Required</i> item and select an interface in the open Select Interface dialog box. |
| | Remove | Removes an item from the Provided/Required Interfaces list. |

To customize ports list

The ports list can be displayed in a separate compartment on the element shape when the **Suppress Ports** check box is cleared from the element shortcut menu, **Presentation Options** submenu.

1. From the element shortcut menu, select **Edit Compartment** and then **Ports**.

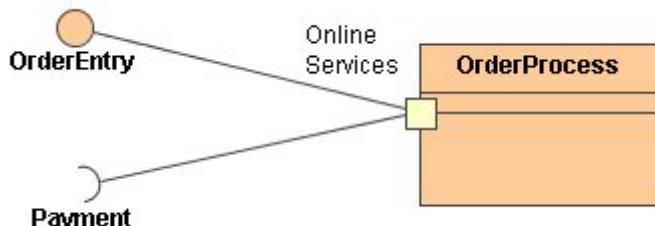
11 MODEL ELEMENTS

Port

2. In the open **Compartment Edit** dialog box, **Ports** tab, add the desired ports to display from **All** to the **Selected** list.

To draw a Port

In the Implementation diagram toolbar, select a Port element to draw on a class shape.



Draw a Realize path from a port to an interface to depict the Provided Interface.

Draw Usage path from port to interface to depict Required Interface.

NOTE

When a Part symbol is created (also when dropping classifiers on composite structure diagrams), all its ports are displayed on the diagram.

To specify the Provided/Required Interfaces for a Port even if the Port type is not specified

When you add a Provided or Required Interface to a Port, the **Select Port Type** dialog will open (Figure 480 on page 754) with the following options:

- **Set Provided Interface as Port Type** (available on Provided Interface creation only). The Provided Interface will be suggested as the Port Type.
- **Create "dummy" port type automatically**. Create a dummy port type and relations between the type and interface.
- **Select or create a port type manually**. The **Select Port Type** dialog will open to allow you to select or create a Port.

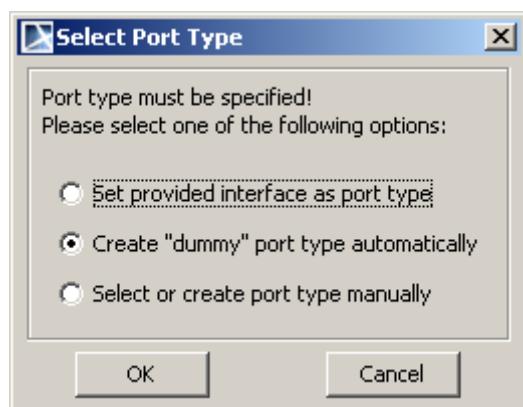


Figure 480 -- The Select Port Type Dialog

Pseudo State

The Pseudo state is typically used to connect multiple transitions into more complex state transitions paths. For example, by combining a transition entering a fork pseudo state with a set of transitions exiting the fork pseudo state, we get a compound transition that leads to a set of orthogonal target states.

For more information about working with symbols, see "Diagramming"

Specify the pseudostates in the **Pseudo State Specification** dialog box.

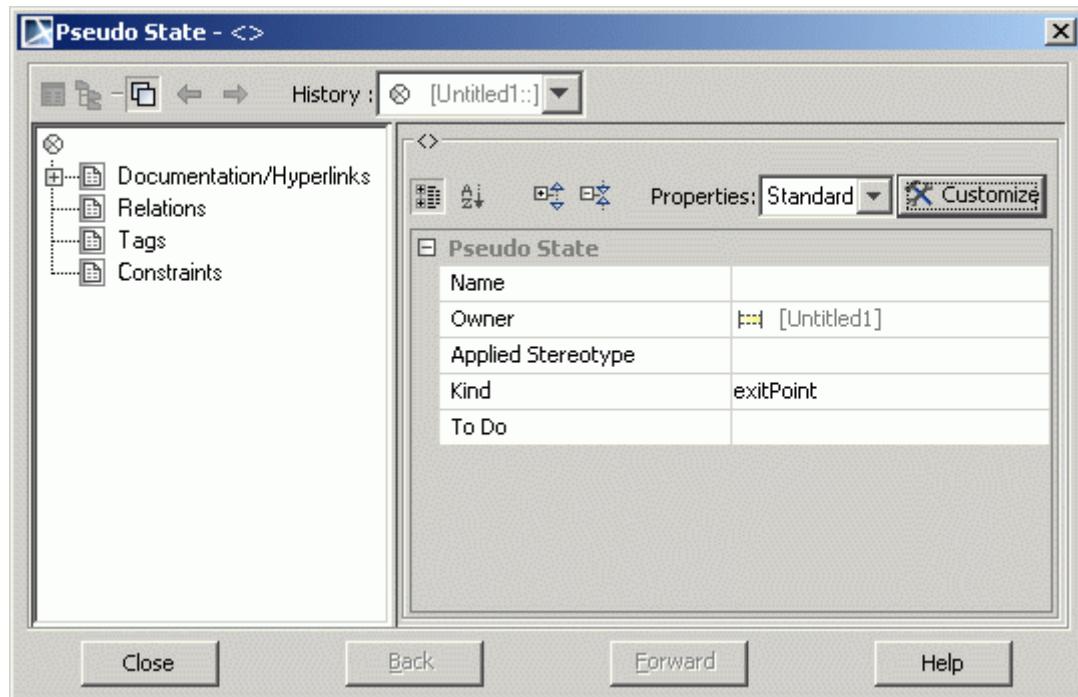


Figure 481 -- Pseudo State Specification dialog box

| Tab name | Property | Function |
|----------------|--------------|--|
| General | Owner | The region, containing a pseudo state. |
| | Kind | The kind, showing the purpose of the pseudo state. |

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

Initial

Every object belongs to a particular state as soon as it is created. So, it is useful to explicitly show that particular state. A solid filled circle represents the initial state of an object. There can only be one initial state for an object. The initial state denotes the starting place for a transition, the target of which is a composite state.

Final state

The final state symbol (a circle surrounding a smaller solid circle) is used to represent the object destruction. The final state is optional in the diagram because there is a system that runs without interruption after the start of the activities. Also, there can be several final states in the same state diagram, denoting that the life of the object may finish depending on several conditions.

Terminate

Entering a terminate pseudo state implies that the execution of the state machine by means of its context object is terminated. The state machine does not exit any states nor does it perform any exit actions other than those associated with the transition leading to the terminate pseudo state.

Entry Point

An entry point connection point reference as the target of a transition implies that the target of the transition is the entry point pseudo state as defined in the submachine of the submachine state. As a result, the regions of the submachine state machine are entered at the corresponding entry point pseudo states.

Exit Point

An exit point connection point reference as the source of a transition implies that the source of the transition is the exit point pseudo state as defined in the submachine of the submachine state that has the exit point connection point defined. When a region of the submachine state machine has reached the corresponding exit points, the submachine state exits at this exit point.

Deep History

The Deep History represents the most recent active configuration of the composite state that directly contains this pseudo state; e.g. the state configuration that was active when the composite state was last exited. A composite state can have at most one deep history vertex.

Shallow History

The Shallow History represents the most recent active substate of its containing state (but not the substates of that substate). A composite state can have at most one shallow history vertex. A transition coming to the shallow history vertex is equivalent to a transition coming to the most recent active substate of a state.

Junction

The junction vertices are semantic-free vertices that are used to chain multiple transitions together. They are used to construct the compound transition paths between states. For example, a junction can be used to combine multiple incoming transitions into a single outgoing transition representing a shared transition path (this is known as merge). Conversely, it can be used to split an incoming transition into multiple outgoing transition segments with different guard conditions.

Choice

The choice vertices, when reached, result in the dynamic evaluation of the guards or the triggers of its outgoing transitions. This realizes a dynamic conditional branch. It allows splitting of transitions into multiple outgoing paths such that the decision on which path to take may be a function of the results of prior actions performed in the same run-to-completion step.

Fork and Join

The fork vertices are used to split an incoming transition into two or more transitions terminating on the orthogonal target vertices (i.e., vertices in different regions of composite state). The segments going out of a fork vertex must not have guards or triggers.

The join vertices are used to merge several transitions emanating from the source vertices in different orthogonal regions. The transitions entering a join vertex cannot have guards or triggers.

Realization

The realization is a specialized abstraction relationship between two sets of model elements, one represents a specification (the supplier) and the other represents an implementation of the latter (the client). The realization can be used to model stepwise refinement, optimizations, transformations, templates, model synthesis, framework composition, etc.

The realization relationship is drawn as a dashed line with a solid triangular arrowhead (a “dashed generalization symbol”). The client (the one at the tail of the arrow) supports at least all of the operations defined in the supplier (the one at the arrowhead), but not necessarily the data structure of the supplier (attributes and associations).

For more information about working with symbols, see "Diagramming"

The realization paths can be grouped in a tree. This feature makes the appearance of the diagram more structural and understandable.

NOTE

In MagicDraw, you will find three kinds of a realization relationship:

- **Interface Realization.** A dashed line with a solid triangular arrowhead. An Interface Realization is a specialized Realization relationship between a Classifier and an Interface. This relationship signifies that the realizing classifier conforms to the contract specified by the Interface.
- **Realization.** A solid line that represents a relationship between a classifier and an interface.
- **Substitution.** A dashed line with an arrowhead and <<substitute>> stereotype. A substitution is a relationship between two classifiers. It signifies that the substituting classifier complies with the contract specified by the contract classifier. This implies that instances of the substituting classifier are runtime substitutable where instances of the contract classifier are expected.

To create a realization tree if a class or an interface already has a number of realization paths attached to it

Select the **Make Sub Tree** command from the class or the interface shortcut menu.

To remove a realization from the tree or to ungroup a tree

- Select the realization and select the **Remove From Tree** command from the path shortcut menu.

11 MODEL ELEMENTS

Realization

- Select a tree head and select the **Ungroup Tree** command from the tree shortcut menu.

To specify the selected realization path in the **Specification** dialog box

- Double-click the path.
- Select **Specification** from the path shortcut menu.
- Select the path and press ENTER.

Realization and its kinds Specification dialog boxes

The realization, interface realization, and substitution relationships are defined in the dialog box of the same structure. They differ from one another only by the corresponding Specification name..

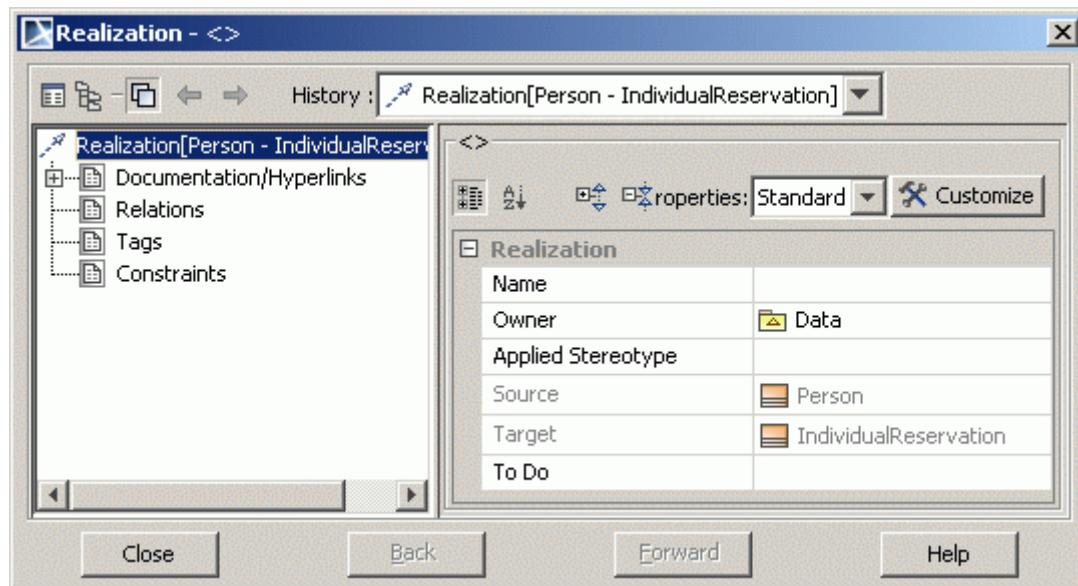


Figure 482 -- Realization Specification dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box | Function |
|---|---------|--|
| General Set general information about the realization relationship | Source | The name of the child element. |
| | Target | The name of the parent element. |
| | Mapping | Type a value, or click the ‘...’ button and edit the value in the Edit Mapping dialog box. Note: Available in the Substitution Specification dialog box only. |

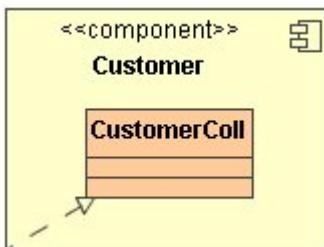
Creating the realizing classifiers

The realizing classifiers are a set of Realizations owned by the Component. The Realizations reference the Classifiers of which the Component is an abstraction (i.e., that realize its behavior).

To create a Realization relationship between a component and a classifier:

1. Drag the classifier shape to the component shape.

2. Select a classifier or component and select **Related Elements** from its shortcut menu, then select the **Display Paths** command. The realization relationship will be displayed on the diagram pane.



Reception

Term

The OMG UML specification (UML 2.2: Superstructure) states:

“A **reception** is a declaration stating that a classifier is prepared to react to the receipt of a signal. A reception designates a signal and specifies the expected behavioral response. The details of handling a signal are specified by the behavior associated with the reception or the classifier itself.”

Signal receptions can be specified for classes or interfaces.

Parent topic: “Model Elements” on page 620.

Related topics:

“Specification Window” on page 237.

“Formatting Symbols” on page 255.

“Class” on page 663.

“Interface” on page 706.

To create a new reception

- Double-click the selected class or select **Specification** from the class shortcut menu. The **Class** specification dialog box opens. Click the **Signal Receptions** tab and then click the **Create** button. The **Select Signal** dialog box opens. Select a signal or create a new one. Click **OK**. The **Signal Reception** specification dialog box opens. Specify a new reception and click **OK**.
- Select a class in the Browser tree. From the class item shortcut menu, select **New** and then select **Signal Reception**.
- Select a class shape and click the small red **Insert New Signal Reception** smart manipulation button.

NOTE The signal reception compartment is suppressed and the smart manipulator button is not visible by default.

11 MODEL ELEMENTS

Send Signal Action

- Drag-and-drop the signal symbol on the class or interface shape in the diagram pane. The signal reception with the assigned signal is created. You may also drag a signal from the Browser to the class or interface shape on the diagram pane.

To open the **Signal Reception** specification dialog box

1. Open the **Class** specification, or the **Interface** specification dialog box.
 2. In the **Signal Receptions** tab, double-click the desired signal reception in the tree, or click the **Create** button.
OR
- Double-click the desired signal reception directly on the diagram.

Refer to “Specification Window” on page 237 for information about the specification elements not covered in this section.

To set the signal for a signal reception

- Create a signal reception. The **Select Signal** dialog box opens. Select an existing or create a new signal for the signal reception.
- In the **Signal Reception** specification dialog box, in the **Signal** drop down menu, select a signal. You may also click the “...” button. In the **Select Element** dialog box, select a signal or click the **Create** button to create a new one.

To display the signal reception on the diagram pane

The signal reception compartment is added to the class and interface shape. This compartment is hidden by default. To show the signal reception compartment:

- From the symbol shortcut menu, select the **Presentation Options** command, then select the **Suppress Signal Receptions** command.
- From the symbol shortcut menu, select the **Symbol(s) Properties** command. In the element **Properties** dialog box, clear the **Suppress Signal Receptions** command.

To show/hide the signal receptions in the Signal Reception compartment

Select the **Edit Compartment** command from the symbol shortcut menu. Select the **Signal Reception command**. The **Edit Compartment** dialog box opens. Only those signal receptions will be displayed in the Signal Reception compartment that are displayed in the **Selected** list.

To change the order of the signal receptions

- From the symbol shortcut menu, select the **Presentation Options** command, then select the **Signal Receptions Sort Order** command.
- From the symbol shortcut menu, select the **Symbol(s) Properties** command. In the element **Properties** dialog box, clear the **Suppress Signal Receptions** check box.

Send Signal Action

The Send Signal Action is an action that creates a signal instance from its inputs and transmits it to the target object, where it may cause the start of the state machine transition or the execution of an activity. The argument values are available to the execution of associated behaviors. The requester continues the execution immediately. Any reply message is ignored and is not transmitted to the requester.

11 MODEL ELEMENTS

State

For more information about working with symbols, see "Diagramming"

Define the selected send signal action in the **Send Signal Action Specification** dialog box.

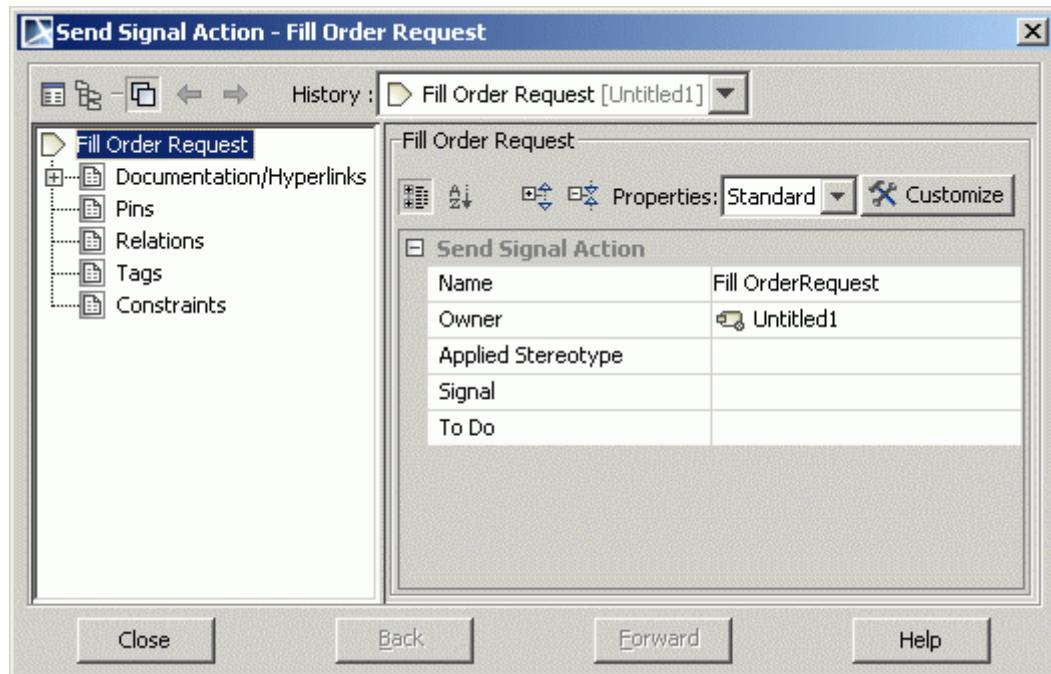


Figure 483 -- Send Signal Action Specification dialog box

Refer to the "Specification Window" on page 237 for information about the specification elements not covered in this section.

| Tab name | Box name | Function |
|---|----------|--|
| General Set a general information about the object node. | Signal | Click the "..." button to assign an existing signal from the model in the Select Elements dialog box, or click Create to assign a new one. |

State

A state is a condition during the lifetime of an object or an interaction during which the object meets certain conditions, performs an action, or waits for an event. The state is defined by the concepts of duration and stability. An object may not be in an unknown or undefined state. A state may have two compartments to provide more information about that state:

- The first compartment is the name compartment, it contains the state name, for example: running, going up.
- The second compartment is the activity compartment, it contains the events and actions of the state.

For more information about working with symbols, see "Diagramming"

11 MODEL ELEMENTS

State

MagicDraw supports four types of states: State, Composite State, Orthogonal State, and Submachine State. Define the selected state in the **Composite State Specification** dialog box.

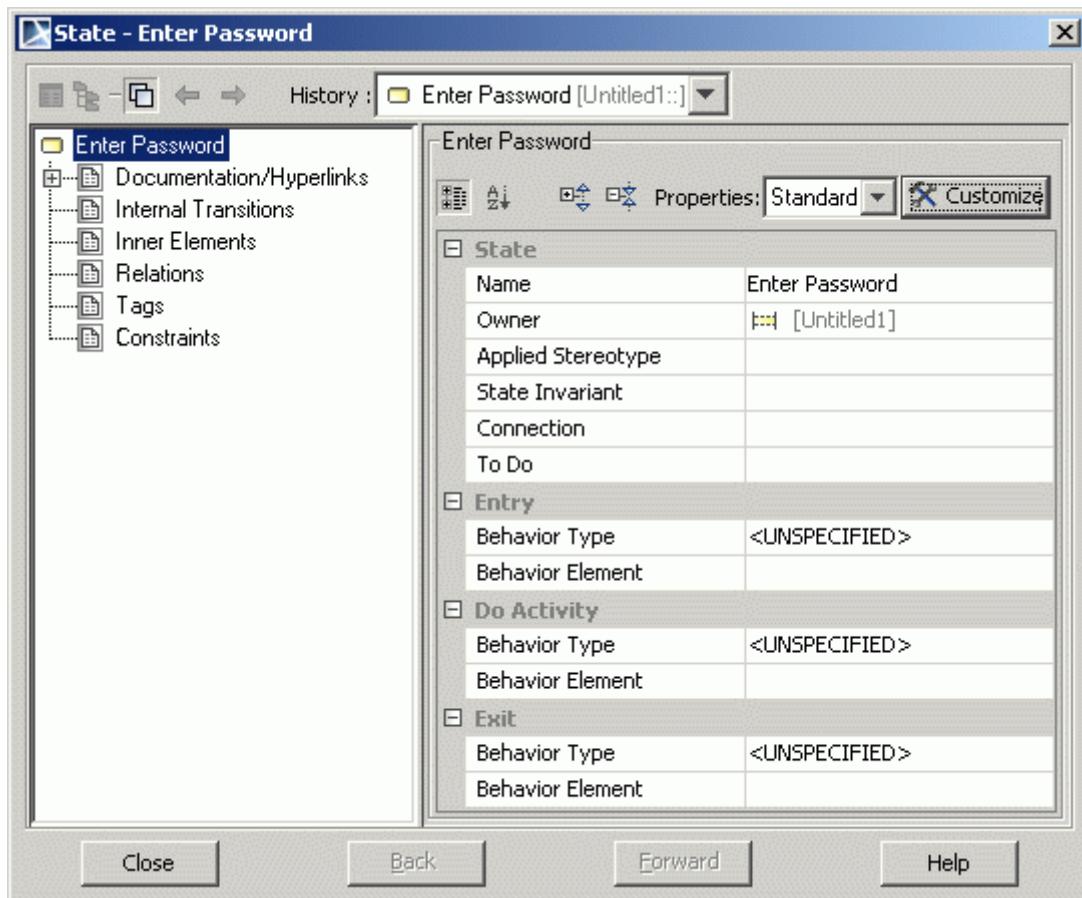


Figure 484 -- State Specification dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box name | Function |
|--|-------------------------|--|
| General Contains the state actions, events and buttons for editing the list. | State Invariant | Type text or click ‘...’ to open the Edit State Invariant dialog box. |
| | Connection | Click “+” to open the Connection Point Reference dialog box. Specify the information for the connection. |
| Actions group box | | |
| Entry | | |
| | Behavior Type | Select a type from the activity, interaction, or state machine items list. |
| | Behavior Element | Click ‘...’ and select an element from the activity, interaction, or state machine items list. The corresponding dialog box opens. |
| Do Activity | | |
| | Behavior Type | Select a type from the activity, interaction, or state machine items list. |
| | Behavior Element | Click ‘...’ and select an element from the activity, interaction, or state machine items list. The corresponding dialog box opens. |
| Exit | | |

11 MODEL ELEMENTS

State

| Tab name | Box name | Function |
|---|-------------------------|--|
| | Behavior Type | Select a type from the activity, interaction, or state machine items list. |
| | Behavior Element | Click ‘...’ and select an element from the activity, interaction, or state machine items list. The corresponding dialog box opens. |
| Inner States The list of inner states and buttons for editing them | Name | The name of the inner state. |
| | Type | The type of the inner state. |
| | Region | The region, to which an inner state was added. |
| Internal Transitions A set of transitions that, if triggered, occurs without exiting or entering the state. | Name | The name of the internal transition. |
| | Create | The Transition Specification dialog box opens. Define the transition. |
| | Delete | Remove the selected transition from the state. |

To suppress/unsuppress the actions compartment

In the state shortcut menu, select/clear the **Suppress Actions** check box.

To insert a new region to the state

- In the state shortcut menu, select **Insert New Region**.

To insert a new inner state to the state

- In the Browser, drag and drop the selected state to a Region.
- On the diagram pane, select the state and drag and drop it on the state symbol.

To display region name on the state symbol

Region name can be optionally displayed on the State symbol on a diagram.

To display or hide region name

- On a diagram from the State symbol shortcut menu check or clear the **Show Region Name** check box.

11 MODEL ELEMENTS

State

- In the State **Properties** dialog box select or clear the **Show Region Name** check box.

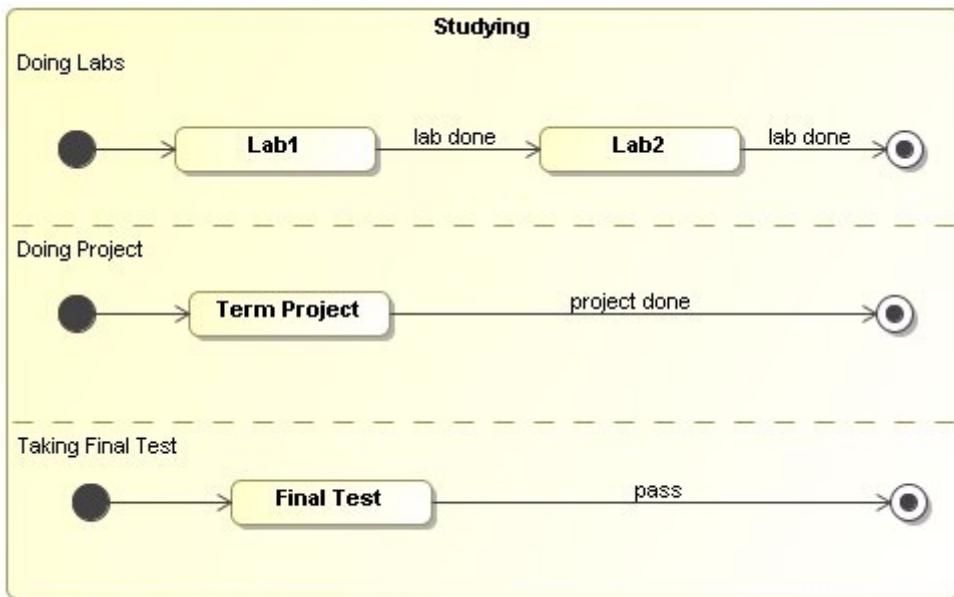


Figure 485 -- Displaying region name on the State symbol

Changing State to Composite/submachine/orthogonal State

You can change your current state to a simple state, composite state, orthogonal state, and submachine state.

To change the state to the composite state

- From the State diagram toolbar, select the State to draw on the diagram pane.
- From the state shortcut menu, select the **Add New Region** command. The region is added and the state shape enlarges.

To change the state to the orthogonal state

- From the State diagram toolbar, select the State to draw on the diagram pane.
- From the state shortcut menu, select the **Add New Region** command. The region is added and the state shape enlarges.
- Open the state shortcut menu again and add the second region to the state.

To change the state to the submachine state

- From the State diagram toolbar, select the State to draw on the diagram pane.
- From the state shortcut menu, select the **Submachine** command and select an existing State Machine from the list, or click **New** to create a new one.

NOTE : To change the state to the submachine state, all regions from the state should be removed. To do this, select the state on the diagram pane and from the shortcut menu, select the **Remove Region** command.

Composite State

A composite state either contains one region or is decomposed into two or more orthogonal regions. Each region has a set of mutually exclusive disjoint subvertices and a set of transitions. A given state may only be decomposed in one of these two ways.

Any state enclosed within a region of the composite state is called a substate of that composite state. It is called a direct substate when it is not contained by any other state; otherwise it is referred to as an indirect substate.

Each region of the composite state may have an initial pseudostate and a final state. A transition to the enclosing state represents a transition to the initial pseudostate in each region.

For more information about working with symbols, see "Diagramming"

Define the selected composite state in the **State Specification** dialog box. For a detailed description of this dialog box, see "State Specification dialog box" on page 762.

For more information on working with states, see "State" on page 761.

To add a new region to the composite state

Select **Add New Region** from the composite state shortcut menu.

To remove a region from the composite state (at least two regions have to be left)

Select **Remove Region** from the composite state shortcut menu and select a region you want to remove.

Submachine

A submachine state specifies the insertion of the specification of a submachine state machine. The state machine that contains the submachine state is called the containing state machine. The same state machine may be a submachine more than once in the context of a single containing state machine.

The submachine state is semantically equivalent to a composite state. The regions of the submachine state machine are the regions of the composite state. The entry, exit, behavior actions, and internal transitions, are defined as part of the state. The submachine state is a decomposition mechanism that allows factoring of the common behaviors and their reuse.

For more information about working with symbols, see "Diagramming"

Define the selected submachine in the **State Specification** dialog box. For a detailed description of this dialog box, see "State Specification dialog box" on page 762.

For more information on working with states, see "State" on page 761.

Adding connection point reference

The connection point reference represent an entry to or exit from the submachine state. It can be used as the source or target of a transition.

To draw the connection point reference on the submachine state

-
1. In the state diagram toolbar, click the **Connection Point Reference** button. Click on the diagram pane on the submachine state shape. The **Select Entry/Exit Point** dialog box opens.
 2. Select *entry point* to define the entry into the submachine state or *exit point* to define the exit from the submachine state. The Connection Point Reference is drawn on the submachine state with a defined entry or exit point.

To see the assigned entry/exit point, open the **Connection Point Reference** dialog box. The **Entry** or **Exit** properties will display the defined entries.

11 MODEL ELEMENTS

Subsystem

To assign the entry/exit points to the Connection Point Reference

Select the Connection Point Reference on the diagram pane. Open the shortcut menu and select the **Select Entry/Exit Point** command.

NOTE: In the **Select Entry/Exit Point** dialog box, only these entry and exit points are listed, which are created at the same State Machine as the submachine state. If there are no entry/exit points at the same state machine, the **Select Entry/Exit Point** dialog box is not opened when drawing the Connection Point Reference.

Defining State Invariant

To define a state condition

- Double-click on the state to open the **State Specification** dialog box and in the **State Invariant** field, type an condition and submit changes.
- Near the **State Invariant** field, click "..." button. The **Edit State Invariant** dialog box opens. Type the condition and close the dialog box by submitting changes.

The State Invariant value is displayed on the diagram pane on the state shape in brackets:

Typing password

[State Invariant expression]

Assigning behavior to state

You may define a behavior to be executed correspondingly to the listed events while being in the state whenever the state is entered and exited.

To assign a behavior to a state

1. Double-click the state to open the **State Specification** dialog box.

In the Do Activity, Entry or Exit group, the Behavior Type combo box, assign one of the behaviors: Activity, Interaction, or State Machine.

Subsystem

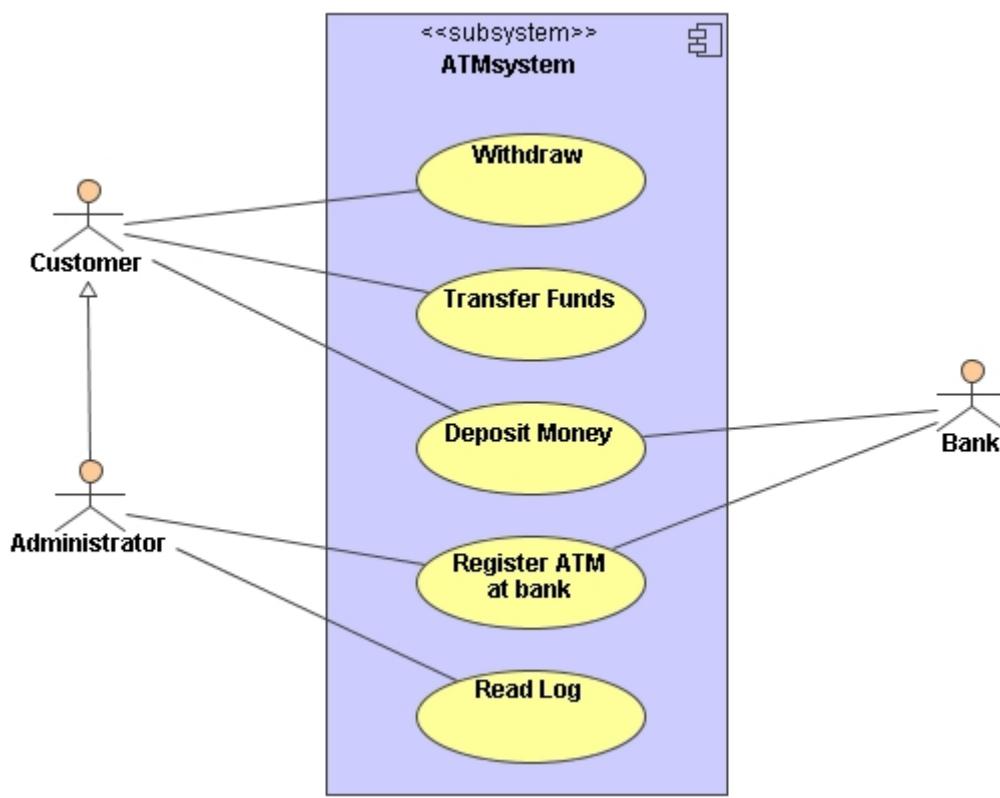
A subsystem is treated as an abstract single unit. It groups model elements by representing the behavioral unit in a physical system.

11 MODEL ELEMENTS

Swimlane

To draw a subsystem

1. In the Use Case diagram toolbar, select the Subsystem element to draw.



Swimlane

Actions and subactivities can be organized into swimlanes in the activity diagrams. The swimlanes are used to organize responsibility for actions and subactivities according to the class. They often correspond to the organizational units in a business model.

The swimlanes limit and provide a view on the behaviors invoked in the activities. They consist of one or more partitions. They may be vertical and horizontal.

An activity diagram can be divided visually into “swimlanes,” each separated from the neighboring swimlanes by vertical or horizontal solid lines on both sides. Each swimlane represents a responsibility for part of the overall activity, and may eventually be implemented by one or more objects. The relative ordering of the swimlanes has no semantic significance, but may indicate some affinity. Each action is assigned to one swimlane. Transitions may cross lanes. There is no significance to the routing of a transition path.

11 MODEL ELEMENTS

Swimlane

To open the **Activity Partition Specification** dialog box

Select **Specification** from the partition (swimlane) shortcut menu, or double-click the partition line.

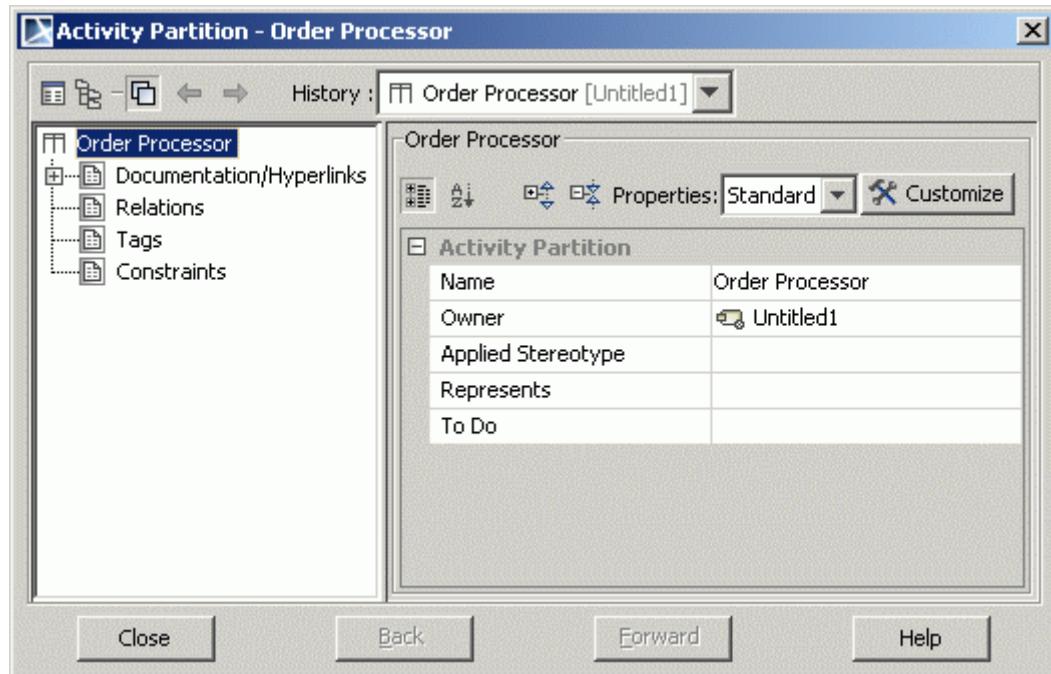


Figure 486 -- Activity Partition Specification dialog box

Refer to the “Specification Window” on page 237 for information about the specification elements not covered in this section.

| Tab name | Box name | Function |
|--|------------|---|
| General Set general information about the partition | Owner | Shows the name of the activity diagram where the partition is placed. |
| | Represents | Click the “...” button to open the Select Elements dialog box. Assign a model element that represents the partition. |

To draw a swimlane on the diagram

Click the **Swimlane** button on the diagram toolbar and click the diagram.

To set a name for a swimlane

1. Double-click the swimlane line or select **Specification** from the line shortcut menu.
2. The **Activity Partition Specification** dialog box opens. Type a name in the **Name** box.
 - Type the name directly on the Diagram pane.

11 MODEL ELEMENTS

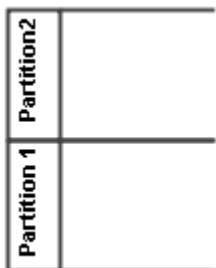
Swimlane

The Horizontal and Vertical Swimlanes

The following is an example of vertical swimlanes:

| Partition1 | Partition2 | Partition3 |
|------------|------------|------------|
| | | |

The following is an example of horizontal swimlanes:



To add an additional partition

Select a swimlane on the diagram pane and from the shortcut menu, select **Insert Vertical Swimlane** or **Insert Horizontal Swimlane**.

To draw multidimensional swimlanes

1. Draw a vertical swimlane.
2. From the swimlane shortcut menu, select the **Insert Horizontal Swimlane** command.
3. Insert as many horizontal and vertical swimlanes as you need.

The following is an example of the multidimensional swimlanes:

| | Seattle | Reno |
|------------------|---------|------|
| Order Processor | | |
| Accounting Clerk | | |

11 MODEL ELEMENTS

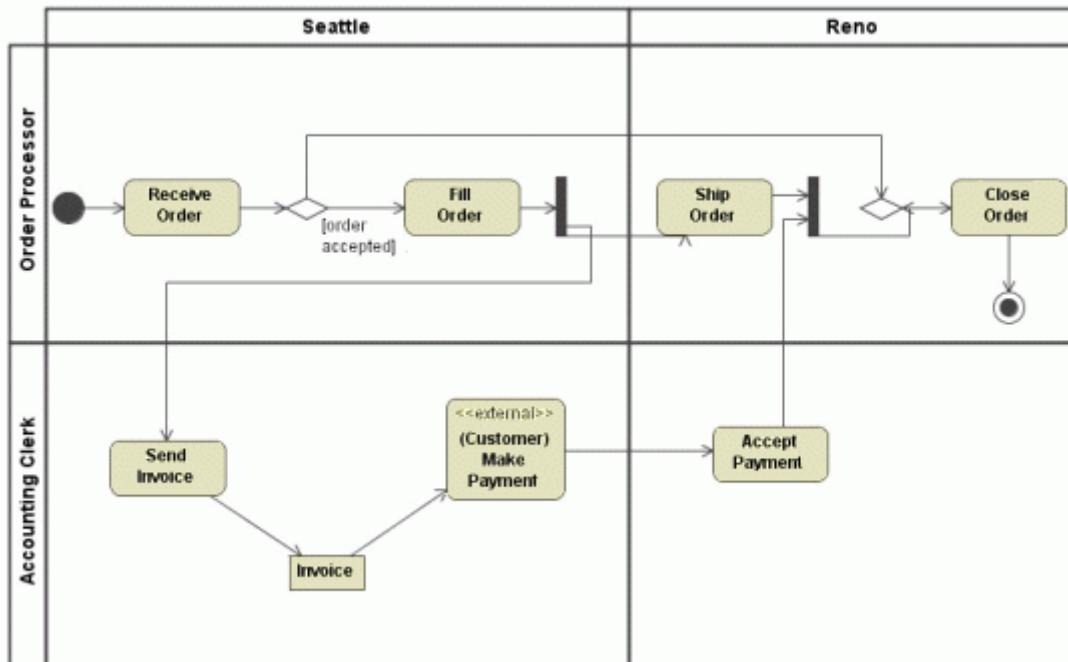
Swimlane

To add model elements to a swimlane

If a swimlane is already drawn in the activity diagram, drawing an action (or any other element) will highlight the swimlane in blue. This means that the action shape will depend on the swimlane symbol.

NOTE: If the model elements depend on a swimlane symbol, they will be deleted if the swimlane symbol is deleted.

The multidimensional swimlanes, with added elements, shown in the Figure below.



Template / Parameterized class

Terms **ParameterableElement**

The OMG UML specification (UML 2.3: Superstructure) states:

A parameterable element is an element that can be exposed as a formal template parameter for a template, or specified as an actual parameter in a binding of a template.

TemplateableElement

The OMG UML specification (UML 2.3: Superstructure) states:

A templateable element is an element that can optionally be defined as a template and bound to other templates.

ClassifierTemplateParameter

The OMG UML specification (UML 2.3: Superstructure) states:

A classifier template parameter exposes a classifier as a formal template parameter.

To define a parameterized class

1. Double-click the class shape or select **Specification** from the class shape shortcut menu. The **Class Specification** dialog box opens. Click the **Template Parameters** tab.
2. To add or edit a parameterized class, click the **Create** button. The **Select Template Parameter Type** dialog box opens. Select the type from the model tree and click **OK**.
3. To remove the selected parameterized class, click the **Delete** button.

To open the **Classifier Template Parameter Specification** window

In the element **Specification** dialog box tree, expand the **Template Parameters** tab and double click the selected template.

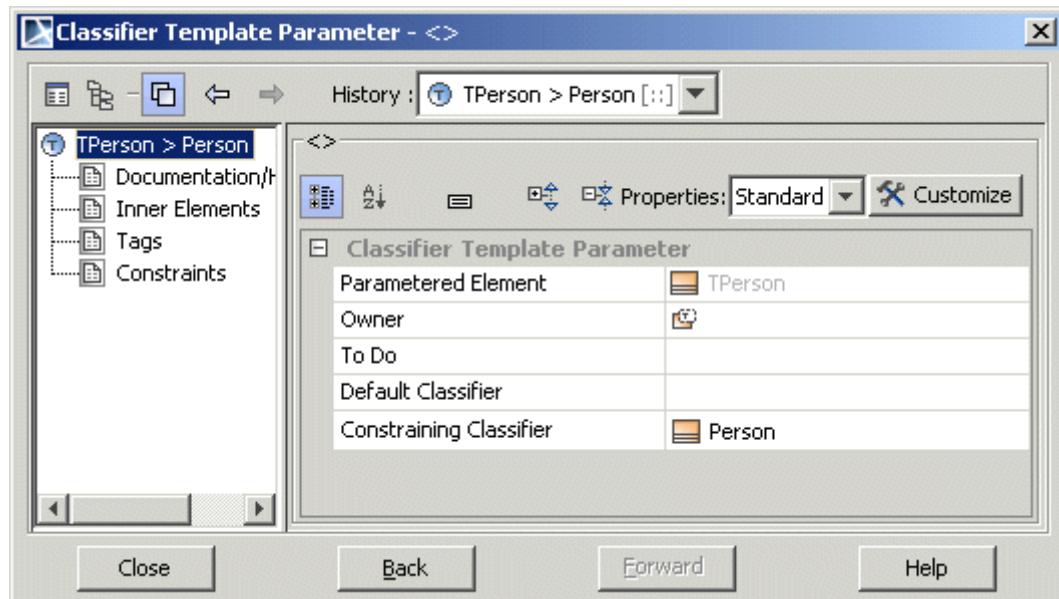


Figure 487 -- Classifier Template Parameter Specification dialog box

Select the **Expert** mode from the **Properties** field to show more properties of the template parameter.

Refer to Section “Specification Window” on page 237 for information about the specification elements not covered in this section

| Tab name | Box name | Function |
|----------------|----------------------------|---|
| General | Parametered Element | The name of the element, which was parametered by a template parameter. |
| | Owner | The name of the element, which contains the template parameter. |

Transition

A transition is a directed relationship between a source vertex and a target vertex. It may be part of a compound transition, which takes the state machine from one state configuration to another, representing the complete response of the state machine to an occurrence of an event of a particular type.

11 MODEL ELEMENTS

Transition

The selected transition could be defined in the transition's Specification window.

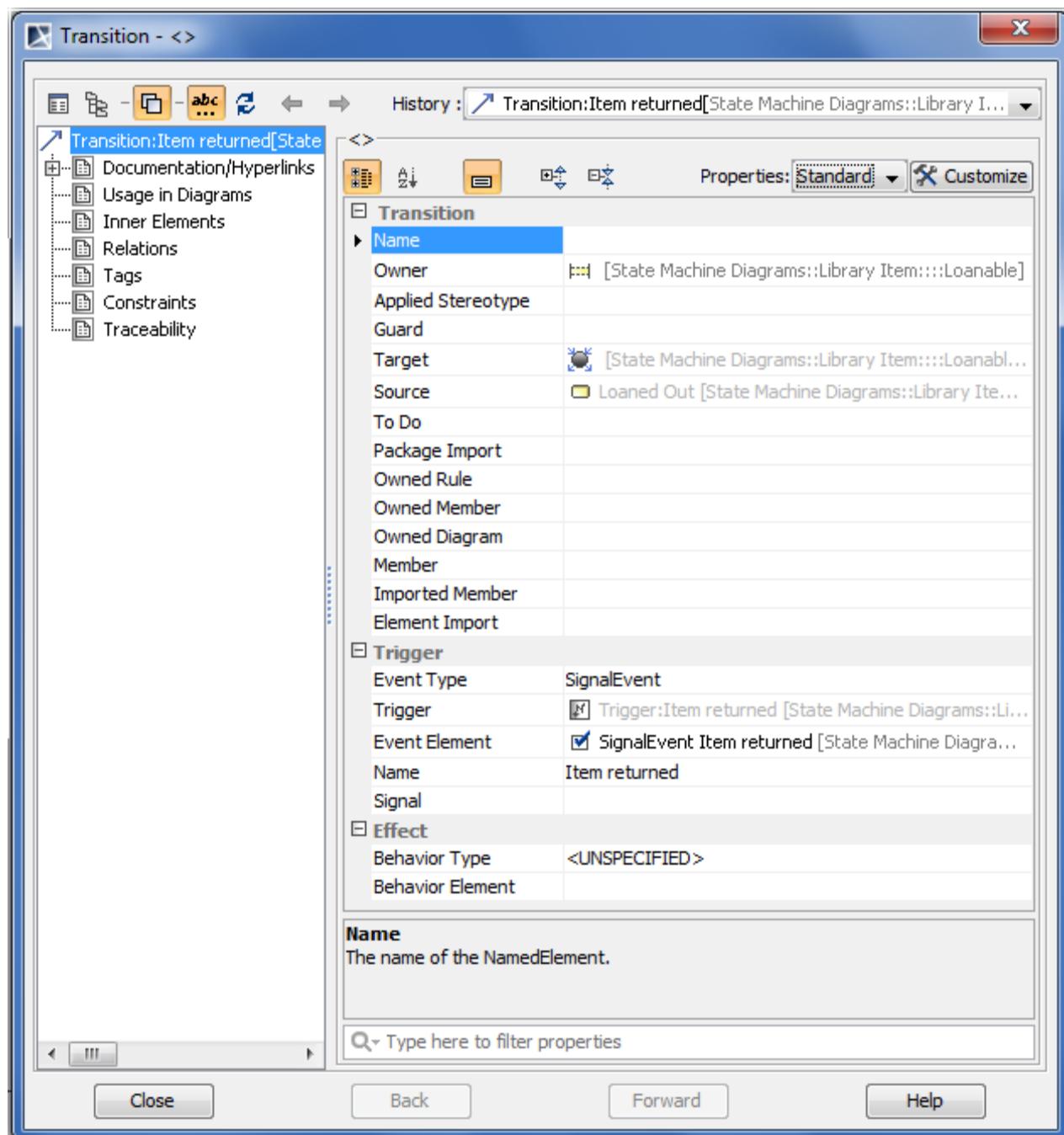


Figure 488 -- Transition's Specification window

11 MODEL ELEMENTS

Transition

For more information about the specification elements not covered in this section, refer to “Specification Window” on page 237.

| Tab name | Property name | Description |
|---|-------------------------|---|
| General Set a general information about the transition. | Guard | A guard is an expression that is attached to a transition as a fine-grained control over its execution. The guard is evaluated when an event instance is dispatched by the state machine. |
| | Target | The name of the model element, which is the target of the transition. |
| | Source | The name of the model element, which is the source of the transition. |
| Trigger property group | | |
| | Event Type | The type of the event. |
| | Trigger | The trigger that specifies the event that causes the transition. |
| | Event Element | The particular event that causes a trigger. |
| Effect property group | | |
| | Behavior Type | The type of the affected action. |
| | Behavior Element | The optional behavior that is performed when the transition fires. |

To assign an event type

1. Open the transition’s Specification window.
2. In the **General** tab, click the **Event Type** property value cell. The list of available event types will open. For more information about event types, refer to Section “Events” on page 684

NOTE The **Event Type** property belongs to the **Trigger** group. If you do not see this property, click the **+** button near the **Trigger** property group to expand its content.

3. Select the event type from the list.
4. According to the selected event type, the **Event Element** property value will be defined as well as additional corresponding properties will appear in the **Trigger** property group.
5. Click the **Close** button.

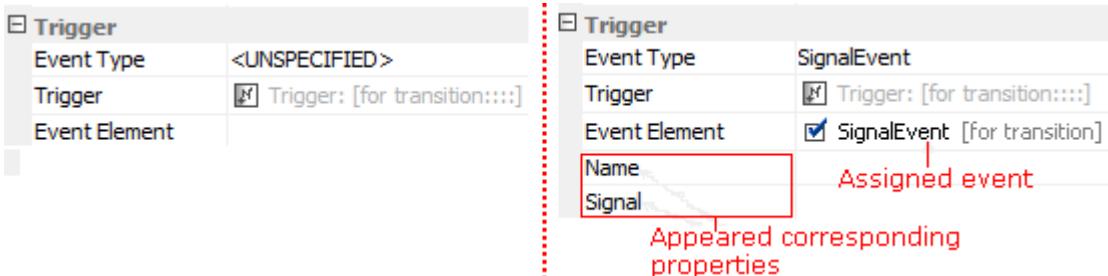


Figure 489 -- Trigger group before assigning event type (on the left) and after assigning event type (on the right)

To assign a behavior type

1. Open the transition’s Specification window.

11 MODEL ELEMENTS

Transition

2. In the **General** ab, click the **Behavior Type** property valuecell. The list of available behaviors will open. For more information about event types, refer to Section “Event types” on page 685.

NOTE The **Behavior Type** property belongs to the **Effect** property group. If you do not see this property, click the **+** button near the **Effect** property group to expand its content.

3. Select the behavior from the list.
4. According to the selected behavior, additional corresponding properties will appear in the **Effect** property group.
5. Click the **Close** button.

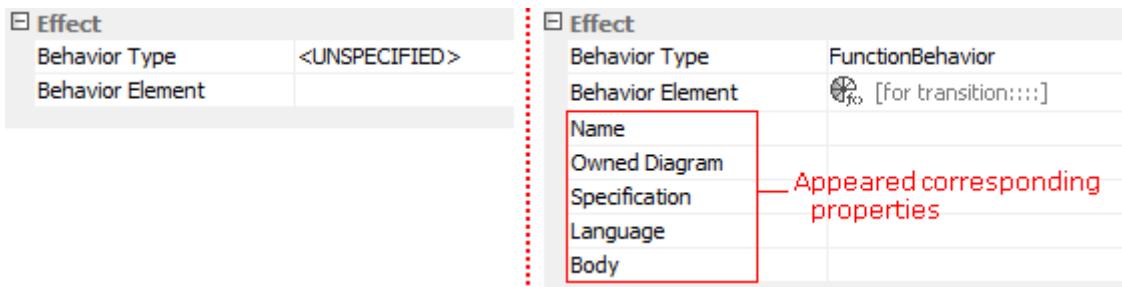


Figure 490 -- Effect group before assigning behavior type (on the left) and after assigning behavior type (on the right)

Easier way of event type assigning

You can assign an event type to a transition by typing an appropriate command on a selected transition path. The event type can be one of the following:

- AnyReceiveEvent
- CallEvent
- ChangeEvent
- TimeEvent

To assign an event type by typing an appropriate command

1. Select a transition.
2. Type a command. The commands that can be used are listed in the following table.

| Event Type | Command Syntax |
|-----------------|---|
| AnyReceiveEvent | all |
| ChangeEvent | when (expression) |
| SignalEvent | signal name |
| TimeEvent | at (time) , if time is specified as relative after (time) , if time is specified as absolute |

NOTE You should specify arguments in brackets according to your project needs.

The typed command with the argument will be displayed on the transition path.



Figure 491 -- Time event assigned to transition

11 MODEL ELEMENTS

Use Case

Transition properties, such as **Event Type**, **Trigger**, and **Event Element**, will be specified automatically after the event is assigned to the transition. To see these properties, open the transition's Specification window.

There is an exception for the event type named Call Event. If an event of this type has an operation defined, the name of the operation will be displayed on the transition path. Use the procedure "To assign an event type" on page 774 to assign this type of event.

IMPORTANT! The event type assignment by typing commands is also available for transitions to self in State Diagrams as well as for protocol transitions and protocol transition to self in Protocol State Diagrams.

Use Case

A use case represents a typical interaction between a user and a system. It captures some of the functionalities and data that the user works with. It is a type of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies a type of behavior, including the variants, that the subject can perform in collaboration with one or more actors. The subject of a use case could be a physical system or any other element that may have the behavior, such as a component, a subsystem, or a class.

An extension point is a reference to one location within a use case where an action sequences from other use cases can be inserted. Each extension point has a unique name within the use case and a description of the location within the behavior of the use case.

A use case is shown as an ellipse with the name inside it. The extension points are listed in the Extension Points compartment of the use case.

Define the selected use case in the **Use Case Specification** dialog box.

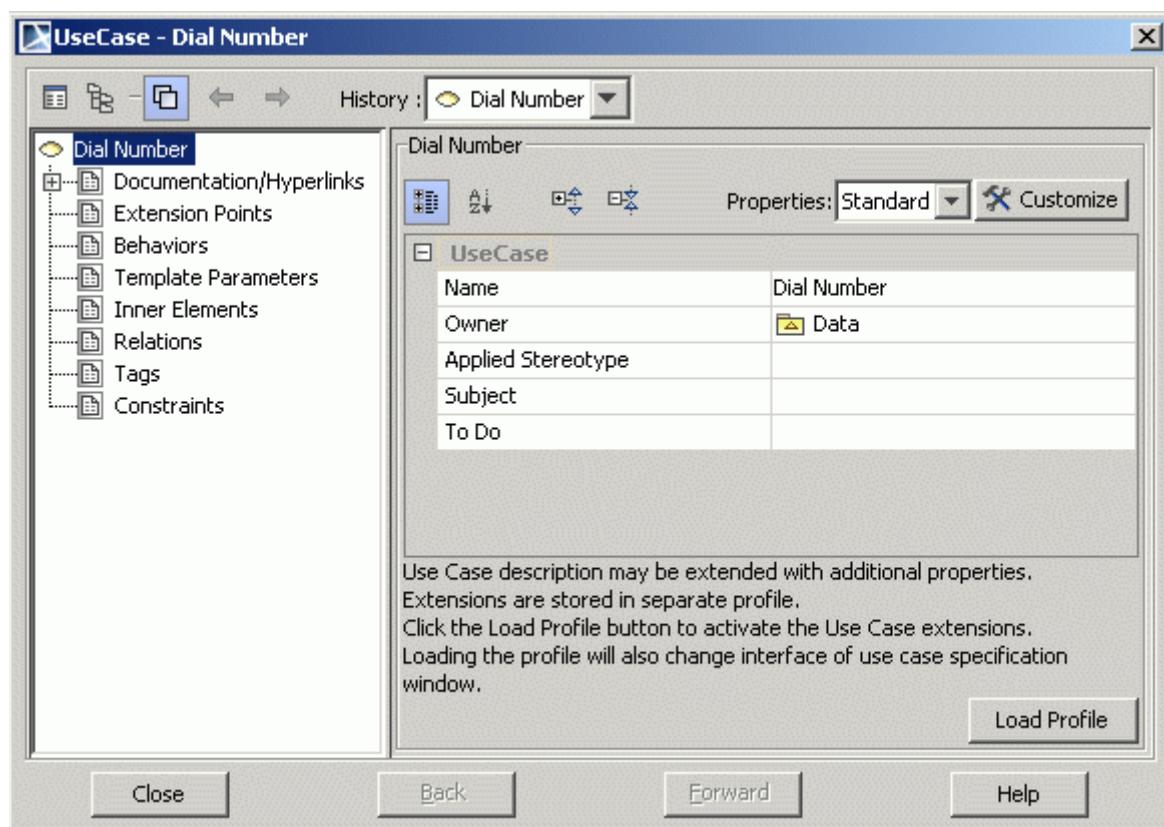


Figure 492 -- Use Case specification dialog box

11 MODEL ELEMENTS

Use Case Extension

Refer to the "Specification Window" on page 237 for information about specification elements..

| Group name | Box | Function |
|---|---------------------|--|
| General | Load Profile | Loads the Use Case description profile that contains specific extensions. NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only. |
| Behaviors From the drop down list, select an activity, state machine, or interaction to create within a use case. | Type | The type of the created element. |
| | Name | The name of the created element. |
| | Create | The list of elements to create opens. Select the element and define the properties in the open properties list. |
| | Delete | Remove the selected diagram from the use case. |
| Extension Points | Create | Define a new extension point. |
| | Delete | Remove the selected extension point from the use case. |
| Inner elements From the drop down list, choose diagrams or constraints to create as an inner element of the use case. | Name | Type the name for a newly created diagram. |
| | Type | The type of the created diagram. |
| | Create | The list of diagrams opens. Select a diagram or constraint and define properties in the open properties list. |
| | Delete | Remove the selected inner element from the use case. |

For more information about working with symbols, see "Diagramming"

To add an extension point for the use case

1. Open the **Use Case Specification** dialog box.
2. Click the **Extension Points** group.
3. Click **Create**. The **Extension Point** properties window opens. Here you may specify the extension point.
 - Press CTRL+ALT+E.
 - From the use case shortcut menu, select **Insert New Extension Point**.
 - From the use case shortcut menu in the Browser, select **New Element**, and then **Extension Point**. Type the name of the extension point.

TIPS:

You may create an extension point directly on a diagram: Draw an extension path between two use cases. Then click **YES** in the message window. Once it is created, specify its name.

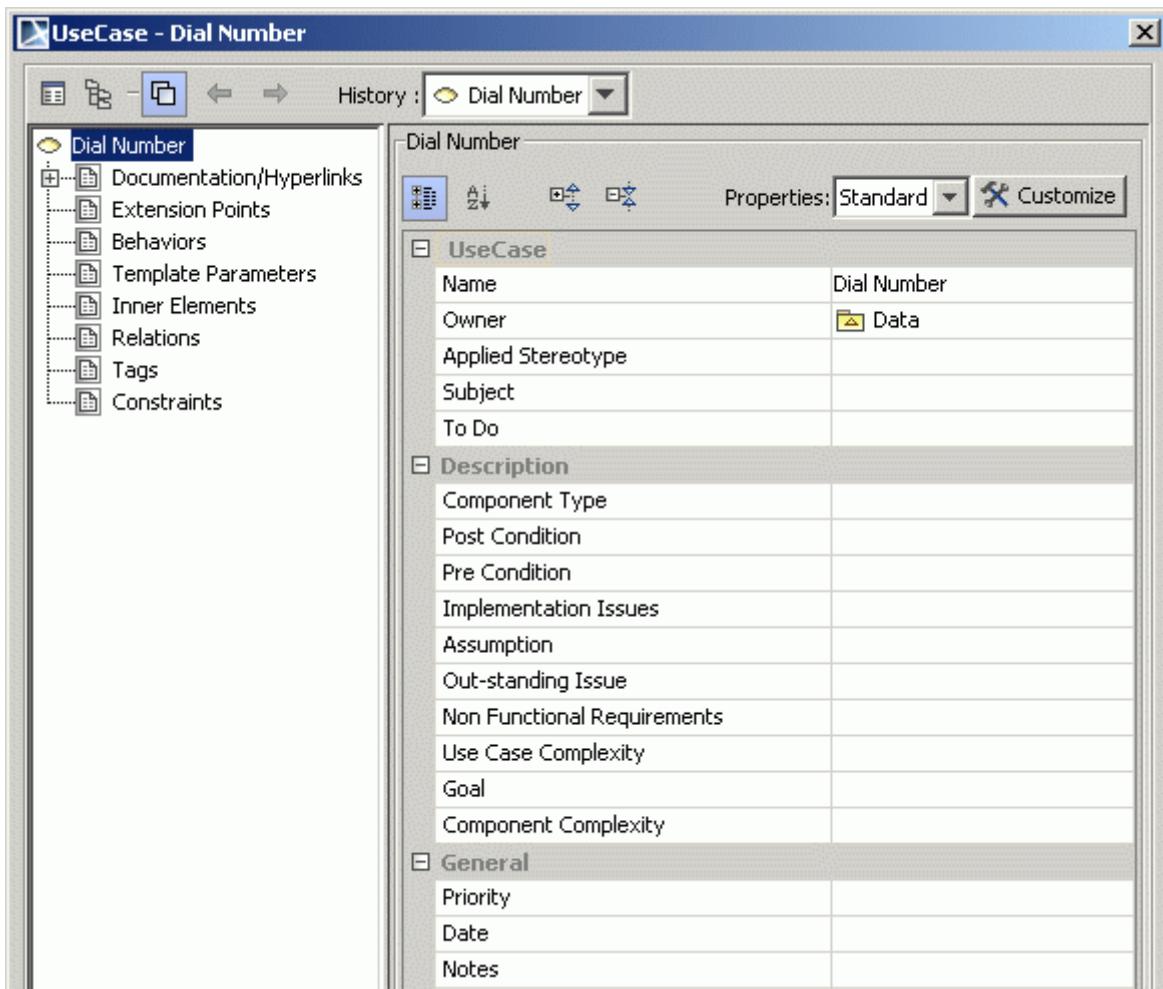
You may define a use case as abstract (for the detailed description, see "Generalizable elements" on page 695.)

Use Case Extension

MagicDraw provides the UseCase description extensions. Now you may define the pre-conditions and post-conditions for the use cases and other extensions, which are implemented as properties and tags in the **Use Case Specification** dialog box. MagicDraw contains the UseCase Description profile, which can be loaded from the **Use Case Specification** dialog box. The properties are available in the **Use Case** Specification window:

11 MODEL ELEMENTS

Use Case Extension



The properties are mapped to tags, which can be listed in the **Specification** dialog box, **Tags** group. If the tags cannot be seen, apply <<requirementUseCase>> stereotype to the use case.

IMPORTANT The tag definitions from the UseCase Description profile are only visible in generated reports, not on the diagrams. For a detailed description about reports, see in “Controlling Merge memory usage” on page 341. For a detailed description of working with tag definitions, see in “Specification Window” on page 237.

To display the tag definitions from the UseCase Description profile on the diagram

1. Open the UseCase Description profile UseCase_Profile.xml as a project (this profile is located in the <MagicDraw installation directory>/profiles folder).
2. In the Browser tree select the property, which represents the tag definition and open its **Specification** dialog box.
3. In the **Applied Stereotype** field, remove <<InvisibleStereotype>>.
4. Save the project.
5. Reload the profile in the project you are working: from the UseCase Description Profile shortcut menu in the Browser tree, select **Modules**, and then **Reload Module**.

Value Specification

You can create a standalone Value Specification in a model using the appropriate element shortcut menu.

To create a Value Specification from the element shortcut menu

1. Select the appropriate element in the Model Browser.
2. Right-click the element. On the shortcut menu point to **New Element > Value Specification** and then select the suitable Value Specification.

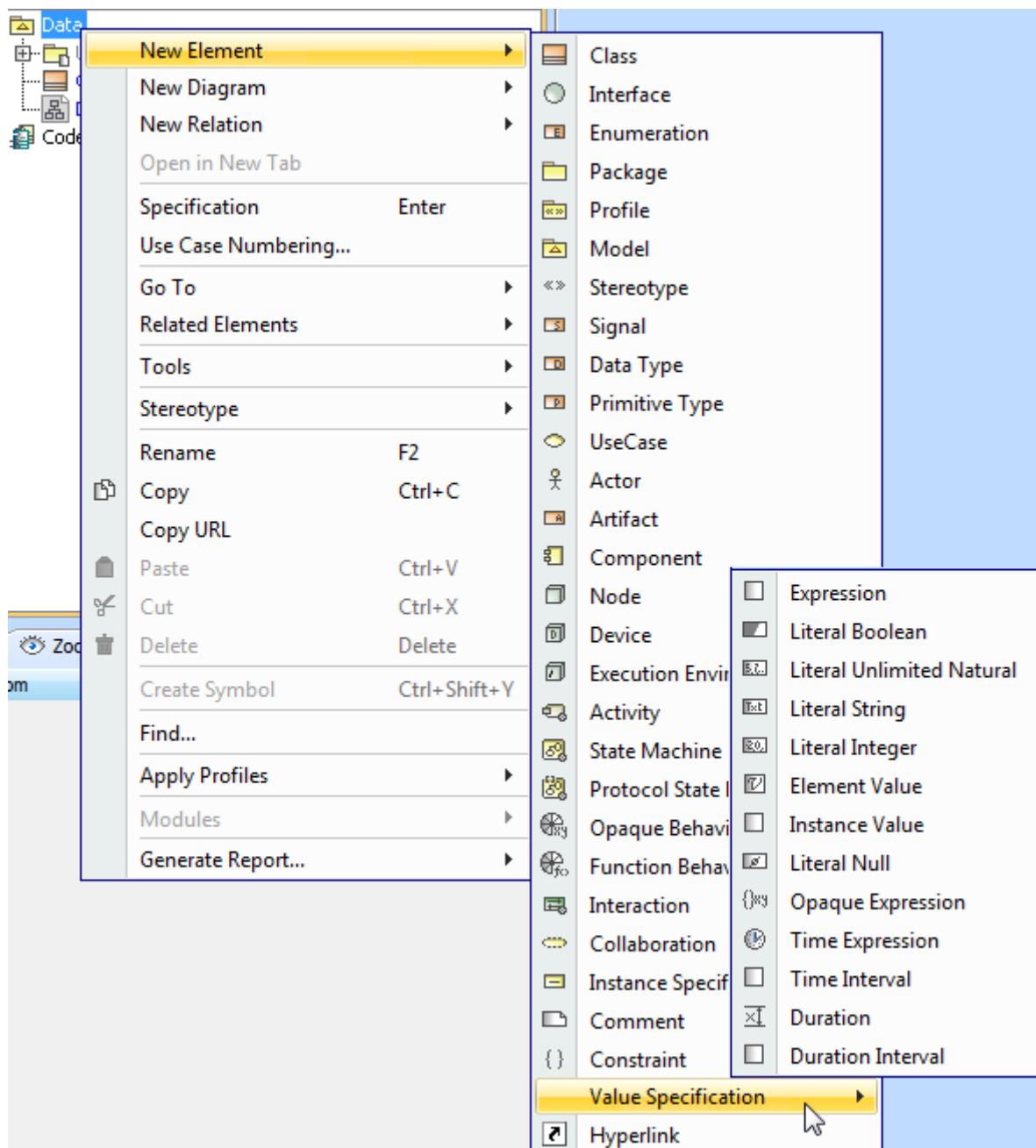


Figure 493 -- Creating Value Specification from the Model Browser

NOTE

Since version 16.0, Value Specifications are displayed in the Containment tree.

11 MODEL ELEMENTS

Value Specification

NEW! Since version 17.0, each property, which can have a Value Specification, has the  button at the end of its row in the Specification window or the **Properties** panel. Click the property value cell to see this button. To change a Value Specification click the button and the shorctut menu with the Value Specifications list will open. Choose the suitable Value Specification from the list.

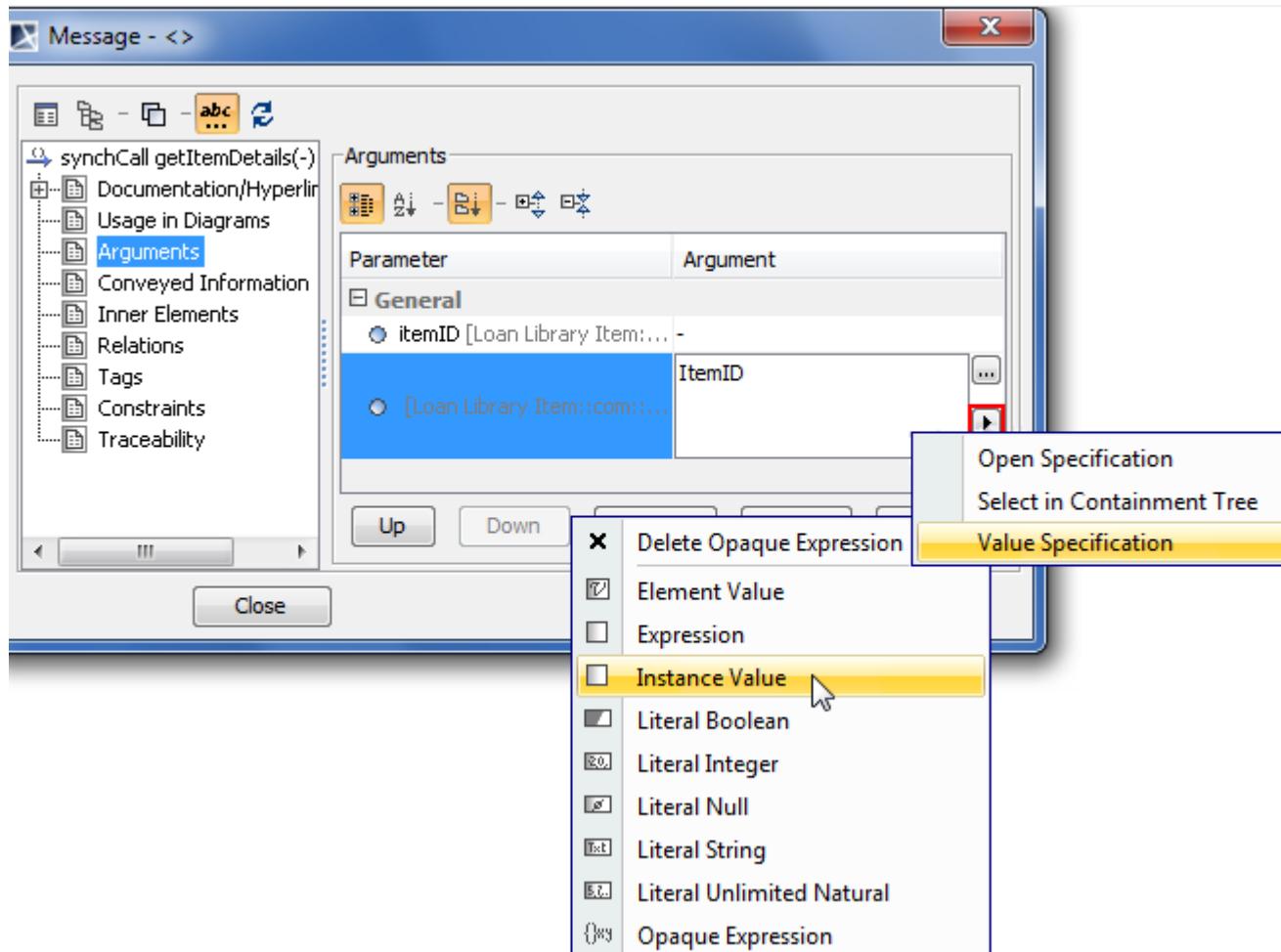


Figure 494 -- Changing property's Value Specification

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

This chapter includes the descriptions of the menu system, the toolbar buttons and the Browser window buttons of the MagicDraw graphical user interface (GUI), and also the descriptions of the icons used in modules and profiling mechanism.

- “Menu System” on page 782
- “Main Toolbars” on page 798.
- “Diagram Toolbars” on page 804
- “Browser Window” on page 807.
- “Icons of general elements” on page 810.
- “Icons of relationships” on page 820.
- “Icons from Modules and Profile mechanism” on page 823.

Menu System

File menu

| Command | Button / Hot keys | Function |
|------------------------------|--|---|
| New Project |  CTRL+N | <p>Creates a new project. Choose an icon to create a new Blank Project, create a Project from an Existing Source, or create a Project from a Template.</p> <p>A project is nameless until you close or save it by choosing the Save Project or Save Project As commands.</p> <p>You may simultaneously create as many new projects as you wish, without saving and closing the previously created or open projects. You may work on only one open or created project at a time. If you wish to work with another project, select the project name in the Projects list on the main window toolbar.</p> |
| Open Project |  CTRL+O | <p>Opens an existing project.</p> <p>The Open dialog box appears. Select a project you wish to open. You may open as many projects as you wish. If you wish to work with another project, select the project name in the Projects list on the main window toolbar.</p> |
| Save Project |  CTRL+S | <p>Saves the open project.</p> <p>To save the current project for the first time, type the name of the project and select the format of the file you wish to save.</p> |
| Save Project As | | <p>Saves the project giving it a name.</p> <p>Use this command to save the current project for the first time or under a different name, the Save dialog box appears. Type the name of the project and select the format of the file you wish to save.</p> |
| Close Project | | <p>Closes an open project.</p> <p>If the open project has unsaved changes, MagicDraw displays a dialog box asking whether the changes should be saved before the project is closed.</p> |
| Close All Projects | | <p>Closes all open projects.</p> <p>If the open projects have unsaved changes, MagicDraw displays a dialog box asking whether the changes should be saved before each project is closed.</p> |
| Open Element from URL | | <p>You can open any elements through their URLs by clicking the Open Element from URL command and the element will be highlighted in the Containment tree or in the diagram. For more information about element URL, see “Copying/Opening Element URLs” on page 291.</p> |
| Use Module | | <p>The Use Module dialog box appears. Choose profile or module for use in the project.</p> |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Menu System

| Command | Button / Hot keys | Function |
|----------------|---|---|
| Import From | | <ul style="list-style-type: none"> • Another MagicDraw Project - The Import dialog box appears. Select the project you wish to import. You may import as many projects as you wish. Imports an existing project to a previously open project. This is the recommended command for importing an existing project to the teamwork server. <p>You may also import the following files:</p> <ul style="list-style-type: none"> • CSV File • UML XMI 1.0, 1.1, 1.2, 2.1 File • MOF XMI File • Eclipse UML 2 (v1.x, v2.x, v3.x) XMI File • Enterprise Architect 2.1 XMI 2.1 File • BPMN 1.1 Project • CA ERwin Data Modeler 7.x • Rational Software Architect/Modeler Project • Rational Rose *.mdl Project File |
| Export To | | <ul style="list-style-type: none"> • Module - saves the content of the selected objects in a separate file. • Template - saves the current project as a template project. • UML XMI 2.1 File • EMF Ecore File • MOF XMI File • Eclipse UML2 (v1.x, v2.x, v3.x) XMI File - exports the MagicDraw model to EMF based UML 2 compatible XMI. • BPEL File - exports the Business Process diagram to the Business Process Execution Language (BPEL), based on BPEL specification v.1.1 |
| Share Packages | | Saves the package as a separate module. |
| Save As Image | | <p>Saves the open diagram as an image of in the selected formats.</p> <p>The Save As Image command is used for saving any item or group of items selected in the diagram pane, in one of the various image file formats.</p> |
| Print |  CTRL+P | <p>The Print dialog box appears. Printing of active diagram, selected symbols, or selected diagrams is available. Select the printer, set the options for the printer, specify the number of copies, and select the specific pages to be printed.</p> |
| Print Preview |  | Preview the diagram before printing. The Print Preview screen opens. |
| Print Options | | The Print Options dialog box opens. |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Menu System

| Command | Button / Hot keys | Function |
|--------------------|-------------------|--|
| Project Properties | | The Project Properties dialog box opens. This dialog box contains the following information: project location, file size, date created, date modified for the last time, and number of diagrams in the project. In the Description tab, type the description of the project or other important information. The Modules tab shows a list of modules the project is using. |
| | | The list shows a specified number of recent project files. Specify the number of files in the Recent Files List Size property in the Environment Options dialog box. |
| Opened Projects | | A list of open projects. |
| Exit | | Exits the application. All open projects are closed. If an open project has unsaved changes, the MagicDraw displays a dialog box asking whether the changes should be saved before closing the project. |

The **File** menu contains a list of recent projects. The shortcuts with numbers to recent projects are displayed. If the shortcut is selected from the **File** menu, the recent project will open instantly.

Edit menu

MagicDraw allows you to use the **Edit** menu commands while drawing the diagrams on the Diagram pane. The commands allow selecting, cutting, copying, and pasting of items and entire diagrams, reversing the actions you have taken while drawing, and finding an item in the current project.

| Command | Button/ Shortcut keys | Function |
|---------|--|--|
| Undo |  CTRL+Z | Reverses the last action you have performed while drawing the diagram on the Diagram pane (moving, dragging, resizing, cutting, copying, pasting, deleting, selecting, editing shapes, setting project and shape properties, etc.). Actions are reversed in the opposite order you have performed them, starting with the most recent. By default, the limit of the undo mechanism is 100 steps backwards. To change the limit, choose Environment from the Options menu. The Environment Options dialog box appears. Change the Undo List Size property. The Undo command is unavailable until you perform any action after loading an existing project or creating a new project. Each command has an easily recognized name. You will be able to see the command history and undo or redo one action or a group of actions. The main window will have two lists of commands: one for the undo commands, another one for the redo commands. |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Menu System

| Command | Button/ Shortcut keys | Function |
|------------------------------------|---|---|
| Redo |  CTRL+Y | Reverses the action of the Undo command (moving, dragging, resizing, cutting, copying, pasting, deleting, selecting, etc.). The Redo command is unavailable until you use the Undo command. |
| Cut |  CTRL+X | Cuts the selected items or group of items on the Diagram pane. The cut items are placed in the clipboard. Later they can be pasted back to the Diagram pane of the current or to another project. The Cut command is unavailable until you select any item or any group of items on the Diagram pane of the current project. |
| Copy |  CTRL+C | Copies the selected items or group of items on the Diagram pane. The copied items are placed in the clipboard. The cut items can be pasted back to the Diagram pane or to another project. The Copy command is unavailable until you select any item or any group of items on the Diagram pane of the current project. |
| Copy URL | | Copy a project element URL to a clipboard and share it with other as a quick reference to model elements. For more information about copying element URL, see “Copying/Opening Element URLs” on page 291. |
| Paste |  CTRL+V | Pastes the cut or copied items or group of items from the clipboard to the Diagram pane of the current project. MagicDraw creates shapes for items, or a group of items, in the current project. You will see the data and shapes of the pasted items and diagrams in the Browser window. |
| Paste With New Data | CTRL+E | Pastes the cut or copied items or groups of items from the Clipboard to the Diagram pane of the current project. MagicDraw creates new data and data shapes in the current project. |
| Delete |  CTRL+D | Deletes data together with symbol. It is unavailable until you select any item or group of items in the current project. |
| Delete Symbol(s) | DELETE | Deletes a symbol of model element from the diagrams, by leaving it in the model. |
| Select All | CTRL+A | Selects all items on the Diagram pane of a particular project. |
| Select All of the Same Type | CTRL+ALT+A | Selects all shapes of the selected types in active diagram. Enabled, when one or more shapes are selected in active diagram. |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Menu System

| Command | Button/ Shortcut keys | Function |
|--------------------|---|---|
| Copy as BPM | CTRL+SHIFT +B | Copies the selected model elements to the system clipboard. If no model elements are selected, the active diagram is copied. |
| Copy as EMF | CTRL+SHIFT +E | Copies the selected model elements to the system clipboard. If no model elements are selected, the active diagram is copied. NOTE Copying as EMF is available only under Windows system. |
| Copy as JPG | CTRL+SHIFT +J | Copies the selected model elements to the system clipboard as a JPG image. If no model elements are selected, the active diagram is copied. NOTE Copying as JPG is available only under Windows system. |
| Copy as PNG | CTRL+SHIFT +P | Copies the selected model elements to the system clipboard as a PNG image. If no model elements are selected, the active diagram is copied. NOTE Copying as PNG is available only under Windows system. |
| Find |  CTRL+F | Opens the Find dialog box. |
| Quick Find | CTRL+ALT+F | Performs a quick search of class/interface, classifier, or diagram. |
| Find TODO | | Performs a search for the TODO tagged value. Results are displayed in the Browser, Search Results tab. |
| Paths | | <ul style="list-style-type: none"> ● Path Style - choose the line style for drawing a path. ● Rectilinear – drawing rectilinear lines. ● Oblique - drawing free form lines. ● Bezier - in computer graphics, a curve that is calculated mathematically to connect separate points in smooth, free-form curves. ● Change Path Style – switches in series between rectilinear, oblique, and bezier path line style. Shortcut is CTRL+L. ● Reset Labels Positions – resets the changed path labels (name, roles, etc.) to the default position. ● Remove Break Points – removes the break points of the path and makes the path a line straight. |
| Symbol | | All commands that are available through the shortcut menu for a particular symbol. |

View menu

| Command | Button / Hot keys | Function |
|---------------------------------|---|---|
| Fit In Window | CTRL+W | Reduces the size of the whole diagram to fit in the Diagram pane. |
| Zoom In |  CTRL+NUMPAD PLUS SIGN (+) | To change the zoom step size choose Environment from the Options menu and set the Zoom Step Size property in the Environment Options dialog box. |
| Zoom Out |  CTRL+NUMPAD MINUS SIGN (-) | Decreases the size of the selected objects in the diagram pane by x percent. To change the zoom step size, choose Environment from the Options menu and set the Zoom Step Size property in the Environment Options dialog box. |
| Zoom 1:1 |  CTRL+NUMPAD SLASH MARK (/) | Restores the original size of the selected diagram symbols. |
| Zoom To Selection | CTRL+NUMPAD ASTERICS MARK (*) | Increases the size of the selected model element on the Diagram pane to the maximum visible size. |
| Refresh | CTRL+R | Repaints all diagram shapes. |
| Grid | | Set grid options. Every diagram may have its own grid settings: <ul style="list-style-type: none">• Show Grid - show/hide grid• Snap Paths to Grid - use/do not use grid for drawing paths.• Snap Shapes to Grid - use/do not use grid for drawing shapes.• Grid Size - change the grid size. Type the number. |
| Recently Closed Diagrams | F12 | Opens a list of diagrams that were last closed. |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Menu System

| Command | Button / Hot keys | Function |
|---|-------------------|---|
| Main Toolbars: <ul style="list-style-type: none">● Menu Bar● File● Diagrams● Custom Diagrams● Opened Projects● Teamwork● Rearrangable● Hidable● Floatable● Expert Menu Mode● Expert Toolbar Mode● Customize | | Clear the check boxes of the toolbars you want to hide or select/clear check boxes to rearrange toolbar modes. |
| Diagram Toolbars: <ul style="list-style-type: none">● Symbol Editing● Path Editing● Edit● View● Layout● Rearrangable● Hidable● Floatable● Expert Mode● Customize | | Clear the check boxes of the toolbars you want to hide or select/clear check boxes to rearrange toolbar modes. |
| Status Line | | Select the check box of Show Status Bar or Show Memory Monitor to display this info on the status line. |

Layout menu

Use the commands of the **Layout** menu for managing the layout of the shapes on the current Diagram pane. You must select more than one shape for other Layout menu commands. A class diagram should be open and activate before using the **Class Diagram** command in this menu.

| Command | Function |
|--|--|
| Layout Options | Opens the Diagram Layout Options dialog box. Layout options for the diagram can be set. |
| Quick Diagram Layout (CTRL+Q) | Applies recommended layout with default options on the active diagram. |
| Layout Class Diagram Style | Applies layout, which uses specific layout algorithms to improve class diagram readability. |
| Layout Activity Diagram Style | Applies layout, which uses specific layout algorithms to improve activity diagram readability. |
| Layout Business Process Diagram Style | Applies layout, which uses specific layout algorithms to improve business process diagram readability. |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Menu System

| Command | Function |
|--|--|
| Layout Hierarchic Style | Applies layout that portrays the main direction or flow of directed graphs. It is ideal for many application areas, especially for Workflow, Software Engineering, Customer relationship management, Configuration management, Process modeling, Database Modeling, and Bio informatics. |
| Layout Tree Style | Applies layout, which specializes in the layout of tree-structured graphs. The need to visualize directed or undirected trees arises in many application areas, e.g. Data-flow analysis, Software Engineering, Network management, Bio informatics. |
| Layout Orthogonal Style | Applies layout that is well suited for medium sized sparse diagrams. It produces compact drawings with no overlapping shapes, few crossings, and few bends. All edges will be routed in an orthogonal style, i.e. only rectilinear style links will be used. |
| Layout Organic Style | Applies layout of the organic style graph. |
| Layout Circular Style | Applies layout of the algorithm that portrays interconnected ring and star topologies. It is excellent for applications in social networking (criminology, economics, ...), network management, WWW visualization, and eCommerce. |
| Route Paths Orthogonal Style | Applies layout, which routes the links of a diagram using only vertical and horizontal line segments, while keeping the positions of the shapes in the diagram fixed. The routed links will usually not cross through any shapes and not overlap any other links. |
| Route Paths Organic Style | Applies layout, which routes the links of a diagram using oblique link style, while keeping the positions of the shapes in the diagram fixed. The routed links will usually not cross through any shapes and not overlap any other links. |
| Make Same Width | Applies layout to the selected shapes according to their width. After the layout, the width of the shapes is equal (according to the widest). |
| Make Same Height | Applies layout to the selected shapes according to their height. After the layout, height of the shapes is equal (according to the tallest). |
| Align • Right • Left • Top • Bottom | Aligns the selected shapes: <ul style="list-style-type: none">• Aligns the selected shape(s) vertically, starting with the rightmost shape(s).• Aligns the selected shape(s) vertically, starting with the leftmost shape(s).• Aligns the selected shape(s) across from the uppermost shape(s).• Aligns the selected shape(s) across from the lowermost shape(s). |
| Center • Horizontally • Vertically | Centers the selected shapes: <ul style="list-style-type: none">• Centers the selected shape(s) on a horizontal line.• Centers the selected shape(s) on a vertical line. |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Menu System

| Command | Function |
|---|---|
| Space Evenly <ul style="list-style-type: none">• Horizontally• Vertically | Sets spaces among the selected shapes evenly. <ul style="list-style-type: none">• Spaces between the selected shapes becomes even on a horizontal line.• Spaces between the selected shapes becomes even on a vertical line. |

Diagrams menu

| Command | Hot keys | Function |
|--------------------------|---|---|
| Diagrams | | Select the required diagram in the list to edit, create, remove, or open it. |
| Customize | | The Customize Diagrams dialog box opens. |
| Diagram Wizards | | Wizards for creating diagrams may be opened: <ul style="list-style-type: none"> ● Class Diagram Wizard ● Package Dependency Diagram Wizard ● Package Overview Diagram Wizard ● Activity Decomposition Hierarchy Wizard ● Hierarchy Diagram Wizard ● Realization Diagram Wizard ● Sequence Diagram from Java Source Wizard ● Content Diagram Wizard |
| Previous Diagram |  CTRL+0 | Activates the previously open diagram. |
| Next Diagram |  CTRL+9 | Activates the next diagram. |
| Load All Diagrams | | If there are unloaded diagrams in the project, this command loads all diagrams. |

Options menu

| Command | Function |
|---------------------|--|
| Project | The Project Options dialog box opens. |
| Modules | The Modules dialog box opens. |
| Environment | The Environment Options dialog box opens. |
| Perspectives | Choose a command from the submenu - to switch to different Perspective or to Customize . |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Menu System

| Command | Function |
|------------------------|--|
| Look and Feel | <p>Choose a style for the MagicDraw (GUI) from the following list:</p> <ul style="list-style-type: none">• Metal.• CDE/Motif.• Vsnet (Windows).• Office 2003 (Windows).• Office 2007 (Windows)• Eclipse (Windows).• Xerto. <p>The Look and Feel Themes command allows you to personalize the GUI: set your favorite colors and fonts.</p> |
| Interface Style | Single Window, or Multiple Windows interface style. |

Tools menu

| Command | Hot key | Function |
|-----------------------------------|---------|--|
| Apply Pattern | | The Pattern Wizard dialog box appears. Create the design pattern for the selected class, interface. |
| Model Transformations | | Opens the Model Transformations Wizard dialog box with a list of all available transformations. |
| Hyperlinks | | Opens the Hyperlink dialog where you can add hyperlinks to any model elements. |
| Report Wizard | | <p>The Report Wizard is the new report engine for MD 14.0 and above. It is designed to solve the several problems of the old engines (XSL/XSLT and JPython).</p> <p>It supports text based templates to generate the output file. The format of output file depends on the type of the template file. The type of template files that the Report Wizard supports are normal text, RTF, HTML, Spreadsheet template (need to be saved as HTML format), and XML template (DocBook or FO).</p> <p>All commercial MagicDraw editions will have full use of all features within the Report Wizard. For more details, see the "<i>MagicDraw ReportWizard UserGuide.pdf</i>", which is located in the MagicDraw installation directory, Manual folder.</p> |
| Quick Reverse | | Choose the language you need (Java, Java Bytecode, C++, C#, CIL, CIL Disassembler, IDL, CORBA IDL, DDL, EJB, EJB 2.0, XML Schema, WSDL). Opens the Round Trip Set dialog box. (For more details, see Code Engineering User's Guide). |
| Generate Code Framework | CTRL+G | Generates code for the selected items in the current diagram. Opens the Message Window with the information appears (For more details, see Code Engineering User's Guide). |
| Check Syntax | CTRL+T | Checks syntax in the model according to the default code engineering language. Opens the Message Window with the information. |
| Set empty tags to defaults | | <p>Set default tag value to tag with empty value. This functionality is needed when the stereotype is already assigned to an element and the new mandatory tag definition with default value is created for the stereotype. After creating such a tag definition, the model elements that have the modified stereotype applied will have newly created tags unset.</p> <p>For more information see "To create default tag values" on page 632.</p> |
| Integrations | | Opens the Integrations dialog box with a list of tools for possible integration with MagicDraw. |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Menu System

| Command | Hot key | Function |
|--|------------------|--|
| Requirements | | <p>Integration with CaliberRM.</p> <ul style="list-style-type: none"> ● CaliberRM |
| <ul style="list-style-type: none"> ● DataHub ● MagicRq | CTRL+ SHIFT+R | <ul style="list-style-type: none"> ● Opens the Login dialog box for login to CaliberRM server. ● Logs out from CaliberRM server. ● Launches CaliberRM Client application. ● Opens the CaliberRM Requirements window in MagicDraw. <ul style="list-style-type: none"> ● Starts Cameo DataHub, if it is enabled. More information about Cmaeo DataHub at http://www.magicdraw.com/cameodatahub. ● Starts MagicRq plugin, if it is enabled. More infomation about MagicRq plugin at http://www.magicdraw.com/magicrq. |
| CVS | | <p>Performs operations with CVS (for detailed description of integration with CVS, see MagicDraw Integrations User's Guide)</p> <ul style="list-style-type: none"> ● Allows the user to enter a CVS command line (like "checkout -c") whenever the CVS command is not available through the menus. The Command Line dialog box opens.  |

Figure 1 -- Command Line dialog box

- Checkout Module
 - Add Project to CVS
 - Commit Project to CVS
 - Update CVS Project
- NOTE:** You can add, update or commit projects to CVS only if they are saved in some checked out directory or subdirectory.
- Commits a project to CVS. The **Commit Project to CVS** dialog box opens.
 - Updates the current project by loading the latest project version from CVS.

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Menu System

| Command | Hot key | Function |
|--|---------|--|
| ProActivity <ul style="list-style-type: none">• Import• Export | | The Import ProActivity Files dialog box opens. Specify ProActivity files to start the import process. The UML diagrams will be generated depending on the input files. The Export to ProActivity wizard guides the MagicDraw model export process to ProActivity. |
| Check Spelling | | Checks spelling for the whole project or for the selection. The Check Spelling dialog opens. |
| Scripts | | Runs the selected script. Select the script from the Scripts dialog. |
| ParaMagic Plugin | | Starts ParaMagic Plugin, if it is enabled. More about ParaMagic Plugin at http://www.magicdraw.com/paramagic . |
| MDA | | Starts CameoMDA plugin, it is enabled. More about CameoMDA Plugin at http://www.magicdraw.com/cameomda . |

Analyze menu

| Command | Shortcut keys | Function |
|---------------------------------|---------------|--|
| Model Visualizer | | Opens the Model Visualizer dialog box with a list of all available wizards. |
| Metrics | | Metrics feature allows the measurement of a project by different viewpoints. Specify the metrics scope in the Metrics Options dialog box. |
| Compare Projects | | Opens the Compare Projects dialog box, where you can choose projects to perform model differencing. |
| Dependency Matrix | | <p>Dependency Matrix is a method of visualizing and representing dependency criteria. Diagrams, UML, and extended UML elements serve as row and column entries. The cells in the matrix show where these elements are associated - related.</p> <p>Choose the Create Blank Matrix command to open the Matrix Dependency View.</p> <p>Choose the Matrix Templates command to open the Dependency Matrix Templates dialog box.</p> |
| Dependency Checker | | Opens the Dependency Checker dialog box to configure the options for the dependency analysis of the whole project. |
| Validation | | <p>Validation functionality allows the completeness and correctness evaluation of the models, created by the user, according to constraints defined in Object Constraint Language (OCL) or java code.</p> <p>For more information about validation, see “Validation” on page 444.</p> |
| Display Paths | | Displays paths among shapes that are already created in the model data. |
| Display Related Elements | | Displays elements related to the selected element. |
| Create Relation Map | | <p>Creates the Relation Map for the selected element.</p> <p>More information about relation map, see “Relation Map” on page 398.</p> |
| Used By | CTRL+ALT+U | Finds a list of all elements that reference the current element. |
| Depends On | CTRL+ALT+D | Finds a list of the elements that depend on the current element. |
| Go To | | This is a feature that allows you to find model elements associated with the selected model element. Enabled when a selected element is related to another model element. |

Teamwork menu

For a detailed description about Teamwork Server, see [MagicDraw Teamwork System User's Guide](#).

Window menu

You may use commands of the **Window** menu to manage the layout of the windows.

| Command | Hot key | Function |
|--|---------------|--|
|  Containment | | Opens the Containment Tree tab in the Browser window. |
|  Inheritance | | Opens the Inheritance Tree tab in the Browser window. |
|  Diagrams | | Opens the Diagrams Tree tab in the Browser window. |
|  Model Extensions | | Opens the Model Extensions Tree tab in the Browser window. |
|  Search Results | | Opens the Search Results tab in the Browser window. |
|  Documentation | | Opens the Documentation tab in the Browser window. |
|  Zoom | | Opens the Zoom tab in the Browser window |
|  Properties | | Opens the Properties tab in the Browser window |
|  Messages Window | CTRL+M | Opens the Messages Window . The Messages Window is used for displaying the warnings and errors that may appear in the project. It appears automatically and is intended to display the warnings and errors after saving, loading, exporting and importing the project. |
| Reset Windows Configuration | | All Browser tabs are placed in their default position. |
| List of opened diagrams | | Displays a list of open diagrams. |
| Close All Diagrams But Current | CTRL+SHIFT+F4 | Closes all open diagrams except the one you currently being used. |
| Close All Diagrams | CTRL+ALT+F4 | Closes all open diagrams. |

The **Window** menu contains a list of open diagrams in the project. The list shows the specified number of the recent diagrams. This number can be customized in the **Recent Windows List Size** property in the **Environment Options** dialog box. For a detailed description on this dialog box, see the Section "Customizing Environment Options" on page 89.

Help menu

| Command | Function |
|---|---|
| Help Contents | Displays a table of contents for the MagicDraw Help. |
| User Manual | Opens the MagicDraw User Manual. |
| Plugins User Manuals | Displays the list of all available user's guides and user's manuals. |
| Tip of the Day | Displays the Tip of the Day screen. |
| Report an Issue | The MagicDraw team always welcomes your initiative. Submit bugs, suggestions, and new feature requests through the Report an Issue dialog directly to Online Support System. |
| View and Submit Internal Errors | <p>View errors received during work with MagicDraw. Send a bug report to the MagicDraw support team.</p> <p>For more information about submitting a bug see "Reporting issues directly from MagicDraw" on page 37.</p> |
| Check for Updates | Opens The HTTP Proxy Server Connection dialog box. Set the data for connection to start the MagicDraw UML update. |
| <Product Name> License Manager | Select evaluation key, activate commercial license, or use the floating license. |
| Finish Offline Floating Session | If you are using offline Floating License, finish the offline session. |
| Show Welcome Screen | If no project is open, you may choose this command to open the Welcome screen. |
| What's New | Opens the New and Noteworthy section: http://www.magicdraw.com/newandnoteworthy |
| News Reader | Read the latest news about new updates, products, resources, and No Magic Inc. events. |
| Resource/Plugin Manager | Check for available updates and new resources in the Resource/Plugin Manager window. |
| MagicDraw on the Web | <p>The WWW pages where you can find additional information about MagicDraw and UML.</p> <ul style="list-style-type: none"> ● Online Support ● Online Demo ● New and Noteworthy ● MagicDraw Home Page ● UML Stuff ● No Magic Home Page |
| Entertainment with UML | <ul style="list-style-type: none"> ● Memory game. Open pictures to find pairs with as few tries as possible. ● Puzzle game. Transpose the separate parts to get the whole picture. |
| About MagicDraw | Displays the information screen about the MagicDraw tool. |

Main Toolbars

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Main Toolbars

The main toolbars are located at the top of the MagicDraw window, below the main menu. They contain the commands for commonly used tasks, so helping to speed up your work with MagicDraw.



Figure 2 -- The fragment of the Main toolbars

For more information about customizing toolbars, see “Customizing toolbars” on page 72.

File Toolbar

IMPORTANT! If you cannot see some of the buttons, please, check the perspective and the mode you are working on.

| Button | Title (shortcut keys) | Description |
|---|--|---|
|  | New Project (Ctrl+N) | To create a new blank project, press the New Project button: <ul style="list-style-type: none"> The New Project dialog box opens. Select the Blank Project icon. Specify the file name in the Name text box. Click the “...” button to select a location to store the newly created project in your computer. Click OK. |
|  | Open Project (Ctrl+O) | To open an existing project, press the Open Project button. |
|  | Save Project (Ctrl+S) | To save the current project. |
|  | Print Active Diagram (Ctrl+P) | To print an open diagram. |
|  | Print Preview | The Print Preview dialog box opens showing how your diagram looks before printing. |
|  | Find (Ctrl+F) Find TODO Quick Find (Ctrl+Alt+F) | Search for an element, symbol or diagram in the project according to your selected criteria. |
|  | Undo action (Ctrl+Z) | Undo the last action you performed while drawing the diagram on the Diagram pane (moving, dragging, resizing, cutting, copying, pasting, deleting, selecting, editing shapes, setting project and shape properties, etc.). Actions are reversed in the order you have performed them. The Undo command is unavailable until you perform an action after loading an existing project or creating a new project. By default, the limit of the undo mechanism is 100 steps backwards. To change the limit, select Environment from the Options menu. The Environment Options dialog box opens. Change the Undo List Size property. Each command has an easily recognized name. You will be able to see the command history and undo or redo action history. The main window will have two lists of commands: one for the undo commands, another one for the redo commands. |
|  | Redo action (Ctrl+Y) | Restore the Undo action (moving, dragging, resizing, cutting, copying, pasting, deleting, selecting, etc.). The Redo command is unavailable until you use the Undo command. |

Diagrams Toolbar

IMPORTANT! If you cannot see some of the buttons, please, check the perspective and the mode you are working on.

| Toolbar button | Title (shortcut keys) | Description |
|---|--|--|
|  | Class Diagram (Ctrl+Shift+1) | A class diagram is a graphic representation of the static structural model. It shows classes and interfaces, along with their internal structure and relationships. The classes represent the types of objects that are handled in a system. ^a |
|  | Use Case Diagram (Ctrl+Shift+2) | A use case is a description of the functionality (a specific usage of a system) that a system provides. |
|  | Communication Diagram (Ctrl+Shift+3) | A communication diagram illustrates the various static connections between objects and it models their interactions. |
|  | Sequence Diagram (Ctrl+Shift+4) | A sequence diagram is a time-oriented view of the interaction between objects. |
|  | State Diagram (Ctrl+Shift+5) | A state diagram describes the lifecycle of an object and its behavior (such as a procedure or operation), or a behaviored feature (such as a use case). |
|  | Protocol State Machine Diagram (Ctrl+Shift+6) | A protocol state machine is always defined in the context of a classifier. It specifies which operations of the classifier can be called in which state and under which condition, thus specifying the allowed call sequences on the classifier operations.* |
|  | Activity Diagram (Ctrl+Shift+7) | The purpose of an activity diagram is to focus on flows driven by the internal processing (as opposed to external events). |
|  | Implementation Diagram (Ctrl+Shift+8) | The implementation diagrams help developers describe the logical software structure inside a computer system or across a large distributed system. |
|  | Composite Structure Diagram (Ctrl+Shift+9) | A composite structure diagram allows a decomposition and modeling of the internal structure of classifiers. |
|  | Interaction Overview Diagram (Ctrl+Shift+0) | The interaction overview diagrams define interactions through a variant of the activity diagrams in a way that promotes the overview of the control flow. |

a. Material from the OMG UML Specification has been duplicated with permission.

Custom Diagrams Toolbar

IMPORTANT! If you cannot see some of the buttons, please, check the perspective and the mode you are working on.

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Main Toolbars

| Toolbar button | Title | Description |
|---|---------------------------------|---|
|  | Business Process Diagram | The Business Process Diagrams convey business processes of different participants in a modeled system. In a MagicDraw model, the Process contains a Business Process Diagram, depicting it as a graph of Flow Objects. |
|  | CORBA IDL Diagram | CORBA IDL diagram facilitates the creation of CORBA IDL elements. Also the following patterns are available for CORBA IDL: Interface, Value Type, Type Definition, Sequence, Array, Fixed, Union, Enumeration, Struct, and Exception. |
|  | Free Form Diagram | In the free form diagram you may draw all types of element shapes and also describe the business workflow. |
|  | Generic DDL Diagram | A generic DDL diagram is used to draw database definition elements. The Generic DDL diagrams simplify the creation of primary key, foreign key, triggers, and so forth. |
|  | Networking Diagram | A networking diagram allows a visual display of a network topology. The Networking Profile contains stereotypes for a network description. |
|  | Oracle DDL Diagram | Use the Oracle DDL diagram to draw database definition elements. This diagram is based on the Generic DDL diagram. |
|  | Robustness Diagram | The robustness diagram represents robustness analysis. It includes elements from the class diagram and actors from the use case diagram with predefined stereotypes. |
|  | Struts Diagram | A struts diagram is an extension of the UML notation. It contains the same model elements which belong to the class diagram, except they have predefined stereotypes. |
|  | Time Diagram | A time diagram is similar to a sequence diagram, except the model elements of the time diagram have predefined stereotypes. |
|  | User Interface Modeling Diagram | The user interface modeling diagrams makes it possible to build prototypes of user interfaces, connect UI mock-ups with the whole architectural model, export them as images, and create browsable reports for presentations. |
|  | WSDL Diagram | A WSDL diagram is used to draw WSDL elements. It enables you to create all the elements used in a wsdl file. |
|  | Web Diagram | A web system consists of server applications, network, communicating protocol, and the browser. |
|  | XML Schema Diagram | The purpose of this diagram is to create the structure of an xml schema file. The Diagram allows a quick drawing of xml schema elements. |
|  | Relation Map Diagram | The relation map diagrams enables reviewing and analyzing relations between the elements of the whole model rapidly. |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Main Toolbars

| Toolbar button | Title | Description |
|---|---------------------------|---|
|  | Dependency Matrix | The Dependency Matrix is a method of visualizing and representing dependency criteria. Diagrams, UML and extended UML elements serve as row and column entries. The cells in the matrix show where these elements are associated or related. |
|  | NEW! Generic Table | A Generic Table allows you to organize your elements in tabular form. This table provides a convenient way to fill in information using a spreadsheet-like tabular format instead of the limited-size fields in the element specification dialog. |
|  | Content Diagram | The purpose of the content diagram is to generate or represent a project structure (diagrams). The relations between diagrams are represented. The content table works as a table of contents of a project. |
|  | oAW Metamodeling Diagrams | The purpose of the oAW metamodeling diagram is to define a model transformation and check rules. This diagram is a simplified class diagram. |

Diagrams Navigation Toolbar

To work with diagram windows and tooltips of model elements, use the buttons on the **Diagrams Navigation** toolbar.

IMPORTANT!

If you don't see some buttons, please, check the perspective and the mode you are working on.

| Toolbar button | Title (shortcut keys) | Description |
|---|--------------------------------|---|
|  | Previous Diagram (Alt+Left) | Opens the previously opened diagram. |
|  | Next Diagram (Alt+Right) | Opens the next diagram. |
|  | ToolTips Style button | Click the ToolTips Style button and select the style of tool tips you wish to display in symbols on the diagram pane: <ul style="list-style-type: none">• Do not show – do not show tips (default).• Object name – show the symbol name and location path from the Data package in the following style: <Element Name> [<path>]• Object Documentation – show the documentation associated with the desired symbol. Note: To see the tool tips, drag the cursor over the desired symbol on the diagram. |

Opened Projects Toolbar

To switch quickly from one recently opened project to the other, use the **Opened Projects** toolbar.

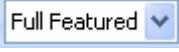
| | |
|-------------------|---|
| IMPORTANT! | If you cannot see some of the buttons, please, check the perspective and the mode you are working on. |
|-------------------|---|

| Toolbar button | Title (shortcut keys) | Description |
|---|-----------------------------|--|
|  | List of the opened projects | You will see the name and location of the currently open project. To see a list of all open projects, click the small arrow on the right side of the drop-down list box. |

Perspectives Toolbar

To change the MagicDraw user perspective, use the **Perspectives** toolbar.

| | |
|-------------------|---|
| IMPORTANT! | If you cannot see some of the buttons, please, check the perspective and the mode you are working on. |
|-------------------|---|

| Toolbar button | Title (shortcut keys) | Description |
|---|---|---|
|  | List of the available MagicDraw user perspectives | Click to change the MagicDraw user perspective. |

Validation Toolbar

To validate the created models, use the buttons on the **Validation** toolbar.

| | |
|-------------------|---|
| IMPORTANT! | If you cannot see some of the buttons, please, check the perspective and the mode you are working on. |
|-------------------|---|

| Toolbar button | Title | Description |
|---|---------------------|---|
|  | Validate | Click to open the Validation dialog box. |
|  | Run Last Validation | Click to run the model validation on options saved in the Validation dialog box. |

Diagram Toolbars

The diagram toolbar contains buttons for working with symbols on the diagram pane. Select any symbol or path on the diagram pane and the required buttons from the diagram main toolbar become active.



Figure 3 -- The fragment of the Diagram toolbars

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Diagram Toolbars

Use the diagram main toolbar to change the symbol layout, path style, symbol properties style, diagram zoom as well as symbol copy/paste, cut, or delete actions.

| Button | Title | Description |
|---|---------------------------|--|
| Layout group | | |
|  | Quick diagram layout | Apply the recommended layout tool with default options to the active diagram. Press the small arrow near the Quick Diagram Layout button to see other available layouts. |
|  | Make same width | Layout the selected shapes according to their width. After the layout, the same width will be applied to the shapes (according to the widest). |
|  | Make same height | Layout the selected shapes according to their height. After the layout, the same height will be applied to the shapes (according to the highest). |
|  | Make same size | Layout the selected shapes according to their width and height. After the layout, the same width and height will be applied to the shapes according to the widest and highest shape. |
|  | Center Horizontally | Center the selected shapes on a horizontal line. |
|  | Center Vertically | Center the selected shapes on a vertical line. |
|  | Space Evenly Horizontally | Space the selected shapes evenly. The space between the selected shapes is equally distributed on a horizontal line. |
|  | Space Evenly Vertically | Space the selected shapes evenly. The space between the selected shapes is equally distributed on a vertical line. |
| Paths Editing group | | |
|  | Remove Break Points | To remove all angles of a path, press the Remove Break Points button. |
|  | Rectilinear | To change a path style to rectilinear lines, press the Rectilinear button. |
|  | Reset Labels Positions | To reset the path labels to the default position, press the Reset Labels Position button. |
| Symbol Editing group | | |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Diagram Toolbars

| | | |
|--|--------------------------------------|--|
| | Fill Color | Press the button to apply the selected fill color or press the small arrow near the button to select another color. |
| | Set Selected Symbol Style as Default | If a new style was set, it will be applied for all newly created elements after drawing them on the diagram pane. |
| | Apply Default Symbol Style | The selected symbol style will be changed to its default style. (The default symbol style is defined in the Project Options dialog box, Symbol property styles branch). |
| | Select All of the Same Type | All shapes of the same style are selected, for example, all classes will be selected. |

View group

| | | |
|--|---|--|
| | Zoom In, Zoom Out, and Percentage drop down box | Using the view group buttons, you can change the view of the diagram by zooming it in and out. NOTE You can also change the diagram view in the Browser, Zoom panel. |
|--|---|--|

Edit group

| | | |
|--|-----------------------------|--|
| | Cut, Copy, Paste and Delete | Using the Edit group buttons, you can copy or delete symbols from the diagram pane. NOTE After a symbol is deleted from the diagram pane, it does not mean that the element is deleted from the project. |
|--|-----------------------------|--|

Browser Window

The Browser is a hierarchical navigational tool that allows you to manage your model data, including packages, components, classes, all UML diagrams, extension mechanisms, and other data.

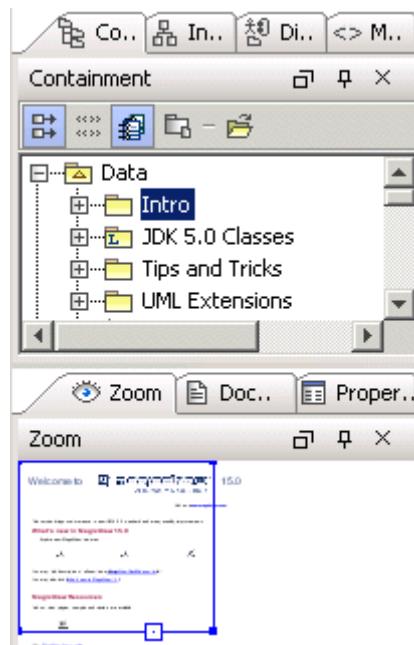


Figure 4 -- Browser window

Floating, Auto-hide and Close Buttons

To each Browser window there are added Toggle Floating, Toggle auto-hide, and Close buttons. You can move or hide a window using these buttons. See the circled buttons in the image below.



| Icon | Title | Description |
|------|------------------|---|
| □ | Toggle Floating | Press the Toggle Floating button, the window is split and you can move the window to any desired position. |
| ⊕ | Toggle auto-hide | Press the Toggle auto-hide button, the current Browser window is hidden. The tab of the hidden window is displayed on the left side of MagicDraw (in vertical position). Bring the mouse over the hidden window tab and the window is displayed. |
| × | Close | Close the current window. |
| □ | Toggle Floating | The floating window is docked again. |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Browser Window

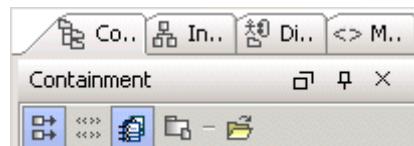
| Icon | Title | Description |
|------|------------------|--|
| | Toggle auto-hide | The auto-hide enabled window is displayed again. |

The browser consists of two parts:

1. Containment tree/Inheritance tree/Diagrams tree/Model Extensions tree/Search Results part.
2. Zoom/Documentation/Properties window.

Buttons from the Containment Tree

In the Browser, the Containment tree is active by default. To open the Containment tree, click the Containment Tree tab at the top of the Browser. The Containment tree displays a model data, and groups it in logical sets.



| Icon | Title | Description |
|------|----------------------------|---|
| | Show Full Types in Browser | To show/hide full information of the operations, attributes, and relationships in the Containment Tree. |
| | Show Stereotypes | To show/hide stereotypes near the element name. |
| | Show Code Engineering Sets | To show/hide Code Engineering Sets branch in the Browser. |
| | Show Auxiliary Resources | To show/hide modules and profiles in the Browser. |
| | Open in New Tab | A new tab with a package name and tree content will be opened in the Browser. |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Browser Window

Buttons from the Inheritance Tree

To open the Inheritance tree  , click the Inheritance Tree tab at the top of the Browser. The Inheritance tree represents the classifiers, packages, data types, stereotypes hierarchy of your project. The inheritance according to the UML Specification is shown using a generalization relationship.



| Icon | Title | Description |
|---|-----------------------|---|
|  | Show only hierarchies | The generalization hierarchies are displayed in the tree. If a classifier has no generalization relationship, it will not be visible on the tree. |
|  | Invert Tree | The hierarchy of general classifiers and children is displayed in the Inheritance tree. After inverting the tree, the classifier tree view will be changed, making the child a root classifier. |

Buttons from the Diagrams Tree

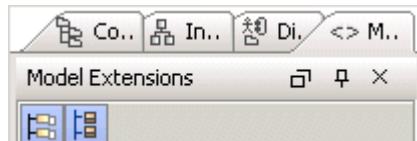
To open the Diagram tree  , click the Diagram Tree tab at the top of the Browser. The Diagram tree in the Browser represents the external structure of a diagram.



| Icon | Title | Description |
|---|-----------------------|---|
|  | Group by Diagram Type | If the Group by Diagram Type button is pressed, diagrams are listed in the packages by diagram type. |

Buttons from the Model Extensions Tree

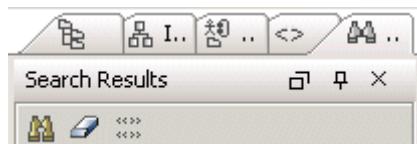
To open the Model Extensions tree , click the Model Extensions tree tab at the top of the Browser. The Model Extensions Tree contains all Stereotypes that are predefined and created manually in the project.



| Icon | Title | Description |
|---|--------------------|---|
|  | Group by Profiles | If the Group by Profiles button is pressed, groups of profiles and the elements inside them are listed. |
|  | Group by Metaclass | If the group by Metaclass button is pressed, groups of Metaclasses are displayed as packages and the elements are listed inside them. |

Buttons from the Search Results Tree

To open the Search Results tree, click the  from the **Edit** main menu, select the **Find** command. The **Search Results** tree shows results of the search.



| Icon | Title | Description |
|---|------------------|--|
|  | Find | Press the Find button and the Find dialog box opens. |
|  | Clear Results | Press the Clear Results button to remove elements from the Search Results tree. |
|  | Show Stereotypes | Press the Show Stereotypes button to display stereotypes beside the element. |

Icons of general elements

This section lists all element icons displayed in the MagicDraw Browser window. To make your icons search easier, the icons of relationships are listed in the next section “Icons of relationships” on

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Icons of general elements

page 820, and the icons used in modules/profiling mechanism are listed in the section “Icons from Modules and Profile mechanism” on page 823.

| Icon | Title | Description |
|------|-----------------------------------|--|
| | Abstract class | An abstract class is a class, which represents the conceptual set of the classifiers. In the Class specification dialog box select the Is Abstract option and a class icon changes to the icon of the abstract class. |
| | Accept Event Action | An accept event action is an action that waits for the occurrence of an event that meets the specified conditions. ^a |
| | Action | An action is a named element that is the fundamental unit of an executable functionality. The execution of an action represents some transformation or processing in the modeled system, be it a computer system or otherwise.* |
| | Activity | An activity element is created on the activity diagram creation. |
| | Activity Parameter Node | An activity final node is a final node that stops all flows in an activity.* |
| | Actor | An actor represents the roles played by the human users, external hardware, and other subjects.* |
| | Artifact | An artifact represents a physical piece of information that is used or produced by a software development process.* |
| | Artifact Instance | An instance of an artifact. |
| | Association Class | An Association Class can be seen as an association that also has class properties, or as a class that also has association properties. Not only it connects a set of classifiers, but also defines a set of features that belong to the relationship itself, not to any of the classifiers.* |
| | Attribute | An attribute is a named property of a class that describes a range of values that can be held by instances of that class. |
| | Call Operation Action | The call operation action transmits an operation call request to the target object, where it may cause the invocation of an associated behavior.* |
| | Central Buffer Node (Object Node) | An object node is an activity node that indicates an instance of a particular classifier, possibly in a particular state, may be available at a particular point in the activity.* |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Icons of general elements

| Icon | Title | Description |
|------|----------------------------|--|
| | Choice | The choice vertices, when reached, result in the dynamic evaluation of the guards of the triggers of its outgoing transitions.* |
| | Class | A class is the descriptor for a set of objects with similar structure, behavior, and relationships. |
| | Collaboration | A collaboration is represented as a kind of classifier and it defines a set of cooperating entities to be played by instances (its roles) as well as a set of connectors that define communication paths between the participating instances.* |
| | Collaboration Use | A collaboration use represents a particular use of collaboration to explain the relationships between the properties of a classifier.* |
| | Combined Fragment | A combined fragment defines an expression of interaction fragments. It is defined by an interaction operator and the corresponding interaction operands.* |
| | Comment | A comment gives an ability to display different remarks on diagrams.* It may be attached to multiple elements. |
| | Component | A component represents all kinds of elements that pertain to piecing together software applications. They can be simple files, such as DLLs or executables. |
| | Component Instance | An instance of a component that may reside on a node instance. |
| | Composite State | A composite state is a state with one region. There is a set of two or more states placed within a region. These are known as substates. The substates describe states within a state. |
| | Orthogonal State | An orthogonal state is a state with two or more regions. In each region, you may draw one life cycle with both a beginning and an end. |
| | Conditional Node | A conditional node is a structured activity node that represents an exclusive choice among some number of alternatives.* |
| | Connection Point Reference | A connection point reference represents the use of entry/exit points. |
| | Constraint | A constraint is a condition or restriction expressed in a natural language text or in a machine readable language for the purpose of declaring some of the semantics of an element.* |
| | Data Store | A data store keeps all tokens that enter it, copies them when they are chosen to move downstream. Incoming tokens containing a particular object replace any tokens in the object node containing that object.* |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Icons of general elements

| Icon | Title | Description |
|------|-----------------------------------|--|
| | Data Type | A data type is a type whose instances are identified only by their value.* |
| | Deep History | A deep history represents the most recent active configuration of the composite state that directly contains this pseudostate.* |
| | Decision Node in activity diagram | A decision node is a control node that chooses between outgoing flows. Each token arriving at a decision node can traverse only one outgoing edge. Decisions are made using guard conditions. They help protect transitions that depend on a guarding condition. |
| | Merge Node in activity diagram | A merge node has multiple incoming edges and a single outgoing edge. It is not used to synchronize concurrent flows but to accept one among several alternate flows. |
| | Deployment | A node deploys and provides a place to store and/or execute an artifact. |
| | Deployment Specification Instance | An instance of a deployment specification element. |
| | Device | A device is a physical computational resource. For example; a piece of hardware such as a desktop computer, a processor, a server, or a human work unit such as a department or team. |
| | Device Instance | An device instance is an instance of a device. |
| | Duration | A duration defines a value specification that specifies the temporal distance between two time instants.* |
| | Duration Constraint | A duration constraint defines a constraint that refers to a duration interval.* |
| | Element Value | A value specification is an abstract metaclass used to identify a value or values in a model. It may make reference to an instance or it may be an expression denoting an instance or instances when evaluated.* |
| | Entry Point | An entry point connection point reference as the target of a transition implies that the target of the transition is the entry point pseudostate as defined in the submachine of the submachine state.* |
| | Enumeration | An enumeration is a kind of data type, whose instances may be any of a number of user-defined enumeration literals. |
| | Enumeration Literal | An enumeration literal defines an extension element of an enumeration data type. |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Icons of general elements

| Icon | Title | Description |
|------|--------------------------------|---|
| | Event (all type of events) | An event is the specification of some occurrences that may potentially trigger effects by an object.* |
| | Execution Environment | An execution environment is a node that offers an execution environment for specific types of components that are deployed on it in the form of executable artifacts.* |
| | Execution Environment Instance | An instance of execution environment. |
| | Exit Point | An exit point connection point reference as the source of a transition implies that the source of the transition is the exit point pseudostate as defined in the submachine of the submachine state that has the exit point connection point defined.* |
| | Expansion Node | An expansion node is an object node used to indicate a flow across the boundary of an expansion region.* |
| | Expansion Region | An expansion region is a structured activity region that executes multiple times corresponding to the elements of an input collection.* |
| | Final State | A special kind of state signifying that the enclosing region is completed. If the enclosing region is directly contained in a state machine and all other regions in the state machine also are completed, then it means that the entire state machine is completed.* |
| | Activity Final | An activity may have more than one activity final node. The first one reached stops all flows in the activity.* |
| | Function Behavior | A function behavior is an opaque behavior that does not access or modify any objects or other external data.* |
| | Hyperlink | An icon with a small arrow image on the left-bottom side indicates that element has a hyperlink to another element/symbol, file, or web page. For more information about hyperlinks, see “Defining Hyperlinks Between Elements” on page 270. |
| | Information Flows | The InformationFlows package provides mechanisms for specifying the exchange of information between entities of a system at a high level of abstraction.* |
| | Information Item | An information item is an abstraction of all kinds of information that can be exchanged between objects.* |
| | Initial Node | An initial node is a control node at which a flow starts when the activity is invoked.* |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Icons of general elements

| Icon | Title | Description |
|------|---------------------------|--|
| | Input Pin | An input pin is a pin that holds input values to be consumed by an action.* |
| | Output Pin | An output pin is a pin that holds output values produced by an action.* |
| | Instance | An instance specification is a model element that represents an instance in a modeled system.* |
| | Interaction | An interaction is a unit of behavior that focuses on the observable exchange of information between connectable elements.* |
| | Interaction Operand | An interaction operand is contained in a combined fragment. It represents one operand of the expression given by the enclosing combined fragment.* |
| | Interaction Use | An interaction use refers to an Interaction. The InteractionUse is a shorthand for copying the contents of the referred Interaction where the interaction use is.* |
| | Interface | An interface is a specifier for the externally-visible operations of a class, component, or other classifier (including subsystems) without a specification of the internal structure. |
| | Interruptible Region | An interruptible region contains activity nodes. When a token leaves an interruptible region via edges designated by the region as the interrupting edges, all tokens and behaviors in the region are terminated.* |
| | Junction | The junction pseudo state corresponds to the merge and the static conditional branch. |
| | Lifeline | A lifeline represents an object. The lifeline runs from the beginning of the interaction, at the top of the diagram, to the end of the interaction at the bottom of the line. |
| | Literal Boolean | A literal boolean is a specification of a boolean value.* |
| | Literal Integer | A literal integer is a specification of an integer value.* |
| | Literal Null | A literal null specifies the lack of value.* |
| | Literal String | A literal string is a specification of a string value.* |
| | Literal Unlimited Natural | A literal unlimited natural is a specification of an unlimited natural number.* |
| | Loop Node | A loop node is a structured activity node that represents a loop with the setup, test, and body sections.* |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Icons of general elements

| Icon | Title | Description |
|------|-----------------------|--|
| | MetaClass | A class whose instances are classes. Metaclasses are typically used to construct metamodels. |
| | Method (Operation) | An operation is a behavioral feature of a classifier that specifies the name, type, parameters, and constraints for invoking an associated behavior.* |
| | Model | A model is an abstraction of a physical system from a particular point of view. |
| | System Boundary | A system boundary element consists of use cases related by the exclude or include (uses) relationships. |
| | Model Library | This icon depicts Model with applied Model Library stereotype. The model library is a package that contains model elements that are intended to be reused by other packages.* |
| | N-ary Association | An n-ary association is an association among two or more classes (a single class may appear more than once). |
| | Node | A node is a computational resource upon which artifacts may be deployed for execution. The nodes can be interconnected through communication paths to define the network structures.* |
| | Node Instance | A node instance is an instance of a node where the component instances may reside. |
| | Opaque Action | An opaque actions is an action with implementation-specific semantics.* |
| | Opaque Behavior | A behavior with implementation-specific semantics.* The Opaque Behavior is introduced for implementation-specific behavior or for use as a place-holder before one of the other behaviors is chosen.* |
| | Opaque Expression | An opaque expression is an uninterpreted textual statement that denotes a (possibly empty) set of values* when evaluated in a context.* |
| | Package Model Library | This icon depicts a Package with applied Model Library stereotype. The model library is a package that contains model elements that are intended to be reused by other packages.* |
| | Package | A package groups classes and other model elements together. |
| | Parameter | A parameter is a specification of an argument used to pass the information on to or out of an invocation of a behavioral feature.* |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Icons of general elements

| Icon | Title | Description |
|------|------------------------------|---|
| | Port | A port is a property of a classifier that specifies a distinct interaction point between that classifier and its environment or between the (behavior of the) classifier and its internal parts.* |
| | Primitive Type | A primitive type defines a predefined data type, without any relevant substructure.* |
| | Protocol State Machine | A protocol state machine is always defined in the context of a classifier. It specifies which operations of the classifier can be called in which state and under which condition, thus specifying the allowed call sequences on the classifier's operations.* |
| | Signal Reception (Reception) | A reception is a declaration stating that a classifier is prepared to react to the receipt of a signal.* |
| | Region | A region is an orthogonal part of either a composite state or a state machine. It contains states and transitions.* |
| | Send Signal Action | A send signal action is an action that creates a signal instance from its inputs, and transmits it to the target object, where it may cause the discharge of a state machine transition or the execution of an activity.* |
| | Sequence Node | A sequence node is a structured activity node that executes its actions in order.* |
| | Shallow History | A shallow history represents the most recent active substate of its containing state.* |
| | Shared Package | Not all module contents are visible in the using project. A module has a shared part and a private part. Only the contents of the shared part are visible in the working project. The shared packages are marked with a hand image. For more information about project partitioning, see "Project Partitioning" on page 141. |
| | Signal | A signal is a specification of send request instances communicated between objects.* |
| | Slot | A slot specifies that an entity modeled by an instance specification has a value or values for a specific structural feature.* |
| | State Machine | A state machines can be used to express the behavior of part of a system.* |
| | State | A state specifies how the object reacts to events occurring around it. |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Icons of general elements

| Icon | Title | Description |
|---|---------------------------------|---|
|  | Stereotype | A stereotype is an extension mechanism that defines a new and more specialized element of the model based on an existing element. |
|  | Structured Activity Node | A structured activity node is an executable activity node that may have an expansion into the subordinate nodes as an activity group.* |
|  | Submachine State | A submachine state specifies the insertion of the specification of a submachine state machine.* |
|  | Subsystem | A subsystem is a unit of hierarchical decomposition for large systems.* |
|  | Subsystem Instance | An instance of the subsystem. |
|  | Swimlane | Actions and subactivities can be organized into a Swimlane in the activity diagrams. The swimlanes are used to organize responsibility for actions and subactivities according to a class. |
|  | Fork Horizontal or Vertical | A fork node is a control node that splits a flow into multiple concurrent flows.* |
|  | Join Horizontal or Vertical | A join node is a control node that synchronizes multiple flows.* |
|  | Template Parameter | A template parameter exposes a parameterizable element as a formal template parameter of a template.* |
|  | Template Parameter Substitution | A template parameter substitution relates the actual parameter(s) to a formal template parameter as part of a template binding.* |
|  | Template Parameter Signature | A template signature bundles the set of formal template parameters for a templated element.* |
|  | Time Constraint | A time constraint specifies the combination of min and max timing interval values. |
|  | Time Expression | A time expression defines a value specification that represents a time value.* |
| | Time Observation | An time observation is a reference to a time instant during an execution. It points out the element in the model to observe and whether the observation is made when this model element is entered or when it is exited.* |
|  | Time Event | A time event specifies a point of time by an expression. The expression might be absolute or might be relative to some other point of time.* |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Icons of general elements

| Icon | Title | Description |
|------|-------------------|--|
| | Trigger | A trigger specification may be qualified by the port on which the event occurred.* A trigger relates an event to a behavior that may affect an instance of the classifier.* |
| | Use Case Instance | A use case instance is an instance of a use case. |
| | Use Case | A use case is a kind of behavior-related classifier that represents a declaration of an offered behavior.* |
| | Value Pin | A value pin is an input pin that provides a value by evaluating a value specification.* |
| | Variable | A variable is considered as a connectable element.* |

a. Material from the OMG UML Specification has been duplicated with permission.

Icons of relationships

| Icon | Title | Description |
|------|-------------------------|--|
| | Relations branch | Most of the relationships are included in the Relations branch in the Browser. |
| | Abstraction | An abstraction is a dependency relationship that relates two elements or sets of elements that represent the same concept at different levels of abstraction or from different viewpoints. ^a |
| | Aggregation association | An aggregation describes a special type of association designed to help cope with complexity. |
| | Assembly Connector | An assembly connector is a connector between two components that defines that one component provides the services that another component requires.* |
| | Association | An association answers to the question why two classes of objects need to know about one another. |
| | Communication Path | The communication path is a subclass of association. It specifies the relationship between nodes by defining the number of nodes that may be connected (multiplicity), and the nature of the connection, via the name of the path or a stereotype. |
| | Association Class | An association class can be seen as an association that also has class properties, or as a class that also has association properties. Not only it connects a set of classifiers, but also defines a set of features that belong to the relationship itself, not to any of the classifiers.* |
| | Call Message | A call message represents the request to invoke a specific operation. |
| | Composition | A composition is used for aggregations where the life span of the member object depends on the life span of the aggregate. |
| | Connector | A connector specifies a link that enables communication between two or more instances.* |
| | Constraint | A constraint is a condition or restriction expressed in a natural language text or in a machine readable language for the purpose of declaring some of the semantics of an element.* |
| | Control Flow | A control flow is an edge that starts an activity node after the previous one is finished.* |
| | Create Message | A create message is message designating the creation of another lifeline object.* |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Icons of relationships

| Icon | Title | Description |
|------|----------------------------------|---|
| | Delegation Connector | A delegation connector is a connector that links the external contract of a component (as specified by its ports) to the internal realization of that behavior by the component parts.* |
| | Dependency | A dependency indicates a semantic relationship between two model elements (or two sets of model elements). |
| | Deployment | A deployment is a relationship between the location and the artifact. |
| | Delete Message | A delete message is message designated to terminate another lifeline.* |
| | Direct Association | A directed relationship represents a relationship between a collection of source model elements and a collection of target model elements.* |
| | Directed Aggregation Association | An aggregation describes a special type of association designed to help cope with the complexity. A directed relationship represents a relationship between a collection of source model elements and a collection of target model elements.* |
| | Direct Composition Association | A composition is used for aggregations where the life span of the member object depends on the life span of the aggregate. A directed relationship represents a relationship between a collection of source model elements and a collection of target model elements.* |
| | Element Import | An element import is defined as a directed relationship between an importing namespace and a packageable element.* |
| | Exception Handler | An exception handler is an element that specifies a body to execute in case the specified exception occurs during the execution of the protected node.* |
| | Extend | A relationship from an extending use case to an extended use case that specifies how and when the behavior defined in the extending use case can be inserted into the behavior defined in the extended use case.* |
| | Extension | An extension point identifies a point in the behavior of a use case where that behavior can be extended by the behavior of some other (extending) use case, as specified by an extend relationship.* |
| | Generalization | A generalization is the relationship from the child element (the more specific element, such as a subclass) to the parent (the more general element, such as a super class) that is fully consistent with the first element and that provides additional information. |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Icons of relationships

| Icon | Title | Description |
|------|---------------------------|---|
| | Generalization Set | A generalization set defines a particular set of generalization relationships that describe the way in which a general classifier may be divided using specific types.* |
| | Include | An include (uses) relationship from the use case A to the use case B indicates that an instance of the use case A will also contain the behavior as specified by B. The behavior is included at the location which is defined in A. |
| | Information Flow | An information flow specifies that one or more information items circulates from its sources to its targets.* |
| | Interface Realization | An interface realization is a specialized realization relationship between a behavioral classifier and an Interface.* |
| | Link | An instance specification whose classifier is an association represents a link and is shown using the same notation as for an association, but the solid path or paths connect the instance specifications rather than the classifiers.* |
| | Manifestation | A manifestation is the concrete physical rendering of one or more model elements by an artifact.* |
| | Package Merge | A package merge is a directed relationship between two packages that indicates that the contents of the two packages are to be combined.* |
| | Message | A message is a named element that defines one specific kind of communication in an Interaction. A communication can be, for example, raising a signal, invoking an operation, creating or destroying an Instance. The message specifies not only the kind of communication given by the dispatching execution specification, but also the sender and the receiver.* |
| | Send Message | The message was generated by an asynchronous send action.* |
| | Non-navigable Association | An association with non-navigable association ends. |
| | Object Flow | An object flow is a technique used to capture how objects participate in activities and how they are affected by the activities.* |
| | Package Import | A package import is defined as a directed relationship that identifies a package whose members are to be imported by a namespace. |
| | Profile Application | A profile application is used to show which profiles have been applied to a package.* |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Icons from Modules and Profile mechanism

| Icon | Title | Description |
|------|-----------------------|--|
| | Protocol Transition | A protocol transition specifies a legal transition for an operation.* |
| | Realization | A realization is a specialized abstraction relationship between two sets of model elements, one representing a specification (the supplier) and the other represents an implementation of the latter (the client).* |
| | Component Realization | A component realization concept is specialized in the Components package to (optionally) define the Classifiers that realize the contract offered by a component in terms of its provided and required interfaces.* |
| | Reply Message | A reply message is a reply message to an operation call.* |
| | Role Binding | A role binding is a mapping between features of the collaboration type and features of the classifier or operation. This mapping indicates which connectable element of the classifier or operation plays which role(s) in the collaboration.* |
| | Substitution | A substitution is a relationship between two classifiers which signifies that the substitutingClassifier complies with the contract specified by the contract classifier.* |
| | Template Binding | A template binding represents a relationship between a templateable element and a template.* |
| | Transition | A state transition usually has an event attached to it, but it is not necessary to attach one. If an event is attached to a state transition, the state transition will be performed when the event occurs. |
| | Usage | A usage is a relationship in which one element requires another element (or set of elements) for its full implementation or operation.* |

a. Material from the OMG UML Specification has been duplicated with permission.

Icons from Modules and Profile mechanism

Module, Profile, and Shared package model elements have their own icons, which are represented in the MagicDraw Browser. You may see the icons, their titles, and descriptions in the following table.

| Icon | Title | Description |
|------|--------|---|
| | Module | If you have (or developing) a large model, which has several weakly dependent parts, it is advisable to split it into several module files. Partitioning has a package level granularity. |

12 APPENDIX I: MENUS, BUTTONS, AND ICONS

Icons from Modules and Profile mechanism

| Icon | Title | Description |
|------|--|--|
| | Profile | A Profile is a kind of Package that extends a reference metamodel. The primary extension construct is the Stereotype, which is defined as part of Profiles. ^a |
| | Profile Model Library exported as module | This icon depicts a profile with applied Model Library stereotype. A package that contains model elements that are intended to be reused by other packages.* |
| | Shared Package | Not all module contents are visible in the working project. A module has a shared part and private part. Only the contents of the shared part are visible in the working project. The shared packages are marked with a hand image. |

a. Material from the OMG UML Specification has been duplicated with permission.

For more information about working with modules and profiling mechanism, see “Project Partitioning” on page 141.

13 APPENDIX II: UML 2.3 SUPPORT

Introduction

MagicDraw always supports the latest versions of UML standards, and the newest version of MagicDraw is no exception.

The MagicDraw version 16.8 supports the changes of UML specification from version 2.2 to 2.3. These changes, including metamodel changes and notation changes, are introduced in the sections of this appendix.

For the summary of the UML metamodel changes, see section “Metamodel Changes” on page 825. This section also gives a brief description of each change.

For the UML notation changes, see section “Notation Changes” on page 827.

For information about opening models created in any version older than MagicDraw 16.8, see section “Opening up Older Models” on page 828.

Metamodel Changes

The following table summarizes the UML metamodel changes from version 2.2 to 2.3 supported in MagicDraw, starting with version 16.8, and gives some brief information about the impact of each change.

| Metamodel Change | Description/Comment |
|--|--|
| PROPERTIES ADDED | |
| Port::isConjugated:Boolean=False | Provided and required interfaces, that are derived from port's type, since now can be easily switched. To do this, set the property to <i>true</i> . The default value is <i>false</i> . |
| Action::isLocallyReentrant:Boolean=False | An action since now can begin a new concurrent execution, even if there is already another execution of the action ongoing. To do this, set the property to <i>true</i> . The default value is <i>false</i> . |
| Classifier::isFinalSpecialization:Boolean=False | A classifier since now can be a final specialization and have no specifying generalizations. To do this, set the property to <i>true</i> . The default value is <i>false</i> . NOTE: Use this property instead of the <i>isLeaf</i> property. |
| StructuredActivityNode::structuredNodeInput:InputPin[0..*] {Subsets <i>Action::input</i> } | A structured activity node (i.e., expansion region, sequence node, and loop node) since now can have input pins. NOTE: The conditional node cannot have input pins. |
| StructuredActivityNode::structuredNodeOutput:OutputPin[0..*] {Subsets <i>Action::output</i> } | A structured activity node (i.e., expansion region, sequence node, conditional node, and loop node) since now can have output pins. |

13 APPENDIX II: UML 2.3 SUPPORT

Metamodel Changes

| Metamodel Change | Description/Comment |
|--|---|
| Package::ownedStereotype:Stereotype[*] {Subsets Package::packagedElement} | A package since now can own stereotypes. NOTE: Stereotypes can be owned only in these packages, which are directly or indirectly owned in profile. |
| PROPERTIES REMOVED | |
| Profile::ownedStereotype | The property has been moved to a package's properties list. |
| SUBSETTING CHANGES | |
| <ul style="list-style-type: none"> ComponentRealization::abstraction {Subsets Dependency::client --> Dependency::supplier} ComponentRealization::realizingClassifier {Subsets Dependency::supplier --> Dependency::client} | The ends of the component realization have been reversed. The supplier is now at the arrowhead of the dependency as in every realization dependency. |
| DEFAULT VALUE CHANGES | |
| Behavior::isReentrant:Boolean=False --> True | The default value of this property has changed to <i>true</i> , meaning that the behavior by default can be invoked while it is still executing from a previous invocation. |
| MULTIPLICITY CHANGES | |
| Clause::test[*] --> [1..*] | A clause since now must have at least one test section. |
| LoopNode::test[*] --> [1..*] | A loop node since now must have at least one test section. |
| ExecutableNode::test[0..*] --> [1..*] | An executable node since now must have at least one test section. |
| WriteStructuralFeatureAction::value[1] --> [0..1] | A value for a write structural feature action since now may not be specified. It became <i>optional</i> . |
| WriteVariableAction::value[1] --> [0..1] | A value for a write variable action since now may not be specified. It became <i>optional</i> . |
| Reception::signal[0..1] --> [1] | A signal for a reception since now must be specified. It became <i>mandatory</i> . |
| DERIVATION CHANGES | |
| Connector::kind is derived | The connector's kind became automatically derived. In the past it was a settable property. |
| CONSTRAINTS ADDED | |
| A component cannot nest classifiers. | Classifiers can be owned in a component only as packaged elements. |
| A component, which is nested in a class, cannot have any packaged elements. | - |
| Stereotypes can be owned only in these packages, which are directly or indirectly owned in profile. | - |
| SPECIALIZATION CHANGES | |
| ActivityGroup specializes Element --> NamedElement | The activity group no longer specializes the element. It specializes the named element instead, meaning that the activity group can have a name. |

| Metamodel Change | Description/Comment |
|---|---|
| OTHER CHANGES | |
| <ul style="list-style-type: none"> “OpaqueBehavior::body:String[*]”+“{nonunique, ordered}” “OpaqueExpression::body:String[*]”+“{nonunique, ordered}” “OpaqueAction::body:String[*]”+“{nonunique, ordered}” | Bodies of opaque behaviors, expressions, and actions became non-unique and ordered. |
| <ul style="list-style-type: none"> “OpaqueBehavior::language:String[*]”+“{ordered}” “OpaqueExpression::language:String[*]”+“{ordered}” “OpaqueAction::language:String[*]”+“{ordered}” | Bodies of opaque behaviors, expressions, and actions became non-unique and ordered. |

For the UML notation changes from version 2.2 to 2.3 supported in MagicDraw as of version 16.8, see section 1.2 "Notation Changes".

Notation Changes

The changes in UML notation from version 2.2 to 2.3 supported in MagicDraw as of version 16.8 are as follows:

1. In the expansion region notation, the keyword <<streaming>> has been changed to <<stream>> (shown in the upper left corner of the expansion region shape, when its mode is set to "stream").



Figure 1 -- An example of the expansion region notation with the replaced keyword <<stream>>

2. In the port notation, when "isConjugated" is true for the port, the port's type is represented with a tilde "~" prefixed. Note that the provided and required interfaces are switched, when the port becomes conjugated.

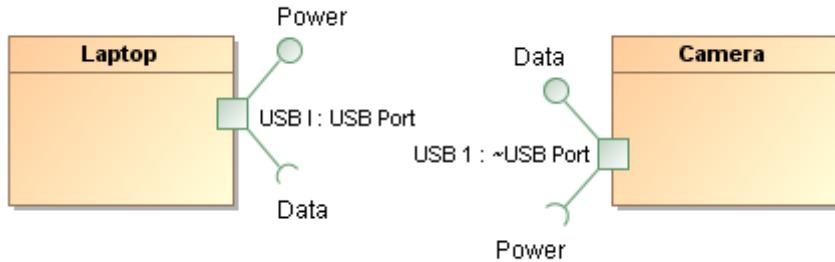


Figure 2 -- An example of the port notation. The port's type with tilde "~" indicates the conjugated port

For the UML metamodel changes from version 2.2 to 2.3 supported in MagicDraw as of version 16.8, see section 1.1 "Metamodel Changes".

Opening up Older Models

Models, which were created using any older version than MagicDraw v16.8, can be opened up with this MagicDraw version as well.

To update older models, simply open them with the MagicDraw version 16.8 and then save them. Models will be automatically converted and next time will be opened as UML v2.3 models. After the conversion to UML v2.3, all property values of these models will be persisted.

INDEX

A

About (command) 108, 112, 1089

abstract

 class 877, 898
 operation 994

Abstract (generalizable element) 343, 941

action state 731

activation bar 710, 966

active

 class 898

activity diagram 729

actor 704

 working with 876

aggregation 698

 creating 885

Align (command) 98, 1079

assigning

 classifier to classifier role 965
 classifier to collaboration 905
 classifier to instance 954
 model element to a package 386
 state to object flow state 988

association 698

 in class diagram 878
 in use case diagram 878

 n-ary 881

 navigability of 885

 with a role 893

association class 880

association end

 multiplicity of 886
 qualifier of 887
 visibility of 886

asynchronous message/stimulus 968

attribute 890

 creating new 890

defining initial value of 954

multiplicity 895

scope of 894

show only public 902

type modifier of 894

type of 893, 1040

attributes

 controlling the list of 898

 representing as association 893

 sorting of 902

 suppressing compartment 902

B

binding dependency 923

browser

 changing position 117

 changing size 117

 closing or reopening 117

 code engineering sets in 123

 Containment tree 117

 creating model elements and diagrams
 in 132

 Diagrams tree 126

 displaying full information in 118

 Documentation tab 134

 functions of 114

 Inheritance tree 128

 Model Extensions tree 129

 multiple selections 133

 sorting alphabetically 117

 structure 115

 Zoom tab 135

Browser (command) 106, 1087

C

Call (event type) 1060

Center (command) 98, 1079

INDEX

- Change (event type) 1060
 - changeable 895
 - Check Syntax (command) 101, 1082
 - class 895
 - active 898
 - creating setters and getters 495
 - defining as abstract, leaf, root and/or active 877, 898
 - description of 895
 - design patterns 900
 - inner elements of 897
 - owner of 386
 - show/hide package name 386
 - suppressing attributes 902
 - suppressing operations 902
 - class diagram 694
 - elements in 695
 - wizard 539
 - classifier
 - assign to an instance 954
 - assigning to a collaboration 905
 - assigning to classifier role 965
 - classifier role
 - assigning classifier to 965
 - in sequence diagram 710
 - Close All Diagrams (command) 107, 1088
 - Close All Projects (command) 88, 1069
 - closing
 - all opened projects 179
 - diagram 262
 - code engineering
 - checking syntax 124
 - creating new set 123
 - editing set 123
 - code generation 124
 - changing properties of 124
 - collaboration
 - assigning classifier to 905
 - collaboration diagram
 - changing numbering in 972
 - compartments of class 895
 - component 744, 907
 - diagram 742
 - instance 744
 - Component View package 178
 - composition 885
 - concurrency of operation 996
 - concurrent
 - operation 996
 - constraints
 - show-hide on symbol 360
 - showing on class 903
 - content diagram 795
 - Contents (command) 108, 111, 1089
 - Copy (command) 91, 1072
 - Copy as EMF (command) 92, 1073
 - Copy as JPG (command) 92, 1073
 - copying/cutting
 - among different projects 132
 - in the Browser 132
 - of text 295
 - symbol on a diagram 294
 - using drag and drop 293
 - CORBA IDL diagram 806, 818
 - creating
 - backup file 186
 - displaying already created relationships 576
 - elements and diagrams in Browser 132
 - new attribute 890
 - new code engineering set 123
 - new diagram 260
 - new operation 989
 - new project style 365
 - paths between shapes 279
 - creating:stereotypes 844
 - Cut (command) 91, 1072
- D
- data
 - creating several shapes of the same 275
 - Data package 178
 - data types 915
 - DDL diagram 808

INDEX

- defining
 - model elements 341
- Delete (command) 92, 1073
- deleting
 - all model elements 133
 - from the Browser 132
 - symbol or model element 277
- dependency
 - binding 923
 - permission 924
 - usage 924, 925
- deployment diagram 743
- design patterns 900
- destroying sequence object 966
- diagram
 - closing 262
 - creating 260
 - defining properties of 371
 - information table 273
 - opening 261
 - renaming 263
 - saving as image 331
- diagrams
 - activity 729
 - class 694
 - collaboration 706
 - content 795
 - CORBA IDL 806, 818
 - DDL 808
 - implementation 742
 - robustness 800
 - sequence 709
 - state 718, 724
 - use case 703
 - web 802
 - WSDL 812
 - XML Schema 814, 817, 825
- Diagrams menu 99, 1080
- display
 - related elements 575
- documentation of MagicDraw 39
- drag and drop
 - copying 293
 - from browser to diagram 293
 - multiple symbols 293
 - source code files 294
- drawing
 - more than one shape 274
 - shape 274
 - symbol from the Browser 132
- E**
 - Edit menu 90, 1071
- editing
 - code engineering set 123
- editions of MagicDraw 24
- Entertainment with UML (command) 108, 1089
- enumeration 917, 1028
- enumeration literal 919
- Environment (command) 100, 1081
- event 719
- Exit (command) 89, 1070
- extend 928
- extension point 1061
 - adding to use case 1063
- F**
 - features of MagicDraw 24
- File menu 87, 1068
- Fit In Window (command) 94, 1075
- G**
 - generalizable elements 941
 - defining as 941
 - generalization
 - grouping into tree 940
 - generate
 - code from the selected set 124
- Generate Framework (command) 101, 1082
- getter 495
- grid 307
 - size 308
 - snapping to 307

INDEX

- style 308
- visibility of 307
- Grid (command) 95, 1076
- grouping
 - diagrams in Browser 126
 - generalization paths into tree 940
 - of model elements 996
 - realizations into tree 1036
- Guarded 996
- H**
 - Help menu 108, 1089
 - HTML text 392
 - hyperlink 382
 - for model element 380
 - hyperlink: for model element 380
- I**
 - image
 - available formats 330
 - saving as 331
 - implementation diagram 742
 - Import Project (command) 88, 1069
 - include 705, 947
 - initial value
 - defining for attribute 954
 - showing on attribute 902
 - inner elements
 - of class (inserting) 897
 - installation
 - other platforms 54
 - system requirements 51
 - UNIX 54
 - using no install 54
 - Windows 54
 - instance 748
 - assign classifier to 954
 - of component 744
 - of node 745
 - interface 747
 - interface style
 - making changes to 174
 - multiple windows 175
 - Interface Style (command) 100, 1081
 - internal transition 963
- J**
 - JVM (Java Virtual Machine)
 - for Solaris 53
 - for Windows 53
- L**
 - layout 308
 - Layout menu 96, 1077
 - Leaf
 - class 898
 - Leaf (generalizable element) 941
 - lifeline 966
 - link
 - in class diagram 701
 - in collaboration diagram 707
 - in implementation diagram 751
 - link attribute 988
 - link to self 707
 - Look and Feel (command) 100, 1081
 - Look and Feel themes 175
- M**
 - Mac OS X
 - installation 54
 - Magic Draw UML on the Web (command) 108, 1089
 - MagicDraw
 - appearance of 174
 - documentation 39
 - editions 24
 - features 24
 - support 45
 - updating 70
 - Make Same Height (command) 98, 1079
 - menus 86, 1068
 - message
 - changing numbering of 972
 - defining action for 971

INDEX

- diagonal 713
- in collaboration diagram 708, 709
 - asynchronous 968
 - synchronous 969
- in sequence diagram 976, 979
- predecessor of 975
- recursive 714
- to self 713
- uninterpreted 712
- Message Window (command) 107, 1088
- model
 - development of 23
 - static structural 694
- model element
 - defining 341
 - saving as image 331
- model management 694
- moving
 - of all model elements 133
 - symbol 293
- MS Office
 - copying to 294
- multiple windows 175
- multiplicity
 - of an attribute 895
 - of association end 886
- N**
 - name
 - specifying for a shape 275
 - n-ary association 881
 - navigability
 - of association 885
- New Project (command) 87, 1068
- Next Diagram (command) 99, 1080
- node 744, 980
- note 838, 840
 - adding hyperlink to 382
 - retrieving documentation 842
 - text as HTML in 392
- note anchor 840
- numbering
 - changing 972
- numbers
 - show/hide on collaboration diagram 971
- O**
 - object flow state
 - assigning state 988
 - defining classifier for 988
- Open Project (command) 87, 1068
- opening
 - diagram 261
 - diagram from Browser 127
 - last project on startup 188
 - project 187
 - specification dialog box 342
- operation
 - concurrency 996
 - creating new 989
 - defining 989
 - defining as abstract, leaf, or root 995
 - functions of 989
 - parameter of 992
 - return type 994
 - return type of 991
 - show only public 902
 - showing signature of 901
 - visibility 995
- operations
 - controlling the list of 898
 - generation of setters and getters 897
 - managing 498
 - sorting of 902
 - suppressing compartment 902
- Options menu 100, 1081
- ordering
 - of attributes 902
 - of operations 902
- P**
 - package 696, 746, 996
 - adding inner elements 997

INDEX

changing header position 998
Component View 178
Data 178
dependency wizard 544, 548
showing assigned model elements 998
parameter
 of an operation 992
parameterized class 1056
 defining of 1056
partition 1051
paste 294
 with new data 294
Paste (command) 91, 1072
Paste With New Data (command) 91, 1072
path 259
 changing style of 280, 281
 defining properties of 371
 drawing 279
Paths (command) 92, 1073
patterns See design patterns
permission 924
predecessor 975
presentation options
 for class 900
Previous Diagram (command) 99, 1080
primitive 920
Print Active Diagram 89, 1070
Print Options (command) 89, 1070
Print Preview (command) 89, 1070
printing 334
project
 dividing into modules 211
 multiple projects 178
 opening of 187
 packages of 178
Project (command) 100, 1081
Projects (command) 89, 1070
properties
 changing code generation 124
pseudostate 719

Q
qualifier
 of association end 887
Quick Reverse (command) 101, 1082

R
realization
 grouping into tree 1036
Redo (command) 90, 1071
Refresh (command) 94, 1075
registering MagicDraw 70
relationship
 displaying on diagram 576
renaming
 diagram in Browser 127
Retrieve DB Structure (command) 101, 1082
return type
 of an operation 994
reverse
 changed files 125
 code engineering set 124
robustness diagram 800
Root
 class 898

S
Save (dialog box) 185
Save As Image (dialog box) 331, 332, 333
Save Project As (command) 87, 1068
saving 185
 as image 331
 formats 185
 in .mdf format 185
 project 185
 project as template 189
Select All (command) 92, 1073
selection 288
 all symbols of the same type 288
 multiple 288
 multiple (in the Browser) 133
 of all model elements 133

INDEX

- of all symbols 288
 - of symbol 288
 - separator 841
 - adding hyperlink to 382
 - sequence diagram
 - model elements in 710
 - overview 709
 - Sequential 996
 - setter 495
 - shape
 - defining properties of 371
 - definition 259
 - drawing more than one shape 274
 - drawing of 274
 - shortcut keys
 - assigning 176
 - Signal (event type) 1060
 - signal receipt
 - trigger event for 868, 872
 - Solaris
 - JVM 53
 - sorting
 - of attributes 902
 - of operations 902
 - Space evenly (command) 98, 1079
 - state 720, 725, 731, 1042
 - state diagram
 - model elements in 719, 725
 - state machine 719
 - statechart diagram See state diagram
 - stereotypes
 - defining properties of 371
 - show/hide on a symbol 360
 - showing on class 903
 - stereotypes:creating of 844
 - stimulus
 - changing numbering 971
 - changing numbering of 972
 - defining action for 971
 - main information about 708, 709
 - Report an Issue (command) 108, 111, 1089
 - support for MagicDraw 45
 - swimlane 1051
 - symbols 259
 - defining properties 359
 - formatting 371
 - moving 293
 - presentation of
 - constraint 295
 - note 838, 295
 - separator 295
 - text box 295
 - synchronous 969
 - system boundary 705
- T**
- tagged values
 - show/hide on a symbol 360
 - showing on class 903
 - template
 - saving as 189
 - text box 295
 - adding hyperlink 382
 - text as HTML in 392
 - text editor
 - choosing 125
 - Time (event type) 1060
 - Tip of the Day (command) 108, 111, 1089
 - toolbars 109, 1090
 - Tools menu 100, 1081
 - transition 735
 - to self 724, 729
 - type modifier 894
 - type of an attribute
 - showing full path 902
- U**
- Undo (command) 90, 1071
 - Unix
 - installation 54
 - Unlock Key (command) 108, 1089
 - updating MagicDraw 70

INDEX

usage 924, 925

use case

- adding extension point 1063
- extension point 1061

V

View menu 94, 1075

visibility

- for operation 995
- of association end 886
- showing on attribute 902
- showing on operation 902

W

web diagram 802

Window menu 106, 1087

Windows

 JVM 53

wizards 404, 537

workspace 178

WSDL diagram 812

X

XMI 186

XML Schema diagram 814, 817, 825

Z

Zoom 1:1 (command) 94, 1075

Zoom In (command) 94, 1075

Zoom Out (command) 94, 1075

Zoom To Selection (command) 94, 1075

zooming 306

 adjusting step size 307

 fit in window 306

 to maximum size 307

 to original size 306

 using Browser 135

 zoom in 306

 zoom out 306