# Лекция 15 Онтологии

Изначально понятие «онтология» появилось в философии еще в 17в., где оно обозначает раздел, изучающий фундаментальные принципы бытия, его наиболее общие сущности и категории, его принципы, структуры и закономерности. Применимо к сфере информационных технологий это понятие впервые использовал в своих работах Грубер (Gruber) в 1993 году [1].

В настоящее время онтологии стали весьма популярными, но четкого понимания их еще не сложилось. При первом ознакомлении с онтологией кажется, что это достаточно абстрактное понятие, не применимое к практической деятельности. Но на самом деле оно было введено именно с практической целью для построения информационных систем.

*Онтология представляет собой попытку всеобъемлющей и детальной формализации некоторой области знаний с помощью концептуальной схемы. Обычно такая схема состоит из структуры данных, содержащей все классы объектов, их связи и правила, приятые в этой области.* [2]Проще говоря, это некоторое формальное описание знаний, понятное компьютеру. Это описание может быть как общих понятий, так и понятий для конкретной предметной области. Онтологии содержат знания об отношении между понятиями. Причем одним из главных принципов онтологии является то, что она должна быть понятна всем, кто может использовать ее.

***На настоящий момент разработаны специальные языки для описания онтологий*** [2]:

* OWL – Web Ontology Language, стандарт W3C, язык для семантических утверждений, разработанный как расширение RDF и RDFS [12];
* KIF (англ. Knowledge Interchange Format – формат обмена знаниями) – основанный на S-выражениях (символических выражениях данных) синтаксис для логики [2];
* Common Logic (CL) – преемник KIF (стандартизован – ISO/IEC 24707:2007);
* CycL – онтологический язык, использующийся в проекте Cyc. Основан на исчислении предикатов с некоторыми расширениями более высокого порядка [2].

Для работы с языками онтологий существует несколько видов технологий: редакторы онтологий (для создания онтологий), СУБД онтологий (для хранения и обращения к онтологии) и хранилища онтологий (для работы с несколькими онтологиями).

Достаточно большой список разработанных онтологий приведен в [3]. Все они были разработаны в редакторе онтологий Protégé. Наиболее широкое распространение онтологии нашли во Всемирной паутине. Онтологии в сети варьируются от больших таксономий, категоризирующих веб-сайты, до категоризаций продаваемых товаров и их характеристик.

## Цели создания онтологий

Как уже говорилось термин «онтология» в сфере информационных технологий впервые появился в работе Грубера [1]. Он рассматривал различные аспекты взаимодействия интеллектуальных систем друг с другом и с человеком. Под интеллектуальными системами понимаются программы, моделирующие некоторые аспекты интеллектуальной деятельности человека. В обычных программах знания, которые вкладываются программистом (алгоритмы) не меняются. Интеллектуальная система в этом смысле более универсальна – в ней знание о том, что надо делать в процессе исполнения программы, не вшито в программу раз и навсегда, но может меняться. Если так, то эти знания необходимо передавать программе как данные, т.е. возникает необходимость их описания.

Идея Грубера состояла в том, чтобы позволить интеллектуальным системам обмениваться между собой заложенными в них знаниями, прежде всего декларативными. Если внутри интеллектуальной системы знания о мире могут быть закодированы как угодно, то для обмена этими знаниями с другой интеллектуальной системой необходимо предоставить описание этих знаний. Это описание должно быть в достаточной степени формальным, чтобы быть понятным другой системе, а также должен быть известен язык этого описания. Кроме того, описание должно быть понятно также и человеку. Для этого Грубер предложил описывать знания двумя способами:

* в канонической форме, которая представляет собой описание знаний на языке логики предикатов (например, в виде фактов языка Prolog);
* в форме онтологии, которая представляет собой множество классов, связанных между собой отношением обобщения (это обратное отношение для отношения наследования).

Таким образом, онтология по Груберу представляет собой описание декларативных знаний, сделанное в виде классов с отношением иерархии между ними. К этому описанию, предназначенному для чтения человеком, присоединено описание в канонической форме, которое предназначено для чтения машинами. Каждая интеллектуальная система может предоставлять несколько таких описаний, соответствующих различным областям хранящихся в ней декларативных знаний и, таким образом, выступает как хранилище библиотеки онтологий. Грубер представлял, что интеллектуальные системы будут выступать как библиотеки онтологий и свободно обмениваться онтологиями между собой. При этом библиотеке онтологий уже не обязательно быть интеллектуальной системой, достаточно просто предоставлять сервис по передаче онтологий по требованию.

Составление описания декларативного знания обычно требует большой работы и определенных навыков. Для обозначения этой работы, а также ее результата, Грубер ввел в обиход специальный термин «концептуализация». Описание он называл «спецификацией». Таким образом, онтология по Груберу определяется как спецификация концептуализации[3].

В [4] выявлены основные причины возникновения потребности в создании онтологий.

*Совместное использование людьми или программными агентами общего понимания структуры информации* является одной из наиболее общих целей разработки онтологий. К примеру, пусть, несколько различных веб-сайтов содержат информацию по медицине или предоставляют информацию о платных медицинских услугах, оплачиваемых через Интернет. Если эти веб-сайты совместно используют и публикуют одну и ту же базовую онтологию терминов, которыми они все пользуются, то компьютерные агенты могут извлекать информацию из этих различных сайтов и накапливать ее. Агенты могут использовать накопленную информацию для ответов на запросы пользователей или как входные данные для других приложений.

*Обеспечение возможности использования знаний предметной области* стало одной из движущих сил недавнего всплеска в изучении онтологий. Например, для моделей многих различных предметных областей необходимо сформулировать понятие времени. Это представление включает понятие временных интервалов, моментов времени, относительных мер времени и т.д. Если одна группа ученых детально разработает такую онтологию, то другие могут просто повторно использовать ее в своих предметных областях. Кроме того, если нужно создать большую онтологию, то можно интегрировать несколько существующих онтологий, описывающих части большой предметной области. Также можно повторно использовать основную онтологию, но расширить ее для описания интересующей предметной области.

*Создание явных допущений в предметной области*, лежащих в основе реализации, дает возможность легко изменить эти допущения при изменении наших знаний о предметной области. Жесткое кодирование предположений о мире на языке программирования приводит к тому, что эти предположения не только сложно найти и понять, но и также сложно изменить, особенно непрограммисту. Кроме того, явные спецификации знаний в предметной области полезны для новых пользователей, которые должны узнать значения терминов предметной области.

*Отделение знаний предметной области от оперативных знаний* – это еще один вариант общего применения онтологий. Можно описать задачу конфигурирования продукта из его компонентов в соответствии с требуемой спецификацией и внедрить программу, которая делает эту конфигурацию независимой от продукта и самих компонентов.

*Анализ знаний в предметной области* возможен, когда имеется декларативная спецификация терминов. Формальный анализ терминов чрезвычайно ценен как при попытке повторного использования существующих онтологий, так и при их расширении.

Приведенные выше причины определяют основные цели создания онтологий. А области применения их чрезвычайно обширны.

## Основные понятия онтологии

Современные онтологии строятся по большей части одинаково, независимо от языка написания. Обычно они состоят из понятий, атрибутов, отношений и экземпляров.

Понятия (также их называют классами) – абстрактные группы, коллекции или наборы объектов. Они могут включать в себя экземпляры, другие классы, либо же сочетания и того, и другого. Классы онтологии составляют *таксономию* – иерархию понятий по отношению вложения.

Объекты в онтологии могут иметь атрибуты. Каждый атрибут имеет, по крайней мере, имя и значение и используется для хранения информации, которая специфична для объекта и привязана к нему.

Отношения выражают свойства понятий (классов) и их зависимости с другими понятиями (классами). Отношение можно представить как атрибут, значением которого является другое понятие (класс).

Экземпляры или индивиды – элементы нижнего уровня онтологии. Экземпляры могут представлять собой как физические объекты (люди, дома), так и абстрактные (числа, слова). Строго говоря, онтология может и обойтись без конкретных объектов, но чаще всего они также включаются.

Отношения и атрибуты также в некоторых источниках называют слотами (или ролями). А ограничения, наложенные на слоты – фацеты (или ограничения ролей). Онтология вместе с набором индивидуальных экземпляров классов образует базу знаний. В действительности, трудно определить, где кончается онтология и где начинается база знаний.

Единого универсального подхода к созданию онтологий, который бы привел к однозначно успешному результату, не существует. Процесс создания онтологий обычно является итеративным, т.е. сначала создается черновой набросок, а затем по мере необходимости происходит возврат для определения деталей, и так продолжается до тех пор, пока онтология не будет отражать концепцию предметной области с определенной степенью.

Практически, создание онтологий включает следующие этапы:

1. Определение понятий (классов) в онтологии
2. Организация понятий (классов) в некоторую иерархию (базовый класс –> подкласс)
3. Определение слотов и их допустимых значений
4. Заполнение значений слотов для экземпляров классов.

В [5] сформулированы основные правила разработки онтологий.

1. Не существует единственного правильного способа моделирования предметной области – всегда существуют жизнеспособные альтернативы. Лучшее решение почти всегда зависит от предполагаемого приложения и ожидаемых расширений.
2. Разработка онтологии – это обязательно итеративный процесс. В процессе создания важно возвращаться к уже созданным классам и отношениям и уточнять и добавлять информацию в случае необходимости. Иногда полезно вносить и кардинальные изменения для улучшения общей структуры онтологии.
3. Понятия в онтологии должны быть близки к объектам (физическим или логическим) и отношениям в интересующей предметной области. Скорее всего, это существительные (объекты) или глаголы (отношения) в предложениях, которые описывают предметную область. Это поможет лучше понимать онтологию людям, не являющимся разработчиками, но заинтересованными в применении ее к своим приложениям.

То есть, знание того, для чего будет использована онтология и насколько детальной или общей она будет, повлияет на многие решения, касающиеся моделирования. Среди нескольких жизнеспособных альтернатив нужно определить, какая поможет лучше решить поставленную задачу и будет более наглядной, более расширяемой и более простой в обслуживании. Также нужно помнить, что онтология – это модель реального мира и понятия в онтологии должны отражать эту реальность. После того, как будет определена начальная версия онтологии, можно оценить и отладить ее, используя ее в приложениях или в методах решения задач и/или обсудив ее с экспертами предметной области. В результате почти наверняка нужно будет пересмотреть начальную онтологию. Этот процесс итеративного проектирования, вероятно, будет продолжаться в течение всего жизненного цикла онтологии.

## Языки описания онтологий

Для разработки демонстрационной онтологии был выбран язык OWL[12], так как он является языком, рекомендованным для разработки онтологий консорциумом Всемирной паутины (World Wide Web Consortium, W3C).

Язык OWL разработан как расширение языка RDF (Resource Description Framework). Язык RDF [6] разработан для того, чтобы описывать содержимое Web. В Semantic Web (следующее поколение World Wide Web, в котором кроме гипертекстовых документов содержатся описания семантики этих документов), когда говорят о каких-то сущностях Web, называют эти сущности ресурсами. RDF представляет собой язык для описания таких ресурсов. Ввиду того что описания семантики документов должны быть понятны компьютерам, необходимо разработать специальные программы-агенты, которые производили бы такое чтение. Также необходимо обеспечить возможность обмена информацией между различными программными агентами. Таким образом, под RDF подразумевается не только сам язык, но также и различные дополнительные программные модули, необходимые для обеспечения полноценного чтения и обмена информацией, записанной на этом языке. Этот факт подчеркивается в названии языка RDF.

Главный элемент языка RDF – это тройка, или триплет. Тройка представляет собой совокупность трех сущностей:

* субъект;
* объект;
* предикат.

Предикаты еще часто называют отношениями. Тройка имеет также представление в виде графа вида субъект–предикат–объект, где субъект и объект представлены как узлы, а предикат выступает в роли ребра, которое эти узлы соединяет.

С математической точки зрения, тройка представляет собой экземпляр некоторого бинарного отношения. Отношение – это множество последовательностей, состоящих в точности из n элементов для некоторого заранее определенного натурального числа n. Если n = 2, то отношение называется бинарным, т.е. бинарное отношение представляет собой множество пар. Например, отношение «семейные пары» задает множество пар элементов вида («муж», «жена»). Каждая тройка определяет одну пару из некоторого бинарного отношения, но кроме этого дополнительно задает еще имя отношения, т.е. если имеется пара («муж», «жена») отношения «семейные пары», то эту пару можно выразить тройкой («муж», «семейные пары», «жена»). Для полноценного описания, кроме задания отношений, необходимо еще задать ограничения на их содержимое. Например, для отношения «семейные пары» необходимо уточнить, что первый элемент каждой пары этого отношения должен быть мужчиной, а второй – женщиной. Для задания такого ограничения необходимо ввести понятия «мужчина» и «женщина», после чего задать ограничение, выражающее тот факт, что если имеется тройка вида («имя 1», «семейные пары», «имя 2»), то сущность «имя 1» должна быть экземпляром понятия «мужчина», а сущность «имя 2» – экземпляром понятия «женщина». Это делается посредством т.н. высказываний. Для высказываний существует свой язык, включающий переменные и логические операции. В качестве логических операций могут выступать: логическое «или» (дизъюнкция), логическое «и» (конъюнкция) и логическое следование (импликация). Имеются также кванторы существования и всеобщности, позволяющие ограничивать область применения высказывания. Язык RDF основан на математическом аппарате дескрипционной логики [7].

Дескрипционная логика (Description Logic – DL) базируется на формализмах семантических сетей и фреймов, но использует аппарат математической логики. В математической логике производится явное разделение на синтаксис и семантику. Синтаксис задает язык, с помощью которого записываются различные высказывания об элементах мира данной логической системы. Семантика задает ту часть описываемого мира, которая удовлетворяет заданным ограничениям. Таких частей может быть более одной или даже бесконечно много. Каждая такая часть мира называется моделью данной логической системы.

Ввиду того, что RDF предполагается использовать для описания ресурсов, распределенных по разным участкам Web, необходимо как-то решить проблему идентификации имен узлов и ребер RDF графа, т.е. элементов троек. Для этого используется стандартный подход: каждый элемент описывается посредством так называемого Унифицированного Идентификатора Ресурса (URI – Uniform Resource Identifer [8]). Обычно URI представляет собой либо URL (Унифицированный Указатель Ресурса–Uniform Resource Locator [9]), содержащий информацию о местонахождении данного ресурса в Web, либо URN (Унифицированное Имя Ресурса – Uniform Resource Name [10]), позволяющий идентифицировать данный ресурс в некотором пространстве имен. Пространство имен представляет собой просто именованное множество элементов и используется, чтобы обеспечить уникальность имен этих элементов в Web. Для более подробной информации по использованию этого языка можно обратится к официальному сайту, посвященному этому языку [6].

Схема RDF (RDF Schema, RDFS [11]) представляет собой расширение языка RDF, позволяющее описывать простые онтологии данных, находящихся в хранилищах RDF. Так же, как схема базы данных описывает структуру базы данных в виде заголовков таблиц и связей между ними, схема RDF позволяет описывать структуру RDF-хранилища. Структура описывает хранилище в терминах типов и отношений между ними.

В RDFS можно задавать классы, которые определяются в дескриптивной логике как унарные отношения. Для этого в RDFS определен специальный объект rdfs:Class – класс всех классов. Вообще, каждый объект RDF – это экземпляр класса rdfs:Resource, и rdfs:Class здесь не исключение. Но, с другой стороны, rdfs:Resource – это класс, а значит должен быть определен как экземпляр объекта rdfs:Class. Таким образом, объекты rdfs:Resource, и rdfs:Class определяются рекурсивно посредством друг друга – случай нередкий в языках описания онтологий.

OWL (Web Ontology Language [12]) представляет собой язык, предназначенный для описания онтологий и разработанный консорциумом W3 специально для этих целей. OWL построен как расширение RDF и RDFS. Это означает, что основная конструкция – это тройка языка RDF. В этом контексте язык OWL можно рассматривать как расширенный вариант RDFS, позволяющий не только описывать классы и свойства, но также задавать ограничения на их использование. На языке дескрипционной логики это означает, что логика, лежащая в основе OWL, содержит кроме описания отношений также и аксиомы, задающие соотношения между данными отношениями и различного рода ограничения последних.

Базовым элементом языка OWL является класс всех классов, определяемый как owl:Class. Любой OWL-класс должен быть задан как экземпляр класса owl:Class.

В языке OWL также присутствуют два предопределенных класса:

* Класс owl:Thing (сущность), обозначающий множество всех индивидов.
* Класс owl:Nothing (ничто), обозначающий пустое множество.

Каждый класс OWL является дочерним классом класса owl:Thing и родительским классом класса owl:Nothing. Наследование классов в языке OWL задается с помощью конструкции rdfs:subClassOf.

В OWL существует разделение свойств на два класса:

* Объектные свойства используются для связывания индивидов друг с другом. Объектные свойства – это экземпляры класса owl:ObjectProperty.
* Свойства типов данных связывают индивидов с так называемыми значениями типов данных (data values). Под значениями здесь подразумеваются RDF-литералы, или типы данных, определенные в XML Schema. Свойства типов данных – это экземпляры класса owl:DatatypeProperty.

Классы owl:ObjectProperty и owl:DatatypeProperty являются дочерними классами класса rdf:Property.

Язык OWL позволяет описывать различные характеристики классов и свойств, которые обычно задаются как разного рода ограничения на структуру связей между своими экземплярами. Эти ограничения выражаются в виде предопределенных соотношений, называемых в языке OWL аксиомами. В этом состоит основное отличие языка OWL от RDFS. Эти ограничения позволяют выражать в онтологии более тонкие вещи, чем с помощью RDFS.

Таких ограничений в языке OWL множество. Но все они подобраны таким образом, чтобы не снизить производительность алгоритма логического вывода по фактам, которые описаны в онтологии. Более подробную информацию о языке OWL можно найти в [12].

## Редактор онтологии

Для создания и редактирования онтологий используются специальные программы – редакторы онтологий. В данный момент в сети Интернет представлено множество онтологических редакторов, большинство из которых кроссплатформенны, то есть могут быть установлены на компьютер с любой операционной системой.

Самые известные из них – Ontolingua, Protégé, OntoEdit, OilEd, WebOnto, OntoSaurus, HOZO и др. [13] Для работы этих онторедакторов необходимо установленное Java ПО, которое бесплатно доступно в Интернете. Функциональность онторедакторов можно определить по следующим параметрам:

* осуществление поиска по онтологии;
* редактирование (ввод, корректировка, удаление);
* логический контроль при вводе;
* тестирование функциональности;
* взаимодействие с другими онтологиями.

Был выбран редактор Protégé, разработанный в Стэнфордском университете (Stanford University) под руководством Марка Мьюсенадля построения онтологий и баз знаний. Редактор Protégé доступен для бесплатного скачивания на официальном сайте [14]. Этот редактор поддерживает язык OWL в последней редакции. Также на официальном сайте [14] можно найти полную документацию по использованию этого редактора. Еще одним важным аспектом является наличие большого количества плагинов, один из которых – OWLViz – позволяет визуализировать онтологию. Также для этой цели служить инструмент OntoGraf для представления онтологии в виде ориентированного графа. Для проверки онтологий на синтаксическую правильность в последней версии Protégé имеются несколько программ семантической проверки, так называемых резонеров (reasoner) для языка OWL DL: Fact++, Pellet, HermiT 1.3.7 и др.

# Обзор существующих онтологий

## Онтология Pizza

Эта онтология описывается в документации по созданию онтологий в Protégé [14] и является распространенным примером для описания возможностей данной программы.

В ней описывается онтология пицц. На рисунке 1 представлена схема онтологии, созданная при помощи плагина программы Protégé OWLViz. Любая пицца (Pizza) состоит из основы (PizzaBase) и начинки (PizzaTopping). В качестве основы рассматриваются два вида: DeepPanBase и ThinAndCrispyBase. Разнообразных начинок гораздо больше, поэтому они объединены в подклассы. Например, подкласс CheeseTopping содержит все виды сырных начинок, а подкласс FishTopping – все виды рыбных начинок и т.д. Также есть подклассы MeatTopping (мясные начинки), FruitTopping (фруктовые начинки), HerbSpiceTopping (начинки из трав и специй), NutTopping (ореховые начинки), SauceTopping (соусы) и VegetableTopping (овощные начинки). Все эти подклассы не могут пересекаться, то есть ни один индивид не может входить в два или несколько этих подклассов одновременно, поэтому между ними существует отношение DisjointWith (не пересекается с).

В классе Pizza существует подкласс NamedPizza, в котором подклассы – названия различных пицц. У каждой пиццы есть основа и есть начинки, поэтому существует два основных отношения или свойства: hasBase (имеет основу) и hasTopping (имеет начинку). Два этих свойства находятся в отношении SubPropertyOf к свойству hasIngredient (иметь ингредиент). Ко всем этим свойствам определены противоположные свойства (то есть находящиеся в отношении InverseOf): isIngredientOf, isBaseOf и isToppingOf. У каждого отношения (свойства) определены Domains – множество классов, на котором определено это отношение и Ranges – множество классов, являющееся областью значений. Например, для отношения hasTopping в качестве Domains выступает класс Pizza, а в качестве Range – класс PizzaTopping.

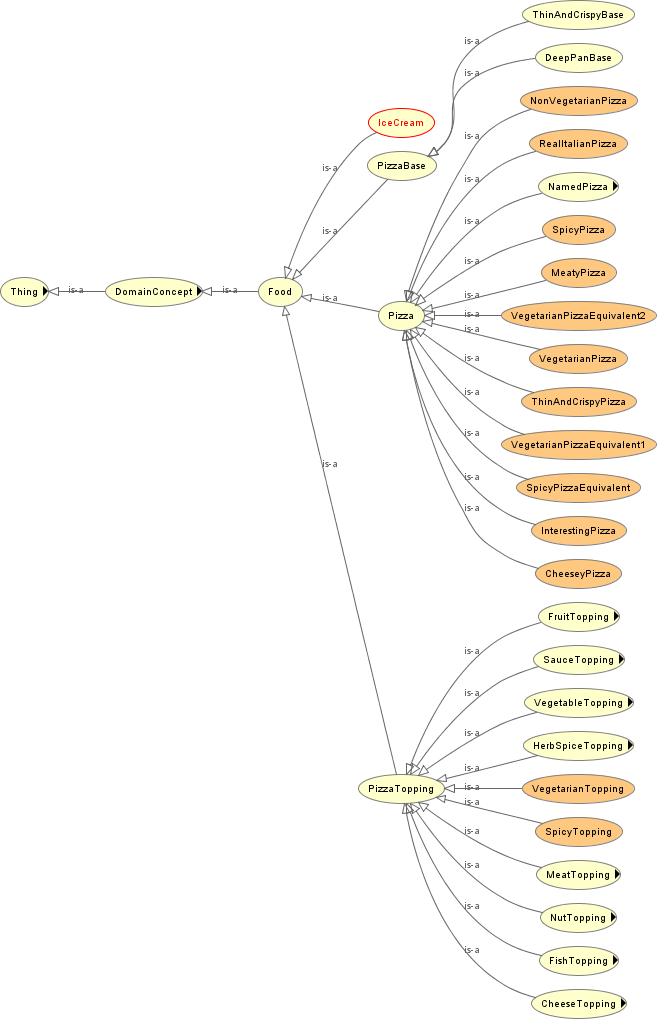


Рисунок 1 – Схема онтологии Pizza

Также в классе Pizza есть подклассы, определенные с помощью отношения EquivalentTo (эквивалентности) некоторому логически заданному классу. Например, класс CheeseyPizza EquivalentTo Pizza and (hasTopping some CheeseTopping), это означает, что класс CheeseyPizza является пересечением двух классов: Pizza и (hasTopping some CheeseTopping). Второй класс представляет собой логический класс, объединяющий всех индивидов, которые имеют хотя бы какую-то начинку из класса CheeseTopping. Аналогичным образом, например, определяется класс MeatyPizza EquivalentTo Pizza and (hasTopping some MeatTopping).

Для определения вегетарианской пиццы используется следующее выражение:VegetarianPizza EquivalentTo Pizza and (not (hasTopping some FishTopping)) and (not (hasTopping some MeatTopping)). То есть вегетарианская пицца – это пицца, которая не имеет рыбных и мясных начинок в своем составе. После этого легко определяется противоположный класс невегетарианской пиццы: NonVegetarianPizza EquivalentTo Pizza and (not (VegetarianPizza)).

Как уже говорилось, в классе Pizza есть подкласс NamedPizza с названиями наиболее распространенных пицц. Для каждой пиццы описывается ее состав. Например, пицца American hasTopping only (MozzarellaTopping or PeperoniSausageTopping or TomatoTopping), hasTopping some MozzarellaTopping, hasTopping some PeperoniSausageTopping, hasTopping some TomatoTopping. Это означает, что пицца American имеет в качестве начинки сыр Mozzarella, колбаса Paperoni, томаты и ничего другого. Исходя из этих сведений устройство логического вывода, в программе Protégé это OWL Reasoner, делает вывод, что пицца American является подклассом для следующих классов: CheeseyPizza, так как имеет в своем составе сыр, MeatyPizza, так как имеет в своем составе колбасу, InterestingPizza, так как имеет в своем составе не менее трех ингредиентов.

Эта онтология может быть использована в ресторанах, кафе и других заведениях, где готовят пиццу. Ее повсеместное использование принесет некоторую систематичность и будет гарантировать, что при заказе пиццы того или иного наименования, заказчик будет знать, что входит в ее состав, и будет уверен, что он получит то, что хочет, вне зависимости от заведения, города или даже страны.

## Онтология Wine

Эта онтология представляет собой онтологию вин и еды, а также подходящие комбинации вин и блюд. Она соединяет в себе две большие онтологии: food (блюда) и wine (вин). В отличие от предыдущей рассмотренной онтологии, данная онтология имеет значительно больше взаимосвязей между понятиями, а также наполнена экземплярами (индивидами), что объясняется более детальным и тщательным подходом к ее разработке и соответствует поставленным целям. Эту онтологию можно будет использовать как основу для приложений в наборе инструментов для управления рестораном. Одно приложение могло бы составлять список вин для меню на текущий день или отвечать на запросы официантов и посетителей. Другое приложение могло бы анализировать инвентарный перечень винного погреба и предлагать категории вин для пополнения и конкретные вина для закупки к следующим меню или для поваренных книг.

В классе Wine есть множество подклассов, которые объединяют в себе различные вина по месту производства (стране), по цвету, по вкусу и т. д. Для описания всех вин используется класс WineDescriptor (описание вин). Подклассы этого класса: WineColor (цвет), определяемый индивидами Red (красное), Rose (розовое), White (белое); WineBody (крепкость); WineFlavor (аромат); WineSugar (наличие сахара). Также для описания происхождения вин используется класс Region, в котором в качестве индивидов перечислены различные районы производства вин. Как уже говорилось, к этой онтологии вин прикреплена еще и онтология блюд. Для построения отношений между винами и блюдами существует два отношения (свойства): hasDrink (употребляется с напитком), hasFood (употребляется с едой). Первое свойство связывает какое-то блюдо с каким-то конкретным вином или классом вин, а второе свойство, обратно, связывает какое-то вино с блюдами, которые рекомендуется употреблять с этим вином. Например, блюда из рыбы (FishCourse) рекомендуется употреблять с винами средней крепости (hasDrink only (hasBody value Medium)) и с сухими винами (hasDrink only (hasSugar value Dry)). На рисунке 2 приведена схема онтологии Wine, построенная OWLViz.

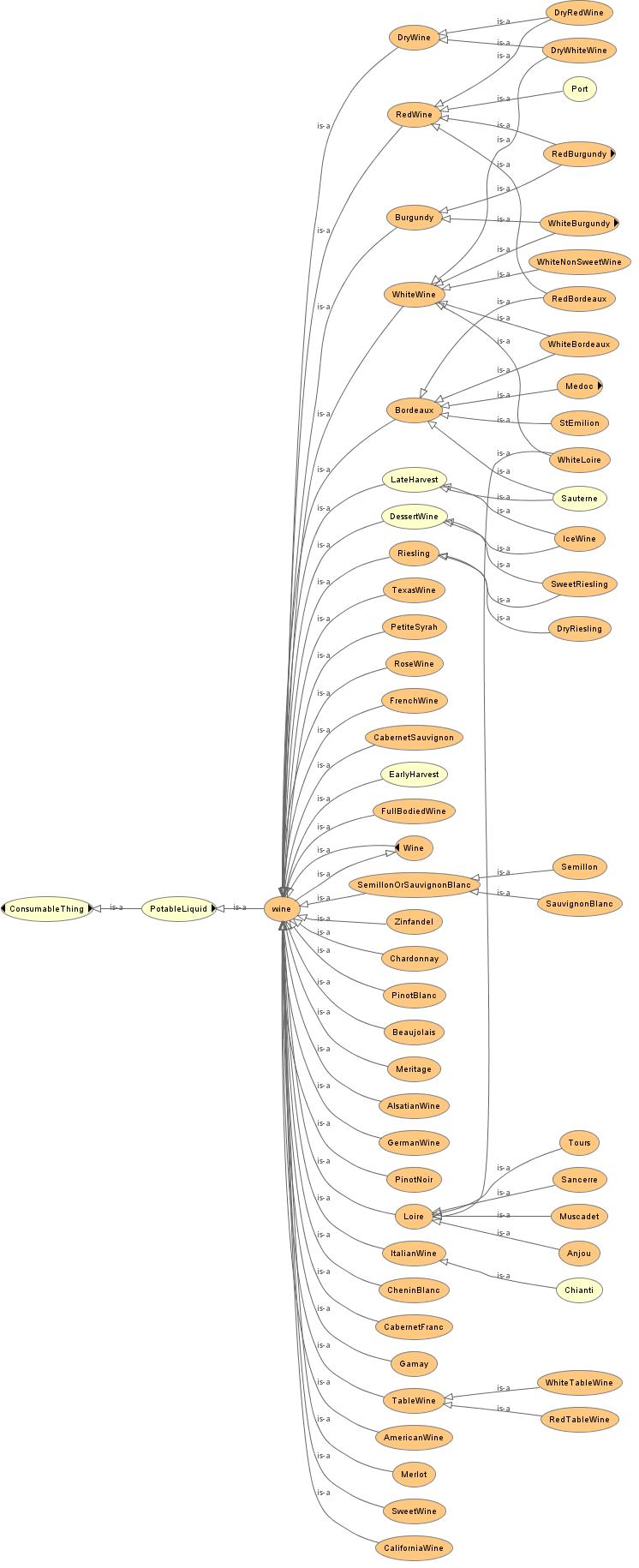


Рисунок 2 – Схема онтологии Wine

## Другие онтологии

Существует множество разработанных онтологий в различных отраслях и сферах. Также существую онтологии, которые являются объединением ранее построенных онтологий.

Например, Biochemical ontologies (биохимические онтологии), содержащие более 30 онтологий, которые описывают атомы, молекулы, связи и процессы, связанные с ними.

Онтология camera описывает различные части фотокамеры и их взаимосвязи друг с другом.

Онтология Finance представляет собой онтологию по финансовым инструментам, а также процессам и процедурам, связанным с ценными бумагами.

Infrastructure Product Ontology – онтология, описывающая коммунальную инфраструктуру и охватывает пять секторов коммунальных услуг: водоснабжение, канализация, газ, электричество и связь.

Protein Ontology – онтология белков, описывающая все декларативные знания о них, в том числе их классификацию, что позволяет строить логические рассуждения.

Tourism – онтология для создания семантики веб-сайтов, связанных с туризмом.

Таких примеров большое количество, они доступны в интернете и могут применяться в различных информационных системах. Их тематика обширна и безгранична. Но остается еще много областей, в которых онтологии еще не получили широкого распространения или вовсе не применялись. Как уже ранее говорилось, онтологии могут существенно упростить работу и взаимодействие различных, распространенных в наше время информационных систем друг с другом. Поэтому проектирование онтологий на данный момент является достаточно актуальным направлением и приобретает все большую популярность.

# Разработка онтологии редакции газеты

## Описание предметной области

Было принято решение о разработке онтологии редакции газеты, так как в настоящее время крупные редакции газет часто используют информационные системы для регулирования издательского процесса. Основные цели ее создания состоят в следующем:

1. Описать структуру редакции, основные роли и функции сотрудников.
2. Определить взаимосвязь статей, издаваемых в газете, и сотрудников, имеющих к ним отношения.

В качестве источника информации о структуре и процессе работы редакции был взят учебник [15], а также официальные сайты крупных газет «АиФ» [16] и «Российская газета» [17].

Процесс подготовки и выпуска газеты в сильнейшей степени зависит от особенностей структуры – строения – коллектива редакции. Эту структуру свободно определяет ее руководитель – главный редактор. При этом он исходит из величины и состава коллектива, типа и уровня издания, характера его читательской аудитории. Оптимальная структура редакции должна способствовать управлению всеми ее подразделениями и их безотказной деятельности. Это возможно лишь, во-первых, при наличии – как в любой функционирующей системе – центра управления коллективом. Во-вторых, при обеспечении свободного притока в редакцию информации, без которой невозможно как управление ее коллективом, так и деятельность его звеньев. И, в-третьих, при наличии в редакции прямой и обратной связи между всеми ее подразделениями и сотрудниками. Прямой связи – сверху вниз – от редактора, редколлегии, ответственного секретаря, генерального менеджера в отделы, к корреспондентам, техническим и коммерческим работникам – в виде планов, приказов, распоряжений, заданий и т.п. И обратной связи – снизу вверх – из отделов, технических и коммерческих служб в секретариат, редколлегию, главному редактору, генеральному менеджеру – информация о выполнении планов, решений и заданий и встречные предложения.

Структура редакции формируется и под воздействием таких субъективных факторов, как принципы редакционной деятельности. Это прежде всего принцип разделения труда. В соответствии с ним весь коллектив делится на части и звенья, у каждого из которых свои задачи и обязанности, определяющие направление и характер их деятельности. Это относится и к каждому творческому, техническому и коммерческому сотруднику редакции. Столь же важен и второй принцип – кооперирования труда и объединения его результатов. Он определяет итоги работы всей редакции: создание коллективного журналистского произведения – номера периодического издания, в который каждый сотрудник вносит свой вклад.

Третий принцип – единоначалия – важен как для руководителей редакции, так и для их подчиненных. В соответствии с ним каждый член редакционного коллектива имеет только одного непосредственного руководителя. Корреспондент подчиняется заведующему отделом, который, в свою очередь, получает указания от ответственного секретаря редакции и т.д. Все работники редакции имеют одного руководителя – главного редактора. Руководителю этот принцип облегчает управление тем подразделением, которое он возглавляет. Он получает право решать все вопросы, связанные с деятельностью этого отдела или службы, осуществлять контроль работы их сотрудников. А подчиненные ему знают, кому они обязаны предъявить результаты своего труда, и кто дает им задания и планирует их деятельность.

Важнейшее значение для деятельности редакционного коллектива имеет функциональное разделение труда. Оно определяет как структуру редакции, так и организацию ее работы и приводит к разделению коллектива на несколько частей. Первая из них – творческая часть редакции**.** В нее входят журналисты, которые выполняют функцию подготовки и выпуска периодического издания: написания и редактирования его материалов, работы с внередакционными авторами, компоновки готовых материалов газеты в отдельные ее выпуски, обеспечения их дизайна – иллюстрирования и верстки, руководства редакцией и процессом производства номера и т.д. К ней примыкает техническая часть коллектива**.** Ее составляют технические сотрудники. Они, с одной стороны, создают оптимальные условия для деятельности журналистов, с другой – обеспечивают техническую подготовку и выпуск номера – держат корректуру, при наличии компьютерной системы производят набор текстов, верстку полос и т.д.

С возникновением в нашей стране информационного рынка в редакциях периодических изданий возникла коммерческая часть коллектива**.** Ее работники стремятся укрепить экономическую базу редакции, обеспечить реализацию тиража газеты, привлечь в нее рекламные и частные объявления и т.д.

В результате в редакционном коллективе формируются три функциональные группы. С понижением уровня этой структуры функциональное разделение труда приобретает все более специализированный характер. Каждая из трех групп разделяется на несколько подразделений. Так, творческая часть редакции формируется из двух звеньев. Первое – звено руководства и управления. В него входят: редакторы – главный редактор и его заместители, редакционная коллегия и секретариат. Второе – звено исполнения, которое охватывает корреспондентов, сгруппированных в отделы редакции, обозревателей, специальных и собственных корреспондентов газеты. Внутри каждого из звеньев происходит дальнейшее разделение труда. Например, редакторат и редколлегия осуществляют общее руководство редакцией, секретариат представляет собой центр оперативного управления коллективом. Важнейшими элементами звена исполнения являются творческие отделы, где создаются тексты и иллюстрации, публикуемые в газете. В технической части редакции функциональное разделение труда приводит к образованию ряда служб – корректорской, компьютерного центра, редакционной библиотеки и др. А работники коммерческой части коллектива сосредоточены в ее подразделениях – отделах рекламы, распространения, маркетинга и других, где и решают стоящие перед ними специфические задачи.

Так в процессе разделения труда формируется пирамидальная структура редакционного коллектива. Наверху нее стоит руководитель – главный редактор, ниже – его заместители, редколлегия, секретариат, а под ними располагаются творческие отделы, технические и коммерческие службы.

Процесс подготовки и выпуска номера печатного периодического издания делится на несколько этапов. Это подготовка информации, формирование номера и его выпуск. В свою очередь подготовка журналистской информации, предназначенной для определенного номера газеты, подразумевает три последовательных операции: отбор информации, ее редактирование, техническую подготовку к публикации.

Важнейшее значение имеет отбор информации**.** Он производится в отделах редакции и в ее секретариате. Информацию может отбирать и ведущий редактор номера, отвечающий за его содержание. Отбор этот идет с использованием нескольких основных источников. Первый из них – оперативная информация, доставляемая корреспондентами и собкорами редакции, а также информационными агентствами. Второй источник – запас готовых материалов, хранящихся в «портфелях» отделов и секретариата.

Отобранная информация должна быть подготовлена к публикации. Эта подготовка происходит, прежде всего, в процессе редактирования текстов. Оно осуществляется на всем пути информации в редакции до ее публикации и носит коллективный характер.

В редактировании каждого текста обычно участвуют несколько журналистов, представляющих разные подразделения и службы редакции. Кроме самого автора – его коллега по отделу или его руководитель, приводящий текст в соответствие с планом отдела; сотрудник секретариата или ответственный секретарь, проверяющий правдивость и точность информации, содержащейся в тексте, и при необходимости сокращающий его размеры; ведущий редактор или сам главный редактор, оценивающий значение информации и определяющий ее место в номере и комплекте издания; и корректор, сигнализирующий в секретариат и автору о смысловых и стилистических ошибках, замеченных в тексте.

Подготовленные для публикации материалы (тексты и иллюстрации) составляют запас – тот массив журналистской информации, который используется для формирования очередного номера печатного периодического издания в секретариате редакции.

## Проектирование основных понятий онтологии

Первым шагом в создании онтологии редакции газеты был поиск понятий, связанных с данной областью. Главными понятиями соответственно являются «редакция» и «газета». Под редакцией понимается как коллектив работников, отделы редакции, так и все результаты труда этого коллектива, то есть непосредственно газета и соответственно все статьи, иллюстрации и реклама, представляющие содержимое газеты. Таким образом, были созданы соответствующие классы с вполне естественной иерархией в программе Protégé в окне Class hierarchy, что изображено на рисунке 3.

Все создаваемые классы являются подклассом предопределенного класса Thing, обозначающего множество всех индивидов. Также следует помнить, что существует противоположный класс Nothing. Одним из признаков нелогичности онтологии является вывод, что какой-нибудь подкласс класса Thing является эквивалентным классу Nothing. В этом случае следует проверить все отношения связанные с этим подклассом, так как в них есть противоречие.

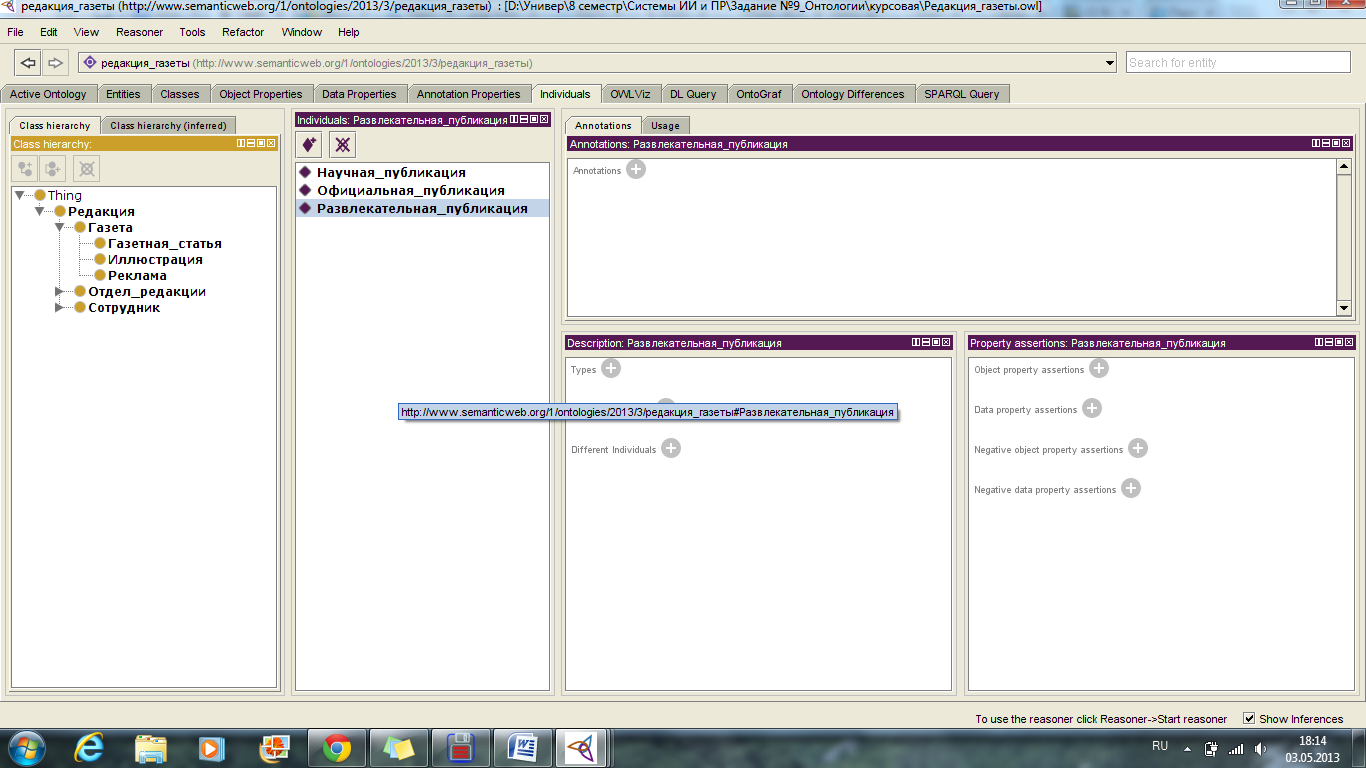


Рисунок 3 – Основные понятия редакции газеты

Коллектив редакции, то есть ее сотрудники делятся на две большие группы: штатные и внештатные сотрудники. В качестве внештатных сотрудников могут выступать различные государственные и социальные организации, информационные агентства, а также внештатные журналисты. Класс штатных сотрудников лучше разделить на две большие группы: редакционная коллегия (управленческое звено) и исполнители. В редакционную коллегию, как уже говорилось в предыдущем пункте, входят главный редактор, заместитель главного редактора, ответственный секретарь и секретариат. В качестве исполнителей выступают редакторы, дизайнеры, корректоры, технические работники, менеджеры по рекламе, а также самый обширный класс – журналисты, в котором в свою очередь есть подклассы: комментатор, обозреватель, корреспондент или репортер. Так как корреспондент и репортер для данной онтологии имеют практически одно и то же значение, то они объявлены эквивалентными (в Protégé это отношение Equivalent To). В свою очередь репортеры бывают разных направлений: военный, спортивный, фоторепортер. Это также следует учесть, так как их деятельность различна.

Все перечисленные исполнители – это совершенно разные специальности (кроме корреспондента и репортера, которые эквивалентны), эти классы не могут пересекаться, поэтому они находятся в отношении DisjointWith. Если поставить такое же отношение между корреспондентом и репортером, то логическое устройство (Reasoner) обнаружит конфликтную ситуацию, что отразится выделением красным цветом этих двух классов. В самом деле, если классы эквивалентны, то любой экземпляр, который относится к классу корреспондент, будет также экземпляром класса репортер, и наоборот. А указание на то, что они не пересекаются, говорит о противоположном отношении, поэтому это логическая ошибка, на которую и указывает устройство логического вывода.

В итоге для сотрудников получилась следующая иерархия, представленная на рисунке 4, которая также создается в окне Class hierarchy.

Как видно из этой иерархии, класс журналистов присутствует как подкласс штатных сотрудников, так и подкласс внештатных сотрудников. При создании второго класса с таким же именем, логическое устройство программы Protégé выдает сообщение, что такой класс уже существует, и в этом случае необходимо либо переименовать создаваемый класс, либо согласится с тем, что это один и тот же класс. В нашем случае эти понятия одинаковы, они отличаются только принадлежностью к разным классам.

В программе Protégé можно создавать комментарии, которые будут использоваться исключительно людьми, заинтересованными в этой онтологии. Эти комментарии не влияют на логическое представление понятий, но позволяют любому человеку сориентироваться в онтологии и понять используемые термины. Также в комментарии можно привести синонимы понятия на этом же языке, в нашем случае русском, или на любом другом языке, например, английском. Пример реализации комментария для сотрудника обозревателя приведен на рисунке 5.

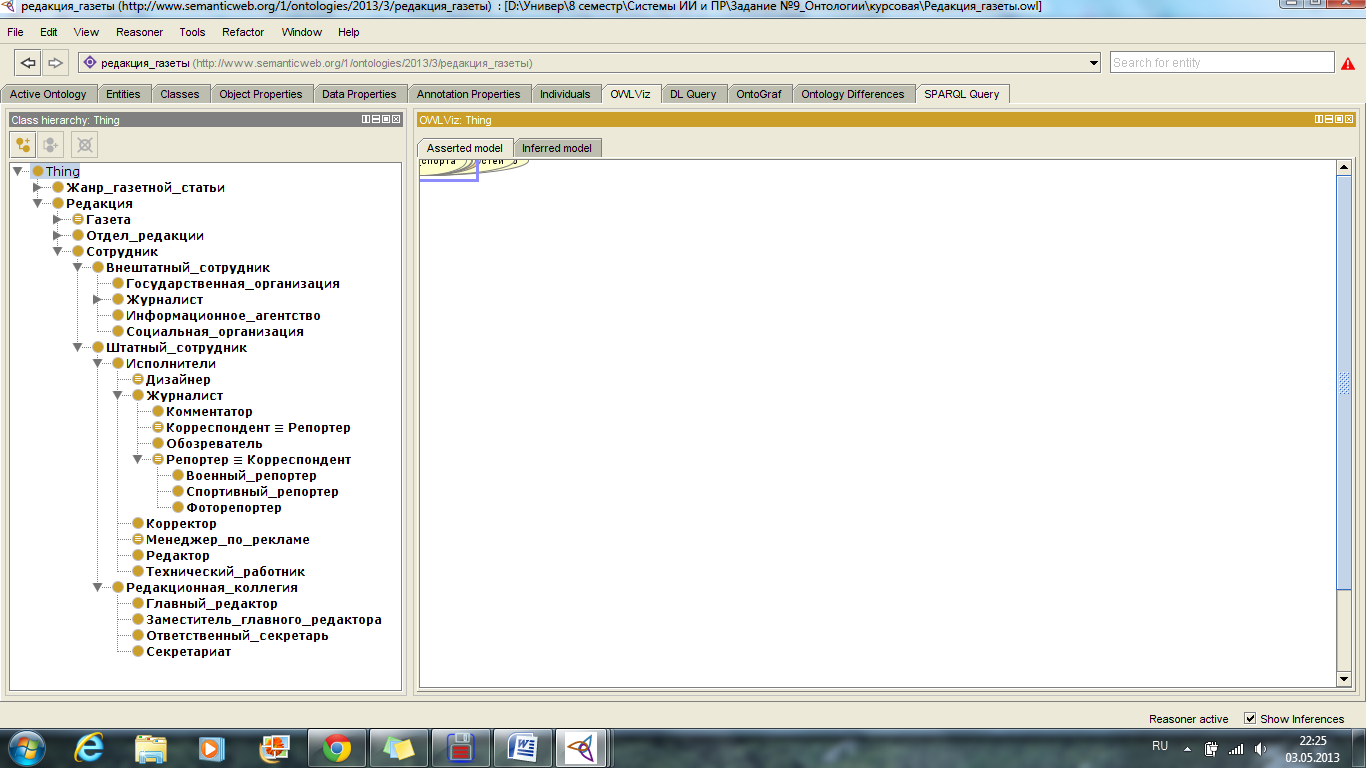


Рисунок 4 – Иерархия сотрудников редакции

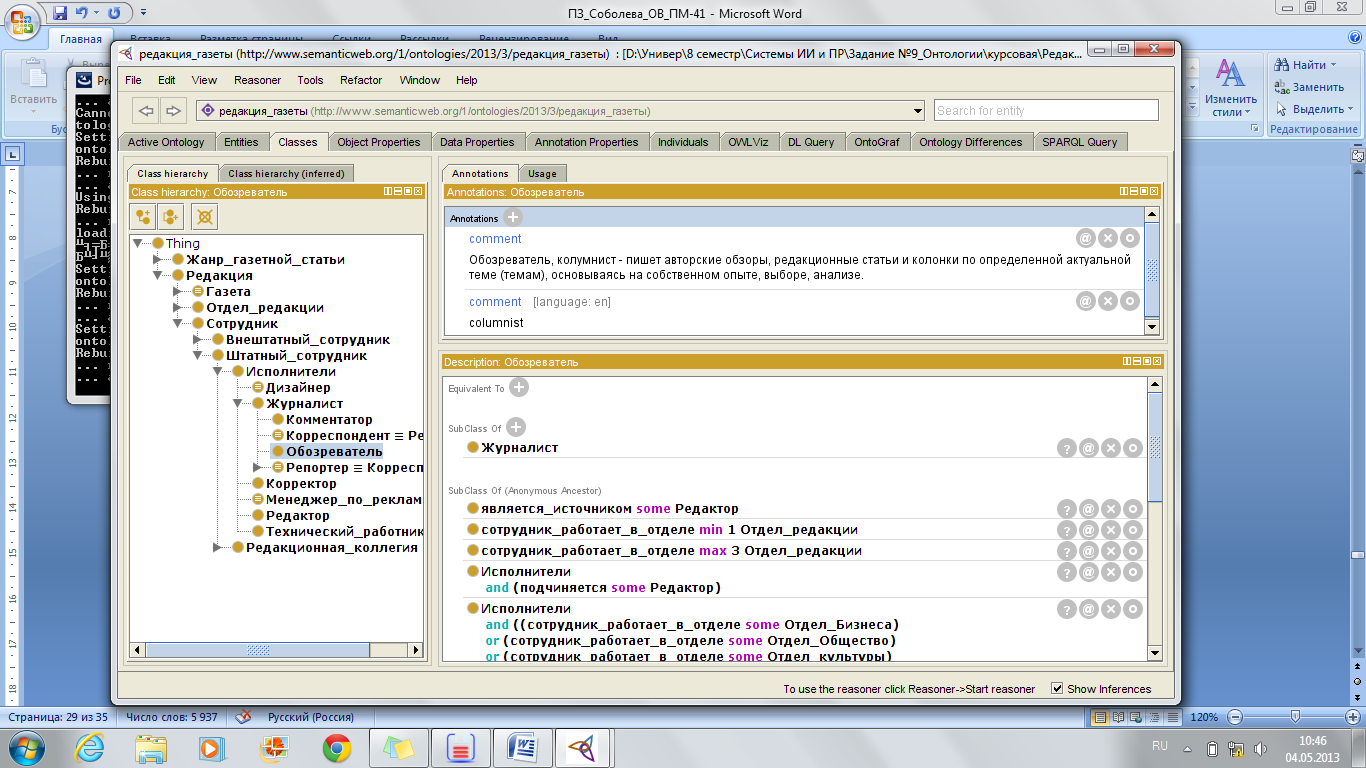


Рисунок 5 – Комментарий к классу «Обозреватель»

Работа сотрудников редакции газеты организована по отделам, то есть существуют определенное разделение работы по типу этой работы или по тематике. Основываясь на сведениях о двух известных крупных редакциях в нашей стране [16], [17], были сформированы представленные на рисунке 6 классы, обозначающие различные отделы.

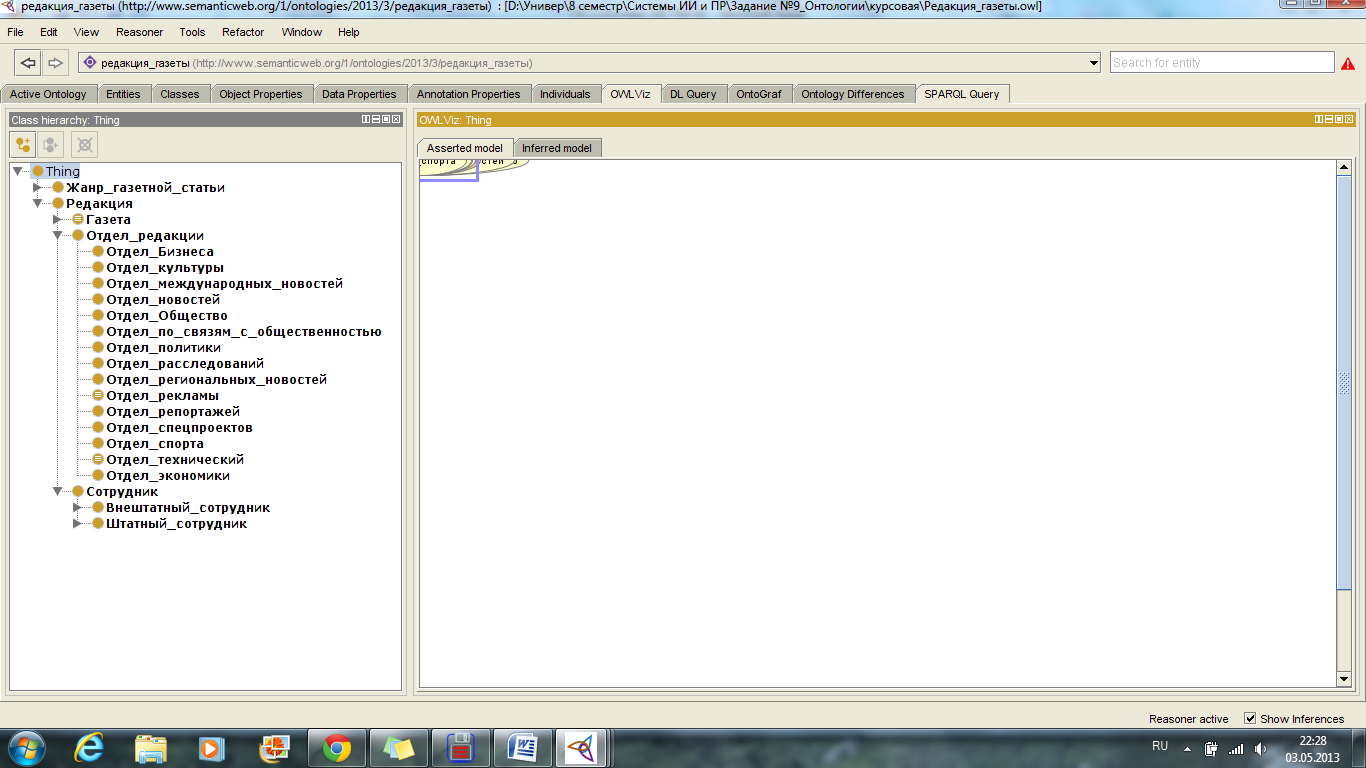


Рисунок 6 – Список отделов редакции

Эти классы также являются непересекающимися, поэтому все находятся в отношении DisjointWith друг с другом.

В газетах встречаются разнообразные жанры статей. В наиболее крупных редакциях сотрудники могут специализироваться по какому-нибудь конкретному жанру или по нескольким. Также жанровое разнообразие статей влияет на интерес читателей к газете, поэтому при формировании очередного номера необходимо учитывать это, наполняя газету как информационными жанрами: интервью, заметками, так и аналитическими: комментариями, рецензиями, статьями и т. д. Исходя из этого, было решено создать класс «Жанр\_газетной\_статьи», в котором присутствуют три подкласса: «Информационный\_жанр», «Аналитический\_жанр» и «Художественно-публицистический\_жанр». Так как к этим группам жанров относятся конкретные жанры, которые не имеют дальнейшего деления на классы или группы, то удобнее всего представить эти конкретные жанры в виде экземпляров (индивидов). Для этого в программе Protégé используется вкладка Individuals, в которой создаются экземпляры с именами всех известных жанров, что представлено на рисунке 7.

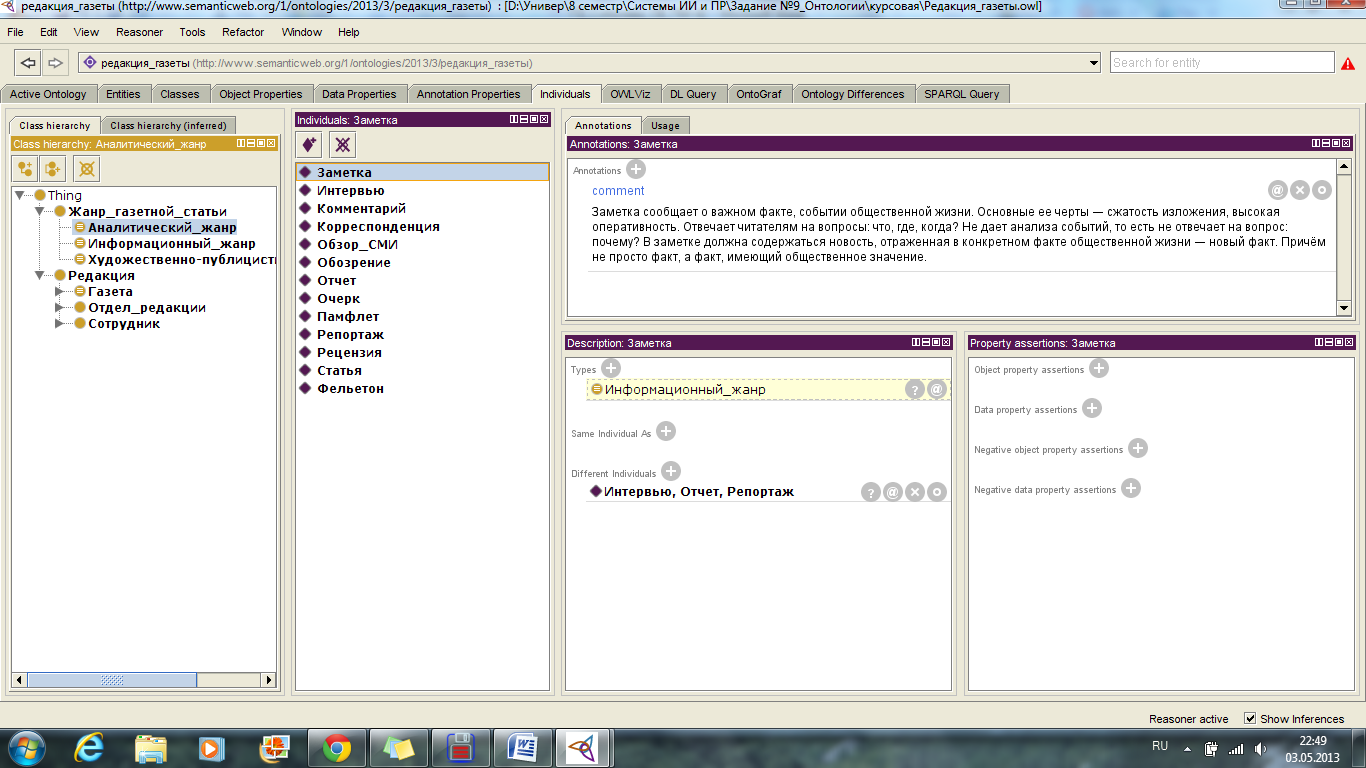


Рисунок 7 – Вкладка Individuals с созданными экземплярами жанров

Также для каждого экземпляра можно задать комментарий, в котором объясняется значение понятия, в данном случае описываются характерные черты того или иного жанра. Понятия жанров, относящихся к одному и тому же типу, не пересекаются друг с другом, поэтому они также находятся в отношении Disjoint Individuals.

Затем для определения класса «Информационный\_жанр» используется следующее определение: «Информационный\_жанр Equivalent To {Заметка, Интервью, Отчет, Репортаж}». Это означает, что класс «Информационный\_жанр» является классом, состоящим только из четырех приведенных индивидов. После запуска логического устройства вывода, для всех индивидов из этого класса автоматически определяется тип – «Информационный\_жанр». Это отображается в пункте Types соответствующих индивидов и подсвечивается специальным цветом, что говорит о том, что это не было указано явно, а является результатом логических рассуждений. Также результатом вывода является то, что для класса «Информационный\_жанр» в пункте Members устанавливаются все индивиды, связанные с этим классом, которые также подсвечиваются специальным цветом, что изображено на рисунке 8.

Следует отметить, что результаты логического вывода отражаются только при включенном устройстве логического вывода (Reasoner), в данном случае он является активным, о чем говорит надпись правом нижнем углу окна: Reasoner active.

Аналогичным образом можно определить классы «Аналитический\_жанр» и «Художественно-публицистический\_жанр».

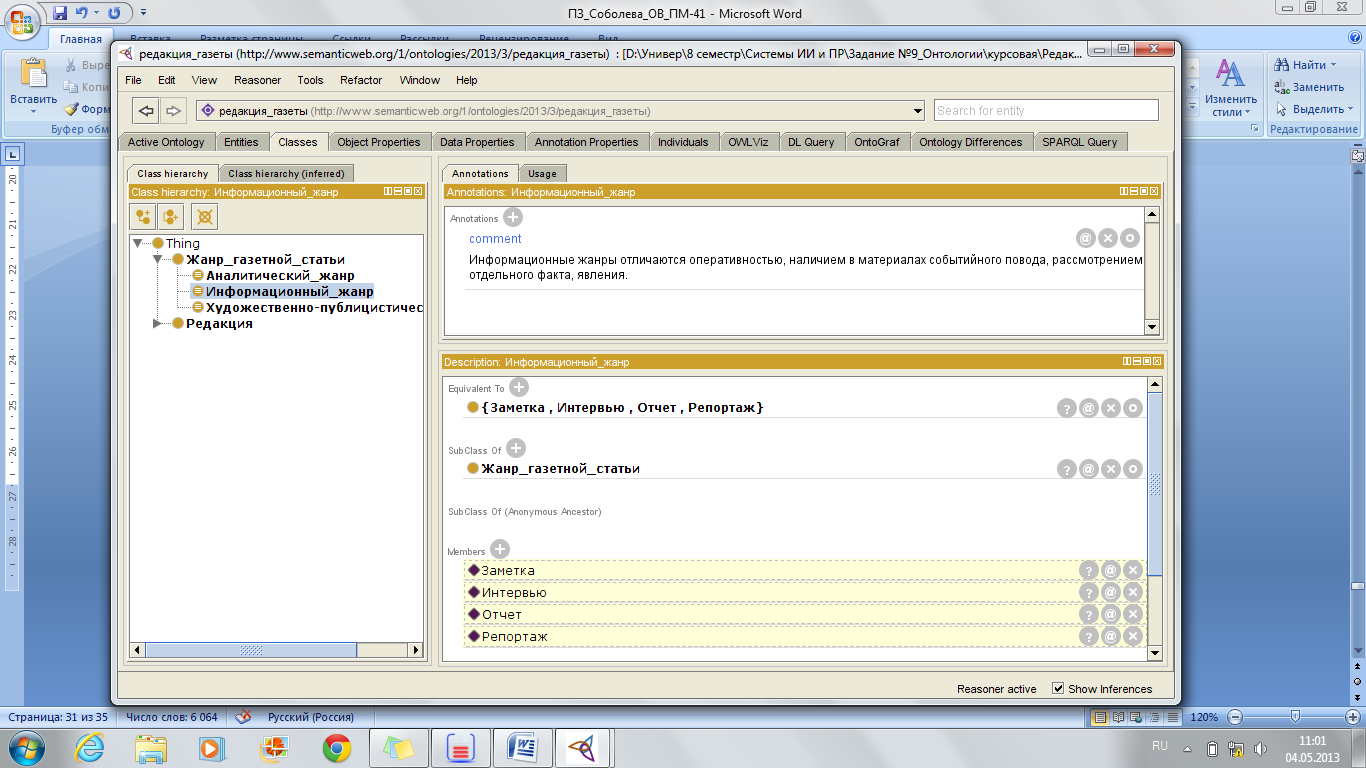


Рисунок 8 – Отражение экземпляров класса «Информационный\_жанр»

## Проектирование отношений между понятиями

В предыдущем подразделе были спроектированы основные понятия выбранной предметной области и базовые отношения между ними: иерархия, непересекаемость некоторых классов, эквивалентность классов между собой и определенному набору экземпляров. Далее следует определить более сложные отношения, которые характеризуют взаимосвязи созданных классов.

Для этого необходимо перейти на вкладку Object Properties. Все создаваемые отношения будут являться экземплярами предопределенного отношения topObjectProperty.

Первым делом определим отношение подчинения, так как созданная иерархия сотрудников этого не отражает. Для этого определим два противоположных отношения: «подчиняется» и «имеет\_в\_подчинении». Факт того что они являются противоположными определяется отношением Inverse Of. Также следует отметить, что эти отношения являются транзитивными, так как например редакторы подчиняются секретариату, который в свою очередь подчиняется ответственному секретарю, из этого следует, что редакторы подчиняются и ответственному секретарю тоже. Этот факт отражается установлением галочки напротив свойства Transitive в окне Characteristics.

Далее следует определить эти отношения между конкретными классами. Для этого необходимо перейти во вкладку Classes. Далее рассмотрим на примере главного редактора. Он является самым главным человеком в редакции и ему должны подчиняться все остальные сотрудники, но непосредственно ему подчиняются его заместители, ответственные секретари и секретариат. Поэтому логичнее всего определить главного редактора с помощью следующего выражения: Главный\_редактор Equivalent To (имеет\_в\_подчинении some Заместитель\_главного\_редактора) and (имеет\_в\_подчинении some Ответственный\_секретарь) and (имеет\_в\_подчинении some Секретариат). Соттветственно «Главный\_редактор» является подклассом следующих классов: (имеет\_в\_подчинении some Заместитель\_главного\_редактора), (имеет\_в\_подчинении some Ответственный\_секретарь), (имеет\_в\_подчинении some Секретариат). Все вышеперечисленные действия будут отражаться в Protégé как показано на рисунке 9.

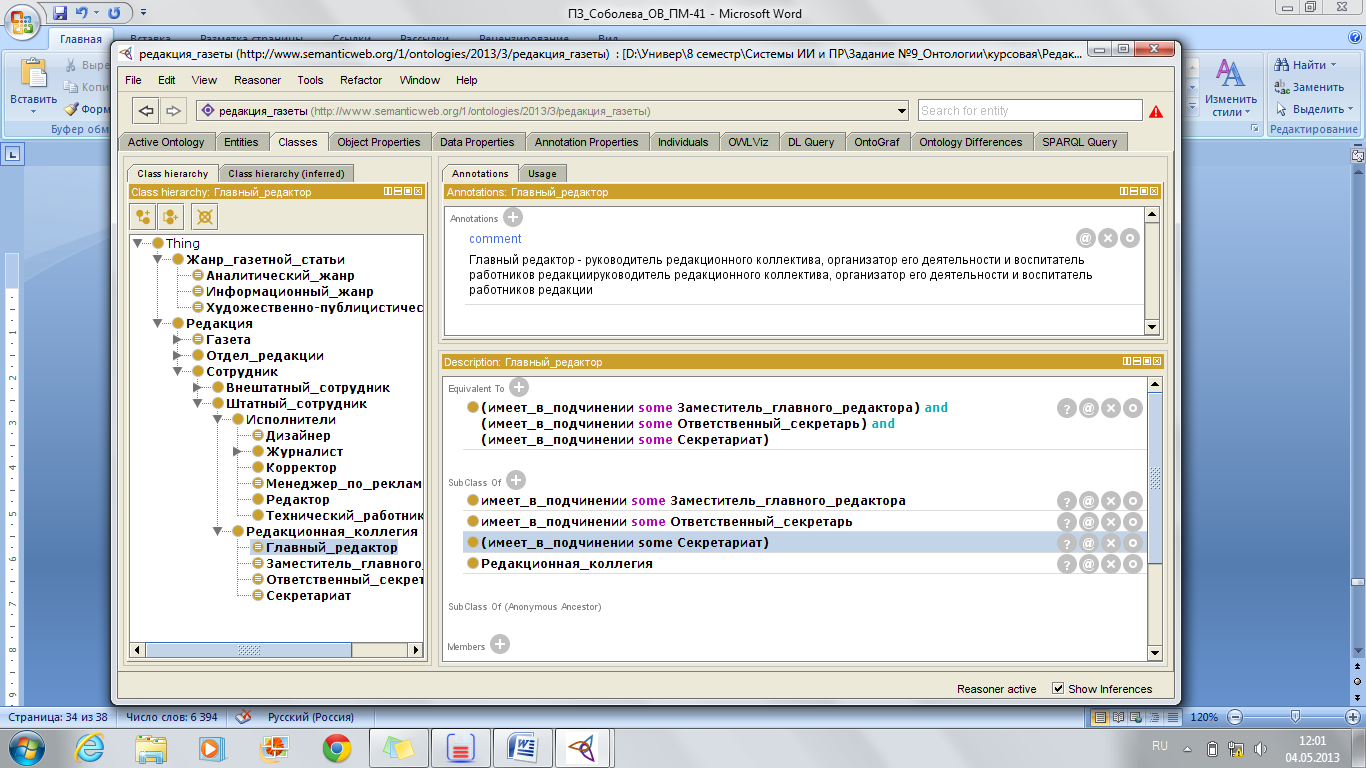


Рисунок 9 – Определение понятия главный редактор

Ключевое слово some означает, что экземпляры определяемого класса имеют хотя бы одно отношение к некоторым экземплярам указанного после ключевого слова some класса. Есть и другое ключевое слово only, оно означает, что определяемый класс имеет только указанное отношение к экземплярам следующего после ключевого слова класса, в данном случае это не подходит.

Исходя из аналогичных рассуждений, можно определить класс «Заместитель\_главного\_редактора», как показано на рисунке 10. В этом случае уже используется как отношение «имеет\_в\_подчинении», так и отношение «подчиняется». Также можно заметить, что определяемый класс является подклассом (подчиняется only Главный\_редактор), так как класс «Заместитель\_главного\_редактора» может иметь только отношение подчинения к классу «Главный\_редактор».

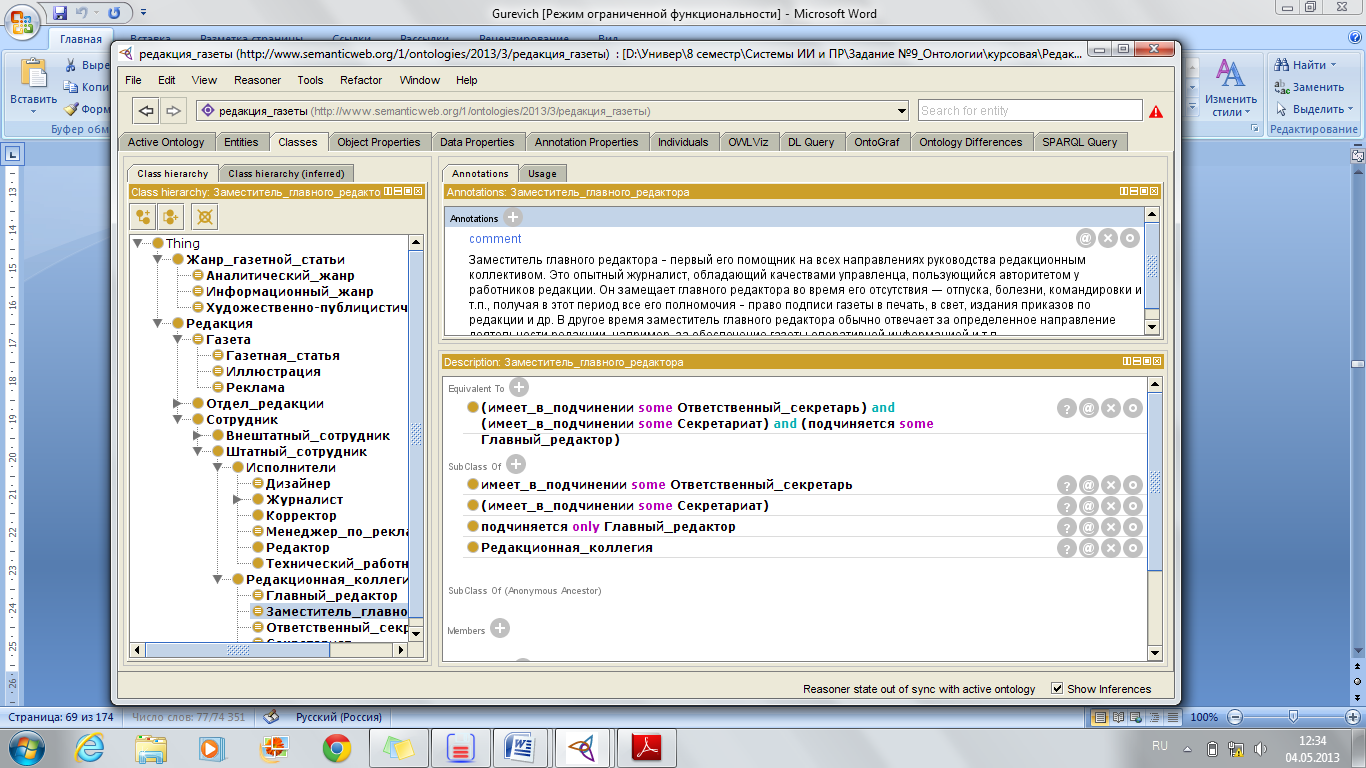


Рисунок 10 – Определение класса «Заместитель\_главного\_редактора»

Таким же образом определяются классы всех остальных сотрудников через отношения подчинения между собой.

Описывая структуру редакции, следует создать также и следующие два отношения: «в\_отделе\_работает\_сотрудник» и «сотрудник\_работает\_в\_отделе». Они также являются противоположными, то есть имеют отношение Inverse Of друг к другу. В отличие от предыдущих двух отношений, в которых было сложно задать область определения и область значений, в данных отношениях это осуществимо. Так для отношения «в\_отделе\_работает\_сотрудник» в качестве области определения (Domains) выступает класс «Отдел\_редакции», а в качестве области значений (Ranges) – класс «Сотрудник». Для противоположного отношения эти классы меняются местами. Далее было определено это отношение между сотрудниками и отделами редакции. Например, для технического работника это будет выглядеть достаточно просто (рисунок 11), так как он может работать только в техническом отделе.

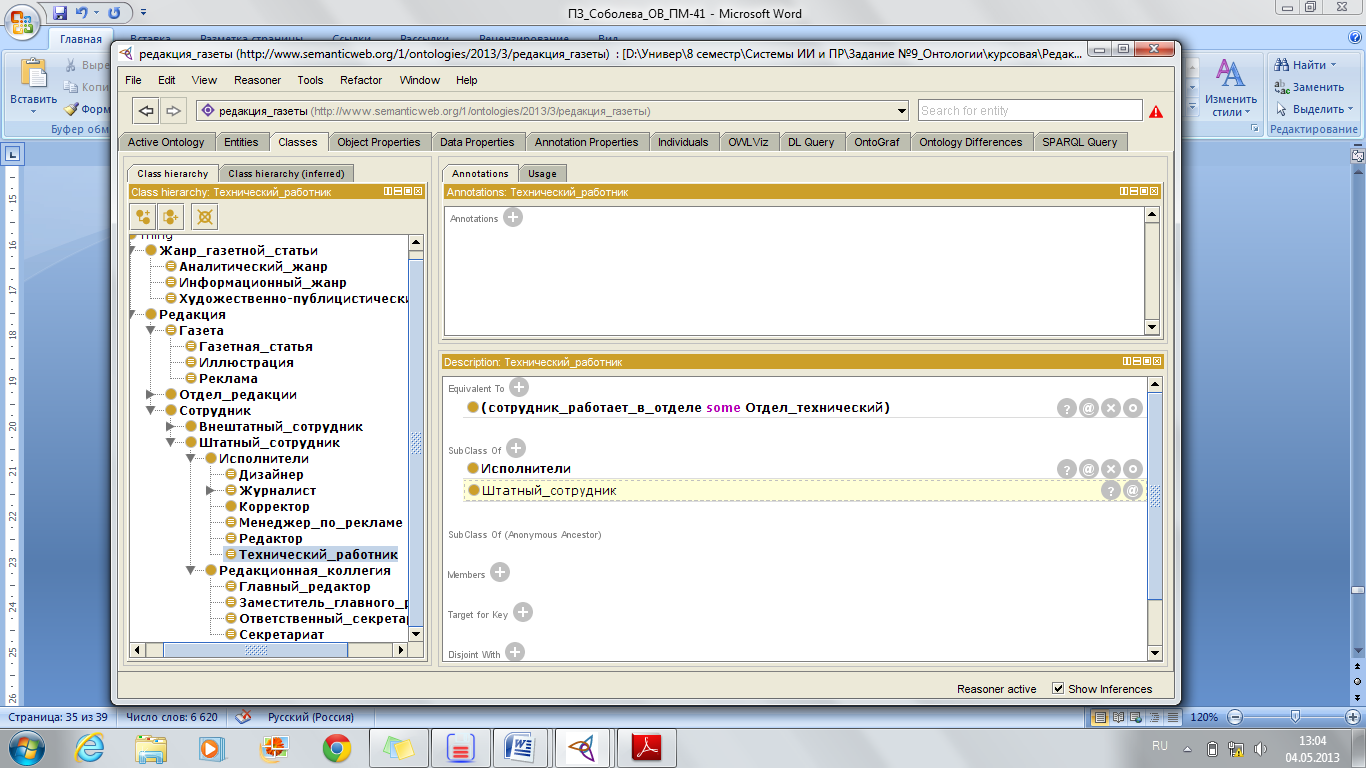


Рисунок 11 – Определение класса «Технический\_работник»

Для журналиста задание отношения к отделам редакции будет сложнее. Во-первых, журналист может работать не во всех отделах редакции, например, в техническом отделе или отделе по рекламе работают другие люди. Во-вторых, журналист может работать на несколько отделов, поэтому следует ввести некоторые ограничения на количество отделов, в которых он может работать. Учитывая эти факты, описание класса «Журналист» будет иметь следующий вид, представленный на рисунке 12.

Как видно из представленного на рисунке определения, оно достаточно громоздко, так как необходимо перечислять все классы, обозначающие те отделы, в которых может работать журналист. Это можно сделать другим способом с использованием вспомогательной частицы not: Журналист SubClass Of Исполнители and not(Отдел\_технический) and not(Отдел\_рекламы). Это говорит о том, что не существует однозначного способа описания понятий.

Но заданное определение для класса «Журналист» как подкласса «Штатный\_сотрудник» является определением и для подкласса «Журналист» в классе «Внештатный\_сотрудник», что противоречит реальной ситуации, так как внештатные сотрудники не могут работать в отделах редакции. Чтобы избежать этой ситуации, класс «Журналист» был удален из класса «Внештатный\_сотрудник» и создан новый класс «Внештатный\_журналист». Для внештатных журналистов не принципиально знать, на чем он специализируется, поэтому было решено не делить этот класс на подклассы.

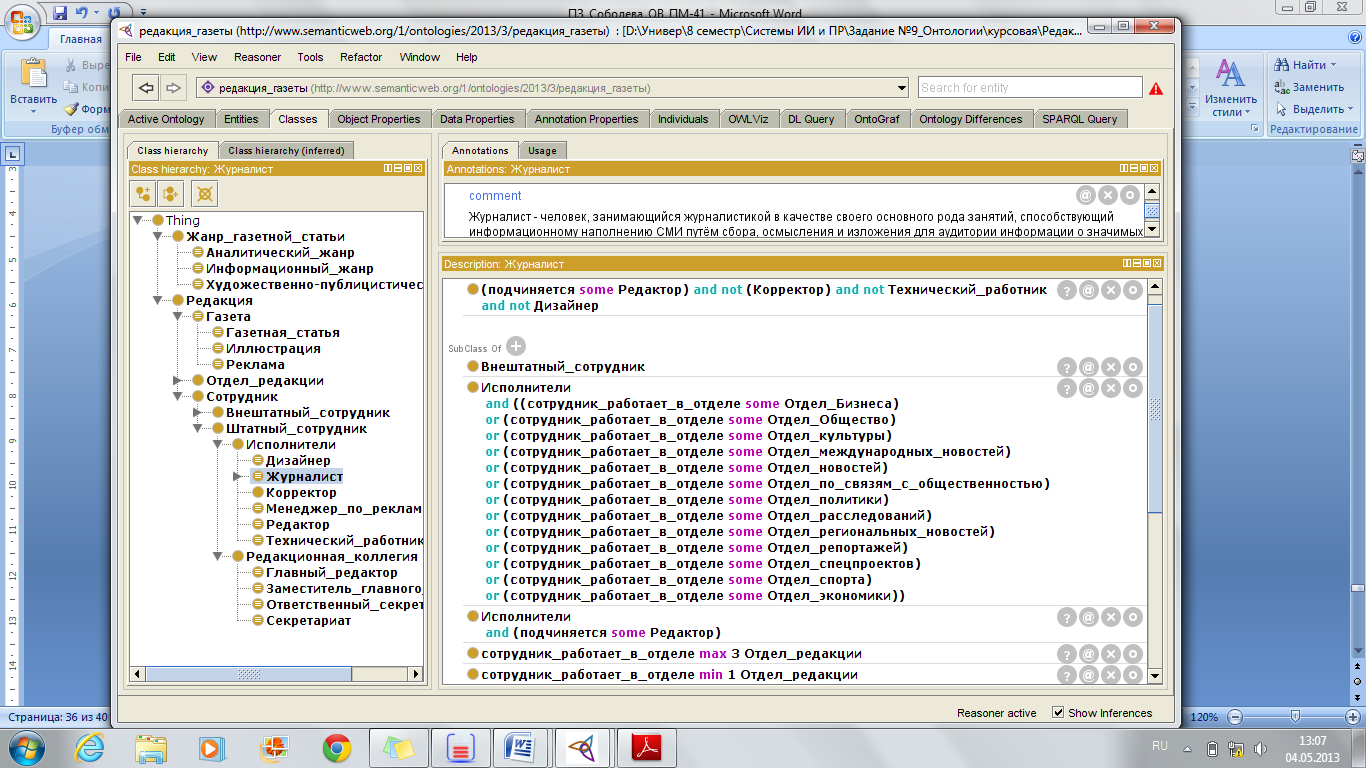


Рисунок 12 – Определение класса «Журналист»

Внештатные сотрудники чаще всего выступают не как самостоятельные авторы, а источники информации, и в любом случае присылаемая от них информация поступает к редакторам, которые и распоряжаются ей. Поэтому было создано два противоположных отношения: «имеет\_источник» и «является\_источником». Где в качестве области определения (Domains) для первого отношения и в качестве области значения (Ranges) для второго выступает класс «Редактор», а Domains для второго и Ranges для первого – класс «Внештатный\_сотрудник». Соответствующие отношения также отражены в описании классов.

Теперь что касается самих газетных статей, у них всегда есть автор, что выражается отношениями «написана» или «создана». Это два эквивалентных в данной онтологии отношения. Это отношение, как и все предыдущие, имеет противоположное отношение «является\_автором». Чтобы описать взаимосвязь редактора с газетной статьей, необходимы два отношения: «отредактирована» и «редактирует». Для описания жанра, в котором написана газетная статья, используется отношение «имеет\_жанр».

Помимо отношений (свойств), связывающих экземпляры друг с другом, есть отношения (свойства), связывающие экземпляры с конкретным типом данных. Эти свойства являются экземплярами класса topDataProperty. Например, в данной онтологии были созданы следующие свойства:

* «Заголовок\_газетной\_статьи» – тип string;
* «Имя\_сотрудник» – тип string;
* «Количество\_сотрудников\_в\_отделе» – тип int.

Таких свойств можно создать достаточно много, но нужно учитывать цель создания онтологии и то, насколько подробно и точно все должно быть описано. Так как в данном случае рассматривается демонстрационная версия онтологии, то этих данных будет вполне достаточно.

Учитывая все созданные отношения, можно определить класс «Газетная\_статья» так, как показано на рисунке 13.

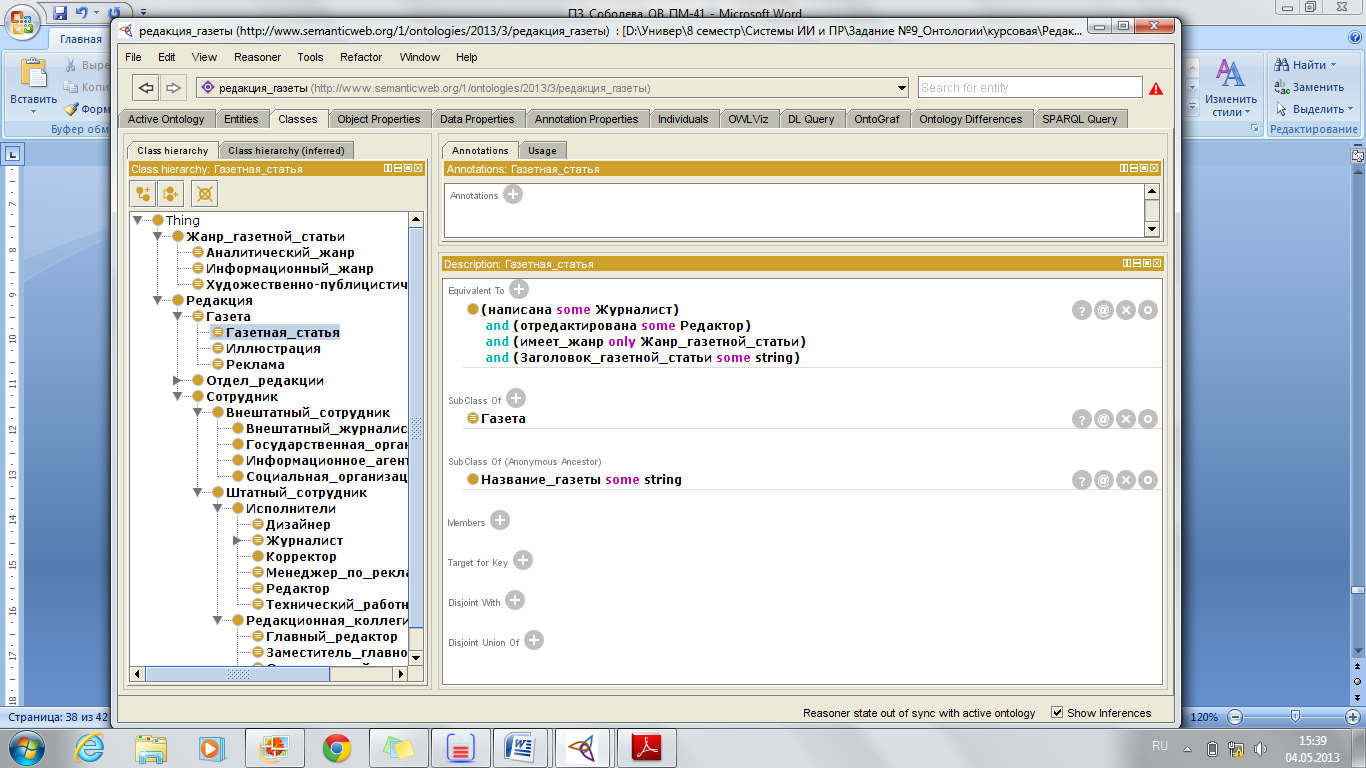


Рисунок 13 – Определение класса «Газетная\_статья»

После совершения любых изменений и внесения дополнений в онтологию необходимо включать устройство логического вывода (Reasoner), который контролирует логичность написанной онтологии. При возникновении конфликтных ситуаций те классы, для которых они возникли, выделяются красным цветом. В этом случае необходимо проверить все отношения и определения, которые связаны с этими классами, и вручную исправить сложившиеся логические ошибки.

В программе Protégé также существует такой инструмент как OntoGraf, который позволяет представить онтологию в виде графа, где вершинами являются классы, а ребра – различные отношения между классами. Можно регулировать глубину этого графа. Вершины, в которых слева от названия указан знак «+», имеют скрытые отношения, то есть отношения, которые не отображены на текущем графе. На рисунке 14 представлен граф, описывающий все отношения двух классов: «Штатный\_сотрудник» и «Внештатный\_сотрудник».

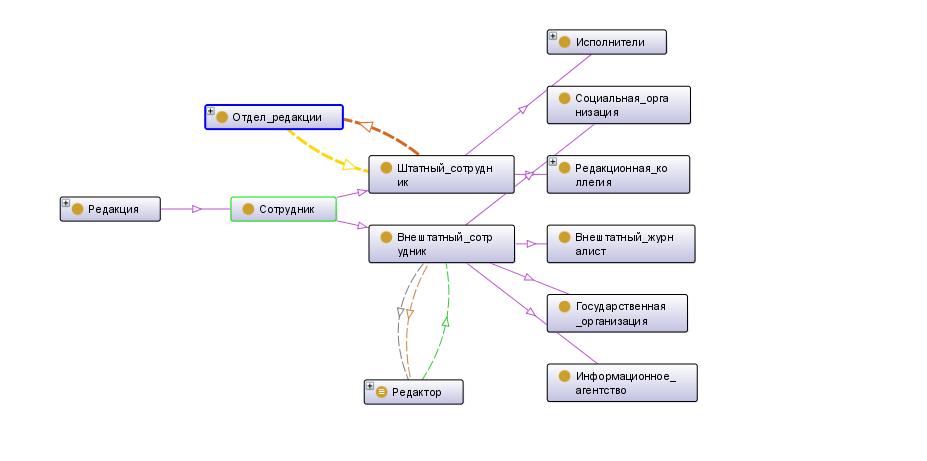


Рисунок 14 – Онтограф отношений классов «Штатный\_сотрудник» и «Внештатный\_сотрудник»

Каждый тип отношений показан на этом графе своим цветом. При наведении указателя мыши на каждую вершину в режиме просмотра непосредственно в окне Protégé появляется всплывающая подсказка, в котором описаны все определения и отношения, связанные с этим классом. При наведении указателя на стрелки также появляется подсказка, в которой указывается, какое это отношение, и какие классы оно связывает.

Все отношения, связанные с классом «Газетная\_статья» изображены на рисунке 15.

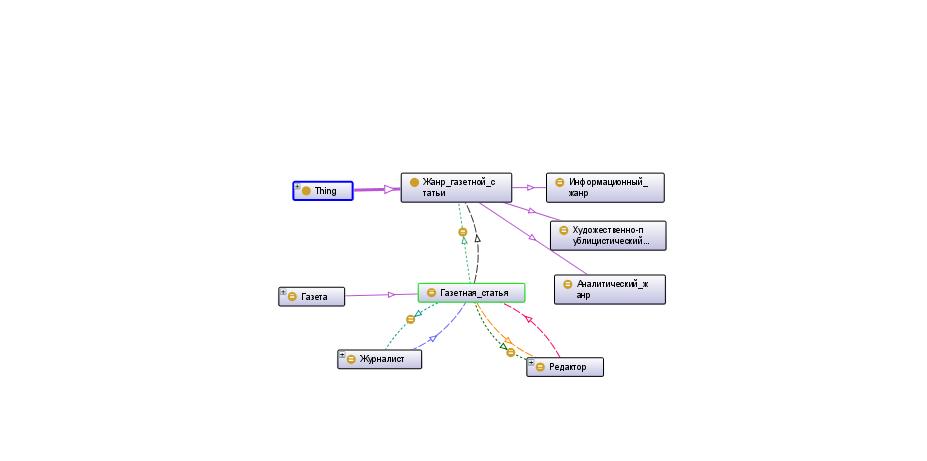


Рисунок 15 – Отношения класса «Газетная\_статья»

Удобство OntoGraf заключается еще и в том, что пользователь может перемещать вершины и делать граф более понятным для просмотра.

Более полная структура онтологии редакции газеты изображена на рисунке 16.

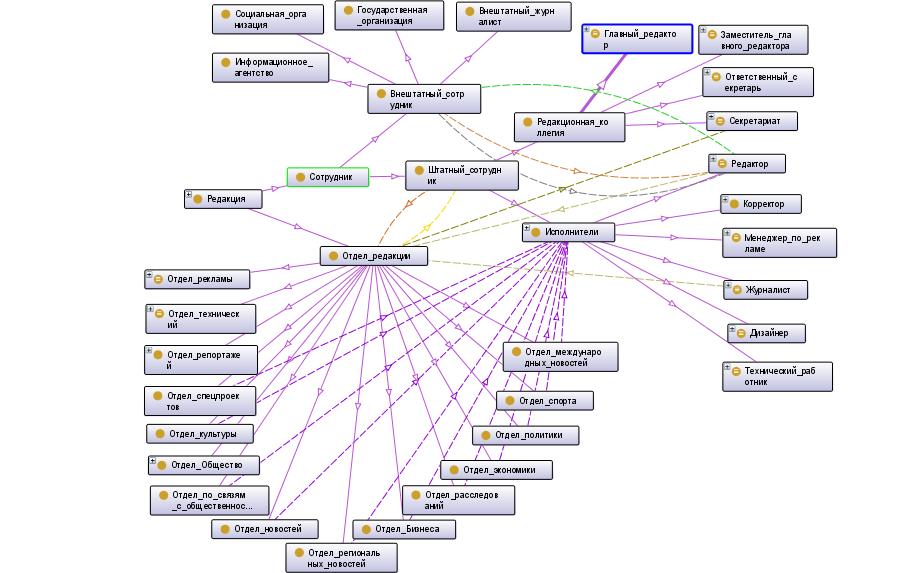


Рисунок 16 – Структура онтологии редакции газеты

Здесь изображены еще не все отношения, которые присутствуют в онтологии, но уже можно заметить насколько сложной может оказаться ее структура. Это объясняется и тем, что количество классов может быть значительно больше, а также тем, что отношений между классами также может быть больше. Таким образом, в крупных онтологиях изображать всю структуру в виде графа бессмысленно, но полезно представлять отдельные части этой структуры, что OntoGraf позволяет делать путем сворачивания некоторых отношений и демонстрацией не всех классов.

Как уже говорилось, редакторы онтологий созданы для того, чтобы было удобнее и понятнее разрабатывать онтологии, а также для того, чтобы формировать описание онтологии, например, на языке OWL. После создания онтологии редакции газеты можно посмотреть ее описание на языке OWL. Фрагмент листинга приведена в приложении А.

# Заключение

Разработана демонстрационная онтология редакции газеты. Следует отметить, что это не единственно возможная онтология редакции газеты. Возможны различные изменения в ее структуре, углубление в изучении каких-то отдельных сторон этой области. Главной целью разработки данной онтологии являлась демонстрация основных принципов проектирования онтологий, что было показано в процессе последовательного ее создания. Как можно заметить, процесс разработки итеративен, на протяжении всего проектирования происходило обращение к уже созданным классам и отношениям, и вносились изменения и уточнения. Вся онтология создавалась на русском языке с использованием общепринятых терминов, существующих в реальной жизни. В будущем возможно указание для всех понятий и отношений синонимов на других языках, например, на английском.

Для реализации проектирования данной онтологии использовалась среда для разработки онтологий – Protégé. Она показала свои достоинства: весьма удобный интерфейс, позволяющий быстро и эффективно редактировать онтологию; встроенное устройство логического вывода (Reasoner), проверяющее правильность создаваемой онтологии; наличие инструмента OntoGraf для представления онтологии в виде ориентированного графа.

Разработанная онтология может применяться в информационных системах, связанных с издательской деятельностью, например, она может быть основой в приложениях для отслеживания работы по созданию статьи, начиная со сбора информации и заканчивая утверждением ее главным редактором или в приложениях, контролирующих работу отделов и взаимодействие их друг с другом.