Model Transformations for Intelligent Systems Engineering

Nikita O. Dorodnykh
and Alexander Yu. Yurin
Matrosov Institute for System Dynamics and Control Theory,
Siberian Branch of the Russian Academy of Sciences (IDSTU SB RAS),
Irkutsk, Russia
tualatin32@mail.ru
iskander@icc.ru

Nikolai Yu. Pavlov, Sergey A. Korshunov
and Dmitriy Yu. Sopp
CentraSib LLC.,
Irkutsk, Russia
Telephone: (3952) 603–911
info@centrasib.ru

***Abstract*—The paper discusses the application of model transformations in the process of intelligent systems engineering. The model-driven approach is used as a basic methodology and the own its implementation is proposed. The conceptual models presented in XML-like formats are used as the initial data. Particular attention is paid to the first stage in the chain of model transformations: the stage of the formation of a computation-independent model. A new domain-specific language – Transformation Model Representation Language (TMRL) and special tools are used for implementing model transformations. The description of the main TMRL constructions and an example of its application are presented.**

***Keywords*—model transformation, code generation, intelligent system, knowledge base, domain-specific language.**

I. Introduction

The model transformation is one of the main principles of the Model Driven Engineering (MDE) [1]. MDE approach is based on the use of software information models of varying degrees of abstraction as the main artifacts in the development of software systems. In this case, the process itself is a sequence (chain) of transformation of these models. In this paper we proposed to consider the application of model transformations in the process of intelligent systems engineering, in particular, for development of knowledge bases (KB) with the use of transformation of conceptual models presented in XML-like formats. The main results are analysis of related works in the field of model transformation and the new domain-specific declarative language for the description of transformations – Transformation Model Representation Language (TMRL), designed for transforming conceptual models to the KB.

II. Model Transformation

Recently, in the field of software engineering there has been a tendency to use approaches that consider models not only as artifacts of documentation (technical specifications), but also as central artifacts in software development providing automatic synthesis of program codes. This allows any to significantly reduce development time, reduce the risk of programming errors, to involve end-users in the process of software engineering. These approaches are related to the Model Driven engineering (MDE) or Model Driven Development (MDD) areas [2]. Today, the main MDE implementations (initiatives) are the following: OMG Model Driven Architecture (MDA), Eclipse Modeling Framework (EMF), Model Integrated Computing (MIC), Microsoft Software Factories, JetBrains MPS.

The central concepts of MDE/MDD are:

- A model is an abstract description of the some characteristics of the system (process) in a formal language. As a rule, the models are visualized with the aid of a certain graphic notation and serialized in XML.
- A metamodel is a model of a language used to create models (model of models).
- A meta-metamodel is a language that describes metamodels. The most common languages for metamodeling are: MOF (Meta-Object Facility), Ecore, KM3 (Kernel Meta Meta Model), etc.
- A four-layer metamodeling architecture is the concept that defines the different layers of abstraction (M0-M3), where the objects of reality are represented at the lowest level (M0), then the level of models (M1), the level of metamodels (M2) and the level of the meta-metamodel (M3).
- A model transformation is automatic generation of a target model from a source model with the accordance of a set of transformation rules [3]. In this case, each transformation rule describes the correspondence between the elements of the source and target metamodels.

A more detailed description of these concepts can be found in [2], [4].

There are many works devoted to the model transformations. At the same time, model transformations can be considered from different viewpoints.

In particular, there are two types of transformation [5]:

- Model-to-Model (M2M);
- Model-to-Text (M2T) and Text-to-Model (T2M). In this case, the output text can be in the form of a source code, documentation, specifications, and etc.

Two types of model transformations in accordance with the modelling languages used to describe the source and target models [5]:

- an endogenous transformation is a transformation between models that are using one modeling language;
- an exogenous transformation is a transformation between models that are using different modeling languages.

The model transformations can be classified according to the abstraction level on which the source and target model are resided [5]:

- a vertical transformation is a transformation of models of different abstraction levels;
- a horizontal transformation is transformation of models of the same abstraction level.

The model transformations can also be classified according to the transformation direction:

- an unidirectional transformation is a transformation where only the target model can be obtained from the source model;
- a bidirectional transformation is a transformation where the target model can be obtained from the source model and vice versa.

Currently, there are some ways to implement the model transformation:

- using graph grammars (graph rewriting) (e.g., VIsual Automated model TRAnsformations (VIATRA2) [6], Graph REwriting And Transformation (GReAT) [7], etc.);
- using visual design of transformation rules and category theory (e.g., Henshin [8]);
- using transformation standards (e.g., Query/View/Transformation [9]);
- using hybrid (declarative-imperative) approach for specifying and constructing transformation rules (e.g., ATLAS Transformation Language [10]);
- using declarative and procedural programming languages [11];
- using languages for transforming XML documents (e.g., eXtensible Stylesheet Language Transformations [12], etc.).

The transformations constructed using these ways should satisfy the following main requirements [13], [14], [15]:

- completeness: it should allows one to represent any necessary transformation in accordance with the defined models;
- formality: it should allows automatic execution;
- flexibility: it should not depends on a specific subject domain.

In this paper, it is proposed to consider the application of model transformations in the process of intelligent systems engineering and KBs based on the implementation of MDE / MDD.

III. Model Transformations for Intelligent Systems Engineering

The MDE/MDD implementation used is based on the principles of MDA [16] and assumes a clear separation of three levels (viewpoints) of the software representation:

- A computation-independent level, which is a description of the basic concepts and relationships of the subject domain, we will express it in the form of ontologies. Various conceptual models can be used to automate its formation. It is proposed to limit the number of conceptual models by XML-like formats, in particular, for representation: UML models – XMI (XML Metadata Interchange), concept maps – CXL (Concept Mapping Extensible Language), event trees – ETXL (Event Tree Mapping Extensible Language).
- A platform-independent level, which provides a representation of the domain model in the context of the formalism used to represent knowledge. In particular, logical rules or frames. It is advisable to use problem-oriented notations or UML profiles, in particular RVML (Rule Visual Modeling Language) [17].
- A platform-dependent level representing a formalized description of KBs, taking into account a certain software platform. In the context of intelligent systems engineering, such a platform is the programming languages for KBs, for example, CLIPS.

Thus, the process of intelligent systems engineering is a sequential transition between the considered levels and can be described by the following sequence of steps:

- Formation of a conceptual model by means of third-party programs. At this step some problem-oriented notations are used: UML, concept maps or memory cards, event trees. At the end of the step, the conceptual model is presented in XML-like formats: XMI (StarUML, IBM Rational Rose), CXL (IHMC CmapTools), ETXL (ET-Editor).
- Analysis of XML document of conceptual model with identification of the concepts and relations. A computation-independent model (CIM) is formed on the basis of the selected concepts, represented in the form of ontology. The automatically generated ontology is edited to clarify it.
- Formation of a platform-independent model (PIM) based on ontology. This model depends on the formalism of knowledge representation, but does not take into account the features of languages and the tools for implementing these formalisms.
- Formation of a platform-specific model (PSM), taking into account the features of languages and means for implementing formalisms.
- Generation of program codes or specifications of KBs and intelligent systems based on generated models.

It is suggested to use model transformations to implement the transitions between the steps and to consider in detail the

transformations of the first step, which are related to the M2M type.

In the context of the MDA approach, it is recommended to use the OMG standard for implementation of M2M transformations, which is called QVT including Operational, Relational and Core model transformation languages.

However, programmers who use the QVT should:

- know the specific syntax of the model transformation language;
- be able to describe transformation rules with the aid of the model transformation language;
- know the additional languages, such as an Object Constraint Language (OCL) that can be used to construct the transformation rules;
- be able to describe metamodels for the source and target languages (to support the transformation process).

It should be noted that all model transformation languages is supported by a specific software tools, that, in turn don't provide an opportunity to visualize the development process, i.e. the transformation rules are defined in special text editors focused on programmers. The combination of these factors makes difficult to use these languages and tools in a practical way by end-users (e.g., subject domain experts, knowledge engineers, analysts, etc.), in particular, when developing KBs and ESs on the basis of conceptual models. So, in practice developers prefer "ad-hoc" solutions for particular tasks with using declarative and procedural programming languages.

We propose a new domain-specific language – TMRL to address these shortcomings.

IV. Transformation Model Representation Language

TMRL is focused on the representation and storage of the so-called a transformation model, which is a scenario (program) for transforming the source conceptual model to the target KB. Thus, TMRL transformation model is the core of the software component for transformation, providing analysis (parsing) of conceptual models and synthesis (generation) of the KB code [18].

The structure of the transformation model can be defined as follows:

$$M_T = \langle\ MM_{IN}, MM_{OUT}, T\ \rangle, \qquad (1)$$

where MM_{IN} is a metamodel of the source (input) conceptual model; MM_{OUT} is a metamodel of the target (output) model of knowledge representation (KB); T is an operator for model transformation (a set of rules).

The TMRL grammar belongs to the class of context-free grammars (LL(1) CF-grammars) [19]. The TMRL constructs allow one to describe the elements of the transformation model in a declarative form, in particular, the rules for the correspondence of metamodel elements, as well as the mechanism of interaction with previously developed (external) software components for the transformations. TMRL specifications meet the requirements of accuracy, clarity and completeness [20], i.e. the TMRL specifications contain all the necessary information (for the considered transformations) to solve the task, all objects of the model are well formalized, while the specifications are compact enough and at the same time understandable (readable).

The main difference between TMRL and existing model transformation languages is its ease of use, achieved through a limited set of elements. TMRL is not an extension of other languages and does not use the constructions of other languages, as other model transformation languages very often do, in particular, ATL uses the OCL. In addition, TMRL has human-readable syntax for the purpose of making the necessary modifications to the model of the transformation manually, if necessary. An additional feature of TMRL is the ability to describe interaction with previously developed software components of the transformation in support of the import of different formats of conceptual models.

A. Transformation model structure in TMRL

The transformation model in TMRL consists of three main blocks. Consider this structure in an example that describes the transformation of a UML class diagram into an ontology model (CIM). In this case, the language elements are highlighted in bold.

Block 1. Description of the elements and relationships of the source metamodel:

```
Source Meta-Model UML-diagram-class {
   Elements [
      Model,
      Class attributes (xmi.id, name),
      ...
   ]
   Relationships [
      Model is associated with Namespace.
          ownedElement,
      Namespace.ownedElement is associated
          with Class,
      DataType(xmi.id) is Attribute(type),
      ...
   ]
}
```

This block contains a description of the UML class diagram: "UML-diagram-class", including elements of the model ("Elements"). In this example, these are the "Model" and "Class" elements, the "Class" element has "xmi.id" and "name" properties. In addition to describing the elements, the source metamodel contains a description of the relationships between the elements of the metamodel ("Relationships"), including by identifiers, for example, linking an attribute to a data type ("DataType(xmi.id) is Attribute(type)").

Block 2. Description of elements and relationships of the target metamodel:

```
Target Meta-Model Ontology {
   Elements [
      Ontology attributes (about),
      Class attributes (id, label),
      ...
   ]
```

```
   Relationships [
      Ontology is associated with Class,
      ...
   ]
}
```

The block contains a description of the ontology model: "Ontology". The structure of the block is similar to the structure of the block of the source metamodel.

Block 3. Description of rules for transforming models:

```
Transformation UML-diagram-class to Ontology {
   Rule Model to Ontology priority 1 [
      Ontology(about) is Model or ModelElement
          .name
   ]
   Rule (Class, ModelElement.name) to Class
       priority 2 [
      Class(label) is Class(name) or
          ModelElement.name
      Class(id) is Class(name)
   ]
   ...
}
```

This block describes the rules for converting the elements of the source ("UML-diagram-class") metamodel to the target ("Ontology") metamodel.

The Knowledge Base Development System (KBDS) is used to support TMRL [21]. KBDS provides interactive visual construction of transformation models and their automatic generation on TMRL.

V. Conclusion

The paper describes the application of model transformations in the process of intelligent systems engineering and, in particular, KBs. The various conceptual models presented in XML-like formats are used as initial data. Particular attention is paid to the stage of formation of CIM. The detailed description of the new domain-specific language (TMRL) is provided. The KBDS is used as tool for supporting TMRL.

References

[1] L. G. Cretu and D. Florin, *Model-Driven Engineering of Information Systems: Principles, Techniques, and Practice*. Apple Academic Press, 2014.

[2] A. R. Da Silva, *Model-driven engineering: A survey supported by the unified conceptual model*. Computer Languages, Systems & Structures. 2015. Vol. 43. P. 139-155.

[3] A. Kleppe, J. Warmer and W. Bast, *MDA Explained: The Model-Driven Architecture: Practice and Promise*, 1rd ed. New York: Addison-Wesley Professional, 2003.

[4] J. Sprinkle, B. Rumpe, H. Vangheluwe and G. Karsai, *3 Metamodelling. State of the Art and Research Challenges*. Model-Based Engineering of Embedded Real-Time Systems. 2010. P. 57-76.

[5] T. Mens and P. V. Gorp, *A Taxonomy of Model Transformations*. Electronic Notes in Theoretical Computer Science. 2006. Vol. 152. P. 125-142.

[6] D. Varro and A. Balogh, *The model transformation language of the VIATRA2 framework*. Science of Computer Programming. 2007. Vol. 63, No. 3. P. 214-234.

[7] D. Balasubramanian, A. Narayanan, C. Buskirk and G. Karsai, *The graph rewriting and transformation language: GReAT*. Electronic Communications of the EASST. 2006. Vol. 1. P. 1-8.

[8] T. Arendt, E. Biermann, S. Jurack, C. Krause and G. Taentzer, *Henshin: advanced concepts and tools for in-place EMF model transformations*. Processing of the 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2010) / Lecture Notes in Computer Science, Springer Berlin Heidelberg. 2010. Vol. 6394. P. 121-135.

[9] *Query/View/Transformation (QVT)*. 2016. Available at: http://www.omg.org/spec/QVT/ (accessed 07.12.2017).

[10] F. Jouault, F. Allilaire, J. Bézivin and I. Kurtev, *ATL: A model transformation tool*. Science of Computer Programming. 2008. Vol. 72, No. 1. P. 31-39.

[11] A. F. Berman, O. A. Nikolaychuk and A. Y. Yurin, *Intelligent planner for control of failures analysis of unique mechanical systems*. Expert Systems with Applications. 2010. Vol. 37, No. 10. P. 7101-7107.

[12] *XSL Transformations (XSLT) Version 2.0*. 2007. Available at: http://www.w3.org/TR/xslt20/ (accessed 07.12.2017).

[13] T. Gardner, C. Griffin, C. Koehler and R. Hauser, *A review of OMG MOF 2.0 Query/Views/Transformations Submissions and Recommendations toward the final standard*. Object Management Group, OMG Document ad/03-08-02, 2003.

[14] S. Sendall and W. Kozaczynski, *Model Transformation – The Heart and Soul of Model-Driven Software Development*. IEEE Software. 2003. Vol. 20, No. 5. P. 42-45.

[15] K. Czarnecki and S. Helsen, *Feature-based survey of model transformation approaches*. IBM Systems Journal. 2006. Vol. 45, No. 3. P. 621-645.

[16] D. Frankel, *Model Driven Architecture: Applying MDA to Enterprise Computing*. New York: Wiley, 2003.

[17] A. Y. Yurin, *Notation for design of knowledge bases of rule-based expert systems*. Object systems. 2016. No. 12. P. 48-54. (In Russ.)

[18] I. V. Bychkov, N. O. Dorodnykh and A. Y. Yurin, *Approach to the development of software components for generation of knowledge bases based on conceptual models*. Computational Technologies. 2016. Vol. 21, No. 4. P. 16-36. (In Russ.)

[19] A. V. Aho, M. S. Lam and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, 2nd ed. Addison Wesley, 2006.

[20] V. N. Agafonov, *Specification of programs: conceptual means and their organization*. Novosibirsk: Nauka, 1987. (In Russ.)

[21] N. O. Dorodnykh, *Web-based software for automating development of knowledge bases on the basis of transformation of conceptual models*. Open Semantic Technologies for Intelligent Systems. 2017. P. 145–150.

ПРИМЕНЕНИЕ МОДЕЛЬНЫХ ТРАНСФОРМАЦИЙ ДЛЯ СОЗДАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Дородных Н.О.*, Коршунов С.А.**, Павлов Н.Ю.**, Сопп Д.Ю.**, Юрин А.Ю.*

*Институт динамики систем и теории управления имени В.М. Матросова Сибирского отделения Российской академии наук (ИДСТУ СО РАН)

**ООО "ЦентраСиб"

В работе рассмотрено применение модельных трансформаций в процессе создания интеллектуальных систем на основе предлагаемой реализации модельно-управляемого подхода. В качестве исходных данных предлагается использовать концептуальные модели, представленные в XML-подобных форматах. Особое внимание уделено первому этапу в цепочке модельных трансформаций: этапу формирования вычислительно-независимой модели. В качестве инструментария реализации модельных трансформаций предложено использовать новый предметно-ориентированных язык – Transformation Model Representation Language (TMRL) и поддерживающее его программное средство. Приведено описание основных конструкций TMRL и пример его применения.