# MOLA 2 Tool

Audris Kalnins, Agris Sostaks[1], Elina Kalnina, Edgars Celms[1] and Oskars Vilītis[1]

MOLA 2 Tool is an academic model transformation tool implementing the MOLA transformation language. It is a principally new version of a tool having a new user-friendly development environment and an efficient compiler. The main existing and expected application area is to serve as a transformation support for MDSD process, another area is the transformation-based generic tool development.

## MOLA Language

**MOLA** is a graphical model transformation language developed at the University of Latvia, IMCS [1,2]. Similarly to most of model transformation languages, MOLA is based on pattern matching. However, its main distinguishing feature is the use of simple procedural control structures governing the order in which pattern matching rules are applied to the source model. The basic idea of MOLA is to merge traditional structured programming as a control structure with pattern-based transformation rules. The key language element is a graphical loop concept. Active usage of MOLA language since 2004 for various MDSD related tasks has confirmed its easy readability and low learning curve. There have been only few extensions necessary for the language during this period.

## MOLA 2 Tool

Since the first presentation of MOLA tool in ECMDA'05 the MOLA support has been completely rebuilt. A new significantly more user friendly Transformation Development Environment (a set of graphical editors) has been built and a set of several optimizing compilers to various runtime environments have been developed.

The new Transformation Development Environment (TDE) has been built on the basis of METAclipse tool building framework [3], which also has been developed by the University of Latvia, IMCS. METAclipse is a metamodel and transformation based tool building platform, which is specially fit for the support of complicated graphical domain specific languages, and MOLA is such a language. From the technical point of view, METAclipse is a set of Eclipse plugins which extend the functionality of standard Eclipse components EMF, GEF and partially, GMF [4,5,6]. It contains advanced presentation engines, which support graphical diagram building, property editing and all other diagram and model related facilities. More precisely, the engines perform all the various visualisation and user interaction related tasks in a standard way typical to Eclipse environment, they do these jobs on the basis of a fixed presentation metamodel. However, the main functionality of a tool based on METAclipse is defined by transformations, which link the domain and presentation (visualisation) models in the tool, fill up property dialogs, and process the updated property values. In METAclipse framework these tool-specific transformations are built in MOLA.

[1] supported partially by ESF (European Social Fund), project 2004/0001/VPD1/ESF/PIAA/04/NP/3.2.3.1/0001/0063

Figure 1 shows the general architecture of MOLA 2 Tool. The editor part of TDE supports the two diagram kinds used in MOLA: class diagrams for defining the combined source/target metamodel and MOLA diagrams displaying the MOLA procedures constituting a transformation. The transformation technology used in METAclipse permits to make the diagram editing very user friendly and safe. For example, very dedicated prompters based on the context-sensitive part of MOLA syntax offer only the relevant elements for referencing, element name modifications are automatically propagated to all referencing elements and so on. Usage experience of about one year has confirmed the high quality of MOLA 2 TDE in practice.

At the same time, since transformations defining MOLA TDE are built in MOLA itself (a bootstrapping approach is used!), the development of MOLA 2 TDE has served as one of large-scale applications of MOLA.

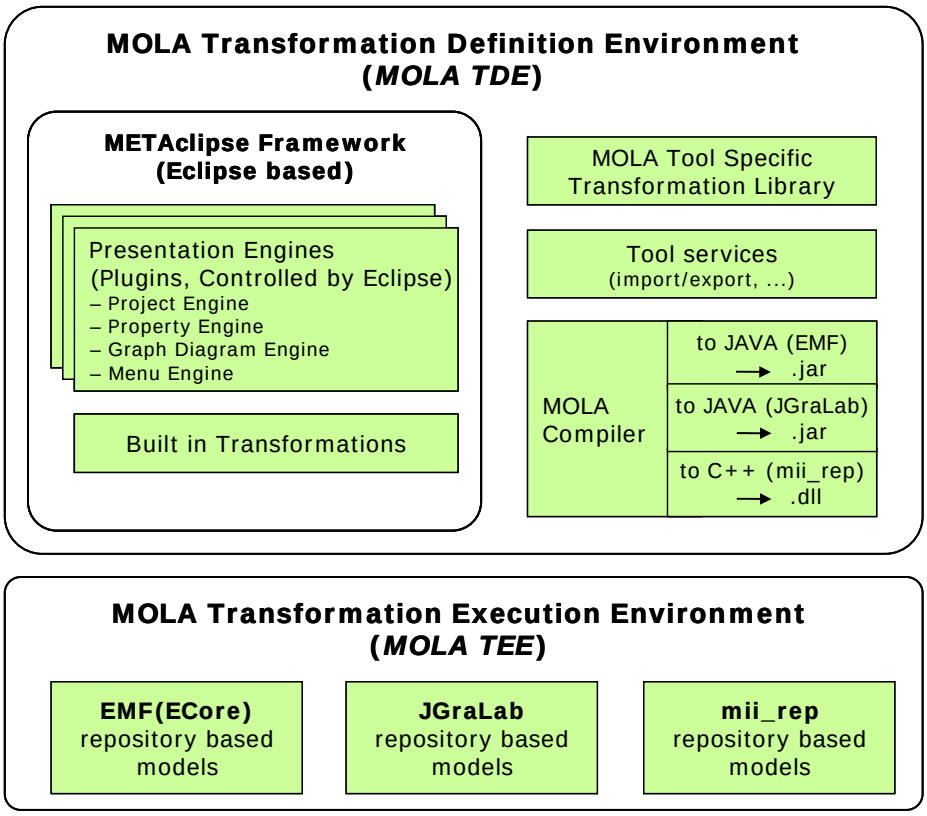


**Figure 1.** MOLA2 Tool architecture

A screenshot of MOLA 2 Tool (TDE part) displaying one metamodel fragment and one MOLA diagram is shown in Figure 2.

Another crucial part of MOLA 2 Tool is the MOLA compiler. A completely new optimizing MOLA compiler has been built, which supports an efficient pattern match. The compiler back-end generates OOPL code from a MOLA transformation, which after the compilation is capable of executing against an appropriate model repository. Actually, three different back-ends have been built. The oldest one generates C++ code against the API of high performance custom model repository mii_rep [7] built

by UL IMCS. Two other back-ends generate Java code, one against the open source high performance graph/metamodel based repository JGraLab [8] built at the University of Koblenz. However, the most important for wide usage of MOLA is the newest back-end generating Java against the API of Eclipse EMF, which is the most popular model repository kind so far. In all cases the final compilation step also can be invoked from TDE, thus the MOLA compiler produces the complete executable library (jar or dll) which can be run against the relevant repository.

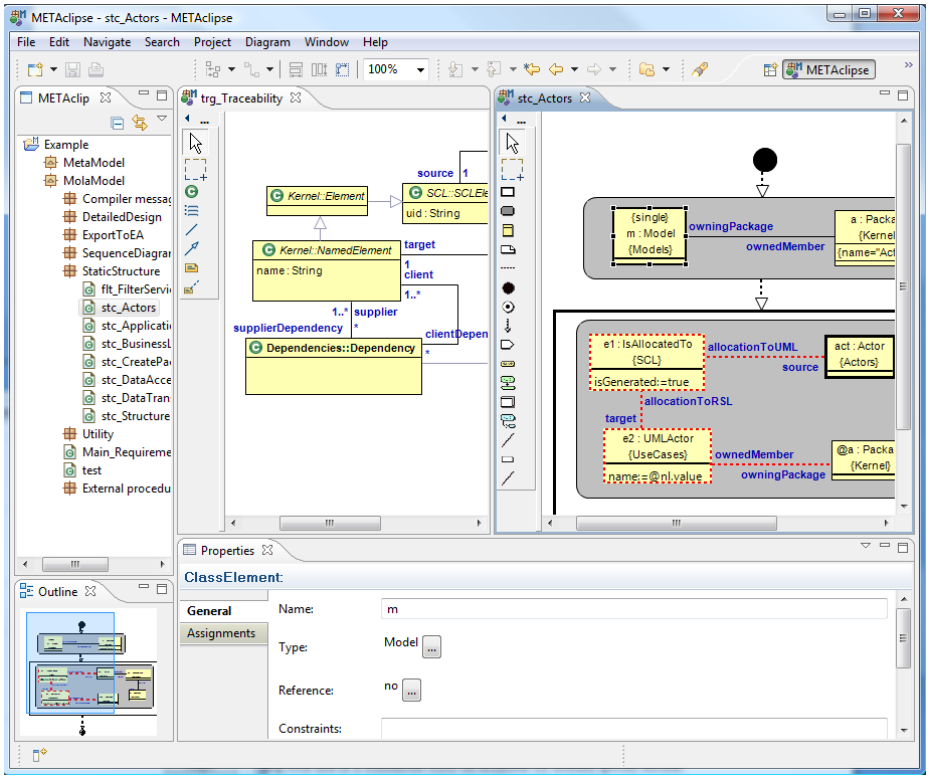


**Figure 2.** MOLA2 TDE screenshot

The MOLA Transformation Execution Environment (TEE) is dependent on the chosen repository for holding the models and metamodels. The widest possibilities are offered by EMF, which is the cornerstone for any Eclipse-based model management. In particular, a start-up model (instance) editor can be built very easy, as long as its metamodel is available in EMF. In addition, there are large libraries of models in EMF, built for other transformation tools. It is planned to extend TDE by a facility for converting MOLA transformation into an Eclipse plugin. Then it will be easy to incorporate MOLA transformations into various Eclipse-based modelling tools, for a flexible support of various model driven development scenarios.

The two other kinds of repositories have their own support for model management, for example, mii_rep is integrated to a model browser, which can be used to analyse and create models and invoke transformations. However, the main use case for these repositories is to facilitate the incorporation of MOLA transformations into other tools

based on these repositories. Thus, the mii_rep repository is used for transformation execution inside METAclipse (via JNI (Java Native Interface) mechanism, see [9]).

## MOLA usage experience and perspectives

One large-scale MOLA usage has already been mentioned – it is the development of transformations for MOLA 2 TDE inside METAclipse. The mii_rep based version was used there because of the bootstrapping requirements (the transformations were initially developed using the previous TDE).

Another large-scale application of MOLA is in the 6-th Framework IST project ReDSeeDS [10]. One of the aspects of this project is a true transformation based software development along refined MDA guidelines. For example, there a special Requirements model (in a UML-like language RSL [11], this model is a sort of refinement for OMG CIM) is used. Non-trivial transformations are used to obtain the initial version of Architecture model (in a rich subset of UML) from the Requirements model. Similarly, the Architecture model can be transformed to a Detailed Design model (an analogue to OMG PSM). The first versions of supporting MOLA transformations have already been built and tested in practice. Here the JGraLab repository is used, since some other tools in ReDSeeDS also use this repository.

The widest applicability of MOLA is expected for the EMF based version, due to reasons already mentioned.

## References


[1] Kalnins A., Barzdins J., Celms E.: Model Transformation Language MOLA. Proceedings of MDAFA 2004, Vol. 3599, Springer LNCS, 2005, pp. 62–76.

[2] UL IMCS, MOLA pages, http://mola.mii.lu.lv/

[3] Kalnins A., Vilitis O., Celms E., Kalnina E., Sostaks A., Barzdins J. Building Tools by Model Transformations in Eclipse. Proceedings of DSM'07 workshop of OOPSLA 2007, Montreal, Canada, Jyvaskyla University Printing House, 2007, pp. 194–207.

[4] Eclipse Modeling Framework (EMF, Eclipse Modeling subproject), http://www.eclipse.org/emf/

[5] Graphical Editor Framework (GEF, Eclipse Tools subproject), http://www.eclipse.org/gef/

[6] Graphical Modeling Framework (GMF, Eclipse Modeling subproject), http://www.eclipse.org/gmf/

[7] Barzdins, J., Barzdins, G., Balodis, R., Cerans, K., Kalnins, A., Opmanis, M., Podnieks, K.: Towards Semantic Latvia. Proceedings of Seventh International Baltic Conference on Databases and Information Systems, Communications, Vilnius, Lithuania, O. Vasileckas, J. Eder, A. Caplinskas (Eds.), Vilnius, Technika, 2006, pp. 203–218.

[8] Universität Koblenz-Landau, Institute for Software Technology, Graph Laboratory http://www.uni-koblenz.de/FB4/Institutes/IST/AGEbert/MainResearch/GraphTechnology/GraLab

[9] Java Native Interface Specification, http://java.sun.com/j2se/1.5.0/docs/guide/jni/spec/jniTOC.html

[10] ReDSeeDS. Requirements Driven Software Development System. European FP6 IST project. http://www.redseeds.eu/, 2007.

[11] Smialek M., Bojarski J., Nowakowski W., Ambroziewicz A., Straszak T.: Complementary Use Case Scenario Representations based on Domain Vocabularies. Proceedings of MODELS 2007, Vol. 4735, Springer LNCS, 2007, pp.544 – 558.