TUTORIAL ONTOLOGÍAS

Jesús Contreras
ISOCO
jcontreras@isoco.com

Juan Antonio Martínez Comeche
Universidad Complutense de Madrid
comeche@ccdoc.ucm.es

OBJETIVO DE LAS ONTOLOGÍAS: LA WEB SEMÁNTICA.

La web actual consiste esencialmente en un conjunto enorme de páginas que contienen texto no estructurado, es decir, texto cuyo contenido no nos hemos preocupado por caracterizar. Básicamente nos hemos limitado a reseñar la manera en que debe visualizarse dicho contenido, como lo demuestra la naturaleza de las etiquetas HTML. Esta simplicidad ha favorecido, sin duda, el éxito de la web actual y justifica su enorme crecimiento en número de páginas y usuarios, pero al tiempo acarrea problemas y dificultades a la hora de manejar y recuperar tal cantidad ingente de información.

Los seres humanos somos incapaces de controlar la información que en un momento dado puede sernos de utilidad en relación a una necesidad informativa entre los millones de páginas existentes en la web, máxime cuando los cambios en la misma se suceden a un ritmo vertiginoso. De hecho, se estima que un 40% de la red se modifica mensualmente. En tales circunstancias, hemos ideado buscadores que nos ayudan a decidir qué páginas pueden incluir información relevante ante un problema cualquiera. Pero dado que la información textual de los sitios web no está estructurada, en cuanto que no está descrita ni caracterizada de alguna forma, los algoritmos de los motores de búsqueda únicamente pueden basarse en la aparición de las palabras consideradas aisladamente.

Ello provoca, sin duda, falta de precisión y exhaustividad en los resultados obtenidos. Falta de precisión por cuanto en los resultados que nos presenta un buscador se hallan páginas que no tienen relación alguna con nuestra necesidad informativa. Eso sucede, por ejemplo, cuando las palabras poseen varios significados. Si consultamos por la palabra banco obtendremos páginas relativas a entidades bancarias, pero también a un tipo de asiento. De igual forma, la falta de exhaustividad puede venir provocada, entre otros motivos, por la utilización de un sinónimo en una página en lugar de la palabra empleada en la consulta. En tal caso, la página no será recuperada pues no contiene estrictamente la palabra introducida en la búsqueda.

Además, los buscadores proporcionan enlaces a documentos que pueden ser útiles para el usuario, pero no son capaces de proporcionar la respuesta concreta que busca en muchas ocasiones. Si una persona busca los coches más baratos entre los concesionarios de una zona geográfica concreta, hoy día el usuario debe ocupar muchas horas comparando la información de los distintos concesionarios que un buscador le ha facilitado previamente.

Otro problema de la web actual consiste en la falta de fiabilidad de las fuentes. El usuario no tiene elementos de juicio sobre la veracidad y confiabilidad de los datos presentes en los sitios web recuperados.

La evolución de la web diseñada por Tim Berners-Lee trata de solucionar los problemas planteados en los párrafos anteriores. Imaginemos por un momento una web donde el contenido de

las páginas está caracterizado y descrito de tal manera que sea capaz de discernir los distintos significados de las palabras, pueda deducir la existencia de relaciones de sinonimia entre palabras en cierto contexto temático, de manera que sea capaz de recuperar páginas útiles en relación a la necesidad informativa del usuario aunque en ellas no aparezcan las palabras introducidas expresamente en la consulta, o que fuese capaz de comparar datos e información procedentes de varias fuentes, efectuar inferencias o deducciones lógicas a partir de ellos para mostrarnos directamente la información que buscábamos (el concesionario cercano con los coches más baratos, por ejemplo). Incluso que fuese capaz de emitir juicios sobre la fiabilidad de los datos presentes en las diversas páginas y considerar en la respuesta exclusivamente los más veraces, desechando los menos confiables.

Tim Berners-Lee ha denominado Web Semántica a la web donde las aplicaciones serán capaces de efectuar un procesamiento de la información mucho más profundo. Esta web estará caracterizada por programas capaces de "comprender" el contenido de las páginas web, y por tanto, de relacionar la información contenida en páginas hoy aisladas, de procesarla, de discriminar la más fiable en un momento dado, e incluso de deducir o inferir información no registrada previamente, tomando decisiones con un cierto grado de autonomía.

Para que estas aplicaciones y servicios más "inteligentes" sean posibles es necesario que la información de las páginas web esté estructurada, esto es, perfectamente descrita y clasificada de manera que su significado exacto esté al alcance de las máquinas. De esta manera los ordenadores podrán manipular y procesar la información adecuadamente. De ahí la denominación de Web Semántica.

La manera que se ha ideado para codificar los significados de la información contenida en las páginas web consiste en el empleo de etiquetas que especifiquen el valor semántico o la interpretación correcta de los contenidos. Así, un número puede indicar, según las circunstancias, un precio, un año o una longitud. Su significado preciso en cada caso se especificará mediante la presencia de una etiqueta.

El marcado y anotación de los contenidos de la web debe realizarse siguiendo unas reglas y formatos comunes, pues de lo contrario sería imposible la manipulación efectiva de la información por parte de los ordenadores. En primer lugar, un marcado consistente implica la estructuración previa del dominio que se representa, detallando las entidades principales que lo componen, su jerarquía y la naturaleza de las relaciones existentes entre ellas. En segundo lugar, debe cuidarse que todos los usuarios empleemos formatos compatibles, pues si coexisten varios conjuntos de etiquetas y no se procura un método para garantizar su utilización conjunta, todos los esfuerzos serían inútiles.

El cumplimiento de ciertas normas necesarias para desarrollar de manera coherente el etiquetado de los contenidos web supone la creación de ontologías sobre el dominio o área de conocimiento que deseamos representar semánticamente. En consecuencia, las ontologías son el medio principal para lograr el objetivo de la web semántica, al facilitar la definición formal de las entidades y conceptos presentes en los diferentes dominios, la jerarquía que les sustenta y las diferentes relaciones que los unen entre sí. De esta manera garantizamos una representación formal legible por las máquinas, basado en un lenguaje común -XML- que puede ser compartido y utilizado por cualquier sistema de manera automática.

No menos importante que los retos tecnológicos y de formalismos se plantea el reto de la explotación y uso de la web semántica. Haciendo un símil con la web actual, que presenció su auge en cuanto se perfilaron nuevos modelos de negocio, se esbozan aquí algunas posibilidades o visiones sobre los tipos de aplicaciones en la web semántica.

La tecnología de la web semántica ofrece la posibilidad de construir contenido de manera formal y completa de acuerdo a modelos semánticos consensuados. La existencia de estos modelos permite que las funcionalidades ofrecidas por estos sistemas abarquen, entre otras, las siguientes aplicaciones:

- **Recuperación de información** mediante buscadores semánticos: las búsquedas semánticas, al contrario que las tradicionales -basadas en palabras clave-, trabajan con el significado de las palabras de acuerdo al modelo subyacente asegurando la precisión del 100% en las búsquedas. El resultado presentado al usuario pasa a ser la información solicitada en forma de conceptos del modelo, en lugar de los documentos posiblemente relacionados, tal como hacen los buscadores actuales.

- **Publicación de la información** de acuerdo al modelo. La navegación y la presentación de la información se podrá hacer de acuerdo a su contenido, de manera que el usuario puede visualizar los conceptos del modelo y consultar los conceptos relacionados independientemente de los documentos presentes en el sistema.

- La presencia del modelo permite la incorporación de **Interfaces inteligentes** como son los basados en lenguaje natural. La posibilidad de formular consultas en un lenguaje cercano al natural asegura la usabilidad del sistema final.

- **Sistema de inferencia y compleción de información**. En base a los axiomas de los modelos de la web semántica es posible validar y aumentar la información mediante sistemas de inferencia automáticos.

- **Intercambio de información** a formatos de aplicaciones específicas. La posibilidad de traducir la información a formatos de otras aplicaciones, como pueden ser aplicaciones educativas, permite aumentar la rentabilidad de la codificación de la misma. Actualmente el gasto de las empresas en hacer compatibles a sistemas heterogéneos supone un 30% del gasto de toda la industria de tecnologías de la información.

CONCEPTO DE ONTOLOGÍA

Aunque existen muy diversas definiciones del concepto de ontología, una de las más ampliamente aceptadas es la siguiente de Thomas Gruber:

> **"Una Ontología es una especificación formal y explícita de una conceptualización compartida":**

- El término "conceptualización" implica que toda ontología desarrolla un modelo abstracto del dominio o fenómeno del mundo que representa. Dicho modelo abstracto se basa esencialmente en el empleo de conceptos, atributos, valores y relaciones.

- Con "especificación explícita" se quiere expresar que una ontología supone la descripción y representación de un dominio concreto mediante conceptos, atributos, valores, relaciones, funciones, etc., definidas explícitamente. Las máquinas no pueden dar nada por supuesto o por obvio, y todo conocimiento -por básico que parezca, mientras sea necesario- debe ser explícitamente representado.

- El término “formal” alude al hecho de que cualquier representación (concepto, atributo, valor, etc.) ha de ser expresada en una ontología mediante un formalismo siempre idéntico, de manera que pueda ser reutilizada y leída por cualquier máquina independientemente del lugar o de la plataforma o idioma del sistema que lo emplee.

- Quizá el término más restrictivo e importante de los que figuran en la definición sea el de “compartida”. En efecto, una ontología lo es cuando dicha conceptualización y su representación formal y explícita ha sido favorablemente acogida por todos los usuarios de la misma. Ello permite por una parte distinguir claramente las ontologías de las bases de datos (en las que también puede hablarse de conceptualización y especificación formal y explícita mediante conceptos, atributos y valores, pero en la que el creador no tiene que lograr el consenso de nadie). Sin embargo, al mismo tiempo plantea una gran dificultad, pues en la práctica es imposible o casi imposible conseguir el consenso de todos los involucrados en un dominio específico (piénsese, por ejemplo, en una ontología sobre sanidad: ¿quién pone de acuerdo a médicos, pacientes, profesores, alumnos, gestores, etc., en la terminología, valores y relaciones en el dominio de la medicina?). De ahí que, en la práctica, se considere imposible desarrollar una ontología de carácter genérico o global, y sin embargo, se desarrollen ontologías en ámbitos mucho más restringidos, porque alcanzar aquí el consenso es factible (un banco para desarrollar servicios online para sus clientes, una empresa que desea una ontología sobre el conocimiento generado internamente por ella, etc.). Cuanto más genérico es el ámbito, en mayor medida el proceso hasta alcanzar el consenso se produce mediante una guerra de estándares inicial (varios organismos lanzan sus ontologías, esperando que cada una de ellas alcance el consenso de los demás), de las que surgen con el tiempo 2 ó 3 estándares de facto por área o sector, normalizándose finalmente hasta alcanzar una variante con el consenso de todos.

La parte formal del contenido está sujeta por el uso de ontologías, que otorgan la capacidad de comprensión a las aplicaciones o los agentges inteligentes. La disponibilidad, facilidad de gestión y divulgación de éstas es otro de los retos que se plantea. Las ontologías, consideradas como repositorios formales de conocimiento, siguen un ciclo de vida que modela desde su construcción, refinamiento, modificaciones, uso o explotación hasta su retiro. Alrededor de esta figura se sitúa el reto de la disponibilidad de las ontologías, que incluye la necesidad de metodologías de construcción, herramientas que las soporten, métodos de evaluación, comprobación y metodologías de evolución como gestión de cambios y de versiones, entre otros. Las ontologías no son formalismos cerrados y están sujetos a procesos evolutivos. Es por eso que metodologías y herramientas que soporten estos procesos se hacen esenciales. El proceso de desarrollo de ontologías debe tener el apoyo necesario tanto desde el punto de vista metodológico como de herramientas que lo faciliten.

Sin ser exhaustivos, podemos clasificar las ontologías, según el dominio y propósito que cubren, en los siguientes apartados:

- **Ontologías de nivel superior**: Permiten modelar los niveles altos de una realidad, ofreciendo conceptos genéricos para la clasificación de términos. Ejemplos de ontologías de estas características son CyC, WordNet, SUMO, etc.

- **Ontologías generales**: Algunos conceptos como el tiempo, el espacio, eventos, etc., pueden reutilizarse a través de diferentes dominios.

- **Ontologías de dominio**: Ontologías de dominios específicos o para aplicaciones concretas modelan las particularidades de las realidades de acuerdo a los propósitos de explotación impuestos.

La mayoría de las metodologías de construcción incluyen pasos que permiten adaptar y modificar ontologías ya existentes para construir otras más adecuadas a los propósitos del dominio modelado. En los escenarios de modelos de negocio previstos, las ontologías de dominios específicos las proveerán los actores interesados en su explotación, normalmente empresas líderes en un sector concreto. El reto se plantea en la disponibilidad de ontologías de propósito general que incluyan términos generales como tiempo, espacio, términos abstractos, etc.

PRINCIPALES VENTAJAS Y DESVENTAJAS DE LAS ONTOLOGÍAS

Hoy día la principal ventaja que aportan las ontologías, desde el punto de vista de las bases de datos y de la recuperación de información, tiene que ver con el empleo simultáneo de bases de datos de muy distinta naturaleza, características y formato a las que poder interrogar simultáneamente para recuperar la información buscada. Estas bases de datos pueden ser de contenido tan dispar como documentación textual poco estructurada (la clásica que forma parte de los buscadores en Internet), documentación fotográfica, documentación geográfica, museográfica, urbanística, espacios naturales, etc.

Otra de las grandes ventajas de las ontologías es la posibilidad de utilizar la información preexistente en dichas bases de datos a través de la propia ontología sin tener que renunciar a la base de datos original de partida, de manera que pueden convivir y seguir empleándose simultáneamente bases de datos iniciales y ontología. Tan solo debemos tener presente la necesidad de actualizar la ontología con los nuevos datos que vayan añadiéndose a las bases de datos. Tampoco es necesario introducir de nuevo manualmente -aunque podría hacerse- las bases de datos en la nueva estructura que hemos desarrollado con la ontología. Se puede automatizar el proceso de volcado de la información mediante el desarrollo informático de programas que transformen el formato de la base de datos inicial en el formato y lugar adecuado en la nueva estructura ontológica. Lógicamente, cuanto más estructurada se halle la información inicial, más sencilla es la transformación correspondiente.

Una de las mayores desventajas, en cambio, radica en la imposibilidad de utilización directa de las ontologías previamente desarrolladas. Es preciso crear programas de interrogación de la ontología, pues las herramientas existentes apenas permiten la visualización de los datos previamente introducidos. Eso exige la participación de especialistas en programación dentro de los equipos de desarrollo de ontologías si deseamos utilizarla posteriormente.

Otra de sus grandes desventajas surge al tratar de sacar el máximo partido de las amplísimas posibilidades de recuperación que proporcionan las ontologías, lo que se consigue mediante el desarrollo de buscadores denominados semánticos. Estos buscadores semánticos involucran la programación de complejos algoritmos que echan mano de herramientas de procesamiento de lenguaje natural poco desarrollados aún. En consecuencia, las ontologías posibilitarían una mejora sustancial en la precisión de la recuperación gracias a la posibilidad real de “entender” los términos y conceptos presentes en las preguntas, pero para ello dependen de complejos algoritmos de comprensión del lenguaje natural que no han alcanzado todavía un desarrollo suficiente. En resumen, suponen una herramienta con grandes posibilidades que, sin embargo, dependen para desarrollar todo su potencial de avances en otras áreas ajenas como la Inteligencia Artificial o el Procesamiento del Lenguaje Natural.

TERMINOLOGÍA BÁSICA EMPLEADA EN ONTOLOGÍA

En este apartado abordamos los principales conceptos que se manejan necesariamente al desarrollar una ontología. Todos ellos constituyen partes sustanciales en cualquier ontología, tanto en lo relativo a la estructura como al conjunto de datos.

- **Clase** (**Concepto)**: Descripción formal de una entidad del universo o dominio que se quiere representar. Constituye la pieza básica de estructuración del conocimiento. La decisión sobre qué considerar una clase del dominio no es fácil. Tal decisión debe ser tomada de acuerdo a los objetivos de la ontología (extraídos de las sesiones con los expertos, preguntas relevantes y de estándares existentes en el dominio). Una clase puede tener subclases que representan conceptos que son más específicos que dicha clase.
- **Clase abstracta:** Clase que no permite que existan instancias de ella. Se usa para agrupar conceptos, introducir cierto orden en la jerarquía, pero suelen ser demasiado generales para admitir instancias.
- **Instancia:** Representan objetos concretos del dominio, pertenecientes a una clase. La colección de instancias constituye la base de hechos (también denominada base de datos o base de conocimiento) del modelo.
- **Instancia indirecta:** Cuando una clase es instancia indirecta de otra, quiere decir que es instancia de alguna de sus clases derivadas. En contraposición a instancia directa, donde no existen clases intermedias.
- **Propiedad (Atributo, Slot):** Característica que permite describir más detalladamente la clase y sus instancias. Establece que la clase o concepto posee una propiedad que se concretará mediante un valor. Los valores de las propiedades o atributos pueden ser tipos básicos como cadenas de caracteres o números, pero también pueden ser otras clases o instancias (vid. Relación).
- **Faceta (Restricción sobre las propiedades):** Es alguna propiedad de la propiedad. Por ejemplo, la cardinalidad, si la propiedad es obligatoria o no, etc.
- **Relación:** Interacción o enlace entre los conceptos o clases del dominio que se modeliza. Algunas relaciones semánticas básicas son: subclase de, parte de, parte exhaustiva de, conectado a, es un, etc. Suelen configurar la taxonomía del dominio.Las relaciones más simples se modelizan mediante una propiedad de una clase cuyo valor es una instancia de otro concepto. Por ejemplo, dadas las clases AUTOR y OBRA, podemos definir la relación CREACIÓN como una propiedad (de cardinalidad múltiple) de la clase AUTOR cuyos valores sean instancias de la clase OBRA. Si las relaciones son más complejas (deben tener a su vez propiedades o deben organizarse en jerarquías, por ejemplo) se suelen definir mediante clases (conceptos) que las representen.
- **Axioma:** Regla que se añade a la ontología y que permite describir el comportamiento de los conceptos o clases. Se establecen a partir de valores específicos de las propiedades. Por ejemplo: “Para todo A que cumpla la condición C, entonces A es B”. Permiten dejar constancia de que ciertos valores de propiedades introducidos son coherentes con las restricciones de la ontología, o bien inferir posteriormente valores de atributos que no se han introducido explícitamente.De esta forma, a través de los axiomas es posible inferir conocimiento no codificado explícitamente en la ontología.
- **Anotación:** Es el proceso de relleno de instancias a partir de texto libre. Existen dos maneras de anotar texto: la más usual es la inclusión de etiquetas semánticas dentro del texto que se está procesando. Esto implica que el formato del texto debe ser editable y procesable. En el caso de no disponer de este formato, la segunda manera consiste en rellenar las instancias directamente en el modelo, dejando el texto original sin modificar.
- **Herencia:** Propiedad de la relación 'es_un' que permite que las clases relacionadas (heredadas) cuenten con los atributos de la clase con la cual se relacionan (clase padre).
- **Herencia múltiple**: Se da cuando una clase dada hereda o cuenta con las propiedades de dos

clases padre con las que establece dos relaciones del tipo 'es_un'.

- **Derivación:** Organización de las clases de la ontología en un árbol de jerarquía mediante sucesivas relaciones 'es_un' (también llamadas kind_of, is_a, o herencias) con la propiedad de herencia. Esta organización permite el encadenamiento sucesivo de herencias desde las clases de nivel superior a las clases situadas en niveles inferiores, llamadas clases derivadas.

METODOLOGÍAS PARA HACER ONTOLOGÍAS

Existen varias metodologías generales para el desarrollo de ontologías, con diverso grado de dificultad y especificación en su aplicación. Todas ellas exponen los procedimientos y las herramientas que pueden usarse para el desarrollo y validación de ontologías. Entre ellas se pueden destacar las siguientes:

- **Diligence**: Metodología de desarrollo de ontologías basada en la colaboración de múltiples participantes, de manera que la creación de una ontología se concibe como un proceso social, distribuido y muy poco controlado. Se hace hincapié en el consenso, facilitando un marco para la discusión de diferentes propuestas y el intercambio de argumentos a favor de las diferentes posibilidades hasta llegar a un acuerdo final. Se asume también que toda ontología debe evolucionar constantemente a lo largo del tiempo, de forma que permite introducir nuevas etapas en ellas, en las cuales los usuarios pueden introducir sucesivas modificaciones.

- **Competency Questions**: La metodología de las “Preguntas Relevantes” o “Preguntas de Verificación” es una de las más sencillas de aplicar. Consiste en determinar el dominio y el alcance de la ontología mediante la lista de preguntas que el sistema debería ser capaz de contestar. Las respuestas a estas preguntas (Grüninger;Fox, 1995) sugieren lo que podrían ser las instancias de la ontología, a partir de las cuales se deducirían (generalizando) las clases de la misma. Al mismo tiempo, estas preguntas servirán como factor decisivo a la hora de evaluar la propia ontología, permitiendo comprobar si se ha representado la suficiente información como para poder responder a dichas cuestiones, especialmente relevantes. Será la que empleemos aquí.

- **Methontology**: Desarrollado en la Universidad Politécnica de Madrid. Propone un ciclo de vida de construcción de la ontología basado en prototipos evolutivos, porque esto permite agregar, cambiar y removar términos en cada nueva versión (prototipo). Para cada prototipo, el proceso consta de los siguientes pasos esenciales:
  - Especificación: Consiste en delimitar los objetivos de su creación (compartir información entre personas o por agentes software; permitir la reutilización del conocimiento de un dominio; hacer explícitas las suposiciones que se efectúan en un dominio; separar el conocimiento del dominio del conocimiento sobre su fabricación o forma de operación; o simplemente analizar el conocimiento del dominio), decidir el dominio de actuación de la ontología (para no modelizar objetos poco relevantes en perjuicio de otros más importantes), quién la usará y para qué, las preguntas a las que deberá responder (ayuda al establecimiento de las dos anteriores), y quién se encargará de su mantenimiento (decidiendo si se limitará a introducir nuevas instancias, se permitirá la modificación de conceptos o atributos, etc.)
  - Conceptualización:Consiste en crear un glosario de términos que pertenecen al dominio, definirlos y crear una taxonomía (estableciendo una clasificación o jerarquía entre los conceptos, sus niveles, las relaciones entre ellos, sus instancias, sus propiedades o atributos, e igualmente los axiomas o reglas).
  - Formalización: Proceso consistente en convertir el modelo anterior en un modelo formal o semi computable.Se puede emplear en este paso una herramienta como Protégé.

    - Implementación: Convierte el modelo formalizado en un modelo computable mediante un lenguaje para construcción de ontologías.Se puede emplear en este paso una herramienta como Protégé.
    - Mantenimiento: Labor que puede acarrerar desde el borrado de instancias ya inútiles o la incorporación de nuevas instancias que se han ido produciendo con el tiempo, hasta las tareas de introducción de cambios en el contenido de la información, ya sea redefiniendo atributos, relaciones o incluso conceptos.

- **On-To-Knowledge**: Desarrollado por la Universidad de Karlsruhe, hace hincapié en las aplicaciones futuras de la ontología a la hora de diseñarla. Los pasos esenciales de que consta son:
    - Estudio de viabilidad: Previo al desarrollo de la ontología y base para el proceso siguiente.
    - Inicio: Proceso de delimitación del dominio y objetivo de la ontología, extracción de las fuentes de conocimiento (libros, revistas, documentos, etc.), junto con la descripción de sus usuarios y aplicaciones futuras.
    - Refinamiento: Producción de una ontología orientada a sus aplicaciones, conforme las especificaciones extraídas del proceso anterior.
    - Evaluación: Prueba de la utilidad de la ontología y del entorno de software asociado a ella en la aplicación para la que fue diseñada.
    - Mantenimiento: Decisión sobre el responsable de esta tarea y de los procesos posibles en ella.

PASOS PARA EL DESARROLLO DE UNA ONTOLOGÍA

Como hemos visto en un apartado anterior, existen diversas metodologías para el desarrollo de ontologías. Nosotros incidiremos aquí en una de las más sencillas de aplicar, basada en el empleo de las Competency Questions.

Sea cual sea el método empleado, ello no obsta para que consideremos el procedimiento de desarrollo de una ontología como un proceso iterativo, compuesto de aproximaciones sucesivas hasta llegar a una versión final de la misma. De este modo, inicialmente dispondremos de un primer borrador que debe ser revisado y refinado, debiendo ocuparnos por completar todos sus detalles.

Desde un comienzo conviene destacar algunas reglas fundamentales en el diseño de ontologías, a las que nos referiremos en muchas ocasiones. Estas tres reglas nos pueden ayudar, sin embargo, a la hora de tomar decisiones relativas al diseño de nuestra ontología:

- No existe un único modo correcto de modelizar un dominio; al contrario, siempre hay alternativas posibles. La mejor solución casi siempre depende del propósito u objetivo final de la ontología, o de las aplicaciones en las que vaya a emplearse.
- El desarrollo de una ontología es necesariamente un proceso iterativo.
- Los conceptos o clases de una ontología deberían ser muy cercanos a objetos o entes (tanto físicos como lógicos) y a las relaciones existentes en nuestro dominio de interés. Estos conceptos o clases serán muy probablemente nombres (objetos) o verbos (relaciones) que encontraremos en frases que describen nuestro dominio.

Esto es, el saber para qué se va a emplear la ontología y en qué medida va a ser general o detallada nos ayudará en muchas de las decisiones que debamos tomar a lo largo del proceso de desarrollo de la misma. Entre varias alternativas posibles, necesitaremos determinar cuál funcionará mejor en la tarea proyectada inicialmente para la ontología. De igual manera, tendremos que tener

presente siempre que una ontología es un modelo de la realidad y que los conceptos o clases de una ontología deben reflejar esta realidad. Después de obtener una versión inicial, podemos evaluarla y corregirla empleándola en la aplicación para la que fue proyectada o discutiendo sus características con expertos en el campo, o de ambas formas simultáneamente. Como resultado, casi con certeza absoluta necesitaremos revisar esta versión primera. Este proceso iterativo de diseño probablemente continuará durante el ciclo de vida completo de la ontología.

El desarrollo de una ontología mediante Competency Questions consta de los siguientes pasos esenciales:

- **1.- Determinar el dominio y alcance de la ontología, además del propósito u objetivo de la misma**, mediante las "Preguntas Relevantes" o "Preguntas de Verificación", esto es, mediante la lista de preguntas que el sistema debería ser capaz de contestar. Es habitual que estas Competency Questions se extraigan, al menos de manera parcial, de un corpus documental (artículos, libros, sitios web, opinión de expertos) representativo del dominio que se desea modelizar.

  En este apartado, mediante las Competency Questions, deberíamos responder las siguientes cuestiones básicas:
  - ¿Qué dominio cubrirá la ontología? Acotar bien el dominio nos permite decidir qué objetos o entes modelizar (son relevantes) y qué otros objetos no nos interesa representar. Por ejemplo, nos interesan los edificios como unidades físicas para gestionar posibles licencias municipales, mientras que no nos interesan sus ventanas.
  - ¿Para qué se va a emplear la ontología? De las funcionalidades de la aplicación final dependerá el punto de vista bajo el cual el diseñador deberá modelizar los conceptos de la realidad. Un mismo dominio se puede modelizar con clases o atributos distintos según sea el objetivo final de la ontología.
  - ¿Qué preguntas debería contestar la ontología? Permite establecer no solamente las cuestiones o el tipo de búsquedas que se efectuarán a la ontología cuando entre en funcionamiento, sino que ayuda en las dos cuestiones anteriores: (1) a delimitar el dominio tal como lo conciben los usuarios finales de la ontología; y (2) a considerar su punto de vista, importante a la hora de modelizar los conceptos de la realidad.
  - ¿Quién usará y mantendrá la ontología? Se debe saber de antemano si la persona encargada de mantener la ontología tiene nociones del dominio (si debe poder crear nuevas clases, relaciones o modificar la jerarquía) o solamente se limitará a introducir instancias.

- **2.- Considerar la reutilización de ontologías existentes**. Conviene comprobar si podemos partir de una o algunas ontologías existentes, refinando o extendiendo sus límites. En otras ocasiones, si nuestro sistema necesita interactuar con otras aplicaciones que emplean determinadas ontologías, la reutilización de tales ontologías es un requisito necesario en nuestro proyecto. Entre los repositorios de ontologías existentes, podemos destacar los siguientes:
  - Ontolingua: http://www.ksl.stanford.edu/software/ontolingua. Conjunto de herramientas que permiten el desarrollo, uso y modificación colaborativos de ontologías.
  - DAML Library: http://www.daml.org/ontologies. Listado de ontologías definidas en el lenguaje DAML, organizadas por diferentes criterios de búsqueda. Contiene unas 170 ontologías.
  - UNSPSC: http://www.unspsc.org. Modelos que permiten clasificar e identificar artículos de venta.
  - RosettaNet: http://www.rosettanet.org. Organización no lucrativa que promociona la utilización y creación de estándares de intercambio de información en el área del comercio electrónico.
  - DMOZ: http://www.dmoz.org. Directorio de contenido WEB.

- **3.- Enumerar términos importantes de la ontología**. Es útil hacer una lista con todos los términos que tienen relación con nuestro dominio. Estos términos pueden seleccionarse a partir del corpus documental (artículos, libros, sitios web, etc.) de donde se obtuvieron algunas de las Competency Questions, junto con la ayuda de los expertos en el dominio. Al principio no debemos preocuparnos por las posibles relaciones entre ellos, si varios se enmarcan en un mismo concepto, o si corresponden a conceptos o propiedades. Algunos de estos términos se transformarán en clases, otros en propiedades y otros en instancias.

- **4.- Definir las clases y la jerarquía de clases**. Este paso, junto al siguiente, son los más importantes en el proceso de desarrollo de una ontología. Existen diversas aproximaciones a la hora de definir la jerarquía de clases:

  - Top-down: comienza con la definición de los conceptos o clases más generales del dominio y posteriormente se lleva a cabo su especialización. Ejemplo: en una ontología sobre vinos, comenzamos con un concepto o clase general Vino, que posteriormente especializamos creando las siguientes subclases: Vino Tinto, Vino Blanco, Vino Rosado. A su vez, podemos categorizar la clase Vino Tinto en Rioja, Valdepeñas, etc.

  - Botom-up: comienza con la definición de los conceptos o clases más específicas, reagrupando posteriormente estas clases en conceptos más generales. Ejemplo: en una ontología sobre comidas, comenzamos con una clase Cordero y con otra clase Cerdo. Luego creamos una superclase común para ellas -Carne- que a su vez es subclase de Comida.

  - Combinación de las anteriores: comienza con la enumeración de los conceptos o clases más destacadas y posteriormente se generalizan y especializan apropiadamente.

    Ninguno de estos métodos es inherentemente mejor que los otros. La aproximación elegida depende en gran medida de la visión personal que el desarrollador tenga del dominio. Habitualmente se parte de la lista creada en el paso anterior, seleccionando aquellos términos que describan objetos con existencia independiente para constituir los conceptos o clases, en detrimento de aquellos términos que describan cómo son esos objetos.

    Sin duda, una de las decisiones más difíciles durante el proceso de modelización es decidir cuándo crear o introducir una nueva clase o cuándo representar una diferencia mediante distintos valores de propiedades.

    Algunas reglas generales que pueden ayudarnos a decidir cuándo introducir nuevas clases son las siguientes, advirtiendo claramente que se trata de una guía y no deben ser consideradas de obligado cumplimiento:

    - Generalmente las subclases de una clase presentan propiedades adicionales que la superclase no posee, o bien tienen restricciones o facetas diferentes de la superclase, o bien participan en relaciones distintas que la superclase. Por ejemplo, los vinos tintos pueden tener diferentes niveles de tanino, mientras que esta propiedad no se emplea para describir los vinos en general. En definitiva, habitualmente introduciremos una nueva clase en la jerarquía solamente cuando hay algo que podemos decir de esa clase que no podemos afirmar de la superclase. En la práctica, pues, cada subclase debería tener o nuevas propiedades o slots añadidos, o tener definidos nuevos valores para las propiedades, o anular algunas facetas de las propiedades heredadas.

    - Sin embargo, en ocasiones puede ser útil crear nuevas clases aunque no introduzcan

nuevas propiedades. De hecho, las clases en jerarquías terminológicas no tienen que introducir nuevas propiedades. Por ejemplo, una ontología médica puede incluir una clasificación de varias enfermedades. Esta clasificación podría ser una lista de términos sin propiedades (o con el mismo conjunto de propiedades). En tal caso, aún es útil organizar los términos en una jerarquía antes que introducirlos todos al mismo nivel en una lista. De esta manera (1) se facilita la exploración y navegación; y (2) se posibilita a un usuario la elección del nivel apropiado de generalidad de los términos. Otro motivo para introducir nuevas clases sin propiedad alguna nueva es modelizar conceptos entre los cuales los expertos del dominio establecen habitualmente una distinción, incluso aunque nosotros hubiésemos decidido no modelizar o representar la distinción en sí misma. Dado que empleamos las ontologías para facilitar la comunicación entre los expertos y entre éstos y los sistemas basados en conocimiento, podemos preferir reflejar en la ontología el punto de vista de los expertos en el dominio.

- En ocasiones la decisión se plantea entre una clase nueva o un valor distinto de una propiedad. Cuando modelizamos un dominio, a menudo tendremos que decidir si modelizar una distinción específica (como vino tinto, blanco y rosado) como un valor de una propiedad o como un conjunto de clases. De nuevo la decisión depende de los límites que hayamos impuesto al dominio de la ontología. ¿Qué importancia tiene el concepto Vino Blanco en nuestro dominio? La pregunta puede hacerse paralelamente de la siguiente manera: ¿Existen preguntas muy específicas en las que esté involucrada la distinción entre vino tinto, blanco y rosado, o por el contrario la distinción es marginal? Si el hecho de que los vinos sean blancos posee una importancia menor o no presenta implicaciones de cara a sus relaciones con otros objetos, en tal caso no deberíamos introducir una clase separada para los vinos blancos. Por ejemplo, para un modelo del dominio empleado en una fábrica de etiquetas para vinos la distinción no es muy importante. Por el contrario, para la representación de vinos, comidas y sus combinaciones apropiadas, un vino tinto es muy diferente de un vino blanco: combinan con diferentes comidas y tienen propiedades distintas. De igual forma, si una distinción es importante en el dominio y pensamos en los objetos con diferentes valores para ese factor como diferentes clases de objetos, entonces deberíamos crear una nueva clase para ese factor distintivo. Por ejemplo, se considera normalmente los vinos Red Merlots y White Merlots como realmente vinos diferentes más que como una única clase que engloba a los dos vinos Merlot. También debemos tener presente que números, colores y localizaciones suelen ser valores de propiedades y no suelen causar la creación de nuevas clases. Ello no implica que no existan excepciones. En este sentido, el color del vino es una excepción notable. Por ejemplo, consideremos una ontología sobre anatomía humana. A la hora de representar las costillas, ¿creamos una clase para cada costilla? ¿O creamos una clase costilla con un slot para el orden y su posición lateral? Si la información acerca de cada costilla es significativamente diferente, deberíamos crear una clase para cada costilla (imaginemos que queremos representar detalles sobre su localización y lo que está a su alrededor, así como funciones específicas u órganos que protege). Pero si estamos modelizando la anatomía a un nivel más alto de generalidad, todas las costillas son muy similares y presentan las mismas potenciales aplicaciones, de manera que convendría simplificar nuestra jerarquía y tener solamente una única clase Costilla con dos slots: posición lateral y orden.

- A veces surge la duda entre considerar un concepto particular como una clase o como una instancia particular. En general, esta decisión depende de las aplicaciones potenciales de la ontología, o lo que es lo mismo, depende de cuál sea el más bajo nivel de granularidad en la representación, pues el nivel de granularidad viene determinado por las potenciales aplicaciones de la ontología. En este caso, la solución viene dada por

las Competency Questions, pues los conceptos más específicos que constituyan las respuestas a estas preguntas son buenos candidatos para constituir las instancias de la base de conocimiento. Por ejemplo, si solamente vamos a emparejar vinos con comidas, no estaremos interesados en las botellas de vino desde un punto de vista físico. Por tanto, los tipos de vinos serían instancias en la base de conocimiento. En cambio, si la ontología se emplease para gestionar un inventario de vinos en un restaurante al tiempo que recopila emparejamientos vino-comida, las botellas individuales de cada vino podrían convertirse en instancias de la base de conocimiento. En cualquier caso, conviene tener siempre presente que las instancias individuales son los conceptos más específicos representados en una base de conocimiento. En otras ocasiones algunas instancias individuales pueden convertirse en un conjunto de clases. Supongamos las regiones vinícolas. Inicialmente, podemos definir como clases grandes regiones como Francia, Estados Unidos, Alemania, etc., y considerar las regiones más específicas dentro de las anteriores como instancias. Por ejemplo, Bourgogne sería una instancia de la clase Francia. Sin embargo, supongamos que también nos gustaría representar que Cotes d'Or es una región dentro de Bourgogne. En tal caso, ello nos obliga a considerar Bourgogne como clase y Cotes d'Or como una instancia de Bourgogne. Ese hecho provocaría que fuese muy difícil distinguir qué regiones son clases y cuáles son instancias. Por lo tanto, una posible decisión sería definir todas las regiones como clases. Protégé permite especificar algunas clases como Abstract, implicando que tales clases no tienen instancias directas. En definitiva, si los conceptos forman una jerarquía natural, podríamos representarlos como clases, aunque fuesen clases abstractas.

Por último, **resumimos las reglas enunciadas anteriormente en relación a las clases**:

- Una nueva subclase de una clase generalmente (1) debería tener nuevas propiedades que no posee la clase; o (2) debería tener diferentes valores para las propiedades que los de la clase; o (3) debería participar en diferentes relaciones que la clase.
- En jerarquías terminológicas, las nuevas clases no tienen por qué introducir nuevas propiedades.
- Si un factor es importante en el dominio y pensamos en los objetos con diferentes valores para ese factor como diferentes clases de objetos, entonces deberíamos crear una nueva clase o clases considerando dicho factor.
- Las instancias son los conceptos más específicos representados en una ontología.
- Si los conceptos de un dominio forman una jerarquía natural, podríamos representarlos como clases, aunque sean clases abstractas.

Posteriormente se organizan las clases en una taxonomía jerárquica. No existe una única jerarquía correcta para un dominio dado. La jerarquía depende de los posibles usos de la ontología, del nivel de detalle necesario para sus futuras aplicaciones, de las preferencias personales y, en ocasiones, de los requisitos de compatibilidad con otros sistemas con los que se va a emplear.

Sin embargo, cuando desarrollemos la jerarquía de clases pueden considerarse ciertas reglas generales y algunos errores habituales que nos ayudarán a tomar una decisión. Al igual que hicimos al presentar las reglas relativas a las clases, advertimos claramente que se trata de una guía y no deben ser consideradas de obligado cumplimiento:

- La jerarquía de clases representa una relación del tipo “is_a”. Por tanto, si una clase A es una superclase de la clase B, entonces toda instancia de B es también una instancia de A. De igual forma, una subclase B de una clase A representa un concepto que es un tipo especial o una subespecie dentro del concepto representado por la superclase.

- Un error frecuente en el modelado de la jerarquía de clases consiste en incluir tanto el singular como el plural de un concepto en la jerarquía, siendo más genérico el plural. Por ejemplo, es incorrecto definir una clase Vinos y una clase Vino como subclase de Vinos. Si se concibe la jerarquía de clases como una relación "tipo especial o subespecie de", este error resulta claro: un único vino no es una subespecie o tipo especial de Vinos. La mejor forma de evitar este tipo de error consiste en emplear siempre bien el singular o bien el plural al nombrar las clases o conceptos. Es más habitual, en este sentido, emplear el singular para la denominación de las clases.

- Transitividad de las relaciones jerárquicas. Si B es una subclase de A y C es una subclase de B, entonces C es una subclase de A. En ocasiones se distingue entre subclases directas y subclases indirectas. Una subclase directa es tal que no existe ninguna otra subclase entre ella y la superclase correspondiente. Una subclase indirecta es tal que existen otras subclases en la jerarquía entre ella y la superclase correspondiente.

- Es importante saber distinguir entre una clase y su denominación. En resumen, las clases representan conceptos del dominio, y no tienen por qué corresponder con el significado de las palabras que empleemos para designarlas. Ello implica que podemos cambiar la denominación de una clase, pero no la realidad objetiva del dominio que representa. En la práctica esta regla supone que no debemos emplear sinónimos para representar clases diferentes. De hecho, muchos sistemas permiten asociar a una clase una lista de sinónimos o traducciones.

- Deben evitarse los ciclos o bucles en la jerarquía de clases. Existe un ciclo o bucle en una jerarquía cuando una clase A tiene una subclase B y al mismo tiempo B es una superclase de A. Cuando suceda esto, habitualmente implica que las clases A y B son equivalentes: todas las instancias de A lo son de B, y todas las instancias de B lo son de A.

- Se denomina clases hermanas aquellas que son subclases directas de la misma clase. En relación a las clases hermanas conviene tener en cuenta que todos los conceptos hermanos en la jerarquía (excepto los que derivan directamente de la raíz) deben presentar el mismo nivel de generalidad. Por ejemplo, Vino Blanco y Ribeiro no deberían ser subclases directas de Vino, pues Vino Blanco es un concepto más general que Ribeiro. En cambio, los conceptos derivados de la raíz (representados a menudo como subclases directas de la clase Thing, la más general), representan secciones básicas del dominio y no tienen por qué representar conceptos parejos o del mismo nivel de generalidad.

- No existen reglas estrictas para el número de subclases que deben poseer las clases. Sin embargo, las clases de ontologías bien estructuradas tienen entre 2 y 12 subclases directas. Por tanto, si una clase tiene solamente una subclase directa, puede que la ontología no sea completa o exista un problema de modelado; y, simultáneamente, si una cierta clase tiene más de una docena de subclases directas puede ser conveniente crear categorías intermedias. Por ejemplo, podríamos listar todos los tipos de vinos como subclases directas de la clase Vino, pero si creamos las categorías intermedias de Vino Tinto, Vino Blanco y Vino Rosado, reflejamos mejor el modelo conceptual del dominio de los vinos que poseen la mayoría de las personas. Sin embargo, si no existen clases en las que se agrupen los conceptos de forma natural, tampoco hay necesidad de crear clases intermedias artificiales: es preferible dejar las clases con esa larga lista de subclases hermanas. Después de todo, la ontología debe ser un reflejo del mundo real: si

no existe una categorización en el mundo real, la ontología debe reflejar este hecho.

- Herencia múltiple. La mayoría de los sistemas de representación del conocimiento permiten la herencia múltiple en la jerarquía de clases, esto es, una clase que es simultáneamente subclase de diversas clases, heredando todas las propiedades y sus facetas de ambas clases padre. Por ejemplo, supongamos que deseamos crear una clase separada denominada Vino de Postre. Podemos definir, entonces, el Vino de Oporto como subclase del Vino Tinto y del Vino de Postre. La clase Vino de Oporto heredará, por tanto, las propiedades y sus facetas correspondientes a ambas clases padre. En consecuencia, heredará el valor 'dulce' para la propiedad 'azúcar' del Vino de Postre y el valor del nivel de tanino y el valor del color correspondientes a las propiedades de la clase Vino Tinto.

- Limitando el alcance. La siguiente regla puede ser muy útil a la hora de decidir hasta qué punto desarrollar la jerarquía de clases: La ontología no debería contener toda la información posible sobre el dominio; no es necesario especializar o generalizar más de lo que necesitemos para la aplicación. En el ejemplo del vino y la comida no necesitamos, por tanto, conocer qué papel se ha empleado en las etiquetas o cómo se cocinan las gambas. Dicho de otro modo, no incluiremos en nuestra ontología todas las propiedades posibles que pudiesen tener los vinos y las comidas, sino únicamente las que nos afectan para representar las interconexiones entre los términos que hemos considerado pertinentes.

- Subclases disjuntas. Muchos sistemas nos permiten especificar explícitamente que existen clases disjuntas, esto es, aquéllas que no pueden tener ninguna instancia en común. Por ejemplo, las clases Vino de Postre y Vino Blanco no son clases disjuntas pues hay muchos vinos que son instancias de ambas clases. En cambio, las clases Vino Tinto y Vino Blanco son clases disjuntas porque no hay ninguna instancia que lo sea simultáneamente de ambas clases. En ocasiones, el declarar ciertas clases como disjuntas permite evaluar mejor la ontología, pues el propio sistema indicaría como error de modelización una clase que fuese simultáneamente subclase indirecta de ambas.

Por último, **resumimos las reglas enunciadas anteriormente en relación a la jerarquía de clases**:

- Si una clase A es una superclase de la clase B, entonces toda instancia de B es también una instancia de A.
- Una subclase de una clase representa un concepto que es un tipo especial o una subespecie dentro del concepto representado por la clase.
- Si B es una subclase de A y C es una subclase de B, entonces C es una subclase de A.
- No debemos emplear sinónimos de un mismo concepto para representar clases diferentes, sino que los sinónimos deben considerarse denominaciones diferentes para un mismo y único concepto.
- No deben aparecer ciclos o bucles en la jerarquía de clases.
- Los conceptos de un mismo nivel de la jerarquía o clases hermanas (excepto los que derivan directamente de la raíz) deben presentar el mismo nivel de generalidad.
- Cada clase debería tener entre 2 y 12 subclases directas.
- La ontología no debería contener toda la información posible sobre el dominio: no es necesario especializar o generalizar más de lo que necesitemos para la aplicación.

- **5.- Definir las propiedades/slots de las clases**. Este paso, junto al anterior, son los más importantes en el proceso de desarrollo de una ontología. Una vez que hemos definido algunas clases, pasamos a describir sus características y estructura. Como ya habíamos seleccionado las

clases de entre la lista de términos, la mayoría de los términos que quedan son probablemente propiedades de dichas clases. Ejemplo: en una ontología sobre vinos, el color, el cuerpo, el contenido de azúcar o la localización de una bodega corresponden a propiedades.

Debemos determinar qué clase describe cada propiedad de la lista. Así, el color, el cuerpo y el azúcar serían propiedades de la clase Vino, mientras que la clase Bodega tendría una propiedad localización.

En general, las características de los objetos que pueden convertirse en propiedades o slots en una ontología son las siguientes:

- Características intrínsecas, como el sabor de un vino.
- Características extrínsecas, como el nombre de un vino.
- Partes del objeto, si el concepto o la clase está estructurado, tanto físicamente como de manera abstracta o convencional. Por ejemplo, en la clase Vino, el azúcar podría ser una propiedad, en cuanto que es uno de sus componentes; si Comida fuese una clase, el primer plato, el segundo plato y el postre podrían ser asimismo propiedades o slots.
- Relaciones con otros conceptos o clases. Por ejemplo, 'fabricante' de un cierto vino representa una relación entre la clase Vino y la clase Bodega.

  Todas las subclases de una clase heredan las propiedades de dicha clase. Por ejemplo, todos los slots de la clase Vino serán heredados por todas las subclases de Vino, Vino Tinto y Vino Blanco entre ellas. Podemos añadir una propiedad adicional, el nivel de tanino (bajo, moderado o alto) a la clase Vino Tinto. En tal caso, el slot 'nivel de tanino' será heredado por todas las subclases de Vino Tinto (como Valdepeñas o Rioja). Por tanto, una propiedad debería ser adscrita a la clase más general que posea dicha propiedad. Por ejemplo, cuerpo y color del vino deberían adscribirse a la clase Vino, pues es la clase más general cuyas instancias tendrán cuerpo y color.

  Debemos ser conscientes de que el valor de una propiedad puede depender de otra propiedad. Por ejemplo, si un vino fue fabricado por una bodega, entonces la bodega produce ese vino. Estas dos relaciones, 'fabricante' y 'produce' se denominan propiedades o relaciones inversas. En tales casos, almacenar la información en ambas direcciones es redundante, pues el sistema puede inferir el valor de la relación en un sentido cuando dispone del valor de la relación en el otro sentido. Sin embargo, desde el punto de vista de la adquisición del conocimiento, es conveniente a veces disponer de ambas piezas de información disponibles explícitamente. Es el caso en que, por los datos disponibles, se pueda rellenar en unos casos directamente en un sentido y en otros casos en el sentido contrario. En tales circunstancias, el sistema suele completar automáticamente el valor complementario sea cual sea el que se rellene inicialmente para garantizar la consistencia de la base de conocimiento. Así, si en nuestro ejemplo tuviésemos un par de propiedades inversas (la propiedad 'fabricante' de la clase Vino y la propiedad 'produce' de la clase Bodega), y el usuario crease una instancia de la clase Vino y completase un valor en la propiedad 'fabricante', el sistema podría añadir automáticamente una instancia de Bodega con ese mismo nombre y añadir también una instancia al slot 'produce' con la instancia de la clase Vino inicial.

  Muchos sistemas permiten la especificación de valores por defecto para las propiedades o slots. Si la mayoría de las instancias de una clase poseen un valor concreto de cierta propiedad, podemos definir este valor como valor por defecto de la propiedad o slot. Así, cuando se cree cada nueva instancia de una clase con esta propiedad, el sistema completará automáticamente la propiedad con el valor por defecto. Posteriormente podremos cambiar el valor a cualquier otro permitido, si no posee el valor por defecto.

Adviértase que esta característica es diferente y no interfiere con los valores de las propiedades. Los valores de las propiedades no pueden ser modificados. Por ejemplo, podemos decir que la propiedad 'nivel de azúcar' de la clase Vino de Postre tiene el valor DULCE. En consecuencia, todas las subclases e instancias de la clase Vino de Postre tendrán el valor DULCE para el slot 'nivel de azúcar'. Este valor no puede ser modificado en ninguna de las subclases ni en ninguna de las instancias de la clase.

Por último, **resumimos las reglas enunciadas anteriormente en relación a las propiedades o slots**:

- Todas las subclases de una clase heredan las propiedades de dicha clase, por lo que una propiedad debería ser adscrita a la clase más general que posea dicha propiedad.
- Garantizar la consistencia de la base de conocimiento cuando existan propiedades o relaciones inversas en una ontología.

- **6.- Definir las facetas o restricciones de las propiedades o slots**. Las propiedades pueden poseer diferentes facetas que describan o caractericen el tipo de valores que posee una cierta propiedad, los valores permitidos, el número de valores posible (cardinalidad), entre otras. Por ejemplo, podemos decir que la propiedad 'nombre' (como en 'nombre de un vino') presenta un tipo de valor que es una cadena. Ello quiere decir que los valores de la propiedad 'nombre' serán siempre un conjunto encadenado de caracteres como 'rioja', 'valdepeñas', etc.; o que la propiedad 'produce' (como en 'tal bodega produce estos vinos') puede tener múltiples valores y que dichos valores serán siempre instancias de la clase Vino. Ello quiere decir que el slot 'produce' presenta un tipo de valor Instancia, con Vino como clase permitida.

  Algunas de las facetas o restricciones más comunes son las siguientes:

  - **Cardinalidad**. Establece cuántos valores puede tener una propiedad o slot. Algunos sistemas distinguen únicamente entre cardinalidad simple (como máximo un valor) y cardinalidad múltiple (se permiten cualquier número de valores). Ejemplo: el cuerpo de un vino será una propiedad de cardinalidad simple, pues un vino solamente puede tener un cuerpo; los vinos producidos por cierta bodega supone un slot 'produce' de cardinalidad múltiple.

  - **Tipo de valor**. Describe qué tipo de valores puede poseer una propiedad. Los más frecuentes son:
    - String [Cadena]. Es el tipo de valor más simple, empleado en propiedades como 'nombre'. Indica que su valor es un conjunto de caracteres.
    - Número. Describe propiedades con valores numéricos. Por ejemplo, el precio de un vino es un número. En ocasiones se utilizan tipos de valor más específicos, como Entero o Float (decimal).
    - Boolean. Son propiedades cuyos valores son SI o NO. Ejemplo: si decidimos no representar los vinos espumosos como una clase independiente, podemos representar esta característica con una propiedad 'espumoso' cuyo valor sea booleano: si el valor es true (SI) indica que el vino es espumoso y si el valor es false (NO) el vino no es espumoso.
    - Enumerado. Las propiedades enumeradas especifican una lista de los valores permitidos para el slot. Por ejemplo, podemos especificar que la propiedad 'sabor' tenga uno de los tres siguientes valores: fuerte, moderado y suave. En Protégé los slots enumerados son de tipo Symbol.
    - Instancia. Las propiedades tipo Instancia permiten definir relaciones entre clases. Los slots con tipo de valor Instancia deben especificar la lista de clases permitidas de las

cuales pueden proceder las instancias que componen la relación. Por ejemplo, la propiedad 'produce' de la clase Bodega puede tener como valores las instancias de la clase Vino.

- **Dominio y Rango** de una propiedad o slot. Se suele denominar Rango de una propiedad a las clases permitidas para una propiedad de tipo Instancia. Así, el rango de la propiedad 'produce' sería la clase Vino. Se suele denominar Dominio de una propiedad al conjunto de clases que describe o caracteriza dicha propiedad. Así, el dominio de la propiedad 'produce' sería la clase Bodega.

  Las reglas básicas para determinar el Dominio y el Rango de un slot son las siguientes:

  - Cuando se definan el Dominio o el Rango de una propiedad, emplee la clase o clases más generales que puedan ser respectivamente Dominio o Rango de dicha propiedad.

  - No defina nunca un Dominio o un Rango que sea excesivamente general: todas las clases del dominio de una propiedad deben ser caracterizadas o descritas por dicha propiedad, mientras que todas las instancias de todas las clases del rango de una propiedad son valores permitidos para dicha propiedad de tipo Instancia.

    De lo anterior se deduce que en lugar de listar todas las subclases posibles de la clase Vino para el rango de la propiedad 'produce', se debe emplear exclusivamente la clase Vino. Al mismo tiempo, no debemos especificar que el rango de una propiedad es la clase THING, la más general de una ontología. Dicho en términos más específicos:

    - Si una lista de clases que definan el rango o el dominio de una propiedad incluye una clase y su subclase, elimina la subclase. Por ejemplo, si el rango de un slot o propiedad contiene tanto la clase Vino como la clase Vino Tinto, podemos eliminar Vino Tinto del rango porque no añade ninguna información nueva: Vino Tinto es una subclase de Vino y, por tanto, el rango Vino incluye implícitamente tanto la subclase Vino Tinto como cualquier otra subclase de la clase Vino.
    - Si una lista de clases que definan el rango o el dominio de una propiedad contiene todas las subclases de una clase A, pero no la clase A misma, el rango debería contener únicamente la clase A y no sus subclases. Por ejemplo, en lugar de definir el rango de una propiedad mediante Vino Tinto, Vino Blanco y Vino Rosado (enumerando todas las subclases de Vino), debemos definir el rango mediante la clase Vino.
    - Si una lista de clases que definan el rango o el dominio de una propiedad contiene todas las subclases de una clase A excepto una o algunas de ellas, considere la posibilidad de que la clase A defina mejor o más apropiadamente el rango de la propiedad.

- **7.- Crear instancias**. El último paso consiste en crear las instancias individuales de las clases en la jerarquía. La definición de una instancia individual de una clase exige: a) elegir una clase; b) crear una instancia individual para esa clase; c) rellenar los valores de las propiedades. Por ejemplo, una instancia de vino de Rioja (Vino-Vino Tinto- Rioja) podría ser Marqués de Riscal. Esta instancia podría tener definidos los siguientes valores:
  - cuerpo: Poco
  - color: Rojo
  - sabor: Suave
  - nivel de tanino: Bajo

- uva: Tempranillo
- fabricante: Marqués de Riscal (instancia de la clase Bodega)
- localización: Rioja Alta (instancia de la clase Región)
- nivel de azúcar: Seco

Como se apuntó en su momento, la decisión sobre qué considerar clases y cuándo introducir las instancias es una decisión sobre el nivel de granularidad o grado de especificidad de la representación que deseemos introducir en la ontología. Entonces se advirtió que la solución a esta decisión viene impuesta por la Competency Questions, pues los conceptos más específicos que constituyan las respuestas a estas preguntas relevantes constituyen las instancias o individuos de nuestro dominio.

HERRAMIENTAS DE EDICIÓN DE ONTOLOGÍAS

Entre las diversas herramientas informáticas empleadas actualmente en el desarrollo de ontologías, destacaríamos las siguientes:

- Protégé: Desarrollada por el grupo SMI (Stanford Medical Informatics) de la Universidad de Stanford. Se puede descargar libremente. Será la que emplearemos aquí.

- KAON: Desarrollada por la Universidad de Karlsruhe (el nombre de KAON procede de KArlsruhe Ontology). Es una herramienta de libre acceso que puede descargarse desde Sourceforge.

- WebOnto: Desarrollado por el Knowledge Media Institute (Kmi) de la Open University (Reino Unido). Se puede acceder libremente a ella, aunque los usuarios deben solicitar una cuenta de usuario a los administradores para disfrutar de todas sus capacidades de edición.

- OntoEdit: Desarrollado inicialmente por el Instituto AIFB en la Universidad de Karlsruhe, actualmente está comercializada por Ontoprise GmbH. Existe una versión libre y otra profesional. Esta última incluye muchas más funciones, estando considerada la más completa del mercado.

A estas herramientas más empleadas actualmente se pueden sumar otras como:

- Ontolingua: Desarrollada por el Knowledge Systems Laboratory (KSL) de la Universidad de Stanford, su principal objetivo es facilitar el desarrollo colaborativo de ontologías y proporcionar un repositorio de las mismas.

- OntoSaurus: Desarrollada por el Information Sciences Institute (ISI) de la Universidad de Southern California para la creación de ontologías con el lenguaje LOOM.

INSTALACIÓN DE PROTÉGÉ

Los pasos que deben seguirse para la instalación del programa Protégé son los siguientes:

- Ir a la página http://protege.stanford.edu/

- Hacer clic en “Downloads”, dentro del menú superior.

- Dentro de “registered Users”, cliquear en “download” page.

- Dentro del apartado Protégé 3.3.1, pinchar en “Download full Protégé”.

- Descargar la instalación recomendada para Tu Plataforma (Windows o Linux, entre otras). Nosotros expondremos aquí más en detalle el proceso completo de instalación para dos plataformas: 1) La plataforma Windows, incluyendo Java VM “Download (86.9M)” ; 2) La plataforma Linux, incluyendo Java VM “Download(110.3M)”[Tarda algún tiempo, dependiendo de la velocidad de la conexión a internet que tengamos].

- INSTALACIÓN EN WINDOWS

    - Una vez descargado, hacer doble clic en el archivo “install_protege.exe”
    - Tras unos minutos en que se prepara la instalación, aparece una ventana de introducción con el mensaje “This program will install Protégé...”. Cliquear en “Next”.
    - Aparece la ventana de información importante. Cliquear en “Next”.
    - En la ventana para la elección de componentes está seleccionado por defecto (en color azul) la opción “Everything”. Si no lo está, seleccionarlo y cliquear en “Next”.
    - En la ventana de elección de subdirectorio de instalación del programa, el valor por defecto suele ser adecuado para la inmensa mayoría de usuarios. A no ser que deseemos modificar dicho subdirectorio, basta con cliquear en “Next”.
    - En la ventana de elección de la máquina virtual Java está seleccionada por defecto la instalación de Java VM. Conviene respetar este valor, a no ser que deseemos emplear específicamente la versión instalada previamente en nuestro sistema. A continuación, cliquear en “Next”.
    - A continuación se nos muestra el resumen de las selecciones efectuadas previamente. Si estamos conformes, cliqueamos en “Install” y comenzará la instalación, que suele durar pocos minutos.
    - Una vez instalado el programa, aparece una ventana indicando que “The installation of Protégé is complete. Press “Done” to quit the installer”. Cliqueamos en “Done”.

- INSTALACIÓN EN LINUX

    - Una vez descargado, abrir una ventana de comandos y situarse en el directorio donde se haya descargado el programa mediante el comando “cd”
    - En ese directorio, teclear **sh ./install_protege.bin**. Aparece una ventana de introducción con el mensaje “This program will install Protégé...”. Cliquear en “Next”.
    - Aparece la ventana de información importante. Cliquear en “Next”.
    - En la ventana para la elección de componentes está seleccionado por defecto (en color azul) la opción “Everything”. Si no lo está, seleccionarlo y cliquear en “Next”.
    - En la ventana de elección de subdirectorio de instalación del programa, el valor por defecto suele ser adecuado para la inmensa mayoría de usuarios. A no ser que deseemos modificar dicho subdirectorio, basta con cliquear en “Next”.
    - En la ventana de elección de la máquina virtual Java está seleccionada por defecto la instalación de Java VM. Conviene respetar este valor, a no ser que deseemos emplear específicamente la versión instalada previamente en nuestro sistema. A continuación, cliquear en “Next”.
    - A continuación se nos muestra el resumen de las selecciones efectuadas previamente. Si

estamos conformes, cliqueamos en “Install” y comenzará la instalación, que suele durar pocos minutos.

- Una vez instalado el programa, aparece una ventana indicando que “The installation of Protégé is complete. Press “Done” to quit the installer”. Cliqueamos en “Done”.

ARRANQUE/FINALIZACIÓN DEL PROGRAMA EN WINDOWS

- Para arrancar el programa, ir sucesivamente a “Inicio”, “Programas”, “Protege_3.3.1” y cliquear en “Protege”. Aparecerá el terminal de comandos que ejecutará una serie de acciones. Es importante no cerrar esta ventana para que el programa funcione correctamente. Si se desea, podemos minimizarla, pero nunca cerrarla.
- Para finalizar el programa, en el menú desplegable de “File” cliquear “Exit Protégé”.

ARRANQUE/FINALIZACIÓN DEL PROGRAMA EN LINUX

- Para arrancar el programa, abrir una ventana de comandos e ir, mediante el comando cd, al subdirectorio donde se instaló el programa, /home/...../Protege_3.3.1. En dicho subdirectorio teclear **sh ./Protege**. Se ejecutarán una serie de acciones. Es importante no cerrar esta ventana para que el programa funcione correctamente. Si se desea, podemos minimizarla, pero nunca cerrarla.
- Para finalizar el programa, en el menú desplegable de “File” cliquear “Exit Protégé”.

FUNCIONAMIENTO DEL PROGRAMA PROTÉGÉ

Comenzaremos el funcionamiento del programa con unas consideraciones acerca de la convención en las denominaciones empleada en Protégé. En esta herramienta no podemos tener simultáneamente una clase Vino y una propiedad Vino. Pero como Protégé distingue las mayúsculas de las minúsculas, podemos tener una clase Vino y un slot 'vino'. De este modo cumplimos, además, la convención habitual que capitaliza los nombres de las clases y emplea minúsculas para los nombres de las propiedades.

Cuando el nombre de un concepto contiene más de una palabra (Vino Tinto, p.ej.), Protégé no permite el empleo de espacios dentro de las denominaciones de clases ni de propiedades, aunque sí en los nombres de instancias. La convención más habitual, además, opta por no usar los espacios ni en las clases ni en las propiedades. En caso necesario, se puede emplear el subrayado para unir palabras o simplemente unirlas con la inicial en mayúsculas: VinoTinto o Vino_Tinto, en nuestro ejemplo. Sin embargo, es importante considerar aquí otros sistemas con los que el nuestro deba o pueda interactuar. Si los otros sistemas usan un cierto método, puede ser útil decantarse por él (empleo de guión bajo, p.ej.). Conviene advertir que, una vez elegido una cierta convención, debemos ser consistentes y emplearla en toda la ontología.

En cuanto al empleo del singular o del plural, el nombre de una clase representa una colección de objetos, por lo que algunos diseñadores prefieren emplear el plural para las clases. Sin embargo, en la práctica, se emplea más el singular. Nosotros emplearemos, pues, el singular, aunque ninguna opción es mejor que la otra. Debemos, eso sí, ser consistentes en dicho empleo a lo largo de toda la ontología. Además, actuando siempre de la misma manera detectaremos posibles

errores de modelizado (como crear una clase Vinos y una subclase suya Vino).

En ocasiones se sugiere el empleo de prefijos o sufijos convencionales para distinguir claramente los slots de las clases. Dos prácticas comunes al respecto son, bien añadir 'tiene-' [has- en inglés] o bien añadir el sufijo '-de' [-of en inglés]. Esta aproximación permite que con solo mirar un término podamos determinar de inmediato si el término es una clase o un slot. Pero si se emplea la capitalización para distinguir clases y propiedades, el uso de prefijos sería redundante.

Cómo empezar a crear una ontología

Una vez arrancado el programa, observaremos una ventana de bienvenida a Protégé: "Welcome to Protégé". En dicha ventana el sistema nos ofrece varias posibilidades: Open Recent, Open Other y New Project.

Como deseamos crear una ontología en lugar de abrir proyectos anteriores, elegimos New Project. Para ello, basta con pinchar en "New Project..."

Aparece entonces la ventana de creación de un Nuevo Proyecto, con la opción de elegir entre varios tipos de formato: Protégé Files, Experimental XML File, OWL/RDF Database, entre otros. Lo más habitual es seleccionar RDF Files. Al pinchar en "RDF Files" veremos que queda resaltado en azul. Después cliqueamos en "Finish".

Después de cliquear en "Finish" entraremos en la ventana inicial del programa, con el título: "<new> Protégé 3.3.1".

La ventana inicial del programa posee, en su parte superior, una barra de menú con las opciones: File, Edit, Project, Window y Help. Debajo, una barra de iconos con algunas de las acciones más habituales, disponibles también en la barra de menús desplegables: New Project, Open, Save Project, Cut, Copy, etc... A su vez, debajo de todo lo anterior, una serie de pestañas nos permiten acceder a las partes esenciales de una ontología: Clases, Slots o Propiedades, Forms, Instancias y Queries o Consultas.

Cómo crear una clase

Situados en la ventana inicial del programa, pinchamos en la pestaña "Classes". Observaremos que la ventana "Classes" está dividida, a su vez, en dos ventanas: Class Browser y Class Editor.

La ventana Class Browser nos informa de la jerarquía de clases existente en nuestra ontología. Inicialmente, una ontología vacía contiene una única clase denominada ":THING". La clase THING constituye la raíz de la jerarquía, e indica que todas las clases creadas por el diseñador son necesariamente subclases de THING. También aparece debajo de ella una subclase ":SYSTEM-CLASS", propia e interna del sistema. Cuando tengamos desarrollado parcialmente nuestra jerarquía o árbol, bastará seleccionar una clase cualquiera de la jerarquía en Class Browser para que a la derecha, en la ventana Class Editor, se nos muestre los datos correspondientes a dicha clase: su nombre, descripción y lista de atributos.

Para crear una clase, seguiremos estos pasos:

- Nos situamos con el cursor en la clase padre de la clase que deseamos crear. En el caso de

nuestra primera clase, pues, nos deberemos situar en la clase THING.

- Seleccionamos la clase padre en la jerarquía (THING en nuestro ejemplo) cliqueando en ella. Aparecerá resaltada y enmarcada.
- En la ventana de la jerarquía (Class Browser), encima de la clase THING, a la derecha, observaremos varios iconos. Pulsamos sobre “Create Class”, una especie de sol. Esto da lugar a la aparición de una nueva clase en el árbol en su lugar correspondiente, que aparecerá con su denominación interna, semejante a: “KB_800434_Class_0”.
- En la ventana Class Editor (a la derecha de la ventana con la jerarquía) podemos cambiar esta denominación interna conforme a las convenciones de nombrado expuestas al principio de este epígrafe sobre el funcionamiento del programa. Siguiendo con nuestro ejemplo sobre vinos, podemos denominar Vino (con mayúscula inicial , siguiendo la regla más habitual) a la nueva clase recién creada. Para ello, en “Name”, tecleamos “Vino” e introducimos Return. La clase Vino aparecerá en la ventana izquierda, bajo “:THING” [y :SYSTEM-CLASS]. Por defecto, el sistema crea la clase como concreta [“Concrete”], esto es, permitiendo futuras instancias de ella. Si deseamos crearla como abstracta [“Abstract”], basta desplegar el menú a la derecha del campo “Role”. En “Documentation” se puede añadir en qué artículo, libro o sitio web del corpus documental figura una alusión a dicho concepto. También se suele incluir aquí una descripción sobre la utilidad y propósito del concepto creado, junto con cualquier comentario considerado relevante en relación a la clase.

  Como ejercicio, el lector puede crear dos nuevas clases, hermanas de Vino, denominadas Bodega y Región, y cuatro clases: Vino Tinto, Vino Blanco, Vino Rosado y Vino de Postre, como subclases de Vino.A su vez, dentro de la clase Vino Tinto, crear las subclases Rioja y Valdepeñas.

  Como ejercicio, crear la clase Vino de Oporto como subclase de 2 clases simultáneamente (herencia múltiple): Vino Tinto y Vino de Postre. Para ello se crea como subclase de una de ellas (Vino Tinto, por ejemplo) y se añade en la sección superclases -debajo de la jerarquía- la otra clase de la que depende (Vino de Postre, siguiendo el ejemplo).
- Para eliminar una cierta clase, porque se ha cometido algún error o porque decidimos modificar la jerarquía, basta situarse sobre ella, seleccionarla, y pulsar sobre “Delete Class”, un icono situado a la derecha del sol, representado mediante un círculo tachado.

Cómo añadir propiedades o relaciones a una clase

Estando en la ventana “Classes” se pueden añadir propiedades o atributos a las clases. Basta tener seleccionada la clase a la que deseamos añadir una propiedad.

Para añadir una propiedad o slot a una clase, seguiremos estos pasos:

- En la pestaña “Classes”, seleccionamos en la jerarquía la clase a la que deseamos añadir una propiedad o slot. En nuestro ejemplo de vinos, supongamos que deseamos añadir la propiedad 'nombre' a la clase Vino. Cliqueamos, pues, sobre la clase Vino.
- En la ventana “Class Editor”, a la derecha de la jerarquía, bajo “Name”, “Role” y “Documentation”, observaremos una ventana “Template Slots”, donde podemos añadir propiedades o slots a la clase previamente seleccionada. Para ello, pulsamos sobre”Create Slot”, una especie de rectángulo azul con rayos. Aparecerá superpuesta la ventana correspondiente a la nueva propiedad, con la denominación interna en el título, semejante a: “KB_800434_Slot_5”.
- En la ventana de la nueva propiedad, incluimos en “Name” la denominación conforme a la convención elegida, en este caso 'nombre' en minúsculas, e introducimos Return.
- En “Value Type” introducimos el tipo de valor que tendrá este slot. En el caso de 'nombre', dejamos el valor por defecto, esto es, String o cadena de texto. Para indicar que solamente puede presentar varios valores predefinidos, elegir Symbol.Una vez elegido “Symbol” surge una

ventana justo debajo, “Allowed Values”, donde introduciremos los valores posibles de la propiedad.

- En “Cardinality” introducimos la cardinalidad de la propiedad. En el caso de 'nombre', no consideraremos la posibilidad de sinónimos en relación a los nombres de los vinos, por lo que señalaremos “1” como valor máximo. Para ello, en “at most” introducimos el número 1.
- Podemos añadir cualquier comentario importante en relación a la propiedad en “Documentation”. Una vez introducidos los valores pertinentes, cerramos la ventana. De regreso en la ventana de las clases, podremos observar que en el área “Template Slots” se ha añadido la propiedad 'nombre' con cardinalidad 'single' y tipo de valor 'String'. Si pinchamos en la pestaña “Slots” observaremos también que se ha añadido la nueva propiedad recién creada 'nombre' a las internas de Protégé que figuran siempre. Seleccionándola, podemos consultar los datos relativos a dicha propiedad: Name, Value Type, Cardinality, su dominio [“Domain”], etc.
- Las relaciones son un tipo especial de propiedad, caracterizada porque el tipo de valor que presenta es una Instancia [“Instance”]. Cuando se elige Instancia como tipo de valor, automáticamente se añade una nueva ventana de Clases Permitidas [“Allowed Classes”] donde se debe añadir la clase con la que se relaciona.

  Como ejercicio, el lector puede añadir a la clase THING la propiedad 'nombre'.

  Como ejercicio, el lector puede añadir a la clase Bodega la propiedad 'produce'. Se trata de una relación entre la clase Bodega y la clase Vino; por tanto, se trata de una propiedad con tipo de valor Instancia [“Instance”]. En la nueva ventana “Allowed Classes” que surge, se incluye la clase Vino con la que se relaciona. Se trata, además, de un slot de cardinalidad múltiple.

  Como ejercicio, el lector puede añadir a la clase Vino las propiedades color (String de cardinalidad 1 con posibles valores: rojo, blanco y rosado, definidos en la ventana “Allowed Values” tras indicar el valor de tipo Symbol), sabor (String de cardinalidad 1 con posibles valores: fuerte, moderado y suave), cuerpo (String de cardinalidad 1 o simple con posibles valores: poco, medio y mucho), uva (String de cardinalidad múltiple con posibles valores: albariño, cabernet sauvignon, cariñena, garnacha, malvasía y tempranillo), nivel de azúcar (String de cardinalidad simple con posibles valores: seco, normal, dulce), fabricante (es una relación entre la clase Vino y la clase Bodega; por tanto, se trata de una propiedad con tipo de valor Instancia [“Instance”] y en la nueva ventana “Allowed Classes” que surge, se incluye la clase Bodega con la que se relaciona; aprovechamos también para indicar aquí que posee como propiedad inversa a 'produce' de la clase Bodega. Para ello, en “Inverse Slot” añadimos 'produce') y localización (relación con la clase Región).

  Como ejercicio, el lector puede especificar que la clase Vino de Postre tendrá como valor único de la propiedad 'nivel de azúcar' -heredada de la clase Vino- el valor dulce. Para ello se seleccionará la propiedad nivel de azúcar, se selecciona “View Slot Overrides” y en la ventana de la propiedad, dentro de “Allowed Values”, se restringen los valores permitidos al valor dulce. Como en este caso el valor es único, podemos añadir este valor 'dulce' como valor por defecto para que lo introduzca automáticamente en todas las instancias de Vino de Postre.

  Como ejercicio, el lector puede especificar que la clase Vino Tinto posee una propiedad añadida, 'nivel de tanino' (con 3 valores posibles: alto, medio, bajo) a las heredadas de la clase Vino.

Cómo añadir una instancia a una clase

Estando en la ventana “Instances” se puede introducir una instancia de una clase. Basta tener seleccionada la clase en la que deseamos introducir la instancia.

Para introducir una instancia de una clase, seguiremos estos pasos:

- Estando en la pestaña Instancias, seleccionamos la clase en la que deseamos introducir la instancia.
- En la ventana “Instance Browser”, pinchamos en “Create Instance”, un rombo con rayos. Aparecerá debajo la nueva instancia con la denominación interna, semejante a: “KB_800434_Instance_21”.
- En la ventana “Instance Editor” aparecen las propiedades de la clase correspondiente. Si rellenamos la propiedad 'nombre', por ejemplo, y deseamos que la instancia aparezca con esa denominación en lugar de la interna, en la ventana “Instance Browser”, junto a “Create Instance” y “Delete Instance”, en el menú desplegable, elegimos “Set Display Slot” y la propiedad que deseamos aparezca para representar la instancia, 'nombre' en este caso.
  Como ejercicio, el lector puede instroducir la siguiente instancia de Vino de Rioja:
  cuerpo: poco
  color: rojo
  sabor: suave
  nivel de tanino: bajo
  uva: tempranillo
  fabricante: marqués de Riscal
  localización: rioja alta
  nivel de azúcar: seco

BIBLIOGRAFÍA

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. (2001). The Semantic Web. Scientific American Magazine, 2001, mayo.

GÓMEZ-PÉREZ, A.; FERNÁNDEZ-LÓPEZ, M.; CORCHO, O. (2004). Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. Springer.

GRUBER, T.R. (1995). Towards principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies, 1995, 43(4-5): 907-928. Disponible en: http://tomgruber.org/wrinting/onto-design.htm.

GRÜNINGER, M.; FOX, M.S. (1995). Methodology for the Design and Evaluation of Ontologies. En: Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95, Montreal.

NOY, Natalya F.; MCGUINNESS, Deborah L. Ontology Development 101: A Guide to Creating Your First Ontology. Disponible en: http://www-ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html