TECHNISCHE UNIVERSITÄT DRESDEN

Fakultt Informatik Institute for Software- und Multimediatechnik, Lehrstuhl fr Softwaretechnologie

AN INTRODUCTION INTO DRESDEN OCL2 FOR ECLIPSE

Claas Wilke

Last update: May 25, 2009

CONTENTS

1 INTRODUCTION

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 a *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* [TUD09d].

The tutorial starts with the installation of the needed *Eclipse* plug-ins. Afterwards, it describes how to load a domain specific model, an instance of such a model, and OCL constraints defined on such a model. 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 [TUD09d].

The procedure described in this tutorial has been realized and tested with *Eclipse 3.4.1* [Ecl09]. We recommend to use the *Eclipse Modeling Tools Edition* which contains all required plug-ins to run *Dresden OCL2 for Eclipse*. Otherwise you need to install at least the plug-ins enlisted in table A.1.

2 HOW TO INSTALL DRESDEN OCL2 FOR ECLIPSE

Four different possibilities exist to install *Dresden OCL2 for Eclipse*. (1) You may install the plug-ins using the update site available at [TUD09c], (2) you may install the plug-ins using the binary distribution available at [TUD09a], (3) you may run the the source code distribution available at [TUD09a], or (4) you may checkout and run the source code distribution from the SVN available at [TUD09b]. This tutorial will explain the possibilities (1) and (4).

2.1 INSTALLING DRESDEN OCL2 FOR ECLIPSE USING THE ECLIPSE UPDATE SITE

To install *Dresden OCL2 for Eclipse* via the *Eclipse Update Site*, you have to start an *Eclipse* instance and select the menu option *Help -> Software Updates ....* A new window called *Software Updates and Add-ons* should open. Select the page *Available Software* and click on the button *Add Site ...* to add the toolkit update site (see figure 2.1).

Enter the path

`http://dresden-ocl.svn.sourceforge.net/svnroot/dresden-ocl/trunk/ocl20forEclipse/updatesite/tudresden.ocl20.updatesite_1.2.0`

and press the *OK* button (see figure 2.2). If an error occurs, please make sure, that you use `http` instead of `https`.

Now you can select the features of *Dresden OCL2 for Eclipse* which you want to install. Select them and click on the *Install ...* button (see figure 2.3). An otherview about all features of *Dresden OCL2 for Eclipse* can be found in table **??**. Follow the wizard and agree with the user license. Then the Toolkit will be installed. Afterwords, you should restart your *Eclipse* application to finish the installation.

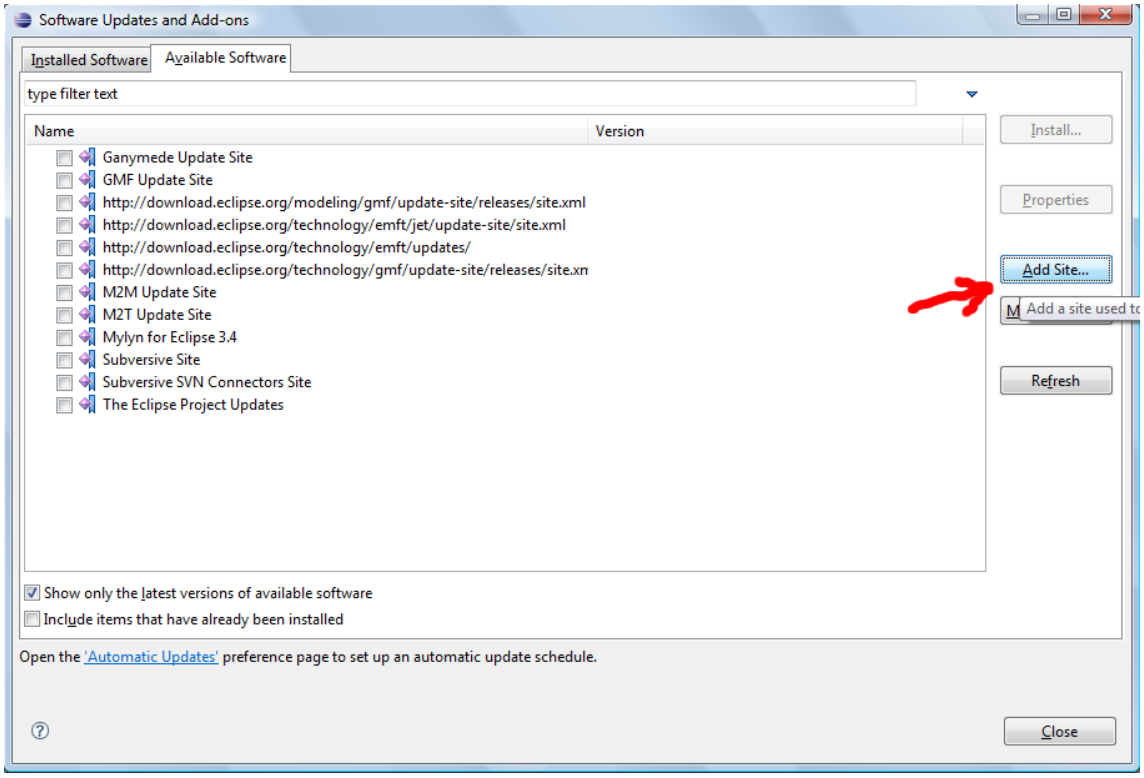

Figure 2.1: Adding an Eclipse Update Site.

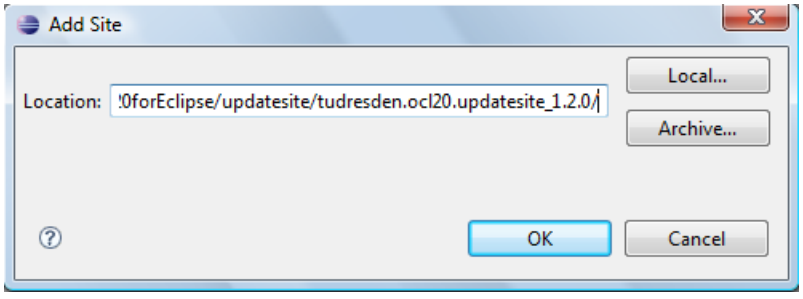

Figure 2.2: Adding the Dresden OCL2 for Eclipse Update Site.

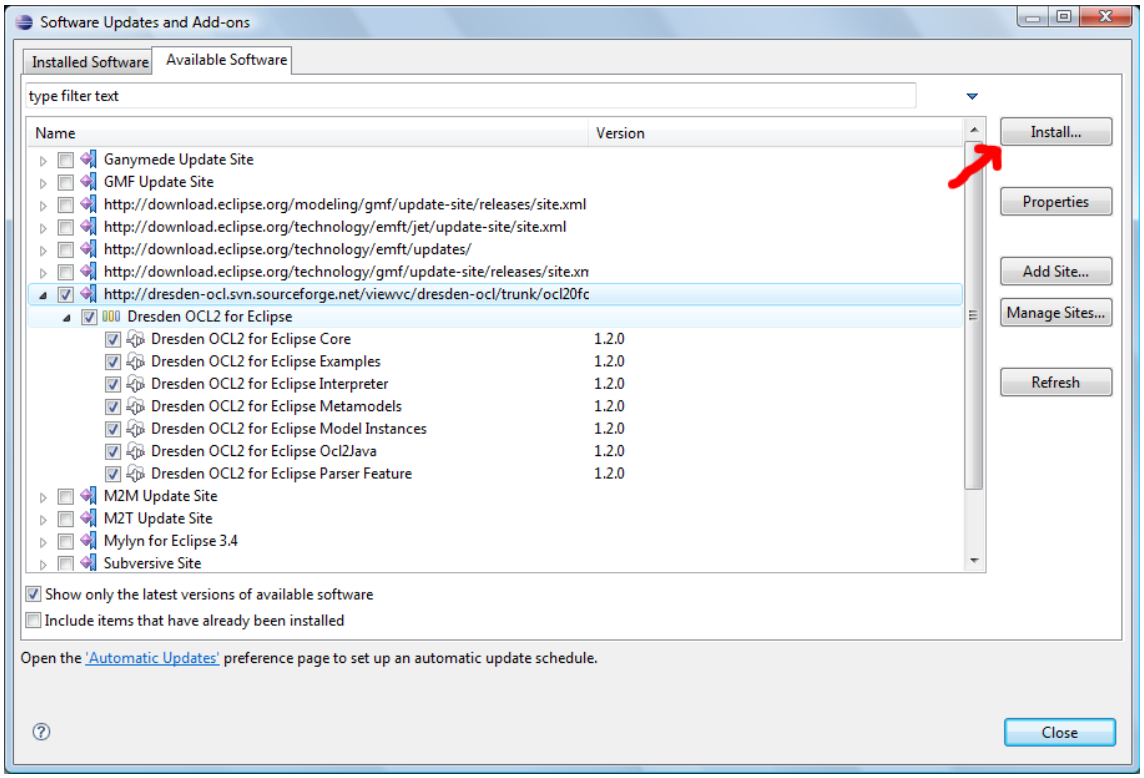

Figure 2.3: Selecting features of Dresden OCL2 for Eclipse.

2.2 IMPORTING DRESDEN OCL2 FOR ECLIPSE FROM THE SVN

To use *Dresden OCL2 for Eclipse* by checking out the source code from the SVN you need to install a SVN client. In the following the use the *Eclipse Subversive* plug-in and at least one of the *SVN Connectors* available at [Pol09].

After installing *Eclipse Subversive*, 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 Repositories* you can add a new repository (see figure 2.4) 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 *SVN Repositories* 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 2.5).

2.3 WHICH PLUG-INS DO I NEED AT LEAST?

An often asked question is "Which plug-ins are at least required to run *Dresden OCL2 for Eclipse*?" Well, the answer is: "That depends."

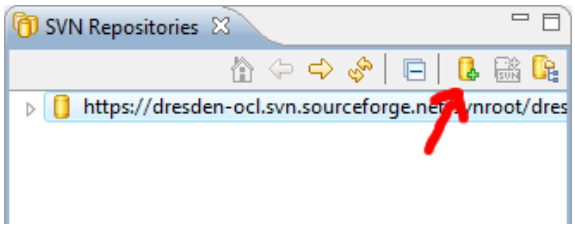

Figure 2.4: Adding an SVN repository.

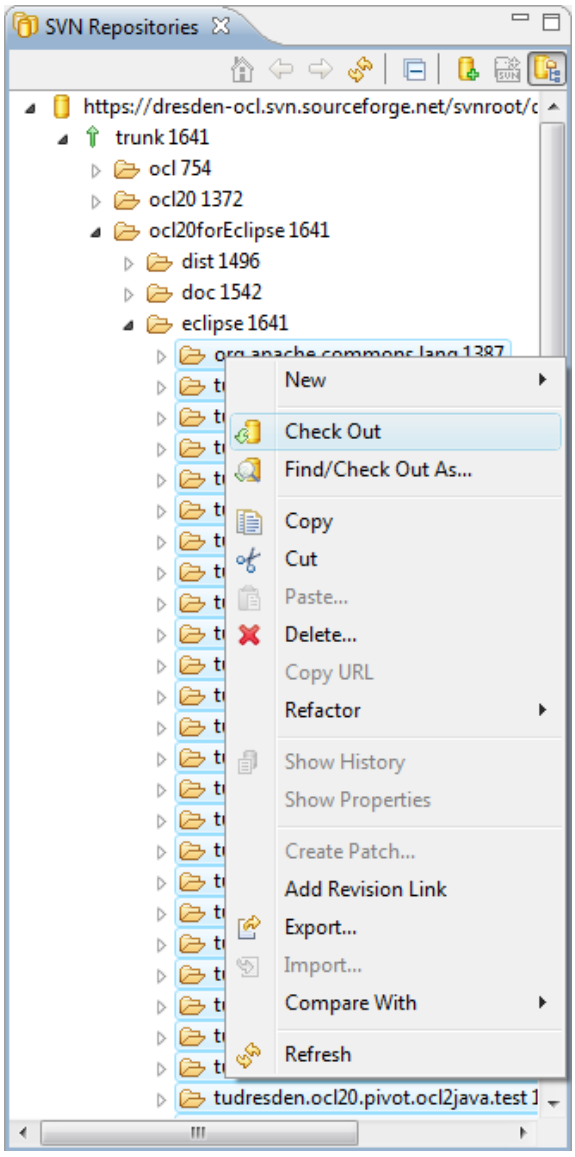

Figure 2.5: Checkout of the Dresden OCL2 Toolkit plug-in projects.

That depends on the things you want to do with *Dresden OCL2 for Eclipse*. Table A.2 shows a list of the current plug-ins of *Dresden OCL2 for Eclipse*. Which are related to different features. You should install at least the *Core* feature, at least one of the *Metamodels*, and the *Parser* feature. The *Interpreter* and *OCL2Java* features are only required if you want to interpret constraints or to generate code. If you import or interpret modelinstances, you need to install the *Model Instances* feature as well. The examples of the *Example* feature are only required to run the examples provided in the tutorials available at [TUD09d]. This tutorial recommends to first install all provided features.

2.4 BUILDING THE OCL2 PARSER

If you decided to run *Dresden OCL2 for Eclipse* as source code plug-ins from an *Eclipse* workspace, you need to build the *OCL2 Parser* via an *Ant* build script. If you installed the Toolkit using the update site, you can skip this section 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 2.6).

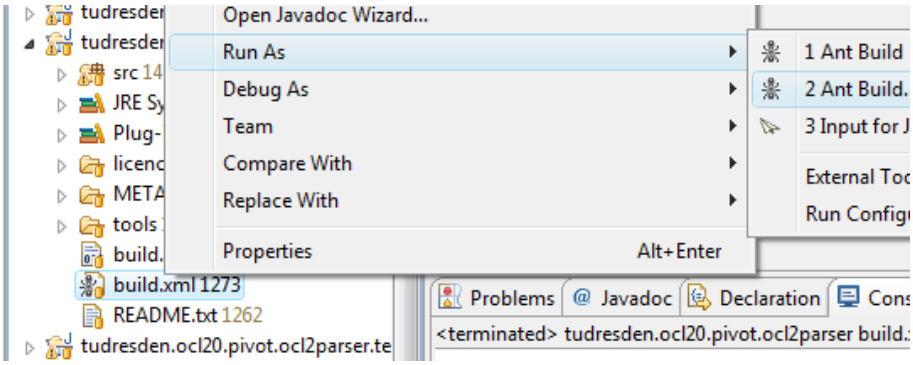

Figure 2.6: Executing the OCL2 parser build script.

A new window should open. 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 2.7). Afterwards the *Ant* parser should be generated without errors. If an error like *Unable to find javac compiler.* occurs, you might be trying to run the *Ant* script with an *Java Runtime Environment* instead of the *Java Development Kit* (For errors like) use the *Installed JREs...* button in the same window to select a *JDK* instead.

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 2.8).

Additionally you need to recompile all depending projects. Select the function *Project > Clean... > Clean all projects* in the *Eclipse* menu to clean all projects. Now all the projects should not contain errors anymore and should be executable.

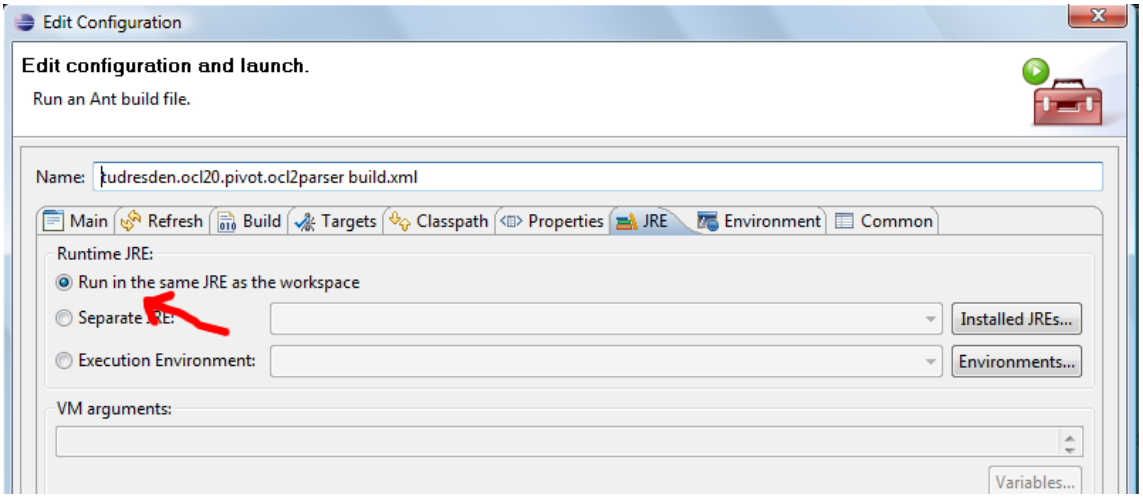

Figure 2.7: Settings of the JRE for the Ant build script.

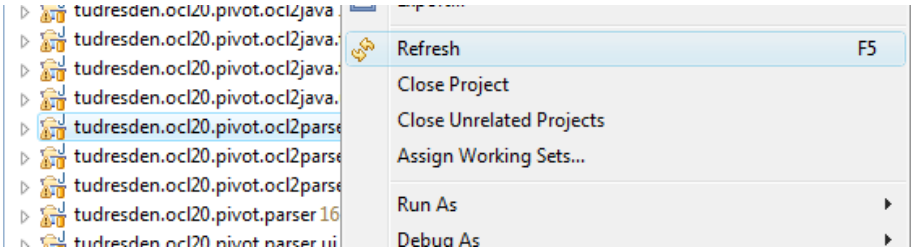

Figure 2.8: Refreshing the project "tudresden.ocl20.pivot.oclparser".

3 LOADING MODELS, MODEL INSTANCES AND OCL CONSTRAINTS

If you installed the *Dresden OCL2 for Eclipse* using the update site, you can execute the toolkit by re-starting your *Eclipse* distribution. If you imported the Toolkit as source code plug-ins into an *Eclipse* workspace, you have to start a new *Eclipse* 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.

3.1 THE SIMPLE EXAMPLE

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 3.1 shows the class diagram of the *Simple Example*.

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 2.0* (based on *Eclipse MDT UML2*) and *EMF Ecore*.

3.2 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* have been installed using the update site, 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.

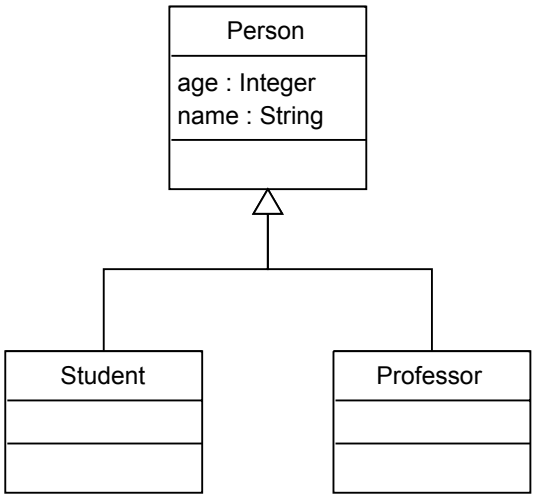

Figure 3.1: The class diagram described by the simple example model.

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 3.2). 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*.

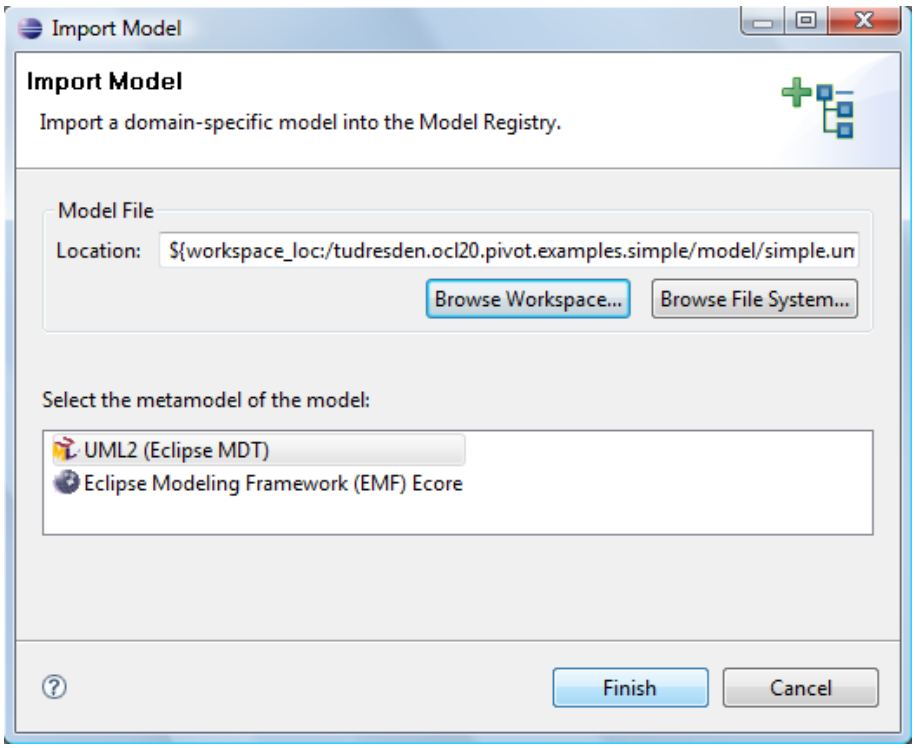

Figure 3.2: Loading a domain specific model.

Figure 3.3 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 3.4).

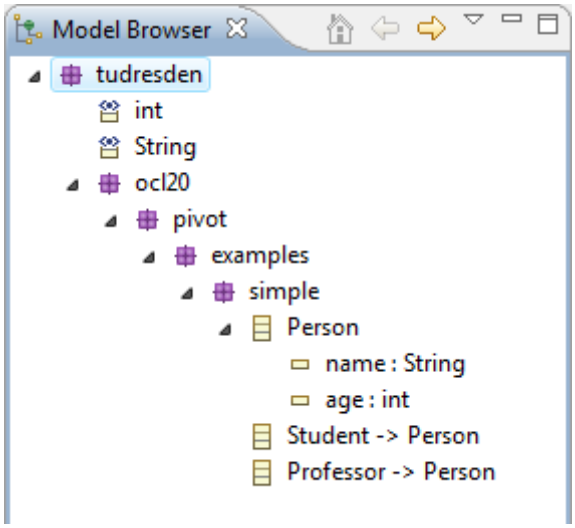

Figure 3.3: The loaded Simple Example model in the model browser.

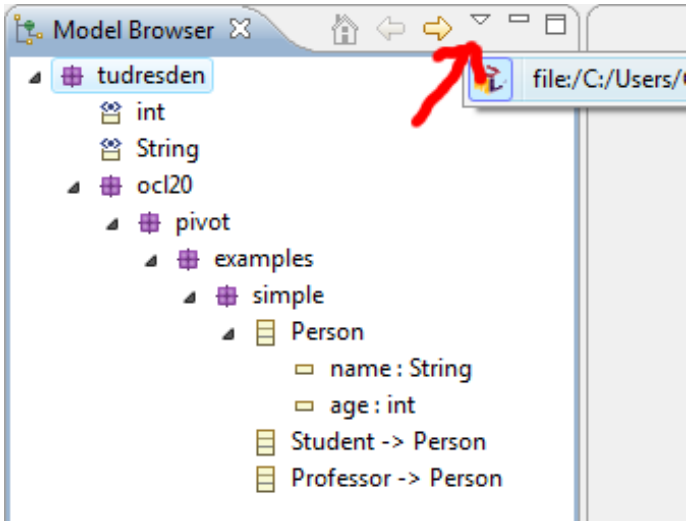

Figure 3.4: You can switch between different models using the little triangle.

3.3 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* a domain-specific model loaded before (see figure 3.5). Besides the model instance resource you have to select a model for which the model instance shall be loaded and the type of model instance you want to load (we want to load a *Java Instance*).

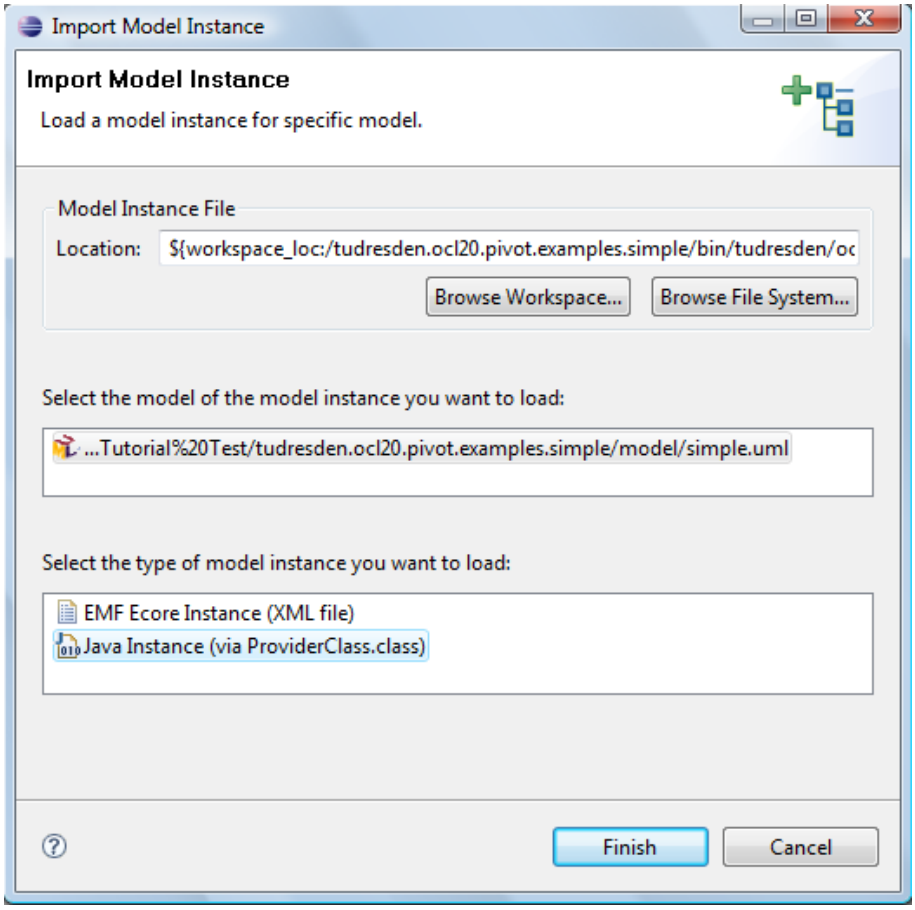

Figure 3.5: Loading a simple model instance.

Figure 3.6 shows the loaded model instance of the *Simple Example* model. Like in the model browser you can switch between different model instances. Note 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.

3.4 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

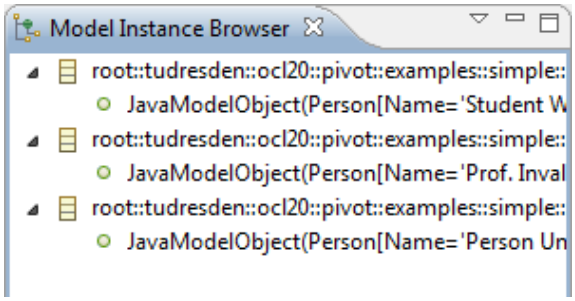

Figure 3.6: A simple model instance in the Model Instance Browser.

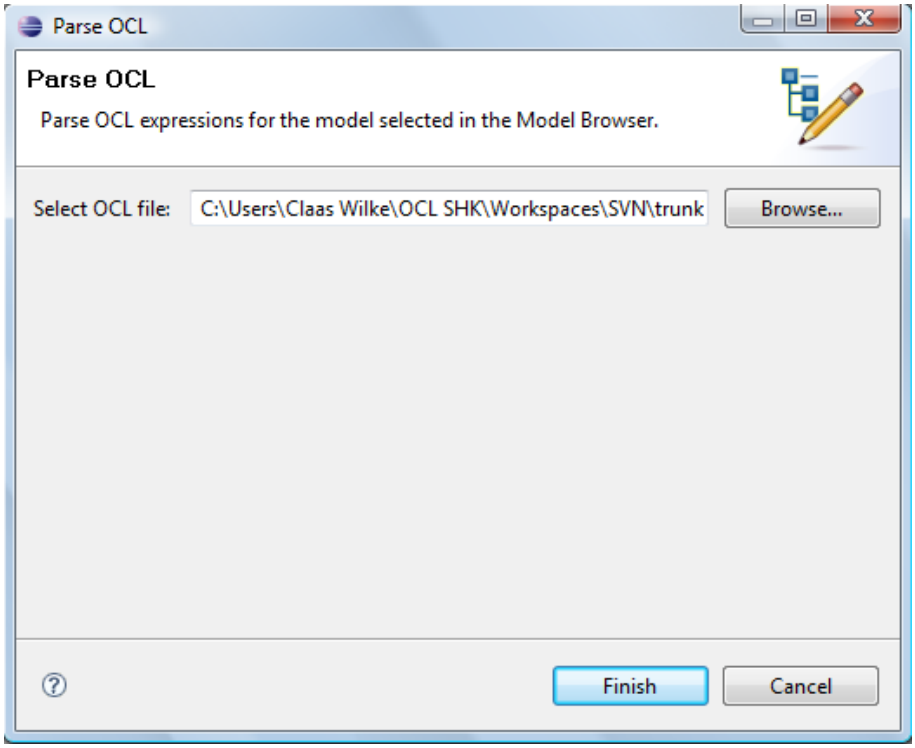

Figure 3.7: The import of OCL expressions.

```
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 3.1: The invariants of the simple examples.

Simple Example, see figure 3.7). The constraints of the file `constraints/invariants.ocl` are shown in listing 3.1.

The expressions of the selected OCL file will be loaded into the actually selected model. Figure 3.8 shows the `Model Browser` containing the model and the parsed expressions.

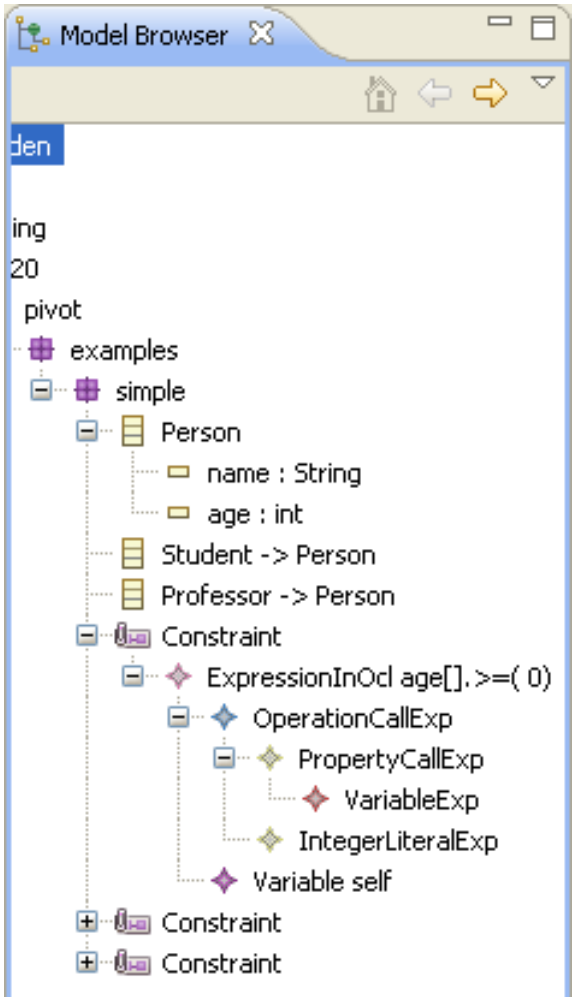

Figure 3.8: Parsed expressions and the model in the Model Browser.

4 CONCLUSION

This tutorial described how to use *Dresden OCL2 for Eclipse*. It explained how to install the Toolkit's plug-ins. Afterwards, the loading of 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 or you can use the *OCL22Java Code Generator* to generate *AspectJ* code for a loaded model and OCL constraints. Tutorials how to use the OCL2 Interpreter and the other tools provided with *Dresden OCL2 for Eclipse* can be found at [TUD09d] in the *Dresden OCL2 for Eclipse > Usage* section.

5 A SHORT NOTE ABOUT LOGGING

Dresden OCL2 for Eclipse uses a *Log4j* logger to log method entries, exits and errors during the Toolkit's execution. If you run *Dresden OCL2 for Eclipse* as source code plug-ins from an *Eclipse* workspace, you might receive exceptions like

```
log4j:ERROR Could not connect to remote log4j server at [localhost].
```

although the Toolkit works correctly. This is because the *Log4j* logger tries to sent the logged events to a server running at `localhost`.

To solve this problem (if you want to) you have to install and setup a logging server at your computer. One logging server you might use is called *Chainsaw* and available at [Apa09]. If you start *Chainsaw*, set up a *SocketReceiver* at port *4445 (Old Style/Standard Chainsaw Port)* (see figure 5.1).

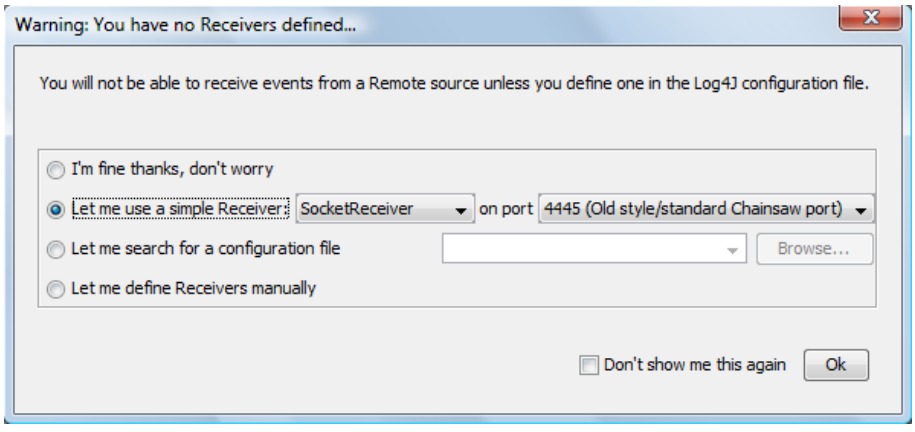

Figure 5.1: Setting up a simple SocketReceiver in Chainsaw.

A TABLES

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/
Eclipse Plug-in Development Environment (only to run the toolkit using the source code distribution)	http://www.eclipse.org/pde/

Table A.1: Software needed to run Dresden OCL2 for Eclipse (**If not using the Eclipse MDT Distribution)**.

Feature	Plug-ins
Core	**Required:** org.apache.commons.lang tudresden.ocl20.pivot.logging tudresden.ocl20.pivot.essentialocl tudresden.ocl20.pivot.essentialcol.edit tudresden.ocl20.pivot.essentialocl.editor tudresden.ocl20.pivot.essentialocl.standardlibrary tudresden.ocl20.pivot.modelbus tudresden.ocl20.pivot.modelbus.ui tudresden.ocl20.pivot.pivotmodel tudresden.ocl20.pivot.pivotmodel.edit tudresden.ocl20.pivot.standardlibrary **Optional:** tudresden.ocl20.pivot.essentialocl.tests tudresden.ocl20.pivot.pivotmodel.tests
Examples	**Optional:** tudresden.ocl20.pivot.examples.living tudresden.ocl20.pivot.examples.pml tudresden.ocl20.pivot.examples.royalandloyal tudresden.ocl20.pivot.examples.royalandloyal.constraints tudresden.ocl20.pivot.examples.simple tudresden.ocl20.pivot.examples.simple.constraints
Interpreter	**Required (for interpretation):** tudresden.ocl20.interpreter tudresden.ocl20.interpreter.ui **Optional:** tudresden.ocl20.interpreter.test
Metamodels	**Required (at least one of the following):** tudresden.ocl20.pivot.metamodels.ecore tudresden.ocl20.pivot.metamodels.uml2
Model Instances	**Required (at least one of the following for interpretation):** tudresden.ocl20.pivot.modelinstancetype.ecore tudresden.ocl20.pivot.modelinstancetype.java
Ocl2Java	**Required (for code generation):** tudresden.ocl20.pivot.ocl2java tudresden.ocl20.pivot.ocl2java.ui **Optional (eventually for code execution):** tudresden.ocl20.pivot.ocl2java.types **Optional:** tudresden.ocl20.pivot.ocl2java.test
Dresden OCL2 for Eclipse Parser Feature	**Required:** tudresden.ocl20.pivot.ocl2parser tudresden.ocl20.pivot.parser tudresden.ocl20.pivot.parser.ui **Optional:** tudresden.ocl20.pivot.ocl2parser.test

Table A.2: The plug-ins of Dresden OCL2 for Eclipse related to their feature.

LIST OF FIGURES

BIBLIOGRAPHY

[Apa09] *Apache Chainsaw*. Apache Logging Services, 2009. – http://logging.apache.org/chainsaw/

[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

[Ecl09] *Eclipse project Website*. Eclipse project Website, 2009. – http://www.eclipse.org/

[Pol09] *Polarion Software: Subversive*. Polarion.org Community, 2009. – http://www.polarion.com/products/svn/subversive.php

[TUD09a] *Sourceforge Project Site of the Dresden OCL Toolkit*. Sourceforge project website, 2009. – http://sourceforge.net/projects/dresden-ocl/

[TUD09b] *SVN of the Dresden OCL Toolkit*. Subversion Repository, 2009. – https://dresden-ocl.svn.sourceforge.net/svnroot/dresden-ocl

[TUD09c] *Update Site of the Dresden OCL Toolkit*. Eclipse Update Site, 2009. – http://dresden-ocl.svn.sourceforge.net/svnroot/dresden-ocl/trunk/ocl20forEclipse/updatesite/tudresden.ocl20.updatesite_1.2.0

[TUD09d] *Website of the Dresden OCL Toolkit*. Project website, 2009. – http://dresden-ocl.sourceforge.net/

[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