УДК004.08

А.Ю. Атисков

ПОДХОДЫ К ТРАНСФОРМАЦИИ МОДЕЛЕЙ ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

Атисков А.Ю. **Подходы к трансформации моделей проектирования информационных систем.**

Аннотация. Отсутствие удобных средств переиспользования моделей проектирования информационных систем обусловлено тем, что в то время как существует несколько хорошо представленных стандартов на моделирование платформенных моделей, еще нет сформировавшегося взгляда на определение трансформаций между такими моделями. В данной статье рассмотрены подходы к автоматизированному созданию новых моделей на основе уже имеющейся модели системы. Особо выделена таксономия определений для классификации подходов.

Ключевые слова: трансформация, правила, синтаксис, семантика, гибридный подход.

Atiskov A.J. **Approaches towards transformation of informational system's models.**

Abstract. Lack of convenient methods for re-usage of informational system's models is conditioned by the absence of standard view on transformation definition between platform models. Besides there is no general opinion concerning transformations between such models. This paper explores approaches towards semi-automatic creation of new models on the basis of existing model of system. Taxonomy of definitions of approaches for classification is shown.

Keywords: transformation, rules, syntax, semantics, hybrid approach.

1. Введение. Современные технологии переиспользования результатов проектирования программного обеспечения направлены в основном на поддержку объектно-ориентированных методов моделирования, на кодогенерацию и современные стандарты. Программные комплексы поддержки этих технологий неудобны в использовании без постоянной помощи со стороны производителя и обладают избыточной сложностью в настройке или адаптации к изменениям, вносимым в стандарты проектирования. Отсутствие общей технологии описания элементов диаграмм проектирования на самом верхнем уровне (например, OWL-описания) приводит к тому, что зачастую невозможно адаптировать существующую информационную систему к современным стандартам.

Далее будет представлена таксономия для классификации некоторых существующих и возможных подходов к трансформации моделей. Таксономия описана с поэлементной моделью, которая делает выбор различных техник для трансформации моделей явным [25].

В то время как существующие OMG стандарты, такие как MOF [28] и UML [35], предлагают хорошо описанную процедуру для определения PIM и PSM, не существует описания для перехода из PIM в

PSM [19]. В 2002 году, чтобы это исправить, OMG инициировала процесс стандартизации, представив заявку на QVT (Запрос/Вид/Трансформация) [31]. Этот процесс, в конце концов, превратился в стандарт OMG для определения трансформации моделей, который представляет интерес не только для PIM–PSM трансформаций, но и для определения взглядов на модели и синхронизацию между моделями. Следуя потребностям OMG, было представлено большое количество подходов к трансформации моделей:

1. Научные разработки: GReAT [1], UMLX, ATOM [7], VIATRA [36], BOTL [10, 24], ATL [9];
2. Подходы, представленные как дополнения к OMG QVT RFP: QVT-Partners [32], CBOP–DSTC–IBM [11], AST+ [6], IOPT [20], Compuware–Sun [14];
3. Реализации MDA инструментов с открытым кодом: Jamda [21], AndroMDA [4], JET, FUUT-je и GMT [18];
4. Реализации коммерческих MDA инструментов: OptimalJ [30], ArcStyler [5], XDE [38], Codagen Architect [13], b+m Generator Framework [8].

2. Классификация методов трансформации моделей. Следует отметить, что таксономия (рис. 1) не является нормативной[15, 16, 17,23]. К сожалению, относительно новая область трансформации моделей содержит большое количество перегруженных терминов, и многие термины, которые использованы в таксономии, часто используются с другими значениями в оригинальном описании различных подходов.

Способ применения правил трансформации позволяет трансформации ограничить части модели, принимающие участие в трансформации. Некоторые подходы поддерживают гибкие границы исходной модели (например, XDE или GReAT), в которых могут быть установлены границы, меньшие, чем целая модель. Последнее может быть важным по причине производительности. Целевая граница — это граница целевой модели, в которую RHS будет раскрываться.

Отношения между исходной и целевой моделью. Некоторые подходы управляют созданием новой целевой модели, которая должна быть отделена от исходной (CDI). В других подходах, целевая и исходная модели всегда являются одной моделью, то есть поддерживают лишь «обновление на месте» (VIATRA, GReAT).

Остальные подходы (например, XDE) позволяют целевой модели быть вновь создаваемой моделью или же старой, трансформированной, моделью. Последние подразумевают редактирование на месте. Более того, подход может позволять деструктивное обновление существующей модели или же обновление только за счет расширения, например, когда существующие элементы модели не могут быть удалены. Подходы, использующие недетерминистическую выборку и

Рис. 1. Таксономия для классификации подходов к трансформации моделей проектирования.

транзакционные итерации, могут ограничивать редактирование на месте только в рамках расширения для поддержания завершенности (VIATRA).

Стратегии применения правил. Правило должно применяться к определенным элементам внутри области применения, так как может быть более одного совпадения для правила внутри данной области. Стратегия может быть детерминистической, недетерминистической или даже итеративной. Например, детерминистическая стратегия может использовать какую-нибудь стандартную обходящую стратегию (например, просмотр в глубину) по всей содержащейся в области

иерархии. STR [33] является примером языка элементного переписывания с богатым механизмом построения обходов в древовидных структурах. Примеры недетерминистических стратегий включают «применение к одной точке», когда правило применяется к одной недетерминистически выбранной области, и «параллельное применение», когда одно правило применяется одновременно ко всем подходящим элементам в области. Иногда возможность применения правила определяется интерактивно.

Расположение цели для правила обычно определяется детерминистически. В случае редактирования на месте исходное положение становится целевым расположением (VIATRA, GReAT). В подходах с разделенными целевой и исходной моделями контрольные связи могут быть использованы для определения цели (CDI): правило может следовать по контрольным связям к какому-нибудь целевому элементу, который был создан другим правилом и использовать этот элемент как собственную цель.

Порядок применения правил. Определяют порядок, в котором конкретное правило применяется. Они делятся на четыре главные категории:

Форма: аспект порядка применения правил может быть выражен неявно или явно. Неявный порядок подразумевает, что человек не имеет явного контроля над алгоритмом порядка применения правил, определенного инструментом (BOTL, OptimalJ). Единственная возможность для пользователя повлиять на системно-определенный алгоритм это создать паттерны и логику правил для гарантирования определенной последовательности применения. Например, определенное правило может проверять какую-нибудь информацию, которую могут дать только другие правила. Явный порядок может быть внутренним или внешним. Внешний порядок имеет специальные конструкции для явного контроля над порядком выполнения. При внешнем порядке существует четкое разделение между правилами и логикой порядка применения (в VIATRA порядок применения правил обеспечивается внешним автоматом с конечными состояниями). В отличие от этого внутренний порядок - это механизм, позволяющий правилу трансформации прямо вызывать другие правила (CDI, Jamda и большинство шаблонных подходов, которые предоставляют возможность вызова других шаблонов)

Выбор правила: правила могут быть выбраны по явным критериям (Jamda). Некоторые подходы допускают недетерминистический выбор (BOTL). В качестве альтернативы может быть предоставлен

механизм разрешения конфликтов, основанный на приоритетах (в то же время ни один из подходов не реализует механизм разрешения конфликтов). Интерактивный выбор правил также возможен (XDE).

Выполнение правил: механизм выполнения правил включает рекурсии, циклы и транзакционные итерации (например, повторное выполнение, пока не произойдут какие-либо изменения).

Фазы: процесс трансформации может быть организован в несколько фаз, каждая из которых имеет определенное назначение, и только определенный набор правил может быть исполнен в каждой фазе. Например, структурно-ориентированные подходы, такие как OptimalJ и IOPT [20] имеют отдельную фазу для создания элементной иерархии для целевой модели и отдельную фазу для расстановки атрибутов и связей в целевой модели.

Организация правил связана с составлением и структуризацией множества правил трансформации. Существуют три области вариации в данном контексте:

Модульные механизмы: некоторые подходы позволяют запаковывать правила в модули (AST+, VIATRA). Модуль может импортировать другой модуль для доступа к его содержимому.

Механизмы переиспользования: механизмы переиспользования предлагают путь для определения правил, основанный на одном или более правиле. В общем случае механизмы порядка применения правил могут быть использованы для определения композиции правил трансформации, однако, некоторые подходы предлагают специальные механизмы переиспользования, такие как наследование между правилами (например, наследование в AST+, расширение в CDT, специализация в QVTP, ответвление в IOPT), наследование между модулями (наследование между частями в AST+), логическая композиция (QVTP).

Организационная структура: правила могут быть организованы в соответствии со структурой исходного языка (как в атрибутных грамматиках, где операции присоединены к элементам в исходном языке) или целевого языка, или могут иметь собственную независимую организацию. Примером организации в соответствии со структурой цели является IOPT. В этом подходе есть одно правило для каждого элемента цели, и правила применяются в соответствии с элементной иерархией целевой метамодели. Например, если целевой язык имеет пакетные конструкции, в которых могут содержаться классы, правило для создания пакетов будет содержать правило для создания классов (которое будет содержать правила для создания атрибутов и методов).

Связи измерений. Трансформация может записывать связи между исходными и целевыми элементами. Эти связи могут быть полезны при анализе влияний (то есть анализа того, как изменения одной модели влияют на зависящие от нее модели), синхронизации между моделями, отладке на основе моделей (например, отображение пошагового исполнения реализации на ее модель высокого уровня) и определении цели трансформации.

Некоторые подходы предоставляют специальную поддержку единства измерений (CDI, IOPT), в то время как другие предполагают, что пользователь сам будет за этим следить, используя те же механизмы, что и для добавления других видов связи в моделях (VIATRA, GReAT). Некоторые подходы со специальной поддержкой связей для единства измерений требуют от разработчиков вручную закодировать создание таких связей в правилах трансформации (CDI), в то время как другие создают такие связи автоматически (IOPT). В таких случаях подход может также предоставлять контроль над тем, как много таких связей будет создаваться (для ограничения количества избыточной информации). Наконец, есть выбор того, где эти связи будут храниться. Например, в исходной и/или целевой модели, или отдельно. Предпочтительнее подход, который подразумевает хранение GUID в каждом элементе модели и сохранение информации по единству измерений отдельно от моделей.

Направленность. Трансформация может быть однонаправленной или двунаправленной. Однонаправленная трансформация может производиться лишь в одном направлении, в таком случае целевая модель создается (или обновляется) на основе исходной модели. Двунаправленная трансформация может быть достигнута с помощью двунаправленных правил или с помощью определения двух отдельных дополнительных однонаправленных правил, по одному на каждое направление.

Правило трансформации (рис. 2) состоит из двух частей: левой и правой (LHS RHS). LHS обращается к исходной модели, тогда как RHS соответствует целевой модели. Обе части могут быть представлены комбинацией следующих терминов:

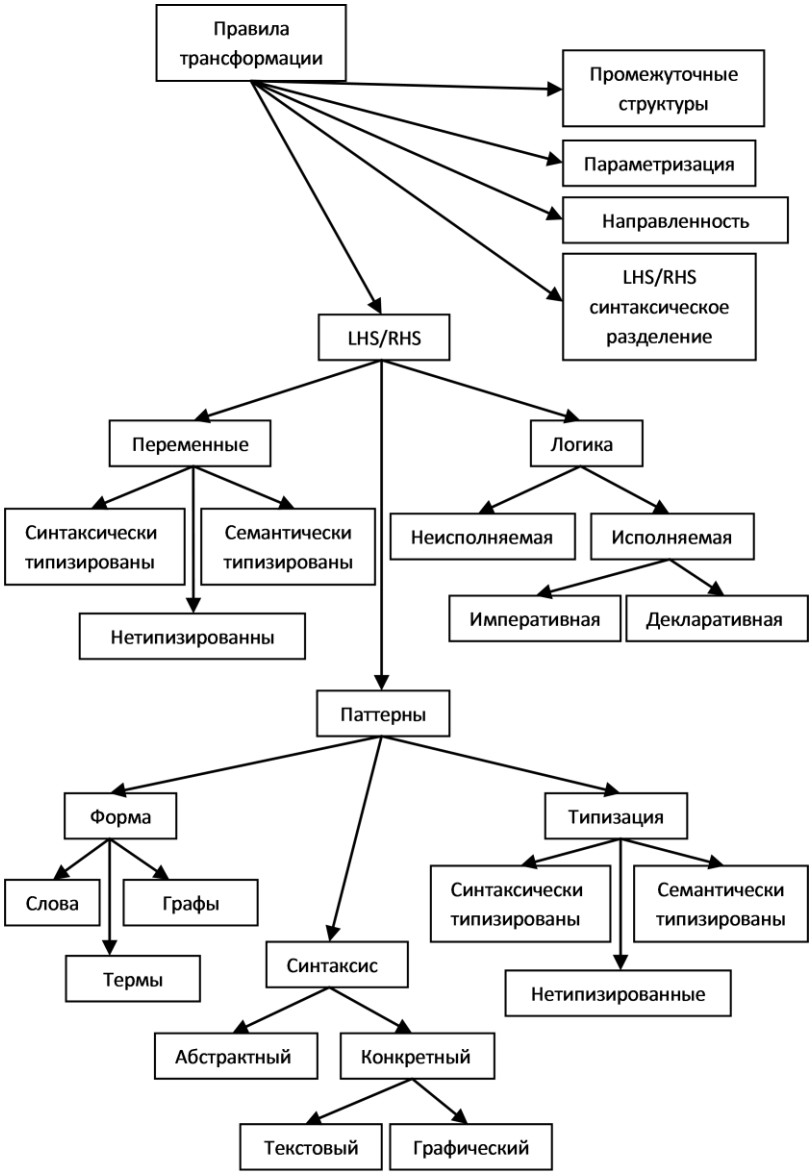

Рис. 2. Свойства правил трансформации.

Логика представляет вычисления и ограничения элементов модели. Логика может быть выполняемой и невыполняемой. Невыполняемая логика используется для определения отношений между моделями [32]. Выполняемая логика может иметь декларативную и императивную форму. Примеры декларативной формы включают OCL-запросы [29] для извлечения элементов из исходной модели [38] и явное создание целевых элементов по ограничениям [11]. Императивная логика часто имеет вид языка программирования с кодом, использующим API для прямого манипулирования моделями. Например, JMI [22] предоставляет Java API для доступа в репозиторий MOF. В контексте QVT стандартизации, семантика UML Action [34] может быть использована для определения императивной логики в форме, которая может быть автоматически отображена в различные языки программирования.

Переменные содержат элементы из исходной/целевой модели (или промежуточные элементы). Они иногда относятся к метапеременным для различия их от переменных, которые могут быть частью трансформируемой модели.

Паттерны— это части модели с одной или многими переменными. Есть строковые, графовые паттерны и паттерны терминов. Строковые паттерны используются в текстовых шаблонах. Трансформация модель–модель обычно использует паттерны графов или терминов. Паттерны могут быть представлены с помощью абстрактного или конкретного синтаксиса соответствующей исходной/целевой модели, а синтаксис может быть текстовым или графическим.

Паттерны и переменные могут быть нетипизированны либо типизированы синтаксически или семантически. В случае синтаксической типизации переменная ассоциируется с элементом метамодели, чьи значения она может принимать. Семантическая типизация позволяет ассоциировать более конкретные свойства. Например, синтаксический тип переменной может быть «выражением», тогда как семантический тип может быть «выражением, выполняемым над целым значением». Последнее не доступно в существующих языках трансформации, но поддерживается в некоторых метаязыках, таких как MetaML и MetaOcaml [26, 27].

Синтаксическое разделение. RHS и LHS могут быть синтаксически разделены. Другими словами, синтаксис правила может специально отличать RHS и LHS, или же не делать синтаксического различия.

Параметризация правила. Правила могут иметь дополнительные контролирующие параметры для конфигурации.

Промежуточные структуры. Некоторые подходы (VIATRA и GReAT) требуют построения промежуточных структур. Это особенно значимо, когда трансформация модели происходит в самой модели (замещение элементов).

Правила трансформации обычно предполагают наличие функциональной черты: имея входные данные из исходной модели, они дают конкретный результат в целевой модели. Декларативные правила (то есть использующие декларативную логику и/или паттерны) могут также часто применяться в обратном направлении. Однако, поскольку разные входы могут вести к одному выходу, инверсия правила может не быть функцией. В таком случае, инверсия может подсчитать количество возможных решений (теоретически их может быть бесконечное число) или только представить часть конкретного результата (потому что часть может быть общей для всех решений) и использовать переменные, значения по умолчанию или значения, которые уже присутствуют в выходных данных для других частей. Возможность инверсии трансформации зависит не только от инверсии правил трансформации, но и также от инверсии логики расписания. Инверсия набора правил может не приводить к какому-либо результату из-за невозможности завершения.

В целом, можно представить трансформацию моделей в код как случай трансформации модель–модель; нужно только описать метамодель для целевого языка программирования. Однако, по практическим соображениям для переиспользования существующих технологий компилирования код зачастую генерируется как простой текст, который затем загружается в компьютер. Поэтому различают трансформации модель–код (которые лучше называть модель–текст, так как можно сгенерировать безкодовые артефакты, например XML) и трансформации модель–модель. Некоторые приложения предлагают оба вида трансформации (Jamda, XDE, OptimalJ).

3. Подходы модель–код.

Подход на основе паттерна «посетитель». Самый общий подход состоит в предоставлении механизма посетителей для прохождения по внутреннему представлению модели и написанию кода в выходной поток. Примером такого подхода является Jamda, который является объектно-ориентированной инфраструктурой, предоставляющей набор классов для описания моделей UML, API для манипулирования моделями и механизм посетителей (так называемых CodeWriters) для генерирования кода.

Подход на основе шаблонов. Большинство доступных на текущий

момент MDA средств поддерживают основанную на шаблонах генерацию модель–код, например “b+m Generator Framework”, JET, FUUT-je, Codagen Architect, AndroMDA, ArcStyler, OptimalJ и XDE (последние два также представляют трансформацию модель–модель). AndroMDA переиспользует существующую технологию генерации на основе шаблонов с открытым кодом: Velocity [37] и XDoclet [39].

Шаблон обычно состоит из целевого текста, содержащего вставки мета-кода для доступа к информации из исходных данных и выполнения выборки кода и итеративного разложения [12]. В соответствии с терминологией LHS использует выполняемую логику для доступа к исходным данным; RHS соединяет нетипизированные, строковые паттерны с выполняемой логикой для селекции кода и итеративного разложения. Не существует синтаксического различия между LHS и RHS.

Логика LHS для доступа к исходной модели может иметь различные формы. Она может быть как простым Java кодом для доступа к API, поддержанную внутренним представлением исходной модели (например, JMI), или это могут быть декларативные запросы (например, в OCL или XPath [40]).

По сравнению с трансформацией на основе посетителей структура шаблонов имеет близкое сходство с кодогенерацией. Шаблоны сами по себе ведут к итеративной разработке, так как они могут быть легко взяты из примеров. Так как подход на основе шаблонов оперирует с текстом, паттерны, которые они содержат, являются нетипизированными и могут представлять синтаксически или семантически неверные фрагменты кода. С другой стороны, текстовые шаблоны независимы от целевого языка и упрощают генерацию любых текстовых артефактов, включая документацию.

4. Подходы модель–модель. Трансформации модель–модель переводят исходную модель в целевую (которые могут быть экземплярами одной или разных метамоделей). Все такие подходы поддерживают синтаксическую типизацию переменных и паттернов.

Большинство существующих MDA средств предоставляют только трансформации модель–код, которые они используют для генерации PSM (в таком случае играя роль лишь создателя кода) из PIM. Когда происходит интеграция сложных абстрактных связей между PIM и PSM, то легче сгенерировать промежуточные модели, чем прямо создавать целевую PSM. Это делает трансформацию более модульной и управляемой. Также промежуточные модели могут быть нужны для оптимизации и настройки (или, как минимум, для отладки).

Подходы прямого манипулирования. Эти подходы предоставляют

внутреннее представление модели и API для манипулирования ею. Обычно они реализованы как объектно-ориентированный каркас разработки, который также может предоставлять минимальную инфраструктуру для организации трансформации. Однако пользователи должны имплементировать правила и расписание в большинстве случаев с нуля, используя языки программирования, такие как Java.

Подходы на основе отношений. В этой категории собраны декларативные подходы, в которых концепцией являются математические отношения (QVTP, CDI, декларативные подходы GLR [*2,* 19] и правила отображения AST+).

Основной идеей является определение исходного и целевого типа элемента отношений и спецификация его ограничений. В чистом виде такая спецификация не выполнима. Однако декларативным ограничениям может быть придана семантика выполнения, такая же, как в логическом программировании. Подход в QVTP различает отношения, которые являются двунаправленными, неисполняемыми спецификациями трансформаций в своем каркасе, и отображения, которые являются исполнимыми, однонаправленными трансформациями, реализующими отношения.

Все подходы на основе отношений не приводят к побочным эффектам. Они часто поддерживают проход с возвратом (например, GRL и QVTP) и, в отличие от подходов прямого императивного манипулирования, создают целевые элементы неявно (например, GRL и CDI). Подходы на основе логического программирования (GRL, CDI) требуют строгого разделения между исходной и целевой моделями, то есть они не позволяют проводить обновление на месте.

Подходы на основе трансформации графов. Эта категория подходов к трансформации выросла на основе теоретических основ графовых трансформаций. В частности, эти подходы оперируют типизированными, раскрашенными графами с атрибутами [*3*], которые являются специально разработанными типами графов для представления UML-подобных моделей. Примерами этого подхода к трансформации моделей являются VIATRA, ATOM, GReAT, UMLX и BOTL.

Правила графовых трансформаций состоят из LHS графого паттерна и RHS графового паттерна. Графовые паттерны могут быть описаны в конкретном синтаксисе соответствующего исходного или целевого языка (VIATRA) или в абстрактном синтаксисе MOF (как в BOTL). Преимуществом конкретного синтаксиса заключается в том, что он более знаком разработчикам, работающим с определенным языком моделирования, чем абстрактный синтаксис. Также для ком-

плексных языков, таких как UML, паттерны в конкретном синтаксисе стремятся быть более лаконичными, чем паттерны в соответствующем абстрактном синтаксисе [24]. С другой стороны, легче предоставить определенную по умолчанию трансформацию для абстрактного синтаксиса, который будет работать для любой метамодели, что полезно, когда недоступен специализированный конкретный синтаксис.

Также как и подходы на основе отношений, подходы на основе графовых трансформаций могут представлять трансформацию моделей в декларативной форме. Однако обеспечение специальных возможностей, таких как графовые паттерны и отображения на основе графовых паттернов, отличает графовую трансформацию от подходов на основе отношений.

Структурные подходы. Подходы в данной категории имеют две четко различных фазы: первая фаза связана с созданием иерархической структуры целевой модели, в то время как вторая фаза устанавливает атрибуты и связи в целевой модели. Весь каркас определяет расписание и стратегии приложения; пользователь имеет возможность только предоставить правила трансформации.

Примером подхода является каркас трансформации модель-модель у OptimalJ. Каркас реализован на Java и предоставляет так называемые инкрементальные копии, которые пользователь должен наследовать для определения собственных правил трансформации. Базовым элементом является идея копирования элементов модели из исходной в целевую, которые затем могут быть адаптированы для предоставления декларативного интерфейса. Правило трансформации реализовано как метод с входным параметром, чей тип определяет исходный тип для правила, а сам метод возвращает Java объект, представляющий собой класс целевого элемента модели. Правилам запрещено иметь побочные эффекты, а расписание полностью определено каркасом.

Другим структурным подходом является IOPT. Специфическое свойство данного подхода — организация правил, ориентированная на целевую модель, где есть одно правило на каждый элемент цели, и положение правила соответствует контейнерной организации в целевой метамодели. Выполнение такой модели может быть представлено как конфигурация «сверху-вниз» целевой модели.

Гибридные подходы. Гибридные подходы являются комбинацией техник из предыдущих категорий. Язык правил трансформации AST+ является композицией декларативного и императивного подхода. Он также может быть классифицирован в категории «отношений». Также

как QVTP, он проводит различие между исходными и целевыми элементами, которые ограничены набором инвариантов. Они похожи на отношения в QVTP и подходят в категорию «отношений».

Операционные правила в TRL представляют собой выполнимые правила трансформации. В отличие от правил отображения операционные правила явно указывают: правила создают, обновляют или удаляют элементы. Расписание явно задано во внутренней форме, в то время как правило явно вызывает другие правила в своем теле. Поддерживается наследование правил. Правила могут быть собраны в модули. Наследование между модулями (с перегрузкой) также поддерживается.

Язык трансформации атласов ATL [9] также является гибридным подходом, несколько похожим на TLR. Правило трансформации в ATL может быть полностью декларативным, гибридным или полностью императивным. LHS полностью декларативного правила (так называемый исходный паттерн) состоит из набора синтаксически типизированных переменных с опциональным OCL ограничением как фильтром или направляющей логикой. RHS полностью декларативного правила (так называемый целевой паттерн) состоит из набора переменных и некоторой декларативной логики для связывания значений атрибутов в целевых элементах. ATL предоставляет и явное, и неявное расписание. Неявный алгоритм расписания начинается с вызова правила, которое специфицировано как входная точка, которая может вызывать следующие правила. После завершения первой фазы, автоматически проверяются совпадения по исходным паттернам и выполняются соответствующие правила. Наконец, выполняется намеченная выходная точка. Явное внутреннее расписание поддерживается за счет возможности вызова правила из императивного блока другого правила.

5. Заключение. Существуют вполне удовлетворительные решения для трансформации модель–текст (такие как шаблонные подходы), однако они не подходят для трансформации модель–модель. Было предложено множество новых подходов к трансформации модель–модель, но имеющегося на данный момент опыта недостаточно, чтобы эффективно применить их в практических целях. Из-за этого исследование трансформаций все еще находится на стадии изучения возможностей и установления требований. Средства моделирования, доступные на рынке, только начинают предлагать некоторые возможности трансформации модель–модель, но они очень ограничены и применяются главным образом в частных случаях, например, без подходящей теоретической основы. Большинство этих средств нацелено на генера-

цию EJB приложений, и модели трансформации, которые они предлагают, были специально созданы, чтобы достичь этой цели. Можно выделить следующие моменты, связанные с описанной ранее терминологией и классификацией:

1. Прямая манипуляция – наиболее низкоуровневый подход. Он предлагает пользователю лишь небольшую поддержку или даже отсутствие таковой. В целом, вся работа должна быть проделана самим пользователем. В долгосрочной перспективе этот подход становится непрактичным.

2. Категория структурных трансформаций содержит прагматические подходы, которые созданы в контексте конкретных видов приложений (и поэтому наиболее подходят к ним), таких как создание EJB реализаций и схем баз данных из UML моделей. Неясным остается, насколько хорошо эти подходы могут поддерживать другие виды приложений.

3. Подходы на основе трансформаций графов имеют объемную теоретическую базу в графовых трансформациях. Эти подходы являются мощными, декларативными, но при этом и самыми сложными. Сложность идет от неопределенности в расписании и стратегии применения, которые требуют аккуратного рассмотрения завершения процесса трансформации и порядка применения правил (включающих свойство «слияния»).

4. Подходы на основе отношений соблюдают хороший баланс между гибкостью и декларативным выражением. Они обеспечивают гибкое расписание и хороший контроль над неопределенностью.

5. Гибридные подходы позволяют пользователям смешивать и сопоставлять различные концепции и парадигмы в зависимости от области применения. Практические подходы, скорее всего, имеют гибридный характер.

Учет различного рода особенностей проектирования для подходов к трансформации моделей требует большого количества экспериментов и практического опыта.

Литература

1. *Agrawal A., Karsai G., Shi F.* Graph Transformations on Domain-Specific Models // Journal on Software and Systems Modeling, 2003. URL: http://atlanmod.emn.fr/www/papers/ext/Agrawal_A_11_0_2003_Graph_Tran.pdf (дата обращения: 10.10.2011).
2. *Akehurst D. H., Kent S.* A Relational Approach to Defining Transformations in a Metamodel // J.-M. Jézéquel, H. Hussmann, S. Cook (Eds.): UML 2002 – The Unified Modeling Language 5th International Conference, Dresden, Germany, September 30 –October 4, 2002. Proceedings, LNCS 2460, pp. 243–258, 2002.

3. *Andries M., Engels G., Habel A., Hoffmann B., Kreowski H.-J., Kuske S., Kuske D., Plump D., Schürr A., Taentzer G.* Graph Transformation for Specification and Programming // Technical Report 7/96, Universität Bremen, 1996, URL: http://citeseer.nj.nec.com/article/andries96graph.html (дата обращения: 10.10.2011).
4. *AndroMDA 2.0.3*.URL: http://www.andromda.org (дата обращения: 10.10.2011).
5. *ArcStyler4.0*.URL: http://www.arcstyler.com (дата обращения: 10.10.2011).
6. *AST+* Alcatel, Softeam, Thales, TNI-Valiosys, Codagen Corporation // MOF Query/Views/Transformations, Revised Submission. OMG Document: ad/03-08-05. URL: http://www.omg.org/cgi-bin/doc?ad/03-08-05 (дата обращения: 10.10.2011).
7. *ATOM3* A Tool for Multi-Paradigm modeling // URL: http://atom3.cs.mcgill.ca (дата обращения: 10.10.2011).
8. *b+m* ArchitectureWare, Generator Framework // OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture. URL: http://www.architectureware.de (дата обращения: 10.10.2011).
9. *Bézivin J., Dupé G., Jouault F., Rougui J. E.* First experiments with the ATL model transformation language: Transforming XSLT into XQuery // In the online proceedings of the OOPSLA'03 Workshop on Generative Techniques in the Context of the MDA, URL: http://www.softmetaware.com/oopsla2003/mda-workshop.html (дата обращения: 10.10.2011).
10. *Braun P., Marschall F.* The Bi-directional Object-Oriented Transformation Language // Technical Report, Technische Universität München, TUM-I0307, May 2003. URL: http://wwwbroy.in.tum.de/publ/papers/TUM-I0307.pdf (дата обращения: 10.10.2011).
11. *CDI* CBOP, DSTC, and IBM // MOF Query/Views/Transformations, Revised Submission. OMG Document: ad/03-08-03. URL: http://www.omg.org/cgi-bin/doc?ad/03-08-03 (дата обращения: 10.10.2011).
12. *Cleaveland C.* Program Generators with XML and Java // Prentice-Hall, 2001, URL: http://www.craigc.com/pg (дата обращения: 10.10.2011).
13. *Codagen Architect*. URL: http://www.codagen.com/products/architect/default.htm (дата обращения: 10.10.2011).
14. *CS* Compuware Corporation and Sun Microsystems // MOF Query/Views/Transformations, Revised Submission. OMG Document: ad/03-08-07. URL: http://www.omg.org/cgi-bin/doc?ad/03-08-07 (дата обращения: 10.10.2011).
15. *Czarnecki K, Helsen S.* Classification of Model Transformation Approaches // Proceedings of the OOPSLA'03 Workshop on the Generative Techniques in the Context Of Model-Driven Architecture, Anaheim, California, USA, 2003. URL: http://www.swen.uwaterloo.ca/~kczarnec/ECE750T7/czarnecki_helsen.pdf (дата обращения: 10.10.2011).
16. *Czarnecki K.* Domain Engineering // Chapter in the Wiley Software Engineering Encyclopedia, Second Edition, John Marciniak, (Eds.), Wiley and Sons, Inc., February 2002, pp. 433–444.
17. *Czarnecki K.* Generative Programming: Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models // Ph.D. Thesis, Computer Science Department, Technical University of Ilmanau, Ilmanau, Germany, 1998, URL: http://www.prakinf.tuilmenau.de/~czarn/diss (дата обращения: 10.10.2011).
18. *FUUT-je* // hosted at the Eclipse Generative Model Transformer (GMT) project website, URL:
http://dev.eclipse.org/viewcvs/indextech.cgi/~checkout~/gmthome/download/index.html (дата обращения: 10.10.2011).

19. *Gerber A., Lawley M., Raymond K., Steel J., Wood A.* Transformation: The Missing Link of MDA // Graph Transformation: First International Conference (ICGT 2002), Barcelona, Spain, October 7–12, 2002. Proceedings. LNCS vol. 2505, Springer-Verlag, 2002, pp. 90–105.
20. *IOPT* Interactive Objects and Project Technology // MOF Query/Views/Transformations, Revised Submission. OMG Document: ad/03-08-13. URL: http://www.omg.org/cgi-bin/doc.ad/03-08-13 (дата обращения: 10.10.2011).
21. *Jamda* The Java Model Driven Architecture 0.2 // May 2003, OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture 17. URL: http://sourceforge.net/projects/jamda (дата обращения: 10.10.2011).
22. *JMI* Java Metadata Interface 1.0.URL: http://java.sun.com/products/jmi (дата обращения: 10.10.2011).
23. *Kang K., Cohen S., Hess J., Nowak W., Peterson S.* Feature-Oriented Domain Analysis (FODA) // Feasibility Study. Technical Report, CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, November 1990. URL: http://www.sei.cmu.edu/reports/90tr021.pdf (дата обращения: 10.10.2011).
24. *Marschall F., Braun P.* Model Transformations for the MDA with BOTL // CTIT Technical Report TR-CTIT-03-27, Enschede, The Netherlands, University of Twente (2003) pp. 25–36.
25. *Mens T., Czarnecki K., Gorp P.* A Taxonomy of Model Transformations // Dagstuhl Seminar Proceedings 04101. 2005. URL: http://drops.dagstuhl.de/opus/volltexte/2005/11 (дата обращения: 10.10.2011).
26. *MetaML* // URL: http://www.cse.ogi.edu/PacSoft/projects/metaml (дата обращения: 10.10.2011).
27. *MetaOcaml*Meta Objective-Caml.URL: http://www.cs.rice.edu/~taha/MetaOCaml (дата обращения: 10.10.2011).
28. *MOF* Meta Object Facility 1.4 // OMG Document: formal/02-04-03. URL: http://www.omg.org/cgi-bin/doc?ad/02-04-03 (дата обращения: 10.10.2011).
29. *OCL* The Object Constraint Language Specification 2.0, OMG Document: ad/03-01-07URL: http://www.omg.org/cgi-bin/doc?ad/03-01-07 (дата обращения: 10.10.2011).
30. *OptimalJ* 3.0. URL: http://www.compuware.com/products/optimalj (дата обращения: 10.10.2011).
31. *QVT* Query/Views/Transformations RFP // MOF 2.0 OMG Document: ad/2002-04-10, revised on April 24, 2002. URL: http://www.omg.org/spec/QVT (дата обращения: 10.10.2011).
32. *QVT-Partners* MOF Query/Views/Transformations // Revised Submission. OMG Document: ad/2003-08-08. URL: http://www.omg.org/cgi-bin/doc.ad/03-08-08 (дата обращения: 10.10.2011).
33. *STR* Strategies for Program Transformation.URL: http://www.stratego-language.org (дата обращения: 10.10.2011).
34. *UML Action* Action Semantics for the UML // Object Management Group, 2001. ad/2001-08-04. URL: http://www.omg.org/cgi-bin/doc?ad/01-08-04 (дата обращения: 10.10.2011).
35. *UML* Unified Modeling Language // Object Management Group. URL: http://www.uml.org (дата обращения: 10.10.2011).
36. *Varro D., Varro G., Pataricza A.* Designing the automatic transformation of visual languages // Science of Computer Programming, vol. 44(2):pp. 205–227, 2002.
37. *Velocity* 1.3.1 // The Apache Jakarta Project, March 2003, URL: http://jakarta.apache.org/velocity (дата обращения: 10.10.2011).

38. *XDE*RationalXDE.URL: http://www.rational.com/products/xde (дата обращения: 10.10.2011).
39. *XDoclet* Attribute Oriented Programming.URL: http://xdoclet.sourceforge.net (дата обращения: 10.10.2011).
40. *XPath* XML Path Language Version 1.0 // W3C, November 1999, URL: http://www.w3.org/TR/xpath (дата обращения: 10.10.2011).

Атисков Алексей Юрьевич— канд. техн. наук; младший научный сотрудник лаборатории информационно-вычислительных систем Учреждения Российской академии наук Санкт-Петербургский институт информатики и автоматизации РАН (СПИИРАН). Область научных интересов: технологии автоматизированной трансформации диаграмм проектирования, OWL-описание предметных областей, метамоделирование информационных систем. Число научных публикаций — 8. atiskov@gmail.com; СПИИРАН, 14-я линия В.О., д. 39, г. Санкт-Петербург, 199178, РФ; р.т. +7(812)328-4369, факс +7(812)328-4450.

Atiskov Alexey Jurievich —researcher, Laboratory of Computer and Informational Systems, St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS). Research interests: technologies of semi-automatic systems that transform modeling diagrams, OWL-description of subjects, metamodeling of informational systems. The number of publications — 8. atiskov@gmail.com; SPIIRAS, 39, 14-thLine V.O., St. Petersburg, 199178, Russia; office phone +7(812)328-4369, fax +7(812)328-4450.

Рекомендовано лабораторией информационно-вычислительных систем СПИИРАН, заведующий лабораторией Воробьев В.И, д-р техн. наук, проф.
Статья поступила в редакцию 12.04.2012.

РЕФЕРАТ

Атисков А.Ю. **Подходы к трансформации моделей проектирования информационных систем.**

Современные технологии переиспользования результатов проектирования программного обеспечения направлены в основном на поддержку объектно-ориентированных методов моделирования, на кодогенерацию и современные стандарты. Программные комплексы поддержки этих технологий неудобны в использовании без постоянной помощи со стороны производителя и обладают избыточной сложностью в настройке. В статье представлена таксономия для классификации некоторых существующих и возможных подходов к трансформации моделей, которая включает в себя способ применения правил, отношения между исходной и целевой моделью, стратегии применения правил, порядок применения правил, организацию правил, связи измерений, направленность и сами правила трансформации, части которых могут быть представлены комбинацией переменных, паттернов, логики, синтаксического разделения, параметризацией правил и промежуточных структур.

Различают трансформации модель–код и трансформации модель–модель.

Подходы модель–код включают в себя подход на основе паттерна «посетитель», который является самым общим подходом, и подход на основе шаблонов, который поддерживается большинством доступных на текущий момент MDA средств.

Подходы модель–модель переводят исходную модель в целевую. Подходы прямого манипулирования предоставляют внутреннее представление модели и API для манипулирования им. Подходы на основе отношений состоят из декларативных подходов. Подходы на основе трансформации графов выросли на основе теоретических основ графовых трансформаций. Структурные подходы имеют две четко различных фазы: первая фаза связана с созданием иерархической структуры целевой модели, в то время как вторая фаза устанавливает атрибуты и связи в целевой модели. Гибридные подходы являются комбинацией техник из предыдущих категорий.

Существуют вполне удовлетворительные решения для трансформации модель–текст, однако они не подходят для трансформации модель–модель. Было предложено множество новых подходов к трансформации модель–модель, но имеющегося на данный момент опыта недостаточно, чтобы эффективно применить их в практических целях. Средства моделирования, доступные на рынке, только начинают предлагать некоторые возможности трансформации модель–модель, но они очень ограничены и применяются главным образом в частных случаях, например, без подходящей теоретической основы. Представленная таксономия для классификации подходов позволяет учитывать различного рода особенности проектирования для подходов к трансформации моделей и проводить сравнение подходов для обоснованного выбора наиболее эффективного подхода.

SUMMARY

Atiskov A.J. **Approaches towards transformation of informational system's models.**

Modern technology reusability of software design is focused mainly on support for object-oriented modeling techniques for code generation and modern standards. Software systems support for these technologies are difficult in their use without constant assistance from the manufacturer and have excessive complexity in configuration. The paper presents a taxonomy for the classification of some existing and possible approaches to model transformation, which includes the method of application of the rules, the relationship between the source and target model, strategies for applying the rules, the order of application of the rules, organization rules, connection dimensions, orientation and transformation rules themselves, parts of which can be represented by a combination of variables, patterns, logic, syntactic, separation, and parameterization of the rules of intermediate structures.

There are several model–code and model–model transformation.

Approaches model–code include the approach based on pattern "visitor", which is the most common approach, and an approach based on a template, which is supported by the most currently available MDA tools.

Model–model approaches transform the original model to the target. Direct manipulation approaches provide the internal representation of the model and API for their manipulating. Approaches based on relationships consist of declarative approaches. Graph transformation approaches are based on the basis of the theoretical foundations of graph transformations. Structural approaches have two distinctly different phases: the first phase involves the creation of a hierarchical structure of the target model, while the second one sets the attributes and relationships in the target model. Hybrid approaches are a combination of techniques from the previous categories.

There are quite satisfactory solutions for the transformation model text, but they are not suitable for transformation model–model. It was suggested a lot of new approaches to model transformation model, but at the moment the available experience is not enough to effectively apply them in practice. Modeling tools available in the market are just beginning to offer some possibility of transforming the model–model, but they are very limited and mainly used in special cases, for example, without a suitable theoretical framework. The presented taxonomy for the classification of approaches provides a possibility to take into account various features of the design for the approaches to model transformation and provides a possibility to compare approaches in order to choose the most effective approach.