Е.М. Бениаминов, В.А. Лапшин

Уровни представлений онтологий, языки, математические модели и проект Веб-сервера онтологий в стиле Веб 2.0[1]

Рассматривается проект Веб-сервера онтологий в стиле Веб 2.0, предоставляющий возможности коллективной разработки онтологий. Обсуждаются уровни представления онтологий и математические модели, лежащие в основе этих представлений. Приведен обзор языка алгебраических спецификаций CASL.

Ключевые слова: *онтология, уровни представлений онтологий, Веб-сервер онтологий, алгебраические спецификации, язык CASL*

1. Введение

В последние годы в Интернет стали очень популярными системы, которые поддерживают процессы объединения людей по интересам, так называемые социальные сети. В таких системах пользователям предоставляется электронная среда в сети Интернет для организации объединения пользователей, для ввода разнородных данных и доступа к данным, а наполнение системы данными производится самими пользователями. Технология создания таких систем получила название технологии Веб 2.0 [1].

Для каждого объединения людей в таких системах определяется область интересов, лежащих в основе их объединения и целей объединения. Степень точности описания области интересов может быть различной и иметь в различной степени развитую онтологию и словарь, которыми владеют участники группы.

Онтология [2, 3, 4] — это формальное описание договоренности группы людей о том, что и как у них называется, какими свойствами может обладать, в каких отношениях участвует и каким ограничениям удовлетворяет. Онтология группы может уточняться и, как правило, уточняется в процессе объединения людей. Более того, по мере определения онтологии предметной области социальной группы, в этой группе формируется свой язык, на котором выражается данная онтология, и выражаются способы ее использования. Степень формальности такого языка зависит от традиций, особенности предметной области, лежащей в основе интересов данной группы, ее близости к общеупотребимой — бытовой области знаний, а также от желания и возможности использования в данной области компьютерных моделей.

В этой работе обсуждаются средства, которые могут предоставляться пользователям, для компьютерной поддержки процессов коллективного формирования онтологий и языка социальной группы.

[1]Работа выполнена при финансовой поддержке РФФИ (проект 09-07-00079-а).

Развитие этих средств шло в следующем направлении. На некотором этапе развития технологии Веб 2.0 стала понятной потребность пользователей в классификации вводимых ими материалов. Для реализации этой потребности в технологию Веб 2.0 были введены средства формирования и использования таксономий[2]. Пользователям в таких системах была обеспечена возможность определять собственные элементы таксономии и использовать их для разметки, вводимых материалов (страниц, фото, видео клипов и т.д.), а также для разметки на формируемых ими страницах ссылок на другие страницы посредством произвольно выбираемых меток, называемых тегами. Такая практика спонтанного сотрудничества группы людей с целью совместной категоризации информации получила название *фолксономии* [5].

Вместе с тем, пользователям стали предоставлять языки использования таксономий и разметок по таксономиям материалов и ссылок для поиска и отображения материалов. Эти языки представляются либо в строгом формате, который пытаются довести до некоторого стандарта, либо предоставляются пользователям в виде графических интерфейсов.

Одновременно с этим, в рамках направления *семантическая паутина* [6], развивалось понимание важности использования формальных моделей онтологий для организации связей между данными в Интернет и построения языков представления онтологий. Использование онтологий в системах стали получать все более широкое распространение. В связи с этим большое значение стали иметь работы по стандартизации языков представлений онтологий и построение онтологий, разделяемых большим числом пользователей. Однако большие онтологии обычно являются коллективной разработкой и складываются в процессе согласований и отладки. Поэтому возникает потребность в программных средствах поддерживающих, коллективную разработку онтологий. Таких систем пока немного[3]. Наиболее развитой среди них является система *Ontolingua* [8, 9], разработанная в Стенфордском университете в лаборатории систем знаний (KSL).

В данной статье описывается проект разработки системы в стиле Веб 2.0 для коллективного формирования баз онтологий и языков работы с ними. В проекте предлагается разработка Веб-сервера в стиле Википедия [10], но для онтологий. То есть предполагается создать среду в Интернет, в которой сами пользователи коллективно наполняют систему онтологиями и создают библиотеки онтологий по различным областям знаний. Онтология представляет собой фиксацию на формальном открытом языке договоренностей пользователей о том, что как называется в их области, какими свойствами обладает и каким ограничением удовлетворяет. Предполагается, что открытый формальный язык, на котором описываются онтологии, также коллективно формируется

---

[2]Таксономия является простейшим типом онтологий, в которых ключевыми элементами являются понятия класса, элемента класса, свойств класса и элементов, отношений класс-подкласс.

[3]Представляется достойным внимания подход, используемый в проекте Semantic Wiki [7]. Целью проекта Semantic Wiki является создание инструмента, позволяющего пользователям производить семантическую разметку добавляемых в Вики текстов.

пользователями по мере пополнения системы онтологиями и шаблонами языка, введенными в онтологиях. В системе должна поддерживаться модульность, многоверсионность разработки онтологий, управление доступом к версиям онтологий и словарям системы, а также средства поиска и отладки онтологий и языка системы. Онтологии, хранящиеся на разрабатываемом Веб-сервере, предназначены для многоразового многоцелевого компьютерного использования в задачах модульного построения формальных информационных моделей сложных объектов и их анализа.

Особое значение для коллективной разработки онтологий является модульная организация сложных онтологий и использование операций над онтологиями. Богатый опыт по модульной организации спецификаций накоплен в алгебраическом подходе к спецификации онтологий. В данной работе обсуждаются основы этого подхода и принципы организации алгебраического языка спецификаций CASL (Common Algebraic Specification Language) [11], который был разработан сообществом CoFI (Common Framework Initiative for algebraic specification and development) [12].

Для эффективного использования онтологий требуется многоуровневое их представление и использование специфических языков и средств на каждом уровне. Уровням представления онтологий посвящен следующий раздел.

2. Уровни и языки представлений онтологий

Использование онтологий в компьютерных системах связано с несколькими уровнями представления онтологий.

2.1. **Уровни представления онтологий**

Ниже описаны три уровня представления онтологии.

Верхний уровень. Это уровень представления пользователей. На этом уровне онтологии представляются для понимания пользователями и для формирования новых онтологий. В настоящее время этот уровень в основном отображается в графических редакторах онтологий типа Protege [13] или KAON [14]. Однако сложные аксиомы онтологий в этих редакторах приходится задавать в логических языках, которые пользователи должны изучать, если хотят пользоваться этим инструментом, и никак не могут подстроить его для своей предметной области. Жесткость и чуждость таких языков представления онтологий для специалистов в предметных областях, далеких от программирования и математики, является основной проблемой, препятствующей массовому использованию этих технологий. В основном этими технологиями пользуются программисты для создания приложений, которые уже разрабатываются для конкретных задач пользователей.

Средний уровень. Это уровень представления онтологий программистами и уровень межмашинного обмена онтологиями. Этот уровень развивался очень бурно. В настоящее время на этом уровне имеются стандарты языков представления — OWL 2.0 [15], языков запросов SPARQL [16] и языков описания ограничений SWRL [17] и RIF [18]. Этот уровень

описан во многих докладах и работах, поэтому здесь он особо обсуждаться не будет.

Нижний уровень. Это уровень представления онтологий в виде набора простейших фактов из бинарных отношений (троек). Для этого уровня разработан язык представления RDF [19]. Язык запросов SPARQL обращается к уровню представления онтологий на языке RDF.

Для работы с онтологиями на уровне выполнения запросов и отображения онтологий, а также для организации взаимодействия онтологий с базами данных, используется загрузка онтологий, представленных в виде RDF, в программную среду в виде RDF-хранилищ. Для этого используются специальные программные средства типа JENA [20], или SESAM [21], или программные средства СУБД ORACLE [22], а также некоторые программные среды логического вывода.

2.2. **Отображения уровней**

Взаимные отображения между уровнями представлений онтологий не всегда однозначны и полны.

Отображение с верхнего уровня на уровень OWL и обратно стараются полностью автоматизировать. Однако некоторые ограничения, сформулированные на уровне пользователя, могут представляться в OWL в виде комментария и не использоваться при компьютерной обработке.

Язык OWL может оказаться недостаточным для отображения пользовательских ограничений. Для расширения языка ограничений язык OWL дополняется различными версиями языков правил SWRL или RIF, выражения которых позволяют генерировать новые факты онтологий из существующих.

Отображение с уровня OWL на уровень RDF также может быть неоднозначным и определяться конкретным программным средством. Например, если некоторое отношение в OWL объявлено транзитивным, то на RDF уровне могут быть отображены все факты транзитивного замыкания исходного отношения, а могут быть отображены исходные факты и правило транзитивного замыкания в программной среде RDF- хранилища.

В общем случае пользовательское определение онтологии, заданное словарем онтологии и ограничениями (аксиомами) может задать неразрешимую теорию. Это значит, что в такой онтологии не существует общего алгоритма для построения ответов на все вопросы. Это значит, что система, использующая такую онтологию, не сможет ответить на все вопросы, которые мы сможем сформулировать к ней, и теория, которая соответствует такой онтологии, не является полной. Неполнота теорий, соответствующих общим онтологиям является их специфическим качеством[4].

Однако для каждой онтологии и уровня подробности ее представления в виде RDF-хранилища определяется фрагмент теории (подмножество запросов), для которой в данной программной среде система эффективно строит ответы. Такой фрагмент можно назвать *вычислительной*

[4]В системах, использующих общие онтологии, данные онтологий дополняются более подробной информацией из приложения.

аппроксимацией данной онтологии. Таким образом, помимо онтологии возникают вычислительные аппроксимации онтологии, определяющие возможности использования онтологии в данной вычислительной среде, но должен быть определен и идеальный объект, приближением которого являются эти аппроксимации.

Аппроксимация зависит от потребностей пользователя, от представления онтологии в RDF-хранилище, программной среды, в которой создано хранилище, и от представленных правил и, наконец, от уровня знаний логических выводов в данной онтологии. Правила выражают ограничения (аксиомы) данной онтологии, но, кроме того, могут также представлять достигнутый уровень процедурных знаний в этой онтологии. Совокупность всех аппроксимаций онтологии представляет собой направленное множество вместе с отношением, отражающим, что одна аппроксимация содержит другую (является более точной и отвечает на большее множество запросов). В пределе аппроксимации онтологии приближаются к идеальной (в общем случае недостижимой и потенциально бесконечной) *математической модели* онтологии.

В среде конкретных онтологий пользователями могут быть придуманы особо эффективные алгоритмы для ответов на определенные запросы[5], и эти процедурные знания должны быть как-то отражены в вычислительных моделях. Следовательно, для эффективной работы с онтологиями мы должны иметь как средства для представления непроцедурных знаний о мире онтологии, так и средства представления процедурных знаний для эффективных вычислений в мире онтологий. Процедурные знания могут выражаться, например, в виде последовательностей запросов или правил переписывания.

Заметим, что языки запросов в онтологиях являются процедурными языками (в отличие от языка формул или вопросов). Пользуясь знаниями об онтологии, пользователи выражают информационную потребность, содержащуюся в вопросе на языке запросов, указывая последовательность действий в данной онтологии и последовательность выполнения подзапросов для получения ответа на вопрос.

2.3. **Уровень описания математической модели онтологии**

Среди уровней представления онтологий дополнительно следует выделить математический (логический) уровень для описания идеальной математической модели онтологии. На этом уровне моделируются потенциальные возможности средств онтологического моделирования, несколько абстрагируясь от языков представления онтологий и удобств пользователей. По существу, на этом уровне представляется семантика возможностей онтологического моделирования. На этом же уровне моделируются операции взаимодействия онтологий. К этому уровню относятся логические модели онтологий в виде дескриптивных логик, представление онтологий в исчислениях предикатов первого порядка и более высоких порядков. С математической точки зрения описание онтологии (ее декларативной части) есть последовательное определение сигнатуры

[5]Бывает и наоборот, для особых алгоритмов описывается онтология, в которой этот алгоритм работает.

логической теории и ее аксиом (то есть описание теории). Имея начальные понятия класса, элементов классов, отношения класс-подкласс, операций и предикатов и связывающие их аксиомы, последовательно вводят новые классы, элементы, предикаты и операции, а также связывающие их аксиомы, отражающие предметную область онтологии и системные конструкции, принятые или вводимые в данной предметной области.

Наравне с логическими моделями онтологий на этом уровне используются алгебраические модели, подходы и языки (алгебра и логика — взаимно дополняющие друг друга разделы математики). Эти подходы может быть не столь известны программистам, использующим онтологии. Они требуют более серьезную математическую подготовку, но в них накоплен опыт и достигнуты наиболее существенные результаты для построения идеальных математических моделей онтологий, приближениями которых являются вычислительные аппроксимации онтологий. В частности, здесь продуманы идеи модульной организации онтологий, параметрические онтологии, методы коллективной разработки онтологий [12].

Один из авторов этой работы давно пропагандирует алгебраические методы в теории баз данных и моделировании средств представления знаний. Уже более 10 лет этот курс читается в РГГУ для студентов специальности «Интеллектуальные системы в гуманитарной сфере», написан учебник этого курса [23]. В этой работе описывается алгебраический подход к моделированию онтологий.

Алгебраический подход к моделированию онтологий был положен в основу алгебраического языка спецификаций онтологий Common Algebraic Specification Language (CASL) [11]. Язык был разработан большой группой специалистов, применяющих алгебраические методы для моделирования, накопившие большой опыт и объединившихся в группу Common Framework Initiative (CoFI) [12] для построения стандарта алгебраического языка спецификаций.

3. Язык CASL

Этот раздел подготовлен на основе доклада А.И. Ильиной, студентки Отделения интеллектуальных систем в гуманитарной сфере РГГУ. Авторы выражают ей благодарность за квалифицированную работу.

Язык алгебраических спецификаций CASL (Common Algebraic Specification Language) был разработан сообществом CoFI (Common Framework Initiative for algebraic specification and development) в 1998 году, но работа над этим языком началась гораздо раньше и язык вобрал в себя достижения многих других алгебраических языков спецификаций.

В основе языка CASL лежит стандартизованный язык предикатов первого порядка. В настоящее время имеются различные расширения и подъязыки CASL, включая диалекты языка логики высшего порядка, модальной логики, OWL DL. В языке CASL имеются средства структурирования спецификаций, включающие использование модулей, инкапсуляции, объединение спецификаций, расширения и т.д.

Алгебраическим язык CASL называется потому, что моделями его спецификаций являются алгебраические системы. Этот язык включает

в себя тщательно отобранные идеи предыдущих языков алгебраических спецификаций и новые оригинальные идеи, чтобы можно было писать спецификации более четко, ясно и в то же время кратко. CASL сразу же привлек к себе всеобщее внимание и уже de facto рассматривается как стандарт. Стоит отметить, что CASL является языком представления, а не языком программирования.

CoFI руководствовались следующими принципами при создании языка спецификаций:

- все типы данных могут быть представлены алгебрами, то есть несущим множеством, операторами на этом множестве и аксиомами;
- абстрактные типы данных могут быть представлены классами таких алгебр;
- спецификации программ можно давать с помощью вводимых абстрактных типов данных и описания последовательности выполнения операций.

Исходя из этих соображений, было решено, что абстрактные типы данных и программы могут быть представлены алгебраическими методами. Таким образом, появился CASL — язык для спецификаций требований к программным продуктам. Позже он стал рассматриваться как язык описания онтологий, использующий модульную организацию, и стал применяться для описания сложных онтологий (см., например, [24] о применении CASL в разработке онтологии верхнего уровня[6] DOLCE).

Язык CASL состоит из следующих частей:

- Базовые спецификации. Сюда входят декларации переменных, определения, аксиомы. В спецификации могут быть определены сорта (классы), подсорта (подклассы), операции (частичные операции) и предикаты. Аксиомы базовых спецификаций являются аксиомами логики первого порядка.
- Структурные спецификации. Сложные спецификации могут быть легко построены из более мелких с помощью нескольких операций, определенных в языке CASL: объединения, расширения, переименования и др.
- Архитектурные спецификации. Служат для структурирования сложной модульной онтологии вплоть до реализации онтологии готового программного продукта.
- Библиотеки спецификаций, Это именованные коллекции именованных спецификаций.

3.1. **Базовые спецификации**

Значение или семантика базовых спецификаций, как правило, состоит из двух частей:

- сигнатуры Σ, соответствующей символам, введенным спецификацией (словарь онтологии);

[6]Онтологиями верхнего уровня принято называть такие онтологии, которые создаются для описания абстрактных понятий и используются затем в основном для согласования онтологий, описывающих конкретные предметные области. Более подробно см. в [4] раздел 2.2 и описание на [25].

- классу моделей сигнатуры Σ (class of Σ-models), соответствующих тем интерпретациям сигнатуры Σ, которые удовлетворяют аксиомам и ограничениям спецификации.

Спецификации CASL могут задавать:

- сорта (классы);
- подсорта (подлассы);
- операции;
- предикаты.

Сорта (sorts). Сорта представляют собой множество, которое называется несущим множеством (carrier set). Элементы несущего множества обычно являются абстрактными представлениями какого-либо типа данных: чисел, букв, списков и.т. д. Сорт в языке CASL является аналогом типа в языках программирования. В CASL также можно задавать составные сорта, например List[int] – список из целых чисел.

Подсорта (subsorts). Подсорта в CASL интерпретируются как вложенные (встроенные) типы. Подсорта могут быть и подмножествами сортов ($Nat < Int$), и вложенными типами ($Char < String$).

Операции (operations) Операции в CASL могут быть определены как полные или частичные. Операции состоят из имени и «совокупности параметров», в которой указывается количество и сорта аргументов, а также указание сорта результирующего значения операции. Если значение какого-либо параметра операции не определено, то результат этой операции также будет не определен. Операции без аргументов называются константами. Они интерпретируются как элементы несущего множества результирующего сорта. Раздел спецификации CASL, в котором вводятся операции в обозначаются словом *ops*.

Приведем пример:

```
ops min(x,y : Elem) : Elem = x when x < y else y;
max(x,y : Elem) : Elem = y when min(x,y) = x else x
```

Предикаты (predicates). Предикаты определяются так же, как и операции, только они не имеют сорта результирующего значения. Следует обратить внимание, что не стоит путать предикаты с операциями с результирующим сортом булевского типа. Предикаты, в отличие от операций, не могут иметь неопределенного значения (если аргумент предиката не определен, то предикат считается не выполненным). Предикаты используются для формирования атомарных формул, а не термов.

Приведем пример:

```
sort Elem
pred __<__ : Elem x Elem
pred  __? __(x,y : Elem) <=> (x < y ? x = y)
```

Имя предиката или операции может быть объявлено с различными наборами типов параметров в одной и той же спецификации. Такой случай называется перегрузкой (overloading). Например: перегрузка константной операции empty: для сорта list и для сорта set. Перегрузка предиката $<$: для Int и для $Char$.

Интерпретация логических связок и кванторов в языке CASL является стандартной. Аксиомы в CASL — это аксиомы логики первого порядка. Все логические связки и кванторы (с добавлением квантора) так же берутся из языка предикатов. Переменные в формулах могут иметь значения из несущего множества каждого соответствующего сорта.

Приведем пример спецификации «частичный порядок»:

```
spec Strict_Partial_Order =  %задали имя%
sort Elem  %задали сорт%
pred__<__: Elem x Elem %задали предикат «меньше»%
? x , y, z : Elem
. not (x < x )  %некоммутативность%
. x < y => not (y < x )  %ассиметричность%
. x < y and y < z => x < z %транзитивность%
%{стоит отметить, что могут существовать такие x и y,
такие что не верно ни x < y, ни y < x.}%
end
```

3.2. **Структурные спецификации**

Семантика структурных спецификаций та же, что и у базовых спецификаций. Структурные спецификации формируются из базовых с использованием разнообразных конструкций составления спецификаций.

Рассмотрим следующие основные способы:

- переименование;
- скрытие;
- объединение и расширение;

Переименование. Переименование используется для составления новых спецификаций из уже имеющихся. Это помогает избежать коллизии имен операций и предикатов и подчеркивает отличие смысла операций в новой спецификации. Для переименования в CASL есть оператор `|->`.

Приведем пример:

```
%На основе списка строится спецификация стек.%
spec Stack[sort Elem] =
List_Selectors [sort Elem]
with
sort List |-> Stack,
ops cons |-> push__onto__, head |-> top, tail |-> pop
end
```

Скрытие. Ненужные в новой спецификации операторы и предикаты можно скрыть. Для этого используется оператор hide.

Приведем пример:

```
Spec Square_Matrix =
Matrix hide num_of_columns
end
```

Объединение и расширение. Конструировать сложные спецификации можно, объединяя уже существующие, более мелкие спецификации. В

CASL это делается с помощью оператора and. Логично это делать одновременно с расширением, то есть новая спецификация расширяет возможности тех спецификаций, из которых она состоит. Для этого в CASL есть оператор `then`.

Приведем пример:

```
%{Построим спецификацию, объединяющую спецификации
список и множество.}%
Spec List_Set [sort Elem] =
List_Selectors [sort Elem]
and Generated_Set [sort Elem]
then op elements__of__ : List > Set
. e:Elem; L:List
. elements_of empty = empty
. elements_of cons(e, L) = { e } V elements_of L
end
```

3.3. **Архитектурные спецификации**

Семантика архитектурных спецификаций отражает их модульную структуру, то есть архитектурные спецификации описывают структуру системы в целом. Они состоят из блоков, которые определяют компоненты системы, и блока результата, который указывает, как компоненты должны быть скомбинированы.

Приведем пример:

```
%Простейшая архитектурная спецификация:%
arch spec System =
units
Unit1 : Specification1
...
UnitN : SpecificationN
result LINK(Uuit1, ..., UnitN)
```

`Unit1...UnitN` — имена блоков, представляющих собой спецификации `Specification1...SpecificationN`.

`LINK` — терм, определяющий что будет результатом этой спецификации. Часто результатом бывает лишь один из блоков спецификации.

Обычно `Specification1...SpecificationN` имеют общие части и используют операции и предикаты друг друга. Для этого в CASL есть оператор `given`, который указывает, что данная спецификация может быть реализована только тогда, когда реализована какая-либо другая спецификация. С его помощью можно разбивать сложные спецификации на несколько более простых.

Приведем пример:

```
arch spec Matr =
units
V : Vector
M : Matrix given V
MM: MatrixMult given M
result MM
```

3.4. **Библиотеки спецификаций**

Семантика библиотек спецификаций — это отображение имен спецификаций в их семантику, то есть библиотеки спецификаций представляют собой именованные коллекции спецификаций. Чаще всего библиотеки являются локальными (все используемые в библиотеке спецификации определены внутри этой библиотеки). Спецификации, которые были определенные позднее, могут использовать внутри себя те, что были определены до них.

Приведем пример:

```
library UserManual/Examples
...
spec Natural = ...
...
spec Natural_Order = Natural then ...
```

Библиотеки также могут включать в себя спецификации из других библиотек. Для этого в CASL используется слово `from`.

Приведем пример:

```
library Basic/StructuredDatatypes ...
from Basic/Numbers get Nat, Int ...
spec List [sort Elem ] given Nat = ...
...
spec Array ... given Int = ...
```

Для библиотеки можно определять инструкции синтаксического разбора, отображения символов, информацию об авторе и дате создания библиотеки.

Приведем пример:

```
library UserManual/Examples ...
%display union %LATEX V
%prec { + , ? } < { * }
%authors( Michel Bidoit <bidoit@lsv.ens-cachan.fr>)%
%dates 15 Oct 2003, 1 Apr 2000 ...
```

Директива `%display union %LATEX V` указывает на то, что оператор под именем union следует отображать как `V`. `%prec { + , ? } < { ? }` определяет приоритет операций и таким образом позволяет опускать скобки, и вместо $a - (b * c) + d$ просто писать $a - b * c + d$.

В настоящее время CASL имеет много диалектов (см. [11, раздел 1.7.2]), в основе которых лежит язык предикатов первого порядка, второго порядка, высших порядков, модальная логика, теория категорий. Предусматриваются подходы слияния языка CASL и наиболее современного языка функционального программирования Haskell [26], в основе которого лежат теоретико-категорные конструкции λ-исчисления.

4. Общее описание проекта построения Веб-сервера онтологий

В этом разделе описывается проект [27], разрабатываемый авторами этой статьи. В проекте предлагается построить Веб-сервер в стиле Википедия с использованием технологий Web 2.0 для библиотек онтологий.

Проектируемая система предназначена для коллективного формирования библиотек взаимодействующих онтологий и языков работы с ними. В соответствии с принципами Веб 2.0. система обеспечивает удобную среду в Интернет для объединения пользователей системы по интересам, распределению ролей между пользователями и формированию библиотек онтологий самими пользователями на формальном языке, который они формируют в процессе построения библиотеки.

Онтология, как упоминалось ранее, — это формальное описание договоренности группы людей о том, что и как у них называется, какими свойствами может обладать, в каких отношениях участвует и каким ограничениям удовлетворяет. Онтология группы может уточняться и, как правило, уточняется в процессе объединения людей. Более того, по мере определения онтологии предметной области социальной группы, в этой группе формируется свой язык, на котором выражается данная онтология, и выражаются способы ее использования. Степень формальности такого языка зависит от традиций и особенности предметной области, лежащей в основе интересов данной группы, а также от желания и возможности использования в данной области компьютерных моделей.

Первоначально пользователям системы предоставляется некоторая базовая онтология ядра системы и язык ядра, обеспечивающие пользователям возможность формировать произвольные онтологии и вводить новые конструкции языка, переопределять старые конструкции — подстраивать язык под конкретную проблемную область.

Далее, пользователи могут объявить любую построенную ими онтологию средой для построения новых онтологий. В этом случае онтология среды и конструкции языка, доступные из среды, становятся доступными для новых конструируемых онтологий.

В ядре системы имеются возможности для модульного объектного построения новых онтологий. В результате деятельности пользователей строятся библиотеки взаимосвязанных онтологий и словарей языков системы. В системе имеются средства, поддерживающие процессы обсуждения вариантов онтологий, построения и хранения черновиков онтологий, версий онтологий, тестирования и отладки онтологий, а также средства управления публикацией онтологий (доступностью онтологий для различных групп пользователей).

В системе вместе с текстом онтологии на построенном формальном языке хранится некоторая вычислительная модель онтологии, автоматически построенная системой на этапе проверки правильности построения текста онтологии (откомпилированный файл онтологии). На основании этой вычислительной модели в системе производится анализ правильности построения запросов к онтологии и построение ответов на запросы. В ядре онтологии имеются некоторые языковые конструкции, управляющие вычислительной моделью онтологии, позволяющие задавать вычисления по правилам переписывания выражений в данной онтологии.

Работа системы обеспечивается взаимодействием двух подсистем:

- Веб-сервера онтологий;
- Веб-приложения.

Веб-сервер онтологий обеспечивает многопользовательский режим работы системы, работу с пользователями, библиотеками онтологий и словарями. Для работы с конкретной онтологией или черновиком онтологии Веб-сервер вызывает Веб-приложение в соответствующем режиме и передает ему требуемые для работы в этом режиме параметры.

Веб-приложение выполняет все функции с отдельной онтологией или черновиком: анализирует запрос к онтологии, строит и выводит ответ запрос; анализирует текст черновика онтологии; строит и выдает сообщения об ошибках; строит вычислительную модель онтологии по тексту онтологии; результаты работы с онтологией отображает в формах пользователя и передает Веб-серверу для сохранения.

В настоящее время Веб-сервер онтологий разрабатывается с помощью программных средств технологии Drupal [28] для создания интернет-порталов. Веб-приложение разрабатывается на языке Пролог с помощью средств Visual Prolog 5.2 [29] и SWI-Prolog [30]. В основу Веб-приложения положены программы, разработанные на Visual Prolog 5.2 для системы представления знаний ЭЗОП, работающей под Windows [31].

Основное отличие проектируемой системы от существующих серверов онтологий (например, Ontolingua [8]) — это то, что в проекте предусматривается использование принципов открытого шаблонного языка для представления онтологий, позволяющего пользователям подстраивать язык представлений по мере пополнения библиотеки онтологиями и шаблонами пользователей к языку предметной области.

Для обмена проектируемой системы онтологиями с другими системами предполагается разработка модулей по загрузке и выводу онтологий на языках OWL и CASL.

Кроме того, существенной особенностью разрабатываемой системы является использование вместе с текстами онтологий откомпилированных по этим текстам внутренних представлений онтологий (вычислительных аппроксимаций онтологий). Внутреннее представление онтологии используется при семантическом анализе выражений языка, при формировании ответов на запросы к онтологии и ее отладке, при межмашинном обмене онтологиями в некотором стандарте и при использовании онтологий в приложениях. Внутренние представления онтологий предполагается строить с использованием современных алгебраических средств и модульной организации построения онтологий на основании подхода, близкого к подходу языка спецификаций CASL. Эти средства обладают достаточными возможностями для моделирования сложных онтологий и проведения вычислений на основе построенных моделей.

В соответствии с принципами Веб 2.0 авторы проекта не предполагают сами разрабатывать онтологии, а собираются разработать программную среду в Веб и ядро системы, в которой пользователям было бы удобно коллективно разрабатывать и отлаживать библиотеки онтологий и сопутствующие им языки. Однако некоторые примеры библиотек онтологий авторы предполагают разрабатывать совместно с некоторыми группами пользователей на этапе отладки системы и создания демонстрационных примеров.

Процесс формирования сложных и больших библиотек онтологий в настоящее время сдерживается сложностью формального языка представления онтологий. Сложность языка (несоответствие его языку прикладной области) ограничивает понимание этих текстов специалистами в прикладных областях и, следовательно, затрудняет процессы проверки, формирования и тестирования онтологий. На преодоления этих недостатков нацелен предлагаемый проект. Основой для решения перечисленных проблем предлагаются разработанные членами коллектива принципы открытого языка представления знаний. Эти принципы обеспечат многослойность представления онтологий:

- на пользовательском уровне онтологии представляются на языке, близком к языку предметной области;
- на уровне межмашинного обмена знаниями онтологии представляются на стандартном языке машинного представления онтологий;
- на внутреннем уровне для задач тестирования онтологий и построения ответов на вопросы к миру построенной онтологии используется алгебраический язык и средства алгебраических вычислений.

В настоящее время в соответствии с проектом создается экспериментальный сервер онтологий с использованием технологий Веб 2.0. На сервере можно будет зарегистрироваться. Пользователям могут быть присвоены различные роли. Система будет работать в многопользовательском режиме. Пользователи смогут искать документацию на сервере, просматривать ее, оставлять комментарии к документам, открывать новые форумы для обсуждений и участвовать в старых, а также открывать блоги для обсуждения работ над онтологиями.

Для формирования онтологии в открытом языке шаблонных выражений, ее отладки и построения ответов на вопросы к онтологии разработана специальная программа (Веб-приложение). Программа работает на сервере с интерфейсом в Веб-браузере и вызывается пользователем с сервера опциями:

- «Задать вопрос» для построения запроса на доступном из текущей онтологии языке, сформированном пользователями, и для получения ответов на запросы к онтологии. Ответы формируются на основании внутреннего представления онтологии, правил переписывания, введенных в онтологии, и алгебраических вычислений.
- «Сделать средой» и «Создать новую онтологию» для создания новых онтологий с возможностью сохранения недоделанных онтологий в виде черновиков.
- «Создать новую версию онтологии» для создания и отладки новых версий существующих онтологий.
- «Удалить онтологию», если она не используется другими онтологиями.

Взаимодействие сервера онтологий и программы работы с текущей онтологией находится в стадии отладки.

Сервер онтологий разрабатывается программными средствами системы управления сайтом Drupal 5 [28]. Система Drupal имеет большое количество модулей и удобную организацию для быстрой разработки серверов, использующих технологии Веб 2.0. В дальнейшем предполагается переход на более свежую версию CMS Drupal 6.

Разработаны программа грамматического анализа для открытого языка шаблонных выражений, на котором описываются онтологии и задаются вопросы к онтологии, написана на языке Пролог, программа вычисления ответов на запросы к онтологии также разрабатывается на языке Пролог. Основу этих программ составляет система (`http://ezop-project.ru/?q=WindowsEzop`) ЭЗОП под Windows [31, 9, 33]. Система ЭЗОП разработана программными средствами Visual Prolog 5.2.

Документация к проекту выставлена на специально разработанном средствами CMS Drupal 6 сайте [27]. На сайте документации приведен список примеров онтологий, разработанных в системе [34].

Основной онтологией системы является онтология с именем «Ядро системы». Эта онтология создана разработчиками системы. В онтологии ядра определены базовые типы, используемые при конструировании онтологий, а также шаблоны базового языка, предоставляемого системой пользователям. Кроме того, в языке ядра определены шаблоны для задания вопросов к онтологии. Наиболее интересные из них:

```
Элементы области @Области?
Подобласти области @Область?
Чему равно @выражение?
```

Здесь выражения, начинающиеся с символа `@`, представляют собой переменные, вместо которых могут подставляться выражения типа, заданного в шаблоне вопроса. Например, в шаблоне `Элементы области @Области?` переменная `@Области` имеет тип «область», который в системе является синонимом типа «класс». Результатом выполнения данного шаблона будет печать списка элементов конкретного класса, имя которого передано в качестве параметра в шаблон. Например, на вопрос `Элементы области материки?` в онтологии, описывающей географические объекты, будет дан ответ: `Евразия, Северная Америка, Южная Америка, Австралия, Антарктида`.

Также, на странице [34] представлены примеры иерархических онтологий (таксономий), спецификации булевой алгебры, онтологий, описывающих физические понятия (в частности, понятие равномерного движения), а также приведен пример спецификации вычислимой функции (онтология «Факториал»).

Онтология «Факториал» содержит грамматический шаблон, позволяющий задавать в текстах онтологий факториалы различных значений. Значение этого шаблона имеет тип `real`, а семантика состоит в вычислении значения факториала непосредственно в тексте онтологии. Ниже приведен текст онтологии «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].
```

В онтологии определяются два шаблона: `fact(@n,@i,@y)` и `@n!`. Первый шаблон вспомогательный и имеет внутреннюю область видимости. Второй шаблон `@n!` состоит, фактически, из параметра `@n`, который используется для передачи натурального числа с целью последующего вычисления значения факториала. В текстах онтологий выражение вида `5!` будет теперь трактоваться как факториал числа 5 и вычисляться соответствующим образом.

5. Заключение

В работе описан проекта построения Веб-сервера онтологий в стиле Веб 2.0. Целью создания проекта является разработка системы, позволяющей пользователям коллективно разрабатывать онтологии. Разработка такой системы требует решения целого ряда вопросов. Возникающая проблематика описана в данной работе.

В работе введено понятие об уровнях представления онтологий. Выделено три уровня представления: верхний, средний и нижний уровни. Верхний уровень доступен пользователям системы, создающим онтологии. На среднем уровне специалисты по разработке систем знаний пишут программы для обмена данными между различными системами. На нижнем уровне разрабатываются модели эффективного доступа к данным. Кроме того, выделяется уровень математической модели представления онтологии.

В статье приведено описание одного из вариантов уровня математической модели представления онтологии, основанного на алгебраическом подходе к моделированию онтологий. Онтологии, описанные с помощью этого подхода, обычно называют алгебраическими спецификациями. В

работе приведен обзор языка алгебраических спецификаций CASL, являющегося сейчас de facto стандартом для такого рода языков.

В работе не были подробно рассмотрены вопросы организации вычислений в онтологиях, построенных на основе алгебраического подхода. Рассматриваемый в работе проект построения Веб-сервера онтологий в стиле Веб 2.0 позволяет производить такие вычисления (см. примеры вопросов к онтологии «Ядро системы» выше). Рассмотрение вопросов, связанных с вычислениями в алгебраических онтологиях, требует отдельной статьи.

Список литературы

[1] Страница Веб 2.0 на сайте Википедия — http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1_2.0.

[2] Gruber T.R. Toward Principles for the Design of Ontologies used for Knowledge Sharing // Knowledge Systems Laboratory, Stanford University, Technical report KSL-93-21, August 23, 1993.

[3] Guarinio N. Formal Ontology and Information Systems // Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998, Amsterdam, IOS Press.

[4] Лапшин В.А. Онтологии в компьютерных системах. М.: Научный мир, 2010.

[5] Страница описания термина «фолксономия» — http://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D0%BB%D0%BA%D1%81%D0%BE%D0%BD%D0%BE%D0%BC%D0%B8%D1%8F.

[6] Страница Semantic Web на сайте Википедии — http://en.wikipedia.org/wiki/Semantic_Web.

[7] Страница Semantic Wiki на сайте Википедии — http://en.wikipedia.org/wiki/Semantic_wiki.

[8] Сайт проекта Ontolingua — http://www.ksl.stanford.edu/software/ontolingua.

[9] Бениаминов Е.М., Болдина Д.М. Система представления знаний Ontolingua - принципы и перспективы // НТИ Сер. 2, Информ. процессы и системы. 1999. №10.

[10] Свободная энциклопедия Википедия — http://ru.wikipedia.org.

[11] Bidoit M., Mosses P.D. CASL User Manual. Introduction to Using the Common Algebraic Specification Language. N.Y: Springer, 1998.

[12] Сайт проекта CoFI — http://www.informatik.uni-bremen.de/cofi/wiki/index.php/CoFI.

[13] Сайт проекта Protege — http://protege.stanford.edu.

[14] Страница проекта KAON. — http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1_2.0.

[15] Обзор языка OWL на сайте консорциума W3 — http://www.w3.org/TR/owl-features.

[16] Описание языка SPARQL на сайте консорциума W3 — http://www.w3.org/TR/rdf-sparql-query.

[17] Обзор языка SWRL на сайте консорциума W3 — http://www.w3.org/Submission/SWRL.

[18] Обзор языка RIF на сайте консорциума W3 — http://www.w3.org/TR/rif-overview.

[19] Страница языка RDF на сайте консорциума W3 — http://www.w3.org/RDF.

[20] Страница проекта JENA на сайте SourgeForge — http://jena.sourceforge.net/documentation.html.

[21] Сайт проекта Open RDF — http://www.openrdf.org.

[22] Страница Database Semantic Technologies на сайте компании ORACLE — http://www.oracle.com/technology/tech/semantic_technologies/htdocs/semtech_training.html.

[23] Бениаминов Е.М. Алгебраические методы в теории баз данных и представлении знаний. М.: Научный мир, 2003.

[24] Страница проекта DOLCE на сайте Laboratory for Applied Ontology — http://www.loa-cnr.it/DOLCE.html.
[25] Википедия об онтологиях верхнего уровня — http://en.wikipedia.org/wiki/Upper_ontology_(computer_science).
[26] Сайт языка Haskell — http://www.haskell.org.
[27] Проект экспериментального Веб-сервера онтологий — http://ezop-project.ru.
[28] Сайт проекта Drupal open source content management platform — http://drupal.org.
[29] Адаменко А., Кучуков А. Логическое программирование и Visual Prolog. М.:БХВ-Петербург, 2003.
[30] Сайт языка SWI-Prolog — http://www.swi-prolog.org.
[31] Бениаминов Е.М., Болдина Д.М. Система представления и обработки знаний ЭЗОП // Материалы конференции Диалог'2001. Прикладные проблемы, 2001.
[32] Бениаминов Е.М. Система представления и обработки понятий, основанная на алгебраическом (категорном) подходе // II Всесоюзн. конф. «Искусственный интеллект-90», Минск, 1990.
[33] Бениаминов Е.М., Манушина М.Ю. Принципы построения открытого языка шаблонных выражений в системе представления знаний // НТИ Сер. 2, Информ. процессы и системы. 2000. №7.
[34] Страница примеров онтологий на сайте проекта ЭЗОП — http://ezop-project.ru/?q=node/166.

Российский государственный гуманитарный университет
E-mail address: ebeniamin@yandex.ru, mefrill@yandex.ru