УДК 004.434

А.О. Сухов

МЕТОДЫ ТРАНСФОРМАЦИИ ВИЗУАЛЬНЫХ МОДЕЛЕЙ

Аннотация: В статье рассматриваются методы трансформации моделей, созданных с помощью визуальных языков моделирования. Подробный анализ позволил выявить существующие в этой области проблемы и сформулировать подходы к их устранению.

Abstract: In paper the methods of model transformations created by means of visual modeling languages is considered. The detailed analysis has allowed to reveal existing problems in this area and to formulate approaches to their elimination.

Ключевые слова: графовые трансформации, графовые грамматики, предметно-ориентированные языки.

Keywords: graph transformations, graph grammars, domain-specific languages.

Введение

В любую развитую CASE-систему встроен целый набор языков моделирования различных уровней: языки описания моделей предметных областей, схем баз данных, бизнес-логики и др. При этом широко используются визуальные языки, т.к. графические нотации позволяют строить наглядные, легко воспринимаемые различными категориями пользователей модели.

В настоящее время не существует единого универсального визуального языка создания программного обеспечения [4]. Сейчас активно используются на практике такие языки визуального моделирования, как UML и ERD – для моделирования предметных областей; IDEF, DFD, EPC, BPEL и BPML – для моделирования бизнес-процессов на разных уровнях и т.п. Для многих областей деятельности ставится задача создания информационных систем, которые допускали бы участие экспертов (не программистов) как в разработке системы, так и в настройке ее на меняющиеся условия эксплуатации, потребности бизнес-процессов и конкретных пользователей. При этом каждому специалисту необходимо обеспечить возможность работы в привычных для него терминах знакомой ему предметной области [3]. Предметно-ориентированные языки и языковые инструментарии, предназначенные для их создания, позволяют решить эту задачу, однако при этом остро встает проблема трансформации построенных моделей, т.к. большинство наиболее развитых на сегодняшний день языковых инструментариев не позволяет производить трансформацию моделей из одной нотации в другую [1].

Для задания синтаксиса визуальных языков используются различные формализмы: автоматная модель, алгоритмические сети, но наиболее распространенным являются графовые грамматики [2]. *Графовые грамматики* – обобщение грамматик Хомского на графы. Чтобы задать грамматику, требуется задать множества терминальных и нетерминальных символов, набор правил вывода, а также выделить в множестве нетерминалов стартовый символ. Причем в качестве правой части продукционного правила может выступать не только помеченный граф, но и код на каком-либо языке программирования, а также фрагмент визуальной модели, описанной в другой нотации. Именно поэтому графовые грамматики могут использоваться как для порождения синтаксически правильных моделей, так и для рефакторинга существующих моделей, генерации кода и трансформации моделей с одного языка моделирования в другой.

Трансформации моделей

Существуют различные подходы к трансформации моделей, некоторые из них имеют формальную основу, так технология GReAT использует для трансформации тройные графовые грамматики, а некоторые – применяют технологии из других областей программной инженерии, например, метод программирования на примере.

Язык трансформации ATL (ATLAS Transformation Language) является частью архитектуры управления моделью ATLAS [8]. ATL – язык, позволяющий описывать

трансформации любой исходной модели в указанную целевую модель. Преобразование производится на уровне метамоделей – моделей языка моделирования. Интеграция в рамках одного языка декларативного и императивного подхода позволяет производить преобразования для сложных предметных областей, при этом декларативный подход скрывает сложность алгоритмов трансформации. В основе ATL лежит стандартизованный язык описания ограничений OCL, что позволяет использовать в процессе описания трансформаций все преимущества данного языка: стандартные типы, фильтры, помощники и др.

К недостаткам данного подхода следует отнести высокие требования, предъявляемые к разработчику преобразования. Поскольку ATL в большинстве случаев использует лишь текстовое определение трансформации, то помимо умения строить исходную и целевую метамодель разработчик должен знать язык описания преобразований. Кроме того, использование императивных конструкций элиминирует преимущества декларативного подхода. Отсутствие навигации по целевой модели затрудняет процесс определения правил преобразования. Еще одним недостатком является однонаправленность правил трансформации, поэтому, чтобы определить взаимное отображение исходной и целевой моделей, необходимо дополнительно построить обратное отображение вручную.

GReAT (Graph REwriting And Transformation) – язык описания преобразований модели, базирующийся на подходе тройных трансформаций графа [6]. Трансформация, описанная на этом языке, представляет собой набор упорядоченных правил перезаписи графа, которые применяются к входной модели и в результате создают выходную модель. GReAT является языком описания трансформаций моделей, который использует диаграммы классов UML и язык OCL для представления преобразований. Данный подход позволяет производить трансформацию сразу для нескольких исходных метамоделей, что является значительным преимуществом по сравнению с другими средствами трансформации. Также, GReAT предоставляет возможность построения вспомогательной метамодели, которая позволяет определить соответствие между исходными и целевыми метамоделями.

Однако это средство трансформации моделей обладает также и недостатками: у пользователя отсутствует возможность выбора языка спецификации метамоделей, изменения его описания, поэтому приходится довольствоваться возможностями, предоставляемыми UML и OCL; трансформации в GReAT однонаправлены, определение обратного преобразования должно быть получено вручную; отсутствует единая математическая основа описания графов, так в источниках [5] и [6] даны различные формальные определения графа, вершины, дуги.

AGG (Attributed Graph Grammar) – средство описания графовых грамматик для типизированных атрибутных графов, которое поддерживает трансформацию графов. Данный подход был интегрирован в инструментальное средство на платформе Java, при этом теоретические понятия реализованы настолько непосредственно насколько возможно, учитывая эффективность работы инструментария.

AGG использует для описания исходной и целевой моделей ориентированные типизированные атрибутные графы, что позволяет применять данный инструментарий практически в любой предметной области. Благодаря использованию в качестве формальной основы алгебраического подхода к трансформации графов существует возможность парсинга графа, проверки графовой модели на противоречивость.

Расширение возможностей графовых грамматик средствами языка Java позволяет приблизить формальную модель к предметной области и реализовать сложные функции поведения системы. Преобразования модели задается правилами перезаписи графа, которые применяются недетерминировано до тех пор, пока ни одно из них не может быть больше выполнено. Если необходимо явно указать порядок применения правил, то пользователь может сгруппировать их по уровням. Еще одним преимуществом данного подхода является

то, что он был реализован в инструментальном средстве [10], которое содержит визуальные редакторы, интерпретатор, а также текстовый редактор описания Java-выражений.

VIATRA – основанный на правилах и паттернах, язык преобразования для управления графовыми моделями, который комбинирует в единую парадигму спецификации два подхода: математический формализм, основанный на правилах трансформации графа, для описания моделей и абстрактные конечные автоматы, предназначенные для описания потока управления [9]. Благодаря использованию конструкций конечных автоматов разработчикам удалось значительно повысить семантику стандартных языков описания паттернов и преобразования графов. Кроме того, мощные конструкции языка позволяют производить многоуровневое метамоделирование предметных областей.

К преимуществам VIATRA следует отнести, во-первых, возможность использования универсальных метапреобразований, которые позволяют производить многоуровневое метамоделирование и повторное использование уже созданных алгоритмов трансформаций. Во-вторых, основанный на шаблонах метод генерации кода для преобразований вида «модель−код» позволяет генерировать файлы, содержащие как статичные части, так и изменяемые данные. В-третьих, возможность использования как текстовой, так и графической нотации языка метамоделирования предоставляет пользователю возможность применять для конкретной задачи более удобный для него способ моделирования. Кроме того, команды языка в объединении с паттернами графа и правилами трансформации формируют удобный математический язык работы с моделями.

Существенным недостатком VIATRA является отсутствие возможности задания двунаправленных трансформаций. Хотя разработчики подхода критикуют стандарт MOF за отсутствие возможности многоуровневого метамоделирования [7], они все же остаются в рамках той же парадигмы при использовании визуального языка описания метамоделей, который используется чаще, чем текстовое представление, особенно пользователями непрофессиональными программистами. Интеграция реализации данного подхода в систему Eclipse накладывает на него ряд ограничений, присутствующих у этой платформы.

QVT (Query/View/Transformation) – предложенный OMG подход к трансформации модели, предоставляющий в распоряжение пользователя как декларативные, так и императивные языки. Преобразование определяется на уровне метамоделей, описанных с помощью MOF. Достоинством данного подхода является существование стандарта его описания, а также использование в процессе построения преобразований модели стандартных языков: OCL и MOF. Еще одно преимущество QVT – большой набор языков описания трансформаций, которые позволяют использовать как стандартные средства, так и их расширения.

Однако эти преимущества имеют и обратную сторону. Использование MOF в качестве языка метамоделирования не позволяет пользователю выбрать удобный для него метаязык, а также изменить описание метаязыка, интегрированного в QVT. Кроме того, при использовании данного подхода затруднено задание трансформаций вида «модель-код», т.к. каждая метамодель должна быть описана с помощью стандарта MOF.

Синтаксис языка описания трансформаций достаточно сложен по сравнению с другими подходами. Хотя разработчики стандарта заявляют о поддержке как визуального, так и текстового представления языков описания трансформаций, в большинстве работ авторы предпочитают использовать лишь текстовый конкретный синтаксис, поскольку визуальное представление оказывается еще более сложным для понимания.

Следует также отметить, что декларативный язык описания трансформации никогда не имел полной реализации, поэтому в настоящее время нет ни одного инструментального средства, поддерживающего весь стандарт QVT.

Подход трансформации модели на примере (MTBE) основан на подходах программирования на примере (PBE) и формировании запросов на примере (QBE). Основная

цель MTBE – автоматическая генерация правил трансформации на основе начального набора обучающих примеров.

Основное преимущество MTBE заключается в том, что для получения правил преобразования пользователю не требуется знание какого-либо языка трансформации моделей, будь то графовые трансформации, ATL и т.п., поскольку в качестве такого языка выступают концепты исходной и целевой моделей, т.е. генерировать правила преобразования может конечный пользователь, владеющий информацией лишь о нотации языка моделирования.

Текущие реализации подхода MTBE позволяет выполнять лишь полные эквивалентные отображения, не учитывая сложные преобразования атрибутов. Еще одним существенным недостатком большинства реализаций MTBE является то, что в них не затрагивается проблема трансформации ограничений, налагаемых на концепты и отношения модели.

Заключение

Подводя итог, можно говорить о том, что с каждым годом интерес научного сообщества к проблеме трансформации моделей растет, однако существующие методы преобразования моделей обладают значительными недостатками. Одним из подходов к решению этих проблем является разработка языка трансформаций, который оперирует конструкциями понятными пользователям – непрофессиональным программистам. Такой язык должен позволять изменять свое описание во время работы системы, без внесения каких-либо изменений в исходный код. При этом необходимо предоставить пользователю возможность выбора наиболее подходящей формы определения трансформации: будет ли она описана с помощью визуальной нотации либо в текстовом виде. Кроме того, этот подход должен позволять выполнять генерацию обратного преобразования автоматически с возможностью внесения необходимых изменений «вручную».

Созданный язык планируется интегрировать в инструментальное средство описания визуальных предметно-ориентированных языков моделирования MetaLanguage [1].

Библиографический список

1. Лядова, Л.Н. Визуальные языки и языковые инструментарии: методы и средства реализации / Л.Н. Лядова, А.О. Сухов // Труды международных научно-технических конференций «Интеллектуальные системы» (AIS'10) и «Интеллектуальные САПР» (CAD-2010), 2010. С. 374-382.
2. Сухов, А.О. Анализ формализмов описания визуальных языков моделирования / А.О. Сухов // Современные проблемы науки и образования. – 2012. – № 2; URL: www.science-education.ru/102-5655 (дата обращения: 02.04.2012).
3. Сухов, А.О. Предметно-ориентированный язык в адаптируемых информационных системах / А.О. Сухов // Материалы конференции «Технологии Microsoft в теории и практике программирования». – Новосибирск: Академгородок, 2008. С. 25-26.
4. Сухов, А.О. Решение проблем традиционного подхода к разработке информационных систем на основе использования предметно-ориентированных языков / А.О. Сухов // Материалы конференции «Технологии Microsoft в теории и практике программирования». – Томск, 2009. С. 180-181.
5. Agrawal, A. Graph Transformations on Domain-Specific Models / A. Agrawal, G. Karsai, F. Shi // International Journal on Software and Systems Modeling. – Nashville, 2003. – P. 1-43.
6. Balasubramanian, D. The Graph Rewriting and Transformation Language: GReAT / D. Balasubramanian [et al.] // Electronic Communications of the EASST. – 2006. – Vol. 1. – P. 1-8.
7. Balogh, A. Advanced model transformation language constructs in the VIATRA2 framework / A. Balogh, D. Varro // ACM Symposium on Applied Computing – SAC. – 2004. P. 1280-1287.
8. Bezivin, J. An Introduction to the ATLAS Model Management Architecture / J. Bezivin, F. Jouault, D. Touzet // Research Report № 05.01. – LINA, 2005. – 24 p.
9. Borger, E. Abstract State Machines. A method for High-Level System Design and Analysis / E. Borger, R. Stark. – Springer-Verlag: New York, 2003. – 448 p.
10. Taentzer, G. AGG: A Graph Transformation Environment for Modeling and Validation of Software / G. Taentzer // Lecture Notes in Computer Science. –2004. – Vol. 3062/2004. – P. 446-453.