MOMENT-QVT: a prototype for the QVT Relations language

Contributed by Artur Boronat
Monday, 11 June 2007
Last Updated Monday, 11 June 2007

In MDA, model transformations have become a relevant issue by means of the standard Query/Views/Transformations (QVT). Since a software artifact can be viewed as a model definition, model transformations are the basic mechanism that permits the manipulation of model-based software artifacts.

The MOMENT-QVT tool is a model transformation engine that provides partial support for the QVT Relations language. MOMENT-QVT implements the metamodel definition QVT, given in the QVT standard, and provides an editor for the QVT Relations language, which permits defining model transformations between EMF metamodels. Fig. 1 shows the editor of the MOMENT-QVT tool, which provides: syntax coloring, editing facilities and parsing facilities. For example, when a model transformation definition is not well-defined, the editor indicates which line contains the error.
Fig. 1. MOMENT-QVT: QVT Editor.

To execute a model transformation that is defined with the QVT Relations language, the metamodels that are referenced in the transformation definition have to be registered in EMF, previously. This can be achieved in several ways: by generating the plugin of the metamodel by means of EMF, or by using the EMF dynamic capabilities, see this article for an introduction. Metamodels can also be registered by means of the MOMENT Extensions facilities, which appear as a pop-up menu when we choose an Ecore model, as shown in Fig. 2. Fig. 2. Registering an Ecore model.

Once the model transformation is defined by using the concrete syntax of the QVT Relations language, we have to parse it as shown in Fig. 3, generating a QVT model definition.After obtaining the QVT model definition that corresponds to the user-defined model transformation, the user can invoke the model transformation by using the invocation wizard that is shown in Fig. 4. To invoke a model transformation, the user has to choose the ModelGen operator in the Operator name panel. After this, the user has to provide the file that contains the QVT model definition, which defines the model transformation. Depending on the definition of the model transformation, the user has to provide the input parameters and the output parameters.Fig. 3. Generation of the QVT model that represents a model transformation.
Fig. 4. Invocation of a QVT transformation.

MOMENT-QVT provides support for traceability, in the sense that a traceability model definition, which records what objects of the target model definition have been generated from objects of the source model definition, is generated in an automated way during a model transformation. Fig. 5 presents the traceability editor of the MOMENT framework. Fig. 5 shows the traceability model definition that is generated by a model transformation that transforms an Ecore model into a model that represent a relational schema in the traceability editor. The traceability editor is constituted by three main frames, the left frame shows an input model definition of the transformation, the right frame shows the output generated model definition, and the frame in the middle shows the traces that relate elements of the input model definition to elements of the target model definition. Traces also provide information about the transformation rule (or relation) that has been applied to source objects to generate the corresponding target objects. Fig. 5. MOMENT-QVT: traceability model