An Implementation of the Ecore to GenModel Case Study using the Epsilon Transformation Language

Dimitrios S. Kolovos\textsuperscript{1}, Louis M. Rose\textsuperscript{1}, Richard F. Paige\textsuperscript{1}, Juan De Lara\textsuperscript{2}

\textsuperscript{1} Department of Computer Science, University of York, Y010 5DD, York, UK, \{dkolovos,louis,paige\}@cs.york.ac.uk
\textsuperscript{2} Universidad Autónoma de Madrid, Juan.deLara@uam.es

1 Introduction

In this short paper we outline an implementation of the Ecore2GenModel transformation challenge \cite{1} using the Epsilon Transformation Language (ETL) \cite{2}. The implemented transformation is part of the EuGENia tool \cite{3} and covers all the requirements defined in \cite{1}. The remainder of the paper is structured as follows. Section 2 provides a brief overview of ETL and its underpinning Epsilon platform \cite{4}. Section 3 outlines the implementation and provides excerpts of the Ecore2GenModel transformation in ETL and section 4 concludes this short paper.

2 The Epsilon Transformation Language

The Epsilon Transformation Language (ETL) is a hybrid rule-based model transformation language that builds on top of the infrastructure provided by the Epsilon model management platform. ETL features a rule-based transformation specification style, rule inheritance, transformation decomposition, access to an arbitrary number of input/output models conforming to different metamodels, and specified using diverse modelling infrastructures (e.g. EMF, MDR, XSD-backed XML, plain XML etc.), a powerful imperative expression language for specifying rule bodies (EOL) and tight integration with Java. A complete specification of the syntax and semantics of ETL is available in Chapter 5 of the Epsilon Book\textsuperscript{1}.

\textsuperscript{1}\url{http://eclipse.org/gmt/epsilon/doc/book/}
3 Implementation of the Ecore2GenModel case study with ETL

The ETL Ecore2GenModel transformation consists of a number of declarative rules with imperative bodies that specify how elements of a source Ecore model shall be transformed into elements of the respective target GenModel model. Listings 1 and 2 demonstrate the EPackage2GenPackage rule which transforms each EPackage in the Ecore model to a GenPackage in the target GenModel model, and the EClass2GenClass rule which transforms each EStructuralFeature in the Ecore model to a GenFeature in the target GenModel model respectively.

Listing 1: The EPackage2GenPackage rule

```java
rule EPackage2GenPackage
transform s : Ecore!EPackage
to t : GenModel!GenPackage {
    genModel.genPackages.add(t);
t.ecorePackage = s;
t.disposableProviderFactory = s.getBooleanAnnotation("disposableProviderFactory", true);
t.prefix = s.getAnnotation("prefix", s.name.firstToUpperCase());
copyAnnotations(s, t);
}
```

Listing 2: The EStructuralFeature2GenFeature rule

```java
@greedy
rule EStructuralFeature2GenFeature
transform s : Ecore!EStructuralFeature
to t : GenModel!GenFeature {
    guard : s.eContainer().isTypeOf(Ecore!EClass)
s.eContainer().equivalent().genFeatures.add(t);
t.ecoreFeature = s;
var defaultProperty;
if (s.isTypeOf(Ecore!EReference)) {
    if (not s.container and not s.containment) {
        if (s.changeable) {
            defaultProperty = GenModel!GenPropertyKind#Editable;
        }
        else {
            defaultProperty = GenModel!GenPropertyKind#Readonly;
        }
    }
    else {
        defaultProperty = GenModel!GenPropertyKind#None;
    }
} else {
    defaultProperty = GenModel!GenPropertyKind#None;
}
t.children = s.getBooleanAnnotation("children", s.isContainment);
t.createChild = s.getBooleanAnnotation("createChild",
t.isChildren and s.isChangeable);
t.notify = s.getBooleanAnnotation("notify", t.isChildren);
```
else {
    if (s.changeable) {
        defaultProperty = GenModel!GenPropertyKind#Editable;
    } else {
        defaultProperty = GenModel!GenPropertyKind#Readonly;
    }
    t.children = s.getBooleanAnnotation("children", false);
    t.createChild = s.getBooleanAnnotation("createChild", false);
    t.notify = s.getBooleanAnnotation("notify", true);
}

t.propertySortChoices = s.getBooleanAnnotation("propertySortChoices",
    s.isTypeOf(Ecore!EReference)
    and defaultProperty = GenModel!GenPropertyKind#Editable);

t.property = s.getEnumerationAnnotation("property", t, defaultProperty);

Listing 3: The pre block of statements

var genModel = new GenModel!GenModel;
var topEPackage = Ecore!EPackage.all.selectOne(p|p.eContainer().isUndefined());
genModel.complianceLevel =
    topEPackage.getEnumerationAnnotation("complianceLevel",
        genModel, GenModel!GenJDKLevel#JDK60);
genModel.copyrightFields =
    topEPackage.getBooleanAnnotation("copyrightFields", false);
genModel.modelPluginId =
    topEPackage.getAnnotation("modelPluginID", pluginName);
genModel.modelDirectory =
    topEPackage.getAnnotation("modelDirectory", "/\+pluginName+"/src");
genModel.modelName =
    topEPackage.getAnnotation("modelName", topEPackage.name.firstToUpperCase());
genModel.importerID =
    topEPackage.getAnnotation("importerID", "org.eclipse.emf.importer.ecore");
genModel.foreignModel.add(foreignModel);

As discussed in [1], one of the improvements of the envisioned Ecore2GenModel transformation over the default Java-based implementation that ships with EMF is to allow users to embed GenModel-specific annotations in the Ecore metamodel which the transformation will then propagate to the GenModel model. In the solution presented in this paper we exploit the tight integration of ETL
with Java, which allows ETL code to invoke Java methods and the reflective facilities provided by EMF. More specifically, we have specified the \textit{copyAnnotations} operation displayed in \ref{lst:4} which can copy String and Boolean values from annotation details attached to elements in the Ecore model to attributes of the respective elements in the target GenModel model. Having defined this generic operation, we can then call it from all rules of the transformation (e.g. line 18 in \ref{lst:3}, line 10 in \ref{lst:1} and line 47 in \ref{lst:2}) to reduce duplication.

### Listing 4: The \texttt{copyAnnotations()} operation

```java
operation copyAnnotations(source : Any, target : Any) {
  for (stringFeature in target.eClass().eAllStructuralFeatures.
    select(sf|sf.eType.name = "String" or sf.eType.name = "EString")) {
    if (source.hasAnnotation(stringFeature.name)) {
      var annotationValue = source.getAnnotation(stringFeature.name, "");
      if (stringFeature.many) {
        var parts = annotationValue.split(",").collect(s|s.trim());
        target.eGet(stringFeature).addAll(parts);
      } else {
        target.eSet(stringFeature, annotationValue);
      }
    }
  }
  for (booleanFeature in target.eClass().eAllStructuralFeatures.
    select(sf|not sf.isMany and
      (sf.eType.name = "Boolean" or sf.eType.name = "EBoolean"))) {
    if (source.hasAnnotation(booleanFeature.name)) {
      target.eSet(booleanFeature, source.getBooleanAnnotation
        (booleanFeature.name, target.eGet(booleanFeature)));
    }
  }
}
```

\section{Conclusions}

At the time of writing, the complete specification of the transformation comprises 270 lines of ETL code and is available in \url{http://bit.ly/bmWbWQ}. A screen-cast demonstrating the transformation in action is also available in \url{http://bit.ly/cusuKd}. Regarding the performance of the transformation, in Table 1 we provide data for the Ecore, UML\textsuperscript{2}, and Java\textsuperscript{3} metamodels.

Here, it's worth noting that Epsilon involves a one-off setup cost of approximately 2 seconds which only applies to the first transformation executed in the


\footnote{\url{http://www.emftext.org/index.php/EMFText_Concrete_Syntax_Zoo_Java_5}}
<table>
<thead>
<tr>
<th></th>
<th>Size (Bytes)</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ecore</td>
<td>30,600</td>
<td>810</td>
</tr>
<tr>
<td>Java</td>
<td>191,600</td>
<td>2012</td>
</tr>
<tr>
<td>UML</td>
<td>1,182,911</td>
<td>6162</td>
</tr>
</tbody>
</table>

Table 1: Ecore2GenModel transformation performance

Eclipse workspace, and that the results in Table 1 were collected using Eclipse 3.5.1, EMF 2.5 and Epsilon 0.8.9.201006211154, on a 2.66 GHz Intel Core 2 Duo MacBook Pro laptop with 4GB of RAM, running Mac OS X 10.6.4.

References


