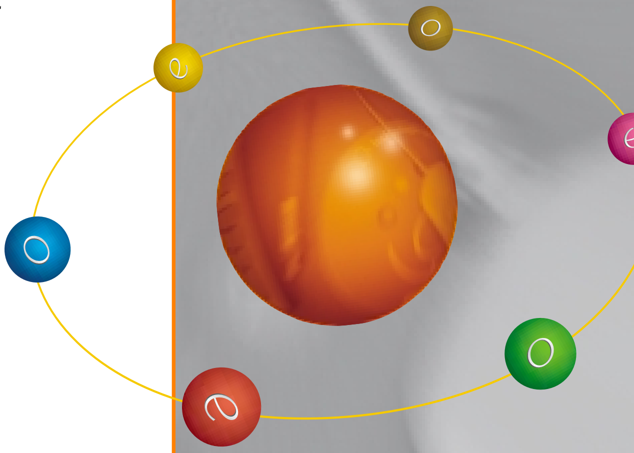
Objecteering/UML

Objecteering/Reverse COM User Guide

Version 5.2.2



Objecteering Software

www.objecteering.com

Taking object development one step further

Contents

Chapter 1: Overview of the Objecteering/Reverse COM module

Principles of the Objecteering/Reverse COM module

Welcome to the *Objecteering/Reverse COM* user guide!

The 1.0 version of the *Objecteering/Reverse COM* module is used to reconstruct a model in Objecteering/UML from the analysis of COM components.

The Objecteering/UML *COM Reverse Engineering* tool works in two distinct phases. The first phase analyzes the COM component which is to be reversed and generates intermediate ASCII files.

For this, COM components are searched for in the Windows registry and proposed in the list.

In the second phase, the classes identified by the component analysis are imported from these ASCII files (as shown in Figure 1-1).

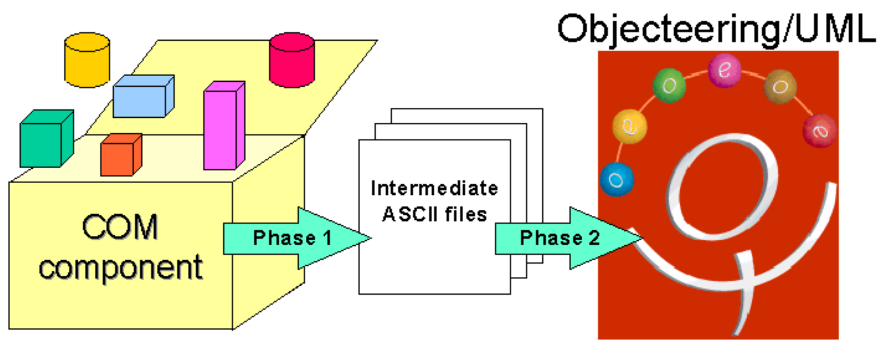


Figure 1-1. Reconstructing a model

The second function of the *Objecteering/Reverse COM* module is to generate C++ "proxy" (or client-side part) classes for the COM classes imported from components. These "proxies" can then be used in place of the real component class.

Composition of the Objecteering/Reverse COM module

The *Objecteering/Reverse COM* module is delivered in externalized form, and contains the following elements:

- the "*COMReverse.exe*" binary (for Windows NT/95/98/2000) in the "*bin/windows*" directory
- HTML documentation (the present document)
- sets and types bindings description files
- configuration and message files in the "*res*" directory

Chapter 2: Using the Objecteering/Reverse COM module

Working with the Objecteering/Reverse COM module

Overview of working with the Objecteering/Reverse COM module

Complete installation of the *Objecteering/Reverse COM* module includes:

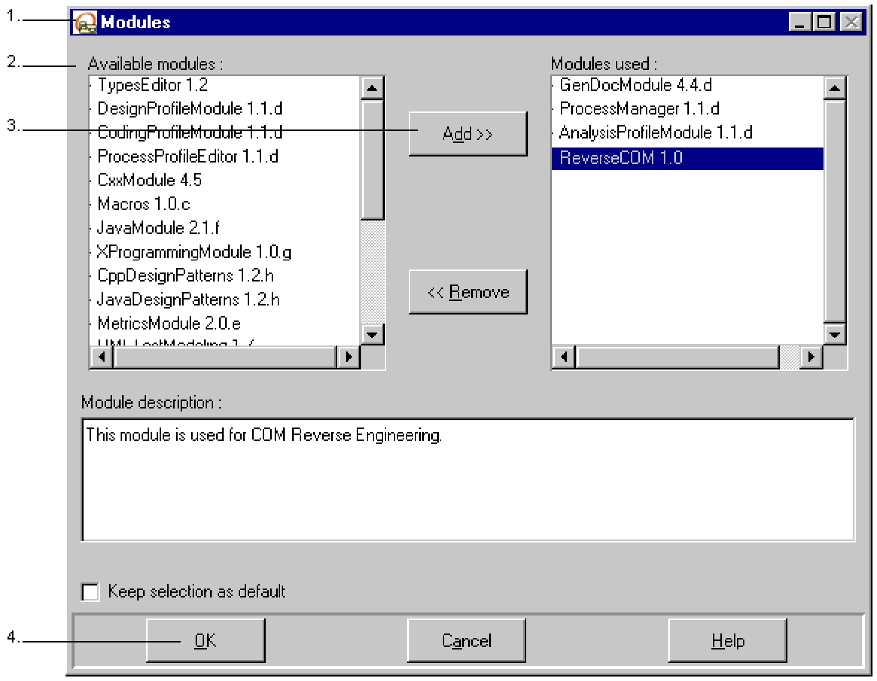
- ♦ delivering the module to your site
- ♦ selecting the module at UML modeling project level
- ♦ selecting the C++ module at UML modeling project level
- ♦ configuring the module

For further information on the selection of the module, please refer to the "*Selecting the Objecteering/Reverse COM module for a UML modeling project*" section in the current chapter of this user guide.

For further details on how to configure the module, please turn to the "*Module parameters*" section in chapter 4 of this user guide.

Selecting the Objecteering/Reverse COM module for a UML modeling project

In order to be able to use the *Objecteering/Reverse COM* module, it must be selected for the current UML modeling project. This selection is made by opening the "*Modules*" dialog box and transferring the *Reverse COM* module from the left-hand "*Modules available*" column into the right-hand "*Modules used*" column (as shown in Figure 2-1).



Available modules :
- TypesEditor 1.2
- DesignProfileModule 1.1.d
- CodingProfileModule 1.1.d
- ProcessProfileEditor 1.1.d
- CxxModule 4.5
- Macros 1.0.c
- JavaModule 2.1.f
- XProgrammingModule 1.0.g
- CppDesignPatterns 1.2.h
- JavaDesignPatterns 1.2.h
- MetricsModule 2.0.e

Modules used :
- GenDocModule 4.4.d
- ProcessManager 1.1.d
- AnalysisProfileModule 1.1.d
- ReverseCOM 1.0

Module description :

This module is used for COM Reverse Engineering.

Figure 2-1. Selecting the *Objecteering/Reverse COM* module

Steps:

1 - Click on the "*UML modeling project modules*" icon to open the "*Modules*" window, or click on the "*Tools/Modules...*" menu.

2 - Click on the *Reverse COM* module in the left-hand "*Modules available*" column.

3 - Click on the "*Add*" button. The module is then transferred into the right-hand "*Modules used*" column.

4 - Confirm by clicking on "*OK*".

Chapter 3: Objecteering/Reverse COM functions

Services available on packages

Making a COM component available in your Objecteering/UML model - phase 1

Once you have delivered the *Objecteering/Reverse COM* module to your site and you have selected it in your UML modeling project, you can now "*import*" any COM component that has been registered on your machine into your Objecteering/UML modeling project, so it can be used as a model element.

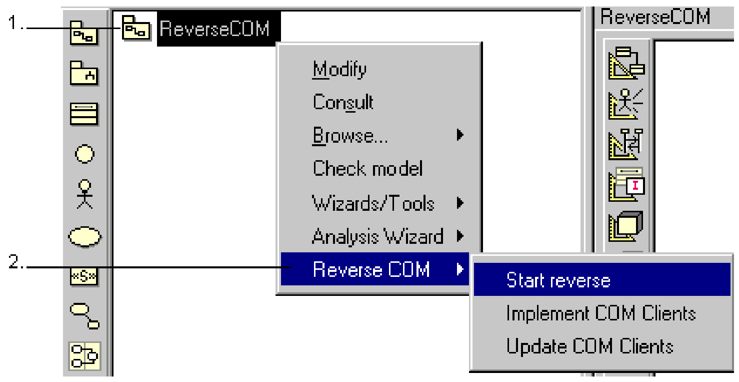


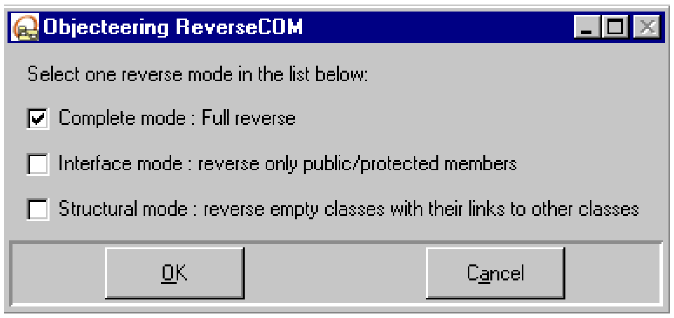
Figure 3-1. Reversing a COM component

Steps:

1 - Right-click on the root package to display the context menu.

2 - Run the "*Reverse COM/Start reverse*" command from the context menu.

When a reverse operation begins, the mode should be selected in the dialog box shown in Figure 3-2.


Objecteering ReverseCOM

Select one reverse mode in the list below:

[✓] Complete mode : Full reverse

[ ] Interface mode : reverse only public/protected members

[ ] Structural mode : reverse empty classes with their links to other classes

Figure 3-2. Selecting the import mode

Once the import mode has been selected, the following dialog boxes appear (as shown in Figure 3-3).

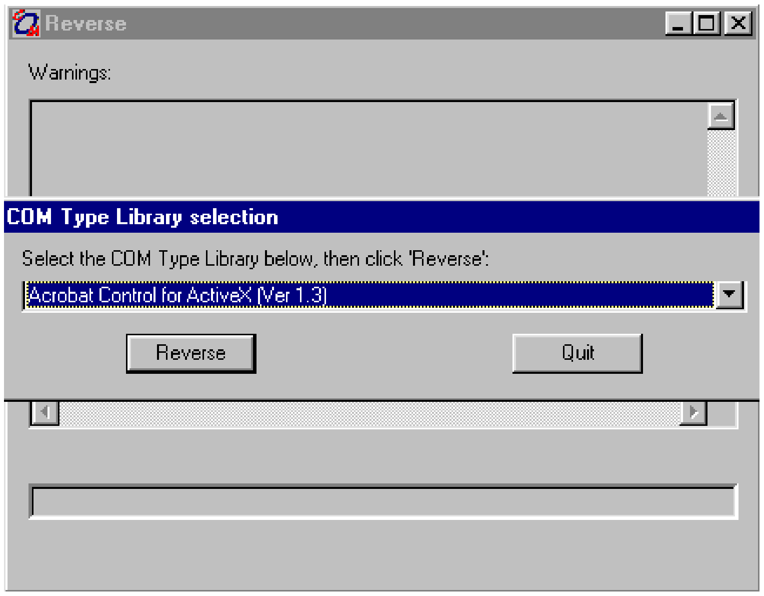

Reverse
Warnings:
COM Type Library selection
Select the COM Type Library below, then click 'Reverse':
Acrobat Control for ActiveX (Ver 1.3)
Reverse
Quit

Figure 3-3. Reverse tool dialog boxes

Open the combo box and select, for example, the item corresponding to Microsoft DAO 3.6 Library (Figure 3-4).

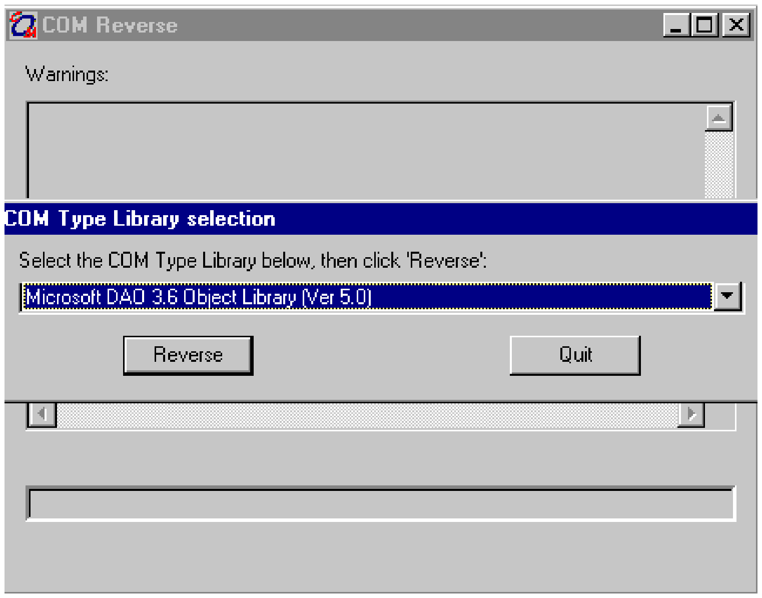
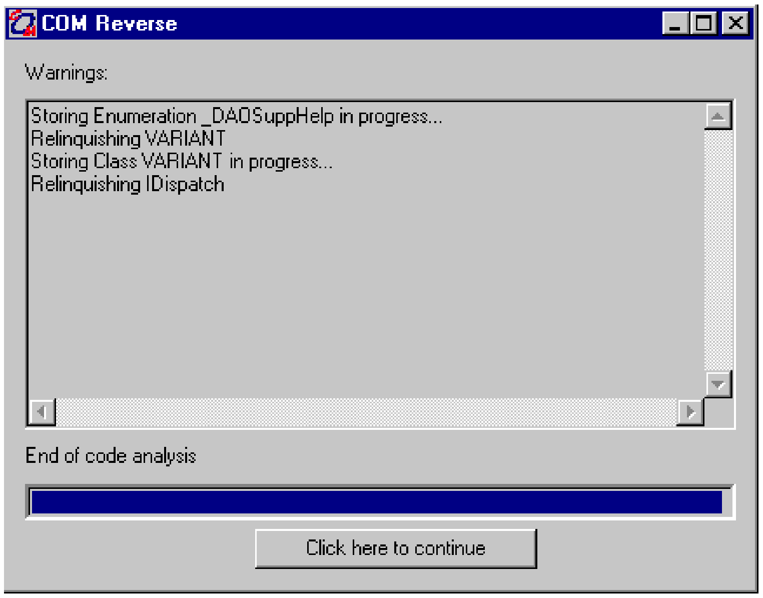

COM Reverse
Warnings:
COM Type Library selection
Select the COM Type Library below, then click 'Reverse':
Microsoft DAO 3.6 Object Library (Ver 5.0)
Reverse
Quit

Figure 3-4. Selecting the COM component

Click on the "*Reverse*" button. A confirmation dialog box then appears. Click on "*OK*" button to confirm.

The component is then analyzed and ASCII files are generated. At the end of analysis, the following window (Figure 3-5) appears:



```
Storing Enumeration _DAOSuppHelp in progress...
Relinquishing VARIANT
Storing Class VARIANT in progress...
Relinquishing IDispatch
```

End of code analysis

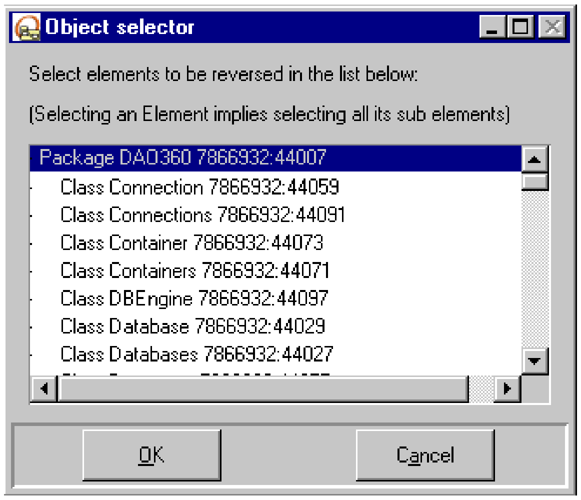
Figure 3-5. End of code analysis

In some cases, as here, you may see certain warnings in the top window. Simply click on the "*Click here to continue*" button, to close this window and view messages in the Objecteering/UML console.

The first phase is now complete, and the second begins.

Making a COM component available in your Objecteering/UML model - phase 2

Before importing elements into Objecteering/UML, the following dialog box (Figure 3-6) appears.



Select elements to be reversed in the list below:

(Selecting an Element implies selecting all its sub elements)

Package DAO360 7866932:44007
Class Connection 7866932:44059
Class Connections 7866932:44091
Class Container 7866932:44073
Class Containers 7866932:44071
Class DBEngine 7866932:44097
Class Database 7866932:44029
Class Databases 7866932:44027

Figure 3-6. Selecting those elements to be imported

After selection, classes from the component are then imported one by one. This can take a while for big components such as Word (which has around 200 classes).

At the end of the import phase, your package contains all the component classes (as shown in Figure 3-7).

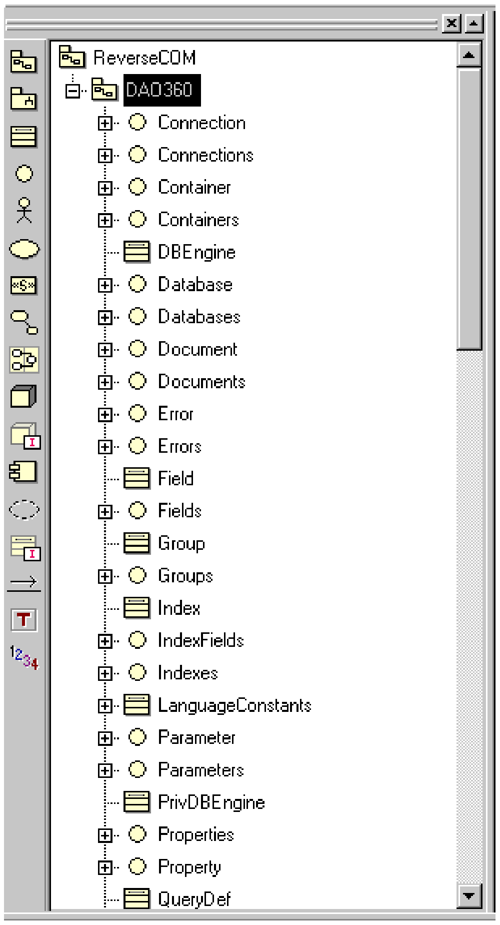


Figure 3-7. Imported classes

Imported objects are Classes, Enumerations and Data Types.

Classes can be divided into five categories:

- "*CoClasses*" are imported as standard classes which implement some interfaces. For example, the "*AddIn*" class in the "*Msdev*" package.
- Dispatch Interfaces are imported as UML interfaces (classes with the <<interface>> stereotype) tagged {dispinterface}. For example, the "*Events*" class in the "*Msdev*" package.
- "basic" interfaces are imported as UML interfaces with no tagged value For example, the "*IExtensibleObject*" class in the "*Msdev*" package.
- Modules are imported as classes stereotyped <<module>>. For example, the "*Constants*" class in the "*Msdev*" package.
- Unions are imported as classes with the {union} tagged value. There is no example of this in the "*Msdev*" package.

To each class, operation, attribute and so on, a "*description*" note is added. The note contains the description of the element, as it is stored in the component.

To classes, enumerates and data types, a {clsid} tagged value is added. It is valued with the CLSID of the element.

The following additional tagged values may be added to these elements.

- {dual}
- {dispatchable}
- {cancreate}
- {appobject}
- {licensed}
- {predeclid}
- {hidden}
- {control}
- {nonextensible}
- {oleautomation}
- {restricted}
- {aggregatable}
- {replaceable}

For examples of such tagged values, have a look at the classes in the "*Msdev*" package.

Class members (attributes, associations, operations) are tagged with a {memid} tagged value with the member identifier. They can also be tagged {bindable}, {source}, {requestedit}, {displaybind}, {defaultbind}, {defaultcollelem}, {nonbrowsable}, {uidefault}, {replaceable}, {immediatebind}, {hidden} and {restricted}.

Operations can be stereotyped either <<PropertyGet>>, <<PropertyPut>> or <<PropertyPutRef>>. They can also be tagged {putref} and {usesgetlasterror}.

Attributes and associations may be tagged {C++TypeExpr} or {readonly}.

Constants are attributes tagged {const}.

Implementing COM clients

Once you have "imported" a COM component into your Objecteering/UML modeling project, you can automatically implement some proxies (or client side parts) of the components' classes.

To do this, you must select the *Objecteering/C++* module and then simply run the "*Reverse COM/Implement COM Clients*" command from the context menu available on your package.

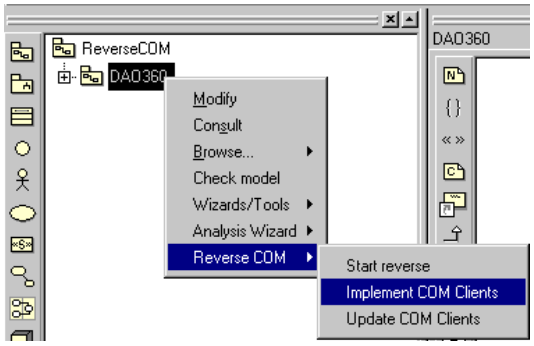


Figure 3-8. Generating COM classes

This operation automatically creates a new package (if it does not already exist) of the same name, suffixed by "*Clients*". For each of the package's dispatch interfaces, a proxy equivalent class with the same name is generated in the client package.

These classes are subclasses of the MFC "*ColeDispathDriver*" class, which provides helpful functions in using COM clients.

Tree default constructors are added to each class, and each operation in the class is implemented by a call to the dispatch methods with the right parameters.

Some kinds of parameters (arrays, for example) cannot be dealt with by the "*ColeDispathDriver*" class. In this case, the corresponding operation is not generated.

You can now generate C++ code for these classes. To do this, consult the *Objecteering/C++* user guide.

Services available on classes

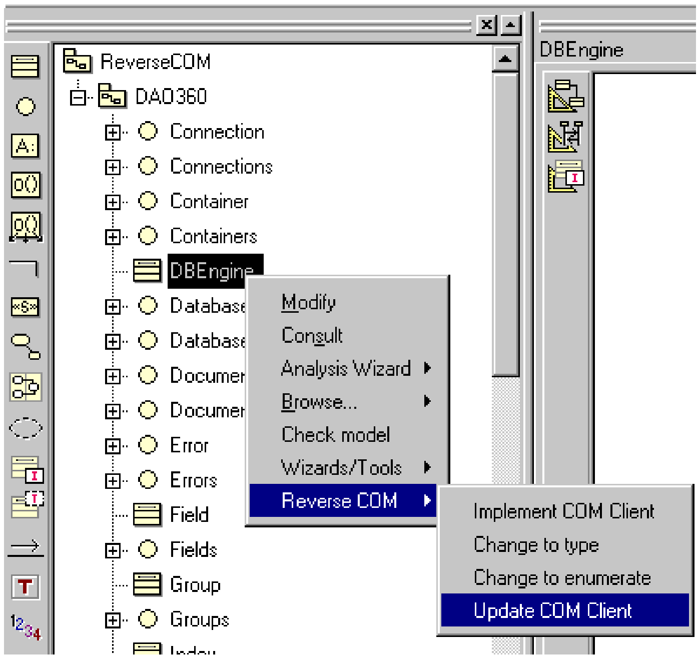


Figure 3-9. Services available on a class

The "*Update COM client*" command is accessible from classes which have already been implemented.

The ... command	triggers ...
Implement COM client	the generation of a proxy equivalent class with the same name in the client package.
Change to type	the transformation of the selected class into a type.
Change to enumerate	the transformation of the selected class into an enumeration.
Update COM client	the re-generation of the proxy equivalent class in the client package.

Chapter 4: Parameterizing the Objecteering/Reverse COM module

Module parameters

When the *Objecteering/ReverseEngineering* module has been selected in your UML modeling project, a certain amount of information must then be defined (as shown in Figure 4-1).

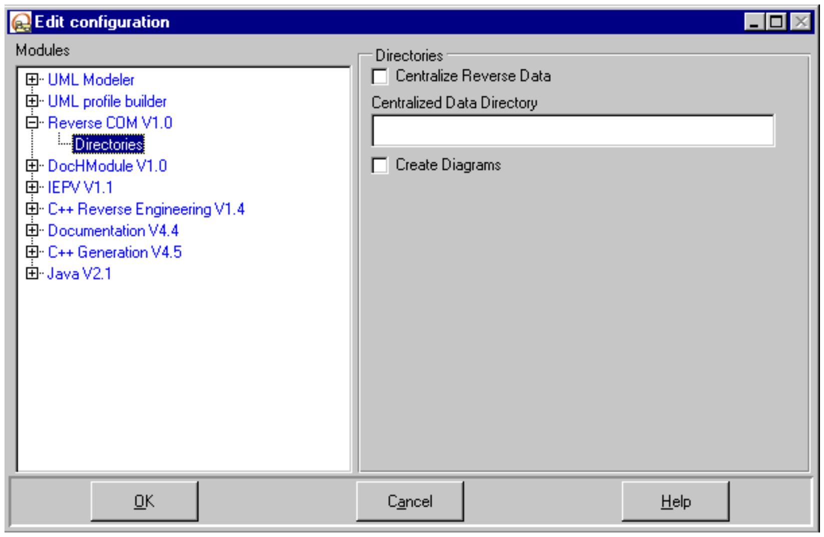


Figure 4-1. Configuring the *Objecteering/Reverse COM* module

Key:

- *Centralize Reverse Data*: This box should be checked, in order to have all reverse data centralized in one directory. If it is not checked, reverse data is stored in a sub-directory of the ReverseCOM directory.
- *Centralized Data Directory*: This is the directory where reverse data is stored, if the "*Centralize Reverse Data*" tickbox is not checked.
- *Create Diagrams*: If this box is checked, reversed class diagrams are created and opened at the end of the reverse procedure.

Index