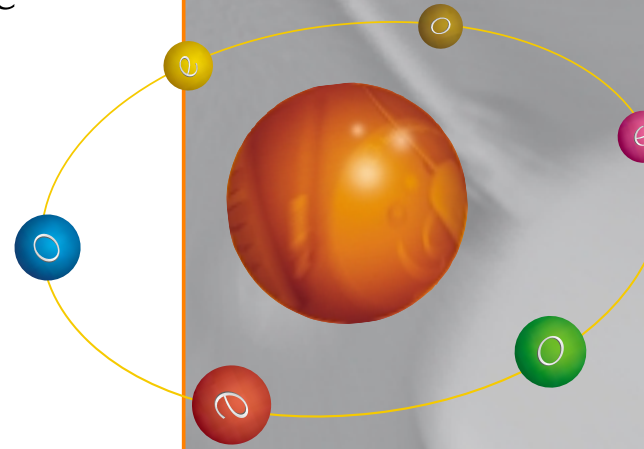


Objecteering/UML

Objecteering/UML Modeler User Guide
Objecteering/Model Dialog Boxes
User Guide

Version 5.2.2



Objecteering

Software

www.objecteering.com

Taking object development one step further

Information in this document is subject to change without notice and does not represent a commitment on the part of Objecteering Software. The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license agreement. The purchaser may make one copy of the software for backup purposes. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser's personal use, without the express written consent of Objecteering Software.

© 2003 Objecteering Software

Objecteering/UML version 5.2.2 - CODOBJ 001/002

Objecteering/UML is a registered trademark of Objecteering Software.

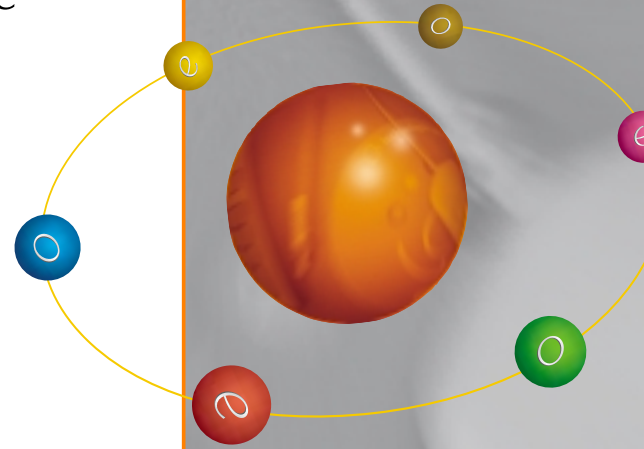
This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

UML and OMG are registered trademarks of the Object Management Group. Rational ClearCase is a registered trademark of Rational Software. CM Synergy is a registered trademark of Telelogic. PVCS Version Manager is a registered trademark of Merant. Visual SourceSafe is a registered trademark of Microsoft. All other company or product names are trademarks or registered trademarks of their respective owners.

Objecteering/UML

Objecteering/UML Modeler User Guide

Version 5.2.2



www.objecteering.com

Taking object development one step further

Objecteering

Software

Contents

Chapter 1: Introduction	
Introducing Objectteering/UML Modeler	1-3
Objectteering/UML windows	1-8
Consistency mechanisms	1-12
Glossary	1-20
Chapter 2: First Steps	
Preparation.....	2-3
Creating a UML modeling project	2-4
Creating elements in a UML modeling project	2-6
Creating a diagram	2-11
Editing a diagram	2-13
Creating elements in a diagram.....	2-16
Creating a link in a diagram	2-18
Resizing an object.....	2-19
Chapter 3: Functions of Objectteering/UML Modeler - Overview	
Launching Objectteering/UML Modeler	3-3
Creating or opening a UML modeling project.....	3-4
Receiving and upgrading UML modeling projects	3-8
The main window	3-12
The explorer	3-15
The properties editor	3-16
The console	3-18
The on-line help search engine	3-19
Diagrams.....	3-21
Work products	3-23
Macros	3-25
Editing UML Modeler configuration	3-26
Transferring elements between UML modeling projects	3-28
Saving a user work session	3-29
Saving your model context	3-31
The search function	3-32
Read-only mode.....	3-33
Removable consistency checks	3-34
Obligatory consistency checks	3-36

Chapter 4: Functions of Objectteering/UML Modeler - Detailed View	
UML Modeler menus	4-3
UML Modeler tools	4-8
Explorer functions	4-16
Browsing in the explorer	4-19
Structural element creation icons	4-22
Modifying elements	4-25
Properties editor functions	4-26
Terminal element creation icons	4-34
Visualizing messages in the console	4-36
Using the on-line help search engine	4-38
Working with Objectteering/UML macros	4-42
UML Modeler parameter sets	4-58
Using the search function	4-66
Read-only mode	4-83
Shortcuts	4-87
Chapter 5: Graphic Editors - General Principles	
Overview of the Objectteering/UML graphic editors	5-3
Creating a graphic element	5-6
Creating links	5-11
Redrawing links	5-13
Handling graphic elements	5-17
Modifying graphic elements	5-21
Masking and showing elements	5-22
Context menus for a diagram	5-25
Chapter 6: Graphic Editors - Detailed View	
Working with graphic elements	6-3
The "Modify" command on a diagram	6-8
Context menu on a diagram element	6-11
The "Graph" menu	6-14
The "Help" menu	6-18

Chapter 7: Specific graphic editors	
Class diagram	7-3
Class diagram - Behavior of graphic elements	7-6
Managing class attributes and operations	7-18
Deployment diagram	7-25
Deployment diagram - Behavior of graphic elements	7-27
Deployment instance diagram	7-31
Deployment instance diagram - Behavior of graphic elements	7-34
Object diagram	7-38
Object diagram - Behavior of graphic elements	7-40
Sequence diagram	7-42
Sequence diagram - Behavior of graphic elements	7-45
Collaboration diagram	7-49
Collaboration diagram - Behavior of graphic elements	7-51
Use case diagram	7-53
Use case diagram - Behavior of graphic elements	7-55
State diagram	7-57
State diagram - Behavior of graphic elements	7-59
Activity diagram	7-69
Activity diagram - Behavior of graphic elements	7-71
Chapter 8: Methodological Hints	
The Model/Instance approach	8-3
Flow Diagrams and DataFlows	8-7
Using flow diagrams	8-10
Examples of Flow Diagrams	8-13
Index	

Chapter 1: Introduction

Introducing Objectteering/UML Modeler

Introduction

Welcome to the *Objectteering/UML Modeler* user guide!

Objectteering/UML Modeler is the central tool of the Objectteering/UML CASE tool and provides model and diagram creation and editing facilities, help with model construction and consistency check functions.

This user guide describes *Objectteering/UML Modeler* editing facilities. It presents an overview of *Objectteering/UML Modeler* services, provides "*First steps*" which we recommend to every new user, and then describes in detail the editing services provided.

Every model element represented in *Objectteering/UML Modeler* has a dedicated dialog box, described in the *Objectteering/Model Dialog Boxes* user guide.

Objectteering/UML windows

Objectteering/UML includes the following windows:

- ◆ a *main window*, which centralizes Objectteering/UML services
- ◆ *graphic editors*, which present diagrams and are used to graphically display models and model elements, and to create, modify or destroy graphic elements
- ◆ *model explorers*, which allow you to browse a model both hierarchically and in detail. A model can be entirely created and edited using this tool.
- ◆ a *properties editor*, containing a number of tabs, each with a specific function. For example, the "*Diagrams*" tab is used to display the icons you will need to create diagrams in Objectteering/UML, as well as to present information on those diagrams which already exist in the modeling element selected.
- ◆ a *console*, which displays execution messages, errors and warnings

When Objectteering/UML is launched, the main window appears. In this window, the user can either create a new UML modeling project or open an existing one.

Once a modeling project has been created or opened, the main window displays an explorer, a properties editor and a console. When a new modeling project is created, a class diagram is automatically created and opened on the UML model root. As many explorers and graphic editors as you like can be opened from the main window.

General ergonomics

Objectteering/UML exists in native source code, on both UNIX and Windows platforms.

Objectteering/UML ergonomics on PC and UNIX are very similar. The main explorer, the console and the properties editor are dockable windows, which can be positioned and docked wherever you wish within the Objectteering/UML workspace, or even outside this workspace. Other windows, such as Objectteering/UML graphic editors, cannot be dragged over these dockable windows.

For further information on dockable windows, please refer to the "*The main window*" section in chapter 3 of this user guide.

General view of Objectteering/UML on PC

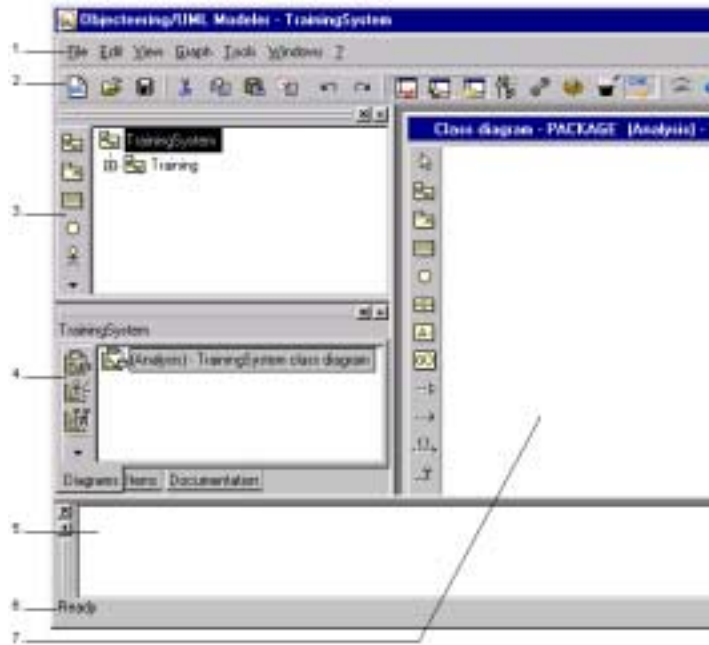


Figure 1-1. General view of Objectteering/UML on PC

Key:

- 1 - Menu bar
- 2 - Tool bar
- 3 - Main explorer
- 4 - Properties editor
- 5 - Console
- 6 - Status bar
- 7 - Graphic editor

Objecteering/UML modules

Objecteering/UML Modeler is the principal module of the Objecteering/UML CASE tool, and is essential for all services linked to the use of UML models. The Objecteering/UML CASE tool provides a large number of complementary modules, which all exploit a UML model for a specialized need. For example, the generation of C++ or Java code, documentation generation and automatic design patterns are all in specific modules. A module is selected through the "Modules" window (see Figure 1-2). When a module is selected within a UML modeling project, it provides menus, icons and specialized annotations, specific to the module in question.

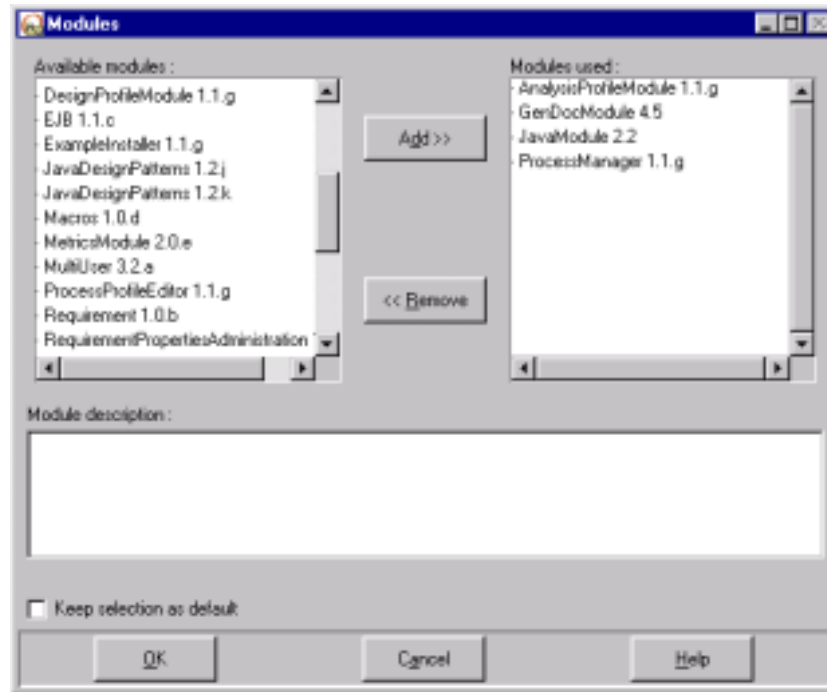


Figure 1-2. Module selection window

The *Wizard/Tools module* is always present. This module cannot be deselected, since it is an integral part of the Objectteering/UML tool.

The *Objectteering/Macros* module can be selected to provide you with a number of standard Objectteering/UML macros, which can be run on your model elements. This module can also be used to create your own macros.

When certain modules are selected, a tab specific to these modules automatically appears in the properties editor (for example, for the *Objectteering/Documentation*, *Objectteering/Java*, *Objectteering/C++* and *Objectteering/Visual Basic* modules). These tabs are designed to make it even easier to enter elements relevant to the module in question, such as summary and description notes for documentation generation.












Objectteering/UML windows

Main window

Once the UML modeling project has been created or opened, the main window appears. The main window centralizes the commands related to information management, general purpose functionalities and UML modeling project configuration. When the main window is opened, the main explorer, the console and the properties editor are displayed. These windows are dockable.

Note: Please note that for new UML modeling projects, a class diagram is automatically created and displayed.

The main window provides Objectteering/UML's general services:

- ◆  creating a new UML modeling project
- ◆  opening an existing UML modeling project
- ◆  saving the user model
- ◆  showing and hiding the console
- ◆  showing and hiding the principal explorer
- ◆  showing and hiding the properties editor
- ◆  launching a new explorer
- ◆  modifying the configuration of the current UML modeling project
- ◆  importing modules into the current UML modeling project
- ◆  activating/deactivating Objectteering/UML process wizards
- ◆  activating/deactivating removable consistency checks

The graphic editors



Graphic editors are used to display diagrams, which are created in the "Diagrams" tab of the properties editor. Firstly, a diagram is created for a model element such as a package, a class or a state machine. A graphic editor is then automatically opened for the newly created diagram.

Graphic editors allow you to:

- ◆ manage the editing of several diagrams
- ◆ graphically represent model elements
- ◆ create a model element at the same time as its graphic representation
- ◆ modify a model element (name, properties, descriptions)
- ◆ modify the graphic representation of a model element (size, positioning, color)
- ◆ show the graphic representation of a model element (showing)
- ◆ hide the graphic representation of a model element (masking)
- ◆ destroy a model element and its graphic representation
- ◆ print a diagram

The explorer



The explorer contains the UML modeling project and allows you to:

- ◆ browse in order to visualize all modeling project components, that is to say, all model elements
- ◆ construct a model
- ◆ create and edit graphic elements

The explorer is used to browse models and to work with model elements.

The properties editor



The properties editor contains a number of tabs, used to facilitate access to specific information:


- ◆ the "*Items*" tab displays icons used to create terminal elements, such as links, notes and tagged values, as well as displaying the terminal elements which already exist for the selected model element
- ◆ the "*Diagrams*" tab provides the icons used to create the different Objectteering/UML diagrams, as well as presenting information on those diagrams which already exist on the selected model element.
- ◆ the "*Java*", "*C++*", "*Visual Basic*" and/or "*Documentation*" tabs are present where the *Objectteering/Java*, *Objectteering/C++*, *Objectteering/Visual Basic* and/or *Objectteering/Documentation* modules have been selected in the current UML modeling project. These tabs facilitate the entry of information specific to the generation in question.

The console



The console provides a trace of operations undertaken in Objectteering/UML, for example the opening and printing of a document. It also provides error messages or warnings when operations are incorrectly executed.

The user can:

- ◆ clear the console, by clicking on the  "*Clear console*" icon or activating the menu option
- ◆ move around in the operations trace using the scroll bars
- ◆ save the contents of the console in a file, using the "*Save the console*" menu option (the file name is an Objectteering/UML configuration parameter)

Diagrams

For each model element, one or several diagrams may be associated. For example, (as shown in Figure 1-3), several different kinds of diagram can be created in a package: class diagram, use case diagram, sequence diagram, and so on. For further information on diagrams associated with specific model elements, please refer to the "Diagrams" section of chapter 3 of this user guide.

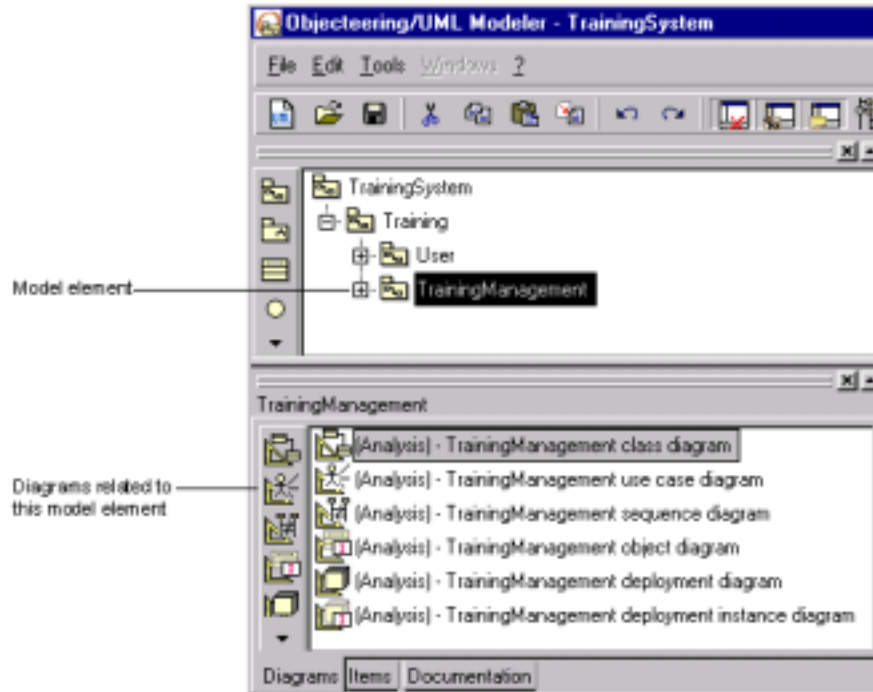


Figure 1-3. Different diagrams related to a model element

Consistency mechanisms

Presentation

Objectteering/UML Modeler provides the following powerful consistency mechanisms:

- ◆ assisted data entry: help during modeling, by providing the list of model elements that can be linked to an edited element
- ◆ consistency management: a guarantee that a model is kept consistent with regard to any changes made to any aspect: modifications made to a model view (graphic editors, the properties editor or explorers) automatically update other open views. For example, a modification of the name of a class will update the name displayed in the properties editor, as well as every diagram which includes the class and every element which refers to that class (operation parameters, instances, and so on).
- ◆ consistency checks: a guarantee that the model entered is consistent. For every modification, it automatically checks the validity of the entry with regard to the rest of the model.

Removable consistency checks

As demonstrated below, the Objectteering/UML CASE tool provides the user with powerful consistency checks, which guarantee the quality and coherence of the model produced.

However, the user may, in some modeling situations, prefer to have a certain degree of flexibility with regard to those consistency checks applied to his model, and for this reason, certain Objectteering/UML consistency checks are removable. For further information on removable consistency checks, please refer to the "*Removable consistency checks*" section in chapter 3 of this user guide.

Assisted data entry

Objectteering/UML Modeler provides a mechanism which helps the user, by listing those elements which may be selected, during model entry phases. This mechanism carries out more than 200 consistency checks in real time on models, which allows you to guarantee the high quality of your models.

The assisted data entry mechanism means that you can avoid the following problems:

- ◆ the entry of the same name several times
- ◆ the manual updating of identical names during modifications
- ◆ the entry of inconsistent names

Examples: An operation parameter's class is chosen from a list, which provides all the classes accessible in the given context. The operation associated with a state transition is chosen from a list, which provides all the operations capable of realizing the transition.

Example of assisted data entry

We are now going to create the non primitive classes C1 and C2 in a package, and then add an attribute to C1, using the explorer (as shown in Figure 1-4).

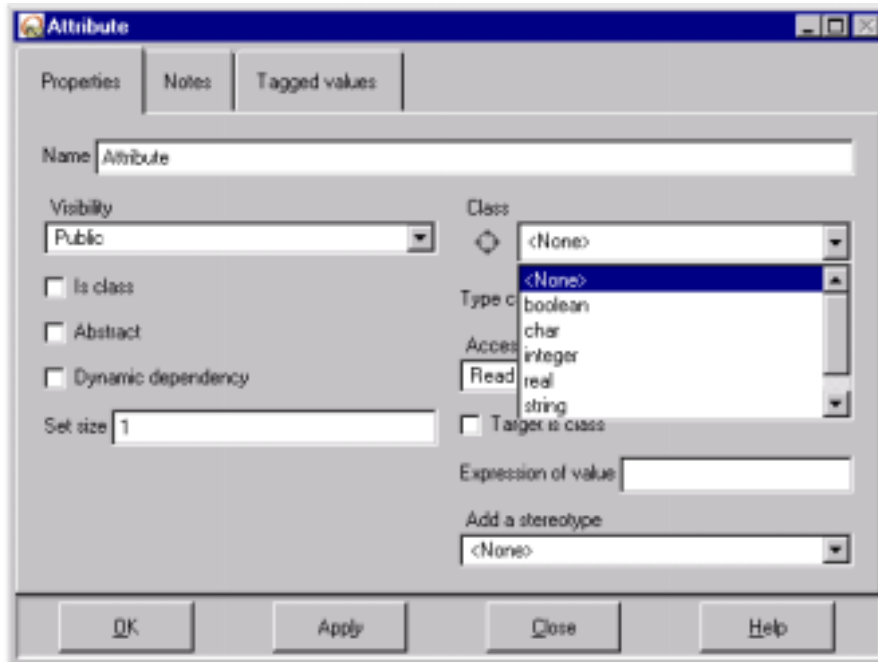


Figure 1-4. The "Attribute" dialog box - List of the possible classes

Result: In the "Attribute" dialog box, shown in Figure 1-3, (for further details on this dialog box, please refer to the "Attribute dialog box" section in chapter 3 of the *Objecteering/Model Dialog Boxes* user guide), a list of classes possible for the attribute is provided. Predefined primitive classes (integer, string, etc.) figure in this list, but not the C1 or C2 classes.

This result is logical, since it is recommended that only primitive classes can be attributes of other classes.

We are now going to define the C2 class as being as "*primitive*", and carry out the operation of creating an attribute in C1 once again (as shown in Figure 1-5). The C2 class then appears in the list of classes which may be chosen as attribute.

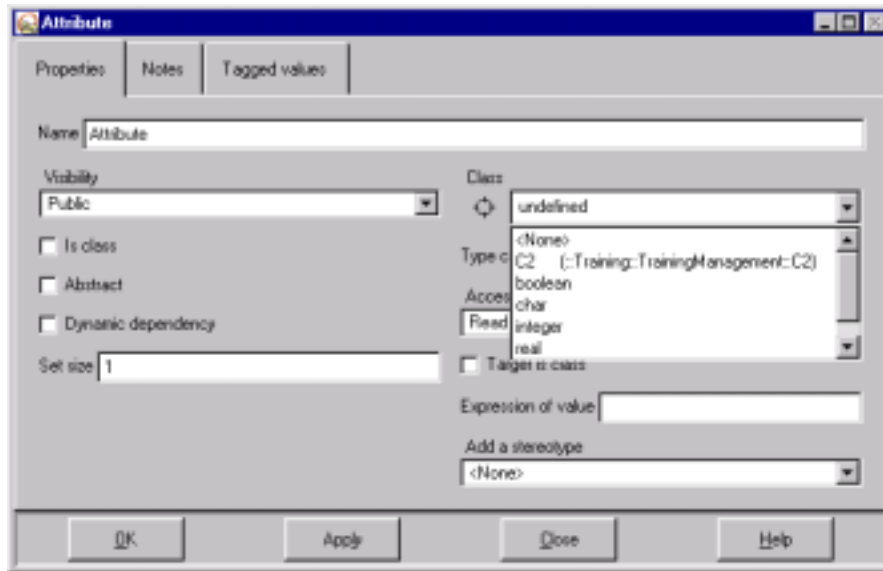


Figure 1-5. The "Attribute" dialog box containing the "C2" class in the list of possible classes for the attribute

Note: C2 is now available.

Consistency management

All the model elements in all the editors are permanently consistent. For example, the name of an operation can appear in a class, in a state diagram, in a sequence diagram, in the properties editor and in the explorer. No matter how many times it appears or how many times it may be modified, it is always consistent.

You will notice that the explorer shows all the operations created with the same names and signatures. The principle of consistency is dealt with here.

Consistency management: Example of a model update

We are now going to change the name of a class in a class diagram. It will be automatically updated in the explorer.

Figure 1-6 below presents a class diagram created in a package. This package contains the "Tea" class, whose name we are going to change to "OrangeJuice".

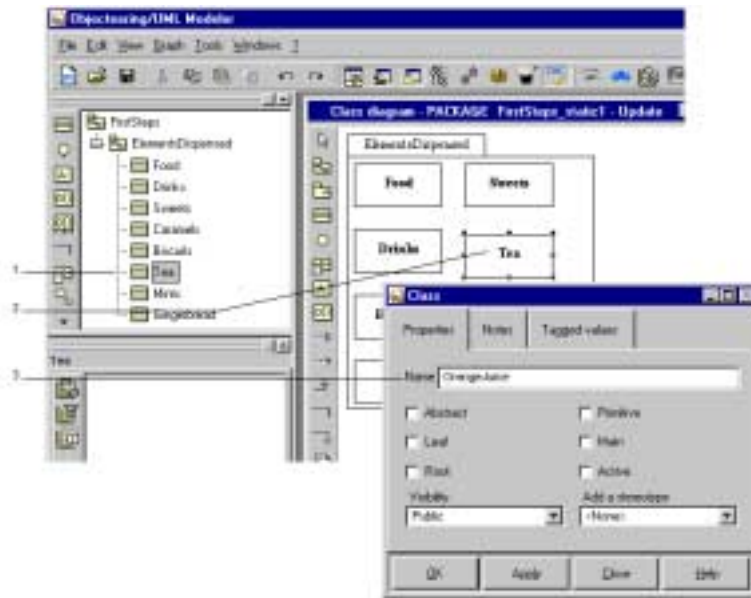


Figure 1-6. Automatically updating the name of the class

Steps:

- 1 - Observe the "Tea" class in the explorer.
- 2 - Observe the "Tea" class in the editor.
- 3 - Open the class dialog box in the explorer or in the class diagram, either by right-clicking on the class and selecting the "Modify" option or by double-clicking on the class, and change the name of the class from "Tea" to "OrangeJuice".

Chapter 1: Introduction

The modification of the name of the "Tea" class, which has been changed to "OrangeJuice", has been taken into account in the class diagram and in the explorer (as shown in Figure 1-7).

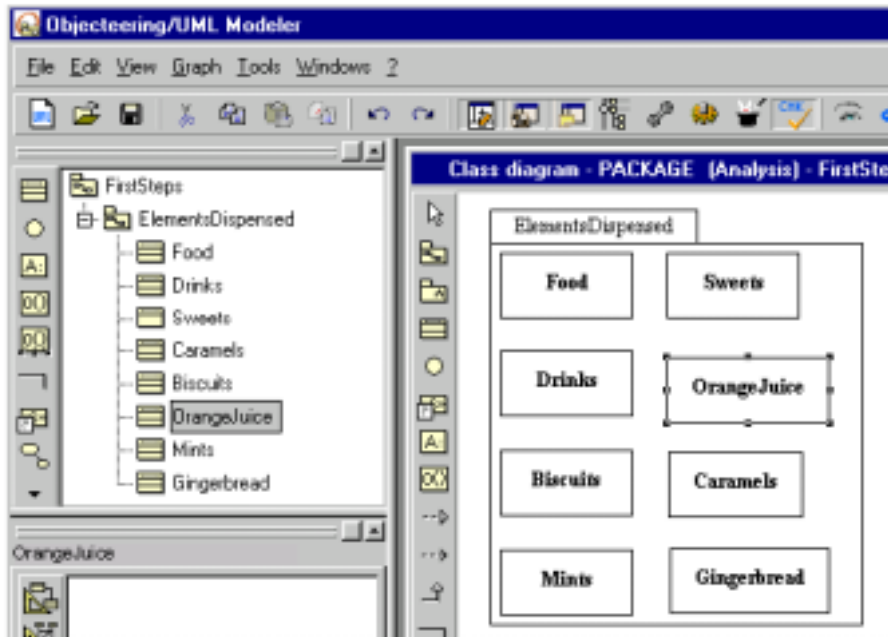


Figure 1-7. Automatic updating in the explorer and in the class diagram

Consistency checks: Example of a forbidden case scenario

Objecteering/UML carries out more than 200 consistency checks on a model in real time. This allows the user to guarantee a high level of quality for models entered in the CASE tool.

Figure 1-8 presents an example of a forbidden operation. The creation of the association between C1 and "primitive" C2, or the change in C2's status (non elementary), the association having already been defined, are actions which are forbidden by Objecteering/UML.

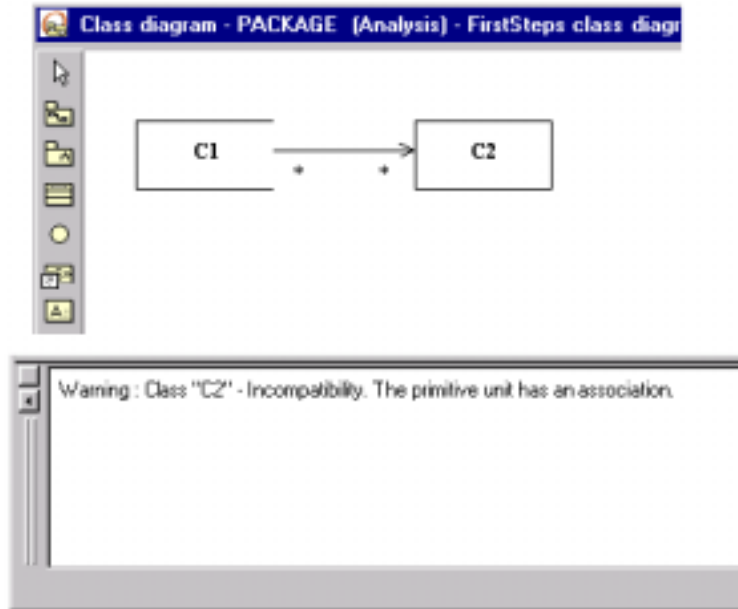


Figure 1-8. Warning message in the console

Glossary

Objectteering/UML terms

Administration: Tool through which operations such as the configuration, creation or repair of your model can be carried out.

Composition tree: The hierarchy which represents the entire UML modeling project at all its levels, including information on, for example, packages, classes and operations.

Configuration: A UML modeling project environment, in terms of selected modules and parameter values given to these modules.

Consistency checks: Mechanisms which provide assisted data entry, consistency management and consistency check functions, used to guarantee model consistency.

Console: Window which provides a trace of operations undertaken in Objectteering/UML, such as the opening and printing of a document, error and warning messages and information on code generation and import administration.

Diagram: The graphic representation of a collection of model elements.

Dialog boxes: The window in which model element information is entered and modified.

Dockable windows: Windows which can be positioned and docked wherever you wish within the Objectteering/UML workspace, or even outside this workspace. Other windows, such as Objectteering/UML graphic editors, cannot be dragged over these dockable windows.

Enterprise Edition: This is the complete multi-user, multi-project version of Objectteering/UML.

Explorer: One of the main Objectteering/UML windows, used to browse a UML modeling project's elements.

Graphic editor: The window used to display and modify diagrams.

Main window: The main window contains the editor's general services, such as launching a new explorer, importing elements from other UML modeling projects or modifying the configuration of the current UML modeling project.

Menu bar: Bar which displays the available menus.

Model element: A specific type of modeling unit, for example, a package or a class, etc.

Module: Group of services, independently packaged, that can be selected in the *Objectteering/UML Modeler*. For example, *Java*, *Documentation* and *Metrics* are examples of modules. A module is a group of UML profiles that can be defined through the *Objectteering/UML Profile Builder* tool (see also the *UML/Objectteering Profile Builder* user guide).

Personal Edition: This is the stand alone version of Objectteering/UML, dedicated to UML modeling.

Professional Edition: This edition of Objectteering/UML provides all the modeling and generation features of the Enterprise Edition, but does not support multi-user teamwork.

Properties editor: Contains a number of tabs, each with a specific function. For example, the "*Diagrams*" tab is used to display the icons you will need to create diagrams in Objectteering/UML, as well as to present information on those diagrams which already exist in the modeling element selected.

Read-only mode: Mode in which model elements cannot be added, modified or deleted.

Removable consistency checks: Consistency checks which may be activated or deactivated by the user, according to his modeling context.

Status bar: Bar which provides complementary information on the element or operation in question.

Structural elements: Elements found in the explorer.'

Sub-system: A sub-system is a grouping of model elements, which represents a behavioral unit in a physical system. A sub-system provides interfaces and has operations.

Terminal elements: Elements which cannot be decomposed, generally created and located in the "*Items*" tab of the properties editor.

UML model root: The original package to which all other elements will be added. This corresponds to the UML notion of a model.

UML modeling project: Work space for building and using models.

UML profiling project: Work space for customizing Objectteering/UML through the *UML Profile Builder* tool.

UML Profile: A UML profile represents a certain angle of vision on a model, for a functional purpose. UML profiles can structure several UML usage domains, and group tagged value definitions and stereotypes.

Work product: A product produced for a specific use, for example, documentation generation or code generation.

Objectteering/UML diagrams

Activity diagram: An activity diagram defines an extended view of a state machine package.

Class diagram: The class diagram allows you to present the internal structure of an element and its relationships with other elements.

Collaboration diagram: The collaboration diagram allows you to present exchanges of messages between roles.

Deployment diagram: The deployment diagram is used to represent the physical architecture of the system.

Deployment instance diagram: The deployment instance diagram is used to present a particular instance of deployment.

Object diagram: The object diagram is used to present a set of class instances with their links and the messages exchanged.

Sequence diagram: A sequence diagram is used to show how different objects cooperate.

State diagram: A state diagram allows you to describe the manner in which objects react to events.

Use case diagram: A use case diagram allows you to describe the most important services rendered by the system.

Chapter 2: First Steps

Preparation

Introduction

These rapid first steps will allow you to become familiar with the Objectteering/UML CASE tool, most notably with the following operations:

- ◆ *creating a UML modeling project*
- ◆ *creating elements in the explorer*
- ◆ *creating a diagram*
- ◆ *creating and manipulating objects in a diagram*

The demonstration constantly refers you to the on-line help, which you can always call up from the tool via the ever-present "*Help*" or "?" buttons.

Launching the tool

Before launching Objectteering/UML, it must have been installed (please refer to chapter 2, "*Installation*", of the *Objectteering/Introduction* user guide).

Environment ...	Action required ...
Windows 95/98/2000/NT4	Select " <i>Objectteering UML Modeler</i> " from the " <i>Start/Programs/Objectteering</i> ", or double-click on the UML Modeler icon in your desktop
UNIX	Run the objing command.

Creating a UML modeling project

Now that we have launched the tool, we are going to create a new UML modeling project (as shown in Figure 2-1).

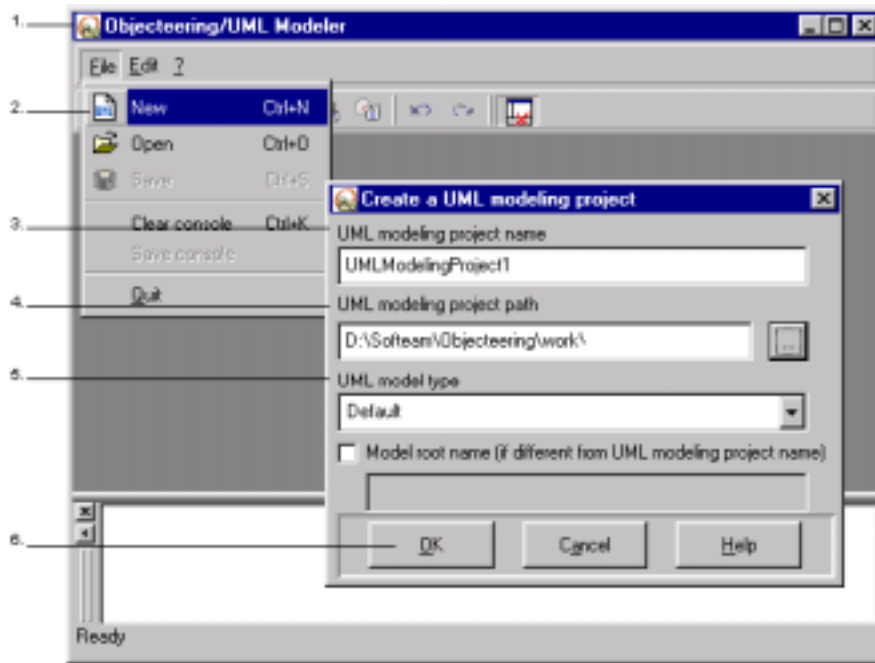




Figure 2-1. Creating a UML modeling project

Steps:

- 1 - Launch the  *Objectteering/UML Modeler* tool by clicking on the *UML Modeler* icon in your desktop. The window shown in Figure 2-1 will then appear.
- 2 - Click on the "*File/New*" menu. The "*Create a UML modeling project*" window will then open.
- 3 - In the "*UML modeling project name*" field, enter the "*NewProject*" name.
- 4 - In the "*UML modeling project path*" field, enter the path of the directory where the new UML modeling project is to be created. You may also use the  icon to open a file browser through which you can select your UML modeling project path.
- 5 - In the "*UML model type*", select a type for the modeling project which is to be created. For example, by selecting the "*DefaultJava*" type, your Objectteering/UML working environment will be automatically configured for Java development with the *Objectteering/Java* and *Objectteering/Design Patterns for Java* modules. Here, we are going to select the "*Default*" UML model type.
- 6 - Confirm by clicking on the "*OK*" button.

Note: By default, the name of the UML model root which appears in the explorer is the same as the name of the UML modeling project itself (in our example, "*NewProject*"). If you should wish to give the UML model root another name, you should simply check the "*Model root name (if different from UML modeling project)*" tickbox, which will then allow you to enter a different name directly in the field below.

Result

The main window appears, containing your new UML modeling project in the explorer, in which we are going to be able to create elements and diagrams. The console, the properties editor and an automatically created class diagram also appear.

Creating elements in a UML modeling project

Working in the explorer

From the explorer (shown in Figure 2-2), we are going to create several elements in our UML modeling project (packages, classes, operations, attributes, etc.) and then make certain modifications to them.

Note: When an element is modified in the explorer, modifications are automatically taken into account in the graphic editors, thanks to the Objecteering/UML consistency mechanisms.

Creating a package


When a new UML modeling project is created and Objectteering/UML launched, a package associated with the UML modeling project, and which has the same name, appears (in our example, the "NewProject" package). This package is the UML model root. To create a new package, simply carry out the steps shown in Figure 2-2.



Figure 2-2. Creating the "P1" package in the package representing the UML model root of the "NewProject" UML modeling project

Steps:

1 - Select the "NewProject" package.

2 - Click on the  "Create a package" icon, and enter the name of the "P1" package over the highlighted text which says "Package". "Package" is the default name assigned automatically by Objectteering/UML.

Creating a sub-system


A sub-system is a kind of package, stereotyped <<sub-system>>, which represents an independent part of the system being modeled. Sub-systems are an important feature of the component-based approach. Sub-systems are groups of model elements, which represent a behavioral unit in a physical system.

To create a sub-system in a package, following the steps shown below (Figure 2-3).



Figure 2-3. Creating a sub-system

Steps:

- 1 - Select the "P1" package.
- 2 - Click on the  "Create a subsystem" icon.
- 3 - Enter the name of the "S1" sub-system over the highlighted text which says "subsystem". "subsystem" is the default name assigned automatically by Objectteering/UML.

Creating classes

We are now going to create classes "C1" and "C2" in the newly created "S1" sub-system (see Figure 2-4).

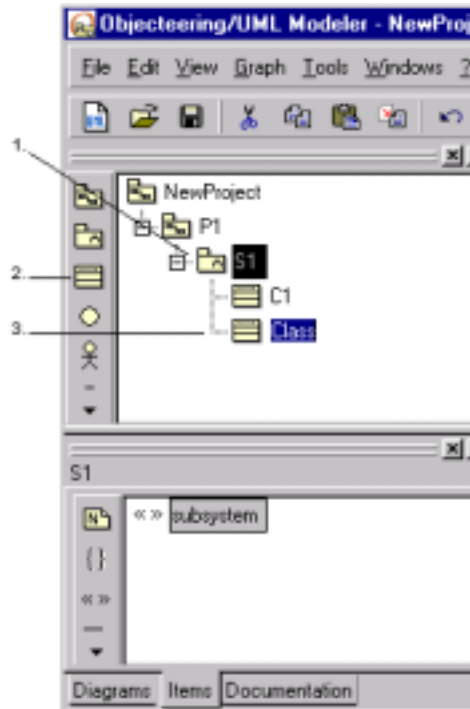



Figure 2-4. Creating the "C1" and "C2" classes in the "S1" sub-system

Chapter 2: First Steps

Steps:

- 1 - Select the "S1" sub-system.
- 2 - Click on the  "Create a class" icon.
- 3 - Enter the "C1" name over the highlighted text which says "Class". "Class" is the default name assigned automatically by Objectteering/UML. Continue by creating the "C2" class.

Most model elements (*packages, sub-systems, classes, operations, parameters, associations, etc.*) can be created in this way.

Note: If you press "Return" after you have created a class, you will see that another class is automatically created. This is the continuous entry creation mode, which is available on all model elements. To exit this mode, simply click elsewhere in the explorer.

Creating a diagram

Creating a class diagram

Various elements can have associated diagrams (packages, classes, etc.). These diagrams can be created through the "Diagrams" tab in the properties editor.

Note: When a new UML modeling project is created and Objecteering/UML launched, a class diagram is automatically created and opened.

We are now going to create a class diagram from sub-system "S1" (as shown in Figure 2-5).

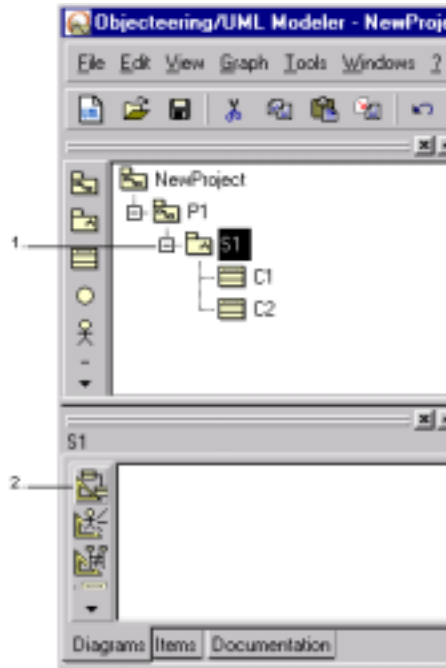



Figure 2-5. Creating a class diagram in the "S1" sub-system

Chapter 2: First Steps

Steps:

- 1 - Select the "S1" package in the explorer.
 - 2 - Select the "Diagrams" tab in the properties editor and click on the  "Create a class diagram" icon. The newly created class diagram then opens automatically.
-

Editing a diagram

Showing elements in a diagram

In the package's class diagram editor, the graphic elements contained in the class diagram are "masked". The "Show" operation allows the user to display in a graphic editor elements which exist in a model (as shown in Figure 2-6).

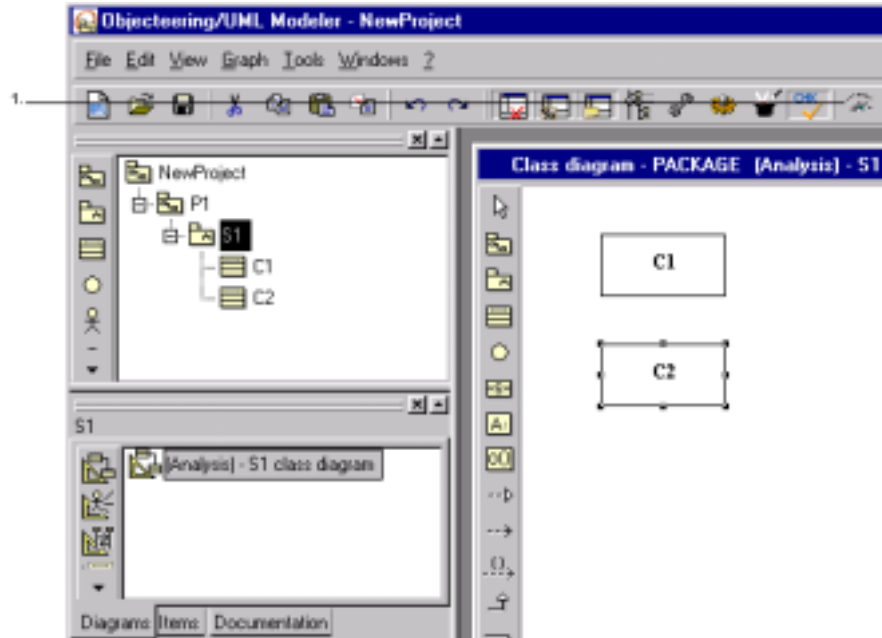



Figure 2-6. Showing classes "C1" and "C2" in sub-system "S1"

Steps:

- 1 - Click on the  "Show contents" icon. All the elements contained in the package then appear.

Masking elements in a diagram

We are now going to mask the "C1" class (as shown in Figure 2-7).

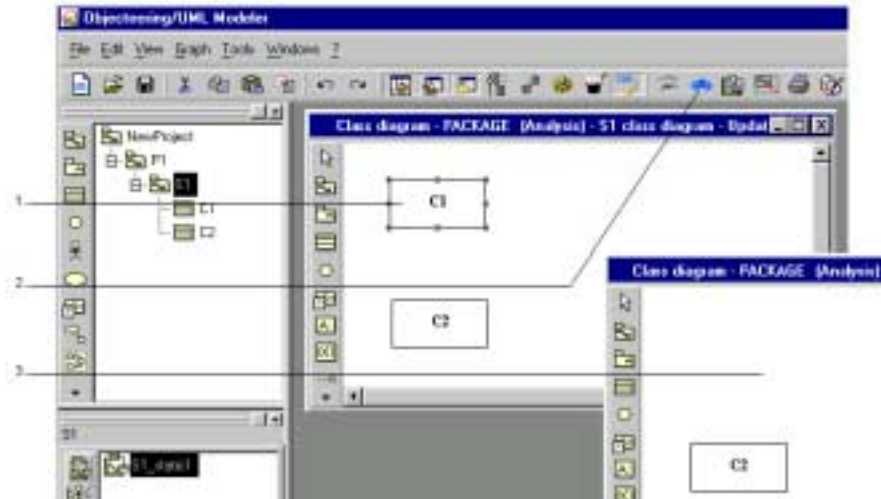



Figure 2-7. Masking the "C1" class

Steps:

- 1 - Select the "C1" class in the diagram.
- 2 - Click on the  "Mask" icon in the menu shortcut bar.
- 3 - The "C1" class is no longer visible. However, please note that it has not been discarded and still exists in the explorer. Continue by masking the "C2" class in the same way.

Showing an element using the "drag and drop" function

We will now show the "C1" and "C2" classes, using the "drag and drop" function (as shown in Figure 2-8).

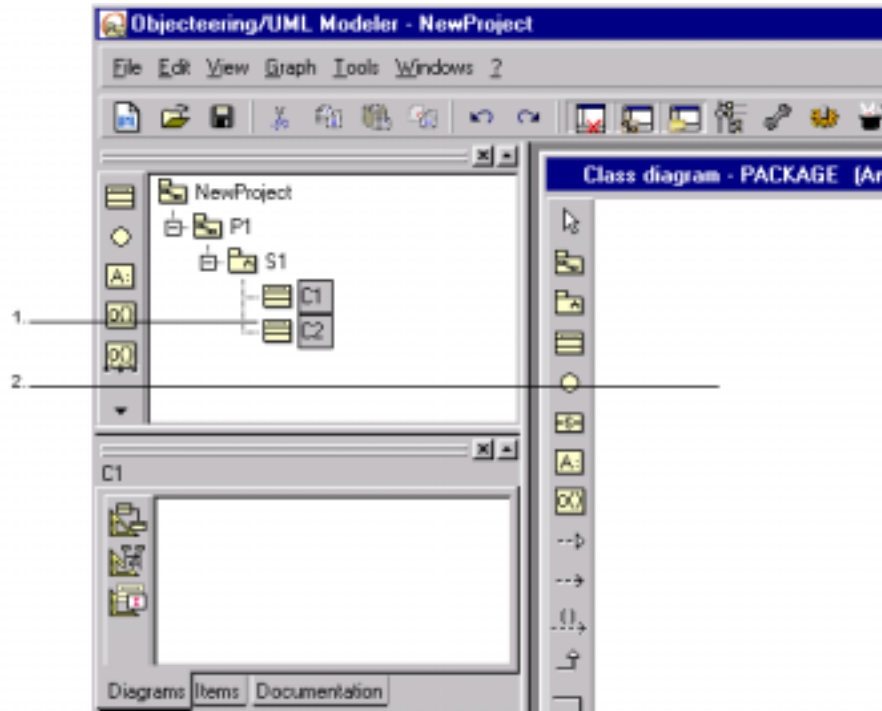


Figure 2-8. Showing classes "C1" and "C2" using the "drag and drop" function

Steps:

- 1 - Select classes "C1" and "C2" in the explorer.
- 2 - Drag the elements into the graphic editor, holding down the left mouse button. The "C1" and "C2" classes are now visible again in the graphic editor.

Creating elements in a diagram

Creating an operation

An operation, like a class, can be created either in the explorer or in a graphic editor.

We are now going to create an operation in the "C1" class (as shown in Figure 2-9).

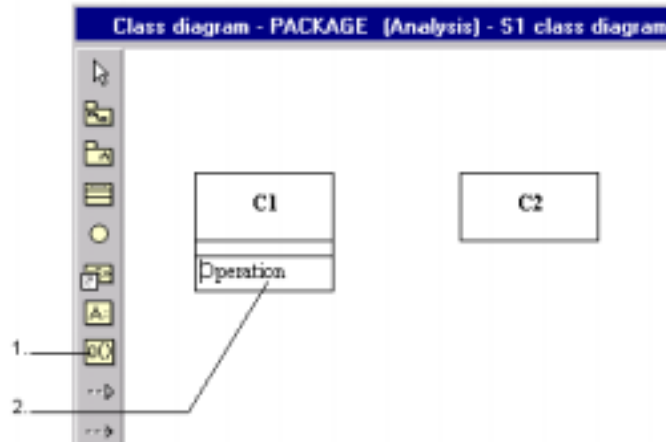



Figure 2-9. Creating an operation in the "C1" class

Steps:

- 1 - Click on the  "Create an operation" button.
- 2 - Click in the "C1" class. Enter the name of the operation ("Operation1") over the highlighted text which shows the default name ("Operation()"). Press "Return" to confirm.

Modifying an element

We will now modify the name of the "Operation1()" operation in the diagram (as shown in Figure 2-10).



Figure 2-10. Modifying the name of the "C1" class operation

Steps:

- 1 - Select the operation by clicking on the right mouse button, and choose the "Modify" option from the context menu. The operation dialog box then appears (double-clicking on the operation is a short cut). For further information on this dialog box, please refer to the *Objecteering/Model Dialog Boxes* user guide.
- 2 - Modify the name of the operation in the entry field or choose the operation name from the list proposed in the combobox.
- 3 - Confirm by clicking on "OK". If you confirm by clicking on "Apply", the dialog box will remain open.

Note 1: To modify the name of an element, you can also simply highlight its name and directly enter the new name.

Note 2: When an element is modified in a graphic editor, modifications are automatically taken into account in the explorer.

Creating a link in a diagram

Creating an association between two classes

A link is created between an origin element and a destination element. We are going to create an association between the "C1" class (the origin class) and the "C2" class (the destination class), as shown in Figure 2-11.

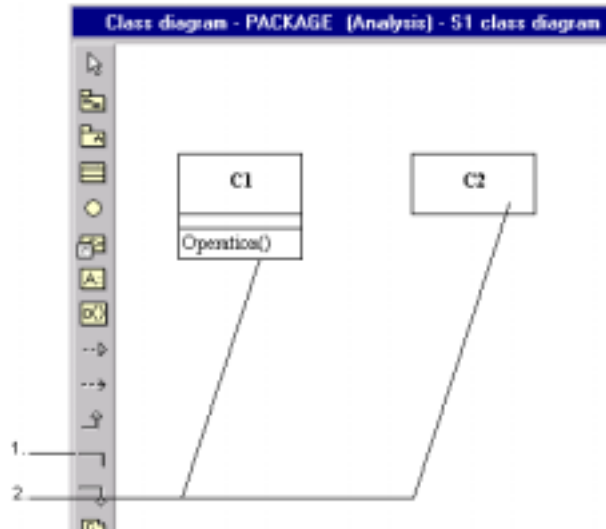



Figure 2-11. Creating an association between the "C1" and "C2" classes


Steps:

- 1 - Click on the  "Create an association" icon.
- 2 - Click on class "C1" and then on class "C2".

Note: It is possible to redraw the association. The redefinition of links is explained in chapter 5 of this user guide.

Resizing an object

Resizing a class

In the graphic editor tool bar, the  "Fit to contents" icon is used to adjust the size of the object to its contents.

First of all, we shall enlarge class "C2" (see Figure 5-9 in the "Handling graphic elements" section in chapter 5 of this user guide), and then continue with the steps shown in figure 2-12.

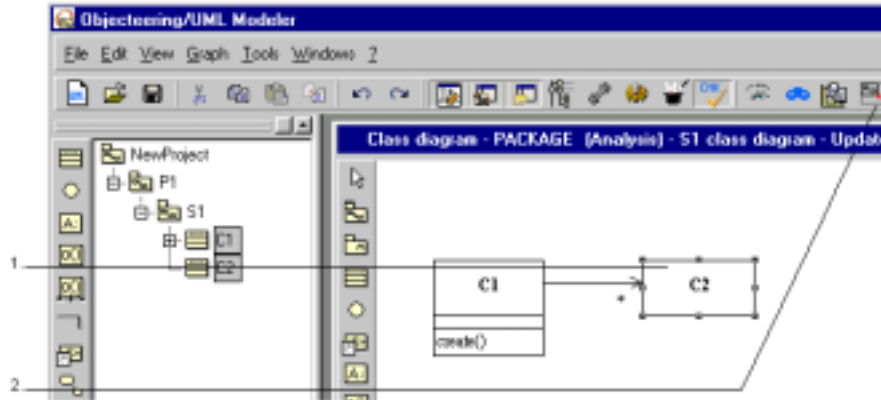



Figure 2-12. Resizing class "C2"

Steps:

- 1 - Select class "C2", which you have just enlarged.
- 2 - Click on the  "Fit to content" icon.

You will see that the element has adjusted itself to the size of its contents. This is true for all objects represented by a box.

Chapter 3: Functions of
Objecteering/UML Modeler
- Overview

Launching Objecteering/UML Modeler

Launching Objecteering/UML

Environment ...	Action required ...
Windows 95/98/2000/NT4	Select " <i>UML Modeler</i> " from the " <i>Start/Programs/Objecteering</i> " menu, or double-click on the UML Modeler icon in your desktop.
Windows NT 3.5	Click on the " <i>Objecteering</i> " icon from the " <i>Objecteering</i> " window.
UNIX	Run the " <i>objing</i> " command.

Note: Please note that when UML Modeler is launched, the user cannot work on UML profiling projects, and when UML Profile Builder is launched, he cannot work on UML modeling projects.

Parameters of the "objing" command

Objecteering/UML can also be launched using the following parameters:

- ◆ "*objing <UML modeling project> <UML model root>*": directly launches the Objecteering/UML modeling project specified.
 - ◆ "*objing <UML modeling project path>*": launches Objecteering/UML by opening the UML modeling project located in the path indicated.
-

Creating or opening a UML modeling project

Creating a new UML modeling project

To create a new UML modeling project, follow the steps shown in Figure 3-1 below.

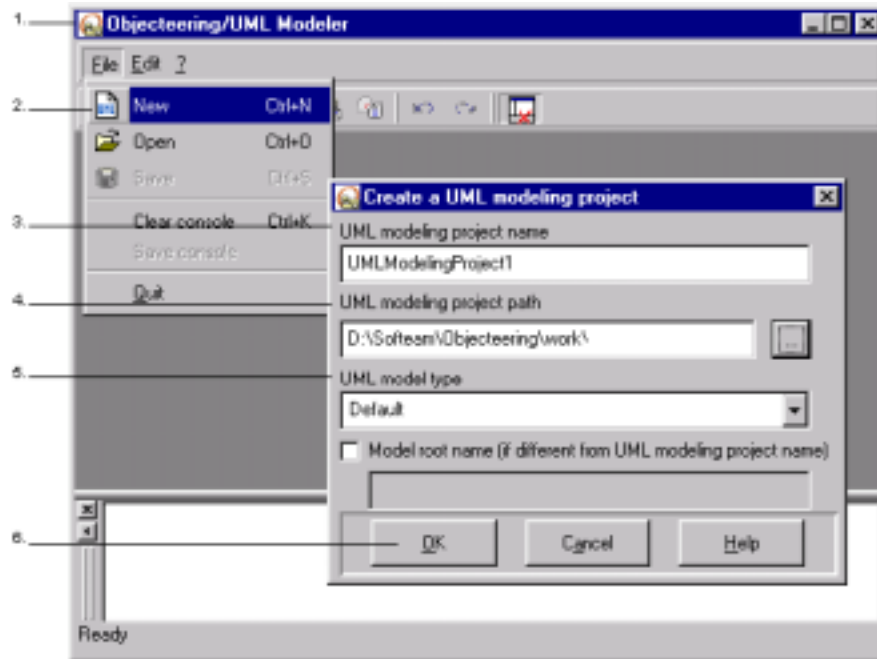



Figure 3-1. Creating a new UML modeling project

Steps:



- 1 - Launch the *Objecteering/UML Modeler* tool by clicking on the *UML Modeler* icon in your desktop. The window shown in Figure 3-1 will then appear.
- 2 - Click on the "*File/New*" menu. The "*Create a UML modeling project*" window will then open.
- 3 - In the "*UML modeling project name*" field, enter the name of the UML modeling project which is to be created.
- 4 - In the "*UML modeling project path*" field, enter the path of the directory where the new UML modeling project is to be created. You may also use the  icon to open a file browser through which you can select your UML modeling project path.
- 5 - In the "*UML model type*", a type for the modeling project which is to be created can be selected. For example, by selecting the "*DefaultJava*" type, your *Objecteering/UML* working environment will be automatically configured for Java development with the *Objecteering/Java* and *Objecteering/Design Patterns for Java* modules.
- 6 - Confirm by clicking on the "*OK*" button.

Note: By default, the name of the UML model root which appears in the explorer is the same as the name of the UML modeling project itself. If you should wish to give the UML model root another name, you should simply check the "*Model root name (if different from UML modeling project)*" tickbox, which will then allow you to enter a different name directly in the field below.

Opening an existing UML modeling project

The procedure for opening an existing UML modeling project is very similar to the procedure for creating a new UML modeling project. Simply carry out the following steps:



- 1 - Click on the *Objecteering/UML Modeler* icon in your desktop. The window shown in Figure 3-1 above will then appear.
- 2 - Click on the "File/Open" menu. The "Open an existing UML modeling project" window will then open.
- 3 - Double-click on the UML modeling project you wish to open. It will then automatically open.

It is also possible to open an existing UML modeling project simply by double-clicking on it in the explorer. This launches Objecteering/UML and opens the UML modeling project you have selected.

Note: Predefined types are not shown unless you check the "Show predefined types" tickbox. For further information on predefined types, please refer to the "Detailed view of the Configuration menu" section in chapter 3 of the *Objecteering/Administrating Objecteering Sites* user guide.

Changing UML modeling project

If you already working on a UML modeling project and you wish to either create a new one or open another existing one, you will be asked if you wish to save the work carried out on the initial UML modeling project, before closing it. For further information, please refer to the "Changing UML modeling project" section in chapter 3 of the *Objecteering/Introduction* user guide.

Receiving and/or upgrading UML modeling projects

As we have just seen, UML modeling projects can be opened simply by double-clicking on the UML modeling project file either in the Windows explorer or in the "*Open an existing UML modeling project*" window.

However, if you wish to work on a UML modeling project created on another user site and/or using an earlier version of Objectteering/UML, the UML modeling project has to be received and/or upgraded. For further information, please refer to the "*Receiving and upgrading UML modeling projects*" section in the current chapter of this user guide.

Receiving and upgrading UML modeling projects

Introduction

Objectteering/UML features a simplified UML modeling project reception and upgrade procedure, thus ensuring backward compatibility and making Objectteering/UML even easier for you to use.

If you wish to work on a UML modeling project either created on a different user site or using an earlier version of Objectteering/UML, you can launch the reception and/or upgrade procedures simply by:

- ◆ double-clicking on the UML modeling project in question in the Windows explorer
- ◆ double-clicking on the UML modeling project in question in the "*Open an existing UML modeling project*" window

Receiving a UML modeling project

There are four possible case scenarios with regard to the reception of UML modeling projects:

- ◆ the reception of a database which is not known to your site, but which exists in the same version of Objectteering/UML. In this case, you can choose to simply receive the database in question on your site.
- ◆ the reception of a database which is not known to your site, and which exists in an earlier version, for example, the 4.3.2 version of Objectteering/UML. In this case, you can choose to receive the database in question on your site, and upgrade it from the previous version to the current version of Objectteering/UML.
- ◆ the reception of a database which is not known to your site, but which has the same name as an existing database and which exists in the same version of Objectteering/UML. In this case, you can choose to receive the database in question on your site and rename it.
- ◆ the reception of a database which is not known to your site, but which has the same name as an existing database and which exists in an earlier version, for example, the 4.3.2 version of Objectteering/UML. In this case, you can choose to receive the database in question on your site, rename it and upgrade it from the previous version to the current version of Objectteering/UML.

In the example shown below (Figure 3-2), a simple reception operation is demonstrated.



Figure 3-2. Receiving the "OutsideProject" UML modeling project by double-clicking in the Windows explorer

Steps:

- 1 - In the Windows explorer, position yourself in the directory containing the UML modeling project you wish to use.
- 2 - Double-click on this UML modeling project file. Objecteering/UML then opens and a confirmation dialog box appears, asking you if you wish to receive the database of the UML modeling project on your site.
- 3 - Click on the "OK" button to confirm the reception of the UML modeling project.

The reception procedure is then launched, and you can follow its progress in the console. This process can take a few minutes. Once completed, the UML modeling project is opened and is ready for use.

Upgrading a UML modeling project

To upgrade a UML modeling project created on your site using, say, the 4.3.2 version of Objectteering/UML, simply carry out the steps below (shown in Figure 3-3).

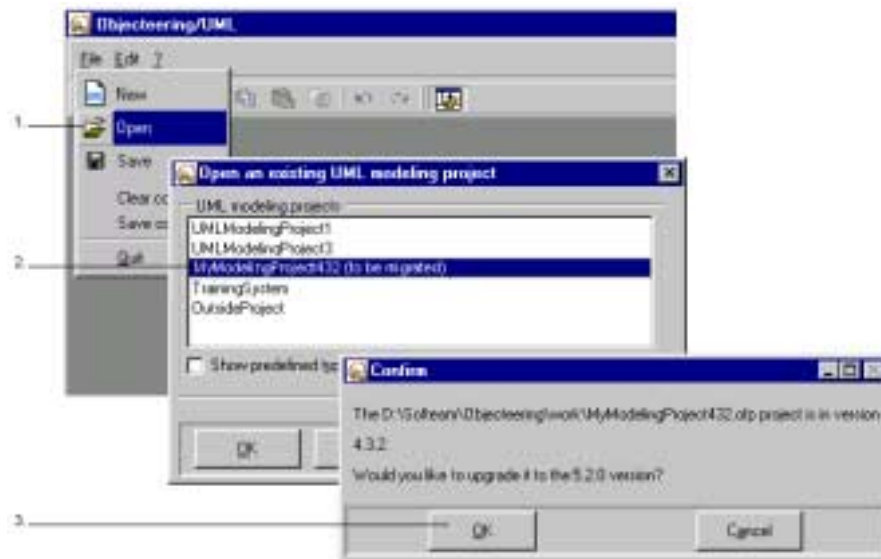


Figure 3-3. Upgrading a UML modeling project via the "Open an existing UML modeling project" window

Steps:

- 1 - Click on "File/Open". The "Open an existing UML modeling project" window will then appear.
- 2 - Double-click on the UML modeling project you wish to upgrade. Please note that UML modeling projects which have not yet been upgraded are listed in this window with the message "to be migrated" shown after their name.
- 3 - A dialog box then appears, telling you which version the UML modeling project is currently in, and asking you if you want to upgrade it. Click on the "OK" button.

The upgrade procedure is then automatically launched, and you can follow its progress in the console. This process can take a few minutes. Once completed, the UML modeling project is opened and is ready for use.

Note: When a UML modeling project is upgraded from a previous version of Objecteering/UML to the current version, the modules contained therein are not upgraded. To upgrade modules, unselect the current version and then select the new version of the module in question.

The main window

Description

The main window of the Objecteering/UML tool is shown in Figure 3-4.

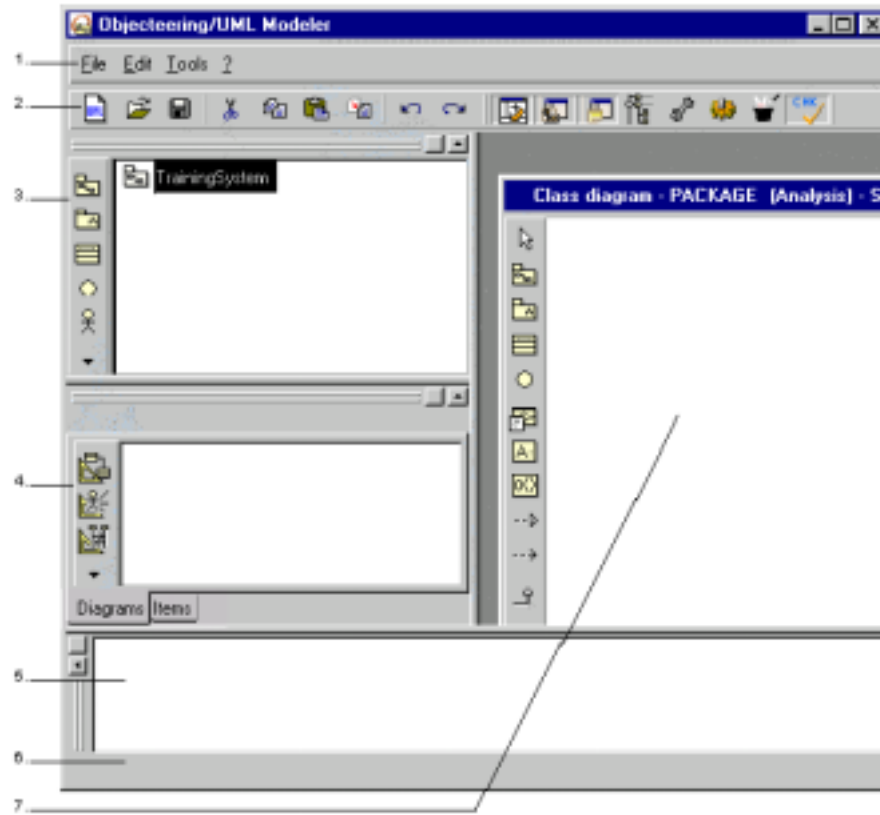


Figure 3-4. The main window for PC

Key:

- 1 - Menu bar: This contains the "*File*", "*Tools*" and "*Administration*" menus.
- 2 - Tool bar: This provides icons associated to certain elements in the menu bar.
- 3 - Explorer: The explorer is one of the main tools used to create and manipulate model elements within a UML modeling project.
- 4 - Properties editor: The properties editor contains a number of tabs, each with a specific function. For example, the "*Diagrams*" tab is used to display the icons you will need to create diagrams in Objecteering/UML, whilst the "*Items*" tab displays icons used to create terminal elements.
- 5 - Graphic editors: Graphic editors present diagrams and are used to graphically display models and model elements, and to create, modify or destroy graphic elements.
- 6 - Console: This contains operation traces, error messages and warnings.
- 7 - Status bar: This provides information complementary to that displayed by the bubbles which appear over the tool's icons.

For details on the menus which appear in the main window, please refer to the "*UML Modeler menus*" section in chapter 4 of this user guide.

For details on the tools available through the main window, please refer to the "*UML Modeler tools*" section in chapter 4 of this user guide.

Dockable windows

Objectteering/UML features dockable windows, in other words, windows which you can position and dock as you wish, either within the Objectteering/UML workspace or even outside it. The main explorer, the console and the properties editor are all dockable windows. Other windows, such as Objectteering/UML graphic editors, cannot be dragged on top of these dockable windows.

Figure 3-5 shows the original Objectteering/UML window layout, as well as a possible alternative arrangement.

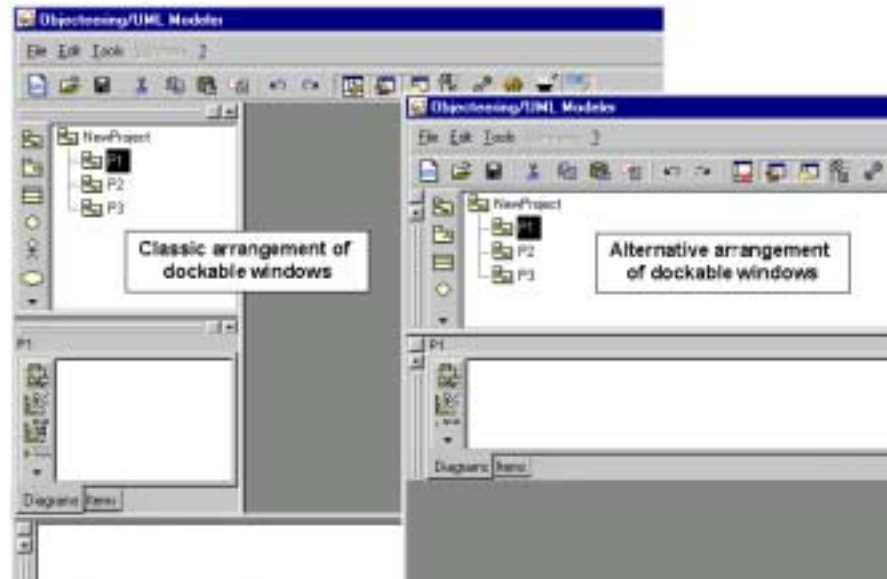


Figure 3-5. Different ways of arranging the dockable Objectteering/UML windows

To reposition dockable windows, simply click on them and drag them into the position of choice. As you will see, a traced outline previews the new position.

The explorer

Graphic representation

The explorer (shown in Figure 3-6) is one of the main tools used to manipulate model elements within a UML modeling project. It provides an automatic representation of all components, starting from a UML model root.

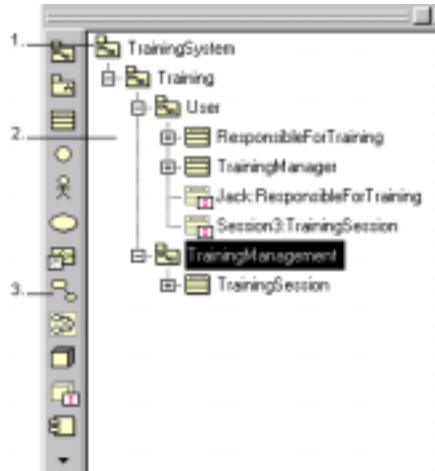


Figure 3-6. The explorer in PC

Key:

- 1 - UML model root
- 2 - Structure of the model
- 3 - Creation icons

Note: For further information on this window, please refer to the "*Explorer functions*" section in chapter 4 of this user guide.

The properties editor

Overview

The properties editor is one of the main Objectteering/UML windows and contains a number of tabs, as shown in Figure 3-7.

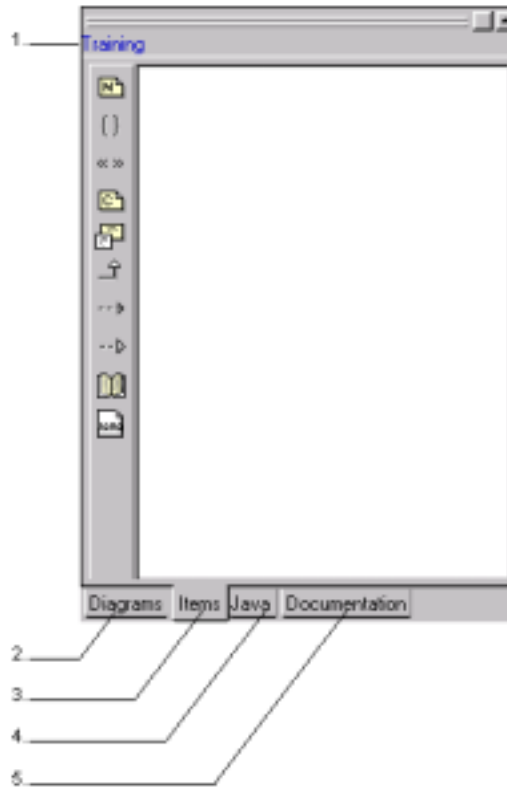


Figure 3-7. The properties editor

Key:

- 1 - The model element selected in the explorer.
- 2 - The "*Items*" tab
- 3 - The "*Diagrams*" tab
- 4 - The "*Java*" tab
- 5 - The "*Documentation*" tab

These tabs contain creation icons for terminal elements and diagrams. Certain modules, for example *Objectteering/Java* and *Objectteering/Documentation*, have their own tab, designed to present information proper to the module concerned. These tabs are only present when the module in question has been selected.

For further information on the properties editor, please refer to the "*Properties editor functions*" section in chapter 4 of this user guide.

The console

Description

The console (shown in Figure 3-8) provides a trace of operations undertaken in Objectteering/UML, such as the opening and printing of a document, as well as error and warning messages, and information on code generation and import administration.



Figure 3-8. Console containing a trace of the opening of a diagram

The user can:

- ◆ clear the contents of the console by selecting the "*Clear console*" option in the "*File*" menu
- ◆ move around in the operation trace, using the scroll bars (where present)
- ◆ visualize an error message or a warning message.
- ◆ save the contents of the console in a file using the "*Save the console*" option. The name of the file is an Objectteering/UML configuration parameter (please refer to the "*The Formalism set*" theme of the "*UML Modeler parameter sets*" section in chapter 4 of this user guide).

Note: The "*Visualizing messages in the console*" section in chapter 4 of this user guide presents the different kinds of messages which may be visualized in the console.

The on-line help search engine

Overview

Objectteering/UML Modeler provides you with a powerful integrated help search engine, used to make it easy for you to find the information you need as quickly as possible.

You can search for information either in:

- ◆ a specific tome of the *Objectteering/UML* on-line help
- ◆ the entire *Objectteering/UML* on-line help database

Chapter 3: Functions of Objecteering/UML Modeler - Overview

Once you have located the sections of the on-line help which contain the information you are looking for, practical hypertext links allow you to jump to the relevant sections.

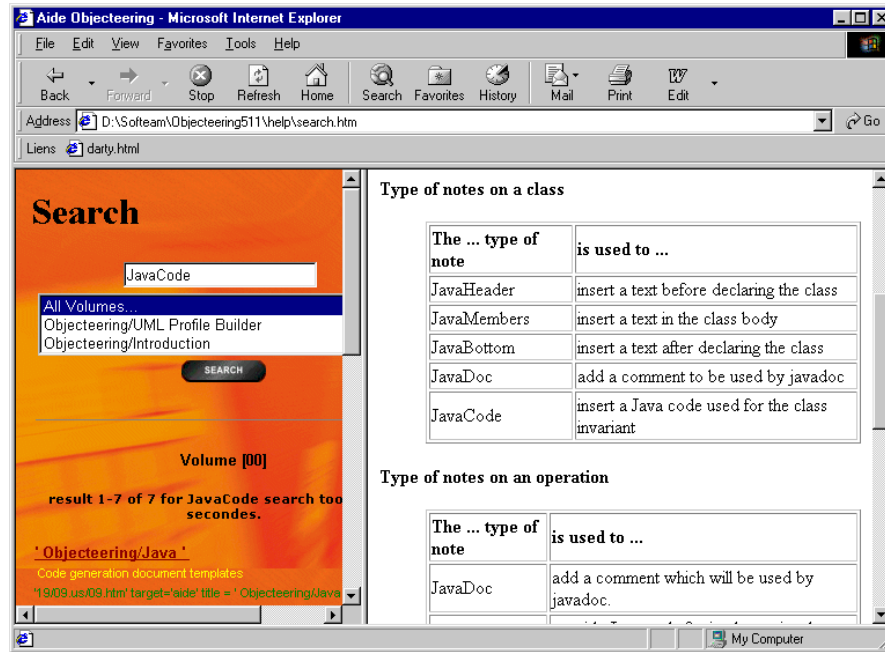


Figure 3-9. The *Objecteering/UML* on-line help search engine

For further information on the search engine, please refer to the "*Using the search engine*" section in chapter 4 of this user guide.






Diagrams





Presentation

Diagrams are an essential feature of UML modeling, and can be created on a certain number of model elements. Several diagrams of the same type can exist for the same element, and consistency between diagrams and the explorer is constantly maintained.

Several diagrams can be edited simultaneously in separate editors.

Model elements and associated diagrams

The ... element	can have	icon ...	is used to represent ...
package, sub-system, class	a class diagram		the internal structure of an element and its relationships with other elements. Classes and packages are the main concepts of this kind of diagram.
package, sub-system	a use case diagram		the most important use cases involved with the current model. Use cases and actors are the main concepts of this kind of diagram.
package, sub-system, class, collaboration, use case	a sequence diagram		a cooperation set between different objects. Instance and messages are the main concepts of this kind of diagram.
package, sub-system, class	an object diagram		a set of class instances with their relationships and the messages exchanged. Instances, links and messages concepts are the main concepts of this kind of diagram.
package, sub-system	a deployment diagram		the physical architecture of the system. Components and nodes are the main concepts of this kind of diagram.


The ... element	can have	icon ...	is used to represent ...
package, sub-system	a deployment instance diagram		a particular instance of deployment. Instances of nodes and components are the main concepts of this kind of diagram.
collaboration	a collaboration diagram		the exchange of messages between roles. Roles instances are the main concepts of this kind of diagram.
state machine	a state diagram		the manner in which objects react to events. It is used to describe a state chart at the level of a class.
activity graph	an activity diagram		a special case of a state machine, which is used to model processes involving one or more classifiers. Its primary focus is on the sequence and conditions for the actions that are taken, rather than on which elements perform those actions.

Work products

Description



Documentation work product icon

Work products are created using specific icons linked to selected modules. For example, the  button represents documentation. This type of work product is present when the *Documentation* module has been selected. Modules such as *C++* or *Java* provide their own work products such as makefile work products or Java Source work products. For more information on these work products, please refer to the associated user guides.

Example

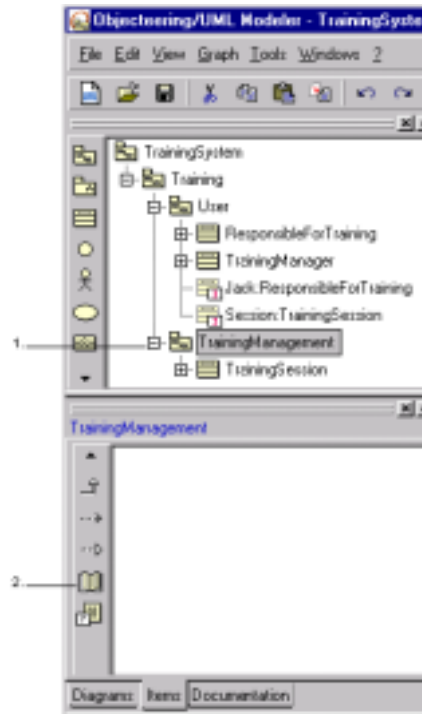



Figure 3-10. Producing a documentation work product from the "TrainingManagement" package

Steps:

- 1 - Select the "TrainingManagement" package.
- 2 - Click on the  "Create a document" icon in the "Items" tab of the properties editor.
- 3 - Fill in the dialog box which appears, and the document work product will then appear in the "Items" tab of the properties editor.

Macros

Overview

A macro is a series of Objectteering/UML commands which can be grouped together within one command, in order to automate repetitive modeling operations.

Objectteering/UML macros are a series of instructions written in our Java-like J language, which can be edited either in the Objectteering/UML tool itself or using an external text editor, such as Word, Wordpad or Notepad.

With Objectteering/UML, you can:

- ◆ execute Objectteering/UML macros, provided as standard with the Objectteering/UML tool
- ◆ create your own macros
- ◆ modify macros
- ◆ delete macros

For further information, please refer to the "*Working with Objectteering/UML macros*" section in chapter 4 of this user guide.

Editing UML Modeler configuration

Overview

The "Edit configuration" dialog box is used to set parameters for modules selected in the current UML modeling project. In this dialog box, a section of hierarchy corresponds to every module selected (for example, *Objectteering/Java*). The sections available depend on the configuration of your site and of your UML modeling project. For example, Figure 3-11 presents a UML modeling project where the *UML Modeler* module (always present) and the *Documentation* module have been selected.

Typically, this window is used to specify generation directories, editing tools, default selections, and so on.

For details on parameters specific to modules, please refer to the user guides associated with the modules in question.

To launch the Objectteering/UML module configuration window, simply click on the



"Modify module parameter configuration" icon or select the "Tools/Modify configuration..." menu.

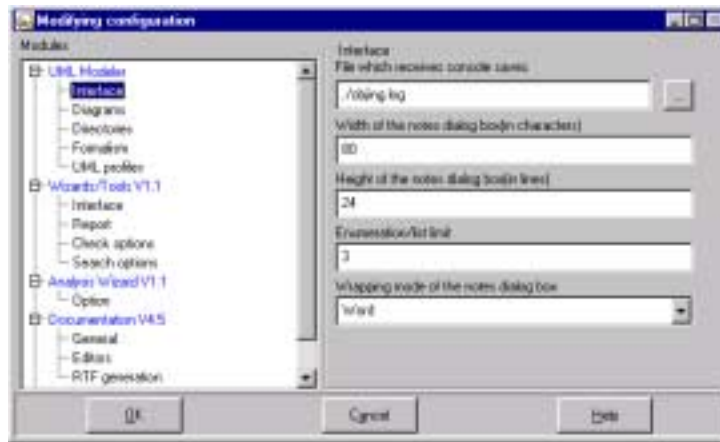


Figure 3-11. The "Modifying configuration" dialog box

Modules frequently present

The following list of modules which are frequently present is by no means exhaustive, and new modules are constantly being created by Objecteering Software and our partners. For new modules, please refer to the user guide specific to the module you wish to configure.

The following modules are frequently present in the "*Edit configuration*" dialog box:

- ◆ *UML Modeler*
- ◆ *UML Profile Builder*
- ◆ *Documentation*
- ◆ *Java*
- ◆ *Macros*

UML Modeler parameters

There exist five sub-sections in the "*UML Modeler*" configuration hierarchy:

- 1 - *Formalism*: general parameters on formalism options
- 2 - *Interface*: general ergonomic options
- 3 - *Diagrams*: general diagram options
- 4 - *Directories*: directories used during file generation (documentation, C++, and so on)
- 5 - *UML profiles*: profiles containing the J rules which parameterize the behavior of the Objecteering/UML graphic interface

Note: These sub-sections are described in the "*UML Modeler parameter sets*" section in chapter 4 of this user guide.

Transferring elements between UML modeling projects

Description of the transfer function

The model element transfer feature gives the user the possibility of importing elements which come from another UML modeling project into his UML modeling project. Imported elements can be packages or classes.

For further information on the transfer of elements between UML modeling projects, please refer to the "*Running principle*" and "*Importing elements between projects*" sections of chapter 4 of the *Objecteering/UML Teamwork User Guide*.

Saving a user work session

Saving the session

This function allows you to save the whole UML modeling project on which you are working.

To activate this function, click on the  "Save" icon or select the "Save" option from the "File" menu.

The "Exit" dialog box

If you wish to exit the tool, simply click on the "File/Quit" menu. If you have not saved your work, Objecteering/UML reminds you of this fact through the following dialog box (shown in Figure 3-12).

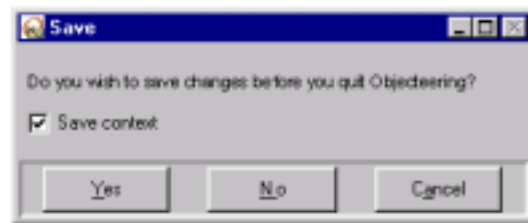


Figure 3-12. The "Exit" dialog box

The ... button	allows you to ...
Yes	exit Objecteering/UML and save the current UML modeling project.
No	exit Objecteering/UML without saving.
Cancel	cancel the "Exit" action and close the dialog box.
Save context?	save your modeling project context

Note: For further information on saving your modeling project context, please refer to the "Saving your work context" section in the current chapter of this user guide.

Changing UML modeling project

It is only possible to work on one UML modeling project at a time.

If you wish to work on another existing UML modeling project, or to create a new one, simply carry out the following steps:

- 1 - Click on the "*File/New*" or "*File/Open*" menu in the menu bar (according to whether you want to create a new modeling project or open an existing one). The "*Create a UML modeling project*" or "*Open a UML modeling project*" dialog box then appears.
- 2 - Enter the relevant information in the dialog box, and click on "OK".
- 3 - A confirmation dialog box then appears, asking you whether you wish to save the work you have done on your original project before closing it and opening the new one. Click on the "Yes" button. Your original project is closed and your new one opened.

Note: For further information on the creation or opening of UML modeling projects, please refer to the "*Creating or opening a UML modeling project*" section in the current chapter of this user guide.

Saving your model context

Saving the model context

When the model context is saved, the size and position of editors open on the model is restored when the model is next used. For explorers other than the main explorer, the selected element is remembered, and diagrams which were open are re-opened.

The model context is saved by:

- ◆ clicking on the  "Save" icon in the tool bar
- ◆ by selecting the "File/Save" menu options
- ◆ by quitting Objecteering/UML

In each case, a confirmation box (like the one shown in Figure 3-13) will appear.

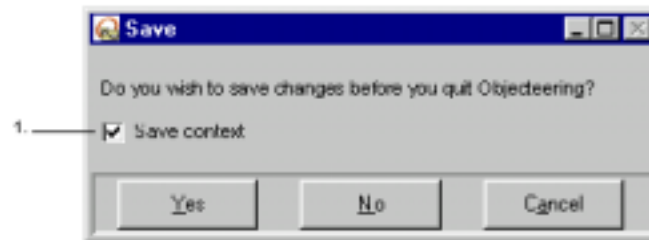


Figure 3-13. Saving your model context

Steps:

- 1 - Check the "Save context?" tickbox.

Note: Information on the model context is saved in the UML modeling project itself.

The search function

Overview

Objectteering/UML provides a powerful integrated search function, which can be used to search for elements in:

- ◆ models
- ◆ metamodels
- ◆ textual elements, such as notes and constraints
- ◆ diagrams



Figure 3-14. The "Search" function window

For further information on the search function, please refer to the "*Using the search function*" section in chapter 4 of this user guide.

Read-only mode

Overview of the Objectteering/UML read-only mode

The read-only mode function is used to protect model elements against potentially incorrect or inappropriate modification. For example, this can be helpful to users working in groups, using, for example, the *Objectteering/Multi-user* module.

It is possible to register a specific module as being the module in charge of the read-only mode, so that at any given time, only this specified module is authorized to change an element's state. For further information on modules which manage the read-only mode, please refer to the *Objectteering/UML Teamwork User Guide*.

Model elements and the read-only mode

Every element in a user model can have two states, read-only and read-write. For read-only elements, no modifications whatsoever can be made by the user. For example, on a class in read-only mode, the user may not change the class' name, add attributes or delete operations.

When an element's state changes from read-write mode to read-only mode, all sub-elements which compose the said element are also put in read-only mode. However, this is not the case for multi-user atomic units. Multi-user atomic units are state change entry points. Elements which are not multi-user atomic units cannot directly receive an order to change state.

Elements in read-only mode are graphically represented differently from elements in read-write mode, and certain context menu items and keyboard shortcuts are no longer available.

Note: For further information on the read-only mode, please refer to the "*Read-only mode*" section in chapter 4 of this user guide.

Locally annotating read-only elements

For certain generators, it can be useful to have the possibility of annotating model elements, without changing the semantics of the annotated objects, even for elements in read-only mode. To this end, local tagged values and local notes are used. For further information on these elements, please refer to the "*Read-only mode*" section in chapter 4 of this user guide.

Removable consistency checks

Introduction

The Objectteering/UML CASE tool provides over 200 consistency checks in real time, used to guarantee the quality and coherence of the model produced. The advantages of these consistency checks are clear:

- ◆ Model consistency is checked when elements are entered, thus ensuring that inconsistent names or elements are not entered.
- ◆ Model consistency is maintained when elements are modified, thus allowing the user to avoid having to manually update all instances of the modified element.

Removable consistency checks

However, the user may, in some modeling situations, prefer to have a certain degree of flexibility with regard to the consistency checks applied to his model. This can be the case, for example, during the preliminary phases of a project (Analysis, Specification, etc.), when users can prefer to have freer, less restrictive use of the CASE tool. Similarly, when importing models from other CASE tools which do not necessarily employ the same level of consistency checks as the Objectteering/UML CASE tool, it can also be useful to be able to deactivate certain checks. For this reason, certain Objectteering/UML consistency checks are removable.

Two types of Objectteering/UML model consistency checks are, therefore, available:

- ◆ Obligatory consistency checks , which are essential to accurate modeling
- ◆ Optional consistency checks, used to assist the user in his modeling activities

Only optional Objectteering/UML model consistency checks may be deactivated by the user. These optional consistency checks are activated or deactivated as a set. In other words, the user can either choose to apply all the optional consistency checks in real time or to apply none of them.

Principle behind removable consistency checks

The principle behind Objecteering/UML removable consistency checks is that the user should be able to:

- ◆ Activate or deactivate optional consistency checks, according to his modeling needs
- ◆ Manually check the model or a part of the model
- ◆ Display and correct errors after re-activating optional consistency checks

Note: For further information on removable consistency checks, please refer to the "*UML Modeler tools*" section in chapter 4 of this user guide.

Obligatory consistency checks

Introduction

Objectteering/UML consistency checks verify the composition tree of a model's elements and the validity of the model itself in Objectteering/UML. These consistency checks are applied to a part of the set of elements contained in Objectteering/UML (Metamodel).

Obligatory consistency checks cannot be deactivated.

For example: A class can belong to a package or to a class, but cannot belong to a data type.

Obligatory Objectteering/UML consistency checks on different elements

The following list presents the different obligatory consistency checks applied by Objectteering/UML to the model element in question.

Element class:

- ◆ An element must be attached to one single container, except in the case of a UML modeling project, a UML profile, a text type and a tag type.

Package class:

- ◆ A package must belong to a UML modeling project or to another package.
- ◆ A package can only specialize packages.
- ◆ No cycles between use links or generalization links are allowed.
- ◆ A package can only use other packages.
- ◆ A package cannot be destroyed if it is used or if it is the parent of another package.
- ◆ A package cannot be destroyed if it represents the UML modeling project.

Class class:

- ◆ A class must belong to a package or a class.
- ◆ A class can only contain classes, data types and enumerations.
- ◆ No cycles between generalization links are allowed.
- ◆ A class cannot be destroyed if it is the parent of another class.

DataType class:

- ◆ A type must belong to a class, a package or a signal.
- ◆ A type cannot contain NameSpaces.
- ◆ A type can neither send nor receive DataFlows.
- ◆ A type cannot implement interface classes.
- ◆ A type cannot have communication links.
- ◆ A type can only specialize a DataType.
- ◆ A type can only use a DataType.
- ◆ No cycles between generalization links are allowed.
- ◆ The *undefined* type cannot be destroyed.

Enumeration class:

- ◆ An enumeration cannot contain NameSpaces.
- ◆ An enumeration cannot have communication links.
- ◆ An enumeration cannot have use links.
- ◆ An enumeration has no generalization links.
- ◆ An enumeration cannot contain DataFlows.
- ◆ An enumeration cannot implement interfaces.
- ◆ An enumeration cannot contain instances.
- ◆ An enumeration cannot be instantiated.
- ◆ An enumeration cannot have members.
- ◆ An enumeration belongs to a package, a class or a signal.

Attribute class:

- ◆ An attribute must belong to a class, an actor, a use case, a component, a node or an association link.
- ◆ An attribute can be typed by a class (its state is of no importance during this phase), a DataType or an enumeration.

Parameter class:

- ◆ A parameter can have a class, a DataType or an enumeration as its type.

Association class:

- ◆ For an association created on a class, only classes, actors, DataTypes and signals are accepted.
- ◆ For an association created on an actor, only classes and actors are accepted.
- ◆ For an association created on a signal, only classes and signals are accepted.
- ◆ For an association created on a node, only nodes are accepted.

Generalization class:

- ◆ There is no generalization on enumeration and component.
- ◆ The NameSpaces which link Generalization have to have the same metaclass, except in the case of Signal, which can specialize a signal or a class.

Use class:

- ◆ A use must occur between two units which are of the same type (*ClassOf*).
- ◆ An actor can use a package, a class, a component, an actor or a signal.
- ◆ A signal can use a signal or a class.

Realization class:

- ◆ Only a class or a component can implement an interface class.

Signal class:

- ◆ There are no DataFlows on Signal.
- ◆ There is no communication on Signal.
- ◆ A signal must belong to a package or a class.
- ◆ No cycles between generalization links are allowed.

DataFlow class:

- ◆ At least one of the "Receive" or "Send" links must be defined.
- ◆ A DataFlow cannot exist on a data type, an enumeration, a use case or a signal.

AttributeRole class:

- ◆ An attribute role must belong to a classifier role.

LinkEnd class:

- ◆ A link end is only linked to one instance.

AssociationEndRole class:

- ◆ An association end role is only linked to a classifier role.

Actor class:

- ◆ An actor must belong to a package.
- ◆ An actor does not implement interface classes ("Realization").
- ◆ An actor cannot be associated with predefined instances ("Instances").
- ◆ An actor cannot contain other NameSpaces.
- ◆ An actor can only specialize actors.
- ◆ No cycles between generalization links are allowed.

UseCase class:

- ◆ A use case can only belong to a package.
- ◆ A use case cannot contain NameSpaces.
- ◆ A use case cannot have any uses.
- ◆ A use case cannot have any associations.
- ◆ A use case neither sends nor receives DataFlows.
- ◆ A use case cannot implement interface classes ("Realization").
- ◆ A use case cannot be associated to predefined instances ("Instances").
- ◆ No cycles between generalization links are allowed.

Communication class:

- ◆ A communication link must link two actors or an actor and a use case.

StateMachine class:

- ◆ A state machine must have a root state.

Transition class:

- ◆ A transition must belong to a StateVertex by the source relationship.
- ◆ A transition must have a destination (*Target* relationship).

InternalTransition class:

- ◆ The Start and Reach relationships must be empty on an internal transition.

Node class:

- ◆ A node must belong to a package.
- ◆ A node contains no NameSpaces.
- ◆ A node contains no uses (Used and empty User).
- ◆ No instances can exist (Declared empty).
- ◆ No implementations can exist (Realized empty).

Component class:

- ◆ A component must belong to a package or to another component.
- ◆ No generalization can exist (relationship to the Generalization class defined on NameSpace).
- ◆ A component contains no instances.

NodeInstance class:

- ◆ A node instance belongs to a package.




ComponentInstance class:

- ◆ A component instance can contain node instances, component instances or instances.
-









Chapter 4: Functions of
Objecteering/UML Modeler
- Detailed View

UML Modeler menus


The "File" menu

The command	... icon ...	is used to ...
New		create a new UML modeling project. When a new project is created, Objecteering/UML will ask you if you wish to save your work on the original project before closing it and opening the newly created one.
Open		open an existing UML modeling project. When another project is opened, Objecteering/UML will ask you if you wish to save your work on the original project before closing it and opening the other one.
Save		carry out a general save of the current UML modeling project.
Clear console	none	erase the information which appears in the console.
Save console	none	save the contents of the console in a file.
Quit	none	exit Objecteering/UML Modeler.





The "Edit" menu

The command ...	icon ...	is used to ...
Undo		cancel the last action.
Redo		redo last action.
Cut		destroy and store the selected element in the clipboard.
Copy		store the selected element in the clipboard.
Paste		create the element stored in the clipboard at the selected position.
Move		move copied elements in the clipboard into the selected element.
Empty clipboard	none	destroy the contents of the clipboard.
Modify		modify the element.
Consult	none	open the selected element's dialog box for consultation purposes.
Delete		destroy the element.




The "View" menu

The ... command	icon ...	is used to ...
Save as	none	open a file browser in which you can enter a name and path, in order to save the current diagram as a separate file.
Print...		open the dialog box used to configure printing parameters and then launch the printing of a diagram.
Copy graph image	none	copy the contents of the active diagram, in order to paste them elsewhere.
Show	none	show the contents of the element selected in the active diagram, or show the links which exist between selected elements.
Mask	none	mask the contents of the element selected in the active diagram, or mask the links which exist between selected elements.
Select all	none	select all elements in the active diagram.

The "Graph" menu

The ... command	icon ...	is used to ...
Align	none	align the selected model elements according to your choice (to the left, to the right, etc.).
Fit to contents		adjust the size of the selected graphic elements to fit the contents.
Zoom forward		increase the display of a diagram's contents, in order to make it easier to read.
Zoom back		reduce the display of a diagram's contents, in order to visualize a larger part of the presentation.
Home	none	reposition graphic elements to the home position in the diagram.
Redraw	none	redraw the selected graphic element.
Resources		display the resources window, which allows you to change certain visual aspects of the diagram (colors, styles, etc.).
Grid	none	activate the grid in a diagram.
Layout	none	move and lay out graphic elements.

The "Tools" Menu

The ... command	icon ...	is used to ...
Modify configuration		modify the configuration of modules installed in your UML modeling project.
Modules...		add modules to or remove modules from your UML modeling project.
Import	none	transfer model parts between different UML modeling projects
Process Wizard		select model examples and model templates, and ensure the coherent selection of techniques used on a UML modeling project in accordance with the development process used (for further information, please refer to the " <i>Objecteering/Process Wizard</i> " user guide)

The "Windows" menu

The ... command	icon ...	is used to ...
Close	none	close the current diagram.
Close all...	none	close all open diagrams.
Next	none	switch to the next diagram.
Previous	none	switch to the previous diagram.
Cascade	none	display all open diagrams in a cascade formation.

UML Modeler tools

Modification of modules used



Objectteering/UML provides a set of modules, which allows you to change the configuration of the CASE tool in order to use specific services (for example, the *Java Generation* module).

The "Modules" window allows you to add or remove Objectteering/UML modules to or from the current UML modeling project (as shown in Figure 4-1). The procedure for selecting a module is explained in chapter 3 of the *Objectteering/Introduction* user guide.

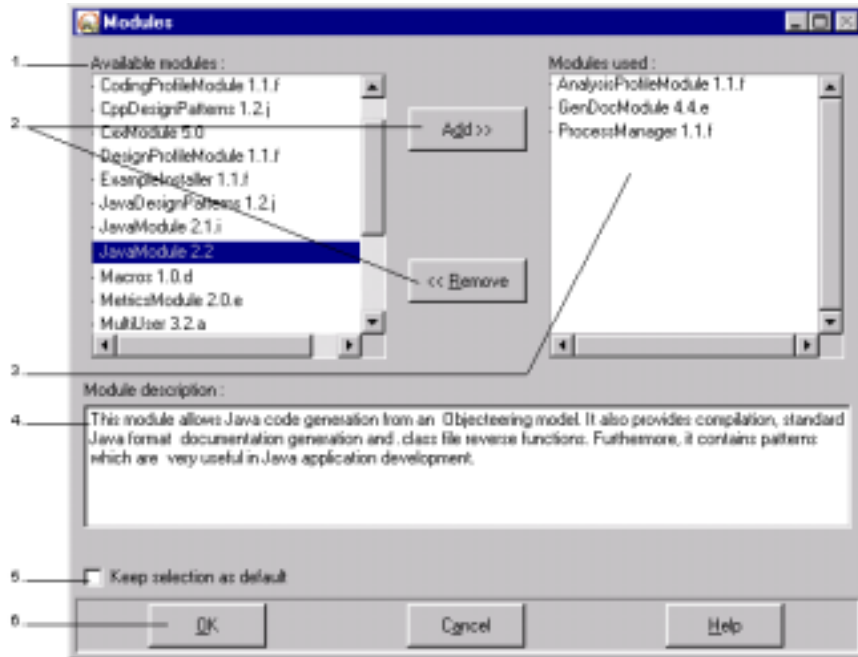


Figure 4-1. Window for selecting modules for a UML modeling project

Key:

- 1 - List of modules.
- 2 - Buttons allowing the addition or withdrawal of modules in a UML modeling project.
- 3 - List of modules used. When a module has been selected from the list of modules on the left-hand side of the window, and after having clicked on "Add", it appears in the list of modules used on the right-hand side of the window.
- 4 - Description of the module selected from the list of modules installed.
- 5 - Default selection button. If this box is checked, the user may specify that this module selection should be carried out automatically for every new UML modeling project created.
- 6 - Click on "OK" to confirm your choices.

Process Wizards

The main domains covered by the "*Process Wizard*" are as follows:

- ◆ follow-up and management of Profile Modules used in the development process
- ◆ assistance during UML modeling project creation
- ◆ assistance to complete the model and to check it during modeling activities
- ◆ improvement in productivity due to specific patterns and impact analysis tools


For further information on this tool, please refer to the *Objecteering/Process Wizard* user guide.


Removable consistency checks



Objectteering/UML provides a function enabling the user to deactivate and then reactivate certain Objectteering/UML consistency checks, thus allowing the user a certain degree of flexibility with regard to the consistency checks applied to his model.



The pressed down  icon in the menu bar (see Figure 4-2) indicates that removable consistency checks are activated. When deactivated, the "switched

off"  icon is displayed. By default, removable consistency checks are active.

Only optional Objectteering/UML model consistency checks may be deactivated by the user. These optional consistency checks are activated or deactivated as a set. In other words, the user can either choose to apply all the optional consistency checks in real time or to apply none of them.

The principle behind Objectteering/UML removable consistency checks is that the user should be able to:

- ◆ Deactivate optional consistency checks
- ◆ Manually check the model or a part of the model
- ◆ Display and correct errors
- ◆ Reactivate optional consistency checks, according to his modeling needs, and fix the model

Deactivating removable consistency checks

Since removable consistency checks are activated by default, we are now going to deactivate them, as shown in Figure 4-2.

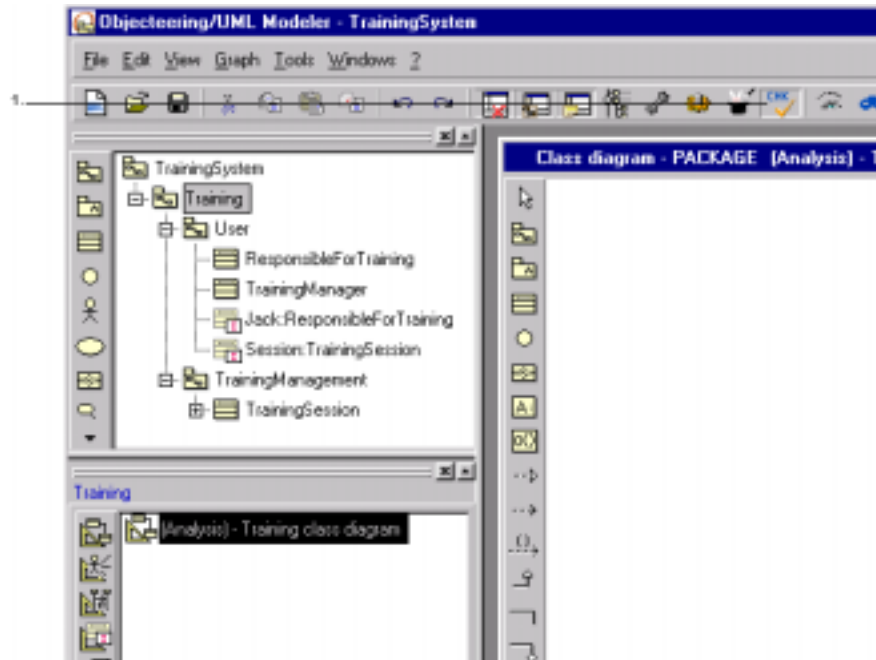




Figure 4-2. Deactivating removable consistency checks

Steps:

- 1 - Click on the  icon, which "switches off" removable consistency checks on the entire UML modeling project. The  icon is then displayed in the menu bar.

Manually checking the model or a part of the model

When removable consistency checks have been deactivated and modifications made, the user may manually check the consistency of his model through the "Check model" context menu entry (as shown in Figure 4-3).

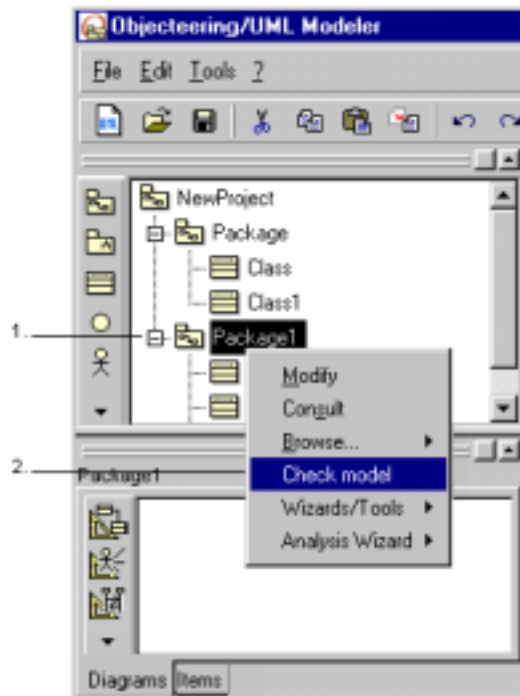


Figure 4-3. Manually checking the "Package1" package

Steps:


- 1 - Select the element to be checked, in this case the "Package1" package.
- 2 - Click on the right-mouse button to open the context menu, and run the "Check model" command.

If any errors in consistency exist, these are displayed in, and may be corrected via, the dialog box shown below in Figure 4-4.

Note: The user is not obliged to run this command on specific elements in his UML modeling project, since when removable consistency checks are reactivated, the consistency of the entire UML modeling project is checked. It can, however, be simpler to have only the errors related to a particular element displayed at any one time, rather than all the errors in the entire UML modeling project.

Displaying and correcting errors

To display and subsequently correct errors in consistency, the user has two options:

- ◆ Clicking on the  icon to reactivate removable consistency checks on the entire UML modeling project
- ◆ Running the "*Check model*" context menu command on his model elements (as shown in Figure 4-3)

In both cases, if errors exist, a dialog box like the one shown in Figure 4-4 is displayed.

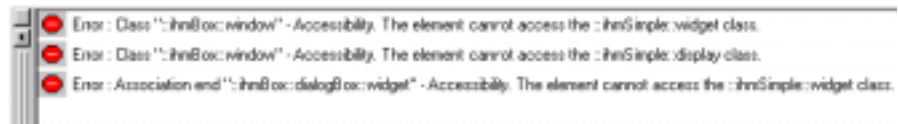


Figure 4-4. An example of the dialog box used to display consistency errors

Chapter 4: Functions of Objectteering/UML Modeler - Detailed View

To correct the errors displayed in this dialog box, the user has two choices:

- ◆ Double-clicking on the error in question, to automatically open the dialog box corresponding to the erroneous element, through which the error can be corrected.
- ◆ Accessing the erroneous element via the explorer and correcting the error.

The ... button	is used to ...
OK	close the dialog box.
Print	print the errors which appear in the dialog box, as well as associated help messages.
Help	display the help message which corresponds to the currently selected error. Certain errors can be rather complicated to understand, and it can be useful for the user to have the possibility of visualizing the associated help message.
Check again	update the error display dialog box. Once an error has been corrected, its correction is not automatically reflected in the dialog box. In order for error corrections to be taken into account, the user must click on the " <i>Check again</i> " button on the bottom right of the error display dialog box.

Reactivating removable consistency checks

To reactivate removable consistency checks, carry out the steps illustrated in Figure 4-5.

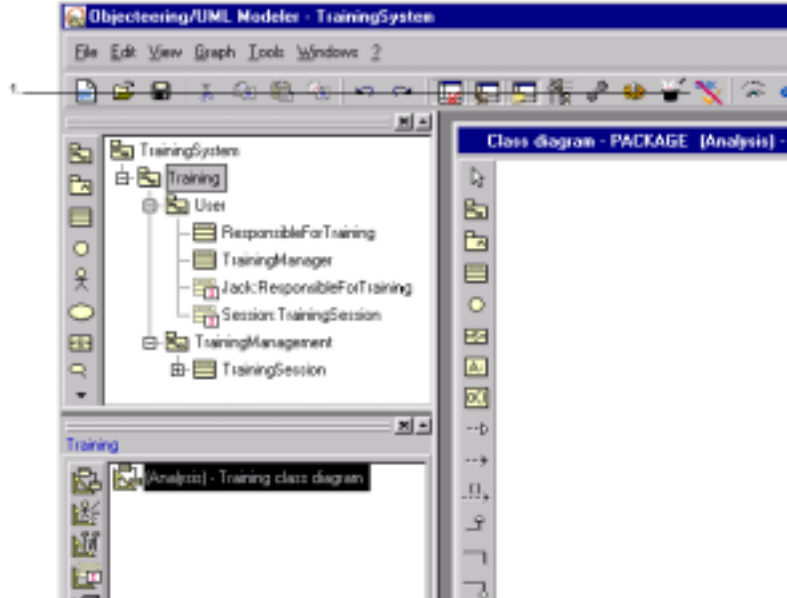





Figure 4-5. Reactivating removable consistency checks

Steps:

- 1 - Click on the  icon, which "switches on" removable consistency checks on the entire UML modeling project.

Note: The  icon has been transformed to , to indicate that removable consistency checks are no longer engaged. The system will only carry out obligatory consistency checks on the model.

Explorer functions

Principal functions



The explorer presents the model to a high level of detail, from the highest level (packages) down to the lowest level, such as operations, parameters, states, transitions.

The explorer allows you to:

- ◆ visualize model elements
- ◆ create and modify elements
- ◆ reference elements
- ◆ move elements using the drag and drop facility
- ◆ destroy elements
- ◆ duplicate elements using the copy/paste facility
- ◆ filter visualized element types

Launching an explorer

An explorer can be launched in the following ways:

- ◆ from the main window, using the  "Explorer" icon in the menu bar. The explorer created by the main window presents the UML model root.
- ◆ from an existing explorer. The new explorer will refer to the same UML modeling project, but the root can be different (see figure 4-8). An element must be selected from the composition tree, before selecting the "Explorer" option from the "Display" menu or clicking on the  button. The element selected becomes the root in the newly created explorer.

Note: The explorer which is dockable and which appears when the project is created is the main explorer.

The user may open as many explorers as he wishes, in the following modes:

- ◆ *update mode*: this allows you to browse the model and to create and/or update elements (Figure 4-6)

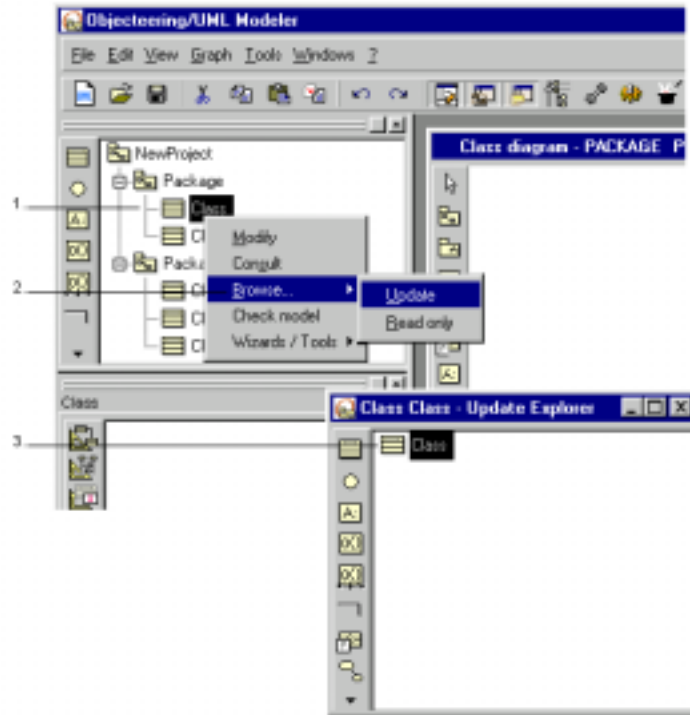


Figure 4-6. The explorer - the "Update" mode on a class

Steps:

- 1 - Select a class.
- 2 - Run the "*Browse.../Update*" command.
- 3 - Edit the update explorer class, to create and/or update elements.

- ◆ *read only mode*: this allows you to browse the model without being able to update elements (Figure 4-7).

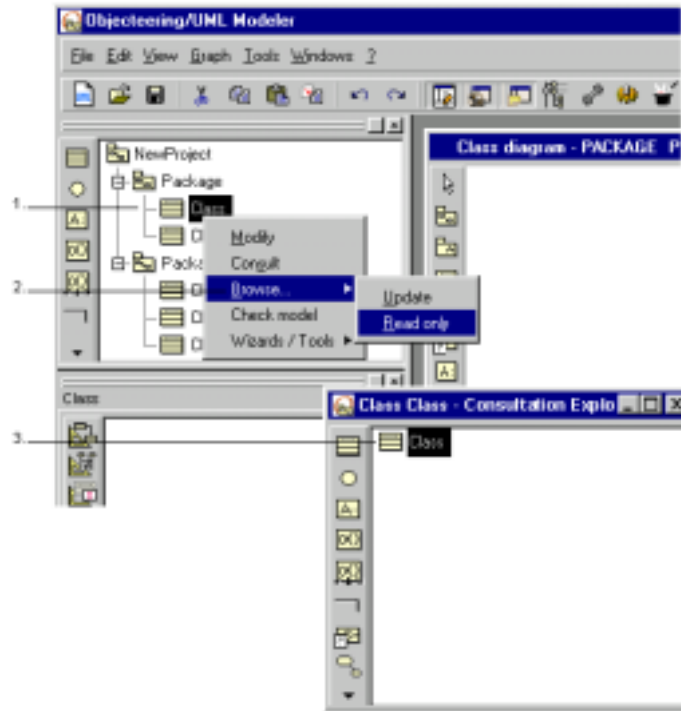


Figure 4-7. The explorer - the "Read-only" mode on the class

Steps:

- 1 - Select a class.
- 2 - Run the "Browse.../Read-only" commands.
- 3 - Edit the read-only explorer class in read-only mode.

Browsing in the explorer

Description

The explorer is used to browse the model to a very high level of detail.

It is possible to browse elements using the mouse, by clicking on the element in question:

- ◆ a click on the "+" or the "-" situated next to each element expands the said element or reduces it.
- ◆ double-clicking on the left mouse button over an element opens a dialog box, which allows the modification of the element in question (with the exception of the name of the UML model root).
- ◆ clicking on the right mouse button over an element opens the context menu which is used to trigger a specific action.

Context icons

In the explorer, the icons in the left-hand vertical palette change according to the element selected in the explorer. The automated creation icons will adapt themselves to the element in question (as shown in Figure 4-8).

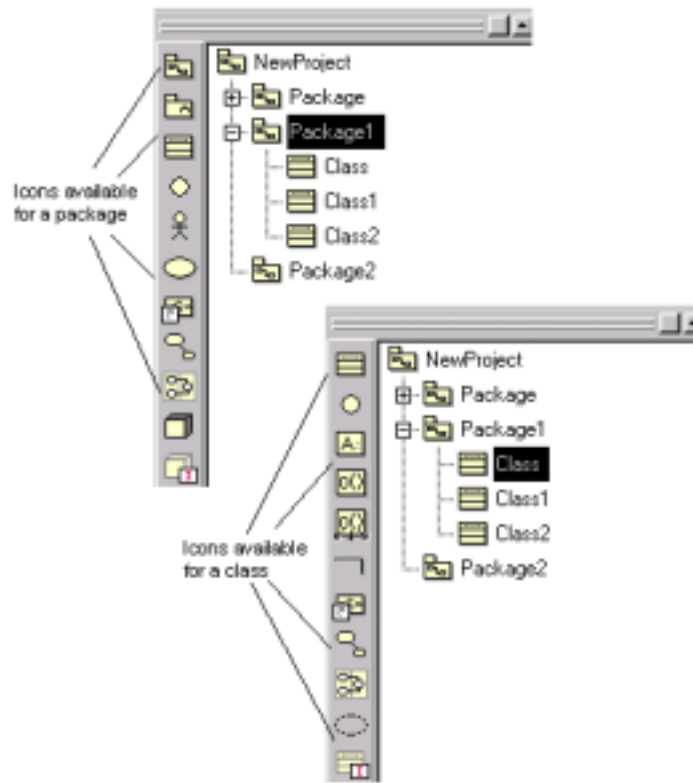



Figure 4-8. The explorer adapts to the element selected

The elements which can be created using the explorer icons are so-called structural elements, such as classes, packages or attributes.

Note 1: Terminal elements, such as diagrams, notes and tagged values are created via the properties editor.

Note 2: If you are using modules which manage the read-only mode, certain elements in the explorer may be displayed with the  icon superimposed over the icon representing the element itself. This icon indicates that these items may not be modified. For further information on the read-only mode, please refer to the *Objecteering/UML Teamwork User Guide*.










Structural element creation icons

















Presentation












The explorer:

- ◆ presents icons which represent elements which can be created on the element selected
- ◆ are updated according to the element selected in the explorer

Structural units

The ... icon	allows you to ...	from ...
	create a package	a package, a sub-system.
	create a sub-system	package, a sub-system.
	create a class	a package, a sub-system, a class.
	create an interface class	a package, a sub-system, a class.
	create an actor	a package, a sub-system.
	create a use case	a package, a sub-system.
	create a signal	a package, a sub-system, a class.
	associate a state machine	a package, a sub-system, a class, an actor, a signal, a node, a component, a state machine, a use case, an operation.
	associate an activity graph	a package, a sub-system, a class, an actor, a use case, a node, an operation, a signal, a component.

The ... icon	allows you to ...	from ...
	create a node	a package, a sub-system.
	create a node instance	a package, a sub-system, a node instance, a data type.
	create a component	a package, a sub-system, a component.
	create a collaboration	a package, a sub-system, a class, a use case, an operation.
	create an instance	a package, a sub-system, a class, a node instance, an instance, a data type.
	create a data flow	a package, a sub-system, a class, an actor, a node, a component.
	declare a type	a package, a sub-system, a class, a signal.
	define an enumeration	a package, a sub-system, a class, a signal.
	create an attribute	a class, an actor, a use case, a signal, a node, a component, a data type.
	create an operation	a class, an actor, a use case, a signal, a node, a component, a data type.
	redefine an operation	a class, an actor, a use case, a signal, a node, a data type.
	create a binary association	a class, an actor, a signal, a node, a data type, an association.
	create a template parameter	a class.
	create a state	a state machine.
	create an event	a state machine, an activity graph.
	create a transition	a state, a sub activity state, an action state, an object flow state.

The ... icon	allows you to ...	from ...
	create an internal transition	a state.
	create a component instance	a node instance.
	create an attribute link	a node instance, an instance, an object.
	create a link	a node instance, an instance, an object.
	create a classifier role	a collaboration.
	add a return parameter	an operation.
	add a parameter	an operation.
	create an action state	an activity graph, a sub activity state.
	create a sub activity state	an activity graph, a sub activity state.
	create an object flow state	an activity graph, a sub activity state.
	create a partition	an activity graph.

Modifying elements

Procedure

To modify an element in the explorer, simply carry out the steps shown in Figure 4-9.



Figure 4-9. Modifying a model element in the explorer

Steps:

- 1 - Select the element.
- 2 - Double-click on the element itself, select the "Modify" option from the pop-up menu, which is displayed by clicking on the right mouse button, or activate the "Edit/Modify" menu.
- 3 - Enter the modifications in the dialog box which appears (see the *Objecteering/Model Dialog Boxes* user guide).

Properties editor functions

Overview

The properties editor is essentially a window designed to aid the user in his modeling, by providing rapid access to various information and services he may need to use.

The properties editor contains a number of tabs:

- ◆ the "*Items*" tab, which displays terminal elements belonging to the selected model element, as well as icons used to create these terminal elements
- ◆ the "*Diagrams*" tab, which presents information on diagrams belonging to the selected model element, as well as the icons used to create them
- ◆ the "*C++*", "*Java*", "*VB*" and/or "*Documentation*" tabs are present where the *Objectteering/C++*, *Objectteering/Java*, *Objectteering/Visual Basic* and/or *Objectteering/Documentation* modules have been selected in the current UML modeling project, and facilitate the entry of information specific to the said domain.

Note: It should be noted that when a module which adds a tab to the properties editor is selected, where earlier versions of the module in question did not provide this service, you should quit and restart Objectteering/UML, in order for the properties editor to be correctly displayed.

The "Items" tab

The "Items" tab of the properties editor contains the icons used to create terminal elements, such as tagged values, notes and generalization links, and displays those terminal elements which already exist on the selected model element (as shown in Figure 4-10).

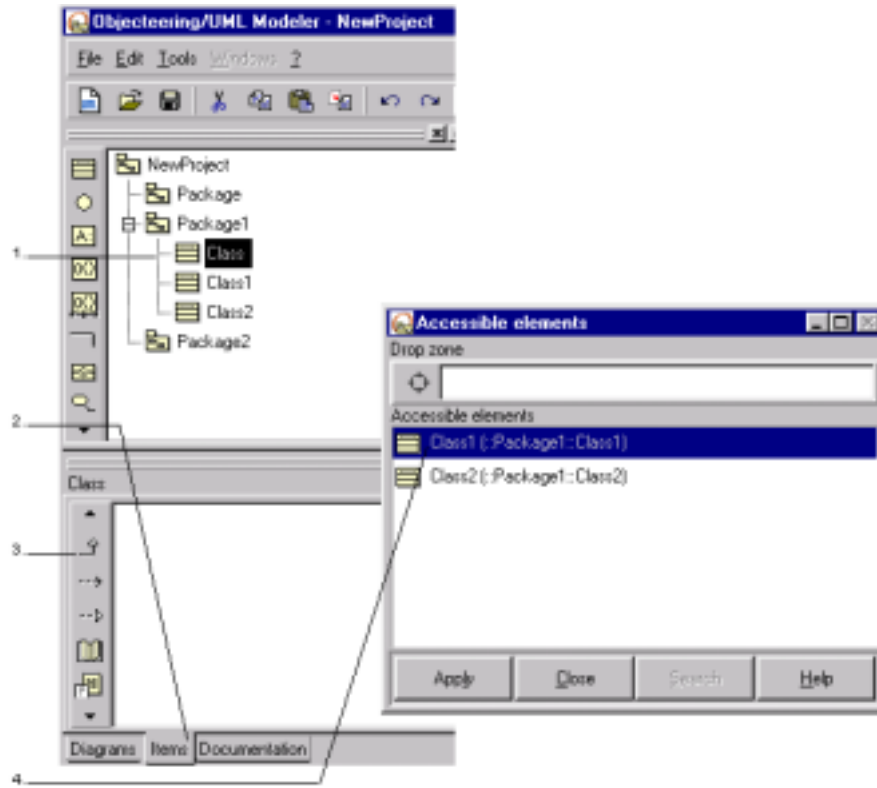



Figure 4-10. Creating a generalization for the "Class" class in the "Items" tab

Chapter 4: Functions of Objectteering/UML Modeler - Detailed View

Steps:

- 1 - Click on the "Class" class in the explorer.
- 2 - Click on the "Items" tab of the properties editor. You will then see any terminal elements belonging to the "Class" class.
- 3 - Click on the  "Specialize" icon. The "Accessible elements" window then appears.
- 4 - Click on the "Search" button and select the relevant element. Confirm by clicking on "Apply". The newly created generalization appears in the properties editor.

The "Diagrams" tab

The "*Diagrams*" tab in the properties editor presents the diagrams which already exist for the selected model element, and provides the icons used to create diagrams (as shown in Figure 4-11).

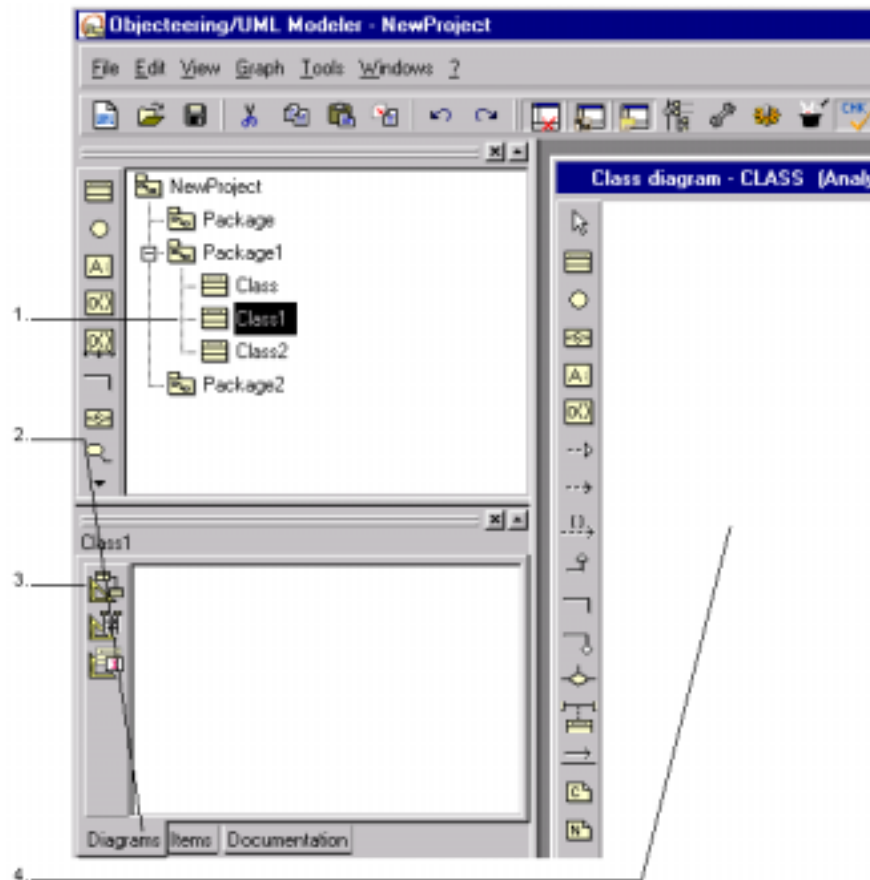



Figure 4-11. Creating a class diagram from the "*Diagrams*" tab in the properties editor

Chapter 4: Functions of Objectteering/UML Modeler - Detailed View

Steps:

- 1 - Select the element for which you wish to create a diagram in the explorer.
- 2 - Click on the "Diagrams" tab in the properties editor.
- 3 - Click on the  "Create a class diagram" icon.
- 4 - The newly created diagram will then automatically open.

The "Documentation" tab

The "Documentation" tab of the properties editor is used to enter summary and description notes for elements selected in the explorer directly in the properties editor itself, without having to activate the "Note" dialog box. You can also indicate whether or not you wish to annotate the selected element using the `{noanalysis}` and `{nodesign}` tagged values.

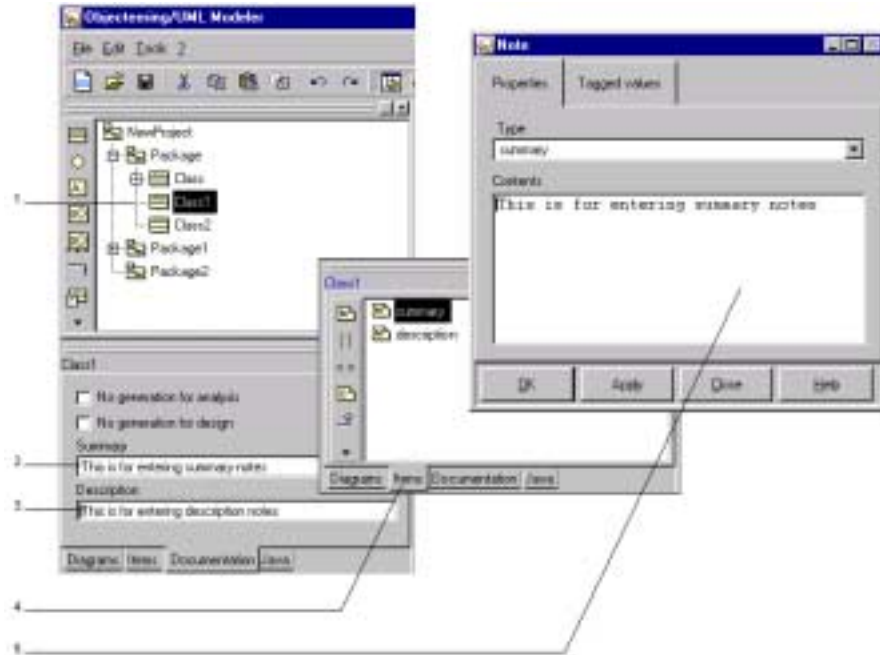




Figure 4-12. The "Documentation" tab

Steps:

- 1 - Select an element in the explorer, and click on the "Documentation" tab in the properties editor.
- 2 - Enter the following text in the "Summary" field: "This is for entering summary notes". Indeed, this field is used to enter summary notes for the element selected in the explorer directly in the properties editor, without having to activate the "Note" dialog box (either by clicking on the  "Add a note" icon or by selecting the "Notes" tab in the element's dialog box).
- 3 - Enter the following text in the "Description" field: "This is for entering description notes". Indeed, this field is used to enter description notes for the element selected in the explorer directly in the properties editor, without having to activate the "Note" dialog box (either by clicking on the  "Add a note" icon or by selecting the "Notes" tab in the element's dialog box).
- 4 - You can check that these notes have been entered, by clicking on the "Items" tab of the properties editor. You will see that two notes have been created.
- 5 - Double-click on one of the notes in order to open the "Note" dialog box. You will see the text you entered in the "Documentation" tab of the properties editor.

Note: The "No generation for analysis" and "No generation for design" tickboxes are used to indicate whether or not these tagged values should be applied to the element selected in the explorer.

The "Java" tab

The "Java" tab of the properties editor contains fields specific to the generation of Java code, as well as buttons to launch code generation, visualization or compilation.

For further information on this tab, please refer to the "*The properties editor and the Java module*" section in chapter 2 of the *Objecteering/Java Developer* user guide.

The "C++" tab

The "C++" tab of the properties editor contains fields specific to the generation of C++ code.

For further information on this tab, please refer to the "*The properties editor and the Objecteering/C++ module*" section in chapter 2 of the *Objecteering/C++ Developer* user guide.

The "VB" tab

The "VB" tab of the properties editor contains fields specific to the generation of VB code, as well as buttons to launch code generation and visualization.

For further information on this tab, please refer to the "*The properties editor and the Objecteering/Visual Basic module*" section in chapter 2 of the *Objecteering/VB Developer* user guide.

Terminal element creation icons







Presentation
















Terminal elements are elements which cannot be decomposed. They are generally notes, graphics and tagged values, and are created in the "*Items*" and "*Diagrams*" tabs of the properties editor.

For a class, the terminal elements which can be created are as follows:

- ◆ constraints
- ◆ notes
- ◆ detailed graphic views
- ◆ tagged values
- ◆ work products (documentation, generated source C++, etc)
- ◆ stereotypes

Terminal elements

The ... icon	is used to	for ...
	add a note	all structural elements.
	associate a tagged value	all structural elements.
	associate a stereotype	all structural elements.
	create a constraint	all structural elements.
	reference a unit	a package.
	create a generalization link between: <ul style="list-style-type: none"> ◆ two classes ◆ two packages ◆ two use cases or two actors ◆ two nodes ◆ two signals ◆ two datatypes 	a package, a class, an actor, a use case, a node, a data type.
The ... icon	is used to	for ...

	create a use link	a package, a class, a operation
	create an implementation link	a class, a component
	create a communication link	an actor, a use case
	redefine a operation	a operation
	deploy a component	a node
	reference a signal	a data flow
	create a class diagram	a package, a class
	create a use case diagram	a package, a class
	create a sequence diagram	a package, a class, a use case
	create an object diagram	a package, a class
	create a deployment diagram	a package
	create a deployment instance diagram	a package
	create a state diagram	a state machine
	create a collaboration diagram	a collaboration
	create an activity diagram	an activity graph

Visualizing messages in the console

Visualizing an error message or a warning

When Objecteering/UML picks up an error during an operation, this error is displayed in the console, with any warnings which are pertinent (as shown in Figure 4-13).

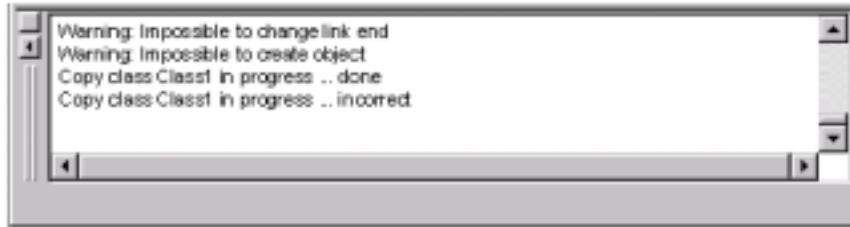


Figure 4-13. Console containing error messages and warnings

Visualizing traces of operations in the console

For all operations carried out on an element, a work product, etc, the console displays the progress of the operation.

During generation on a work product , the console displays its progress and then displays the final result of the generation (as shown in Figure 4-14).

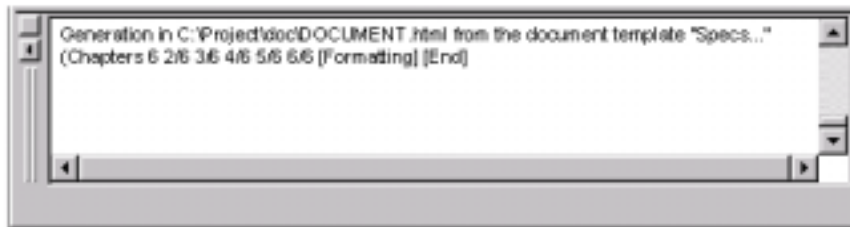


Figure 4-14. Visualizing the result of documentation generation

Messages and import traces in the console

During an import (UML modeling project, module, etc.), the console displays the progress of the import (as shown in Figure 4-15), as well as the final result. If the import of a component has not been correctly carried out, the console displays the errors which have occurred.

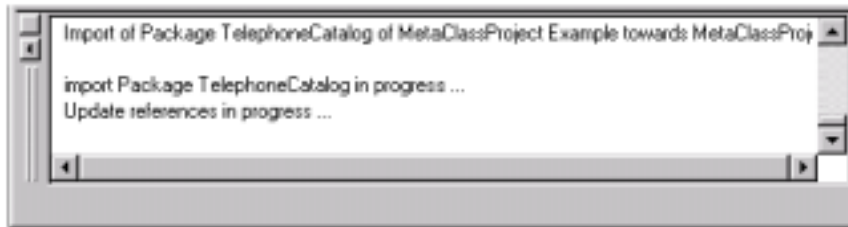


Figure 4-15. Display of import progress traces

Using the on-line help search engine

Launching the search engine

To launch the *Objectteering/UML* on-line help search engine, the user has two options:

- ◆ either launch the search engine, by clicking on the "Search Engine" menu entry in the "Start/Programs/Objectteering" menu
- ◆ or click on the "?" menu within *Objectteering/UML* itself (as shown in Figure 4-16 below)



Figure 4-16. Launching the on-line help search engine

Steps:

- 1 - Click on the "?" menu.
- 2 - Click on "Search engine".
- 3 - The *Objectteering/UML* on-line help search engine is then started up.

The left-hand part of the search engine window is used to:

- ◆ enter the information you wish to search for
- ◆ specify the on-line help tomes in which you wish to carry out your search

The right-hand part of the search engine window is used to display the relevant pages of the on-line help, through the hypertext links which appear after a successful search.

Searching for information

The following example (Figure 4-17) will show you just how easy it is to search for information within the *Objectteering/UML* on-line help, using the on-line help search engine.

Let's imagine we want to look for information on the `{JavaStatic}` tagged value.



Figure 4-17. Searching for information using the on-line help search engine

Steps:

- 1 - In the first entry field, enter "JavaStatic".
- 2 - Select the on-line help tome in which you wish to run the search. In this case, we have selected the *Objectteering/Java* on-line help. You can also run a search in the entire on-line help database, by selecting "All volumes...".
- 3 - Click on the "Search" button.

The result of this search is shown below in Figure 4-18.



Figure 4-18. The result of the search carried out

To access the on-line help sections which contain the information you are looking for, simply click on the hypertext links. The relevant section then appears in the right-hand side of the search engine window.

Working with Objecteering/UML macros

Selecting the Objecteering/Macros module

In Objecteering/UML, macros are developed and implemented using the *Objecteering/Macros* module, which is automatically installed during installation of the Objecteering/UML tool itself.

To use the *Objecteering/Macros* module, carry out the steps shown below (Figure 4-19).

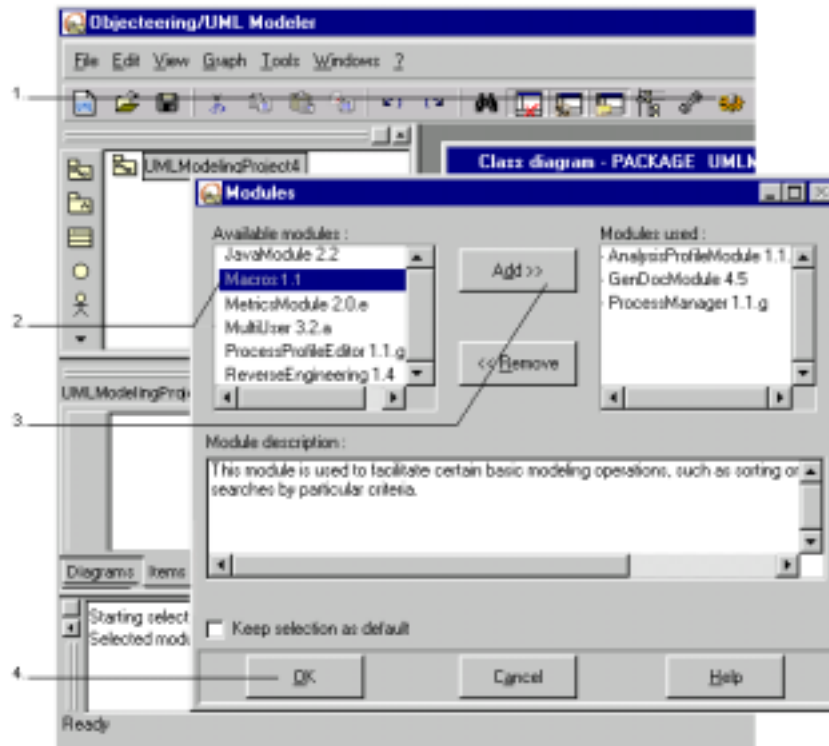



Figure 4-19. Selecting the *Macros* module

Steps:

- 1 - Click on the  "UML modeling project modules" icon.
 - 2 - Select the *Macros* module.
 - 3 - Click on "Add". The *Macros* module is then transferred from the left-hand "Available modules" list to the right-hand "Modules used" list.
 - 4 - Confirm by clicking on "OK".
- The *Objecteering/Macros* module can now be used!

Parameterizing the Objectteering/Macros module

The *Objectteering/Macros* module has one sub-section containing two parameters (as shown in Figure 4-20).

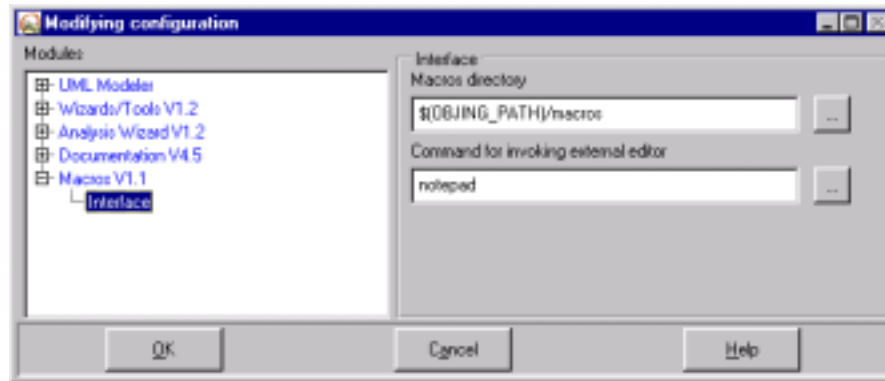




Figure 4-20. Parameters of the *Macros* module

Key:


- ◆ "*Macros directory*": This field is used to indicate where standard Objectteering/UML macros, as well as macros created by the user, are stored. By default, it is set to \$OBJING_PATH/macros. To change this directory, click on the  icon to open a file browser, and select the directory of your choice.
- ◆ "*Command for invoking external editor*": Macros can be edited in Objectteering/UML itself. The user can, however, choose to edit his macros using a tool other than Objectteering/UML, such as Word, Wordpad or Notepad. To select this tool, click on the  icon to open a file browser, and make your selection. Where this parameter has been defined, the tool specified is automatically used to open macro file, and takes precedence over the optional Windows jmf file association.

Note: Objectteering/UML macro files take the *.jmf* suffix, meaning *J macro file*.

Macro commands

Once the *Objecteering/Macros* module has been selected, four macro operations can be carried out:

- ◆ executing a macro
- ◆ creating a macro
- ◆ modifying a macro
- ◆ deleting a macro

To access macro commands, the user can either activate the "*Macros*" context menu, available by right-clicking on an element, or click on the  "*Macro commands*" icon situated in the toolbar, to open the "*Macro commands*" window.

The "Macro commands" window

The "Macro commands" window interface changes according to your *Objectteering/UML* installation mode.

The macros contained in "Local" are those which can be accessed from the current workstation, whilst those contained in "Server" are the macros which can be run from the server.

In "Local", macros can be executed, created, modified or deleted. In "Server", macros can only be executed.

Figure 4-21 shows the "Macro commands" window for a heavyweight/lightweight client installation.

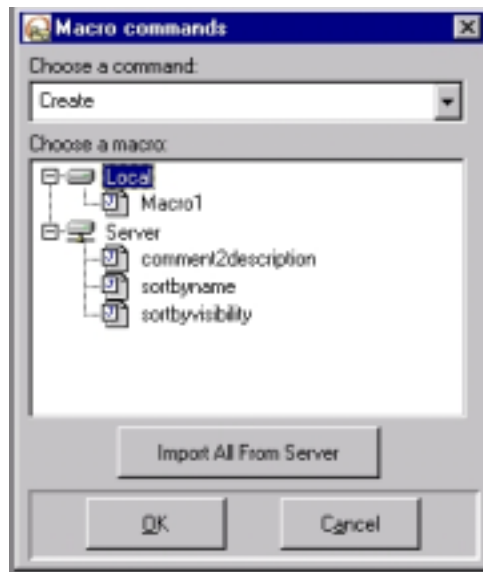


Figure 4-21. The "Macro commands" window in heavyweight or lightweight client mode

The "Import all from server" button is used to import all macros from the server onto the current workstation.

Figure 4-22 shows the "Macro commands" window for a standalone installation.

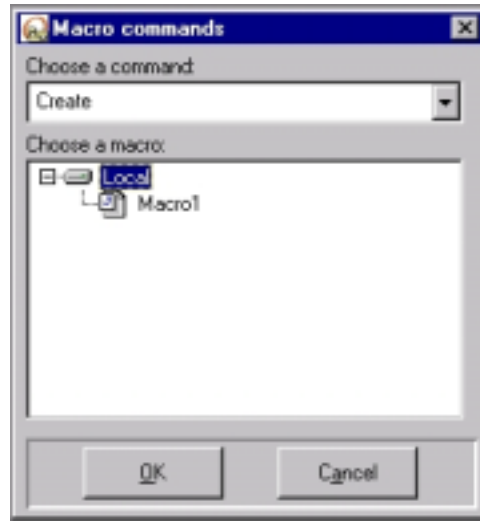


Figure 4-22. The "Macro commands" window in standalone mode

Figure 4-23 shows the "Macro commands" window for a server installation.

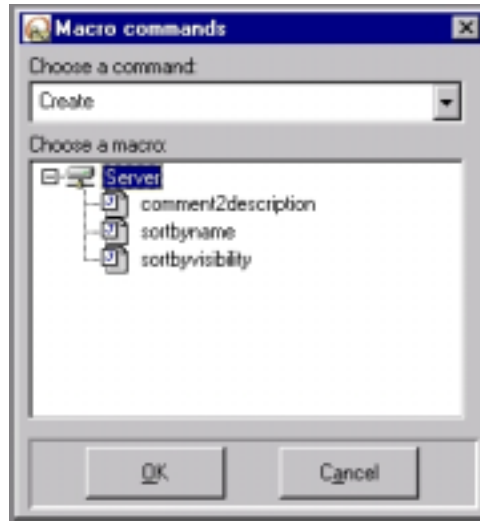



Figure 4-23. The "Macro commands" window for a server installation

Executing macros

Objecteering/UML comes with a number of standard macros, allowing you to:

- ◆ sort model elements by name
- ◆ sort model elements by visibility
- ◆ transform comment notes into description notes

To run an existing macro, you can either run it from the context menu or by clicking on the  "Macro commands" icon. Figure 4-24 below shows an existing macro being run from the context menu.

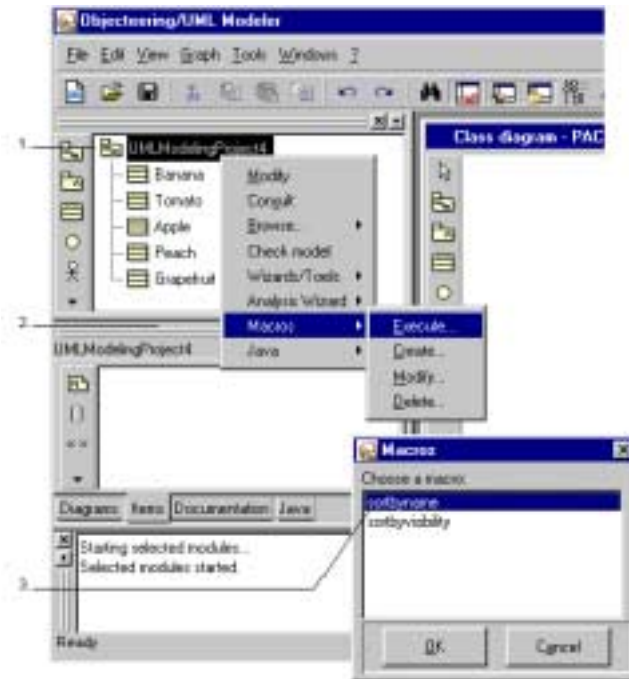




Figure 4-24. Running the "sortbyname" macro on the "UMLModelingProject4" package

Chapter 4: Functions of Objectteering/UML Modeler - Detailed View

Steps:

- 1 - Select the "*UMLModelingProject4*" package in the explorer by right-clicking.
- 2 - Select the "*Macros/Execute...*" commands from the context menu which then appears. The "*Macros*" window then appears.
- 3 - Select the "*sortbyname*" macro. The macro is then run on the selected element and its contents.

Note: To run this command through the  "*Macro commands*" icon, simply click on the "*UMLModelingProject4*" package in the explorer, click on the  icon, select the "*sortbyname*" macro and click on "OK".

The result (the package's classes are now displayed in alphabetical order) is shown below (Figure 4-25).

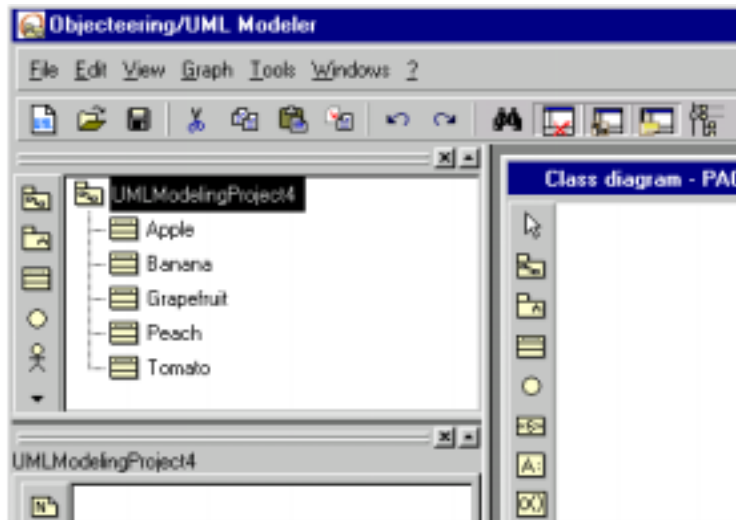




Figure 4-25. The result of running the "*sortbyname*" macro on the "*UMLModelingProject4*" package - its classes are now displayed in alphabetical order

Creating macros

To create your own macros, you can either create them from the context menu or by clicking on the  "Macro commands" icon. Figure 4-26 below shows a new macro being created through the  icon.

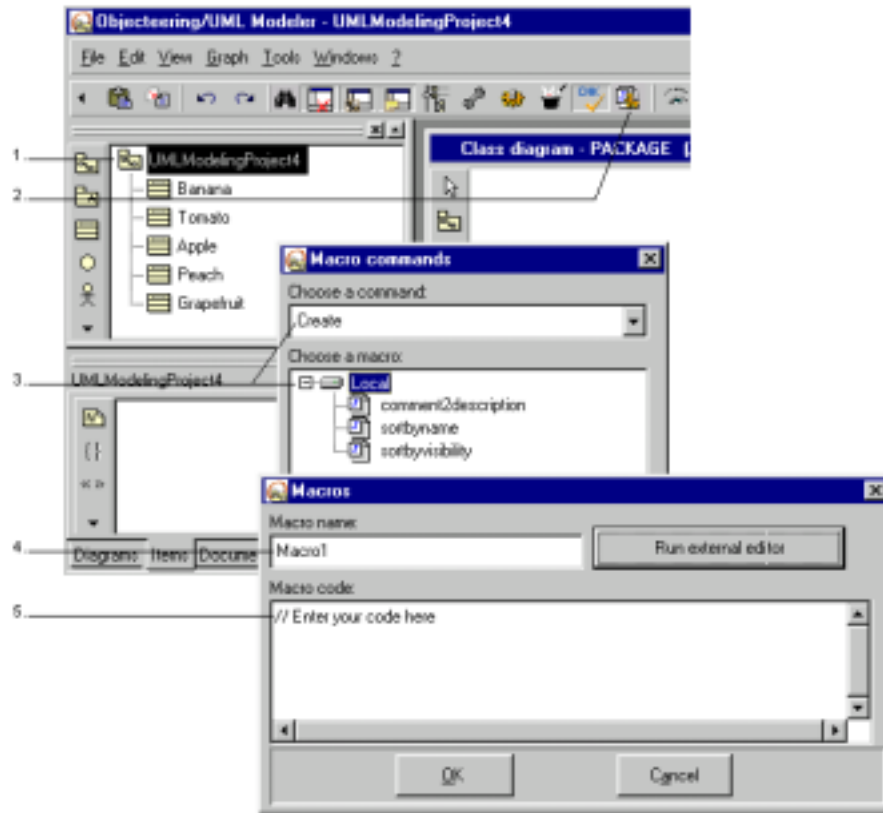



Figure 4-26. Creating the "Macro1" macro

Steps:

- 1 - Select the "*UMLModelingProject4*" package in the explorer.
- 2 - Click on the  "*Macro commands*" icon in the toolbar.
- 3 - Click on "*Local*" and select the "*Create*" command in the combobox. The "*Macros*" window then appears.
- 4 - Enter a name for your macro. Macro file names must be in lower case and must not contain blanks.
- 5 - Enter the necessary J code, and confirm by clicking on "*OK*".

Note 1: If you prefer to enter your J code using an external text editor (which has been previously specified at module parameter configuration level), simply click on the "*Run external editor*" button. This opens the text editor specified at module parameter configuration level (see Figure 4-20 above).

Note 2: To run this command through the context menu, simply right-click on "*UMLModelingProject4*" in the explorer, select the "*Macros/Create...*" commands from the context menu and then continue as described above.


The following guidelines should be respected when creating your own macros:

- ◆ Macro files should have a header in the following form:

```
/*
 * <file name>
 *
 * Version: <Macro version>
 * Date: <Last modified>
 * Author: Copyright (c) Softeam 1996-2001. All rights
 reserved.
 *
 * Valid for: <Metaclass on which the macro can be run>.
 *
 * <Description of the macro>
 *
 */
```

- ◆ Macro file J code should be clearly annotated, in order to facilitate comprehension of the macro.
- ◆ Code lines should not be longer than 80 characters. Where this is the case, lines are split according to Java coding conventions.
- ◆ The "*return*" and "*exit*" instructions must not be used.
- ◆ The macro must check those metaclasses on which it can be executed, so as not to produce J errors. Where a metaclass is not valid, the following error message occurs: "*Error: this macro is not available on <metaclass> metaclass*".

Modifying macros

To modify existing macros, you can either modify them through the "Macros/Modify..." context menu command, or through the  "Macro commands" icon. Figure 4-27 shows a macro being modified through the context menu.

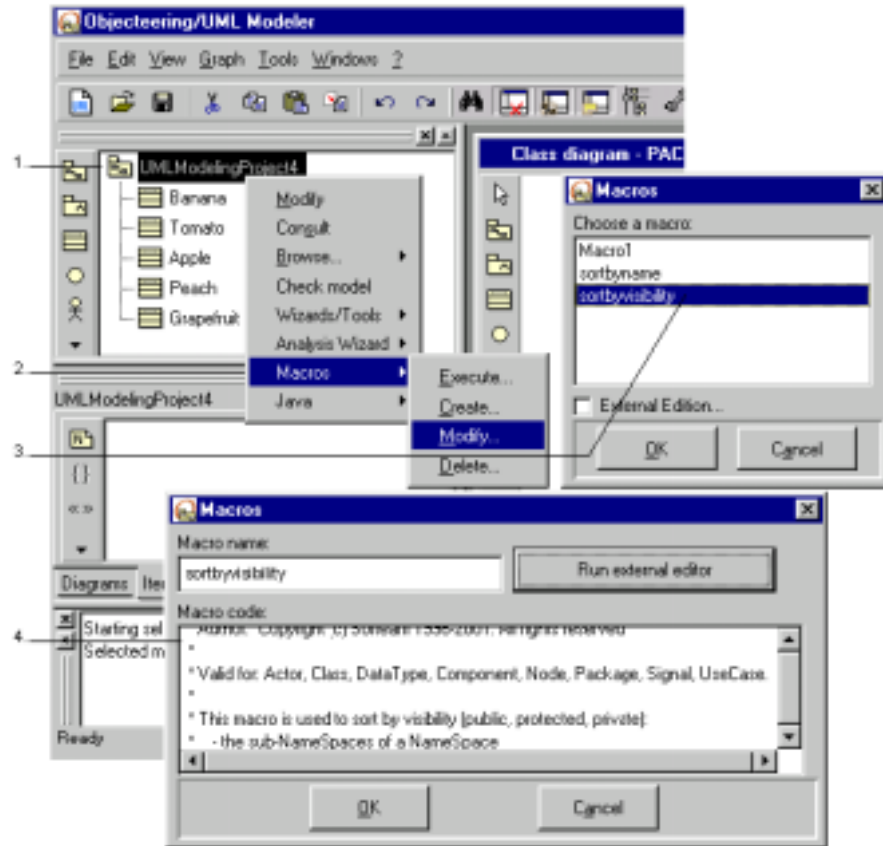






Figure 4-27. Editing the "sortbyvisibility" macro

Steps:

- 1 - Select the "*UMLModelingProject4*" package in the explorer by right-clicking.
- 2 - Select the "*Macros/Modify...*" commands from the context menu which then appears. The "*Macros*" window then appears.
- 3 - Select the "*sortbyvisibility*" macro. If you wish to open this macro file using an external editor, check the "*External Edition...*" tickbox. The file will then be opened using the text editor specified at module parameter configuration level.
- 4 - Make the desired modifications to the J code and confirm by clicking on the "*OK*" button.

Note: To modify the macro through the  "*Macro commands*" icon, simply select the "*UMLModelingProject4*" package in the explorer, click on the  icon, select the "*sortbyvisibility*" macro, select "*Modify*" in the combobox and click on "*OK*". Then continue with the steps described above.

Deleting macros

To modify existing macros, you can either modify them through the "Macros/Delete..." context menu command, or through the  "Macro commands" icon. Figure 4-28 shows a macro being modified through the  icon.

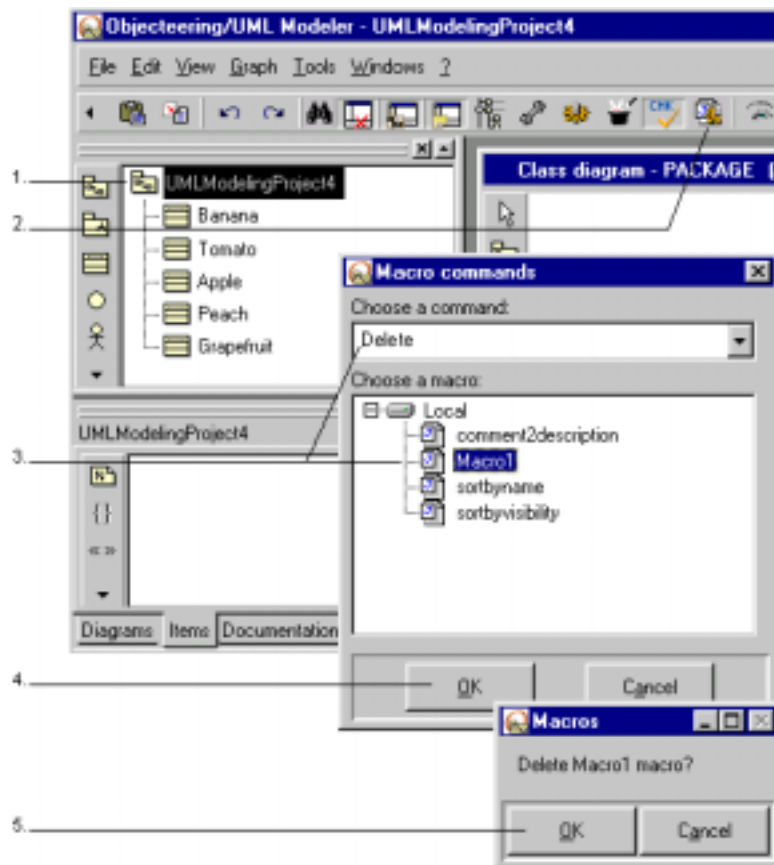



Figure 4-28. Deleting a macro

Steps:

- 1 - Select the "*UMLModelingProject4*" package in the explorer.
- 2 - Click on the  "*Macro commands*" icon.
- 3 - Click on "*Macro1*" and select "*Delete*" in the combobox.
- 4 - Confirm by clicking on the "*OK*" button.
- 5 - A confirmation box will then appear, asking you to confirm deletion of the macro. Click on the "*OK*" button to confirm. The macro will then be deleted.

Note: To run this command through the context menu, simply right-click on "*UMLModelingProject4*" in the explorer, select the "*Macros/Delete...*" commands from the context menu and then continue as described above.

UML Modeler parameter sets

The "Formalism" set

The parameters displayed in the "*Edit configuration*" dialog box allow you to define general formalism options edited in diagrams (as shown in Figure 4-29).

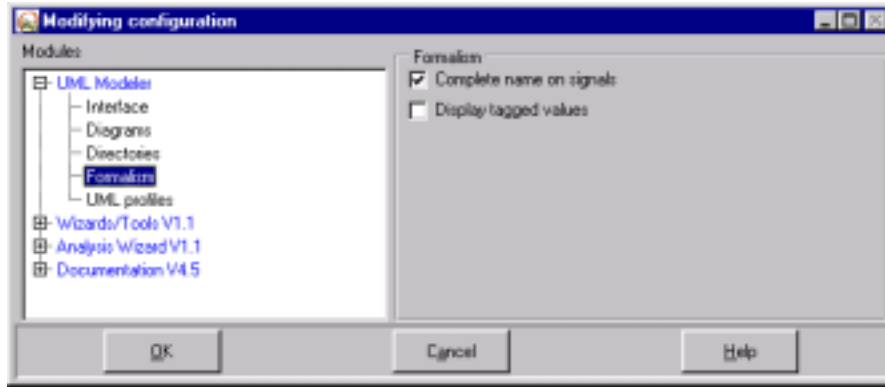


Figure 4-29. Editing "Formalism" configuration

The "Formalism" set: Description

The ... field or button	allows you to ...
Complete name on signals	use long names (flow name and associated class name) instead of short names (only the flow name) of the object flows or events on the graphic views.
Display tagged values	display tagged values when a new diagram is created.

The "Interface" set

This dialog box (shown in Figure 4-30) presents general interface presentation options.

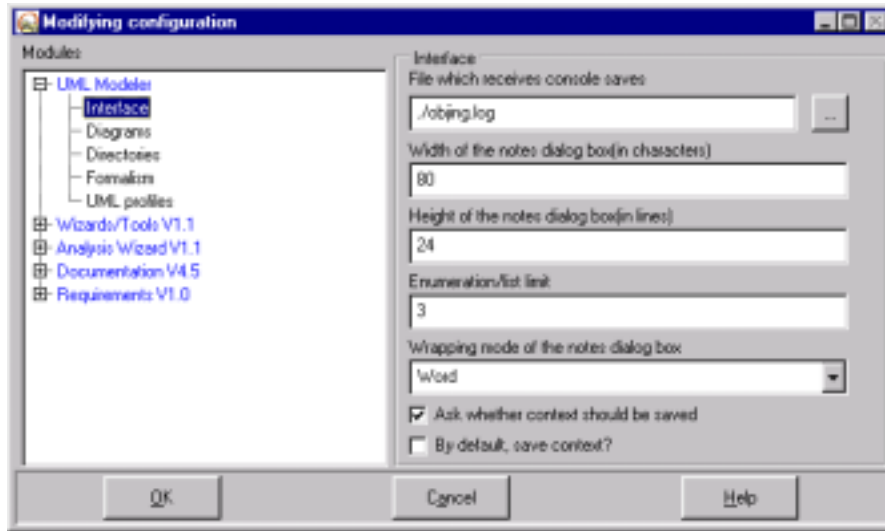



Figure 4-30. Editing "Interface" configuration

The "Interface" set: Description

The ... field or button	allows you to ...
File which receives console saves	modify the name or path of the file which contains the history of all information saved from the console.
Width of the notes dialog box (in characters)	specify the width in number of characters of the dialog box which allows the entry of notes. <u>Note:</u> the actual number can vary slightly according to the platform
Height of the notes dialog box (in lines)	specify the height in lines of the dialog box which allows the entry of notes.
Enumeration/list limit	fix the limit from which a choice passes from radio button enumeration form to list form in the entry dialog boxes.
Wrapping mode of the notes dialog box	define the wrapping mode : none, on the character or on the word.
Ask whether context should be saved?	specify whether or not you wish Objectteering/UML to propose a context save when you save or quit.
By default, save context?	indicate whether you want to save the context by default or not save the context by default.

Note: The  "Browse" icon is used to open a file browser, making it easier for you to specify directory and/or file names and path.

The "Diagrams" set

This dialog box (shown in Figure 4-31) allows you to parameterize the main graphic options of the element.

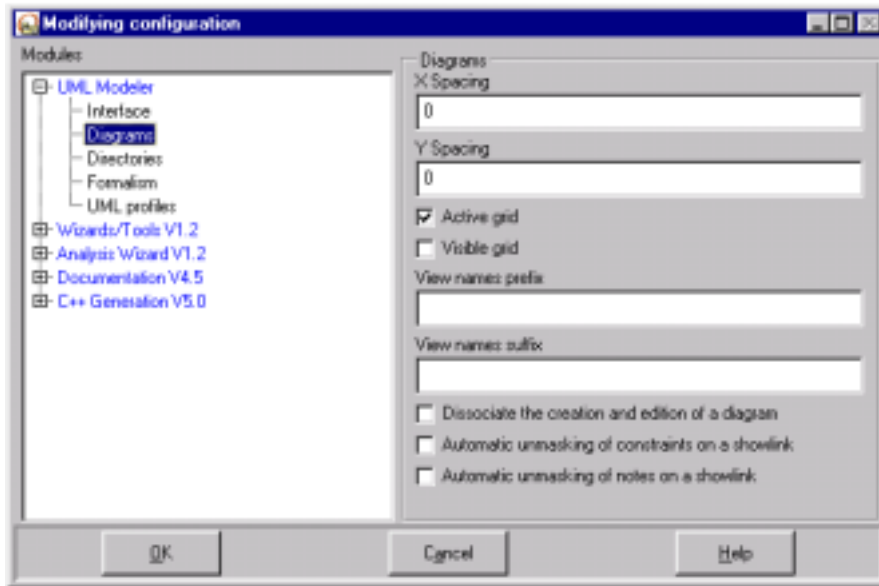


Figure 4-31. Editing "Diagrams" configuration

The "Diagrams" set: Description

Diagrams allow the activation of a magnetic grid which is used to help position objects (for further details, please refer to the "*Options menu*" section in this user guide).

The ... field or button	allows you to ...
X Spacing	choose the spacing of the vertical background grid lines.
Y Spacing	choose the spacing of the horizontal background grid lines.
Active grid	activate the grid.
Visible grid	make the grid visible.
View names prefix	give a prefix to the naming of diagrams.
View names suffix	give a suffix to the naming of diagrams.
Dissociate the creation and edition of a diagram	dissociate the creation from the opening of the diagram.
Automatic unmasking of constraints on a showlink	specify whether or not constraints should be automatically unmasked on a showlink operation.
Automatic unmasking of notes on a showlink	specify whether or not notes should be automatically unmasked on a showlink operation.

The "Directories" set

This dialog box (shown in Figure 4-32) allows the user to specify the default directory to be used for all generation. Objecteering/UML generation modules such as *RDB*, *C++* or *Java* will, by default, use this root directory.

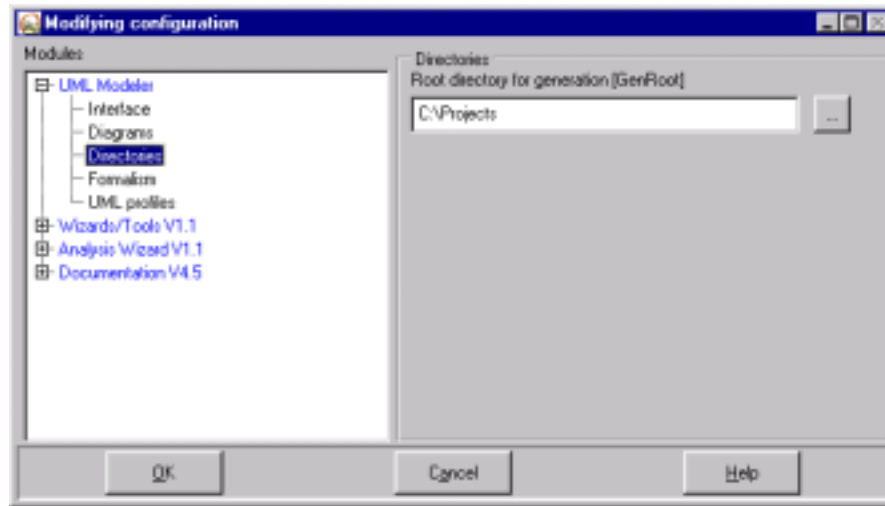



Figure 4-32. Editing "Directories" configuration

The "Directories" set: Description

The ... field or button	allows you to ...
Root directory for generation (GenRoot)	give the path of the root directory in which the generation of modules will happen. This path is memorized in the GenRoot variable, which is used to construct different sub-directories for the different types of generation, for example \$(GenRoot) for the generation of C++ files.

Note: The  "Browse" icon is used to open a file browser, making it easier for you to specify directory and/or file names and path.

The "UML Profiles" set

This dialog box (shown in Figure 4-33) allows you to define the UML profile used by the UML Modeler interface and the UML profile used to start and stop Objecteering/UML.

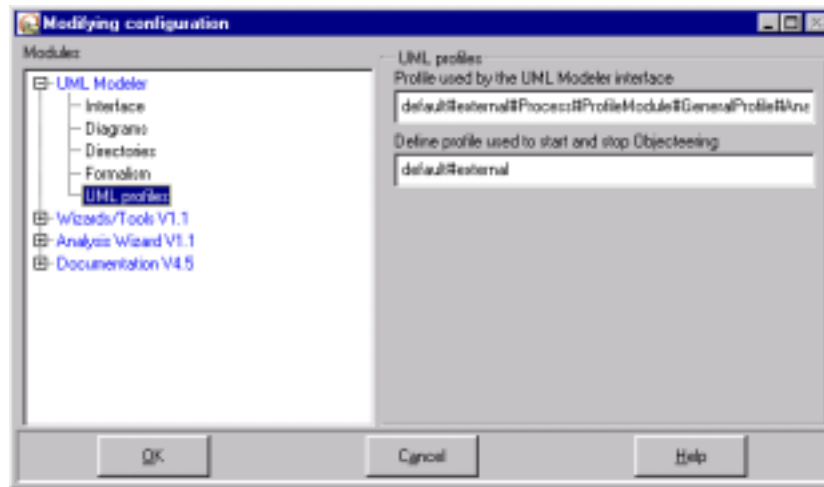


Figure 4-33. Editing "UML profiles" configuration

The "UML Profiles" set: Description

The ... field or button	is used to ...
Profile used by the UML Modeler interface	contain the rule for calculating diagram names by default.
Define profile used to start and stop Objecteering.	Indicate the profile used to start and stop Objecteering/UML.

Using the search function

Launching the search function

To open the "Search" window and launch a search, the user has three possibilities:


- ◆ using the "Edit/Search" menu (as shown in Figure 4-34)
- ◆ clicking on the  "Search" icon in the *Objectteering/UML* toolbar
- ◆ pressing CTRL F, either from a specific element or from anywhere within *Objectteering/UML* (for further information on keyboard shortcuts, please refer to the "Shortcuts" section in the current chapter of this user guide)

Figure 4-34 illustrates the first of these methods.



Figure 4-34. Launching the "Search" window through the "Edit/Search" menu

Steps:

- 1 - Click on the "Edit" menu and select the "Search..." command.

Simple search mode

The *Objecteering/UML* search function operates in two different modes, simple and advanced.

The simple search mode allows you to search for expressions, with or without distinguishing between uppercase and lowercase characters. You can specify whether you want to search for elements which begin or end with the expression, which contain the expression or which exactly match the expression.

Searches are launched from either a particular element or from the predominant *Objecteering/UML* structure according to which *Objecteering/UML* tool you are using (a model when using *Objecteering/UML Modeler* or a metamodel when using *Objecteering/UML Profile Builder*).

The simple search mode uses the "Search" window shown in Figure 4-35.

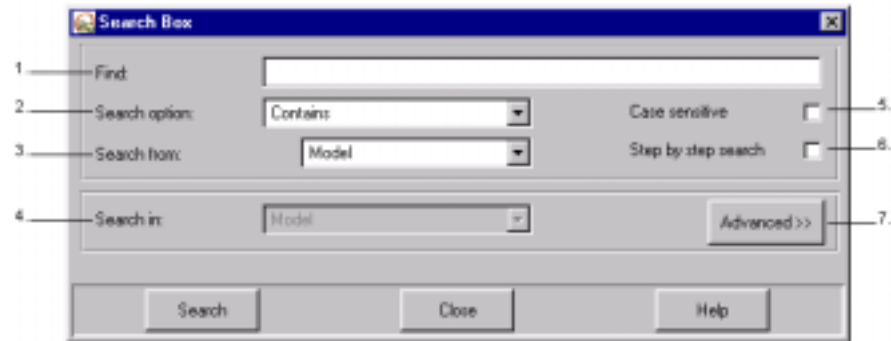


Figure 4-35. The "Search" window in simple mode

Key:

- 1 - The "*Find*" field is where you should enter the expression you wish to search for.
- 2 - The "*Search options*" field is used to select a search option, from "*Contains*", "*Begins with...*", "*Ends with...*" and "*Matches exactly*".
- 3 - The "*Search from*" field is used to indicate where you wish to search from (either the current element, the model or the metamodel). If you want to search from a particular element, you can drag it directly into the "*Search from*" zone.
- 4 - The "*Search in*" field is used to indicate where you wish to search. In simple mode, this field is not accessible and is automatically set to "*Model*".
- 5 - The "*Case sensitive*" checkbox is used to indicate whether or not your search should distinguish between uppercase and lowercase characters.
- 6 - The "*Step by step search*" checkbox, when checked, lets you jump from the first instance of the element searched for to the second to the third, and so on.
- 7 - The "*Advanced>>*" key is used to switch to the advanced search mode, thereby extending the "*Search box*" and making additional fields available.

The "*Search from*" field

The "*Search from*" field in the "*Search*" window is used to indicate where you wish to run your search from (a particular element, the model or the metamodel).

To run your search from a particular element, you have two options:

- ◆ If the "*Search*" window is not open, select the element and press CTRL F. The "*Search*" window then appears, with the selected element appearing automatically in the "*Search from*" field.
- ◆ If the "*Search*" window is already open, select the element, drag it to the "*Search*" window and drop it in the "*Search from*" field.


Example of a simple search

Figure 4-36 provides an example of a simple search for the expression "Training" in the model.



Figure 4-36. Carrying out a simple search

Steps:

- 1 - Click on the  "Search" icon to open the "Search" box.
- 2 - Enter the expression you want to search for, in our example "Training".
- 3 - Choose your search option mode ("Contains", "Begins with...", "Ends with..." or "Matches exactly"), in our example "Contains".
- 4 - Choose where you want your search to be carried out.
- 5 - Click on "Search" to launch the search operation.

For our model, the following results are obtained (Figure 4-37).

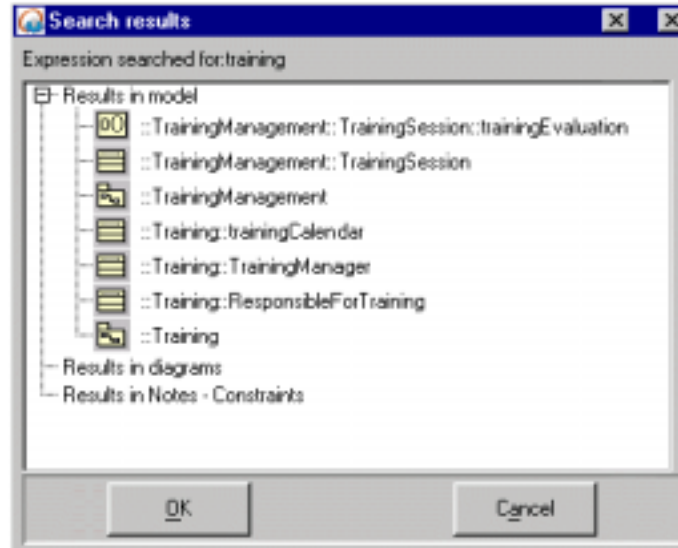


Figure 4-37. Search results

Results are shown with their associated icon, indicating the type of element concerned (package, class, operation, and so on).

Note: If we had checked the "Case sensitive" tickbox when searching for "Training", the "trainingEvaluation" operation and the "trainingCalendar" class would not have figured in the "Search results" window.

If you double-click on a search result, you automatically jump to the element concerned (as shown in Figure 4-38).

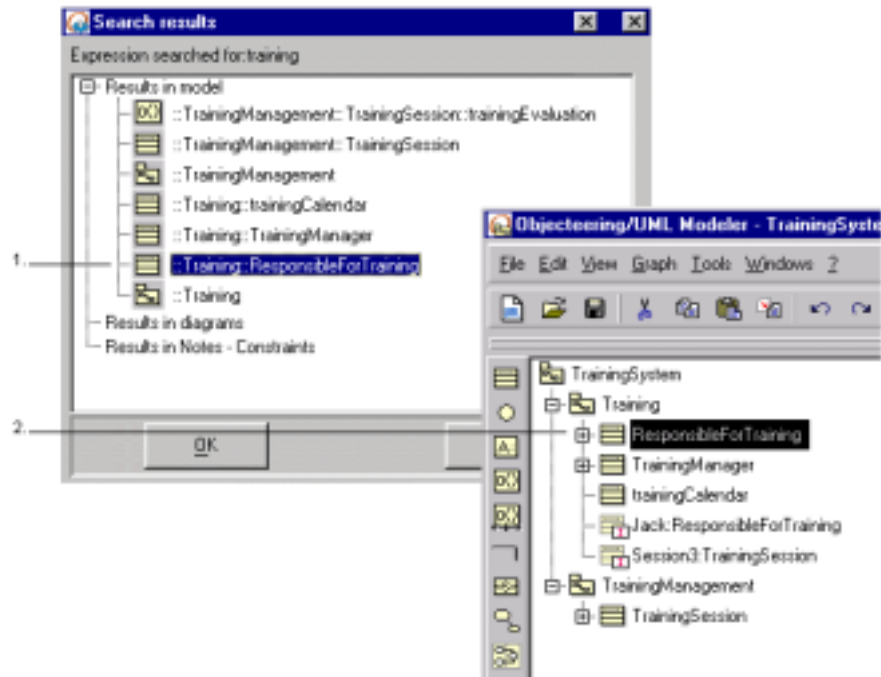


Figure 4-38. Jumping to elements from the "Search results" window

Steps:

- 1 - In the "Search results" window, select the element you wish to jump to.
- 2 - The focus then shifts to the requested element.

Advanced search mode

As previously explained, the *Objectteering/UML* search function operates in two different modes, simple and advanced.

The advanced search mode uses the same "Search" window as the simple mode, but with a certain number of additional fields (as shown in Figure 4-39). To pass to advanced mode from simple mode, you should simply click on the "Advanced>>" button.

In the advanced mode, you have access to the same fields as with the simple mode, and you can also specify whether you want to run your search in the model, in notes and constraints, in diagrams or in all of the above. It is also possible to indicate the type of element you want to search for and define a filter condition using the J language.

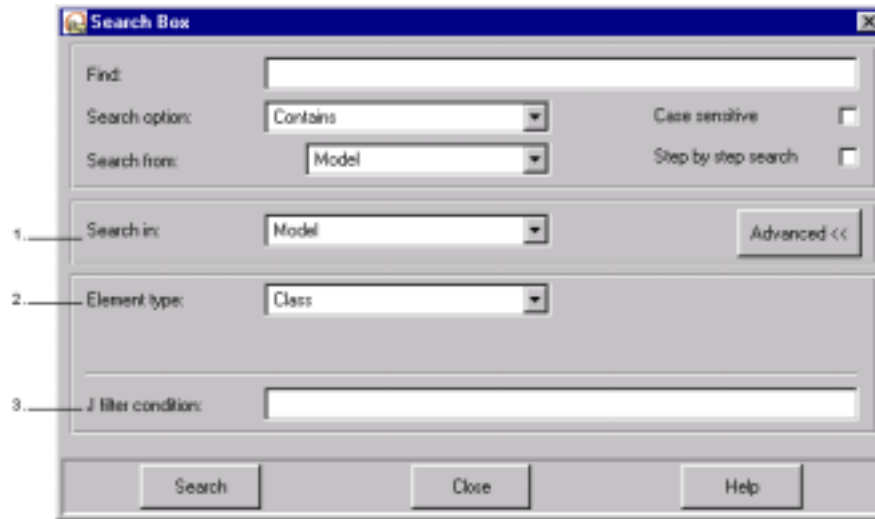


Figure 4-39. The "Search" window in advanced mode

Key:

- 1 - The "*Search in*" field (which is visible but not accessible in simple mode) is used to indicate whether you wish to search in the model, in notes and constraints, in diagrams or all of the above. If you select a search in diagrams or in notes and constraints, certain additional fields appear (please see below for more details).
- 2 - The "*Element type*" field is used to indicate the type of element you wish to search for. The available types depend on the type of search to be carried out.
- 3 - The "*J filter condition*" field is used to enter a J expression (which will return a boolean) which will be used to filter all those elements found. If this expression returns true, the element is retained. Otherwise, it is deleted from the results list.

Different types of search in advanced mode

In advanced mode, the following types of search are available with the *Objectteering/UML* search facility:

- ◆ search in the model (select "*Model*" in the "*Search in*" field of the "*Search*" window)
- ◆ search in notes and constraints (select "*Notes and constraints*" in the "*Search in*" field of the "*Search*" window)
- ◆ search in diagrams (select "*Diagrams*" in the "*Search in*" field of the "*Search*" window)
- ◆ search in all of the above (select "*All*" in the "*Search in*" field of the "*Search*" window)

Searching in the model

Running a search in the model allows you to browse the elements which figure in the model, in view of selecting them in an explorer. The elements found correspond to model or metamodel elements.

To run a search for an expression in the model, you should simply carry out the steps shown in Figure 4-40.

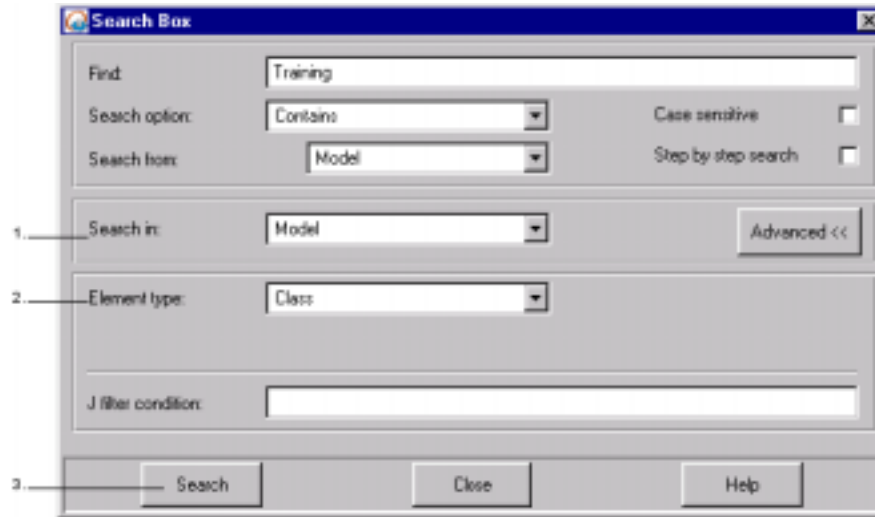


Figure 4-40. Searching in the model

Steps:

- 1 - After defining the expression you wish to find, as well as your search option and where you wish to search from, select "Model" in the "Search in" field.
- 2 - In the "Element type" field, select the element type you want to search for (in our example, we have indicated that we want to search for classes only).
- 3 - Click on "Search" to run the search.

Searching in notes and constraints

Running a search in notes and constraints allows you to browse textual elements and search inside notes and constraints.

To run a search for an expression in the model, you should simply carry out the steps shown in Figure 4-41.

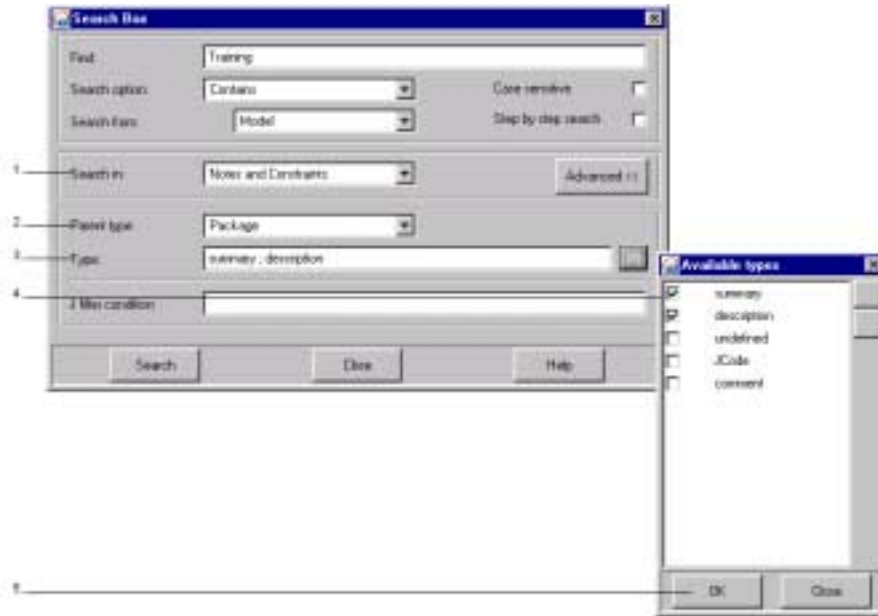



Figure 4-41. Searching in notes and constraints

Steps:

- 1 - After defining the expression you wish to find, as well as your search option and where you wish to search from, select "*Notes and constraints*" in the "*Search in*" field. This action opens the "*Parent type*" and "*Type*" fields (see step 3).
- 2 - In the "*Parent type*" field, select the type of the parent element inside which you want to search notes and constraints (in our example, we have indicated that we want to search notes and constraints belonging to packages).
- 3 - Click on the  "*Browse*" icon to the right of the "*Type*" field. This opens the "*Available types*" window.
- 4 - In the "*Available types*" window, select the types of note or constraint you wish to search. The two icons on the right of this window are used to select or unselect all the proposed types.
- 5 - Click on "*OK*" to confirm (the selected note and constraint types then appear in the "*Type*" field) and then click on "*Search*" to run your search.

Searching in diagrams

Running a search in diagrams allows you to only search for the required expression within the diagrams which exist within your UML modeling or profiling project.

To run a search for an expression in diagrams, you should simply carry out the steps shown in Figure 4-42.

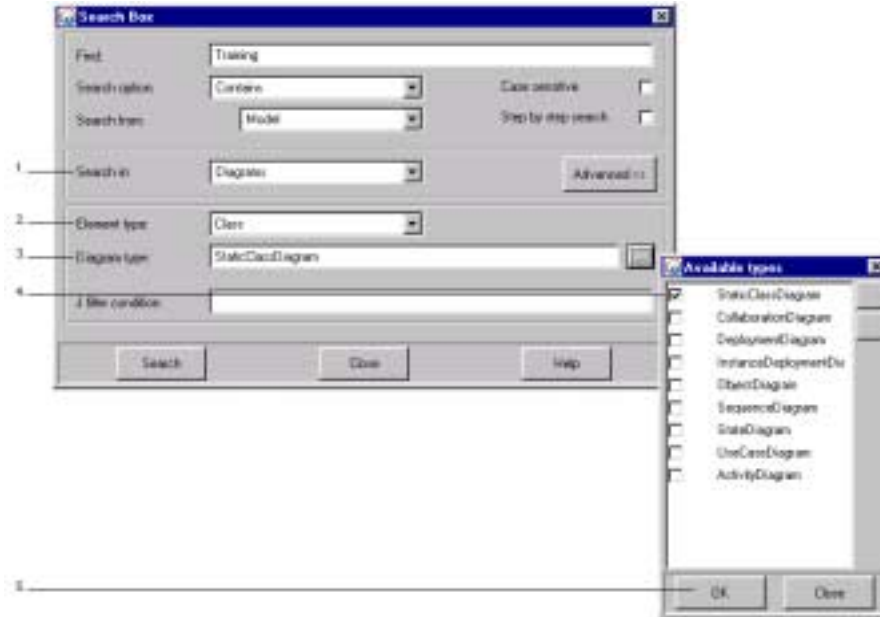



Figure 4-42. Searching in diagrams

Steps:

- 1 - After defining the expression you wish to find, as well as your search option and where you wish to search from, select "*Diagrams*" in the "*Search in*" field. This action opens the "*Diagram type*" field (see step 3).
- 2 - In the "*Element type*" field, select the element type you want to search for (in our example, we have indicated that we want to search for classes only).
- 3 - Click on the  "*Browse*" icon to the right of the "*Diagram type*" field. This opens the "*Available types*" window.
- 4 - In the "*Available types*" window, select the types of diagram inside which you want to search for your expression. The two icons on the right of this window are used to select or unselect all the proposed diagram types.
- 5 - Click on "*OK*" to confirm (the selected diagram types then appear in the "*Diagram type*" field) and then click on "*Search*" to run your search.

Searching in the model, notes, constraints and diagrams

To run a search in the model, notes, constraints and diagrams, simply carry out the steps shown in Figure 4-43.

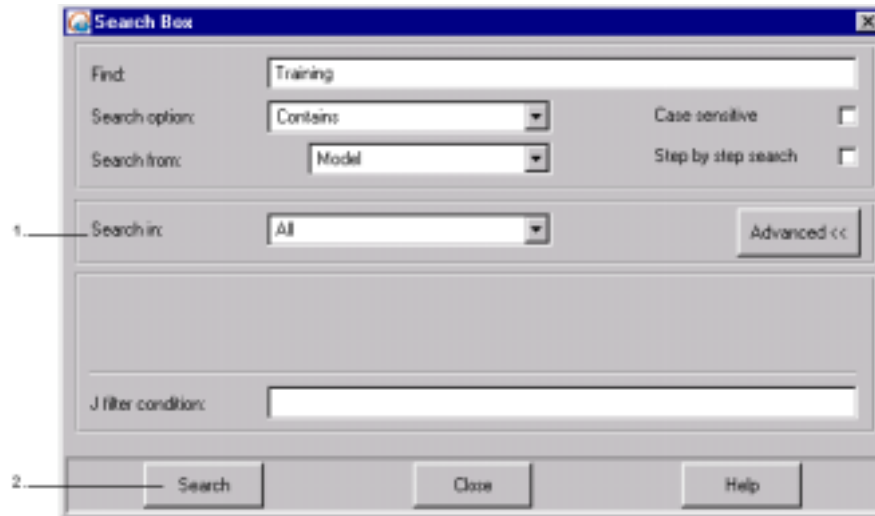


Figure 4-43. The "All" option in the "Search in" field

Steps:

- 1 - After defining the expression you wish to find, as well as your search option and where you wish to search from, select "All" in the "Search in" field.
- 2 - Click on "Search" to run the search.

Note: This type of search can be time-consuming, as it combines the three previously discussed types of search.

Example of an advanced search

Figure 4-44 provides an example of an advanced search for representations of the "ResponsibleForTraining" class in diagrams.

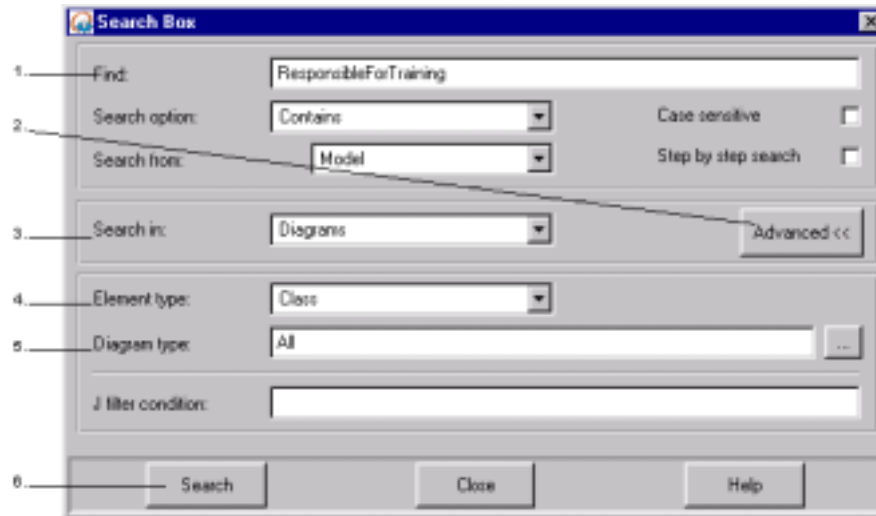


Figure 4-44. Carrying out a search in advanced mode

Steps:

- 1 - Enter the name of the element you wish to search for, in this case, the "ResponsibleForTraining" class.
- 2 - Click on the "Advanced>>" button to access the advanced search mode.
- 3 - Select "Diagrams" in the "Search in" field.
- 4 - Choose the type of element you wish to search for in the "Element type" field. In this case, we have specified that we are searching for a class.
- 5 - If necessary, define the different types of diagram you wish to search. In our example, we have chosen to search all types of diagram.
- 6 - Click on the "Search" button to run the search.

The result of this search is shown in Figure 4-45 below.

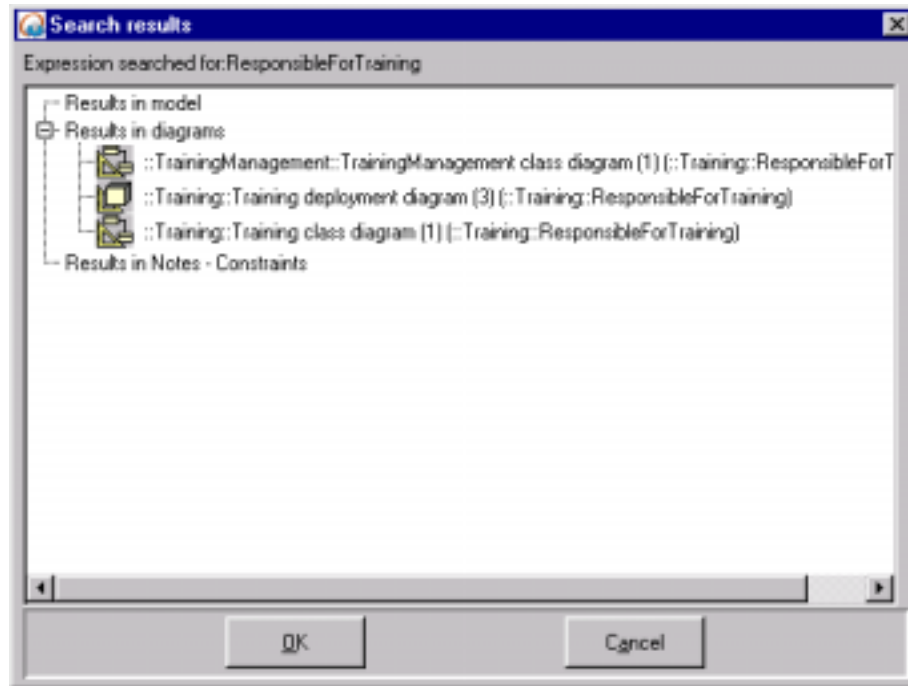




Figure 4-45. The results of the search

Read-only mode

Graphic representation of read-only elements

Read-only elements are displayed differently from read-write elements in the explorer, the properties editor and graphic editors. Where a module, for example the *Objecteering/Multi-user* module, has been registered as the module which manages the read-only mode, elements in read-only mode, i.e. those elements which may not be modified, are shown with the  icon superimposed over the icon representing the element in question.

However, where no module has been designated as the module which manages the read-only mode, modifiable elements are shown with the  icon superimposed over the icon representing the element in question. This is only the case at UML profiling project level.

Note 1: The read-only mode on links in graphic editors is not graphically represented. The same is true for attributes, operations, attribute links and attribute roles, since the information is already graphically represented on the class or the instance.

Note 2: There is no graphic representation of the read-only mode in state diagrams, since they are associated with state machines. If state machines are in read-only mode, all elements which compose them are also in read-only mode.

Handling elements and the read-only mode

Elements in read-only mode cannot trigger operations, for example, delete, cut or move, which are likely to modify either themselves and/or their components, and when model elements are put in read only mode, certain operations are no longer possible on the element(s) in question. For example, new sub-elements may only be created on an element in read-write mode. By default, all new elements are created in read-write mode.

To destroy a model element, both the element and its components must be in read-write mode.

As regards paste operations, all pasted elements are created in read-write mode, regardless of their state when cut or copied. The same is true for drag and drop operations.

Imported items are all created in read-write mode.

In Objectteering/UML's explorer, properties editor and graphic editors, several operations which are likely to modify the model are normally available via the menu, icons in the tool bar, context menus and creation palettes.

When the read-only mode has been activated, the following changes take place, with regard to the above possibilities:

- ◆ menu items which are no longer available are grayed out
- ◆ icons in the tool bar are deactivated
- ◆ context menu items are filtered, and only available menu items appear
- ◆ icons in the creation palette which are no longer available are grayed out

Links and the read-only mode

The addition or deletion of links on elements in read-only mode depends on the nature of the link in question. If the link is oriented (as shown in Figure 4-46), it can only be added or deleted if the origin element is in read-write mode, whilst the mode of the destination element is of no importance.



Figure 4-46. Adding an oriented link

The "Class1" class is in read-write mode, and it is, therefore, possible to add a dependency oriented from the "Class1" class towards the "Class2" class. The fact that "Class2" is in read-only mode has no impact on the addition of the dependency.



However, in order to add or delete a non-oriented link (as shown in Figure 4-47), all elements which feature in the link must be in read-write mode.



Figure 4-47. Adding a non-oriented link

Neither the actor nor the use case are in read-only mode, therefore a non-oriented communication link can be added.

Locally annotating read-only elements

Elements in read-only mode can be annotated using local tagged values and local notes. These local tagged values and local notes can be seen in the "Items" tab of the properties editor. They are represented by the  icon for local notes and the  icon for local tagged values. This graphic representation is very similar to that of classic tagged values and notes, the only difference being that their illustrations are shown in yellow.

Handling local tagged values and local notes

Local tagged values and local notes are subject to certain constraints in terms of their handling. Furthermore, they behave in a particular way when the elements they annotate are handled.

Local tagged values and local notes cannot be copied, pasted, moved or destroyed by the user. However, if the element they annotate is destroyed, they are automatically destroyed.

If an element is copied or imported, any local tagged values or local notes annotating the element in question are not carried over to the newly created element.

If an element is moved, any local tagged values or local notes annotating the element in question are also moved.

Shortcuts

The following keyboard shortcuts are available in Objecteering/UML:

The ... shortcut	available in...	is used to ...
Ctrl A	graphic editors	select all elements in the graphic editor.
Ctrl C	explorer	copy the selected element.
Ctrl D	explorer	move the selected element down one level.
Ctrl E	graphic editors	show the contents of the selected element.
Ctrl F	graphic editors and explorers	launch the search function.
Ctrl G	graphic editors	copy the graph image.
Ctrl I	graphic editors	reset the zoom function.
Ctrl K	main window	clear the contents of the console.
Ctrl L	graphic editors	show links on the selected element.
Ctrl M	graphic editors	mask the selected element.
Ctrl N	main window	create a new UML modeling project.
Ctrl O	main window	open an existing UML modeling project.
Ctrl P	graphic editors	print the current diagram.
Ctrl R	graphic editors	mask the contents of the selected element.
Ctrl S	graphic editors and explorers	carry out a save operation.

The ... shortcut	available in...	is used to ...
Ctrl U	explorer	move the selected element up one level.
Ctrl V	explorer	paste the selected element.
Ctrl X	explorer	cut the selected element.
Ctrl Y	graphic editors and explorers	redo the last cancelled operation.
Ctrl Z	graphic editors and explorers	undo the last operation carried out.
+	graphic editors	zoom in on a diagram.
-	graphic editors	zoom out of a diagram.

Chapter 5: Graphic Editors - General
Principles

Overview of the Objecteering/UML graphic editors

Introduction

Graphic editors are used to view and work with diagrams. They are also used to create and modify elements represented in the diagram.

Several types of diagram exist. In the 5.2.2 version of Objecteering/UML, the following types of diagram are supported:

- ◆ *Activity diagrams*: The activity diagram graphically illustrates activity graphs, which show a procedure or a workflow.
- ◆ *Class diagrams*: The class diagram allows you to present the internal structure of an element and its relationships with other elements.
- ◆ *Collaboration diagrams*: The collaboration diagram allows you to present exchanges of messages between roles.
- ◆ *Deployment diagrams*: The deployment diagram is used to represent the physical architecture of the system.
- ◆ *Deployment instance diagrams*: The deployment instance diagram is used to present a particular instance of deployment.
- ◆ *Object diagrams*: The object diagram is used to present a set of class instances with their links and the messages exchanged.
- ◆ *Sequence diagrams*: The sequence diagram is used to show how different objects cooperate.
- ◆ *State diagrams*: The state diagram allows you to describe the manner in which objects react to events.
- ◆ *Use case diagrams*: The use case diagram allows you to describe the most important services rendered by the system.

Objecteering/UML allows the creation of several diagrams of the same type on the same element. For example, several class diagrams can exist on the same package. We then talk of different views on the same model.

The graphic editors, the properties editor and the explorer allow the same model elements to be manipulated, created, modified and activated. Consistency is always maintained.

Every element is presented in the explorer, whereas in a diagram, only elements which a designer decides to present appear.

Editing window

Figure 5-1 below shows a class diagram.

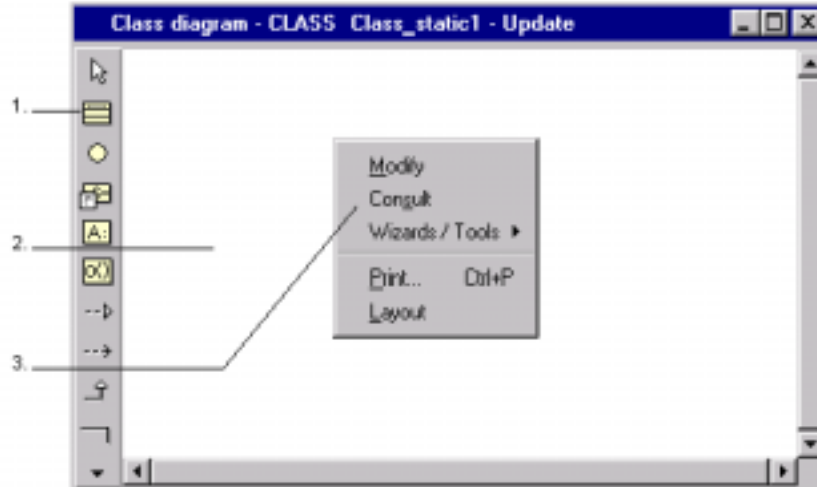



Figure 5-1. A class diagram

Key:

- 1 - Creation icon palette, where model elements to be created are selected.
- 2 - Diagram, which is the drawing area.
- 3 - Pop-up menu, which gives the functions possible on the diagram.

Menus and tool bar icons can also be used in graphic editors.

Note: If you are using modules which manage the read-only mode, certain elements in the explorer may be displayed with the  icon superimposed over the icon representing the element itself. This icon indicates that these items may not be modified. For further information on the read-only mode, please refer to the *Objecteering/UML Teamwork User Guide*.

Principal functions

Graphic editors allow you to:

- ◆ create or destroy elements
- ◆ modify or visualize elements
- ◆ move and lay out elements
- ◆ mask or unmask elements
- ◆ activate generation on elements
- ◆ navigate between diagrams
- ◆ change the presentation of elements (size, color, icons, etc)
- ◆ print or copy paste the diagram to the environment (external tools such as Word, etc.)

Launching a graphic editor

When a new UML modeling project is created, a class diagram is automatically created and opened.

A diagram is opened in a graphic editor by:

- ◆ double-clicking on the name of an existing diagram in the "*Diagrams*" tab of the properties editor
 - ◆ creating a diagram for an element, by clicking on the creation button in the "*Diagrams*" tab of the properties editor
-

Creating a graphic element

Graphic object categories

The graphic objects which correspond to the editor icons are as follows:

- ◆ boxes (packages, classes, instances, states, use cases, actors)
- ◆ links (associations, generalizations, uses, data flows, sequence messages, control transitions, trigger transitions, events, signals, co-operations, dependencies)

Creating several elements of the same kind

Very often, there is a need to create several different occurrences of the same kind of element.

For example, you may want to create several or several associations. The "CTRL" key, pressed during the selection of the creation icon, will activate the repeated entry system (as shown in Figure 5-2). Double-clicking on the icon has the same effect.

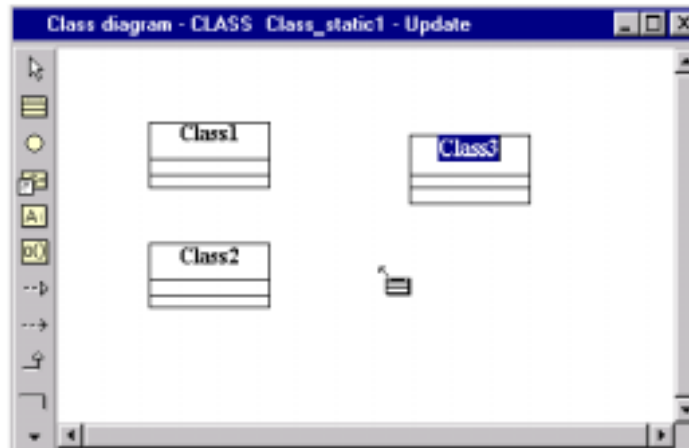


Figure 5-2. Creating several classes using the repeated entry system

Creating a box

To create a box:

- 1 - Click on the icon which corresponds to the type of element desired.
- 2 - Click in the graphic zone at the point where you wish to create the element.
- 3 - Enter the name directly in the highlighted zone.

A new element is then created at the selected location.

Example: Creating a class in a class diagram

To create a class in an editor, proceed as shown in Figure 5-3:

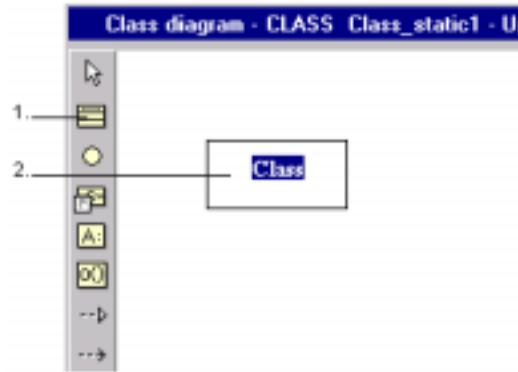



Figure 5-3. Creating a class

Steps:

- 1 - Click on the  "Create a class" button.
- 2 - Click in the diagram at the point where you wish to position the class, and enter the name directly in the highlighted text zone (which proposes the element name by default).

Creating a link

To create a link, carry out the following steps:

- 1 - Click on the icon which corresponds to the type of link desired.
- 2 - Click on the two elements between which you wish to create the link, in "*origin* - *destination*" order.
- 3 - A new link is created at the selected location.

Note: You can modify the values of this link directly in the diagram, or by double-clicking, an operation which will open the link modification entry box (Figure 5-4).

Example: Creating an association between two classes

We will now create an association between the "Class" class and the "Class1" class (as shown in Figure 5-4).

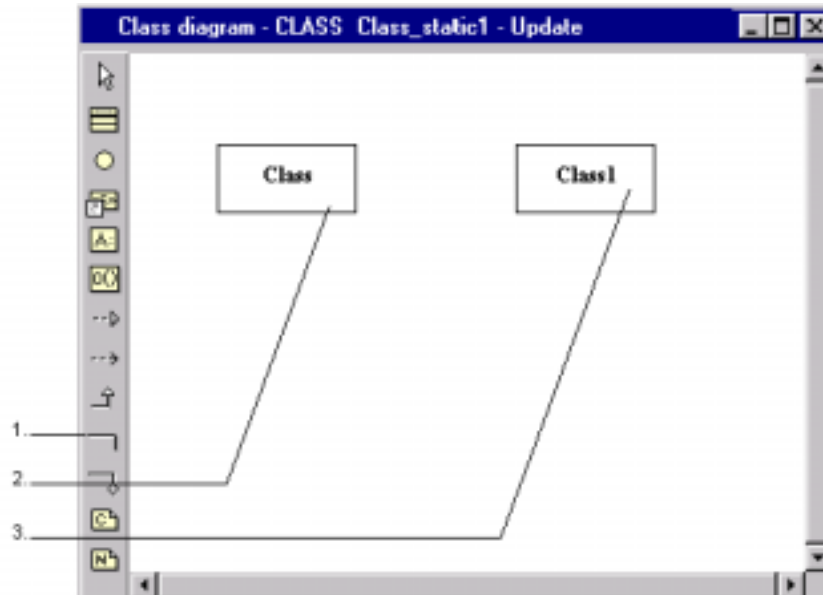



Figure 5-4. Creating an association between two classes

Steps :

- 1 - Click on the  "Create an association" button.
- 2 - Click on the start class.
- 3 - Click on the destination class.

Note: By default, the association will contain 0..* quantity values.

Modifying a link's values in a graphic editor

A link's values can be changed by editing the "Link" dialog box, or by directly entering over its values (as shown in Figure 5-5).

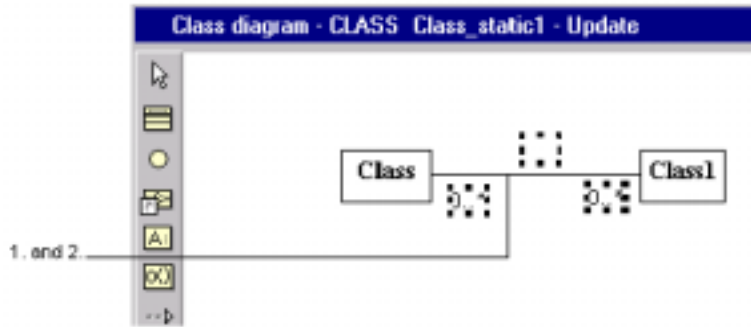


Figure 5-5. Direct modification of an association's values

Steps:

- 1 - Select the link.
 - 2 - Click once again on the link to make all the labels appear (including those which are empty).
 - 3 - Highlight the association's value.
 - 4 - Enter the new value directly over the highlighted zone.
-

Creating links


Different tracing of links

A link may be drawn in several different ways:

- ◆ orthogonal, which is a link containing one or more right angles
- ◆ free, which is a link presented in the manner of the user's choice
- ◆ shared target, which is a rake bar link, principally used in generalization

The choice of link affects the clarity of the diagram.

Color presentation of links

With Objecteering/UML, it is possible to present aspects of your diagram in different colors, through the "*Graph/Resources*" menu or the  "*Resources*" icon. For more information on this feature, please refer to the "*The 'Graph' menu*" section in chapter 6 of this user guide.

Drawing an orthogonal link

By default, the drawing of links is always free. To modify this tracing, use the "Shift" button as shown in the following steps (as shown in Figure 5-6).

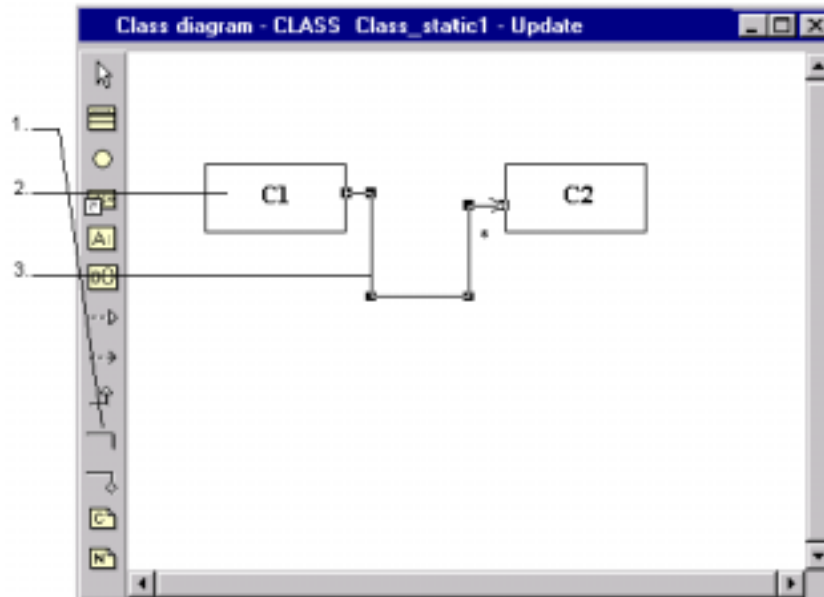



Figure 5-6. Drawing an association between two classes in orthogonal mode

Steps:

- 1 - Click on the  "Create an association" button.
 - 2 - Select the start class.
 - 3 - Holding down the "Shift" button, draw your link, with desired control points, and click on the destination class.
-

Redrawing links

Moving a link

To move a link, simply click on the link in question, whilst at the same time holding down the left mouse button. Move the link to the desired point, and then release the mouse button. The link has been moved to its new position.

If you wish to alter the form of a link, be it an orthogonal or a free link, simply click on the desired point of the link whilst holding down the "*Ctrl*" key, and drag the point to the new desired position. If the link is free, acute angles will be produced. Conversely, if the link is orthogonal and this operation is carried out, the link will be shown in its new form, but only using right angles.

Moving a link end

To move a link end, simply click on the left mouse button over the link end which you wish to move, and drag it to the new link end point. When you release the left mouse button, the link end is repositioned. This procedure can be carried out for either the starting point of the link or its end point.

Redefining a free link in orthogonal mode

The context menu on a link which is obtained by clicking on the right mouse button allows you to modify a link's drawn line. The starting point is automatically selected.

Select the link by clicking on the right mouse button and choose the "Redraw Link" option in the context menu. Continue with the operations represented in Figure 5-7.

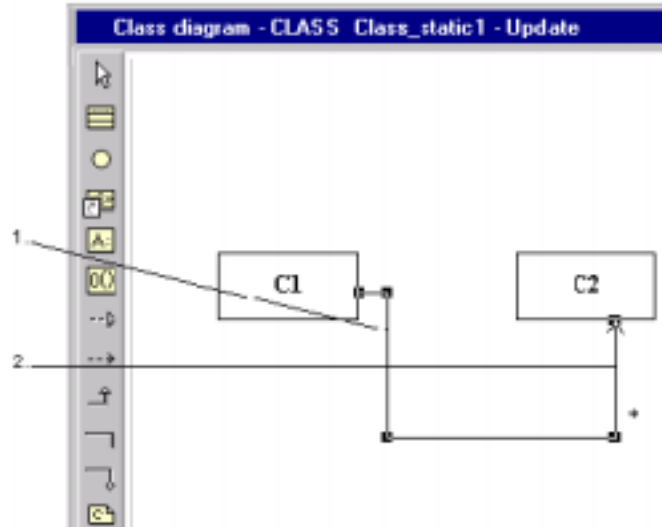


Figure 5-7. Redefining a free link in orthogonal mode

Steps:

- 1 - Holding down the "Shift" key, click on the left mouse button, to fix the different control points between the two elements.
- 2 - Click on the destination element using the left mouse button, to finish the redefinition link.

Redefining a free link in free mode

By default, the creation of a link between two elements is in free mode. However, this link can be modified by adding several control points between these two elements (Figure 5-8).

Select the link by clicking on the right mouse button, then select the "Redraw Link" option, and finally continue with the operations represented in Figure 5-8 below.

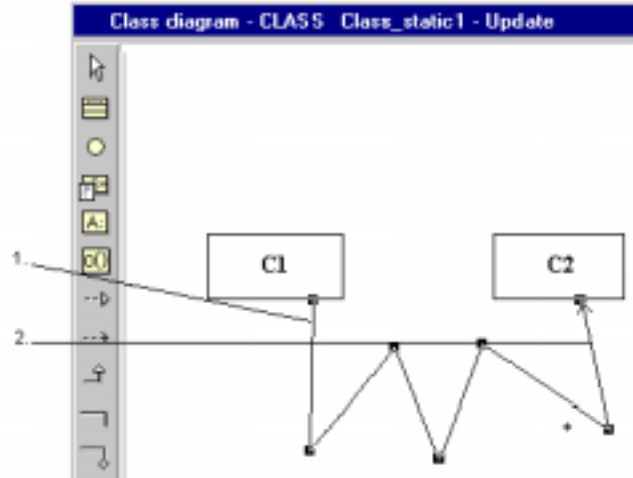


Figure 5-8. Redefining a free link in free mode between two classes

Steps:

- 1 - Using the left mouse button, click on the different locations to position the control points.
- 2 - Click on the destination element to finish the link.

Note: Control points can be moved. To do this, simply select the link by clicking on the right mouse button, and then select a point, holding down the left mouse button, and move it. When you release the left mouse button, the control point is frozen.

Chapter 5: Graphic Editors - General Principles

If you wish to undo control points:

- 1 - Select the link using the right mouse button by choosing the link redefinition option.
- 2 - Using the left mouse button, click directly over the destination class.

The control points no longer exist.

It is also possible to modify a straight link, by holding down the "*Ctrl*" button and then clicking on the link, and dragging it into an acute angle as desired. Control points are thus added.

Handling graphic elements

Mouse - Illustration

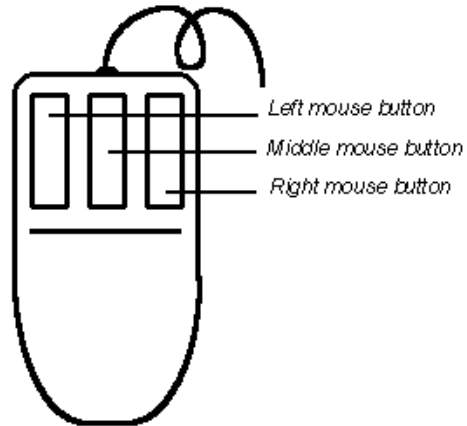


Figure 5-9. A mouse with three buttons

The ... button	is used to ...
left	designate, modify, move and resize elements, directly select, and redefine the tracing of a link.
middle	activate the redefinition of a link.
right	activate the context menus on the selected element.
left double click	activate the entry window of the element.

Note: For mice with only two buttons, clicking on both mouse buttons simultaneously corresponds to clicking on the middle button.

Selecting an element

An element, designated by the pointer, is selected by clicking on the left mouse button.

Selecting several elements

To select several elements, there are two possibilities:

- 1 - Hold down the "*Shift*" button and select the different elements.
- 2 - Select elements by clicking in the graphic zone, and then move the mouse, whilst holding down the left mouse button and drawing a rectangle round the elements. A dotted rectangle surrounds the selected elements. The selection is made when the left mouse button is released. Each selected element then appears with control points.

Moving an element

An element can be moved, by selecting the said element by holding down the left mouse button, and by moving the selected object to the desired position.



Result: The position of the object is visualized while it is being moved.

Dependencies (associations, generalizations, etc.) graphically "*follow*" the object moved.

Using the "Grab" function

The "*Grab*" function is used in Objectteering/UML to move the entire contents of a diagram within a graphic editor, without having to select each element.

To use the "*Grab*" function, simply carry out the following steps:

- 1 - Select one of the elements in the graphic editor.
 - 2 - Right-click, holding down the right mouse button until you see the small  hand icon appear.
- Note: Don't forget that if you right-click and do not hold down the mouse button, a context menu will appear.
- 3 - Still holding down the right mouse button, drag the  hand within the graphic editor. The entire contents of the diagram are then moved to wherever you move them.

Re-sizing an element

Control points are used to re-size an object (Figure 5-10).

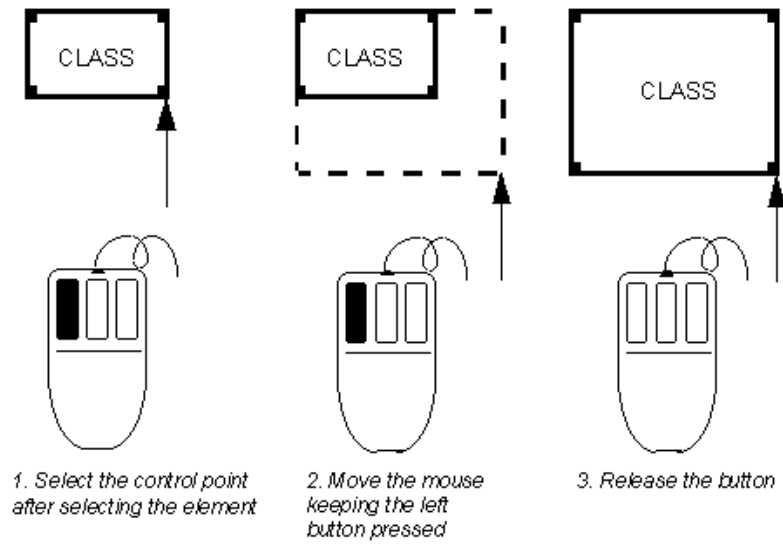


Figure 5-10. The re-sizing of a class

Modifying graphic elements

Procedure

If you wish to modify only the name of an element, you need simply highlight the name of the element and directly enter the new name.

To modify the contents of an element, carry out the following steps:

- 1 - Select the element.
- 2 - Double-click on the element, select the "*Modify*" function in the pop-up menu, by clicking on the right mouse button, or activate the "*Edit/Modify*" menu.
- 3 - Enter the modifications in the displayed dialog box.
- 4 - Confirm by clicking on "*OK*" or "*Apply*".

Note: If you click on "*Apply*", the element is updated, but the modification dialog box remains active. It can be used to modify another selected element.


Masking and showing elements

Presentation

It is possible to hide or reveal components or referenced elements in each diagram.

Elements can be shown using the drag and drop function from the explorer to a diagram (Figure 5-11). This function is used to show the desired elements (a single class, for example).

The principle behind showing elements

By default, all elements of a diagram (i.e. all model elements that are part of the diagram) are shown by clicking on the  "Show contents" icon. To show only certain selected elements, the drag and drop function from the explorer towards the diagram is preferable.

Showing using drag and drop from the explorer

We will now show the contents of a package using the drag and drop function (as shown in Figure 5-11).

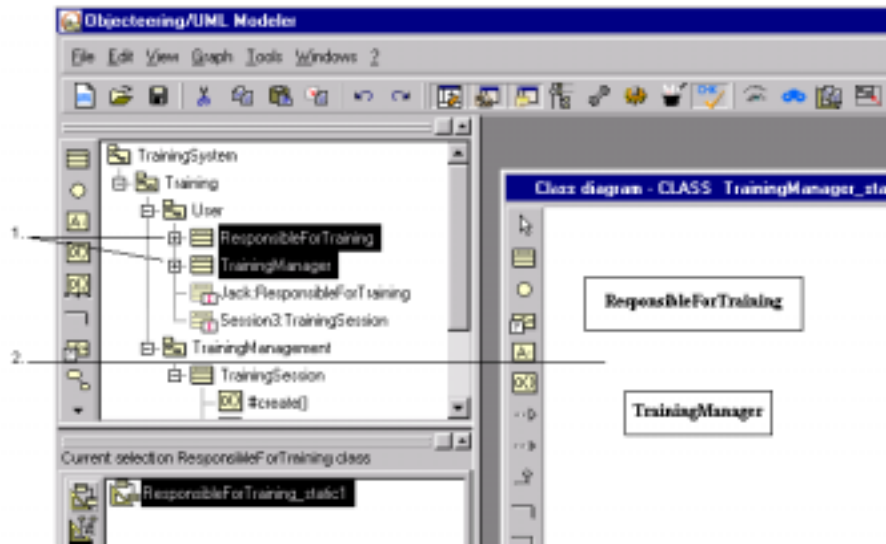


Figure 5-11. Showing classes using the drag and drop function

Steps:

- 1 - Select the elements to be shown and hold down the left mouse button.
- 2 - Holding down the left mouse button, drag the elements towards the diagram editor.


Showing links

To show links or parts of a model element, carry out the following steps:

- 1 - Click on the origin element (that is to say, the element from which the link stems), and click on the right mouse button. A context menu then appears.
- 2 - Select the "*Show links*" option from the context menu. The link then reappears.

Masking an element

This function is used to mask elements of a diagram. To mask an element:

- 1 - Select the element(s) to be masked.
- 2 - Click on the  "*Mask*" icon.

Note: Elements may also be masked by selecting the element in question, clicking on the right mouse button to display the context menu and selecting the "*Mask*" option.

Context menus for a diagram

Activating a context menu

A context menu can be activated by:

- ◆ selecting an object with the right mouse button, in order to display the corresponding menu, if it exists.
- ◆ selecting an option in the displayed menu.

Deactivating a context menu

A menu on a selected element may be deactivated by clicking outside the menu.

Context menu for a diagram

Clicking on the right mouse button in the background of a diagram displays the following context menu (Figure 5-12).

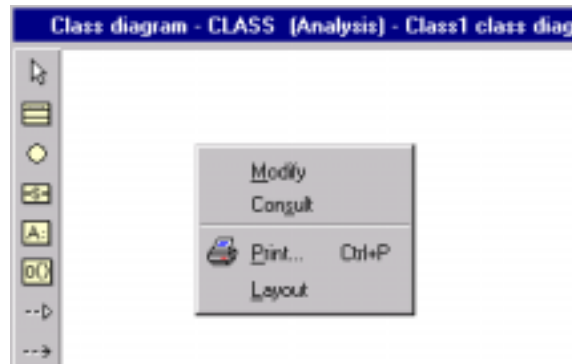


Figure 5-12. Menu of a diagram


The ... command	is used to
Modify	edit the modification dialog box for a diagram.
Consult	launch the modification dialog box for the diagram in read-only mode.
Print...	launch the diagram's print dialog box.
Layout	modify the diagram layout.

Chapter 6: Graphic Editors - Detailed View

Working with graphic elements

Modifying the size of elements

Diagram elements have a predefined size. The user may adjust this size, by using graphic element selection. It is then possible to adjust their size to their contents and to recover the pre-defined size. One or several elements can be adjusted at the same time.

To adjust the size of a diagram, you need simply select one or several elements, and then click on the  button (Figure 6-1).

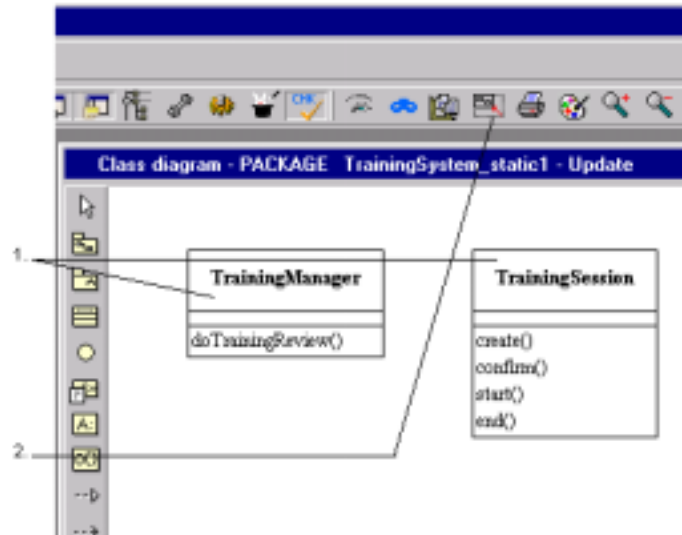




Figure 6-1. Adjusting a class in the diagram

Steps:

- 1 - Select the classes.
- 2 - Click on the  "Resize" icon.

Using the zoom function in a diagram

 The "Zoom forward" menu (shown in Figure 6-2) allows you to increase the display of a diagram's contents, in order to make it easier to read.



 The "Zoom back" menu allows you to reduce the display of a diagram's contents, in order to visualize a larger part of the presentation.



Figure 6-2. The "Zoom forward" menu on a diagram

Printing a diagram

The  "Print..." button edits the window used to print a diagram (as shown in Figure 6-3). This includes a field which concerns "Printer parameterization".

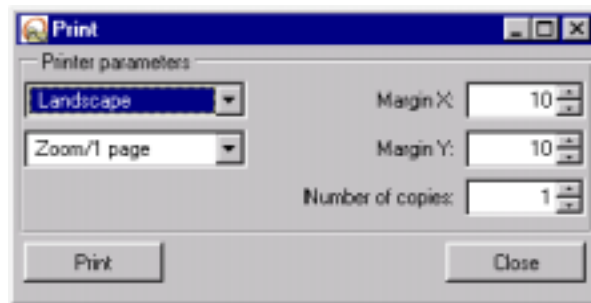


Figure 6-3. The "Print" dialog box

The ... field	allows you to ...
Printer parameters	choose the print format, margins and the number of copies

Note: By default, the OBJING_PRINTER environment variable is not set, and lp is used (the installation env.sh can be used to automatically set this variable to this value). However, this variable can be set to lp -c.

Saving a diagram in a file

To save a diagram in a file, follow the steps shown in Figure 6-5 below.



Figure 6-5. Saving a diagram in a file

Steps:

- 1 - Click on the "View" menu and select the "Save as" option.
- 2 - Choose the physical location where your file will be recorded.
- 3 - Enter the name of your diagram.
- 4 - Click on the "Save" button.

Switching from an element in a diagram to the same element in the explorer

It can be useful when working in a graphic editor to be able to switch back to a specific element in the explorer, without having to manually select the explorer and then the element concerned.

To do this, simply activate the context menu on the element in question in the graphic editor and select the "*Select in explorer*" option. This action automatically activates the explorer, displaying it in the foreground of the tool, and automatically selects the element which was selected in the graphic editor.

Note 1: This function is only available on box elements (in other words, it is not available on links).

Note 2: The highlight function only functions for the main explorer.

The "Modify" command on a diagram

The "Modify" command on a diagram

This command edits the modification dialog box, used to modify the selected diagram (Figure 6-5). This dialog box is identical for a class diagram, a deployment diagram, a deployment instance diagram, a collaboration diagram, an activity diagram, a state diagram and an object diagram.

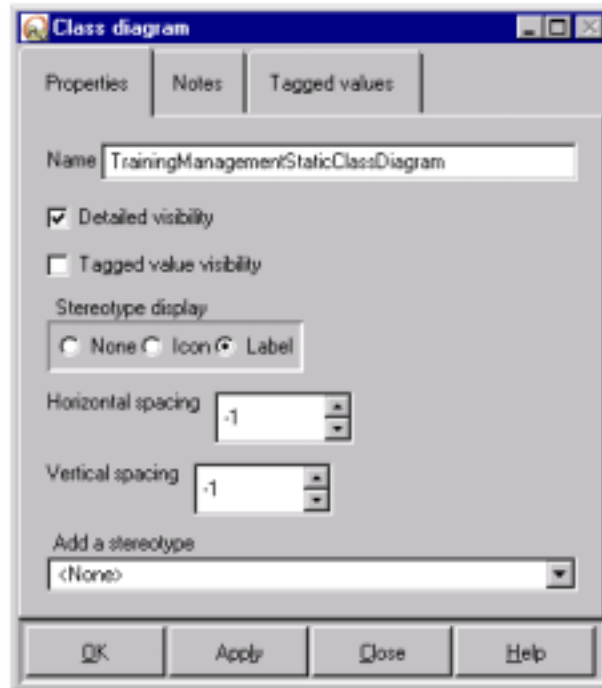


Figure 6-5. The "Class diagram" dialog box - Properties tab

The ... field or button	is used to ...
Name	change the diagram's name.
Detailed visibility	display the visibility symbols (+,-,#) and the complete path names (package:className) of the elements.
Tagged value visibility	make tagged values visible.
None	not display stereotypes.
Icon	display diagram elements by icon, if they are not expanded.
Label	display diagram elements by label.
Horizontal spacing	configure the automatic positioning of the diagram.
Vertical spacing	configure the automatic positioning of the diagram.
OK	confirm changes.
Close	close the modification dialog box.
Help	launch the on-line help concerning this dialog box.

Note: The addition of an element (for example, an operation or an attribute for a class) to an icon redisplay it in the format of an element. The masking of the contents of the element allows you to represent it using an icon.

The "*Notes*" tab of this dialog box allows you to add a note to a diagram. By clicking on the "*Add*" button, the "*Notes*" entry dialog box is edited. Further details on this dialog box are provided in the *Objectteering/Model Dialog Boxes* user guide.

The "*Tagged values*" tab allows you to add a tagged value to a diagram. By clicking on the "*Add*" button, the "*Tagged values*" entry dialog box is edited. Further details on this dialog box are provided in the *Objectteering/Model Dialog Boxes* user guide.

The "Modify" command on a sequence diagram

For sequence diagrams, this dialog box contains two extra elements with regard to the class diagram modification dialog box:

- ◆ the "*Show focus of control*" command, which deletes the *focus* representation on the diagram
- ◆ the "*Show return messages*" command, which deletes the representation of return messages between two sequence objects on the diagram

The "Modify" command on a use case diagram

This "*Modify*" command on a use case diagram contains a field additional to the class diagram's modification dialog box, namely the "*Representation of the system boundary*" field. Use cases are surrounded by a frame which is deleted when this command is deactivated.

Context menu on a diagram element

Context menu on a diagram element

The context menu on a diagram element changes according to the element in question.

The ... command	is used to ...	and can be activated on ...
Modify	edit the modification dialog box	all existing elements, as well as their contents.
Consult	edit the read-only dialog box	all existing elements, as well as their contents.
Browse	create an explorer from the element or edit an explorer from the element for reading only	a package, a class.
Check model	manually check model consistency from the selected element	a package, a class.
Wizards/Tools	runs the Process Wizard tool. The Wizards / Tools ... menu is different according to the element on which the command is run	all existing elements.
Resources	edit the graphic resources of the element	all existing elements.
Mask	mask the selected element	all existing elements.
Mask contents	mask an element's contents	all elements which can contain other elements.
Show contents	show the contents of an element	all elements which can contain other elements.
Show links	show an element's existing links	all elements which contain a link.

The ... command	is used to ...	and can be activated on ...
Select in explorer	activate the explorer on the model and select the element corresponding to the selected graphic element in the graphic editor.	all existing box elements, but not link elements.
Options	activate one of three possible modes: <i>detailed display</i> mode, <i>visible tags</i> mode and <i>stereotype display</i> mode	all existing elements.
Automatic	display the attributes and operations of a class or not, according to the user's choice	classes.

Note: You can open an element's modification dialog box by double-clicking on the element in question.

Graphic representation of a context menu on a class

Figure 6-6 below illustrates the visualization of a context menu on a class. Clicking on the right mouse button over the class in question activates this menu.

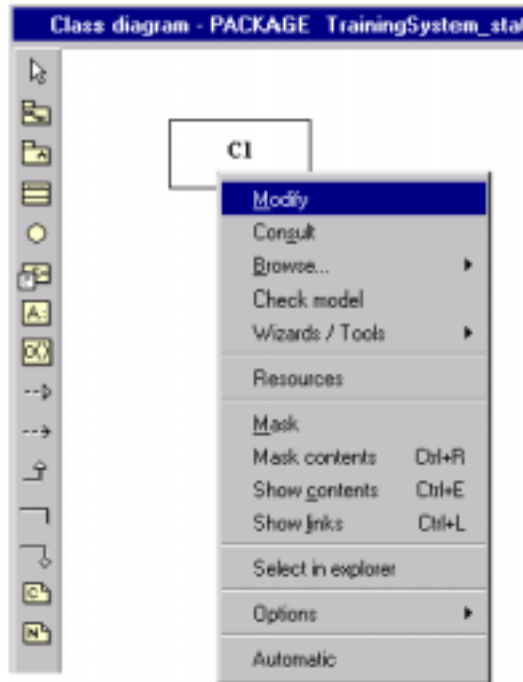


Figure 6-6. Context menu on a class in a diagram

The "Graph" menu

The "Resources" dialog box

The "Resources" option in the "Graph" menu opens the "Resources" dialog box (shown in Figure 6-8). This option is used to change various visual aspects of the diagram, notably:

- ◆ the element font
- ◆ the background and foreground colors
- ◆ the style and width of lines

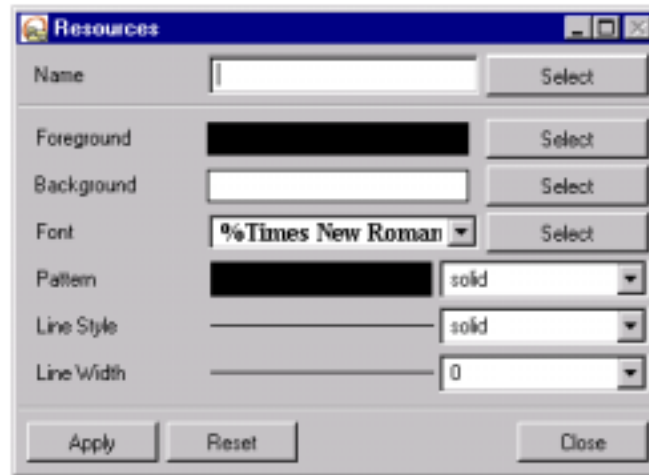


Figure 6-8. The "Resources" dialog box

The ... field	displays ...	which allows you to...
Name	the window containing the list of available palettes	select a palette from the proposed list.
Foreground	the " <i>Foreground</i> " window	choose a color for one or several elements.
Background	the " <i>Background</i> " window	choose a background color for the diagram.
Font	the " <i>Font Chooser</i> " window	choose the character style, the font size and so on.
Pattern	a context menu	choose an element's pattern.
Line Style	a context menu	choose the style of line or elements' surround desired.
Line width	a context menu	choose the width of line or elements' surround for the selected elements.

The "Grid" dialog box

The "Graph" menu can be used to open the "Grid" dialog box (as shown in Figure 6-9). This dialog box is used to display a background grid in your diagram, which will allow you to have marker points used to position elements in the aforementioned diagram.

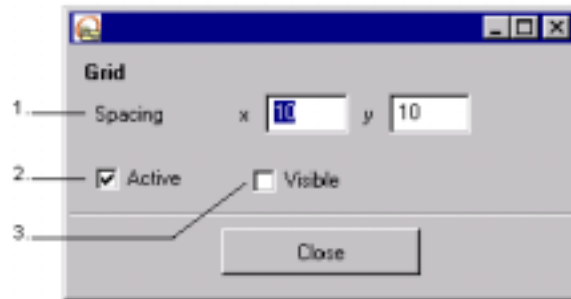


Figure 6-9. The "Grid" dialog box

Key:

- 1 - The "Spacing" field allows you to choose the horizontal and vertical spacing of the lines in your grid.
- 2 - If the "Active" box is checked, elements are positioned on the grid lines.
- 3 - If the "Visible" box is checked, the grid is displayed in your diagram.

The "Align" menu

The ... command	is used to ...
Left	align the selected objects to the object furthest left.
Center vertically	vertically align the selected objects.
Right	align the selected objects to the object furthest right.
Up	align the selected objects to the object furthest up.
Center horizontally	horizontally align the selected objects.
Down	align the selected objects to the object furthest down.

The "Help" menu

The "?" (help) menu

The command	...	is used to ...
Help		launch the on-line help.
General contents		open the " <i>General contents</i> " page of our on-line help. From this page, you can select the module you require assistance with.
Search engine		launch the Objectteering/UML search engine, used to search for information in the Objectteering/UML on-line documentation.
Volumes		select an on-line help volume. The volumes which figure in this list are those which are most frequently present in an Objectteering/UML configuration.
Objectteering Software on the Web		access either the " <i>Frequently Asked Questions</i> " page, the " <i>Objectteering Software home page</i> " or the " <i>UML Open Edition home page</i> " on the Web.
What's new		open a page, with information on the latest developments made to your version of Objectteering/UML.
About		provide general information on your version of Objectteering/UML.

Chapter 7: Specific graphic editors

Class diagram

Definition



The class diagram (an example of which is shown in Figure 7-1) allows you to present the internal structure of an element and its relationships with other elements (referenced by it). Class diagrams can present the greatest variety of elements.

The main elements presented in a class diagram are classes, packages, associations, generalizations and dependencies.

A class diagram is created in a package, a sub-system or a class. For information on the creation of a diagram, please refer to chapter 2 of this user guide.

Example of a class diagram - packages and dependencies

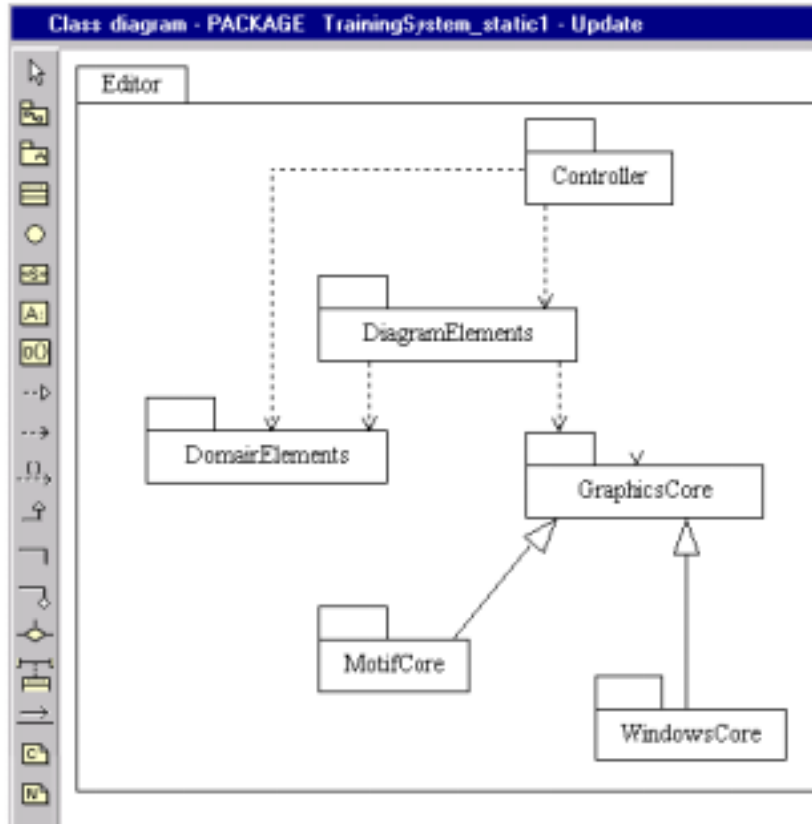




















Figure 7-1. Class diagram of a package which contains several generalized packages and associations

Elements which can be created or referenced

The ... icon	is used to create ...
	a package
	a sub-system
	a class
	an interface class
	a signal
	an attribute
	an operation
	an implementation link
	a dependency
	a throw or catch exception link
	a generalization link
	an association
	an aggregation
	an n-ary association
	a class association
	a dataflow
	a constraint
	a note

Class diagram - Behavior of graphic elements

Expanding elements in a class diagram

Certain graphic elements can be represented in two ways, summarized and developed. This is the case for classes, whose members can either be presented or not, and for packages, which can either present the elements they contain or not. The expansion of an element transforms a non-developed (summarized) representation into a developed representation.

Example: expanding a package

The example below (Figure 7-2) presents packages which contain a set of elements (classes, links, etc.). We are going to expand a package which contains several classes. The unmasking function will be accessed through the context menu applied to the element.

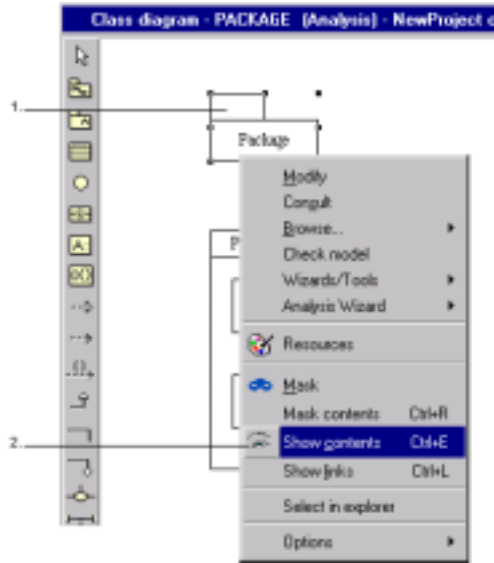


Figure 7-2. Expanding a package in a class diagram

Steps:

- 1 - Select a package containing several classes, by clicking on the right mouse button.
- 2 - Select the "Show contents" option in the element's context menu.

Conversely, an element can be downsized by using the "Mask contents" option in the context menu applied to the element in question.

Creating a sub-system in a class diagram

A sub-system is a kind of package, stereotyped <<sub-system>>, which represents an independent part of the system being modeled. Sub-systems represent an important aspect of the component-based approach.

To create a sub-system in a class diagram, following the steps shown below (Figure 7-3):

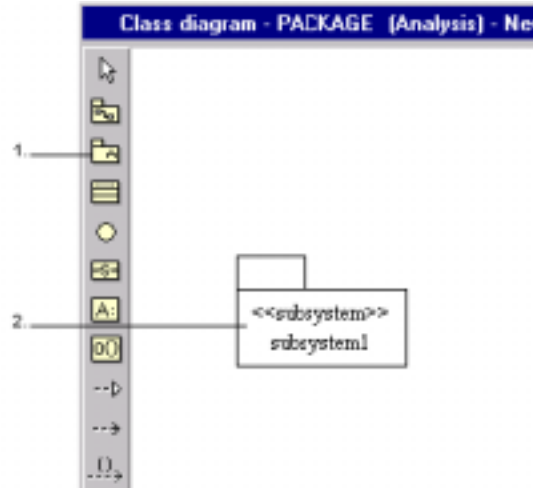



Figure 7-3. Creating a sub-system in a class diagram

Steps:

- 1 - Click on the  "Create a subsystem" icon.
- 2 - Click in the diagram at the point where you wish to position your sub-system.

Note: As you can see, a sub-system is simply a package stereotyped <<subsystem>>.

Creating a class association

A class association is a class which relates other classes, and is both a class and an association (see also the ClassAssociation metaclass). A class association is a component of an association. For further information on this class, please refer to the *Objectteering/Model Dialog Boxes* user guide.

The example below (Figure 7-4) presents the creation of a class association between two classes.

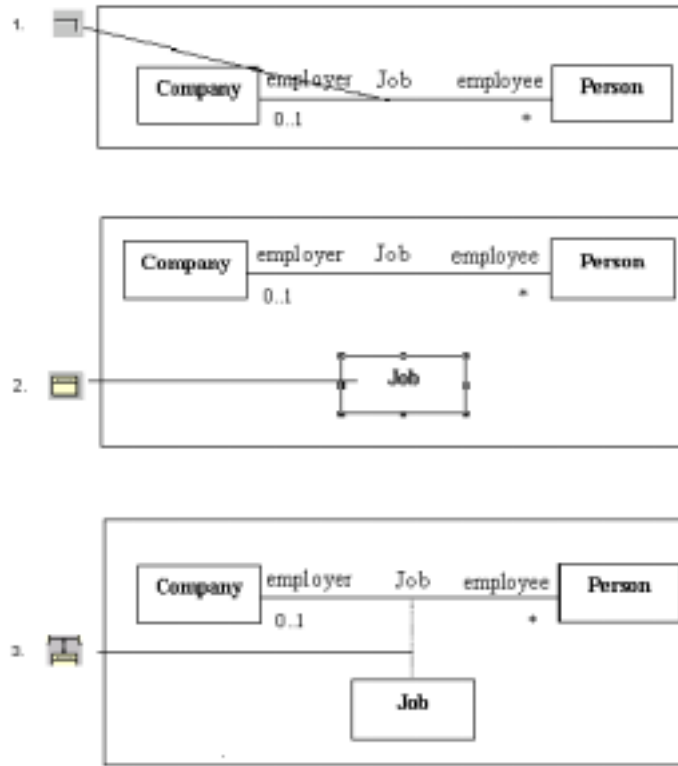






Figure 7-4. Creating a class association

Chapter 7: Specific graphic editors

Steps:

- 1 - Create an association between the "Company" class and the "Person" class, by clicking on the  "Create an association" button, and then clicking on the origin class and the destination class. The link then appears.
- 2 - Create the class which will serve as class association, by clicking on the  "Create a class" button.
- 3 - Create the class association, by clicking on the  "Create a class association" button, clicking on the class and then on the original association between the two original classes.

Throwing and catching exceptions

In Objectteering/UML, exceptions are represented by signals. The  ("Throw or catch an exception") icon is used, as its name suggests, to create use links stereotyped <<throw>> or <<catch>> between operations and signals.

Three uses of this feature are available:

- ◆ The creation of a use stereotyped <<catch>> (from an operation to an exception)
- ◆ The creation of a use stereotyped <<throw>> (from an exception to an operation).
- ◆ The creation of a dependency between an operation and a class or a signal.

Note: Please note that for dependencies between an operation and a class or a signal, the Objectteering/UML C++ and Java generators do not yet take these dependencies into account.

Figure 7-5 shows an example of a class diagram which contains a use link stereotyped `<<catch>>` (from the "serialize" operation belonging to the "Document" class towards the "FileSystemFull" signal) and another stereotyped `<<throw>>` (from the "write" operation belonging to the "fileServices" class towards the "FileSystemFull" signal).

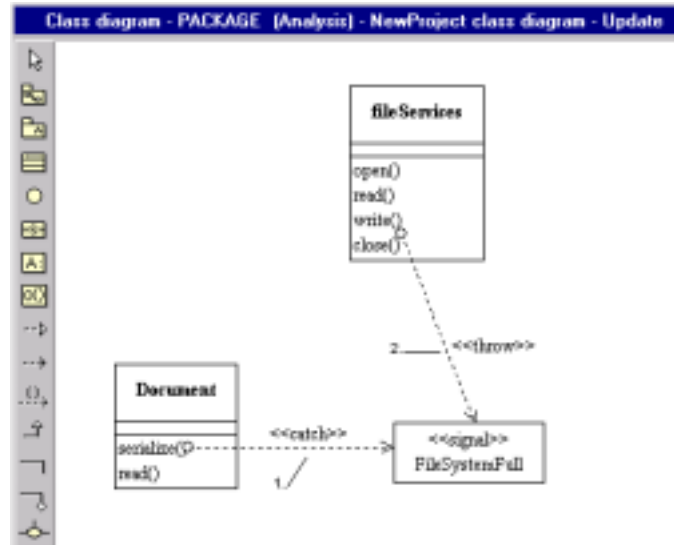





Figure 7-5. Example of a class diagram including both `<<catch>>` and `<<throw>>` use links

To catch an exception, simply click on the  ("Throw or catch an exception") icon, and then click on the operation in question and then on the signal (as shown in example 1 in Figure 7-5).

To throw an exception, click on the  ("Throw or catch an exception") icon, and then firstly click on the signal and then on the operation in question (as shown in example 2 in Figure 7-5). Please note that even though you first click on the signal and then on the operation, the link is created from the operation to the signal.

Creating a qualifier on an association

The creation of a qualifier on an association link can only be carried out in the explorer, by selecting the association concerned and clicking on the  ("Create a *qualifier*") icon. The qualifier created is, however, automatically presented in the class diagram after its creation in the explorer.

Generalization - creating rake bar links

Generalization links are the only links which can be presented in rake bar form (as shown in Figure 7-6). At the outset, a generalization is already created between two elements.

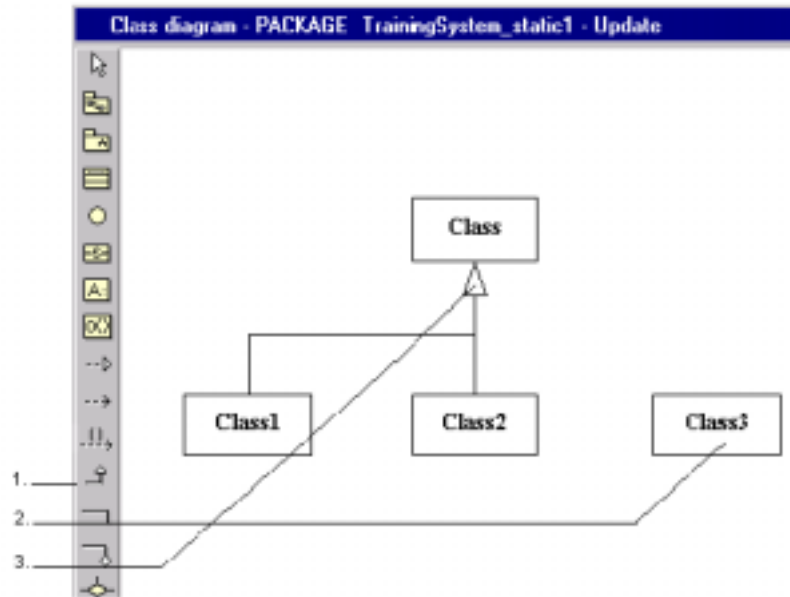



Figure 7-6. Tracing rake bar links between classes

Steps:

- 1 - Click on the  "Create a generalization" icon.
- 2 - Select the "Class3" origin class, by clicking on the left mouse button.
- 3 - Select an existing generalization link going to the same destination class. The generalization is then transformed into rake bar form.

Redefining rake bar links

To redefine a generalization rake bar link, proceed as follows (as shown in Figure 7-7):

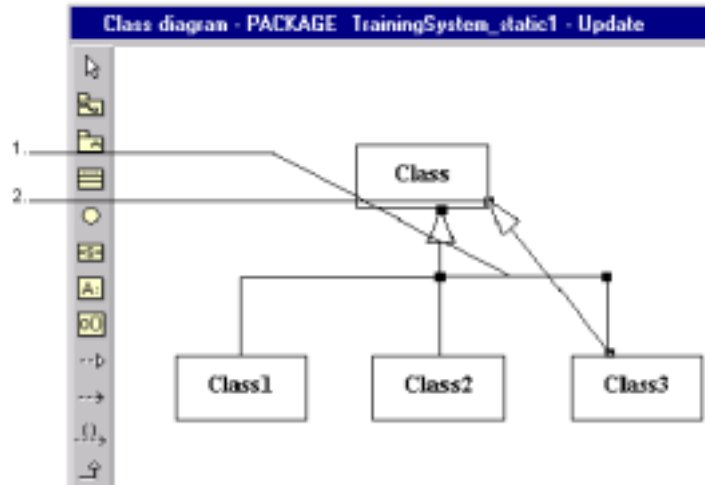


Figure 7-7. Modifying the drawing of generalization rake bar links by a free drawn line

Steps:

- 1 - Select the link using the left mouse button.
- 2 - Select the link end (the arrival point of the link), and hold down the left mouse button.
- 3 - Release the left mouse button over the link's destination class.

Modifying the drawing of a generalization link

To alter the drawing of the generalization rake bar vertical and/or horizontal bars, select and then move the bars in question, by holding down the left mouse button.

"Automatic" and "Non-automatic" modes

Classes can have attributes and operations, and the user can decide whether or not to show everything (all attributes and operations) by either activating or deactivating the "*Automatic*" mode option from the context menu available on the element in question. If everything is displayed, the user may find that visibility is impaired.

When the "*Automatic*" mode has been activated, everything which is created or modified in the explorer appears automatically within the class in the diagram. If the "*Automatic*" mode has been deactivated, this is not the case.

If the user selects the "*Show contents*" option from the context menu on the element in question, the "*Automatic*" is automatically activated. Conversely, if the "*Mask contents*" option is selected from the context menu, the "*Automatic*" mode is automatically deactivated.

Example

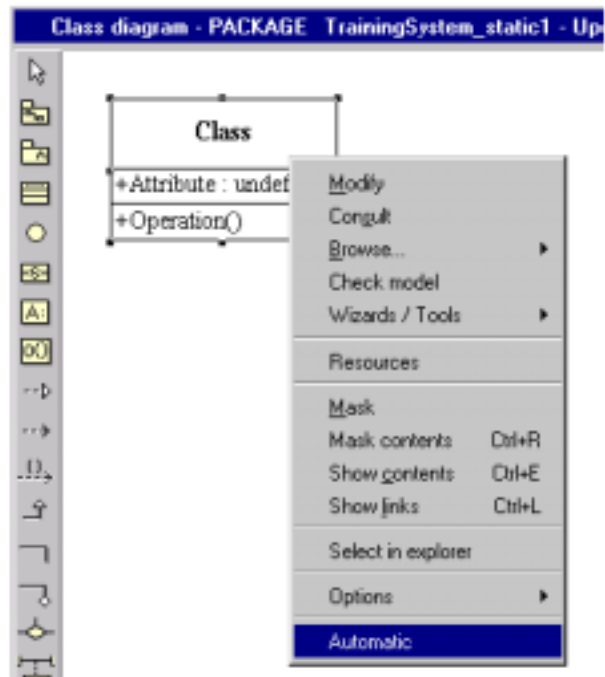


Figure 7-8. The "Automatic" option in the context menu on a class

Steps:


- 1 - Select the class.
- 2 - Click on the right mouse button and proceed with either the activation or deactivation of the "Automatic" mode option.

Managing class attributes and operations

Introduction

Objectteering/UML gives the user the possibility of creating and graphically editing class attributes and operations in graphic editors. These can be directly selected in diagrams.

Creating an operation

The  "Create an operation" icon is used to create an operation, by designating the owner class (as shown in Figure 7-9).

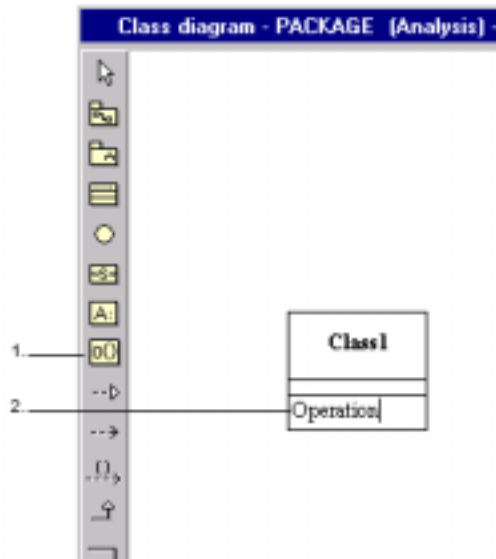



Figure 7-9. Creating an operation

Steps:

- 1 - Click on the  "Create an operation" button.
- 2 - Click on the class. Enter the name of this method over the text highlighted by default (*operation()*).

The operation dialog box

By double-clicking on the operation created, you may edit the "Operation" entry dialog box (Figure 7-10), which is used to carry out modifications or to add values to an operation.

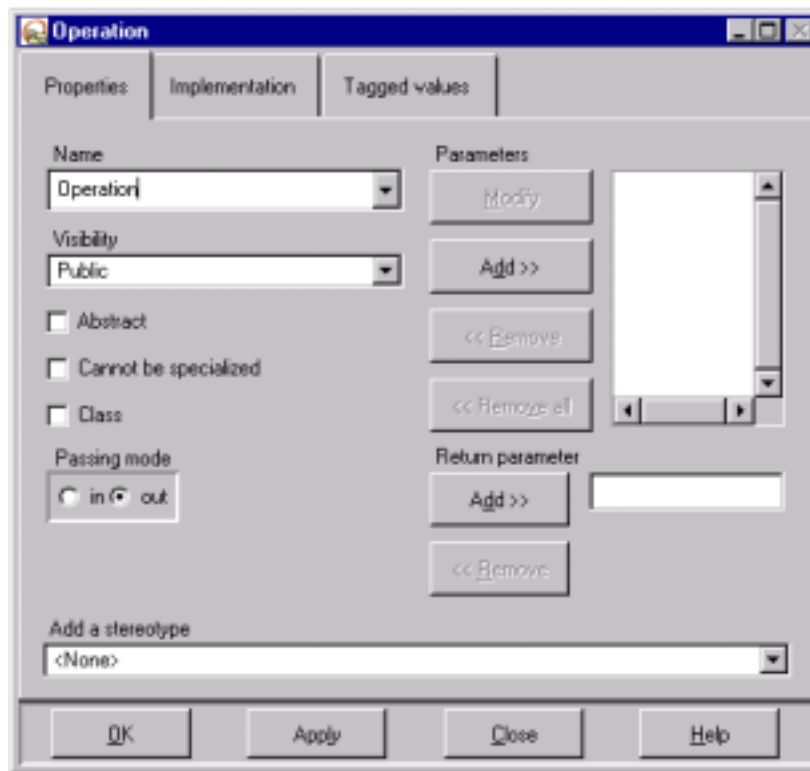



Figure 7-10. Operation modification dialog box

For further information on this dialog box, please refer to the *Objectteering/Model dialog boxes* user guide.

Creating an attribute in a graphic editor

To create an attribute in a graphic editor, click on the  "Create an attribute" icon, and then designate the owner class (as illustrated in Figure 7-11):

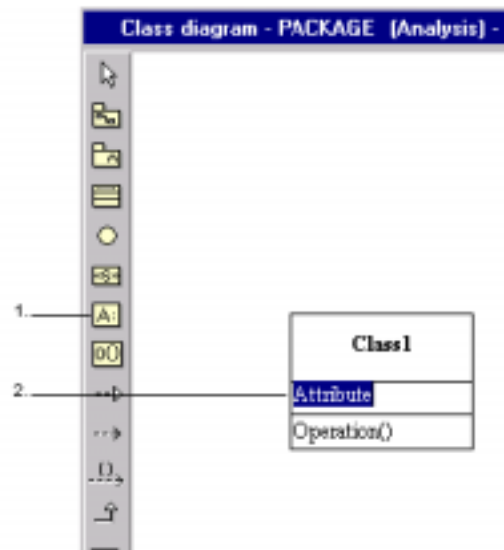



Figure 7-11. Creating an attribute

Steps:

- 1 - Click on the  "Create an attribute" button
- 2 - Click on the owner class. Enter the name of the attribute over the text highlighted by default.

Modifying an attribute

Double-clicking on an attribute allows you to immediately modify it, by direct designation (Figure 7-12).

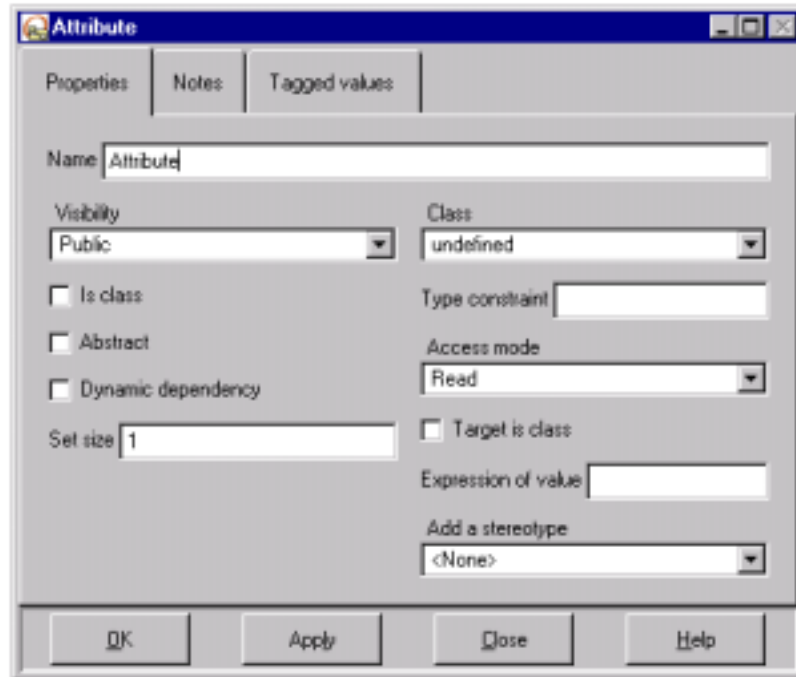


Figure 7-12. Modifying an attribute from the graphic editor

For further information on this dialog box, please refer to the *Objecteering/Model Dialog Boxes* user guide.

Modifying a class

If you wish to change the name of a class, you may either launch the modification dialog box for the class in question by clicking on the "Modify" button or edit the context menu, by clicking on the right mouse button, and select the "Modify" option (as shown in Figure 7-13). You can also directly enter the new class name by highlighting the displayed name.

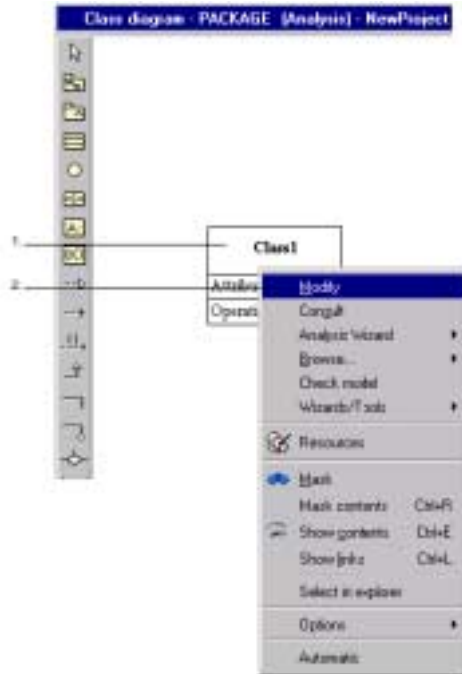



Figure 7-13. Editing the "Class" dialog box from a class diagram

Steps:

- 1 - Select the class by clicking on the right mouse button.
- 2 - Choose the "Modify" option in the context menu. The class dialog box is then edited.

Deleting an attribute or an operation

To delete an attribute or a operation:

- 1 - Select the attribute (or the operation) concerned.
- 2 - Activate the  "Edit/Delete" button, click on the "backspace" key or click on the "Delete" key.

Template class

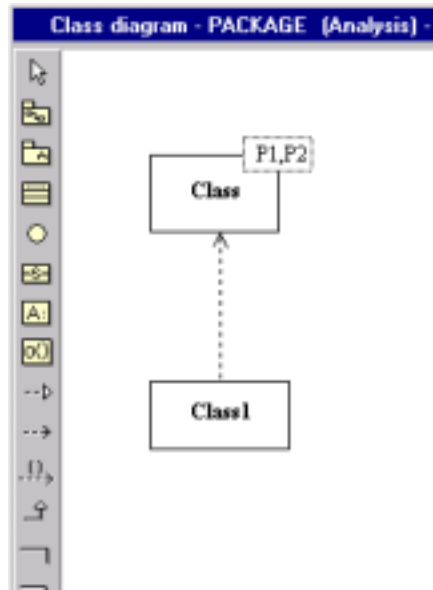



Figure 7-14. Class with template parameters (P1,P2)

Objecteering/UML supports template classes. Template parameters may be created only through the explorer, using the  "Create a template parameter" icon. A class may be bound to a template class by using the `{bind}` tagged value on the class itself.

Deployment diagram

Definition



The deployment diagram (an example of which is shown in Figure 7-15) is used to represent the physical architecture of the system. It presents the distribution of the software components on the set of execution units (Nodes).

Nodes and components are the main concepts in a deployment diagram.

Example

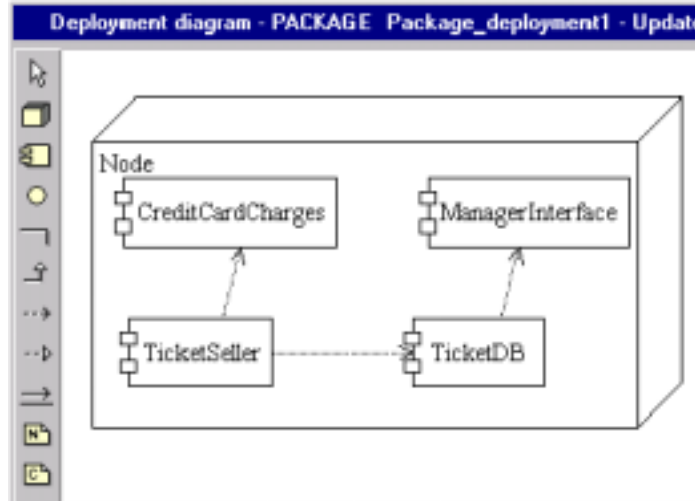











Figure 7-15. Diagram which represents a node containing several components

Elements which can be created or referenced

The ... icon	allows you to create ...
	a node
	a component
	an interface class
	an association
	a generalization
	a dependency
	an implementation link
	a note
	a constraint

Deployment diagram - Behavior of graphic elements

Creation mode


A deployment diagram can only be created at the level of a package or a sub-system (as shown in Figure 7-16).



Figure 7-16. Creating a deployment diagram

Steps:

1 - Select a package.

2 - Click on the  "Create a deployment diagram" icon in the "Diagrams" tab of the properties editor. The newly created diagram then opens automatically.

Creating a component inside a node

A node is a run-time physical object which represents a computational resource.

The example below (shown in Figure 7-17) presents the creation of a component in a node. The embedding function is used.

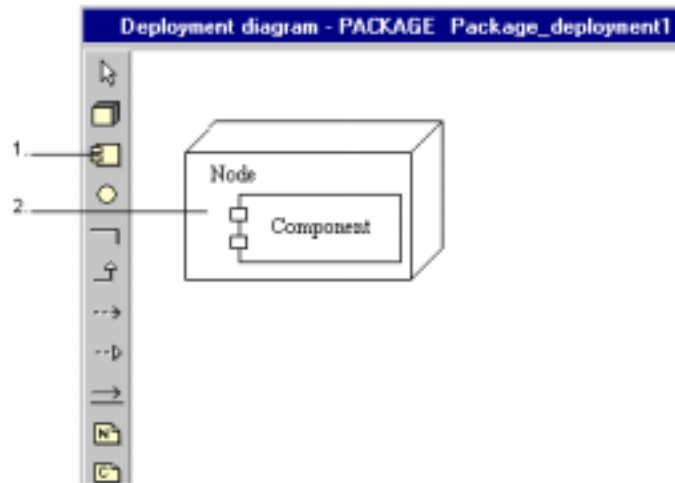



Figure 7-17. Creating a component inside a node

Steps:

- 1 - Click on the  "Create a component" icon.
- 2 - Click in the "Node" box.

The component is now contained within the node.

For most objects such as packages, classes, nodes, etc, embedded creation is carried out, and the consistency of the UML model is respected.

Referencing elements within a component

With Objectteering/UML, elements such as classes, instances and datatypes can be referenced by a component.


This operation is carried out by dragging and dropping a model element (classes, instances, datatypes) from the explorer into the component in the deployment diagram (as shown in Figure 7-18).



Figure 7-18. Referencing a class in a component through the drag and drop feature

Steps:

- 1 - Select the element to be referenced in the explorer.
- 2 - Drag it into the component in the deployment diagram.

It is also possible to create one or several references to the component using the  "Reference an element" icon, and to carry out a "Show contents" operation on the component in the diagram (as shown in Figure 7-19).

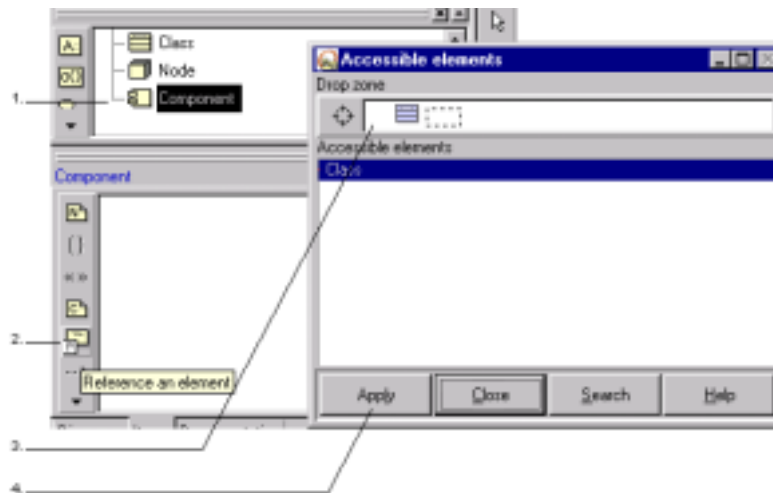



Figure 7-19. Referencing a class in a component through the "Reference an element" icon

Steps:

- 1 - Select the component which must reference the elements.
- 2 - Click on the  "Reference a component" icon.
- 3 - Drag and drop the element you wish to reference into the drop zone.
- 4 - Click on "Apply" to confirm the reference.

The element is now referenced. To make it appear in the diagram, you can either drag and drop this reference into the diagram, or run the "Show contents" operation on the component within the diagram.

Deployment instance diagram

Definition



The deployment instance diagram (an example of which is shown in Figure 7-20) presents a particular instance of deployment.

It represents instances of nodes and instances of components that can correspond to an example illustration component and deployment diagrams.

Restriction: "*Dependencies*" between components cannot be represented in these diagrams (for further details on strategies for connecting instances to their models, please refer to the "*The Model/Instance approach*" section of chapter 8 of this user guide).

Example of a deployment instance diagram

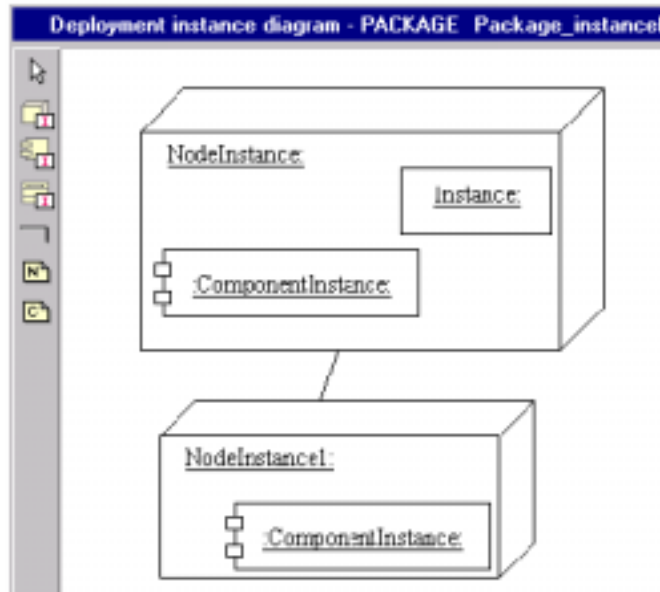








Figure 7-20. Instances of nodes and components

Elements which can be created or referenced

The ... icon	allows you to create ...
	a node instance
	a component instance
	a class instance
	a binary link between two objects
	a note
	a constraint

Deployment instance diagram - Behavior of graphic elements

Creation mode

The deployment instance diagram can only be created at the level of a package or a sub-system (as shown in Figure 7-21).

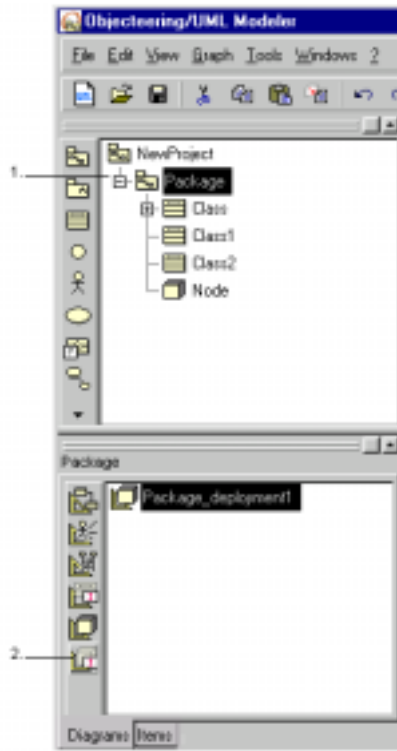



Figure 7-21. Creating a deployment instance diagram

Steps:

1 - Select a package.

2 - Click on the  "Create a deployment instance diagram" in the "Diagrams" tab of the properties editor. The newly created diagram then opens automatically.

Creating a node instance

A node instance is an instance of a node. The example below (shown in Figure 7-22) presents the creation of a component instance and two instances, as well as a link between the two instances, inside a node instance. The embedding creation function is used.

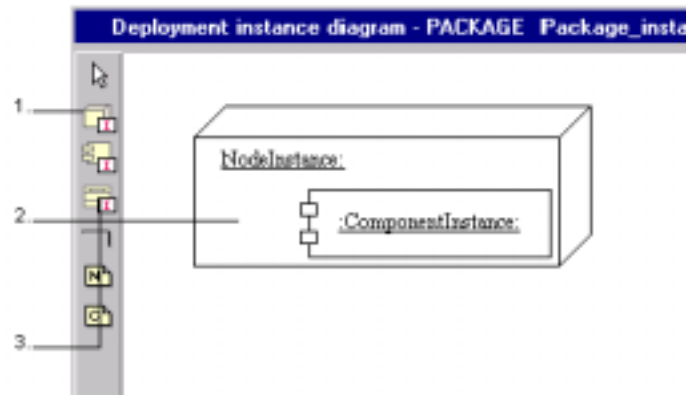




Figure 7-22. Creating a component instance and two instances within a node instance

Steps:

- 1 - Click on the  "Create a component instance" icon.
- 2 - Click inside the node instance, where you wish to position the component instance.
- 3 - Continue by clicking on the  "Create an instance" icon and creating one instance within the node instance and another within the component instance. Finish up by creating a link between the two instances. The result of these operations can be seen in Figure 7-23.

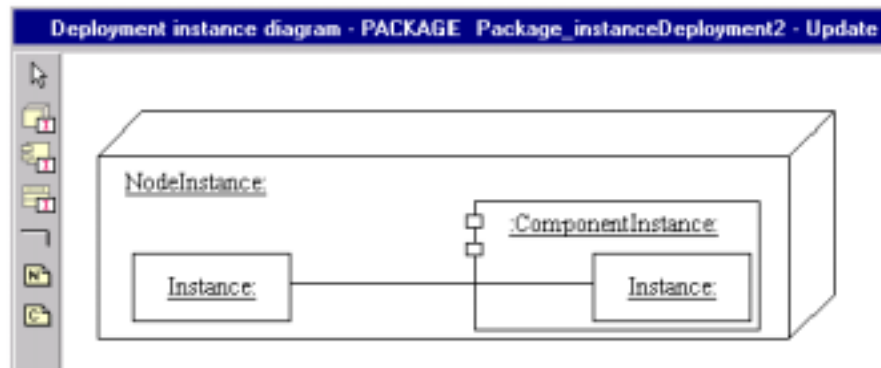
Result

Figure 7-23. Two instances and a component instance within a node instance

Note: Please note that the link between the two instances goes from the first instance, which is embedded in the node instance, to the second instance, which is embedded inside the component instance. Links can connect elements embedded inside other elements.

Object diagram

Definition



The object diagram (an example of which is shown in Figure 7-24) presents a set of class instances with their links and the messages exchanged. It can be created from a package, a subsystem or a class. Objects and links can be created without being linked to a class or an association. Messages are directly added to an existing link: if the link is oriented, the message is created with the same orientation; if not, it is created oriented towards the box nearest to the point where the user has clicked.

A synchronous message is represented near the link in the form of a complete arrow and its label. An asynchronous message is represented near the link in the form of a empty half arrow and its label.

It is not possible to create or represent a message for an n-ary link.

Objects can be connected to existing classes, or created independently for those classes. Connecting objects to classes will then allow you to connect links to associations and messages to operations (for further details on strategies for connecting instances to their models, please refer to the "The Model/Instance approach" section of chapter 8 of this user's manual).

Example of an object diagram

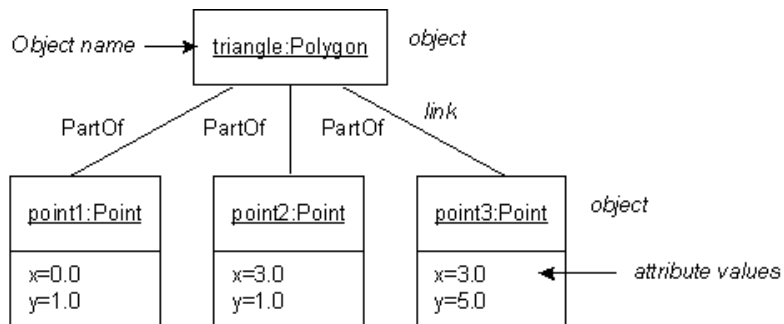









Figure 7-24. Representation of several instances

Elements which can be created or referenced

The ... icon	allows you to create ...
	an instance
	an instance attribute
	a binary link between two objects
	an n-ary link between objects
	a message from one object to another
	a constraint
	a note

Object diagram - Behavior of graphic elements

Creation mode


An object diagram can be created from a package, a sub-system or a class (as shown in Figure 7-25):



Figure 7-25. Creating an object diagram

Steps:

1 - Select a package.

2 - Click on the  "Create an object diagram" icon in the "Diagrams" tab of the properties editor. The newly created diagram then opens automatically.

Creating instances, associations and messages

Figure 7-26 illustrates how to create instances, associations and messages in an object diagram.

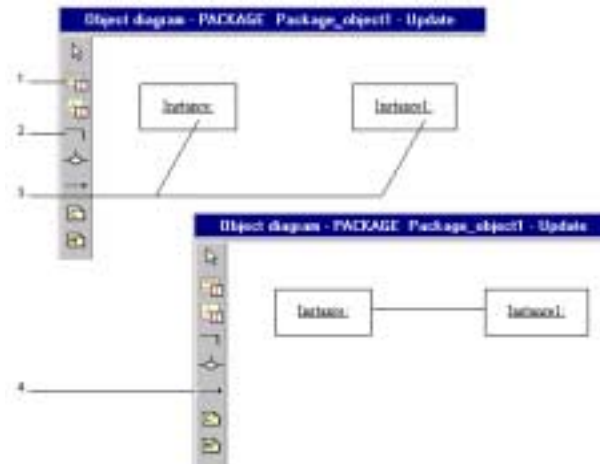





Figure 7-26. Creating instances, associations and messages

Steps:

- 1 - Click on the  "Create an instance" icon and then click in the background of the diagram where you wish to position your newly created instance. Create two instances in this way.
- 2 - Click on the  "Create an association" icon.
- 3 - Click first on the origin instance and then on the destination instance.
- 4 - Click on the  "Create a message" icon and then on the association between the instances. According to where you click, the message will be positioned closer to one end of the association than the other.

Sequence diagram

Definition



A sequence diagram (an example of which is shown in Figure 7-27) shows how different objects cooperate. The objects (vertical bars) can be defined a priori, and can be roles or class instances. Cooperation between objects is represented by the sending of messages between objects (horizontal arrows), and their sequence (order from top to bottom). A sequence diagram can be created on:

- ◆ a package
- ◆ a subsystem
- ◆ a class
- ◆ a use case
- ◆ a collaboration

If the sequence diagram is created in a collaboration, the objects are ClassifierRoles. If it is created in a package, a class or a use case, the objects are instances.

Objects can be connected to existing classes, or created independently from any class. Connecting objects to classes will then allow you to connect messages to operations (for further information, please refer to the "*Creating a graphic object*" section in chapter 5 of this user guide).

Example of a sequence diagram in a collaboration

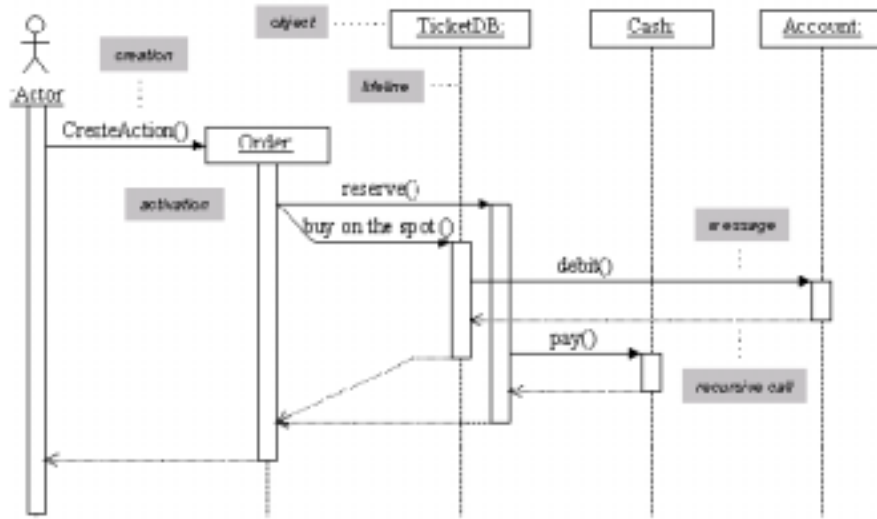










Figure 7-27. Sequence diagram

Elements which can be created or referenced in a sequence diagram in a collaboration

The ... icon	allows you to create ...
	a role instance
	a sequence message between two ClassifyRole objects. The return message is also created
	a creation message between two objects
	a destruction message from one object to another
	an asynchronous message
	a branch or fork message between two objects
	a constraint
	a note

Sequence diagram - Behavior of graphic elements

Creating a sequence diagram


A sequence diagram can be created in a collaboration, a sub-system, a class, a use case or a package (as shown in Figure 7-28).



Figure 7-28. Creating a sequence diagram

Steps:

1 - Select the root element in the explorer.

2 - Click on the  "Create a sequence diagram" in the "Diagrams" tab of the properties editor. The newly created diagram then opens automatically.

Messages

A creation or activation message is always created with a return message. If the creation or activation message is destroyed, the corresponding return or termination message will also be destroyed. In the same way, if a termination or return message is destroyed, the corresponding creation or activation message is also destroyed.

If an activation message is moved, the activation block in its entirety is also moved (including embedded activations).

A context menu item on an activation message is used to transform the message into a creation message. In the same way, a context menu item on a return message is used to transform the message into a termination message. Furthermore, a context menu item on a message allows you to declare the said message as being synchronous or asynchronous.

A sequence diagram's objects contain a lifeline on which the user creates messages. These messages have an activation time (checking focus), represented by a column between the start of the message and the return. Figure 7-29 below shows examples of messages between objects.

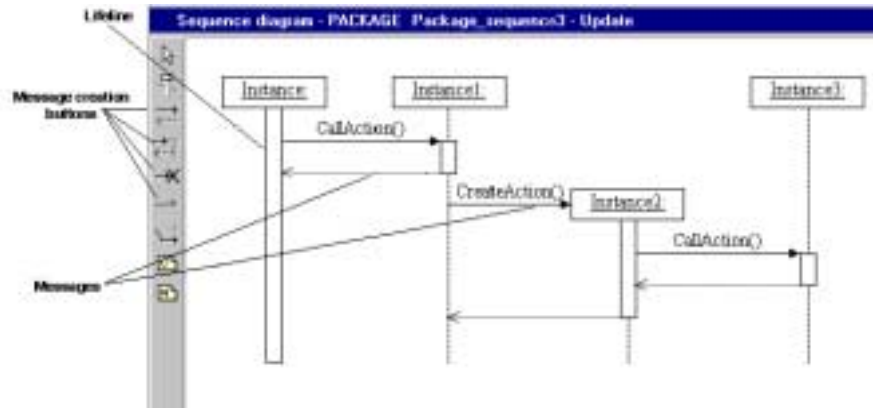


Figure 7-29. Examples of messages between instances

Note: A message created from one sequence object to another is graphically represented by an arrow. The object which receives the message is moved down.

Conditions and branches

Conditions may be defined on sequence, creation, destruction and asynchronous messages via the "Sequence message" dialog box (for further details on this dialog box, please refer to the "Sequence message dialog box" section in chapter 8 of the *Objecteering/Model Dialog Boxes* user guide), to indicate that an action is only carried out if certain defined conditions are fulfilled.

Conditions are often used in conjunction with forks and branches. In the context of sequence diagrams, a fork represents two simultaneous actions, whilst a branch is used to show two or more different possible outcomes, each with its own condition.

Figure 7-30 shows an example of a branch with conditions.

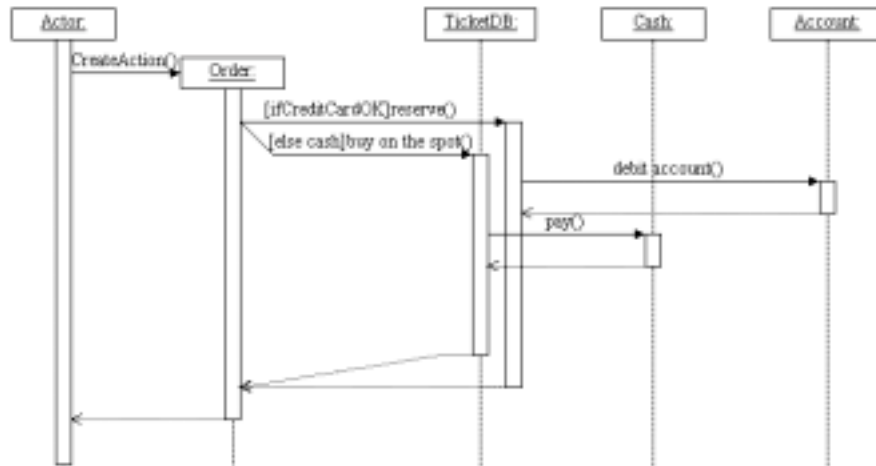


Figure 7-30. A sequence diagram containing a branch with conditions

Collaboration diagram

Definition



The collaboration diagram (an example of which is shown in Figure 7-31) is used to present exchanges of messages between roles. It is semantically very close to the object diagram. This diagram is created for a collaboration.

A collaboration defines a context in which roles can exist.

Roles can be connected to instances and classes (for further details on strategies for connecting instances to their models, please refer to the "The Model/Instance approach" section of chapter 8 of this user guide).

Example of a collaboration diagram

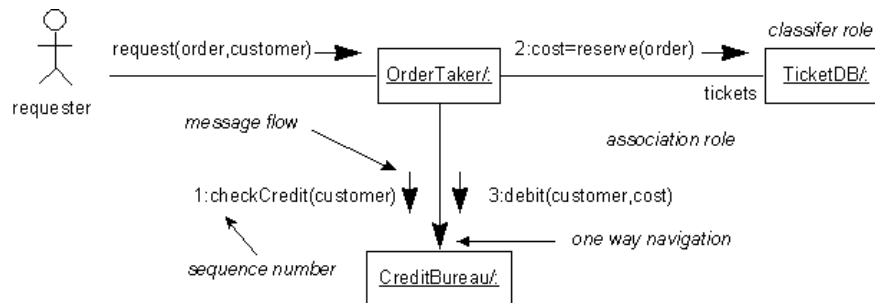









Figure 7-31. Example concerning role types

Elements which can be created or referenced

The ... icon	allows you to create ...
	a role instance. The modification dialog box allows you to attach a role to a class and/or an instance. The role is created with the label " <i>RoleName</i> :", " <i>RoleName/ObjectName</i> :" " <i>RoleName/ClassName</i> ", " <i>RoleName/ObjectName: ClassName</i> ", according to the information entered. The role name can be directly edited in the diagram.
	an attribute value.
	an association between roles. The modification dialog box allows you to attach a link to an association between classes.
	an n-ary association.
	a message. If the link is oriented, the message is created in the same direction; if not, it is created oriented towards the box nearest to the point at which the user has clicked.
	a constraint.
	a note.

Collaboration diagram - Behavior of graphic elements

Creating a collaboration diagram



A collaboration diagram is created from a collaboration, represented by the  icon (as shown in Figure 7-32).



Figure 7-32. Creating a collaboration diagram

Steps:

- 1 - Select the collaboration in the explorer.
- 2 - Click on the  "Create a collaboration diagram" icon in the "Diagrams" tab of the properties editor. The newly created diagram then opens automatically.

Orientation of messages

The orientation of the messages on a link is defined in the following way:

- ◆ If the link is not associated with a navigable association, the message is oriented towards the element nearest the positioning "click" of the message.
 - ◆ If the link is associated with a navigable association, then the message follows the direction of the navigability.
-

Use case diagram

Definition



The use case diagram (an example of which is shown in Figure 7-33) allows you to describe the most important services rendered by the system. Starting with actors, external participants who interact with the system, they represent the most important cases of system operating. A use case may then be sub-divided into sequence diagrams, which detail the different functions of one use case.

Example of a use case diagram

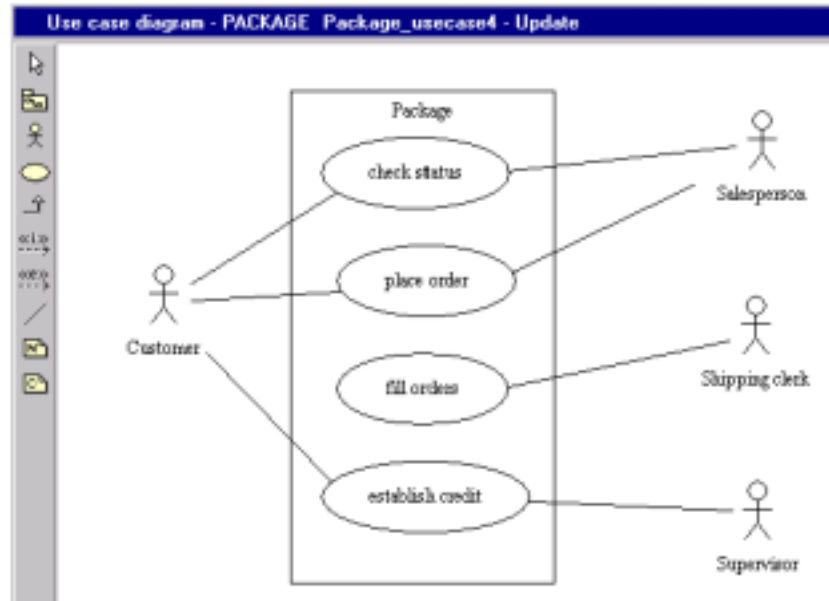











Figure 7-33. Use case diagram

Elements which can be created or referenced in a use case diagram

The ... icon	allows you to create ...
	a package
	an actor
	a use case.
	a generalization between actors or UseCases.
	an inclusion relation between UseCases
	an extension relation between UseCases
	a communication link
	a note
	a constraint

Use case diagram - Behavior of graphic elements


Creation mode

A use case diagram (shown in Figure 7-34) is created in a package or a sub-system.



Figure 7-34. Creating a use case diagram

Steps:

- 1 - In the explorer, select the package to be edited.
- 2 - Click on the  "Create a use case diagram" button in the "Diagrams" tab of the properties editor. The newly created diagram then opens automatically.

Use case boundaries

The "*boundaries*" of use cases can be displayed or not displayed, according to the user's choice. This option is available in the "*Edit*" menu for a diagram.

State diagram

Definition



State diagrams (an example of which is shown in Figure 7-35) are defined on a state machine, which can be created on a class, a sub-system, a package or an operation. A state diagram allows you to describe the manner in which objects react to events. It is used to describe a state machine at the level of a class.

State diagrams can also represent protocol.

Example of a state diagram

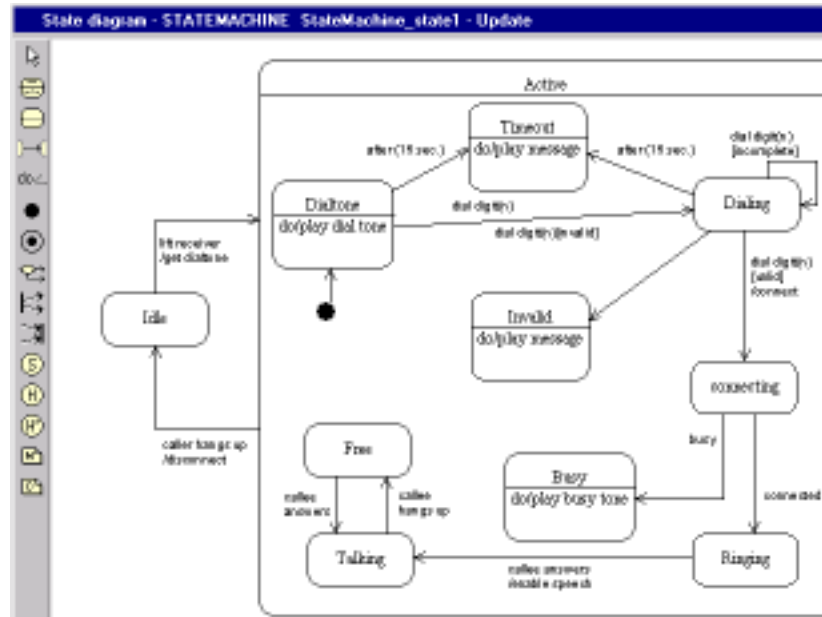
















Figure 7-35. Transition syntax

Elements which can be created or referenced in a state diagram

The ... icon	allows you to create ...
	a state. A state can be created in a state. In this case, the parent state is developed and the child state created inside.
	a concurrent state. This can be created in a state or in another concurrent state. Threads can be created within concurrent states (these threads can themselves be concurrent).
	a simple transition of an origin state to a destination state.
	an internal transition, to define the corresponding attributes. Chains are displayed in the internal transitions compartment.
	an initial pseudo-state.
	a final pseudo-state on a minimum of three existing states / pseudo-states. Transitions will be created between elements on which you have clicked and created a pseudo-state.
	a pseudo-state Branch, on a minimum of three existing states / pseudo-states. Transitions will be created between elements on which you have clicked and created the pseudo-state.
	a pseudo-state Fork, on a minimum of three existing states / pseudo-states. Transitions will be created between elements on which you have clicked and created the pseudo-state.
	a pseudo-state Join, on a minimum of three existing states / pseudo-states. Transitions will be created between elements on which you have clicked and created the pseudo-state.
	a synchronization state. The bound is either a positive integer or a star ("*") where unlimited. Synchronization states are drawn on the boundary between two regions when possible. It is the responsibility of the user to correctly place the synchronization state between two threads.
	a history.
	an in-depth history.
	a note.
	a constraint.

State diagram - Behavior of graphic elements

Creation mode

In order to be able to create a state diagram (shown in Figure 7-36), you must first create a state machine on the current element in the explorer.

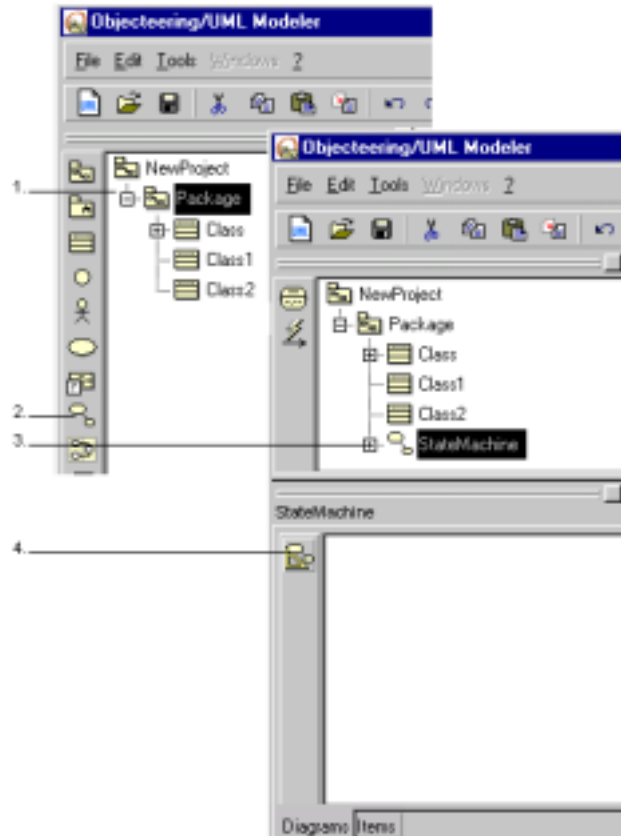




Figure 7-36. Creating a state diagram

Chapter 7: Specific graphic editors

Steps:

- 1 - Select a package or a class.
- 2 - Click on the  "Associate a state machine" icon.
- 3 - Enter the name of the state machine and confirm.
- 4 - Select this state machine in the explorer and click on the  "Create a state diagram" icon in the "Diagrams" tab of the properties editor. The newly created diagram opens automatically.

Creating sub-states

Embedded states can be created by creating new states in existing states (as shown in Figure 7-37). The parent state (State1 in Figure 7-36) can mask or unmask its contents. When the contents are masked, then transitions appear as "stubbied transitions".

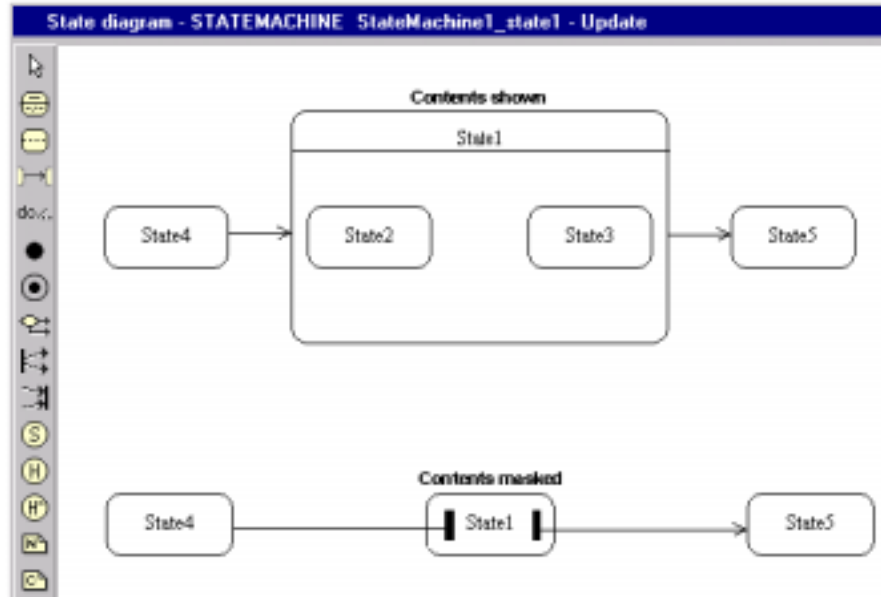





Figure 7-37. Embedded states

Creating pseudo states

 (branch),  (join),  (fork) are pseudo states that convert at least three states. These pseudo states can all be created at the same time, or pseudo states can be created individually, and then connected with other states (see Figure 7-38).

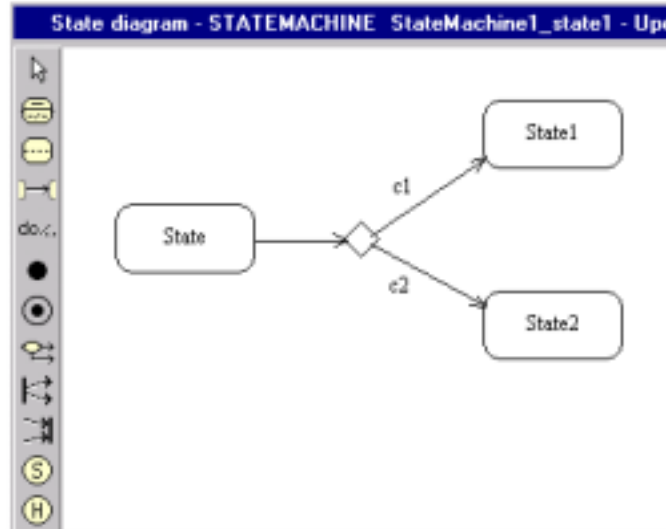



Figure 7-38. Branch pseudo state

Concurrent states and threads

To create a concurrent state in Objecteering/UML, the user may:

- ◆ either choose to designate the said state as being concurrent, by simply checking the "*Concurrent*" tickbox in the "*State*" dialog box (for further details on this dialog box, please refer to the "*State dialog box*" section in chapter 5 of the *Objecteering/Model Dialog Boxes* user guide)
- ◆ or create the concurrent state, by clicking on the  "*Create a concurrent state*" icon

Note: If the state edited contains a child state, the "*Concurrent*" checkbox is grayed out.

Chapter 7: Specific graphic editors

When a thread (or parallel state) is created within a concurrent state, the two states are graphically represented as shown in Part 1 of Figure 7-38. However, if subsequent threads are then created, the graphic representation differs, as shown in Parts 2 and 3 of Figure 7-39.

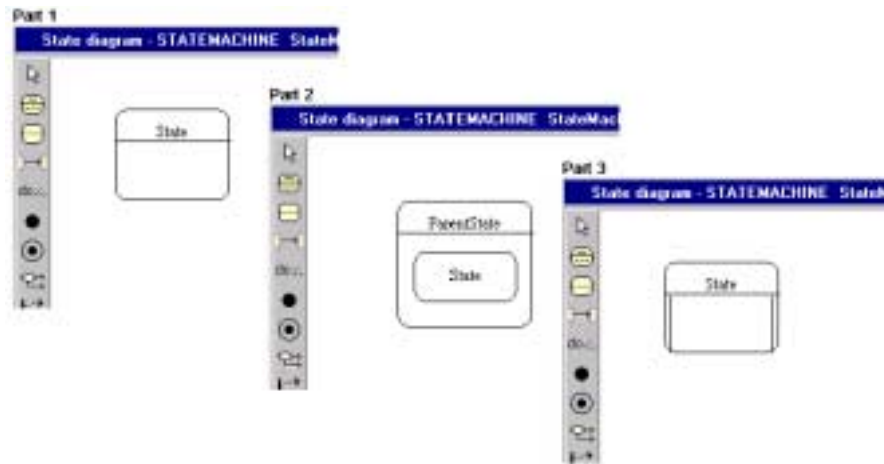


Figure 7-39. Concurrent states and threads

Key:

- 1 - Part 1 shows a concurrent state containing one thread.
- 2 - Part 2 shows a parent concurrent state containing a thread, which itself contains another thread. Graphic representation shows that the first thread is not concurrent, since the second thread is simply embedded therein.
- 3 - Part 3 shows the same parent concurrent state as in Part 2. Graphic representation shows that the first of these threads, created in the original concurrent state, is itself concurrent. This is shown by the twin bars, used to represent the second thread. For each subsequent thread created within a concurrent state, a set of bars appears to graphically represent the addition.

Where two threads are created in the same concurrent state, they are graphically represented as follows (Figure 7-40).

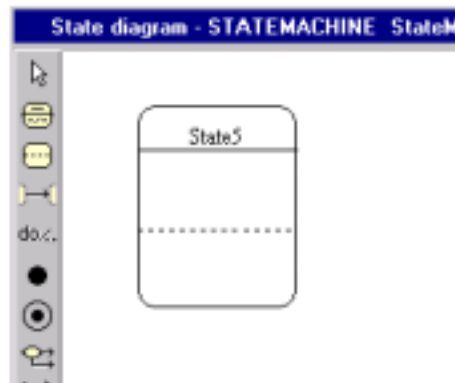


Figure 7-40. Two threads created in a concurrent state and represented by a dotted line

Note: If you wish to create several threads in the same concurrent state, you must click on the creation icon and then over the header of the concurrent state.

Creating substates in threads

To create substates in threads, you should simply click on the icon used to create the desired state and then click inside the thread (as shown in Figure 7-41).

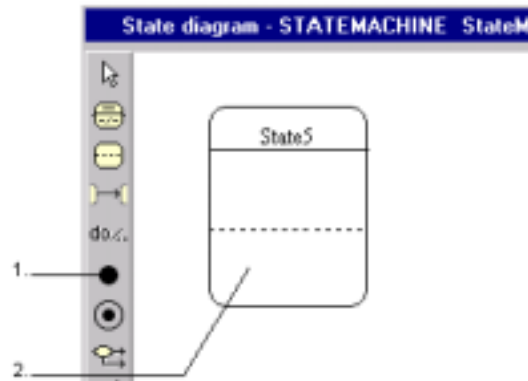




Figure 7-41. Creating substates in threads

Steps:

- 1 - Click on the  "Create an initial state" icon.
- 2 - Click inside the thread in which you wish to create the substate.

Creating and referring to events

There are two ways of creating events in a state diagram:

- ◆ creation in the explorer in the state machine, using the  "Create an event" icon
- ◆ creation via the state dialog box, where events are entered as deferred events. Once these events exist, they can be referred to from any transition dialog box (see Figure 7-42).

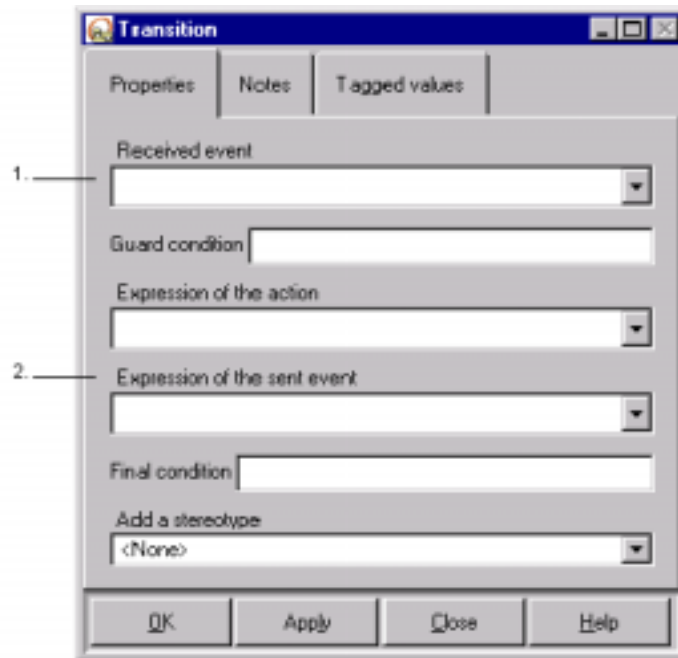


Figure 7-42. Transition dialog box

Key:

- 1 - "*Received event*": This is the event received which triggers the transition. The received event can be text entered in the field, or a reference to events defined in the current state machine.
- 2 - "*Sent event*": This is an event sent by the transition once it has been triggered. An emitted event can be text entered in the field, or a reference to existing events in the current state machine (combobox). Signals can also be referenced (shorthand for Signal sending event).

Received or sent events in the transition dialog boxes can refer to existing events. As a practical shorthand, sent events may also directly refer to signals. Actions which frequently activate an operation of the class can directly refer to the class' operations.

Activity diagram

Definition

















Activity diagrams (an example of which is shown in Figure 7-43) are created on an activity graph, which can be created for a class, a sub-system, a component, a node, a signal, a use case, an actor, an operation and a package. An activity diagram is used to show a procedure or workflow.

Example of an activity diagram



Figure 7-43. Activity diagram

Elements which can be created or referenced in an activity diagram

The ... icon	allows you to create ...
	an action state.
	a sub-activity state.
	a simple transition of an origin state to a destination state.
	an initial pseudo-state.
	a final pseudo-state.
	a pseudo-state Branch, on a minimum of three existing states/pseudo-states. Transitions will be created between elements on which you have clicked and created the pseudo-state.
	a pseudo-state Fork, on a minimum of three existing states/pseudo-states. Transitions will be created between elements on which you have clicked and created the pseudo-state.
	a pseudo-state Join, on a minimum of three existing states/pseudo-states. Transitions will be created between elements on which you have clicked and created the pseudo-state.
	a sender signal. This state has no actions, but the out transition sends a signal.
	a receiver signal. This state has no actions, but the in transition receives a signal.
	a partition. Partitions are used to organize the activity graph into different entities. All activities can be assigned.
	an object flow state. This is used to define a certain state.
	a note.
	a constraint.

Activity diagram - Behavior of graphic elements

Creation mode

In order to be able to create an activity diagram (shown in Figure 7-44), you must first create an activity graph on the current element in the explorer.

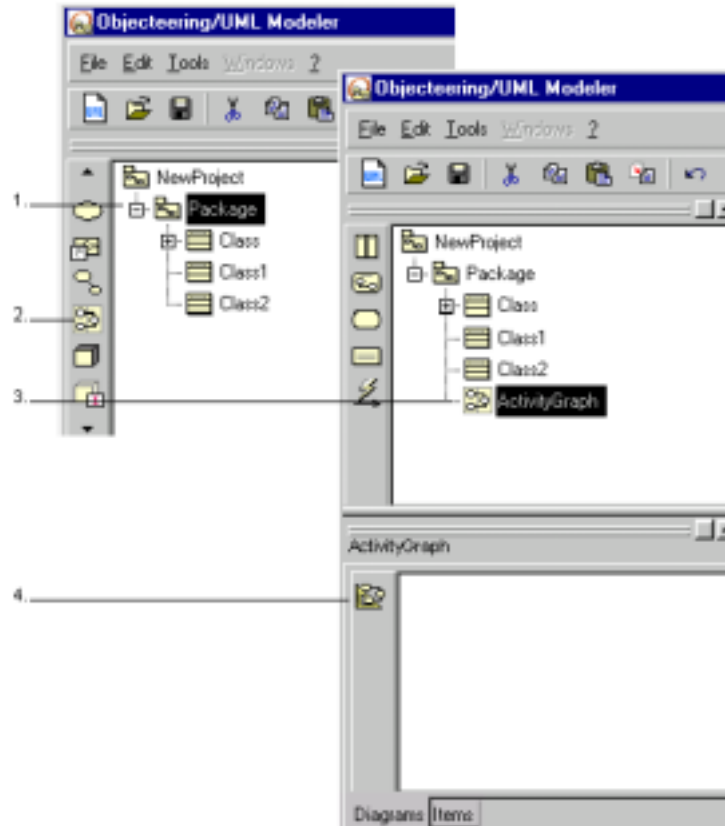




Figure 7-44. Creating an activity diagram

Steps:

- 1 - Select the element on which you wish to create the activity graph (in this example, a package).
- 2 - Click on the  "Associate an activity graph" icon in the "Diagrams" tab of the properties editor.
- 3 - Select the activity graph.
- 4 - Click on the  "Create an activity diagram" icon.

Behavior of graphic elements

The behavior of graphic elements in activity diagrams is similar to that of elements in a state diagram, since the activity diagram and the state diagram are closely related.

Action states, sub activity states and object flow states

An action state, which cannot be broken down, represents the execution of an atomic action, typically the invocation of an operation. An action state belongs to an activity graph or sub activity state, and always has an associated internal transition, used to describe the triggered processing.


Sub activity states are states which can be broken down in another activity graph similar to action states, except that they may have children. A sub activity state belongs to an activity graph or to another sub activity state, and has no associated internal transitions.

Object flow states define an object flow between actions in an activity diagram and represent a state at a certain time in its lifecycle.

Partitions

Partitions, also known as swimlanes, are used to group states (action states, pseudo states, sub activity states and object flow states) within an activity diagram, and often correspond to organizational units in a business model. They belong to an activity diagram.

Action states, object flow states and sub activity states can be divided into partitions or not, as the user wishes.

To create a partition in an activity diagram, simply click on the  "Create a *partition*" icon. Partitions are graphically shown stuck together. They can be moved by selecting the partition in question and dragging it to its new position. Regardless of the positioning of partitions, they always remain stuck together, and cannot be superimposed over one another.

Note 1: States in an activity diagram can be assigned to several partitions, but appear only once in the diagram. In the explorer, however, they appear in each partition to which they have been assigned. To assign states to a partition, simply drag the states in question, one by one, into the partition.

Note 2: Action states, pseudo states (except joins and forks) and sub activity states cannot straddle two partitions. Object flow states, joins and forks, however, can straddle two partitions, since they are not assigned to a specific partition, and represent different states in the evolution of an object.

Chapter 8: Methodological Hints

The Model/Instance approach

Presentation

The Objectteering/UML CASE tool is very much oriented towards defining a general model and then detailing this model using examples which refer to it. There are two possible approaches, particularly concerning the management of a model with regard to its instances:

- ◆ *Approach 1*: we start with a model, and then construct the instances which refer to this model.
- ◆ *Approach 2*: we define the instances as the first examples, before having created the model. The model will then try to represent these instances, and it will be necessary, if you should so desire, to associate each of the instances to a model element.

The aim of the first approach is to show model examples, in order to validate or illustrate the model itself. The second approach, on the other hand, is used to provide examples of the domain concerned, in order to then create the model. Models which deal with instances in Objectteering/UML (*object*, *sequence*, *collaboration* and *deployment instance* diagrams) can handle both these approaches.

Example of the first approach

Carry out the following steps:

- 1 - In the explorer, create the following three classes, with the operations indicated: the "Product" class with the "create" and "deliver" operations, the "Worker" class with the "build" operation, and finally the "Manager" class.
- 2 - Continue by creating the sequence diagram shown in Figure 8-1. First create three instances, then display them and associate them to their classes using the model dialog box shown in Figure 8-2.
- 3 - Finally, create the messages, then edit them, and associate them to their operation using the model dialog box shown in Figure 8-3.

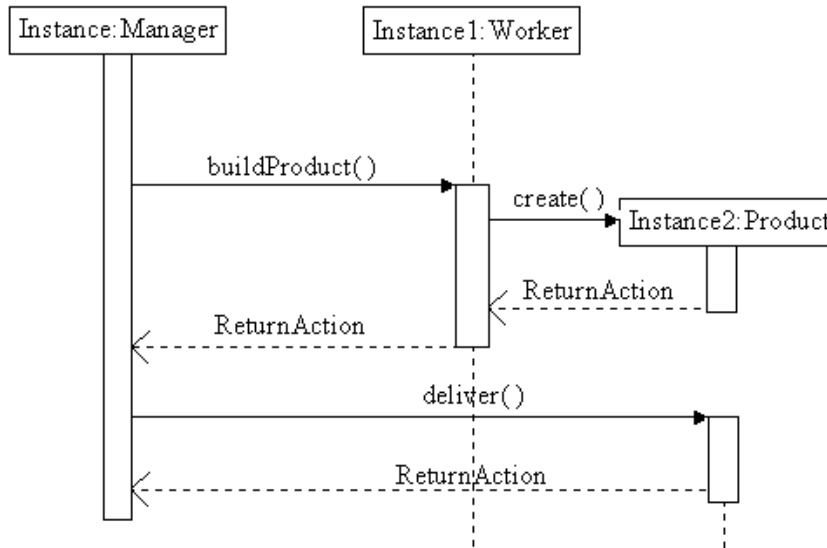


Figure 8-1. Example of a sequence diagram

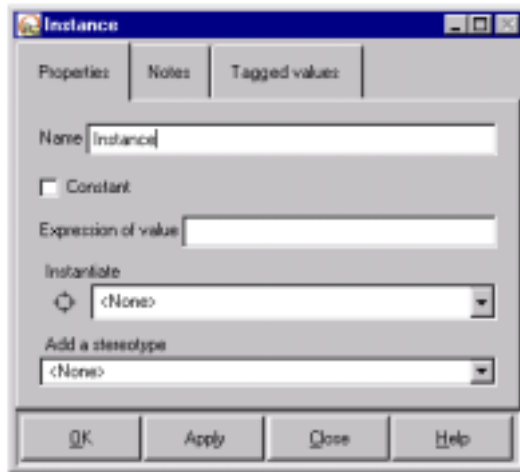


Figure 8-2. The "Instance" dialog box, used for associations to classes

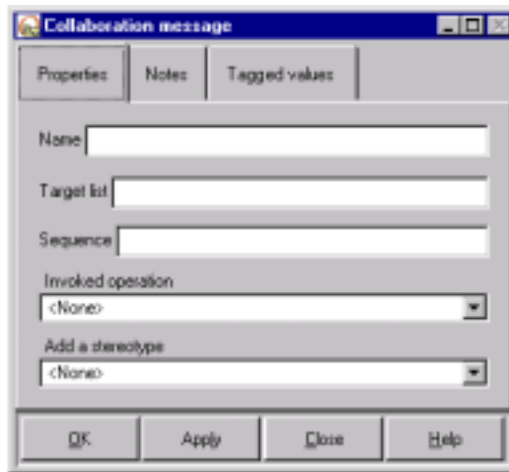


Figure 8-3. The "Message" dialog box, used for associations to operations

Example of the second approach

We shall firstly create the sequence diagram shown in Figure 8-1. Neither the classes nor the model already exist. Object and message names are entered directly, without being associated to existing operations and classes.

The next step is to enter the classes and operations mentioned in the previous paragraph. They have been displayed as shown in the example.

Finally, the sequence diagram instances and the message are displayed and connected to the class model which we have just created.

If we had used an object diagram, the procedure would have been the same, but also with links connected to associations and attribute instances linked to attributes.

Flow Diagrams and DataFlows

Overview

Flow diagrams are an extension added to UML. This type of model provides a useful representation of the "*preliminary analysis*" and "*preliminary requirements*" phases. It is also used to represent functional or technical architecture (*preliminary design*).



It is often difficult to develop a global vision of the functions of large systems. UML can provide:

- ◆ a global vision of the system, through package diagrams
- ◆ an overview of the services provided by the system through actors and use cases
- ◆ a detailed view of the dynamics, through state, collaboration and sequence diagrams

Flow diagrams broadly present the circulation of information between different system entities, and are effectively the offspring of a procedure called "*systematic analysis*", used in the 1980s. They have also been used since 1990 in an object method called "*ClassRelation*".

UML extensions

Two simple extensions have been applied to UML:

- ◆  The "*DataFlow*", which represents the flow of information between two system entities (please refer to the "*Data flow dialog box*" section in chapter 3 of the *Objectteering/Model Dialog Boxes* user guide, and to the "*DataFlows and Signals*" section of chapter 5 of the *Objectteering/Metamodel User Guide*).
- ◆  The "*Signal*", which extends the definition of a piece of information which can be transmitted between two parts of the system.

A DataFlow can refer to a signal. A signal can represent a model element which will subsequently implement a transmissible piece of information, in other words :

- ◆ a class
- ◆ an operation parameter
- ◆ an operation (please refer to the "*Signal dialog box*" section in chapter 3 of the *Objectteering/Model Dialog Boxes* user guide, and to the "*DataFlows and Signals*" section of chapter 5 of the *Objectteering/Metamodel User Guide*).

Example of notation

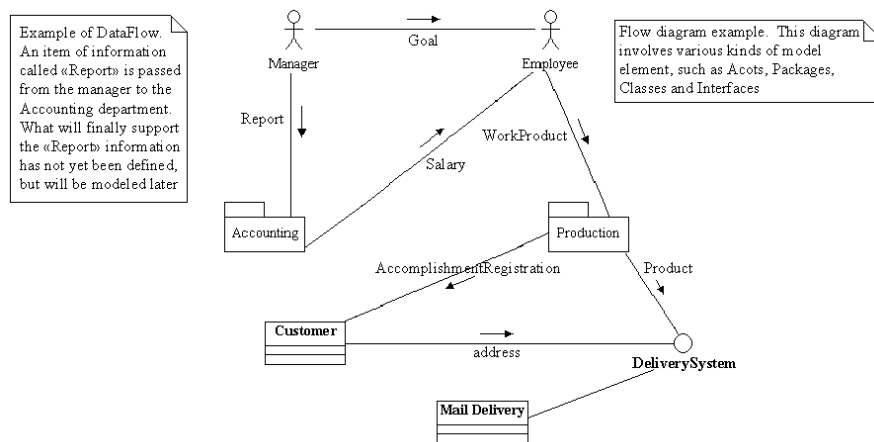


Figure 8-4. DataFlow with various examples

In the example shown in Figure 8-4, "*DataFlow*" exchanges are shown between actors, classes, interfaces and packages. Components, nodes and use cases can also be shown. A flow diagram simply presents an example of information being exchanged between different system entities, and is used only to show certain exchanges. For example, we determine here that a "*manager*" gives "*goals*" to an "*employee*", and that the "*accounting*" department (package) pays a "*salary*" to the "*employee*".

Each of these "*DataFlows*" can be associated with a signal. For example, a "*salary*" signal or an "*address*" signal can exist. These signals will subsequently be associated with a model element which will truly represent them. For example, the "*salary*" signal can be associated with the "*salary*" class, or with a "*paySalary*" operation. The final model will express how these signals are effectively transmitted.

Using flow diagrams

Overview

Flow diagrams are a prior study tool and are first defined through discussion and confirmation with project players. A precise, detailed model is then constructed, to exactly represent the system which is to be realized.

Step 1: Defining communicating entities

You must first decide Who and What will figure in the first draft of these diagrams. The Who and What can be:

- ◆ actors who interact with the system
- ◆ packages which represent "*sub-systems*"
- ◆ important components introduced into the system, etc.

These entities must be significant to all the users who will handle or validate these diagrams. For example, it is possible to represent classes in this type of diagram, but they are often too detailed for players to comprehend.

If these entities are numerous (more than 7), they must be structured.

Step 2: Defining the nature of information exchanged in the system and which we wish to present

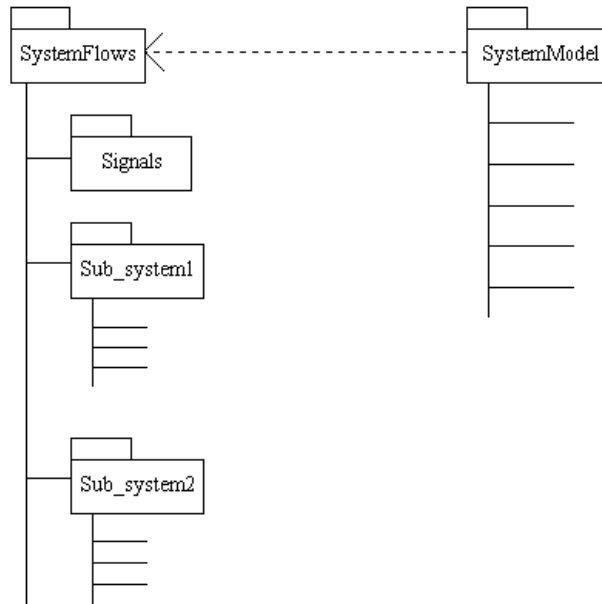


Figure 8-5. Typical structure of a model

We can use a structure of the type shown in Figure 8-5. In the system flow diagram, we will create a package dedicated to the definition of signals, as well as structure our "*communicating entities*", if necessary.

In the "*signals*" package, we will create signals which define every piece of information exchanged.

Note: We recommend that you document communicating entities and signals as you go along, by entering "*description*" notes.

Step 3: Drawing flow diagrams

Starting from communicating entities, define as many diagrams as necessary to illustrate different communications. Data Flows will reference signals created during Step 2.

Note: Steps 2 and 3 are often conducted parallel to each other. It is possible, for example, to create a Data Flow, then name it, and then subsequently create a signal which corresponds to this type of information, and connect the Data Flow to the signal.

Check that all Data Flows are connected to a signal, and verify that no signals have been duplicated. Use generalization between signals where information specialization exists.

Step 4: Making the detailed model application

Communicating entities are extremely good for structuring the application model. Each Data Flow asks three questions, which the detailed model must answer:

- ◆ Which model element represents the information conveyed (an operation, a parameter or a class)? The signal which corresponds to the Data Flow must be linked to this model element.
- ◆ What is sending this signal? Generally speaking, a class is at the origin of a signal.
- ◆ What deals with this signal? Once again, it is usually a class which deals with signals.

At the end of detailed modeling, check that the three questions have been answered for each signal.

Conclusion

Flow Diagrams are a useful preliminary analysis tool. They act alongside other tools, such as use cases and dictionaries.

Each of these models can be used to justify the detailed model, by providing a high level angle of vision.

Examples of Flow Diagrams

Overview

Two examples of flow diagrams will be presented here. The first is relative to preliminary analysis, and the second to preliminary design. In each case, flow diagrams are used to show the information which can be exchanged between system entities.

Company organization example

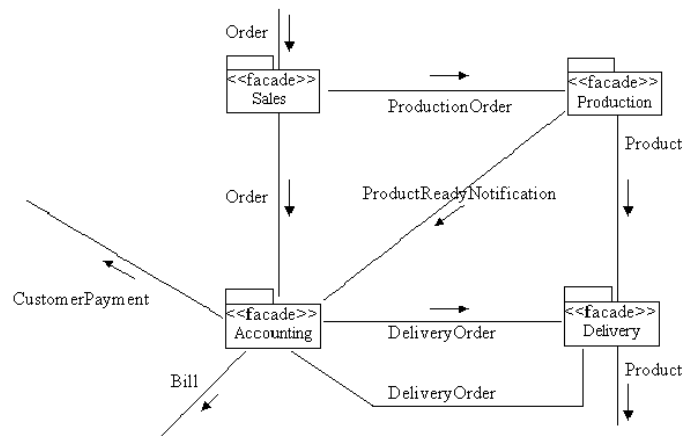


Figure 8-6. Company organization diagram

In this example, packages represent the various company departments. These packages are "facade" packages, since they do not necessarily constitute the definitive structure of the model once completed.

We attempt to clarify the role of each department, by representing the information it receives and sends. In this diagram, the following is seen:

- ◆ A *Data Flow* cannot have an origin or a destination ("order" or "product" case).
- ◆ Several *Data Flows* can represent the same information. For example, there are two "order" and two "product" *Data Flows*. The "order" or "product" information will be materialized by the creation of the signal, which itself can only exist once.

When this model has been accepted, detailed modeling is then carried out, in order to represent exactly which model element will translate the signal (a class, an operation or a parameter).

Architecture example

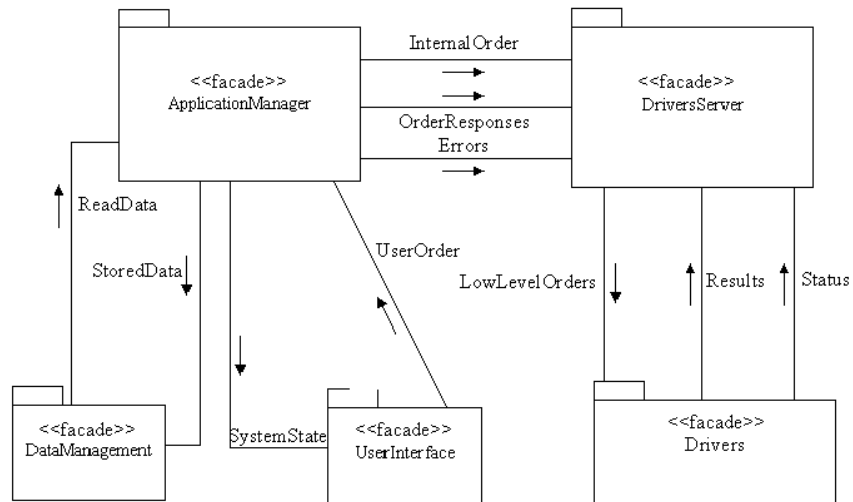


Figure 8-7. Architecture example

In this example, we use facade packages to show how models which make up separate final physical units are grouped. Future libraries, processes and applications are represented.

It is useful to regroup elements before realizing a deployment diagram. Information which must be exchanged between these elements is then expressed.

Technical questions will then be asked, in order to find out how to implement these information flows:

- ◆ by sending messages
- ◆ through inter-process communication, etc

Index

- 4-88
- :Importing modules into the current UML modeling project 1-8
- {noanalysis} tagged value 4-31
- {nodesign} tagged value 4-31
- + 4-88
- About 6-18
- Accessing macro commands 4-45
- Action states 7-72, 7-73
- Activating generation on elements 5-5
- Activating/deactivating Objecteering/UML process wizards 1-8
- Activating/deactivating removable consistency checks 1-8
- Activity diagram 1-22, 4-34, 6-8, 7-71
 - Definition 7-69
 - Elements which can be created or referenced 7-70
 - Example 7-69
- Activity diagram: 5-3
- Activity graph 5-3, 7-71
- Actor 3-38, 3-40, 4-34, 4-85, 5-6, 7-53, 7-69, 8-7, 8-10
- Adding modules 4-8
- Advanced mode search types 4-74
- Advanced search mode 4-72
- Assisted data entry 1-12
 - Example 1-14
- Association 2-18, 3-40, 5-6, 5-18, 7-3, 7-9, 7-13, 7-38
- Association end role 3-39
- Association link 3-38
- Asynchronous messages 7-48
- Attribute 4-83, 6-9, 7-16, 7-18, 7-21
- Attribute dialog box 1-14
- Attribute link 4-83
- Attribute role 3-39, 4-83
- Available types 4-77, 4-79
- Backward compatibility 3-8
- Begins with 4-68
- Branches 7-48
- Case sensitive 4-68, 4-70
- Changing the presentation of elements in diagrams 5-5
- Changing UML modeling project 3-6, 3-30
- Changing UML modeling project and saving 3-30
- Checking focus 7-47
- Class 1-12, 1-16, 1-20, 2-16, 3-28, 3-36, 3-38, 3-39, 4-20, 4-34, 4-83, 5-6, 6-9, 7-3, 7-6, 7-9, 7-16, 7-28, 7-38, 7-40, 7-42, 7-57, 7-68, 7-69, 8-12, 8-14
- Class association 7-9
- Class diagram 1-11, 1-22, 2-5, 2-11, 2-13, 4-34, 5-3, 5-4, 6-8
 - Automatic and non-automatic modes 7-16
 - Creating a class association 7-9
 - Creating a qualifier on an association 7-13
 - Creating an attribute 7-21
 - Creating an operation 7-19
 - Creating rake bar links 7-14
 - Definition 7-3
 - Deleting an attribute or an operation 7-24
 - Elements which can be created or referenced 7-5
 - Example 7-4
 - Expanding elements 7-6

- Managing class attributes and operations 7-18
- Modifying a class 7-23
- Modifying an attribute 7-22
- Operation dialog box 7-20
- Redefining generalization rake bar links 7-15
- Template class 7-24
- Class instance 7-38, 7-42
- Class instances 1-22
- ClassAssociation metaclass 7-9
- Classifier role 3-39, 7-42
- Clearing the contents of the console 3-18
- Code generation 1-20, 1-21, 3-18
- Collaboration 4-34, 7-42, 7-49, 7-51
- Collaboration diagram 1-22, 4-34, 5-3, 6-8, 8-3, 8-7
 - Definition 7-49
 - Elements which can be created or referenced 7-50
 - Example 7-49
 - Orientation of messages 7-52
- Commands 1-8
- Communication 3-39
- Communication link 3-40, 4-34, 4-85
- Communication links 3-37
- Component 3-38, 3-41, 4-34, 7-25, 7-28, 7-31, 7-69, 8-10
- Component instance 3-41
- Component-based approach 2-8
- Composition tree 1-20, 3-36
- Concurrent state 7-65
- Concurrent states 7-64
- Conditions 7-48
- Configuration 1-20
 - Modules frequently present 3-27
 - UML Modeler tab 3-27
- Consistency checks 1-3, 1-12, 1-20, 3-36, 4-10
 - Flexibility 1-12, 3-34
 - Obligatory 3-34, 3-36
 - Optional 3-34
 - Removable consistency checks 1-12
- Consistency management 1-12
 - Example 1-17
- Consistency mechanisms
 - Assisted data entry 1-13
 - Consistency management 1-16
- Console 1-3, 1-10, 1-19, 1-20, 2-5, 3-11
 - Clearing the contents of the console 3-18
 - Description 3-18
 - Messages and generation traces 4-36
 - Messages and import traces 4-37
 - Saving console contents in a file 3-18
 - Visualizing a warning 4-36
 - Visualizing an error message 4-36
 - Visualizing error messages 3-18
 - Visualizing warnings 3-18
- Console functions
 - Clearing the console 1-10
 - Saving the contents of the console 1-10
 - Tracing operations 1-10
- Constraints 4-34
- Contains 4-68
- Context menu
 - On a class in a diagram 6-13
- Context menu on a diagram element 6-11

- Context menus 4-19
- Continuous entry creation mode 2-10
- Control transition 5-6
- Copy/Paste operation 4-16
- Copy/Pasting diagrams 5-5
- Creating a box 5-7
- Creating a collaboration diagram 7-51
- Creating a deployment diagram 7-27
- Creating a deployment instance diagram 7-34
- Creating a diagram 5-5
- Creating a new UML modeling project 1-8, 3-4
- Creating a state diagram 7-59
- Creating a sub-system in a class diagram 7-8
- Creating a use case diagram 7-55
- Creating an activity diagram 7-71
- Creating an object diagram 7-40
- Creating elements in diagrams 5-5
- Creating several elements of the same kind 5-6
- Creation icons
 - Actor 4-22
 - Class 4-22
 - Node 4-22
 - Signal 4-22
 - State machine 4-22
 - Use case 4-22
- Creation icons
 - Action state 4-23
 - Activity graph 4-22
 - Association 4-22
 - Attribute 4-22
 - Attribute link 4-23
- Classifier role 4-23
- Collaboration 4-22
- Component 4-22
- Component instance 4-23
- Data flow 4-22
- Enumeration 4-22
- Event 4-22
- Instance 4-22
- Interface class 4-22
- Link 4-22
- Node instance 4-22
- Object flow state 4-24
- Operation 4-22
- Package 4-22
- Parameter 4-23
- Partition 4-24
- Return parameter 4-23
- State 4-22
- Structural units 4-22
- Sub-activity state 4-23
- Sub-system 4-22
- Template parameter 4-22
- Type 4-22
- Creation messages 7-48
 - Ctrl A 4-87
 - Ctrl C 4-87
 - Ctrl D 4-87
 - Ctrl E 4-87
 - Ctrl F 4-87
 - Ctrl G 4-87
 - Ctrl I 4-87
 - Ctrl K 4-87
 - Ctrl L 4-87
 - Ctrl M 4-87
 - Ctrl N 4-87
 - Ctrl O 4-87

- Ctrl P 4-87
- Ctrl R 4-87
- Ctrl S 4-87
- Ctrl U 4-88
- Ctrl V 4-88
- Ctrl X 4-88
- Ctrl Y 4-88
- Ctrl Z 4-88
- Cut operation 4-84
- Data flow 5-6, 8-8, 8-12, 8-14
- Data type 3-36, 3-39
- Data types 3-37
- DataFlow 3-39, 3-40
- DataFlows 3-37
 - Overview 8-7
- DataType 3-38, 4-34
- Delete operation 4-84
- Dependency 5-6, 5-18, 7-3
- Deployment diagram 1-22, 4-34, 5-3, 6-8, 8-15
 - Creating a component inside a node 7-28
 - Definition 7-25
 - Elements which can be created or referenced 7-26
 - Example 7-25
 - Referencing elements within a component 7-29
- Deployment instance diagram 1-22, 4-34, 5-3, 6-8, 8-3
 - Creating a node instance 7-36
 - Definition 7-31
 - Elements which can be created or referenced 7-33
 - Example 7-32
- Destroying elements in diagrams 5-5
- Destruction messages 7-48
- Diagram
 - Activating a context menu 5-25
 - Context menu 5-26
 - Deactivating a context menu 5-25
- Diagram creation
 - Properties editor 2-11
- Diagrams 1-11, 1-20, 2-5, 3-17, 3-21, 4-21, 4-26, 4-58, 5-3, 7-7
 - Activity diagram 3-22
 - Class diagram 3-21
 - Collaboration diagram 3-22
 - Context menu 7-16
 - Deployment diagram 3-21
 - Deployment instance diagram 3-22
 - Model elements 3-21
 - Modify command 6-8
 - Object diagram 3-21
 - Sequence diagram 3-21
 - State diagram 3-22
 - Use case diagram 3-21
- Diagrams set
 - Description 4-62
- Dialog box 1-20
- Directories set
 - Description 4-64
- Dockable windows 1-4, 1-8, 1-20, 3-14
- Documentation generation 1-21
- Documentation work product 3-23
- Drag and drop 4-16
- Drag and drop 5-22
- Drag and drop function 2-15, 5-22
- Drag and drop operation 4-84
- Edit menu 4-4
 - Consult command 4-4
 - Copy command 4-4

- Cut command 4-4
- Delete command 4-4
- Empty clipboard command 4-4
- Modify command 4-4
- Move command 4-4
- Paste command 4-4
- Redo command 4-4
- Undo command 4-4
- Edit/Search 4-66
- Element type 4-73, 4-79
- Elements
 - Masking 5-24
 - Showing 5-22
- Embedded creation 7-28
- Ends with 4-68
- Enterprise Edition 1-20
- Enumeration 3-38, 3-39
- Enumerations 3-37
- Error messages 1-10, 1-20, 3-18
- Event 5-6, 7-67
- Events 1-22
- Example of a simple search 4-69
- Example of an advanced search 4-81
- Exceptions 7-11
- Exiting the tool
 - Exiting and saving 3-29
- Expanding elements in a class diagram
 - Expanding a package 7-7
- Explorer
 - Context icons 4-20
- Explorer 1-9, 1-12, 1-16, 1-20, 2-5, 2-6, 2-14, 2-15, 2-16, 3-13, 3-15, 3-21, 4-19, 4-21, 4-83, 4-84, 5-3, 5-4, 5-22, 7-13, 7-24, 7-59, 7-67, 7-71, 7-73
- Browsing 4-19
- Graphic representation 3-15
- Launching an explorer 4-16
- Principal functions 4-16
- Read-only mode 4-18
- Update mode 4-17
- Explorer functions
 - Constructing a model 1-9
 - Creating and editing graphic elements 1-9
- Explorer functions
 - Browsing 1-9
 - Creating elements 4-16
 - Destroying elements 4-16
 - Duplicating elements 4-16
 - Filtering visualized element types 4-16
 - Modifying elements 4-16
 - Moving elements 4-16
 - Referencing elements 4-16
 - Visualizing UML modeling project components 4-16
- Explorers 1-3
- Facade package 8-15
- File menu 4-3
 - Clear console command 4-3
 - New command 4-3
 - Open command 4-3
 - Quit command 4-3
 - Save command 4-3
 - Save console command 4-3
- Filter conditions 4-72
- First Steps
 - Creating a class diagram 2-11
 - Creating a diagram 2-3
 - Creating a link in a diagram 2-18
 - Creating a package 2-7

- Creating a sub-system in the explorer 2-8
- Creating a UML modeling project 2-3
- Creating an operation 2-16
- Creating and manipulating objects in a diagram 2-3
- Creating classes 2-9
- Creating elements in the explorer 2-3
- Launching the tool 2-3
- Masking elements in a diagram 2-14
- Modifying an element 2-17
- Preparation 2-3
- Resizing an object 2-19
- Showing elements in a diagram 2-13
- Showing elements using the drag and drop function 2-15
- Working in the explorer 2-6
- Flow diagram
 - Example of notation 8-9
 - Overview 8-7
 - UML extensions 8-8
- Flow diagrams 8-13
- Formalism set
 - Description 4-58
- General contents 6-18
- General ergonomics 1-4
- General view of Objectteering/UML on PC 1-5
- Generalization 3-41, 4-34, 5-6, 5-11, 5-18, 7-3, 7-14, 8-12
- Generalization link
 - Modifying the drawing 7-15
- Generalization links 3-36, 3-37, 3-39, 3-40, 4-27
- Grab function 5-19
- Grabbing elements 5-19
- Graph menu 4-6
 - Align command 4-6, 6-17
 - Fit to contents command 4-6
 - Grid command 4-6, 6-16
 - Home command 4-6
 - Layout command 4-6
 - Redraw command 4-6
 - Resources command 4-6, 6-14
 - Zoom back command 4-6
 - Zoom forward command 4-6
- Graphic editor 2-16
 - Description 5-4
 - Pop-up menu 5-4
 - Principal functions 5-5
- Graphic editor functions
 - Creating a model element and its graphic representation simultaneously 1-9
 - Destroying a model element 1-9
 - Diagram management 1-9
 - Graphic representation of model elements 1-9
 - Masking 1-9
 - Modifying a model element 1-9
 - Modifying the graphic representation of a model element 1-9
 - Printing a diagram 1-9
 - Showing 1-9
- Graphic editor tool bar 2-19
- Graphic editors 1-3, 1-9, 1-12, 1-20, 2-6, 3-13, 4-83, 4-84
- Graphic elements 1-9, 2-13
 - Zoom function 6-4
- Graphic object

- Categories 5-6
- Creating a class in a class diagram 5-7
- Creating a link 5-8
- Creating an association 5-9
- Modifying link values 5-10
- Graphic objects
 - Boxes 5-6
 - Links 5-6
- Help 6-18
- help menu 6-18
- Highlight function 6-7
- Hypertext links 3-20, 4-39, 4-41
- Icons 1-6
- Implementation link 4-34
- Implementations 3-40
- Import administration 1-20, 3-18
- Import all from server 4-46
- Import operations 4-37
- Importing elements from other UML modeling projects 3-28
- Instance 3-39, 3-41, 4-83, 5-6, 7-31
- Instances 3-40
- Interface class 3-38, 3-39
- Interface classes 3-40
- Interface set
 - Description 4-60
- Interfaces 3-37
- Internal transition 7-72
- J expressions 4-73
- J filter conditions 4-72
- J language 4-72
- J rules 3-27
- Jumping to elements 4-71
- Keyboard shortcuts 4-66
- Launching a graphic editor 5-5
- Launching a new explorer 1-8
- Launching Objectteering/UML 3-3
- Launching the search 4-66
- Launching the search engine 4-38
- Laying out elements in diagrams 5-5
- Link 4-83, 4-85, 5-8, 7-7, 7-38
- Link end 3-39
- Link forms
 - Free 5-11, 5-13
 - Orthogonal 5-11, 5-13
 - Rake bar 5-11, 7-15
 - Shared target 5-11
- Links 1-22, 2-18
 - Color presentation 5-11
 - Drawing links 5-11
 - Drawing orthogonal links 5-12
- Local 4-46
- Local notes 3-33, 4-86
 - Handling 4-86
- Local tagged values 3-33, 4-86
 - Handling 4-86
- Looking for information 4-40
- Lowercase characters 4-67
- Macro commands
 - Heavyweight/Lightweight client 4-46
 - Server 4-48
 - Standalone 4-47
- Macros 1-7
 - Comment to description notes 4-49
 - Creating macros 4-51
 - Deleting macros 4-56
 - Executing macros 4-49
 - External editors 3-25
 - Macro commands 4-45

- Modifying macros 4-54
- Overview 3-25
- Parameterizing the Macros module 4-44
- Selecting the Macros module 4-42
- Sort by name 4-49
- Sort by visibility 4-49
- Main explorer 1-4, 4-16, 6-7
- Main window 1-3, 1-8, 1-20, 2-5, 4-16
 - Console 3-13
 - Description 3-12
 - Explorer 3-13
 - Graphic editors 3-13
 - Menu bar 3-13
 - Properties editor 3-13
 - Status bar 3-13
 - The console 1-10
 - Tool bar 3-13
- Manipulating graphic elements
 - Modifying the size of elements 6-3
- Manipulating model elements within a UML modeling project 3-15
- Masking elements 5-22
- Masking elements in diagrams 5-5
- Matches exactly 4-68
- Members 3-37
- Menu bar 1-20, 3-13
- Menus 1-6
- Message 5-3, 7-38, 7-42, 7-46, 7-49
 - Asynchronous 7-46
 - Synchronous 7-46
- Message lifeline 7-47
- Messages 1-22
- Metamodel 4-67
- Model elements 1-3, 1-9, 1-12, 1-13, 1-16, 1-20, 3-13, 3-15, 3-21
 - Associated diagrams 1-11
- Model/Instance
 - Example of the first approach 8-4
 - Example of the second approach 8-6
- Model/Instance approach
 - Presentation 8-3
- Models 4-67
- Modification of modules used 4-8
- Modifying a graphic element 5-21
- Modifying elements
 - Procedure 4-25
- Modifying elements in diagrams 5-5
- Modifying UML modeling project configuration 1-8
- Module 1-21
- Modules 1-6, 3-23, 3-26, 3-27, 4-9
- Modules in charge of the read-only mode 3-33
- Move operation 4-84
- Moving an element 5-18
- Moving elements in diagrams 5-5
- Multi-user atomic units 3-33
- NameSpaces 3-37, 3-38, 3-39, 3-40, 3-41
- N-ary link 7-38
- Navigating between diagrams 5-5
- Node 3-38, 3-40, 7-28, 7-31, 7-36, 7-69
- Node instance 3-41, 7-36
- Notes 4-21, 4-27, 4-34
- Notes and constraints 4-77
- Object 7-42
- Object diagram 1-22, 4-34, 5-3, 6-8, 8-3, 8-6
 - Definition 7-38
 - Elements which can be created on referenced 7-39

- Example 7-38
- Object diagrams
 - Creating associations 7-41
 - Creating instances 7-41
 - Creating messages 7-41
- Object flow state 7-73
- Object flow states 7-72
- Objectteering/C++ 1-6, 3-23, 4-26
- Objectteering/Design Patterns for C++ or Java 1-6
- Objectteering/Design Patterns for Java 2-5, 3-5
- Objectteering/Documentation 1-6, 1-21, 3-23, 3-26, 3-27, 4-26
- Objectteering/Introduction 4-8
- Objectteering/Java 1-6, 1-21, 2-5, 3-5, 3-17, 3-23, 3-26, 3-27, 4-8, 4-26
- Objectteering/Macros 1-7
- Objectteering/Metamodel 8-8
- Objectteering/Metrics 1-21
- Objectteering/Model Dialog Boxes 1-3, 2-17, 4-25, 6-9, 7-20, 7-48, 7-63, 8-8
- Objectteering/Multi-user 3-33
- Objectteering/Process Wizard 4-9
- Objectteering/Process Wizards 1-7
- Objectteering/UML configuration 4-8
- Objectteering/UML consistency checks 3-34
- Objectteering/UML consistency mechanisms 2-6
- Objectteering/UML diagrams 1-22
- Objectteering/UML environment variables
 - OBJING_PRINTER 6-5
- Objectteering/UML Modeler 2-5, 3-5, 3-6
- Objectteering/UML modules 1-6
- Objectteering/UML obligatory consistency checks on different elements 3-36
- Objectteering/UML Profile Builder 1-21, 3-27
- Objectteering/UML Teamwork 3-28, 5-4
- Objectteering/UML terms
 - Glossary 1-20
- Objectteering/UML windows 1-3
 - Console 1-3
 - Explorers 1-3
 - Graphic editors 1-3
 - Main window 1-3
- Objectteering/Visual Basic 4-26
- Objects 1-22
- objing command
 - Parameters 3-3
- Obligatory consistency checks 3-36
 - Actor class 3-39
 - Association class 3-38
 - AssociationEndRole class 3-39
 - Attribute class 3-38
 - AttributeRole class 3-39
 - Class class 3-37
 - Communication class 3-40
 - Component class 3-41
 - ComponentInstance class 3-41
 - DataFlow class 3-39
 - DataType class 3-37
 - Element class 3-36
 - Enumeration class 3-37
 - Generalization class 3-38
 - InternalTransition class 3-40
 - LinkEnd class 3-39
 - Node class 3-40
 - NodeInstance class 3-41

- Package class 3-36
- Parameter class 3-38
- Realization class 3-38
- Signal class 3-39
- StateMachine class 3-40
- Transition class 3-40
- Use class 3-38
- UseCase class 3-40
- Opening a diagram 5-5
- Opening a graphic editor 5-5
- Opening an existing UML modeling project 1-8, 3-6
- Opening the search window 4-66
- Operation 1-16, 2-16, 2-17, 4-16, 4-34, 4-83, 6-9, 7-16, 7-18, 7-42, 7-57, 7-68, 7-69, 8-12, 8-14
- Operation traces 3-18
- Package 1-9, 1-11, 1-17, 1-20, 3-28, 3-36, 3-39, 3-40, 4-16, 4-20, 4-34, 5-6, 5-23, 7-3, 7-6, 7-28, 7-34, 7-38, 7-40, 7-42, 7-55, 7-57, 7-69, 8-10, 8-14
- Parameter 4-16, 8-12, 8-14
- Parameter sets
 - Diagrams 4-61
 - Directories 4-63
 - Formalism 4-58
 - Interface 4-59
 - UML Profiles 4-65
- Parameters 3-26
- Parent type 4-77
- Partitions 7-73
- Paste operation 4-84
- Personal Edition 1-21
- Predefined types 3-6
- Printing a diagram 6-5
- Printing diagrams 5-5
- Process Wizards 4-9
- Professional Edition 1-21
- Properties editor 1-3, 1-10, 1-12, 1-16, 1-21, 2-5, 3-13, 4-21, 4-84
- C++ tab 1-10, 4-33
- Diagrams tab 1-10, 2-11, 3-17, 4-29, 5-5
- Documentation tab 1-10, 3-17, 4-31
- Items tab 1-10, 3-17, 4-27
- Java tab 1-10, 3-17, 4-33
- Overview 4-26
- Tabs added for specific modules 1-7
- VB tab 4-33
- Visual Basic tab 1-10
- Pseudo state 7-62, 7-73
 - Branch 7-62
 - Fork 7-62
 - Join 7-62
- Read-only elements
 - Graphic representation 4-83
 - Handling elements and the read-only mode 4-84
- Read-only mode 1-21, 3-33, 4-18, 4-21, 4-83, 5-4
 - Destroying a model element 4-84
 - Graphic representation 3-33
 - Handling elements and the read-only mode 4-84
 - Links 4-85
 - Local annotation 3-33, 4-86
 - Model elements 3-33
 - Overview 3-33
- Read-write mode 3-33
- Receiving a UML modeling project 3-8
- Receiving and renaming a UML modeling project 3-8

- Receiving and upgrading a UML modeling project 3-8
- Receiving UML modeling projects 3-7
- Receiving, renaming and upgrading a UML modeling project 3-8
- Redefining a free link
 - Free mode 5-15
 - Orthogonal mode 5-14
- Redefining links
 - Moving a link 5-13
 - Moving a link end 5-13
- Redrawing links 2-18
- Referencing elements 7-29
- Referencing elements within a component 7-29
- Removable consistency checks 1-12, 1-21, 3-34, 4-10
 - Checking the model 4-12
 - Correcting errors 4-13
 - Deactivation 4-11
 - Displaying errors 4-13
 - Principle 3-35
 - Reactivation 4-15
- Removing modules 4-8
- Repeated entry system 5-6
- Re-sizing an element 5-20
- Role 5-3, 7-42, 7-49
- Roles 1-22
- Saving 1-8
- Saving a diagram 6-6
- Saving a work session 3-29
- Saving console contents in a file 3-18
- Saving the model context 3-31
- Search 3-32
- Search engine 3-19, 4-38, 6-18
 - Search engine window 4-39
 - Search from 4-68
 - Search function 3-32
 - Search in 4-68, 4-73, 4-77, 4-79
 - Search in all 4-74
 - Search in diagrams 4-74
 - Search in notes and constraints 4-74
 - Search in the model 4-74
 - Search options 4-68, 4-75
 - Begins with 4-68
 - Contains 4-68
 - Ends with 4-68
 - Matches exactly 4-68
 - Search results 4-70
 - Search window 3-32, 4-66
 - Searching diagrams 3-32
 - Searching for information 4-40
 - Searching for information in on-line documentation 3-19
 - Searching in all 4-80
 - Searching in diagrams 4-78
 - Searching in notes and constraints 4-76
 - Searching in the model 4-75
 - Searching metamodels 3-32
 - Searching models 3-32
 - Searching textual elements 3-32
 - Selecting an element 5-18
 - Selecting modules
 - Tabs added to the properties editor 1-7
 - Selecting several elements 5-18
 - Sequence diagram 1-11, 1-16, 1-22, 4-34, 5-3, 7-53, 8-3, 8-6, 8-7
 - Creating 7-45
 - Definition 7-42

Elements which can be created or referenced 7-44
 Example 7-43
 Messages 7-46
 Modify command 6-10
 Sequence message 5-6
 Sequence message activation time 7-47
 Sequence messages 7-48
 Server 4-46
 Shortcuts 4-66
 + 4-88
 Ctrl P 4-87
 Ctrl A 4-87
 Ctrl C 4-87
 Ctrl D 4-87
 Ctrl E 4-87
 Ctrl F 4-68, 4-87
 Ctrl G 4-87
 Ctrl I 4-87
 Ctrl K 4-87
 Ctrl L 4-87
 Ctrl M 4-87
 Ctrl N 4-87
 Ctrl O 4-87
 Ctrl R 4-87
 Ctrl S 4-87
 Ctrl U 4-88
 Ctrl V 4-88
 Ctrl X 4-88
 Ctrl Y 4-88
 Ctrl Z 4-88
 Showing and hiding the console 1-8
 Showing and hiding the principal explorer 1-8
 Showing and hiding the properties editor 1-8
 Showing contents 7-30
 Showing elements 5-22
 Drag and drop 5-23
 Links 5-24
 Showing elements in a diagram
 The drag and drop function 2-15
 Signal 3-37, 3-38, 3-39, 4-34, 5-6, 7-68, 7-69, 8-12, 8-14
 Signals 4-58
 Simple search
 Example 4-69
 Simple search mode 4-67
 Site 3-26
 Softeam on the Web 6-18
 State 4-16, 5-6, 7-72
 State diagram 1-16, 1-22, 4-34, 4-83, 5-3, 6-8, 7-72, 8-7
 Concurrent states 7-63
 Creating and referring to events 7-67
 Creating pseudo states 7-62
 Creating sub-states 7-61
 Creating substates in threads 7-66
 Elements which can be created or referenced 7-58
 Example 7-57
 Threads 7-63
 State machine 1-22, 3-40, 4-34, 4-83, 7-57, 7-59, 7-60
 Status bar 1-21, 3-13
 Step by step search 4-68
 Stereotypes 1-21, 4-34
 Structural element 1-21
 Structural elements 4-20
 Sub activity state 7-73
 Sub activity states 7-72

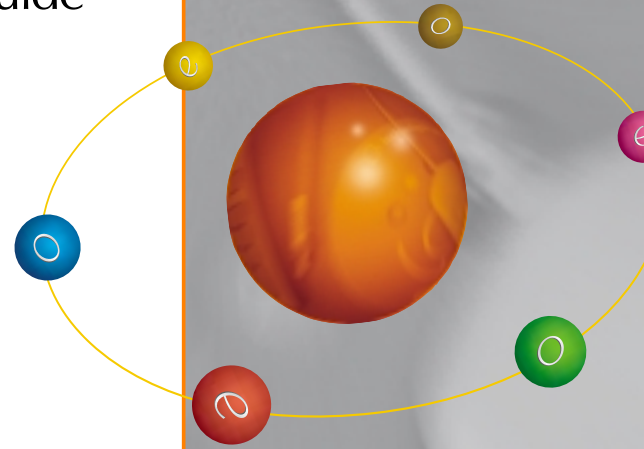
- Sub-system 1-21, 2-8, 7-42, 7-55, 7-69
- Swimlanes 7-73
- Switching from an element in a diagram to the same element in the explorer 6-7
- Tag type 3-36
- Tagged value 7-24
- Tagged value definitions 1-21
- Tagged values 1-21, 4-21, 4-27, 4-34, 4-58, 4-86
- Template class 7-24
- Template parameter 7-24
- Terminal element creation icons 3-17, 4-26
- Terminal elements 1-10, 1-21, 4-26
 - Icons 4-34
- Text type 3-36
- Thread 7-65
- Threads 7-64
- Throwing and catching exceptions 7-11
- Tool bar 3-13
- Tools menu 4-7
 - Import command 4-7
 - Modify configuration command 4-7
 - Modules command 4-7
 - Process wizard command 4-7
- Transfer function
 - Description 3-28
- Transferring elements between UML modeling projects 3-28
- Transition 3-40, 7-67
- Trigger transition 5-6
- Type 4-77
- Type of UML modeling project 2-5, 3-5
- UML extensions 8-8
- UML model root 1-3, 1-21, 2-5, 2-7, 3-5, 3-15, 4-16, 4-19
- UML model root name tickbox 2-5, 3-5
- UML model type 2-5, 3-5
- UML Modeler
 - Diagrams 3-27
 - Directories 3-27
 - Formalism 3-27
 - Interface 3-27
 - UML profiles 3-27
- UML modeling project 1-8, 1-9, 1-21, 2-6, 3-3, 3-6, 3-13, 3-26, 3-28, 3-29, 3-36, 5-5
- UML profile 1-21, 3-36
- UML profiles 1-21
- UML Profiles set
 - Description 4-65
- UML profiling project 1-21, 3-3, 4-83
- Unmasking elements in diagrams 5-5
- Upgrading a UML modeling project 3-10
- Upgrading UML modeling projects 3-7
- Uppercase characters 4-67
- Use 3-40, 5-6
- Use case 3-38, 3-39, 3-40, 4-85, 5-6, 7-42, 7-53, 7-69, 8-7, 8-12
- Use case diagram 1-11, 1-22, 4-34, 5-3
 - Definition 7-53
 - Elements which can be created and referenced 7-54
 - Example 7-53
 - Modify command 6-10
 - Use case boundaries 7-56
- Use link 4-34

Use links 3-36, 3-37
Using flow diagrams
 Overview 8-10
Using the grab function 5-19
View menu 4-5
 Copy graph image command 4-5
 Mask command 4-5
 Print command 4-5
 Save as command 4-5
 Select all command 4-5
 Show command 4-5
Visualizing elements in diagrams 5-5
Volumes 6-18
Warnings 1-10, 1-19, 1-20, 3-18
What's new 6-18
Windows menu 4-7
 Cascade command 4-7
 Close all command 4-7
 Close command 4-7
 Next command 4-7
 Previous command 4-7
Wizards 1-7
Work product 1-21
Work products 4-34
 Description 3-23
 Example 3-24

Objecteering/UML

Objecteering/Model Dialog Boxes User Guide

Version 5.2.2



Objecteering

Software

www.objecteering.com

Taking object development one step further

Contents

Chapter 1: Overview and general ergonomics	
Introduction	1-3
Creating a new element	1-4
Creating elements using the continuous entry creation mode	1-7
Modifying an existing element	1-10
Consulting an existing element	1-13
Entering references between elements	1-14
Standard dialog box tabs	1-19
Glossary	1-25
Chapter 2: Extensibility mechanism and general element dialog boxes	
Constraint dialog box	2-3
Note dialog box	2-6
Tag parameter dialog box	2-8
Tagged value dialog box	2-9
Chapter 3: Static model dialog boxes	
Package dialog box	3-3
Class and interface dialog box	3-6
Data type dialog box	3-9
Attribute dialog box	3-12
Operation dialog box	3-15
Parameter dialog box	3-20
Return parameter dialog box	3-23
Binary association dialog box	3-25
N-ary association dialog box	3-28
Class association dialog box	3-30
Enumeration dialog box	3-32
Enumeration literal dialog box	3-35
Signal dialog box	3-36
Data flow dialog box	3-38
Generalization dialog box	3-40
Use dialog box	3-42
Realization dialog box	3-44
Template parameter dialog box	3-46

Chapter 4: Use case model dialog boxes	
Use case dialog box	4-3
Actor dialog box	4-5
Communication link dialog box	4-7
Use case dependency dialog box	4-9
Chapter 5: State machine model dialog boxes	
State machine dialog box	5-3
State dialog box	5-5
Pseudo state dialog box	5-7
Transition dialog box	5-9
Internal transition dialog box	5-12
Event dialog box	5-15
Chapter 6: Activity model dialog boxes	
Activity graph dialog box	6-3
Action state dialog box	6-5
Sub activity state dialog box	6-8
Object flow state dialog box	6-11
Partition dialog box	6-14
Chapter 7: Physical model dialog boxes	
Node dialog box	7-3
Component dialog box	7-5
Node instance dialog box	7-7
Component instance dialog box	7-9
Chapter 8: Sequence and collaboration model dialog boxes	
Collaboration dialog box	8-3
Instance dialog box	8-5
Classifier role dialog box	8-8
Attribute link dialog box	8-11
Attribute role dialog box	8-13
Link dialog box	8-15
Collaboration message dialog box	8-17
Sequence message dialog box	8-20

Chapter 9: Diagram dialog boxes	
Class diagram dialog box	9-3
Deployment diagram dialog box	9-6
Deployment instance diagram dialog box	9-9
Object diagram dialog box	9-12
Sequence diagram dialog box	9-15
Collaboration diagram dialog box	9-18
Use case diagram dialog box	9-21
State diagram dialog box	9-24
Activity diagram dialog box	9-27

Index

Chapter 1: Overview and general ergonomics

Introduction

General presentation

Welcome to the *Objectteering/Model Dialog Boxes* user guide!

Model elements are entered in Objectteering/UML through the various different Objectteering/UML dialog boxes.

These dialog boxes can be opened from:

- ◆ the explorer
- ◆ the properties editor
- ◆ the graphic editors

Shortcuts exist to create large numbers of elements in a row (continuous entry creation mode), or to rapidly modify several existing elements, without having to close or open many different dialog boxes.

Operations on model elements

The following operations are possible:

- ◆ the creation of a new element
- ◆ the creation of elements using the continuous entry creation mode
- ◆ the editing of an element
- ◆ the consultation of an element

Elements may be edited or consulted from the explorer or graphic editor which is currently active.

Non "modal" windows

Objectteering/UML dialog boxes are non "modal", meaning they do not block the use of Objectteering/UML. You are, therefore, able to access other windows and browse the model (explorer, graphic editors, properties editor) in read only mode while a dialog box is active.

Creating a new element

Explorer

To create an element in the explorer, carry out the following steps (see figure 1-2 for an example of the creation of a new element):

- 1 - Select an embedding element.
- 2 - Click on a creation icon.
- 3 - Press "*Return*" to accept the name proposed by default, or click over the highlighted text in order to change this suggested name.

Example: Creating an attribute in a class

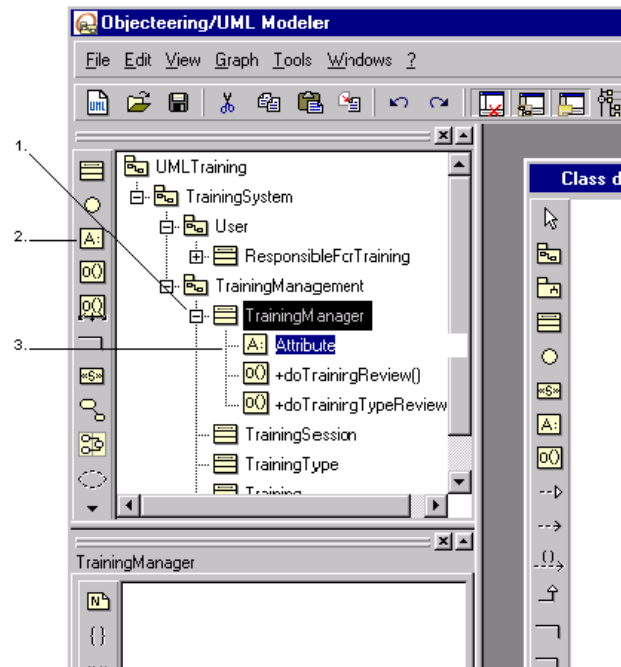



Figure 1-1. Creating an attribute in the "TrainingManager" class

Steps:

- 1 - Select a class.
- 2 - Click on the  "Create an attribute" icon.
- 3 - Press "Return" to accept the name suggested by default, or click over the highlighted text in order to change the suggested name. Confirm by pressing "Return".

Properties editor

The properties editor can be used to create certain elements, such as diagrams and links. To create an element in the properties editor:

- 1 - Select the embedding element in the explorer.
- 2 - Click on the relevant tab in the properties editor ("*Diagrams*" to create a diagram and "*Items*" to create different links and other terminal elements).
- 3 - Define the relevant information in the window which then appears.
- 4 - Confirm.

Note: Certain notes and tagged values specific to modules such as *Objecteering/Documentation* and *Objecteering/Java* can be created or associated through the "*Documentation*" and "*Java*" tabs of the properties editor. For further information, please refer to the user guide for the module in question.

Graphic editors

To create an element in a graphic editor:

- 1 - Click on a creation button in the palette.
 - 2 - Click in the editing zone where you wish to position the element.
 - 3 - Press "*Return*" to accept the name proposed by default, or click over the highlighted text in order to change the suggested name. Confirm by pressing "*Return*".
-

Creating elements using the continuous entry creation mode

Description

The continuous entry creation mode allows you to create several occurrences of the same type of element in the same dialog box (for example, classes). This mode is activated by clicking on the icon which corresponds to the type of element selected, whilst at the same time holding down the "*Ctrl*" key, or by double-clicking on the icon, and is recommended for entering large numbers of elements. The continuous entry creation mode functions in the explorer. In diagrams, the repeated entry system is used.

Creating from the explorer

To create elements using the continuous entry creation mode in the explorer:

- 1 - Select the embedding element.
- 2 - Click on a creation button in the palette holding down the "*Ctrl*" key, or double-click on the creation button.
- 3 - Still holding down the "*Ctrl*" Key, press "*Return*" to accept the name suggested by default, or click over the highlighted text in order to change the suggested name.
- 4 - Repeat this step as many times as necessary (step 3).
- 5 - To exit the continuous entry creation mode, clicking on the "*Escape*" key cancels both the creation in progress and the continuous entry mode itself. Alternatively, clicking at any other point in the explorer instead of pressing "*Return*" confirms the creation in progress and exits the continuous entry mode.

Example: Creating a class in a package

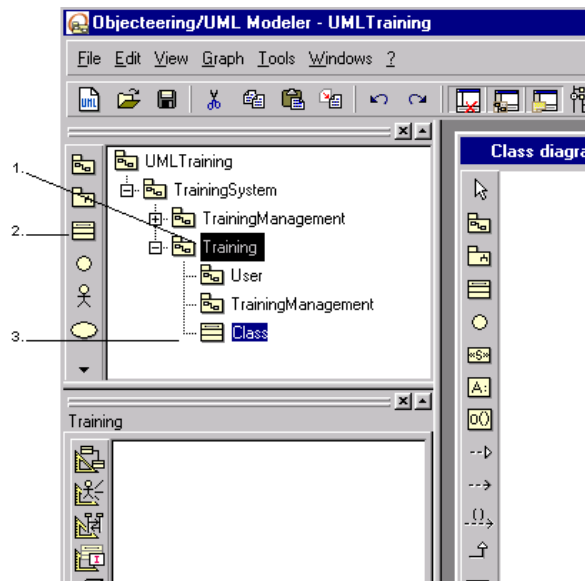



Figure 1-2. Creating a class using the continuous entry creation mode in the "TrainingSystem" package

Steps:

- 1 - Select the package in which the classes are to be created.
- 2 - Click on the  "Create a class" icon.
- 3 - Either accept the name proposed by default, or click over the highlighted text in order to change the suggested name.
- 4 - Press "Enter" to create the next class, and continue in this way until as many classes as necessary have been created.
- 5 - You may exit the continuous entry creation mode by clicking elsewhere in the explorer.

Creation from a graphic editor

To create elements using the repeated entry system in a graphic editor:

- 1 - Click on a creation button in the palette holding down the "*Ctrl*" key, or double-click on the creation button.
 - 2 - Click in the editing zone where you wish to position the element.
 - 3 - Press "*Return*" to accept the name proposed by default, or click over the highlighted text to change the suggested name.
 - 4 - Click once again on the left mouse button where you wish to locate the next element.
 - 5 - Repeat this step as many times as necessary to create as many elements as you wish.
 - 6 - To exit the repeated entry system, press the "*Escape*" key.
-

Modifying an existing element

Description

You can modify all the visible attributes of an existing element through its dialog box. The dialog box can be used to modify several occurrences of the same type of element, without the user having to close the dialog box.

Modifying from the explorer or the properties editor

To modify an element's name, simply select the element in question, and then click once on the left mouse button. The current name is highlighted, and you may type over it to make the necessary modifications.

Another way of modifying elements in the explorer and the properties editor is to carry out the following steps:

- 1 - Select the element.
- 2 - Click on the "*Modify*" item in the context menu or the "*Edit*" menu (you may also double-click on the element in question, which automatically edits the modification dialog box). Pressing "*Return*" also displays the modification dialog box.
- 3 - Modify the data in the dialog box which then appears.
- 4 - Click on "*Apply*".
- 5 - Redo steps 1 to 3 to modify another of the same element's properties.
- 6 - Click on "*Close*" when you have finished your modifications.

Modifying from a graphic editor

To modify the elements in the graphic editors without exiting the dialog box:

- 1 - Select the element.
- 2 - Double-click on the graphic element, or indeed click on the right mouse button and choose the "Modify" option from the context menu which appears. Pressing "Return" also displays the modification dialog box.
- 3 - Modify data in the dialog box which appears.
- 4 - Click on "Apply".
- 5 - Redo steps 1 to 3 to modify another of the same element's properties.
- 6 - Click on "Close" when you have finished your modifications.

Graphic representation

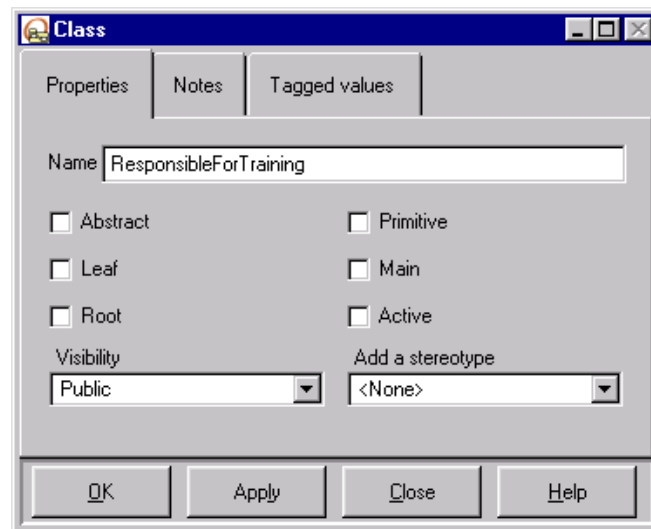


Figure 1-3. Editing the "Class" dialog box

Dialog box buttons

The ... button	is used to ...	Keyboard shortcut...
OK	confirm the entry and quit the dialog box.	Return
Apply	apply the changes.	Return
Close	exit the dialog box, without recording the entered data.	Escape
Help	provide online help on the element's entry fields.	F1

Consulting an existing element

Description

Dialog boxes are the same in consult mode as in edit mode, except that their generic buttons are not identical (the "OK" and "Apply" buttons are not available in consult mode) and no modifications can be made.

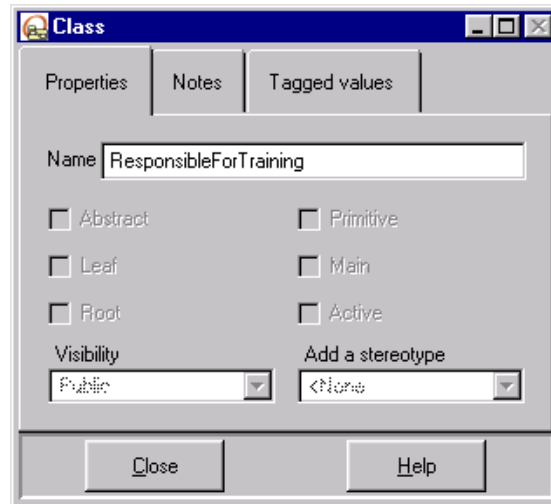


Figure 1-4. Consulting a class

Entering references between elements

Overview

In the dialog boxes or the explorer, it is often necessary to define links between elements, such as, for example, a reference between a package and a class, or a class which types an attribute, and so on. Three methods of entering references exist:

- ◆ *Selection from a list*: Objectteering/UML provides a limited list of possible elements.
- ◆ *Free selection*: Here, Objectteering/UML does not guide the user in his choice, which can be very wide, and the user will establish the reference by using drag and drop from the explorer.
- ◆ *Mixed selection*: Here, Objectteering/UML makes an incomplete suggestion to the user, with a list which is sometimes too extensive. The free selection mode is then proposed as an alternative.

Selecting from a list

To create references using the "selection from a list" mode, carry out the following steps, illustrated in Figure 1-5:

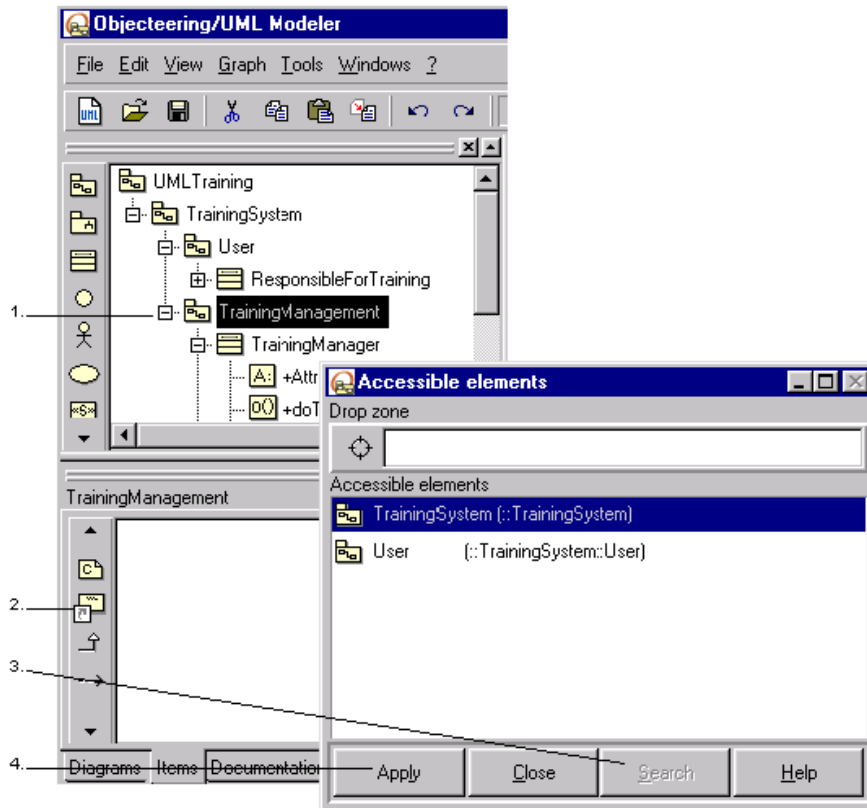



Figure 1-5. Selecting references from a list

Chapter 1: Overview and general ergonomics

Steps:

- 1 - Select a package, node instance, component, data flow or object in the explorer.
- 2 - Click on the  "Reference a unit" icon in the "Items" tab of the properties editor.
- 3 - Choose the element to be referenced from the combobox in the window. If necessary, click on the "Search" button to carry out a search of all elements which may be referenced.
- 4 - Click on "Apply" to confirm.

Free selection

To create references using the "free selection" mode, carry out the following steps, illustrated in Figure 1-6:

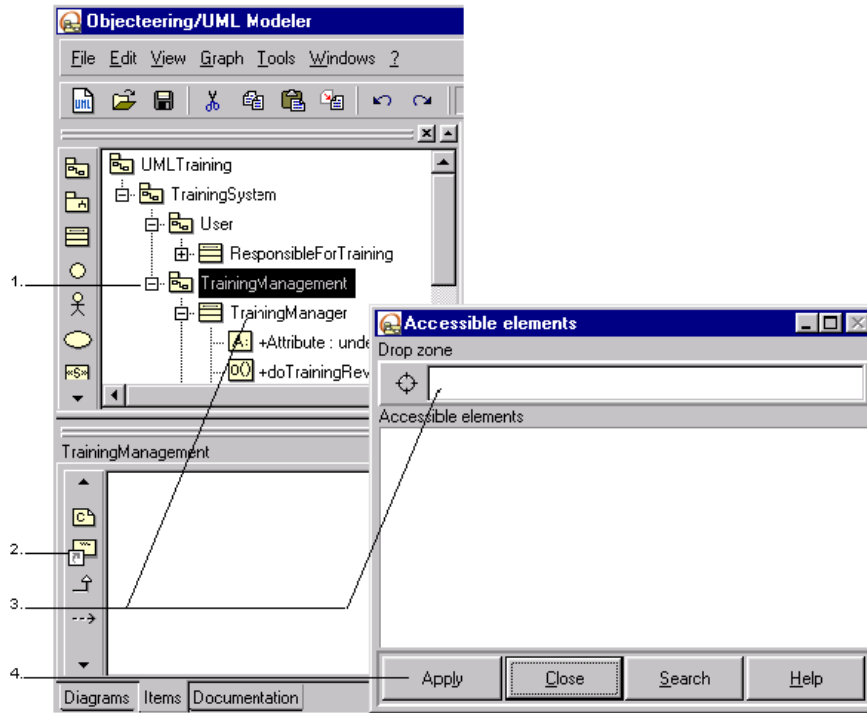



Figure 1-6. Using the "free selection" mode

Chapter 1: Overview and general ergonomics

Steps:

- 1 - Select a package, node instance, component, data flow or object.
 - 2 - Click on the  "Reference a unit" icon.
 - 3 - Select the item to be referenced, and using the drag and drop function, transport it to the "Drop zone" field. It will then appear both in the "Drop zone" field and the "Accessible elements" box.
 - 4 - Click on "Apply" to confirm.
-

Standard dialog box tabs

Overview

Standard dialog boxes contain three tabs: the "*Properties*" tab, the "*Notes*" tab and the "*Tagged values*" tab.

Properties tab

Figure 1-7 shows the "*Properties*" tab in a modification dialog box.

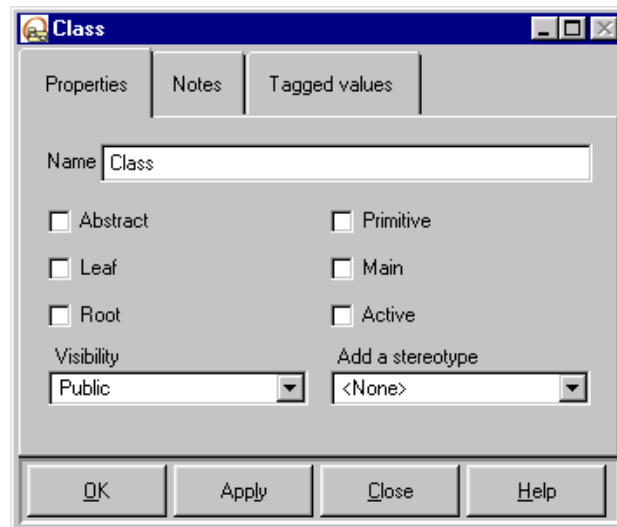



Figure 1-7. The "*Properties*" tab in the "*Class*" dialog box

The "*Properties*" tab is used to enter the essential attributes of an element (name, etc.).

Notes tab

 Each element can have several associated descriptive texts. Each text has a name, chosen from a list of authorized names, defining its nature and function. For example, a text called "description" can be found in generated documentation, a text called "C++" in generated C++ code, and so on.

Note: The note types available depend on which Objectteering/UML modules have been chosen for the project.

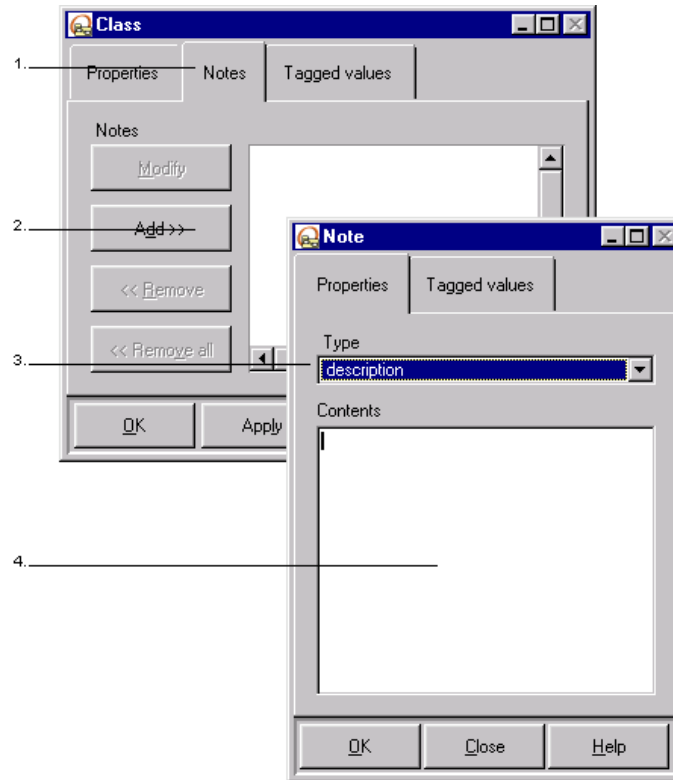


Figure 1-8. The "Notes" tab in the "Class" dialog box

Steps:


- 1 - Select the "Notes" tab.
- 2 - Click on "Add" to open a new entry dialog box.
- 3 - Click on the scrolling list and select the "*description*" note type.
- 4 - Enter your text.
- 5 - Confirm.

Modifying a description

To modify a description, carry out the following steps:

- 1 - Select the element in the right-hand list.
- 2 - Click on the "*Modify*" option from the context menu, available by clicking on the right mouse button over the selected element (double-clicking on the element in question also edits the modification dialog box), or the "*Edit*" menu. Pressing "*Return*" also displays the modification dialog box.
- 3 - Modify the data in the dialog box which appears.
- 4 - Click on "*OK*" to confirm.

Tagged values tab

 Tagged values are used to annotate model elements, in order to add a special meaning to them. For example, adding the `{persistent}` tagged value to a class will make the class persistent.

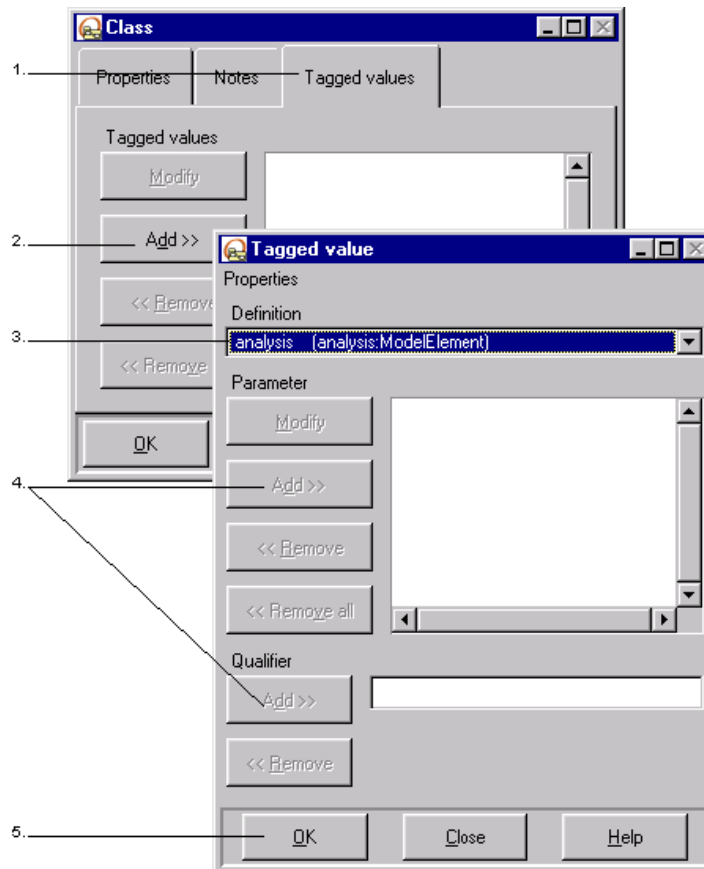


Figure 1-9. The "Tagged values" tab in the "Class" dialog box

To enter a tagged value, carry out the following steps:

- 1 - Select the "*Tagged values*" tab.
- 2 - Click on the "*Add*" button. The "*Tagged value*" dialog box then appears.
- 3 - Click in the scrolling list to select the tagged value definition.
- 4 - Click on the "*Add*" buttons in the "*Parameter*" and "*Qualifier*" zones, if these are to be added. (When you click on the "*Add*" button in these zones, the "*Tag parameter*" dialog box is displayed, and the value for the qualifier or parameter may be entered.)
- 5 - Click on "*OK*" to associate a text component to an element.

Note: The tagged values available depend on which Objectteering/UML modules have been chosen, and on the nature of the model element itself.

Modifying a tagged value

To modify a tagged value, carry out the following steps:

- 1 - Select the element in the "*Items*" tab of properties editor.
 - 2 - Click on the "*Modify*" option from the context menu, available by clicking on the right mouse button over the selected element (double-clicking on the element in question also edits the modification dialog box), or in the "*Edit*" menu. Pressing "*Return*" also displays the modification dialog box.
 - 3 - Modify the tagged value in the dialog box which appears.
 - 4 - Click on "*OK*" to confirm.
-

Glossary

Dialog box: Window in which model element values are entered.

Element: Model element the user can display, manipulate, or enter (for example, *class*, *attribute*, *note*, ...).

Continuous entry creation mode: Mode used to enter large numbers of elements of the same nature "in a row".

Property: Value which describes an element's characteristics and which will have an effect on the model.


Note: Free text, characterized by its type and content. This text completes the attached element for documentation and programming purposes.

Tagged value: Model annotation in the form of a "{tag}". It completes the properties and provides additional information which will be used by the Objecteering/UML modules.

Chapter 2: Extensibility mechanism
and general element
dialog boxes

Constraint dialog box

Entering constraints

 A constraint expresses a semantic restriction on an element or a set of elements being modeled. A constraint appears in the form of more or less formal text, which can be interpreted by a programming language.

The "*Constraint*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on constraints, please refer to the "*Constraint class*" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Constraint" dialog box

This screen (shown in Figure 2-1) is used to enter constraint values.

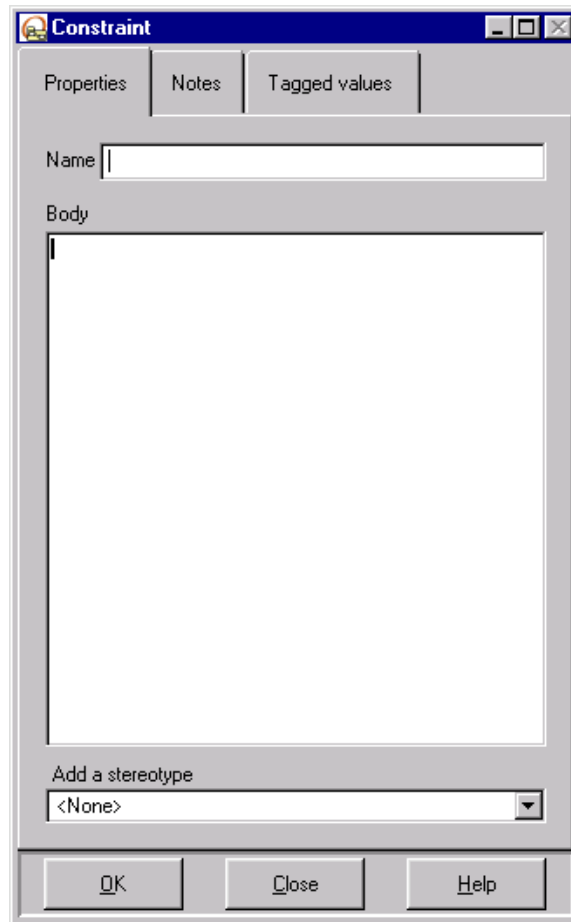



Figure 2-1. The "Properties" tab of the "Constraint" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This field presents the name of the constraint (a scrolling menu presents existing constraints from which the user can select). This field is exclusive of the *"Body"* field.
 - ◆ *"Body"*: This field allows the user to specify the body of the constraint, where its expression has not been reduced to a token or a symbol. This field is exclusive of the *"Name"* field.
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to *the Objectteering/UML Profile Builder* user guide). Pre- and post-conditions or invariant specific cases are defined here.
-

Note dialog box

Entering notes

 A note is any text part attached to a model element. Notes are used to add comments, descriptions, code and so on. In Objectteering/UML, notes are typed and may contain tagged values.

A note is made up of a name indicating its nature, and a textual content. A text must conform to the definition provided by its *"type"* (*TextType*).

The list of text types depends on the Objectteering/UML modules available.

The *"Note"* dialog box contains two tabs - *"Properties"* and *"Tagged values"*. For information on these standard dialog box tabs, please refer to the *"Standard dialog box tabs"* section of this user guide.

For further information on notes, please refer to the *"Note class"* section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Note" dialog box

This screen (shown in Figure 2-2) is used to enter values for the note.

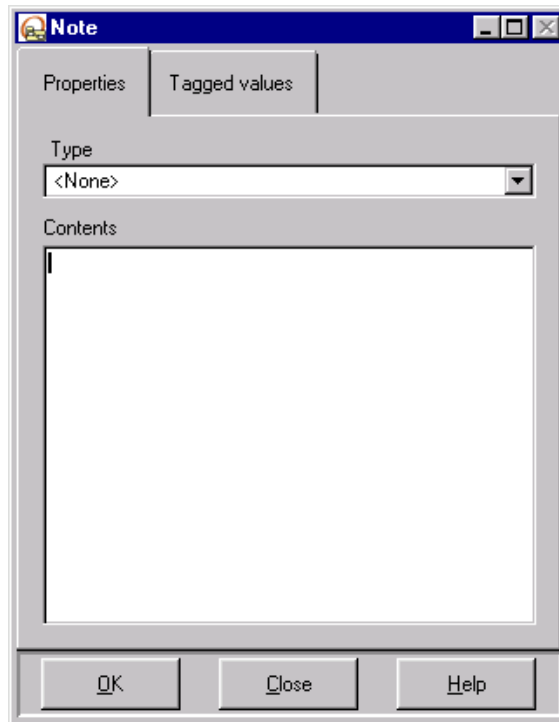


Figure 2-2. The "Properties" tab of the "Note" dialog box

Description of "Properties" tab fields

- ◆ "Type": This field allows the user to choose the note type from the scrolling list which appears, when the arrow is clicked on.
- ◆ "Contents": This field allows the user to enter the associated text.

Tag parameter dialog box

Entering tag parameters

A tag parameter is a parameter added to a tagged value, used to impose certain criteria on the tagged value in question. The selected tagged value definition imposes the existence and number of parameters.

For further information on tag parameters, please refer to the "*TagParameter class*" section of the *Objecteering/Metamodel User Guide*.

The "Tag parameter" dialog box

This screen (shown in Figure 2-3) is used to enter values for a tag parameter.

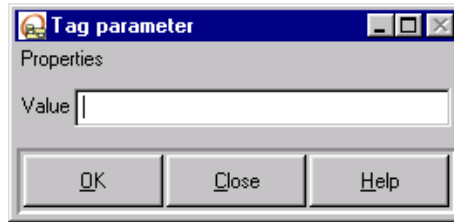



Figure 2-3. Entering properties in the "Tag parameter" dialog box

Description of fields in the "Tag parameter" dialog box

- ◆ "Value": This field allows the user to enter the parameter value for the tag parameter.

Tagged value dialog box

Entering tagged values

 A tagged value allows the user to refine the semantics of existing classes, through the addition of new attributes defined in a use case.

Tagged values are typed (see "Tag Type" dialog box) and can have values (see "Tag Parameter" dialog box).

A tagged value is expressed as follows: "{tagged value_name:Qualifier(p1,p2,p3)}".

The "Tagged value" dialog box contains no standard tabs, but instead contains three zones - "Definition", "Parameter" and "Qualifier".

For further information on tagged values, please refer to the "TaggedValue class" section of the *Objectteering/Metamodel User Guide*.

Tagged value dialog box

This screen (shown in Figure 2-4) is used to enter the values of a tagged value.

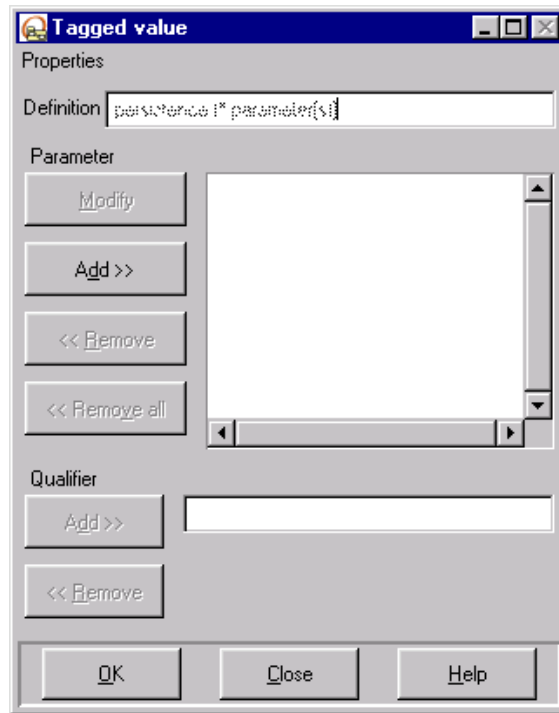


Figure 2-4. The "Tagged value" dialog box

Description of fields in the "Tagged value" dialog box

- ◆ **"Definition"**: Tagged values are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
 - ◆ **"Parameter"**: This field is used to add, remove or modify parameters which will be added to the tagged value. When you wish to add or modify a parameter, the *"Tag parameter"* dialog box is displayed (please refer to the *"Tag parameter dialog box"* section of this user guide. The buttons in this dialog box are used as follows:
 - ◆ **"Modify"**: This button is used to modify an existing parameter.
 - ◆ **"Add"**: This button is used to add a new parameter. When you click on this button, the *"Tag parameter"* button is displayed, and the value for the parameter which is to be added may be entered.
 - ◆ **"Remove"**: This button is used to remove a parameter from the tagged value in question.
 - ◆ **"Remove all"**: This button is used to remove all parameters from the tagged value in question.
 - ◆ **"Qualifier"**: A tagged value qualifier is a specific parameter. A maximum of one may exist for each tagged value. The selected tagged value definition either enforces or does not enforce the existence of qualifiers. Qualifiers are used to qualify the information entered for the tagged value in question.
 - ◆ **"Add"**: This is used to add a qualifier to the tagged value in question. When you click on this button, the *"Tag parameter"* dialog box is displayed, and the value for the qualifier may be entered.
 - ◆ **"Remove"**: This button is used to remove a qualifier from the tagged value in question.
-

Chapter 3: Static model dialog boxes

Package dialog box

Entering packages



A package is a general purpose mechanism used to organize elements into groups of model elements and diagrams. Packages may contain other packages.

The "Package" dialog box contains three tabs - "Properties", "Notes" and "Tagged values". For information on these standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on packages, please refer to the "Package class" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Package" dialog box

This screen (shown in figure 3-1) is used to enter values for a package.

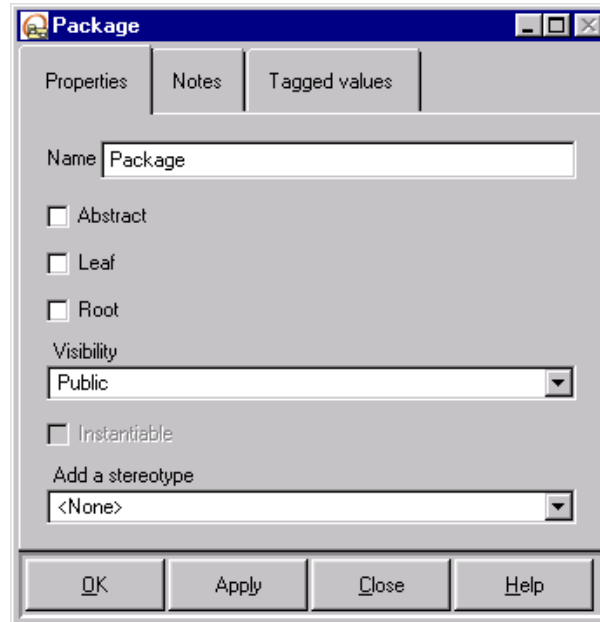


Figure 3-1. The "Properties" tab of the "Package" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: The package's name. This name must be unique in the model.
 - ◆ *"Abstract"*: This indicates if the package is abstract (that is, which cannot be directly instantiated) or not.
 - ◆ *"Leaf"*: This indicates if the package is a leaf package (a generalizable package with no children in the generalization hierarchy) or not.
 - ◆ *"Root"*: This indicates whether the package is a root package (with no ancestors) or not.
 - ◆ *"Visibility"*: This indicates the visibility of the package (public, protected, private or none).
 - ◆ *"Instantiable"*: This indicates whether or not the package may be instantiated.
 - ◆ *"Add a stereotype"* : This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
-

Class and interface dialog box

Entering classes



A class represents a pattern for an object's creation.(see also the "*Class*" metaclass).

A class has an extended description in a model. For example, enumerates are classes in the general sense, but are not developed in a model. A class represents its instances, and has operations and attributes.

Classes can have dependency links between them, and also support generalization links and associations.



Interfaces are created through the same dialog box, with the "*interface*" stereotype.

A class can be:

- ◆ "*abstract*": This means that it cannot have direct *instances*.
- ◆ "*leaf*": This means that it cannot be re-decomposed.
- ◆ "*root*": This means that it specializes no other classes, except "*Object*".
- ◆ "*primitive*": This means that its value cannot be decomposed. It can then be used as a type of attribute.
- ◆ "*main*": This means that it represents the application.
- ◆ "*active*": This means that the class can receive events, be multithreaded, or have specific active behavior.
- ◆ "*interface*": This means that the class is an interface. It will be defined by a set of services, and can be implemented by classes. This feature is frequently used for targets such as *Java*, *Corba*, etc.
- ◆ "*public*" or "*private*": This defines the visibility of this class from outside the package.

The "*Class*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on classes, please refer to the "*Class class*" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Class" dialog box

This screen (shown in Figure 3-3) is used to enter values for a class.

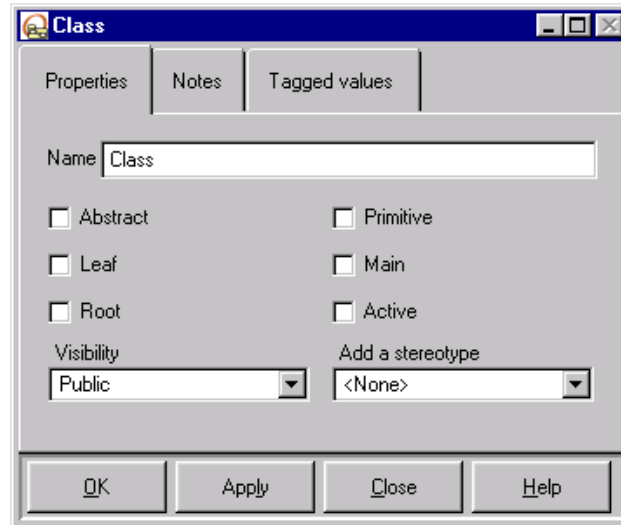



Figure 3-2. The "Properties" tab in the "Class" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This indicates the name of the class.
 - ◆ *"Abstract"*: An abstract class is defined very generally, and does not own direct instances.
 - ◆ *"Primitive"*: This determines if the class is primitive. A class is primitive if its value is not de-composable, and if its instances are not managed by the application. For example, *"integer"* and *"boolean"* are primitive classes, whereas *"Human"* or *"peripheral"* generally are not.
 - ◆ *"Leaf"*: This indicates if the class is a leaf class (a generalizable class with no children in the generalization hierarchy) or not.
 - ◆ *"Main"*: A main class is a class whose unique instance represents the application.
 - ◆ *"Root"*: This indicates if the class is a root class (with no ancestors) or not.
 - ◆ *"Active"*: This indicates if the class is an active class (a class whose instances are active objects) or not.
 - ◆ *"Visibility"*: This only applies if a class belongs to a leaf package. The visibility can either be public or protected. A public class is accessible from any user package of the current package. A private class can only be accessed from the current package or by an heir package.
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to *the Objecteering/UML Profile Builder* user guide).
-

Data type dialog box

Entering data types

 A data type is a descriptor of a set of primitive values which lacks identity, and which allows the user to define base types. Data types include numbers, strings, and enumerated values. They are passed by value and are immutable entities.

The "*Data type*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on data types, please refer to the "*Data Type class*" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Data type" dialog box

This screen (shown in Figure 3-3) is used to enter values for a data type.

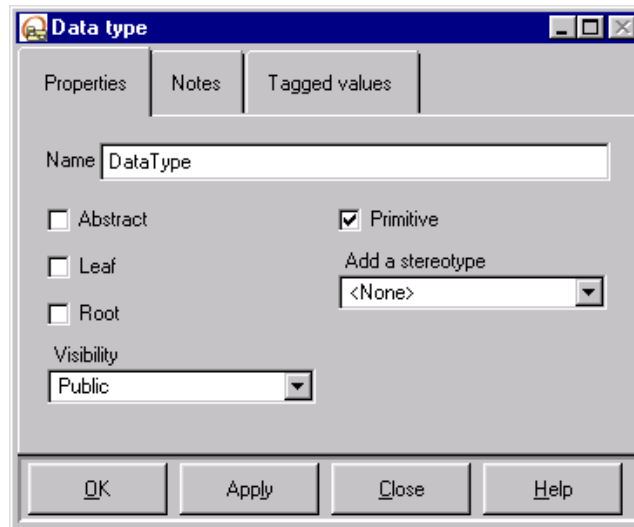


Figure 3-3. The "Properties" tab of the "Data type" dialog box

Description of "Properties" tab fields

- ◆ "Name": This indicates the name of the data type.
 - ◆ "Abstract": This indicates whether the data type is abstract or not.
 - ◆ "Primitive": This indicates whether the data type is primitive or not (this property should always be true).
 - ◆ "Leaf": This indicates whether the data type is leaf (a generalizable data type with no children in the generalization hierarchy) or not.
 - ◆ "Root": This indicates if the data type is a root data type (with no ancestors) or not.
 - ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to *the Objecteering/UML Profile Builder* user guide).
 - ◆ "Visibility": This indicates the visibility of the data type (public, protected, private or none).
-

Attribute dialog box

Entering attributes



An attribute specifies a primitive property of a class.

Its value is shared by all the instances of the class. It is characterized by a name, a type, and, optionally, by a default value.

An attribute can be:

- ◆ a class attribute, in which case it is related to the class itself.
- ◆ an instance attribute, in which case it belongs to each of the class instances. Its value is particular to a given instance.
- ◆ a functional dependency attribute, in which case its value depends on other values (for example other attributes). It is dynamically evaluated.

The "Attribute" dialog box contains three tabs - "Properties", "Notes" and "Tagged values". For information on these standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on attributes, please refer to the "Attribute class" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Attribute" dialog box

This screen (shown in Figure 3-4) is used to enter values for an attribute.

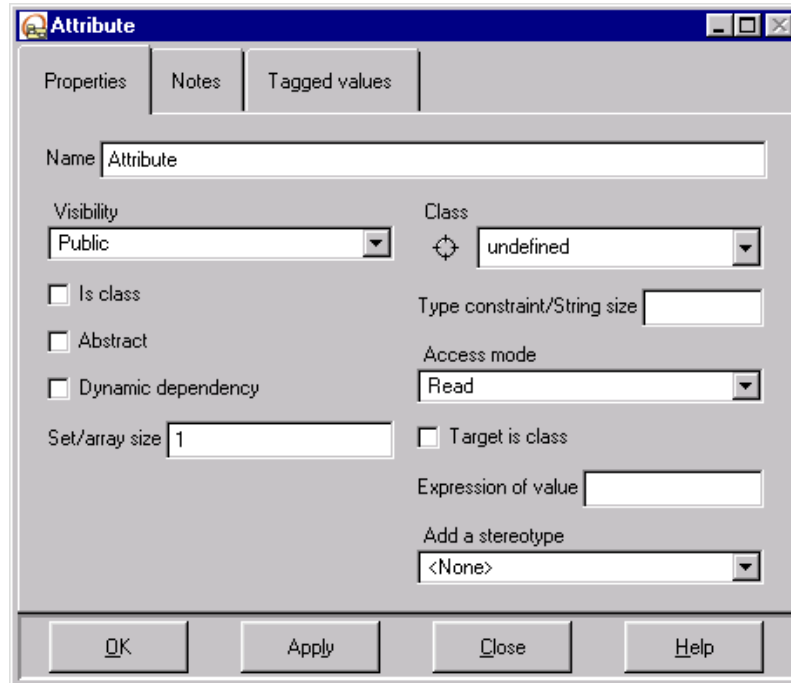


Figure 3-4. The "Properties" tab of the "Attribute" dialog box

Description of "Properties" tab fields

- ◆ "Name": This is the attribute's name.
 - ◆ "Visibility": This is the visibility of the attribute (*undefined*, *public*, *protected* or *private*).
 - ◆ "Class": This determines which class will give a type to the attribute. A list selects those classes that are "primitive" and that are accessible by the current class.
 - ◆ "Is class": This determines whether the attribute is static or not.
 - ◆ "Type constraint/String size": This is used to indicate the size of string-type attribute, and gives an indication of instantiation of the attribute's primitive class.
 - ◆ "Abstract": This indicates whether the attribute is abstract or not.
 - ◆ "Access mode": This determines the access mode of the attribute (read, write, read/write or neither).
 - ◆ "Dynamic dependency": This determines whether the attribute is a dynamic dependency, i.e. whether its value is calculated dynamically through an expression. This also corresponds to the "derived" attributes.
 - ◆ "Set/array size": This is used to indicate the size of the set. The attribute is not a set if the value is 1. Otherwise, it is a set with the indicated size (* for the sets with unlimited size).
 - ◆ "Target is class": This indicates that the attribute type is a *metaclass*.
 - ◆ "Expression of value": If the attribute is a dynamic dependency, then this field will contain the expression of the dynamic calculation in the target language (such as C++ or Java). If not, the associated field contains the attribute's default value.
 - ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to *the Objecteering/UML Profile Builder* user guide).
-

Operation dialog box

Entering operations



or (redefinition of an operation). This is the definition of a service provided by a class and its descendent classes. It characterizes the messages that its class instances can deal with.

The "Operation" dialog box contains two standard tabs - "Properties" and "Tagged values" - and one additional tab, "Implementation". For information on the two standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on operations, please refer to the "Operation class" section of the *Objectteering/Metamodel User Guide*.



Note: The "Redefine an operation" icon is used to redefine operations of parent classes, without re-entering its properties.

The "Properties" tab of the "Operation" dialog box

This screen (shown in Figure 3-5) is used to enter values for an operation.

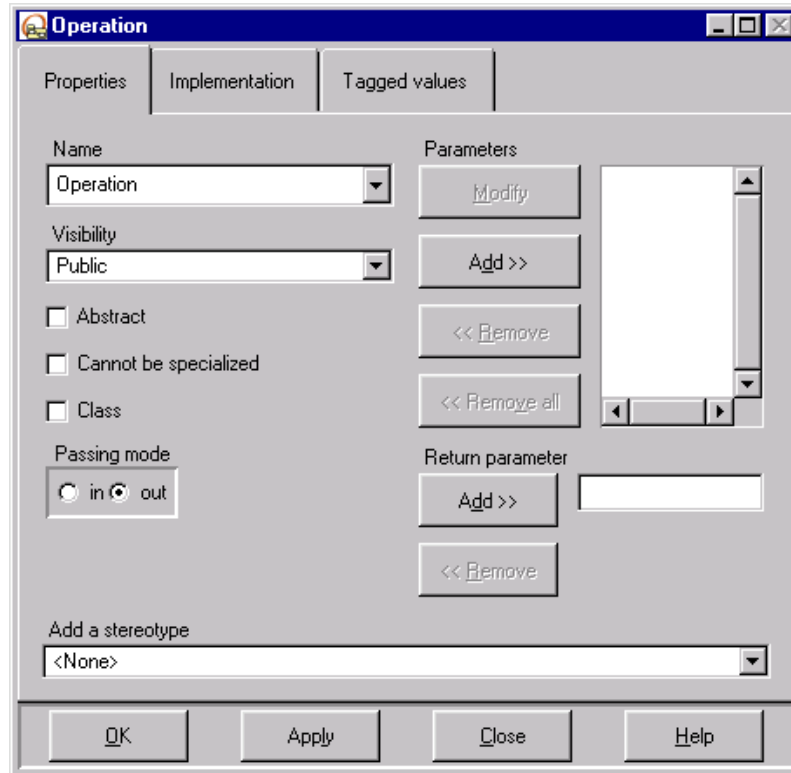


Figure 3-5. The "Properties" tab in the "Operation" dialog box

Description of "Properties" tab fields

- ◆ **"Name"**: This is the name of the operation.
- ◆ **"Visibility"**: This is the visibility of the member (*none*, *public*, *protected* or *private*).
- ◆ **"Abstract"**: This determines abstract methods.
- ◆ **"Cannot be specialized"**: This indicates that the operation may not be specialized, that is to say made more specific. It is a leaf operation.
- ◆ **"Class"**: This defines a *"class"* operation, i.e. shared by all its instances.
- ◆ **"Passing mode"**: This is the operation's passing mode (*in* or *out*). The default is *"out"*. This mode indicates whether the object receiving the message is modified (*out*), or not (*in*) by the operation's execution.
- ◆ **"Parameters"** (See *Parameter*): This defines the parameters that the operation receives.
- ◆ **"Return parameter"** (See *Return parameter*): This defines the possible return parameter.
- ◆ **"Add a stereotype"**: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling level (for further information, please refer to *the Objecteering/UML Profile Builder* user guide).

The "Implementation" tab of the "Operation" dialog box

This screen (shown in Figure 3-6) allows the user to add, modify or remove the operation's code.

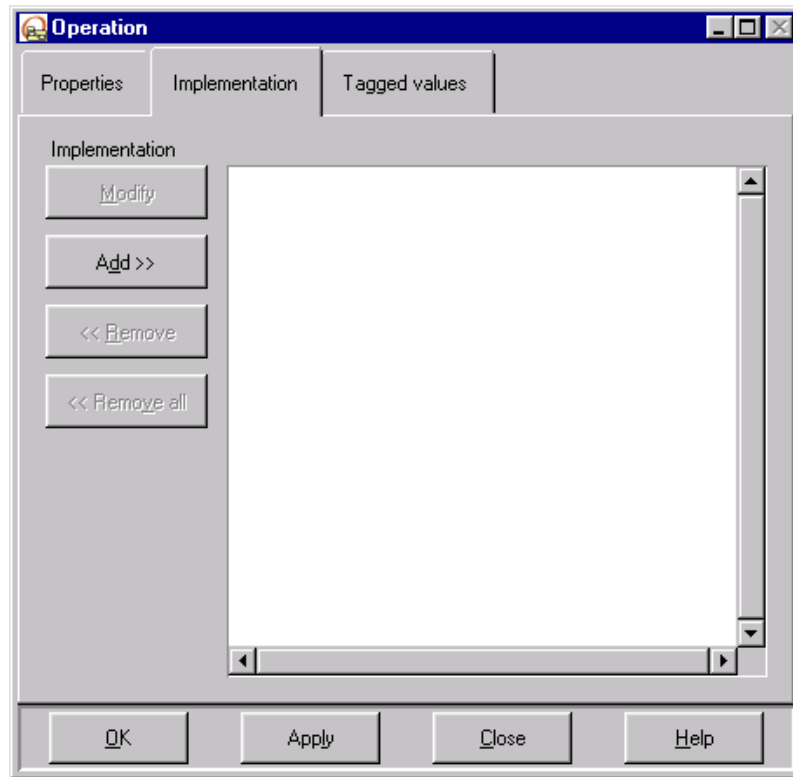



Figure 3-6. The "Implementation" tab of the "Operation" dialog box

Description of "Implementation" tab fields

- ◆ *"Implementation"*: This is text which represents the operation's content in the target language. It may be added, modified or removed by clicking on the relevant button.
-

Parameter dialog box

Entering parameters

 Parameters are information received by an operation, and may be entered either from the explorer, or from an operation's dialog box.

Parameters are defined for each operation. Their name, type and passing mode are essential information.

A parameter has:

- ◆ a passing mode (in, out, in/out)
- ◆ a compulsory or optional feature, indicating whether the caller must supply it or not

The "*Parameter*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on parameters, please refer to the "*Parameter class*" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Parameter" dialog box

This screen (shown in Figure 3-8) is used to enter values for a parameter.

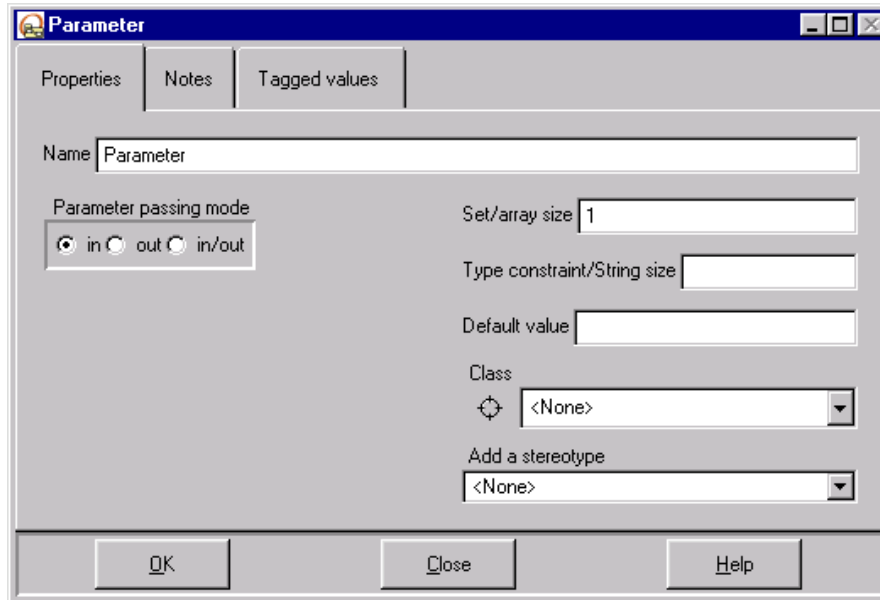


Figure 3-7. The "Properties" tab of the "Parameter" dialog box

Description of "Properties" tab fields

- ◆ "Name": This is the name of the element
 - ◆ "Parameter passing mode": This indicates whether the parameter is in "read only" mode (in) or may be modified (out or in/out).
 - ◆ "Set/array size": If the value is other than 1, the parameter is a set with the indicated size (* if unlimited, constant, or integer).
 - ◆ "Type constraint/String size": This is the constraint on the parameter type (for example, the size of the character string)
 - ◆ "Default value": This is the possible default value of the parameter.
 - ◆ "Class": This defines the class to which the parameter belongs.
 - ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to *the Objecteering/UML Profile Builder user guide*).
-

Return parameter dialog box

Entering return parameters

P» Return parameters are information sent back after an operation has been carried out. Return parameters can be entered either from the explorer, or from an operation's dialog box.

The "Return parameter" dialog box contains three tabs - "Properties", "Notes" and "Tagged values". For information on these standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on parameters, please refer to the "Parameter class" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Return parameter" dialog box

This screen (shown in Figure 3-8) is used to enter values for a parameter.

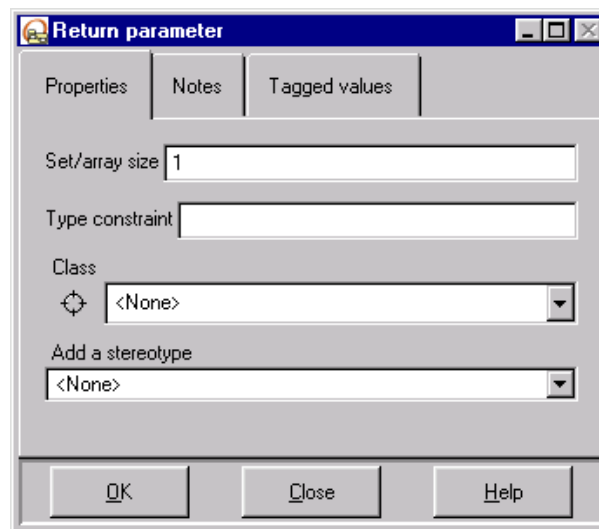



Figure 3-8. The "Properties" tab of the "Return parameter" dialog box

Description of "Properties" tab fields

- ◆ "Set/array size": If the value is other than 1, the parameter is a set with the indicated size (* if unlimited, constant, or integer).
 - ◆ "Type constraint": This is the constraint on the parameter's type (for example, size of the character string)
 - ◆ "Class": This defines the class to which the parameter belongs.
 - ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to *the Objecteering/UML Profile Builder* user guide).
-

Binary association dialog box

Entering binary associations

 An association (or aggregation) specifies a stable link between two or more instances or sets of instances, and can be binary or n-ary. Where a binary association is concerned, the association entry dialog box allows the entry of both the association and the association ends. For n-ary associations there exists a dialog box for each extremity, as well as for the association itself.

The "*Binary association*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*First link Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on associations, please refer to the "*Association class*" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Binary association" dialog box

This dialog box (shown in Figure 3-9) is used to enter values for a binary association.

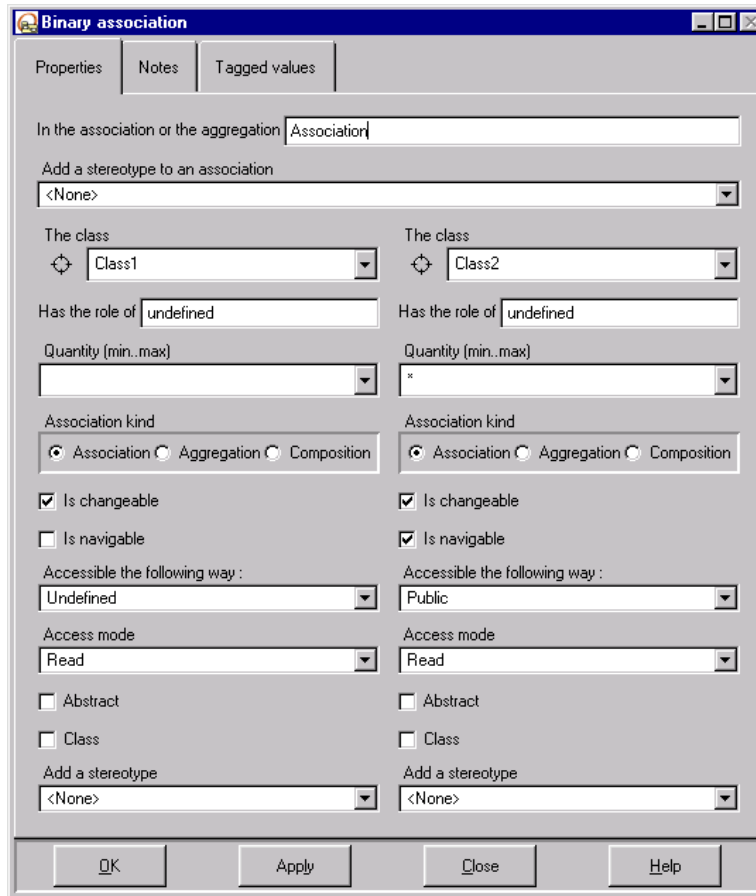


Figure 3-9. The "Properties" screen of the "Binary association" dialog box

Description of "Properties" tab fields

- ◆ *"In the association or the aggregation"*: This specifies the binary association name.
 - ◆ *"Add a stereotype to association"*: This field allows the user to add a stereotype to the association.
 - ◆ *"The class"*: This defines the names of the classes at both ends of the link.
 - ◆ *"Has the role of"*: This is the role played by the class with regard to the class at the other end of the link.
 - ◆ *"Quantity (min-max)"*: This is the interval multiplicity written "min*max". "min" gives the minimum number of instances of the class, and "max" indicates the maximum number of instances of the class. The "*" symbol means unlimited.
 - ◆ *"Association kind"*: This is used to define the kind of association concerned. "Association" is for standard associations, "Aggregation" represents "shared aggregations" (indicated by a white triangle), and "Composition" represents strong composition (graphically shown by a black triangle).
 - ◆ *"Is changeable"*: When placed on a target end, this specifies whether or not an instance of the association may be modified from the source end.
 - ◆ *"Is navigable"*: This specifies whether or not the association (which must be binary) can be traversed from the opposite class to the class attached to the association end in question.
 - ◆ *"Accessible the following way"*: This is the visibility of the member (public, protected or private). Choosing visibility orientates the relation between the current class and the class at the other end (navigability).
 - ◆ *"Access mode"*: This determines the access mode of the association (read, write, read/write or no access).
 - ◆ *"Abstract"*: This determines if the role is abstract or not.
 - ◆ *"Class"*: This determines if the role is class or not.
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to *the Objectteering/UML Profile Builder* user guide).
-

N-ary association dialog box

The "Properties" tab of the "N-ary association" dialog box



This screen (shown in Figure 3-10) is used to enter values for an n-ary association or aggregation.

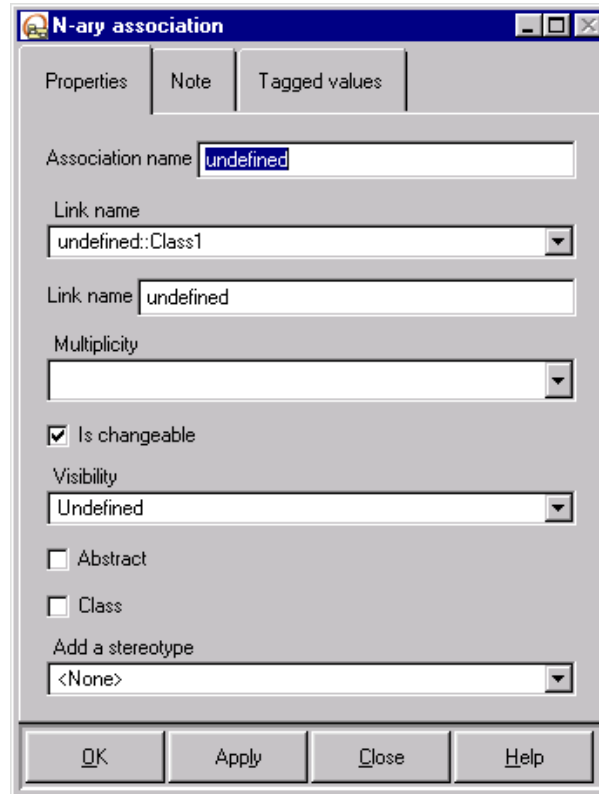



Figure 3-10. The "Properties" screen of the "N-ary association" dialog box

Description of "Properties" tab fields

- ◆ "*Association name*": This is the name of the n-ary association or aggregation.
 - ◆ "*Link name*": This is the name given to the link.
 - ◆ "*Multiplicity*": This indicates the multiplicity of the n-ary association.
 - ◆ "*Is changeable*": When placed on a target end, this specifies whether or not an instance of the association may be modified from the source end.
 - ◆ "*Visibility*": This indicates the visibility of the package (public, protected, private or none).
 - ◆ "*Abstract*": This determines if the role is abstract or not.
 - ◆ "*Class*": This determines if the role is class or not.
 - ◆ "*Add a stereotype*": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to *the Objecteering/UML Profile Builder* user guide).
-

Class association dialog box

Entering class associations

 This is a class which relates other classes by adding features to associations, and is both a class and an association. A class association is a component of an association.

The "Class association" dialog box contains three tabs - "Properties", "Notes" and "Tagged values". For information on these standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on class associations, please refer to the "ClassAssociation class" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Class association" dialog box

This screen (shown in Figure 3-11) is used to enter values for a class association.



Figure 3-11. The "Properties" tab of the "Class association" dialog box

Description of "Properties" tab fields

- ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to *the Objecteering/UML Profile Builder* user guide).
-

Enumeration dialog box

Entering enumerations



An enumeration defines a finite sub-set of positive integers, in which each value has a symbolic name.

The enumerate types represented here can correspond either to C++ enumerate types or to Pascal or Ada enumerate types (which have different semantics). All the possible symbolic values of the enumerations are defined by a "*Literal value*" class.

The "*Enumeration*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on enumerations, please refer to the "*Enumeration class*" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Enumeration" dialog box

This screen (shown in Figure 3-12) is used to enter values for an enumeration.

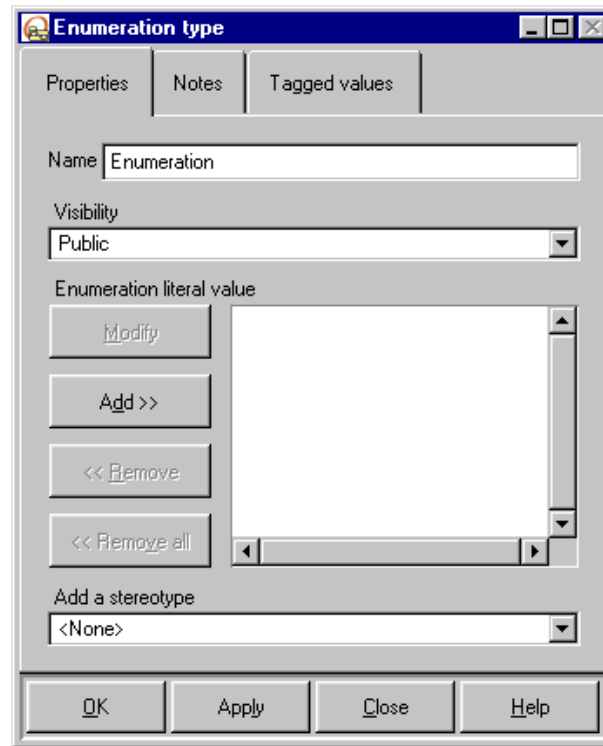



Figure 3-12. The "Properties" tab of the "Enumeration" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This is the name of the enumeration.
 - ◆ *"Visibility"*: This indicates the visibility of the enumeration, which can be public, protected, private or none.
 - ◆ *"Enumeration literal value"*: (see the *"Enumeration literal"* referenced element) This is the link with the "Enumeration literal" representing the possible values of the type.
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Enumeration literal dialog box

Entering enumeration literals

 An enumeration literal is a symbolic value of an enumeration type element. For further information, please refer to the "*EnumerationLiteral class*" section of the *Objectteering/Metamodel User Guide*.

An enumeration type is defined as an integer type sub-set, composed of several possible values and defined with a symbolic name. The enumeration literal is this symbolic name.

The "Enumeration literal" dialog box

This screen (shown in Figure 3-13) is used to enter values for an enumeration literal.



Figure 3-13. Entering an enumeration literal

Description of fields

- ◆ "*Literal name*": This is the name of the enumeration literal, and represents a symbolic value of the associated enumeration.
-

Signal dialog box

Entering signals



Signals are specifications of asynchronous stimuli communicated between instances. Events and Data flows are representations (occurrences) of signals. By extension, signals are a representation of any kind of information that can travel between packages, classes, instances or messages. This information can be an object or a request.

The "Signal" dialog box contains three tabs - "Properties", "Notes" and "Tagged values". For information on these standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on signals, please refer to the "Signal class" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Signal" dialog box

This screen (shown in Figure 3-14) is used to enter values for a signal.

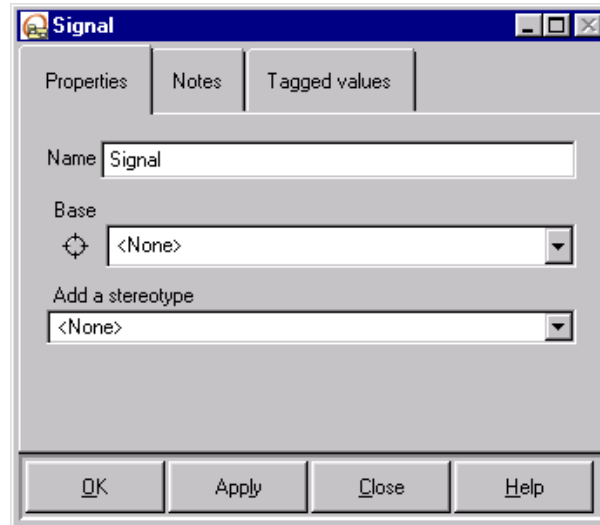


Figure 3-14. The "Properties" screen of the "Signal" dialog box

Description of "Properties" tab fields

- ◆ "Name": This is the name of the signal.
- ◆ "Base": This field is used to show represented information. The user can choose from the combo box. A signal can be a representation of a parameter, passing, or message call, or of any class instance navigation.
- ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).

Data flow dialog box

Entering data flows



A data flow is a circulation of information between model elements, the representation of all types of information that can be transmitted between elements. Data flows can be objects or requests.

The "*Data flow*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on data flows, please refer to the "*DataFlow class*" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Data flow" dialog box

This screen (shown in figure 3-15) is used to enter values for a data flow.

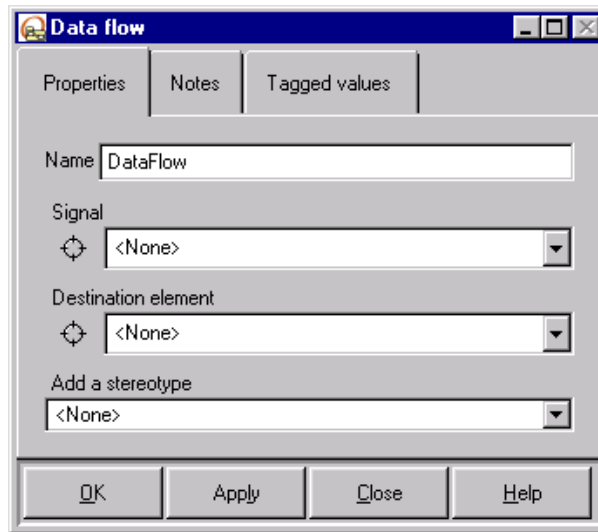


Figure 3-15. The "Properties" tab of the "Data flow" dialog box

Description of "Properties" tab fields

- ◆ "Name": This is the data flow's name. This name must be unique in the model.
- ◆ "Signal": This combobox allows the user to choose a signal in the UML modeling project. The dataflow will indicate that the signal may circulate between its origin and destination.
- ◆ "Destination element": This combobox indicates the destination of the data flow.
- ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to *the Objecteering/UML Profile Builder* user guide).

Generalization dialog box

Entering generalizations



A generalization is a generalization link between classes, which represents the hierarchy of packages or classes (see also the "*Generalization*" metaclass).

A generalization link can have tagged values.

The "*Generalization*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on generalizations, please refer to the "*Generalization class*" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Generalization" dialog box

This screen (shown in Figure 3-16) is used to enter values for a generalization.

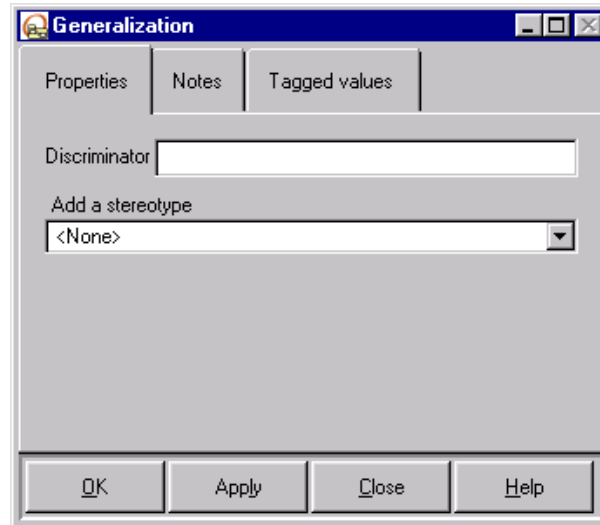


Figure 3-16. The "Properties" tab of the "Generalization" dialog box

Description of "Properties" tab fields

- ◆ "*Discriminator*": This field designates the partition to which the generalization link belongs.
 - ◆ "*Add a stereotype*": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to *the Objecteering/UML Profile Builder* user guide).
-

Use dialog box

Entering uses



A use is a usage dependency between model elements; in other words, one element requires the presence of another element for its correct functioning or implementation.

The "Use" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on uses, please refer to the "*Use class*" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Use" dialog box

This screen (shown in Figure 3-17) is used to enter values for a use.



Figure 3-17. The "Properties" screen of the "Use" dialog box

Description of "Properties" tab fields

- ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
-

Realization dialog box

Entering realizations



A realization is an implementation link between a class and its interface, or between a component and its interface.

The "*Realization*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on realizations, please refer to the "*Realization class*" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Realization" dialog box

This screen (shown in figure 3-18) is used to enter values for a realization.




Figure 3-18. The "Properties" tab of the "Realization" dialog box

"Properties" Tab: Description

- ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
-

Template parameter dialog box

Entering template parameters

 A template parameter is a parameter for templated elements. Typically, parameters are classifiers which represent attribute types, but they can also represent integers or even operations.

For further information on template parameters, please refer to the "*TemplateParameter class*" section of the *Objecteering/Metamodel User Guide*.

The "Template parameter" dialog box

This screen (shown in Figure 3-19) is used to enter values for a template parameter.

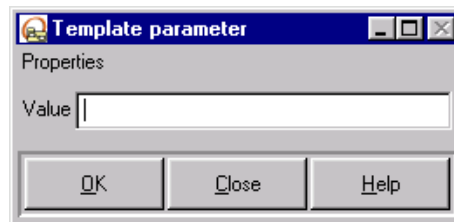


Figure 3-19. The "Template parameter" dialog box

Description of fields

- ◆ "Value": This is the value assigned to the template parameter.
-

Chapter 4: Use case model dialog
boxes

Use case dialog box

Entering use cases



A use case represents a system's functions. One or more sequence diagrams show how these functions or activities are carried out by external users called actors and objects of the system.

The relationships between uses and actors are described in use case diagrams.

The "Use case" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on use cases, please refer to the "*UseCase class*" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Use case" dialog box

This screen (shown in Figure 4-1) is used to enter the values of a use case.

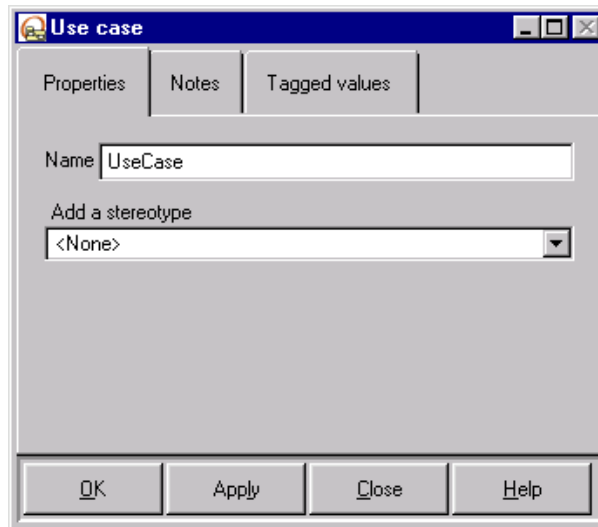


Figure 4-1. The "Properties" tab of the "Use case" dialog box

Description of "Properties" tab fields

- ◆ "Name": This is the use case's name. This name must be unique in the model.
 - ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
-

Actor dialog box

Entering actors



An actor is defined in a package.

An actor symbolizes a user and his relationship to an application model. It will be used by use case diagrams and sequence diagrams and represents the modeled system's external participants.

The "Actor" dialog box contains three tabs - "Properties", "Notes" and "Tagged values". For information on these standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on actors, please refer to the "Actor class" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Actor" dialog box

This screen (shown in figure 4-2) is used to enter values for an actor.



Figure 4-2. The "Properties" tab of the "Actor" dialog box

Description of "Properties" tab fields

- ◆ "Name": The actor's name. This name must be unique in the model.
 - ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Communication link dialog box

Entering communication links



A communication link represents the interaction that a user external to the system can have with the system in specific use modes. Communication links are defined in a use case.

The "*Communication link*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on communication links, please refer to the "*Communication class*" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Communication" dialog box

This screen (shown in Figure 4-3) is used to enter values for the communication link.

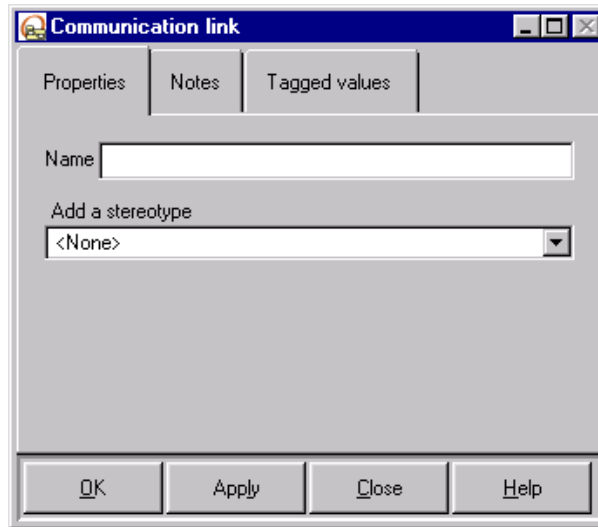


Figure 4-3 The "Properties" tab of the "Communication link" dialog box

Description of "Properties" tab fields

- ◆ "Name": The communication link's name.
 - ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
-

Use case dependency dialog box

Entering use case dependencies



A use case dependency is a use link between two use cases. UML defines two use types between use cases. These are presented in the form of a stereotype («extend» and «include») attached to the use link.

The "Use case dependency" dialog box contains three tabs - "Properties", "Notes" and "Tagged values". For information on these standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on use case dependencies, please refer to the "UseCaseDependency class" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Use case dependency" dialog box

This screen (shown in Figure 4-4) is used to enter values for a use case dependency.

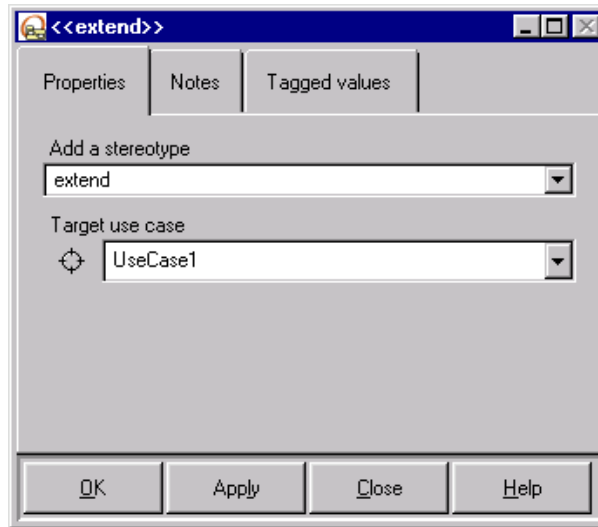


Figure 4-4. The "Properties" tab of the "Use Case dependency" dialog box

Description of "Properties" tab fields

- ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
 - ◆ "Target use case": In a dependency between use cases, this combobox is used to define the link to the target use case.
-

Chapter 5: State machine model dialog
boxes

State machine dialog box

Entering state machines



A state machine is a graph of states and transitions which describes the dynamic behavior of objects, that is to say, the sequence of states that an object or an interaction goes through in response to events during its life, together with its responsive actions.

In Objectteering , a state machine belongs to a package, an operation, a use case or a class. Its natural position is to belong to a class.

The "*State machine*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on state machines, please refer to the "*StateMachine class*" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "State machine" dialog box

This screen (shown in Figure 5-1) is used to enter the values for a state machine.

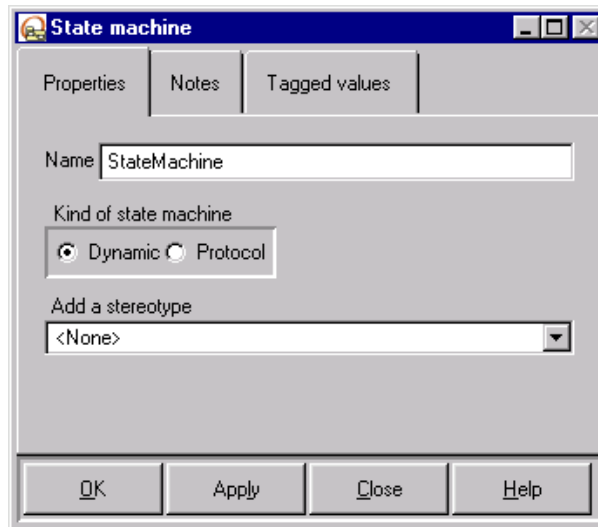


Figure 5-1. The "Properties" tab of the "State machine" dialog box

Description of "Properties" tab fields

- ◆ "Name": This is the name of the state machine.
 - ◆ "Kind of state machine": Here, the user checks either the "Dynamic" button to indicate that the state machine is dynamic, or the "Protocol" button to indicate that the state machine is protocol.
 - ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

State dialog box

Entering states



A state represents either a period of time during which an object waits for certain events to occur, or a period of time during which an object performs an ongoing activity. States are interconnected by transitions.

The "State" dialog box contains three tabs - "Properties", "Notes" and "Tagged values". For information on these standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on states, please refer to the "State class" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "State" dialog box

This screen (shown in Figure 5-2) is used to enter values for a state.



Figure 5-2. The "Properties" tab of the "State" dialog box

Description of "Properties" tab fields

- ◆ "Name": This is the name of the state.
 - ◆ "Concurrent": This indicates whether or not the state is concurrent.
 - ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Pseudo state dialog box

Entering pseudo states

A pseudo state is an abstraction of different types of nodes in the state machine graph.

Pseudo states are used to link transition segments, and a transition to one implies a further automatic transition to another state, without an event being necessary.

The "*Pseudo state*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on pseudo states, please refer to the "*PseudoState class*" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Pseudo state" dialog box

This screen (shown in Figure 5-3) is used to enter values for a pseudo state.

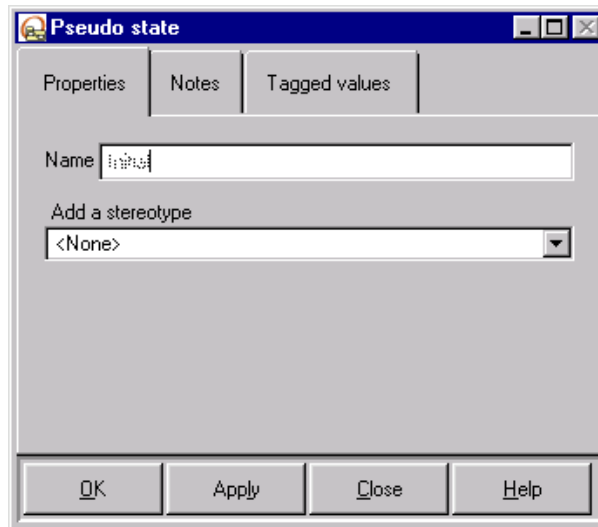


Figure 5-3. The "Properties" tab of the "Pseudo state" dialog box

Description of "Properties" tab fields

- ◆ "Name": This is the name of the pseudo state.
 - ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Transition dialog box

Entering transitions



A transition is a relationship between two states, indicating that an object in the first state will perform specified actions and enter the second state when a specified event occurs and specified guard conditions are satisfied.

The "*Transition*" dialog box contains two tabs - "*Properties*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on transitions, please refer to the "*Transition class*" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Transition" dialog box

This screen (shown in Figure 5-4) is used to enter values for a transition.

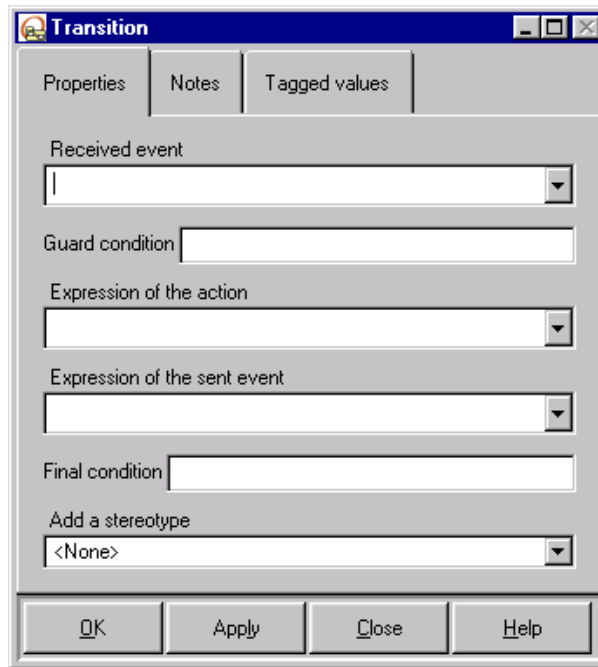



Figure 5-4. The "Properties" tab of the "Transition" dialog box

Description of "Properties" tab fields

- ◆ *"Received event"*: This is the event received which triggers the transition. The received event can be text entered in the field, or a reference to events defined in the current state machine.
 - ◆ *"Guard condition"*: This is the condition under which a transition may be triggered (for further information on conditions, please refer to the *"Condition class"* section of the *Objectteering/Metamodel User Guide*).
 - ◆ *"Expression of the action"*: This is an action realized when the transition is triggered. The combobox makes it possible to simply designate an operation (message call).
 - ◆ *"Expression of the sent event"*: This is an event sent by the transition once it has been triggered. A sent event can be text entered in the field, or a reference to existing events in the current state machine (combobox). Signals can also be referenced (shorthand for Signal sending event).
 - ◆ *"Final condition"*: This is a condition obtained once the transition has occurred (this can be useful for protocol state diagrams).
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
-

Internal transition dialog box

Entering internal transitions

 An internal transition is a transition which is internal to a state. It is related to a state. It may be triggered upon entering or exiting the state, or can describe an activity that is performed whilst in the state.

The "*Internal transition*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on internal transitions, please refer to the "*InternalTransition class*" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Internal transition" dialog box

This screen (shown in Figure 5-5) is used to enter values for an internal transition.

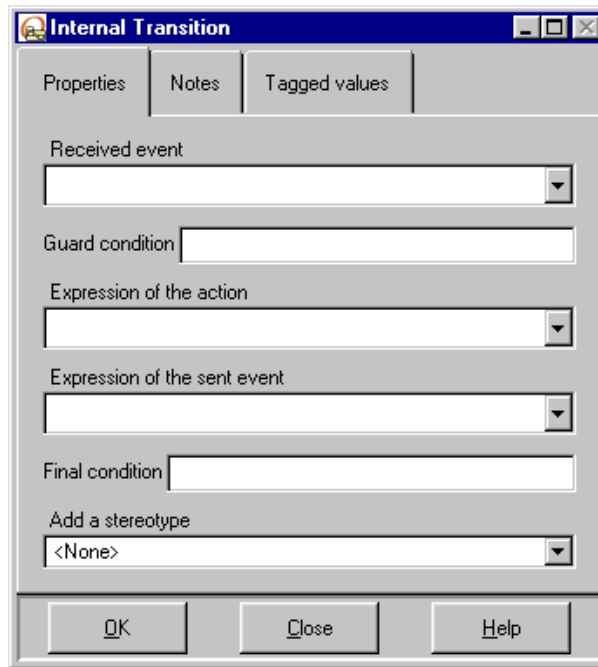


Figure 5-5. The "Properties" screen of the "Internal transition" dialog box

Description of "Properties" tab fields

- ◆ *"Received event"*: This is the received event which triggers the internal transition.
 - ◆ *"Guard condition"*: This is the condition under which an internal transition is triggered.
 - ◆ *"Expression of the action"*: This is an action realized when the transition is triggered. The combobox makes it possible to simply designate an operation (message call).
 - ◆ *"Expression of the sent event"*: This is an event sent by the internal transition once it has been triggered.
 - ◆ *"Final condition"*: This is a condition obtained once the transition has occurred (this can be useful for protocol state diagrams).
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Event dialog box

Entering events



An event is a specification of a significant occurrence that has a location in time and space. An instance of an event can lead to the activation of a behavioral feature in an object.

An event can be either an occurrence of a signal, a message occurrence or a time or change expression occurrence.

The "Event" dialog box contains three tabs - "Properties", "Notes" and "Tagged values". For information on these standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on events, please refer to the "Event class" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Event" dialog box

This screen (shown in Figure 5-6) is used to enter values for an event.

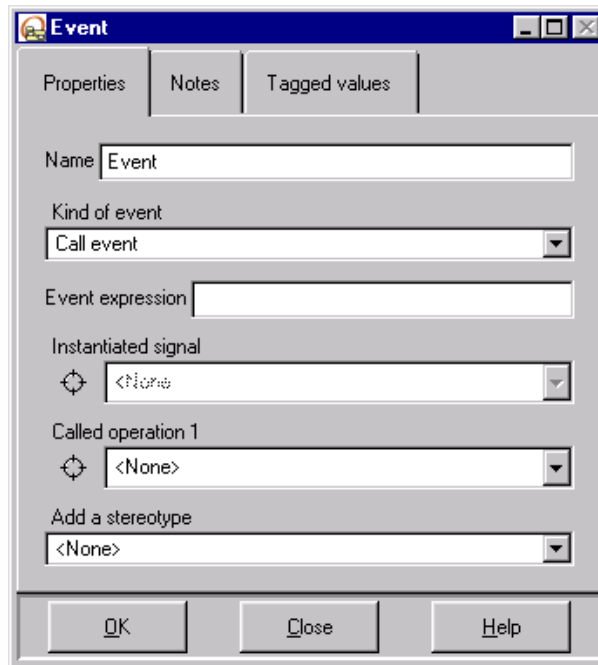


Figure 5-6. The "Properties" screen of the "Event" dialog box

Description of "Properties" tab fields

- ◆ "*Name*": This is the name of the event.
 - ◆ "*Kind of event*": This defines the nature of the event (Time, Signal occurrence, and so on).
 - ◆ "*Event expression*": This is an expression which initiates the event. It can be a time expression, or a triggering condition, and may contain parameter values in the case of operation call events, and so on.
 - ◆ "*Instantiated signal*": This is the signal from which the event is an occurrence.
 - ◆ "*Called operation 1*": This is a direct link to an operation in the case of a call event.
 - ◆ "*Add a stereotype*": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
-

Chapter 6: Activity model dialog boxes

Activity graph dialog box

Entering activity graphs



An activity graph is a special case of a state machine in which all or most of the states are activity states or action states, and in which all or most of the transitions are triggered by completion of an activity in the source states.

An activity graph emphasizes the sequential and concurrent steps of a computational procedure.

The "Activity graph" dialog box contains three tabs - "Properties", "Notes" and "Tagged values". For information on these standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on activity graphs, please refer to the "ActivityGraph class" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Activity graph" dialog box

This screen (shown in Figure 6-1) is used to enter and modify values for an activity graph.

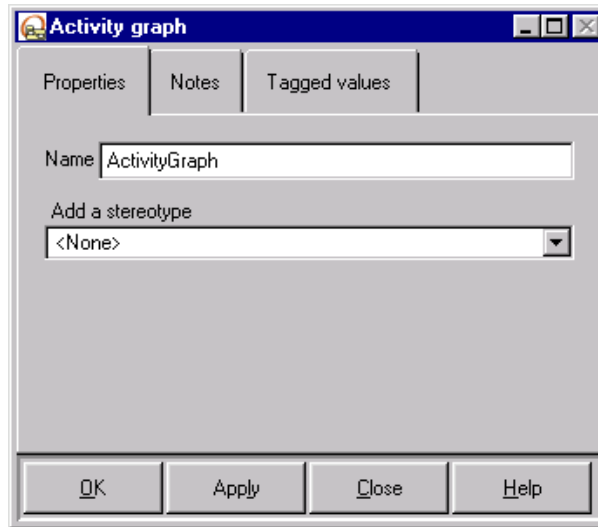


Figure 6-1. The "Properties" screen of the "Activity graph" dialog box

Description of "Properties" tab fields

- ◆ "Name" : This is the name of the activity graph.
 - ◆ "Add a stereotype" : This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Action state dialog box

Entering action states



An action state is a state which cannot be further broken down and which describes an action. The purpose of an action state is to execute an action and then transition to another state.

The "Action state" dialog box contains three tabs - "Properties", "Notes" and "Tagged values". For information on these standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on action states, please refer to the "ActionState class" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Action state" dialog box

This screen (shown in Figure 6-2) is used to enter and modify values for an action state.

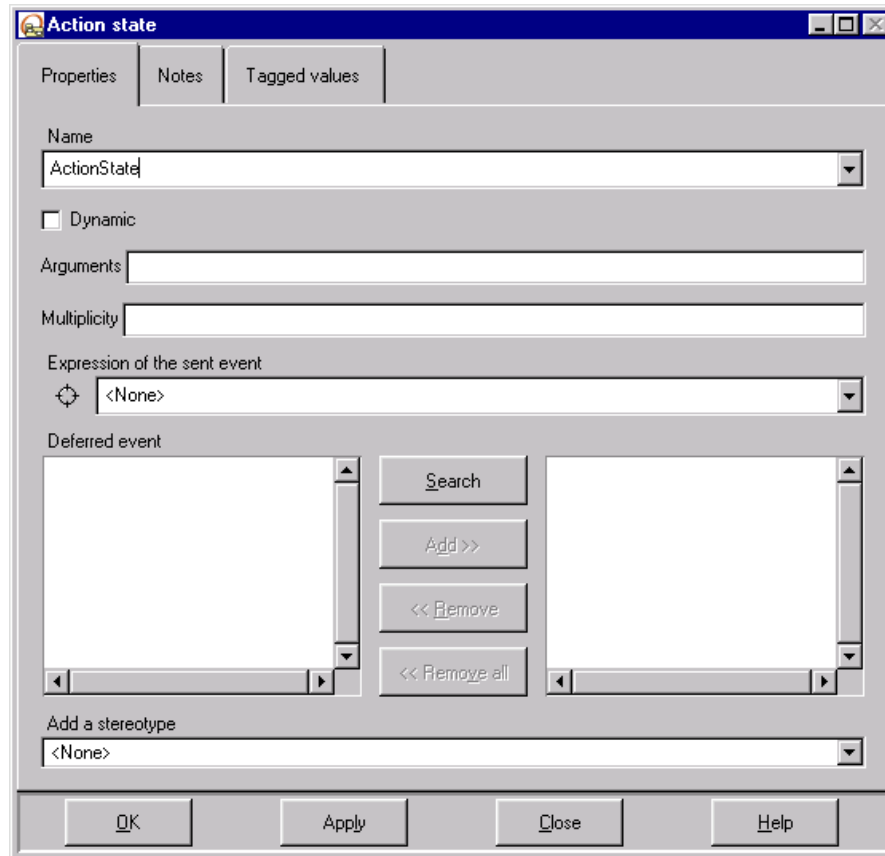


Figure 6-2. The "Properties" screen of the "Action state" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This is the name of the action state.
 - ◆ *"Dynamic"*: This specifies whether or not the action state's actions can be executed concurrently.
 - ◆ *"Arguments"*: This determines at runtime the number of parallel executions of the action state's actions. The value must be a set of lists of objects, each list serving as an argument for one execution.
 - ◆ *"Multiplicity"*: This indicates the multiplicity of the action state.
 - ◆ *"Expression of the sent event"*: This is linked to the associated transition. The assisted data entry function suggests, if the case arises, the names of the operations belonging to the current class (which contains the activity diagram). If a partition representing a classifier is associated, Objectteering/UML proposes the names of the methods belonging to the classifier.
 - ◆ *"Deferred event"*: This indicates those events not handled by the action, but postponed until later.
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
-

Sub activity state dialog box

Entering sub activity states



A sub activity state represents the execution of a non-atomic sequence of steps that has some duration (in other words, it consists internally of a set of actions and possibly waiting for events). A sub activity state can be split into another activity graph.

The "*Sub activity state*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on sub activity states, please refer to the "*SubActivityState class*" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Sub activity state" dialog box

This screen (shown in Figure 6-3) is used to enter and modify values for a sub activity state.

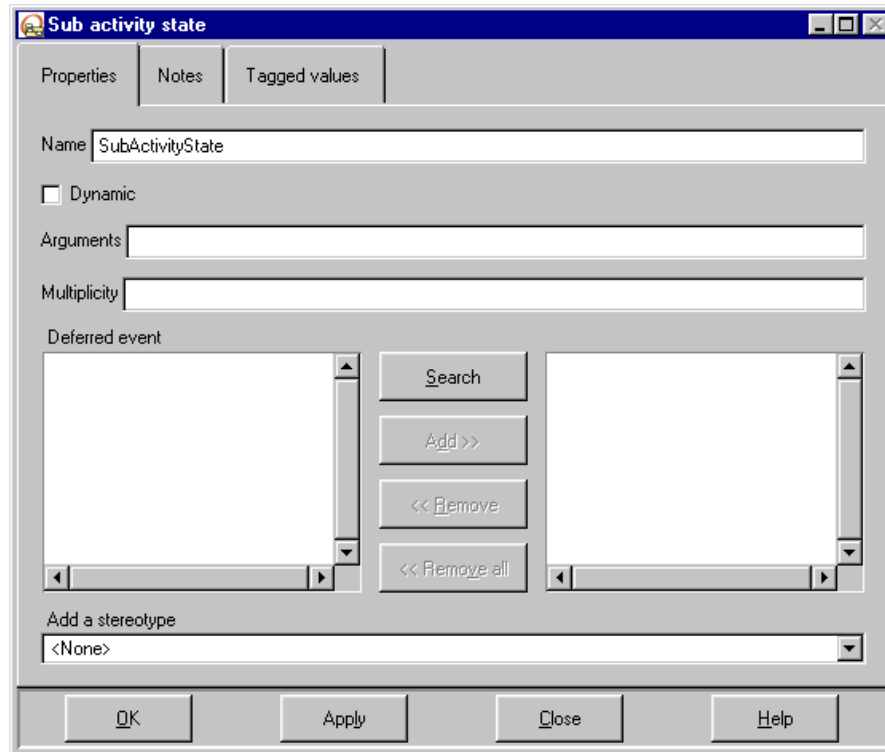


Figure 6-3. The "Properties" screen of the "Sub activity state" dialog box

Description of "Properties" tab fields

- ◆ "*Name*": This is the name of the sub activity state.
 - ◆ "*Dynamic*": This specifies whether or not the sub activity state's actions can be executed concurrently.
 - ◆ "*Arguments*": This determines at runtime the number of parallel executions of the sub activity state's actions. The value must be a set of lists of objects, each list serving as an argument for one execution.
 - ◆ "*Multiplicity*": This indicates the multiplicity of the sub activity state.
 - ◆ "*Deferred event*": This indicates those events not handled by the action, but postponed until later.
 - ◆ "*Add a stereotype*": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
-

Object flow state dialog box

Entering object flow states



An object flow state is a state which represents the existence of an object of a particular class at a specific point within a computation. It defines an object flow between actions in an activity graph.

The "*Object flow state*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on object flow states, please refer to the "*ObjectFlowState class*" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Object flow state" dialog box

This screen (shown in Figure 6-4) is used to enter and modify values for an object flow state.

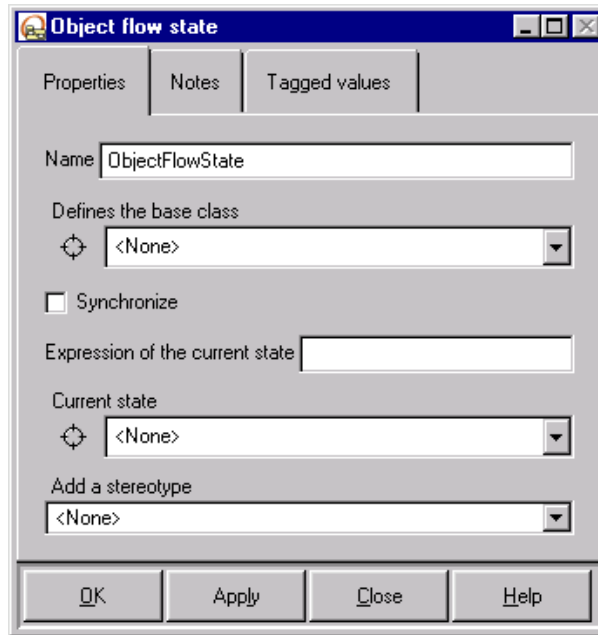


Figure 6-4. The "Properties" screen of the "Object flow state" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This is the name of the object flow state.
 - ◆ *"Defines the base class"*: This defines the base class on which the object flow state is based.
 - ◆ *"Synchronize"*: This indicates whether or not an object flow state is used as a synch state.
 - ◆ *"Expression of the current state"*: This is the value of the current state. If the *"ClassifierInState"* association is set, then this value has no meaning. By extension, this value can be a boolean expression or any text expressing a current situation. This is a more flexible way of representing *"ClassifierInState"* in an activity diagram.
 - ◆ *"Current state"*: This indicates the current state of the object flow state.
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
-

Partition dialog box

Entering partitions



A partition in an activity graph organizes responsibilities for activities. Partitions do not have a fixed meaning, but often correspond to organizational units in a business model.

The "*Partition*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on partitions, please refer to the "*Partition class*" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Partition" dialog box

This screen (shown in Figure 6-5) is used to enter and modify values for a partition.

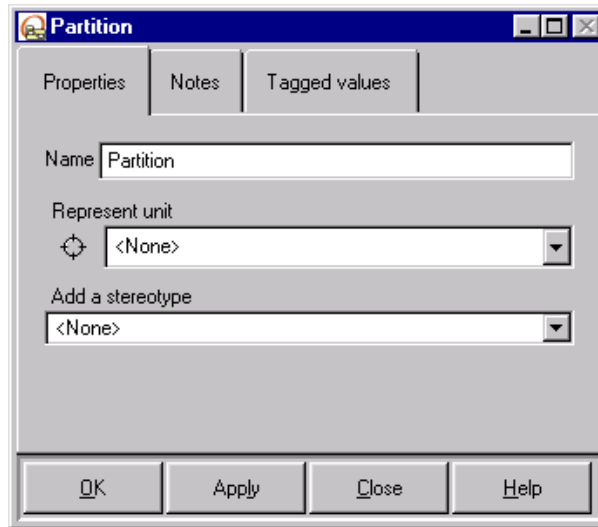


Figure 6-5. The "Properties" screen of the "Partition" dialog box

Description of "Properties" tab fields

- ◆ "Name": This is the name of the partition.
- ◆ "Represent unit": In Objectteering/UML, partitions can represent "NameSpaces". They very often represent "Classifiers" as active elements, or "Packages" as organizational units.
- ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).

Chapter 7: Physical model dialog
boxes

Node dialog box

Entering nodes



A node is a run-time physical object which represents a computational resource. Nodes generally have at least a memory and often processing capability as well. Associations between nodes represent communication paths.

The "Node" dialog box contains three tabs - "Properties", "Notes" and "Tagged values". For information on these standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on nodes, please refer to the "Node class" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Node" dialog box

This screen (shown in Figure 7-1) is used to enter values for a node.

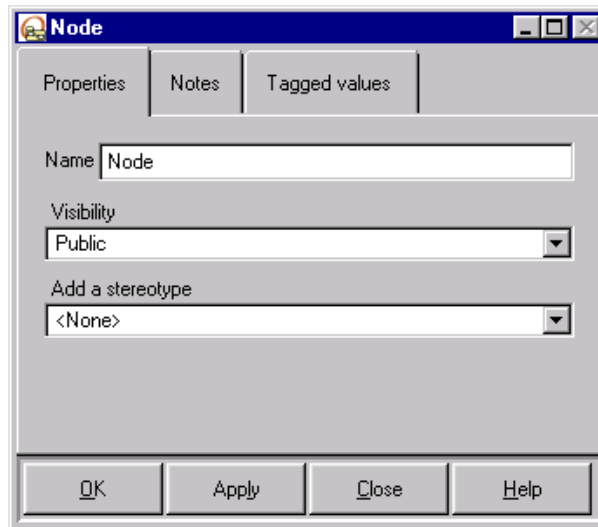


Figure 7-1. The "Properties" tab of the "Node" dialog box

Description of "Properties" tab fields

- ◆ "Name": This is the name of the node.
 - ◆ "Visibility": This indicates the visibility of the node (public, protected, private or none).
 - ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Component dialog box

Entering components



A component is a physical unit of implementation with well-defined interfaces that is intended to be used as a replaceable part of a system. Each component embodies the implementation of certain classes from the system design. Any physical element in a software development can be represented by a component. By extension, any work product (a C++ source, documentation etc) can be a specific kind of component.

The "*Component*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on components, please refer to the "*Component class*" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Component" dialog box

This screen (shown in Figure 7-2) is used to enter values for a component.

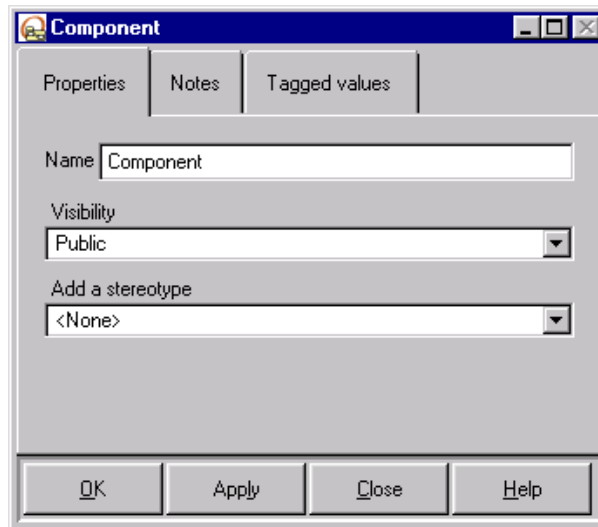


Figure 7-2. The "Properties" tab of the "Component" dialog box

Description of "Properties" tab fields

- ◆ "Name": This is the name of the component.
 - ◆ "Visibility": This indicates the visibility of the component (public, protected, private or none).
 - ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Node instance dialog box

Entering node instances



A node instance is an instance of a node. Nodes represent a kind of executable unit, whereas node instances represent examples of communicating nodes. Through clustering associations, node instances can present the instances that they contain, such as objects or components.

The "*Node instance*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on node instances, please refer to the "*NodeInstance class*" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Node instance" dialog box

This screen (shown in Figure 7-3) is used to enter values for a node instance.

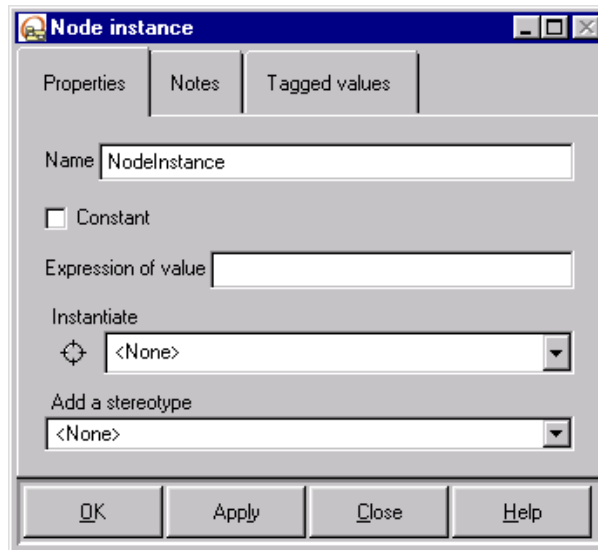


Figure 7-3. The "Properties" tab of the "Node instance" dialog box

Description of "Properties" tab fields

- ◆ "Name": This is the name of the node instance.
- ◆ "Constant": This is used to specify that the value of the node cannot change.
- ◆ "Expression of value": This is the expression of the initial value in the target language, used to instantiate the object.
- ◆ "Instantiate": This combobox is used to indicate the element to be instantiated.
- ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).

Component instance dialog box

Entering component instances



A component instance is an instance of a component. The semantic range of components starts from a C++ source code, documentation, an executable library, from the component model dedicated to business objects such as Java ejb, for example. Component instances can be deployed in specific node instances, and their behavior is specific.

The "*Component instance*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on component instances, please refer to the "*ComponentInstance class*" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Component instance" dialog box

This screen (shown in Figure 7-4) is used to enter values for a component instance.

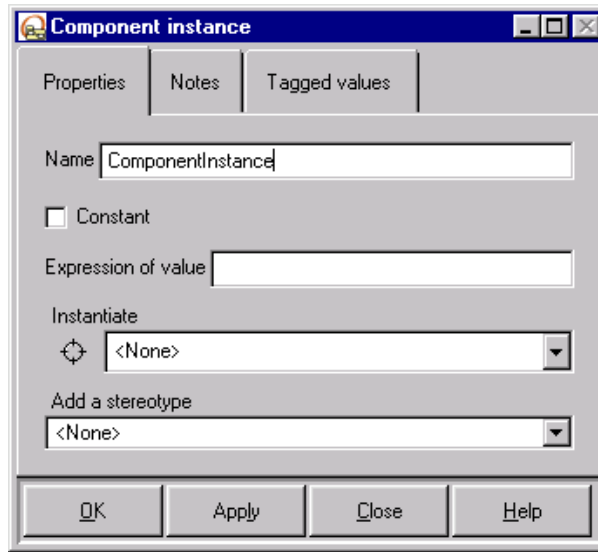


Figure 7-4. The "Properties" tab of the "Component instance" dialog box

Description of "Properties" tab fields

- ◆ "*Name*": This is the name of the component instance.
 - ◆ "*Constant*": This is used to specify that the value of the component cannot change.
 - ◆ "*Expression of value*": This is the expression of the initial value in the target language, used to instantiate the object.
 - ◆ "*Instantiate*": This combobox is used to indicate the element to be instantiated.
 - ◆ "*Add a stereotype*" : This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
-

Chapter 8: Sequence and
collaboration model dialog
boxes

Collaboration dialog box

Entering collaborations



A collaboration is the description of a general arrangement of objects and links that interact within a context to implement a behavior, such as a use case or operation.

The "*Collaboration*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on collaborations, please refer to the "*Collaboration class*" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Collaboration" dialog box

This screen (shown in figure 8-1) is used to enter values for a collaboration.

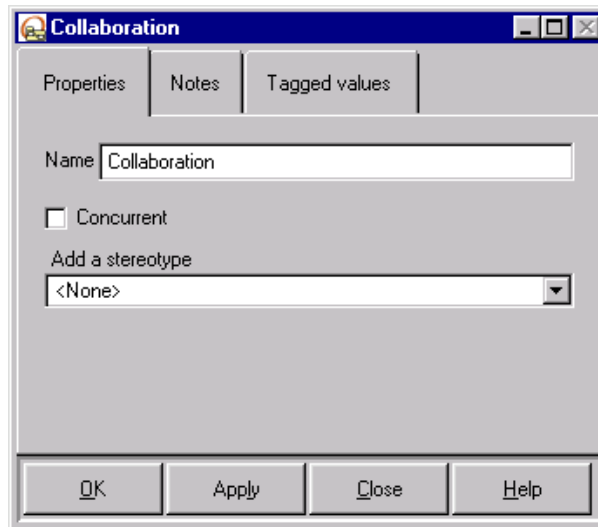


Figure 8-1. The "Properties" tab of the "Collaboration" dialog box

Description of "Properties" tab fields

- ◆ "Name": This is the collaboration's name. This name must be unique in the model.
- ◆ "Concurrent": When this box is checked, it is possible for two or more activities to be carried out at the same time, without the implication that these activities are synchronized.
- ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).

Instance dialog box

Entering instances



An instance is the result of the instantiation of a class.

The "*Instance*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on instances, please refer to the "*Instance class*" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Instance" dialog box

This screen (shown in figure 8-2) is used to enter values for an instance.

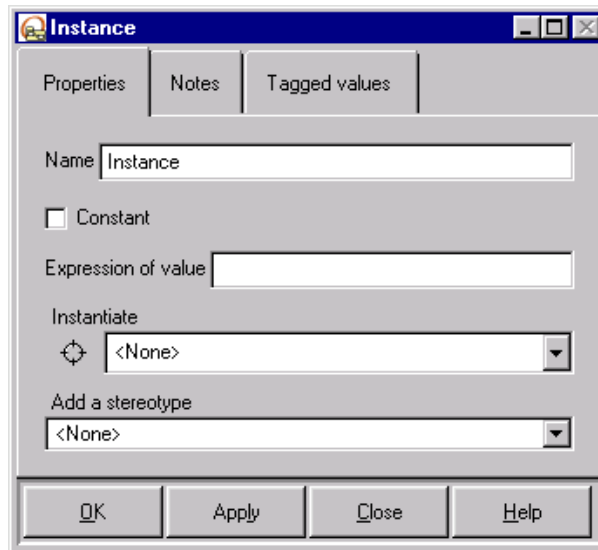


Figure 8-2. The "Properties" tab of the "Instance" dialog box


Description of "Properties" tab fields

- ◆ *"Name"*: This is the name of the instance. This name must be unique in the model.
 - ◆ *"Constant"*: When this box is checked, all values on attribute roles and association roles remain the same.
 - ◆ *"Expression of value"*: This is the expression of the initial value in the target language, used to instantiate the object.
 - ◆ *"Instantiate"*: This combobox is used to indicate the element to be instantiated.
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
-

Classifier role dialog box

Entering classifier roles



(in a collaboration) or  (in a sequence diagram) A classifier role is a slot in a collaboration that describes the role played by a participant in a collaboration. It has a reference to a classifier (the base) and a multiplicity, and may have a name or be anonymous.

The "*Classifier role*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on classifier roles, please refer to the "*ClassifierRole class*" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Role" dialog box

This screen (shown in figure 8-3) is used to enter values for a classifier role.

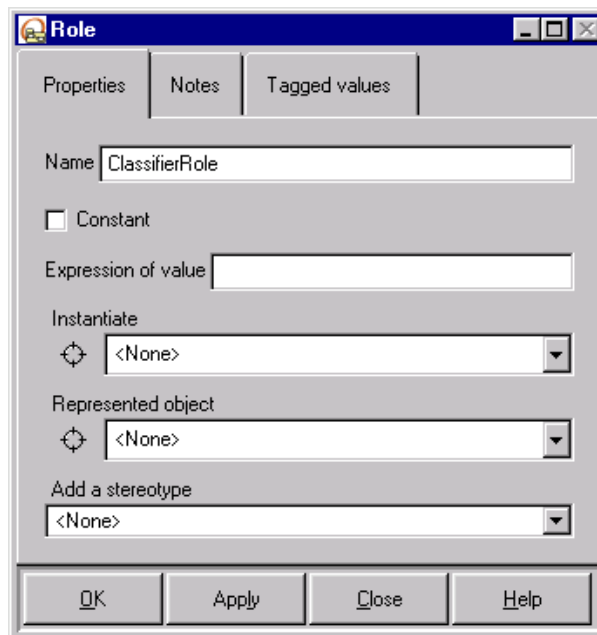


Figure 8-3. The "Properties" tab of the "Classifier role" dialog box

Description of "Properties" tab fields

- ◆ "*Name*": This is the name of the classifier role. This name must be unique in the model.
 - ◆ "*Constant*": When this box is checked, all values on attribute roles and association roles remain the same.
 - ◆ "*Expression of value*": This is the expression of the initial value in the target language, used to instantiate the object.
 - ◆ "*Instantiate*": This combobox is used to indicate the class which is represented.
 - ◆ "*Represented object*": The relation between the role and the instance.
 - ◆ "*Add a stereotype*": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Attribute link dialog box

Entering attribute links



An attribute link is an occurrence of an attribute in an instance. In particular, it contains an attribute value.

The "*Attribute link*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on attribute links, please refer to the "*AttributeLink class*" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Attribute link" dialog box

This screen (shown in figure 8-4) is used to enter values for an attribute link.

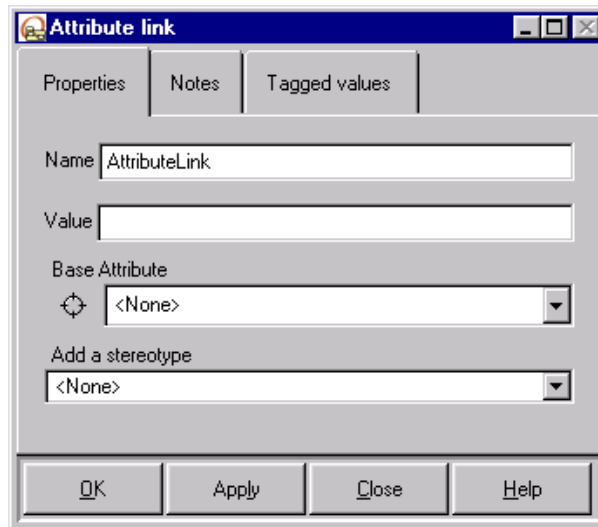


Figure 8-4. The "Properties" tab of the "Attribute link" dialog box

Description of "Properties" tab fields

- ◆ "Name": This indicates the name of the attribute link. This name must be unique in the model.
- ◆ "Value": This indicates the value of the attribute link. This field may be freely defined.
- ◆ "Base Attribute": This is the attribute represented by the attribute link.
- ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).

Attribute role dialog box

Entering attribute roles



An attribute role is an instance of an attribute in a role.

The "Attribute role" dialog box contains three tabs - "Properties", "Notes" and "Tagged values". For information on these standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on attribute roles, please refer to the "AttributeRole class" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Attribute role" dialog box

This screen (shown in figure 8-5) is used to enter values for an attribute role.

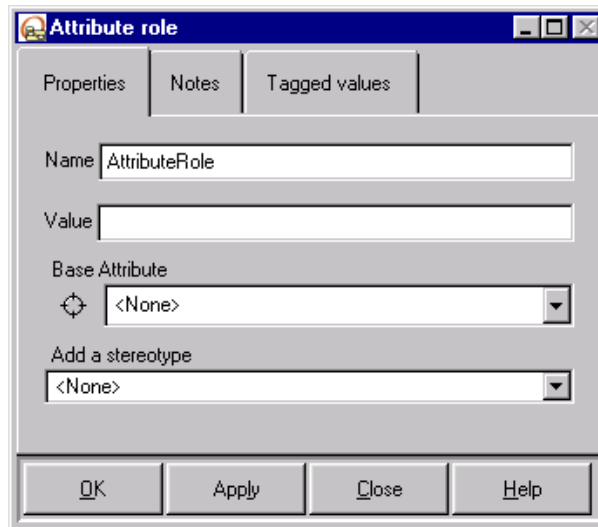



Figure 8-5. The "Properties" tab of the "Attribute role" dialog box

Description of "Properties" tab fields

- ◆ "Name": This indicates the name of the attribute role. This name must be unique in the model.
- ◆ "Value": This indicates the value of the attribute role. This field may be freely defined.
- ◆ "Base Attribute": This is the attribute represented by the attribute link.
- ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).

Link dialog box

Entering links

 A link is a tuple of object references that is an instance of an association or an association role.

The "Link" dialog box contains three tabs - "Properties", "Notes" and "Tagged values". For information on the standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on links, please refer to the "Link class" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Link" dialog box

This screen (shown in Figure 8-6) is used to enter values for a link.

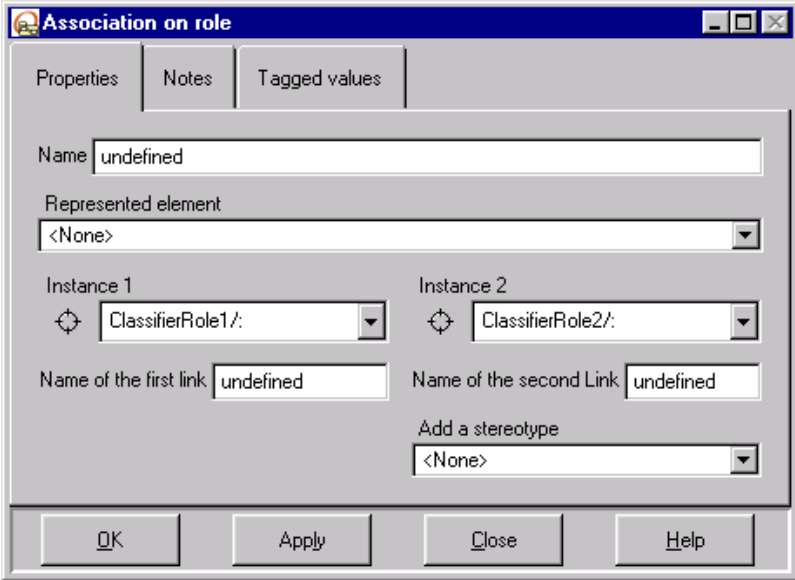


Figure 8-6. The "Properties" tab of the "Link" dialog box

Description of "Properties" tab fields

- ◆ "Name": This indicates the link's name. This name must be unique in the model.
 - ◆ "Element represented": This defines the element which is represented.
 - ◆ "Instance 1": This is the name of the first instance.
 - ◆ "Instance 2": This is the name of the second instance.
 - ◆ "Name of the first link": This is the name of the first link.
 - ◆ "Name of the second link": This is the name of the second link.
 - ◆ "Add a stereotype": This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Collaboration message dialog box

Entering collaboration messages



A collaboration message is a message sent by a role to another role in a collaboration diagram.

The "*Collaboration message*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on collaboration messages, please refer to the "*CollaborationMessage class*" section of the *Objectteering/Metamodel User Guide*.

The "Properties" tab of the "Collaboration message" dialog box

This screen (shown in figure 8-7) is used to enter values for a collaboration message.

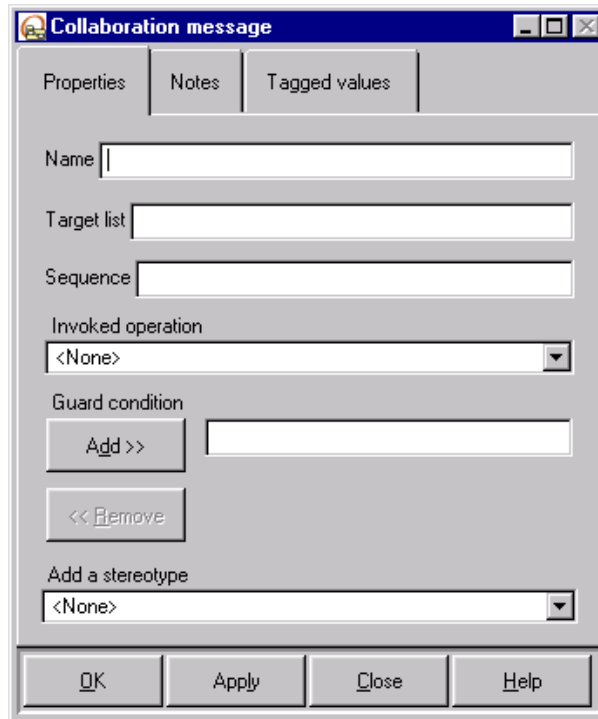


Figure 8-7. The "Properties" tab of the "Collaboration message" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This is the name of the collaboration message. This name must be unique in the model. The name may only be freely defined where the *"Invoked operation"* field is set to *"<None>"*. If an operation has been selected in the *"Invoked operation"* field, the *"Name"* field is automatically defined in accordance with the operation selected.
 - ◆ *"Target list"*: This is the list of arguments for the operation.
 - ◆ *"Sequence"*: This field allows the user to give the order number of the message.
 - ◆ *"Invoked operation"*: This field allows the user to select an existing operation on the destination object of the message. If an operation is selected, the *"Name"* field is automatically defined in accordance.
 - ◆ *"Guard condition"*: This is the condition under which a collaboration message may be triggered (for further information on conditions, please refer to the *"Condition class"* section of the *Objectteering/Metamodel User Guide*).
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
-

Sequence message dialog box

Entering sequence messages



A sequence message is a message between two roles or two instances, expressed in a sequence diagram.

The "Sequence message" dialog box contains three tabs - "Properties", "Notes" and "Tagged values". For information on these standard dialog box tabs, please refer to the "Standard dialog box tabs" section of this user guide.

For further information on sequence messages, please refer to the "SequenceMessage class" section of the *Objecteering/Metamodel User Guide*.

The "Properties" tab of the "Sequence message" dialog box

This screen (shown in Figure 8-8) is used to enter values for a sequence message.

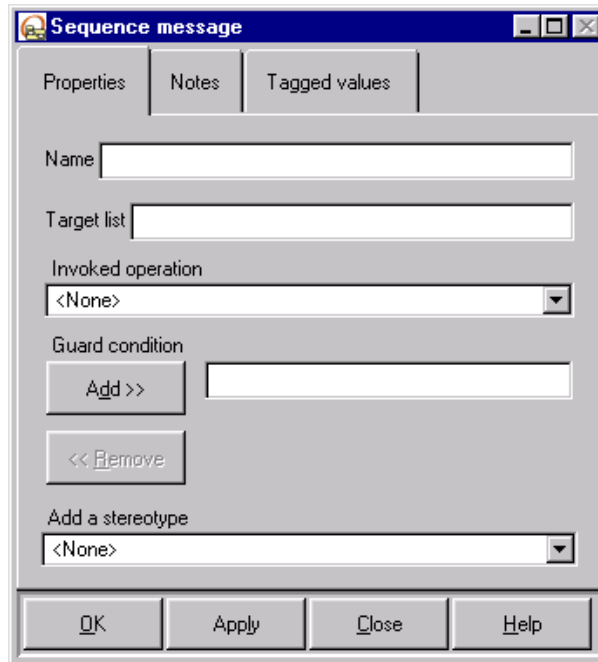


Figure 8-8. The "Properties" tab of the "Sequence message" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This is the name of the sequence message. This name must be unique in the model. The name may only be freely defined where the *"Invoked operation"* field is set to *"<None>"*. If an operation has been selected in the *"Invoked operation"* field, the *"Name"* field is automatically defined in accordance with the operation selected.
 - ◆ *"Target list"*: This is the list of arguments for the operation.
 - ◆ *"Invoked operation"*: This field allows the user to select an existing operation on the destination object of the message. If an operation is selected, the *"Name"* field is automatically defined in accordance.
 - ◆ *"Guard condition"*: This is the condition under which a sequence message may be triggered (for further information on conditions, please refer to the *"Condition class"* section of the *Objectteering/Metamodel User Guide*).
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
-

Chapter 9: Diagram dialog boxes

Class diagram dialog box

Entering class diagrams



Class diagrams allow you to present the internal structure of an element and its relationships with other elements (referenced by it). They are editors capable of presenting a great variety of elements.

The main elements of class diagrams are classes, packages, associations, generalizations and dependencies.

A class diagram is created in a package or a class. For further details on the creation of a diagram, please refer to the "*Class diagram*" section in chapter 7 of the *Objectteering/UML Modeler* user guide.

The "*Class diagram*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on class diagrams, please refer to the "*Diagram class*" section of the *Objectteering/J Libraries User Guide*.

The "Properties" tab of the "Class diagram" dialog box

This screen (shown in Figure 9-1) is used to modify values for a class diagram.



Figure 9-1. The "Properties" tab of the "Class diagram" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This is the name of the class diagram.
 - ◆ *"Detailed visibility"*: This indicates whether or not the visibility and path are detailed. For example, the "+,-,#" symbols are shown on class members, and the class name includes the package paths "P1:P2:C1".
 - ◆ *"Tagged value visibility"*: This indicates whether or not tagged values are presented in the diagram.
 - ◆ *"Stereotype display"*: This indicates the form of the stereotype display. Stereotypes are presented as icons only if the element contains no other elements (for example, a class can present an icon stereotype if it does not show its members).
 - ◆ *"Horizontal spacing"*: This indicates the horizontal spacing between boxes after automatic positioning (layout).
 - ◆ *"Vertical spacing"*: This indicates the vertical spacing between boxes after automatic positioning (layout).
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objectteering/UML Profile Builder* user guide).
-

Deployment diagram dialog box

Entering deployment diagrams



The deployment diagram is used to represent the physical architecture of the system. It presents the distribution of software components on the set of execution units (Node).

Nodes and components are its main concepts.

A deployment diagram is created in a package or a class. For further details on the creation of a diagram, please refer to the "*Deployment diagram*" section in chapter 7 of the *Objectteering/UML Modeler* user guide.

The "*Deployment diagram*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on deployment and component diagrams, please refer to the "*Diagram class*" section of the *Objectteering/J Libraries User Guide*.

The "Properties" tab of the "Deployment diagram" dialog box

This screen (shown in Figure 9-2) is used to modify values for a deployment diagram.



Figure 9-2. The "Properties" tab of the "Deployment diagram" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This is the name of the deployment diagram.
 - ◆ *"Detailed visibility"*: This indicates whether or not the visibility and path are detailed. For example, the "+,-,#" symbols are shown on class members, and the class name includes the package paths "P1:P2:C1".
 - ◆ *"Tagged value visibility"*: This indicates whether or not tagged values are presented in the diagram.
 - ◆ *"Stereotype display"*: This indicates the form of the stereotype display. Stereotypes are presented as icons only if the element contains no other elements (for example, a class can present an icon stereotype if it does not show its members).
 - ◆ *"Horizontal spacing"*: This indicates the horizontal spacing between boxes after automatic positioning (layout).
 - ◆ *"Vertical spacing"*: This indicates the vertical spacing between boxes after automatic positioning (layout).
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Deployment instance diagram dialog box

Entering deployment instance diagrams



The deployment instance diagram presents a particular instance of deployment.

It represents instances of nodes and instances of components that can correspond to an example illustration component and the deployment diagrams.

A deployment instance diagram is created in a package or a class. For further details on the creation of a diagram, please refer to the "*Deployment instance diagram*" section in chapter 7 of the *Objectteering/UML Modeler* user guide.

The "*Deployment instance diagram*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on deployment instance diagrams, please refer to the "*Diagram class*" section of the *Objectteering/J Libraries User Guide*.

The "Properties" tab of the "Deployment instance diagram" dialog box

This screen (shown in Figure 9-3) is used to modify values for a deployment instance diagram.

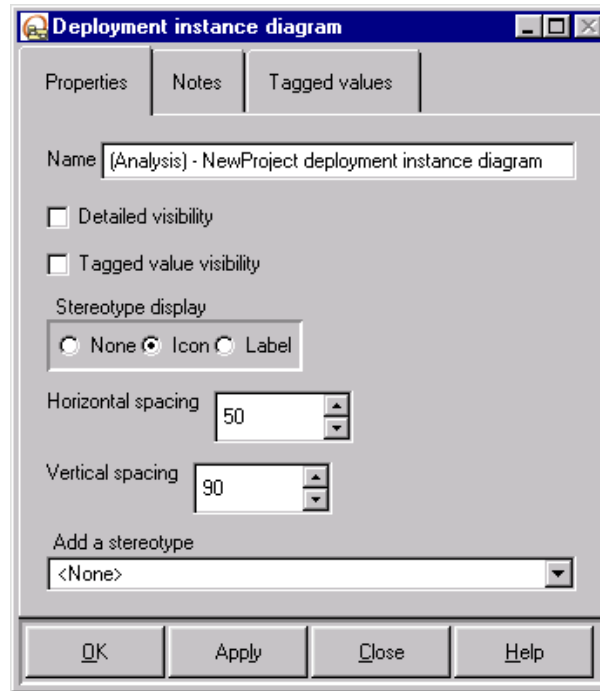


Figure 9-3. The "Properties" tab of the "Deployment instance diagram" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This is the name of the deployment instance diagram.
 - ◆ *"Detailed visibility"*: This indicates whether or not the visibility and path are detailed. For example, the "+,-,#" symbols are shown on class members, and the class name includes the package paths "P1:P2:C1".
 - ◆ *"Tagged value visibility"*: This indicates whether or not tagged values are presented in the diagram.
 - ◆ *"Stereotype display"*: This indicates the form of the stereotype display. Stereotypes are presented as icons only if the element contains no other elements (for example, a class can present an icon stereotype if it does not show its members).
 - ◆ *"Horizontal spacing"*: This indicates the horizontal spacing between boxes after automatic positioning (layout).
 - ◆ *"Vertical spacing"*: This indicates the vertical spacing between boxes after automatic positioning (layout).
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Object diagram dialog box

Entering object diagrams



The object diagram presents a set of class instances with their links and the messages exchanged. It can be created from a package or a class. Objects and links can be created without being linked to a class or an association. Messages are directly added to an existing link; if the link is oriented, the message is created with the same orientation; if not, it is created oriented towards the box nearest to the point where the user has clicked.

A synchronous message is represented near the link in the form of a complete arrow and its label. An asynchronous message is represented near the link in the form of a empty half arrow and its label.

It is not possible to create or represent a message for an n-ary link.

Objects can be connected to existing classes, or created independently for those classes. Connecting objects to classes will then give the ability to connect links to associations and messages to operations.

An object diagram is created in a package or a class. For further details on the creation of a diagram, please refer to the "*Object diagram*" section in chapter 7 of the *Objectteering/UML Modeler* user guide.

The "*Object diagram*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on object diagrams, please refer to the "*Diagram class*" section of the *Objectteering/J Libraries User Guide*.

The "Properties" tab of the "Object diagram" dialog box

This screen (shown in Figure 9-4) is used to modify values for an object diagram.

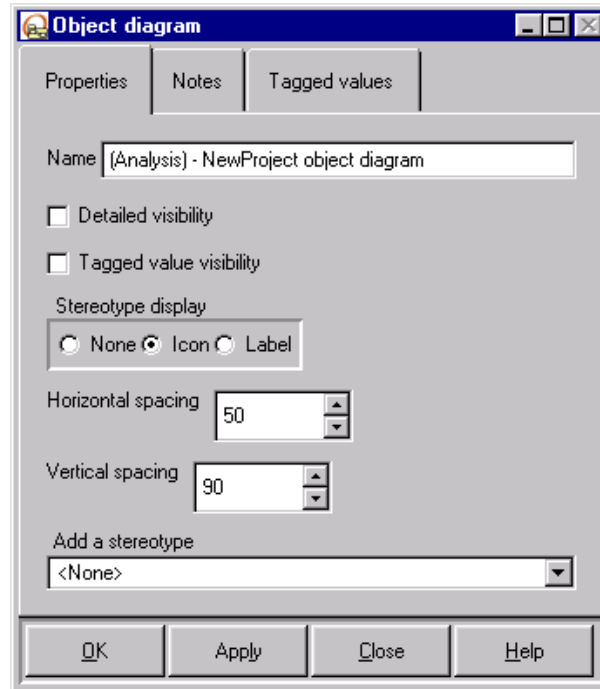


Figure 9-4. The "Properties" tab of the "Object diagram" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This is the name of the object diagram.
 - ◆ *"Detailed visibility"*: This indicates whether or not the visibility and path are detailed. For example, the "+,-,#" symbols are shown on class members, and the class name includes the package paths "P1:P2:C1".
 - ◆ *"Tagged value visibility"*: This indicates whether or not tagged values are presented in the diagram.
 - ◆ *"Stereotype display"*: This indicates the form of the stereotype display. Stereotypes are presented as icons only if the element contains no other elements (for example, a class can present an icon stereotype if it does not show its members).
 - ◆ *"Horizontal spacing"*: This indicates the horizontal spacing between boxes after automatic positioning (layout).
 - ◆ *"Vertical spacing"*: This indicates the vertical spacing between boxes after automatic positioning (layout).
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Sequence diagram dialog box

Entering sequence diagrams



A sequence diagram shows how different objects cooperate. The objects (vertical bars) can be defined a priori, and can be roles or class instances. Cooperation between objects is represented by the sending of messages between objects (horizontal arrows), and their sequence (order from top to bottom). A sequence diagram can be created in:

- ◆ a package
- ◆ a class
- ◆ a use case
- ◆ a collaboration

If the sequence diagram is created in a collaboration, the objects are classifier roles. If it is created in a package, a class or a use case, the objects are instances.

Objects can be connected to existing classes, or created independently from any class. Connecting objects to classes will then give the ability to connect links to associations and messages to operations.

A sequence diagram is created in a package or a class. For further details on the creation of a diagram, please refer to the "*Sequence diagram*" section in chapter 7 of the *Objectteering/UML Modeler* user guide.

The "*Sequence diagram*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on sequence diagrams, please refer to the "*Diagram class*" section of the *Objectteering/J Libraries User Guide*.

The "Properties" tab of the "Sequence diagram" dialog box

This screen (shown in Figure 9-5) is used to modify values for a sequence diagram.

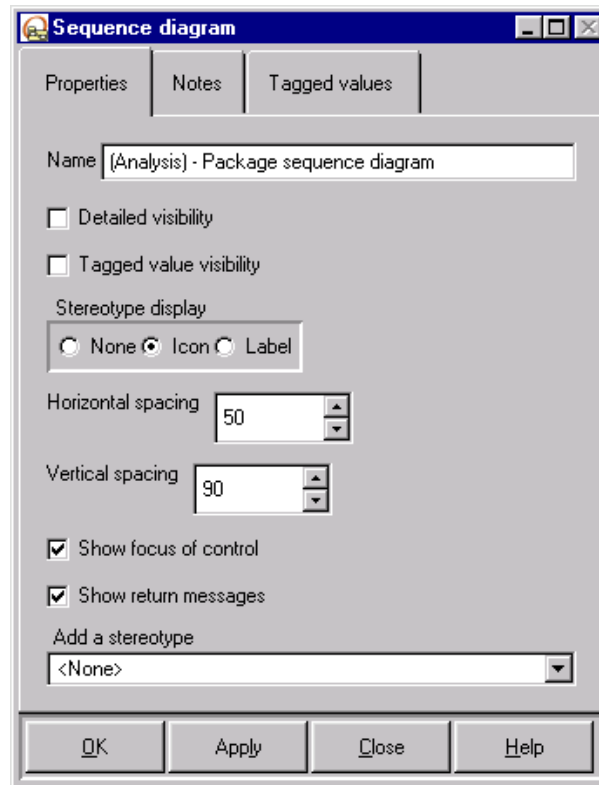


Figure 9-5. The "Properties" tab of the "Sequence diagram" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This is the name of the sequence diagram.
 - ◆ *"Detailed visibility"*: This indicates whether or not the visibility and path are detailed. For example, the "+,-,#" symbols are shown on class members, and the class name includes the package paths "P1:P2:C1".
 - ◆ *"Tagged value visibility"*: This indicates whether or not tagged values are presented in the diagram.
 - ◆ *"Stereotype display"*: This indicates the form of the stereotype display. Stereotypes are presented as icons only if the element contains no other elements (for example, a class can present an icon stereotype if it does not show its members).
 - ◆ *"Horizontal spacing"*: This indicates the horizontal spacing between boxes after automatic positioning (layout).
 - ◆ *"Vertical spacing"*: This indicates the vertical spacing between boxes after automatic positioning (layout).
 - ◆ *"Show focus of control"*: Focus of control of messages can be presented or hidden by this button.
 - ◆ *"Show return messages"*: The dotted arrow which symbolizes the return of messages can be hidden or not.
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Collaboration diagram dialog box

Entering collaboration diagrams



The collaboration diagram allows you to present exchanges of messages between roles. It is semantically very close to the object diagram. This diagram is created for a collaboration.

A collaboration defines a context in which roles non exist.

A collaboration diagram is created in a package or a class. For further details on the creation of a diagram, please refer to the "*Collaboration diagram*" section in chapter 7 of the *Objectteering/UML Modeler* user guide.

The "*Collaboration diagram*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on collaboration diagrams, please refer to the "*Diagram class*" section of the *Objectteering/J Libraries User Guide*.

The "Properties" tab of the "Collaboration diagram" dialog box

This screen (shown in Figure 9-6) is used to modify values for a collaboration diagram.

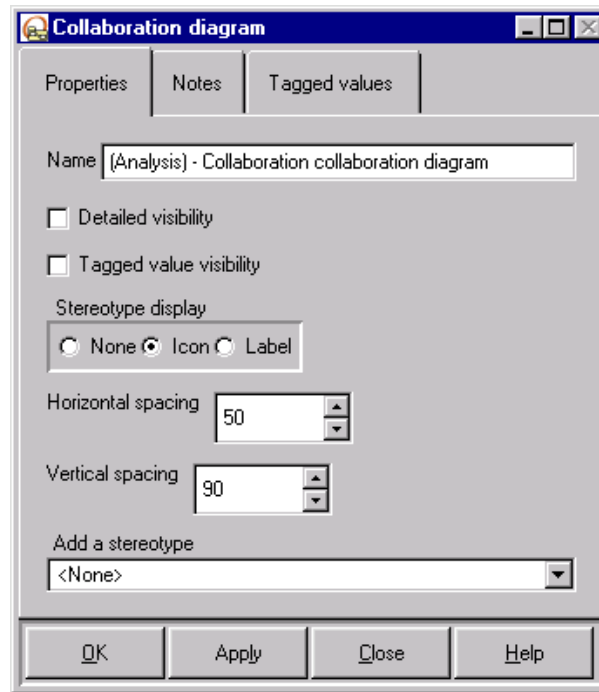


Figure 9-6. The "Properties" tab of the "Collaboration diagram" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This is the name of the collaboration diagram.
 - ◆ *"Detailed visibility"*: This indicates whether or not the visibility and path are detailed. For example, the "+,-,#" symbols are shown on class members, and the class name includes the package paths "P1:P2:C1".
 - ◆ *"Tagged value visibility"*: This indicates whether or not tagged values are presented in the diagram.
 - ◆ *"Stereotype display"*: This indicates the form of the stereotype display. Stereotypes are presented as icons only if the element contains no other elements (for example, a class can present an icon stereotype if it does not show its members).
 - ◆ *"Horizontal spacing"*: This indicates the horizontal spacing between boxes after automatic positioning (layout).
 - ◆ *"Vertical spacing"*: This indicates the vertical spacing between boxes after automatic positioning (layout).
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Use case diagram dialog box

Entering use case diagrams



The use case diagram is used to describe the most important services rendered by the system. Starting with actors, external participants who interact with the system, they represent the most important cases of system operating. A use case may then be sub-divided into sequence diagrams, which detail the different functions of one use case.

A use case diagram is created in a package or a class. For further details on the creation of a diagram, please refer to the "*Use case diagram*" section in chapter 7 of the *Objectteering/UML Modeler* user guide.

The "*Use case diagram*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on use case diagrams, please refer to the "*Diagram class*" section of the *Objectteering/J Libraries User Guide*.

The "Properties" tab of the "Use case diagram" dialog box

This screen (shown in Figure 9-7) is used to modify values for a use case diagram.

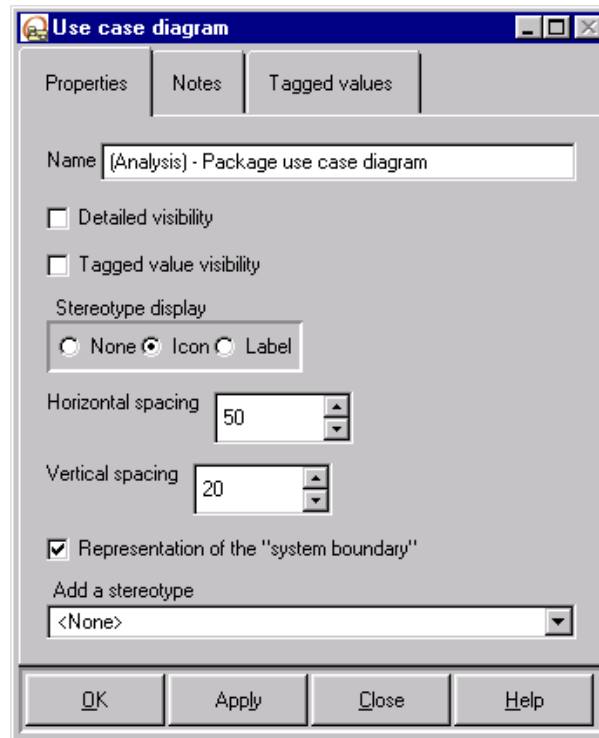


Figure 9-7. The "Properties" tab of the "Use case diagram" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This is the name of the use case diagram.
 - ◆ *"Detailed visibility"*: This indicates whether or not the visibility and path are detailed. For example, the "+,-,#" symbols are shown on class members, and the class name includes the package paths "P1:P2:C1".
 - ◆ *"Tagged value visibility"*: This indicates whether or not tagged values are presented in the diagram.
 - ◆ *"Stereotype display"*: This indicates the form of the stereotype display. Stereotypes are presented as icons only if the element contains no other elements (for example, a class can present an icon stereotype if it does not show its members).
 - ◆ *"Horizontal spacing"*: This indicates the horizontal spacing between boxes after automatic positioning (layout).
 - ◆ *"Vertical spacing"*: This indicates the vertical spacing between boxes after automatic positioning (layout).
 - ◆ *"System boundary representation"*: This indicates whether or not the system boundary is represented (square symbolizing the owner package).
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

State diagram dialog box

Entering state diagrams



State diagrams can be defined from a class, a package or an operation. A state diagram allows you to describe the manner in which objects react to events. It is used to describe a state machine at class level.

State diagrams can also represent protocol.

A state diagram is created in a state machine created on a package or a class. For further details on the creation of a diagram, please refer to the "*State diagram*" section in chapter 7 of the *Objectteering/UML Modeler* user guide.

The "*State diagram*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on state diagrams, please refer to the "*Diagram class*" section of the *Objectteering/J Libraries User Guide*.

The "Properties" tab of the "State diagram" dialog box

This screen (shown in Figure 9-8) is used to modify values for a state diagram.



Figure 9-8. The "Properties" tab of the "State diagram" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This is the name of the state diagram.
 - ◆ *"Detailed visibility"*: This indicates whether or not the visibility and path are detailed. For example, the "+,-,#" symbols are shown on class members, and the class name includes the package paths "P1:P2:C1".
 - ◆ *"Tagged value visibility"*: This indicates whether or not tagged values are presented in the diagram.
 - ◆ *"Stereotype display"*: This indicates the form of the stereotype display. Stereotypes are presented as icons only if the element contains no other elements (for example, a class can present an icon stereotype if it does not show its members).
 - ◆ *"Horizontal spacing"*: This indicates the horizontal spacing between boxes after automatic positioning (layout).
 - ◆ *"Vertical spacing"*: This indicates the vertical spacing between boxes after automatic positioning (layout).
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Activity diagram dialog box

Entering activity diagrams



Activity diagrams graphically illustrate activity graphs, which show a procedure or a workflow. An activity graph is a special instance of a state machine in which all or most of the states are activity states or action states, and in which all or most of the transitions are triggered by completion of activity in the source states.

The main elements of activity diagrams are action states, sub activity states, object flow states and partitions.

An activity diagram is created in an activity graph on a package or a class. For the creation of a diagram, please refer to the "*Activity diagram*" section in chapter 7 of the *Objectteering/UML Modeler* user guide.

The "*Activity diagram*" dialog box contains three tabs - "*Properties*", "*Notes*" and "*Tagged values*". For information on these standard dialog box tabs, please refer to the "*Standard dialog box tabs*" section of this user guide.

For further information on activity diagrams, please refer to the "*Diagram class*" section of the *Objectteering/J Libraries User Guide*.

The "Properties" tab of the "Activity diagram" dialog box

This screen (shown in Figure 9-9) is used to modify values for an activity diagram.

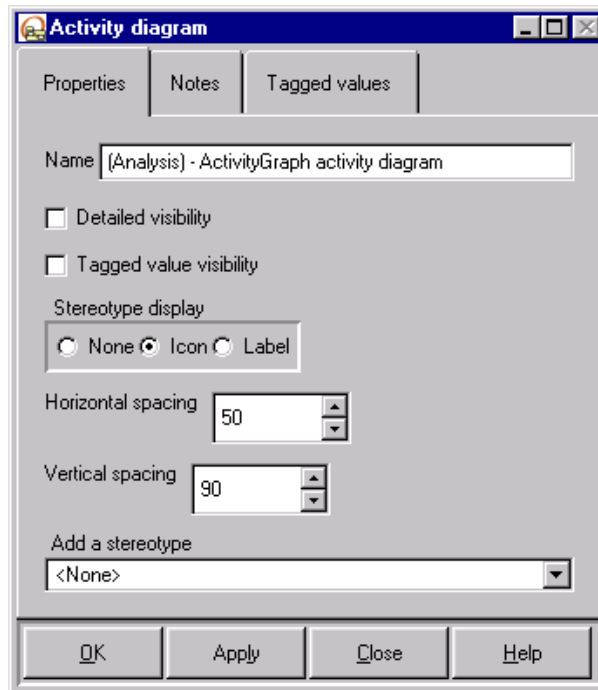


Figure 9-9. The "Properties" tab of the "Activity diagram" dialog box

Description of "Properties" tab fields

- ◆ *"Name"*: This is the name of the activity diagram.
 - ◆ *"Detailed visibility"*: This indicates whether or not the visibility and path are detailed. For example, the "+,-,#" symbols are shown on class members, and the class name includes the package paths "P1:P2:C1".
 - ◆ *"Tagged value visibility"*: This indicates whether or not tagged values are presented in the diagram.
 - ◆ *"Stereotype display"*: This indicates the form of the stereotype display. Stereotypes are presented as icons only if the element contains no other elements (for example, a class can present an icon stereotype if it does not show its members).
 - ◆ *"Horizontal spacing"*: This indicates the horizontal spacing between boxes after automatic positioning (layout).
 - ◆ *"Vertical spacing"*: This indicates the vertical spacing between boxes after automatic positioning (layout).
 - ◆ *"Add a stereotype"*: This field allows the user to specify the semantics of a given existing class, if he wishes. Stereotypes are previously defined at UML profiling project level (for further information, please refer to the *Objecteering/UML Profile Builder* user guide).
-

Index

{persistent} tagged value	1-23	Class	1-14, 1-25, 2-9, 3-12, 3-30, 3-36, 3-44, 5-3, 8-5, 9-3, 9-9, 9-12, 9-15, 9-24
<<extend>>	4-9	Properties tab	3-7
<<include>>	4-9	Properties tab description	3-8
Action state	6-3, 9-27	Class association	3-30
Properties tab	6-6	Properties tab	3-30
Properties tab description	6-7	Properties tab description	3-31
Activity diagram	9-27	Class diagram	9-3
Properties tab	9-28	Properties tab	9-4
Properties tab description	9-29	Properties tab description	9-5
Activity graph	6-3, 6-8, 6-11, 6-14, 9-27	Class instance	9-12, 9-15
Properties tab	6-4	Classifier role	8-8
Properties tab description	6-4	Properties tab	8-9
Activity state	6-3, 9-27	Properties tab description	8-10
Actor	4-3, 4-5, 9-21	Classifier role	9-15
Properties tab	4-6	Collaboration	8-3, 8-8, 9-15, 9-18
Properties tab description	4-6	Properties tab	8-4
Aggregation	3-25	Properties tab description	8-4
Association	3-6, 3-25, 3-30, 7-3, 8-15, 9-3, 9-12, 9-15	Collaboration diagram	8-17
Association role	8-15	Properties tab	9-19
Asynchronous message	9-12	Properties tab description	9-20
Attribute	1-14, 1-25, 2-9, 3-6, 8-11, 8-13	Collaboration message	8-17
Properties tab	3-13	Properties tab	8-18
Properties tab description	3-14	Properties tab description	8-19
Attribute link	8-11, 8-14	Communication	4-7
Properties tab	8-12	Properties tab	4-8
Properties tab description	8-12	Properties tab description	4-8
Attribute role	8-13	Communication link	4-7
Properties tab	8-14	Component	1-16, 3-44, 7-5, 7-7, 7-9, 9-6, 9-9
Properties tab description	8-14	Properties tab	7-6
Binary association	3-25	Properties tab description	7-6
Properties tab	3-26	Component instance	7-9
Properties tab description	3-27	Properties tab	7-10
C++	1-20	Properties tab description	7-11

- Condition 5-11
- Constraint
 - Properties tab 2-4
- Constraint 2-3
 - Properties tab description 2-5
- Consult mode 1-13
- Consulting an element 1-3
- Consulting an existing element
 - Description 1-13
- Continuous entry creation mode 1-3, 1-7, 1-25
- Creating a diagram 9-6, 9-9, 9-12, 9-15, 9-18, 9-21, 9-24, 9-27
- Creating a new element 1-3
- Creating a new element in a graphic editor 1-6
- Creating a new element in the explorer 1-4
- Creating a new element in the properties editor 1-6
- Creating elements using the continuous entry creation mode
 - Creating from the explorer 1-7
 - Description 1-7
- Creating elements using the repeated entry system
 - Creating from a graphic editor 1-9
- Creation new elements using the continuous entry creation mode 1-3
- Data entry
 - Shortcuts 1-3
- Data flow 1-16, 3-36, 3-38
 - Properties tab 3-39
 - Properties tab description 3-39
- Data type 3-9
 - Properties tab 3-10
 - Properties tab description 3-11
- Dependency 9-3
- Dependency link 3-6
- Deployment diagram 9-6
 - Properties tab 9-7
 - Properties tab description 9-8
- Deployment instance diagram 9-9
 - Properties tab 9-10
 - Properties tab description 9-11
- Diagrams 1-7
- Dialog box 1-25
- Dialog box buttons 1-12
- Documentation 1-20, 1-25
- Drag and drop 1-14
- Drag and drop function 1-18
- Editing an element 1-3
- Element 1-25
- Entering action states 6-5
- Entering activity diagrams 9-27
- Entering activity graphs 6-3
- Entering actors 4-5
- Entering attribute links 8-11
- Entering attribute roles 8-13
- Entering attributes 3-12
- Entering binary associations 3-25
- Entering class associations 3-30
- Entering class diagrams 9-3
- Entering classes 3-6
- Entering classifier roles 8-8
- Entering collaboration diagrams 9-18
- Entering collaboration messages 8-17
- Entering collaborations 8-3
- Entering communication links 4-7
- Entering component instances 7-9
- Entering components 7-5
- Entering constraints 2-3

- Entering data flows 3-38
- Entering data types 3-9
- Entering deployment diagrams 9-6
- Entering deployment instance diagrams 9-9
- Entering enumeration literals 3-35
- Entering enumerations 3-32
- Entering events 5-15
- Entering generalizations 3-40
- Entering instances 8-5
- Entering internal transitions 5-12
- Entering links 8-15
- Entering n-ary associations 3-28
- Entering node instances 7-7
- Entering nodes 7-3
- Entering notes 2-6
- Entering object diagrams 9-12
- Entering object flow states 6-11
- Entering operations 3-15
- Entering packages 3-3
- Entering parameters 3-20
- Entering partitions 6-14
- Entering pseudo states 5-7
- Entering realizations 3-44
- Entering references between elements 1-14
 - Free selection 1-14
 - Mixed selection 1-14
 - Selection from a list 1-14
- Entering return parameters 3-23
- Entering sequence diagrams 9-15
- Entering sequence messages 8-20
- Entering signals 3-36
- Entering state diagrams 9-24
- Entering state machines 5-3
- Entering states 5-5
- Entering sub activity states 6-8
- Entering tag parameters 2-8
- Entering tagged values 2-9
- Entering template parameters 3-46
- Entering transitions 5-9
- Entering use case dependencies 4-9
- Entering use case diagrams 9-21
- Entering use cases 4-3
- Entering uses 3-42
- Enumerates 3-6
- Enumeration
 - Properties tab 3-33
 - Properties tab description 3-34
- Enumeration literal
 - Description of fields 3-35
- Event 3-36, 5-15, 9-24
 - Properties tab 5-16
 - Properties tab description 5-17
- Explorer 1-3, 1-7, 1-10, 1-14, 1-16, 3-20, 3-23
 - Creating new elements 1-4
- Generalization 3-40, 9-3
 - Properties tab 3-41
 - Properties tab description 3-41
- Generalization link 3-6
- Graphic editors 1-3
 - Creating new elements 1-6
- Instance 3-6, 3-36, 5-15, 7-9, 8-5, 8-11, 8-20
 - Properties tab 8-6
 - Properties tab description 8-7
- Interface 3-6, 3-44
 - Properties tab description 3-8
 - Properties tab screen 3-7
- Internal transition 5-12
 - Properties tab 5-13

- Properties tab description 5-14
- Link 8-15, 9-12
 - Properties tab 8-15
 - Properties tab description 8-16
- Links between elements 1-14
- Message 3-36, 9-12, 9-15, 9-18
 - Asynchronous 9-12
 - Synchronous 9-12
- Model elements 2-3
- Modifying a description 1-22
- Modifying a tagged value 1-24
- Modifying an existing element
 - Description 1-10
 - Modifying from a graphic editor 1-11
 - Modifying from the explorer or the properties editor 1-10
- N-ary association 3-25, 3-28, 9-12
 - Properties tab 3-28
 - Properties tab description 3-29
- Node 7-3, 7-7, 9-6, 9-9
 - Properties tab 7-4
 - Properties tab description 7-4
- Node instance 1-16, 7-7
 - Properties tab 7-8
 - Properties tab description 7-8
- Non-modal windows 1-3
- Note 1-25, 2-6
 - Properties tab 2-7
 - Properties tab description 2-7
- Notes on model elements 1-20
- Object 1-16
- Object diagram 9-12, 9-18
 - Properties tab 9-13
 - Properties tab description 9-14
- Object flow state 6-11, 9-27
 - Properties tab 6-12
 - Properties tab description 6-13
- Objecteering/Documentation 1-6
- Objecteering/Java 1-6
- Objecteering/UML modules 1-25
- Operation 3-6, 3-20, 3-23, 5-3, 5-11, 8-3, 9-12, 9-15, 9-24
 - Implementation tab 3-18
 - Implementation tab description 3-19
 - Properties tab 3-16
 - Properties tab description 3-17
- Operations on model elements 1-3
- Package 1-14, 1-16, 3-3, 3-36, 4-5, 5-3, 9-3, 9-9, 9-12, 9-15, 9-24
 - Properties tab 3-4
 - Properties tab description 3-5
- Parameter 3-20
 - Properties tab 3-21
 - Properties tab description 3-22
- Parameter: 3-17
- Partition 6-14, 9-27
 - Properties tab 6-15
 - Properties tab description 6-15
- Properties editor 1-3, 1-10, 1-16
 - Creating new elements 1-6
- Property 1-25
- Pseudo state
 - Properties tab 5-8
 - Properties tab description 5-8
- Realization
 - Properties tab 3-45
 - Properties tab description 3-45
- Repeated entry system 1-7
- Return parameter 3-17, 3-23
 - Properties tab 3-23
 - Properties tab description 3-24

- Role 8-13, 8-20, 9-15, 9-18
- Sequence diagram 4-3, 4-5, 8-8, 8-20, 9-15, 9-21
 - Properties tab 9-16
 - Properties tab description 9-17
- Sequence message 8-20
 - Properties tab 8-21
 - Properties tab description 8-22
- Signal 5-11, 5-15
 - Properties tab 3-37
 - Properties tab description 3-37
- Standard dialog box tabs
 - Notes tab 1-20
 - Overview 1-19
 - Properties tab 1-19
 - Tagged values tab 1-23
- State 5-3, 5-5, 5-9, 5-12, 6-3, 6-5, 9-24
 - Properties tab 5-6
 - Properties tab description 5-6
- State diagram 5-11, 5-14, 9-24
 - Properties tab 9-25
 - Properties tab description 9-26
- State machine 5-3, 5-7, 5-11, 9-24, 9-27
 - Properties tab 5-4
 - Properties tab description 5-4
- Stereotypes
 - <<extend>> 4-9
 - <<include>> 4-9
- Sub activity state 6-8, 9-27
- Properties tab 6-9
- Properties tab description 6-10
- Synchronous message 9-12
- Tag parameter 2-8
 - Description of fields 2-8
- Tagged value 1-25, 2-8, 2-9
 - Description of fields 2-11
- Tagged value dialog box 2-10
- Tagged values
 - {persistent} tagged value 1-23
- Template parameter 3-46
 - Description of fields 3-46
- Transition 5-3, 5-5, 5-9, 6-3, 9-27
 - Properties tab 5-10
 - Properties tab description 5-11
- Use
 - Properties tab 3-43
 - Properties tab description 3-43
- Use case 2-9, 4-3, 4-7, 4-9, 5-3, 8-3, 9-15
 - Properties tab 4-4
 - Properties tab description 4-4
- Use case dependency 4-9
 - Properties tab 4-10
 - Properties tab description 4-10
- Use case diagram 4-3, 4-5, 9-21
 - Properties tab 9-22
 - Properties tab description 9-23
- Work product 7-5