LIANTIS GmbH & Co KG
St.-Anton-Straße 69 – 71
47798 Krefeld

Phone +49 2151 1540893
Fax +49 2151 1540894

info@liantis.com
www.liantis.com

User Manual

Octopus/OCL Plugin for Magic Draw UML

Trademarks

UML™ is a trademark of the Object Management Group.

Magic Draw™ is a trademark of No Magic Inc.

Table of content

1 Introduction

This manual describes how to use the Octopus/OCL Plugin for Magic Draw UML. This plugin enables you to do the following:

- Create a UML2 class diagram with Magic Draw UML
- Export this UML2 model to .uml files for Octopus/OCL (by using the provided plugin)
- Refresh your Octopus Eclipse project to import your model
- Use the Octopus OCL editor to edit OCL2 constraints for your model. The editor will not only do a syntax check but also a check if the constrained elements exit in the model.

1.1 Why not use the build-in Magic Draw OCL Editor?

Currently the Magic Draw OCL editor does not support OCL 2.0 (Magic Draw Version 11). The next mayor version of Magic Draw will greatly improve OCL support.

I will prefer to use this plugin even in the future for the following reason:

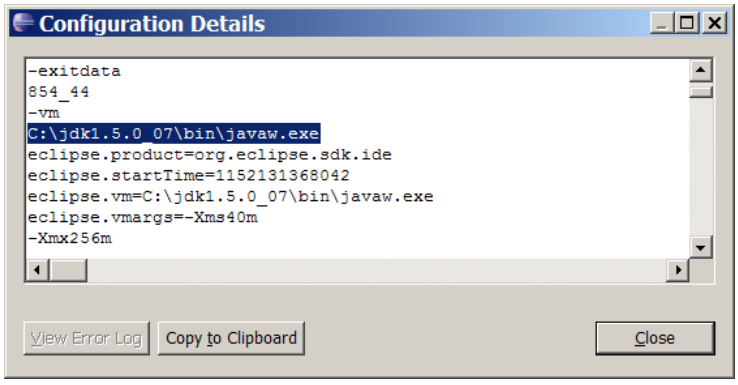
Using OCL2 in a real world project results in a large amount of constraints. Editing, organising understanding and refactoring all these constrains in a text editor is much easier than doing this directly in the UML model. Displaying OCL constraints in the class diagram just makes sense for PowerPoint™ software architects.

1.2 Required software

- Magic Draw UML 10 or better (a free community edition is available from the No Magic website http://www.nomagic.com).
- Java Virtual Machine with support for Java 1.5 or better (http://www.javasoft.com).
- Eclipse 3.1 or better running with the above Java VM (http://www.eclipse.org).
- The Octopus/OCL plugin for Eclipse 2.2.0 or better (http://forgeforge.net/projects/octopus).
- The Octopus/OCL plugin for Magic Draw UML (http://www.szallies-informatik.de/sware/ocl/).

1.3 Installation

- Install Magic Draw UML.
- Copy the Octopus/OCL plugin for Magic Draw into the <installdir>/plugin directory. Start Magic Draw. The menu bar should contain an entry “Octopus/OCL”.
- Install the Java VM.
- Install Eclipse. Start Eclipse. Check if Eclipse uses the correct VM by selecting “Help”/”About Eclipse SDK” in the menu. Click the “Configuration details” button. The parameter “vm” shows the used VM.



- Copy the files of the Octopus Plugin to the Eclipse Folder. Restart Eclipse.

1.4 Support

Don't hesitate to contact me: constantin.szallies@liantis.com

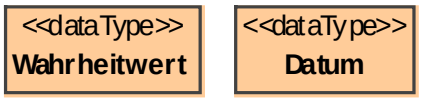
2 Getting started

The following sections provide an overview of using Magic Draw with Octopus/OCL.

2.1 Magic Draw: Define your type system

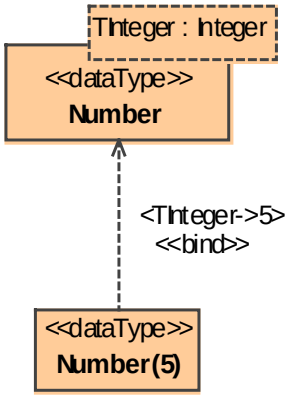
All your used primitive types (like Date, String, Boolean etc) must be modelled using datatypes or primitives. It's not possible to use any types defined in the "UML standard profile" package of Magic Draw.

For example create the following datatypes:



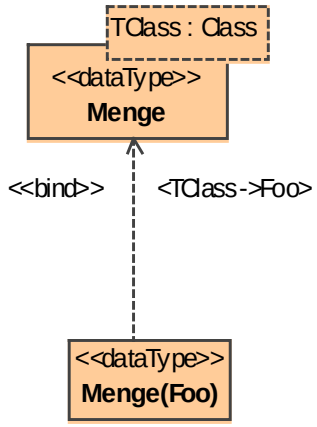
"Wahrheitswert" is the German translation for "Boolean" and "Datum" means "Date".

It's possible to define template datatypes with template parameters:



This defines an "abstract" template data type "Number" that can be instantiated by providing a concrete value for the template parameter in a binding dependency. For example "Number(5)" is a number with at most 5 digits.

It's also possible to define template datatypes for object containers:

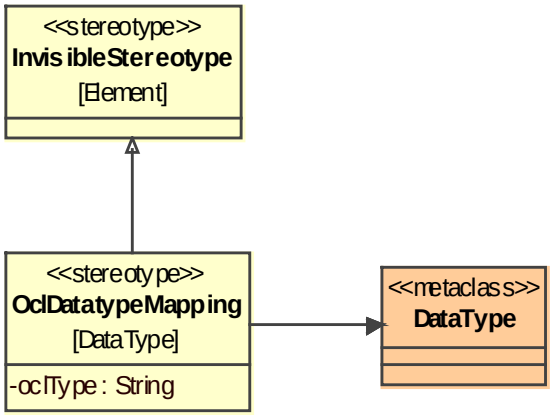


"Menge" is German for "Set" and "Menge(Foo)" defines a data type that is a set of Foo objects.

2.2 Magic Draw: Define an OCL type mapping

OCL defines standard datatypes like "Integer", "Real" and "String" and the four container types "Set", "OrderedSet", "Bag" and "Sequence".

You can define a mapping from your own type system used in the UML model to the OCL type system. To do this, you first have to add a profile to your project. In this profile, you have to define a stereotype called "OclDatatypeMapping" with a property "oclType : String"[1]:



The stereotype "OclDatatypeMapping" can be applied to datatypes and inherits from "InvisibleStereotype".

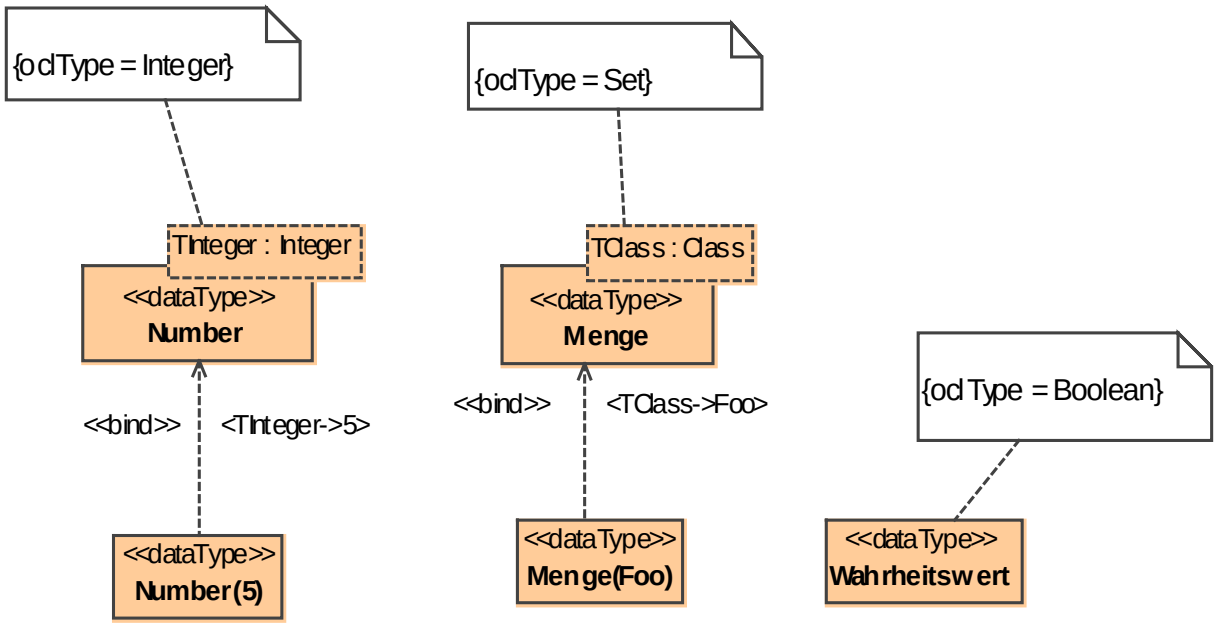
"InvisibleStereotype" is a stereotype defined in "UML Standard Profile::Magic Draw Profile". Inheriting from this stereotype has the effect that the stereotype's name is not displayed class diagrams. This is reasonable because the "OclDatatypeMapping" stereotype is only used for mapping purposes and has no business meaning.

To map "Wahrheitswert" to OCL "Boolean" apply the stereotype "OclDatatypeMapping" to "Wahrheitswert" and add a value for the property "oclType".

---

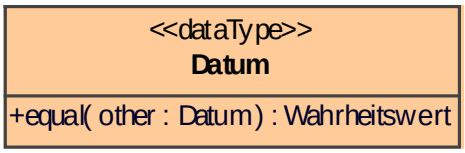
[1] Note that you have to use the types defined in "UML Standard Profile::UML Standard Profile::datatypes' for stereotype properties.

The following diagram shows an example of a type mapping.



OCL does not define a standard type for dates, therefore no type mapping is applied.

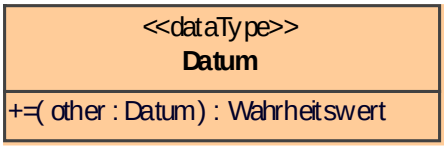
The type "Datum" can be used unaltered in OCL constraints. In this case, you can define your own operations. Let's say in your OCL constraints you need to check if two dates are equal:



You can use this operation in your constraints:

```
inv:
      bagOf->forAll(d : types::Datum | d.equal(d))
```

It would be nicer to use the equal sign instead of "equal". Just change the name of the operation:



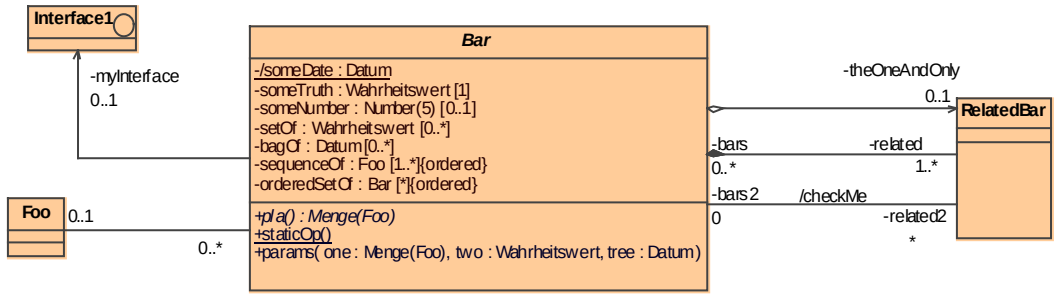
The new constraint looks like this:

```
inv:
      bagOf->forAll(d : types::Datum | d = d )
```

Defining + - / < > <= >= = <> * as operation names always results in an infix operation.

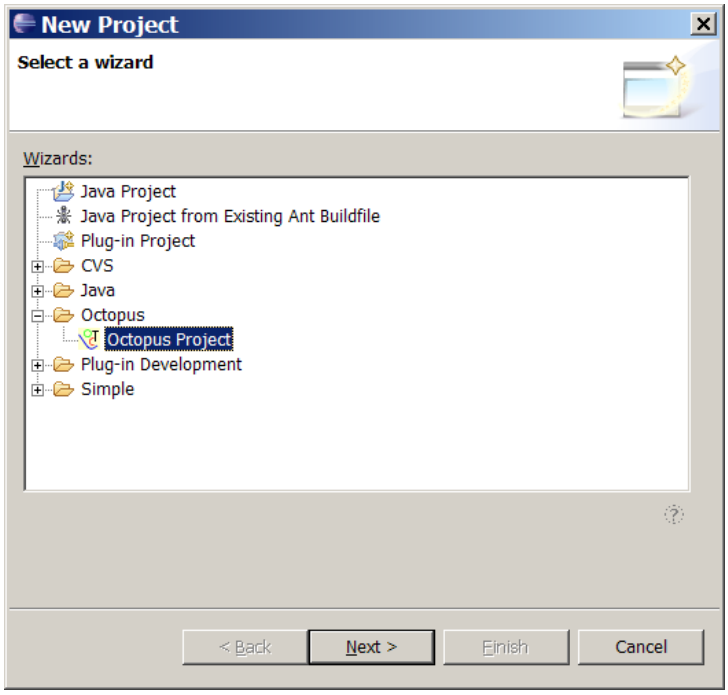
2.3 Magic Draw: Define your class model

Create a class model like the following:



2.4 Eclipse: Create an Octopus/OCL project

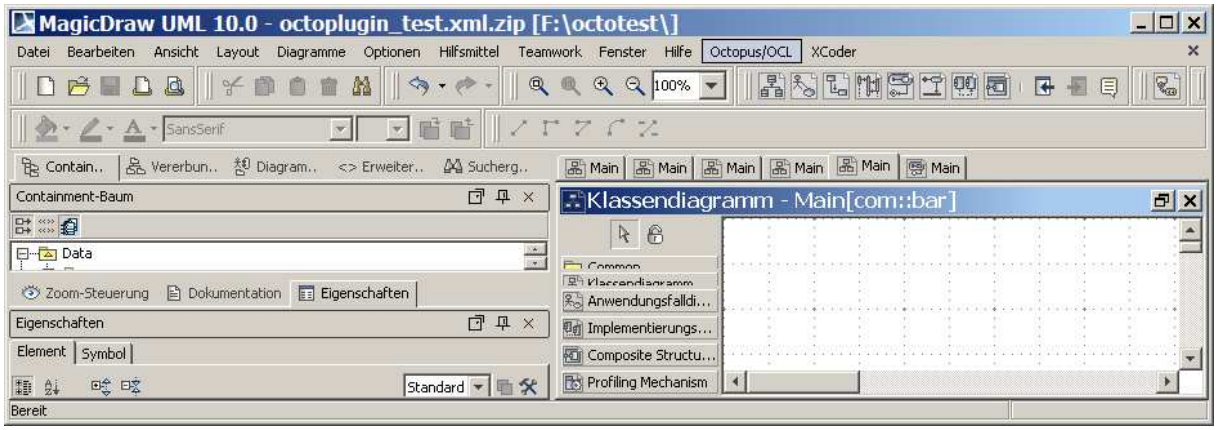
Start Eclipse and create a new Octopus project by selecting "File/New/Project…" from the menu. Select "Octopus/Octopus Project".



Click on "Next >", enter the project name "octotest" and "Finish".

2.5 Magic Draw: Export your class model

In Magic Draw select "Octopus/OCL" and "Export" from the menu.



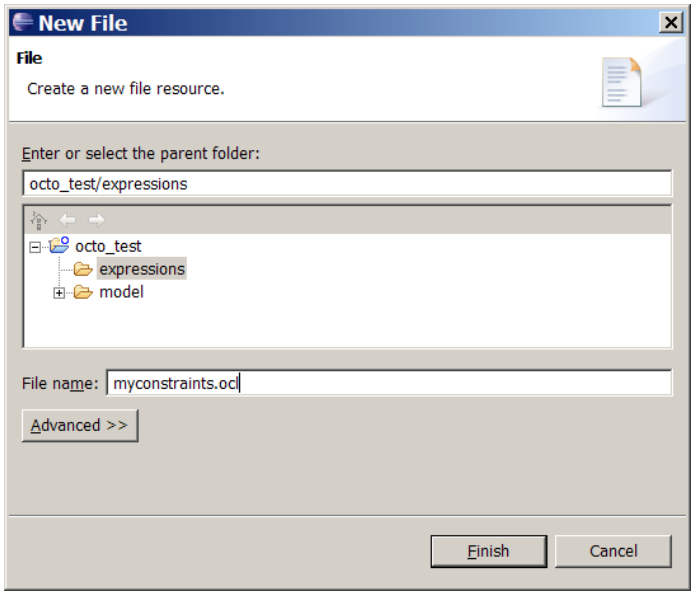
A file chooser dialog appears. Select the model directory of your Eclipse Octopus project (<workspace>/octotest/model).

2.6 Eclipse: Load class model

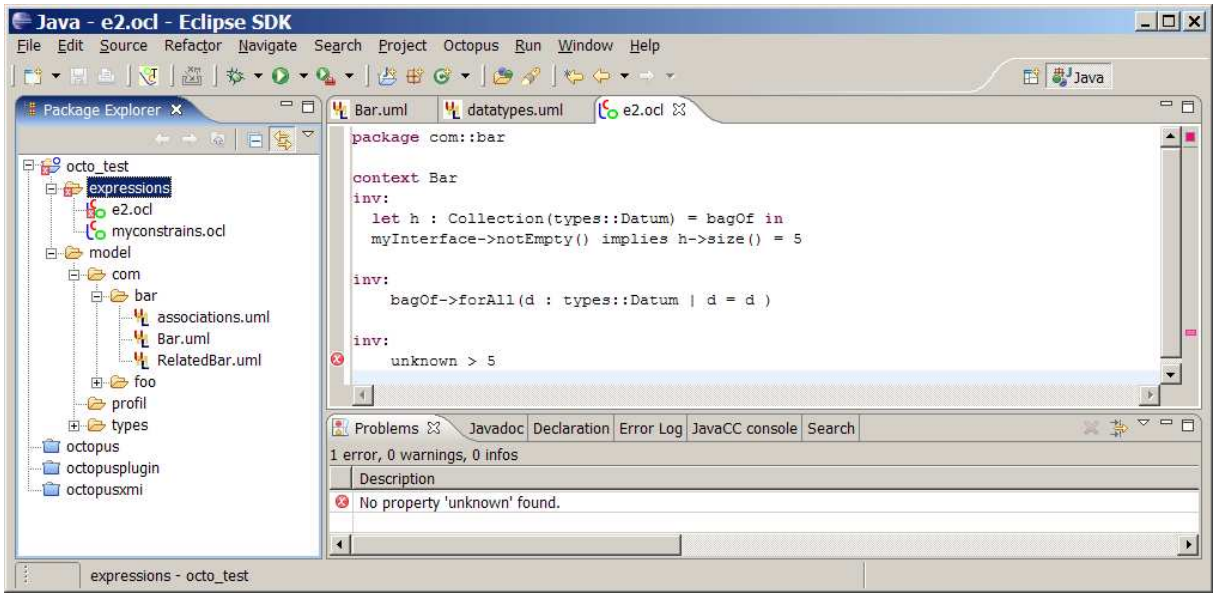
In your project select the directory “model”. Right-click on the directory to show the context menu. Select “Refresh”.

Your class model shows up in the tree view on the right side.

Create a new OCL file by selecting the directory “expressions”. Right-click to show the context menu. Select “New/File…”. Type the name of the OCL file with “.ocl” postfix.



Edit your OCL file:



Happy OCL hacking!