

Разработка онтологий 101: руководство по созданию Вашей первой онтологии¹

Наталья Ф. Ной (Natalya F. Noy) и Дэбора Л. МакГиннесс (Deborah L. McGuinness)
Стэнфордский Университет, Стэнфорд, Калифорния, 94305
noy@smi.stanford.edu и dln@ksl.stanford.edu

[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

Аннотация

Онтологии стали центральными компонентами многих больших приложений, хотя учебный материал не соответствует растущему интересу. В этой работе обсуждается вопрос, зачем строить онтологию, и предлагается методология создания онтологий, основанная на системах представления декларативных знаний. Она использует опыт двух авторов в построении и поддержке онтологий в ряде онтологических сред, включая Protege-2000, Ontolingua и Chimaera. В ней представлена методология на примере учебной базы знаний по винам. Несмотря на то, что статья адресована пользователям фреймовых систем, она может быть полезна для построения онтологий в любой объектно-ориентированной системе.

1. Зачем создавать онтологию?

В последние годы разработка онтологий - формальных явных описаний терминов предметной области и отношений между ними (Gruber 1993) - переходит из мира лабораторий по искусственному интеллекту на рабочие столы экспертов по предметным областям. Во всемирной паутине онтологии стали обычным явлением. Онтологии в сети варьируются от больших таксономий, категоризирующих веб-сайты (как на сайте Yahoo!), до категоризаций продаваемых товаров и их характеристик (как на сайте Amazon.com). Консорциум WWW (W3C) разрабатывает RDF (Resource Description Framework) (Brickley and Guha 1999), язык кодирования знаний на веб-страницах, для того, чтобы сделать их понятными для электронных агентов, которые осуществляют поиск информации. Управление перспективных исследований и разработок министерства обороны США (The Defense Advanced Research Projects Agency, DARPA) в сотрудничестве с W3C разрабатывает Язык Разметки для Агентов DARPA (DARPA Agent Markup Language, DAML), расширяя RDF более выразительными конструкциями, предназначенными для облегчения взаимодействия агентов в сети (Hendler and McGuinness 2000). Во многих дисциплинах сейчас разрабатываются стандартные онтологии, которые могут использоваться экспертами по предметным областям для совместного использования и аннотирования информации в своей области. Например, в области медицины созданы большие стандартные, структурированные словари, такие как SNOMED (Price and Spackman 2000) и семантическая сеть Системы Унифицированного Медицинского Языка (the Unified Medical Language System) (Humphreys and Lindberg 1993). Также появляются обширные общецелевые онтологии. Например, Программа ООН по развитию (the United Nations Development Program) и компания Dun & Bradstreet объединили усилия для разработки онтологии UNSPSC, которая предоставляет терминологию товаров и услуг (<http://www.unspsc.org/>).

Онтология определяет общий словарь для ученых, которым нужно совместно использовать информацию в предметной области. Она включает машинно-интерпретируемые формулировки основных понятий предметной области и отношения между ними.

Почему возникает потребность в разработке онтологии? Вот некоторые причины:

¹ В большинстве американских колледжей вступительный курс любого предмета имеет номер «101»: «Химия 101», «Биология 101» и т.д. Следующие два более углубленных курса по химии назывались бы «Химия 102» и «Химия 103» соответственно. В США номер «101» означает «Введение». Т.е., название статьи нужно понимать как «Введение в разработку онтологий: Руководство по созданию Вашей первой онтологии». (Здесь и далее в тексте примечания переводчика. Авторские примечания вынесены в конец статьи самими авторами.)

- Для совместного использования людьми или программными агентами общего понимания структуры информации.
- Для возможности повторного использования знаний в предметной области.
- Для того чтобы сделать допущения в предметной области явными.
- Для отделения знаний в предметной области от оперативных знаний.
- Для анализа знаний в предметной области.

Совместное использование людьми или программными агентами общего понимания структуры информации является одной из наиболее общих целей разработки онтологий (Musen 1992; Gruber 1993). 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 нам нужно создать большую онтологию, мы можем интегрировать несколько существующих онтологий, описывающих части большой предметной области. Мы также можем повторно использовать основную онтологию, такую как UNSPSC, и расширить ее для описания интересующей нас 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. Мы можем 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 (McGuinness and Wright 1998). После этого мы можем разработать онтологию компонентов и характеристик ЭВМ и применить этот алгоритм для конфигурирования нестандартных ЭВМ. Мы также можем использовать тот же алгоритм для конфигурирования лифтов, если мы предоставим ему онтологию компонентов лифта (Rothenfluh et al. 1996).

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 (McGuinness et al. 2000).

Часто онтология предметной области сама по себе не является целью. Разработка онтологии сродни определению набора данных и их структуры для использования другими программами. Методы решения задач, доменно-независимые приложения и программные агенты используют в качестве данных онтологии и базы знаний, построенные на основе этих онтологий. К примеру, в этой статье мы разрабатываем онтологию вин и еды, а также подходящие комбинации вин и блюд. Затем 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.

Об этом руководстве

Мы основываемся на нашем опыте использования Protege-2000 (Protege 2000), Ontolingua (Ontolingua 1997), Chimaera (Chimaera 2000) в качестве сред для редактирования онтологий. В этом руководстве для наших примеров мы используем Protege-2000.

Пример вина и еды, который мы используем на протяжении всей статьи, сделан на основе примерной базы знаний, которая представлена в статье, описывающей CLASSIC – систему представления знаний, основанную на описательно-логическом подходе (Brachman et al. 1991). В учебном пособии по CLASSIC (McGuinness et al. 1994) этот пример получил дальнейшее развитие. Protege-2000 и другие фреймворковые системы описывают онтологии декларативным образом, определяя явным образом, какова классовая иерархия и к каким классам принадлежат индивидуальные концепты.

Некоторые идеи по разработке онтологий в этом руководстве берут свое начало в литературе по объектно-ориентированному проектированию (Rumbaugh et al. 1991; Booch et al. 1997). Однако разработка онтологий отличается от проектирования классов и отношений в объектно-ориентированном программировании. Объектно-ориентированное программирование сосредотачивается главным образом на методах классов – программист принимает проектные решения, основанные на *операторных* свойствах класса, тогда как разработчик онтологии принимает эти решения, основываясь на *структурных* свойствах класса. В результате структура класса и отношения между классами в онтологии отличаются от структуры подобной предметной области в объектно-ориентированной программе.

Невозможно охватить все трудности, которые, возможно, придется преодолеть разработчику онтологии, и в этом руководстве мы не пытаемся затронуть их всех. Вместо этого мы пытаемся дать отправную точку, исходное руководство, которое могло бы помочь неопытному проектировщику онтологий в их разработке. В конце мы предлагаем источники, в которых можно посмотреть пояснения к более сложным структурам и механизмам разработки, если они потребуются для предметной области.

В конечном счете, единственной правильной методологии разработки онтологий не существует, и мы не пытались определить таковую. Представленные здесь идеи мы сочли полезными, исходя из нашего опыта разработки онтологий. В конце этого руководства мы предлагаем список ссылок на альтернативные методологии.

2. Из чего состоит онтология?

В литературе по искусственному интеллекту содержится много определений понятия онтологии, многие из которых противоречат друг другу. В этой статье **онтология** – формальное явное описание понятий в рассматриваемой предметной области (**классов** (иногда их называют **понятиями**)), свойств каждого понятия, описывающих различные свойства и атрибуты понятия (**слов** (иногда их называют **ролями** или **свойствами**)), и ограничений, наложенных на слоты (**фацетов** (иногда их называют **ограничениями ролей**)). Онтология вместе с набором индивидуальных **экземпляров** классов образует **базу знаний**. В действительности, трудно определить, где кончается онтология и где начинается база знаний.

В центре большинства онтологий находятся классы. Классы описывают понятия предметной области. Например, класс **вин** представляет все вина. Конкретные вина – экземпляры этого класса. Вино Bordeaux в бокале перед вами, когда вы читаете этот документ, – это экземпляр класса **вин Bordeaux**. Класс может иметь **подклассы**, которые представляют более конкретные понятия, чем надкласс. Например, мы можем разделить класс **всех вин** на **красные**, **белые** и **розовые вина**. В качестве альтернативы мы можем разделить класс **всех вин** на **игристые** и **не игристые вина**.

Слоты описывают свойства классов и экземпляров: вино Chateau Lafite Rothschild Pauillac – крепкое, оно производится на винном заводе Chateau Lafite Rothschild. У нас есть два слота, которые описывают вино в этом примере: слот **крепость** со значением «крепкое» и слот **производитель** со значением «винный завод Chateau Lafite Rothschild». Мы можем сказать, что на уровне класса у экземпляров класса Вино есть слоты, которые описывают вкус, крепость, уровень сахара, производителя вина и т.д.^[1]

Все экземпляры класса Вино и его подкласс Pauillac имеют слот **производитель**, значение которого является экземпляром класса **Винный завод** (Рис. 1). Все экземпляры класса

Винный завод имеет слот производит, относящийся ко всем винам (экземплярам класса Вино и его подклассов), которые производятся на этом заводе.

На практике разработка онтологии включает:

- определение классов в онтологии;
- расположение классов в таксономическую иерархию (подкласс – надкласс);
- определение слотов и описание допускаемых значений этих слотов;
- заполнение значений слотов экземпляров.

После этого мы можем создать базу знаний, определив отдельные экземпляры этих классов, введя в определенный слот значение и дополнительные ограничения для слота.

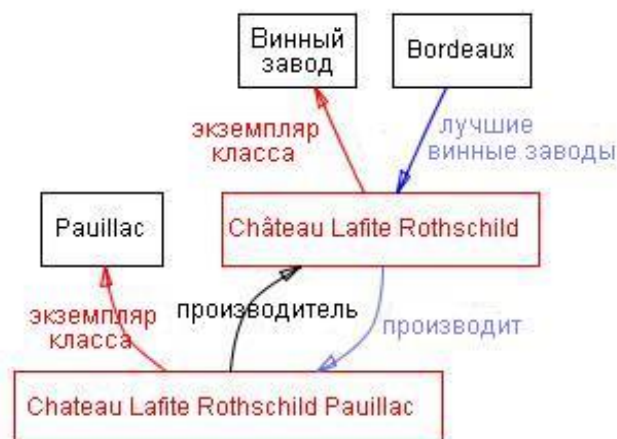


Рис. 1. Некоторые классы в области вин, экземпляры и отношения между ними. Черным мы обозначили классы, а красным – экземпляры. Прямые связи обозначают слоты и внутренние связи, такие как «экземпляр [класса]» и «подкласс [класса]».

3. Простая методология инженерии знаний

Как мы сказали выше, не существует единственного «правильного» способа или методологии разработки онтологий. Здесь мы обсуждаем общие моменты, которые нужно учитывать, и предлагаем один из возможных способов разработки онтологии. Мы описываем итеративный подход к разработке онтологии: мы начинаем с первого чернового просмотра онтологии. Затем мы проверяем и уточняем получаемую онтологию и добавляем детали. Попутно мы обсуждаем решения, касающиеся моделирования, которые должен принять разработчик, а также «за» и «против» и результаты принятия различных решений.

Во-первых, мы бы хотели выделить некоторые фундаментальные правила разработки онтологии, к которым мы будем неоднократно обращаться. Эти правила могут показаться довольно категоричными. Тем не менее, во многих случаях они могут помочь принять проектные решения.

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.

3) Понятия в онтологии должны быть близки к объектам (физическим или логическим) и отношениям в интересующей вас предметной области. Скорее всего, это существительные (объекты) или глаголы (отношения) в предложениях, которые описывают вашу предметную область.

То есть, знание того, для чего вы собираетесь использовать онтологию и насколько детальной или общей она будет, повлияет на многие решения, касающиеся моделирования. Среди нескольких жизнеспособных альтернатив нам нужно определить, какая поможет лучше решить поставленную задачу и будет более наглядной, более расширяемой и более простой в обслуживании. Нам также нужно помнить, что онтология – это модель реального мира и понятия в онтологии должны отражать эту реальность. После того, как мы определим начальную версию онтологии, мы можем 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 нам

нужно будет пересмотреть начальную онтологию. Этот процесс итеративного проектирования, вероятно, будет продолжаться в течение всего жизненного цикла онтологии.

Шаг 1. Определение области и масштаба онтологии

Мы предлагаем начать разработку онтологии с определения ее области и масштаба. То есть, ответим на несколько основных вопросов:

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

Ответы на эти вопросы могут измениться во время процесса проектирования онтологии, но в любой заданный момент времени они помогают ограничить масштаб модели.

Рассмотрим онтологию вина и еды, которую мы представили ранее. Область нашей онтологии – представление еды и вин. Мы собираемся использовать эту онтологию для приложений, которые будут предлагать хорошие сочетания вин и еды.

Конечно, в нашу онтологию будут включены понятия, описывающие различные типы вин, основные виды еды, понятие хорошего и плохого сочетания вина и еды. В то же время, маловероятно, что онтология будет включать понятия для управления инвентарем на винном заводе или служащими в ресторане, даже хотя эти понятия отчасти связаны с понятиями вина и еды.

Если онтология, которую мы проектируем, будет использоваться для помощи при обработке естественного языка статей в журналах о винах, то, возможно, понадобится включить в онтологию синонимы понятий и информации о частях речи. Если онтология будет использоваться для того, чтобы помочь посетителям ресторана решить, какое вино заказать, нам нужно будет включить информацию о розничных ценах. Если она будет использоваться для помощи покупателям вина в создании запасов в винном погребе, то могут понадобиться сведения об оптовых ценах и о наличии вин. Если люди, которые будут поддерживать онтологию, опишут предметную область языком, отличающимся от языка пользователей онтологии, то нам может потребоваться предоставить таблицу соответствий между языками.

Вопросы для проверки компетентности

Один из способов определить масштаб онтологии – это набросать список вопросов, на которые должна ответить база знаний, основанная на онтологии, т.е. **вопросы для проверки компетентности** (Gruninger and Fox 1995). Эти вопросы будут служить лакмусовой бумажкой: Содержит ли онтология достаточно информации для ответа на эти типы вопросов? Требуется ли для ответов особый уровень детализации или представление определенной области? Эти вопросы для проверки компетентности являются всего лишь формальными и не должны быть исчерпывающими.

В области вина и еды возможны следующие вопросы для проверки компетентности:

1. Какие характеристики вина мне следует учитывать при выборе вина?
2. Вино Bordeaux красное или белое?
3. Хорошо ли сочетается Cabernet Sauvignon с морскими продуктами?
4. Какое вино лучше всего подойдет к жареному мясу?
5. Какие характеристики вина влияют на его сочетаемость с блюдом?
6. Влияет ли с год производства вина на его букет или крепость?
7. Какие урожаи Napa Zinfandel были хорошими?

Судя по этому списку вопросов, онтология будет включать информацию о различных характеристиках вина и типах вин, годах производства вин (хороших и плохих), классификациях еды, которые нужно учесть при выборе подходящего вина, рекомендуемых сочетаниях вина и еды.

Шаг 2. Рассмотрение вариантов повторного использования существующих онтологий

Почти всегда стоит учесть, что сделал кто-то еще, и проверить, можем ли мы улучшить и расширить существующие источники для нашей конкретной предметной области и задачи.

Повторное использование существующих онтологий может быть необходимым, если нашей системе нужно взаимодействовать с другими приложениями, которые уже вошли в отдельные онтологии или контролируемые словари. Многие онтологии уже доступны в электронном виде и могут быть импортированы в используемую Вами среду проектирования онтологий. Формализм онтологии часто не имеет значения, т.к. многие системы представления знаний могут импортировать и экспортировать онтологии. Даже если система представления знаний не может работать напрямую с отдельным формализмом, задача перевода онтологии из одного формализма в другой обычно не является сложной.

В литературе и всемирной паутине существуют библиотеки повторно используемых онтологий. Например, мы можем использовать библиотеку онтологий Ontolingua (<http://www.ksl.stanford.edu/software/ontolingua/>) или библиотеку онтологий DAML (<http://www.daml.org/ontologies/>). Существует также ряд общедоступных коммерческих онтологий (например, UNSPSC (www.unspsc.org), RosettaNet (www.rosettanet.org), DMOZ (www.dmoz.org)).

К примеру, база знаний по французским винам уже может существовать. Если мы можем импортировать эту базу знаний и онтологию, на которой она основана, то у нас будет не только классификация французских вин, но и первый шаг к классификации характеристик вин, использующихся для разделения и описания вин. Списки свойств вина уже могут быть доступны на коммерческих веб-сайтах, таких как <http://www.wines.com/>, которые клиенты используют при покупке вин.

Тем не менее, в этом руководстве мы будем считать, что соответствующих онтологий еще не существует, и начнем разрабатывать онтологию с нуля.

Шаг 3. Перечисление важных терминов в онтологии

Полезно составить список всех терминов, о которых мы хотели бы сказать что-либо или которые хотели бы объяснить пользователю. Какие термины мы бы хотели рассмотреть? Какие свойства имеют эти термины? Что бы мы хотели сказать об этих терминах? Например, в число важных терминов, связанных с винами, входят вино, виноград, винный завод, местоположение, цвет вина, его крепость, вкус и содержание сахара; различные виды еды, такие как рыба и черное мясо; типы вина, такие как белое вино и т.д. В начале важно получить полный список терминов, не беспокоясь о пересечении понятий, которые они представляют, об отношениях между терминами, о возможных свойствах понятий или о том, чем являются понятия – классами или слотами.

Следующие два шага – разработка иерархии классов и определение свойств понятий (слотов) – тесно переплетены. Сложно выполнить сначала один из них, а потом – другой. Обычно в иерархии мы даем несколько формулировок понятий и затем описываем свойства этих понятий и т.д. Также эти два шага – самые важные шаги в процессе проектирования онтологии. Здесь мы опишем их вкратце, а затем в следующих двух главах рассмотрим более сложные проблемы, которые необходимо принять во внимание, часто встречающиеся трудности, решения, которые нужно принять, и т.д.

Шаг 4. Определение классов и иерархии классов

Существует несколько возможных подходов для разработки иерархии классов (Uschold and Gruninger 1996):

- Процесс **нисходящей** разработки начинается с определения самых общих понятий предметной области с последующей конкретизацией понятий. Например, мы можем начать с создания классов для общих понятий Вино и Еда. Затем мы конкретизируем класс Вино, создавая его подклассы: Белое вино, Красное вино, Розовое вино. Мы можем еще дальше категоризировать класс Красное Вино, например, в Syrah, Red Burgundy, Cabernet Sauvignon и т.д.
- Процесс **восходящей** разработки начинается с определения самых конкретных классов, листьев иерархии, с последующей группировкой этих классов в более общие понятия. Например, сначала мы определяем классы для вин Pauillac и Margaux. Затем мы создаем общий надкласс для двух этих классов – Medoc, который, в свою очередь является подклассом Bordeaux.
- Процесс **комбинированной** разработки – это сочетание нисходящего и восходящего подходов: Сначала мы определяем более заметные понятия, а затем соответствующим образом обобщаем и

ограничиваем их. Мы могли бы начать с нескольких понятий высшего уровня, таких как Вино, и нескольких конкретных понятий, таких как Margaux. Затем мы можем соотнести их с понятием среднего уровня, таким как Medoc. После этого нам может понадобиться сформировать все классы вин из области Франции, формируя таким образом ряд понятий среднего уровня.

На рис. 2 показано возможное деление на различные уровни обобщения.

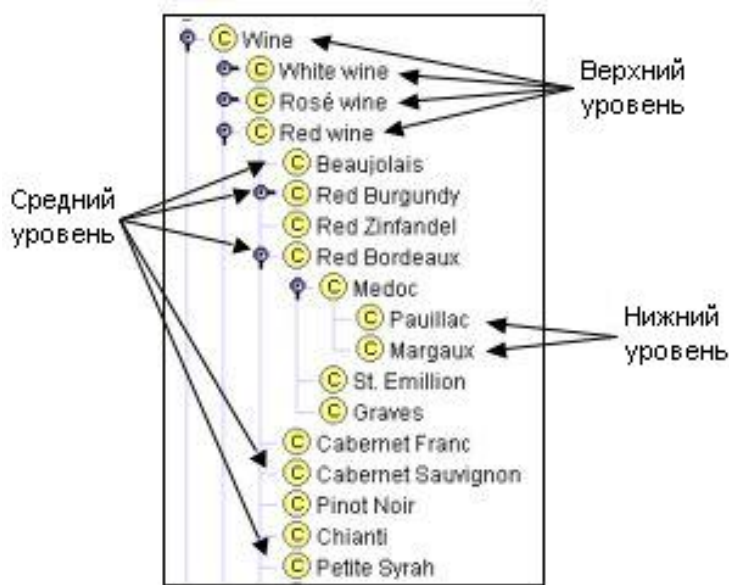


Рис. 2. Различные уровни таксономии Вино: Вино, Красное вино, Белое вино, Розовое вино – более общие понятия, верхний уровень. Pauillac и Margaux – самые конкретные классы в иерархии, нижний уровень.

Ни один из этих трех методов не лучше других по своей сути. Выбор подхода в большой степени зависит от личного взгляда на предметную область. Если разработчик склонен к рассмотрению предметной области сверху вниз, то ему, возможно, больше подойдет нисходящий метод. Часто для многих разработчиков онтологий самым простым является комбинированный метод, т.к. понятия, находящиеся «посередине», имеют тенденцию быть самыми наглядными понятиями в предметной области (Rosch 1978).

Если вы склонны делать сначала самую общую классификацию вин, то вам больше подойдет нисходящий метод. Если вы бы начали приводить конкретные примеры, то более подходящим является восходящий метод.

Какой метод мы бы ни выбрали, обычно мы начинаем с определения классов. Из списка, составленного в Шаге 3, мы выбираем термины, которые описывают объекты, существующие независимо, а не термины, которые описывают эти объекты. В онтологии эти термины будут классами и станут точками привязки в иерархии классов^[2]. Мы организуем классы в иерархическую таксономию, задавая вопрос: если объект является экземпляром одного класса, будет ли он обязательно (т.е. по определению) экземпляром некоторого другого класса?

Если класс A – надкласс класса B, то каждый экземпляр B также является экземпляром A.

Другими словами, класс B представляет собой понятие, которое является «разновидностью» A.

Например, каждое вино Pinot Noir – обязательно красное вино. Поэтому класс Pinot Noir – подкласс класса Красное вино.

На рис. 2 показана часть иерархии классов онтологии по винам. В 4-й главе детально рассмотрено, что нужно искать при определении иерархии классов.








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 ^I	Instance	single	classes={Winery}	
 name	String	single		
 sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}	

Рис. 3. Слоты класса Вино и фацеты этих слотов. Значок “I” рядом со слотом производитель указывает, что у слота есть обратный слот (Глава 5.1.).

Шаг 5. Определение свойств классов – слотов

Классы сами по себе не предоставляют достаточно информации для ответа на вопросы проверки компетентности из Шага 1. После определения некоторого количества классов мы должны описать внутреннюю структуру понятий.

Мы уже выбрали классы из списка терминов, который мы создали на Шаге 3. Большинство оставшихся терминов, вероятно, будут свойствами этих классов. Эти термины включают, 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. Таким образом, у класса Вино будут следующие слоты: цвет, крепость, вкус и сахар. А у класса Винный завод будет слот местоположение.

Вообще, в онтологии слотами могут стать несколько типов свойств объектов:

- «внутренние» свойства, такие как вкус вина;
- «внешние» свойства, такие как 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 индивидуальными концептами; это отношения между отдельными членами класса и другими элементами (например, производитель вина, представляющий отношение между вином и винным заводом, и виноград, из которого произведено вино).

Таким образом, в дополнение к ранее определенным свойствам, к классу Вино нам нужно добавить следующие слоты: название, область, производитель, виноград. На рис. 3 показаны слоты класса Вино.

Все подклассы класса **наследуют** слот этого класса. Например, все слоты класса Вино будут унаследованы всеми подклассами этого класса, включая Красное Вино и Белое Вино. К классу Красное Вино мы добавим дополнительный слот уровень танина (низкий, средний или высокий). Слот уровень танина будет унаследован всеми классами, представляющими красные вина (такие как Bordeaux и Beaujolais).

Слот должен быть привязан к самому общему классу, у которого может быть данное свойство. Например, крепость и цвет вина нужно будет привязать к классу Вино, т.к. это самый общий класс, чьи экземпляры будут иметь крепость и цвет.

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. Например, значение слота название (как в «название вина») – одна строка. То есть, название – это слот с типом значения Строка. Слот производит (как в выражении «винный завод производит эти вина») может иметь множественные значения, которые являются экземплярами класса Вино. То есть, производит – это слот с типом значения Экземпляр, и разрешенным классом является Вино.

Сейчас мы опишем несколько общих фацетов.

Мощность слота

Мощность слота определяет, сколько значений может иметь слот. В некоторых системах различаются только единичная мощность (возможно только одно значение) и множественная мощность (возможно любое число значений). Крепость вина будет слотом единичной мощности (вино может иметь только одну крепость). Вина, производимые на конкретном заводе, заполняют слот множественной мощности производит класса Винный завод.

Некоторые системы позволяют определить минимальную и максимальную мощность для того, чтобы более точно описать количество значений слота. Минимальная мощность N означает, что слот должен иметь не менее N значений. Например, слот виноград класса Вино имеет минимальную мощность 1: каждое вино делается, как минимум, из одного сорта винограда. Максимальная мощность M означает, что слот может иметь максимум M значений. Максимальная мощность слота виноград для вин из одного сорта винограда равняется 1. Иногда полезно установить максимальную мощность в 0. Эта установка будет означать, что для определенного подкласса слот не может иметь значений.

Тип значения слота

Факет типа значения описывает, какие типы значений можно ввести в слот. Вот список наиболее общих типов значений:

- **Строка** – самый простой тип значения, который используется в таких слотах, как название: значением является простая строка.
- **Число** (иногда используются более конкретные типы значений: Float (Число с плавающей запятой) и Integer (Целое число)) описывает слоты числовыми значениями. Например, стоимость вина может иметь тип Float.
- **Булевы** слоты – это простые флаги «да - нет». Например, если мы не будем представлять игристые вина как отдельный класс, то принадлежность к игристым винам может быть показана значением булевого слота: если значение «истина» («да»), то вино игристое, а если значение «ложь» («нет»), то вино не игристое.
- **Нумерованные** слоты определяют список конкретных разрешенных значений слота. Например, мы можем установить, что слот вкус может принять одно из трех возможных значений: сильный, умеренный и мягкий. В Protege-2000 нумерованные слоты имеют тип Символ.
- **Слоты-экземпляры** позволяют определить отношения между индивидными концептами. Слоты с типом значения Экземпляр также должны определять список разрешенных классов, экземпляры которых можно использовать. Например, слот производит класса Винный завод в качестве значений может иметь экземпляры класса Вино^[3].

На рис. 4 показано определение слота производит класса Винный завод.

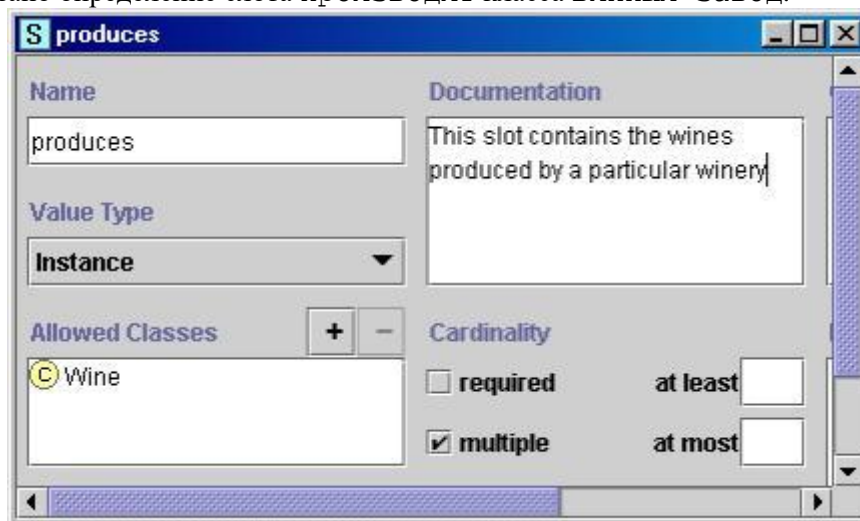


Рис. 4. Определение слота производит, который описывает вина, производимые на винном заводе. Слот имеет множественную мощность и значение типа Экземпляр.

Разрешенным классом для значений этого слота является класс Вино.

Домен слота и диапазон значений слота

Разрешенные классы для слотов типа Экземпляр часто называют **диапазоном значений** слота. В примере на рис. 4 класс Вино является диапазоном значений слота производит. Некоторые системы позволяют ограничить диапазон значений слота, если слот привязан к определенному классу.

Классы, к которым слот привязан, или классы, свойство которых слот описывает, называются **доменом** слота. Класс Винный завод – домен слота производит. В системах, где мы *привязываем* слоты к классам, домен слота обычно составляют классы, к которым привязан слот. Нет необходимости отдельно определять домен.

Основные правила определения домена слота и диапазона значений слота схожи друг с другом:

При определении домена или диапазона значений слота найдите наиболее общие классы или класс, которые могут быть соответственно доменом или диапазоном значений слотов.

*С другой стороны, не определяйте слишком общий домен и диапазон значений: все классы в домене слота должны быть описаны слотом, а экземпляры всех классов в диапазоне значений слота должны являться потенциальными заполнителями слота. Не выбирайте слишком общий класс для диапазона значений (то есть, вы не захотите делать **THING** диапазоном значений, а захотите выбрать класс, который охватит все заполнители.)*

Вместо того чтобы перечислить все возможные подклассы класса Вино для диапазона значений слота производит, просто внесите в список класс Вино. В то же время, нам не нужно определять диапазон значений слота как **THING** (самый общий класс в онтологии).

Конкретнее:

Если список классов, определяющих диапазон значений слота или домен слота, включает класс и его подкласс, удалите подкласс.

Если диапазон значений слота содержит и класс Вино, и класс Красное Вино, мы можем удалить Красное Вино из диапазона значений, т.к. он не добавляет новую информацию: Красное Вино – это подкласс класса Вино, и поэтому диапазон значений слота уже неявно включает его, также как и все другие подклассы класса Вино.

Если список классов, определяющих диапазон значений слота или домен слота, включает все подклассы класса A, но не включает сам класс A, то в диапазон значений должен входить только класс A, а не его подклассы.

Вместо указания того, что диапазон значений слота включает Красное Вино, Белое Вино и Розовое Вино (перечисление всех прямых подклассов класса Вино), мы можем ограничить диапазон значений самим классом Вино.

Если список классов, определяющих диапазон значений слота или домен слота, включает почти все подклассы класса A, подумайте, может, для определения диапазона значений лучше подойдет класс A.

В системах, где привязка слота к классу равнозначна добавлению класса к домену слота, к привязке слота применяются те же правила: С одной стороны, нам нужно постараться сделать его как можно более общим. С другой стороны, мы должны гарантировать, что каждый класс, к которому мы привязываем слот, на самом деле имеет свойство, которое представляет слот. Мы можем привязать слот уровень танина к каждому классу, представляющему красные вина (например, Bordeaux, Merlot, Beaujolais и т.д.). Однако, т.к. все красные вина имеют свойство «уровень танина», то вместо этого нам нужно прикрепить этот слот к более общему классу Красные Вина. Будет неправильно дальше обобщать домен слота уровень танина (привязка его к классу Вино), т.к. мы не используем уровень танина для описания, к примеру, белых вин.

Шаг 7. Создание экземпляров

Последний шаг – это создание отдельных экземпляров классов в иерархии. Для определения отдельного экземпляра класса требуется (1) выбрать класс, (2) создать отдельный экземпляр этого класса и (3) ввести значения слотов. Например, мы можем создать отдельный экземпляр Chateau-Morgon-Beaujolais для представления определенного типа вина Beaujolais. Chateau-Morgon-Beaujolais – это экземпляр класса Beaujolais,

представляющего все вина Beaujolais. У этого экземпляра определены следующие значения слотов (рис. 5):

- 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



Рис. 5. Определение экземпляра класса Beaujolais. Экземпляром является вино Chateau Morgon Beaujolais из области Beaujolais, произведенное из винограда Gamay на заводе Chateau Morgon. Оно легкое, с мягким вкусом, красное, с низким уровнем танина. Это сухое вино.

4. Определение классов и иерархии классов

В этой главе говорится о том, за чем нужно следить, и об ошибках, которые легко сделать при определении классов и иерархии классов (Шаг 4 из Главы 3). Как мы уже говорили ранее, для любой предметной области не существует единственной правильной иерархии классов. Иерархия зависит от возможных способов применения онтологии, уровня детализации, необходимого для приложения, личных предпочтений и иногда от требований по совместимости с другими моделями. Тем не менее, мы рассматриваем несколько руководящих принципов, которые нужно учитывать при разработке иерархии классов. После определения значительного количества новых классов полезно остановиться и проверить, соответствует ли возникающая иерархия этим руководящим принципам.

4.1. Обеспечение правильности иерархии классов

Отношение “is-a”¹

Иерархия классов представляет отношение “is-a”: класс А – это подкласс В, если каждый экземпляр В также является экземпляром А. Например, Chardonnay – подкласс класса Белое Вино. Другой способ подхода к таксономическому отношению – это отношение “kind-of”²: Chardonnay – вид Белого вина. Реактивный лайнер – вид самолета. Мясо – вид еды.

Подкласс класса представляет понятие, которое является «разновидностью» понятия, представляемого надклассом.

Отдельно взятое вино не является подклассом всех вин

¹ Буквально «является».

² Буквально «вид, разновидность [чего-то]».

Распространенная ошибка при моделировании – это включение в иерархию варианта одного и того же понятия как в единственном, так и во множественном числе, сделав первое подклассом второго. Например, будет неправильно определить класс Вина и класс Вино как подкласс класса Вина. Как только вы начинаете считать, что иерархия представляет собой отношение “kind-of”, то ошибка при моделировании становится очевидной: отдельное Вино не является **видом** Вин. Лучший способ избежать таких ошибок – всегда использовать имена классов или в единственном, или во множественном числе (присваивание имен понятиям подробно рассмотрено в Главе 6).

Транзитивность иерархических отношений

Отношение подкласса транзитивно:

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

Например, мы можем определить класс Вино, а потом определить класс Белое вино как подкласс класса Вино. Затем мы определяем класс Chardonnay как подкласс класса Белое вино. Транзитивность отношения подкласса означает, что класс Chardonnay также является подклассом класса Вино. Иногда мы различаем прямые и косвенные подклассы. **Прямой подкласс** – самый близкий подкласс класса: в иерархии между классом и его прямым подклассом нет других классов. То есть, между классом и его прямым надклассом в иерархии нет других классов. В нашем примере Chardonnay – это прямой подкласс класса Белое вино и не прямой подкласс класса Вино.

Развитие иерархии классов

Поддержание последовательной иерархии классов может вызывать сложности по мере того, как развиваются предметные области. Например, много лет все вина Zinfandel были красными. Поэтому мы определяем класс вин Zinfandel как подкласс класса Красное вино. Тем не менее, производители вин иногда начали выжимать виноград и сразу удалять цветообразующие вещества из винограда, изменяя таким образом цвет получаемого вина. Так мы получаем «белое Zinfandel» розового цвета. Теперь нам нужно разбить класс Zinfandel на 2 класса zinfandel – Белое zinfandel и Красное zinfandel – и классифицировать их как подклассы классов Розовое вино и Красное вино соответственно.

Классы и их имена

Важно различать класс и его имя:

Классы представляют понятия предметной области, а не слова, которые обозначают эти понятия.

Имя класса может измениться, если мы выберем другую терминологию, но сам термин представляет объективную реальность мира. Например, мы можем создать класс Shrimps, а потом переименовать его в Prawns – класс представляет все то же понятие. Вина, которые подходили к блюдам из shrimp, должны подходить к блюдам из prawn¹.

В действительности нужно все время соблюдать правило:

Синонимы одного и того же понятия не представляют различные классы.

Синонимы – всего лишь разные имена понятия или термина. Следовательно, у нас не должно быть класса с именем Shrimp и одновременно с этим класса с именем Prawn, а также класса с именем Crevette². Предпочтительнее будет иметь один класс с именем Shrimp или Prawn. Многие системы позволяют ассоциировать с классом список синонимов, переводов или имен представления. Если система не позволяет осуществлять такие ассоциации, то синонимы всегда можно перечислить в документации к классу.

Избежание циклов классов

Нам следует избегать **циклов** в иерархии классов. Мы говорим, что в иерархии есть цикл, когда у некоторого класса A есть подкласс B и в то же время B – это надкласс A. Создание такого цикла в иерархии равнозначен объявлению того, что классы A и B эквивалентны: все экземпляры

¹ «Shrimp» и «prawn» - это синонимы, которые переводятся одинаково – «креветка».

² Также переводится как «креветка».

А – это экземпляры В, а все экземпляры В также являются экземплярами А. Действительно, поскольку В – подкласс А, то все экземпляры В должны быть экземплярами класса А. Поскольку А – подкласс В, то все экземпляры А также должны быть экземплярами класса В.

4.2. Анализ узлов-братьев в иерархии классов

Узлы-братья в иерархии классов

Узлы-братья в иерархии – это классы, которые являются прямыми подклассами одного и того же класса (см. Раздел 4.1).

Все узлы-братья в иерархии (кроме тех, что находятся в корне) должны располагаться на одном уровне обобщения.

Например, Белое вино и Chardonnay не должны быть подклассами одного и того же класса (скажем, класса Вино). Белое вино – более общее понятие, чем Chardonnay. Узлы-братья должны представлять понятия, которые находятся «на одной линии», так же как разделы одного уровня в книге должны находиться на одном уровне обобщения. В этом смысле требования к иерархии классов похожи на требования к структуре книги.

Однако понятия, которые находятся в корне иерархии (и которые всегда представлены как прямые подклассы некоторого самого общего класса, такого как Thing), представляют основные деления в предметной области и не должны быть схожими понятиями.

Что называть «слишком много», а что – «слишком мало»?

Не существует жестких правил относительно того, сколько прямых подклассов должен иметь класс. Тем не менее, во многих онтологиях с четкой структурой имеется от двух до дюжины прямых подклассов. Отсюда два руководящих принципа:

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.

Первое правило похоже на правило набора текста, которое гласит, что у маркированного текста никогда не должна быть лишь одна маркированная точка. Например, большинство красных бургундских вин – это вина Cotes d’Or. Предположим, что мы хотели представить только этот основной тип бургундских вин. Мы могли создать класс Red Burgundy и затем единственный подкласс Cotes d’Or (рис. 6а). Тем не менее, если в нашем представлении красные бургундские вина и вина Cotes d’Or, по существу, эквивалентны (все красные бургундские вина являются винами Cotes d’Or и все вина Cotes d’Or – это красные бургундские вина), то нет необходимости создавать класс Cotes d’Or, и он не добавит в представление новую информацию. Если бы нам нужно было включить вина Cotes Chalonaise (более дешевые бургундские вина из области к югу от Cotes d’Or), то мы бы создали два подкласса класса Burgundy: Cotes d’Or и Cotes Chalonaise (рис. 6б).

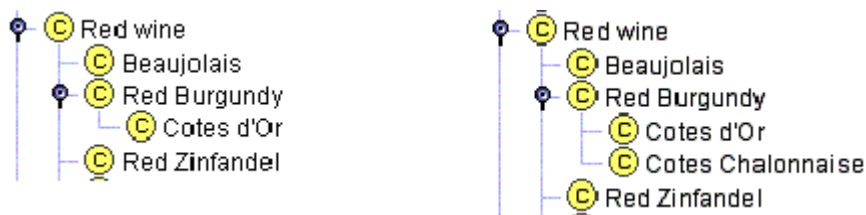


Рис. 6. Подклассы класса Red Burgundy.

Наличие у класса только одного подкласса указывает на ошибку при моделировании.

Теперь предположим, что мы перечислили все вина как прямые подклассы класса Вино. Тогда этот список будет включать такие более общие сорта, как Beaujolais и Bordeaux, так же как и более конкретные вина, как Paulliac и Margaux (рис. 7а). У класса Вино слишком много прямых подклассов и, действительно, для того чтобы онтология отражала различные сорта вин более четко, Medoc должно быть подклассом Bordeaux, а Cotes d’Or должно быть подклассом Burgundy. Кроме того, наличие таких промежуточных категорий как Красное вино и Белое

Вино также будет отражать ту понятийную модель предметной области вин, которая есть у многих людей (рис. 7б).

Тем не менее, если не существует естественных классов для группировки понятий в длинный список узлов-братьев, то не нужно создавать искусственные классы – оставьте все, как есть. В конце концов, онтология – это отражение реального мира, и если в действительности категоризации нет, то онтология должна это отражать.

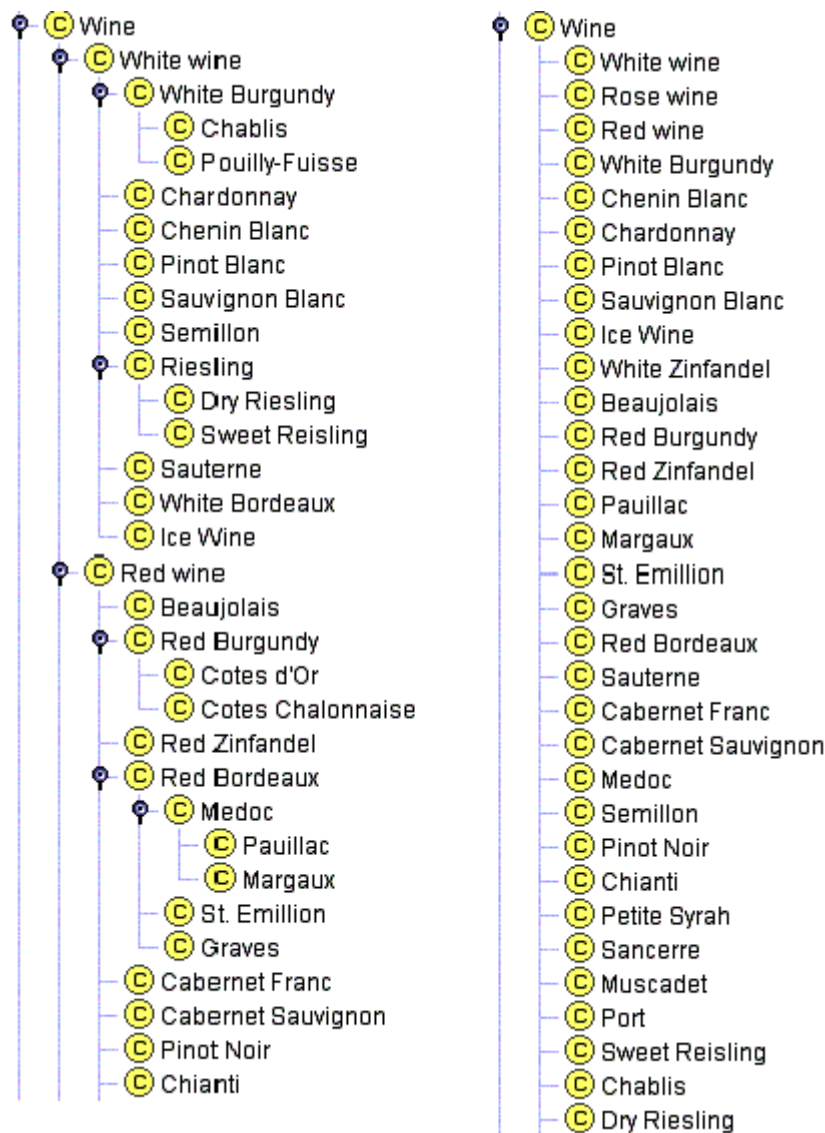


Рис. 7. Категоризация вин. Простое перечисление всех вин против нескольких уровней категоризации.

4.3. Множественное наследование.

Большинство систем представления знаний позволяют осуществлять **множественное наследование** в иерархии классов: класс может быть подклассом нескольких классов. Предположим, что мы хотим создать отдельный класс десертных вин - класс Десертное вино. Вино Port является и красным, и десертным вином^[4]. Следовательно, мы определяем, что у класса Port есть 2 надкласса: Красное вино и Десертное вино. Все экземпляры класса Port будут экземплярами как класса Красное вино, так и класса Десертное вино. Класс Port унаследует слоты и факеты от обоих родителей. Таким образом, он унаследует значение СЛАДКОЕ из слота Сахар класса Десертное вино, а также слот уровень танина и значение слота цвета класса Красное вино.

4.4. Когда вводить (или не вводить) новый класс

Одно из самых сложных решений, которое нужно принять во время моделирования, – это определить, когда ввести новый класс или когда сформулировать различие с помощью разных значений свойств. Сложно ориентироваться как в иерархии с очень большой степенью вложенности и множеством посторонних классов, так и в очень плоской иерархии, где очень мало классов, но в их слотах закодировано слишком много информации. Найти подходящий баланс нелегко.

Существует несколько практических способов определения того, когда в иерархию следует ввести новые классы:

Обычно подклассы класса (1) имеют дополнительные свойства, которых нет у надкласса, или (2) ограничения, отличные от тех, которые есть у надкласса, или (3) состоят в других отношениях, нежели надклассы.

У красных вин разные уровни танина, тогда как это свойство не используется для описания вин в общем. Слот сахар класса Десертное вино имеет значение СЛАДКОЕ, тогда как для надкласса класса Десертное вино это не так. Вина Pinot Noir могут хорошо сочетаться с морскими продуктами, тогда как другие красные вина – нет. Другими словами, мы обычно вводим в иерархию новый класс только тогда, когда мы можем сказать про этот класс что-то такое, чего мы не можем сказать о надклассе.

На практике к каждому подклассу нужно добавить новые слоты или определить у него новые значения слотов и переопределить некоторые facets наследованных слотов.

Однако иногда может быть полезно создать новые классы, даже если они не вводят никаких новых свойств.

Классы в иерархиях терминов не обязаны новые свойства.

Например, некоторые онтологии включают большие иерархии ссылок обычных терминов, используемых в предметной области. К примеру, онтология, которая лежит в основе электронной системы записи медицинской информации, может включать классификацию различных болезней. Эта классификация может быть как раз такой – иерархией терминов, без свойств (или с одним и тем же набором свойств). В этом случае по-прежнему полезно организовать термины в иерархию, а не в линейный список, потому что она (1) облегчит изучение и навигацию и (2) позволит врачу легко выбрать подходящий уровень общности термина.

Другая причина введения новых классов без новых свойств – это моделирование понятий, среди которых эксперты в предметной области обычно проводят разграничение, даже несмотря на то, что мы могли принять решение не моделировать само разграничение. Так как мы используем онтологии для содействия коммуникации между экспертами в предметной области, а также между экспертами и системами, основанными на знаниях, то в онтологии мы бы хотели отразить точку зрения эксперта на предметную область.

В конце концов, нам не следует создавать подклассы класса для каждого дополнительного ограничения. К примеру, мы ввели классы Красное вино, Белое вино и Розовое вино, так как это разделение является естественным в мире вин. Мы не ввели классы для изысканного вина, для вин с умеренным вкусом и т.д. Когда мы определяем иерархию классов, наша цель – добиться равновесия между созданием классов, которые полезны для организации классов, и созданием слишком большого числа классов.

4.5. Новый класс или значение свойства?

При моделировании предметной области нам часто нужно решать, моделировать ли определенное разграничение (такое как белое, красное или розовое вино) как значение свойства или как набор классов, что опять же зависит от масштаба предметной области и от решаемой задачи.

Создать ли нам класс Белое вино или просто создать класс Вино и ввести различные значения слота цвет? Ответ всегда зависит от того масштаба, который мы определили для онтологии. Насколько важно понятие Белое вино в нашей предметной области? Если важность вин в предметной области незначительна и то, белое это вино или нет, не особо влияет на его отношения с другими объектами, то нам не нужно вводить отдельный класс белых вин. Для модели предметной области, используемой на предприятии, где производятся винные этикетки, правила для этикеток вин различных цветов одни и те же и разделение не столь важно. И наоборот, для представления вина, еды и их подходящих сочетаний, красное вино сильно отличается от

белого: оно сочетается с другой едой, у него другие свойства и т.д. Также, цвет вина важен для базы знаний по винам, которую мы можем использовать для определения последовательности дегустации вин. Поэтому мы создаем отдельный класс Белое вино.

Если понятия с разными значениями слота становятся ограничениями для различных слотов в других классах, то для разделения нам следует создать новый класс. В противном случае разделение представляется в значении слота.

Подобным образом, в нашей онтологии вин есть такие классы как Красное Merlot и Белое Merlot, а не единый класс для всех вин Merlot: красные Merlot и белые Merlot – это действительно разные вина (сделанные из одного винограда), и если мы разрабатываем подробную онтологию, то это разделение представляет важность.

Если в предметной области разграничение представляет важность и объекты с другими значениями разграничения мы считаем другими типами объектов, то для осуществления разграничения нам нужно создать новый класс.

При принятии решения о том, вводить новый класс или нет, также может быть полезно рассмотреть потенциальные отдельные экземпляры класса.

Класс, к которому принадлежит отдельный экземпляр, не должен часто меняться.

Обычно, когда для определения различий между классами мы используем внешние, а не внутренние свойства понятий, экземпляры этих классов должны часто будут перемещаться из одного класса в другой. Например, Охлажденное вино не должно являться классом в онтологии, описывающей бутылки вин в ресторане. Свойство охлажденное должно быть просто атрибутом вина в бутылке, так как экземпляр класса Охлажденное вино может легко перестать быть экземпляром этого класса, а затем снова стать его экземпляром.

Обычно числа, цвета, местоположения являются значениями слотов и не приводят к созданию новых классов. Тем не менее, вино – особое исключение, так как цвет вина имеет первостепенную важность для его описания.

В качестве другого примера рассмотрим онтологию человеческой анатомии. Когда мы представляем ребра, создаем ли мы класс для «1-го левого ребра», потом класс для «2-го левого ребра» и т.д.? Или у нас есть класс Ребро со слотами для последовательности и стороны (левое - правое)?^[5] Если информация о каждом из ребер, которые мы представляем в онтологии, существенно отличается, то нам, действительно, нужно создать класс для каждого ребра. То есть, если мы хотим подробно представить информацию о смежности и положении (которые различны для всех ребер), а также особые функции, которые выполняет каждое ребро, и какие органы оно защищает, то нам нужно создать классы. Если мы моделируем анатомию на чуть более низком уровне обобщения и для наших потенциальных приложений все ребра очень похожи (мы говорим всего лишь о том, какое ребро сломано, судя по рентгеновскому снимку, безотносительно к другим частям тела), то нам потребуется упростить иерархию и оставить только класс Ребро с двумя слотами: сторона и порядковый номер.

4. 6. Экземпляр или класс?

Определение того, чем является определенное понятие - классом в онтологии или отдельным экземпляром - зависит от потенциальных приложений онтологии. Определение того, где заканчиваются классы и начинаются отдельные экземпляры, начинается с определения нужной глубины детализации в представлении. Глубина детализации, в свою очередь, определяется потенциальным приложением онтологии. Другими словами, какие самые конкретные элементы будут представлены в базе знаний? Если возвратиться к вопросам для проверки компетентности, которые мы определили в Шаге 1 Главы 3, самые конкретные понятия, которые будут ответами на эти вопросы, лучше всего подойдут на роль индивидуальных концептов в базе знаний.

Отдельные экземпляры - самые конкретные понятия, представленные в базе знаний.

Например, если мы собираемся говорить только о подборе сочетаний вина и еды, то нас не будут интересовать конкретные материальные бутылки вина. Поэтому такие термины как Sterling Vineyards Merlot, вероятно, будут самыми конкретными используемыми нами терминами. Следовательно, Sterling Vineyards Merlot будет экземпляром в базе знаний.

С другой стороны, если бы мы хотели поддерживать в ресторане ассортимент вин в дополнение к базе знаний хороших сочетаний «вино-еда», то отдельными экземплярами в нашей базе знаний могли бы стать отдельные бутылки каждого вина.

Аналогично, если бы мы хотели записать различные качества для каждого конкретного урожая Sterling Vineyards Merlot, то экземпляром в базе знаний стал бы конкретный урожай вина, а Sterling Vineyards Merlot стал бы классом, содержащим экземпляры всех его урожаев.

Другое правило может «переместить» некоторые отдельные экземпляры в разряд классов:

Если понятия формируют естественную иерархию, то нам нужно представить их, как классы.

Рассмотрим винные области. В начале мы можем определить основные винные регионы, такие как Франция, США, Германия и т.д. как классы, а конкретные винные области внутри этих регионов как экземпляры. Например, область Bourgogne – это экземпляр класса регион Франция. Однако мы бы также хотели отметить, что область Cotes d’Or – это область Bourgogne. Поэтому область Bourgogne должна быть классом (чтобы иметь подклассы или экземпляры). Однако представление области Bourgogne как класса, а области Cotes d’Or как экземпляра области Bourgogne кажется произвольным: очень сложно четко разграничить, какие области являются классами, а какие – экземплярами. Поэтому мы определяем все винные области как классы. Protege-2000 дает возможность пользователю определить некоторые классы как Абстрактные, показывая, что у класса не может быть прямых экземпляров. В нашем случае, все классы областей являются абстрактными (рис. 8).

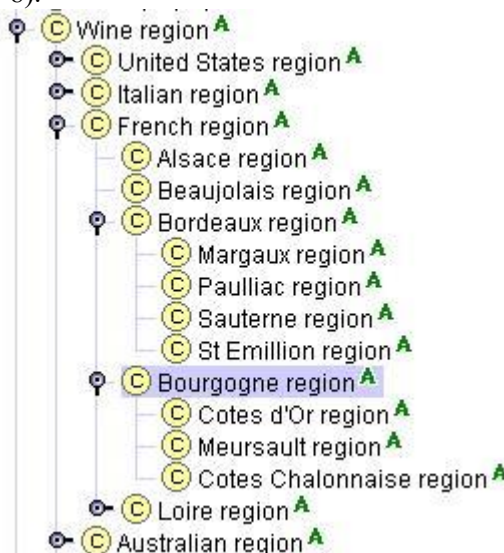


Рис. 8. Иерархия винных областей. Иконка «А» рядом с именами классов указывает на то, что это абстрактные классы и у них не может быть прямых экземпляров.

Так же иерархия классов была бы неправильной, если бы мы пропустили в имени класса слово «область». Мы не можем сказать, что класс Alsace – это подкласс класса Франция: Alsace – это не разновидность Франции. Тем не менее, область Alsace – разновидность региона Франция.

В виде иерархии можно представить только классы – в системах представления знаний нет категории «подэкземпляр». Поэтому, если среди терминов существует естественная иерархия, такая как в иерархиях терминов в Разделе 4.2, то нам нужно охарактеризовать эти термины как классы, даже если они могут и не иметь своих экземпляров.

4.7. Ограничение масштаба

В качестве последнего замечания по составлению иерархии классов, следующий набор правил всегда поможет при ответе на вопрос, закончено ли определение онтологии:

Онтология не должна содержать всю возможную информацию о предметной области: вам не нужно конкретизировать или обобщать больше, чем вам нужно для вашего приложения (не более 1 дополнительного уровня в каждую сторону).

Для нашего примера с вином и едой нам не требуется знать, из какой бумаги делаются этикетки, или как готовятся блюда из креветок.

Также онтология не должна содержать все возможные свойства классов и различия между классами в иерархии.

В нашу онтологию мы, естественно, не включаем все свойства, которые могли бы иметь вино или еда. В нашей онтологии мы представили самые значимые свойства классов элементов. Даже несмотря на то, что в книгах по винам можно узнать о размере виноградин, мы не включили эту информацию. Так же, в нашу систему мы не включили все возможные отношения между всеми терминами. К примеру, мы не включаем в онтологию такие отношения, как любимое вино или любимая еда, просто для того, чтобы сделать более полное представление обо всех взаимосвязях между определенными нами терминами более полным.

Последние правила также применимы для установления отношений между понятиями, которые мы уже включили в онтологию. Рассмотрим онтологию, описывающую биологические эксперименты. Эта онтология, вероятно, будет содержать понятие Биологические организмы. Она также будет содержать понятие Экспериментатор (включающее имя, место работы и т.д.), который проводит эксперимент. Верно то, что экспериментатор как человек также является биологическим организмом. Тем не менее, мы, наверное, не должны отражать эту особенность в онтологии: в целях этого представления экспериментатор не является биологическим организмом, и мы, наверное, никогда не будем проводить эксперименты на самих экспериментаторах. Если бы в онтологии мы делали представление всего того, что мы можем сказать о классах, то Экспериментатор бы стал подклассом класса Биологический организм. Однако нет необходимости включать это знание для возможных приложений. Фактически, включение этой дополнительной классификации существующих классов действительно мешает: теперь у экземпляра Экспериментатора будут слоты для веса, возраста, вида и других данных, которые относятся к биологическому организму, но совершенно неуместны в контексте описания эксперимента. Тем не менее, для пользователей, которые будут иметь дело с этой онтологией и которые могут не знать о задуманном нами приложении, нам нужно отразить это проектное решение в документации.

4.8. Дизъюнктивные подклассы

Многие системы позволяют нам явным образом задать, что несколько классов являются **дизъюнктивными**. Классы дизъюнктивные, если у них не может быть общих экземпляров. Например, в нашей онтологии классы Десертное вино и Белое вино *не* являются дизъюнктивными: существует множество вин, являющихся экземплярами обоих классов. Одним из таких примеров является Rothermel Trochenbierenauslese Riesling, экземпляр класса Сладкое Riesling. В то же время, классы Красное вино и Белое вино дизъюнктивны: ни одно вино не может быть одновременно и белым, и красным. Определение классов как дизъюнктивных позволяет системе лучше проверять правильность онтологии. Если мы объявим классы Красное вино и Белое вино дизъюнктивными и затем создадим класс, который будет подклассом и Riesling (подкласс Белого вина) и Port (подкласс Красного вина), то система может показать, что имеется ошибка в моделировании.

5. Определение свойств – более подробно

В этой главе мы затронем еще несколько деталей, которые нужно иметь при определении слотов в онтологии (Шаги 5 и 6 в Главе 3). В основном, мы обсуждаем обратные слоты и значения слота по умолчанию.

5.1. Обратные слоты

Значение слота может зависеть от значения другого слота. Например, если вино было произведено на винном заводе, то винный завод производит это вино. Эти два отношения, производитель и производит, называются **обратными отношениями**. Излишне хранить информацию и о том, и о другом. Когда мы знаем, что вино производится на винном заводе, то приложение, которое использует базу знаний, всегда может вывести значение для обратного отношения: винный завод производит вино. Тем не менее, с точки зрения приобретения знаний удобно иметь оба блока информации доступными в явном виде. Этот подход позволяет пользователям указать вино в одном случае и винный завод в другом. После это система

приобретения знаний может автоматически заполнить значение для обратного отношения, обеспечивая согласованность базы знаний.

В нашем примере есть пара обратных слотов: слот производитель класса Вино и слот производит класса Винный завод. Когда пользователь создает экземпляр класса Вино и заполняет значение слота производитель, система автоматически добавляет вновь созданный экземпляр к слоту производит соответствующего экземпляра класса Винный завод. Например, когда мы говорим, что Sterling Merlot производится на заводе Sterling Vineyard, система автоматически добавляет Sterling Merlot к списку вин, которые производит завод Sterling Vineyard (рис. 9).

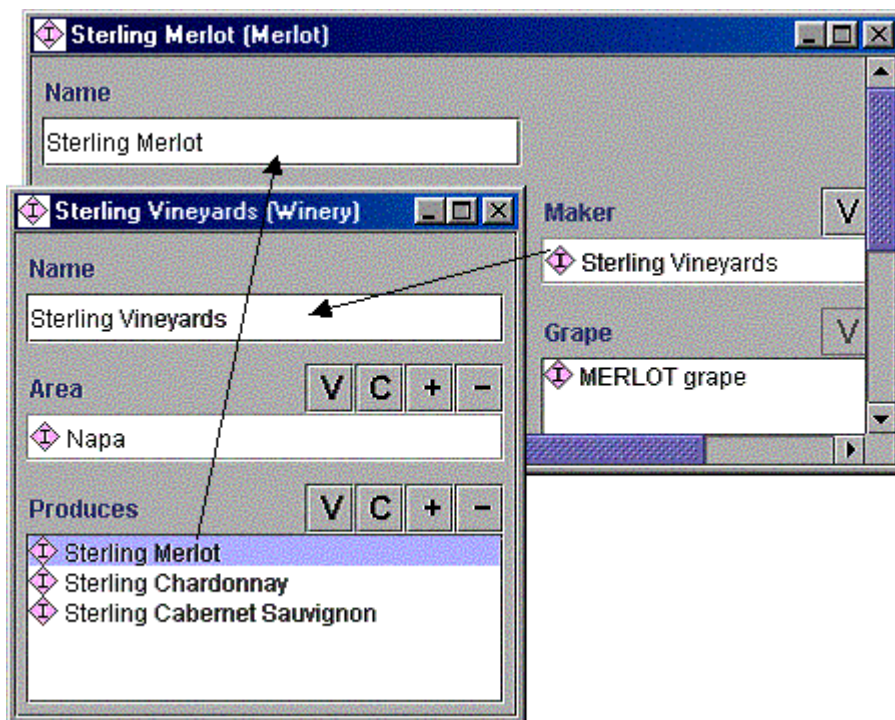


Рис. 9. Экземпляры с обратными слотами. Слот производит класса Винный завод является обратным для слота производитель класса Вино. Заполнение одного из слотов приводит к автоматическому обновлению другого.

5.2. Значения по умолчанию

Многие фреймвые системы позволяют определить для слотов значения по умолчанию. Если значение определенного слота одинаково для большинства экземпляров класса, то мы можем определить это значение как **значение слота по умолчанию**. Затем, когда создается каждый экземпляр класса, имеющего этот слот, система автоматически заполняет значение по умолчанию. После этого мы можем изменить это значение на любое другое, которое позволяют факеты. То есть, значения по умолчанию созданы для удобства: в любом случае они не накладывают какие-либо ограничения на модель или никак ее не меняют.

Например, если большинство вин, о которых мы собираемся говорить, являются крепкими, то значение крепости вина мы можем сделать «крепкое» по умолчанию. Тогда все вина, которые мы определяем, будут крепкими, если мы не укажем иное.

Обратите внимание, что это отличается от **значений слота**. Значения слота не могут быть изменены. Например, мы можем сказать, что слот сахар класса Десертное вино имеет значение «СЛАДКОЕ». Тогда у всех подклассов и экземпляров класса Десертное вино значение слота сахар будет «СЛАДКОЕ». Для всех подклассов или экземпляров этого класса это значение изменить нельзя.

6. Об именах

Определение единых правил присваивания имен понятиям в онтологии, а затем строгое соблюдение этих правил не только делает онтологию более простой для понимания, но также помогает избежать некоторых общих ошибок при моделировании. Существует много вариантов

присваивания имен понятиям. Обычно нет особой причины для выбора того или иного варианта. Тем не менее, нам нужно

Определить единые правила присваивания имен классам и слотам и придерживаться их.

На выбор правил присваивания имен влияют следующие особенности системы представления знаний:

- Имеет ли система одно и то же пространство имен классов, слотов и экземпляров? То есть, позволяет ли система иметь класс и слот с одинаковым именем (как, например, класс винный завод и слот винный завод)?
- Различает ли система регистр букв? То есть, считает ли система разными имена, которые отличаются только регистром (как Винный завод и винный завод)?
- Какие разделители в именах позволяет использовать данная система? То есть, могут ли имена содержать пробелы, запятые, звездочки и т.д.?

К примеру, Protege-2000 имеет единое пространство имен для всех своих фреймов. Она различает регистр букв. Таким образом, у нас не может быть класса винный завод и слота винный завод. Однако у нас может быть класс Винный завод (не прописные буквы) и слот винный завод. С другой стороны, CLASSIC не различает регистр букв и имеет разные пространства имен для классов, слотов и индивидуальных концептов. Таким образом, с точки зрения системы, мы можем с легкостью присвоить имя Винный завод и классу, и слоту.

6.1. Заглавные буквы и разделители

Во-первых, мы можем значительно улучшить читаемость онтологии, если мы все время будем писать названия понятий с большой буквы. Например, общепринято начинать имена классов с большой буквы, а имена слотов – с маленькой (предполагая, что система различает регистр букв).

Когда имя понятия содержит больше одного слова (как в Винный завод), нам нужно разделить слова. Вот возможные варианты:

- Использовать пробел: Винный завод (многие системы, включая Protege, позволяют использовать пробелы в именах понятий).
- Соединить слова вместе и каждое слово написать с большой буквы: ВинныйЗавод.
- Использовать в имени подчеркивание или тире, или другой разделитель: Винный_Завод, Винный_завод, Винный-Завод, Винный-завод (если вы используете разделитель, вам также нужно решить, писать каждое слово с большой буквы или нет).

Если система представления знаний позволяет использовать пробелы в именах, то для многих разработчиков онтологий пробелы могут быть самым естественным решением. Однако важно учитывать другие системы, с которыми может взаимодействовать ваша система. Если в этих системах не используются пробелы или ваше средство представления не очень хорошо обрабатывает пробелы, то может быть лучше использовать другой метод.

6.2. Единственное или множественное число

Имя класса представляет набор объектов. Например, класс Вино в действительности представляет все вина. Поэтому для многих разработчиков было бы естественнее дать классу имя Вина, а не Вино. Ни один из вариантов не лучше и не хуже другого (хотя на практике для имен классов чаще используется единственное число). Тем не менее, каким бы ни был выбор, его следует придерживаться на протяжении всей онтологии. Некоторые системы даже требуют от своих пользователей заранее объявить, какое число (единственное или множественное) они будут использовать в именах классов, и не дают им отклоняться от своего выбора.

Использование все время одной и той же формы также предотвращает такие ошибки разработчика при моделировании, как создание класса Вина, а затем создание класса Вино как его подкласса (см. Раздел 4.1).

6.3. Договоренность в отношении использования префиксов и суффиксов

Некоторые методологии по базам знаний советуют придерживаться договоренности в отношении использования префиксов и суффиксов в именах для того, чтобы различать классы и

слоты. Существует две распространенных традиции: добавлять к именам слотов *has-*¹ или предлог *-of*². Таким образом, наши слоты меняются на *его-производитель* и *его-винный_завод*, если мы выберем использование *его-*. Слоты меняются на *maker-of* и *winery-of*³, если мы выберем использование *of-*. Этот подход позволяет любому, кто посмотрит на термин, сразу же определить, что это: класс или слот. Однако имена терминов становятся немного длиннее.

6.4. Другие соображения по присваиванию имен

Еще несколько моментов, которые нужно иметь в виду при определении правил присваивания имен:

- Не добавляйте к именам понятий такие строки как «класс», «свойство», «слот» и т.д. Из контекста всегда ясно, что это, к примеру, класс или слот. В дополнение к тому, что для классов и слотов вы используете разные правила присваивания имен (скажем, пишете их с большой и с маленькой буквы соответственно), само имя будет показывать, чем является это понятие.
- Обычно лучше не сокращать имена понятий (то есть, используйте Cabernet Sauvignon, а не Cab).
- Имя надкласса должно входить или во все имена прямых подклассов, или ни в одно из них. Например, если мы создаем два подкласса класса Вино для представления красных и белых вин, то подклассы должны называться или Красное Вино и Белое Вино, или Красное и Белое, но не Красное Вино и Белое.

7. Другие ресурсы

В наших примерах в качестве среды разработки онтологий мы использовали Protege-2000. Duineveld с коллегами (Duineveld et al. 2000) описывает и сравнивает ряд других сред для разработки онтологий.

Мы постарались рассказать о самом основном о разработке онтологий и не коснулись многих углубленных тем или альтернативных методологий разработки онтологий. Gymez-Рйrez (Gymez-Рйrez 1998) и Uschold (Uschold and Gruninger 1996) представляют альтернативные методологии разработки онтологий. В руководстве по Ontolingua (Farquhar 1997) говорится о некоторых формальных аспектах моделирования знаний.

В настоящее время исследователи придают особое значение не только разработке онтологий, но также и анализу онтологий. Чем больше онтологий будет создаваться и повторно использоваться, тем больше будет инструментальных средств для анализа онтологий. К примеру, Chimaera (McGuinness et al. 2000) предоставляет диагностические инструментальные средства для анализа онтологий. Анализ, который осуществляет Chimaera, включает как проверку логической верности онтологии, так и диагностику типичных ошибок при проектировании онтологий. Разработчик онтологий может провести диагностику разрабатываемой онтологии с помощью Chimaera, чтобы определить соответствие общим способам моделирования онтологий.

8. Заключение

В этом руководстве мы описали методологию разработки онтологии для декларативных фреймовых систем. Мы перечислили шаги при разработке онтологии и затронули сложные вопросы определения иерархий классов и свойств классов и экземпляров. Тем не менее, помимо всех правил и советов, следует помнить одну из важнейших вещей: *для любой предметной области не существует единственно правильной онтологии*. Проектирование онтологии – это творческий процесс и две онтологии, разработанные разными людьми, никогда не будут одинаковыми. Потенциальные приложения онтологии, а также понимание разработчиком предметной области и его точка зрения на нее будут, несомненно, влиять на принятие решений

¹ «has» в данном случае можно перевести как «его».

² В английском языке предлог «of» используется для отношений, которые в русском языке передаются родительным падежом без предлога. Поэтому использование этого суффикса для русскоязычных пользователей теряет смысл.

³ *maker-of* – производитель [чего-то], а *winery-of* – винный завод [по производству чего-то]

при проектировании онтологии. “Цыплят по осени считают” – мы можем оценить качество нашей онтологии, только используя ее в приложениях, для которых мы ее разработали.

Благодарности

Protege-2000 (<http://protege.stanford.edu/>) была разработана группой Марка Мьюсена (Mark Musen) в Stanford Medical Informatics¹. Некоторые рисунки мы сделали с помощью OntoViz – плагина² к Protege-2000. Исходную версию онтологии вин мы импортировали из библиотеки онтологий Ontolingua (<http://www.ksl.stanford.edu/software/ontolingua/>), в которой, в свою очередь, использовалась версия, опубликованная Brachman и коллегами (Brachman et al. 1991) и распространяемая с системой представления знаний CLASSIC. Затем мы модифицировали онтологию, чтобы представить принципы концептуального моделирования для декларативных фреймовых онтологий. Подробные комментарии Рэя Фергесона (Ray Ferguson) и Мор Пелег (Mor Peleg) к ранним черновым вариантам значительно улучшили эту статью.

Литература

- Booch, G., Rumbaugh, J. and Jacobson, I. (1997). *The Unified Modeling Language userguide*: Addison-Wesley.
- Brachman, R.J., McGuinness, D.L., Patel-Schneider, P.F., Resnick, L.A. and Borgida, A. (1991). Living with CLASSIC: When and how to use KL-ONE-like language. *Principles of Semantic Networks*. J. F. Sowa, editor, Morgan Kaufmann: 401-456.
- Brickley, D. and Guha, R.V. (1999). Resource Description Framework (RDF) Schema Specification. Proposed Recommendation, World Wide Web Consortium: <http://www.w3.org/TR/PR-rdf-schema>.
- Chimaera (2000). Chimaera Ontology Environment. www.ksl.stanford.edu/software/chimaera
- Duineveld, A.J., Stoter, R., Weiden, M.R., Kenepa, B. and Benjamins, V.R. (2000). WonderTools? A comparative study of ontological engineering tools. *International Journal of Human-Computer Studies* **52**(6): 1111-1133.
- Farquhar, A. (1997). Ontolingua tutorial. <http://ksl-web.stanford.edu/people/axf/tutorial.pdf>
- Gymez-Pérez, A. (1998). Knowledge sharing and reuse. *Handbook of Applied Expert Systems*. Liebowitz, editor, CRC Press.
- Gruber, T.R. (1993). A Translation Approach to Portable Ontology Specification. *Knowledge Acquisition* **5**: 199-220.
- Gruninger, M. and Fox, M.S. (1995). Methodology for the Design and Evaluation of Ontologies. In: *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95*, Montreal.
- Hendler, J. and McGuinness, D.L. (2000). The DARPA Agent Markup Language. *IEEE Intelligent Systems* **16**(6): 67-73.
- Humphreys, B.L. and Lindberg, D.A.B. (1993). The UMLS project: making the conceptual connection between users and the information they need. *Bulletin of the Medical Library Association* **81**(2): 170.
- McGuinness, D.L., Abrahams, M.K., Resnick, L.A., Patel-Schneider, P.F., Thomason, R.H., Cavalli-Sforza, V. and Conati, C. (1994). Classic Knowledge Representation System Tutorial. <http://www.bell-labs.com/project/classic/papers/ClassTut/ClassTut.html>
- McGuinness, D.L., Fikes, R., Rice, J. and Wilder, S. (2000). An Environment for Merging and Testing Large Ontologies. *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000)*. A.G. Cohn, F. Giunchiglia and B. Selman, editors. San Francisco, CA, Morgan Kaufmann Publishers.
- McGuinness, D.L. and Wright, J. (1998). Conceptual Modeling for Configuration: A Description Logic-based Approach. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing - special issue on Configuration*.
- Musen, M.A. (1992). Dimensions of knowledge sharing and reuse. *Computers and Biomedical Research* **25**: 435-467.

¹ Stanford Medical Informatics (SMI) – университетская исследовательская группа кафедры медицины Школы Медицины при Стэнфордском университете.

² Плагин (модуль расширения, подключаемый модуль) – дополнительный программный модуль, способный функционировать как составная часть основной программы.

Ontolingua (1997). Ontolingua System Reference Manual.<http://www-ksl-svc.stanford.edu:5915/doc/frame-editor/index.html>

Price, C. and Spackman, K. (2000). SNOMED clinical terms.*BJHC&IM-British Journal of Healthcare Computing & Information Management***17**(3): 27-31.

Protege (2000). The Protege Project.<http://protege.stanford.edu>

Rosch, E. (1978). Principles of Categorization.*Cognition and Categorization*. R. E. and B. B. Lloyd, editors. Hillside, NJ, Lawrence Erlbaum Publishers:27-48.

Rothenfluh, T.R., Gennari, J.H., Eriksson, H., Puerta, A.R., Tu, S.W. and Musen, M.A. (1996). Reusable ontologies, knowledge-acquisition tools, and performance systems: PROTEGE-II solutions to Sisyphus-2.*International Journal of Human-Computer Studies***44**: 303-332.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. (1991). *Object-oriented modeling and design*. Englewood Cliffs, New Jersey: Prentice Hall.

Uschold, M. and Gruninger, M. (1996). Ontologies: Principles, Methods and Applications.*Knowledge Engineering Review***11**(2).

^[1] Имена классов мы начинаем с большой буквы, а имена слотов – с маленькой. Также для всех терминов из онтологии, приведенной в качестве примера, мы используем шрифт печатной машинки.

^[2] Мы также можем рассматривать классы как унарные предикаты – вопросы с одним аргументом. Например, «Этот предмет является вином?» Унарные предикаты (или классы) противоположны бинарным предикатам (или слотам) – вопросам с двумя аргументами. Например, «Этот предмет имеет сильный вкус?» «Какой вкус у этого предмета?»

^[3] Некоторые системы только определяют тип значения с помощью класса и не требуют специальной формулировки для слотов-экземпляров.

^[4] В нашей онтологии мы решили представить только красные вина Port: белые вина Port существуют, но они очень редко встречаются.

^[5] Здесь мы предполагаем, что каждый анатомический орган является классом, поскольку мы также хотели бы говорить о «1-м левом ребре Джона». В нашей онтологии отдельные органы реально существующих людей будут представлены как индивидные концепты.