УДК 004.415

ПРЕОБРАЗОВАНИЕ OWL-ОНТОЛОГИЙ ДЛЯ ВИЗУАЛИЗАЦИИ И ИСПОЛЬЗОВАНИЯ В КАЧЕСТВЕ ОСНОВЫ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

П.А. Ломов, М.Г. Шишаев, В.В. Диковицкий

Институт информатики и математического моделирования технологических процессов Кольского научного центра РАН

lomov@iimm.kolasc.net.ru, shishaev@iimm.kolasc.net.ru, dikovitsky@iimm.kolasc.net.ru

Аннотация

В статье рассматриваются вопросы, связанные с упрощением применения онтологий, описанных на языке OWL, в качестве основы функционирования графических пользовательских интерфейсов. В частности предлагается создание промежуточного модели – онтологии пользовательского представления, для произвольной OWL-онтологии, ориентированной в первую очередь на непосредственную визуализацию в виде графовой структуры. Рассматривается структура такой модели, а также порядок отображения в ней аксиом исходной OWL-онтологии и их визуализация.

Ключевые слова: *онтология, OWL, SKOS, пользовательский интерфейс.*

Введение

На сегодняшний день использование онтологий является ключевым аспектом в семантическом представлении и обработке информации. Круг технологий, связанных с применением онтологических моделей, весьма широк и включает в себя мультиагентные системы, автоматическое извлечение знаний из текстов на естественном языке, поиск информации, интеллектуальное аннотирование и другие. Как правило, онтологии рассматриваются как машинопонимаемые спецификации предметной области или задачи, которые позволяют обеспечить использование единой системы понятий людьми и/или программными агентами и тем самым наладить диалог между ними. В этой связи очень привлекательным является также использование онтологий при разработке и функционировании пользовательского интерфейса [1-4], что так или иначе предполагает визуализацию совокупности элементов онтологии, как правило, в виде графовой структуры. Это дает возможность пользователю оперировать известными ему понятиями, осуществлять интуитивно понятную навигацию между ними и итеративно формировать сложный запрос, внося в него дополнительные объекты поиска, определяя ограничения и не используя при этом каких-либо искусственных поисковых языков. Пользователь также имеет возможность ознакомиться с предметной областью без совершения запросов, для более ясного виденья контекста того или иного термина, определяемого его структурным положением в онтологии.

В то же время большую популярность, благодаря богатым выразительным свойствам и вместе с тем формальной разрешимости, получил язык веб-онтологий (OWL, Ontology Web Language) [5]. Предложенный и развиваемый консорциумом W3C, OWL на сегодняшний день является де-факто стандартом описания онтологий для их использования в Интернет. В его основе лежит дескрипционная логика ALC (Attributive Language with Complement) [6], что позволяет формально описывать понятийные системы задач и предметных областей таким образом, что такой спецификацией могут оперировать как люди, так и программные агенты. Однако онтологии, описанные с помощью данного языка (OWL-онтологии), являют-

ся, по сути, системой логических утверждений (аксиом) дескрипционной логики. В этой связи их визуальное представление в виде графовой структуры далеко не всегда тривиально и требует от разработчика программных систем понимания формальной семантики аксиом, того что неявно из них следует, а также специальных приемов онтологического моделирования. В целом эти требования можно свести к необходимости так называемого понимания онтологии (ontology comprehension) [7, 8], что в свою очередь может потребовать существенных усилий со стороны программиста, а также его взаимодействия с автором онтологии.

Преодолеть трудности, связанные с визуализацией OWL-онтологий, можно двумя способами. Первый заключается в упрощении OWL-онтологии при ее визуальном представлении, то есть учет только тех аксиом, которые могут быть представлены в виде вершин и ребер графа. Такой способ применяется в большинстве современных программных средств графического представления онтологий GraphViz [9], TGVizTab [10], OWLViz [11], OntoSphere [12]. Второй способ, напротив, заключается в усложнении языка графического представления с целью повышения его выразительных возможностей до уровня OWL, что позволило бы сопоставить каждому элементу последнего некоторый визуальный образ. Однако, с учетом того обстоятельства, что OWL-онтология может использоваться как основа пользовательского графического интерфейса, применение первого способа может повлечь существенную потерю формальной семантики в графическом представлении и, как следствие, невозможность создания с помощью него сложных запросов к исходной OWL-онтологии. Недостаток второго способа связан с излишним усложнением визуального представления, что затруднит его понимание пользователем. Таким образом, разработчик интерфейса в ходе визуализации должен правильно выбрать «золотую середину», что также может потребовать существенных затрат его труда и времени.

В данной работе рассматривается проблема снижения трудоемкости задачи визуального представления OWL-онтологии произвольной сложности для ее использования в качестве основы графического пользовательского интерфейса (ГПИ) информационных систем или веб-ресурсов. Для этого предлагается создавать на основе исходной OWL-онтологии ее модификацию - онтологию пользовательского представления (ОПП), которая может быть использована разработчиком интерфейса без предварительного анализа исходной онтологии. Рассматривается структура ОПП, а также порядок представления в нем системы аксиом OWL-онтологии.

1 Онтология пользовательского представления

Для более ясного понимания дальнейшего материала следует уточнить, что подразумевается под онтологией в данном случае. Так как речь идет об онтологических моделях, описываемых с помощью языка OWL, основанного на дескрипционной логике, то наиболее полно их суть, с точки зрения авторов, отражает определение Н. Гуарино [13]: «Онтология – это логическая теория, явно частично отражающая подразумеваемое значение формального словаря». Формальный словарь образует множество классов или концептов, обозначающих понятия предметной области, и отношения (как правило, бинарные), определяемые между классами. С каждым классом и отношением сопоставляется функция интерпретации, отображающая класс во множество объектов реального мира, а отношение - в декартовое произведение этого множества на самого себя (рисунок 1). В OWL функции интерпретации задаются не явно, а через утверждения дескрипционной логики. Последние в свою очередь формально ограничивают множества интерпретации классов и отношений и, тем самым, частично определяют их семантику.

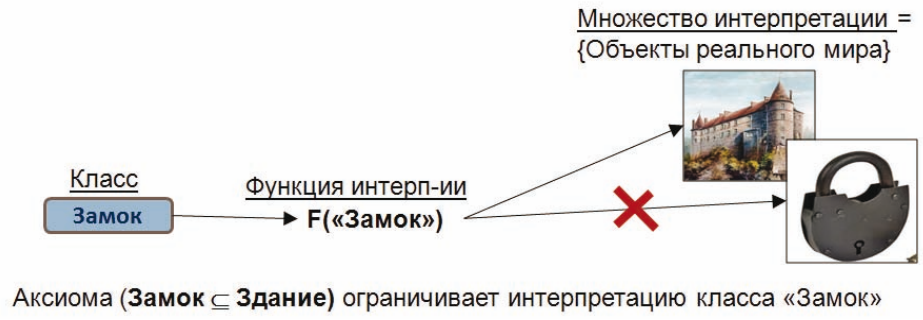

Рисунок 1 – Определение понятий через ограничения их множеств интерпретации аксиомами

Таким образом, OWL-онтологию можно рассматривать как совокупность утверждений дескрипционной логики, задающих ограничения на множества интерпретации классов и отношений. Полагается, что эти утверждения, ввиду того, что формулируются экспертом предметной области, верно отражают смысл понятий, поэтому их называют аксиомами. А сама онтология, которую они составляют, является логической теорией.

Среди основных аксиом, определяющих формальную семантику понятий, можно выделить:

- аксиомы тождественности (equivalent-to), далее обозначаются знаком «≡». Формально они определяют равенство множеств интерпретаций классов. Например, ЭВМ ≡ КОМПЬЮТЕР, декларирует то, что любая сущность, отнесенная к классу «ЭВМ», будет также отнесена к классу «КОМПЬЮТЕР» и наоборот;
- аксиомы наследования (subclass/superclass), далее обозначаются «⊆» и «⊇». Определяют вхождение множества интерпретации одного класса (подкласса) во множество другого (суперкласса). Например, «ЭВМ» ⊆ «ФИЗИЧЕСКИЙ-ОБЪЕКТ» означает, что если сущность принадлежит к классу «ЭВМ», то она также будет отнесена и классу «ФИЗИЧЕСКИЙ-ОБЪЕКТ», но не наоборот;
- аксиома разделения (disjoin) классов, указывает на то, что множества интерпретации классов не пересекаются. Например, аксиома «ЭВМ» disjoint «МАШИНА» означает, что если сущность будет причислена к классу «ЭВМ», то она не будет входить в класс «МАШИНА» и наоборот;
- аксиома перечисления (enumeration), позволяет задать класс экстенсионально, то есть путем перечисления всех его экземпляров. Это предполагает, что данный класс не содержит других экземпляров, кроме указанных непосредственно.

Для того чтобы использовать какую-либо OWL-онтологию в программных разработках или исследованиях следует ясно представлять интерпретации ее классов и отношений. В противном случае любая ее модификация может привести к появлению противоречивых аксиом и, как следствие, возможности получения неверных результатов ее логического анализа и ошибкам или сбоям в работе основанных на ней программ. В свою очередь проблема понимания онтологии (ontology comprehension), как это отмечено в работе [7], является на данный момент достаточно новой и требует дальнейшего исследования.

Для использования OWL онтологии для ГПИ предлагается формировать на ее основе более простую для понимания онтологию пользовательского представления - ОПП. ОПП должна быть изоморфна исходной онтологии в следующем (не строгом математическом) смысле: с одной стороны ОПП должна включать элементы исходной онтологии, существенные для визуализации в рамках ГПИ, и быть достаточно простой, для того чтобы ее визуализация являлась чисто технической задачей, с другой – набор ее элементов должен позволять формировать не только простые (поиск подклассов/экземпляров класса), но и вложенные за-

просы к исходной онтологии. Наряду с этим в нее также можно включить элементы, позволяющие отразить информацию о поисковых потребностях пользователя, и на ее основе определить предпочтительный способ визуального представления понятийной системы предметной области. Это позволит должным образом персонифицировать интерфейс и тем самым повысить удобство работы с ним конечного пользователя (рисунок 2).

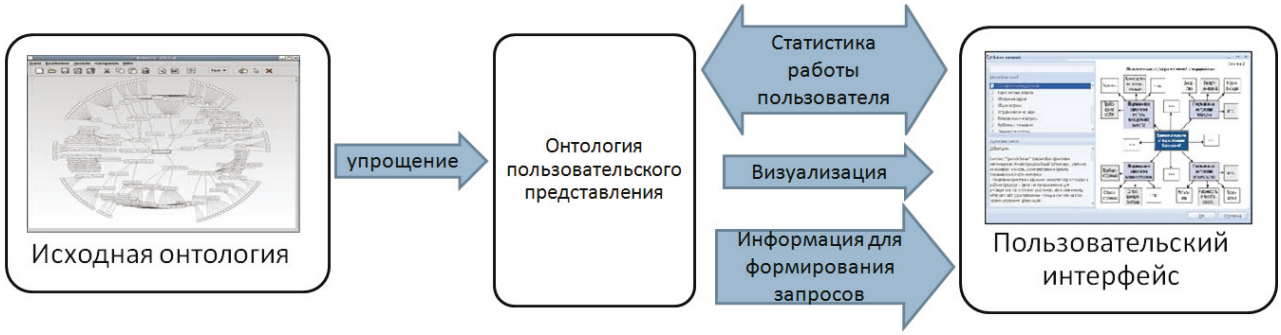

Рисунок 2 – Применение онтологии пользовательского представления

Разработку ОПП имеет смысл производить не «с нуля», а взяв за основу какую-либо из уже существующих моделей представления знаний. Это облегчит создание ОПП, а также позволит при оперировании ее элементами использовать существующие для выбранной в качестве основы модели инструментальные средства.

В качестве общих критериев выбора такой основы можно предложить следующие:

- наличие элементов структурно и семантически схожих с элементами языка OWL, что упростит отображение OWL-онтологий в ОПП;
- распространенность модели представления знаний, что позволит использовать полученную в итоге ОПП в рамках большего числа информационных систем, а также избавит от необходимости создания специальных программных средств для работы с ней;
- простота построения графического представления модели в виде графовой структуры.

С точки зрения авторов на роль такого основания подходит модель простой системы организации знаний (Simple Knowledge Organization System, SKOS) [14]. Ее категории элементов представлены на рисунке 3.

Среди них основными являются:

- концепт (Concept) – обозначает какое-либо понятие предметной области, близок по смыслу с классом или экземпляром класса в OWL;
- коллекция (Collection) – набор концептов, схожих по некоторому признаку;
- отношения «шире»/«уже» (broader/narrower) – вместе со своими транзитивными вариантами «broaderTransitive» и «narrowerTransitive» - позволяют задавать иерархию на концептах и схожи с отношениями «подкласс»/«суперкласс» (subclass/superclass) в OWL;
- метки «prefLabel», «altLabel» и «hiddenLabel» – представляют основное, альтернативное и служебное наименование концепта;
- отношения семантической близости (Mapping relation) определяют различные варианты близости концептов, принадлежащих различным концептуальным схемам (Concept scheme). В OWL для этой цели используется аксиома тождественности (equivalent-to).

Необходимо заметить, что сама модель SKOS описывается ее авторами языком OWL и представляет собой простую OWL-онтологию [15]. Ее категории элементов, такие как концепт (Concept), коллекция (Collection), концептуальная схема (ConceptScheme), представляются в виде OWL-классов. Конкретные элементы SKOS-модели представляются экземплярами соответствующих OWL-классов и связываются между собой отношениями. Описание модели SKOS с помощью языка OWL позволяет осуществлять взаимодействие с ней через те

же программные интерфейсы, что ориентированы на работу с OWL онтологиями, например, OWL API [16]. Кроме того, модель SKOS имеет статус рекомендации W3C, которая является куратором развития стандартов и технологий, используемых в сети Интернет.

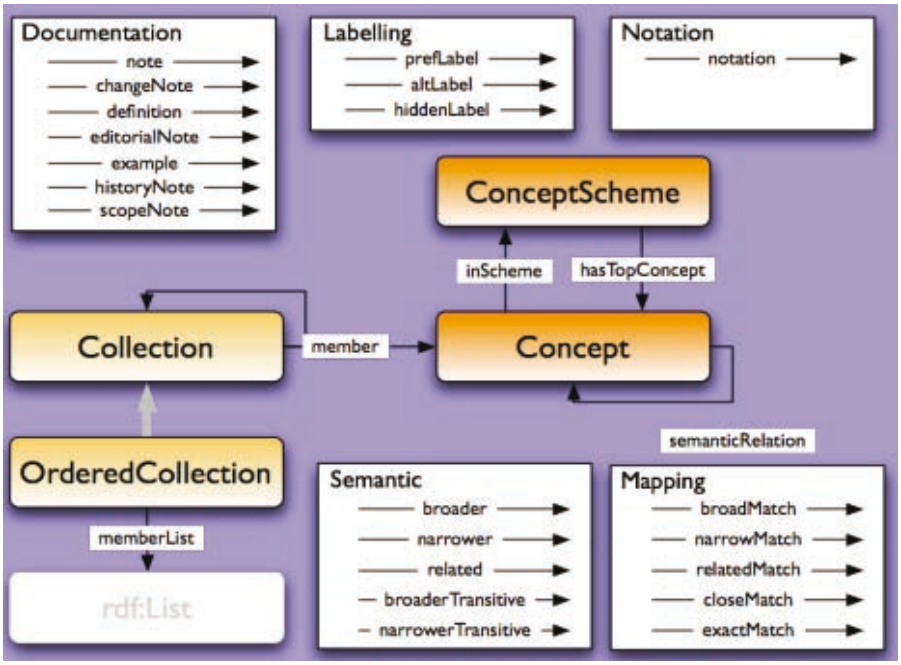

Рисунок 3 – Основные элементы модели простой системы организации знаний (Simple Knowledge Organization System, SKOS)

Что же касается визуализации элементов SKOS, то в данном случае их совокупность достаточно просто представляется в виде графа: его вершинами являются SKOS-концепты, а дугами - отношения между ними. Напротив, для аксиом OWL, включающих пересечение, объединение, отрицания OWL-классов, а также различные ограничения на свойства, определение соответствующих элементов графа требует их предварительного анализа.

Таким образом, модель SKOS является удобной основой для построения из исходной OWL-онтологии изоморфной ей онтологии пользовательского представления.

2 Отображение в онтологии пользовательского представления элементов OWL-онтологий

Несмотря на то, что наборы элементов OWL и SKOS являются схожими, далеко не всегда OWL-онтологию можно непосредственно представить в виде модели SKOS. Главным образом это связано с тем, что язык OWL является более выразительным языком, чем язык модели SKOS. Поэтому при отображении синтаксических конструкций первого в конструкции второго неизбежно приходиться опускать некоторые элементы, тем самым обедняя семантику исходного выражения. При решении вопроса о том, чем необходимо пожертвовать из исходного выражения необходимо удовлетворить два требования:

1) результат упрощения должен быть представлен пользователю так, чтобы он смог понять смысл того или иного понятия;
2) элементов ОПП должно быть достаточно для формирования запросов (в том числе – сложных) на поиск экземпляров в исходной онтологии.

С точки зрения способов представления в SKOS аксиомы OWL могут быть разделены на две группы – простые и составные. Простые аксиомы, имеют в правой части один именован-

ный или неименованный класс, определенный ограничением на одно свойство, а составные - несколько именованных и/или неименованных классов. Для ясности заметим, что в языке OWL неименованным или анонимным является класс, не имеющий интернационализированного идентификатора ресурса (IRI, Internationalized Resource Identifier) и заданный через ограничение его экстенсионала, например, в виде объединения, пересечения классов, ограничений на свойство, а также их комбинаций. Простые аксиомы могут быть непосредственно представлены в виде совокупности элементов SKOS-модели, тогда как в случае составных необходим их предварительный анализ. В ходе такого анализа происходит выявление компонентов аксиомы, значимых при визуализации, а также определение соответствующих им совокупностей элементов модели SKOS (рисунок 4).

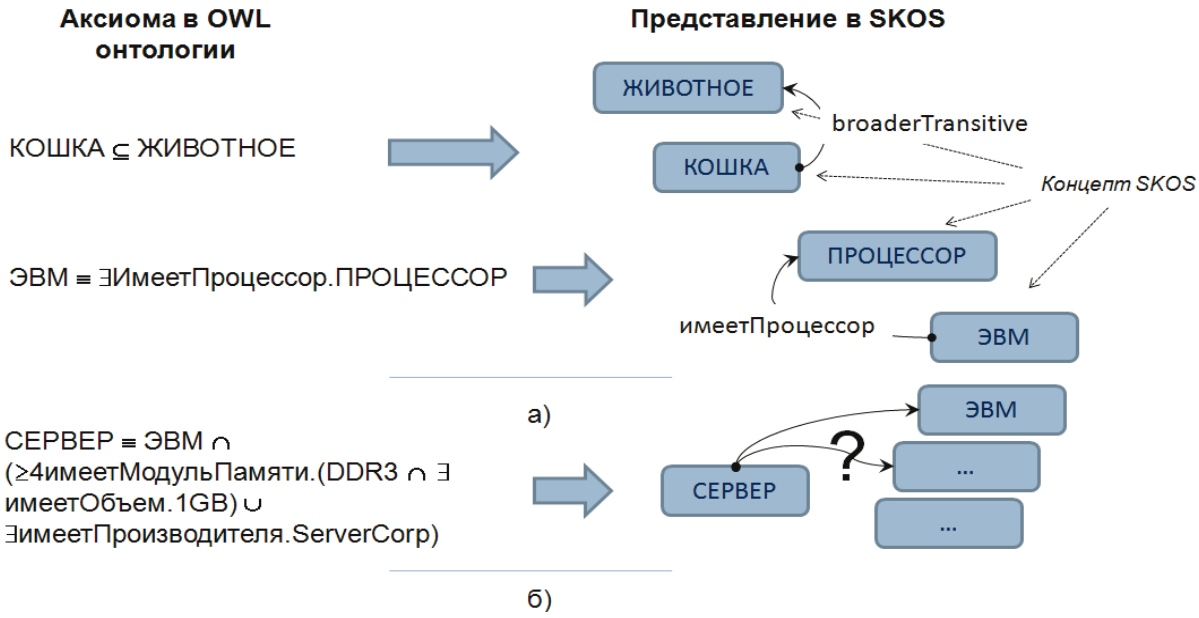

Рисунок 4 – Представление в SKOS и визуализация а) простых и б) составных аксиом OWL

Исходя из изложенного, определим общий порядок формирования ОПП на основе исходной OWL онтологии как набор следующих шагов:

1) обработка машиной логического вывода исходной онтологии и сохранения в ней выведенных утверждений в явном виде;
2) создание для каждого именованного класса исходной OWL-онтологии, соответствующего SKOS-концепта в ОПП;
3) задание отношений иерархии на множестве полученных концептов ОПП путем определения между ними транзитивных отношения SKOS «шире» (broaderTransitive), в соответствии с иерархией между именованными классами исходной OWL-онтологии;
4) создание для каждого свойства в исходной OWL-онтологии аналогичного свойства в ОПП;
5) определение принадлежности свойств к классам в исходной OWL-онтологии и привязка соответствующих свойств в ОПП к SKOS-концептам;
6) анализ оставшихся сложных аксиом в формальных определениях классов и свойств в исходной OWL-онтологии и их представление в ОПП.

Начальная обработка исходной онтологии машиной вывода позволит производить последующий ее анализ с учетом выведенных утверждений, которые являются не менее важными при визуализации, чем определенные явно. Для большей ясности шагов 4 и 5 заметим,

что в языке OWL определено несколько видов свойств в зависимости от типа элементов в их области определения (Property domain) и области значений (Property range). Основными видами свойств являются объектные свойства (Object properties) или отношения, имеющие в области значения и определения именованный или неименованный класс, и типизированные свойства (Datatype properties), область значений которых, в отличие от объектных, задается указанием типа данных (целочисленный, строковый, дата и т.д.). Возможность наличия неименованных классов в областях значений и/или определений свойств делает необходимым шаг 5, так как в модели SKOS отношение может быть привязано только к SKOS-концептам, которые соответствуют именованным классам OWL-онтологии.

Рассмотрим принципы разбора и визуализации составной аксиомы на примере аксиомы:

$$\text{СЕРВЕР} \equiv \text{ЭВМ} \cap (\geq 4\text{имеетМодульПамяти}.(\text{DDR3} \cap \exists\ \text{имеетОбъем.1GB}) \cup \exists\text{имеетПроизводителя.ServerCorp})$$

Здесь и далее при записи аксиом объединение и пересечение классов обозначается знаками «∪» и «∩» соответственно.

Опуская формальную сторону можно сказать, что данная аксиома определяет понятие «Сервер» как ЭВМ, которая либо имеет не менее 4 модулей памяти типа DDR3 объемом 1 гигабайт, либо произведена некоторой фирмой «ServerCorp». Исходя из этого, можно предположить, что поиск экземпляров данного класса в исходной онтологии можно производить двумя способами:

- искать среди экземпляров класса «ЭВМ» те, что имеют отношения принадлежности с экземплярами, представляющими необходимые модули памяти;
- искать среди экземпляров класса «ЭВМ» те, что имеют отношение «имеетПроизводителя» с экземплярами класса «ServerCorp».

Визуальное представление данных альтернатив даст возможность пользователю легко интерпретировать смысл понятия и сформулировать критерии поиска экземпляров данного класса в исходной онтологии.

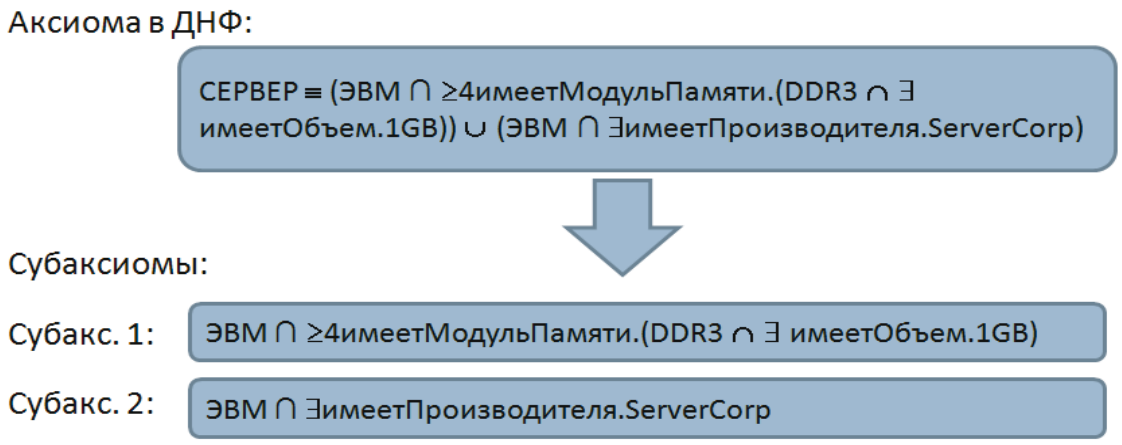

Рисунок 5 – Разделение аксиомы на субаксиомы

Далее рассмотрим подробно разбор данной аксиомы и ее представление в ОПП. Введем понятие «Субаксиома», которым будем обозначать часть какой-либо аксиомы, заданной именованным или неименованным классом, а также их комбинацией. Разложим аксиому на несколько субаксиом, приведя ее предварительно к дизъюнктивной нормальной форме. В этом случае аксиома будет состоять из одной или нескольких субаксиом, соединенных дизъюнкцией (рисунок 5).

Каждая субаксиома определяет альтернативные способы трактовки смысла некоторого понятия, а также разные способы поиска экземпляров, соответствующего ему класса. Субаксиомы представляются в виде SKOS-концептов, которые связываются отношением «related» со SKOS-концептом, соответствующим именованному классу, определяемому ими. Имя SKOS-концепта, представляющего субаксиому (далее SKOS-концепт-субаксиома) формируется путем конкатенации имен, входящих в субаксиому классов. В нашем примере (рисунок 6) имена концептов будут иметь вид «ЭВМ – имеетПроизводителя» и «ЭВМ - имеетМодульПамяти». В свою очередь аксиому целиком представим как именованную SKOS-коллекцию, членами которой будут соответствующие SKOS-концепты-субаксиомы. В качестве имени такой коллекции назначается имя класса, определяемого аксиомой.

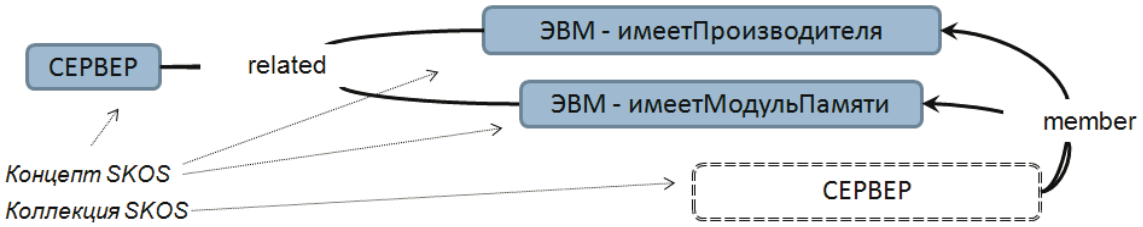

Рисунок 6 – Пример представления аксиомы в виде SKOS концептов-субаксиом

С каждой субаксиомой связывается также список свойств, который составляют свойства, присущие классам, входящим в нее. Если в субаксиоме присутствует неименованный класс, заданный в виде ограничения на свойство, то его следует рассматривать как именованный, которому присуще определяющее его свойство. Для иллюстрации рассмотрим пример привязки свойств к субаксиоме (рисунок 7).

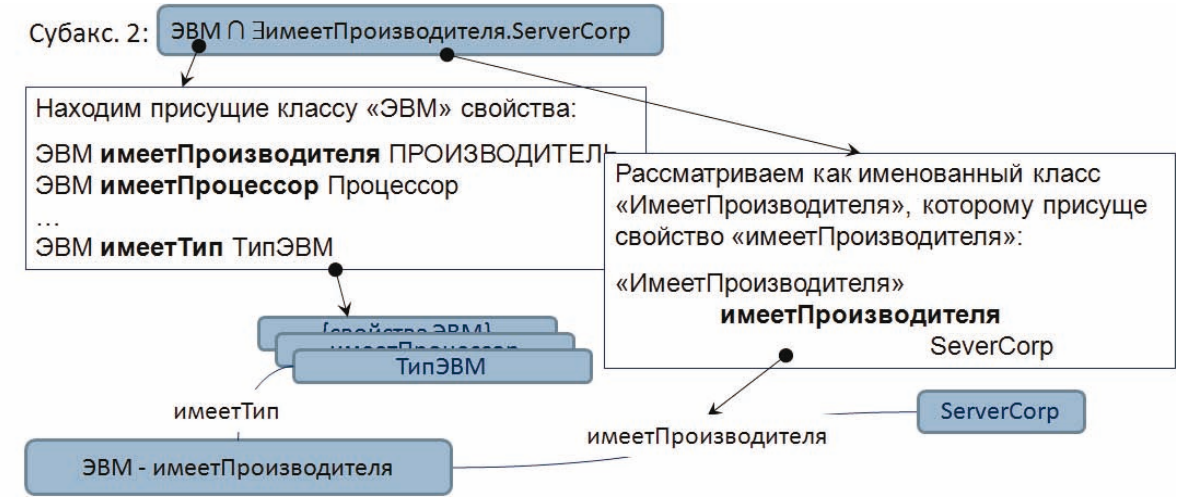

Рисунок 7 – Разбор субаксиомы

В данном случае в исходной OWL-онтологии производится поиск аксиом, задающих объектные и типизированные свойства, у которых в область определения (Property domain) входит класс «ЭВМ». Найденные свойства «привязываются» к концепту-субаксиоме. Далее рассматривается неименованный класс «∃имеетПроизводителя.ServerCorp», который определяет сущности, имеющие хотя бы одно отношение «имеетПроизводителя» с сущностями, отнесенными к классу «ServerCorp». Данный неименованный класс рассматривается как

именованный «имеетПроизводителя» с присущим ему свойством «имеетПроизводителя», которое также добавляет в список свойств концепта-субаксиомы.

Заметим также, что для прикрепляемых свойств рекурсивно запускается анализ их области значения (Property range), которую также может определять простая или составная аксиома. В случае простой аксиомы, включающей только один именованный класс, рекурсия завершается, а SKOS-концепт, соответствующий классу в области значений, связывается со свойством. В противном случае аксиома в области значений разбивается на субаксиомы, каждая из которых привязывается к свойству как альтернативное значение. Для субаксиом в свою очередь также запускается процесс поиска присущих им свойств (рисунок 8).

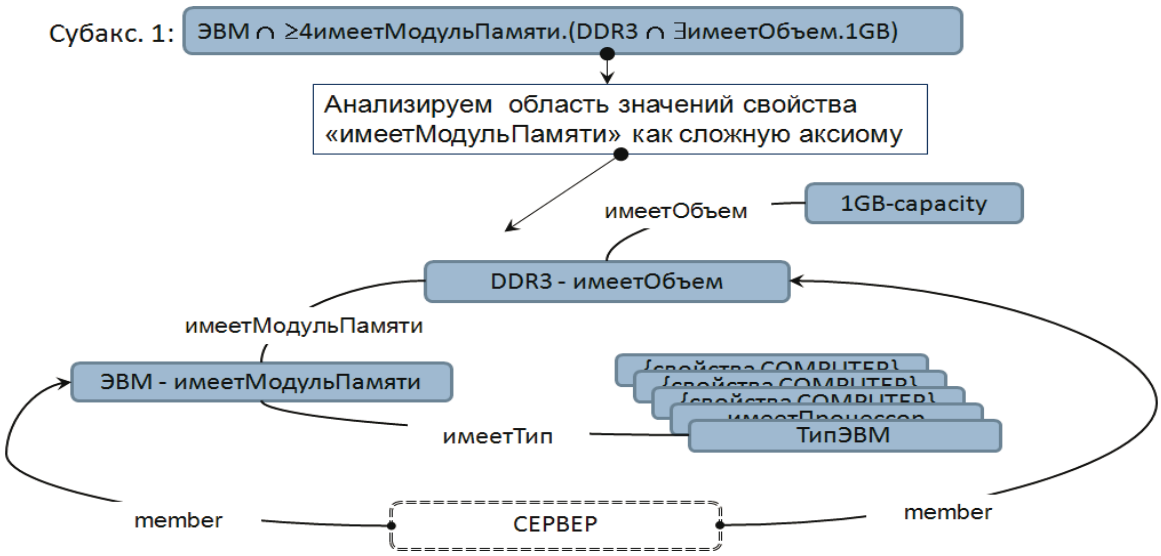

Рисунок 8 – Разбор аксиомы в области значений (Property domain) объектного OWL-свойства

Заметим, что концепты SKOS, созданные в процессе анализа областей значений прикрепленных свойств, также включаются в SKOS-коллекцию, представляющую исходную анализируемую аксиому. В рассматриваемом примере такой SKOS-коллекцией является коллекция «Сервер» (рисунок 8).

В субаксиоме могут также присутствовать отрицания классов. Отрицания позволяют задавать множество интерпретации одного класса через дополнение множества интерпретации другого. Например, аксиома: СЕРВЕР ≡ ¬НАСТОЛЬНАЯ_ЭВМ, определяет понятие «СЕРВЕР» как нечто, не отнесенное к классу «НАСТОЛЬНАЯ_ЭВМ». Заметим, что отображение отрицаний в ОПП позволяет главным образом более полно передать смысл понятия пользователю. С точки же зрения формирования поискового запроса, как указание на исключение некоторых классов из области поиска, отрицание бесполезно. Так, исходя из ранее приведенной аксиомы, при поиске экземпляров класса «СЕРВЕР» экземпляры, причисленные к классу «НАСТОЛЬНАЯ_ЭВМ» рассматриваться не будут, так как предполагается, что машина вывода на первом шаге этапе должным образом произвела классификацию элементов исходной онтологии.

Исходя из изложенного, можно сделать вывод о том, что наличие отрицаний имеет смысл учитывать при отображении в ОПП элементов исходной OWL-онтологии. Однако осуществляется это более упрощенно, чем требуется исходя из формальной семантики языка OWL. Например, в случае простой аксиомы, состоящей из отрицания именованного класса (рисунок 9), создается соответствующий ему новый SKOS-концепт, с которым связываются другие элементы аксиомы и который заноситься в соответствующую аксиоме SKOS-

коллекцию. Хотя формально (с учетом того, что класс «Сервер» отождествляется с дополнением (отрицанием) класса «Настольная ЭВМ»), более правильным было бы связать его со всеми именованными классами онтологии (кроме, разумеется, подклассов класса «Настольная ЭВМ»). Однако полагается, что для понимания пользователем смысла понятия «Сервер» визуализации упрощенного варианта будет достаточно.

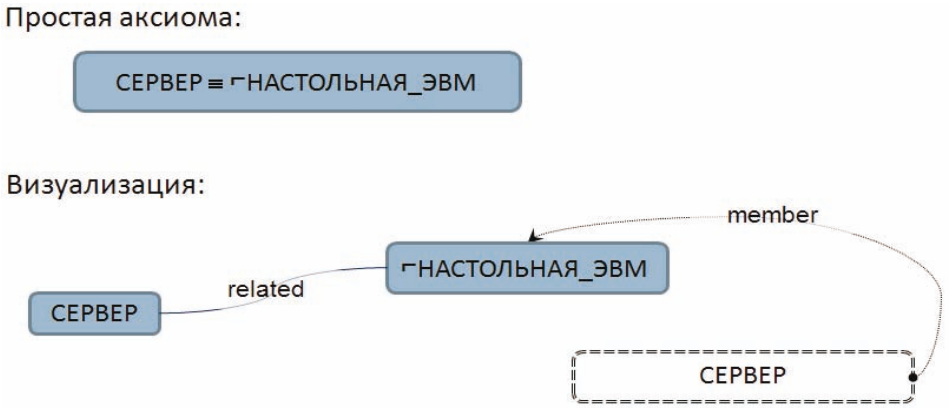

Рисунок 9 – Визуализация простой аксиомы с отрицанием

В случае если отрицание именованного класса присутствует внутри субаксиомы, то в виде отдельного SKOS-концепта оно не выносится, а остается в наименовании концепта-субаксиомы. Также возможны ситуации, когда отрицание именованного класса единолично присутствует в области значений (Property range) свойства. В этом случае данное свойство связывается с создаваемым SKOS-концептом, соответствующим отрицанию класса. Данный SKOS-концепт заносится в соответствующую SKOS-коллекцию (рисунок 10).

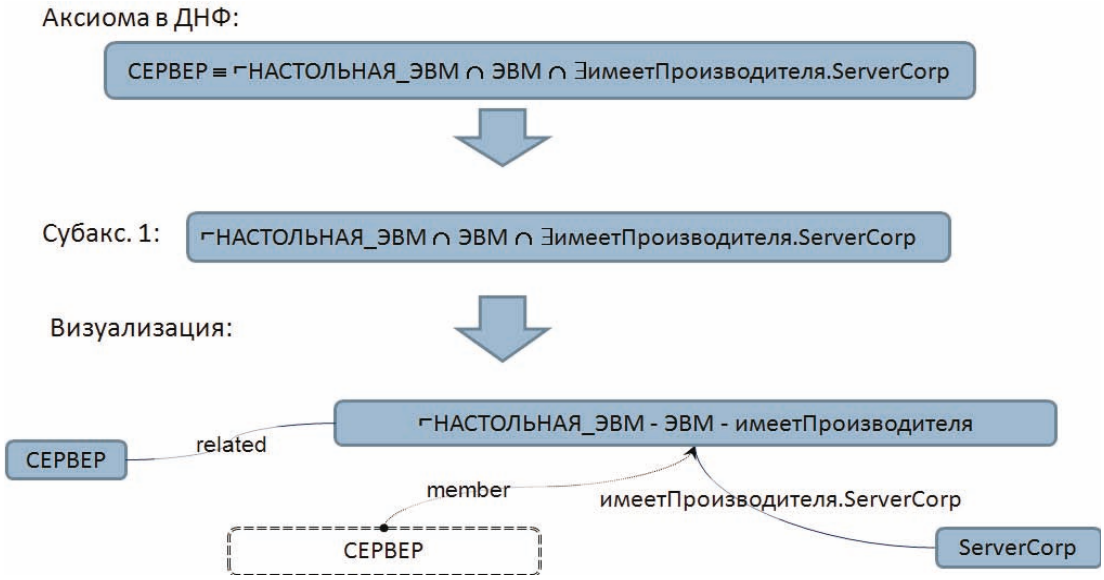

Рисунок 10 – Визуализация отрицания в составе субаксиомы

Остальные ситуации с использованием отрицаний сводятся к рассмотренным.

Далее рассмотрим представление в ОПП основных видов аксиом (тождественности, наследования, несвязности и перечисления) с точки зрения их использования для задания OWL-классов.

Аксиомы тождественности, определяющие классы, представляются в виде отношения SKOS «related» между SKOS-концептом, соответствующим OWL-классу, и с субаксиомами, как это было показано ранее (рисунок 6). Аксиомы наследования представляются иерархическими транзитивными SKOS-отношениями «шире» (broaderTransitive) и «уже» (narrowerTransitive), если они (аксиомы) являются простыми и включают один именованный класс. Если аксиомы являются составными, то они анализируются и представляются также как аксиомы тождественности.

Если при задании OWL-класса использована аксиома несвязности, то ее можно приближенно представить как аксиому тождественности, включающую отрицание (рисунок 11), которая в свою очередь представляется в ОПП рассмотренным ранее способом.

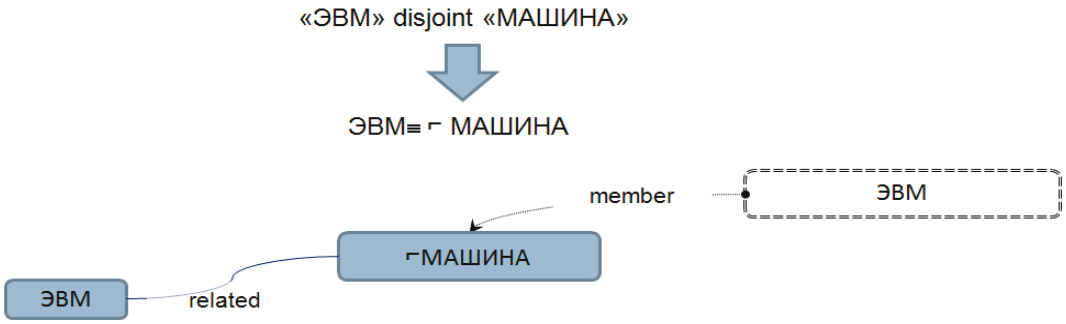

Рисунок 11 – Представление аксиомы несвязности (disjoint)

Что касается классов, заданных путем перечисления, то они представляются в ОПП аналогично именованным классам, заданным с помощью аксиомы «equivalent-to», то есть в виде SKOS-концептов, с которыми связываются отношением «related» SKOS-концепты, соответствующие перечисляемым экземплярам данного класса.

Рассмотренный в данном разделе порядок представления OWL-онтологии в ОПП, позволяет явно отразить в последней некоторую семантику OWL-аксиом в виде совокупностей SKOS-элементов. Визуализация всей ОПП сводиться к отображению в виде графовой структуры SKOS-концептов и отношений между ними. В этом случае при показе пользователю всей онтологии можно скрыть концепты, которые являются членами каких-либо SKOS-коллекций. Это вызвано тем обстоятельством, что в коллекциях находятся SKOS-концепты, соответствующие неименованным OWL-классам исходной онтологии и их отображение усложнит понимание структуры онтологии. При выборе же конкретного SKOS-концепта имеет смысл показать ранее скрытые концепты, что позволить пользователю получить более полное представление о понятии. Формирование поисковых запросов на основе графического представления ОПП предполагается осуществлять путем отслеживания выбора цепочки SKOS-концептов пользователем и конкатенации частей запроса, сопряженных с каждым элементом цепочки на этапе отображения исходной онтологии в ОПП.

Заключение

В данной работе рассмотрен процесс создания для OWL-онтологии ее упрощенной модификации – онтологии пользовательского представления - ОПП, направленной на непосредственную визуализацию ее элементов в рамках ГПИ. В качестве основы ОПП используется модель SKOS, совокупности элементов которой могут быть наглядно представлены в виде графовой структуры.

Основное внимание в статье уделено процессу разбора и отображения основных видов аксиом исходной OWL-онтологии в ОПП. Авторы полагают, что полученная в итоге ОПП

позволить упростить проектирование и разработку ГПИ информационной системы или веб-ресурса, в основе которого используется сложная OWL-онтология предметной области, за счет отсутствия необходимости анализа составных аксиом и определения способов их визуализации.

К основным направлениям дальнейшей работы можно отнести исследование возможностей хранения в ОПП данных о поисковых потребностях пользователя, а также определение с учетом их различных способов отображения элементов ОПП. Это позволит специфицировать представление знаний об одной или нескольких предметных областях для конкретных пользователей в соответствии с их задачами, что является особенно важным для мультипредметных веб-ресурсов, в которых могут использоваться несколько онтологий с большими системами понятий. Важным вопросом является также создание ОПП для произвольной OWL-онтологии. Для этого также предполагается разработка автоматизированного средства, которое могло бы быть использовано как автором онтологии, так и разработчиком ГПИ.

Благодарности

Исследование поддержано программой ОНИТ «Интеллектуальные информационные технологии, системный анализ и автоматизация», проект 2.2.

Список источников

[1] Catarci T., Dongilli P., Dimascio T., Franconi E., Santucci G., Tessaris S. An Ontology-based Visual Tool for Query Formulation Support // Proc. of the 16th European Conference on Artificial Intelligence, 2004.
[2] Bechhofer S., Stevens R., Ng G., Jacoby A., Goble C. Guiding the user: An ontology driven interface // UIDIS, 1999, pp.158–161.
[3] Feikje H., Chris M., Peter E. Evaluating an Ontology-Driven WYSIWYG Interface // Proc. of the 5th International Conference on Natural Language Generation, 2008, pp.138-146.
[4] Грибова В.В., Клещев А.С. Управление проектированием и реализацией пользовательского интерфейса на основе онтологий // Проблемы управления. 2006. №2. - С. 58-62.
[5] OWL2 - Web Ontology Language. Overview, 2009. - http://www.w3.org/TR/owl2-overview
[6] Schmidt-Schauss M., Smolka G. Attributive concept descriptions with complements // Artificial Intelligence. 1991. 48. 1–26.
[7] Bergh J.R. Ontology comprehension. Master's Thesis - University of Stellenbosch, 2010.
[8] Bauer J. Model exploration to support understanding of ontologies. Master's thesis. - Technische Universität Dresden, 2009.
[9] Ellson J., Gansner E.R., Koutsofios L., North S., Woodhull G. Graphviz. Open source graph drawing tools proceedings // Graph Drawing. 2002. P. 483-484.
[10] Alani H. TGVizTab: An Ontology Visualisation Extension for Protégé // Knowledge Capture 03 - Workshop on Visualizing Information in Knowledge Engineering Sanibel Island, FL: ACM (2003), p. 2-7.
[11] Katifori A., Halatsis C., Lepouras G., Vassilakis C., Giannopoulou E. OWLViz. Ontology visualization methods – A survey // ACM computing surveys. 2007. 39(4):10.
[12] Bosca A., Bonino D., Pellegrino P. OntoSphere: More than a 3D ontology visualization tool // SWAP, the 2nd Italian semantic web workshop, 2005.
[13] Guarino N. Formal Ontology and Information Systems // Proc. 1st Int'l Conference on Formal Ontology in Information Systems. Guarino, N. (ed.). 1998. 3-15. IOS Press/Ohmsha.
[14] SKOS Simple Knowledge Organization System Reference, W3C Recommendation, 2009 – http://www.w3.org/TR/skos-reference.
[15] Jupp S., Bechhofer S., Stevens R. SKOS with OWL: Don't be Full-ish! // Proc. of the Fifth OWLED Workshop on OWL: Experiences and Directions, Karlsruhe, Germany, October 26-27, 2008. - http://www.webont.org/owled/2008/papers/owled2008eu_submission_22.pdf
[16] Bechhofer S., Lord P., Volz R. Cooking the Semantic Web with the OWL API // ISWC, the 2nd International Semantic Web Conference. Lecture Notes in Computer Science, v. 2870. Sanibel Island, Florida, October 2003. Springer.

Сведения об авторах

Ломов Павел Андреевич, 1982 г.р., кандидат технических наук, младший научный сотрудник Института информатики и математического моделирования технологических процессов Кольского научного центра РАН. Область научных интересов: онтологическое моделирование, технологии Semantic Web , сетевые технологии, защита информации, 18 печатных работ.

Lomov Pavel Andreevich, (b. 1982) PhD, younger research associate of Institute for Informatics and Mathematical Modeling of Technological Processes of the Kola Science Center RAS. Research interests: the ontological modeling technology Semantic Web, network technology, information security, 18 publications.

Шишаев Максим Геннадьевич, доктор технических наук, Институт информатики и математического моделирования технологических процессов Кольского научного центра РАН, заведующий лабораторией. Автор более 130 печатных работ. Область научных интересов: технологии построения и модели распределенных информационных систем, методы логического проектирования информационных систем, модели и методы построения интегрированных научно-образовательных информационных систем, прикладные проблемы информационной безопасности, проблемы региональной информатизации.

Maxim Gennadyevich Shishaev, doctor of technical sciences, senior researcher, head of Regional Information Systems Laboratory Institute of Informatics and Mathematical Modeling of Technological Processes of the Kola Science Center RAS. Author of over 130 publications. Research interests: technology and models of distributed information systems, methods of logical design of information systems, models and methods of design of integrated scientific and educational information systems, applied information security problems, problems of regional informatization.

***Диковицкий Владимир Витальевич*,** 1987 г.р., аспирант Института информатики и математического моделирования технологических процессов Кольского научного центра РАН. Область интересов: семантическая обработка информации, визуализация данных, человеко-машинное взаимодействие, опубликовано 11 работ.

***Dikovitsky Vladimir Vitalyvich*,** (b. 1987) post-graduate of Institute for Informatics and Mathematical Modeling of Technological Processes of the Kola Science Center RAS. Field of Interest: semantic information processing, data visualization, human-computer interaction, published 11 papers.