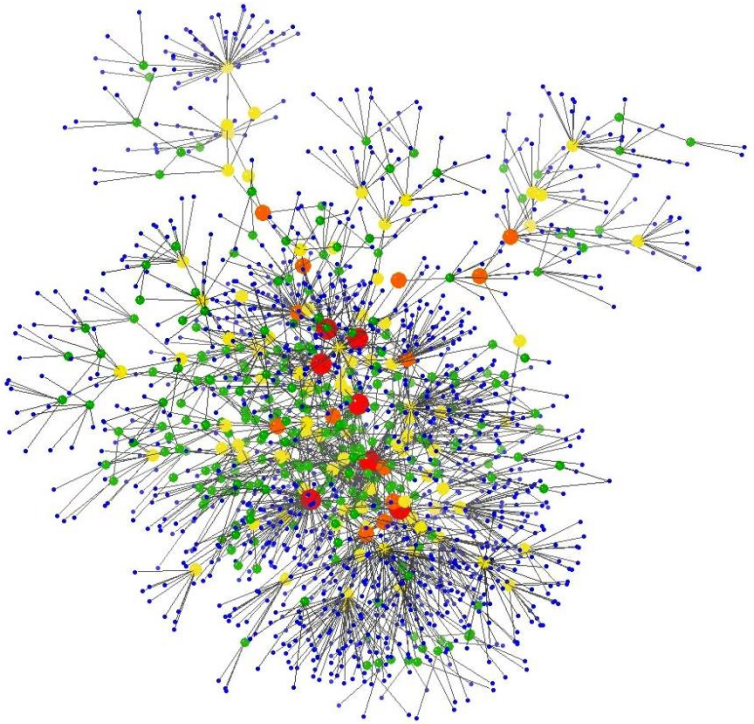
Карпов В.Э.

Онтологии



Проблема понимания

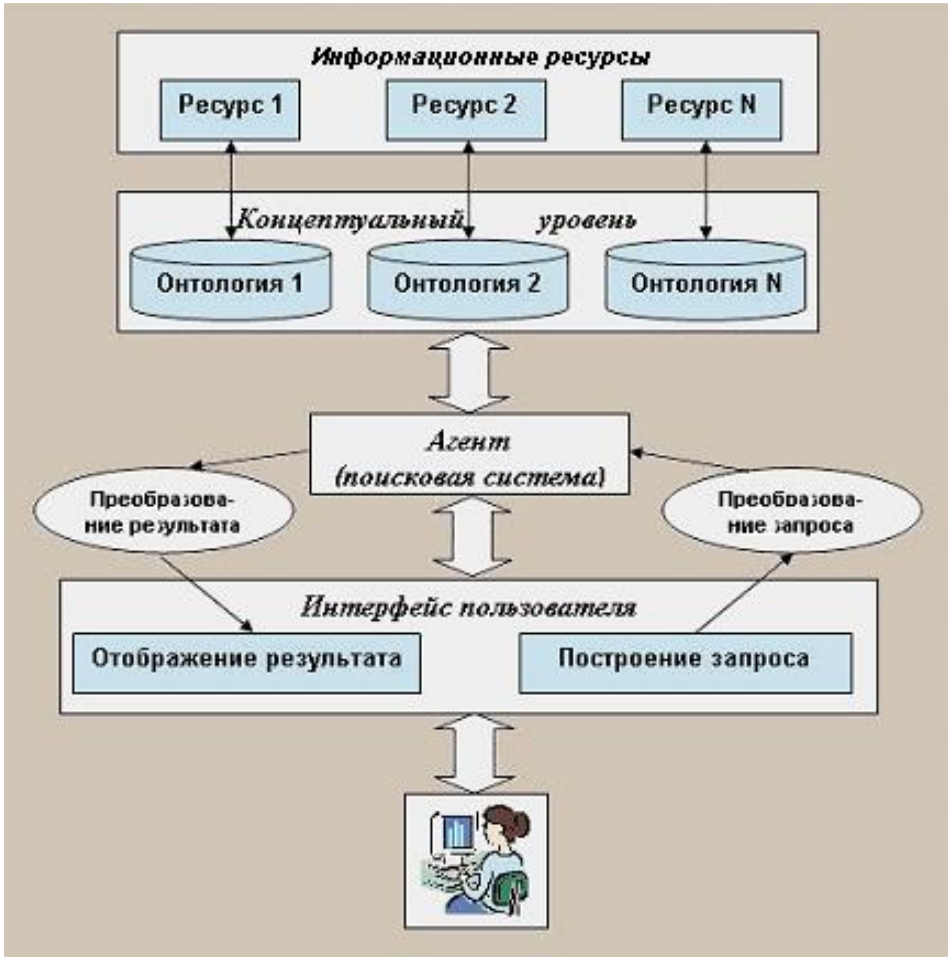
- "Многоязычность" современной науки
- Проблема совместного использования информации (знаний) специалистами из разных областей
- Общий язык - утопия

Онтология

- "Онтология" - обозначение системы знаний, которые относятся к окружающему миру (философск.).
- "Онтология" – учение о бытии
- Термин был введён Аристотелем при попытке классифицировать предметы в мире.

Онтология

- Онтология определяет **общий словарь** для тех, кому нужно совместно использовать информацию в предметной области.
- С точки зрения ИТ О. включает машинно-интерпретируемые формулировки основных понятий предметной области и **отношения** между ними.



Онтология с точки зрения ИТ

Том Грубер (*1993)*. **О. - спецификация концептуализации**. Здесь концептуализация означает абстрактное представление предметной области.

Или: формализованное представление основных понятий и связей между ними

Прочие определения онтологии:

- Общее понимание некоторой области интереса.
- Набор определений (на формальном языке) фрагмента декларативных знаний, ориентированный на совместное многократное использование различными пользователями в своих приложениях.

Под онтологией понимают также:

- надежный смысловой (семантический) базис в определении содержания;
- общую логическую теорию, которая состоит из словаря и набора утверждений на некотором языке логики;
- основу для коммуникации между людьми и компьютерными агентами.

Задачи онтологии

1. Совместное использование людьми или программными агентами общего понимания структуры информации;
2. Возможность повторного использования знаний в предметной области;
3. Сделать допущения в предметной области явными;
4. Отделение знаний в предметной области от оперативных знаний;
5. Анализ знаний в предметной области.

Совместное использование

- *Совместное использование (людьми или программными агентами) общего понимания структуры информации* - основная цель разработки онтологий.
- Например, 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.

Повторное использование знаний

Например, для моделей многих различных ПО 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.

Для создания большой онтологии можно интегрировать несколько существующих онтологий, описывающих части большой предметной области (например, расширение большой базовой онтологии UNSPSC для описания интересующей ПО).

Явные допущения в ПО

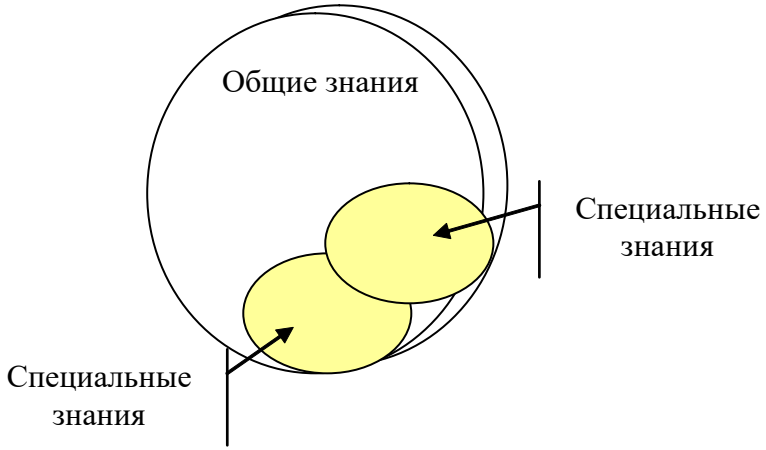
- *Создание явных допущений в предметной области* дает возможность легко изменить эти допущения при изменении наших знаний о предметной области.
- Жесткое кодирование предположений о мире на языке программирования приводит к тому, что эти предположения не только сложно найти и понять, но и также сложно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

Отделение знаний предметной области от оперативных знаний

1. Пусть имеется общая задача конфигурирования продукта из его компонентов в соответствии с требуемой спецификацией. Можно создать систему, которая делает эту конфигурацию независимой от продукта и самих компонентов. Далее:

- разрабатывается онтология компонентов и характеристик ЭВМ. Система конфигурирует нестандартные ЭВМ.
- разрабатывается онтология компонентов и характеристик изделий машиностроения.
- разрабатывается онтология компонентов и характеристик транспортных средств
- …

2. Избыточность БЗ



Анализ знаний в ПО

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

Онтологии и ООП

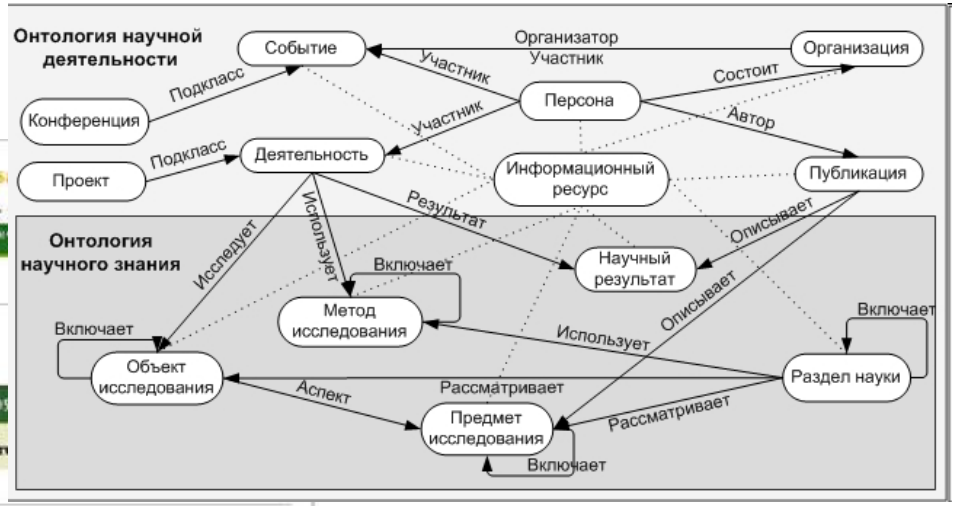
Основные идеи по разработке онтологий берут свое начало в ООП. Однако разработка онтологий отличается от проектирования классов и отношений в ООП:

- ООП сосредотачивается главным образом на методах классов – программист принимает проектные решения, основанные на *операторных* свойствах класса
- При разработке онтологии основываются на *структурных* свойствах класса.
- В результате структура класса и отношения между классами в онтологии отличаются от структуры подобной предметной области в ООП.

Онтологии на практике

Онтологии в сети – это большие таксономии и категоризаторы:

- категоризация запросов (Yahoo! и проч) - множество словарей-тезаурусов
- категоризация продаваемых товаров и их характеристик (Amazon.com).



RDF

RDF (Resource Description Framework) - язык кодирования знаний на веб-страницах, для того, чтобы сделать их понятными для электронных агентов, которые осуществляют **поиск информации**.

Запросы RDF (RDF Query Engines) и язык запросов SPARQL

Примеры шаблонов троек (объект, субъект и отношение):

- ?w lit:wrote lit:KingLear. -- Кто написал Короля Лир?

- lit:Shakespeare ?r lit:KingLear. -- Какое **отношение** связывает Шекспира и Короля Лир?
- lit:Shakespeare lit:wrote ?p. -- Что написал Шекспир?

{?person bio:livedIn ?place.
?place geo:isIn geo:England.
?person lit:wrote lit:KingLear.}

«Найти человека, который жил в месте, которое находится в Англии и который также написал Короля Лир»

Язык Разметки для Агентов

- Язык Разметки для Агентов (DARPA Agent Markup Language, DAML, DARPA в сотрудничестве с W3C).
- Расширение RDF более выразительными конструкциями, предназначенными для облегчения взаимодействия **агентов в сети**.

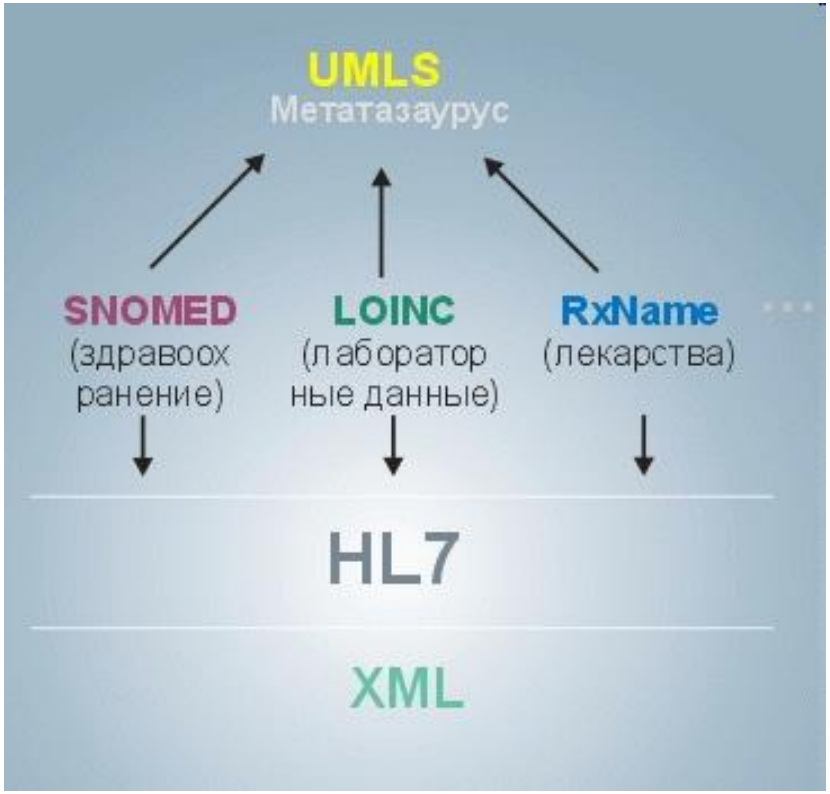
Стандартные онтологии

Стандартные онтологии могут использоваться экспертами по предметным областям для совместного использования и аннотирования информации в своей области.

Медицина. Большие стандартные, структурированные словари:

- SNOMED
- UMLS - Система Унифицированного Медицинского Языка (the Unified Medical Language System) - средство для разработки компьютерных систем «понимающих» биомедицинскую информацию и информацию в сфере здравоохранения.

UMLS имеет три базы знаний: Метатезаурус, Семантическая Сеть, SPECIALIST-лексикон.



Общецелевые онтологии:

Онтология UNSPSC (терминология товаров и услуг, Программа ООН по развитию и компания Dun & Bradstreet)

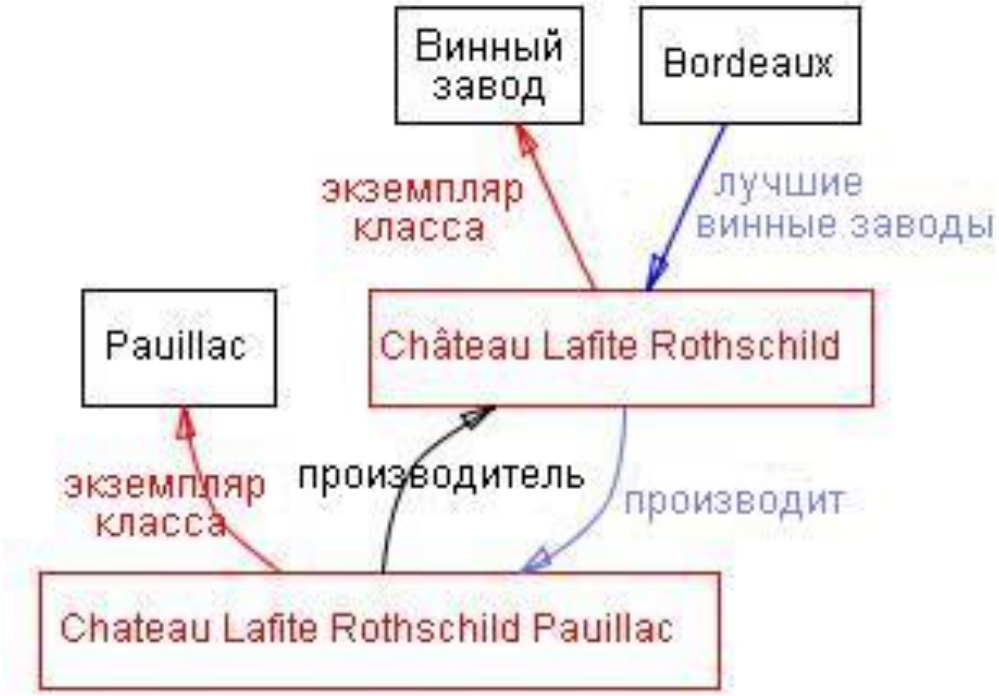
Пример создания онтологии

Терминология

- **Онтология** – формальное явное описание **понятий** в рассматриваемой ПО.
- Понятия представлены **классами**.
- Класс может иметь **подклассы**, которые представляют более конкретные понятия, чем суперкласс.
- Свойства и атрибуты понятия описываются **слотами** (**ролями, свойствами**).
- Ограничения, накладываемые на слоты описываются **фацетами** (**ограничениями ролей**).
- Онтология вместе с набором индивидуальных **экземпляров** классов образует **базу знаний**.

Этапы создания онтологии

- определение классов в онтологии;
- расположение классов в таксономической иерархии (класс – суперкласс);
- определение слотов и описание ограничений значений этих слотов;
- заполнение значений слотов экземпляров.



Шаг 1. Определение области и масштаба онтологии

Необходимо ответить на следующие вопросы:

- Какую область будет охватывать онтология?
- Для чего мы собираемся использовать онтологию?
- На какие типы вопросов должна давать ответы информация в онтологии?
- Кто будет использовать и поддерживать онтологию?

Шаг 2. Можно ли взять готовое решение?

- Рассмотрение вариантов повторного использования существующих онтологий

Шаг 3. Определение основных терминов

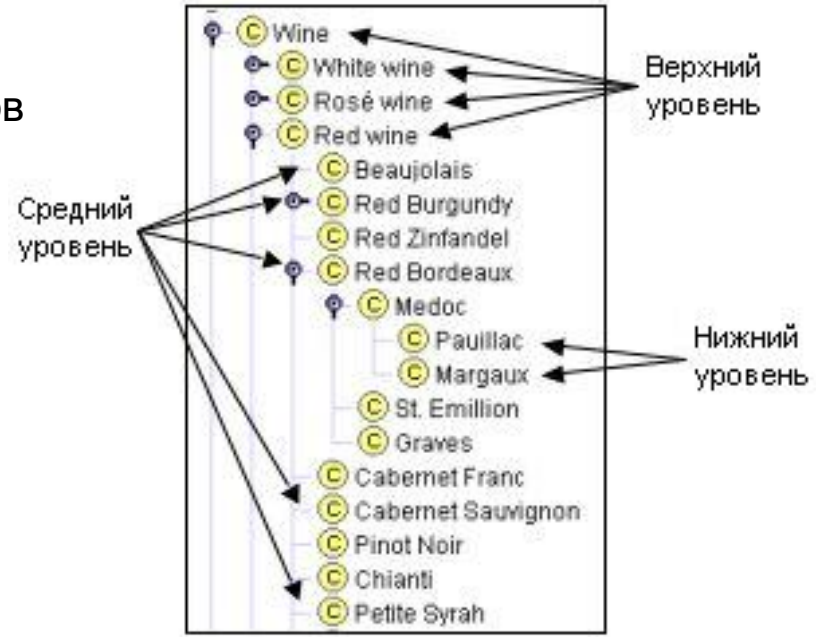
Перечисление важных терминов в онтологии

- *"… вино, виноград, винный завод, местоположение, цвет вина, его крепость, вкус и содержание сахара; различные виды еды (рыба, мясо); типы вина (белое, красное) и т.д."*
- Важно получить полный список:
  - терминов (не беспокоясь о пересечении понятий, которые они представляют),
  - отношений между терминами
  - возможных свойств понятий или о том, чем являются понятия – классами или слотами.

Шаг 4. Определение классов и иерархии классов

Существует несколько возможных подходов для разработки иерархии классов:

1. Процесс **нисходящей** разработки (начинается с определения самых общих понятий).
   Например, можноначать с создания классов для общих понятий Вино и Еда. Затем мы конкретизируем класс Вино, создавая его подклассы: Белое вино, Красное вино, Розовое вино. и т.д.
2. Процесс **восходящей** разработки (начинается с определения самых конкретных классов, листьев иерархии, с последующей группировкой этих классов в более общие понятия).
   Например, определить классы для вин Pauillac и Margaux. Затем создать общий надкласс для двух этих классов – Medoc, который является подклассом Bordeaux и т.д.
3. Процесс **комбинированной** разработки (сочетание нисходящего и восходящего подходов).



Шаг 5. Определение свойств классов – слотов

- Список терминов включает, к примеру, *цвет вина, его крепость, вкус и содержание сахара, а также местоположение винного завода*.
- Для каждого свойства из списка мы должны определить, какой класс оно описывает. Эти свойства станут **слотами**, привязанными к классам. Т.о., у класса **Вино** будут следующие слоты: цвет, крепость, вкус и сахар. А у класса **Винный завод** будет слот местоположение.
- В онтологии слотами могут стать несколько типов свойств объектов:
  - «внутренние» свойства (вкус вина);
  - «внешние» свойства (название вина и область, в которой оно было произведено);
  - части, если объект имеет структуру; они могут быть как физическими, так и абстрактными «частями» (например, блюда, входящие в обед);
  - отношения с другими концептами (например, **производитель вина**, представляющий отношение между вином и винным заводом, и **виноград**, из которого произведено вино).

Т.о., к классу **Вино** нужно добавить слоты: название, область, производитель, виноград.

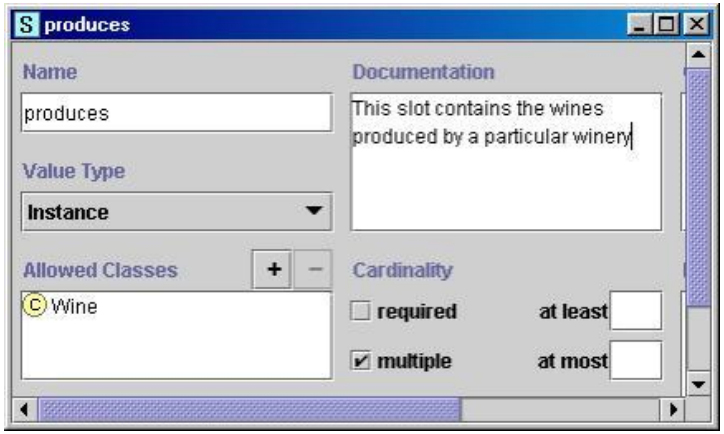
Template Slots

Name	Type	Cardinality	Other Facets
body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}
color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}
flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}
grape	Instance	multiple	classes={Wine grape}
maker	Instance	single	classes={Winery}
name	String	single	
sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}

Шаг 6. Определение фацетов слотов

Слоты могут иметь различные **фацеты**, которые описывают тип значения, разрешенные значения, число значений (мощность) и другие свойства значений, которые может принимать слот.

- **Мощность слота** определяет, сколько значений может иметь слот. Крепость вина будет слотом единичной мощности. Вина, производимые на конкретном заводе, заполняют слот **множественной** мощности производит класса Винный завод.
- **Тип значения слота**
  - **Строка**
  - **Число**
  - **Булевы** слоты.
  - **Нумерованные** слоты (определяют список конкретных разрешенных значений слота).
  - Слоты-**экземпляры** позволяют определить отношения между индивидными концептами. Например, слот "*производит"* класса Винный завод в качестве значений может иметь экземпляры класса **Вино**.

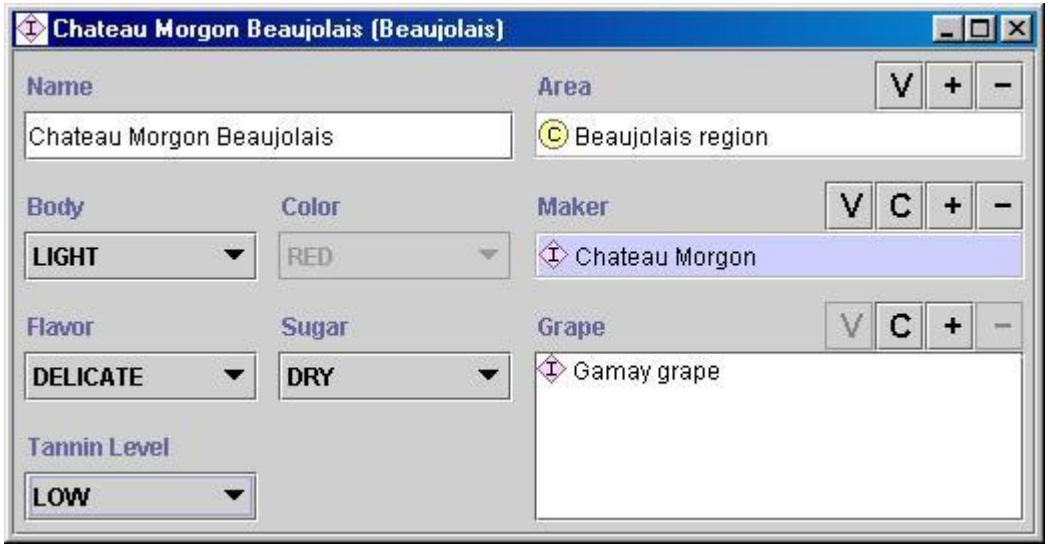


Шаг 7. Создание экземпляров

Для определения отдельного экземпляра класса требуется:

- выбрать класс
- создать отдельный экземпляр этого класса
- ввести значения слотов.

Например, экземпляр Chateau-Morgon-Beaujolais для представления определенного типа вина Beaujolais.



Значения слотов экземпляра:

- Крепость: Легкое
- Цвет: Красный
- Вкус: Мягкий
- Уровень танина: Низкий
- Виноград: Gamay (экземпляр класса Виноград для изготовления вин)
- Производитель: Chateau-Morgon (экземпляр класса Винный завод)
- Область: Beaujolais (экземпляр класса Винная область)
- Сахар: Сухое

Особенности проектирования

1. Транзитивность иерархических отношений

Отношение подкласса транзитивно:

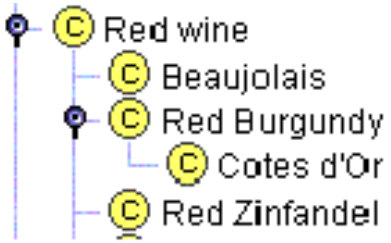
Если B – это подкласс A, а C – подкласс B, то C – подкласс A.

2. Недопустимость циклов классов

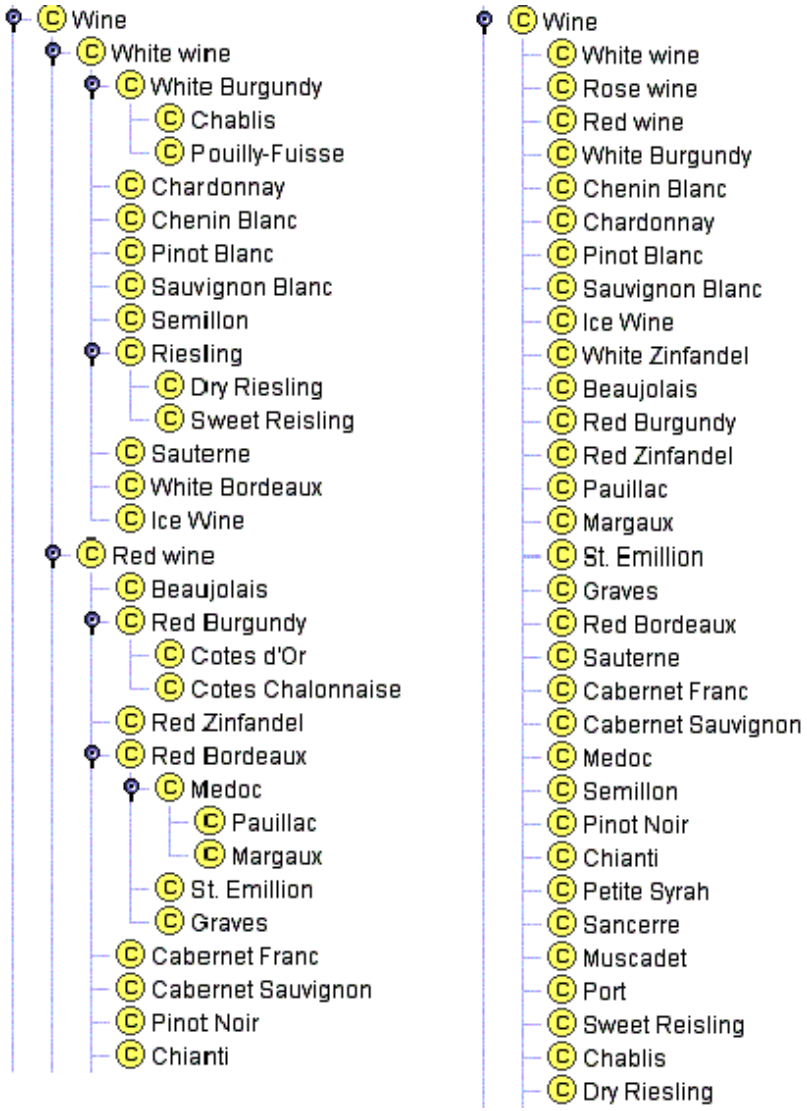
В иерархии есть цикл, когда у некоторого класса A есть подкласс B и в то же время B – это суперкласс A.

3. Сбалансированность иерархии

- Во многих онтологиях с четкой структурой имеется от двух до дюжины прямых подклассов. Отсюда:
- *Если класс имеет только один прямой подкласс, то, возможно, при моделировании допущена ошибка или онтология неполная.*
- *Если у данного класса есть более дюжины подклассов, то, возможно, необходимы дополнительные промежуточные категории.*



Простое перечисление и уровни



Когда вводится подкласс

Самый сложный вопрос. Считается, что подклассы*:*

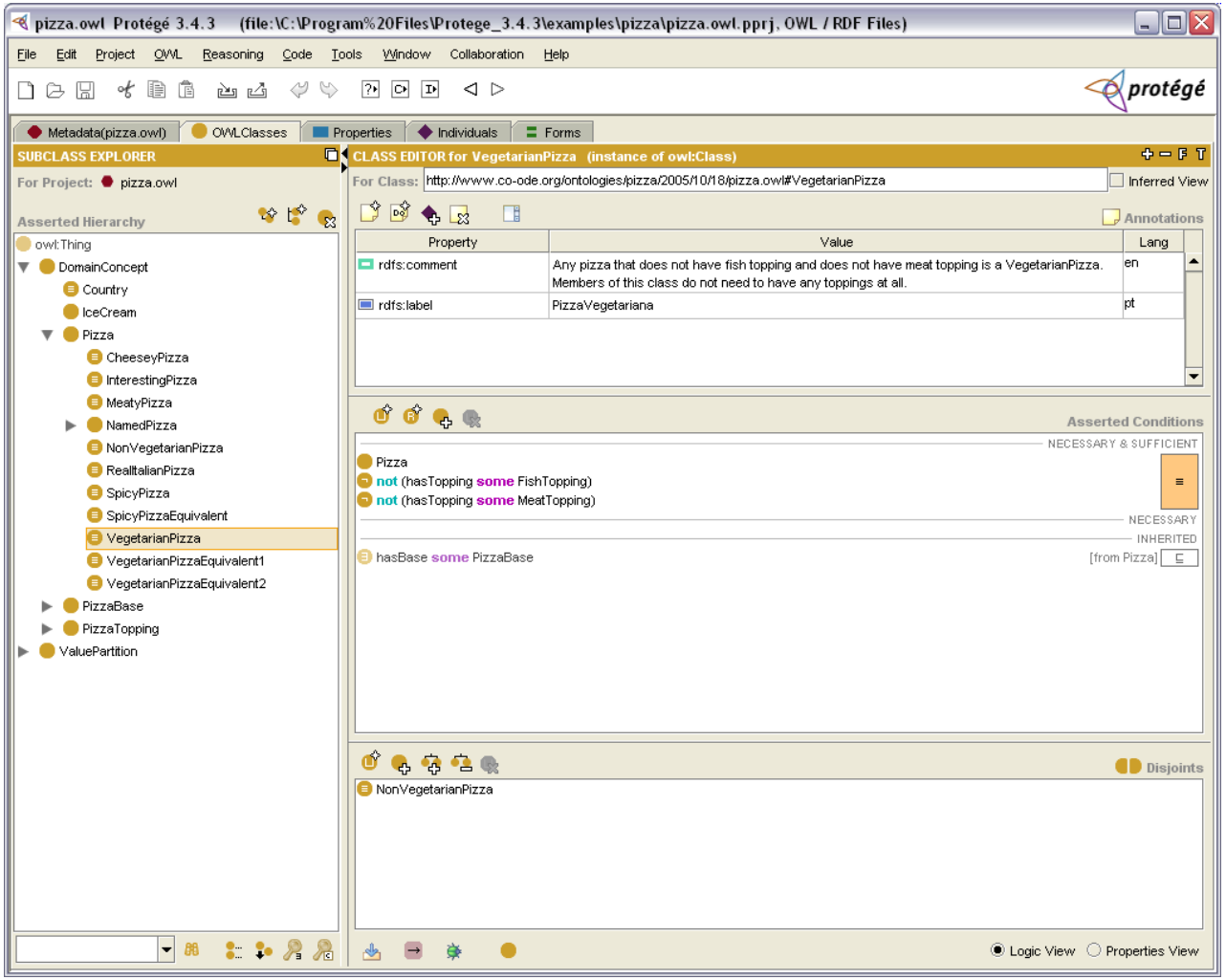
1. *Имеют дополнительные свойства, которых нет у надкласса*
2. *Имеют ограничения, отличные от тех, которые есть у суперкласса*
3. *Состоят в других отношениях, нежели надклассы.*

Инструменты построения онтологий

- **Protege** – локальная Java-программа, разработанная группой медицинской информатики Стенфордского университета.
- Включает **редактор онтологий**. Структура онтологии сделана аналогично иерархической структуре каталога. На основе сформированной онтологии Protege может генерировать формы получения знаний для введения экземпляров классов и подклассов.
- Protege основан на **фреймовой модели представления знания** и снабжен рядом плагинов, что позволяет адаптировать его для редактирования моделей в разных форматах (стандартный текстовый, базы данных, UML, языков XML, XOL, SHOE, RDF и RDFS, DAML+OIL, OWL).

	OilEd	OntoEdit	Ontolingua	OntoSaurus	Protégé	WebODE	WebOnto
Общая информация							
Разработчик	IMG, University of Manchester	Ontoprizse	KSL, Stanford University	ISI, University of Southern California	SMI, Stanford University	Ontology Group, Polytechnic University of Madrid	KMI, Open University
Версия	3.5.5 Oct2003	2.6.6 Mar2004	1.0.650 Oct2002	1.9 Mar2002	2.1.1 June2004	2.1 Mar2003	2.3 May2001
Доступность	Открытый код	Свободная лицензия	Свободный доступ	Открытый код, свободный доступ	Открытый код	Свободный доступ	Свободный доступ
Поддержка методологией	-	On-To-Knowledge	-	-	-	METH-ONTOLOGY	-
Архитектура программного обеспечения							
Архитектура приложения	3–х уровневая	3–х уровневая	Клиент/ сервер	Клиент/ сервер	3–х уровневая	n-уровневая	Клиент/ сервер
Расширяемость		Плагины			Плагины	Сервер приложения	
Хранение онтологий	файлы	файлы	файлы	файлы	файлы, СУБД	СУБД	файлы
Язык ПО	Java	Java	Lisp	Lisp	Java	Java	Java+ Lisp
Модель знания							
Формализм	DL	Фреймы + FOL	Фреймы + FOL	DL	Фреймы + FOL	Фреймы + FOL	Фреймы + FOL
Основной язык представления знания	DAML+OIL	OXML	Ontolingua	LOOM	OKBC	-	OCML
Формальный язык аксиом	-	FLogic	KIF	LOOM	PAL	WAB	OCML
Редакторы онтологий							
Интерфейс пользователя	Локальное приложение	Локальное приложение	HTML	HTML	Локальное приложение	HTML и апплеты	Апплеты
Графическое редактирование таксономии концептов	-	+	-	-	+	+	+
Редактор формальных аксиом	+	-	-	-	+	+	-
Совместная разработка онтологи	-	+	+	+	-	+	+
Машина вывода	FaCT (встроенная) RACER DIG	OntoBroker	JTP	классификатор LOOM	PAL (встр.) Jess, FaCT, Prolog, FLORA Algermon	Prolog (встроенная) Jess	Система представления знаний OCML
Проверка непротиворечивости	+	+	-	+	+	+	+
Интероперабельность							
С другими инструментами	-	OntoAnnotate OntoMat Semantic-Miner	Chimaera OKBC	OKBC	Prompt OKBC ArgoUML	ODE-KM ODE-SeW ODE-SWS ODE-Annotate Protégé	MnM
Импорт	RDF(S) OIL DAML+OIL SHIQ	OXML RDF(S) DAML+OIL FLogic	Ontolingua KIF CML IDL	LOOM PowerLOOM Stella IDL	XML RDF(S) XML Schema XMI	XML RDF(S) DAML+OIL OWL CARIN	OXML RDF(S)
Экспорт	RDF(S) OIL DAML+OIL OWL SHIQ DIG	OXML RDF(S) DAML+OIL FLogic	Ontolingua KIF LOOM CLIPS CML Epikit Prolog IDL	LOOM PowerLOOM KIF Ontolingua Stella IDL C++	XML RDF(S) XML Schema FLogic CLIPS Java XMI	XML RDF(S) OIL DAML+ OIL OWL CARIN FLogic Prolog, Jess, Java	OXML Ontolingua RDF(S)

Protege



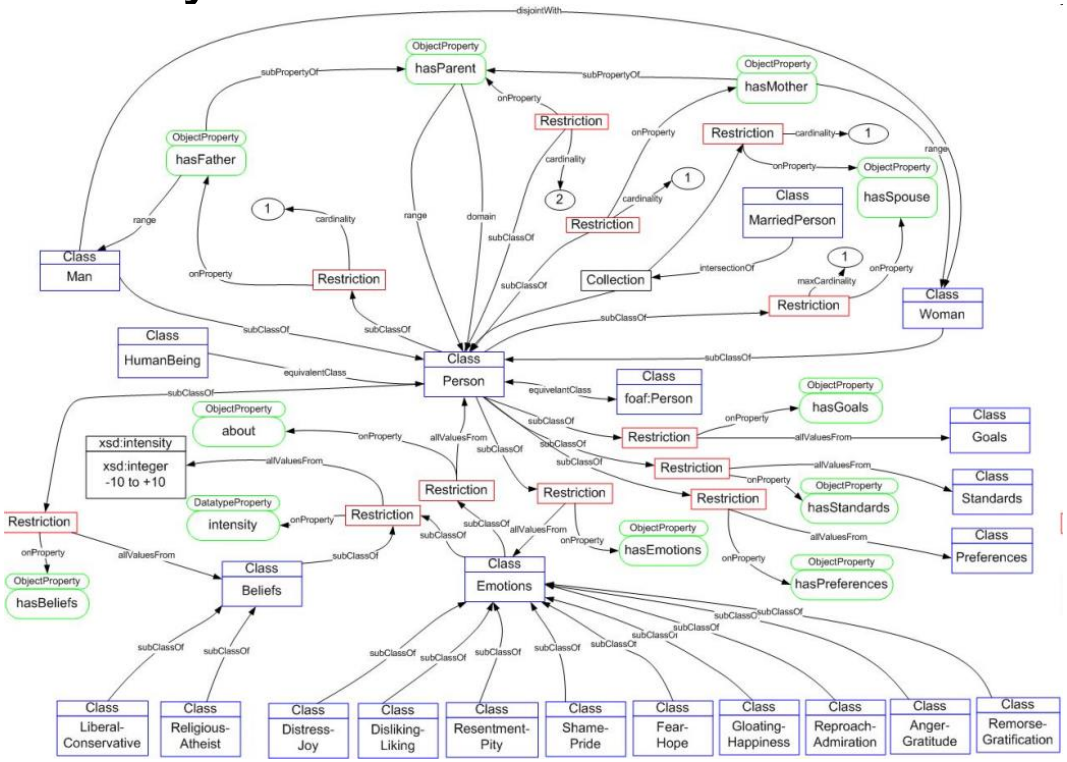
Внешний вид

Онтологии и тезаурусы

- **Тезаурус** – полный систематизированный набор данных о какой-либо области знаний, позволяющий человеку или вычислительной машине в ней ориентироваться.
- Тезаурус более закреплен за лексикой, а онтология за семантикой.
- Универсальные тезаурусы строятся на основе иерархической структуры, которая является естественной для текстовых тезаурусов, но никак не адекватна в реальных конкретных предметных областях.

Заключение

1. Для любой ПО не существует единственно правильной онтологии.
2. Проектирование онтологии – это творческий процесс и две онтологии, разработанные разными людьми, никогда не будут одинаковыми.
3. Онтология субъективна.



Источники

- Natalya F. Noy and Deborah L. McGuinness. ``Ontology Development 101: A Guide to Creating Your First Ontology''. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001. http://protege.stanford.edu/publications/ontology_development/ontology101.html

  Наталья Ф. Ной (Natalya F. Noy) и Дэбора Л. МакГиннесс (Deborah L. McGuinness) Разработка онтологий 101: руководство по созданию Вашей первой онтологии // Стэнфордский Университет, Стэнфорд, Калифорния, 94305
- «Использование онтологий в системах управления знаниями» Т.А. Гаврилова (http://big.spb.ru/publications/bigspb/km/use_ontology_in_suz.shtml)
- Обзор инструментов инженерии онтологий. http://www.impb.ru/~rcdl2004/cgi/get_paper_pdf.cgi?pid=26
- Место онтологий в единой интегрированной системе РАН. http://www.benran.ru/Magazin/cgi-bin/Sb_03/pr03.exe?!15
- От тезауруса к онтологии и обратно. http://www.dialog-21.ru/archive_article.asp?param=7360&y=2002&vol=6077
- Кентавр по имени ТЕОН: Тезаурус + Онтология. http://www.dialog-21.ru/archive_article.asp?param=6771&y=2001&vol=6077
- *Gruber T*. A translation Approach to Portable Ontology Specifications // Knowledge Acquisition Journal, vol. 5, pages 199-220, 1993.