Российский государственный гуманитарный университет

Отделение интеллектуальных систем

Работа в системе коллективного формирования библиотек онтологий ЭЗОП

(Руководство пользователя)

Е. М. Бениаминов

2015

Содержание:

Глава 1. Введение

Это руководство описывает возможности системы **Э**лементов **З**адач и **ОП**ределений (ЭЗОП) для создания онтологий, работы с ними и их использования. Система ЭЗОП представляет собой Web-сервер коллективного конструирования библиотек онтологий пользователями, объединяющимися для разработки разделов библиотеки онтологий на сервере по адресу http://ontoserver.rsuh.ru.

Особенностью системы является предоставление пользователям возможностей открытого шаблонного языка системы для представления онтологий. Открытость языка системы ЭЗОП обеспечивает пользователям возможность самим настроить язык для формирования онтологий раздела библиотеки и запросов к онтологиям, сделав его близким к языку предметной области раздела. При этом новые шаблоны языка вводятся в онтологиях вместе с определениями новых понятий.

Минимальный набор языковых средств системы, действий, связанных с этими средствами и исходная онтология называется *ядром системы*.

В ядро системы входят:

- шаблоны языка ядра системы;
- онтология ядра;
- алгоритмы действий, связанных с шаблонами ядра.

Незарегистрированный пользователь может просматривать в системе открытые разделы библиотеки онтологий, просматривать доступные шаблоны языка разделов, задавать вопросы к онтологиям на шаблонном языке и получать ответы. Кроме того он может просматривать онтологии в графическом виде и получить выбранную онтологию в виде текстового файла на языке OWL - стандартном языке представления онтологий для использования в других компьютерных системах.

Зарегистрированный в системе пользователь, кроме того, может создавать собственные группы пользователей для разработки новых разделов онтологий, участвовать в других группах, создавать и отлаживать черновики новых онтологий и опубликовывать их.

Данное руководство дает общее представление о работе в системе. В следующей главе обсуждается что такое онтология, ее основных характеристики и для чего они нужны. В главе 3 дается краткий обзор интерфейса системы с показом картинок интерфейса.

О компонентах онтологий и о том, как они строятся, говорится в главе 4. Здесь обсуждаются как стандартные составляющие онтологий: классы и элементы классов, так и специфические для принятого в данной системе подхода: функции между классами, операции, которые по исходным классам, элементам и функциям строят новые классы элементы и функции. В системе ЭЗОП принят алгебраический подход к моделированию онтологий. В этой же главе на примерах рассматриваются модульность в построении онтологий и возможности открытого языка определения онтологий и языка запросов к онтологиям.

В главе 5 кратко описывается интерфейс для создания группы пользователей по созданию раздела онтологий. Описывается технология создания онтологий.

Заданию вопросов к онтологиям посвящена глава 6. В этой главе показаны основные шаблоны языка запросов и на примерах показано, как могут строиться тексты вопросов к онтологиям. Здесь рассматриваются как простейшие вопросы для вычисления арифметических или логических выражений, так и тексты вопросов, в которых перед заданием вопроса дополнительные построения, и, даже, SQL-подобные запросы к онтологии, в ответ на которые система выдает таблицы. Эти примеры могут служить образцами вопросов, по которым пользователи могут задавать вопросы к выбранным ими онтологиям.

В главе 7 рассматриваются примеры использования шаблонов настройки языка системы и принципы видимости в онтологии шаблонов, введенных в других онтологиях.

В восьмой главе приводятся примеры онтологий, в некоторой мере демонстрирующие возможности системы.

Глава 9 предназначена для объяснения системных принципов вычислений в онтологиях и перечня основных категорных операций для определения классов, функций и отношений. Система категорных операций существенно расширяет возможности построения онтологий.

Глава 2. Что такое онтология

Онтология представляет собой фиксацию на формальном языке договоренностей группы специалистов о том, что как у них называется, какими свойствами обладает и каким аксиомам удовлетворяет. Онтологии предназначены для многоцелевого, многоразового использования специалистами и компьютерными системами.

На философском уровне онтология является структурным описанием фрагмента мира в виде системы понятий при некотором взгляде на мир. На теоретическом уровне онтология – это теория, заданная системой обозначений (сигнатурой), аксиомами и операциями, связанными правилами логического вывода с фиксированной семантикой. На вычислительном уровне с онтологией связывается система вычисления, как правило, основанная на системе правил переписывания, предназначенная для проверки правильности (синтаксической правильности и непротиворечивости) текста онтологии и построения ответов на вопросы.

Простейшими примерами онтологий являются системы классификации объектов прикладной области. Наиболее конкретными онтологиями являются онтологии задач, представляющие собой модель ситуации, описанной в задаче, и вопрос задачи. Конкретные онтологии и большие онтологии должны строиться из онтологий-фрагментов, описывающих общие ситуации, в виде модулей. Наборы онтологий-модулей организуются в системе в виде разделов библиотеки онтологий для многоразового использования. Каждая онтология разрабатывается в среде некоторой другой онтологии. Онтология-среда предоставляет новой разрабатываемой онтологии средства и понятия, доступные из онтологии-среды. Содержание онтологии существенно зависит от целей ее использования и традиций предметной области, для которой эта онтология создается. Существенным является то, что содержание онтологии должно пониматься по ее тексту специалистами в предметной области и интерпретироваться компьютерными системами при построении ответов на вопросы и решении задач.

Текст онтологии представляется в виде последовательности предложений. Каждое предложение заканчивается точкой или вопросительным знаком (для вопросов). Текст предложения строится из композиции шаблонов, которые заданы в языке ядра системы или определены пользователем в онтологии с помощью специальных шаблонов для введения новых шаблонов или изменения старых. Композиция шаблонов - это подстановка в шаблон вместо переменных шаблонных выражений соответствующих типам этих переменных.

Комментарии в тексте онтологии начинается с /* и заканчиваются */, либо начинается с %% до конца строки.

По тексту онтологии (комментарии игнорируются) система строит внутреннее представление онтологии, которое используется для проверки правильности построения текста, ответов на вопросы к онтологии и компьютерного использования в других системах.

Внутреннее представление онтологии может быть выведено в разных форматах, в том числе в виде OWL-файлов, для межмашинного обмена знаниями и в графическом виде для цельного восприятия онтологии пользователями.

Глава 3. Элементы интерфейса системы

Главная страница системы находится по адресу http://ontoserver.rsuh.ru и имеет вид:

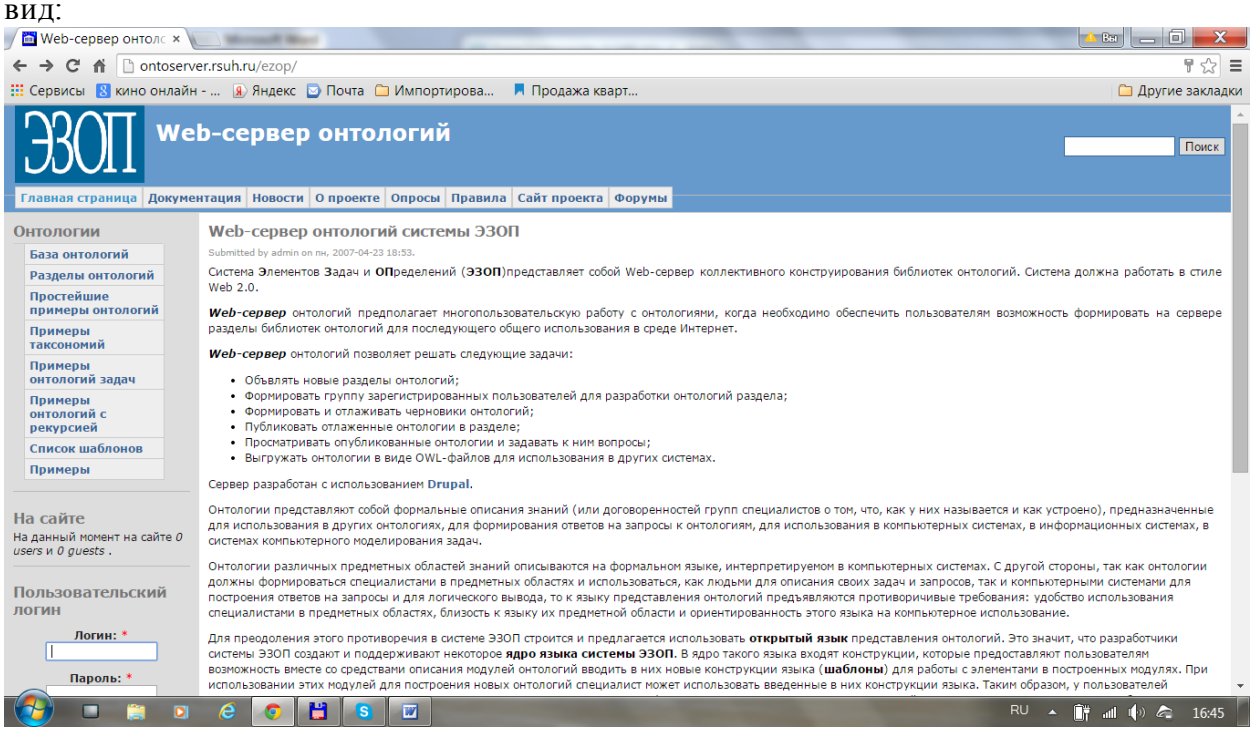

Рис. 2.1. Главная страница системы ЭЗОП

В левом поле экрана системы находится меню «Онтологии», которое позволяет просматривать список всех доступных онтологий, примеры разделов онтологий, демонстрирующие возможности системы, а также список всех шаблонов языка ядра системы.

При выборе пункта «База онтологий» в меню «Онтологии» раскрывается список онтологий, в котором указывается имя онтологии и некоторые ее свойства.

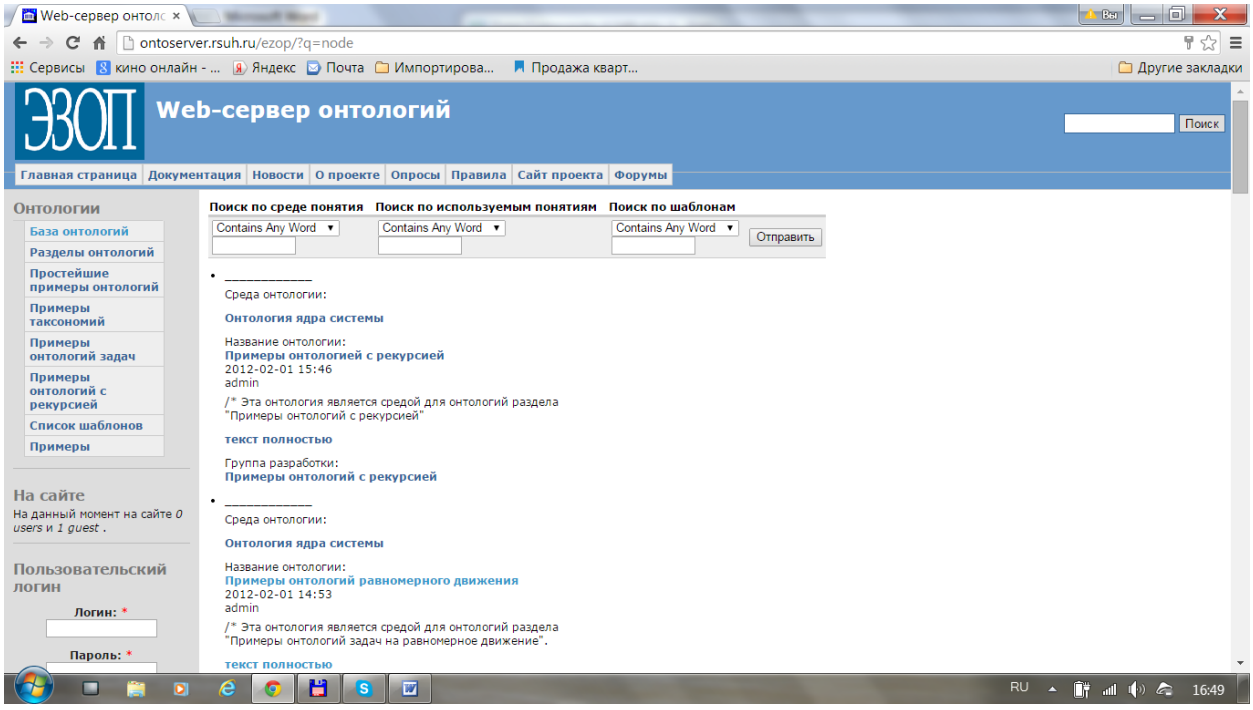

Рис. 2.2. Перечень доступных онтологий

При выборе онтологии из списка раскрывается текст онтологии.

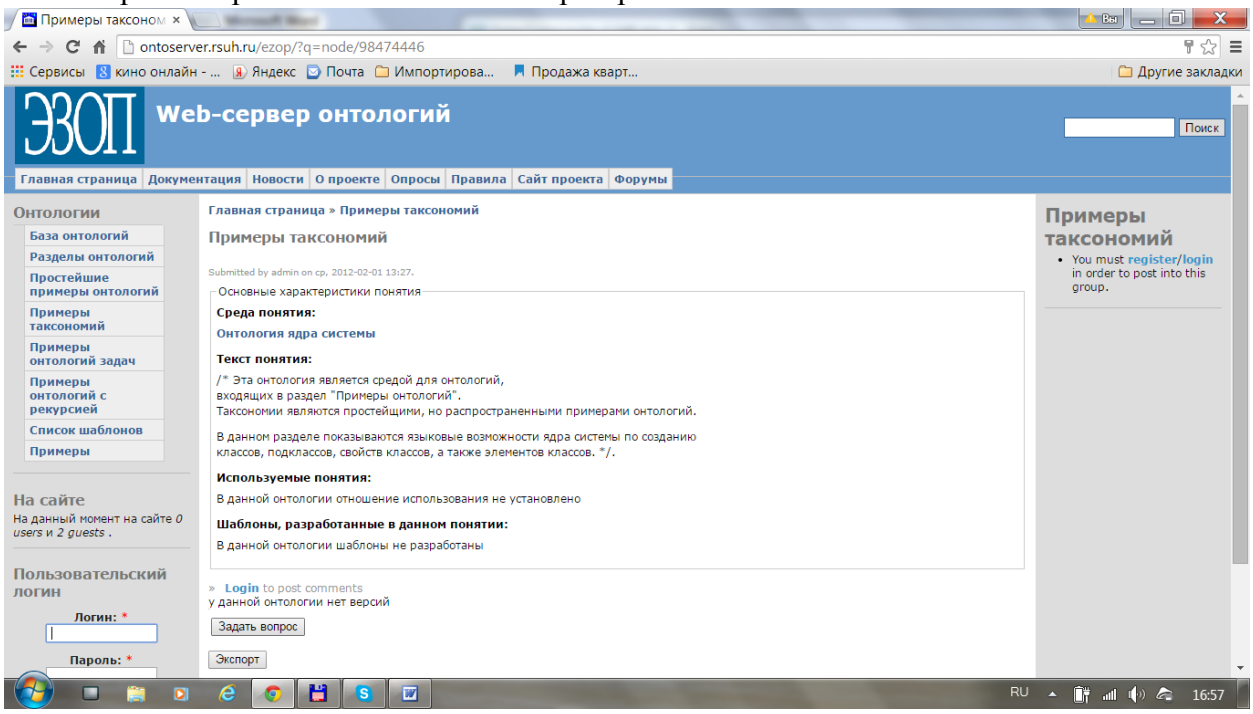

Рис. 2.3. Форма для просмотра текста онтологии

К онтологии можно обращаться с вопросом, если на форме открытой онтологии нажать кнопку «Задать вопрос». При этом открывается форма, в которой можно сформировать текст вопроса, запустить команду «Выполнить» и получить ответ на вопрос.

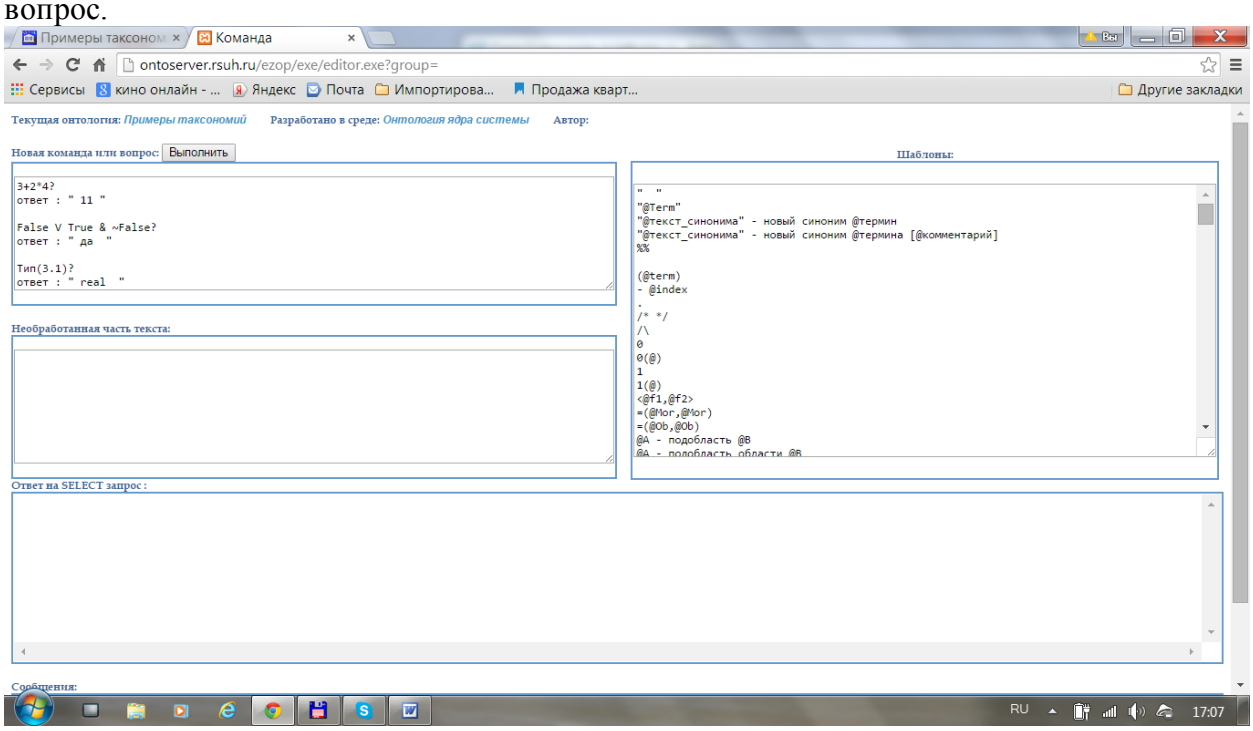

Рис. 2.4. Форма для задания вопросов к онтологии

Подробнее о формировании вопросов написано в соответствующей главе.

Если в меню «Онтологии» главного окна выбрать пункт «Разделы», то откроется список доступных разделов онтологий. Выбирая элементы из этого списка, можно увидеть список онтологий соответствующего раздела.

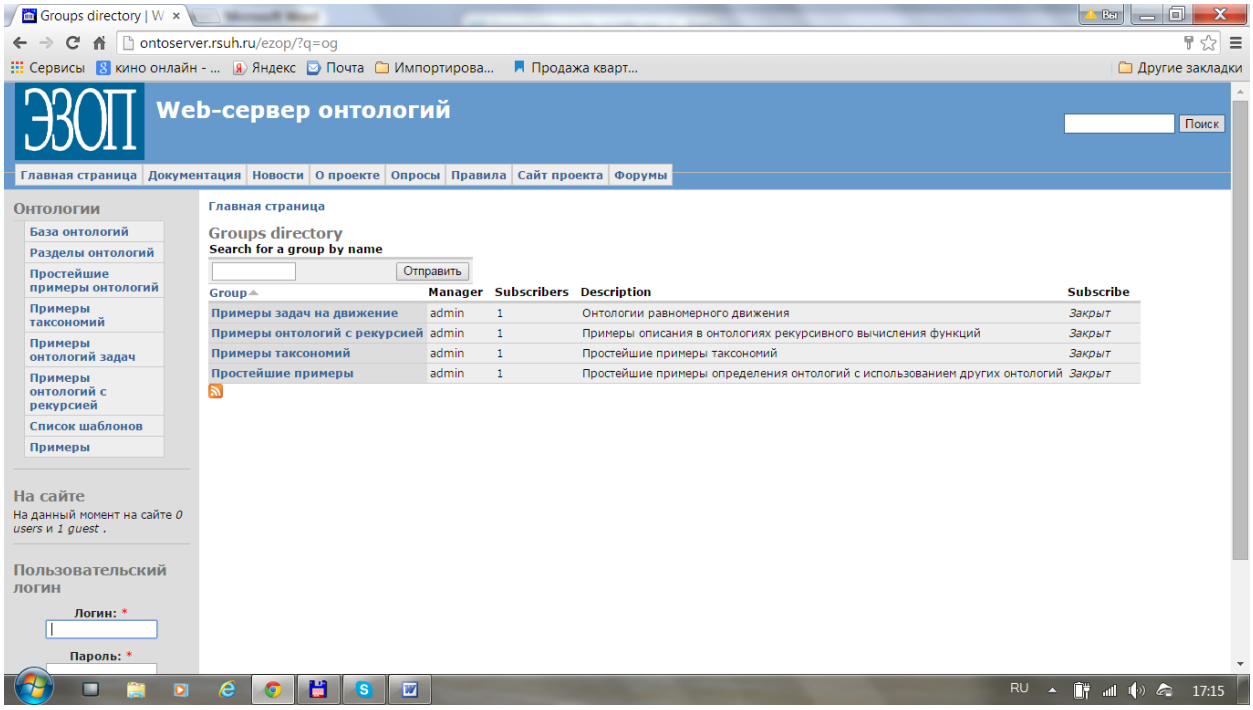

Рис. 2.5. Форма разделов онтологий

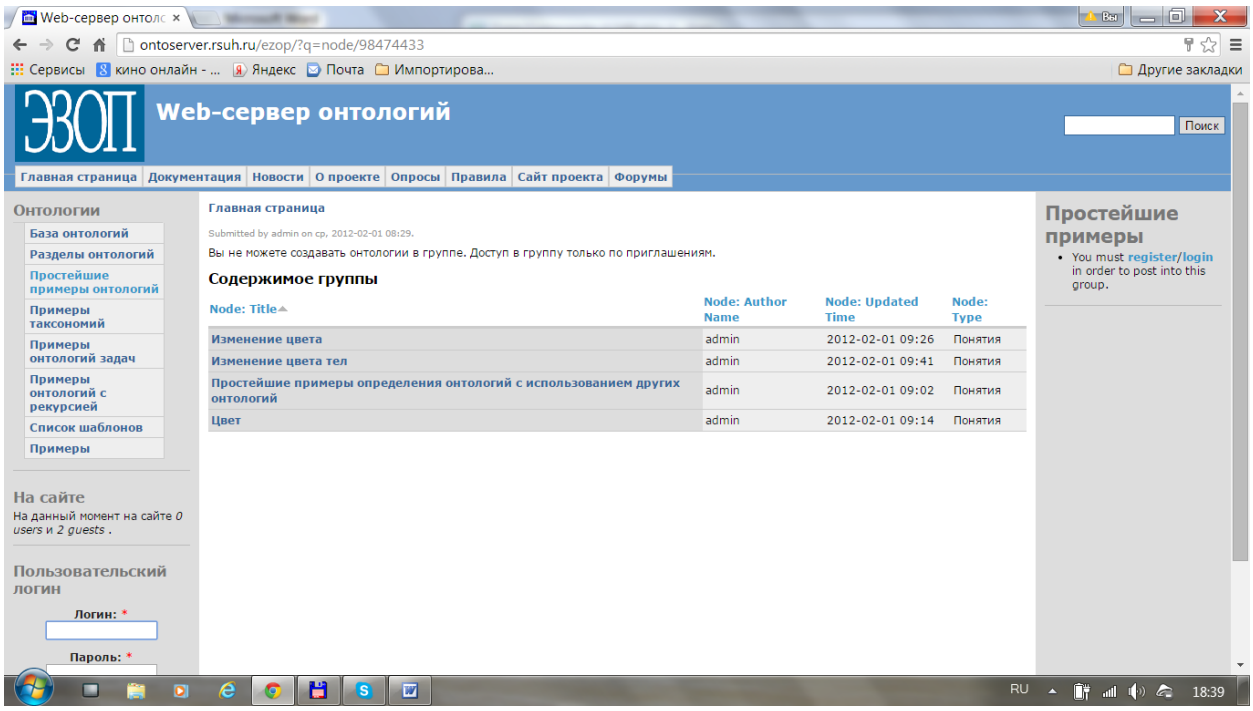

Рис. 2.6. Форма содержания раздела онтологии

Выбирая онтологии в списке онтологий раздела, можно увидеть текст онтологии и задать к ней вопрос.

Если в форме открытой онтологии нажать «Экспорт» и выбрать «Схема онтологии», то открывается форма, в которой дается графическое представление онтологии.

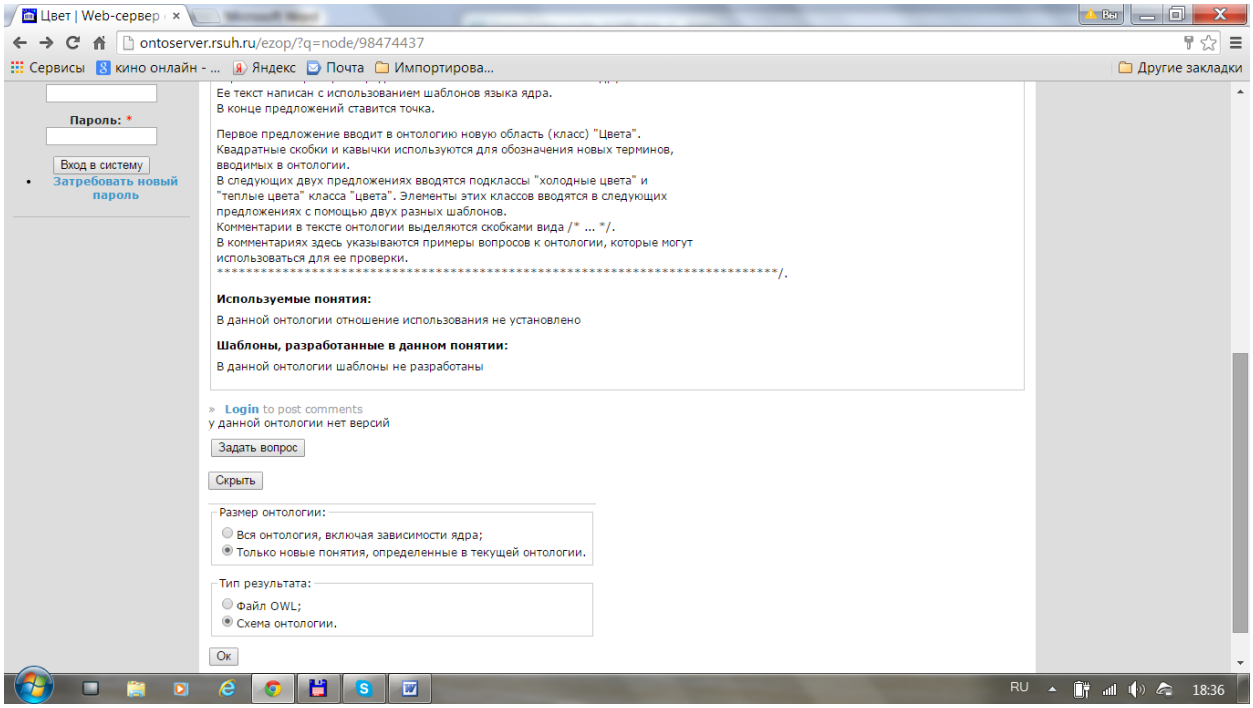

Рис. 2.7. Запуск графического представления онтологии (схемы онтологии)

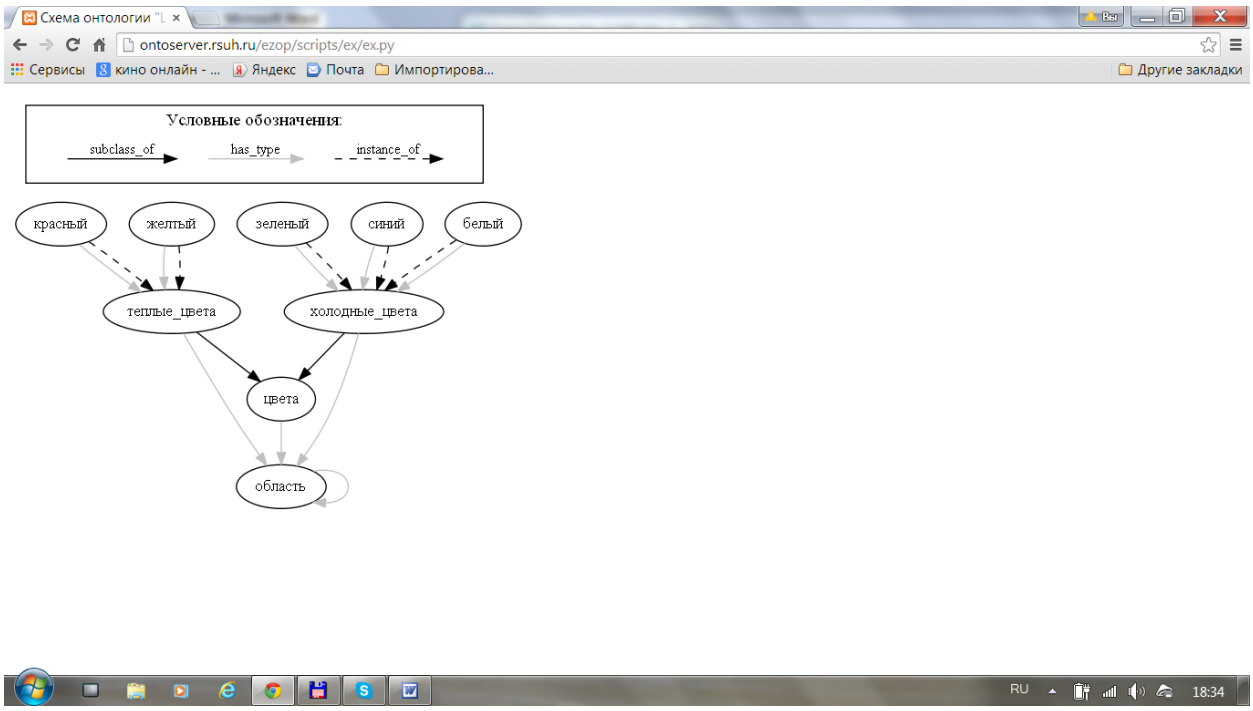

Рис. 2.8. Графическое представление онтологии

Графическое представление онтологии, которое автоматически строится по тексту онтологии, позволяет пользователям получить целостное представление об онтологии.

При выборе в меню «Онтологии» главного окна пункта «Список шаблонов» открывается окно со списком шаблонов ядра системы.

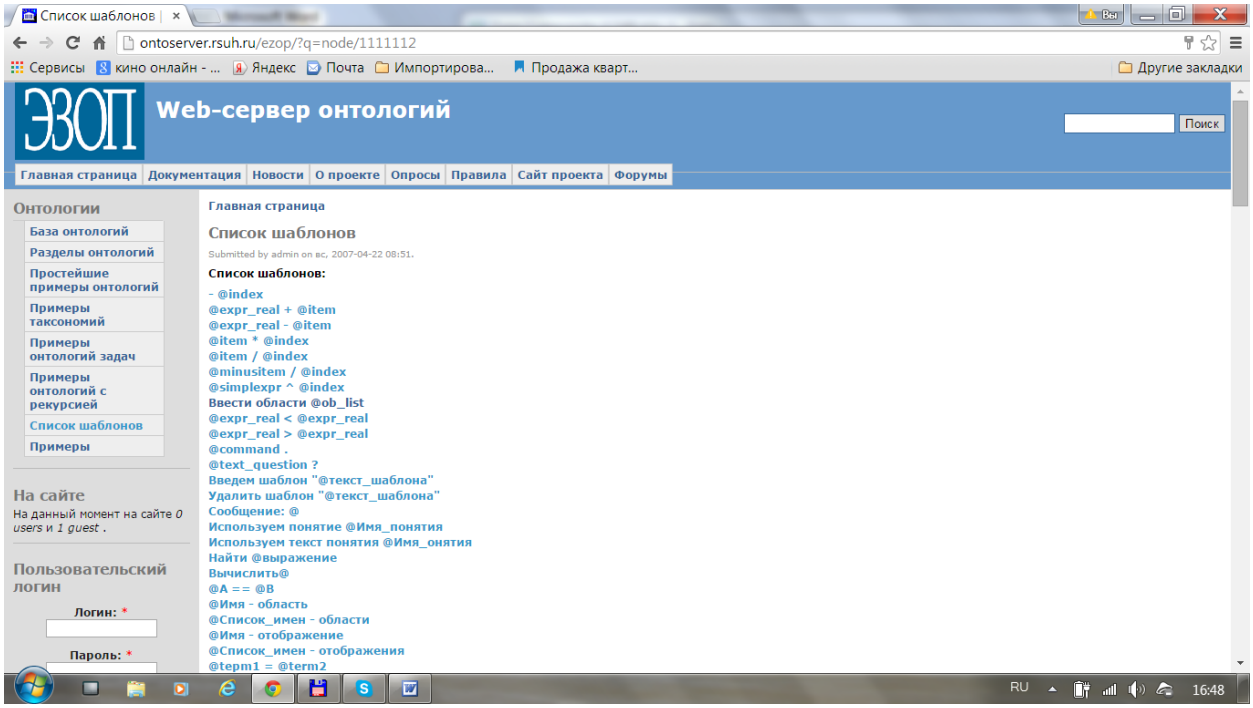

Рис. 2.9. Список шаблонов ядра системы ЭЗОП

Выбирая шаблон в списке шаблонов, можно увидеть описание шаблона.

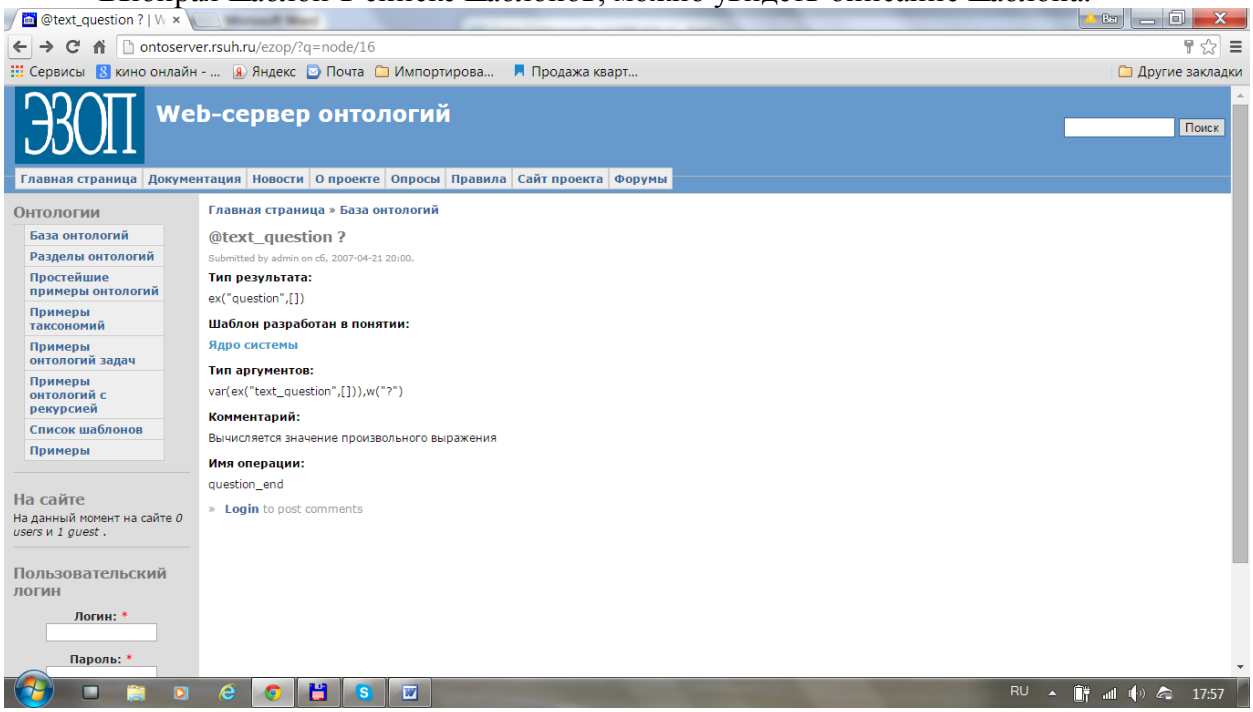

Рис. 2.10. Пример описания шаблона языка ядра системы

Описание примеров онтологий дано в главе 8.

Глава 4. Компоненты онтологий

4.1. Элементы и классы

Основными компонентами онтологий являются элементы и классы. Элементы и классы между собой связываются отношениями принадлежности, подкласс и равенства. Эти отношения удовлетворяют стандартным аксиомам:

- если элемент *a* принадлежит *B*, то *B* обязан быть классом (принадлежать классу *Класс*);
- если элемент *a* принадлежит *B*, и *B* – подкласс класса *C*, то *a* принадлежит *C*;
- если *B* – подкласс класса *C*, и *C* – подкласс класса *B*, то *B* равно *C* (*B*=*C*).

В языке ядра ЭЗОП имеются шаблоны, для ввода новых терминов в качестве имен элементов и классов, ввода отношения принадлежности к классу и ввода отношения класс-подкласс.

Примеры шаблонов для определения новых классов (областей) и элементов имеют вид:

@Новый_термин – область; %% вводится новый класс (область)
@Список_новых_ терминов – области; %% вводится несколько новых классов
@новый_элемент - элемент области @область; %% вводится новый элемент области
@Список_элементов - элементы области @область. %% вводятся несколько элементов.

Слова в шаблонах, начинающиеся с символа @ - это переменные, вместо которых могут быть подставлены новые термины или шаблонные выражения. Типы этих шаблонных выражений должны соответствовать типам переменных, вместо которых подставляются эти выражения.

Примеры шаблонов для задания отношения подкласс класса имеют вид:

@Класс1 < @Класс2; %% первый класс делается подклассом второго
Подобласти @Класса: @Список_классов %% классы из списка делаются подклассами.

Используя приведенные выше шаблоны можно определить простейшие онтологии, задав иерархию классов и выделив некоторые элементы классов.

По умолчанию считается, что разные имена (термины) задают разные элементы. Если система выводит равенство двух разных терминов, то выводится сообщение об ошибке. Однако пользователь, при желании, может задавать синонимы: разные имена для одного и того же элемента, используя специальные шаблоны языка.

В качестве примера простейшей онтологии рассмотрим онтологию «Цвет», находящуюся по адресу http://ontoserver.rsuh.ru/ezop/?q=node/98474437 :

Цвет

Среда понятия:

Простейшие примеры определения онтологий с использованием других онтологий

Текст понятия:

```
[Цвета] - область.
"теплые цвета" < цвета.
"холодные цвета" < цвета.
"Красный", "желтый" - элементы области теплые цвета.
Пусть зеленый,синий, белый - холодные цвета.

/* Примеры вопросов к онтологии:
Элементы области цвета?
Желтый - цвета?
Оранжевый - цвета?
Холодные цвета < цвета?
Равны ли желтый и оранжевый?
*/.
```

Эта онтология разработана в среде онтологии "Простейшие примеры определения онтологий с использованием других онтологий". Ее текст написан с использованием шаблонов языка ядра системы. В конце предложений ставится точка.

Первое предложение вводит в онтологию новую область (класс) "Цвета". Квадратные скобки и кавычки используются для обозначения новых терминов, вводимых в онтологии. В следующих двух предложениях вводятся подклассы "холодные цвета" и "теплые цвета" класса "цвета". Элементы этих классов вводятся в следующих предложениях с помощью двух разных шаблонов. Комментарии в тексте онтологии выделяются скобками вида /* ... */. В комментариях здесь указываются примеры вопросов к онтологии, которые могут использоваться для ее проверки.

4.2. Функции

Функции – это элементы особого класса *Mor* (класса функций). На функциях определены специальные операции:

- если *f* принадлежит классу *Mor*, то терм *dom(f)* правильно построен и принадлежит классу *Ob* (синонимом имени операции *dom* является имя «Область_определения»);
- если *f* принадлежит классу *Mor*, то терм *cod(f)* правильно построен и принадлежит классу *Ob* (синонимом имени операции *cod* является имя «Область_значений»);
- если *f* и *g* принадлежат классу *Mor* и *cod(f)=dom(g)*, то терм *com(g, f)* правильно построен и принадлежит *Mor* (операция *com* называется композицией функций *f* и *g* ; шаблон для инфиксной записи композиции имеет вид @g * @f .

При этом выполняются аксиомы:

- *dom(com(g,f))=dom(f) и cod(com(g,f))=cod(g)*;
- если *cod(f)=dom(g)* и *cod(g)=dom(h)*, то *com(com(h, g), f)=com(h, (com(g, f))*.

%% ассоциативность композиции.

Кроме того,

- если *T* – класс (элемент области *Ob*), то терм *id(T)* правильно построен и принадлежит классу *Mor* (тождественная функция на классе *T*), и *dom(id(T))=T, cod(id(T))=T*.

При этом выполняются аксиомы композиции с тождественным отображением:

- если *f* – элемент *Mor*, то *com(id(codom(f), f)=f* и *com(f, id(dom(f))=f*.

Введенные выше аксиомы означают, что классы и отображения онтологии с введенными операциями образуют алгебраическую структуру, которую в математике называют категорией.

Примерами шаблонов, с помощью которых в онтологии вводятся новые функции являются шаблоны:

@Новый_термин – отображение;	%% вводится новое отображение
@Список_терминов – отображения;	%% вводятся несколько отображений
@f : @A -> @B	%% вводится отображение

Теперь мы введем операции онтологии, связывающие элементы классов и функции:

- если *a* элемент класса *A* и *f* – функция с областью определения *A* и областью значений *B*, то есть *f : A -> B*, то терм *appl(f, a)* правильно построен и задает элемент области *B*.

Шаблон для задания терма *appl(f, a)* имеет вид *@F(@El)*, в который вместо переменной для функции *@F* подставлена функция *f*, а вместо переменной для элемента *@El* – элемент *a*, то есть пишется *f(a)*.

Для применения функции к элементам выполняется аксиома конгруенции:

- если *appl(f, a)* и *appl(f', a')* – правильно построенные выражения и *f=f'*, и *a=a'*, то *appl(f, a)= appl(f', a')*.

В целом в ядре языка ЭЗОП имеются операции (см. категорные операции) и аксиомы, связывающие классы, функции, элементы и подклассы так, что алгебра отологии, построенная из терминов онтологии с помощью этих операций и аксиом является топосом. В частности, в ядре определены операции декартова произведения классов (областей), образ и прообраз подобласти относительно функции и т. д.

В качестве простого примера использования в онтологии функций рассмотрим онтологию «Bool» вида;

```
bool -область.
f,t -элементы области bool.
AND: bool x bool ->bool.
OR : bool x bool -> bool.
NOT : bool ->bool.

/* Таблица действия логических операций */

NOT(f)=t. NOT(t)=f.
AND(f; f)=f. AND(f; t)=f. AND(t; f)=f. AND(t; t)=t.
OR(f; f)=f. OR(f; t)=t. OR(t; f)=t. OR(t; t)=t.
```

В качестве вопросов к этой онтологии можно задавать любую композицию объявленных функций, например «Чему равно OR(AND(NOT(f);t);f)?». Система обрабатывает этот вопрос и выдает «ответ : " t "».

4.3. Операции и термы

В предыдущих разделах введены примеры операций, с помощью которых строятся новые классы и элементы, а также строятся соотношения между ними. Из этих операций и терминов может быть построено сложное выражение – терм.

В общем случае, терм – это правильно построенное выражение из имен операций, терминов и переменных. Термы строятся с помощью шаблонов. По шаблонному выражению программа грамматического анализа строит терм шаблонного выражения, который представляет собой структуру дерево, в корне которого находится имя шаблона или идентификатор шаблона, если он введен пользователем, а ветвями термы, соответствующие подвыражениям, стоящим вместо переменных в шаблоне.

Все термы являются элементами типа Терм.

Задание вопросов к текущей онтологии и построение текущей онтологии происходит на открытом языке шаблонных выражений текущей онтологии.

Программная обработка шаблонного выражения состоит из последовательной обработки шаблонного выражения программами грамматического анализа и вычисления.

Программа грамматического анализа преобразует шаблонные выражения в термы префиксного вида (деревья). В вершине дерева ставится идентификатор шаблона, а ветками являются деревья, соответствующие шаблонным выражениям, поставленным в соответствие переменным шаблона. Далее, построенный терм передается программе вычисления, которая выполняет этот терм в соответствии с действиями, привязанными к

шаблонам. Первым действием является верхний шаблон. Он может управлять выполнением дальнейших действий: снизу-вверх или сверху-вниз, слева-направо или справа-налево или переписываниями веток-аргументов по правилам переписывания. При этом может изменяться внутреннее представление онтологии и множество доступных шаблонов языка текущей онтологии. По ходу выполнения программ грамматического анализа и вычисления программы могут выдавать:

- сообщения об успешном или неуспешном выполнении этапов программы;
- диагностические сообщения с указанием возможных ошибок;
- сообщения о произведенных изменениях в базе данных;
- вопросы пользователю, если требуется дополнительная информация при выполнении действий;
- ответы, если терм шаблонного выражения был вопросом.

Если предложение шаблонного выражения является командой, вызывающей изменения в текущей онтологии (включая комментарии), то это предложение помещается в текст текущей онтологии.

4.4. Модульность

В системе предполагается, что каждая онтология может быть средой для разработки новой онтологии. При этом среда предоставляет новой онтологии, все средства, которые были определены в онтологии-среде или доступны из нее. Бинарное отношение «среда» связывает онтологии в структуру дерева, в корне которого находится онтология «Онтология ядра системы».

Кроме того, при определении новой онтологии или при задании вопроса можно использовать любые существующие онтологии в качестве модулей. Шаблоны языка ядра системы, обеспечивающие эту функцию имеют вид:

```
«Используем понятие @Имя_онтологии»
«Используем текст понятие @Имя_онтологии»
```

Первый шаблон включает во внутреннее представление определяемой онтологии внутреннее представление используемой онтологии. Второй шаблон включает в текст определяемой онтологии текст используемой онтологии, но не показывает его.

В качестве примера рассмотрим онтологии раздела «Простейшие примеры», который находится на странице http://ontoserver.rsuh.ru/ezop/?q=node/98474433 .

Рассмотрим онтологию "цвет":

```
/*****************************************************/
[Цвета] - область.
"теплые цвета" < цвета.
" холодные цвета" < цвета.
"Красный", "желтый" - элементы области теплые цвета.
Пусть зеленый, синий, белый - холодные цвета.

/* Примеры вопросов к понятию:
Элементы области цвета?
Желтый - цвета?
Оранжевый - цвета?
Холодные цвета < цвета?
Равны ли желтый и оранжевый?
```

```
*/.
    /*****************************************************/
```

Эта онтология разработана в среде онтологии "Простейшие примеры определения онтологий с использованием других онтологий".

Ее текст написан с использованием шаблонов языка ядра.

Первое предложение вводит в онтологию новую область (класс) "Цвета". Квадратные скобки и кавычки используются для обозначения новых терминов, вводимых в онтологии. В следующих двух предложениях вводятся подклассы "холодные цвета" и "теплые цвета" класса "цвета". Элементы этих классов вводятся в следующих предложениях с помощью двух разных шаблонов. Комментарии в тексте онтологии выделяются скобками вида /* ... */. В комментариях здесь указываются примеры вопросов к онтологии, которые могут использоваться для ее проверки.

Следующая онтология, которую мы рассмотрим, называется "изменение цвета":

```
    /*****************************************************/
Используем понятие [цвет].
[измененный] - отображение.
Измененный : Цвета -> Цвета.
Измененный (красный) = синий.
Измененный (синий) = зеленый.
Измененный (зеленый) = зеленый.

/* Примеры вопросов к понятию:
Элементы области цвета?
Чему равно измененный(синий)?
Чему равно измененный(желтый)?
Желтый - цвет?
*/.
    /*****************************************************/
```

Эта онтология также разрабатывается в среде "Простейшие примеры определения онтологий с использованием других онтологий" с использованием шаблонов ядра. В первом предложении в разрабатываемую онтологию загружается предыдущая онтология. В следующих предложениях вводится новое отображение из класса "цвета" в этот же класс, и определяются значения введенного отображения на элементах с помощью равенств.

И, наконец, рассмотрим третью онтологию с названием "изменение цвета тел":

```
    /*****************************************************/
"тела" - область.
Пусть [куб], [пирамида],[шар] - тела.
Используем понятие [изменение цвета].
[цвет] - отображение.
цвет: тела -> Цвета.
цвет(куб) = красный.
цвет(пирамида) = синий.
цвет(шар) = красный.

/* Примеры вопросов к понятию:
```

```
Элементы области цвета?
Чему равно измененный(цвет(шар))?
Цвет(пирамида)==измененный(цвет(шар))?
Равны ли цвет(пирамида) и измененный(цвет(шар))?
Равны ли цвет(куб) и измененный(цвет(шар))?
Равны ли измененный(цвет(пирамида)) и
измененный(измененный(цвет(шар)))?
*/.
    /*****************************************************/
```

В первых двух предложениях вводится класс "тела" и некоторые его элементы. Далее, в определяемую онтологию вводится онтология "изменение цвета", вводится функция, задающая для каждого тела его цвет, и равенствами связываются элементы, введенные в текущей онтологии и в предыдущих онтологиях.

Другой способ использования онтологий как модулей – это неоднократное использование внутри данной онтологии другой онтологии с разными значениями параметров. В этом случае приобретают особое значение *параметрические онтологии*.

Рассмотрим в качестве примера онтологию «Равномерное движение»:

Равномерное движение

Среда понятия:

Примеры онтологий равномерного движения

Текст понятия:

```
Путь, скорость, время: real.

Путь= скорость*время.
Время= путь/скорость.
Скорость= путь/время.

Введем шаблон "@Тело движется равномерно"
с переменными: "Тело: new"
и переменной результата " x: команда ";
Пояснения: [Вводится объект @Тело, движущийся равномерно]
Условие применения шаблона:
[]
Действие шаблона:
[x=пустая команда;
тело - объект понятия "равномерное движение". ]
Тип доступа шаблона:[внешний].

Введем шаблон "@Тело равномерно движется со скоростью @V"
с переменными: "Тело: new; V: real выражение"
и переменной результата " x: команда " ;
Пояснения: [Вводится объект @Тело, равномерно движущийся \ \n со
скоростью @V ]
Условие применения шаблона:
[]
Действие шаблона:
[x=пустая команда;
тело - объект понятия "равномерное движение";
```

```
тело's скорость =V. ]
Тип доступа шаблона:[внешний].

/*****************************************************/.
```

В первых четырех строках онтологии вводятся переменные параметры, являющиеся характеристиками равномерного движения, которые связываются уравнениями равномерного движения. Далее вводятся два внешних к этой онтологии шаблона, использование которых в других онтологиях позволяет создавать в них экземпляры объектов равномерного движения, обладающих характеристиками равномерного движения, связанными уравнениями равномерного движения. Эта онтология используется в задачах на равномерное движение. В определении действий введенных шаблонов используется шаблон @имя - объект понятия @понятие. В этом шаблоне в качестве первого аргумента вводится имя для нового объекта, а в качестве второго – имя онтологии в кавычках. Результатом действия выражения «`тело – объект понятия "равномерное движение"`», которое построено по этому шаблону, является создание в текущей онтологии класса с именем онтологии `"равномерное движение"`, и создание (загрузку в текущую онтологию) экземпляра объекта типа `"равномерное движение"` с именем, которое будет подставлено вместо переменной «Тело». С другой стороны, шаблон @Имя_экземпляра's @шаблон, по которому строятся выражения «`тело's скорость`», «`тело's путь`», «`тело's время`» позволяет обращаться к параметрам разных экземпляров одной и той же онтологии, используемых в текущей онтологии.

Примером использования нескольких экземпляров одной онтологии в другой онтологии является онтология следующей задачи.

```
Задача1
```
Среда понятия:

Примеры онтологий равномерного движения

Текст понятия:

```
Пешеход равномерно движется со скоростью 5. Пешеход's
время =2.
Велосипедист равномерно движется со скоростью 6*пешеход's
скорость. Велосипедист's время = 3*пешеход's время.

/*Чему равно велосипедист's путь?*/.

/*****************************************************
По тексту задачи система строит внутреннее представление
онтологии задачи.
Заметим, что в тексте задачи используется шаблон языковой
конструкции,введенный в онтологии "равномерное движение".
В ответ на представленный здесь вопрос система ответит:
"Вопрос: Чему равно велосипедист's путь?
Ответ: 180."
*****************************************************/.
```

Эти примеры показывают простейшие возможности работы с онтологиями, как с модулями.

4.5. Открытость языка шаблонов

Одна из важных функций системы – это открытость языка представления онтологий. Так как предполагается, что онтологии в системе должны, в основном, представлять не программисты, а специалисты в области, к которой относятся онтологии, то нужно сделать этот процесс удобным для таких пользователей. В системе предполагается, что вместе с определением онтологии, пользователь может вводить новые шаблоны языка, которые будут доступными при использовании этой онтологии. Для этого в ядре системы имеется специальный шаблон:

```
«Введем шаблон "@текст_шаблона"
с переменными:
"@список_переменных"
и переменной результата " @переменная_:_Тип " ;
Пояснения: [@текст_пояснения]
Условие применения шаблона:
[@список_условий]
Действие шаблона:
[@действие]
Тип доступа шаблона:[@внешний_наследуемый_локальный]
```

Тип доступа шаблона служит для управления видимостью шаблона. Локальный тип доступа означает, что вводимый шаблон будет видим только из понятия, в котором этот шаблон вводится.

Кроме того, имеется шаблон для введения синонимов существующих терминов: «"@текст_синонима" - новый синоним @термин»».

Для замены (редактирования) шаблона можно воспользоваться шаблоном:

«Заменить шаблон "@текст_шаблона"
с переменными:
"@список_переменных"
и типом результата @тип
на [@Новый_шаблон_с_теми_же_переменными]
и пояснением [@текст_пояснения]».

При этом в новом шаблоне должны быть те же переменные, но могут быть в другом порядке. Этот шаблон производит удаление старого шаблона и вводит новый шаблон с тем же действием.

Удаление шаблонов можно выполнить с помощью шаблона:

«Удалить шаблон "@текст_шаблона"
с переменными типа:
"@список_переменных"
и типом результата @Тип».

Используя эти шаблоны, пользователи могут настроить язык раздела отологий так, что специалисты при работе с онтологиями могут пользоваться шаблонами языка, принятыми в этой области знаний.

Более подробно о настройке языка системы и использовании приведенных выше шаблонов можно узнать в соответствующей главе настоящего пособия.

Глава 5. Создание раздела библиотеки онтологий

Создать новый раздел онтологии или участвовать в разработке онтологий в какой-либо группе пользователей может только зарегистрированный в системе пользователей.

Для зарегистрированного пользователя в левом поле открываются новые пункты меню, среди которых есть «Создать группу» и «Данные обо мне».

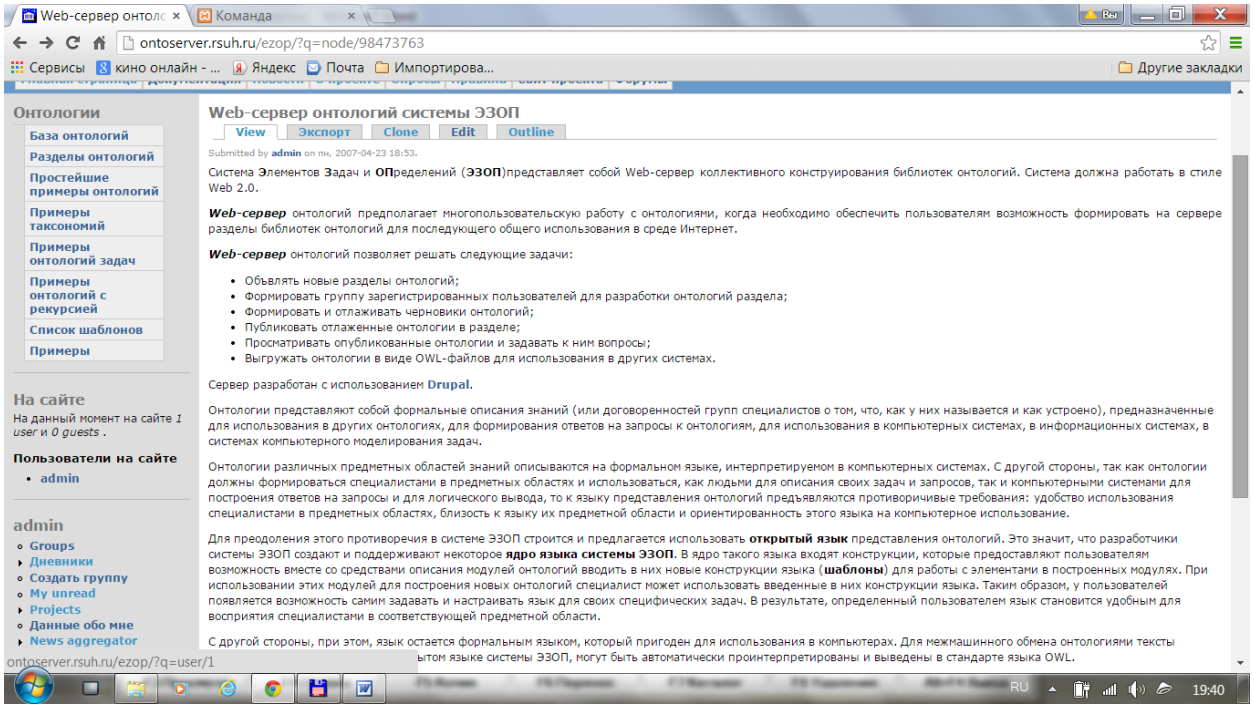

Рис. 5.1. Меню для зарегистрированного пользователя

Если выбрать пункт «Данные обо мне», то открывается окно, в котором перечисляются группы пользователей (разделы), в которых участвует пользователь, и перечисляются онтологии, разработанные данным пользователем.

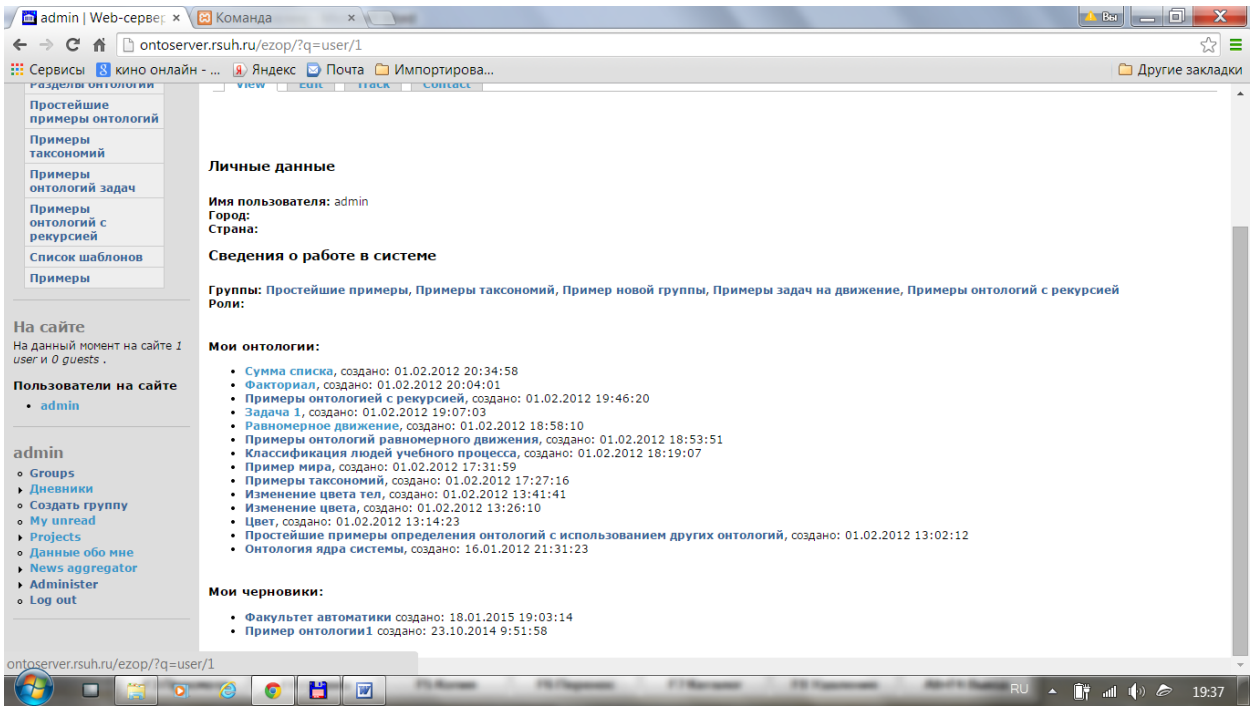

Рис. 5.2. Форма «Данные обо мне»

Для создания нового раздела онтологий (группу пользователей) нужно в меню выбрать пункт «Создать группу». При этом открывается форма создания новой группы.

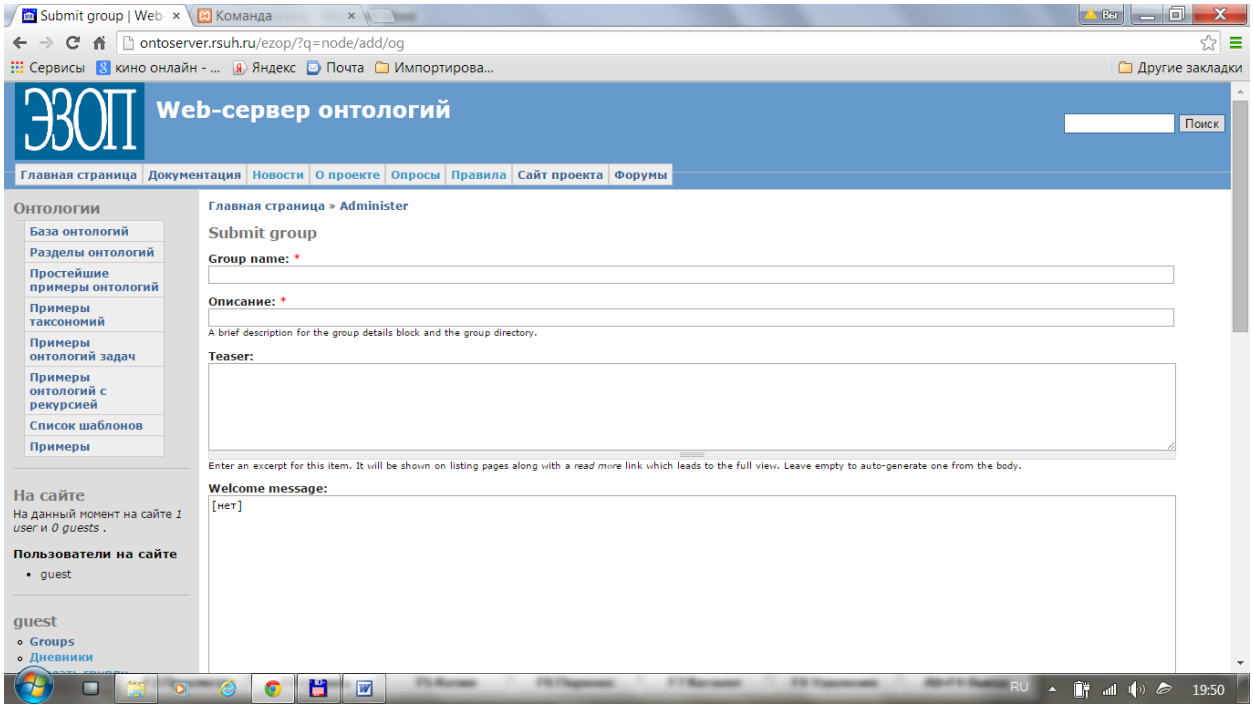

Рис. 5.3. Форма создания новой группы

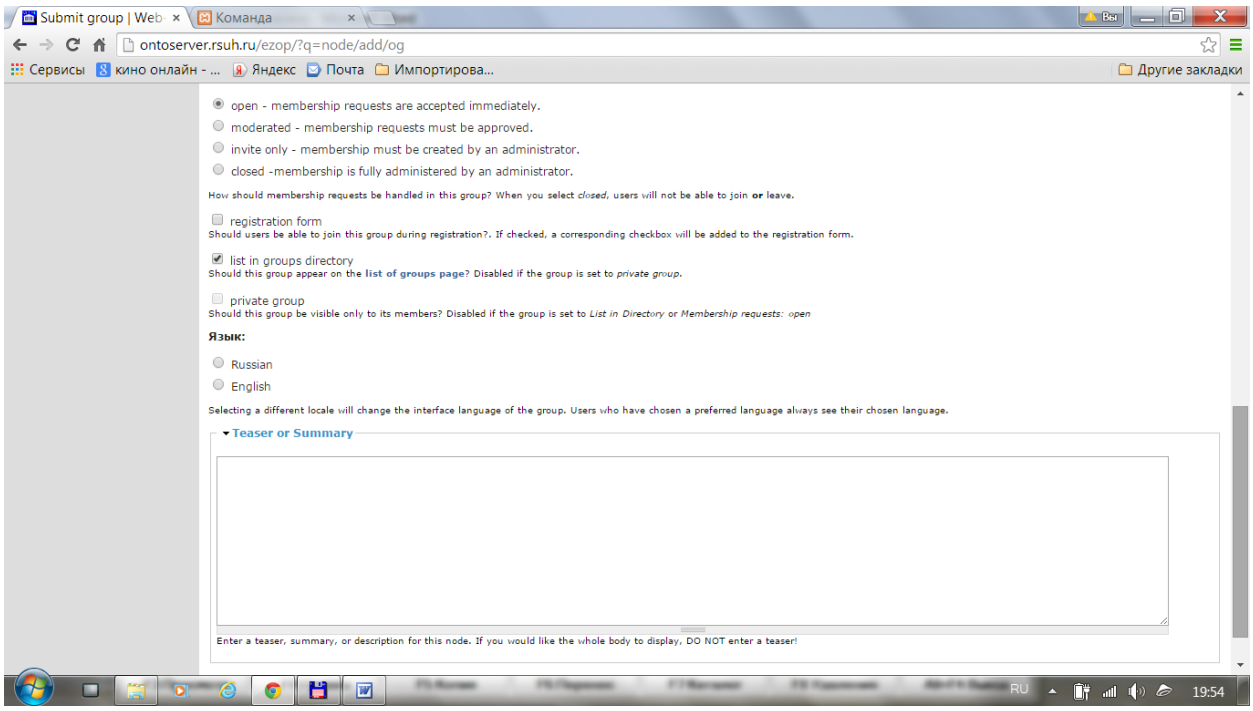

Рис. 5.4. Форма создания новой группы (продолжение)

В этой форме заполняются поля, задающие имя группы и тип группы с типом доступа в группу.

Для того, чтобы создать новую онтологию в группе нужно выбрать пункт меню «Groups» и выбрать, соответствующую группу. Открывается форма группу, в которой для зарегистрированного пользователя становится доступным пункт «Создать онтологию». Если выбрать этот пункт, то открывается список онтологий, среди которых пользователь может выбрать онтологию для среды новой онтологии. Нажав на этой форме кнопку «Выполнить», открывается форма для создания черновика новой онтологии.

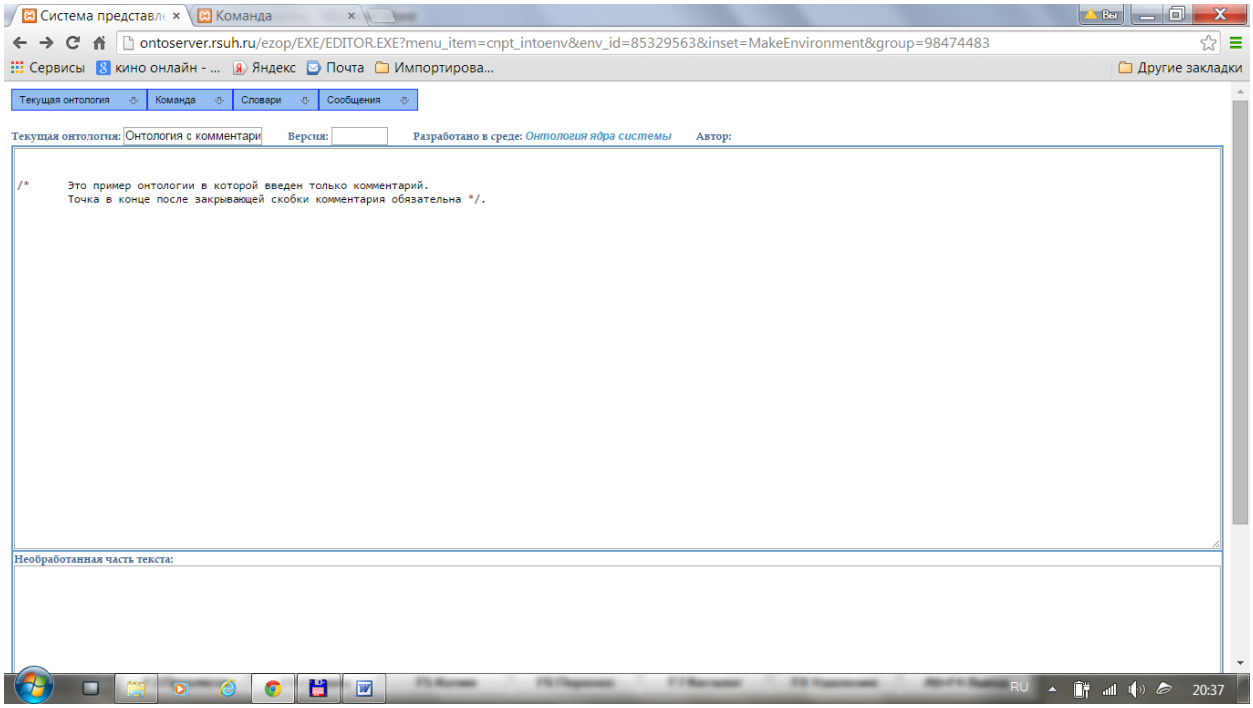

Рис. 5.5. Форма для создания и редактирования черновика онтологии

В этой форме нужно в поле «Текущая онтология» ввести вместо текста «Новая онтология» название новой онтологии, а в основном окне – текст новой онтологии. Для проверки правильности построения онтологии нужно выбрать в горизонтальном меню пункт «Текущая онтология». При этом открывается подменю с пунктами «Построить все», Сохранить черновик» и «Окончательное сохранение».

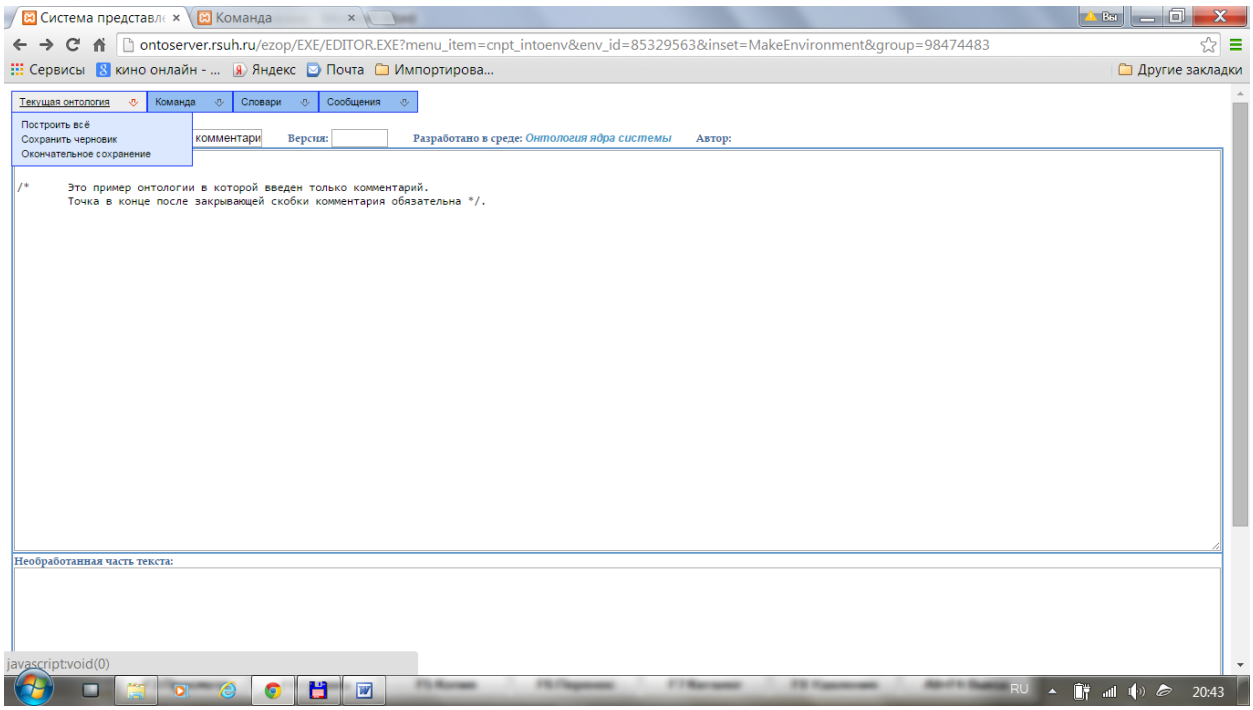

Рис. 5.6. Меню «Текущая онтология» в форме создания и редактирования черновика онтологии

При выборе пункта меню «Построить все» проводится грамматический анализ текста онтологии, выполняются предложения онтологии, и строится внутреннее

представление онтологии. Если в тексте обнаруживается ошибка, то неразобранный текст помещается в окно «Необработанная часть текста».

Сообщения об обработке текста онтологии выводится в окне «Сообщения».

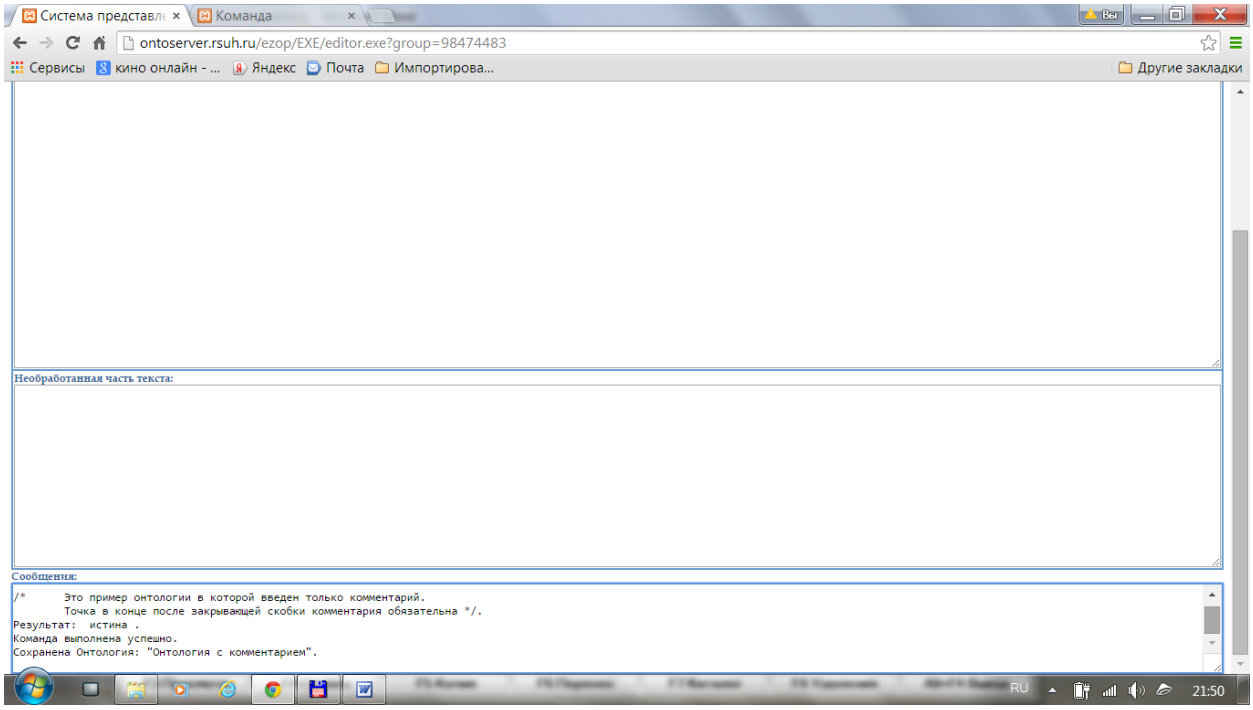

Рис. 5.7. Окно «Сообщения» в форме создания и редактирования онтологий

Если далее выбрать пункт меню «Текущая онтология/Сохранить черновик», то эта онтология появится в списке онтологий группы и в форме «Данные обо мне».

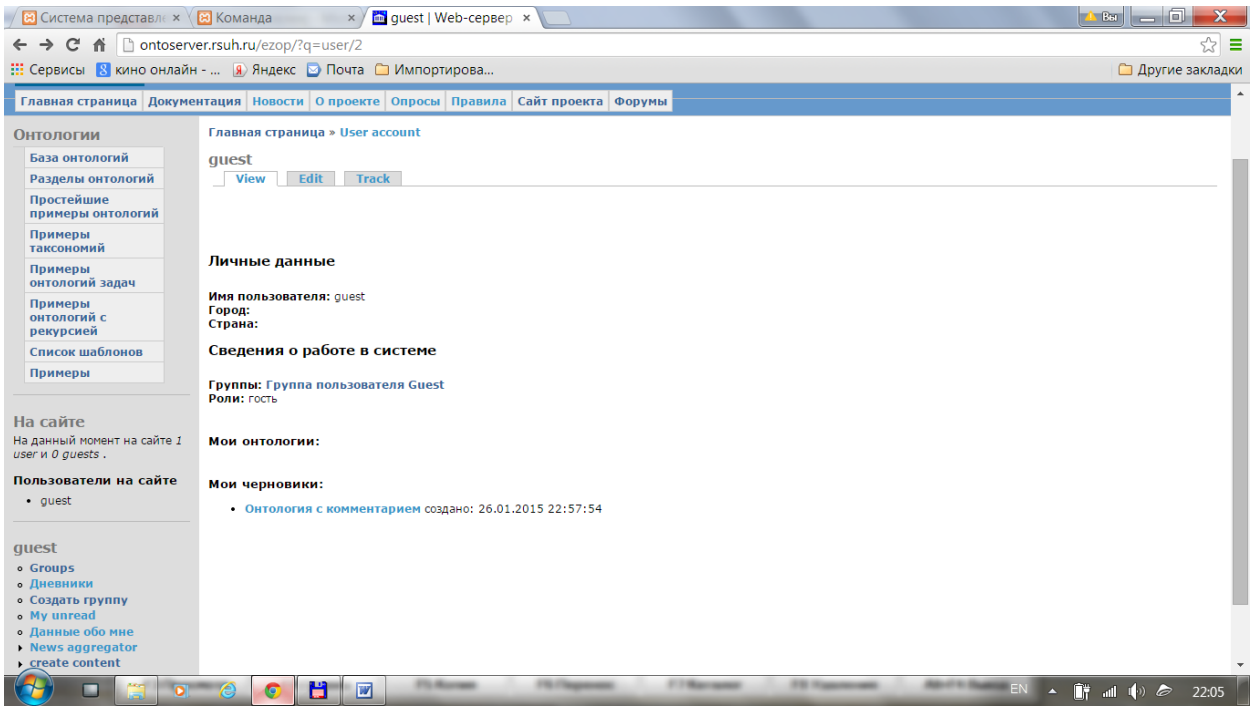

Рис. 5.8. Появление названия новой онтологии в форме «Данные обо мне»

Если в горизонтальном меню выбрать пункт меню «Команда/Новая команда», то открывается форма для формирования вопросов к онтологии. По ответам на вопросы к черновику пользователь может судить о том, правильно ли определена онтология. Если

пользователь считает, что черновик отлажен, он может выбрать пункт меню «Текущая онтология/Окончательное сохранение». В этом случае черновик онтологии становится онтологией, которую могут видеть не только члены группы – разработчиков раздела, в котором разрабатывается данная онтология.

Если в форме редактирования черновика онтологии выбрать пункт меню «Словарь», то открывается форма словаря шаблонов, в котором перечисляются шаблоны языка, доступные из данной онтологии.

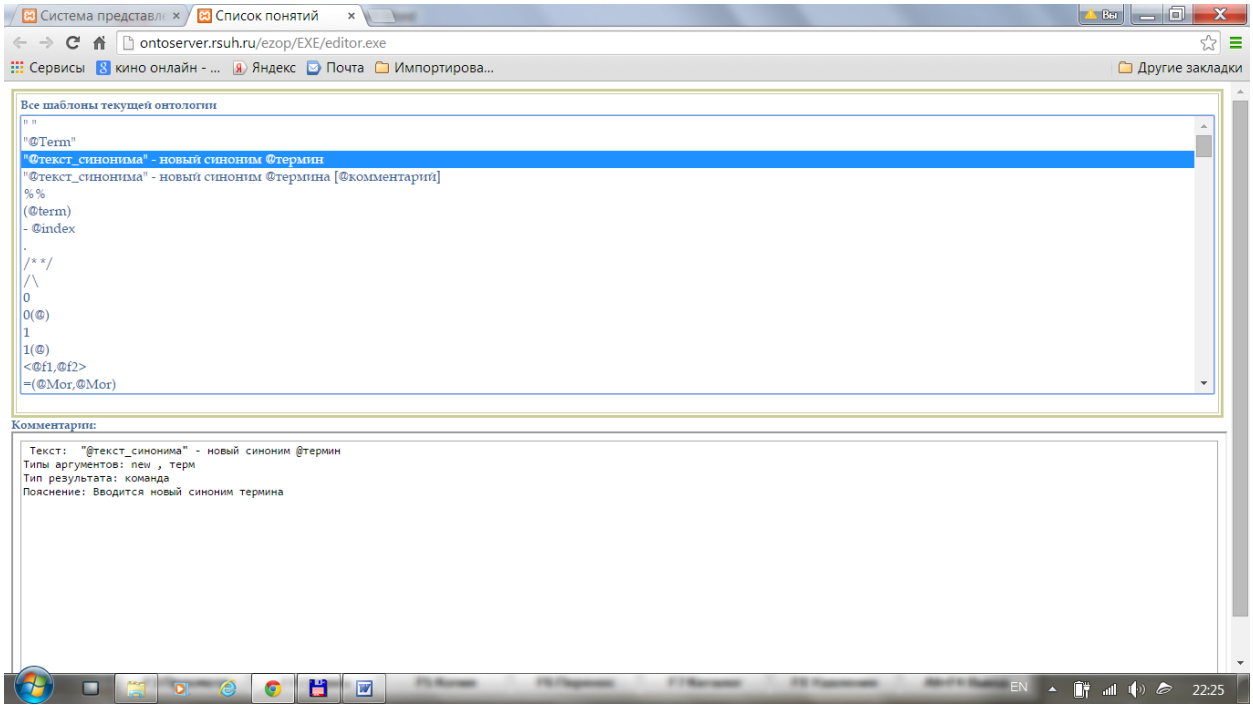

Рис. 5.9. Форма словаря системы

При этом, если в этой форме выбран некоторый шаблон, то в окне «Комментарии» появляется текст, поясняющий этот шаблон. Выбранные в словаре шаблоны можно использовать для формирования текста онтологии или вопросов.

Глава 6. Формирование запросов к онтологии

6.1. Задание простейших вопросов к онтологии

В системе ЭЗОП поддерживается функция обращения произвольных пользователей с вопросами к доступным им онтологиям. Для этого нужно найти соответствующую онтологию, открыть ее и нажать кнопку «Задать вопрос».

Для примера откроем «Онтологию ядра системы»

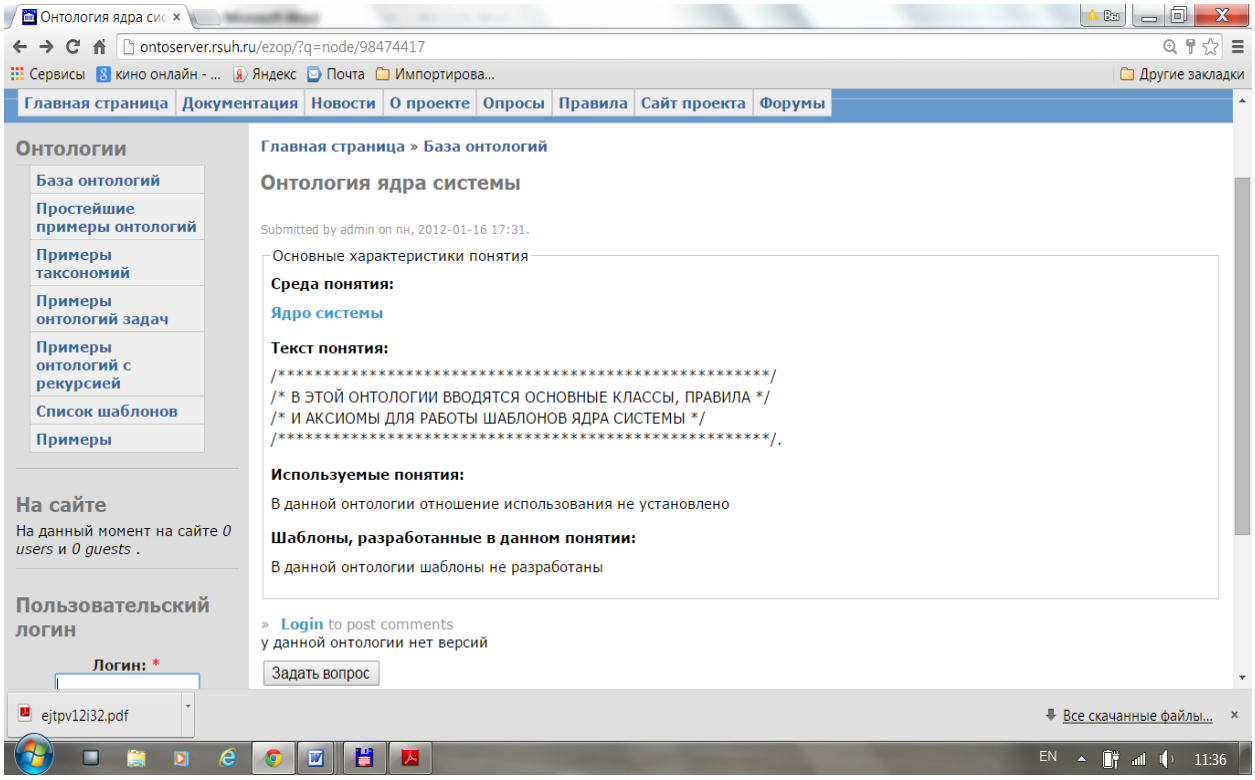

Рис. 6.1. Форма просмотра онтологии

и нажмем кнопку «Задать вопрос». Откроется форма для формирования вопросов и получения ответов на вопросы к онтологии.

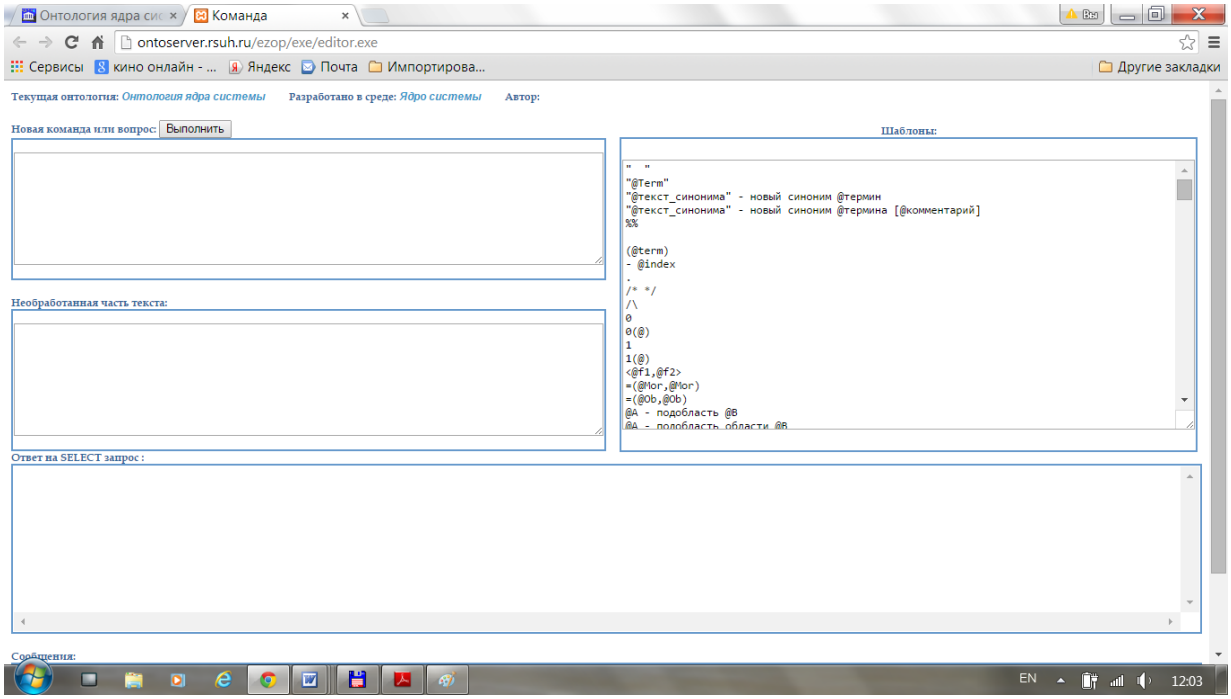

Рис. 6.2. Форма для вопросов к онтологии, ответов и сообщений

В правом верхнем окне формы представлены все шаблоны языка системы, доступные из данной онтологии. В левом верхнем окне пользователь может сформулировать текст вопроса и получить ответ после нажатия кнопки «Выполнить». В среднем левом окне выводится часть текста вопроса, которую система не смогла распознать. В нижнем окне выводятся ответы в виде таблиц на SQL-подобные запросы к

онтологии. В нижней части формы под окном «Ответ на SELECT запрос» находится окно «Сообщения», в котором система выдает сообщения о выполненных действиях.

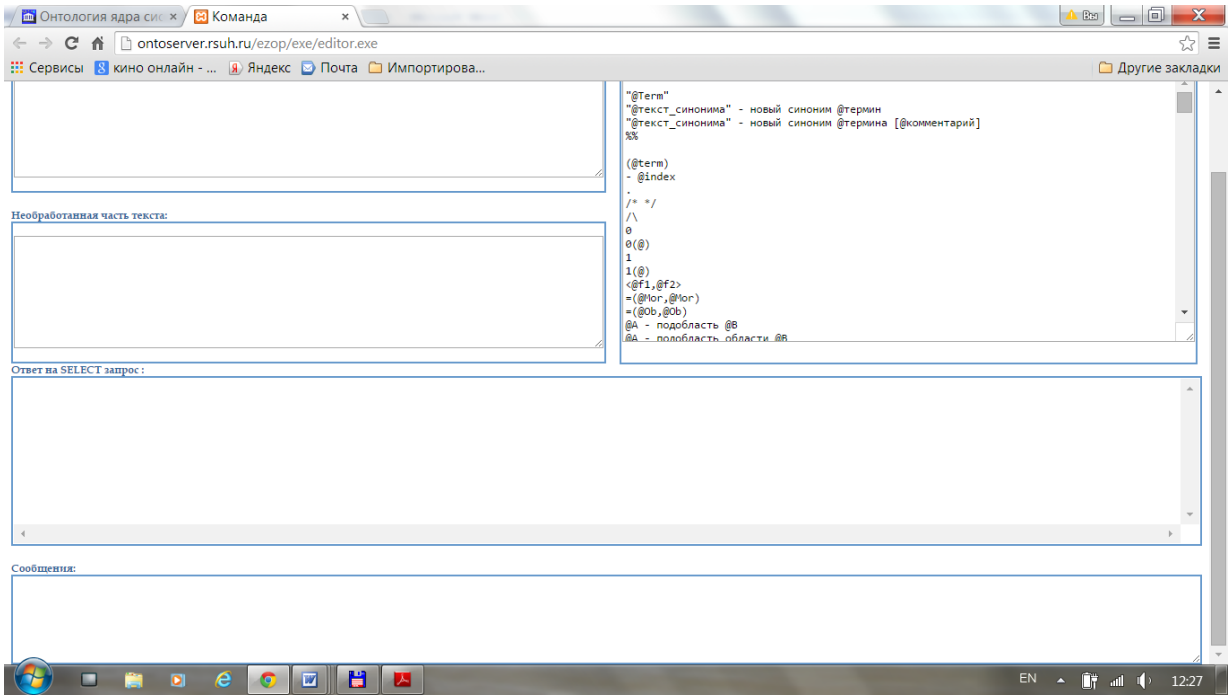

Рис. 6.3. Окно сообщений в форме ответов на вопрос

Для примера, проверим выполнение в онтологии аксиом класс-подкласс и элементов класса и введем в окне вопроса текст:
«[Мейн-кун], кот, животное - области.
Мейн-кун - подобласть области кот. Кот < Животное. Феликс - мейн-кун.
Феликс - животное?
Мейн-кун - животное?
Мейн-кун - подобласть животное?»
и нажмем кнопку «Выполнить», то система выдаст ответы:
«[Мейн-кун], кот, животное - области.
Мейн-кун - подобласть области кот. Кот < Животное. Феликс - мейн-кун.
Феликс - животное?
ответ : " да "
Мейн-кун - животное?
ответ : " нет "
Мейн-кун - подобласть животное?
ответ : " да " »

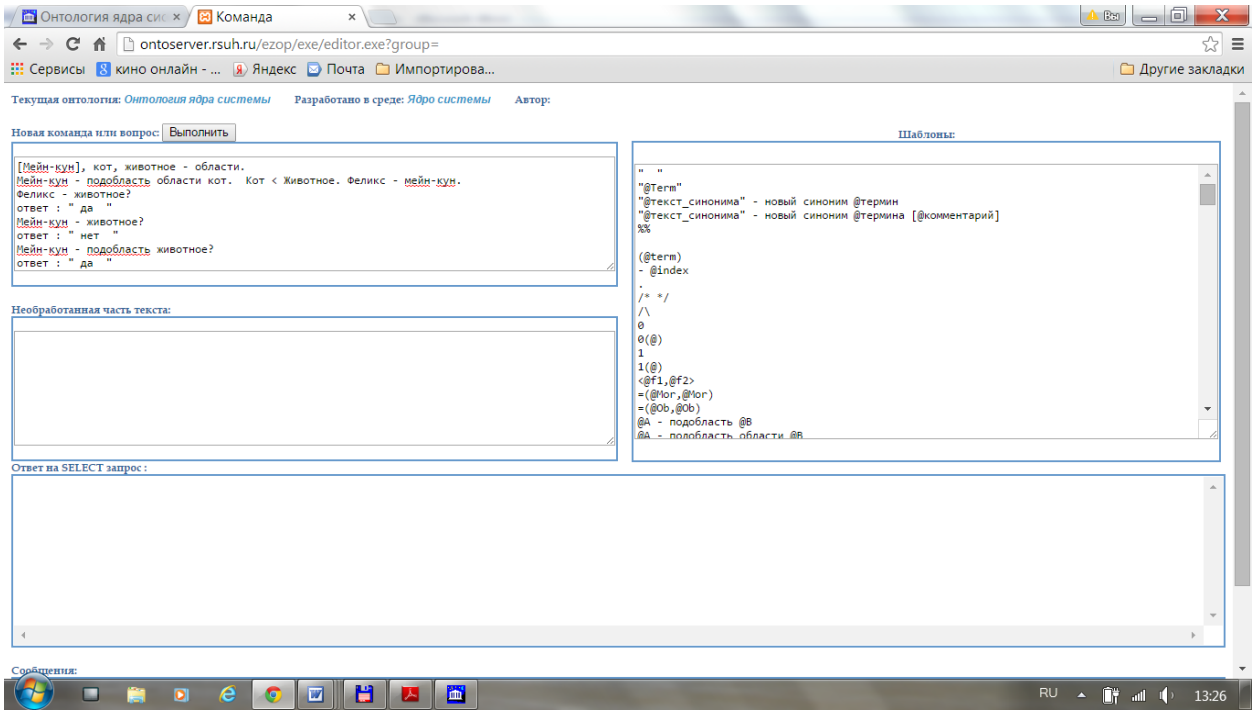

Рис. 6.4. Ответ на вопросы в окне «Новая команда или вопрос»

В онтологии ядра системы имеются типы и шаблоны, поддерживающие арифметические вычисления. Для проверки этой функции введем в окне вопросов выражение: «3*5+35/5-2^3+7*(5-2)?». Нажмем кнопку «Выполнить», то получим ответ в виде:

«3*5+35/5-2^3+7*(5-2)?
ответ : " 35 " ».

Можно также определить новую функцию и задать ее действие правилом переписывания. Для примера введем следующий текст:

«sq: Real_выражение->Real_выражение.
Для всех [t:Real] [sq(t)] => [t^2] ("тело функции sq").
sq(3*4 -5)? »

и выполним его. В результате получим ответ:

«sq: Real_выражение->Real_выражение.
Для всех [t:Real] [sq(t)] => [t^2] ("тело функции sq").
sq(3*4 -5)?
ответ : " 49 " ».

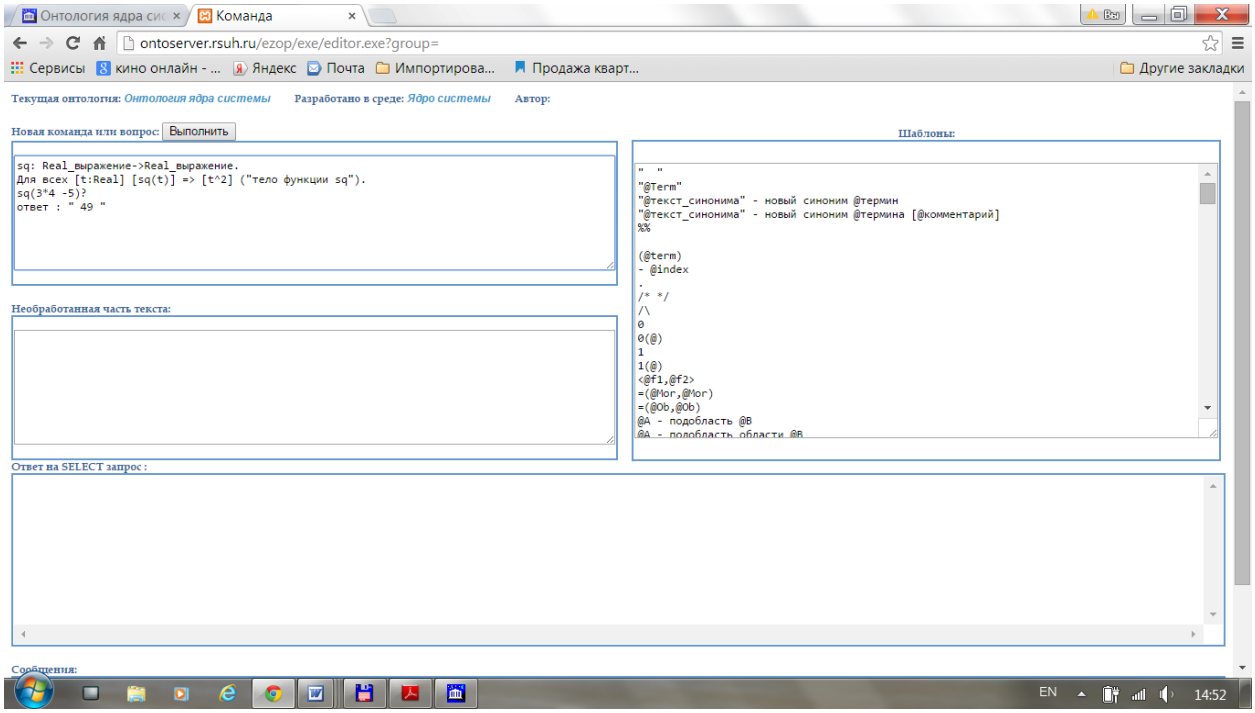

Рис. 6.5. Пример задания функции

Таким способом в онтологии можно водить элементы программ для задания и вычисления функций.

Аналогично, в онтологии ядра системы имеются типы и шаблоны, поддерживающие логические вычисления. Для проверки этой функции введем в окне вопросов выражение: «True &True V ~False --> False & (True V False)?» с операциями конъюнкции, дизъюнкции, импликации и скобками. Нажмем кнопку «Выполнить», то получим ответ в виде:
«True &True V ~False --> False & (True V False)?
ответ : " нет " ».
В системе термин «True» является синонимом термина «да», а термин «False» синонимичен термину «нет».

В булевых выражениях вместо констант можно использовать также предикаты равенства «@Term1 == @Term2», неравенства «@Term1 <> @Term2», быть подобластью «@A - подобласть @B» или «@A < @B», быть элементом области «@Выражение элемент @Области», или «@Выраж принадлежит @Области», или «@Выражение - @Область».

6.2. SQL- подобные запросы

Для SQL- подобных запросов в ядре системы имеется 3 шаблона:

- «Таблица свойств @Об_признаки для @Область»;
- «Select @Области from @Область»;
- «Select "@Функции_от переменных" from "@Переменные" where "@Условие"».

Первые два шаблона синонимичны. В качестве первого аргумента перечисляются некоторые свойства класса, который ставится в качестве второго аргумента этого шаблона. При этом свойства должны быть заданы шаблонами:

- «Свойство @Область: @Имя»;
- «Свойства @Области : @список_свойств».

Для примера рассмотрим онтологию «Факультет информатики»:

```
сотрудник, профессор, доцент, лаборант - области.
профессор < сотрудник. доцент < сотрудник. лаборант < сотрудник.
Свойства сотрудник: фамилия, пол, "год рождения", "год
```

```
поступления на работу".
сотрудник < фамилия.
фамилия of сотрудник = вложение сотрудник в фамилия.
Цаленко, Иванов - элементы области профессор.
Ганнушкина, Вайнтроб - элементы области доцент.
пол of сотрудник (Цаленко) = "мужской".
пол of сотрудник (Иванов) = мужской.
пол of сотрудник (Вайнтроб) = мужской.
пол of сотрудник (Ганнушкина) = "женский".
год рождения of сотрудник (Цаленко) = "1938 г.".
год рождения of сотрудник (Ганнушкина) = "1942 г.".
год поступления на работу of сотрудник (Цаленко) = "1988 г.".
год поступления на работу of сотрудник (Ганнушкина) = "1980 г.".
фамилия of сотрудник (Цаленко) = Цаленко.
фамилия of сотрудник (Иванов) = Иванов.
фамилия of сотрудник (Вайнтроб) = Вайнтроб.
фамилия of сотрудник (Ганнушкина) = Ганнушкина.
/*
Примеры запросов:
Таблица свойств фамилия, пол, год рождения для сотрудник?
Select фамилия, пол, год рождения, год поступления на работу
from сотрудник?
*/.
```

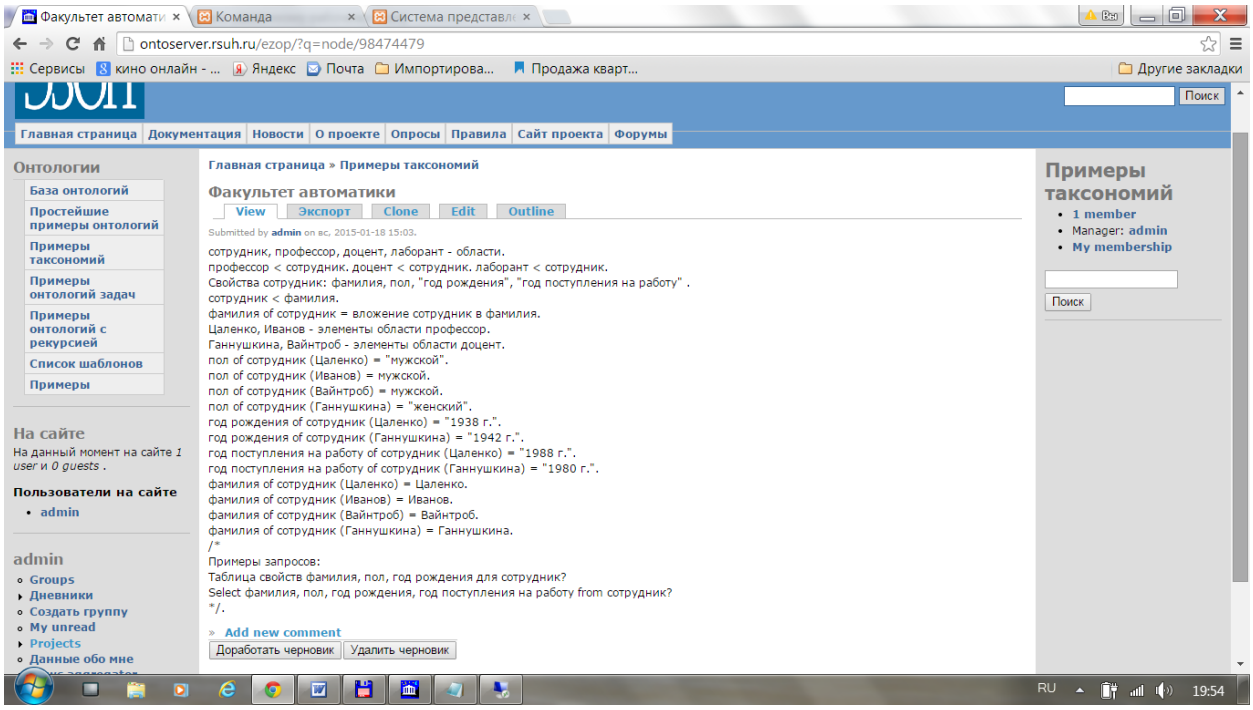

Рис. 6.6. Онтология «Факультет автоматики»

Если к этой онтологии обратиться с запросом: «Таблица свойств фамилия, пол, год рождения для сотрудник?», то ответ получится в табличной форме вида:

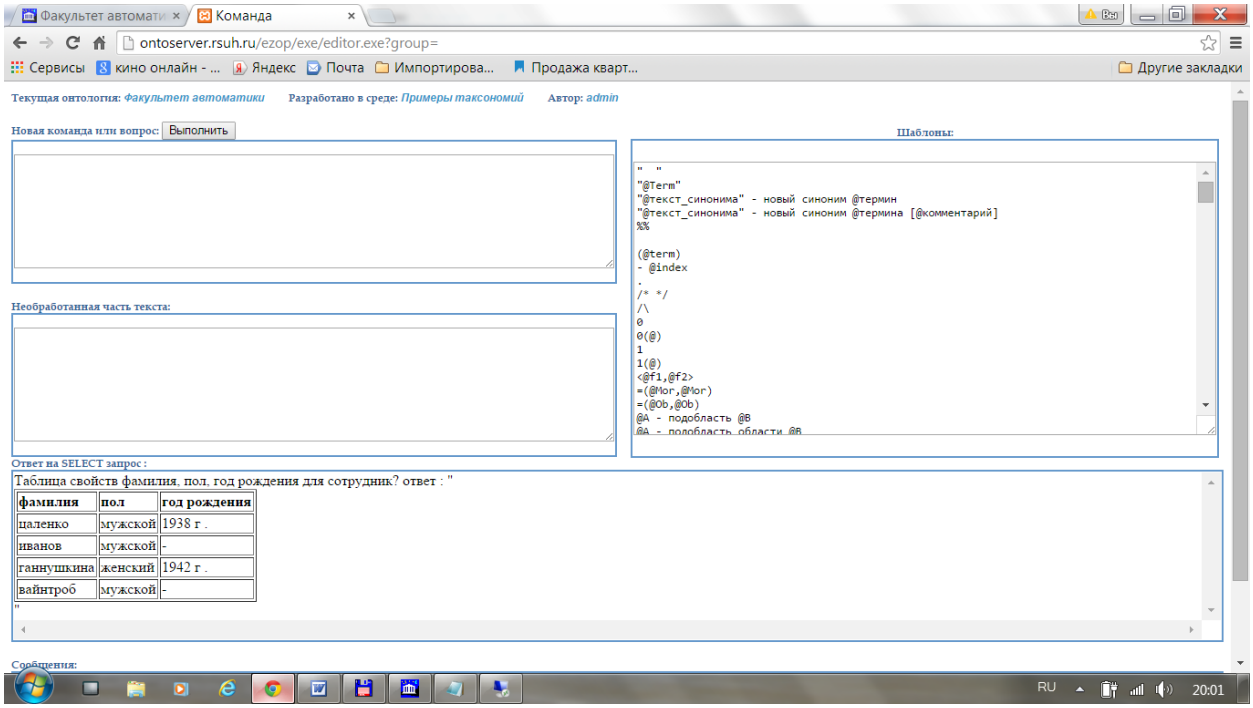

Рис. 6.7. Ответ на запрос «Таблица свойств фамилия, пол, год рождения для сотрудник?»

Шаблон «Таблица свойств @Об_признаки для @Область» является устаревшим по сравнению с шаблоном «Select "@Функции_от переменных" from "@Переменные" where "@Условие"».

В первом окне (вместо первой переменной) в шаблоне «Select "@Функции_от переменных" from "@Переменные" where "@Условие"» подставляется список функциональных термов от переменных, которые входят в список переменных определяемых в списке переменах, перечисляемых во втором окне этого шаблона. В качестве условия в этом шаблоне пишется булево выражение от тех же переменных.

Результатом вычисления этого запроса является таблица, строками которой являются результаты вычисления данного списка функциональных термов при всех значениях перечисленных переменных, удовлетворяющих условию запроса.

Для примера обратимся к онтологии «Изменение цвета тел»:

```
"тела" - область.
Пусть [куб], [пирамида],[шар] - тела.
Используем понятие [изменение цвета].
[цвет] - отображение.
цвет: тела -> Цвета.
цвет(куб) = красный.
цвет(пирамида) = синий.
цвет(шар) = красный.
```

с запросом:

```
«Для всех[ у: Тела ] [ id(Тела)(у) ] => [у] ("Тождество").
Select "id(Тела)(т),Цвет(т), Измененный (цвет(т))" from "т:
тела" where "true"?».
```

В ответ получим таблицу:

id(тела)(т)	цвет(т)	измененный (цвет(т))
куб	красный	синий
пирамида	синий	зеленый
шар	красный	синий

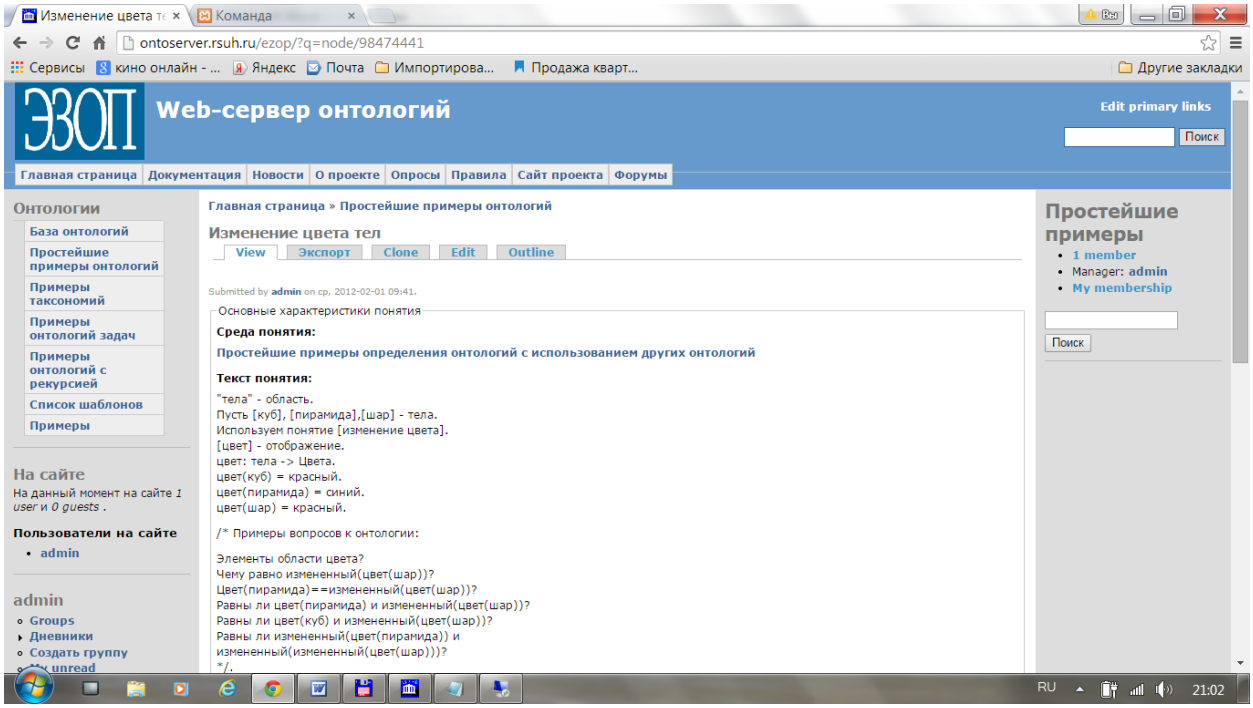

Рис. 6.8. Онтология «Изменения цвета тел»

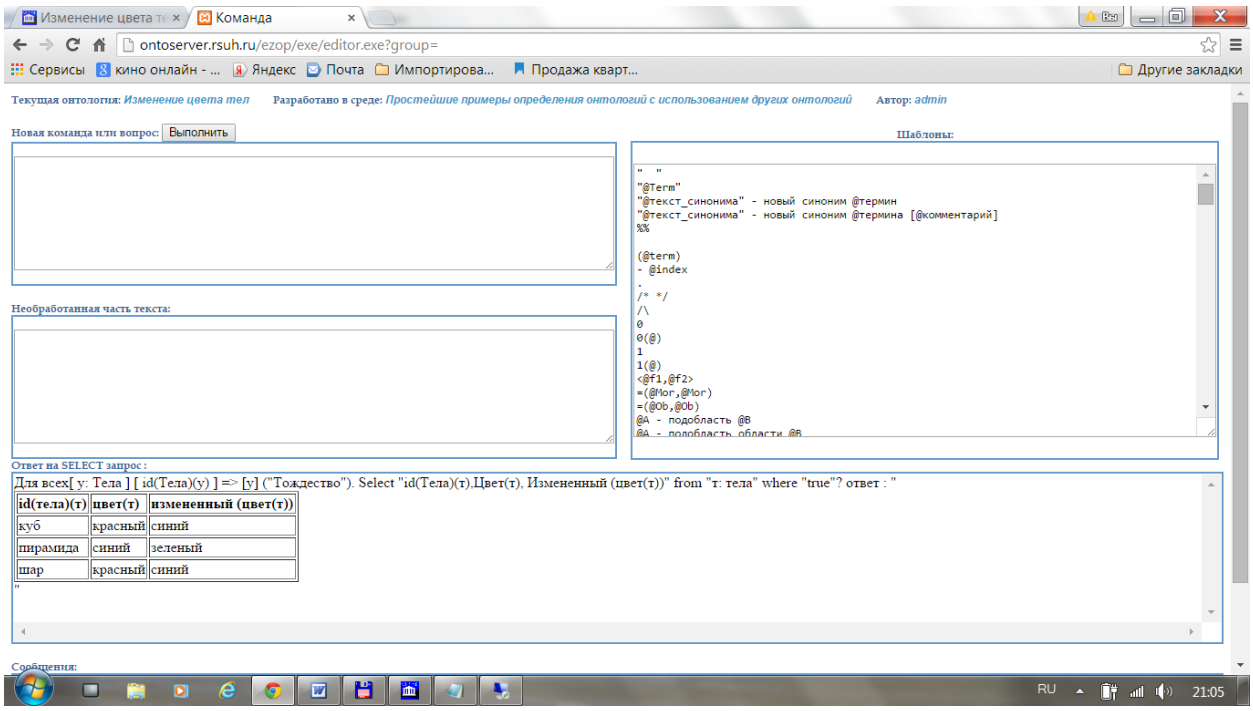

Рис. 6.9. Ответ на SQL-подобный запрос к онтологии «Изменение цвета тел»

Глава 7. Настройка языка пользователей раздела

Как говорилось ранее, в системе ЭЗОП есть специальные возможности настройки языка. В частности, можно переопределить скоки комментариев, ввести синонимы терминов, удалить шаблоны (сделать их невидимыми из текущей онтологии), добавить новые языковые шаблоны и определить действия этих шаблонов.

7.1. Переопределение скобок комментариев

С помощью шаблонов введения синонимии «"@текст_синонима" - новый синоним @термин» пользователь может изменить системные шаблоны для кавычек и выделения комментариев. Покажем это в следующем примере:

Пример определения новых скобок комментария

```
/* Это комментарий */
/* Введем новые скобки для комментария:           */

" < >" - новый синоним "[/*]  [*/]".

< Это комментарий в новых скобках>.
<*********************************************************
*******>

%% Это комментарий.
%% Введем новые скобки для комментария этого типа:
      "%  [
]" - новый синоним "[/*]  [*/]".

% Это комментарий в новых обозначениях  от знака процента до
% конца строки
.
%*******************************************************
%   С помощью шаблона синонимии можно изменить скобки для
% кавычек
%*******************************************************

["//"  "//"] - новый синоним ["[" "]"].
 % Одинарные / будут конфликтовать
 %со скобками комментария /*  */

//два слова// - область.     % Проверка новых кавычек
.
```

Для введения нового термина можно также воспользоваться шаблоном: «Заменить термин "@Термин" типа @Тип на "@Новый_термин" [@текст_пояснения]».

Действие этого шаблона состоит в построении синонима термина и удаление самого термина из текущего понятия (внешние шаблоны не удаляются, а становятся недоступными из текущего понятия).

7.2. Ввод шаблонов

При вводе шаблона общего вида пользователи указывают вид шаблона, типы переменных в определяемом шаблоне, тип результата действия шаблона и само действие шаблона. При необходимости пользователи могут определить условие применения определяемого шаблона. Для ввода шаблонов пользователями имеется специальный шаблон в системе в паке шаблонов «Управление шаблонами» вида:

```
Введем шаблон "@текст_шаблона"
с переменными:
```

```
"@список_переменных"
и переменной результата " @переменная_:_Тип " ;
Пояснения: [@текст_пояснения]
Условие применения шаблона:
[@список_условий]
Действие шаблона:
[@действие]
Тип доступа шаблона:[@внешний_наследуемый_локальный]
```

Тип доступа шаблона служит для управления видимостью шаблона. Локальный тип доступа означает, что вводимый шаблон будет видим только из понятия, в котором этот шаблон вводится.

Приведем пример ввода нового шаблона Sum(@x1,@x2) для сложения чисел. Это делается с помощью выражения:

```
Введем шаблон " Sum(@x1, @x2)"
с переменными:
"x1, x2 : real"
и переменной результата " z: real";
Пояснения: [Сумма двух чисел, но не выражений.]
Условие применения шаблона:
[]
Действие шаблона:
[z=x1+x2]
Тип доступа шаблона: [локальный].
```

Теперь, если вы введете в окне «Команда или вопрос» выражение «Sum(3,5)?» система поймет вас и выдаст в ответе число 8. Однако, если вы зададите выражение «Sum(3,5*2)?» система выдаст сообщение об ошибке, так как выражение 5*2 имеет тип не real, а real_выражение.

Если вы хотите, чтобы в качестве аргументов нового шаблона можно было бы использовать не только числа, а произвольные числовые выражения, нужно отредактировать (в главном меню выбрать пункт «Текущее понятие»/«Редактировать») предыдущий текст определения шаблона в виде:

```
Введем шаблон " Sum(@x1,@x2)"
с переменными:
"x1, x2 : real_выражение"
и переменной результата " z: real " ;
Пояснения: [Сумма двух чисел, но не выражений]
Условие применения шаблона:
[]
Действие шаблона:
[z=(x1)+(x2)]
Тип доступа шаблона:[локальный].

/* Пример вопроса:  3+Sum(3*2, 3^2)/3? */.
```

и запустить в меню пункт «Построить все».

Теперь система сможет производить любые арифметические вычисления с использованием нового шаблона. Например, если в окне «Команда или вопрос» вы введете выражение «3+Sum(3*2, 3^2)/3?», то система правильно вычислит это выражение.

Введем пример шаблона с условием применения. В качестве примера введем новый шаблон для операции деления чисел в виде:

```
Введем шаблон " @x1 div @x2"
с переменными:
"x1: слагаемое;  x2 : степенное real_выражение "
и переменной результата "z: слагаемое";
Пояснения: [Деление числа на число, неравное 0]
Условие применения шаблона:
[ ~x2==0]
Действие шаблона:
[z=(x1)/(x2)]
Тип доступа шаблона: [локальный].
```

В этом примере условием применения шаблона является неравенство нулю первого аргумента.

Вводимый шаблон может использоваться в определении действия самого шаблона, т.е. допустимо рекурсивное определение действия шаблона. Пример такого определения используется в понятии «Факториал»:

```
     /* Определение понятия (ортологии факториал*/
Введем шаблон "fact(@n,@i,@y)"
с переменными:
"n,i,y :real_выражение"
и переменной результата " z:real_выражение  " ;
Пояснения: [Вычисляет факториал]
Условие применения шаблона:
[]
Действие шаблона:
[
if n==i then z=y else z=fact(n,i+1,(y)*(i+1))
]
Тип доступа шаблона:[локальный].

Введем шаблон "@n!"
с переменными:
"n :real"
и переменной результата " z:real  ";
Пояснения: [Вычисляет факториал]
Условие применения шаблона:
[]
Действие шаблона:
[
z=fact(n,1,1)
]
Тип доступа шаблона:[внешний].
```

Более короткий вариант определения факториала имеет вид:

```
Введем шаблон "factorial(@n)"
с переменными:
"n :real_выражение"
и переменной результата " z:real_выражение  ";
Пояснения: [Вычисляет факториал]
```

```
Условие применения шаблона:
[]
Действие шаблона:
[
if n==1 then z=1 else z=(factorial(n-1))*(n)
]
Тип доступа шаблона:[внешний].
```

Дополнительные скобки в выражении (factorial(n-1))*(n) введены для того, чтобы можно было применить шаблон умножения « @*@» к factorial(n-1) и n, которые имеют тип real_выражение.

Удаление шаблонов можно выполнить с помощью шаблона:

«Удалить шаблон "@текст_шаблона"
с переменными типа:
"@список_переменных"
и типом результата @Тип».

Например, для удаления шаблона factorial(@) можно ввести командную строку в окне «Команда или вопрос» вида:

```
Удалить шаблон " factorial(@k)"
с переменными типа:
"k : real"
и типом результата  real.
```

Эта команда на самом деле не удаляет шаблон из системы (если только текущим понятием не является то, в котором это понятие было введено), а делает его невидимым из текущего понятия. После выполнения этой команды она будет отражена в тексте текущего понятия, и этот шаблон будет невидим из текущего понятия при вводе последующих команд.

Для замены (редактирования) шаблона можно воспользоваться шаблоном:

«Заменить шаблон "@текст_шаблона"
с переменными:
"@список_переменных"
и типом результата @тип
на [@Новый_шаблон_с_теми_же_переменными]
и пояснением [@текст_пояснения]».

При этом в новом шаблоне должны быть те же переменные, но могут быть в другом порядке. Этот шаблон производит удаление старого шаблона (в том же смысле, как было объяснено выше) и вводит новый шаблон с тем же действием.

7.3. Видимость шаблонов

Принципы видимости отражены на рис.1 и могут быть перечислены в следующем виде.

1. Из текущей онтологии видны все термины и шаблоны, введенные в этой онтологии и не удаленные в последующих предложениях.

2. Из текущей онтологии видны все термины и шаблоны, введенные в онтологиях, используемых в текущей онтологии, и не удаленные в последующих предложениях, за исключением тех терминов и шаблонов, которые были объявлены как локальные в этих онтологиях.

3. Шаблоны, определенные как внешние, из текущей онтологии видны только те, которые введены в онтологиях, построенных в онтологиях-средах, используемых в этой текущей онтологии.

4. При удалении шаблонов, не введенных в текущей онтологии, шаблоны не удаляются, а становятся невидимыми из текущей онтологии.

Системные шаблоны введены в ядре системы, которое является частью среды для любой онтологии. Поэтому шаблон ядра системы виден из любой онтологии, если только в самой этой онтологии или в ее среде (или части среды) не удален (или заменен) этот шаблон.

Для перевода системных шаблонов на английский, французский или немецкий языки можно, например, построить онтологии в среде ядра системы с именами «English», «French» или «German», в которых переводятся на соответствующий язык системные шаблоны. Далее, используя построенные онтологии в качестве среды, появляется возможность работать с системными шаблонами на разных языках. Если потребуется в некоторой онтологии работать с системными шаблонами на двух языках, например английском и французском, одновременно, то в эту онтологию (или ее среду) можно подгрузить онтологии «English»и «French», и тогда это станет возможным.

Объявление онтологий в качестве сред, может использоваться для организации разделов онтологий со своими вариантами онтологий и шаблонов языка. Так, в среде ядра системы можно построить онтологии «Алгебра», «Механика», «Системные примеры», «Бениаминов», которые будут содержать лишь тексты комментариев. Если эти онтологии использовать в качестве сред для последующих онтологий, то мы получим соответственно разделы онтологий алгебры, механики, системных примеров, Бениаминова, и шаблоны, введенные в одном разделе, не будут видны из онтологий другого раздела. С другой стороны, внешние шаблоны, введенные в онтологиях одного раздела видны из онтологий этого же раздела. Если в текущей онтологии требуется открыть доступ к шаблонам некоторого раздела, например алгебры, то, добавив команду «Используем понятие [Алгебра]», из текущего понятия открывается доступ к внешним шаблонам, введенным в понятиях раздела «Алгебра».

Для каждого вводимого внешнего шаблона типы его аргументов и тип результаты должны быть видимы в среде онтологии, в которой вводится этот шаблон. Тогда, когда будет видим сам шаблон, то будут видны и доступны типы его аргументов и результата.

Глава 8. Примеры онтологий

В этом разделе описываются примеры онтологий и работы с ними.

Основная онтология системы называется "Ядро системы". Эта онтология создана разработчиками системы. В ней определены базовые классы и базовые шаблоны языка вместе с действием этих шаблонов.

8.1. Ядро системы

Если сделать "Онтологию ядра системы" текущей онтологией, то можно увидеть текст онтологии, поясняющей ее содержание.

Если сделать онтологию "Онтологию ядра системы" средой, то в этой среде вы можете определить новую онтологию, используя все возможности языка шаблонов ядра системы.

Нажав на пункт меню "Словарь", вам откроется словарь всех шаблонов языка ядра, с помощью которых вы можете описать новые онтологии, определяя новые классы (области), новые элементы классов, новые функции и связать их отношениями класс-подкласс и равенствами. В словаре имеются шаблоны для определения новых конструкций языка: новые шаблоны, синонимы, новые типы комментарий.

Кроме того, в языке ядра системы имеются шаблоны для формулирования вопросов к онтологии, включая:

- "Элементы области @Области?";
- "Подобласти области @Область?";
- "Чему равно @выражение?".

8.2. Простейшие примеры определения онтологий с использованием других онтологий

Рассмотрим онтологию "цвет":

```
    /*****************************************************/
[Цвета] - область.
"теплые цвета" < цвета.
" холодные цвета" < цвета.
"Красный", "желтый" - элементы области теплые цвета.
Пусть зеленый, синий, белый - холодные цвета.

/* Примеры вопросов к понятию:
Элементы области цвета?
Желтый - цвета?
Оранжевый - цвета?
Холодные цвета < цвета?
Равны ли желтый и оранжевый?

*/.
    /*****************************************************/
```

Эта онтология разработана в среде онтологии "Ядро системы".
Ее текст написан с использованием шаблонов языка ядра.

Первое предложение вводит в онтологию новую область (класс) "Цвета". Квадратные скобки и кавычки используются для обозначения новых терминов, вводимых в онтологии. В следующих двух предложениях вводятся подклассы "холодные цвета" и "теплые цвета" класса "цвета". Элементы этих классов вводятся в следующих предложениях с помощью двух разных шаблонов. Комментарии в тексте онтологии выделяются скобками вида /* ... */. В комментариях здесь указываются примеры вопросов к онтологии, которые могут использоваться для ее проверки.

Следующая онтология, которую мы рассмотрим, называется "изменение цвета":

```
    /*****************************************************/
Используем понятие [цвет].
[измененный] - отображение.
Измененный : Цвета -> Цвета.
Измененный (красный) = синий.
Измененный (синий) = зеленый.
Измененный (зеленый) = зеленый.

/* Примеры вопросов к понятию:
Элементы области цвета?
```

```
Чему равно измененный(синий)?
Чему равно измененный(желтый)?
Желтый - цвет?
*/.
    /*****************************************************/
```

Эта онтология также разрабатывается в среде "Ядра системы" с использованием шаблонов ядра. В первом предложении в разрабатываемую онтологию загружается предыдущая онтология. В следующих предложениях вводится новое отображение из класса "цвета" в этот же класс, и определяются значения введенного отображения на элементах с помощью равенств.

И, наконец, рассмотрим третью онтологию с названием "изменение цвета тел":

```
    /*****************************************************/
"тела" - область.
Пусть [куб], [пирамида],[шар] - тела.
Используем понятие [изменение цвета].
[цвет] - отображение.
цвет: тела -> Цвета.
цвет(куб) = красный.
цвет(пирамида) = синий.
цвет(шар) = красный.

/* Примеры вопросов к понятию:

Элементы области цвета?
Чему равно измененный(цвет(шар))?
Цвет(пирамида)==измененный(цвет(шар))?
Равны ли цвет(пирамида) и измененный(цвет(шар))?
Равны ли цвет(куб) и измененный(цвет(шар))?
Равны ли измененный(цвет(пирамида)) и
измененный(измененный(цвет(шар)))?
*/.
    /*****************************************************/
```

В первых двух предложениях вводится класс "тела" и некоторые его элементы. Далее, в определяемую онтологию вводится онтология "изменение цвета", вводится функция, задающая для каждого тела его цвет, и равенствами связываются элементы, введенные в текущей онтологии и в предыдущих онтологиях.

Эти примеры показывают простейшие возможности работы с онтологиями.

8.3. Примеры онтологий-таксономий

Таксономии также могут быть созданы в среде онтологии "Ядро системы" с использованием шаблонов ядра.

Простейшие системы могут состоять из классов, связанных отношением класс-подкласс и указанием элементов классов.

Примером такой онтологии является онтология "пример мира":

```
    /*****************************************************/
/* ПРИМЕР ОПИСАНИЯ МИРА */
[ТЕЛА],[ОДУШЕВЛЕННЫЕ ТЕЛА],[НЕОДУШЕВЛЕННЫЕ ТЕЛА],
[ПРЕДМЕТЫ], [ЖИВОТНЫЕ],[РЫБЫ],[ЛЮДИ],[МУЖЧИНЫ],[ЖЕНЩИНЫ] -
области.
ОДУШЕВЛЕННЫЕ ТЕЛА - подобласть области ТЕЛА.
НЕОДУШЕВЛЕННЫЕ ТЕЛА - подобласть области ТЕЛА.
ПРЕДМЕТЫ < НЕОДУШЕВЛЕННЫЕ ТЕЛА.

ЖИВОТНЫЕ < ОДУШЕВЛЕННЫЕ ТЕЛА.
РЫБЫ < ЖИВОТНЫЕ.
ЛЮДИ < ЖИВОТНЫЕ.

МУЖЧИНЫ - подобласть области ЛЮДИ.
ЖЕНЩИНЫ - подобласть области ЛЮДИ.
[Петя], [Коля] - элементы области МУЖЧИНЫ.

стул, стол - элементы области ПРЕДМЕТЫ.

/* Пример вопроса:

Элементы области одушевленные тела?

*/.
    /*****************************************************/
```

Другой пример таксономий дает онтология "Классификация людей учебного процесса":

```
    /*****************************************************/
/* Пример взят из книги Цаленко "Моделирование семантики в базах
данных" */

[люди] - область.

Свойства люди :
фамилия,
[год рождения],
пол.

Подобласти люди:
сотрудник,
учащийся.

Свойства сотрудник:
факультет,
[год поступления на работу],
оклад.

Подобласти сотрудник:
преподаватель,
```

```
[административный персонал],
[обслуживающий персонал].

Подобласти преподаватель:
[заведующий кафедрой],
профессор,
доцент,
ассистент.

Подобласти административный персонал:
[заведующий лабораторией],
инспектор,
декан.

Подобласти обслуживающий персонал:
лаборант,
техник.

Свойства учащийся:
факультет,
"год поступления".

Подобласти учащийся:
студент,
аспирант.
    /*****************************************************/
```

В этом примере вводятся не только классы, но и наборы их свойств.

Аналогично могут быть описаны онтологии-таксономии "Классификация оборудования учебного процесса" и "Классификация помещений учебного процесса". Причем каждая классификация делается специалистом в этой области.

Эти онтологии-таксономии могут использоваться при построении онтологии, описывающей конкретный учебный процесс конкретного ВУЗа в определенный момент времени. В такой онтологии загружаются необходимые таксономии и определяются элементы классов с указанием для них значений требуемых свойст. В результате получится онтология, описывающая состояние некоторого учебного процесса. К этой онтологии можно обращаться с вопросами, построенными из шаблонов языка вопросов, описанных в ядре системы, а система строит ответы на запросы к онтологии даннного учебного процесса в данном состоянии.

Шаблоны языка запросов, описанные в ядре, к онтологиям-таксономиям должен быть SQL-подобным, и в идеале соответствовать принятым стандартам, например SPARQL.

8.4. Примеры алгебр

В этом разделе описываются примеры онтологий, описывающие алгебры. Каждая алгебра представляется в виде набора множеств и набора операций, действующих на них.

Рассмотрим пример онтологии "Булева алгебра":

```
    /*****************************************************/
bool -область.
f,t -элементы области bool.
AND: bool x bool ->bool.
OR : bool x bool -> bool.
NOT : bool ->bool.

/* Таблица действия логических операций */

NOT(f)=t. NOT(t)=f.
AND(f; f)=f. AND(f; t)=f. AND(t; f)=f. AND(t; t)=t.
OR(f; f)=f. OR(f; t)=t. OR(t; f)=t. OR(t; t)=t.

Введем шаблон "@u & @v"
с переменными:
"u : bool; v : bool"
и переменной результата " w:bool " ;
Пояснения: [операция конъюнкции]
Условие применения шаблона:
[]
Действие шаблона:
[w=AND(u;v)]
Тип доступа шаблона: [внешний].

Введем шаблон "@u V @v"
с переменными:
"u : bool; v : bool"
и переменной результата " w:bool " ;
Пояснения: [операция дизъюнкции]
Условие применения шаблона:
[]
Действие шаблона:
[w=OR(u;v)]
Тип доступа шаблона: [внешний].

Введем шаблон " ~@u "
с переменными:
"u : bool"
и переменной результата " w:bool " ;
Пояснения: [операция отрицания]
Условие применения шаблона:
[]
Действие шаблона:
[w=NOT(u)]
Тип доступа шаблона: [внешний].

    /*****************************************************/
```

В данном примере вводятся булевы операции в префиксном виде, как имена функций. В инфиксном виде операции задаются в виде шаблонов: "~@u", "@u v @v", "@u & @v".

Заметим, что для инфиксных операций в этом примере не заданы приоритеты и все операции выполняются слева-направо. Чтобы в запросе задать нужную последовательность выполнения операций нужно использовать скобки.

Примерами запросов к этой онтологии являются выражения:
"Чему равно t v t&f?",
"Чему равно t v (t&f)?".

Если вы хотите, чтобы в выражениях сначала выполнялась операция ~, затем &, а затем v, то в онтологии нужно ввести дополнительные грамматические типы булевых выражений и использовать их при определении шаблонов операций. Это реализовано в следующем примере онтологии "boolean".

```
    /*****************************************************/
/************************************************************/
/* Определения встроенного типа даных boolean */
/************************************************************/

False,True -элементы области boolean.
AND: boolean x boolean ->boolean.
OR : boolean x boolean -> boolean.
NOT : boolean ->boolean.
/* Таблица действия логических операций */

NOT(False)=True. NOT(True)=False.
AND(False; False)=False. AND(False; True)=False.
AND(True; False)=False. AND(True; True)=True.

OR(False; False)=False. OR(False; True)=True.
OR(True; False)=True. OR(True; True)=True.

/*синтаксические типы*/
bool_expr-область.
diz_expr < bool_expr.
con_expr < diz_expr.
not_expr < con_expr.
s_bool_expr < not_expr.
boolean < s_bool_expr.

/************************************************************/
/* Шаблоны инфиксной записи логических операций */
/************************************************************/
Введем шаблон "@u & @v"
с переменными:
"u : con_expr; v : not_expr"
и переменной результата " w : con expr " ;
Пояснения: [операция конъюнкции]
Условие применения шаблона:[]
Действие шаблона:
```

```
[w=AND(u;v)]
Тип доступа шаблона: [наследуемый].

Введем шаблон "@u V @v"
с переменными:
"u : diz_expr; v : con_expr"
и переменной результата " w : diz_expr " ;
Пояснения: [операция дизъюнкции]
Условие применения шаблона:[]
Действие шаблона:
[w=OR(u;v)]
Тип доступа шаблона: [наследуемый].

Введем шаблон " ~@u "
с переменными:
"u : not_expr"
и переменной результата " w:not expr " ;
Пояснения: [операция отрицания]
Условие применения шаблона:[]
Действие шаблона:
[w=NOT(u)]
Тип доступа шаблона: [наследуемый].

Введем шаблон "( @u ) "
с переменными:
"u : bool_expr"
и переменной результата " w:s_bool_expr " ;
Пояснения: [операция отрицания]
Условие применения шаблона:[]
Действие шаблона:
[w=u]
Тип доступа шаблона: [наследуемый].

    /*****************************************************/
```

Примером вопроса к данной онтологии является выражение:
'True v ~False&False == True?'.
(Ответ: "Да".)

8.5. Задачи на равномерное движение

В этом разделе рассматривается пример введения понятия равномерного движения и его использования при описании онтологий задач.

Онтология "равномерное движение" представляется в следующем виде:

```
    /*****************************************************/
Путь, скорость, время: real.

Путь= скорость*время.
Время= путь/скорость.
Скорость= путь/время.
```

```
Введем шаблон "@Тело движется равномерно"
с переменными: "Тело: new"
и переменной результата " x: команда " ;
Пояснения: [Вводится объект @Тело, движущийся равномерно]
Условие применения шаблона:
[]
Действие шаблона:
[x=пустая команда;
тело - объект понятия "равномерное движение". ]
Тип доступа шаблона:[внешний].

Введем шаблон "@Тело равномерно движется со скоростью @V"
с переменными: "Тело: new; V: real выражение"
и переменной результата " x: команда " ;
Пояснения: [Вводится объект @Тело, равномерно движущийся \ \n со
скоростью @V ]
Условие применения шаблона:
[]
Действие шаблона:
[x=пустая команда;
тело - объект понятия "равномерное движение";
тело's скорость =V. ]
Тип доступа шаблона:[внешний].

    /*****************************************************/
```

В первых четырех строках вводятся переменные параметры, являющиеся характеристиками равномерного движения, которые связываются уравнениями равномерного движения.

Далее вводятся два внешних к этой онтологии шаблона, использование которых в других онтологиях позволяет создавать в них экземпляры объектов равномерного движения, обладающих характеристиками равномерного движения, связанными уравнениями равномерного движения.

Рассмотрим примеры использования этой онтологии при описании задач.

Текст задачи1 имеет вид:

```
    /*****************************************************/
Пешеход равномерно движется со скоростью 5. Пешеход's
время =2.
Велосипедист равномерно движется со скоростью 6*пешеход's
скорость. Велосипедист's время = 3*пешеход's время.

/*Чему равно велосипедист's путь?*/.

    /*****************************************************/
```

По тексту задачи система строит внутреннее представление онтологии задачи. Заметим, что в тексте задачи используется шаблон языковой конструкции, введенный в онтологии "равномерное движение".

В ответ на представленный здесь вопрос система ответит:

"Вопрос: Чему равно велосипедист's путь? Ответ: 180."

8.6. Факториал

Рассмотрит онтологию "Факториал".

В этом примере определяется рекурсивно функция n!. Для этого вводится шаблон "fact(@n,@i,@y)" локальной функции, рекурсивно накапливающей результат. Далее вводится внешний шаблон "@n!", действие которого определяется через ранее определенную локальную функцию fact(n,i,y).

Текст онтологии "Факториал" имеет вид:

```
    /*****************************************************/
Введем шаблон "fact(@n,@i,@y)"
с переменными:
"n,i,y :real_выражение"
и переменной результата " z:real_выражение " ;
Пояснения: [Вычисляет факториал]
Условие применения шаблона:
[]
Действие шаблона:
[
if n==i then z=y else z=fact(n,i+1,(y)*(i+1))
]
Тип доступа шаблона:[локальный].

Введем шаблон "@n!"
с переменными:
"n :real"
и переменной результата " z:real " ;
Пояснения: [Вычисляет факториал]
Условие применения шаблона:
[]
Действие шаблона:
[
z=fact(n,1,1)
]
Тип доступа шаблона:[внешний].
    /*****************************************************/
```

Примеры вопросов к онтологии:
"Вопрос: 5! ?
Ответ: 120";
"Вопрос: 132! ?
Ответ: 1.118248651E+224".

Заметим, что, так как эта онтология разработана в среде ядра, и шаблон "@n!" был определен как внешний, то этот шаблон будет видим из любой онтологии (если он в этой онтологии не удален) и, следовательно, в этих онтологиях теперь может вычисляться факториал по запросу.

Другой способ описать функцию факториал с помощью правил переписывания термов приводится в онтологии "FactorialRul2":

```
    /*****************************************************/
FactorialRul: real_выражение->real_выражение.
FactorialRul(1)=1.

Для всех [n, k: real ]
[ (n)*(k) ] => [n*k] ("произведение").

Для всех[n: real_выражение ],
если[ n>1 ],
то [ FactorialRul(n)] => [(FactorialRul(n-1))*(n)] ("шаг
факториала"),
если, кроме того,[].

Для всех [n, k, m: real выражение ]
[((n)*(k))*(m)] => [ (n)*((k)*(m)) ] ("ассоциативность").

/*Примеры вопросов:
" Вопрос: FactorialRul(5)?
Ответ: 120"
"Вопрос: FactorialRul(3+2*3)?
Ответ: 362880"*/.
    /*****************************************************/
```

Здесь вводится имя функции и три правила переписывания для ее вычисления.

Глава 9. Алгебраические модели онтологий. Категорные операции

В предыдущих главах мы уже обсуждали, что при определении онтологии пользователь задает имена элементов, классов, функций и операций. Все имена доступные из данной онтологии называются *сигнатурой онтологии* (обозначениями онтологии).

Правильно построенные простые и составные выражения из имен называются термами онтологии.

Кроме имен, пользователь в онтологии задает равенства термов и правила переписывания. Все доступные из данной онтологии равенства и правила переписывания называются *соотношениями онтологии.*

Соотношения онтологии позволяет определить на множестве замкнутых термов (термов без переменных) данной онтологии наименьшее отношение эквивалентности $\approx$, удовлетворяющее свойству конгруэнтности и свойству конкретизации соотношений, где

- (свойство конгруэнтности) если F – имя операции, $F(t_1, \ldots, t_s)$ – правильно построенный терм и $t_1 \approx t'_1, \ldots, t_s \approx t'_s$, то $F(t_1, \ldots, t_s) \approx F(t'_1, \ldots, t'_s)$;
- (свойство конкретизации соотношений) если $L(x_1, \ldots, x_s) = R(x_1, \ldots, x_s)$ – соотношение онтологии и $L(t_1, \ldots, t_s)$, $R(t_1, \ldots, t_s)$ – правильно построенные термы, то $L(t_1, \ldots, t_s) \approx R(t_1, \ldots, t_s)$.

Алгеброй A, соответствующей данной онтологии называется фактор множество всех замкнутых термов данной онтологии по отношению эквивалентности $\approx$. Алгебра A является идеальным объектом, отражающим все знания, заключенные в данной онтологии. В реальности в общем случае алгебра A нам недоступна, и в системе отражаются, доступные в данный момент знания об онтологии. Это аппроксимация этой

алгебры, содержащая доступные нам термы онтологий и эквивалентности между ними. В простейшем случае это термы и соотношения, непосредственно используемые в определении онтологии. Однако у пользователя есть возможность расширять аппроксимацию онтологии, представляя в онтологии доказательства некоторых утверждений о мире данной онтологии. Тогда эти утверждения могут быть использованы в вычислениях в онтологии.

На вопрос «Чему равно Term?» система выдает Term1, эквивалентный в данной аппроксимации терму Term, и самый простой среди эквивалентных термов в данной аппроксимации. Такие термы называются *дескрипторами* в аппроксимации. Добавляя к онтологии предложения с вводом новых терминов, операций и соотношений, мы меняем аппроксимацию, добавляя новые термы в аппроксимацию и изменяя множество дескрипторов. В системе на каждом шаге сохраняется внутреннее представление онтологии, в котором отражается текущая аппроксимация онтологии. Если при разборе текста система встречает правильно построенный терм, которого нет в аппроксимации, но подтермы уже есть в аппроксимации, то система добавляет этот терм с аргументами-дескрипторами в качестве нового дескриптора в аппроксимацию. Если по тексту аппроксимации должно быть выполнено приравнивание двух термов, то система вычисляет дескрипторы для каждого из двух термов в существующей аппроксимации и выбирает из полученных дескрипторов наиболее короткий в качестве дескриптора *d1* в новой аппроксимации, а второй дескриптор *d2* всюду в аппроксимации заменяется на выбранный дескриптор *d1*. После этого пара *(d2, d1)* помещается в таблицу аппроксимации *result_op* в виде *result_op*(*d2*,*d1*). Эта таблица используется при вычислениях и, в частности, при вычислении термов. Если в процессе вычисления будет появляться подтерм *d2*, то он будет заменяться на дескриптор *d1*.

Отличие классов от множеств, используемых в онтологиях, состоит в том, что классы не определяются своими элементами, известными в данной онтологии. Более того, для общих отологий, как правило, вводятся классы, элементы которых в данной онтологии не определяются. Эти элементы будут вводиться в других онтологиях, использующих данную онтологию. Причем в разных онтологиях один и тот же класс может иметь разные наборы известных элементов. Поэтому построение теоретико-множественных конструкций из классов, таких как декартово произведение классов и многих других, исходя из их элементов классов, невозможно. А так как для выразительности языка описания онтологий нам хочется пользоваться аналогами всех теоретико-множественных конструкций для классов, то мы вынуждены обратиться к определениям этих конструкций, введенных в теории категорий. В теории категорий были придуманы и исследованы аналоги теоретико-множественных конструкций, определение которых дается через условия на классы (объекты) и функции (морфизмы). В нашем случае это будут классы и морфизмы алгебраической модели онтологии.

В онтологию ядра системы включены шаблоны и соотношения, связанные с этими шаблонами, которые определяют основные теоретико-категорные операция. Композиции этих операций позволяют строить аналоги всех теоретико-множественных конструкций над классами и функциями.

Перечень категорных операций ядра системы ЭЗОП довольно велик, порядка 60 шаблонов. Для подробного их изложения и приведения примеров их использования потребуется отдельная книга. В этой главе будет приведено только несколько примеров.

В предыдущих главах уже приводились и использовались шаблоны категорных операций *dom(@F), cod(@F), @F:@T1 -> @T2, com(@F,@G), @F*@G, id(@T),* которые задают область определения функции, область значения функции, функцию с заданными областями значений и определений, композицию функций и тождественную функцию для заданного класса, а также соотношения между этими операциями, которые, собственно, делают алгебру каждой онтологии системы категорией.

Для того, чтобы в онтологии построить декартово произведение классов A и B в ядре системы есть шаблон @T1 x @ T2. По выражению A x B система строит в аппроксимации новый класс и две функции pr1(A;B): A x B -> A и pr2(A;B): A x B -> B. Если f: C ->A и g:C -> B – пара функций онтологии, то шаблон <@f1,@f2> позволяет построить новую функцию <f,g>: C -> A x B, которая удовлетворяет уравнениям: com(pr1(A;B), <f,g>)=f и com(pr2(A;B), <f,g>)=g. Кроме того, если h: C -> A x B – функция онтологии, то в онтологии вводится соотношение h=<com(pr1(A;B), h), com(pr2(A;B), h)>.

Если *a* – элемент класса A и *b* – элемент класса B в онтологии, то шаблон *@el1 ; @el2* позволяет построить элемент в декартовом произведении классов в виде *a;b*, который в онтологии делается элементом класса A x B.

Если A и B – классы онтологии, то шаблон @T1 => @ T2 позволяет в онтологии строить новый класс A => B, элементами которого являются функции из класса A в класс B. Шаблон *lambda(@F,@T;@T1)* позволяет по отображению p:A x B ->D в онтологии построить отображение *lambda(p,A;B):*B -> (A=>D). При этом операция *lambda* устанавливает биекцию между множеством функций из A x B в D и множеством функций из B в A=>D.

Если f: C ->A – функция в онтологии и M – подкласс класса C, то шаблон *Im @функция(@подобласть)* позволяет строить подкласс Im f(M) в A – образ подкласса M относительно отображения f. При этом в онтологии вводятся соответствующие аксиомы для образа функции.

Если *a* делается в онтологии элементом классов A и B, то он становится элементом класса A п B – пересечения классов A и B, построенного с помощью шаблона *@область* п *@область*.

В целом в ядре системы должна поддерживается система операций и соотношений, позволяющая рассматривать алгебру каждой онтологии системы как алгебраический топос. В теории топосов известно, что в топосах могут быть реализованы аналоги всех теоретико-множественных конструкций. Эта система операций и соотношений, а также примеры использования этих операций должны быть подробно описаны в отдельном руководстве.