Model Transformation By-Example

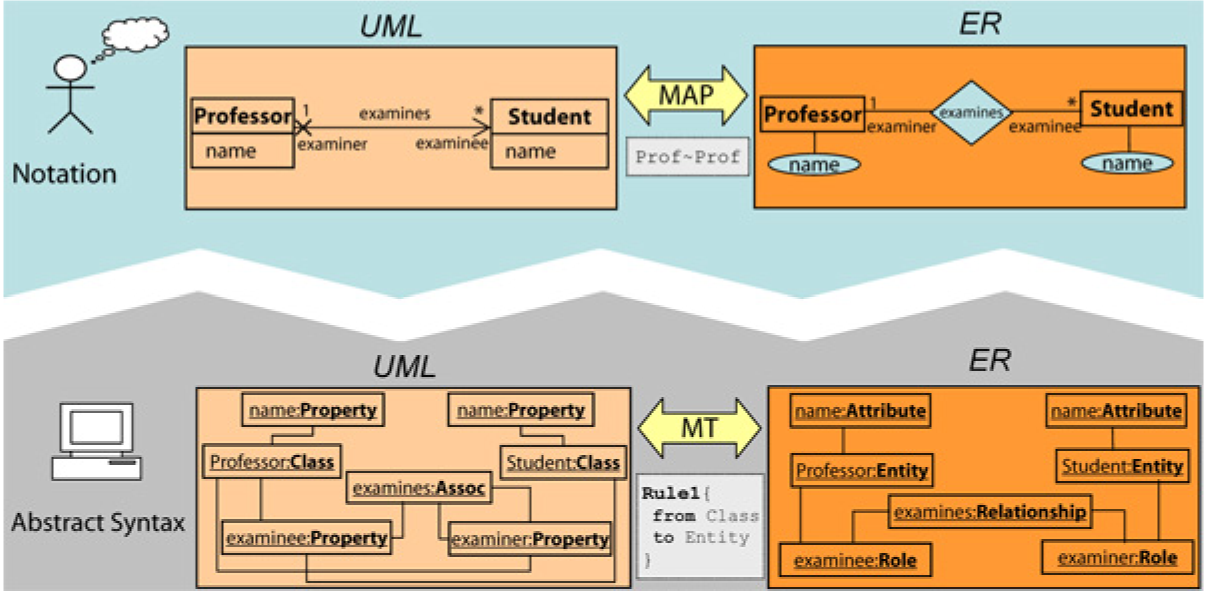
Table of contents

1. Overview

With the advent of Model-Driven Engineering (MDE) several model transformation approaches and languages have been developed in the last 5 years. Most of these existing approaches are metamodel-based with metamodels representing both an abstract syntax of the corresponding modeling language and also a data structure for storing models. However, this implementation specific focus makes it difficult for users of modeling languages to develop model transformations, because metamodels do not necessarily define all language concepts explicitly which are available for notation purposes. Therefore, we propose a by-example approach for defining inter-model mappings representing semantic correspondences between concrete domain models which is more user-friendly then directly specifying model transformation rules or mappings based on the abstract syntax. The inter-model mappings between domain models can be used to generate the model transformation rules, by-example, taking into account the already defined mapping between abstract and concrete syntax elements. With this approach the user’s knowledge about the notation of the modeling language is sufficient for the definition of model transformations regarding semantic correspondences. Hence, no detailed knowledge about the metamodel and the model transformation language is required.

2. Motivation

The following picture shows the gap between the way people reason about models and the way a computational unit processes a model internally. This uge gap leaded us to shift the task of model transformation specification from the meta level to the concrete syntax level.



3. MTBE Prototype / Downloads

In the following you find all files necessary to test our MTBE implementation. The following requirements are however given:
Eclipse 3.4M3
GMF 2.1M3
GEF 3.4M3
EMF 2.3.1
EMFT, OCL, ATL 2006, AMW, AM3 (latest versions)

In order to get the Higher Order Transformation to run, an adjustment has to be made to the Eclipse global preferences. For Ant there has to be the ATLExtractor.jar added as an external library. This is because our prototype executes an Ant script in order to launch the ATL transformation and extract the generated ATL code from the ATL model. Also we still work on the HOT so it produces just a basic transformation code at this moment.

er_gmf_editor.zip: a sample ER editor providing all GMF and EMF models for generating a complete editor

uml_gmf_editor.zip: a sample UML editor

workspace_uml2er_project.zip: the GMF project generated by our MTBE plug-in for ER und UML to work with org.tuwien.mtbe4.project_1.0.0.jar: this is the core eclipse plug-in that has to be copied into the eclipse plugin folder

org.tuwien.mtbe4.featureReasoning_1.0.0.jar: Reasoning Extension for the example below

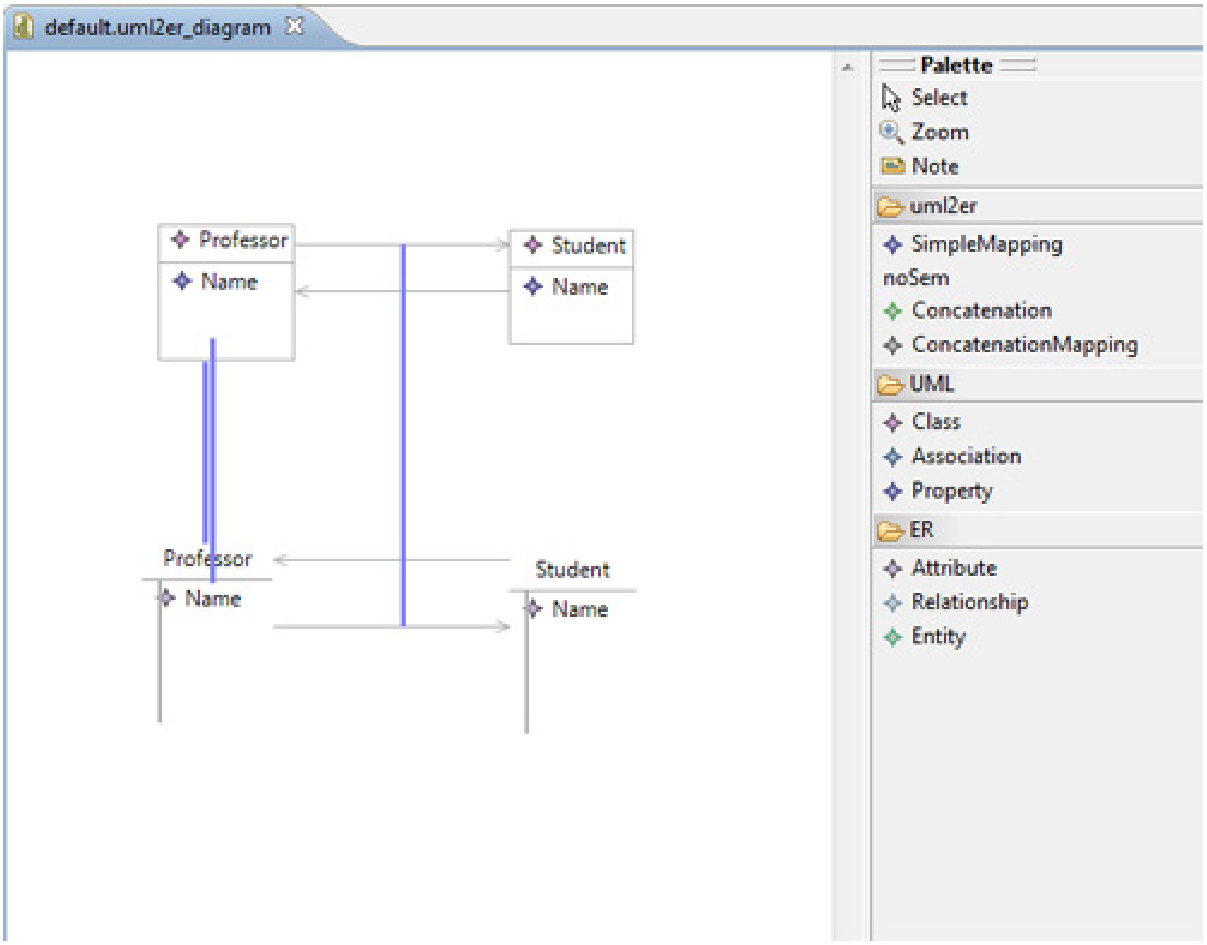
org.tuwien.mtbe5.uml2er_1.0.0.jar

org.tuwien.mtbe5.uml2er.diagram_1.0.0.200802201558.jar

We will soon provide a manual how to use the MTBE plug-in in practice.

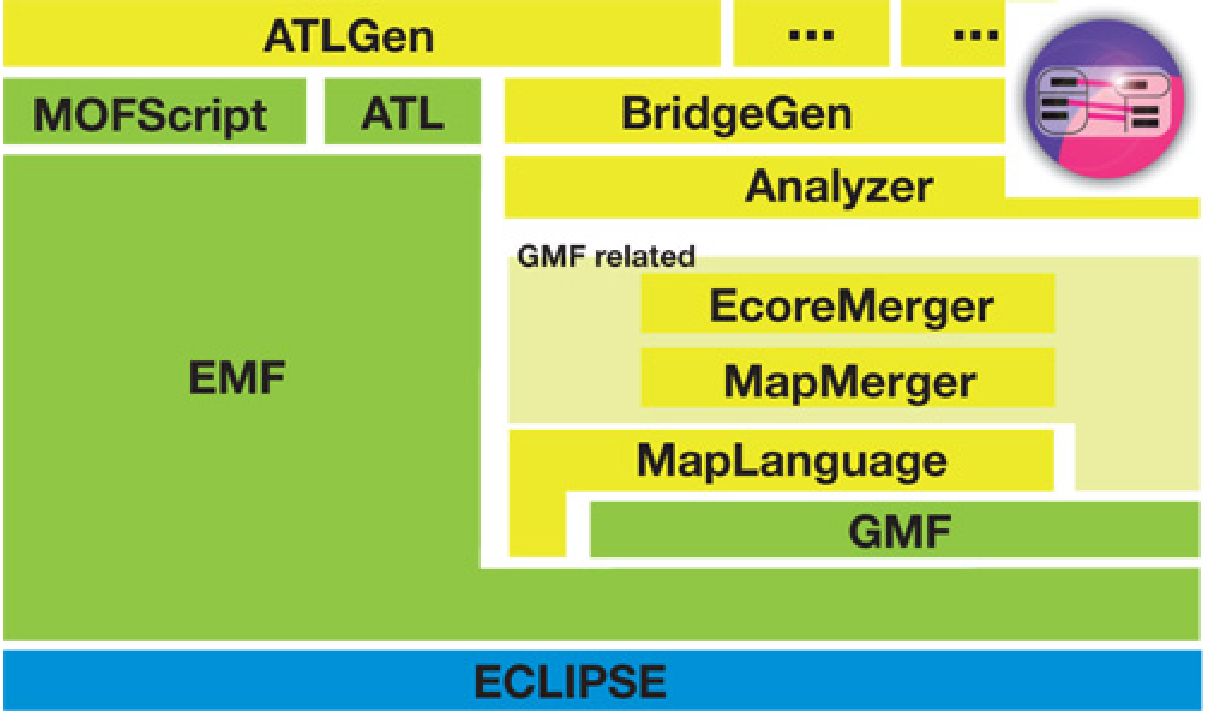
4. Frontend GMF Mapping Editor

Early prototype of the visual GMF mapping editor:



5. Framework components

The next graphic depicts the components of MTBE we are relying upon and the ones we are currently implementing.



6. Publication

`M. Wimmer, M. Strommer, H. Kargl, G. Kramler:` **`Towards Model Transformation Generation By-Example`** `, HICSS-40 Hawaii International Conference on System Sciences, Hawaii, USA, January 2007` [PDF]