# From Uml2 to Babel

Drikus Kleefsman

June 30, 2009

## Abstract

*Keywords: Code generation, Uml, Uml2, Text2Uml, TextUml, Uml2Babel*

*Using TextUml a Uml2 model can be created and saved in the Eclipse workbench. A nice feature is that TextUml supports Action sementics. The way actions are saved in TextUml is used to generate code in different languages, at least that's our objective. The way we interface with TextUml Uml2-models is described here (version 0.31).*

## 1 Introduction

The aim is to have one UML2 model that can generate code in many languages.

Section 2 Installation. Section 3 Classes to interface with. Section 4 Example generating a language.

Software can map a UML2 model to a specific language. We restrict ourselves to object oriented languages and script languages that support some kind af class notion. The maturity of a mapper will be measured by three tests. A mapper should indicate in the documentation in the sourcecode their comments on the tests (if it passes the test and how it passes the test).

A mapper should at least indicate to wich extent it passes the following two tests:

- make a textfile from the example Pet store. The pet store in textUml can be found at the TextUml site. The mapper should produce code for all included .tuml files. The pet store example does not contain any action semantics. The text files must compile in the target language of the mapper.
- We are busy making an example A-Star algorithm containing much action semantics. The mapper can indicate if it supports all the action semantics in the algorithm or what is missing

## 2 Installation

### 2.1 TextUml

TextUml can be downloaded from http://abstratt.com/update/milestones/ using the Eclipse update facility (in Eclipse use menu Help — Software Updates...). More on building and installing TextUml can be found at http://sourceforge.net/apps/mediawiki/textuml/index.php?title=Contributor_Guide. Read it carefully as you will have to build the code step by step. In the process you will generate code for the extensions TextUml defines. Especially the package com.abstratt.mdd.core.meta is important as it contains the Factory and Package for the extensions. Look at the class MetaFactory with its create-methods (createClosure(), createCollectionLitteral, createMetaValueSpecification, createSignatue etc.) We have included the three packages com.abstratt.mdd.core.meta, com.abstratt.mdd.core.meta.impl and com.abstratt.mdd.core.meta.util in our own project.

### 2.2 Uml2Babel

Use of Uml2Babel.

Uml2Babel has the following packages:

1. an inclusion of EMF related elements. Especially important is the class TUmlResource that should ease the loading of textUml files
2. a number of support functions in drikus.net.common. It contains message functions for warning, debugging and notifying that produce text messages on the console. And FiledFiles that can be used to create and write to outputfiles and to create directories.
3. the interface we use with the eclipse uml2 library. The packagenames start with drikus.net.mdd.uml.classes.
4. some mappers. One mapper produces tuml code from a uml2 model and another produces java code.

## 3 Classes

Some general remarks on the classes in the packages having a name that starts with drikus.net.mdd.uml.classes:

1. Most classes contain a Uml2 element. All these classes have a method getUml2(). If a feature is not supported by our classes there is propably resort to a solution using this method.
2. No use is being made of enumerations. In the uml2 library there are many enumerations. In this library strings are used.
3. To be able to navigate back in the hierarchie some classes that are subclasses of UmlNamedElement have a method getParent. In the case of a UmlClass for instance the getParent() returns the UmlPackage the class belongs to.
4. Apart from the stereotypes used in TextUml an extra stereotype ”Constructor” is used. In the class UmlOperation the stereotype is used to tell which operation is a constructor.
5. class UmlEnvironment can be used to pass extra information to the methods in a mapper. In the Uml2JavaMapper it is used for passing the indentation, the output rootdirectory and the current package. There are two methods to increment and decrement the indentation.
6. UmlTraversal starts the mapping creating a mapper, an initial environment and loading the model(s) before calling the map() method on the mapper.

### 3.1 UmlNamedElement

Subclasses of UmlNamedElement

All the subclasses of UmlNamedElement are found in the package drikus.net.mdd.uml.classes. The exception is UmlTemplateBinding, which is not a subclass and of course UmlNamedElement itself. All these classes have a getName() but also some support for templates. Of a subclass of UmlNamedElement you can ask if it is a template itself (isTemplate()) and you can ask if it is bound to a template (isBoudToTemplate()). Also the method getAppliedStereotypes() can be called on a UmlNamedElement.

### 3.2 Getting the method activity

An UmlOperation has the method UmlActivity getActivity() to retrieve the activity having the same name as the operation.

### 3.3 Structured Activities and Statements

A UmlStructuredActivity is seen as a complex statement, containing one or many statements. An UmlStructuredActivity has the crucial getStatements() method. A UmlStatement contains a tree of ActionNodes that together make up the statement. Many classes in package drikus.net.mdd.uml.classes.complexnodes can contain a UmlStructuredActivity, for instance a LoopStatement and a ConditionalStatement. GetStatements() first gets all the Uml2 ActionNodes and makes this a statement if:

1. it is a LoopNode: a UmlLoopStatement is created
2. it is a ConditionalNode: a UmlConditionalStatement is created
3. it is a StructuredActivityNode: a UmlStructuredStatement is created
4. it is a validFirst Action: a UmlStatement is created with a tree of ActionNodes. The root-element of the tree is created in createActionNode(). The root is a subclass of UmlActionNode and the varieties are UmlCallNode, UmlAddStructuralFeatureNode, MetaValueSpecificationNode and UmlReturnVariableActionNode or otherwise the plain vanilla UmlActionNode.

### 3.4 ActionNodes

In package drikus.net.mdd.uml.classes.nodes are classes that are used to build a UmlStatement tree. The most general is UmlActionNode. All other classes in this package extend UmlActionNode. This class has a method getText() that returns a text specific to the differt types of nodes. For instance if it is a ReadSelfAction the returned text is "self" and if it is a AddVariableValueAction then the name of the variable is returned. Also the Nodes are given a type-id in the method String shortType(). As said before in this package a number of specializations of UmlActionNode are defined. These classes have the same type-id as their superclass. UmlActionAddStructuralFeature contains a Uml2 AddStructuralFeatureValueAction, UmlActionCallNode contains a Uml2 CallOperationAction, UmlActionReturnVariableNode contains a AddVariableValueAction and UmlActionMetaValueNode contains a ValueSpecificationAction. So, the type-id of a UmlActionMetaValueNode and a ValueSpecificationAction have the same value: "ValueSpecification" (shortType() deletes the substring Action).

The UmlActionMetaValueNode is used to process Closures. TextUml, when using the base sterotypes, supports closures. UmlActionMetaValueNode contains the closure in the form of an activity. It has the method getClosureActivity() to get that activity. Closure is not created by the createActionNode method of StructuredActivity. A Closure has a method getParameters().

## 4 Languages

The following source is used to map a uml2 association back to textUml. First a local copy is made of the passed environment. When in a method other methods are called that use the environment parameter we make a local copy and call the method using localEnv as in mapStereotypes(a, localEnv). Only associations that are not references from a class are mapped. The references in a class are mapped by mapClass().

```
public void mapAssociation(UmlAssociation a, UmlEnvironment env) {
UmlEnvironment localEnv = env.copy();
String outputFileName = (String)env.get("outputFileName");

if (a.isOwnedAssociation()) {
//is already mapped
}
else {
ff.write(outputFileName, "\n");
if (a.hasAppliedStereotype()) mapStereotypes(a, localEnv);
ff.write(outputFileName, "\n"+
indent(localEnv.getIndent())+"association "+a.getName());
localEnv.incIndent();
for (UmlProperty p: a.getMembers()) {
String nav =""; if (p.isNavigable()) nav="navigable ";
String occurs = occurs(p);

ff.write(outputFileName, "\n"+
indent(localEnv.getIndent())+nav+"role "+p.getName()+
": "+p.getType()+occurs+";");
}
localEnv.decrIndent();
ff.write(outputFileName, "\n"+indent(localEnv.getIndent())+"end;");
}
}
```

Of course very similar code is used for generating java code or CSharp code. Then the method mapAssociation() will produce a new class instead of a new association.