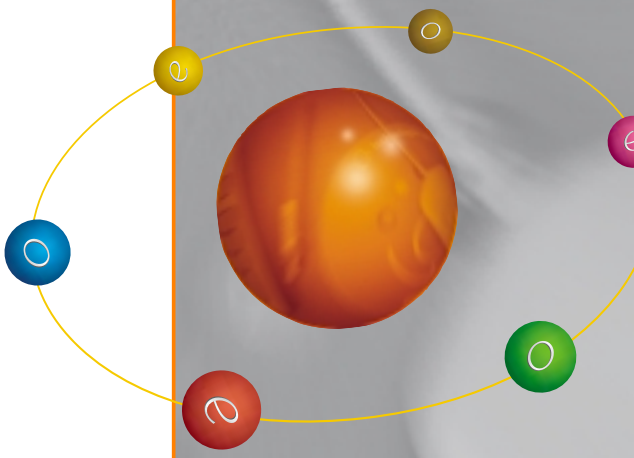
Objecteering/UML

Objecteering/XMI User Guide

Version 5.2.2



Objecteering

Software

www.objecteering.com

Taking object development one step further

Objecteering/UML version 5.2.2 - CODOBJ 001/001

Contents

Chapter 1: Objecteering/XMI general presentation

Introduction

Overview

Welcome to the *Objecteering/XMI* user guide!

Two versions of *Objecteering/XMI* are now available to users:

- the *Objecteering/XMI module*, which is used to generate and re-read XMI files
- the *Objecteering/XMI module import*, which is only used to re-read XMI files

The file format is the standard UML format for exchanging models between different tools. XMI exchange is based on the OMG UML 1.3 and UML 1.4 standards.

Functions

The *Objecteering/XMI module* has three main functions:

- the generation of XMI files from an Objecteering/UML model
- the creation of an Objecteering/UML model from an XMI file
- the update of an Objecteering/UML model from an XMI file (not available in the Personal Edition of Objecteering/UML)

The *Objecteering/XMI module import* has two main functions:

- the creation of an Objecteering/UML model from an XMI file
- the update of an Objecteering/UML model from an XMI file (not available in the Personal Edition of Objecteering/UML)

XMI file generation

Generalities

XMI file generation operations consist of producing XMI files for the packages of a model. These operations take into account all the metaclasses used for static diagrams, use case diagrams, state diagrams, sequence diagrams, object diagrams and activity diagrams.

XMI file generation is only available with the *Objecteering/XMI module*, and not with *Objecteering/XMI module import.*

Generating types

The XMI file generator incorporates into the XMI file all types which are pre-defined in Objecteering/UML. They are declared as DataType type objects. Pre-defined types are as follows:

- boolean
- char
- integer
- real
- string

Strategy chosen for generation

During the generation of XMI files, links can exist towards elements which will not be integrated into the XMI file (for example, an association or a generalization towards a class belonging to a non-exported package). In this case, the link is ignored. This link can be a generalization link, an association or a dependency.

XMI file generation and model consistency checks

Files may be generated from the UML model regardless of whether consistency checks are active or inactive. However, when generation is launched, a message informs the user that he is in the process of generating code on a model which may potentially not conform to the UML modeling rules checked by Objecteering/UML (as shown in Figure 1-1). This message is only displayed when consistency checks have been removed.

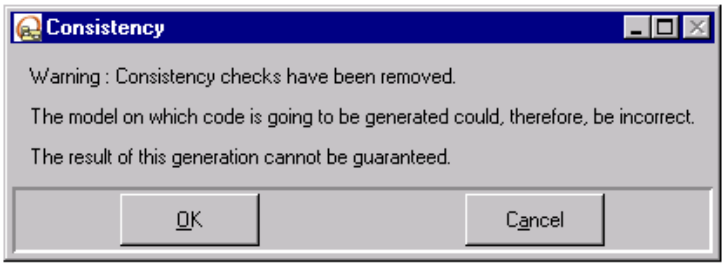


Figure 1-1. Message informing the user that consistency checks have been removed

Note: It should be noted that code generation in command line mode (please see objingcl) is assured, whatever the state of the consistency checks at the time of code generation.

XMI file reverse

Generalities

The XMI file reverse operation does the opposite of the generation operation. It creates an Objecteering/UML model from an XMI model. All the metaclasses used for static diagrams, use case diagrams, state diagrams, sequence diagrams, object diagrams and activity diagrams are handled.

Reverse and model consistency checks

A model can be reversed from files external to Objecteering/UML regardless of whether consistency checks are active or inactive. However, when the reverse is launched, a message informs the user that consistency checks are active and that construction of the model, which may potentially not conform to the UML modeling rules checked by Objecteering/UML, may be refused (as shown in Figure 1-2).

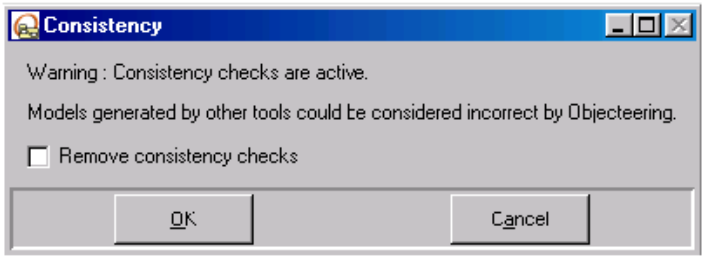


Figure 1-2. Message informing the user that consistency checks are active

If the user wishes to deactivate consistency checks, he should simply check the "*Remove consistency checks*" tickbox.

Note: It should be noted that code reversal in command line mode (please see objingcl) is ensured, whatever the state of the consistency checks when the reverse command is run.

Restrictions

Three versions of Objecteering/UML exist: the Personal Edition, the Professional Edition and the Enterprise Edition. The reverse strategy is different according to the Objecteering/UML edition you have. The principal limitations are as follows:

- There is no exchange by identifier, as the universal identification mechanism is not used.
- There is no exchange of diagrams, except for sequence diagrams and object diagrams.
- There is no update import on Personal Edition versions. The reverse command can only, therefore, be run on the UML model root.

XMI generation and reverse capabilities

Introduction

The *Objecteering/XMI* module is able to generate and reverse the elements contained in six kinds of diagram: static diagrams, use case diagrams, state diagrams, sequence diagrams, object diagrams and activity diagrams. However, in some cases, there may be differences between the model and its XMI representation. The reverse process may also be incomplete, depending on the differences between the UML metamodel and the Objecteering/UML metamodel.

The use case model

The use case model is almost complete, given that the only part not generated or reversed is the ExtensionPoint. ExtensionPoints do not exist in Objecteering/UML and therefore can neither be generated nor reversed.

The state machine model

The state machine model is the most complicated, because there are several differences between the UML metamodel and the Objecteering/UML metamodel.

In Objecteering/UML, an event belongs to a StateMachine, whilst in UML it belongs to a NameSpace. Generation will, therefore, attach an event to the owner of the StateMachine. The opposite process is impossible during reverse, so an XMI extension has been added to Event, so as to allow a parent StateMachine to be referenced. As regards the import of XMI files not generated by Objecteering/UML, Events are attached to the first StateMachine of the owner.

The sequence model

The entire sequence diagram is imported, except for two pieces of information which are lost: the fact that a message is a fork, and the asynchronous nature of a message.

The particularity of a sequence diagram is that the diagram's graphics are automatically generated, which means that messages linked to the sequence diagram are linked.

The object model

The object diagram is completely imported, except for instance embedding which is lost in XMI 1.0/UML 1.3.

As with the sequence diagram, object diagram graphics are automatically generated, which means that messages attached to AssociationOccurences can be seen.

The activity model

The activity model encounters the same Event management problems as the state machine model. The same solutions are used.

Tagged values

Since Objecteering/UML's tagged values may contain more than one value, the generation process will concatenate this value in one single string, using "\objing;" as the separator.

Notes

It should be noted that note indentation may be lost during generation and reverse operations.

Stereotypes

Objecteering/UML stereotypes are imported. However, other stereotypes are imported in the form of an {*xmiStereotype*} tagged value, with the following parameters:

- the name of the stereotype
- its icon
- its base class

Installing Objecteering/XMI module and Objecteering/XMI module import

Objecteering/XMI module import

Objecteering/XMI module import version 2.1 or later is delivered as standard with Objecteering/UML version 5.2.1 or later. No particular installation procedure is necessary.

Objecteering/XMI module import is installed in the $OBJING_PATH/modules/XMIModuleImport directory.

Objecteering/XMI module

Objecteering/XMI module is subject to a valid *Objecteering/UML* license.

Objecteering/XMI module is installed in the $OBJING_PATH/modules/XMIModule directory. The contents of this directory are as follows:

- bin
- doc
- FirstStepsContainer
- res
- tmp

Using Objecteering/XMI module and Objecteering/XMI module import

Introduction

This section will detail the operations necessary for the *Objecteering/XMI* module to function in a UML modeling project. The *Objecteering/XMI module import* functions in the same way, except that only the import command is available.

Creating a new UML modeling project

Details on how to create a UML modeling project are given in the "*Creating a new UML modeling project*" section in chapter 3 of the *Objecteering/Introduction* user guide.

Selecting the XMI module for the newly created UML modeling project

Details on how to select a module are provided in the "*Selecting modules in the current UML modeling project*" section in chapter 3 of the *Objecteering/Introduction* user guide.

XMI glossary

DTD: Document Type Definition. A DTD is a dictionary written in XML format and used to define markers allowed in an XMI file, as well as their hierarchy. It is also used to express constraints in the order in which they appear, due to the necessity of defining certain values and attributing default values. It can be considered as a sort of grammar.

Enterprise Edition: This is the complete version of Objecteering/UML.

Personal Edition: This edition of Objecteering/UML does not contain group work services.

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

XMI: XML-based Metadata Interchange. This is a file format based on XML and used to exchange UML models. An XMI file references a DTD, corresponding to a UML metamodel of a given version.

XML: eXtensible Markup Language. This is a marked file format, defined by the World Wide Web Consortium, and used to transport all kinds of data. It is the standard defined in 1997 with the aim of facilitating the exchange of metadata.

Chapter 2: XMI First Steps

XMI first steps - Introduction

Introduction

This first steps UML modeling project will allow you to become familiar with the different functions of the *Objecteering/XMI* module, step by step.

Sources

This example is a simple bank account application, extracted from *Learn Java Now*, by Stephen R Davis, Microsoft Press.

Importing the first steps UML modeling project

A first steps UML modeling project is delivered as standard with the *Objecteering/XMI* module. It is located in *$OBJING_PATH/modules/XMIModule/FirstStepsContainer*.

Carry out the following steps:

1 - Create a new UML modeling project (named, for example, "*FirstSteps*").

2 - Select the XMI module for this newly created UML modeling project.

3 - Import the contents of the "*First Steps*" UML modeling project into the UML modeling project you have just created (as shown in Figure 2-1).

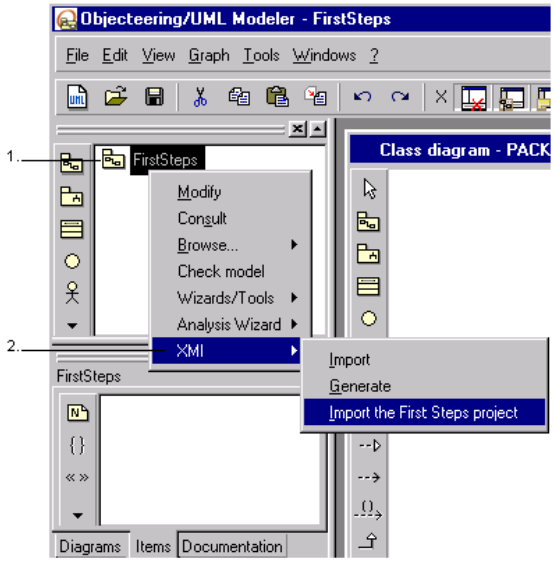


Figure 2-1. Importing the First Steps UML modeling project

Steps:

1 - Select the UML model root by right-clicking.

2 - Select the "*XMI/Import the First Steps project*" command from the context menu which then appears.

Generating XMI code

Triggering XMI code generation

An XMI-type file is generated for each package of the model on which the "*Generate*" command is called.

In this way, it is possible to generate code from a package. An XMI file is then generated for each of the packages contained in it.

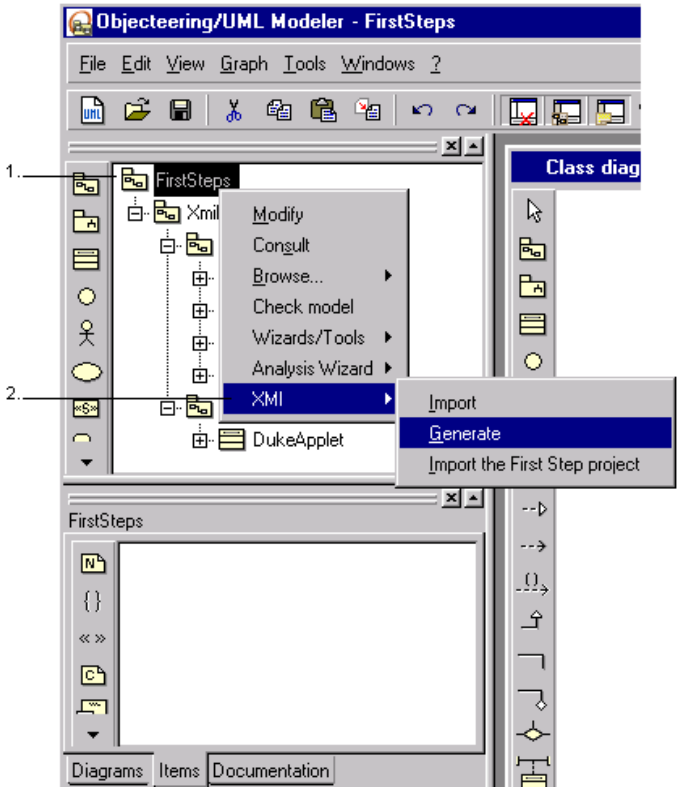


Figure 2-2. The "*Generate*" command

The dialog box shown in Figure 2-3 then appears.

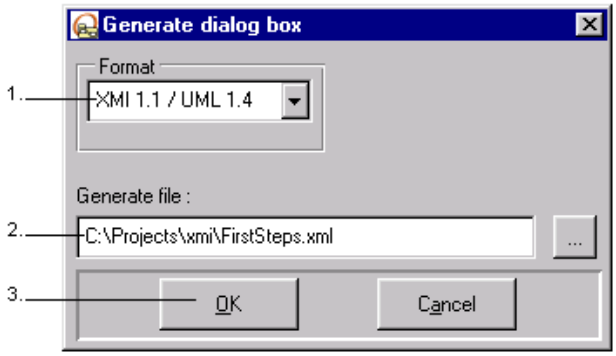


Figure 2-3. The dialog box requesting generation information

Steps:

1 - Enter the XMI format you are working with.

2 - Enter the generation file, or use the  icon to open a file browser, through which you may select your generation file.

3 - Click on "*OK*" to confirm.

Generation is then run.

Reversing a model from an XMI file

Introduction

The reverse operation is used to build a UML model from an XMI file. The aim of this is to retrieve a model realized, for example, using another tool, or to update a model modified on another site.

We are now going to proceed by reversing the entire "*FirstSteps*" package we have just generated.

Launching the command

For the "*Reverse*" command, carry out the following steps.

1 - First, create a new UML modeling project (for example, "*FirstStepsXMIReverse*"), and select the *Objecteering/XMI* module.

2 - Next, carry out the steps shown in Figure 2-4.

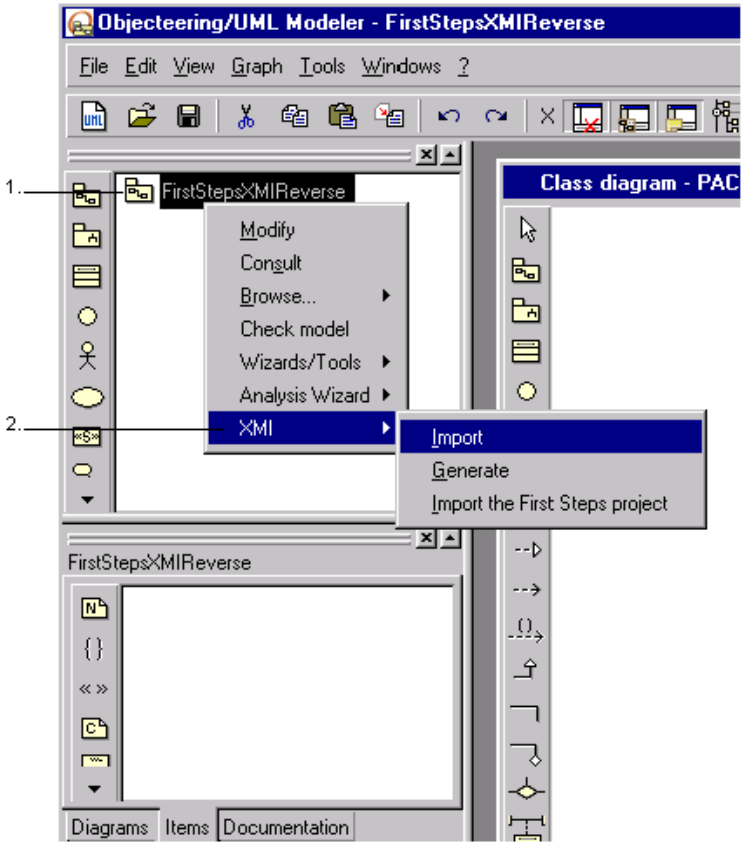


Figure 2-4. The "*Import*" command

Steps:

1 - Select the "*FirstStepsXMIReverse*" package by right-clicking. The context menu then appears.

2 - Run the "*XMI/Import*" commands.

A dialog box asking you to select the .xml file in question then appears (as shown in Figure 2-5).

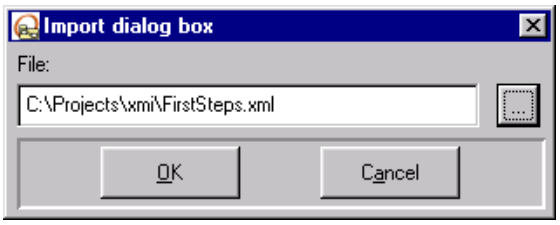


Figure 2-5. Locating the file to import

Steps:

1 - Simply indicate the .xml file you wish to import into the newly created UML modeling project, and confirm by clicking on "*OK*". You can also use the  icon to open a file browser, through which you can select the file.

Note: A dialog box regarding consistency checks then appears. Click on "*OK*" to confirm.

Result

The "*FirstSteps*" UML modeling project previously created has been imported.

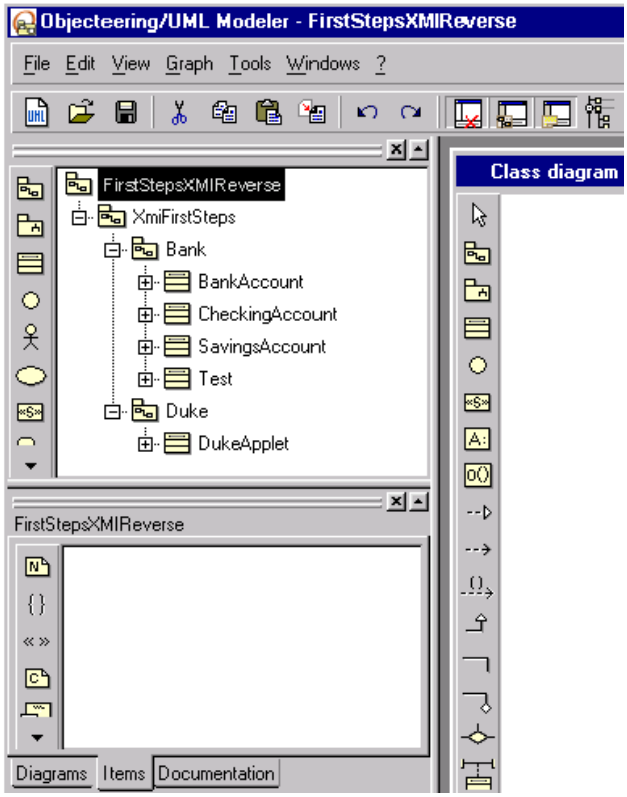


Figure 2-6. Result of the XMI import

Rename the imported "*XmiFirstSteps*" package "*GenFirstSteps*", and run the "*XMI/Import the first steps project*" command on the UML model root ("*FirstStepsXMIReverse*").

Result

A package named "*XmiFirstSteps*" appears under the UML model root. This package contains the complete "*GenFirstSteps*" package. You can go ahead and compare them.

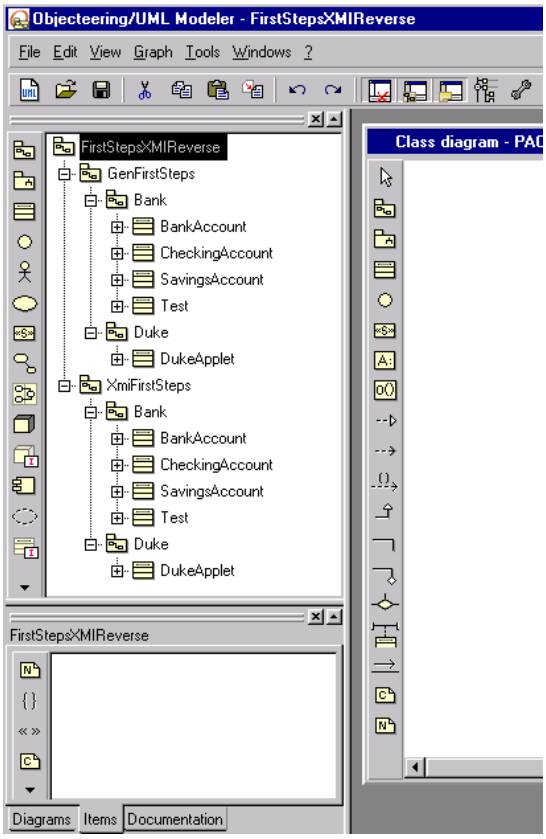


Figure 2-7. Result after running the "*XMI/Import the first steps project*" command on the UML model root

Chapter 3: XMI generation and reverse

Generating an XMI file

Overview

This operation concerns the generation of XMI files from a model built in Objecteering/UML. This generation is based on the DTD used and provided with Objecteering/UML. This DTD is the final XMI DTD, and conforms to the UML 1.3 or UML 1.4 standard, according to your generation choices

Generation dialog box fields

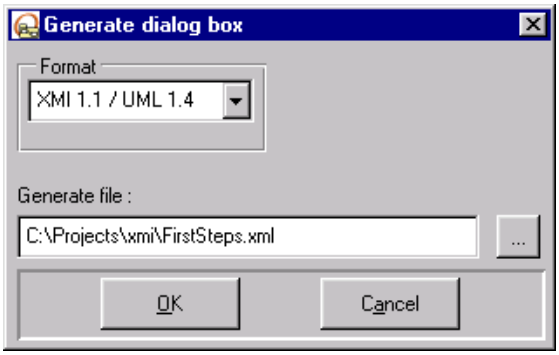


Figure 3-1. The generate dialog box

Key:

- The "*Format*" field is used to select the version of XMI you are using (either XMI1.0 with UML 1.3 or XMI1.1 with UML 1.4).
- The "*Generate file*" field is used to indicate the location of the generated file.

Strategy used during XMI file generation

During file generation, the DTD (Document Type Definition) file is copied to the same location as the generated XML file. An XML file has no value without its DTD.

In the case of generation using the UML 1.3 standard, the DTD is named uml13.dtd, whereas for generation using the UML 1.4 standard, the DTD is named UML14.dtd.

Restrictions

During the generation of XMI files, links directed towards elements not integrated into the XMI file can exist (this is the case for associations and generalization links directed towards a class of a non-exported package). If this is the case, the link is ignored. This link can be a generalization link, an association or a dependency.

Reversing an XMI file

Presentation

The biggest advantage of the XMI standard is that it allows the transfer of UML models between different modeling tools. It is necessary, therefore, to be able to import XMI files into Objecteering/UML, which is what is carried out by the XMI reverse command.

This operation is translated by the reconstruction of the modeling of classes and packages from the available XMI files.

Reverse dialog box fields

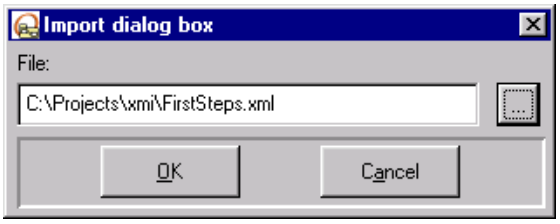


Figure 3-2. The reverse dialog box

Key:

- The *"File"* field indicates the name of the source XMI file, as well as its complete path.

Strategy used in the reverse

The reverse strategy employed varies according to the version of Objecteering/UML used.

Personal edition

This edition cannot be used for group work. In this version, XMI does not provide a model update service.

In the personal edition, the reverse operation destroys all elements of the model into which the user imports. It then creates new elements from the data present in the XMI file.

The reverse command must be used only on the UML model root.

Enterprise edition

Models are compared on the basis of symbols (class names, package names, etc.), and not on the basis of universal identifiers as is the case with the Objecteering/UML import service.

In the enterprise edition, the reverse updates the existing model from the data present in the file. If an element already exists, it is also updated (that is to say, its attributes and links are updated). If the element does not exist, it is created. Finally, if an element present in Objecteering/UML is no longer present in the XMI file, it is destroyed.

Special case

Generally, objects created from XMI files are created as components of the package from which the reverse has been launched. However, during the import of a model type object from a root package, the root package and this object are merged.

DTD file

An XMI file is obsolete without its DTD file. Before running the import command, you should check that the DTD file is indeed present at the location indicated in the XMI file. In most cases, and for files resulting from Objecteering/UML generation, the DTD file is in the same location as the XML file.

Restrictions

A class created by reverse receives an identifier in the same way as an object created in an explorer or a graphic editor.

If two people reverse the same class in two different UML modeling projects, Objecteering/UML considers there to be two different objects.

In order for the Objecteering/UML import not to lose links directed towards reversed classes, the XMI reverse must be carried out in a reference project from which everyone will import the classes they use.

Chapter 4: Calling XMI module commands

Calling XMI module commands on line

Overview

Module commands can be launched through a command line using the *objingcl* executable, delivered as standard with Objecteering/UML.

Calling commands

The generation command is called on-line using an instruction whose syntax is as follows:

```
objingcl  -db <database_name>
          -prj <project_name>
          -mdl XMIModule
          -cmd
          default#external#Code#XMI#GenXMI#generate
          <metaclass>::<object_name> --
<Format><Generate_file>
```

The "*Format*" parameter is used to define the XMI generation format (XMI1.0/UML1.3 or XMI1.1/UML1.4).

The "*Generate_file*" parameter is used to indicate the location of the generated file.

The reverse command is called on-line using an instruction whose syntax is as follows:

```
objingcl  -db <database_name>
          -prj <project_name>
          -mdl XMIModule
          -cmd
          default#external#Code#XMI#RevXMI#reverse
          <metaclass>::<object_name> -- <Reverse_file>
```

The "*Reverse_file*" parameter is used to specify the file which is to be imported.

Note: <object_name> must be a complete name of the Namespace1::NameSpace2 type.

Commands which can be invoked

Command ...	Metaclass ...	Action ...
generate	Package	generation of an XMI file containing the package and the entire package sub-hierarchy
reverse	Package	reverse of an XMI file

Chapter 5: Parameterizing the Objecteering/XMI and Objecteering/XMI module import modules

Defining module parameters

Configuring the module

The *Objecteering/Multi-user* module can be parameterized through the "*Modifying configuration*" dialog box, which is launched by clicking on the  "*Modify module parameter configuration* " icon.

Objecteering/XMI parameter sets

The "General options" parameter set

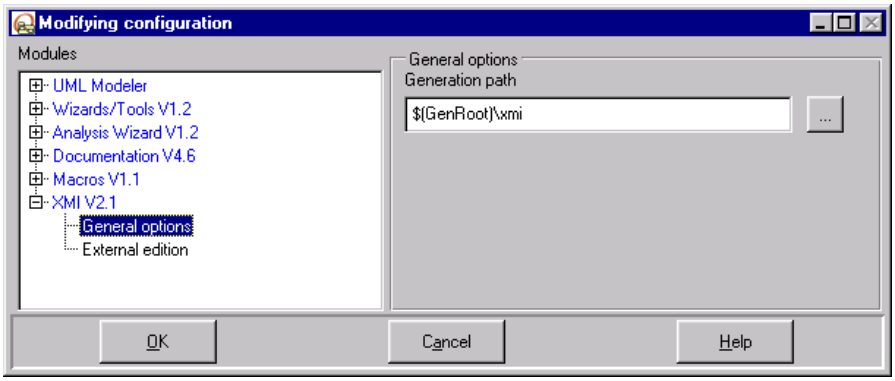


Figure 5-1. The "*General options*" parameter set

The ... parameter	is used to ...
Generation path	indicate the directory in which XMI files will be generated.

The "External edition" parameter set

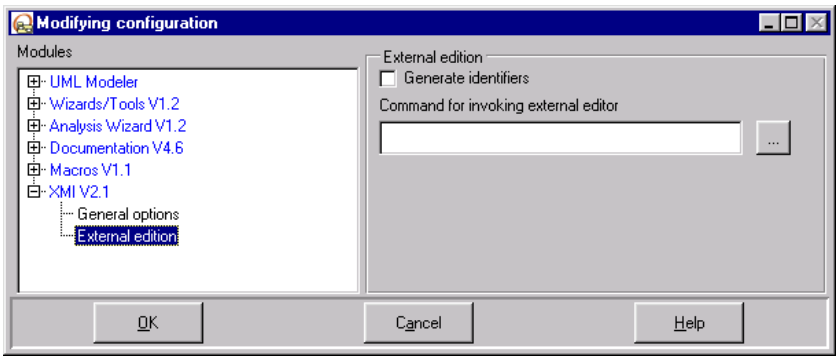


Figure 5-2. The "*External edition*" parameter set

The ... parameter	is used to ...
Generate identifiers	insert markers into the generated code files. We recommend that it remain selected, in order to take advantage of the code/model consistency management system.
Command for invoking external editor	the command used to launch an editor to modify the generated code.

Note: The "*Generate identifiers*" and "*Command for invoking external editor*" parameters are not used by the *Objecteering/XMI* module.

Objecteering/XMI module import parameter sets

The "General options" parameter set

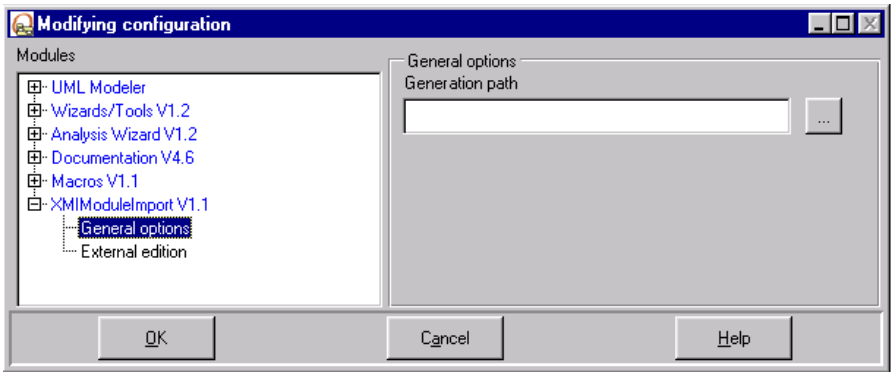


Figure 5-3. The "*General options*" parameter set

The ... parameter	is used to ...
Generation path	indicate the directory in which XMI files will be generated.

The "External edition" parameter set

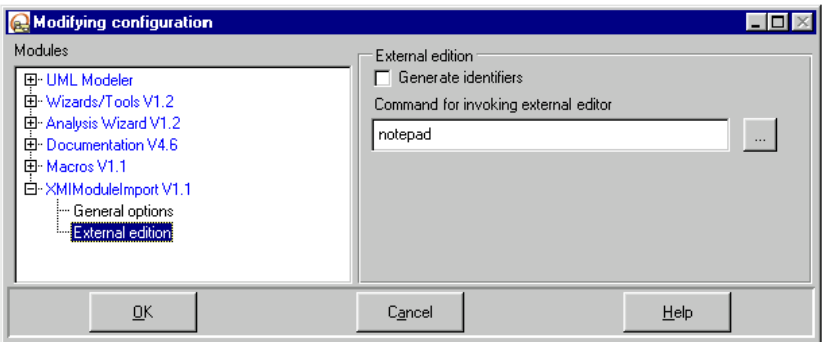


Figure 5-4. The "*External edition*" parameter set

The ... parameter	is used to ...
Generate identifiers	insert markers into the generated code files. We recommend that it remain selected, in order to take advantage of the code/model consistency management system.
Command for invoking external editor	the command used to launch an editor to modify the generated code.

Note: The "*Generate identifiers*" and "*Command for invoking external editor*" parameters are not used by the *Objecteering/XMI module import* module.

Index