A DOMAIN-SPECIFIC LANGUAGE FOR TRANSFORMATION MODELS

Nikita O. Dorodnykh[(1)], *Alexandr Yu. Yurin*[(1)]

[(1)] Matrosov Institute for System Dynamics and Control Theory SB RAS, Irkutsk, Russia

The efficiency of the intelligent systems engineering on the basis of conceptual models can be improved through the use of specialized languages. The paper describes a domain-specific language designed for describing transformation models namely Transformation Model Representation Language (TMRL). The descriptions of the basic language constructions, as well as comparison with analogues, are given.

Keywords: model, model transformation, domain-specific language, Transformation Model Representation Language, TMRL

ЯЗЫК ДЛЯ ОПИСАНИЯ МОДЕЛЕЙ ТРАНСФОРМАЦИЙ

Дородных Н.О. [(1)], *Юрин А.Ю.* [(1)]

[(1)] Институт динамики систем и теории управления им. В.М. Матросова СО РАН, г. Иркутск

Повышение эффективности разработки интеллектуальных систем и их компонентов на основе концептуальных моделей может быть достигнуто путем применения специализированного лингвистического обеспечения. В работе описывается пример подобного обеспечения в форме предметно-ориентированного языка для описания моделей трансформаций – Transformation Model Representation Language (TMRL). Приводится описание основных языковых конструкций, а также сравнение с аналогами.

Ключевые слова: модель, трансформация моделей, предметно-ориентированный язык, Transformation Model Representation Language, TMRL

Введение. Концептуальные информационные модели различного типа (например, DFD, IDEF0, IDEF5, UML, BPMN и др.) активно используются в процессе целенаправленного моделирования и проектирования программного обеспечения, в том числе и интеллектуального, в частности, баз знаний (БЗ). При этом задача разработки БЗ на основе концептуальных моделей предметных областей может быть сведена к задаче их трансформации.

В общем случае трансформация моделей представляет собой процесс автоматической генерации целевой модели по исходной модели в соответствии с некоторым набором правил преобразования [9]. По существу, трансформация моделей является логическим продолжением (развитием) области преобразования программ. Так, трансформация модели может привести к трансформации программы, если программа основывается на модели. Тем не менее, подходы к трансформации, разработанные в обеих областях, имеют несколько разные характеристики. В то время как системы программных трансформаций, как правило, основаны на математически-ориентированных концепциях, таких как пере-

писывание (term rewriting), атрибутивных грамматиках и функциональном программировании, системы трансформации моделей обычно применяют объектно-ориентированный подход для представления и манипулирования различными предметными моделями (абстракции системы и/или ее окружения) [4]. В специальной литературе по трансформации моделей рассматривается широкий спектр артефактов в качестве потенциальных объектов трансформации для процесса разработки программного обеспечения. К ним относятся различные UML-модели, характеристические модели (feature models), спецификации интерфейсов, схемы данных, компонентов и программный код.

Принято различать два вида трансформации моделей [4]:

- Модель-Модель (Model-to-Model, M2M);
- Модель-Текст (Model-to-Text, M2T или «pretty printing» – понятие из программной трансформации) и Текст-Модель (Text-to-Model, T2M).

При этом преобразование M2T в качестве целевых (выходных) текстовых артефактов может рассматривать исходный код (трансформация Модель-Код – Model-to-Code, M2C), документацию, спецификации и т.д.

В настоящий момент наиболее распространенными языками трансформации моделей являются:

- QVT (Query/View/Transformation) – спецификация консорциума OMG, определяющая три языка трансформации моделей [10].
- ATL (ATLAS Transformation Language) [8] – язык описания трансформаций моделей базирующейся на стандарте QVT и стандартизованном языке описания ограничений OCL (Object Constraint Language).
- VIATRA2 (VIsual Automated model TRAnsformations) – основанный на правилах и паттернах язык преобразования для управления графовыми моделями [11].
- GReAT (Graph REwriting And Transformation) – язык описания преобразований модели, базирующийся на подходе тройных трансформаций графа (правила перезаписывания графа применяются к входной модели и в результате создают выходную модель) [2].
- Henshin – язык трансформации моделей, основанный на переписывании графа и использующий правила на основе шаблона, которые могут быть структурированы во вложенные единицы преобразования с четко определенной семантикой [1].
- Epsilon – семейство языков и инструментальных средств для трансформации моделей, генерации кода, проверки достоверности моделей, миграций и рефакторинга [7].
- XSLT (eXtensible Stylesheet Language Transformations) – язык преобразования XML-документов [12]. Данный язык является спецификацией консорциума W3C.

Основным недостатком данных языков трансформации являются высокие требования к пользователю (специалисту) при разработке правил (сценария) трансформаций. В частности, пользователю необходимо знать не только синтаксис и семантику определенного языка трансформации моделей, который может быть достаточно сложен, но и языки метамоделирования (например, MOF (Meta-Object Facility), Ecore, KM3 (Kernel Meta Meta Model) и др.), используемые для описания входной и выходной моделей, а также различ-

ных сопутствующих (дополнительных к основным языкам) языковых конструкций (например, OCL). Существенным ограничением (недостатком) почти всех языков трансформации моделей является жесткая привязка к определенному программному инструментарию, в частности, к платформе Eclipse, где поддержка языков реализована в виде модулей/плагинов и, в частности, EMF (Eclipse Modeling Framework) [6]. Совокупность этих факторов затрудняет практическое использование этих языков и программных средств непрограммирующими пользователями (специалистами-предметниками, инженерами по знаниям, аналитиками и т.д.), в частности, при разработке БЗ на основе трансформации концептуальных моделей и обуславливает актуальность задачи разработки нового предметно-ориентированного языка.

Модель трансформации. Для решения задачи разработки БЗ на основе трансформации концептуальных (информационных) моделей, необходимо определить модель трансформации. Под моделью трансформации понимается некоторый сценарий (программа), описывающая процесс автоматической генерации целевой модели из исходной модели в соответствии с некоторым набором правил трансформации. При этом, каждое отдельное правило трансформации – это описание того, как одна или более конструкций на исходном языке могут быть преобразованы в одну или несколько конструкций на целевом языке [9]. В частности, как конструкции некоторого исходного языка концептуального моделирования (например, UML и др.) могут быть преобразованы в конструкции некоторого целевого языка представления знаний (ЯПЗ).

Таким образом, модель трансформации может быть представленная следующим образом:

$$M_T = \langle MM_{IN}, MM_{OUT}, T \rangle$$

где MM_{IN} – метамодель исходной (входной) концептуальной модели; MM_{OUT} – метамодель целевой (выходной) модели представления знаний (модели БЗ); T – оператор преобразования моделей. При этом использование метамоделирования является одним из основных подходов к определению абстрактного синтаксиса языков, в том числе концептуальных языков моделирования и ЯПЗ.

Подробное описание отдельных элементов модели трансформации приводиться в [15].

Язык описания моделей трансформаций. Для описания моделей трансформаций предлагается специализированный предметно-ориентированный язык – Transformation Model Representation Language (TMRL). Грамматика TMRL принадлежат к классу контекстно-свободных грамматик (КС-грамматик – LL(1)) [14]. Конструкции TMRL позволяют в декларативном виде описывать элементы модели трансформации, в частности, правила соответствия элементов метамоделей. Созданные на TMRL спецификации удовлетворяют требованиям точности, понятности и полноты [13], т.е. в спецификациях на TMRL содержится вся необходимая (в рамках предложенного метода) информация для решения поставленной задачи, все объекты модели хорошо формализованы, при этом спецификации достаточно компактны и в то же время понятны (читабельны).

Структура программы на TMRL состоит из трех основных блоков, рассмотрим ее на примере программы, описывающей преобразование диаграммы классов UML в модель онтологии (элементы языка выделены жирным шрифтом).

Блок 1. Описание элементов и отношений исходной метамодели:

```
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), … ]  }
```

Данный блок содержит описание диаграммы классов UML: «`UML-diagram-class`», включая элементы модели (раздел «Elements»). В данном примере – это элементы «`Model`» и «`Class`», при этом элемент «Class» обладает свойствами «`xmi.id`» и «`name`». Помимо описания элементов, исходная метамодель содержит описание связей между элементами метамодели (раздел «`Relationships`»), в том числе по идентификаторам, например, связь атрибута с типом данных («`DataType(xmi.id) is Attribute(type)`»).

Блок 2. Описание элементов и отношений целевой метамодели:

```
Target Meta-Model Ontology {
   Elements [
        ExtendedOntology attributes (id, name),
        Class attributes (id, name), … ]
 Relationships [
        Ontology is associated with Class, … ]  }
```

Блок содержит описание модели онтологии: «`Ontology`». Структура блока аналогична структуре блока исходной метамодели.

Блок 3. Описание правил преобразования моделей:

```
Transformation UML-diagram-class to Ontology {
 Rule Model to Ontology priority 1 [
        Ontology(name) is Model or ModelElement.name
        Ontology(id) is Model(xmi.id) ]
 Rule (Class, ModelElement.name) to Class priority 2 [
        Class(name) is Class(name) or ModelElement.name
        Class(id) is Class(xmi.id) ] …
 Rule (AssociationEnd, MultiplicityRange) to Lhs priority 7 [
        Lhs is AssociationEnd
        Lhs(operator) is "AND" [
 if (MultiplicityRange(lower) is "1") and (MultiplicityRange(upper) is "-
1") … ]… ]…}
```

Данный блок содержит описание правил преобразования элементов исходной («`UML-diagram-class`») метамодели в целевую («`Ontology`»).

В создаваемых правилах могут использоваться логические операторы («**and**», «**or**») и оператор условного выбора («**if**»), в частности, с целью задания определенного значения (константы) для целевого элемента метамодели в соответствии с выполнением условия, определенного в блоке «**if**» (например, исходя из определенного значения исходного элемента).

Дополнительно к описанию модели трансформации на TMRL возможно задать взаимодействие с другими ранее разработанными программными компонентами трансформации при помощи специальной конструкции «`Call`»:

```
Call <название программного компонента трансформации> , "<путь к концептуальным моделям>" , "<путь сохранения баз знаний>"
```

TMRL содержит 15 специальных элементов (лексем), из них: для описания блока исходной и целевой метамодели используется – 7; для блока трансформации – 7; для блока вызова программного компонента трансформации – 1. В качестве начального нетерминала (стартового символа) используется – «`Модель трансформации на TMRL`».

Сравнение с аналогами. TMRL предназначен для представления и хранения модели трансформации, вследствие чего он не имеет прямых аналогов, с которыми можно было бы осуществить полное и корректное сравнение. Однако наиболее близкими к TMRL являются языки модельных трансформаций (Model Transformation Language, MTL), например, ATL [8] или языки стандарта QVT (QVT-R, QVT-C, QVT-O) [10] и др.

Приведем краткое формальное сравнение синтаксиса правила преобразования элемента «`Class`» из диаграммы классов UML в онтологию OWL на TMRL и ATL.

Правило трансформации на TMRL:

```
Rule Class to Class priority 1 [Class(ID) is Class(name)]
```

Правило трансформации на ATL:

```
rule UMLClass2OWLClass {
  from
    c : UML!uml::Class (
      c.oclIsTypeOf(UML!uml::Class) and
        not thisModule.sequenceOfUnionClass.includes(c)
    )
  to
    oc : OWL!OWLClass (uriRef <- u),
    u : OWL!URIReference (fragmentIdentifier <- l,uri <- uri),
    l : OWL!LocalName (name <- c.name),
    uri : OWL!UniformResourceIdentifier (name <- c.name)
}
```

Заключение. Повышение эффективности описания модельных трансформаций, в частности, при разработке интеллектуальных систем, обуславливает необходимость разработки новых специализированных предметно-ориентированных языков. Примером подобных языков является Transformation Model Representation Language (TMRL).

Главным отличием TMRL от существующих языков трансформации моделей общего назначения является простота его использования, достигаемая за счет ограниченного набора конструкций. TMRL не является расширением других языков и не использует конструкции других языков, как это очень часто делают MTL, в частности, ATL использует язык ограничений OCL. Кроме того, TMRL обладает человекочитаемым синтаксисом с целью обеспечения возможности внесения необходимых дополнений (уточнений) в модель трансформации вручную. Также особенностью TMRL является способность описывать взаимодействие с ранее разработанными программными компонентами трансформации по части поддержки импорта различных форматов концептуальных моделей.

В настоящий момент поддержка TMRL обеспечивается в программном средстве Knowledge Base Development System (KBDS) [5], которое использовалось при создании БЗ экспертных систем для поддержки экспертизы промышленной безопасности [3].

Работа выполнена при финансовой поддержке РФФИ (грант № 18-37-00006).

Результаты получены при использовании сетевой инфраструктуры ЦКП «Интегрированная информационно-вычислительная сеть Иркутского научно-образовательного комплекса» (http://net.icc.ru).

ЛИТЕРАТУРА

[1] Arendt T., Biermann E., Jurack S., Krause C., Taentzer G. 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.

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

[3] Berman A.F., Nikolaichuk O.A., Yurin A.Yu., Kuznetsov K.A. Support of Decision-Making Based on a Production Approach in the Performance of an Industrial Safety Review // Chemical and Petroleum Engineering. 2015. Vol.50. Issue 1-2. P.730-738.

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

[5] Dorodnykh N.O. Web-based software for automating development of knowledge bases on the basis of transformation of conceptual models // Open Semantic Technologies for Intelligent Systems. 2017. Vol.7. P. 145-150.

[6] Eclipse Modeling Framework (EMF) [Электронный ресурс]. URL: http://www.eclipse.org/modeling/emf/ (дата обращения: 14.06.2018). 13

[7] Epsilon [Электронный ресурс]. URL: http://www.eclipse.org/epsilon/ (дата обращения: 14.06.2018).

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

[9] Kleppe A., Warmer J., Bast W. MDA Explained: The Model-Driven Architecture: Practice and Promise (1st ed.). Addison-Wesley Professional, 2003. 192 p.

[10] Query/View/Transformation (QVT) Version 1.3 // OMG Document formal/2016-06-03. 2016 [Электронный ресурс]. URL: http://www.omg.org/spec/QVT/1.3/ (дата обращения: 14.06.2018).

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

[12] XSL Transformations (XSLT) Version 2.0 [Электронный ресурс]. URL: http://www.w3.org/TR/xslt20 (дата обращения: 14.06.2018).

[13] Агафонов В.Н. Спецификация программ: понятийные средства и их организация. Новосибирск: Наука, 1987. 240 с.

[14] Ахо А.В., Лам М.С., Сети Р., Ульман Дж.Д. Компиляторы: принципы, технологии и инструментарий, 2-е изд. М.: Вильямс, 2008. 1184 с.

[15] Бычков И.В., Дородных Н.О., Юрин А.Ю. Подход к разработке программных компонентов для формирования баз знаний на основе концептуальных моделей // Вычислительные технологии. 2016. Т. 21, № 4. С. 16-36.