The VIATRA2 Transformation Framework
Model Transformation by Graph Transformation

Dániel Varró[1,2], András Balogh[1,2], and András Pataricza[1,2]

[1] Budapest University of Technology and Economics
Department of Measurement and Information Systems
H-1117 Budapest, Magyar tudósok körútja 2.
{varro,abalogh,pataric}@mit.bme.hu
[2] OptXware Research and Development LLC.
www.optxware.com
{varro,balogh,pataric}@optxware.com

Abstract. VIATRA2 (VIsual Automated model TRAnsformations) is a general-purpose model transformation engineering framework that aims at supporting the entire life-cycle, i.e. the specification, design, execution, validation and maintenance of transformations within and between various modeling languages and domains in the MDA. This paper reports on the current status of the tool and sketches some future directions.

1 VIATRA2: A tool for MDA transformations

The increasing success of the Model Driven Architecture (MDA) highly relies on an effective tool support for engineering models of different modeling languages, application domains and executing software platforms. While model engineering (*modelware*) has already become a widely accepted software engineering discipline, many researchers and practitioners identify the increasing need for well-founded engineering methods and powerful tools for model transformation engineering (*transware*).

Objectives of Viatra2. The objective of the VIATRA2 (VIsual Automated model TRAnsformations) is a general-purpose model transformation engineering framework that aims at supporting the entire life-cycle, i.e. the specification, design, execution, validation and maintenance of transformations within and between various modeling languages and domains in the MDA. The current paper reports on the state-of-the-art of the VIATRA2 tool in achieving this very ambitious goal.

Tool history. The tool is being developed at the Fault Tolerant Systems Research Group at the Budapest University of Technology and Economics. Our research on model transformations of UML models is dated back to the European ESPRIT project HIDE [5]. The first versions of the tool [7] (written in Prolog) were developed between 2000 and 2003 and primarily focused on designing automated transformations of UML models into various mathematical domains of model analysis tools.

Based on these initial experiments, VIATRA2 has been reengineered from scratch since 2004 (following the guidelines of a PhD thesis [10]) to better meet the requirements of MDA transformations. VIATRA2 is now written in Java and fully integrated into the Eclipse framework [1] and accepts models of several off-the-shelf industrial modeling tools. Our tool is intended to serve as a powerful academic prototype publicly available for academic use.

Currently, VIATRA2 is available as part of the Eclipse GMT Subproject [2].

2 Models and transformations in Viatra2

Model descriptions. VIATRA2 uses the *VPM metamodeling* approach [11] for describing modeling languages and models. The main reason for selecting VPM instead of a MOF-based metamodeling approach is that VPM supports *arbitrary metalevels in the model space*. As a direct consequence, models taken from conceptually different domains (and/or technological spaces) can be easily integrated into the VPM model space. The flexibility of VPM is demonstrated by a large number of already existing model importers accepting the models of different BPM formalisms, UML models of various tools, XSD descriptions, and EMF models.

Ongoing development aims at providing support for domain-specific languages within the VIATRA2 framework. Initial experiments in this direction have been carried out when designing the current visual editor for the VPM model space.

Specification of model transformations. VIATRA2 combines visual rule and pattern-based paradigm of *graph transformation (GT)* [8] and the very general, high-level formal paradigm of *abstract state machines (ASM)* [6] into a single framework for capturing transformations within and between modeling languages.

- *Queries* on models are intuitively captured by graph patterns (that may have negative conditions and interact with other patterns in turn).
- *Elementary model manipulations* specified by graph transformation rules can also reuse predefined graph patterns in their left-hand side and right-hand side graphs.
- Then *complex transformation programs* are assembled by using abstract state machine constructs that provide higher-level control structures for elementary manipulation steps. Both ASM and GT rules are allowed to have input and output parameters to support information hiding.
- *Code generation* is treated as ordinary model-to-code model transformations, and it is supported by intelligent (model-driven) print ASM rules, and a code formatter mechanism to split the generated code into different source files.

A unique feature of VIATRA2 is the support of *generic and meta-transformations* [12] that allow type parameters and manipulate transformations as ordinary models, respectively. Initial experiments have been carried out to gain more practical experience in this novel field.

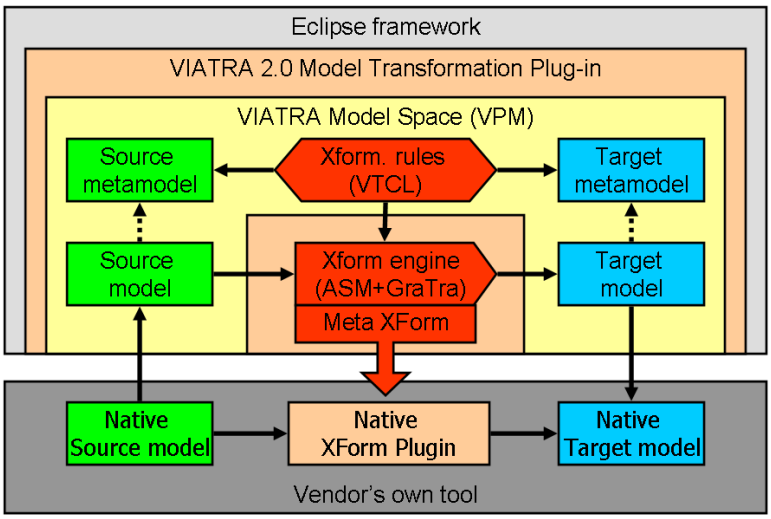

Fig. 1. An overview of the Viatra2 transformation execution

Execution of transformations. Transformations are primarily executed within the framework (see Fig. 1) by using the VIATRA2 interpreter, which uses constraint satisfaction techniques for matching graph patterns (which is typically the most expensive step of a model interpreter). Furthermore, a first detailed experiment [4] have been carried out to externalize transformations by generating platform-specific, stand-alone transformation plugins (transformers) in the evolving EJB3 standard. The main practical relevance of such transformers from an MDA perspective is that they can be integrated into an existing piece of software without the use of the entire VIATRA2 framework.

While VIATRA2 transformations are unidirectional, ongoing research aims at developing a bidirectional and incremental model transformation approach built on top of the current framework. Moreover, the CheckVML tool [9] will soon be integrated into VIATRA2 to support the model checking based formal validation of model transformations.

Extension mechanisms. VIATRA2 provides an extensible framework for handling a large variety of MDA models and transformations. Arbitrary model importers can be easily implemented using the simple metamodeling API of VPM metamodeling core, and a large set of importer plugins are already available. Furthermore, VIATRA2 transformations may call external Java methods if necessary to integrate external tools into a single tool chain.

User interface. Currently, VIATRA2 provides a textual language (called VTCL [3]) for describing transformations and a separate language (called VTML) for representing models, and metamodels, and a graphical user interface for viewing and editing the VPM model space including the creation and deletion of various model elements (and traditional undo and copy operations). This GUI provides a tree view of the model space. Rich textual editors support the editing of VTCL transformations, and VTML models and meta models. Support for a graphical transformation editor of GT and ASM descriptions is an ongoing activity.

3 Applications of Viatra2

Model transformations already carried out using the VIATRA2 framework can be grouped into the two main categories:

- *PIM-PSM mappings for model-driven design.* Traditional PIM-to-PSM mappings are being carried out for the model-driven design and deployment of dependable embedded components in e.g. automotive systems.
- *PSM to source code generation* PSM-to-code mappings include initial code generation support from UML models to EJB3 and the automation of BPM/BPEL models. As well as the generation of middleware configuration files for various platforms.
- *PIM-semantic domain mappings for formal model analysis.* The traditional line of research in our group aims at a transformation-based verification and validation of models that allows an early validation of model-level design decisions. Sample transformations project behavioral UML models (taken from IBM Rational XDE or Software Architect) or BPM models (created in IBM Business Integrator Modeler) via dataflow networks into various back-end analysis tools to (i) automatically catch conceptual flaws, (ii) perform security evaluation according to the Bell-LaPadula approach, or (iii) to highlight dependability bottlenecks and assess error propagation paths.

These typical transformations arise in various application domains of VIATRA2 including the following ones:

- *Business Process Models* Business Process Modeling tools support the visualization and simulation of business processes and workflows. Our work (partially supported by an IBM Faculty Award) aims at the extension of BPM tools with formal analysis and direct code generation for workflow automation languages like BPEL. Research work has also be done for security analysis of BPM models and fault simulation in workflows.
- *Embedded systems* In the scope of the DECOS European IP we developed various domain-specific editors for modeling embedded systems, and used VIATRA2 as a transformation and tool integration platform for PIM to PSM mapping and software-hardware integration in the embedded domain. VIATRA2 is also used for model validation and consistency checking based on transformations.

– *Service-Oriented Architecture* VIATRA2 has been successfully applied in the SOA domain for generating analysis models from UML-based models and did performance and availability analysis of applications. We did also research work in the field of service analysis and deployment, and middleware configuration generation. This work is partly supported by the SENSORIA European IP and the SA Forum group.

4 Future Development

Based on our experiences with VIATRA2 we will carry out further development in the near future to increase the usability and interoperability of the framework.

– *Tight coupling with EMF* As Eclipse Modeling Framework (EMF) is widely used by several modeling tools we try to create a strong coupling between the EMF and VPM model stores. Currently, VIATRA2 imports EMF models from files, using an appropriate importer plugin. The goal of the development is the online synchronization between EMF and VPM models.
– *Native transformation support for EMF* Based on our experience on the generation of native transformation plugins for the EJB 3.0 platform we plan to extend this generation for the EMF platform to create transformation modules executed directly on EMF models. This enables the integration between various EMF-based modeling tools.
– *Visual debugging of transformation* With the refactoring and upgrading of the transformation interpreter we will integrate it with the Eclipse Debug framework. This will enable the visual debugging of VIATRA2 transformations.
– *Domain Specific Modeling framework* We are currently developing an extension to VIATRA2 that will support the definition of domain specific modeling languages and their visual notations. This allows the automatic generation of graphical editors for domain specific modeling.
– *Transformation development by example* One of our further research directions is the support for the "transformation by example" approach [**?**]. This enables the definition of transformations without a specific transformation language only by using example source and target models.

References

1. The Eclipse project. `www.eclipse.org`.
2. *VIATRA2 Framework*. An Eclipse GMT Subproject (`http://www.eclipse.org/gmt/`).
3. A. Balogh and D. Varró. Advanced model transformation language constructs in the VIATRA2 framework. In *ACM Symposium on Applied Computing — Model Transformation Track (SAC 2006)*, pp. 1280–1287. ACM Press, Dijon, France, 2006.

4. A. Balogh, G. Varró, D. Varró, and A. Pataricza. Compiling model transformations to EJB3-specific transformer plugins. In *ACM Symposium on Applied Computing — Model Transformation Track (SAC 2006)*, pp. 1288–1295. ACM Press, Dijon, France, 2006.
5. A. Bondavalli, M. Dal Cin, D. Latella, I. Majzik, A. Pataricza, and G. Savoia. Dependability analysis in the early phases of UML based system design. *International Journal of Computer Systems - Science & Engineering*, vol. 16(5):pp. 265–275, 2001.
6. E. Börger and R. Stärk. *Abstract State Machines. A method for High-Level System Design and Analysis*. Springer-Verlag, 2003.
7. G. Csertán, G. Huszerl, I. Majzik, Z. Pap, A. Pataricza, and D. Varró. VIATRA: Visual automated transformations for formal verification and validation of UML models. In J. Richardson, W. Emmerich, and D. Wile (eds.), *Proc. ASE 2002: 17th IEEE International Conference on Automated Software Engineering*, pp. 267–270. IEEE Press, Edinburgh, UK, 2002.
8. H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg (eds.). *Handbook on Graph Grammars and Computing by Graph Transformation*, vol. 2: Applications, Languages and Tools. World Scientific, 1999.
9. Á. Schmidt and D. Varró. CheckVML: A tool for model checking visual modeling languages. In P. Stevens, J. Whittle, and G. Booch (eds.), *Proc. UML 2003: 6th International Conference on the Unified Modeling Language*, vol. 2863 of *LNCS*, pp. 92–95. Springer, San Francisco, CA, USA, 2003.
10. D. Varró. *Automated Model Transformations for the Analysis of IT Systems*. Ph.D. thesis, Budapest University of Technology and Economics, Department of Measurement and Information Systems, 2004.
11. D. Varró and A. Pataricza. VPM: A visual, precise and multilevel metamodeling framework for describing mathematical domains and UML. *Journal of Software and Systems Modeling*, vol. 2(3):pp. 187–210, 2003.
12. D. Varró and A. Pataricza. Generic and meta-transformations for model transformation engineering. In T. Baar, A. Strohmeier, A. Moreira, and S. Mellor (eds.), *Proc. UML 2004: 7th International Conference on the Unified Modeling Language*, vol. 3273 of *LNCS*, pp. 290–304. Springer, Lisbon, Portugal, 2004.