An Introduction into Dresden OCL2 for Eclipse

Claas Wilke and Ronny Brandt

March 8, 2009

This tutorial generally introduces into *Dresden OCL2 for Eclipse*. *Dresden OCL2 for Eclipse* is the last version of the *Dresden OCL Toolkit* and is based on the new infrastructure called the *pivot model*. The pivot model was developed by Matthias Bräuer and is described in his minor thesis (Großer Beleg) [Brä07]. Further information about the toolkit is available at the website of the *Dresden OCL Toolkit* [WWW09b].

The tutorial starts with the installation of the needed *Eclipse* plug-ins. Then it describes how to load a domain specific model and a model instance. Further activities possible with *Dresden OCL2 for Eclipse* are not in the scope of this tutorial. Documentation about such activities such as interpreting constraints or how to generate Java code for constraints can be found at [WWW09b].

The procedure described in this tutorial has been realized and tested with *Eclipse 3.4.1* [WWW09a]. The tutorial should also run with *Eclipse 3.3.x*. Besides *Eclipse* you also need to install some required plug-ins. Table 1 shows all required software to run *Dresden OCL2 for Eclipse*.

1 How to Install Dresden OCL2 for Eclipse

To use *Dresden OCL2 for Eclipse* you need to install the toolkit as *Eclipse* plug-ins, or to import them into your *Eclipse workspace*. Both possibilities are explained in the following.

Software	**Available at**
Eclipse 3.4.x	http://www.eclipse.org/
Eclipse Modeling Framework (EMF)	http://www.eclipse.org/modeling/emf/
Eclipse Model Development Tools (MDT) (only with the UML2.0 meta model)	http://www.eclipse.org/modeling/mdt/
Supclipse plug-in (only to import the toolkit from the SVN)	http://subclipse.tigris.org/
Eclipse Plugin Development Environment (only to run the toolkit using the source code distribution)	http://www.eclipse.org/pde/

Table 1: Software needed to run Dresden OCL2 for Eclipse.

1.1 Installing Dresden OCL2 for Eclipse as JAR Files

To install *Dresden OCL2 for Eclipse* as *Eclipse* plug-in, you need to have the jar archives of the toolkit. The jar archives are available at `http://dresden-ocl.sourceforge.net/`. You need to copy the jar archives into the `plugins` directory of your *Eclipse SDK* distribution. Then you can start the *Eclipse SDK* and you can work with *Dresden OCL2 for Eclipse*.

1.2 Importing Dresden OCL2 for Eclipse into an Eclipse Workspace

Alternatively you can import *Dresden OCL2 for Eclipse* as plug-in projects into an *Eclipse* workspace. Two different options exists to import the toolkit into an workspace.

On the one hand you can import the plug-ins from your file system, if you have already downloaded them (e.g. from `http://sourceforge.net/projects/dresden-ocl/` as a source code distribution). One the other hand you can import the plug-ins directly from the *SVN* (*Subversion*) location of *Dresden OCL2 for Eclipse*. Both possibilities are described below.

1.2.1 Import the Plug-ins from the Local File System

Let's say you have the plug-ins located in a directory `XYZ` of your file system. To import them into your *Eclipse* workspace you can use the *Eclipse import wizard*. Open the wizard via the menu `File > Import...` and select `General > Existing Projects into Workspace` (see figure 1). In the following window you select the directory `XYZ` as root. Then you select the plug-ins you want to import (if not selected automatically) and activate the check box `Copy projects into workspace` (see figure 2). After pressing the `Finish` button the plug-ins will be imported as projects into your workspace.

1.2.2 Import the Plug-in Projects from SVN

To import the plug-ins directly from the *SVN* repository, you need to install an additionally *Eclipse* plug-in to connect with the *SVN*. The needed plug-in is called *Subclipse*. The installation of *Subclipse* is explained on the *Supclipse* website [Sub09].

Attention: The *Subclipse* plug-in does not work with some distributions of *Eclipse 3.4*. To access the *SVN* Repository with *Eclipse 3.4* use another *SVN* plug-in for *Eclipse* or use an external tool to access the *SVN* repository.

After installing *Subclipse*, a new *Eclispe perspective* for access to *SVN* should exist. The perspective can be opened via the menu `Window > Open Perpective > Other... > SVN Repository Exploring`. In the view *SVN Repository* you can add a new repository (see figure 3) using the URL `https://dresden-ocl.svn.sourceforge.net/svnroot/dresden-ocl/`. After pressing the `Finish` button the SVN repository root should the visible in the *repository view*.

To checkout the plug-ins, you now select them in the repository directory `trunk/ocl20forEclipse/eclipse` and use the `Checkout...` function in the context menu (see figure 4). The given settings could be used and after a click on the `Finish` button the plug-ins should be imported.

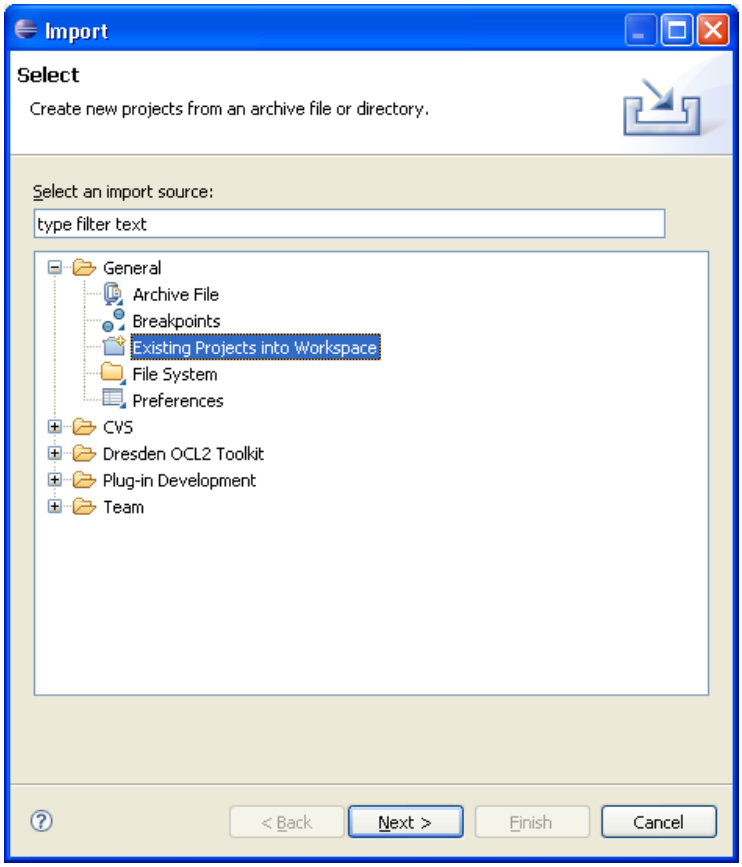

Figure 1: Plug-in import from local file system (1).

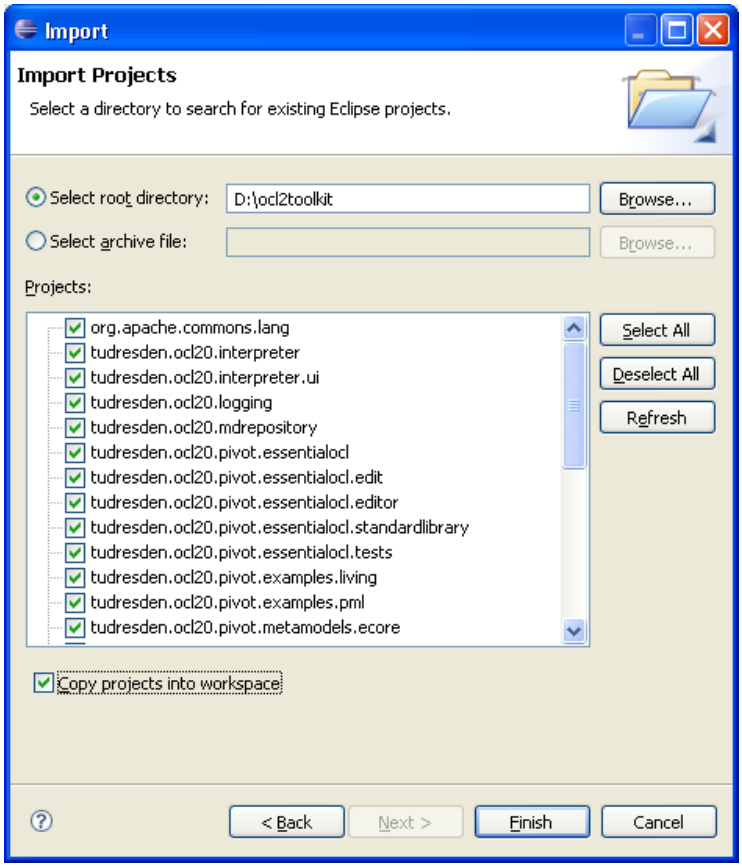

Figure 2: Plug-in import from local file system (2) (not all required projects are shown in the picture).

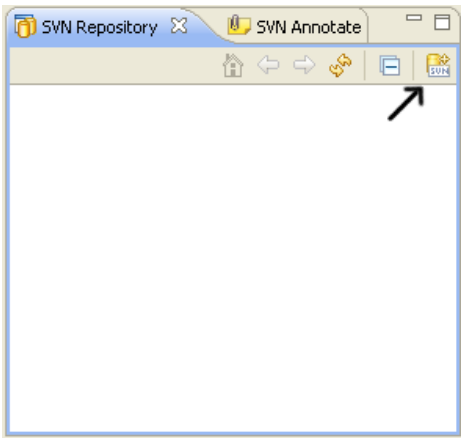

Figure 3: Adding an SVN repository.

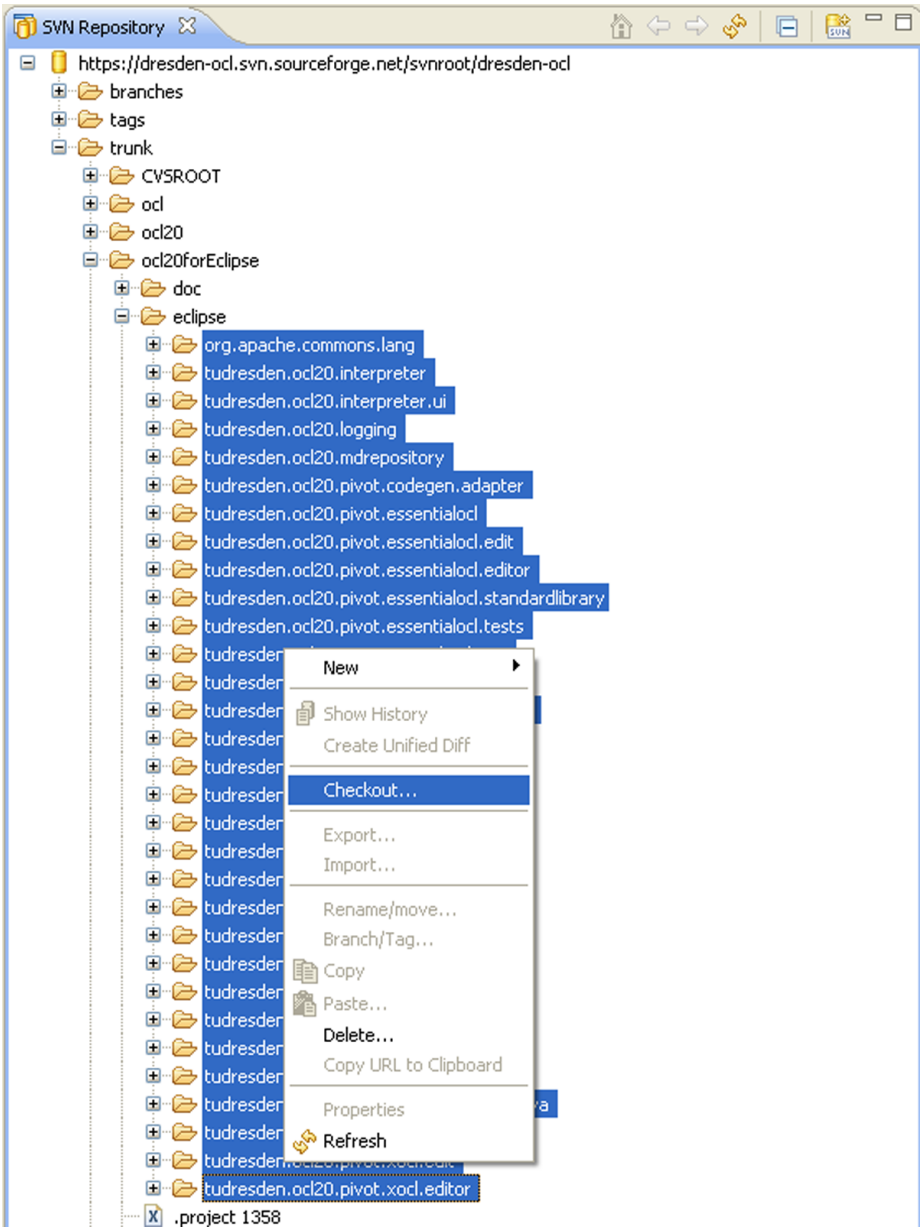

Figure 4: Checkout of the Dresden OCL2 Toolkit plug-in projects (not all plug-ins are shown in the picture).

2 Building the OCL2 Parser

If you decided to run *Dresden OCL2 for Eclipse* as project plug-ins into your workspace, you need to build the *OCL2 Parser* via an *Ant* build script. If you installed the Toolkit using jar archives, you can skip this chapter of the tutorial.

To build the *OCL2 Parser* select the file `build.xml` in the project `tudresden.ocl20.pivot.ocl2parser` and open the context menu via a right mouse click. Select the function `Run As ... > Ant Build` (see figure 5).

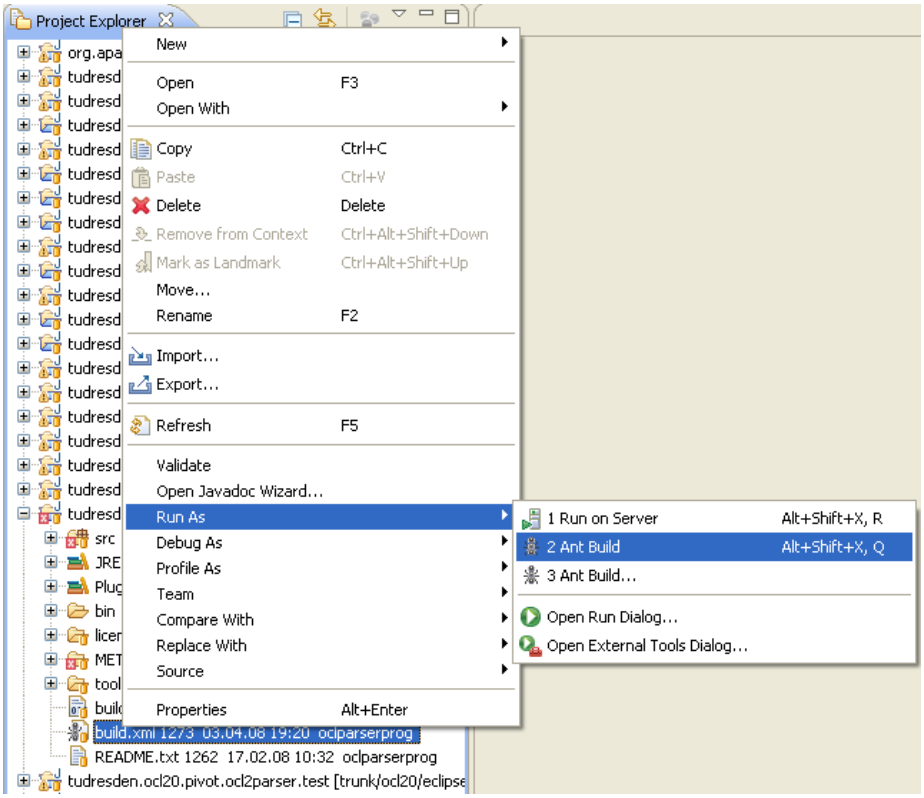

Figure 5: Executing the OCL2 parser build script.

If an error like `Problem: failed to create task or type eclipse.refreshLocal` occurs, you need to change the configuration of the *Ant* script. Open the function `Properties` in the context menu of the `build.xml`. A new window should open. Select the topic `Run/Debug settings` and then the configuration for `tudresden.ocl20.pivot.oclparser build.xml`. Click on the button `Edit`. In the new opened window select in the sub menu `JRE` the check box `Run in the same JRE as the workspace` and click on the button `OK` (see figure 6). Afterwards the *Ant* script should be executable without errors.

After executing the build script successfully you need to update the projects in your workspace. Update the project `tudresden.ocl20.pivot.oclparser` via context menu (`Refresh`, see figure 7).

Additionally you need to recompile all depending projects. Select the function `Project > Clean... > Clean all projects` in the *Eclipse* menu to

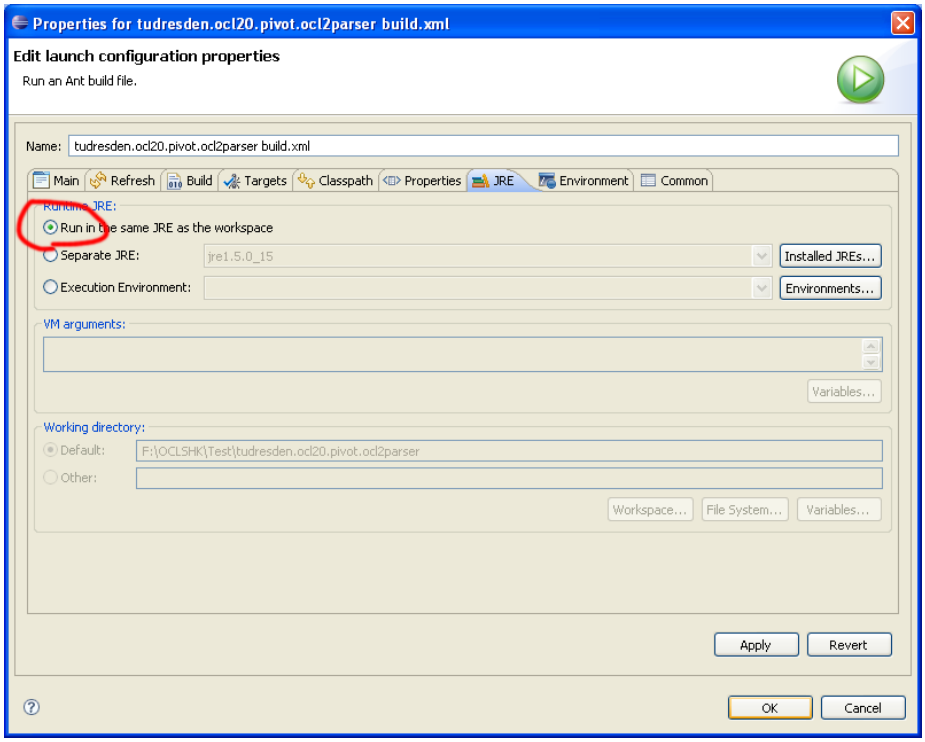

Figure 6: Settings of the JRE for the Ant build script.

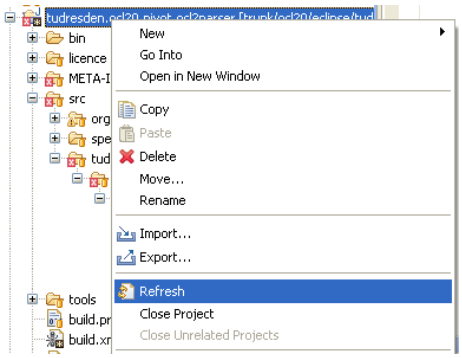

Figure 7: Refreshing the project "tudresden.ocl20.pivot.oclparser".

clean all projects. Now all the projects should not contain errors anymore and should be executable.

3 Loading Models into Dresden OCL2 for Eclipse

If you installed the *Dresden OCL2 for Eclipse* using jar archives, you can execute the toolkit by starting your *Eclipse SDK*. If you imported the Toolkit as plug-in projects into an *Eclipse* workspace, you have to start a new *Eclipse SDK* instance. You can start a new instance via the menu `Run > Run As > Eclipse Application`. If the menu `Eclipse Application` is not available or disabled you need to select one of the plug-ins of the toolkit first.

This tutorial explains *Dresden OCL2 for Eclipse* using the *Simple Example* which is located in the plug-in package `tudresden.ocl20.pivot.examples.simple`. Figure 8 shows the class diagram of the *Simple Example*.

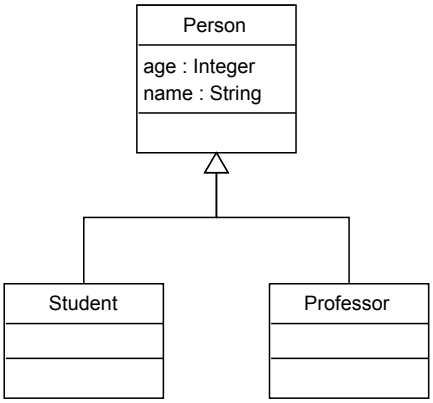

Figure 8: The class diagram described by the simple example model.

Dresden OCL2 for Eclipse provides more examples than the *Simple Example*. The different examples use different meta models which is possible with the *pivot-model* architecture of the toolkit. An overview about all examples provided with *Dresden OCL2 for Eclipse* can be found in [Wil09].

The *Simple example* can be used with two different meta models. These are *UML 1.5* (based on *Netbeans MDR*) and *UML 2.0* (based on *Eclipse MDT UML2*).

3.1 Loading a Domain-Specific Model

After starting *Eclipse* you have to load a model into the toolkit.

If the plug-ins of *Dresden OCL2 for Eclipse* were installed as jar archives, the *Simple example* plug-in has to be imported into the *Workspace* first. Create a new Java project into your *Workspace* and select the *import wizard* `General > Archive File`. In the following window select the `plugins` directory in your *Eclipse* root folder, select the archive `tudresden.ocl20.pivot.examples.simple_1.0.0.jar` and click the `Finish` button.

Now you can load a model. Select the menu option `Dresden OCL2 > Load Model`. In the opened wizard you have to select a model file and a meta model for the model (see figure 9). Click the button `Browse Workspace...` and select the file `model/simple.uml` inside the *Simple example project*. Then select the meta model *UML2* and press the button `Finish`.

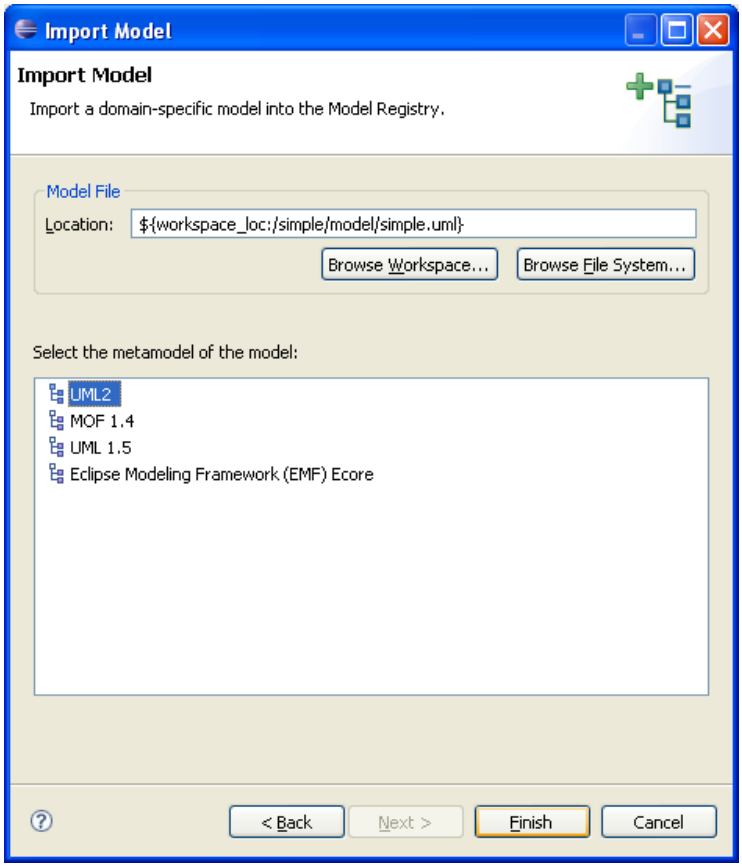

Figure 9: Loading a domain specific model.

Figure 10 shows the loaded *simple* model, which uses *UML2* as its meta model. Via the menu button of the *Model Browser* (the little triangle in the right top corner) you can switch between different models (see figure 11).

3.2 Loading a Model Instance

After loading the model, you can load a *model instance* using another *wizard*. Use the menu option `Dresden OCL2 > Load Model Instance`. In the opened wizard you have to select a model instance (in this tutorial we used the file `bin/tudresden/ocl20/pivot/examples/ModelProviderClass.class` of the *simple example* and a domain-specific model loaded before (see figure 12).

Figure 13 shows the loaded model instance of the *simple example* model. Like in the model browser you can switch between different model instances. Note

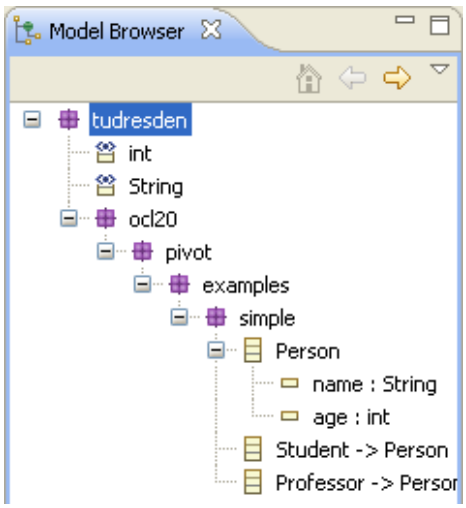

Figure 10: The loaded simple example model in the model browser.

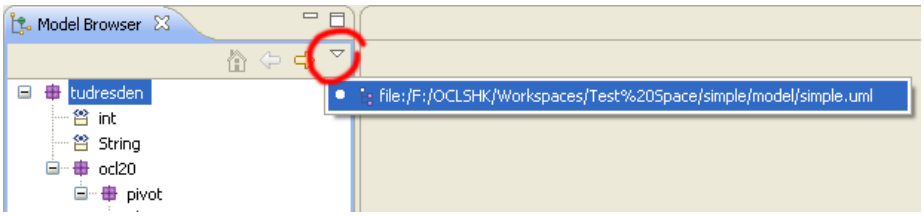

Figure 11: You can switch between different models using the little triangle.

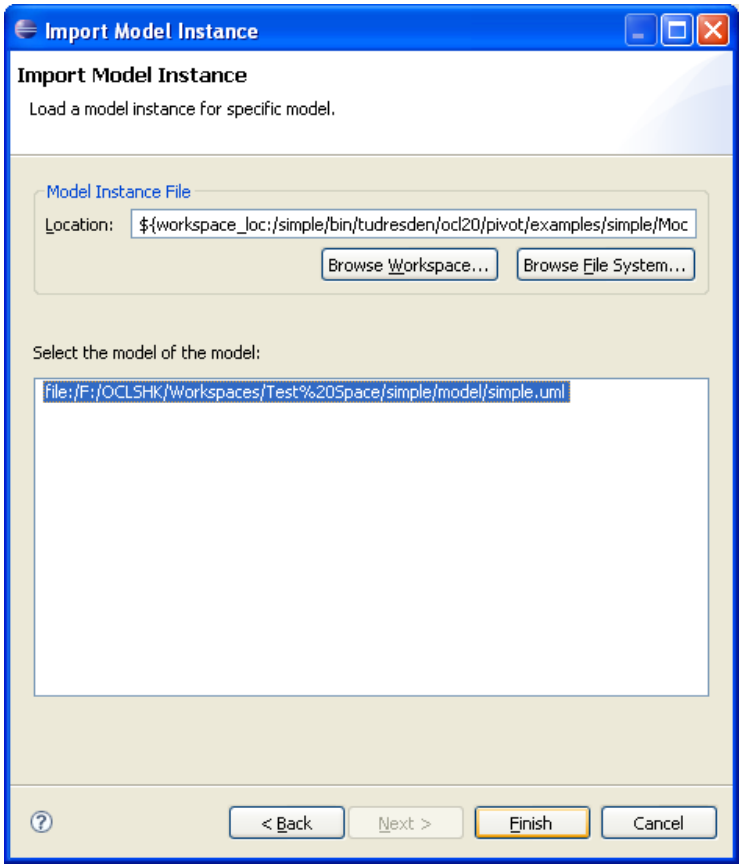

Figure 12: Loading a simple model instance.

that the model instance browser only shows the model instances of the model actually selected in the model browser. By switching the domain specific model, you also switch the pool of model instances available in the model instance browser.

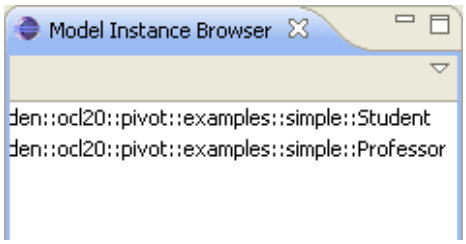

Figure 13: A simple model instance in the Model Instance Browser.

3.3 Parsing OCL expressions

Before you can interpret OCL constraints you have to load them like the domain-specific model and the model instance before. Use the menu option `Dresden OCL2 > OCL Expressions` and select an OCL file (in this tutorial we used the OCL file `constraints/invariants.ocl` of the *simple example*, see figure 14). The constraints of the file `constraints/invariants.ocl` are shown in listing 1.

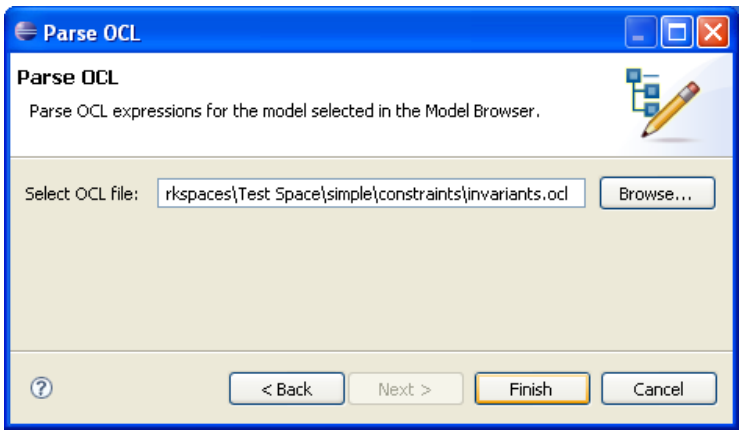

Figure 14: The import of OCL expressions.

The expressions of the selected OCL file will be loaded into the actually selected model instance. Figure 15 shows the `Model Browser` containing the model and the parsed expressions.

4 Conclusion

This tutorial described how to use *Dresden OCL2 for Eclipse*. It explained how to install or import and start the Toolkit's plug-ins. Afterwards the loading of

```
1  package tudresden::ocl20::pivot::examples::simple
2
3  -- The age of Person can't be negative.
4  context Person
5  inv: age >= 0
6
7  -- Students should be 16 or older.
8  context Student
9  inv: age > 16
10
11 -- Proffesors should be at least 30.
12 context Professor
13 inv: not (age < 30)
14
15 endpackage
```

Listing 1: The invariants of the simple examples.

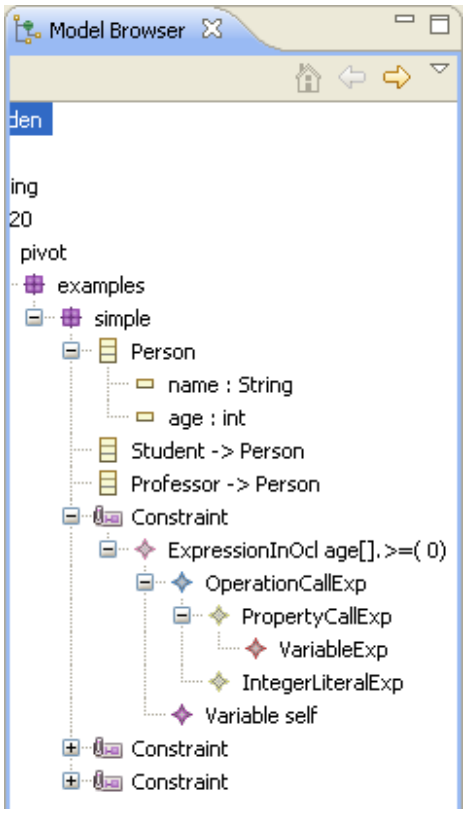

Figure 15: Parsed expressions and the model in the Model Browser.

domain-specific models, model instances and OCL constraints into the toolkit has been explained.

Now the imported models can be used to use the tools provided with *Dresden OCL2 for Eclipse*. For example you can use the OCL2 Interpreter of *Dresden OCL2 for Eclipse* to interpret OCL constraints for a given model and model instance. Tutorials how to use the OCL2 Interpreter and the other tools provided with *Dresden OCL2 for Eclipse* can be found at [WWW09b] in the *Dresden OCL2 for Eclipse* > *usage* section.

References

[Brä07] Bräuer, Matthias: *Models and Metamodels in a QVT/OCL Development Environment*. Großer Beleg, May 2007. – Available at http://dresden-ocl.sourceforge.net/gbbraeuer/index.html

[Sub09] *Installation of the Subclipse plugin*. Subclipse project Website, 2009. – Available at http://subclipse.tigris.org/install.html

[Wil09] Wilke, Claas: *Examples provided with Dresden OCL2 for Eclipse*. Published at the Dresden OCL Toolkit Website., February 2009. – Available at http://dresden-ocl.sourceforge.net/4eclipse_usage.html

[WWW09a] *Eclipse project Website*. Eclipse project Website, 2009. – Available at http://www.eclipse.org/

[WWW09b] *Website of the Dresden OCL Toolkit*. Sourceforge project Website, 2009. – Available at http://dresden-ocl.sourceforge.net/