УДК 004.434

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА СОЗДАНИЯ ВИЗУАЛЬНЫХ ПРЕДМЕТНО-ОРИЕНТИРОВАННЫХ ЯЗЫКОВ МОДЕЛИРОВАНИЯ

Сухов А.О.

ФГБОУ ВПО «Пермский государственный национальный исследовательский университет», Пермь, e-mail: sukhov_psu@mail.ru

Неспособность универсальных языков моделирования повысить уровень абстракции инструментального программного обеспечения и сделать возможным участие специалистов предметной области в процессе разработки информационных систем приводит к широкому распространению предметно-ориентированных языков, которые более выразительны, просты в применении и понятны различным категориям пользователей. В статье рассмотрены подходы к разработке инструментального средства MetaLanguage, предназначенного для создания визуальных предметно-ориентированных языков моделирования. Приведено описание метаязыка системы MetaLanguage, подробно рассмотрены подходы к заданию и выполнению горизонтальных трансформаций моделей. Данный инструментарий позволяет проводить многоуровневое метамоделирование предметных областей, причем изменение описания создаваемых языком можно производить даже на этапе построения моделей. Используя компонент трансформации, пользователь может выполнять преобразования созданных им моделей в текст или визуальную модель, описанную в другой нотации.

Ключевые слова: визуальные предметно-ориентированные языки моделирования, языковой инструментарий, метамоделирование, трансформация моделей

THE LANGUAGE WORKBENCH FOR VISUAL DOMAIN-SPECIFIC MODELING LANGUAGES CREATION

Sukhov A.O.

Perm State National Research University, Perm, e-mail: sukhov_psu@mail.ru

The inability of the general-purpose modeling languages to increase abstraction level of the instrumental software and to make possible an involvement of domain experts in information systems development process leads to a widespread of the domain-specific languages that are more expressive, are simple on applying and are easy to understand for different categories of users. The article describes the approaches to development of the language workbench MetaLanguage that is designed for visual domain-specific modeling languages creation. The description of the meta-language of MetaLanguage system is given, approaches to the definition and applying of horizontal models transformations are considered. This workbench allows to fulfill multilevel meta-modeling of the domains. Modifications in the description of created languages can be done even at a stage of models creation. Using a transformation component, the user can perform the transformations of the created models to a text or a visual model described in other notation.

Keywords: visual domain-specific modeling languages, language workbench, meta-modeling, model transformation

С течением времени информационные технологии используются все в более широких областях деятельности. При этом важной задачей становится разработка информационных систем (ИС), которые в большей степени ориентированы на конкретную предметную область и обладают возможностью последующей адаптируемости к постоянно меняющимся условиям внешней среды и потребностям пользователей. При создании таких ИС важной задачей становится извлечение данных и знаний о предметной области, их формализация и дальнейшее использование на различных этапах жизненного цикла системы. Активное участие в процессе разработки предметно-ориентированных ИС должны принимать специалисты в соответствующей области, поскольку именно они знают все ее особенности и недостатки уже функционирующих в ней систем.

Традиционные средства разработки систем не способны решить эти задачи, поэтому все более востребованными и актуальными становятся модельно-ориентированные подходы к созданию систем, в которых модели являются центральным артефактом на всех этапах жизненного цикла ИС: MBE (Model-Based Engineering), MDE (Model-Driven Engineering), MDD (Model-Driven Development) и др.

Под *моделью* понимают абстрактное описание системы (объекта, процесса), содержащее существенные с точки зрения цели моделирования характеристики системы, особенности ее функционирования. Модели создаются с помощью определенных языков моделирования. *Метамодель* – язык, используемый для разработки моделей. Для создания метамоделей также используются языки моделирования – *мета-метамодели* (*метаязыки*).

Для разработки отдельных компонентов ИС могут применяться различные DSL: языки описания схем баз данных, бизнес-процессов (БП), шаблонов документов и др., но для создания единой модели системы недостаточно проанализировать все

ее части в отдельности, необходимо рассмотреть всю систему в целом. Для этого требуется наличие средств трансформации, позволяющих построить единую модель системы на основе моделей, описывающих ИС с различных точек зрения.

Поскольку универсальные языки моделирования, такие как UML, не способны сделать возможным участие специалистов предметной области в процессе разработки ИС, то для реализации модельно-ориентированных подходов используются *предметно-ориентированные языки* моделирования (Domain Specific Languages, DSL), созданные для работы в конкретных предметных областях. Предметно-ориентированные языки более выразительны, просты в применении и понятны различным категориям пользователей, поскольку они оперируют привычной для них терминологией. Для поддержки процесса разработки и сопровождения DSL используется специальный вид программного обеспечения, получивший название *языковой инструментарий, или DSM-платформа*. Существуют различные средства создания DSL с возможностью задания собственной графической нотации: MetaEdit + [11], MS DSL Tools [7], Eclipse GMF [9], QReal [1] и др.

В результате анализа существующих DSM-платформ были обнаружены следующие основные ограничения, присущие большинству систем [5]:

1. Отсутствие возможности многоуровневого метамоделирования и изменения описания метаязыка. Наличие такой возможности делают систему более гибкой, поскольку она позволяет изменять описание метаязыка, добавлять в него новые конструкции, тем самым приближая его к предметной области.

2. Внесение изменений в описание DSL можно произвести только через повторную генерацию кода.

3. Наличие дополнительной функциональности, невостребованной конечным пользователем, что делает работу с системой сложной для специалистов в предметной области.

4. Отсутствие возможности трансформации моделей, которая позволяет не только создать единую модель системы, но и сгенерировать код по указанному пользователем шаблону, либо произвести преобразование модели в другую нотацию, например, в один из широко распространенных языков моделирования.

Устранение отмеченных недостатков DSM-платформ – цель разработки языкового инструментария MetaLanguage, предназначенного для создания визуальных динамически настраиваемых предметно-ориентированных языков моделирования, позволяющего производить построение моделей для различных предметных областей и выполнять преобразование их описания с одного языка моделирования в другой.

Метаязык системы MetaLanguage

Одним из основных элементов любого языкового инструментария является *метаязык* (мета-метамодель) – язык для описания других языков (метамоделей). Именно благодаря наличию метаязыка DSM-платформа позволяет создавать предметно-ориентированные языки для различных предметных областей, оперирующие привычными для пользователя понятиями. Отличие метаязыка системы MetaLanguage от подхода MOF (Meta Object Facility), используемого в большинстве DSM-платформ, заключается в том, что благодаря интерпретации описания моделей различных уровней абстракции, а не генерации на их основе исходного кода, появляется возможность изменения конструкций создаваемых DSL в динамике, во время построения моделей. Кроме того, процесс создания метамодели становится многоуровневым. Так, определив метамодель и выбрав ее в качестве метаязыка, разработчик может использовать эту мета-метамодель для построения других метамоделей, и этот процесс может быть бесконечным.

Базовыми элементами метаязыка системы MetaLanguage являются сущность, отношение, ограничение. «Сущность» описывает определенную конструкцию языка моделирования, т.е. объект предметной области, важный с точки зрения решаемой задачи. Конструкции визуальных языков в редких случаях существуют независимо, чаще всего они каким-либо образом взаимосвязаны друг с другом, поэтому при создании метамодели важно не только определить основные понятия создаваемого языка, но и правильно задать связи между ними. «Отношение» используется для обозначения физической или концептуальной связи между сущностями. Метамодель позволяет создавать отношения трех типов: ассоциация, агрегация, наследование.

На практике достаточно часто встречаются случаи, когда на сущности и связи между ними необходимо наложить какие-либо *ограничения*. Часть ограничений задается при описании структуры метамодели, а часть через непосредственное их определение на некотором языке. Все ограничения, налагаемые на метамодель, могут быть разделены на две группы: ограничения, налагаемые на сущности, и ограничения, налагаемые на отношения.

Для проверки корректности построения моделей необходимо выбрать математический аппарат и формально описать алгоритмы работы с элементами метамоделей и моделей. Анализ различных средств формального описания синтаксиса визуальных языков показал, что наиболее подходящим формализмом, учитывающим назначение системы MetaLanguage, являются ориентированные псевдо-метаграфы [3]. Этот вид графов позволяет сократить число дуг в графе и сделать модель более структурированной, логичной. В работе [6] описана многоуровневая математическая модель предметной области и приведен пример ее построения.

Трансформация моделей в системе MetaLanguage

Трансформация моделей – центральная часть модельно-ориентированного подхода к разработке, поскольку существование в одной системе моделей, выполненных с разных точек зрения, с разной степенью детализации и использующих для своего описания разные языки моделирования, требует наличия средств преобразования моделей как между различными уровнями абстракции, так и внутри одного уровня: при переходе от одного языка моделирования к другому для построения единой модели системы.

Остается также нерешенной проблема экспорта моделей из одной ИС в другую, например, бизнес-процессы, описанные в одной системе, не могут быть исполнены в другой из-за того, что эти системы используют различные нотации описания бизнес-процессов.

Использование предметно-ориентированных языков и инструментальных средств их создания затрагивает также проблему трансформации, поскольку появляется потребность экспорта созданных пользователем моделей во внешние системы, которые, как правило, используют один из стандартных языков моделирования, отличающийся от разработанного DSL.

Для задания преобразований моделей применяются различные подходы и инструментальные средства. Анализ методов описания трансформаций, приведенный в работе [4], выявил недостатки этих подходов, которые ограничивают их применимость для описания трансформаций в системе MetaLanguage. Наиболее подходящим и перспективным, с точки зрения авторов, является алгебраический подход с одинарным выталкиванием [8].

В статье [10] приведено подробное рассмотрение подходов к заданию вертикальных трансформаций в системе MetaLanguage, поэтому большее внимание здесь уделим описанию горизонтальных преобразований моделей.

Горизонтальные трансформации в MetaLanguage описываются на уровне метамоделей, что позволяет задать преобразования, которые могут быть применены ко всем созданным на их основе моделям. Для создания трансформации необходимо выбрать исходную и целевую метамодели и задать продукционные правила, описывающие преобразование.

Для задания правила следует выделить в исходной метамодели объекты (сущности и отношения), задать ограничения на вхождение паттерна и определить правую часть правила. В зависимости от вида трансформации правой частью будет либо текстовый шаблон для генерации кода, либо фрагмент целевой метамодели. Правила трансформации выполняются в соответствии с их порядком. Сначала будут найдены все вхождения паттерна первого правила, для каждого из них система выполнит правую часть правила, затем система перейдет ко второму правилу и приступит к его выполнению и т.д.

После выборки очередного правила необходимо найти в исходной модели все вхождения экземпляров левой части правила и для каждого из них выполнить правую часть правила. Предположим, что система выбрала очередное продукционное правило трансформации и пытается его выполнить. Для реализации применения правила необходимо описать два алгоритма: алгоритм поиска паттерна в исходном графе-хосте и алгоритм выполнения правой части правила.

Отличием рассматриваемого подхода от классической постановки задачи сопоставления графов, которую можно решить с помощью алгоритмов Ульмана, Шмидта и Дрюфелла, Венто и Фоггиа или Nauty-алгоритма [2], является то, что в данном случае необходимо найти паттерн графа метамодели в графе модели, т.е. требуется провести сопоставление графов, принадлежащих различным уровням абстракции, при этом необходимо учитывать типизацию вершин и типизацию дуг, поскольку между двумя вершинами графа метамодели может быть проведено несколько дуг различного типа.

Использованный при реализации системы MetaLanguage алгоритм поиска паттерна в графе модели является разновидностью алгоритма перебора с возвратом. Поскольку количество дуг в графе модели, как правило, меньше количества вершин, каждая дуга однозначно идентифицирует инцидентные ей вершины, а степень вершины может

быть более двух, что не позволяет с первого раза выбрать следующую вершину графа модели, входящую в паттерн, было принято решение производить поиск подграфа в графе модели на основе поиска дуг определенного типа.

Алгоритм состоит из трех этапов. На первом этапе работы алгоритма будут найдены все экземпляры некоторого произвольного отношения из графа-паттерна, т.е. осуществляется поиск начальной дуги, с которой начнется выполнение второй части алгоритма. На втором этапе требуется найти одно из возможных вхождений экземпляров всех отношений графа-паттерна в граф исходный модели. Третий этап добавляет в искомый граф необходимые вершины и производит выполнение правой части правила. Алгоритм выполнения правой части правила будет зависеть от вида трансформации: имеет ли трансформация вид «модель-модель» или «модель-текст».

Трансформация вида «модель-текст» позволяет пользователю инструментария по заданным им шаблонам сгенерировать на основе построенных моделей исходный код на каком-либо целевом языке программирования, а также любое другое текстовое представление модели, например, ее описание в виде XML. Правая часть продукционного правила в этом случае будет содержать некоторую часть шаблона, состоящую как из статических элементов, которые не зависят от найденного паттерна, так и из динамических частей, т.е. элементов, которые меняются в зависимости от найденного фрагмента модели. Для выполнения трансформации необходимо найти в исходном графе все вхождения паттерна и произвести вставку соответствующего фрагмента текста с заменой динамической части на соответствующие имена сущностей, отношений, значения их атрибутов и др.

Трансформация вида «модель-модель» позволяет произвести преобразование модели из одной нотации в другую или выполнить какие-либо операции над моделью (создание новых элементов, редукция и др.). Такая трансформация позволит не только экспортировать модель во внешние системы, но и предоставит возможность преобразования предметно-ориентированного языка, созданного пользователем в один из широко распространенных языков моделирования, например, UML, ERD, IDEF0 и др.

Левая часть продукционного правила трансформаций такого вида представляет собой паттерн, являющийся некоторым фрагментом исходной метамодели, а правая часть правила – это некоторый фрагмент целевой метамодели. При задании продукционного правила также необходимо описать правила преобразования атрибутов сущностей и отношений.

Все преобразования вида «модель-модель» представляются в виде комбинации элементарных преобразований «сущность-сущность», «отношение-отношение», «сущность-отношение» и «отношение-сущность», для которых описаны алгоритмы трансформации.

Архитектура системы MetaLanguage

Архитектура создаваемого языкового инструментария включает в себя следующие основные компоненты: среда разработки, метаязык, графический редактор моделей, браузер объектов, валидатор, трансформатор.

Среда разработки – это реализация общих сервисных функций создаваемой системы. Она обеспечивает интеграцию всех компонентов в единое целое. *Метаязык* – визуальный язык для описания предметно-ориентированных языков. *Графический редактор* представляет собой рабочую область для построения как моделей, так и метамоделей предметной области, при этом используется единый пользовательский интерфейс. Для создания нового элемента модели достаточно переместить его из списка допустимых конструкций языка и указать необходимые свойства элемента (имя, атрибуты и др.). Графический редактор предоставляет пользователю возможность располагать на рабочем листе различные фигуры, применять к ним наборы операций (наполнение текстом, увеличение/уменьшение, перемещение, соединение друг с другом линиями и т.д.), задавать для них различные графические свойства (выбор цвета, характеристик шрифтов).

Браузер объектов – компонент, предназначенный для выполнения различных операций над моделями и метамоделями (просмотр, редактирование, проверка ограничений, трансформация и др.). Используя браузер, пользователь может получить доступ ко всем элементам метамодели: моделям, созданным с помощью текущей метамодели, сущностям, отношениям, а также к их атрибутам.

Валидатор проверяет соответствие модели ограничениям, заданным пользователем. *Трансформатор* – это компонент, позволяющий выполнять вертикальные и горизонтальные трансформации моделей в текст на целевом языке программирования, либо в визуальную модель в другой нотации.

Заключение

Итогом исследования стала программная реализация системы MetaLanguage, которая позволяет создавать визуальные предметно-ориентированные языки моделирования. При этом система поддерживает возможность многоуровневого метамоделирования, изменения описания мета-метамоделей и метамоделей без повторной генерации кода. Наличие средств трансформации делает возможным выполнять преобразование созданных пользователем моделей в текст или другую графическую модель. Система имеет достаточно удобный и простой пользовательский интерфейс, что позволяет работать с ней не только профессиональным разработчикам, но и специалистам в предметной области.

Работа подготовлена при финансовой поддержке РФФИ (проект № 12-07-00763-а).

Список литературы

1. Терехов А.Н., Брыксин Т.А., Литвинов Ю.В. Архитектура среды визуального моделирования QReal // Системное программирование. – 2009. – Вып. 4. – С. 171–196.

2. Серый А.П. Алгоритмы сопоставления графов для решения задач трансформации моделей на основе графовых грамматик / Математика программных систем: межвуз. сб. науч. ст. – Пермь: Изд-во Перм. гос. нац. исслед. ун-та, 2012. – Вып. 9. – С. 60–73.

3. Сухов А.О. Анализ формализмов описания визуальных языков моделирования / Современные проблемы науки и образования. – 2012. – № 2. – С. 1–9. [Электронный ресурс]. – URL: www.science-education.ru/102-5655 (дата обращения: 01.03.2013).

4. Сухов А.О. Методы трансформации визуальных моделей / Технологии разработки информационных систем ТРИС-2012: материалы III международной научно-технической конференции. – Таганрог: Изд-во Технол. ин-та ЮФУ, 2012. – Т. 1. – С. 120–124.

5. Сухов А.О. Сравнение систем разработки визуальных предметно-ориентированных языков / Математика программных систем: межвуз. сб. науч. ст. – Пермь: Изд-во Перм. гос. нац. исслед. ун-та, 2012. – Вып. 9. – С. 84–111.

6. Сухов А.О. Формальное описание метаязыка системы MetaLanguage / Современные проблемы математики и её прикладные аспекты: Труды всероссийской научно-практической конференции. – Пермь: Изд-во Перм. гос. ун-та, 2010. – С. 154–159.

7. Domain-Specific Development with Visual Studio DSL Tools / S. Cook, G. Jones, S. Kent [et al.]. – Reading : Addison-Wesley, 2007. – 560 p.

8. Fundamentals of algebraic graph transformation / H.Ehrig, K.Ehrig, U. Prange [et al.]. – New York: Springer-Verlag, 2006. – 388 p.

9. Gronback R.C. Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit. – Reading : Addison-Wesley, 2009. – 706 p.

10. Sukhov A.O., Lyadova L.N. MetaLanguage: a Tool for Creating Visual Domain-Specific Modeling Languages / Proc. of the 6th Spring/Summer Young Researchers' Colloq. on Software Engineering. – M.: Изд-во Института системного программирования РАН, 2012. – P. 42–53.

11. Tolvanen J.-P., Rossi M. MetaEdit + : defining and using domain-specific modeling languages and code generators / Proc. of the conference on Object-oriented programming, systems, languages, and applications. – New York: ACM Press, 2003. – P. 92–93.

References

1. Terekhov A.N., Bryksin T.A., Litvinov YU.V. Arkhitektura sredy vizualnogo modelirovaniya QReal. System Programming, 2009, no. 4, pp. 171–196.

2. Seryy A.P. Algoritmy sopostavleniya grafov dlya resheniya zadach transformatsii modeley na osnove grafovykh grammatik. Mathematics of Program System, 2012, no. 9, pp. 60–73.

3. Sukhov A.O. Analiz formalizmov opisaniya vizualnykh yazykov modelirovaniya. Modern Problems of Education and Science, 2012, no. 2, pp. 1–9, available at: www.science-education.ru/102-5655 (accessed 1 March 2013).

4. Sukhov A.O. Metody transformatsii vizualnykh modeley. Proc. of the 3th International Conference «Information Systems Development Technologies». Gelendzhik, 2012, pp. 120–124.

5. Sukhov A.O. Sravneniye sistem razrabotki vizualnykh predmetno-oriyentirovannykh yazykov. Mathematics of Program System, 2012, no. 9, pp. 84–111.

6. Sukhov A.O. Formalnoye opisaniye metayazyka sistemy MetaLanguage. Proc. of the All-Russian Scientific Conference "Modern problems of mathematics and its applications". Perm, 2010, pp. 154–159.

7. Cook S., Jones G., Kent S. Domain-Specific Development with Visual Studio DSL Tools. Reading: Addison-Wesley, 2007, 560 p.

8. Ehrig H., Ehrig K., Prange U. Fundamentals of algebraic graph transformation. New York: Springer-Verlag, 2006, 388 p.

9. Gronback R.C. Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit. Reading: Addison-Wesley, 2009, 706 p.

10. Sukhov A.O., Lyadova L.N. MetaLanguage: a Tool for Creating Visual Domain-Specific Modeling Languages. Proc. of the 6th Spring/Summer Young Researchers' Colloquium on Software Engineering. Perm, 2012, pp. 42–53.

11. Tolvanen J.-P., Rossi M. MetaEdit + : defining and using domain-specific modeling languages and code generators. Proc. of the Conference on Object-oriented programming, systems, languages, and applications. New York: ACM Press, 2003, pp. 92–93.

Рецензенты:

Пенский О.Г., д.т.н., доцент, профессор кафедры процессов управления и информационной безопасности Пермского государственного национального исследовательского университета, г. Пермь;

Тюрин С.Ф., д.т.н., профессор кафедры автоматики и телемеханики Пермского национального исследовательского политехнического университета, г. Пермь.

Работа поступила в редакцию 07.03.2013.