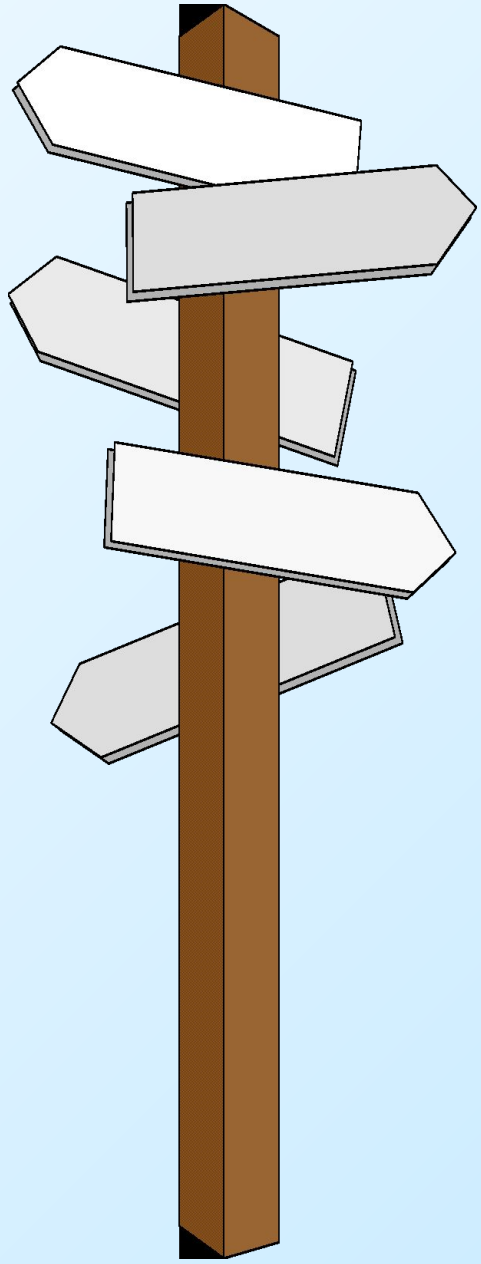
д.т.н. Валькман Юрий Роландович, зав. отд. распределенных интеллектуальных систем
Международного научно-учебного Центра информационных технологий и систем
Киев, yur@valkman.kiev.ua

ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ 2-1:
Разработка онтологий 101-1

Основные темы лекции

1. Зачем создавать онтологию?
2. Из чего состоит онтология?
3. Простая методология инженерии знаний.
   - Шаг 1. Определение области и масштаба онтологии
   - Шаг 2. Рассмотрение вариантов повторного использования существующих онтологий
   - Шаг 3. Перечисление важных терминов в онтологии
   - Шаг 4. Определение классов и иерархии классов
   - Шаг 5. Определение свойств классов – слотов
   - Шаг 6. Определение фацетов слотов
   - Шаг 7. Создание экземпляров
4. Определение классов и иерархии классов
   - 4.1. Обеспечение правильности иерархии классов
   - 4.2. Анализ узлов-братьев в иерархии классов
   - 4.3. Множественное наследование
   - 4.4. Когда вводить (или не вводить) новый класс
   - 4.5. Новый класс или значение свойства?
   - 4.6. Экземпляр или класс?
   - 4.7. Ограничение масштаба
   - 4.8. Дизъюнктивные подклассы
5. Определение свойств – более подробно
6. Об именах
7. Другие ресурсы
8. Заключение



Разработка онтологий 101: руководство по созданию Вашей первой онтологии

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 работы нужно понимать как «Введение в разработку онтологий: Руководство по созданию Вашей первой онтологии».

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 ИИ 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:

- 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.

1. 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.

2. 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.

3. 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.

4. 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).

5. 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.

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, здесь 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:

1. Одно приложение могло бы составлять список вин для меню на текущий день или

2. отвечать на запросы официантов и посетителей.

3. Другое приложение могло бы анализировать инвентарный перечень винного погреба и предлагать категории вин для пополнения и конкретные вина для закупки к следующим меню или

4. для поваренных книг.

Об этом руководстве

Мы основываемся на нашем опыте использования Protege-2000, Ontolingua, Chimaera 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 работы, сделан на основе примерной базы знаний, которая представлена в работе, описывающей CLASSIC – систему представления знаний, основанную на описательно-логическом подходе. В учебном пособии по CLASSIC этот пример получил дальнейшее развитие.

Protege-2000 и другие фреймовые системы описывают онтологии декларативным образом, определяя явным образом, какова классовая иерархия и к каким классам принадлежат индивидные концепты.

Некоторые идеи по разработке онтологий в этом руководстве берут свое начало в литературе по объектно-ориентированному проектированию. 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 ИИ содержится много определений понятия онтологии, многие из которых противоречат друг другу.

В этой работе ОНТОЛОГИЯ – формальное явное описание

- ***понятий*** **в рассматриваемой предметной области (классов (*иногда их называют понятиями*)),**
- ***свойств каждого понятия*****, описывающих различные свойства и**
- ***атрибуты понятия*** **(слотов (*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 *Chвteau Lafite Rothschild Pauillac* - крепкое,**
- **оно производится на винном заводе *Chвteau 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 Chвteau 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.**

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

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

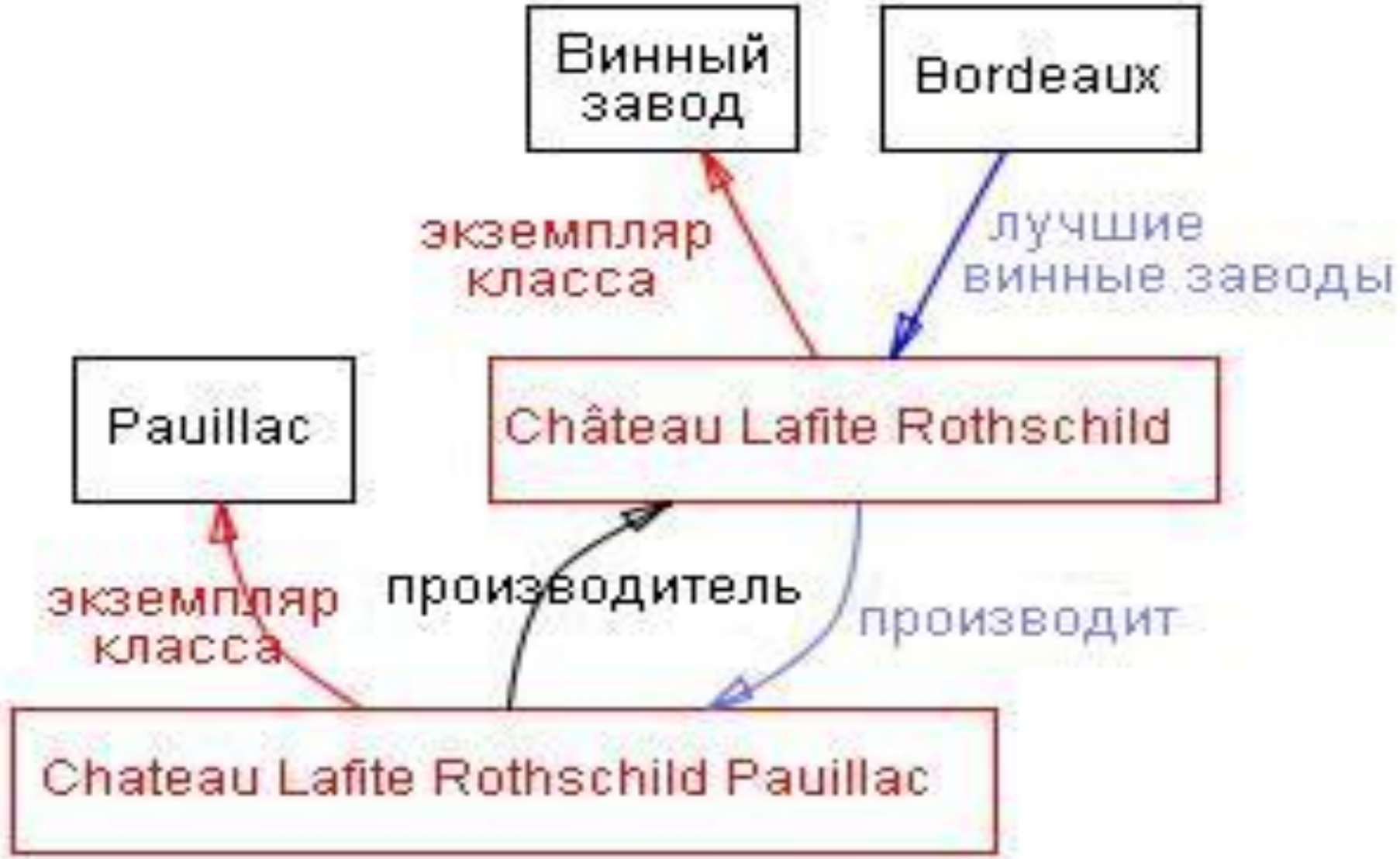
На практике разработка онтологии включает:

- 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.

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:

- **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?**

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.

1. Область нашей онтологии – представление еды и вин.

2. Мы собираемся использовать эту онтологию для приложений, которые будут предлагать хорошие сочетания вин и еды.

3. 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.

4. Если онтология, которую мы проектируем, будет использоваться для помощи при обработке естественного языка статей в журналах о винах, то, возможно, понадобится включить в онтологию синонимов 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.

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**

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 онтологии.

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:

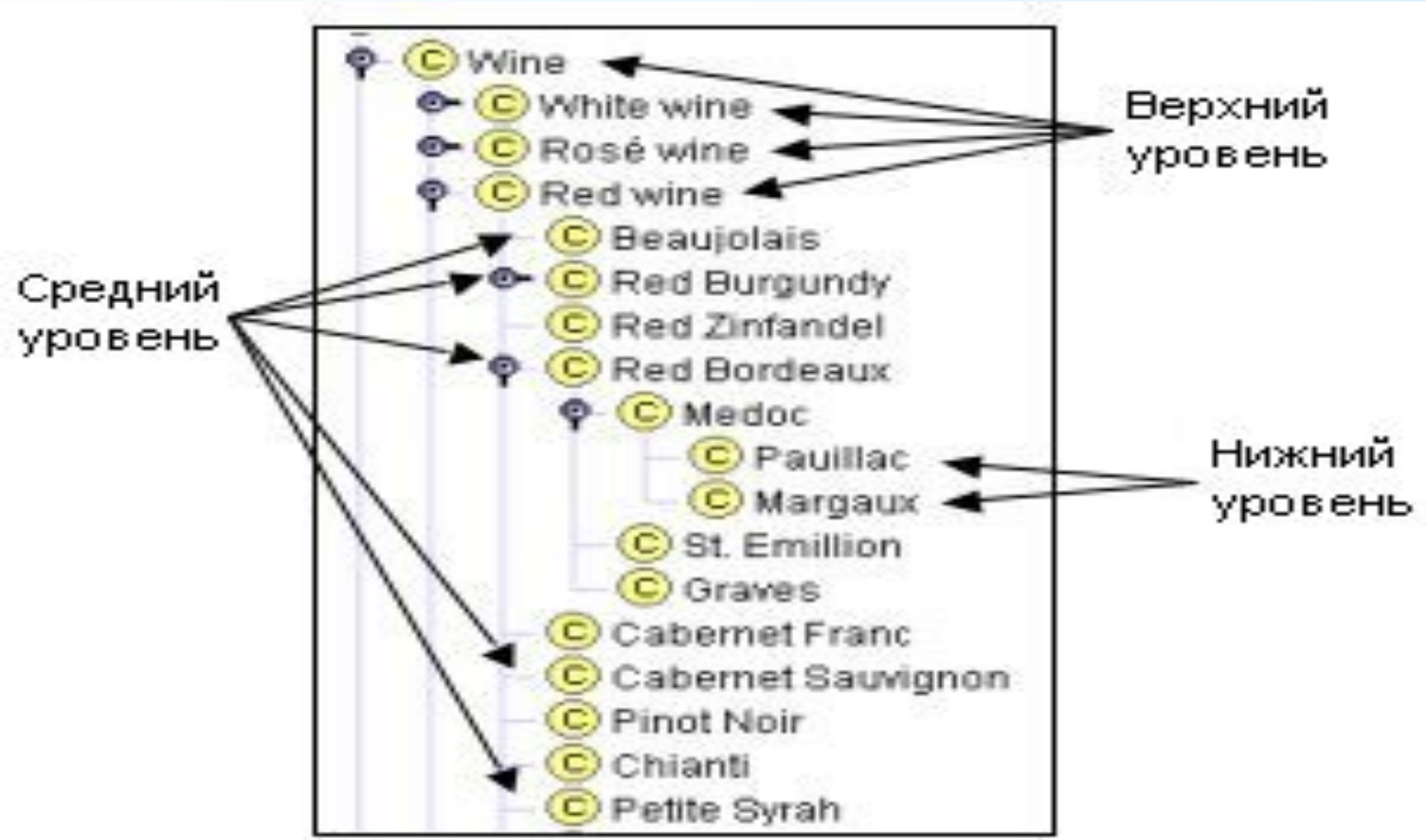
1) **Процесс нисходящей разработки начинается с определения самых общих понятий предметной области с последующей конкретизацией понятий. Например, мы можем начать с создания классов для общих понятий *Вино* и *Еда*. Затем мы конкретизируем класс *Вино*, создавая его подклассы: *Белое вино, Красное вино, Розовое вино*. Мы можем еще дальше категоризировать класс *Красное Вино*, например, в *Syrah, Red Burgundy, Cabernet Sauvignon* и т.д.**

2) **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*.**

3) **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. показано возможное деление на различные уровни обобщения.

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.

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 избрали, 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.

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. показана часть иерархии классов онтологии по винам.

Далее детально рассмотрено, что нужно искать при определении иерархии классов.

Слоты класса Вино и фацеты этих слотов

Значок “I” рядом со слотом производитель указывает, что у слота есть обратный слот.

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}

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 индивидными 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. 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*.

Слоты класса Вино и фацеты этих слотов

Значок “I” рядом со слотом производитель указывает, что у слота есть обратный слот.

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}

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

Фацет 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 будет 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.**

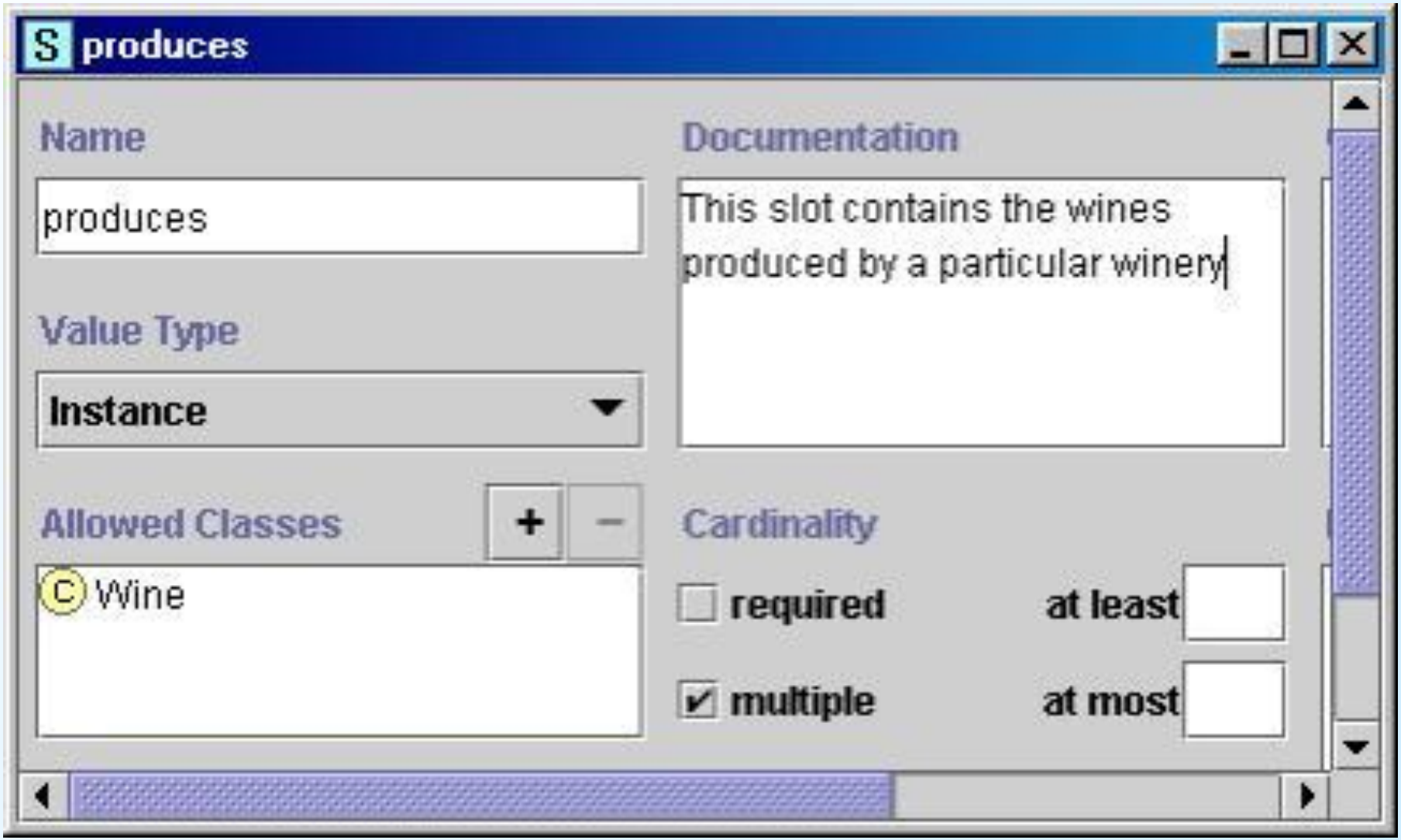
- **Нумерованные слоты определяют список конкретных разрешенных значений слота. Например, мы можем установить, что слот вкус может принять одно из трех возможных значений: сильный, умеренный и мягкий. В Protege-2000 нумерованные слоты имеют тип Символ.**

- **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.

На рис. показано определение слота производит класса Винный завод.

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. 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:

1. 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.

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

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

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.

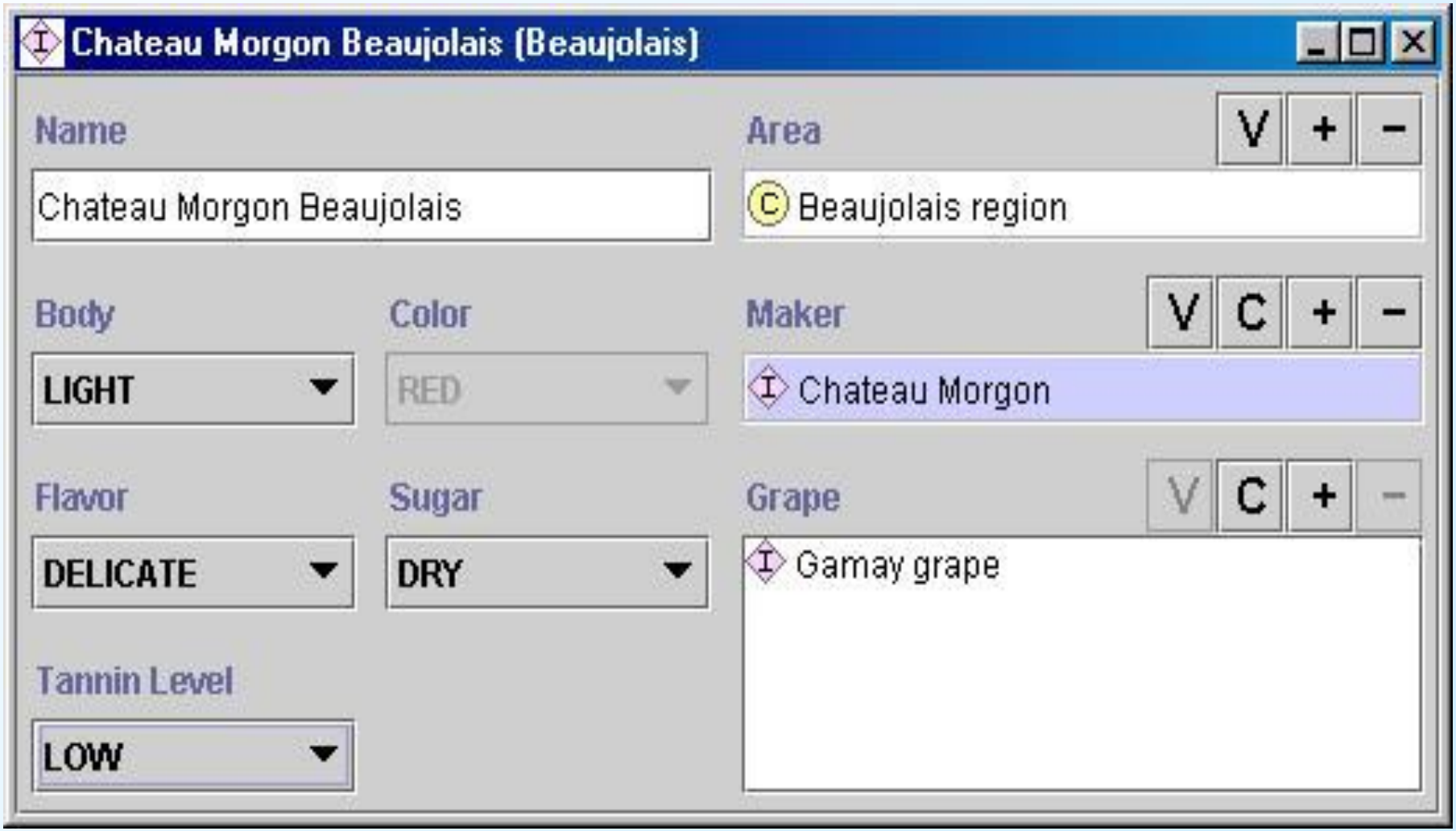
Например, мы можем создать отдельный экземпляр *Chateau-Morgon-Beaujolais* для представления определенного типа вина *Beaujolais*.

Chateau-Morgon-Beaujolais – это экземпляр класса Beaujolais, представляющего все вина Beaujolais.

У этого экземпляра определены следующие значения слотов:

- **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

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



СПАСИБО ЗА ВНИМАНИЕ,

ТВОРЧЕСКИХ УСПЕХОВ!