ИНТЕЛЛЕКТУАЛИЗАЦИЯ ЭКСПЕРТНЫХ СИСТЕМ С ПОМОЩЬЮ ОНТОЛОГИЙ

Глибовец Н.Н., Красиков Д.С.

***Аннотация**: В данной работе разработана методология построения онтологических экспертных систем с использованием правил вывода продукционного типа. В прикладной части описана реализация экспертной системы с использованием организационной онтологии.*

***Keywords**: база знаний, семантическая сеть, фреймворк, экспертные системы (ЭС), онтологические ЭС (ОЭС).*

***ACM Classification Keywords**: D.2.2 Design*

Введение

Одним из современных подходов совершенствования экспертных систем (ЭС) является использование онтологий [Gruber, 1993]. Явная спецификация предметной области храниться в виде иерархии понятий и связей между ними в базе знаний ЭС совместно с правилами вывода, например, в формате OWL [OWL] или CycL [OpenCyc]. Заданная иерархия объектов используется во время общения экспертов и инженеров знаний, построения правил логического вывода, во время проверки конфликтов в базе знаний.

В данной работе рассмотрим разработанную методологию построения онтологических ЭС (ОЭС) с использованием правил вывода продукционного типа. В прикладной части опишем реализацию ЭС для торговой организации с использованием организационной онтологии.

В работах [Cheng, 2008, Trausan-Matu, 2008, ZONG-YONG, 2007] описано некоторые подобные предложения объединения онтологий и продукционных правил. Но в них невозможно устранение противоречий в онтологии во время поддержки ЭС. Кроме того в них используются несовершенные форматы сохранения правил.

В работе [PARK, 2008] предложена концепция использования правил вывода продукционного типа для построения интеллектуальных систем. В описанных тут исследованиях не используют всех возможностей продукционных систем (императивных команд и команд, которые имеют возможность изменять структуру содержания онтологии). В подобных работах не предлагается ни методологии, ни фреймворка построения ОЭС в общем случае.

1. Методология построения ОЭС

Методологию построения (ОЭС) можно отобразить в виде следующего процесса.

1. Создание онтологии предметной области [MARK, 1993]

1.1. *Накопление знаний о предметной области.* Проводится экспертиза соответствующих информационных ресурсов, при этом формально определяются основные термины которыми будет описана выбранная предметная область. Эти определения должны быть собраны с учетом возможности их представления на общем языке, выбранном для описания онтологии.

1.2. *Формирование онтологии*. Для этого разрабатываются полную понятийную структуру предметной области. Вероятно, это потребует распознавания главных, ключевых понятий предметной области и их свойств, определение связей между понятиями, создание абстрактных понятий, выделение понятий, содержащих экземпляры, а также, возможно, привлечение вспомогательных онтологий.

1.3. *Расширение и конкретизация онтологии*. Понятие связи, атрибуты, экземпляры, аксиомы добавляются тех пор, пока уровень детализации обеспечит удовлетворение целей онтологии.

1.4. *Проверка выполненной работы*. На этом этапе устраняются синтаксические, логические и семантические несогласованности элементов онтологии и происходит проверка достоверности информации. Это можно выполнить с помощью машин вывода [Pellet].

2. Внедрение онтологии путем интеграции онтологических объектов с объектами и правилами вывода в ЭС

2.1. Классы и экземпляры онтологии переносятся в среду ЭС. Как правило, они отображаются на классы и экземпляры языка программирования, на котором реализуется ЭС

2.2. Аксиомы, построенные на этапе создания онтологии, отражаются частично в структуре классов среды программирования ЭС, а частично в правила ЭС. После выполнения каждого правила, которое изменяет, удаляет или добавляет факты, в онтологии происходит контроль противоречий, что будет защищать ее от вхождения в противоречивое состояние (рис.1).

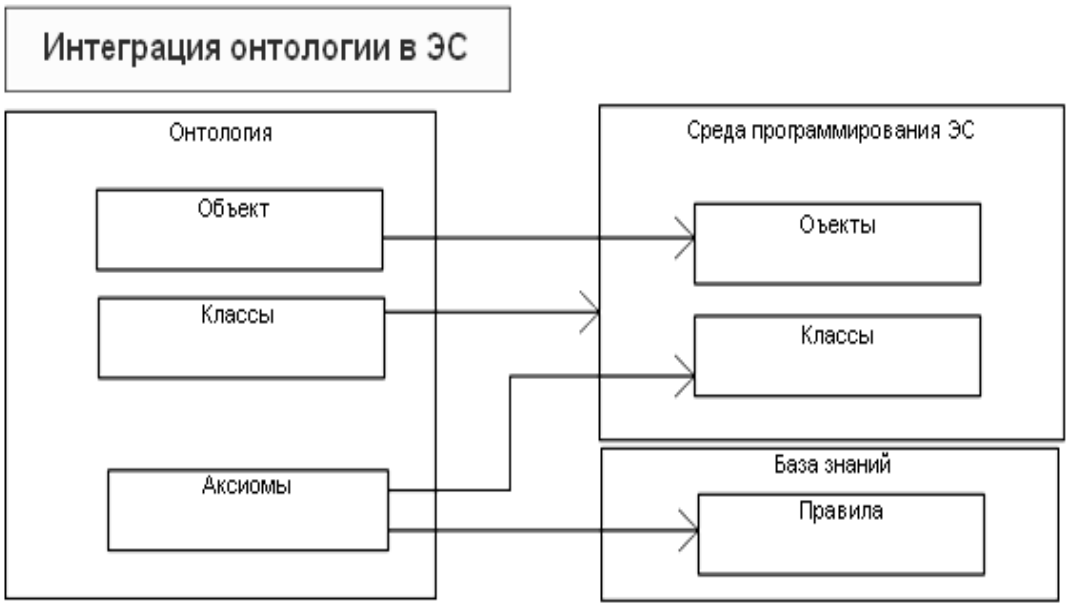

Рис.1. Интеграция онтологии в ЭС.

3. **Согласование**

3.1. Во время выполнения ЭС, перед каждой попыткой изменить базу знаний, будет консультироваться с метаданными, которые хранятся в онтологии, на непротиворечивость. В случае выявления конфликтов, изменения не происходят.

Предлагаемая архитектура ОЭС приведена на рис. 2.

Использование указанной методологии позволяет преодолеть традиционные недостатки ЭС.

Наличие стандартизированного словаря концептов предметной области упрощает общение между экспертом и инженером знаний при проектировании базы знаний. Использование онтологии делает возможным представления родовых / структурных понятий в продукционных системах представления знаний. Онтология как концептуальная схема предметной области поможет глубже анализировать поставленные задачи, принимать более точные решения, а иногда отвечать на принципиально новые вопросы. Существенно упрощается интеграция различных ЭС. Теперь для интеграции достаточно создать

соответствие между концептами двух онтологий, т.е. нет необходимости искать и выделять концепты продукционных правил. Кроме того, если онтология задается одним из общепринятых языков описания, то соответствие можно задавать с помощью стандартных утилит редактирования онтологий (Protégé, Ontolingua, MetaMaker). Использование онтологии создает дополнительную возможность контроля противоречий для сущностей базы знаний, упрощая контроль конфликтов у базе знаний.

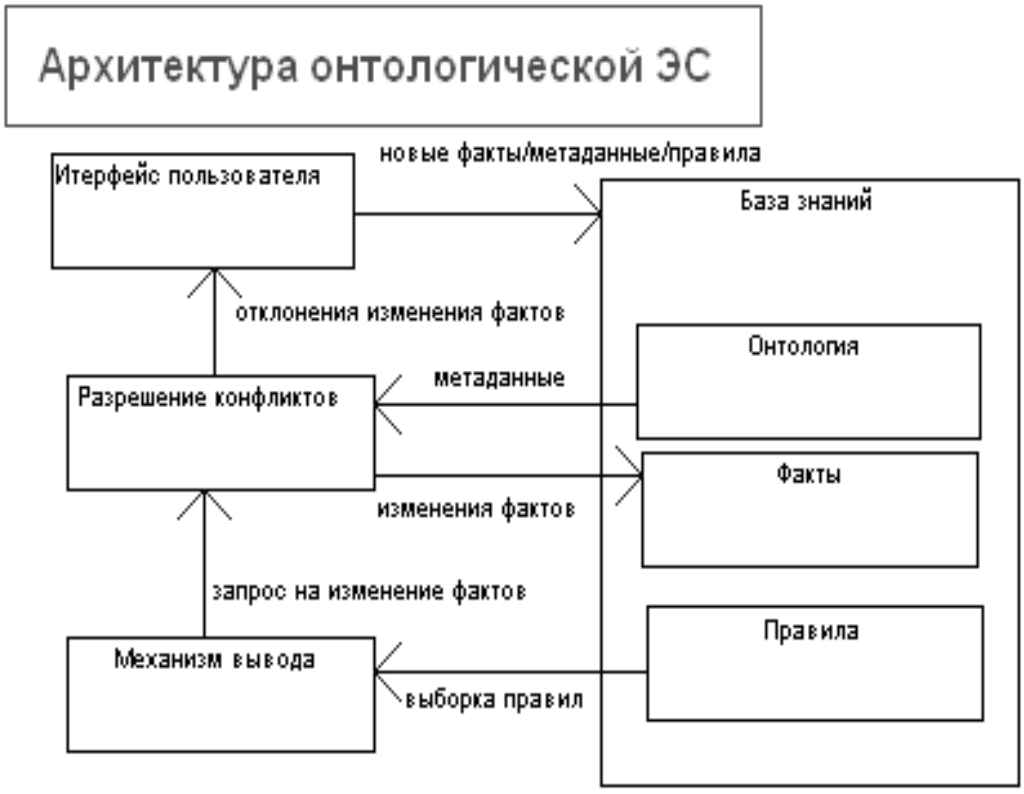

Рис. 2. Структура онтологической ЭС.

При наличии формального описания предметного домена в ЭС упрощается модификация и пополнение БЗ. Так как онтологии неплохо изучены, инженеры знаний могут использовать стандартные языки для описания онтологий (OWL, DC, CycL, KIF) и утилиты (Protégé [Protégé], Pellet) для редактирование базы знаний и логического вывода.

Понятно, что привнесение онтологий в ЭС приводит и к определенным недостаткам. Использование онтологической структуры вызывает потребность в дополнительном анализе и контроле противоречий, что приводит повышению требований к мощности ЭВМ. На этапе проектирования добавляется дополнительный шаг "создание онтологии», что также требует дополнительных затрат.

2. Организационные онтологии и ЭС

В качестве прикладной части данной работы предлагается рассмотреть построение ЭС продукционного типа для торговой организации с использованием организационной онтологии.

Выбор инструментария построения

Мы считаем наиболее эффективной моделью представления знаний гибридную модель – сочетание семантической сети (в данном случае роль семантической сети выполняет онтология) и продукционных правил. Поэтому, выбор фреймворка разработки существенно упростился. Было решено использовать

Drools (JBoss Rules) [Drools] для обработки продукционных правил и язык OWL вместе с библиотекой OWL API для задания обработки онтологии.

Выбор Drools обусловлен хорошей реализацией следующих возможностей. Можно использовать любую операционную систему, поддерживающую Java. Наличие Web-интерфейса упрощает работу пользователя. Поддержка DSL (Domain Specific Language) и деревьев принятия решений в формате Excel-файла (XLS) значительно облегчает создание правил и позволяет редактировать БЗ не только инженерам по разработке знаний, но и обычным пользователям. Проект находился в стадии активной разработки и поддержки, имеет открытый код.

Среди недостатков Drools отметим низкое быстродействие по сравнению с ЭС написанными на C, CLIPS и отсутствие подсистемы объяснений (кроме логирования).

Поэтому мы можем утверждать, что разработанная нами ЭС отвечает основным требованиям к ЭС нового поколения.

Библиотека

Созданная библиотека состоит из двух частей. Первая содержит программный код, который интегрирует онтологию и продукционную систему (считывает с помощью библиотеки OWL API файл онтологии; с помощью Javassist воспроизводит структуру классов в среде Java Virtual Machine; создает соответствующие экземпляры класса, заполняя атрибуты и устанавливая связи между экземплярами). Код второй части, перед каждой сменой в базе данных (изменение, удаление, добавление объектов) считывает метаданные онтологии, проверяет правомерность изменения и производит смену или отклоняет ее.

Обзор предметной области

Рассмотрим построение ЭС для оптимизации розничной торговли некоторой организации. Ключевыми объектами которой являются:

- *Продавец* (Seller) - это организация, которая продает товары в розницу и использует данную ЕС для обслуживания продаж. Главной целью продавца является увеличение прибыли, которая зависит от оборота продаж.
- *Конкурент продавца* (SellerRival) - организация, которая является конкурентом продавца (возможно множество объектов этого класса).
- *Продукт* (Product) - товар, который доступен в продаже для *Покупателя*. Имеет тип, название и количество (единицы или килограммы).
- *Ассортимент* (Assortment) - коллекция товаров и их доступные количества у продавца или в *Конкурента* продавца.
- *Покупатель* (Buyer) - человек, осуществляющий покупки в *Продавцов*, которые оформляются в виде *Заказов*, что в свою очередь формируют *Историю Заказов*.
- *Заказ* (Order) - список товаров и его количество.
- *История Заказов* (OrderHistory) - список *Заказов* с соответствующими датами.

Структуру взаимодействия в организации описывает диаграмма, представленная на рисунке 3.

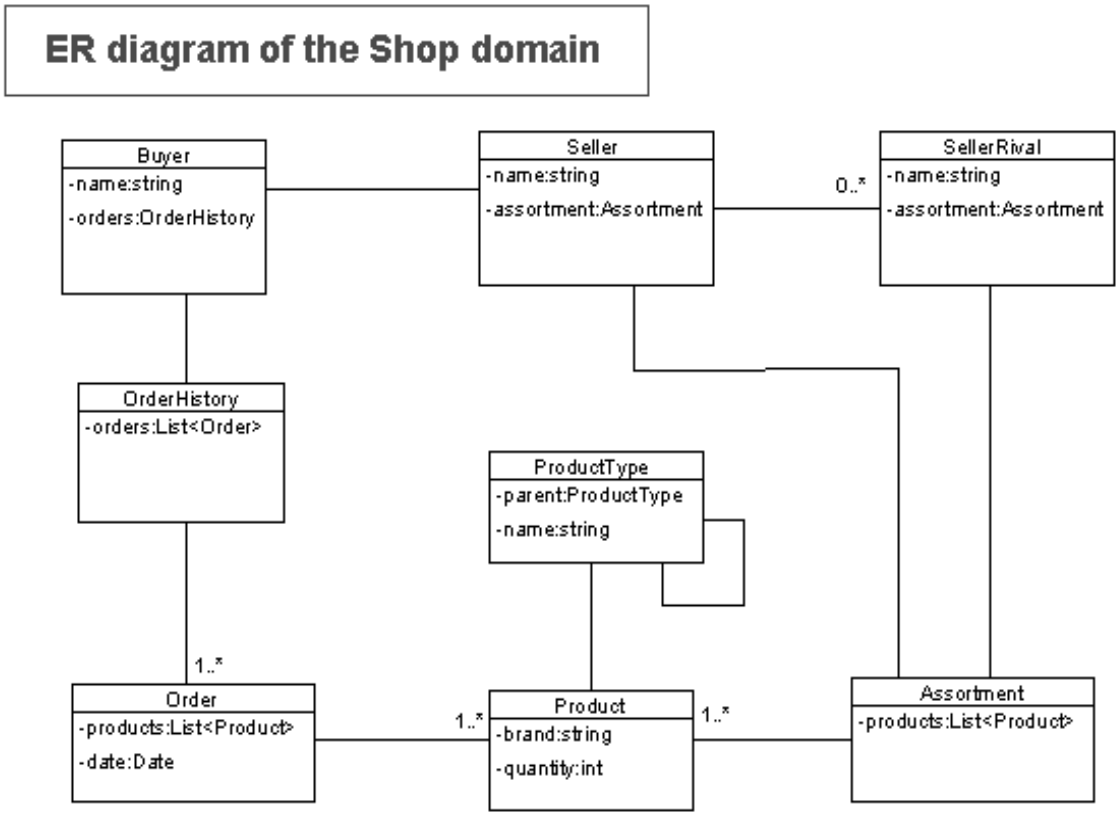

Рис. 3. Структура взаимодействия в организации.

Сценарии использования

Покупатель заходит в магазин, выбирает определенные товары. Его выбор анализируется на основе данного *Заказа*, а также истории *Заказов*. Основываясь на этих данных, система:

- выбирает акционную скидку (пример Drools-правил):

```
rule "Apply 5% Discount"
            agenda-group "checkout"
        dialect "mvel"
        when
                $order : OrderOriginal( grossTotal >= 10 && < 20 )
        then
                $order.discountedTotal = $order.grossTotal * 0.95;
            textArea.append( "discountedTotal total=" +
$order.discountedTotal + "\n" );
        end

        rule "Apply 10% Discount"
            agenda-group "checkout"
        dialect "mvel"
        when
                $order : OrderOriginal( grossTotal >= 20 )
        then
                $order.discountedTotal = $order.grossTotal * 0.90;
            textArea.append( "discountedTotal total=" +
$order.discountedTotal + "\n" );
        end
```

- напоминает о *Товарах* которые он, возможно, забыл купить (пример Drools-правила):

```
        rule "Free Caviar Sample"
            agenda-group "evaluate"
        dialect "mvel"
        when
            $order : OrderOriginal()
```

```
            not ( $p : Product( $type : productType &&
eval($type.get(0). equals("Caviar"))) && PurchaseOriginal( product ==
$p ) )
                $caviar : Product( $type : productType &&
eval($type.get(0)..equals("Caviar")) )
        then
                System.out.println( "Adding free caviar to cart " );
                $caviar.price.set(0,0);
                System.out.println( $caviar.price.get(0) );
                PurchaseOriginal purchase = new PurchaseOriginal($order,
$caviar);
                insert( purchase );
                $order.addItem( purchase );
        end
```

Система дает советы *Продавцу* по закупке, учитывая ассортимент конкурентов продавца и спрос на данную продукцию (*История заказов*), акциях и скидках (что надо делать, чтобы увеличить товарооборот и, соответственно, прибыль организации).

Сценарий построения ЭС

Эксперт (в данном случае это менеджер по продажам) и инженер знаний создают организационную онтологию в Protégé. Помимо классов в OWL будут заданы и экземпляры, которые будут представлять ассортимент магазина. Инженер знаний загружает Eclipse, создает продукционные правила на Drools и графический интерфейс на Java. Фреймворк импортирует онтологию с Protégé в рабочую память продукционной системы Drools и контролирует изменение фактов в базе знаний – наличие противоречий.

Выводы

В данной работе было предложено улучшение классической архитектуры ЭС добавлением онтологии в ее базу знаний. Это придало ЭС следующие преимущества:

• улучшило коммуникацию между экспертами и инженерами созданием на этапе проектирования словаря терминов;

• углубило знания о предметной области до уровня объектов и связей между ними;

• позволило лучше контролировать противоречия, которые появляются во время эксплуатации;

• использование языка описания онтологии OWL улучшило интероперабельность базы знаний и подготовило необходимые технологический базис ЭС для использования в семантическом вебе (Web 3.0);

Кроме создания методологии построения и разработки архитектуры онтологических ЭС, был сделан прототип фреймворка (Drools + Javassist + OWL API). Продемонстрировано его применение. Разработанная библиотека считывает онтологию, заносит ее в базу знаний продукционной системы и отслеживает изменения в базе знаний, чтобы предотвратить создание противоречий.

Библиография

[Gruber, 1993] Gruber T. R. A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220, 1993

[OWL] Web ontology Language (OWL) http://www.w3.org/2007/OWL/wiki/OWL_Working_Group

[OpenCyc] Open Source version of top-level ontology Cyc. http://www.cyc.com/cyc/opencyc/

[Cheng, 2008] Cheng Gang, Du Qingyun. The Design and implementation of ontology and rules based knowledge base for transportation. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. XXXVII. Part B2. Beijing 2008

[Trausan-Matu, 2008] Trausan-Matu, Stefan. A Framework for an Ontology-Based Information System for Competence Management. Economy Informatics, 1-4/2008, p.105

[Park, 2008] Park Han-Saem , Cho Sung-Bae. A Fuzzy Rule-Based System with Ontology for Summarization of Multi-camera Event Sequences. L. Rutkowski et al. (Eds.): ICAISC 2008, LNAI 5097, pp. 850–860, Springer-Verlag Berlin Heidelberg.

[Zong-yong, 2007] Zong-yong Li, Zhi-xu Wang, Ai-hui Zhang, Yong Xu. The Domain Ontology and Domain Rules Based Requirements Model Checking. 2007.

[Mark, 1993] Mark S. Fox, John F. Chionglo, Fadi G. Fadel. A Common-Sense Model Of The Enterprise. In Proceedings of the 2nd Industrial Engineering Research Conference, volume 1, pages 425-429, Norcross GA, USA, 1993.

[Pellet] OWL-reasoner. http://pellet.owldl.com/

[Protégé] OWL-editor. http://protege.stanford.edu/

[Drools] Business Rules Management System. http://www.jboss.org/drools/

Информация о авторах

Глибовец Николай Николаевич *– доктор ф.-м. н., проф., декан Факультета информатики Национального университета "Києво-Могилянська Академія", 04655, Украина, Киев, ул. Сковороды; e-mail: glib@ukma.kiev.ua*

Основные области научных исследований: искусственный интеллект, облачные вычисления, WEB 3.0

Красиков Дмитрий *– разработчик в компании Midnight Coders, Росия, e-mai: dkrasikov@gmail.com*

Основные области научных исследований: облачные вычисления, разработка мобильных приложений, компьютерные алгоритмы