OSTIS-2016
(Open Semantic Technologies for Intelligent Systems)

УДК 004.89

ПОДХОД К АВТОМАТИЗАЦИИ СОЗДАНИЯ БАЗ ЗНАНИЙ НА ОСНОВЕ ТРАНСФОРМАЦИИ КОНЦЕПТУАЛЬНЫХ МОДЕЛЕЙ

Дородных Н.О., Юрин А.Ю.

Институт динамики систем и теории управления имени В.М. Матросова СО РАН, г. Иркутск, Россия

tualatin32@mail.ru

iskander@icc.ru

Рассмотрена концепция подхода к созданию программных компонентов, обеспечивающих автоматизированную разработку баз знаний (OWL и CLIPS) на основе трансформации концептуальных моделей, представленных в формате XML. Предлагается архитектура сервис-ориентированной программной системы (и ее основных элементов), реализующая данный подход. Описана модель типового программного компонента, включая модель трансляции, на основе которой создаются (специализируются) компоненты. Описан метод (алгоритм) трансформации концептуальных моделей в программный код баз знаний с возможностью уточнения (моделирования) продукций в нотации RVML.

Ключевые слова: получение знаний; сервис-ориентированная система; концептуальная модель; базы знаний; трансформация моделей.

Введение

В настоящее время разработка новых методов и подходов к созданию интеллектуальных систем (ИС) остается перспективной областью научных исследований. При этом особый интерес представляет процесс формирования баз знаний (БЗ), который всегда считался традиционно «узким местом». На этапе разработки БЗ решаются задачи моделирования предметной области, идентификации (получения), концептуализации (структурирования) и формализации (представления) знаний и их описание на определенном языке представления знаний (ЯПЗ, или языке программирования баз знаний – ЯПБЗ) [Гаврилова и др., 2000]. Дополнительные сложности возникают, если этот процесс носит удаленный и распределенный характер и требует согласования мнений экспертов.

Эффективность создания БЗ может быть повышена путем автоматизации анализа и повторного использования (трансформации) концептуальных моделей предметных областей, построенных при помощи программных средств концептуального, когнитивного, онтологического моделирования или CASE-средств. Существующие в данной области решения обладают рядом недостатков, в частности: отсутствие возможности совместной, распределенной и одновременной работы пользователей; отсутствие или ограниченность генерации кода БЗ на различных ЯПБЗ; ограниченный набор поддерживаемых форматов концептуальных моделей, а также сложность описания самих моделей для генерации кода.

В связи с этим актуально создание веб-ориентированной (сервис-ориентированной) программной системы, обеспечивающей возможность проектирования БЗ и синтеза кода на различных ЯПБЗ путем трансформации концептуальных моделей.

Таким образом, целью работы является разработка моделей, методов и средств, обеспечивающих автоматизированное создание продукционных БЗ на основе концептуальных моделей, используя промежуточное их представление в виде онтологии и уточнение с использованием специальной графической нотации Rule Visual Modeling Language (RVML) [Грищенко и др., 2013]. Для достижения поставленной цели требуется разработать концепцию сервис-ориентированной программной системы, включая метод трансформации исходных концептуальных моделей в код БЗ на целевом ЯПБЗ.

В качестве источников концептуальных моделей предлагается использовать модели, синтаксис которых выражен на XML (наиболее распространённом формате хранения моделей).

Например, могут быть использованы диаграммы классов UML (Unified Modeling Language) [Booch et al., 2005], представленные в формате XML в соответствии со стандартом XMI (XML Metadata Interchange) [XMI] или концепт-карты, представленные с использованием стандарта XTM [XTM] и др. В качестве целевых ЯПБЗ выбраны CLIPS [Частиков и др., 2003] и OWL [Grau et al., 2008], как одни из широко используемых языков при создании ИС различного назначения.

1. Концепция

На основе результатов анализа моделей, методов и средств автоматизации создания БЗ на основе концептуальных моделей разработана концепция специализированной сервис-ориентированной программной системы.

Основными элементами концепции являются:

- архитектура сервис-ориентированной системы и ее основных элементов, основанные на принципах SaaS;
- модель типового программного компонента, включая модель трансляции и предметно-ориентированный (декларативный) язык для представления и хранения модели трансляции (ЯПМТ);
- модели онтологии и продукций, как основного информационного «ядра» сервис-ориентированной системы;
- метод (алгоритм) создания программных компонентов на основе «клонирования» типового программного компонента и его настройки (специализации), путем установления соответствий (правил трансформации) между элементами метамоделей;
- метод трансформации исходных концептуальных моделей в код БЗ на целевом ЯПБЗ, включающий моделирование (уточнение) продукций в нотации RVML.

1.1. Архитектура системы

Для разработки концептуальной архитектуры сервис-ориентированной системы исследовалась и специфицировалась модель «Программное обеспечение как услуга» (Software-as-a-Service, SaaS) облачного сервиса (системы).

Приведем формальную постановку первой основной задачи. В общем виде SaaS-модель облачного сервиса может быть представлена в виде:

$$SaaS = \langle R, S, A \rangle, \quad (1)$$

где R – набор предоставляемых сервисом ресурсов (услуг) пользователям; S – программное обеспечение, предоставляющее возможность повсеместного и удобного сетевого доступа к ресурсам R; A – аппаратное обеспечение, как вычислительные узлы (основа) для размещения программного обеспечения S.

Решая задачу разработки архитектуры, специализируем SaaS-модель (1) для сервис-ориентированной системы:

$$SaaS^* = \langle R^*, S^*, A \rangle, \quad (2)$$

где R^* – набор предоставляемых сервис-ориентированной системой ресурсов (услуг) пользователям; S^* – набор сервисов (подсистем), предоставляющий возможность повсеместного и удобного сетевого доступа к ресурсам сервис-ориентированной системы R^*. При этом:

$$S^* = \langle S^*_{SYS}, S^*_{USR}, I \rangle, \quad (3)$$

где S^*_{SYS} – множество системных сервисов обеспечивающих базовое взаимодействие пользователей с сервис-ориентированной системой и предоставляющих возможность проектирования, разработки и регистрации (развертывания) пользовательских прикладных сервисов (программных компонентов) синтеза БЗ; S^*_{USR} – множество разработанных и зарегистрированных прикладных сервисов (программных компонентов) в составе сервис-ориентированной системы; I – интерфейс взаимодействия с внешними ИС.

В свою очередь:

$$I = \{i_1, ..., i_n\}, i_j = \langle name_j, command_j \rangle, j \in \overline{1,n}, \quad (4)$$

где $name_j$ – наименование j метода взаимодействия; $command_j$ – управляющая команда метода.

Используя (2), уточним множество предоставляемых сервис-ориентированной системой ресурсов (функций) R^* для набора системных сервисов S^*_{SYS}:

- разработка программного компонента (веб-сервиса), на основе типового программного компонента, включая визуальное и текстовое построение соответствий (правил трансформации) элементов исходной и целевой метамоделей;
- хранение информации с использованием специальной онтологической модели и продукций;
- визуальное отображение и редактирование понятий предметной области и их отношений в виде графа (онтологической модели);
- визуальное отображение и редактирование (моделирование) продукций в нотации RVML.

Используя (2), уточним множество предоставляемых сервис-ориентированной системой ресурсов (функций) R^* для набора прикладных сервисов S^*_{USR}:

- формирование модели онтологии на основе автоматизированного анализа концептуальных моделей предметных областей;
- генерация кода БЗ на целевом ЯПБЗ, путем автоматической трансформации модели онтологии;
- генерация кода БЗ на целевом ЯПБЗ на основе прямой автоматической трансформации концептуальных моделей предметных областей.

Таким образом, концептуальная архитектура сервис-ориентированной системы (рисунок 1) позволяет описать ее структуру, включая состав и типы элементов, а также принципиальные особенности функционирования.

Основные типы элементов:

- информационные ресурсы – предназначены для хранения служебной информации, которая используется системными и прикладными компонентами (подсистемами) сервис-ориентированной системы, как для обеспечения собственного функционирования, так и для решения задач автоматизированного формирования БЗ;
- системные сервисы – представляют собой набор всех предлагаемых пользователям сервисов (подсистем), обеспечивающих базовое взаимодействие пользователей с сервис-ориентированной системой и предоставляющие инструментарий для создания прикладных программных компонентов на основе типового;
- прикладные сервисы – представляют собой набор разработанных пользователями программных компонентов, обеспечивающих возможность автоматического синтеза БЗ путем трансформации концептуальных моделей. Программные компоненты создаются на основе типового программного компонента, путем его «клонирования» и настройки (специализации).

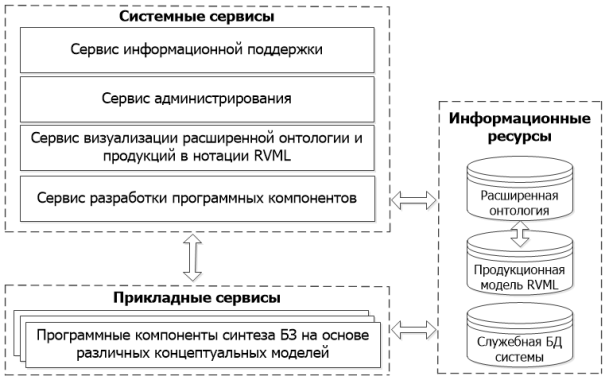

Рисунок 1 – Концептуальная архитектура сервис-ориентированной системы

Для унифицированного хранения и представления знаний, извлеченных из концептуальных моделей, разработана специальная онтологическая модель. Разработанная модель позволяет абстрагироваться от особенностей описания знаний в различных ЯПБЗ, используемых при реализации БЗ (например, CLIPS, Jess, Drools, RuleML, OWL, SWRL и др.), и позволяет хранить знания в собственном независимом формате.

Подробное описание метамодели онтологии приводится в работе [Дородных и др., 2015].

Для поддержки возможности работы со знаниями в виде правил разработана специальная продукционная модель. Главной особенностью продукционной модели является использование специализированной графической нотации Rule Visual Modeling Language (RVML) [Грищенко и др., 2013].

В зависимости от решаемых задач и типов входных данных (исходных концептуальных моделей или модели онтологии) на основе типового программного компонента могут быть созданы разные виды программных компонентов, приведем их классификацию по различным основаниям.

По отношению к информационным ресурсам сервис-ориентированной системы:

- автономные – обеспечивают выполнение операций трансформации и кодогенерации, без использования модели онтологии и продукций (трансформация происходит без использования информационных ресурсов сервис-ориентированной системы);
- интегрируемые – обеспечивают выполнение операций с использованием информационных ресурсов сервис-ориентированной системы.

Интегрируемые программные компоненты имеют доступ к системным сервисам (средствам) визуального проектирования (редактирования) БЗ. Данными средствами являются два специализированных редактора: редактор онтологической модели (визуальное представление и редактирование онтологии) и редактор продукционной модели (визуальное представление и редактирование продукций в нотации RVML).

По типу выполняемых преобразований:

- интегрируемые компоненты анализа – обеспечивают преобразование концептуальных моделей в онтологическую модель и продукций (M2M-преобразование);
- интегрируемые компоненты кодогенерации – обеспечивают преобразование элементов онтологической модели в код БЗ на целевом ЯПБЗ (M2C-преобразование);
- автономные компоненты кодогенерации – обеспечивают преобразование концептуальных моделей в код БЗ на целевом ЯПБЗ (M2C-преобразование).

По характеру использования:

- внешние – фактически размещенные на других серверах, в программной системе непосредственно размещается только описание программных интерфейсов, для реализации используя структурный шаблон «адаптер»;

- внутренние – размещенные непосредственно в составе сервис-ориентированной системы.

Таким образом, программный компонент представляет собой простейший транслятор, который переводит входные концептуальные модели, представленные на одном языке, в выходные БЗ на другом языке.

Одним из важнейших элементов данной технологии является – модель типового программного компонента, на основе которой создается соответствующий программный компонент.

1.2. Модель типового программного компонента

Для повышения эффективности разработки программных компонентов предлагается использовать специальную модель типового программного компонента:

$$M_{TPC} = \langle M_T, A_{IN}, CG_{OUT} \rangle, \quad (5)$$

где M_T – модель трансляции; A_{IN} – анализатор (парсер) входных моделей (исходных концептуальных моделей или расширенной онтологии); CG_{OUT} – генератор кода выходных моделей (расширенной онтологии или БЗ на целевом ЯПБЗ).

Архитектура типового программного компонента может быть представлена на рисунке 2.

Рисунок 2 – Архитектура типового программного компонента

В процессе создания модели типового программного компонента M_{TPC} пользователю необходимо сформировать модель трансляции M_T, на основе которой осуществляется преобразование исходных концептуальных моделей в целевые БЗ.

1.3. Трансформация моделей

Используя (5), определим модель трансляции M_T:

$$M_T = \langle MM_{CM}, MM_{KB}, T \rangle, \quad (6)$$

где MM_{CM} – метамодель исходной (входной) концептуальной модели; MM_{KB} – метамодель целевой (выходной) модели представления знаний (БЗ); T – оператор преобразования моделей.

При этом MM_{ONT} – метамодель онтологии, может выступать в качестве как исходной, так и целевой метамодели в процессе построения M_T.

Используя (6), подробнее опишем элементы модели трансляции M_T:

$$MM_{CM} = \langle E_{CM}, R_{CM} \rangle, \quad (7)$$

где E_{CM} – множество элементов (сущностей) метамодели исходной концептуальной модели; R_{CM} – множество отношений между элементами метамодели исходной концептуальной модели.

$$E_{CM} = \{e_1^{cm}, ..., e_n^{cm}\}, e_i^{cm} = \langle id_i, name_i \rangle, i \in \overline{1,n}, \quad (8)$$

где id_i – идентификатор i элемента; $name_i$ – наименование i элемента или ссылка на другой элемент $e_{link}^{cm} = \langle id_{link}, name_{link} \rangle$, где id_{link} – идентификатор элемента ссылки; $name_{link}$ – наименование элемента ссылки.

$$R_{CM} = \{r_1^{cm}, ..., r_m^{cm}\}, r_j^{cm} = \langle r_{LHS_j}^{cm}, r_{RHS_j}^{cm} \rangle, j \in \overline{1,m}, \quad (9)$$

где $r_{LHS_j}^{cm}$ – левая часть связи; $r_{RHS_j}^{cm}$ – правая часть связи. В свою очередь $r_{LHS_j}^{cm} = e^{cm}$ и $r_{RHS_j}^{cm} = e^{cm}$, где e^{cm} – ссылка на элемент.

$$MM_{KB} = \langle E_{KB}, R_{KB} \rangle, \quad (10)$$

где E_{KB} – множество элементов (сущностей) метамодели целевой БЗ; R_{KB} – множество отношений между элементами метамодели целевой БЗ.

Описание элементов MM_{KB} идентично описанию элементов MM_{CM}.

Основным элементом модели трансляции M_T является оператор преобразования T. Таким образом, формализовать постановку третей основной задачи можно следующим образом. Необходимо определить оператор трансформации концептуальной модели T:

$$T : CM \to KB, \quad (11)$$

где CM – исходная концептуальная модель; KB – целевая БЗ.

При этом $KB \sim M_{KRL}$, где M_{KRL} – целевая модель на ЯПЗ (*KRL*).

В соответствии с видами синтаксиса (абстрактного и конкретного) концептуальных языков моделирования и ЯПБЗ согласно [Greenfield et al., 2004] уточним (11):

$$T = \langle T_{CM-KB}, T_{CM-ONT}, T_{ONT-KB} \rangle,$$
$$T_{CM-KB} : M_{XML} \Leftrightarrow MM_{CM} \to MM_{KB} \Leftrightarrow Code_{KRL},$$
$$T_{CM-ONT} : M_{XML} \Leftrightarrow MM_{CM} \to MM_{ONT} \Leftrightarrow ONT,$$
$$T_{ONT-KB} : ONT \Leftrightarrow MM_{ONT} \to MM_{KB} \Leftrightarrow Code_{KRL},$$

где T_{CM-KB} – оператор преобразования исходной концептуальной модели в код БЗ на целевом ЯПБЗ; T_{CM-ONT} – оператор преобразования исходной концептуальной модели в расширенную онтологию; T_{ONT-KB} – оператор преобразования расширенной онтологии в код БЗ на целевом ЯПБЗ; M_{XML} – исходная концептуальная модель, представленная в формате XML; $Code_{KRL}$ – код БЗ, представленный на целевом ЯПБЗ; ONT – онтологическая модель;

Оператор преобразования моделей T состоит из трех подоператоров в соответствии с приведенной выше классификацией программных компонентов (по типу выполняемых операций).

Уточним подоператор преобразования T_{CM-KB}: процесс трансформации M_{XML} в $Code_{KRL}$ на целевом ЯПБЗ осуществляется путем установления соответствий (определение правил трансформации) между абстрактными элементами исходной метамодели MM_{CM} и целевой метамодели MM_{KB}.

Введем оператор R_T – набор установленных правил соответствия элементов метамоделей (правила трансформации):

$$R_T = (r_1, r_2 ... r_n) : MM_{CM} \to MM_{KB}, \quad (12)$$

где r_i – правило трансформации (продукция). При этом:

$$r_i = \langle e_i^{in}, e_i^{out}, p_i \rangle, i \in \overline{1,n}, \quad (13)$$

где e_i^{in} – исходный элемент метамодели MM_{CM} (левая часть правила); e_i^{out} – целевой элемент метамодели MM_{KB} (правая часть правила); p_i – приоритет выполнения правила, определяющий последовательность выполнения правил, $p_i \in \overline{1,k}$.

Для представления и хранения модели трансляции M_T разработан специальный предметно-ориентированный (декларативный) язык (DSL) – Язык Представления Модели Трансляции (ЯПМТ).

Грамматика разработанного ЯПМТ принадлежат к классу контекстно-свободных грамматик (LL(1)). Конструкции ЯПМТ позволяют в декларативном виде описывать элементы модели трансляции (в основном, установление правил соответствия элементов метамоделей).

2. Метод трансформации концептуальных моделей в БЗ

Метод трансформации представляет собой систематизированную совокупность действий, которые нацелены на решение задачи автоматизированного формирования кода БЗ на целевом ЯПБЗ, путем трансформации исходных концептуальных моделей.

В процессе синтеза БЗ может быть использована модель онтологии, которая представляется либо как исходная модель (концептуальная модель), на основе которой синтезируется код БЗ, либо как целевая модель (БЗ), полученная путем автоматизированного анализа исходных концептуальных моделей.

Метод определяет горизонтальную, экзогенную, одностороннюю трансформацию [Mens et al., 2006], т.е. преобразование исходной и целевой моделей одного уровня иерархии, но описанных на различных языках моделирования (разные концептуальные языки моделирования и ЯПБЗ).

Основные принципы разрабатываемого метода (подхода) трансформации:

- возможность использования двухступенчатой трансформации (на первом этапе M2M-преобразование, затем M2C-преобразование), что повышает технологичность процесса разработки БЗ, за счет использования онтологии;
- использование продукционной модели, представленной в нотации RVML и обеспечивающей унифицированное описание правил, для дополнения (уточнения) полученных знаний.

В общем виде метод (алгоритм) трансформации исходных концептуальных моделей в код БЗ на целевом ЯПБЗ может быть представлен в виде последовательности действий:

На этапе 1 средствами внешних программ пользователь строит концептуальную модель предметной области, которая представляется в формате XML. Этот формат является универсальным и наиболее распространенным способом интеграции программных систем и обеспечения обмена информацией между приложениями.

На этапе 2 в процессе автоматизированного

анализа XML-структуры концептуальной модели выделяются понятия предметной области и их отношения. Анализ происходит на основе разработанной модели трансляции M_T (правил трансформации).

Далее (этап 3) на основе извлеченных понятий и их связей формируется онтологическая модель, как универсальное представление знаний, независящее от исходной концептуальной модели или ЯПБЗ.

На этапе 4 при помощи специальной графической нотации RVML предоставляется возможность визуального отображения и модификации (проверки) полученных зависимостей (продукций).

На этапе 5 происходит автоматическая генерация кода БЗ в формате CLIPS или OWL на основе онтологической модели.

Следует отметить, что этапы 3 и 4 могут отсутствовать, т.к. преобразование исходной концептуальной модели может происходить напрямую в код БЗ, без использования онтологической модели.

Заключение

Решение задач связанных с получением, структурированием и формализацией знаний позволяет повысить эффективность процесса разработки БЗ ИС. В настоящей работе предлагается автоматизировать программную реализацию (синтез кода) БЗ на основе анализа концептуальных моделей.

При решении задач исследования был проведен комплексный анализ области автоматизированного получения (приобретения) знаний, методов и средств трансформации моделей, подходов к разработке программных систем, а также различных стандартов (языков) концептуального, когнитивного и онтологического моделирования. В качестве источников концептуальных моделей предлагается использовать модели представленные на XML. В качестве целевого ЯПБЗ использованы CLIPS и OWL.

На основе разработанных моделей и методов был реализован исследовательский прототип сервис-ориентированной системы для разработки программных компонентов, синтеза БЗ. Одной из возможностей данной системы является обеспечение совместной, распределенной, коллективной работы специалистов-предметников (инженеров по знаниям, программистов и др.) в процессе формирования БЗ для ИС.

Безусловно, данный подход (технология) не позволяет исключить ошибки обусловленные неточностью или неполнотой анализируемых концептуальных моделей, однако, автоматическая генерация декларативного кода на основе этих моделей позволяет использовать принцип быстрого прототипирования при создании БЗ с последующей их проверкой и тестированием (дополнением) в сторонних средствах. По результатам проверки возможно внесение необходимых изменений в модели (расширенная онтология или продукционная модель RVML) и повторная генерация кода БЗ.

Результаты, представленные в статье, получены при частичной финансовой поддержке РФФИ, проекты 15-37-20655, 15-07-03088

Библиографический список

[Гаврилова и др., 2000] Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. СПб.: Питер, 2000. – 384 с.

[Грищенко и др., 2013] Грищенко М.А., Юрин А.Ю., Павлов А.И. Разработка экспертных систем на основе трансформации информационных моделей предметной области // Программные продукты и системы. – 2013. – №3. – С. 143-147.

[Дородных и др., 2015] Дородных Н.О., Юрин А.Ю. Использование диаграмм классов UML для формирования продукционных баз знаний // Программная инженерия. – 2015. – №4. – С. 3-9.

[Частиков и др., 2003] Частиков А.П., Гаврилова Т.А., Белов Д.Л. Разработка экспертных систем. Среда CLIPS. СПб.: БХВ-Петербург, 2003. – 608 с.

[Booch et al., 2005] Booch, G., Rumbaugh, J., Jacobson, I. The Unified Modeling Language User Guide, 2nd Edition. Addison-Wesley, New York, 2005, 496 p.

[Grau et al., 2008] Grau B.C., Horrocks I., Motik B., Parsia B., Patel-Schneider P., Sattler U. OWL 2: The next step for OWL // Web Semantics: Science, Services and Agents on the World Wide Web. – 2008, 6(4), 309-322.

[Greenfield et al., 2004] Greenfield J., Short K., Cook S., Kent S., Crupi J. Software factories: assembling applications with patterns, models, frameworks, and tools. Wiley Pub., 2004, 696 p.

[Mens et al., 2006] Mens T., Gorp Van P. A Taxonomy of Model Transformations. Electronic Notes in Theoretical Computer Science. - 2006, №152, 125-142.

[XMI] Документация спецификации XML Metadata Interchange (XMI). 2015. URL: http://www.omg.org/spec/XMI (дата обращения: 10.11.2015).

[XTM] Документация спецификации XML Topic Maps (XTM). 2015. URL: http://www.topicmaps.org/xtm (дата обращения: 10.11.2015).

AN APPROACH FOR DESIGN OF KNOWLEDGE BASES ON THE BASIS OF COMPUTER-AIDED TRANSFORMATION OF CONCEPTUAL MODELS

Dorodnykh N.O., Yurin A.Yu.

Matrosov Institute for System Dynamics and Control Theory, Siberian Branch of the Russian Academy of Sciences (IDSTU SB RAS), Irkutsk, Russia

tualatin32@mail.ru

iskander@icc.ru

The paper describes the concept of the approach for creation of software components designed for development of knowledge bases (OWL and CLIPS) on the basis of the transformation of conceptual models represented in XML. The architecture of the service-oriented software implementing this approach is proposed. The model of a typal software component (including a translation model) and the algorithm for transformation of conceptual models to the program codes with the opportunity to clarify of logical rules in the RVML notation are described.