Ontologías y web semántica

Jordi Duran Cals
Jordi Conesa i Caralt
Robert Clarisó Viladrosa

PID_00199503

UOC
Universitat Oberta de Catalunya

Índice

Introducción

En sus años de vida, la World Wide Web (WWW), más conocida como la **web**, ya es una herramienta indispensable para la vida cotidiana de las personas, y ha llegado a ser el principal medio de comunicación mundial de información. Desde sus inicios, alrededor del año 1990 y fruto de la existencia de la red Internet, la web ha experimentado un crecimiento casi exponencial en términos de contenido.

Red de redes

Internet es un sistema de conexión de redes de computadoras que permite la comunicación distribuida por medio de un conjunto de protocolos estándar.

Este contenido no es solo utilizado por personas, sino que cada vez hay una mayor necesidad de que pueda ser consumido por mecanismos automáticos. Es en este contexto y por este motivo por lo que los avances tecnológicos enfocados al tratamiento automático del contenido son cada vez más necesarios y habituales. Para realizar tratamientos automáticos de datos y de información hay que identificar el significado de los datos de forma explícita: hay que ampliar la web con información semántica; convertirla en una web semántica.

Los buscadores web, como por ejemplo Google, Yahoo o Bing procesan automáticamente un gran volumen de contenido web y se benefician de tener acceso a información organizada (metadatos) del propio contenido para poderla clasificar y ordenar, además de ofrecer resultados más relevantes. Por ejemplo, si en una página web aparece el término *banco*, este puede tener diferentes significados: "establecimiento financiero" o "asiento", entre otros. Sería muy útil desambiguar el significado en el contexto dentro de la página web. Así, si una persona busca el término *banco* como "establecimiento financiero", no le tendrían que aparecer aquellas referencias en páginas web de bancos de jardín. Hace falta, pues, dotar a la web de mecanismos y estándares para distinguir los diferentes significados, la semántica, de los recursos.

Conseguir que la web semántica sea una realidad es un proyecto muy ambicioso. El World Wide Web Consortium (W3C), una comunidad internacional destinada al desarrollo de estándares para la web, es uno de los principales promotores y arquitectos.

El consorcio W3C

El consorcio W3C está dirigido actualmente por Tim Berners-Lee, creador de la WWW.

En este módulo presentaremos cuál es el estado actual de la web semántica, los estándares más relevantes y qué papel juega el campo de la representación del conocimiento y, en particular, las ontologías.

Objetivos

Con el estudio de este módulo didáctico alcanzaréis los objetivos siguientes:

1. Adquirir pautas para conocer lo que significa la web semántica y descubrir qué potencial tiene.

2. Conocer con más detalle la creación y gestión de ontologías.

3. Conocer de qué formatos estándares está formada la web semántica y cómo utilizarlos.

4. Reconocer un conjunto de recursos web y herramientas para trabajar con datos semánticos.

1. La evolución de la web

En el año 1989, la World Wide Web (WWW, o simplemente, la web) surge de una propuesta de Tim Berners-Lee para utilizar el hipertexto como mecanismo para intercambiar información. Actualmente, la web es el servicio más conocido y popular de la red Internet.

Lectura recomendada

Podéis leer más sobre la propuesta inicial del WWW de Tim Berners-Lee en el siguiente recurso web:

"WorldWideWeb: Proposal for a HyperText Project"

La web es una red de páginas escritas en hipertexto y conectadas entre sí por medio de enlaces. Estas páginas están alojadas en diferentes servidores conectados entre ellos y que utilizan un protocolo (Hypertext Transfer Protocol o HTTP), que permite descargar y consultar las páginas de hipertexto y los recursos que enlazan: imágenes, vídeo, audio, documentos, ...

Cuando los usuarios quieren consultar alguna página, utilizan programas llamados **navegadores web**[1] para visualizar las páginas de hipertexto y guiar la navegación a través de los diferentes enlaces. En este sentido, el hipertexto incluye diferentes características que definen cómo se tiene que presentar la información de una página.

(1)En inglés, *browsers*.

Aunque hoy en día la web está orientada principalmente a su uso por humanos, también hay programas especializados en visitar páginas web. Estos programas, denominados *bot*, *web spider* o web *crawler*, pueden descargar el contenido de las páginas web, ya sea para hacer copias de seguridad, calcular estadísticas o indexar contenidos para facilitar búsquedas, por ejemplo. Desgraciadamente, así como todas las páginas web incluyen información sobre la presentación de la información, muy pocas incorporan anotaciones semánticas que expliquen qué tipo de información contienen y cómo se tiene que interpretar. Algunos ejemplos podrían ser: esta es una página sobre un producto, esto es un precio, esto una valoración de un usuario, esto es el nombre del fabricante, etc. Precisamente, este es uno de los retos de la web semántica.

Para entender la evolución del contenido de la web, a continuación pasamos a exponer cuál ha sido la evolución en sus usos y en la tecnología subyacente.

Lectura recomendada

Podéis encontrar más información sobre los estándares web y su evolución en la obra siguiente:

"Introducción al currículo de estándares web/contenidos", *Mosaic*.

1.1. La web 1.0

Web 1.0 es el término con el que se suelen denominar las primeras páginas web, caracterizadas por ofrecer un conjunto de información estática, que constituyen un mecanismo de comunicación unidireccional. Es decir, el usuario que accede al contenido únicamente puede leerlo de manera pasiva, sin la posibilidad de contribuir en ningún caso a su ampliación. Por lo tanto, el usuario es un simple consumidor de una web enfocada a la lectura.

Tecnológicamente aparece el lenguaje de marcas HTML[2], lenguaje empleado para la creación de páginas web especialmente idóneo para el enlace de contenido web por medio de los hipervínculos[3]. También aparece un nuevo protocolo de Internet, el HTTP, y el sistema de localizador de recursos web URI.

(2)HTML es la sigla de Hyper Text Mark-Up Language.

(3)En inglés, *hyperlinks*.

Ejemplo 1. Páginas HTML estáticas

Un dentista quiere utilizar la web para potenciar su negocio y se publicita con una página web escrita con HTML para anunciar su consulta. En ella figura información como el nombre de la clínica, el teléfono y la dirección. En una web 1.0 tendría la siguiente representación:

```
<div>
   <strong>Clínica Dientes Blancos</strong>
   <ul>
      <li>Teléfono: 553443333</li>
      <li>Dirección: C/ Mayor, 5 - Barcelona</li>
   </ul>
</div>
```

1.2. La web 2.0

Casi una década más tarde, aparece el concepto de web 2.0, también conocida como web colaborativa. Se considera la evolución de la primera generación de la web, con el objetivo de permitir la interacción con el contenido, es decir, dotar las páginas de mecanismos para la colaboración de usuarios en la transformación y creación de nuevo contenido. Así, el usuario pasa a ser también productor, y se consigue una comunicación bidireccional con la web.

El término web 2.0

El significado del término web 2.0 es muy difuso y no está claramente definido. Tampoco hay ningún "salto" desde lo que se conoce como web 1.0, a la 2.0, sino una evolución lenta y progresiva.

Para que el usuario tome este nuevo rol, el de productor de contenidos, hace falta que tecnológicamente se permita la modificación de contenidos con el mínimo conocimiento tecnológico. Es por eso por lo que aparecen herramientas web tales como los blogs o las wikis, entre otros. Estamos, pues, ante la convergencia entre los medios de comunicación y el contenido, y además, en el contexto para la aparición de las primeras redes sociales, como theglobe.com, MySpace, YouTube, etc.

La Wikipedia

Las contribuciones del usuario son las que hacen crecer la web 2.0. El ejemplo más paradigmático es la Wikipedia, donde libremente se construye el conocimiento.

Por otro lado, en paralelo se avanza en tecnologías para el cliente: se consolida el apoyo a Javascript en los navegadores y las técnicas Ajax, con los diferentes entornos de trabajo[4] como Prototype y jQuery, y también el estándar de intercambio de datos JSON.

(4)En inglés, *frameworks*.

Ejemplo 2. Páginas HTML que permiten intercambio y colaboración

El dentista del ejemplo 1, aprovechando el paradigma de web 2.0, quiere mostrar en su web los mensajes que se publican en su línea de tiempo de Twitter, además de permitir a los usuarios que le envíen tuits. Para hacerlo, tiene que añadir un fragmento de código HTML y Javascript que carga esta información de manera dinámica en la página.

```
<a class="twitter-timeline" href=https://twitter.com/YourDentist
   data-widget-id="368186125186">
   Tweets de @YourDentist
</a>
<script>
   !function(d,s,id) {
      var js,fjs = d.getElementsByTagName(s)[0];
      if ( !d.getElementById(id) ) {
         js = d.createElement(s);
         js.id = id;
         js.src = "//platform.twitter.com/widgets.js";
         fjs.parentNode.insertBefore(js,fjs);
      }
   }
   (document, "script", "twitter-wjs");
</script>
```

1.3. La web semántica

La web semántica se considera el próximo paso en la evolución de la web, con el objetivo de poder ser manipulada de manera automática. Tiene el reto de transformar el extenso contenido, mayoritariamente pensado para ser consumido por humanos, en datos e información manipulable por programas.

Lectura recomendada

T. Berners-Lee; J. Hendler; O. Lassila (2001). "The Semantic Web". *Scientific American* (mayo, pág. 34-43).

Para conseguirlo, antes que nada hay que dar un significado a los recursos de la web. Es decir, se tiene que clasificar, estructurar y anotar semánticamente cada recurso para que pueda ser entendido por los programas que lo tendrán que procesar.

Cabe destacar que no estamos hablando de procesar automáticamente el lenguaje natural, tarea que sería muy compleja computacionalmente, sino de estructurar toda la información de manera útil.

Ved también

Hablaremos de URI, RDF, RDFa, RDF Schema y OWL más adelante en este módulo.

Esto no es posible con formatos como el lenguaje de marcas HTML, centrado únicamente en la presentación del contenido. Hacen falta pues nuevos formatos y tecnologías que sean capaces de describir de manera explícita y unívoca los recursos. Por lo tanto, habrá que utilizar un mecanismo para identificar recursos, el *Universal Resource Identifier* (URI), junto con nuevos lenguajes de marcas para añadir **metadatos** a las páginas web, como por ejemplo:

Metadatos

Los **metadatos** son datos estructurados que dan información sobre el resto de datos de un documento (origen, tipo, significado, fecha de creación, etc.).

- *eXtensible Hypertext Mark-up Language* (XHTML)
- *Resource Description Framework* (RDF)
- *Resource Description Framework in Attributes* (RDFa)
- *Resource Description Framework Schema* (RDF Schema)
- *Web Ontology Language* (OWL)

Ejemplo 3. HTML con RDFa

Nuestro dentista quiere favorecer su posicionamiento en los buscadores, y por eso introduce información semántica en su página web. Para conseguirlo, transforma la página HTML a un formato XHTML con marcas RDFa, que permiten, además de la lectura por parte de los humanos, su procesamiento automático:

```
<div xmlns:v="http://rdf.data-vocabulary.org/#"
   typeof="v:Organization">
   <strong>
      <span property="v:name">Clínica Dientes Blancos</span>
   </strong>
   <ul>
      <li>Teléfono:
         <span property="v:tel">553443333</span>
      </li>
      <li>Dirección:
         <span property="v:street-address">C/ Mayor, 5</span>
         <span property="v:locality">Barcelona</span>
      </li>
   </ul>
   <a href="http://ortodoncia.wordpress.com" rel="v:url">Blog</a>
</div>
```

Gracias al hecho de utilizar esta representación, un programa puede ser capaz de identificar la dirección y la población de la calle del dentista y mostrar esta información directamente sobre un mapa, o bien identificar que el número 553443333 es el teléfono del dentista y permitir enlazarlo con el programa de telefonía utilizado para permitir que el usuario pueda llamar al dentista directamente desde la página web.

A lo largo de este módulo, veremos con más detalle cómo funcionan estas tecnologías que ahora solo hemos mencionado (XHTML, RDF, OWL, ...) y qué uso se puede dar a las marcas que se han añadido en este ejemplo.

2. La web semántica

La web semántica[5] no debe verse como una nueva web, sino como una extensión de la actual: una extensión que complementa la web 2.0 al permitir a humanos y dispositivos trabajar juntos.

(5)En inglés, *semantic web*.

2.1. Visión inicial

Tim Berners-Lee y otros definen la web semántica como la web donde las aplicaciones serán capaces de entender el contenido de las páginas web, y por lo tanto, relacionar la información que contienen, favoreciendo la interrelación de recursos.

Así, el planteamiento inicialmente postula la web como una red de interconexión de datos a los que puede accederse fácilmente y que pueden ser comprendidos por cualquier aplicación.

Ejemplo 4. Aplicación de la web semántica: buscadores inteligentes

A través de un buscador web podríamos formular la búsqueda "quiero ir al concierto de Madonna de este próximo fin de semana". El buscador tendría que ser capaz de saber qué días constituyen el próximo fin de semana y en qué ciudades actúa Madonna en aquellas fechas, todo a través de información disponible en la web.

Conocida esta información, y suponiendo que hay un único concierto de Madonna y está en París, el buscador nos proporcionaría un listado de webs para comprar entradas. Podría, además, buscar automáticamente un listado de vuelos a París, puesto que ha identificado que estamos ubicados en la ciudad de Barcelona.

Además de datos, la web semántica está formada también por aplicaciones y tecnologías con capacidad para interactuar con la web y aprovechar la información semántica para ofrecer nuevos servicios. Hablamos de software como por ejemplo los **servicios web**[6] y los **agentes**.

(6)En inglés, *web services*.

El W3C define los servicios web como:

"[...] un sistema de software diseñado para permitir la interacción máquina-máquina sobre una red. Tiene una interfaz descrita en un formato procesable automáticamente (específicamente WSDL). Otros sistemas interactúan con el *web service* en la forma fijada por su descripción utilizando mensajes SOAP, típicamente enviados a través del protocolo HTTP con una serialización en formato XML en conjunción con otros estándares relacionados con la web."

"Web service", Wikipedia

WSDL y SOAP

El *Web Service Description Language* (WSDL) es un formato XML utilizado para describir la funcionalidad pública ofrecida por un servicio web.

El *Simple Object Access Protocol* (SOAP) es un protocolo de comunicación basado en XML y diseñado para el intercambio de información estructurada para acceder a servicios web de Internet.

Así pues, los servicios web no proporcionan una interfaz gráfica para usuarios humanos. En lugar de eso, ofrecen públicamente funcionalidades tales como lógicas de negocio, datos y procesos accesibles, que pueden ser llamadas directamente desde un programa de forma remota. Hará falta pues que los desarro-

lladores aprovechen estas funcionalidades llamando a los servicios desde su software y que, en caso de ofrecer el acceso al servicio de usuarios humanos, creen mecanismos amigables para utilizar estos servicios, como por ejemplo interfaces web o bien interfaces gráficas de usuario (GUI).

Por otro lado, los agentes son programas que, utilizando técnicas de inteligencia artificial, son capaces de percibir el entorno y actuar de manera autónoma para lograr unos objetivos. Para poder tener autonomía –el elemento clave en un agente– es necesario que el agente sea capaz de comprender la información de las páginas web y tenga mecanismos para realizar acciones, por ejemplo, mediante servicios web.

Ejemplo 5. Aplicación de la web semántica: agentes inteligentes

Siguiendo con el ejemplo anterior, en el momento de escoger un vuelo y comprar las entradas, un programa inteligente efectuaría también las reservas de hotel, localizaría el concierto y gestionaría el desplazamiento de manera automática.

Ejemplo 6. Un servicio web para la consulta y petición de reservas de vuelo

Reanudando los últimos ejemplos, para poder realizar la tarea de reserva de hoteles de manera autónoma y automática, el agente necesitará que estén disponibles un conjunto de servicios web. Concretamente, necesita disponer de (1) funciones para la consulta de disponibilidad y ubicación, y (2) poder finalmente tramitar la reserva en la opción que mejor optimice la distancia al concierto y el coste.

Por todos estos motivos, los datos y la información contenida en la web tienen que ser expresados de manera unívoca, es decir, de modo que su significado no lleve a confusión. Por el contrario, lo que nos encontramos ahora es una gran cantidad de información textual en lenguaje natural, que puede ser ambigua y no está estructurada. La manera de estructurarla pasa por representar el conocimiento que contiene por medio de etiquetas que indican cuál es la interpretación correcta de la información contenida en la página.

Ejemplo 7. Tipo de información en una página web semántica

En el caso del concierto de Madonna (ejemplo 4), un sistema semántico emplearía la información del país para desambiguar la ciudad del concierto. De este modo, podría descartar de manera automática que el concierto no tendrá lugar en Texas (donde hay un pueblo llamado Paris), ya que Texas no pertenece a Francia.

Esto sería posible porque el sistema conocería de antemano que las ciudades no se identifican por su nombre y que toda ciudad está ubicada en un país.

Por lo tanto, la tecnología web tiene que ser capaz de distinguir entre información comercial y académica, o datos pulidos y en bruto, e incluso entre consumo humano y tratamiento automático.

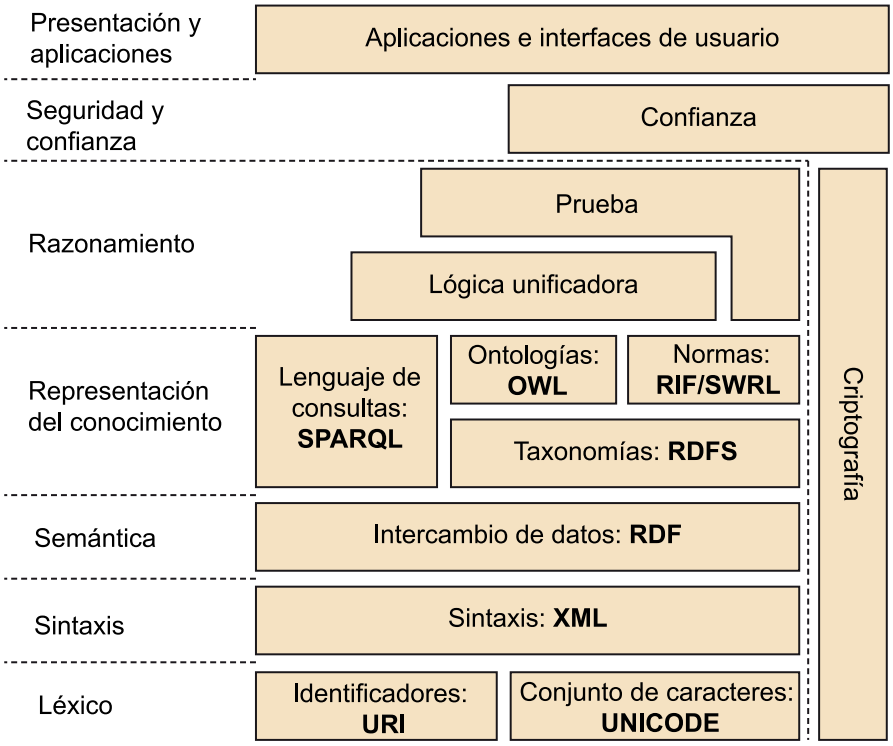
Con relación a la representación del conocimiento, la web semántica tiene el reto de proporcionar un lenguaje que exprese tanto los datos como las reglas mismas para el razonamiento sobre los datos.

2.2. Arquitectura

La web semántica se sostiene sobre un conjunto de tecnologías y estándares para representar la información y añadir anotaciones semánticas que indiquen cómo interpretarla. En este subapartado ofrecemos una panorámica general de estos estándares, que serán estudiados con más detalle en el resto del módulo.

Como todo sistema complejo, la web semántica se puede ver como una arquitectura de capas. Cada nivel resuelve un aspecto concreto del problema de incorporar y utilizar información semántica, utilizando las capas inferiores o siendo utilizado por las capas superiores. En la figura 1 tenemos una representación de esta jerarquía de capas, basada en la ilustración llamada *Semantic web stack* extraída de la presentación "Semantic Web - XML2000" de Tim Berners-Lee.

Figura 1. Arquitectura de la web semántica



Podemos ver que la arquitectura está organizada en una jerarquía de capas:

a) La capa inferior define el **léxico** de una página web semántica, es decir, cuáles son los símbolos que podemos utilizar para construirla. En este nivel consideraremos dos aspectos, qué caracteres se pueden utilizar y cómo se denominan los recursos:

- **Unicode** es un estándar de codificación de conjuntos de caracteres internacional, que incluye alfabetos como el latín, el cirílico, el chino o el árabe. Este estándar posibilita que todos los idiomas del mundo puedan ser utilizados para leer o escribir en la web de manera estándar.

- **Uniform Resource Identifier** (**URI**) es un formato estandarizado que por medio de una secuencia de caracteres ASCII identifica de manera única recursos, como pueden ser documentos, imágenes, etc. Un subconjunto de este formato, y el más conocido de todos, es el **Uniform Resource Locator** (**URL**), que tiene la finalidad de expresar un mecanismo de acceso y localización de documentos en la red, como por ejemplo: http://www.un.org. El uso de las URI es importante para sistemas distribuidos, puesto que proporciona una identificación inteligible de todos los recursos.

URI o IRI?

Una generalización del estándar URI es **Internationalized Resource Identifier** (IRI), que posibilita la utilización de caracteres Unicode en el identificador de recurso.

b) La capa siguiente nos especifica la sintaxis de una página web semántica, es decir, el modo como los símbolos definidos previamente se agrupan para construir una página válida.

Lectura recomendada

Podéis encontrar más información sobre XML en el documento "Lenguajes de marcas", que encontraréis en el espacio de recursos del aula.

Los estándares web codifican la información mediante lenguajes basados en XML (*eXtensible Mark-up Language*), un lenguaje de propósito general para el marcaje de información estructurada en documentos.

Un documento XML está formado por un conjunto de elementos, delimitados por marcas[7] de inicio (como por ejemplo <*title*> o <*table*>) y de final (como por ejemplo <*/title*> o <*/table*>). Cada elemento puede tener información asociada en forma de atributos en la marca de inicio (como por ejemplo la URL en la marca <*a href="www.uoc.edu"*>) o bien en forma de contenido, que se encuentra entre la marca de apertura y la de cierre (como por ejemplo un elemento de una lista en <*li*>*Barcelona*</*li*>). Un documento XML bien formado tiene que respetar una serie de normas básicas, entre otras:

(7)En inglés, *tags*.

- El documento XML consta de una parte de cabecera y de un contenido, donde hay un único elemento raíz. El resto de elementos aparecen dentro del contenido de la raíz o de algún elemento que dependa de ella.

- Todo elemento debe tener una marca de inicio y de final. En caso de que no haya contenido, la marca puede indicar inicio y final como
.

- Las marcas de inicio y final tienen que estar correctamente anidadas: Por ejemplo, es incorrecto, tendría que ser .

- Los nombres de las marcas son sensibles a mayúsculas: a y A no hacen referencia a la misma marca.

- Los valores de los atributos tienen que ir entre comillas: "París".

- Los símbolos especiales como por ejemplo "<", ">", o "/" son caracteres reservados que no pueden aparecer directamente dentro del contenido sino que se codifican (por ejemplo, "<" se codifica como <).

- El documento puede incluir comentarios (<*!--Comentario XML--*>) o bien instrucciones para el programa que procese el documento XML (<*? instrucción atributos ?*>). Esto se puede utilizar, por ejemplo, para indicar el tipo de documento en la cabecera.

- En documentos XML complejos puede ser interesante definir una estructura concreta para sus elementos, por ejemplo, indicar que el XML de un libro tiene que tener siempre un elemento "título" y uno o más elementos "autor". Este tipo de restricciones sobre la estructura se pueden definir utilizando notaciones como por ejemplo **DTD**[8] o **XML Schema**. Por otro lado, para evitar que dos tipos de documentos utilicen la misma marca para diferentes conceptos, se pueden definir **XML Namespaces**, concepto análogo a los paquetes (los *packages* de Java) o espacios de nombres (los *namespaces* de C++) en los lenguajes de programación para evitar colisiones en nombres de tipos y de variables.

[8]DTD es la sigla de *document type definition*.

c) La capa siguiente especifica cómo asignar **significado** a los elementos de una página web. Es aquí donde se encuentra el núcleo del formato de representación de datos para la web semántica, el *Resource Description Framework* (RDF).

Ved también

El formato RDF se presenta con más detalle en el subapartado 4.1 de este módulo didáctico.

RDF es una notación estándar para la representación e intercambio de información sobre recursos. Inicialmente fue concebido para representar metadatos para recursos de la web, como por ejemplo los títulos, autores, fechas de la página web, pero realmente puede ser usado para representar cualquier otro dato. Este formato está basado en la especificación de tripletas **sujeto-predicado-objeto**, a partir de los cuales construye una representación de los datos en forma de grafo.

Para todos los datos dentro de la web semántica, utilizaremos RDF como lenguaje base para la representación de conocimiento. Con objeto de poder ser interpretado de manera automática necesitamos que esté almacenado (**serializado**) en un formato textual. Hay varias serializaciones posibles, como por ejemplo XML o Notation3, que veremos más adelante en este módulo.

Ved también

En el subapartado 4.1 de este módulo didáctico se trata sobre diferentes formatos para serializar RDF.

d) Utilizando RDF, se pueden realizar un conjunto de actividades de **representación del conocimiento**. Por ejemplo, RDF permite añadir anotaciones semánticas, pero en algún momento habrá que definir cuál es el **vocabulario** de términos que se pueden utilizar para hacer las anotaciones: cuáles son las clases de entidades y sus propiedades, y cuál es la jerarquía y relaciones entre estos términos. Con este objetivo, **RDF Schema** (**RDFS**) es un estándar para describir vocabularios de términos y relaciones sencillas entre ellos, como por ejemplo jerarquías de conceptos. Para poder representar definiciones y relaciones más complejas –en definitiva, para poder representar una ontología– disponemos de otro estándar que nos ofrece extensiones respecto a RDFS, el **Web Ontology Language** (**OWL**).

Otros estándares no se centran en la descripción de vocabularios sino de otro tipo de conocimiento: conocimiento inferencial en forma de reglas de producción. Aquí destacaremos dos estándares/especificaciones, el **Rule Interchange Format** (**RIF**) y el **Semantic Web Rule Language** (**SWRL**).

Reflexión

En el módulo "Introducción a la representación del conocimiento" hemos presentado las reglas de producción como un esquema de representación del conocimiento. No entraremos más en detalle en estos lenguajes dentro de este módulo.

Por último, además de estándares para describir conocimiento, también disponemos de un lenguaje para hacer consultas: **Simple Protocol and RDF Query Language** (**SPARQL**), un estándar para la consulta de información basada en RDF, como pueden ser ontologías RDFS y OWL.

SQL

El lenguaje SQL (*Structured Query Language*) es el lenguaje estándar para la consulta y modificación de datos en bases de datos relacionales.

SPARQL es un lenguaje inspirado en SQL que utiliza tripletas RDF y recursos tanto para hacer coincidir la consulta como para devolver los resultados de esta. Además, hay que tener en cuenta que SPARQL no es solo un lenguaje de consulta, sino también es un protocolo para acceder a datos RDF.

e) Una vez que hemos descrito información semántica y modelizado los conceptos y propiedades que aparecen, es el momento de utilizar este conocimiento en actividades de razonamiento para relacionar el conocimiento y hacer inferencias.

Los estándares descritos previamente definen una semántica formal que permite razonar sobre ellos. Destacamos en particular la **lógica descriptiva**, una familia de lenguajes formales que inspiran la notación OWL y que permiten hacer inferencias, como por ejemplo detectar información inconsistente.

Dado que ciertos tipos de razonamiento tienen un elevado coste computacional, esta capa todavía no está resuelta satisfactoriamente. De hecho, algunos consideran que este es uno de los talones de Aquiles de la web semántica, puesto que es complicado conseguir mecanismos de razonamiento que sean escalables para grandes volúmenes de información y que a la vez sean capaces de extraer conclusiones no triviales.

f) Alejándonos de la perspectiva de la inteligencia artificial y acercándonos más a las aplicaciones, los agentes que utilicen información semántica deberán poder asegurar la integridad de la información, establecer el grado de confianza de una fuente de conocimiento o mantener la confidencialidad de los datos. Con este objetivo, será necesario utilizar técnicas criptográficas (como por ejemplo el cifrado o la firma digital) y de gestión de la reputación para asegurar la **seguridad y confianza** de las aplicaciones en la web semántica.

g) Finalmente, es en la capa superior donde se sitúan las **aplicaciones** que utilizan la web semántica para aportar un valor añadido y las **interfaces** que permiten a los usuarios acceder a este conocimiento de forma amigable.

Más adelante analizaremos con detalle algunos de los formatos anteriormente mencionados: RDF, RDFS y OWL. Antes, sin embargo, consideraremos cuál es la situación de la web semántica y qué iniciativas y aplicaciones se han desarrollado en ella.

2.3. Estado actual de la web semántica

El objetivo de la web semántica es muy ambicioso y requiere de varios factores para lograrlo:

- En primer lugar, hay que disponer de estándares adecuados para añadir información semántica a las páginas web.

- En segundo lugar, los productores de contenidos deben disponer de herramientas adecuadas para etiquetar semánticamente e invertir recursos suficientes en el etiquetado, siempre respetando los estándares definidos previamente.

- Por último, tiene que existir un ecosistema de aplicaciones que puedan aprovechar los datos semánticos de forma efectiva y eficiente para proporcionar servicios de valor añadido.

Lo que se ha conseguido hasta ahora es sentar las bases tecnológicas para que el planteamiento inicial de la web semántica sea factible, mediante una infraestructura basada en estándares que lo sostienen.

A pesar de todo, el grado de adopción de los estándares por parte de la comunidad web está siendo lento. Esto hace que la visión original de la web semántica esté tardando un tiempo en hacerse realidad, aunque la tendencia actual de la web va en esta dirección: enlazar todos los datos de la web de manera global. Aun así, desde el 2010 se empieza a apreciar algún cambio, en concreto en servicios web muy populares que empiezan a incluir funcionalidad para tratar contenido semántico.

En el 2010, Facebook adopta el estándar RIF en el Open Graph Protocol (OGP), el cual tiene la misión de permitir a cualquier página web existente llegar a ser un objeto dentro de una red social; esto requiere que cada página incluya un conjunto básico de metadatos.

Lectura complementaria

Podéis consultar más sobre el **Open Graph Protocol** en el siguiente enlace:
"The Open Graph protocol"

En el 2011, los buscadores Google, Bing (Microsoft) y Yahoo! se ponen de acuerdo para adoptar el mismo vocabulario para el marcaje de información semántica dentro de páginas web. De aquí surge la iniciativa Schema.org, que provee de conjuntos organizados de vocabularios que puede utilizar cualquier página web para identificar su contenido.

Lectura complementaria

Podéis consultar más sobre Schema.org en el enlace siguiente:
"What is Schema.org?"

Pero, dejando de lado los estándares que se utilicen para codificar la información semántica, se tiene que empezar a generar y publicar contenido etiquetado semánticamente. Este contenido puede presentarse de dos maneras diferentes: en forma de metadatos incrustados dentro de documentos web, o bien como documentos exclusivamente de datos.

En la primera línea de añadir anotaciones semánticas a páginas web, los sistemas de gestión de contenidos (CMS[9]) evolucionan tecnológicamente para permitir la publicación de contenido anotado semánticamente. Un ejemplo es la herramienta Drupal, una de las más utilizadas, que en su versión 7 incorpora el estándar RDFa con este propósito.

(9)CMS es la sigla de la expresión inglesa *content management systems*.

Otra posibilidad para fomentar las anotaciones semánticas en páginas web es favorecer el etiquetado social, también llamado **folcsonomía**[10]. Utilizando este paradigma, se pide a un conjunto de personas que propongan etiquetas (palabras clave) para describir un recurso: una fotografía, un lugar web, un producto, ... De este esfuerzo colaborativo surge una clasificación en forma de conjunto plano de etiquetas que aprovecha la distribución del esfuerzo entre los miembros del colectivo y la sabiduría de los grupos[11].

(10)En inglés, *folksonomy*.

(11)En inglés, *wisdom of crowds*.

Por último, en la línea de la publicación de ficheros de datos anotados semánticamente, existen dos iniciativas muy interesantes, el *linked data* y el *open data*, que describiremos a continuación.

2.4. *Linked data*

> "Para hacer que la web semántica o denominada asimismo web de datos sea una realidad, es necesario disponer de un gran volumen de estos y tenerlos accesibles en un formato estándar y manejable. Además, las relaciones entre los datos también tienen que estar representadas. A toda esta colección de datos relacionados entre sí en la web se los denomina *linked data*. También visto como el corazón de la web semántica: la integración y razonamiento a gran escala sobre los datos en la web."
>
> "Linked data", W3C

El *linked data* fue caracterizado por Tim Berners-Lee, "Linked Data Design Issues" (2006), por medio de cuatro principios (también denominados reglas):

Lectura recomendada

T. Berners-Lee (2006). *Linked Data Design Issues*

1) Utilizar las URI para identificar los recursos públicos en la web.

2) Aprovechar el HTTP de la URI para que se puedan localizar y consultar estos recursos.

3) Proporcionar información útil del recurso (URI) por medio de los estándares (RDF*[12], SPARQL).

(12)RDF* engloba las tecnologías RDF, RDFa y RDF Schema.

4) Incluir enlaces hacia otras URI relacionadas con los datos contenidos en el recurso, de modo que se potencie el descubrimiento de información en la web.

Cabe decir que actualmente hay una gran ambigüedad en cuanto a la naturaleza exacta de *linked data*. El debate se centra principalmente en torno a si el uso del RDF y el SPARQL es obligatorio. Es decir, si el RDF está integrado o no en el concepto *linked data* o si, por el contrario, es opcional.

Cita

"Voy a hacer referencia a los cuatro principios como reglas, pero son realmente expectativas de comportamiento. Incumpliéndolas no se destruye nada, pero se pierde la oportunidad de crear datos interconectados. Esta situación también limita las diferentes maneras en que más tarde se pueden reutilizar inesperadamente. Es en concreto la inesperada reutilización de la información lo que confiere el valor añadido a la web."

Tim Berners-Lee, "Linked Data"

Así lo observa Paul Miller, en el "Linked Data Horizon Scan" del JISC (2010):

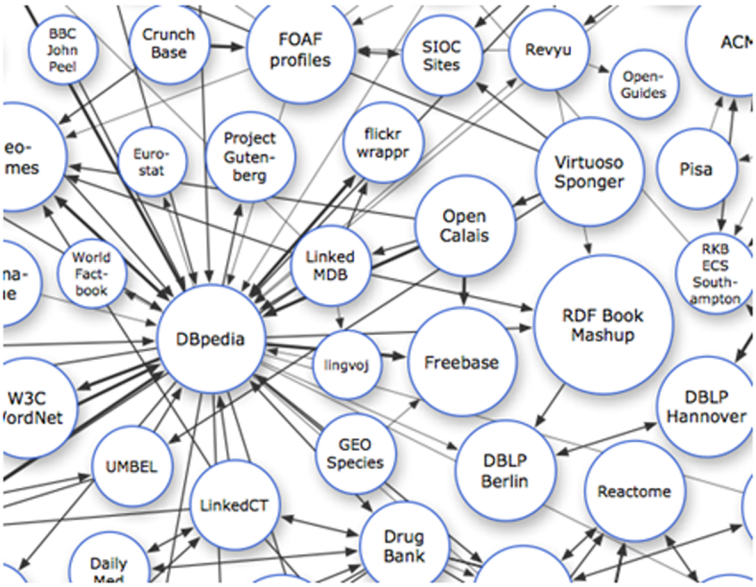
"Aunque la formulación exacta de estas declaraciones ha cambiado ligeramente desde que fue expuesta por primera vez en el 2006, continúa habiendo dudas sobre la firmeza de las necesidades para los estándares específicos; los acrónimos enmarcan un simple pero potente comportamiento.

- Nombre de los objetos y recursos sin ambigüedades.
- Hacer uso de la estructura de la web.
- Facilitar el descubrimiento de información sobre el nombre del objeto o los recursos.
- Si se conocen objetos o recursos relacionados, enlazadlos también.

Se gana mucho al comprender la filosofía que hay detrás de estas reglas, de manera separada a los estándares y a las especificaciones necesarias, para desarrollar todo su potencial."

Paul Miller, "Linked Data Horizon Scan" (2010)

Figura 2. Detalle del diagrama en nube de datos abiertos relacionados de Richard Cyganiak y Anja Jentzsch



Fuente: "The Linking Open Data cloud diagram", en: Miller, *Linked Data Horizon Scan*

Para saber más

P. Miller (2010). "Linked Data Horizon Scan". *Joint Information Systems Committee* (enero).

2.4.1. Algunos ejemplos de proyectos *linked data*

Presentamos a continuación dos ejemplos de proyectos que siguen el paradigma *linked data*: DBpedia y Linkeddata.org.

DBpedia

El objetivo de la DBpedia es extraer datos estructurados de la Wikipedia de modo que estén disponibles en la web. Esto permite utilizar DBpedia para responder consultas complejas y enlazar datos de la Wikipedia con otros conjuntos de datos dentro de la web.

Para saber más

Podéis encontrar más información en la URL de DBpedia.

Para hacerse una idea de sus dimensiones, actualmente la ontología de la Wikipedia contiene 359 clases, 1.775 propiedades y unas 2.350.000 instancias.

El proyecto tiene una filosofía abierta y colaborativa y está mantenido por la Universidad libre de Berlín, la Universidad de Leipzig y OpenLink Software.

Linkeddata.org

Esta iniciativa ofrece un lugar web (linkeddata.org/) para poder enlazar los recursos de la comunidad *linked data*, con objeto de reducir las barreras al acceso a los datos de la web. Por eso contiene información de cómo publicar datos enlazados a partir de artículos, manuales y software.

El poder de esta iniciativa no está en sus ontologías, sino en sus interrelaciones. Las ontologías están relacionadas entre sí, lo que posibilita que los programas naveguen de un concepto hacia otros conceptos relacionados, ya sea en un mismo dominio o en dominios diferentes. Por ejemplo, este sistema permitiría que la ontología LinkedGeoData, que contiene información sobre OpenStreetMaps, estuviera relacionada con DBpedia, que contiene información sobre Wikipedia y permite que puntos de interés de LinkedGeoData (como por ejemplo la ciudad de Gerona) se puedan relacionar con su entrada en Wikipedia vía DBpedia, o incluso con fotos de Gerona vía la ontología de Flickr.

2.5. *Open data*

> "**Datos abiertos** es una filosofía y una práctica que requiere que ciertos datos sean de acceso libre para todo el mundo, sin restricciones por derechos de autor, patentes u otros mecanismos de control. Tiene un espíritu parecido a un cierto número de otros movimientos "abiertos" y a comunidades como las de código abierto y de acceso abierto."
>
> "Open data", Wikipedia

Es en estos movimientos o comunidades que fomentan el acceso abierto a los datos[13] donde está cobrando más importancia el concepto de *conocimiento abierto*[14], que es más amplio que el de *datos abiertos*. El **conocimiento abierto** es una serie de principios y metodologías en lo referente a la producción y difusión del conocimiento de modo que se garanticen los derechos de acceso, uso, modificación y redistribución de la información.

(13) En inglés, *open data*.

(14) En inglés, *open knowledge*.

Por eso, el movimiento Open Knowledge Definition (OKD) trabaja para dar respuesta a lo que significa *apertura*[15] también en términos de contenido y datos.

(15) En inglés, *openness*.

> "Una parte de contenido o de datos es abierta cuando cualquiera es libre de utilizarla, reutilizarla y redistribuirla, sujeto como máximo a la exigencia del reconocimiento y/o compartición."
>
> "Open Knowledge Definition"

Para saber más

Podéis consultar la Open Knowledge Definition en el siguiente sitio web:

"Open Knowledge Definition"

En consecuencia, el concepto de apertura cubre cualquier forma de datos y contenido, como música, libros, imágenes o cualquiera otro material.

Entonces, la relación entre *open data* y *linked data* se produce cuando tenemos *linked data* que se publica explícitamente bajo una licencia abierta, que da lugar a lo que se conoce como ***linked open data***. Por lo tanto, no todas las *linked data* publicadas lo son en abierto, y no todas las *open data* están enlazadas siguiendo los principios de en Berners-Lee. Por lo tanto, se tiene que tener en cuenta utilizar el término apropiado según sea el caso.

2.5.1. Algunos ejemplos de proyectos *open data*

Dos ejemplos de proyectos que siguen la filosofía del *linked open data* son OpenStreetMap y MusicBrainz.

OpenStreetMap

OpenStreetMap es un proyecto colaborativo para crear y distribuir de manera libre datos geográficos del mundo, siguiendo el modelo de la Wikipedia. Dispone de todos los mecanismos para utilizar los datos que contiene, incluida una API para realizar nuevas aportaciones libres.

Para saber más

Podéis encontrar más información en la URL siguiente:
"OpenStreetMap"

El contenido de este proyecto se publica con una licencia abierta, la ODbL (Open Database License).

Este proyecto también se ha vinculado a la iniciativa de Open Linked Data. En particular, se ha creado una ontología que contiene la información de OpenStreetMap en formato RDF y vinculado a un conjunto de ontologías abiertas, como por ejemplo DBPedia. Esta ontología contiene actualmente más de un millar de millones de elementos y puede consultarse en linkedgeodata.org.

MusicBrainz

MusicBrainz es una enciclopedia de información musical libre en línea. La enciclopedia contiene información de artistas, sus grabaciones y las relaciones entre ellos.

Para saber más

Podéis encontrar más información en la URL siguiente:
"MusicBrainz"

El contenido de este proyecto se publica bajo la licencia Creative Commons CC BY-NC-SA-2.0. El usuario puede interactuar con el contenido a través del software cliente libre, libmusicbrainz, distribuido también bajo una licencia libre, en este caso: GNU Lesser General Public License.

Igual que en el caso de OpenStreetMap, MusicBrainz también se ha vinculado al conjunto de ontologías libres enlazadas.

Para saber más

Podéis encontrar más información sobre este proyecto en la siguiente URL:
"LinkedBrainz"

2.6. Aplicaciones de la web semántica

La web semántica tiene múltiples campos de aplicación, pero todos están orientados hacia una de estas dos perspectivas:

1) Semántica: Posibilitar el uso de técnicas de inteligencia artificial al proporcionar el significado del texto y otros tipos de datos en una página web.

2) Interoperabilidad: Facilitar el intercambio de información entre diferentes sistemas.

Los **buscadores semánticos** son una de las aplicaciones más conocidas de la web semántica. Representan un paso más en la evolución de los motores de búsqueda tradicionales, que utilizan información semántica para mejorar la precisión de los resultados o bien para proporcionar una respuesta concreta a la consulta. Como ejemplos, tenemos los buscadores SenseBot, DuckDuckGo y SWSE.

Sin alejarnos de los buscadores, otra área de aplicación es el **comercio electrónico**[16], en el que las tecnologías semánticas potencian la publicación de servicios y/o productos en Internet. La iniciativa GoodRelations va en la línea de facilitar un vocabulario estándar para hacer este marcaje semántico, con términos como por ejemplo *gr:BusinessEntity* o *gr:OpeningHourSpecification*.

(16)En inglés, *e-commerce*.

También en esta área se potencian las transacciones entre negocios[17]. En este campo tienen mucho que ver los servicios web, puesto que favorecen el intercambio de datos entre proveedores y clientes repartidos globalmente.

(17)En inglés, *business-to-business*.

Para saber más

Podéis encontrar más información sobre el vocabulario GoodRelations en la URL siguiente:
"Good Relations, The Web Vocabulary for E-Commerce"

Otro campo de aplicación es en la educación, en concreto en el ámbito del *e-learning*, en que se pueden desarrollar herramientas para la personalización de materiales docentes. Así, a partir del conocimiento previo del bagaje de un estudiante, se pueden adaptar los contenidos para una mejor comprensión.

Pero la lista de áreas en las que se aplica no termina aquí, también se trabaja en aplicaciones para la administración electrónica (e-gobernanza), la asistencia sanitaria, etc. En estos ámbitos, las aplicaciones están claramente relacionadas con el intercambio de información y la interoperabilidad.

3. Ontologías y la web

Las **ontologías**, también denominadas **modelos del dominio**, tienen un papel muy importante en la web semántica, dado que posibilitan la compartición de conocimiento dentro de la web. La definición más generalizada de las ontologías es la siguiente.

> Una ontología es la representación de la conceptualización explícita compartida de un dominio en particular.

A partir de su definición, podemos identificar las tres principales propiedades de las ontologías. Una ontología:

1) utiliza una representación explícita; por lo tanto, suele estar escrita en un lenguaje formal y en un soporte digital que puede ser leído e interpretado por programas informáticos.

2) es una conceptualización compartida; por lo tanto, representa la información que un conjunto de personas tienen con respecto a un dominio de discurso. Una ontología no puede representar el punto de vista de un solo individuo, como podría suceder, por ejemplo, con un esquema de base de datos.

3) representa un dominio en particular; por lo tanto, representa el dominio de discurso relevante para un problema concreto.

Las ontologías son el medio principal para conseguir el objetivo de la web semántica, al facilitar la definición formal de los conceptos presentes en el dominio, la jerarquía que los fundamenta y las diferentes relaciones que los unen entre sí.

Ved también

En el módulo "Introducción a la representación del conocimiento" hemos presentado las ontologías y las hemos relacionado con diferentes esquemas de representación.

En este apartado se mostrará la parte más teórica de las ontologías: ventajas e inconvenientes, procesos de construcción y metodologías de uso. Se entrará en el detalle de la implementación y en la elección de los formatos en el apartado siguiente de este módulo.

3.1. Ventajas e inconvenientes de las ontologías

Como ventaja principal de las ontologías destacaremos que el uso de ontologías en la web semántica ha proporcionado un mecanismo para la especificación común y compartida de información de dominio que puede ser transmitida entre personas y aplicaciones.

Aunque la web semántica está muy ligada al uso de las ontologías, hay que recordar que estas también se aplican en otros campos, como es el caso de la lingüística.

También ha dado lugar al resurgimiento de la disciplina de la representación del conocimiento, área de la inteligencia artificial que facilita la compartición y la reutilización del conocimiento.

Además, si se compara con las bases de datos clásicas, una ontología posibilita el uso de datos de diferente naturaleza, características y formato para inferir conocimiento nuevo. Por medio de unas reglas de inferencia y un motor de razonamiento se pueden utilizar los datos de la ontología para inferir conclusiones sobre ella misma; y todo, sin la intervención humana.

Ejemplo 8. Buscadores de contenido web

Un ejemplo de uso de las ontologías en la web semántica se encuentra en los buscadores de contenido web. Utilizando ontologías, estos pueden realizar búsquedas más allá del uso de palabras clave. Así, se pueden buscar conceptos que, aun sin aparecer literalmente en una página, sí que aparecen con un significado semántico similar o equivalente.

Por otro lado, también ha hecho emerger problemas clásicos de la representación del conocimiento:

- Hay un conflicto de requisitos entre la capacidad de expresión de los lenguajes utilizados para anotar semánticamente y la escalabilidad de los sistemas que los utilizan.

- La integración de diferentes ontologías puede llegar a ser al menos tan difícil como la integración de los mismos recursos que describen.

- El volumen de datos por representar en una ontología, aparte de aumentar la complejidad en el proceso de creación, también dificulta la manipulación de los mismos archivos que la contienen.

Ejemplo 9. La ontología eClassOWL

La ontología eClassOWL destinada a describir los tipos y propiedades de productos y servicios de la web semántica, en su versión 5.1.4 en formato RDF/XML ocupa 38,3 Mb.

Además, han emergido otros nuevos problemas relacionados con la aplicación de las ontologías en el campo de la web:

- La necesidad de crear anotaciones pertinentes para el tratamiento automático frente al tratamiento humano.

- La necesidad de validar el nivel de calidad de los contenidos, que puede variar mucho entre diferentes fuentes.

- La falta de mecanismos apropiados para comprender, visualizar y recoger las ontologías previamente desarrolladas.

- La necesidad de madurar algunos aspectos de la arquitectura de la web semántica que todavía no están suficientemente desarrollados (por ejemplo, la capa de razonamiento).

Ejemplo 10. Limitaciones de las folcsonomías

Si queremos usar la clasificación generada mediante una folcsonomía, debemos tener en cuenta que algunas de las etiquetas pueden tener faltas de ortografía, que puede haber etiquetas sinónimas, que algunas de ellas pueden ser generadas de forma maliciosa para generar *spam*, etc.

3.2. Clasificación de las ontologías

Hay diferentes formas de clasificar las ontologías, desde perspectivas tan diversas como la orientación filosófica, el propósito, el grado de formalidad, etc. Pero una muy usada es la que propone N. Guarino (1998), que las clasifica respecto a su grado de generalidad. Según esta clasificación, podemos distinguir los siguientes tipos de ontologías:

- **Ontologías de alto nivel**[18]: describen conceptos generales como el espacio, el tiempo, los objetos, los acontecimientos, las acciones, etc. Estos conceptos son independientes de un problema o dominio en particular.

(18)En inglés, *top-level ontologies*.

Ejemplo 11. Ontología de alto nivel

La ontología Cyc es un ejemplo de ontología de alto nivel, y representa el conocimiento general del mundo.

Cabe decir, sin embargo, que a causa del alcance y objetivo de esta ontología, también contiene información de dominio y de aplicación.

- **Ontologías de dominio**[19]: describen un vocabulario relacionado con un dominio genérico (por ejemplo, medicina, biología o automovilística) especializando y/o instanciando los conceptos introducidos en las ontologías de alto nivel.

(19)En inglés, *domain ontologies*.

Ejemplo 12. Ontologías del dominio médico

Hay muchos ejemplos de estos tipos de ontologías en diferentes áreas. Por ejemplo, en biomedicina encontramos SNOMED CT, MeSH y UMLS.

- **Ontologías de tareas o actividades**[20]: describen el vocabulario especializado de una tarea o actividad genérica (como quizás la diagnosis médica o la venta en general) por medio de la especialización de las ontologías de alto nivel.

(20)En inglés, *task ontologies*.

Ejemplo 13. Ontologías de tareas en SIG

La utilización de ontologías de tareas en sistemas de información geográfica (SIG; en inglés: GIS) favorece la interoperabilidad y la reutilización de datos entre aplicaciones SIG, y a la vez permite nuevos mecanismos de análisis, como son la recuperación de información o la generación automática de mapas.

A tal efecto, se han desarrollado ontologías referentes a fenómenos cartográficos e hidrológicos. Un ejemplo es la ontología Hydrology creada por la Agencia Nacional de Cartografía de Gran Bretaña (Ordnance Survey). Esta ontología está compuesta por las relaciones funcionales, topológicas y meteorológicas involucradas en el transporte y almacenamiento de aguas superficiales.

Para saber más

Podéis encontrar más información sobre la ontología *Hydrology* en el siguiente recurso web:

"Ordnance Survey Ontologies"

- **Ontologías de aplicación**[21]: son las ontologías más específicas de todas y describen conceptos que dependen de las ontologías de dominio o de tareas; y a menudo son especializaciones de ambas. Tienen una reutilización limitada, puesto que dependen de un ámbito concreto y de los requisitos de una aplicación en particular. Estas ontologías suelen estar diseñadas a medida para cada aplicación específica.

(21)En inglés, *application ontologies*.

Ejemplo 14. Ontologías en aplicaciones de genética

Las tecnologías semánticas ejercen un papel cada vez más importante en la captación y la modelización de conocimiento biológico. En consecuencia, la iniciativa *Semantic Systems Biology* propone soluciones para el desarrollo de sistemas de biología basándose en la utilización de conocimiento semántico del propio dominio.

Con objeto de facilitar la integración de datos para el análisis (generación de hipótesis) en el campo de la biología genética, se han publicado un conjunto de ontologías de aplicación bajo el proyecto The Gene Expression Knowledge Base (GeXKB):

- *Gene Expression Ontology* (GeXO)
- *Regulation of Gene Expression Ontology* (ReXO)
- *Regulation of Transcription Ontology* (ReTO)

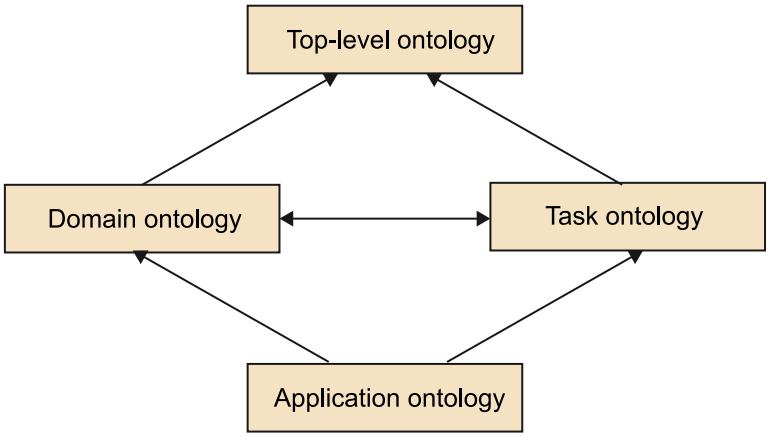
Para saber más

Para conocer más sobre la iniciativa *Semantic Systems Biology* y poder acceder a sus ontologías, podéis consultar el siguiente recurso web:

"Semantic Systems Biology"

La ventaja de esta clasificación es que potencia el propósito de la reutilización del conocimiento de las ontologías. Así, estableciendo un diseño modular, se permite simplificar tanto el desarrollo como su propósito, la reutilización. Una vez que se crea una ontología para un dominio, tendría que ser (al menos hasta cierto punto) reutilizable por otras aplicaciones en el mismo dominio. Además, el diseño modular utiliza la herencia de la información que contienen las ontologías, donde las ontologías de alto nivel describen el conocimiento general, y las ontologías de aplicación, el conocimiento para describir una determinada aplicación. Podemos ver las relaciones de herencia reflejadas en la figura 3.

Figura 3. Relaciones entre los diferentes tipos de ontología



3.3. Proceso de diseño y construcción de una ontología

Para poder utilizar una ontología, el primer paso es construirla: decidir qué conceptos y relaciones tiene que contener y la mejor manera de representarlas (nomenclatura, taxonomía, ...).

Lectura recomendada

N. F. Noy; D. L. McGuinness (2001). ""Ontology development 101: A guide to creating your first ontology". *Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report* (Informe SMI-2001-0880, marzo).

En el proceso de construcción de una ontología, también llamado *ontology engineering*, aparecen cuestiones como por ejemplo qué metodología se tiene que escoger, cuál es la herramienta más apropiada o qué lenguaje hay que utilizar para describirla. Para dar respuestas a algunas de ellas, a continuación exponemos unas reglas fundamentales enfocadas al diseño y la construcción de ontologías basadas en N. F. Noy y D. L. McGuinness (2001):

1) No existe ninguna metodología que podamos considerar perfecta o la mejor de todas.

2) El proceso de construcción de ontologías es necesariamente iterativo.

3) No hay una forma correcta de modelizar un dominio. Puede haber diferentes alternativas, pero todo dependerá del dominio y el uso que queramos hacer de la ontología.

4) Los conceptos en la ontología tienen que ser cercanos a los objetos (físicos o lógicos) y las relaciones en el dominio de interés. Probablemente son sustantivos (objetos) o verbos (relaciones) en las frases que describen el dominio.

A continuación listaremos unos pasos básicos que nos guiarán en el proceso de construcción de una ontología.

3.3.1. Paso 1 – Determinar el dominio y el alcance de la ontología

Por medio de un conjunto de preguntas relacionadas con el uso de la ontología, pondremos límite al alcance del modelo:

- ¿Cuál es el dominio que se quiere cubrir?
- ¿Cuál es la finalidad de la ontología?
- ¿Qué tipos de preguntas tendrá que permitir responder?
- ¿Quién la utilizará y quién la mantendrá?

Ejemplo 15. Dominio de una ontología: competiciones deportivas

Se quiere desarrollar una ontología en el campo deportivo encargada por el Comité Olímpico Internacional (COI).

Se planifica utilizarla en las aplicaciones destinadas a informar de todo lo que se relacione con los acontecimientos olímpicos, como también otros acontecimientos de competiciones mundiales o continentales.

Por lo tanto, contendrá conceptos que describan la información de torneos deportivos a través de sus acontecimientos, los tipos de disciplinas que implica un acontecimiento, los roles de los participantes en cada competición, y los premios asociados.

Principalmente será utilizada por los periodistas que cubran los acontecimientos olímpicos.

Además, es aconsejable listar un conjunto de preguntas de muestra que la ontología tendría que ser capaz de responder, llamadas preguntas de verificación[22] (o relevancia), según Gruninger y Fox (1995). Estas tienen una doble finalidad: por un lado, determinar el alcance de la ontología, y por el otro, serán un mecanismo de control de calidad.

[22] En inglés, *competency questions.*

Ejemplo 16. Preguntas de verificación para la ontología de competiciones deportivas

- ¿En qué disciplinas olímpicas ha competido Ben Johnson?
- ¿Cuál es el máximo de medallas que ha obtenido un mismo atleta en una olimpiada?
- ¿Hay algún año en que el equipo chino haya superado en medallas al de Estados Unidos?
- ¿Michael Phelps es un nadador?
- ¿Cuántos árbitros tiene un partido de waterpolo?

Analizando las preguntas, podemos comprobar que la ontología tendrá información de los equipos olímpicos de cada país, organizado por años y por disciplinas, etc.

3.3.2. Paso 2 – Considerar la reutilización de ontologías existentes

Conviene comprobar si podemos construir una ontología a partir de una o algunas ya existentes, refinándola o ampliando sus límites para nuestro dominio o tarea en particular. O también nos puede interesar transformar una ontología de un formalismo a otro, puesto que una ontología puede tener el contenido que necesitamos pero estar representada en un formalismo inapropiado.

Pero también la reutilización puede venir como un requisito necesario, cuando nuestra aplicación tiene que interactuar con otras aplicaciones que estén empleando ontologías ya existentes.

Así pues, será necesario hacer las siguientes consideraciones sobre la aplicación que se quiere desarrollar:

- ¿Hay alguna ontología relacionada que podamos aprovechar o ampliar? ¿En qué lenguaje está representada?
- ¿Hay que interactuar con ontologías existentes? ¿En qué lenguajes están representadas?
- Si tenemos que hacer una conversión de la ontología entre diferentes lenguajes, ¿qué convertidores están disponibles? ¿Cuál es el que minimiza la pérdida de conocimiento en la conversión?

La elección de un lenguaje

La elección de un lenguaje puede afectar, por ejemplo, a las herramientas de edición que podremos utilizar para crear la ontología. Veremos esta cuestión con más detalle en el apartado 4 de este módulo.

La lista siguiente muestra una relación de recursos web donde se pueden buscar ontologías:

- DAML
- Ontolingua
- Swoogle
- Hakia

Ejemplo 17. Reaprovechamiento de una ontología

El buscador Swoogle nos ha permitido encontrar una ontología sobre Juegos Olímpicos en:

http://swat.cse.lehigh.edu/resources/onto/olympics.owl

Antes de continuar con nuestro desarrollo, es importante comprobar si la podemos reaprovechar, ampliar, recortar o bien basarnos en ella como ejemplo para desarrollar nuestra propia ontología.

Ejemplo 18. Búsqueda de un convertidor

Imaginémonos que el responsable del desarrollo de la ontología deportiva había trabajado anteriormente en otro proyecto de desarrollo de software destinado al registro de acontecimientos deportivos. En aquel trabajo, se habían generado un conjunto de diagramas UML que contenían la definición de conceptos sobre esta misma temática deportiva destinados a la definición de la estructura de datos del software.

Con el fin de poder aprovechar aquel trabajo hecho previamente, se busca si existe algún convertidor de UML al formato RDF. Antes que nada, para saber si es posible hacer la conversión, hay que consultar el recurso web del W3C ConverterToRdf, que contiene una lista actualizada de las herramientas de conversión que hay. Al verificar la existencia de convertidores, seguidamente se escoge uno de estos: el TopBraid Composer, que, aparte de convertidor, también es un editor muy potente para modelización de datos.

3.3.3. Paso 3 – Enumerar los términos relevantes de la ontología

En este punto es importante listar todos los términos que tengan relación con nuestro dominio. Podemos quererlos utilizar para crear enunciados, o bien nos pueden interesar porque aparecen en las respuestas a las preguntas de verificación. A la hora de proponer términos, no debemos limitarnos solo a los conceptos del dominio, sino también a sus propiedades.

Ejemplo 19. Términos relevantes de la ontología de competiciones deportivas

Términos importantes relativos a los deportes olímpicos pueden ser: *acontecimiento*, *competición*, *final*, *medalla*, *programa*, *equipo*, *pista atlética*, *polideportivo*, *disciplina*, *ciclismo*, *tiempo*, *sesión*, *vuelta*, *partido*, *clasificación*, *tenis*, etc.

3.3.4. Paso 4 – Definir las clases y la jerarquía

Para realizar una jerarquía de clases podemos optar por utilizar diferentes metodologías, como explican M. Uschold y M. Gruninger (1996):

- ***Top-down***: antes que nada se crearán las clases para definir los conceptos más generales del dominio, y posteriormente se irán creando las especializaciones de estos.

Ejemplo 20. Estrategia *top-down* en la ontología de competiciones deportivas

En la ontología sobre los deportes olímpicos, empezaremos por crear clases generales, como pueden ser *Acontecimiento deportivo* y *Disciplina deportiva*.

Un *Acontecimiento deportivo* puede estar espcializado en subclases para *Juegos olímpicos*, *Mundiales de fútbol*, *Mundiales de atletismo*, entre otros.

A la vez podemos categorizar la clase *Juegos olímpicos* en *Juegos olímpicos de verano*, *Juegos olímpicos de invierno*, *Juegos paralímpicos*, etc.

- ***Bottom-up***: empezaremos por las clases más específicas, que serán las hojas de la jerarquía, y posteriormente las clases se irán agrupando en conceptos más generales.

Ejemplo 21. Estrategia *bottom-up* en la ontología de competiciones deportivas

Por ejemplo, *Equipo olímpico francés de voleibol* se puede generalizar como *Equipo olímpico*, y si todavía generalizamos más, tendremos *Equipo deportivo*.

- **Combinado de *top-down* y *bottom-up***: se empieza definiendo los conceptos más destacados, y posteriormente se irán o bien especializando o bien generalizando.

A priori, ninguno de estos tres métodos es mejor que los otros. Cuál sea el más apropiado para cada caso concreto dependerá mucho del dominio concreto y de qué visión se tenga de él. Dicho esto, posiblemente el más utilizado por los desarrolladores es el combinado de *top-down* y *bottom-up* por su mayor flexibilidad.

En particular, la estrategia *bottom-up* se ha utilizado con frecuencia al crear ontologías a partir de un conjunto de datos ya existentes, como por ejemplo en las ontologías del *linked data*, en que mayoritariamente se han creado las ontologías a partir de los datos que ya había en la web. Por otro lado, la estrategia *top-down* ha sido empleada a menudo en ontologías de dominio y en ontologías con un gran número de conceptos, que deben estar muy bien organizadas, como por ejemplo la ontología Cyc.

3.3.5. Paso 5 – Definición de las propiedades de las clases

En este paso se definen las propiedades de cada una de las clases de la jerarquía. Hay que tener en cuenta que el paso 4 y el paso 5 están estrechamente ligados, y muchas veces es difícil realizar uno y después el otro. Lo que finalmente suele hacerse es crear unas cuantas definiciones de conceptos dentro de la jerarquía y a continuación pasamos a describir sus propiedades. Y así vamos iterando entre la creación de conceptos y la descripción de propiedades sucesivamente. Hay que destacar que estos son los dos pasos más críticos en todo el proceso de construcción de la ontología.

Ved también

Os recomendamos la revisión del subapartado 2.9 del módulo "Introducción a la representación del conocimiento" de esta asignatura, donde encontraréis la descripción teórica de cómo están formadas las ontologías.

Como podréis suponer, las clases aisladas no proveen de suficiente información para responder las preguntas de verificación propuestas en el paso 1. Por lo tanto, es necesario relacionar y describir los conceptos. Los términos del paso 3 que no se han utilizado en el paso 4 serán muy probablemente propiedades de las clases.

Ejemplo 22. Propiedades de la ontología de competiciones deportivas

La clase *Competición* tiene una *Descripción*, una *Ubicación*, una *Fecha de inicio*, una *Fecha de fin*, una *Organización*, una *Disciplina de competición* y en ella toman parte unos *Participantes*.

3.3.6. Paso 6 – Definir las restricciones de las propiedades

Cada propiedad puede tener diferentes restricciones que la describan o que caractericen el tipo de valor que admite la propiedad.

Ejemplo 23. Restricción sobre el tipo de valor

Una clase *Competición* tiene la propiedad *Disciplina de competición*, cuyo valor solo admite instancias de la clase *Disciplina*, como puede ser el *fútbol*, el *ciclismo*, el *atletismo*, etc.

Describimos a continuación algunas de las restricciones más habituales:

- **Cardinalidad**: indica cuántos valores puede tener una propiedad. El valor de la propiedad también puede estar condicionada por una cardinalidad mínima o máxima que debe cumplir.

Ejemplo 24. Restricción de cardinalidad

Un *Acontecimiento deportivo* solo tiene la propiedad *Fecha de inicio*, la cual puede tener una única fecha. En cambio, puede tener más de un *Participante*.

- **Tipo de valor**: describe el tipo de valor que es admitido para asignar la propiedad. Los más comunes son: cadenas de caracteres, numérico, booleano, enumerado e instancia. El tipo instancia nos permite definir relaciones entre clases por medio de una propiedad. El ejemplo 23 expone una propiedad (la disciplina de la competición) que solo admite instancias de la clase *Disciplina*.

- **Dominio y rango**: se denomina *rango* las clases que definen los posibles valores que puede tomar una propiedad. Se denomina *dominio de una propiedad* el conjunto de clases que tienen asociada aquella propiedad.

Ejemplo 25. Restricción de rango

En el ejemplo *Competición olímpica*, la clase *Disciplina* es el rango de la propiedad *Disciplina de competición*.

Ejemplo 26. Restricción de dominio

En el mismo ejemplo 23, la clase *Competición* es el dominio de la propiedad *Disciplina de competición*.

A la hora de definir restricciones de integridad más complejas, hay que tener en cuenta el coste que representará comprobar estas restricciones (que puede llegar a ser muy grande). También hay que ir con cuidado de no introducir restricciones contradictorias.

3.3.7. Paso 7 – Crear instancias

Finalmente, el último paso consiste en crear las instancias individuales de las clases de la jerarquía. Para definir una instancia individual de una clase hace falta (1) escoger una clase, (2) crear una instancia individual de la clase y (3) dar valor a las propiedades.

Ejemplo 27. Instancia de una clase de la ontología

Si creamos una instancia de *Competición* para declarar el Fifa World Cup 2010 (la Copa del Mundo de fútbol), en la instancia definiremos los siguientes valores:

Descripción: Copa del mundo de fútbol 2010

Ubicación: África del sur

Fecha de inicio: 2010-06-11T15:00:00-03:00

Fecha de fin: 2010-07-11T15:00:00-03:00

Organización: FIFA

Disciplina de competición: Fútbol

Participantes: Australia, Japón, Corea del Norte, Corea del Sur, Nigeria, Uruguay, Eslovenia, España, ... (en total 32 equipos)

A la hora de crear las instancias, es posible que detectemos errores en la ontología, como pueden ser propiedades relevantes que no se han recabado, una taxonomía inapropiada o restricciones que faltan. Esto puede hacernos retroceder y repetir algunos de los pasos anteriores para corregir los defectos detectados. Y es que, tal como hemos dicho, el diseño de la ontología es un proceso iterativo.

3.4. Metodologías para construir ontologías

A continuación listamos algunas metodologías que describen procedimientos, tareas y ciclos de vida para la construcción y validación de las ontologías.

Ontology engineering

El proceso de construcción de una ontología se denomina ingeniería de ontologías (*ontology engineering*).

Tal como se ha comentado, no hay ninguna metodología que sea la más adecuada en todos los casos. Es necesario evaluar en cada caso cuál va a ser la metodología más adecuada, que puede variar en función de diferentes aspectos: condiciones de reutilización y compartición, resultado esperado, explotación que se hará de la ontología, existencia de ontologías de alto nivel que puedan ser reutilizadas, disponibilidad de gran cantidad de documentos que contengan información sobre el dominio, etc.

3.4.1. Methontology

Lectura recomendada

M. Fernández-López; A. Gómez-Pérez; N. Juristo (1997). *Methontology: from ontological art towards ontological engineering*.

Methontology es una metodología propuesta por el Laboratorio de Inteligencia Artificial de la Universidad Politécnica de Madrid (UPM). Permite empezar la creación de ontologías desde cero o a partir de la reutilización de otras ya existentes. Por este motivo, es uno de los métodos de ingeniería de ontologías más completo.

Esta metodología identifica un conjunto de actividades en el proceso de desarrollo de la ontología, como por ejemplo la evaluación, la configuración, la gestión, la conceptualización y la integración. Esta definición de actividades se inspira en las actividades identificadas del proceso de desarrollo de software propuesto por la organización IEEE (estándar IEEE 1074). Así, propone un

ciclo de vida de construcción basado en prototipos evolutivos, con objeto de poder agregar, cambiar y eliminar elementos en cada nueva versión (llamado **prototipo**). La metodología especifica los pasos para llevar a cabo cada una de las actividades, las técnicas utilizadas, los resultados que se tendrían que obtener y el proceso para hacer la evaluación.

Como defecto, la metodología se centra principalmente en la construcción centralizada de una ontología y no aporta orientación para el desarrollo de ontologías descentralizadas.

Para cada prototipo, el proceso consta de los siguientes pasos:

a) Especificación: Define el alcance y la granularidad de la ontología. Consiste en decidir y delimitar diferente información:

- Los objetivos de su creación, como por ejemplo permitir la reutilización del conocimiento de un dominio en particular.

- Decidir el dominio de actuación, evitando la modelización de elementos poco relevantes frente a otros más importantes.

- Determinar por quién será utilizada y también para qué, información que indirectamente ayudará a definir los puntos (a) y (b).

- Definir quién será el responsable del mantenimiento, que será quien decida qué acciones hay que emprender sobre las nuevas instancias, o qué modificación de conceptos o propiedades se permitirá, entre otros.

b) Conceptualización: Es una actividad destinada a organizar y estructurar el conocimiento adquirido. Consiste en crear un glosario de términos que pertenezcan al dominio, definirlos y estructurarlos. Con este objetivo se pueden usar diferentes formalismos: tablas, la notación UML, taxonomías, etc. De esta forma, se consigue establecer una clasificación o jerarquía entre los conceptos, las relaciones entre ellos, sus instancias, propiedades o atributos.

c) Formalización: Es un proceso que consiste en convertir el conocimiento del paso anterior en un modelo formal utilizando alguno de los esquemas de representación del conocimiento.

d) Integración: Un concepto inherente al uso de las ontologías es la reutilización. En este sentido, este paso está destinado a la integración de otras ontologías existentes a la que se construye para reducir costes y evitar duplicidades.

e) Implementación: El siguiente paso es convertir el modelo formalizado en un modelo computable por medio de un lenguaje para la construcción de ontologías, por ejemplo, RDF, OWL, etc.

Ved también

Los lenguajes para la construcción de ontologías se presentan en el apartado 4 de este módulo. El apartado 5 presenta las herramientas de edición de ontologías que se utilizarán en esta etapa.

f) Evaluación: Hay que comprobar si el funcionamiento de la ontología se ajusta a los objetivos planteados en la especificación.

g) Documentación: Si se quiere permitir la reutilización y compartición por parte de terceros, hay que documentar la estructura de la ontología y las decisiones tomadas.

h) Mantenimiento: Finalmente, hay que establecer qué líneas de mantenimiento y modificación se llevarán a cabo.

3.4.2. On-To-Knowledge

On-To-Knowledge fue un proyecto financiado por la Unión Europea destinado al desarrollo de metodologías y herramientas para facilitar la gestión de conocimiento por medio de ontologías, y destinado a profesionales no especializados en ciencias de la computación.

Lectura recomendada

D. Fensel y otros (oct., 2000). "On-to-knowledge: Ontology-based tools for knowledge management". *Proceedings of the eBusiness and eWork Conference 2000* (pág. 18-20). Madrid.

Dentro del proyecto se creó uno de los primeros lenguajes de representación de ontologías para la infraestructura de la web semántica, denominado Ontology Interchange Language (OIL). OIL se basa en conceptos desarrollados en la lógica descriptiva y la representación de marcos, sin abandonar la compatibilidad con el RDF.

Reflexión

En este módulo didáctico no presentaremos el lenguaje OIL, sino uno de sus sucesores, OWL.

La metodología On-To-Knowledge (OTKM) tiene la finalidad de facilitar la introducción al uso y el mantenimiento de aplicaciones basadas en la gestión de conocimiento mediante ontologías, focalizándose en los procesos ***knowledge meta process*** y ***knowledge process***. En este sentido, es una metodología más general que otras que se focalizan exclusivamente en la construcción de la ontología. A continuación listamos sus fases:

Knowledge meta process* y *knowledge process

Knowledge Meta Process es el proceso donde se abordan los aspectos para la introducción de una nueva solución de gestión del conocimiento, así como su mantenimiento.

Knowledge Process es el proceso para la manipulación de una solución de gestión del conocimiento.

a) Estudio de la viabilidad[23]: Hay una fase inicial para el estudio de viabilidad del proyecto. Aquí se identifican las partes involucradas, el área del problema y las soluciones potenciales. El estudio se concibe para apoyar tanto la parte técnica del proyecto como la económica.

(23)En inglés, *feasibility study*.

b) Inicio[24]: Es el proceso por el cual se obtienen las especificaciones de los requisitos de la ontología, que delimita la ontología y describe los objetivos, el dominio, el alcance, las aplicaciones compatibles con la ontología, las fuentes de conocimiento, los usuarios potenciales, los escenarios de uso, el conjunto de posibles consultas al sistema, y las ontologías potencialmente reutilizables.

(24)En inglés, *kick-off*.

c) Refinamiento[25]: El objetivo de esta fase es generar una ontología basada en las especificaciones de la fase anterior. Se descompone a su vez en varias subfases:

(25)En inglés, *refinement*.

- Se creará una taxonomía inicial que contiene todos los conceptos de la fase (Inicio).
- Seguidamente, con expertos del dominio, se creará una primera versión de la ontología, que establece relaciones entre los conceptos de la taxonomía y añade axiomas.
- Para terminar, se formalizará la ontología transfiriéndola a un lenguaje de representación.

d) Evaluación[26]: Fase para validar la utilidad de la ontología y el entorno al software asociado para el cual fue diseñada, es decir, validar los requisitos iniciales. En este punto, se pueden tener que llevar a cabo diferentes ciclos en las fases de evaluación y refinamiento antes de que se cumplan todos los requisitos y se obtenga la ontología deseada.

(26)En inglés, *evaluation*.

e) Evolución[27]: Establecer tanto los responsables como las tareas para hacer evolucionar la ontología, garantizando en todo momento su versionado.

(27)En inglés, *evolution*.

3.4.3. Text2Onto

Text2Onto (y su versión previa TextToOnto) es un entorno de trabajo para la creación de ontologías a partir técnicas de *data mining*. Se desarrolló en el contexto del proyecto KAON2 de la Universidad de Karlsruhe y del Research Center for Information Technologies de Karlsruhe (Alemania).

Para saber más

Podéis consultar más sobre el proyecto Text2Onto en el siguiente recurso web:
"KAON2"

3.4.4. SENSUS-Based

Este método es un enfoque *top-down* para derivar ontologías de dominio específico a partir de una ontología de alto nivel, como por ejemplo Cyc, ConceptNet, WordNet o SENSUS.

SENSUS

SENSUS es una extensión y reorganización de WordNet para el uso del procesamiento del lenguaje natural creada en el Information Science Institute.

Estos materiales varían la descripción original de la metodología, y en lugar de utilizar la ontología SENSUS como referencia, indicamos "ontología de referencia" para enfatizar que puede ser cualquier otra.

Los pasos son:

1) Se toman un conjunto de términos como conjunto inicial, denominados términos semilla.

2) Cada uno de los términos semilla se vinculan a la ontología de referencia.

3) Se incluyen en la nueva ontología todos los conceptos que hay en el camino entre los términos semillas y la raíz de la ontología de referencia.

4) Se incluyen también aquellos términos que son relevantes en el dominio y que no han aparecido en los pasos anteriores.

5) Finalmente, para aquellos términos que no han sido seleccionados, pero que tienen un gran número de caminos que pasan por ellos, se añade todo el subárbol en la nueva ontología.

3.4.5. Grüninger y Fox

Esta metodología se ha desarrollado a partir de la experiencia obtenida en el proyecto Toronto Virtual Enterprise (TOVE), cuyo objetivo fue desarrollar un conjunto de ontologías para modelizar procesos y actividades de negocio. La metodología lleva el nombre de sus investigadores Michael Grüninger y Mark S. Fox.

Para saber más

Para conocer más sobre el proyecto TOVE podéis consultar el siguiente recurso web:
"TOVE Ontology Project"

Los pasos propuestos en esta metodología son los siguientes:

1) Captura de los escenarios inspiradores: El primer paso consiste en identificar problemas o ejemplos en el dominio de la aplicación que no quedan resueltos por ontologías existentes.

2) Formulación informal de preguntas de competencia: Basándose en los escenarios obtenidos en la etapa anterior, se recoge un conjunto de requisitos en forma de preguntas en lenguaje natural. La ontología que estamos desarrollando tiene que ser capaz de representar el conocimiento de estas preguntas y de sus respectivas respuestas.

3) Especificación de la terminología: En este paso, se extraen los términos y palabras clave de las preguntas informales y se codifican en un lenguaje formal.

4) Formulación formal de las preguntas de competencia: Utilizando la terminología del paso anterior se describen las preguntas de competencia en una notación formal.

5) Desarrollo de la ontología: En un proceso iterativo, partiendo de la terminología se van definiendo los conceptos y las relaciones necesarias para responder a las preguntas de verificación.

6) Establecer las condiciones de evaluación de la ontología: Finalmente, habrá que fijar los criterios que usaremos para evaluar si la ontología es lo bastante completa como para dar una respuesta correcta a las preguntas de verificación.

3.5. La lógica descriptiva (DL)

La lógica descriptiva (DL[28]) es una familia de lenguajes formales que permiten expresar propiedades relevantes para la representación del conocimiento.

[28]DL es la sigla de la expresión inglesa *description logic*.

El elemento característico de DL es identificar dos niveles de discurso: la terminología (llamada **TBox**), que caracteriza conceptos y relaciones binarias entre ellos (denominadas **roles**), y la **descripción del universo** (llamada **ABox**), que describe instancias concretas de los conceptos y roles. Al conjunto de terminología y descripción del universo se lo denomina **base de conocimiento**.

Para saber más

Podéis encontrar más información sobre la lógica descriptiva en la referencia siguiente:

M. Antònia Huertas (2006). "Lógicas para la web". *Azafea, revista de filosofía* (núm. 8, pág. 85-102). ISSN 0213-3563

Ejemplo 28. Base de conocimiento DL en el dominio de los seres vivos

Considerando el dominio de los seres vivos, en la TBox podemos hacer **definiciones de conceptos** abstractos que hacen referencia a otros conceptos, como por ejemplo "un ser vivo es procarionte o eucarionte"; o bien **inclusiones de conceptos** como por ejemplo: "un roedor es un tipo de mamífero":

$$\mathtt{SerVivo} \doteq \mathtt{Procarionte} \sqcup \mathtt{Eucarionte}$$
$$\mathtt{Roedor} \sqsubseteq \mathtt{Mamifero}$$

Dentro de la Tbox también podemos establecer definiciones o inclusiones de relaciones binarias entre conceptos (roles), por ejemplo "*tieneDescendiente* es la relación inversa de tieneProgenitor":

$$\mathtt{tieneDescendiente} \doteq \mathtt{tieneProgenitor}^{-}$$

Los contenidos de la Tbox, las definiciones y las inclusiones de conceptos se denominan de forma general **axiomas terminológicos** o simplemente **axiomas**.

Por otro lado, en el ABox hablaríamos de seres vivos concretos, llamados **individuos**. Podemos indicar el concepto al cual instancian, como por ejemplo "Lassie es un perro", "Charles Darwin es un hombre", o bien el rol al cual instancian, "George Darwin es un descendiente de Charles Darwin".

$$\mathtt{LASSIE}: \mathtt{Perro}$$
$$\mathtt{CHARLES\_DARWIN}: \mathtt{Humano}$$
$$(\mathtt{CHARLES\_DARWIN}, \mathtt{GEORGE\_DARWIN}): \mathtt{tieneDescendiente}$$

Otra característica de la lógica descriptiva es su paradigma de mundo abierto (*open world assumption*). En los formalismos de mundo cerrado (*closed world assumption*) se asume un conocimiento completo del mundo, de modo que si no tenemos constancia de que "Fred es un pintor", significa que "Fred no es un pintor". Aquí, en cambio, el hecho de no tener constancia de algo no implica su negación: sencillamente puede ocurrir que no sepamos si Fred es un pintor o no lo es.

Los lenguajes de lógica descriptiva aportan las siguientes ventajas al campo de las ontologías:

- **Expresividad**: Describen un conjunto de operadores para definir restricciones y propiedades que es particularmente apropiado para tareas de representación del conocimiento, como se puede ver en la separación entre conceptos (TBox) e instancias (ABox).

- **Formalidad**: Proporcionan una definición precisa y no ambigua del significado de cada operador y de qué significa que una propiedad sea cierta o falsa. De hecho, la aparición de la DL viene motivada por la falta de semántica formal de las primeras aproximaciones a la representación del conocimiento, como por ejemplo, los marcos o las redes semánticas.

- **Razonamiento**: Permiten un mecanismo sistemático para razonar sobre una base de conocimiento. Por ejemplo, puede ser relevante calcular si una definición de concepto es "satisfactible" (es posible construir instancias de aquel concepto sin violar ninguna restricción) o si la descripción del universo es consistente (ninguna instancia del ABox contradice las restricciones de la TBox). También puede ser relevante responder consultas sobre si un individuo es instancia de una clase o si hay una relación entre dos instancias concretas.

Ejemplo 29. Ejemplo de concepto no satisfactible en DL

Si en la TBox tenemos las definiciones "un Virus es un tipo de ser vivo" y "un ser vivo no es un virus", tendremos que el concepto *Virus* no es satisfactible:

$$\texttt{Virus} \sqsubseteq \texttt{SerVivo}$$
$$\texttt{SerVivo} \doteq \neg\texttt{Virus}$$

Con esta base de conocimiento tan pequeña, la comprobación es sencilla, pero cuando la base de conocimiento crece, detectar este tipo de situaciones no es en absoluto trivial.

La diferencia entre cada lenguaje dentro de la familia DL es el conjunto de operadores que se pueden utilizar para construir propiedades. Un vocabulario de operadores más amplio da lugar a lenguajes más expresivos, pero a la vez, el razonamiento requiere más recursos de cálculo en tiempo y memoria. Dado que el tamaño de las bases de conocimiento puede llegar a ser grande, la elección en el nivel de expresividad frente a eficiencia será muy relevante y habrá que ajustarla al problema concreto que se quiera resolver.

La lógica que se suele usar como punto de partida es la lógica $\mathcal{AL}$ (del inglés *attributive logic*), que ofrece los siguientes operadores para construir conceptos y roles:

Nomenclatura

Cada lenguaje de DL se denomina con una secuencia de caracteres que indica los operadores que admite. Por ejemplo, $\mathcal{ALC}$, $\mathcal{ALCN}$ o $\mathcal{EL}$.

- El concepto universal ($\top$), que contiene todos los individuos.

- El concepto vacío ($\bot$), que no contiene ningún individuo.

- Nombres de concepto atómicos (que no aparecen en la parte izquierda de ninguna definición).

- Negación de conceptos atómicos (¬).

- Intersección de conceptos (⊓).

- Cuantificación existencial limitada (∃ `Rol` ⊤), el conjunto de individuos que participan en un rol.

Ejemplo 30. Ejemplos de lógicas descriptivas

La lógica $\mathcal{AL}$ permite hacer definiciones como "un omnívoro es herbívoro y carnívoro a la vez".

`Omnivoro` ≐ `Herbivoro` ⊓ `Carnivoro`

Esta lógica tiene limitaciones, por ejemplo, para describir el complementario de un concepto. Así, si ya hemos definido el concepto *Vertebrado*, entonces no podemos definir "un Invertebrado es un Animal no Vertebrado".

`Invertebrado` ≐ `Animal` ⊓ ¬`Vertebrado`

Hay una extensión de la lógica $\mathcal{AL}$ denominada $\mathcal{ALC}$ ($\mathcal{C}$ de complementario) que permite utilizar la operación complementario sin ningún tipo de restricciones. Como efecto colateral, al poder definir el complementario de una intersección, esta lógica también admite la unión de conceptos (⊔), como complementario de la intersección.

A continuación presentamos algunas extensiones de la lógica $\mathcal{AL}$, indicando el nombre de la extensión, el operador que se introduce y un ejemplo de cómo se aplica.

Extensión	Operador	Significado	Ejemplo
$\mathcal{C}$	¬*Concepto*	Complementario de conceptos no atómicos	"Un invertebrado es un animal no vertebrado" `Invertebrado` ≐ `Animal` ⊓ ¬`Vertebrado`
$\mathcal{E}$	∃ `Rol` *Concepto*	Cuantificador existencial: conjunto de individuos que participan en un rol e instancian un concepto concreto	"Un simbionte es un ser vivo que está en alguna relación de simbiosis con cualquier otro ser vivo" `Simbionte` ≐ `SerVivo` ⊓ ∃ `tieneSimbiosis.SerVivo`
$\mathcal{U}$	*Concepto* ⊔ *Concepto*	Unión de conceptos: nuevo concepto que contiene el conjunto de individuos de ambos conceptos	"Un ser vivo es procarionte o eucarionte" `SerVivo` ≐ `Procarionte` ⊔ `Eucarionte`
$\mathcal{N}$	$\leq n$ `Rol` (*máximo*) $\geq n$ `Rol` (*mínimo*)	Restricciones numéricas: mínimo y máximo de instancias que participan en el rol	Podemos definir la clase de los mamíferos que han establecido como mínimo dos relaciones de simbiosis: `X` ≐ `Mamifero` ⊓ $\geq$`2tieneSimbiosis`
$\mathcal{Q}$	$\leq n$ `Rol` *Concepto* (*máximo*) $\geq n$ `Rol` *Concepto* (*mínimo*)	Restricciones numéricas calificadas: como $\mathcal{N}$, pero indicando el concepto al que pertenecen las instancias	Podemos definir la clase de seres vivos que participan como mucho en 3 relaciones de simbiosis con plantas: `Y` ≐ $\leq$3 `tieneSimbiosis.Planta`

Extensión	Operador	Significado	Ejemplo
O	{*y 1,..., iN*}	Nominales: concepto con un conjunto de instancias acotado y predefinido	"Hay dos mecanismos de reproducción, la sexual y la asexual" `MecanismoReproduccion ≐ { SEXUAL, ASEXUAL }`
$\mathcal{F}$	–	Restricciones funcionales: se puede forzar que algunos roles tengan que ser funciones, es decir: `si (a,b): Rol y (a,c): Rol, entonces b = c`	El rol *reproduccion* de un ser vivo (*MecanismoReproduccion*) es un rol funcional: un ser vivo sólo puede tener un tipo de reproducción concreto.

Más adelante, hablaremos de la notación concreta que se utiliza en la práctica para codificar propiedades de DL: el lenguaje OWL. Como veremos, muchos de los términos y de los operadores en OWL tienen su origen en la notación formal de la lógica descriptiva.

Ved también

El lenguaje OWL (*web ontology language*) se presenta en el subapartado 4.3 de este módulo.

4. Formatos

El estallido de la popularidad de la web ha dado lugar al crecimiento y desarrollo de nuevos formatos de representación de datos para adaptarse a nuevas necesidades.

La mayoría de estas representaciones están basadas en el paradigma de los lenguajes de marcas, en concreto XML (*eXtensible Mark-up Language*). XML es esencial para el intercambio de datos e información estructurada entre diferentes plataformas. Pero además, también tiene un papel relevante en la manipulación de información semántica.

En este apartado se exponen los diferentes formatos utilizados para la representación de información semántica, que tienen como característica común que acaban codificándose como documentos XML.

4.1. RDF

Resource Description Framework (RDF) es un estándar del W3C para el intercambio de datos y la descripción de su semántica.

Para saber más

RDF se convierte en febrero de 1999 en una recomendación del W3C.

"RDF/XML Syntax Specification (Revised)"

Concretamente, RDF proporciona un modelo de datos que permite la integración de varias fuentes de datos, incluso de esquemas diferentes, y soporte para la evolución de los esquemas a lo largo del tiempo. Aunque inicialmente RDF fue concebido para representar metadatos de páginas web, es un paradigma de propósito general que se puede utilizar para añadir metadatos a cualquier tipo de información.

En otras palabras, RDF nos permite escribir "metainformación" para cualquier tipo de datos y trabajar con ellos. Podemos describir las propiedades de los documentos web y cualquier otro recurso, y garantizar en todo momento la interoperabilidad entre aplicaciones diferentes sin perder el significado de los datos.

4.1.1. El modelo de datos de RDF

Los estándares de RDF definen el modelo de datos de manera separada de su serialización, puesto que el mismo modelo puede admitir diferentes representaciones textuales.

El modelo RDF se basa en relacionar **entidades** por medio de relaciones binarias, también denominadas **enunciados**. Cada uno de estos enunciados es una **tripleta** formada por tres elementos:

Enunciado y tripleta

Los términos enunciado y tripleta (*triple*) son intercambiables.

- un **sujeto**, la entidad origen de la relación;
- un **predicado**, también denominado **propiedad**, que identifica el arco de la relación;
- y un **objeto**, entidad de destino de la relación.

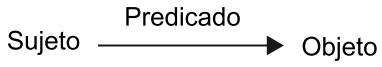
Gráficamente podemos mostrar cada tripleta como la secuencia de tres elementos:

Sujeto - Predicado - Objeto

Reflexión

En el módulo "Introducción a la representación del conocimiento" de esta asignatura se describen los esquemas de representación basados en redes, para los que esta representación en forma de grafo resulta especialmente cercana y adecuada.

O bien, podemos mostrar la tripleta como un arco dentro de un grafo dirigido:



Los conjuntos de tripletas RDF se denominan **grafo RDF**, puesto que el documento RDF se puede ver como un grafo dirigido etiquetado donde las entidades son vértices del grafo, las tripletas son arcos y los predicados son etiquetas dentro de cada arco.

Ejemplo 31. Enunciado simple

En un enunciado donde se describe una propiedad de un recurso, como por ejemplo "El coche de color azul", los tres elementos que distinguimos son:

- el sujeto es: coche
- la propiedad es: color
- el objeto es: azul

Lógicamente, no todo el conocimiento se puede describir con una única tripleta, sino que cuando hay información compleja, puede ser necesario descomponerla en un conjunto de tripletas.

Ejemplo 32. Enunciados de una dirección

"La Clínica Dientes Blancos se encuentra en la calle Mayor, 5 CP 08001 de Barcelona"

Descompondremos el enunciado compuesto de la dirección de una clínica en un conjunto de enunciados únicos:

- el sujeto es: Clínica Dientes Blancos
- la propiedad es: dirección
- el objeto es: calle Mayor, 5 CP 08001 de Barcelona

- el sujeto es: calle Mayor, número 5 CP 08001 de Barcelona
- la propiedad es: calle

- el objeto es: Mayor, 5

- el sujeto es: calle Mayor, número 5 CP 08001 de Barcelona
- la propiedad es: código postal
- el objeto es: 08001

- el sujeto es: calle Mayor, número 5 CP 08001 de Barcelona
- la propiedad es: población
- el objeto es: Barcelona

4.1.2. El identificador unívoco de recursos

El modelo de datos RDF utiliza un mecanismo para identificar recursos de manera única: la especificación URI (*Uniform Resource Identifier*). Este estándar se utiliza para identificar entidades y propiedades de manera unívoca, y garantizar una estructura semántica no ambigua y a la vez efectiva, puesto que representa conseguir un denominador mínimo común a todos los modelos de información.

RDF utiliza lo que se denomina URIref (*URI references*) para nombrar el sujeto, la propiedad o el objeto de un enunciado. Un URIref es una URI con un identificador al final precedido por un separador, el símbolo #.

Ejemplo 33. URIref

La URIref *http://www.nytimes.com/#world* está compuesta por la URI *http://www.nytimes.com/* y el identificador *world*, precedido por un separador (#). Esta URIref hace referencia a la sección de noticias del mundo del periódico *New York Times*.

Conviene subrayar que una URIref nos sirve para identificar cada elemento del enunciado (sujeto, predicado y objeto), a excepción del objeto que en algunos casos puede ser también un literal. A la hora de asignar una URIref, podemos crear una nueva o bien seleccionar alguna que ya esté definida por alguna organización o iniciativa, como por ejemplo la Dublin Core.

La Dublin Core

La iniciativa de metadatos de Dublin Core (DCMI) es una organización dedicada a fomentar y promover los vocabularios especializados de metadatos para la descripción de recursos.

Ejemplo 34. Enunciado de la dirección de una clínica

La figura siguiente muestra un enunciado que relaciona dos entidades: un sitio web de una clínica y su dirección física:

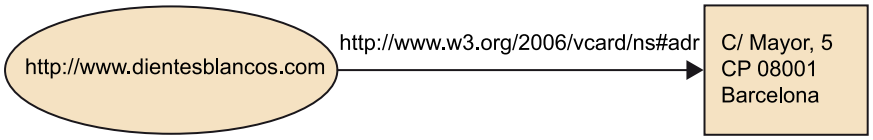
- Sujeto: (sitio web) http://www.dientesblancos.com.

- Predicado: identifica la propiedad del sujeto, en este caso "la dirección postal",

- Objeto: el valor de la propiedad "dirección": "calle Mayor, 5 CP 08001 Barcelona", que es un literal.

Hay que destacar el uso de dos URI: una para identificar el recurso "página web de la clínica", y la otra para la propiedad "dirección postal", esta última identificada por el término *adr* dentro del vocabulario de tarjetas personales electrónicas (*vCard*) proveído por el W3C.

Figura 4. Tripleta correspondiente a una dirección postal en RDF

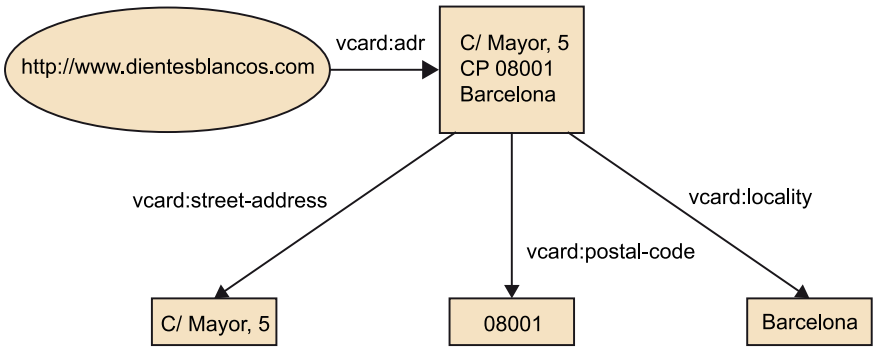


Cuando se serializan enunciados RDF, se pueden abreviar las URI utilizando la sintaxis que permite definir espacios de nombres. De este modo, podemos sustituir http://www.w3.org/2006/vcard/ns por únicamente *vcard*. Por lo tanto, la propiedad http://www.w3.org/2006/vcard/ns#adr quedaría como *vcard:adr*.

Ejemplo 35. Descomposición de la dirección postal de una clínica

La figura 5 muestra la descomposición de la dirección postal de la clínica del ejemplo anterior en diferentes enunciados siguiendo el mismo criterio escogido en el ejemplo 32.

Figura 5. Descomposición en diferentes enunciados de una dirección



4.1.3. Contenedores

Los contenedores[29] sirven para describir una agrupación de entidades denominadas miembros del contenedor. El significado es igual que cuando en los lenguajes de programación definimos estructuras de datos como *lista*, *pila* o *vector*. En el caso de RDF, se contemplan tres posibles tipos de contenedores:

[29] En inglés, *containers*.

- ***Bag*** (*<rdf:Bag>*): Contenedor multiconjunto donde se permiten duplicados y el orden de los miembros no es significativo.

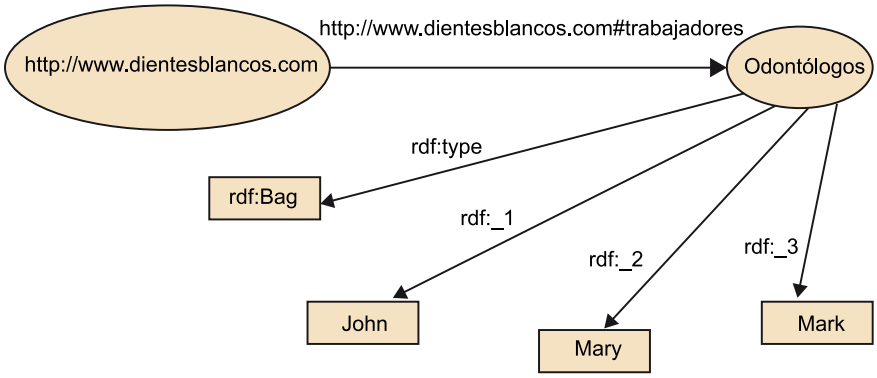
- ***Sequence*** (*<rdf:Seq>*): Contenedor secuencia donde se permiten duplicados y donde el orden de los miembros sí es significativo.

- ***Alternative*** (*<rdf:Alto>*): Contenedor que guarda una lista de recursos o literales alternativos, de los cuales se tiene que seleccionar uno. Un ejemplo podría ser la lista de literales “Retry”, “Ignore”, “Cancel”, de entre los cuales se tiene que elegir uno.

Ejemplo 36. Odontólogos que trabajan en la clínica Dientes Blancos

La figura 6 muestra el grupo de odontólogos que trabajan en la clínica del ejemplo anterior: John, Mary y Mark. Se utilizará una nueva propiedad, *trabajadores*, para asociar esta información. En concreto, se crea una entidad llamada *Odontologos*, que es una agrupación sin orden (*Bag*).

Figura 6. Conjunto de odontólogos que trabajan en la clínica



4.1.4. Colecciones

Una limitación de los contenedores es que son listas abiertas, es decir, que no hay ninguna limitación sobre el número de miembros que pueden tener. Por lo tanto, los contenedores no tienen ningún mecanismo para evitar que en otras partes del documento RDF se añadan nuevos miembros al contenedor.

Con el fin de sostener la creación de listas cerradas, es decir, describir grupos de entidades que contienen únicamente unos elementos especificados, RDF provee lo que se denomina **colecciones**[30]. Se utiliza un vocabulario específico para estructurar este tipo de lista: las propiedades *rdf:first* y *rdf:rest* para estructurar los nodos, y la entidad *rdf:nil* para indicar la finalización de la lista.

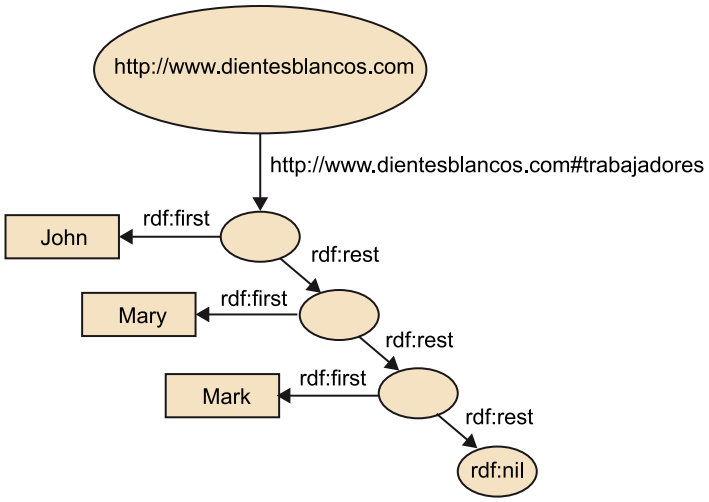
[30]En inglés, *collections*.

Una lista se estructura encadenando el conjunto de entidades que va a contener (pueden ser elementos en blanco, como veremos en el próximo ejemplo), donde cada una tiene una relación con una entidad, y otra con el resto.

Ejemplo 37. Lista cerrada de odontólogos que trabajan en la clínica Dientes Blancos

La figura 7 muestra el grupo de odontólogos que trabajan en la clínica del ejemplo anterior: John, Mary y Mark. Ahora definimos este grupo de trabajadores utilizando una lista cerrada.

Figura 7. Colección cerrada de odontólogos que trabajan en la clínica



4.2. Esquema RDF

Como hemos ido explicando a lo largo de este material, RDF provee una manera de expresar conocimiento sobre unos recursos en forma de tripletas. No obstante, RDF por sí mismo no proporciona ningún mecanismo para describir propiedades, ni las relaciones entre estas propiedades ni los recursos correspondientes. Es decir, no aporta ninguna ayuda para describir la semántica.

Para saber más

RDF Schema se convierte en febrero del 2004 en una recomendación del W3C.

"RDF Vocabulary Description Language 1.0: RDF Schema"

De hecho, en los enunciados RDF es fundamental utilizar términos que transmitan un significado inequívoco para que las aplicaciones los entiendan cuando procesen el documento. En RDF, este significado se expresa por medio de un esquema.

El **esquema RDF** (**RDFS**[31]) (también conocido como *RDF vocabulary*) es el encargado de proveer una extensión para definir vocabularios y conjuntos de propiedades semánticas en el contexto de una comunidad en particular (como por ejemplo, las direcciones postales). Es decir, un esquema RDF describe clases, jerarquías y propiedades.

[31]RDFS es la sigla de *RDF Schema.*

Clases y propiedades del esquema RDF

El sistema de clases y propiedades es muy similar al que se emplea en la programación orientada a objetos.

Así, podemos pensar que un esquema es una especie de diccionario que define los términos que se utilizarán en un enunciado para otorgarle significados específicos.

Recapitulando, RDFS permite definir los términos que utilizarán los enunciados RDF y se les otorgará significados específicos. Además, para evitar definiciones conflictivas con el mismo término, se utilizarán espacios de nombres. De este modo se consigue asociar cada uso de una palabra al esquema donde tiene una definición determinada.

4.2.1. Clases y subclases en los esquemas RDF

Una clase puede representar casi cualquier categoría de entidad, por ejemplo, páginas web, personas, documentos, conceptos abstractos, y un largo etc. Para crearlas, se utilizarán los recursos *rdfs:Class* y *rdf:Resource*, y las propiedades *rdf:type* y *rdfs:subClassOf*.

Instancias

Los recursos que pertenecen a una clase se denominan instancias.

En un RDFS, una clase es cualquier recurso que tenga una propiedad de tipo *rdf:type* con un valor *rdfs:Class*. Una subclase será cualquier recurso que tenga la propiedad *rdfs:subClassOf* y el objeto sea un recurso de clase.

Ejemplo 38. Vocabulario de acontecimientos deportivos: clases

Supongamos que queremos crear un RDFS para describir el vocabulario asociado a las competiciones deportivas. De entrada utilizaremos la URIRef abreviada *sport* para representar el espacio de nombres inventado http://www.schema.org/sport.

Antes que nada declaramos las clases "disciplinas deportivas" y "acontecimientos deportivos":

```
sport:SportsDiscipline rdf:type rdfs:Class
sport:SportingEvent rdf:type rdfs:Class
```

Seguidamente podemos describir clases adicionales como pueden ser los deportes: atletismo, ciclismo y fútbol. También los acontecimientos como una competición o un torneo.

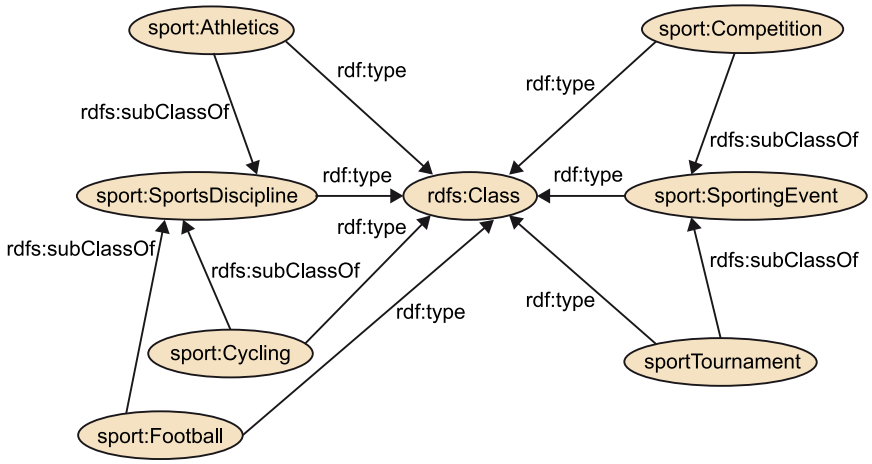
```
sport:Athletics rdf:type rdfs:Class
sport:Cycling rdf:type rdfs:Class
sport:Football rdf:type rdfs:Class
sport:Competition rdf:type rdfs:Class
sport:Tournament rdf:type rdfs:Class
```

Y finalmente se indicará la relación de jerarquía entre las clases:

```
sport:Athletics rdfs:subClassOf sport:SportsDiscipline
sport:Cycling rdfs:subClassOf sport:SportsDiscipline
sport:Football rdfs:subClassOf sport:SportsDiscipline
sport:Competition rdfs:subClassOf sport:SportingEvent
sport:Tournament rdfs:subClassOf sport:SportingEvent
```

La figura 8 muestra cómo queda en forma de grafo.

Figura 8. Esquema RDF para definir el vocabulario de acontecimientos deportivos.



4.2.2. Propiedades

Además de describir clases y organizarlas, también podemos describir predicados (propiedades) específicos y su relación con los datos: qué tipo de datos tiene (objeto) y a qué clases es aplicado (sujeto).

Antes que nada, para definir un predicado, utilizaremos un enunciado con el predicado *rdf:type* y el objeto *rdfs:Property*.

Seguidamente indicaremos los valores que puede tener la propiedad mediante un enunciado con el predicado *rdf:range*, cuyo objeto será alguna clase o tipo de datos que contenga los valores que puede tomar el predicado.

Finalmente, indicaremos en qué clases se asigna la propiedad. Por eso, se utiliza un enunciado donde el predicado es *rdf:domain* y el objeto es la clase (o tipo de datos) en que la propiedad que estamos definiendo es aplicable.

Además, también podemos tener una jerarquía de propiedades utilizando el predicado *rdfs:subPropertyOf*.

Ejemplo 39. Ampliación del vocabulario de acontecimientos deportivos, las propiedades

Queremos continuar ampliando el RDFS del ejemplo anterior para describir el vocabulario asociado a las propiedades de una competición: tipo de competición, fecha de inicio, ubicación y participantes. En algunos casos también habrá que ampliar con clases nuevas y nuevos tipos de datos.

Declaramos los enunciados para definir la propiedad "disciplina de competición", en qué clase será aplicado y el tipo de datos:

```
sport:CompetitionType rdf:type rdfs:Property
sport:CompetitionType rdfs:domain sport:Competition
sport:CompetitionType rdfs:range sport:SportDiscipline
```

En cuanto a la fecha de inicio, declararemos que será una propiedad y que su dominio serán las competiciones. Pero para declarar que el tipo de datos será una fecha, también se define el tipo de datos "fecha".

```
sport:StartDate rdf:type rdfs:Property
sport:StartDate rdfs:domain sport:Competition
xsd:date rdf:type rdfs:Datatype
sport:StartDate rdfs:range xsd:date
```

La propiedad "tener una ubicación" se definirá por medio de un literal de texto. Por eso también hay que crear el recurso "localización":

```
xsd:string rdf:type rdfs:Datatype
sport:Location rdf:type xsd:string
sport:hasLocation rdf:type rdfs:Property
sport:hasLocation rdfs:domain sport:Competition
sport:hasLocation rdfs:range sport:Location
```

Los participantes de una competición pueden presentarse como un jugador individual o en grupo. Por este motivo se utilizará el vocabulario FOAF (*friend of a friend*), un recurso externo a nuestro vocabulario que permite describir personas y las relaciones que se establecen entre ellas, como por ejemplo, grupos.

```
sport:competesIn rdf:type rdfs:Property
sport:competesIn rdfs:domain foaf:Agent
sport:competesIn rdfs:range sport:Competition
sport:Team rdfs:subClassOf foaf:Agent
sport:Player rdfs:subClassOf foaf:Agent
```

El proyecto FOAF

El proyecto FOAF, que define y amplía el vocabulario FOAF, se inició en el 2000 de la mano de Miller y Libby Brickley Dan. Se puede considerar como la primera aplicación social de la web semántica, puesto que combina la tecnología RDF con el interés por la web social.

Ejemplo 40. Uso del vocabulario de acontecimientos deportivos

Ahora, con el RDFS que hemos ido creando con los dos ejemplos anteriores, podemos definir el recurso "La Copa del mundo de fútbol 2010 tuvo lugar en África del sur y su fecha de inicio fue el 11 de junio del 2010". En este contexto utilizaremos un espacio de nombres ex:

```
ex:FIFAWorldCup2010 sport:CompetitionType sport:Football
ex:FIFAWorldCup2010 sport:hasLocation "South Africa"
ex:FIFAWorldCup2010 sport:StartDate 2010-06-11T15:00:00-03:00
```

Además, también crearemos el recurso que identifica al equipo de Japón como uno de los participantes de la Copa del Mundo. Por brevedad, solo añadiremos dos jugadores, Takuto Hayashi y Jaito Yamoto. Recordamos que se utilizará el vocabulario FOAF, y el espacio de nombres para este vocabulario será foaf:

```
ex:JapanTeam rdf:type foaf:Group
ex:JapanTeam foaf:name "Japan Team"

ex:JaitoYamoto rdf:type foaf:Person
ex:JaitoYamoto foaf:name "Kaito Yamamoto"

ex:TakutoHayashi rdf:type foaf:Person
```

La potencia del RDF

Este ejemplo pretende solo proporcionar un ejemplo de la potencia de utilizar RDF para la representación del conocimiento y de reutilizar especificaciones conocidas como FOAF. Una alternativa de representación sería utilizar la entrada de la DBPedia de Kaito Yamamoto en el predicado *member*. Ello permitiría reutilizar conocimiento ya existente, y, aparte, proporcionaría nueva información actualizada sobre este jugador, como por ejemplo su nombre, el equipo en el que juega, los acontecimientos deportivos en que ha participado y su entrada en *wikipedia*, entre otros.

```
ex:TakutoHayashi foaf:name "Takuto Hayashi"

ex:JapanTeam foaf:member ex:JaitoYamoto
ex:JapanTeam foaf:member ex:TakutoHayashi
```

4.3. OWL

OWL (*Web Ontology Language*) proporciona un lenguaje para la definición de ontologías estructuradas basadas en la web. Los lenguajes anteriores se idearon pensando en ontologías para comunidades de usuarios específicas (sobre todo en el campo de las ciencias y en aplicaciones de comercio electrónico), pero no se definieron para ser compatibles con la arquitectura de la WWW en general, y con la web semántica en particular.

OWL amplía RDF Schema para poder expresar relaciones complejas entre diferentes clases y ofrece una mayor precisión a la hora de establecer restricciones en clases y propiedades específicas. OWL aprovecha las características de RDF para añadir las siguientes capacidades a las ontologías:

Para saber más

OWL está aprobado por el W3C y ha despertado gran interés en el campo académico, médico y comercial:

"Web Ontology Language (OWL)"

- Capacidad de ser distribuidas por diferentes sistemas
- Escalabilidad según las necesidades de la web
- Compatibilidad con los estándares web de accesibilidad e internacionalización
- Ontologías abiertas y extensibles
- Capacidad de razonamiento basado en lógica descriptiva

OWL permite definir ontologías que pueden ser utilizadas a través de diferentes sistemas para compartir información específica de un dominio, como pueden ser finanzas, medicina, deporte, etc. Las ontologías se encargan de definir los conceptos necesarios para describir y representar un determinado dominio y las relaciones existentes entre estos conceptos. En este sentido, mediante OWL se pueden definir y aplicar restricciones sobre clases (conceptos), propiedades de clases y relaciones entre clases.

El diseño del OWL

El diseño del OWL se ha visto influido por los lenguajes anteriores de descripción de ontologías, como son el SHOE, DAML y OIL.

4.3.1. Versiones del lenguaje OWL

Actualmente existen dos versiones de la notación OWL, OWL 1 y OWL 2, también llamadas OWL y OWL2 respectivamente. Las separan cinco años de diferencia: la recomendación W3C de OWL aparece al 10 de febrero del 2004, y la segunda versión llega el 27 de octubre del 2009.

Para saber más

Para saber más sobre las especificaciones OWL 1, podéis consultar el siguiente recurso web:

"OWL Web Ontology Language Reference"

Para saber más sobre las especificaciones OWL 2, podéis consultar el siguiente recurso web:

"OWL 2 Web Ontology Language, Documento Overview (2.ª ed.)"

Si queréis conocer más el detalle de los *profiles* del OWL 2, podéis consultar el siguiente recurso web:

"OWL 2 Web Ontology Language, Profiles (2.ª ed.)"

Para saber más sobre la sintaxis del OWL 2, podéis consultar el siguiente recurso web:

"OWL 2 Web Ontology Language, Manchester Syntax (2.ª ed.)"

OWL 2 es una extensión y a la vez una revisión de la OWL 1; por lo tanto, hereda de las características del lenguaje, las decisiones de diseño y casos de uso para OWL 1. De este modo, en todo momento se garantiza completamente la compatibilidad, de manera que todas las ontologías OWL 1 continúan siendo válidas como ontologías OWL 2.

OWL 1 está compuesto por tres sublenguajes: OWL Lite, OWL DL y OWL Full, que veremos con más detalle en el subapartado siguiente. OWL 2 añade tres perfiles nuevos, OWL 2 Profiles: EL, QL y RL, y también una nueva sintaxis, OWL 2 Manchester Syntax.

Los nuevos perfiles de OWL tienen como objetivo facilitar la elección del subconjunto de elementos de OWL más adecuado según la estructura y explotación de la ontología que se desarrolle. En particular, el perfil EL permite tratar ontologías con un gran número de clases y propiedades en tiempo polinomial. El perfil QL está enfocado a ontologías con un gran número de instancias y en el que la tarea de inferencia más utilizada está relacionada con la consulta de información. Este perfil va a permitir resolver las inferencias en tiempo polinomial. Finalmente, el perfil RL es el que permite una mayor expresividad a cambio de efectuar inferencias más costosas.

De las mejoras aportadas a la versión 2 respecto a OWL 1, algunas de ellas son solo sintácticas, mientras que hay otras que ofrecen mayor expresividad, como por ejemplo, nuevos tipos de restricción (restricciones numéricas cualificadas) o de propiedades (asimétricas, reflexivas o disjuntas).

4.3.2. Sublenguajes de OWL: Lite, DL y Full

Reflexión

En el subapartado 3.5 de este módulo hemos repasado diferentes lenguajes de lógica descriptiva y los operadores que soportaba cada uno. Aquí cada sublenguaje OWL apoya a un lenguaje de lógica descriptiva diferente.

OWL es un lenguaje basado en la lógica descriptiva. Esto hace que haya una relación directa entre su expresividad y la complejidad computacional a la hora de realizar razonamientos. Por este motivo, OWL se presenta en tres niveles de lenguajes: Lite, DL y Full, cada uno con diferentes niveles de expresividad. De este modo, se puede elegir uno u otro según las necesidades de expresividad y eficiencia de nuestra aplicación.

Así, de menos a más expresivo:

- **OWL Lite**: Es la versión más sencilla de ontología y, por lo tanto, la más restringida en expresividad; ofrece sólo la definición de una jerarquía y restricciones simples de cardinalidad (0 o 1). Su lógica descriptiva permite

unas restricciones numéricas muy limitadas. Por su simplicidad, es ideal para la migración rápida desde otros lenguajes de ontologías.

- **OWL DL**: Aporta ampliaciones respecto a OWL Lite y está diseñada para tener la máxima expresividad posible pero garantizando la completitud computacional (todas sus conclusiones son computables) y decidibilidad (el tiempo de razonamiento siempre será finito). Como contrapartida a poder expresar propiedades más complejas, el tiempo de razonamiento es superior que en el caso de OWL Lite.

- **OWL Full**: Este nivel presenta un cambio conceptual considerable respecto de los niveles Lite y DL. Aunque en lo que se refiere a la sintaxis no aumenta respecto de OWL DL –es la misma–, sí que permite mantener la misma relajación del RDF Schema. Por ejemplo, Lite y DL tienen muy separados los conceptos instancias, clases y propiedades, en cambio Full permite que una URI sea clase e instancia a la vez –lo que se conoce como una metaclase– o incluso una propiedad. Por este motivo, puede haber procesos de inferencia que no finalicen nunca.

Para entender con más detalle lo que representa cada uno de los sublenguajes o niveles de OWL, introducimos algunas de las definiciones de la terminología básica adoptada por el documento W3C OWL Reference Guide:

a) *Class* <*owl:Class*>: Una clase permite agrupar un grupo de instancias relacionadas porque comparten propiedades. Dos conceptos relacionados con las clases son *Thing* y *Nothing*:

Las clases de OWL

El término ***class*** en OWL equivale a los **conceptos** que se definen en la TBox de DL.

- *Thing* <*owl:Thing*>: es la superclase de todas las clases.

- *Nothing* <*owl:Nothing*>: es una clase que no tiene instancias y es una subclase de todas las clases.

Las clases *Thing* y *Nothing*

Las clases *Thing* y *Nothing* equivalen a los conceptos **universal** ($\top$) y **vacío** ($\bot$) de DL.

b) *Individuals*: Son las instancias de las clases.

Los *individuals* de OWL

Los ***individuals*** de OWL son precisamente los individuos que se describen en el ABox de DL.

c) *Property* (propiedad): Es una relación binaria que permite relacionar instancias (<*owl:ObjectTypeProperty*>) o instancias con literales (<*owl:DataTypeProperty*>).

d) *RDF Resources* (*recurso RDF* o simplemente *recurso*): Es cualquier elemento que tiene un identificador único (o URI).

El término *property* en OWL

El término ***property*** en OWL encaja con los **roles** de DL.

Una de las diferencias principales entre los tres sublenguajes es debido a cómo cada uno de ellos define los elementos *owl:class* y *owl:ObjectTypeProperty*.

A continuación matizamos estas diferencias:

- En el nivel OWL Full, *owl:Class* está definido como un equivalente de *rdfs:class*. Así cualquier clase que es una subclase de *rdfs:class* es también una subclase de *owl:Class*, de este modo se garantiza que cualquier documento RDF válido es un documento OWL Full. Por el contrario, en el nivel Lite y DL, *owl:Class* está definido como una subclase de *rdfs:class*, y provoca que no todos los documentos RDF sean válidos en estos niveles.

- En el nivel OWL Full, *owl:ObjectTypeProperty* es considerado un equivalente de *rdf:Property*. Como consecuencia, *owl:DataTypeProperty*, que es una subclase de *rdf:Property*, es también una subclase de *owl:ObjectTypeProperty*. De este modo, cualquier propiedad OWL que sea definida como un tipo de datos, puede también ser interpretada como un *ObjectTypeProperty*. Esto provee más expresividad en este nivel. Por el contrario, en los otros dos niveles, *owl:DataTypeProperty* y *owl:ObjectTypeProperty* son también subclases del *rdf:Property* pero disjuntas. Por este motivo, las propiedades *InverseOf*, *TransitiveProperty*, *SymmetricProperty*, *FunctionalProperty*, *InverseFunctionalProperty* no pueden ser propiedades de un tipo de datos, debido a que una propiedad de un tipo de datos define una relación de una instancia a un literal.

En la tabla siguiente resumimos las diferencias entre los tres sublenguajes de OWL Lite, DL y Full:

OWL	OWL Lite	OWL DL	OWL Full
Compatibilidad RDF	No todos los documentos RDF son compatibles con OWL Lite	No todos los documentos RDF son compatibles con OWL DL	Todos los documentos RDF son sintácticamente y semánticamente compatibles con OWL Full
Restricciones en las definiciones	Hay una separación conceptual de lo que son clases, instancias, propiedades y tipos de datos	Hay una separación conceptual de lo que son clases, instancias, propiedades y tipos de datos	Las clases pueden ser a la vez instancias y propiedades
Operadores descriptores de clases	IntersectionOf	Enumeration UnionOf ComplementOf IntersectionOf	enumeration UnionOf ComplementOf IntersectionOf
Restricciones de cardinalidad	Restringida a un valor booleano (0 o 1) Cardinality MinCardinality: booleana MaxCardinality: booleana	Permite cualquier valor no negativo Cardinality MinCardinality MaxCardinality	Permite cualquier valor no negativo Cardinality MinCardinality MaxCardinality
Restricciones sobre valores	allValuesFrom someValueFrom	allValuesFrom someValueFrom hasValue	allValuesFrom someValueFrom hasValue
Permite la meta modelización	No	No	Sí

Fijaos que algunos de los operadores de OWL que aparecen en la tabla coinciden con los operadores descritos por la lógica descriptiva: intersección (*IntersectionOf*), unión (*UnionOf*) y complementario (*ComplementOf*) de conceptos; restricciones numéricas; nominales (*enumeration*); etc.

Para saber más

Podéis consultar con más detalle la diferencia de los diferentes lenguajes en el siguiente recurso web:

"OWL Web Ontology Language, Overview"

Aunque hay muchos factores que influirán a la hora de escoger el sublenguaje OWL más apropiado para una aplicación concreta, mostramos a continuación unas reglas muy simples que pueden ayudar a guiar la elección:

- La elección entre OWL Lite y OWL DL se puede basar en el hecho de si la sintaxis OWL Lite es suficiente o no para lo que necesitamos.

- La elección entre OWL DL y OWL Full se puede basar en el hecho de si es más importante llevar a cabo razonamiento automático en la ontología, o si por el contrario, conviene tener más capacidad de expresión y facilitar así la modelización del conocimiento (por ejemplo, si hay que definir clases de clases).

Ejemplo 41. Uso de la restricción *minCardinality*

Imaginémonos que tenemos una ontología que recaba información sobre la participación en los diferentes maratones del mundo. En esta figuran los corredores, los diferentes maratones y las marcas de tiempos conseguidas. Además, cada corredor que ha logrado el tiempo mínimo para poder participar en un maratón olímpico tiene tantas propiedades *hasMinOlimpicTime* como tiempos logrados. Para restringir la clase llamada *FutureOlimpicAthlete*, que agrupa aquellos corredores que cumplan los requisitos para participar en el maratón olímpico, habrá que indicar que todas sus instancias deben tener como mínimo 1 propiedad *hasMinOlimpicTime*. Esta restricción podrá ser expresada por medio del sublenguaje OWL Lite.

Ejemplo 42. Uso de la restricción *hasValue*

Imaginémonos ahora que queremos declarar una clase destinada a ser superclase de las competiciones deportivas masculinas –le llamaremos *Men's Competition*–; por lo tanto, hará falta que todas sus subclases tengan como restricción el valor *male* en la propiedad *gender* de sus participantes. Esta restricción solo podrá ser expresada por medio de los sublenguajes OWL DL u OWL Full.

Ejemplo 43. Recurso que es a la vez clase e instancia

En las olimpiadas de Londres 2012 se quiere representar todas las pelotas utilizadas en cada una de las competiciones. Así se crea una ontología que para cada pelota incluirá su marca, modelo y en qué disciplina es utilizada. Para que la ontología permita que *Molten GL7* identifique a la vez la clase de las pelotas utilizadas en cada uno de los partidos de baloncesto y una instancia de la clase *Pelotas de reglamento*, nos hará falta formalizarlo con el sublenguaje OWL Full.

4.4. Serialización

Como ya hemos visto, RDF es un modelo de datos, y no un formato en sí mismo, de manera que está desvinculado de una sintaxis en particular. El concepto **serialización**[32] se refiere a la representación de datos en un formato para almacenarlos, transportarlos o cualquier otro motivo. En este contexto hablaremos de serializaciones en formato textual, puesto que el objetivo es el

(32)En inglés, *serialization*.

procesamiento automático por parte de aplicaciones y no hace falta que sean legibles para los seres humanos. También se podría plantear una serialización en formato binario para reducir su tamaño.

En este subapartado veremos dos sintaxis para serializar datos RDF: RDF/XML y Notation3. Hay que decir, sin embargo, que existen otros, como por ejemplo Turtle[33], que es un subconjunto de Notation3, pero que no se verán en este módulo didáctico.

(33)De *Terse RDF Triple Language.*

4.4.1. RDF/XML

XML permite la existencia de documentos con cualquier tipo de información. Es un metalenguaje extensible de etiquetas (tal y como indica su nombre) desarrollado por el W3C. En esencia, la tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible.

Por eso, la serialización de datos RDF más común es la realizada en XML, donde RDF provee una sintaxis sobre XML para describir la representación de las tripletas conocida como **RDF/XML**.

Para saber más

Para conocer con más detalle la sintaxis del formato RDF/XML, podéis consultar el enlace web:

"RDF/XML Syntax Specification (Revised)"

A continuación listaremos las pautas para crear un documento RDF/XML.

Identificación del documento

En primer lugar, puesto que todo documento XML tiene que tener una única raíz, hay que añadir un nodo raíz RDF donde se indicará que se utiliza el espacio de nombres del W3.org llamado *rdf*. Este espacio de nombres informa de que el presente documento es RDF a cualquier mecanismo que lo lea, como también que la etiqueta *rdf:RDF* forma parte del mismo. A continuación mostramos cómo queda un documento con el nodo raíz:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <!-- Body Code Omitted -->
</rdf:RDF>
```

Añadir enunciados al documento

Un documento RDF puede contener más de un enunciado (tripleta), donde cada uno de ellos se agrega utilizando la etiqueta *rdf:Description*. Podemos añadir tantos enunciados como haga falta sobre el mismo sujeto.

La etiqueta *rdf:Description* utiliza el atributo *rdf:about* para indicar el identificador único del sujeto. En el código siguiente se añade un enunciado sobre un sujeto genérico de ejemplo identificado por Vocabulary.org.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
```

```
    <rdf:Description rdf:about="http://vocabulary.org/subj">
       <!-- Statement Code Omitted -->
    </rdf:Description>
</rdf:RDF>
```

Añadir predicados a los enunciados

Como hemos visto, los enunciados RDF describen las características de sus sujetos utilizando propiedades, también denominadas *predicados* en la terminología RDF.

Las propiedades se añaden como nuevos nodos que cuelgan del nodo *rdf:Description* del sujeto que se caracteriza, y contendrán la identificación de la propiedad, así como la del objeto del enunciado. Fijaos en que no es necesario identificar el sujeto de la propiedad, ya que es el que está definido en el nodo *rdf:Description*.

En cuanto al identificador de la propiedad, será el nombre del nodo, es decir, la etiqueta XML. Para facilitar su serialización, y especialmente su lectura se utilizará la forma abreviada por medio del espacio de nombres. Así, en el nodo raíz se indicará qué espacio de nombres se utilizará para abreviar la URIref.

A continuación, las propiedades pueden contener dos tipos de objeto: un literal o un identificador de recurso.

A modo de ejemplo, a continuación mostramos un código donde se ha añadido al sujeto de un enunciado dos propiedades en las que los objetos respectivos son: un literal XX y una referencia a un recurso http://vocabulary.org/subj. Conviene destacar que cuando hagamos referencia a un objeto recurso y no a un literal, su serialización no irá como valor tal y como lo hace el literal, sino como atributo del nodo de la propiedad.

Además de esto, se ha utilizado un espacio de nombres *terms*, declarado como atributo en el nodo raíz, *xmlns:terms="http://www.vocabulary.org/"*, para abreviar los identificadores únicos.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:terms="http://www.vocabulary.org/">
   <rdf:Description rdf:about="http://vocabulary.org/suj">
      <terms:pred_1>XX</terms:pred_1>
      <terms:pred_2 rdf:resource="http://vocabulary.org/obj"/>
   </rdf:Description>
</rdf:RDF>
```

Ejemplo 44. Serialización de la dirección de una clínica

A continuación mostramos cómo quedaría la serialización del ejemplo 35 que descomponía la dirección postal de una clínica en formato RDF:

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:vcard="http://www.w3.org/2006/vcard/ns">

   <rdf:Description rdf:about="http://www.dientesblancos.com">
      <vcard:adr>
         <vcard:street-address>C/ Mayor, 5</vcard:street-address>
         <vcard:postal-code>08001</vcard:postal-code>
         <vcard:locality>Barcelona</vcard:locality>
      </vcard:adr>
   </rdf:Description>
</rdf:RDF>
```

En el ejemplo anterior, se puede observar que uno de los nodos que representa la propiedad de dirección *vcard:adr* se describe mediante un conjunto de nodos hijos. Es decir, en la serialización de unos datos RDF pueden anidarse nodos que dependen de otros nodos tantas veces como sea necesario.

Puede pasar que un objeto de una propiedad del documento no haga referencia a ninguna URI y tenga que ser solicitado en diferentes puntos del documento (como sujeto u objeto de diferentes tripletas). En estos casos, es útil utilizar lo que se denomina *nodos blancos*. Un **nodo blanco** es un nodo que contiene un identificador por medio del atributo *rdf:nodeID*, relativo únicamente al documento en cuestión. Es decir, disponemos de un mecanismo para crear una URI local en el documento que nos sirva como identificador auxiliar.

Ejemplo 45

A continuación mostramos cómo quedaría la serialización del ejercicio "Serialización de una dirección de una clínica" utilizando un nodo blanco:

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:vcard="http://www.w3.org/2006/vcard/ns">

   <rdf:Description rdf:about="http://www.dientesblancos.com">
      <vcard:adr rdf:nodeID="adrdientesblancos"/>
   </rdf:Description>

   <rdf:Description rdf:nodeID="adrdientesblancos">
      <vcard:street-address>C/ Mayor, 5</vcard:street-address>
      <vcard:postal-code>08001</vcard:postal-code>
      <vcard:locality>Barcelona</vcard:locality>
   </rdf:Description>
</rdf:RDF>
```

Añadir los elementos para definir esquemas

Para la serialización de esquemas RDF haremos uso extensivo de dos atributos: *xml:base* y *rdf:ID*. El primero especifica la URI del contenido dentro de la etiqueta *rdf:RDF*. Esto quiere decir que todas las URIrefs del documento serán interpretadas con relación a ella, es decir, que no hay que escribir la dirección

absoluta, sino que se puede escribir la dirección relativa respecto a la *xml:base*. El segundo atributo, *rdf:ID*, define un nombre único relativo a la URI del documento, indicado por *xml:base*.

Ejemplo 46

Podéis consultar la serialización del ejemplo "Vocabulario de acontecimientos deportivos" en el epígrafe "Serialización del vocabulario de acontecimientos deportivos" del anexo del documento.

Dado que una clase RDFS es un recurso, podemos abreviar el ejemplo anterior utilizando el elemento *rdfs:Class* en lugar del *rdf:Description*, y eliminar la etiqueta de información *rdf:type*.

Ejemplo 47

Podéis consultar la serialización abreviada del ejemplo anterior en el apartado "Serialización abreviada del vocabulario de acontecimientos deportivos" del anexo del documento.

Añadir información legible para los humanos

El elemento *rdfs:comment* es una instancia de la *rdfs:Property* para añadir descripciones textuales a los recursos que hagan el documento comprensible para los humanos. El dominio de este elemento es un recurso (*rdfs:Resource*) y su rango es un literal (*rdfs:Literal*). Con las mismas propiedades también está el elemento *rdfs:label*, destinado a suministrar una etiqueta de versión.

Ejemplo 48. Comentarios y versionado en RDF

Añadimos un comentario y una versión al recurso clase que define la disciplina deportiva ciclismo.

```
<rdf:Description rdf:ID="Cycling">
   <rdfs:label>Cycling</rdfs:label>
   <rdfs:comment>The class of Cycling Discipline.</rdfs:comment>
   <rdf:type
      rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
   <rdfs:subClassOf rdf:resource="#SportDiscipline" />
</rdf:Description>
```

4.4.2. Notation3

Notation3 (N3) es otra forma de serializar datos RDF que no utiliza XML. Este formato tiene la finalidad de facilitar la lectura humana y la compactación, de modo que los documentos serializados tienen un aspecto muy diferente al que tendrían con RDF/XML.

Para saber más

Podéis consultar su sintaxis en el manual del siguiente enlace:

"Primero: Getting into RDF & Semantic Web using N3"

Así, un enunciado RDF (tripleta) quedaría de la siguiente forma:

Sujeto Predicado Objeto.

Ejemplo 49. Enunciado: "el coche de color azul"

```
<#coche> <#color> azul .
```

Todas las partes, sean sujeto, predicado u objeto, son identificadas con una URI. La única excepción es la parte objeto, que puede ser un literal. Además, en caso de que no haya una URI antes del símbolo #, como pasa en el ejemplo anterior con *#coche* y *#color*, puede referirse a cualquier elemento en el propio documento.

Para saber más

Encontraréis una *suite* de trabajo, con test y ejemplos en la siguiente dirección:

"Notation3 (N3): A readable RDF syntax"

La anotación también permite trabajar con espacios de nombres, que hay que indicar al inicio del documento para poderlos usar en el resto del documento. Para declararlos, se utilizará la misma distribución en tres elementos (como si fuera una tripleta):

```
@prefix namespace: <URI #> .
```

Ejemplo 50. Serialización en Notation3

A continuación mostramos cómo quedaría la serialización de la dirección de una clínica, que en el ejemplo 44 se mostraba en formato RDF/XML:

```
@prefix rdf <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix vcard <http://www.w3.org/2006/vcard/ns#> .

<http://www.dientesblancos.com#> <vcard:adr> <#adrdientesblancos> .
<#adrdientesblancos> <vcard:street-address> C/Mayor,5 .
<#adrdientesblancos> <vcard:postal-code> 08001 .
<#adrdientesblancos> <vcard:locality> Barcelona .
```

La gramática de Notation3 permite otras muchas funcionalidades, como por ejemplo las siguientes: diferentes formas de representar los datos para reducir texto y crear atajos, representar la semántica con operaciones, y creación de vocabulario entre otros.

Gramática de N3

Podéis consultar la gramática completa de Notation3 en el enlace siguiente:

"Notation 3 Logic"

4.5. Fragmentos de códigos HTML enriquecidos

El contenido semántico de la web no solo se encuentra en grandes ficheros RDF/XML. Al contrario, las páginas web están llenas de información válida que espera ser identificada como relevante. Esto requiere identificar de qué trata el documento web y qué fragmentos de información contiene.

Una forma de hacerlo es enriqueciendo los documentos web, identificando las estructuras para que los buscadores, las arañas[34] y los navegadores web puedan comprender el contenido y aprovecharlo para ofrecer nuevos servicios.

(34)En inglés, *web crawlers*.

Buscadores con entorno de test

Algunos buscadores también ofrecen un entorno de test para verificar el correcto marcaje. En el caso de Google, encontraréis las herramientas en el siguiente enlace:

"Herramientas para administradores web"

Los diferentes formatos que permiten llevar a cabo este enriquecimiento semántico de páginas web son: RDFa, Microdata y Microformatos, todos admitidos por los grandes buscadores que hay actualmente (Google, Bing y Yahoo!).

Tecnológicamente, estos formatos se componen de un conjunto de atributos de marcas que amplían la sintaxis de marcaje de documentos web. La finalidad de estos atributos es doble: por un lado se amplía el contenido visual de la web

con datos legibles por máquinas, y por el otro, se facilita la conectividad entre los datos de la web para no tenerlos aislados. Así pues, estamos ante una de las formas de publicar *linked data*.

4.5.1. RDFa

Resource Description Framework in attributes (RDFa) es un mecanismo que posibilita marcar datos RDF (enunciados con sujeto-predicado-objeto) dentro de documentos HTML de manera directa. En realidad no solo se puede aplicar a documentos HTML (HTML4 y HTML5), sino también a cualquier otro documento basado en el formato XML, como son XHTML, XML y SVG. A la vez, se posibilita la extracción de esta información estructurada (tripletas).

RDFa en XHTML

RDFa se convierte en octubre del 2008 en una recomendación del W3C:

"RDFa in XHTML: Syntax and Processing"

Como ya hemos ido mencionando, la esencia de RDFa es el conjunto de atributos que permiten añadir metadatos a los lenguajes de marcas, y que a continuación expondremos. Antes de continuar, hay que matizar que los ejemplos expuestos en este subapartado serán exclusivamente en HTML, pero como ya podéis suponer, son ampliables a cualquier otro lenguaje basado en XML.

Para saber más

Para consultar toda la información relacionada con el uso y la definición del RDFa (herramientas, sintaxis, desarrollos, etc.), podéis consultar el siguiente enlace:

"Linked Data in HTML"

La idea principal de RDFa es marcar lo que se denomina **entidad** o **ítem** con un tipo de elemento y, para cada una de sus propiedades, con los valores respectivos.

Ejemplo 51. Ítems en una página web

En una página web podemos identificar un acontecimiento, una crítica culinaria, una persona, una empresa. Cada ítem marcado podrá contener la información de las propiedades (los ítems pueden estar anidados). Por ejemplo, de una persona, se puede marcar su nombre, su dirección web, su titulación, entre otros.

Los atributos básicos para el marcaje de datos RDF son: *vocab*, *typeof*, *property* y *resource*. Estos constituyen un subconjunto de los atributos totales que proporciona el RDFa Core, que podéis conocer y ampliar consultando los enlaces recomendados.

a) *vocab*: http://schema.org/ es un recurso web que define vocabularios sobre las cosas comunes en la web: personas, lugares, acontecimientos, recetas, ... impulsados por las empresas de los grandes buscadores web. Antes de marcar información dentro del documento web, habrá que indicar qué vocabulario se va a utilizar. Este atributo puede contener por ejemplo http://schema.org. Alternativamente a este atributo, también se puede utilizar la marca de espacio de nombres *xmlns* para indicar cómo localizar el vocabulario que se va a emplear.

b) *typeof*: Este atributo indica el tipo del sujeto que se declara, es decir, de qué se habla.

c) *property*: Este atributo permite especificar las propiedades del sujeto.

Ejemplo 52. Uso de RDFa

Enriquecemos el siguiente fragmento de código HTML para estructurar los datos que contiene para que puedan ser procesados automáticamente.

Fragmento de código HTML sin marcaje de datos:

```
<div>
   <strong>
      <span>Clínica Dientes Blancos</span>
   </strong>
   <ul>
      <li>Teléfono:
         <span>553443333</span>
      </li>
      <li>Dirección:
         <span>C/ Mayor, 5</span>
         <span>Barcelona</span>
      </li>
   </ul>
   <a href="http://ortodoncia.wordpress.com">
      http://ortodoncia.wordpress.com
   </a>
</div>
```

Fragmento de código HTML marcado con RDFa:

```
<div vocab="http://schema.org/" typeof="Organization">
   <strong>
      <span property="name">Clínica Dientes Blancos</span>
   </strong>
   <ul>
      <li>Teléfono:
         <span property="telephone">553443333</span>
      </li>
      <li property="address" typeof="PostalAddress"
         vocab="http://schema.org/">Dirección:
         <span property="streetAddress">C/ Mayor, 5</span>
         <span property="addressLocality">Barcelona</span>
      </li>
   </ul>
   <a property="url" href="http://ortodoncia.wordpress.com">
      http://ortodoncia.wordpress.com
   </a>
</div>
```

Es importante recalcar que el ejemplo anterior se puede marcar de diferentes maneras, y todo dependerá de qué vocabulario utilizamos para hacerlo.

Ved también

Podéis ver otras alternativas de marcaje en el ejemplo 3 del subapartado 1.3, y en los dos subapartados 4.5.2 y 4.5.3 siguientes.

d) ***resource***: Sirve para identificar de manera única el ítem del cual se trata en un documento. Consecuentemente, desde otro documento web podremos hacer referencia a él de manera única.

Ejemplo 53

El siguiente fragmento de código HTML muestra cómo identificar un ítem de manera única.

```
<p vocab="http://schema.org/" resource="#john" typeof="Person">
   My name is <span property="name">John Smith</span>
   and my telephone number is
   <span property="telephone">612 550 221</span>
</p>
```

4.5.2. Microdatos

Los microdatos[35] proporcionan al HTML la posibilidad de contener información semántica por medio del uso de los propios atributos de las etiquetas HTML. El marcaje consiste en agrupar conjuntos de propiedades (parejas de nombre-valor) llamados *ítems*.

(35)En inglés, *microdata*.

Para saber más

Para conocer más sobre las especificaciones W3C de los microdatos, podéis consultar el siguiente enlace:
"HTML Microdata"

Los atributos utilizados son:

- **itemscope**: Atributo booleano que indica un ítem, y por lo tanto, estará en un bloque que agrupe un conjunto de propiedades.

- **itemtype**: Atributo que acompaña el *itemscope* para indicar el tipo de información que encontraremos en sus propiedades. Hace referencia a un diccionario o vocabulario por medio de una dirección URL.

- **itemid**: Atributo utilizado para dar un identificador único a un ítem.

- **itemref**: Atributo que permitirá hacer referencia a un ítem dentro del propio documento.

- **itemprop**: Atributo para definir las propiedades, el nombre y el valor que se quiere asignar.

Ejemplo 54

Vamos a enriquecer el mismo fragmento de código HTML del ejemplo 52 (Uso de RDFa), esta vez mediante el uso de microdatos:

```
<div itemscope itemtype="http://schema.org/Organization">
   <strong>
      <span itemprop="name">Clínica Dientes Blancos</span>
   </strong>
   <ul>
      <li>Teléfono:
         <span itemprop="telephone">553443333</span>
      </li>
      <li property="address" itemscope
         itemtype="http://schema.org/PostalAddress">Dirección:
         <span itemprop="streetAddress">C/ Mayor, 5</span>
         <span itemprop="addressLocality">Barcelona</span>
      </li>
   </ul>
   <a itemprop="url"
      href="http://ortodoncia.wordpress.com">
         http://ortodoncia.wordpress.com
   </a>
</div>
```

4.5.3. Microformatos

Los microformatos[36] son una extensión de HTML para publicar información sobre entidades tan habituales y comunes como son personas, organizaciones, acontecimientos, ubicaciones, entradas a blog, productos, opiniones, resúmenes, recetas, etc. Esta publicación se hace por medio de pequeños patrones de código HTML que describen entidades con sus propiedades.

(36)En inglés, *microformats*.

Para saber más

Para conocer mejor lo que son los microformatos, podéis consultar el siguiente recurso web:
"About Microformats"

Microformatos

"Designed for humans first and machines second, microformats are a set of simple, open data formats built upon existing and widely adopted standards."

Dan Cederholm y Tantek Çelik, "microformats.org" (2005)

Técnicamente, los microformatos hacen uso de los atributos *class* y *rel* de las etiquetas HTML para asignar nombres a las entidades y sus propiedades sobre un HTML muy estructurado pensado para ser leído por un humano. Como los microformatos usan atributos estándares de HTML que ya son admitidos por todas las aplicaciones de la web, esta aproximación tiene la ventaja de no "romper" ninguna aplicación existente.

La lista de los microformatos disponibles en la actualidad en una versión estable incluye, entre otros:

- *hCalendar*: acontecimientos
- *hCard*: personas, organizaciones y contactos
- *rel-license*: licencias del contenido
- *rel-nofollow*: enlaces que no se confían de contenidos de terceros
- ...

Ejemplo 55

Vamos a enriquecer el mismo fragmento de código HTML del ejemplo 52, esta vez mediante microformatos:

```
<div class="vcard">
   <strong>
      <span class="fn org">Clínica Dientes Blancos</span>
   </strong>
   <ul>
      <li>Teléfono:
         <span class="tel">553443333</span>
      </li>
      <li class="adr">Dirección:
         <span class="street-address">C/ Mayor, 5</span>
         <span class="locality">Barcelona</span>
      </li>
   </ul>
   <a class="url" href="http://ortodoncia.wordpress.com">
      http://ortodoncia.wordpress.com
   </a>
</div>
```

5. Recursos

En este último apartado, queremos presentar algunos de los recursos y herramientas disponibles para manipular información semántica. En primer lugar vamos a ver los editores de ontologías más populares de hoy en día. Posteriormente, vamos a estudiar los entornos de trabajo que hay y los almacenes de tripletas que permiten acceder y manipular la información semántica. Finalmente, veremos qué aportan los razonadores y qué tipos de razonadores existen.

5.1. Editores de ontologías

El número de herramientas para la manipulación de ontologías es realmente muy grande. Muchas de ellas son iniciativas particulares que provienen de centros de investigación universitarios, pero también las hay de empresas privadas, como pueden ser industrias farmacéuticas. Esto ha provocado una proliferación de herramientas libres, pero como veremos, también privadas.

Editores de ontologías

El W3C mantiene una lista de editores en el siguiente recurso web:
"Ontology editors"

A continuación compararemos un conjunto de estas herramientas (Protégé, TopBraid Composer, OntoStudio, OilEd y Swoop) basándonos en criterios como, por ejemplo, el propósito, la licencia de uso, la escalabilidad, la documentación, los formatos que admiten, etc. La selección de estos editores se ha hecho principalmente sobre la base de su popularidad.

5.1.1. Protégé

Protégé es uno de los editores de ontologías más utilizados y apoyado por una gran comunidad de usuarios. Se trata de un editor libre de código abierto y multiplataforma desarrollado por la Universidad de Stanford, en concreto por el Centro Stanford para la Investigación Informática en Biomedicina (Center for Biomedical Informatics Research).

Para saber más

Podéis encontrar más información sobre Protégé en el siguiente recurso web:
"Protégé"

El editor ofrece una serie de herramientas para modelizar (crear, visualizar y manipular) el conocimiento en forma de ontologías destinadas a representar dominios o conocimiento para aplicaciones. Además, permite la exportación a una gran variedad de formatos, como RDF, RDF Schema, OWL y XML Schema.

El editor permite dos maneras de modelizar conocimiento: construyendo ontologías basadas en marcos[37] o bien construyendo ontologías destinadas a la web semántica, particularmente con el formato OWL.

(37) En inglés, *frames*.

Una de las características que hay que destacar de Protégé es su escalabilidad y ampliabilidad, que ofrece la posibilidad de adaptar el editor a nuestras necesidades para trabajar con ontologías. Por ejemplo, la ampliabilidad permi-

(38) En inglés, *plugins*.

te desde crear interfaces específicas hasta ejecutar cualquier proceso que las manipule: fusiones, inferencias, etc. Para hacerlo, pone a disposición dos mecanismos: una arquitectura de extensiones[38] basadas en Java y una API que puede ser llamada desde un programa externo.

Además, hace falta mencionar otras características, como son su robustez a la hora de construir y procesar de manera eficiente ontologías de gran volumen, edición colaborativa, y la gestión de versiones.

Para saber más

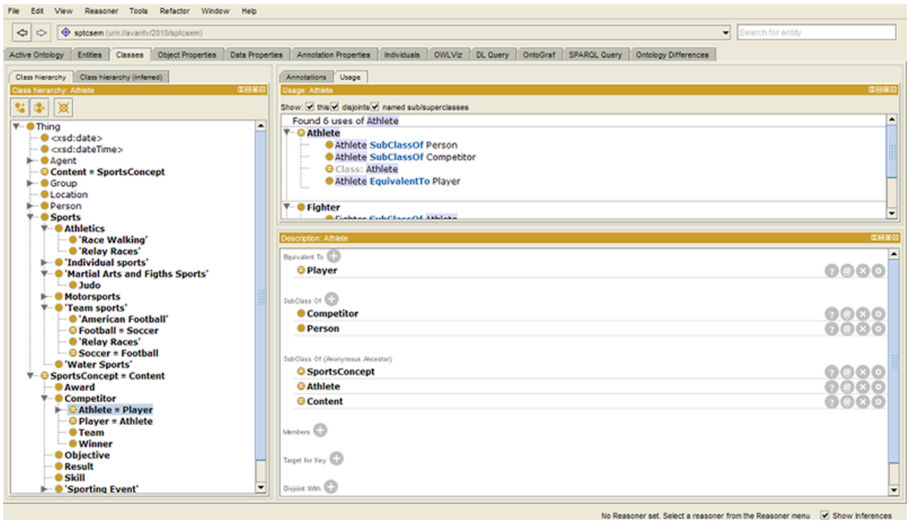
Para conocer qué extensiones están disponibles, podéis consultar el recurso web:

"Welcome to the Protege Plugin Library!"

Para conocer con más detalle cómo utilizar el API de Protégé, podéis consultar el siguiente recurso web:

"Protege-OWL API Programmer's Guide"

Figura 9. Captura de pantalla del editor Protégé



5.1.2. TopBraid Composer

TopBraid Composer es un entorno para modelizar datos especialmente destinado al desarrollo de ontologías basándose en los estándares W3C. El editor admite los formatos OWL-RDF/XML, N3 y N-Triples.

Para saber más

Podéis consultar más en detalle el editor TopBraid Composer en el recurso web:

"TopBraid Composer"

Es un software propietario de la empresa TopQuadrant (USA) basado en la plataforma Eclipse[39] y Apache Jena[40]. Se puede descargar en tres distribuciones diferentes, Free Edition, Standard Edition y Maestro Edition:

a) Free Edition: Es la única distribución gratuita, exclusiva para usos no comerciales y sin ningún tipo de coste. Por eso es una distribución de funcionalidades muy básicas.

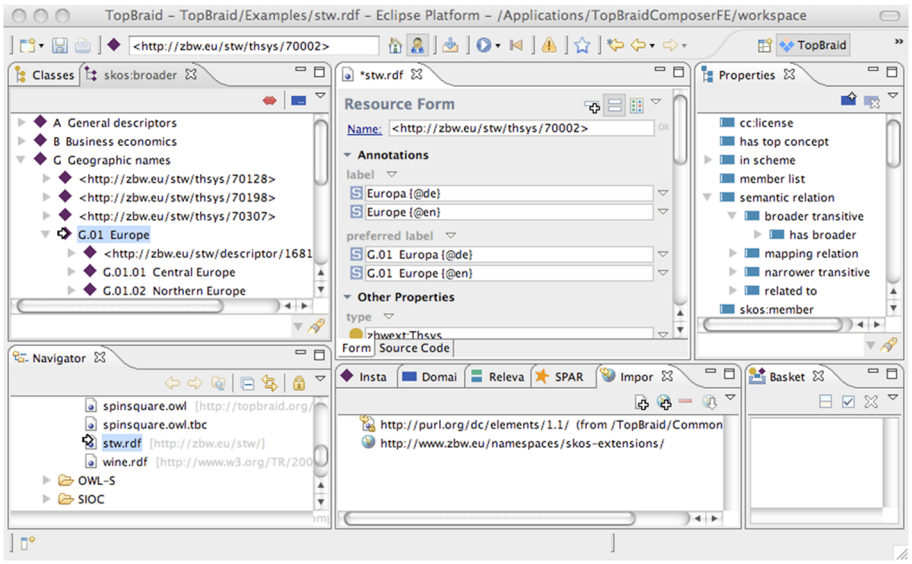
b) Standard Edition: Incluye todas las características de la versión Free Edition y además permite la edición con interacción visual, visualización gráfica, facilidades en la importación, razonadores, conectividad con bases de datos RDF, etc.

c) Maestro Edition: Incluye todas las características de las distribuciones anteriores, además de ofrecer una plataforma para el desarrollo de aplicaciones y servicios web.

(39)Eclipse es un entorno de desarrollo integrado (IDE) de código abierto multiplataforma. Podéis encontrar más información en el recurso web Eclipse.org.

(40)Apache Jena es un entorno de trabajo de Java para la construcción de aplicaciones para la web semántica. Incluye mecanismos para el tratamiento de formatos tales como RDF y OWL, y también razonadores y mecanismos de consulta, y es eficiente en la gestión de grandes volúmenes de datos.

Figura 10. Captura de pantalla del editor TopBraid Composer



5.1.3. OntoStudio

OntoStudio es otro editor propietario de la empresa Ontoprise GmbH (Alemania), dedicada a las tecnologías semánticas. El editor es de pago pero se ofrece una versión libre de tres meses para su evaluación.

Este editor está basado en una arquitectura cliente-servidor que permite el trabajo colaborativo, que centraliza las ontologías en un servidor y permite el acceso de múltiples clientes (editores). Hay una versión cliente basada en web, Web OntoStudio, que permite trabajar mediante un navegador y ofrece un subconjunto de las funcionalidades básicas.

Los formatos aceptados para modelizar el conocimiento son: RDF, RDF Schema, RIF y ObjectLogic, pero también tiene la capacidad de importar otros formatos, como son UML, Database Schemas (Oracle, DB2, MySQL, MS-SQL), tablas de Excel, correos electrónicos y estructuras de directorios.

Para saber más

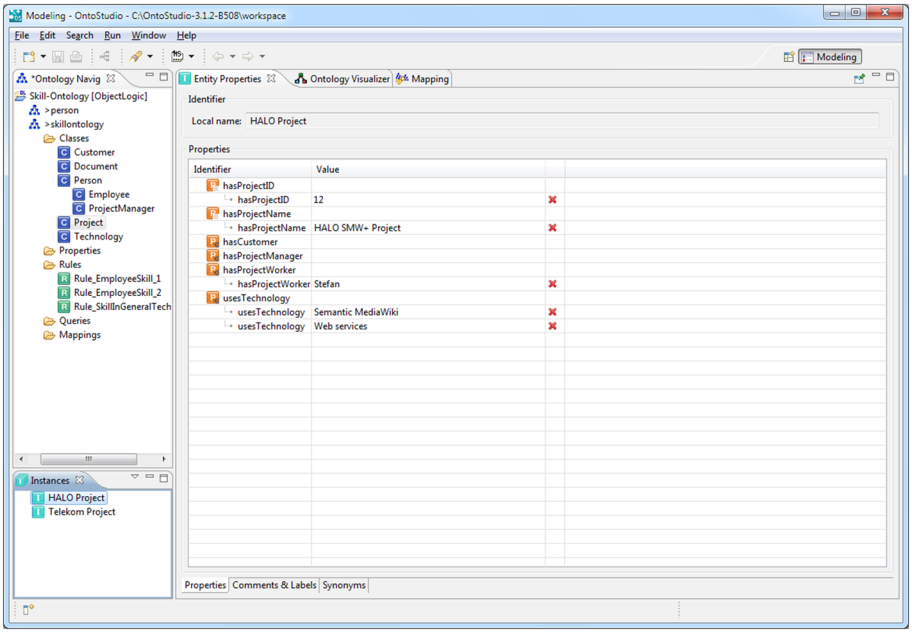
Podéis consultar más sobre OntoStudio y el lenguaje ObjectLogic en el siguiente recurso web

"Semafora helpsystem: semafora WebHelp"

ObjectLogic es un lenguaje de bases de datos orientado a objetos que combina la semántica declarativa y la expresividad de las bases de datos deductivos con objeto de modelizar conocimiento compatible con modelos orientados a objetos.

Otras características que hay que destacar son la posibilidad de trabajar con un API y la herramienta de inferencia, OntoBroker.

Figura 11. Captura de pantalla del editor OntoStudio



5.1.4. OilEd

OilEd es un editor de ontologías en código abierto (licencia GPL) que en actualmente ya no tiene mantenimiento. Fue desarrollado por Sean Bechhofer and Gary Ng de la universidad de Manchester.

Para saber más

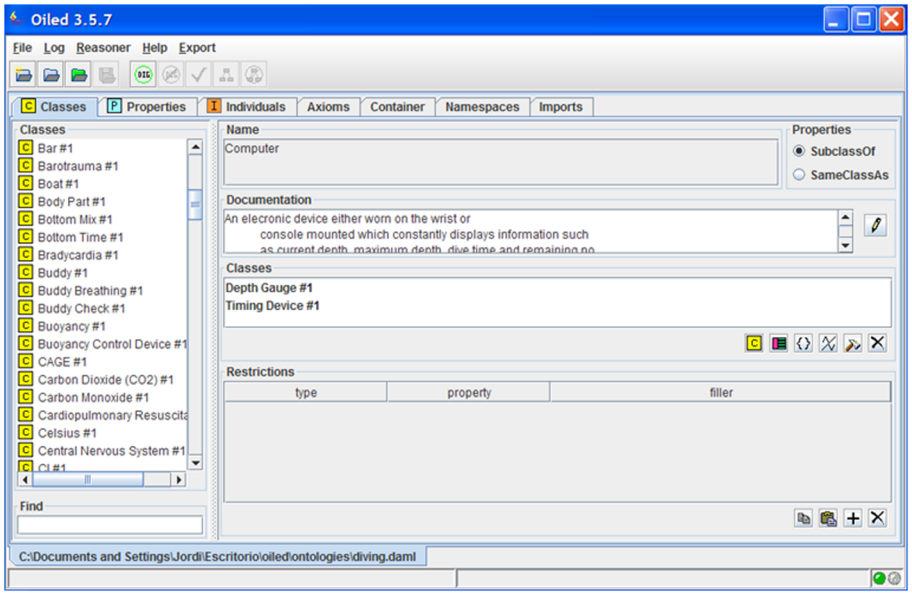
Podéis encontrar más información de OilEd en el siguiente recurso web:

"The OilEd web site"

Podéis consultar con más detalle acerca del editor en el siguiente artículo:

Sean Bechhofer; Ian Horrocks, Carole Goble; Robert Stevens. "OilEd: a Reasonable Ontology Editor for the Semantic Web"

Figura 12. Captura de pantalla del editor OilEd



El editor trabaja con el formato DAML-OIL, pero permite la importación de formatos como OWL, RDF (XML), GO, OIL-Text y SHIQ Knowledge Bases, y del mismo modo también tiene una amplio abanico de formatos de exportación.

Este es un editor muy simple que está centrado en la expresividad y el razonamiento de las ontologías. Es decir, propone una plataforma destinada a demostrar cómo los esquemas de representación basados en marcos pueden mejorar la expresividad en la modelización de conocimiento por medio de lenguajes, y cómo el razonamiento puede apoyar al diseño y el mantenimiento de las ontologías.

Por el contrario, no permite el trabajo colaborativo, el versionado, la integración o la fusión con otras ontologías.

El formato DAML-OIL

OIL (*Ontology Inference Layer*) es una extensión del RDF Schema y fue uno de los primeros formatos para la representación de ontologías en la WS; utiliza una sintaxis basada en marcos.

DAML (*DARPA Agent Mark-up Language*) es otro lenguaje para la definición de ontologías similar al RDF Schema.

Los dos lenguajes se fusionaron en un único lenguaje llamado **DAML-OIL**.

Podéis encontrar más información en el siguiente recurso web:

"The DARPA Agent mark-up Language Homepage"

5.1.5. Swoop

Para saber más

Podéis consultar más sobre el editor Swoop en el siguiente recurso web:

"Swoop", *Google*

Swoop es a la vez un navegador y un editor de ontologías de código libre basado en el formato OWL que tiene origen en el MIND Lab de la Universidad de Maryland. Inicialmente concebido como un navegador web de ontologías, Swoop pretende aprovechar la misma naturaleza de la web semántica (abierta, distribuida y escalable) para desarrollar y compartir conocimiento.

Swoop

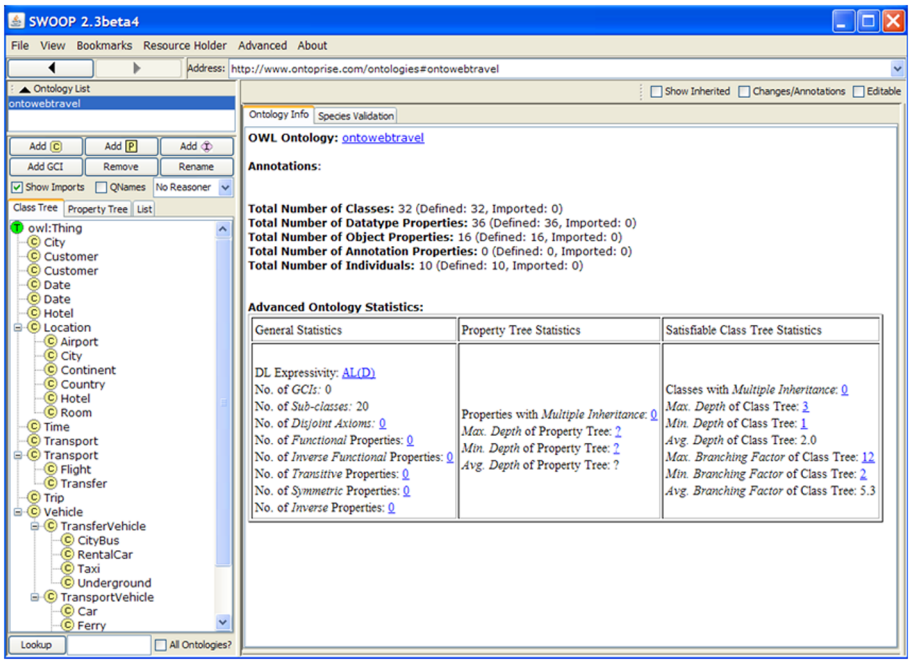
"Herramienta de desarrollo de la web semántica para escalar fácilmente múltiples ontologías con la finalidad de facilitar las tareas de creación, navegación, edición, búsqueda, enlace, unión y división de ontologías en OWL [...]."

Descrito por: B. Cuenca Grau; U. Sattler; S. Tessaris; A. Y. Turhan (2009). *Software Tools for Ontology Integration and Merging Deliverable TONES-D22*.

Los usuarios pueden hacer uso de ontologías externas enlazándolas, o bien importándolas (completamente). No es posible hacer importaciones parciales, pero sí que se posibilita la búsqueda de conceptos a través de las múltiples ontologías importadas o enlazadas que se tengan.

Finalmente, cabe añadir que el editor ofrece la posibilidad de utilizar diferentes sintaxis de visualización: Abstract Syntax, N3.

Figura 13. Captura de pantalla del editor Swoop



Para dar una visión de conjunto de los diferentes editores que pueda ayudar a seleccionar el más apropiado para un proyecto concreto de representación del conocimiento, presentamos a continuación una tabla resumen que compara las herramientas presentadas:

	Protégé	**TopBraid Composer**	**OntoStudio**	**OilEd**	**Swoop**
Licencia	Código abierto	Software propietario	Software propietario	Código abierto	Código abierto
Arquitectura	Programa de escritorio	Programa de escritorio, extensión de Eclipse	Programa de escritorio y versión web restringida	Programa de escritorio	Programa de escritorio
Arquitectura WS	Cliente-servidor	Cliente-servidor	Cliente-servidor	–	Cliente-servidor
Ampliable	Extensiones	Extensiones	Extensiones	Extensiones	Extensiones
Formatos de exportación	XML, XML Schema, RDF, RDF Schema, OWL, Clips, SWRL-IQ, Instance Selection, MetaAnalysis y OWLDoc	RDF, RDF Schema, XML, XML Schema, N3, N-Triples,	XML, XML Schema, OWL, RDF, RDF Schema, UML y OpenXML	RDF, RDF Schema, DAML-OIL, SHIQ, HTML, Dotty, FaCT Lisp, FaCT++ Lisp, DIG, OWL	OWL, XML, RDF y otros formatos de texto
Formatos de importación	XML, XML Schema, RDF, RDF Schema, OWL, Excel, DB Relacional (DataMaster), BioPortal	RDF, RDF Schema, XML, XML Schema, N3, N-Triples, hojas de cálculo, UML, RSS/Atom Feeds, BD relacionales, correo electrónico, XHTML	XML, XML Schema, OWL, RDF, RDF Schema, UML, DB schemas (Oracle, MS-SQL, DB2, MySQL), Outlook, sistemas de ficheros y Remote OntoBroker	OWL, RDF (XML), GO, OIL-Text y SHIQ Knowledge Bases	OWL, XML, RDF y otros formatos de texto
Esquemas de representación del conocimiento	Marcos, esquemas lógicos, metaclases	RDF, OWL	Marcos, esquemas lógicos	Marcos	OWL

* Limitado únicamente a la compartición de anotaciones.

	Protégé	TopBraid Composer	OntoStudio	OilEd	Swoop
Contiene una herramienta de inferencia	Sí	Sí	Sí (OntoBroker)	Sí	Sí
Permite el trabajo colaborativo	Sí	Sí	Sí	No	Sí*

* Limitado únicamente a la compartición de anotaciones.

5.2. Entornos de trabajo y *triplestores*

A medida que la tecnología de la web semántica avanza, su uso se extiende y surge la necesidad de gestión, tratamiento, recuperación y almacenamiento eficiente de conjuntos de datos RDF. Esto ha llevado a la aparición de un número considerable de herramientas que cubren una parte o la totalidad de estas necesidades.

Estas herramientas se pueden catalogar según las funcionalidades que ofrecen. Según este criterio, las podemos clasificar en dos grupos: **entornos de trabajo** y ***triplestores***. Los entornos de trabajo para la web semántica se orientan a proveer API de programación para el procesamiento del RDF. Por otro lado, las ***triplestores*** son sistemas de gestión de bases de datos (SBGD) para el almacenamiento y la recuperación de tripletas RDF, es decir, son SGBD semánticos. Hay que tener en cuenta que la variedad de funcionalidades que cada vez más incorporan estas herramientas vuelven más difusa esta clasificación.

Como hemos dicho, los entornos de trabajo son API para el desarrollo de aplicaciones que requieren funcionalidad para la manipulación de contenido RDF (grafos) en memoria. Otras funciones que ofrecen son la extracción y la escritura de grafos RDF, con la posibilidad de diferentes formatos para la serialización.

Una herramienta *triplestore* está destinada al almacenamiento y consulta de datos RDF. A diferencia de los entornos de trabajo, posee un mecanismo para la persistencia y acceso de los datos. La aparición de cada vez más herramientas de esta tipología está acompañada por iniciativas que desarrollan tecnologías semánticas de consulta y protocolos de acceso. Un *tiplestore* puede ser categorizado de tres maneras según su arquitectura:

- ***in-memory***: Guarda las tripletas del grafo RDF en la memoria principal. Tienen la ventaja de efectuar ciertas operaciones con mucha eficiencia, como por ejemplo la inferencia. Por el contrario, no son herramientas pensadas para guardar grandes volúmenes de datos. Aquí incluiríamos los entornos de trabajo, que se pueden ver también como un *in-memory triplestore*. Por ejemplo: Jena, Sesame.

- **nativa**: Provee persistencia en el almacenamiento de los datos por medio de su propia implementación. Es la categoría más dominante en términos de herramientas implementadas. Por ejemplo: Sesame, Virtuoso, OWLIM.

- **no nativa y no *in-memory***: Son herramientas pensadas para el almacenamiento persistente de tripletas en bases de datos de terceros. Añaden una capa intermedia para utilizar las bases de datos relacionales, como MySQL, PostgreSQL u Oracle. Por ejemplo: Jena SDB.

A continuación detallaremos cada una de las herramientas anteriormente indicadas como ejemplo. Hay que remarcar, que teniendo en cuenta que son herramientas que cambian y están en constante evolución, la página web de cada herramienta ofrecerá siempre la información más actualizada.

5.2.1. Jena

Actualmente, Jena es un proyecto de Apache en código abierto. Se trata de una API para Java especializada en trabajar con RDF y OWL, tanto por lo que respecta a su manipulación, como por el razonamiento e inferencia sobre ellas. La API ofrece mecanismos para publicar datos RDF para otras aplicaciones utilizando varios protocolos, entre ellos SPARQL. Además, diferentes componentes permiten trabajar sobre bases de datos relacionales, ya sea tanto para el almacenamiento de las tripletas como para hacer consultas SQL sobre las tripletas.

Para saber más

Para conocer mejor la herramienta Jena, podéis consultar el recurso web:

"Apache Jena"

5.2.2. Sesame

Sesame es un entorno de trabajo de código abierto para el almacenamiento, la inferencia y la consulta de datos RDF desarrollado por la empresa alemana Aduna. Como aspectos destacables de esta, podemos mencionar que tiene las mismas funcionalidades que Jena, pero además ofrece un almacenamiento de tripletas propio y una API muy extensa.

Para saber más

Para conocer mejor la herramienta Sesame, podéis consultar el recurso web:

"openrdf.org/"

5.2.3. Virtuoso

Tal y como se ha indicado, es una herramienta *triplestore* nativa que se ofrece tanto en código abierto como con una licencia comercial. A diferencia de Sesame y Jena, su API no es muy extensa. Pero por otro lado, es especialmente recomendada para manejar grandes cantidades de tripletas. Además, Virtuoso ofrece un puente para ser utilizado con Jena y Sesame.

Para saber más

Para conocer mejor la herramienta Virtuoso, podéis consultar el recurso web:

"Virtuoso Universal Server"

5.2.4. OWLIM

OWLIM es otra *triplestore* implementada en Java que está disponible en tres versiones: la gratuita OWLIM Lite, la comercial OWLIM SE y la avanzada OWLIM Enterprise. La versión Lite es *in-memory* y por lo tanto está limitada al tamaño de los conjuntos de datos que puede soportar. La versión SE añade almacenamiento nativo, mientras que la versión Enterprise permite la ejecución de la *triplestore* en un clúster para poder replicar los datos y ejecutar tareas en paralelo.

Para saber más

Para conocer mejor la herramienta OWLIM, podéis consultar el recurso web:
"OWLIM"

5.3. Razonadores para la web semántica

Igual que en el caso de los editores de ontologías, hay un gran número de herramientas disponibles para realizar tareas de consulta, inferencia y razonamiento sobre ontologías. Algunos ejemplos de tareas que pueden ser resueltas mediante un razonador son las siguientes:

a) Satisfactibilidad: Comprobar si una clase de ontología se puede llegar a instanciar.

b) Subsunción: Comprobar si alguna restricción de la ontología queda implícita por el resto de restricciones.

c) Clasificación

d) Calcular explícitamente el conjunto de subclases de cada clase, es decir, determinar si hay subclases implícitas.

e) Calcular si dos conceptos son sinónimos.

f) Calcular la clase más específica a la que pertenece una instancia.

g) Comprobar si las instancias definidas dentro de una ontología satisfacen todas las restricciones.

h) Respuesta a consultas: Comprobar qué instancias de la ontología satisfacen una cierta consulta. Generalmente se imponen restricciones sobre el formato que puede tener esta consulta, como por ejemplo una condición expresada como una conjunción de restricciones[41] que hay que satisfacer o una consulta en un lenguaje como SPARQL.

(41)En inglés, *conjunctive query*.

Muchas de estas herramientas tienen su origen en el ámbito de académico y han surgido a partir de grupos o proyectos de investigación y desarrollados desde la universidad. A pesar de ello, muchos han acabado teniendo una vertiente comercial y ya están siendo desarrollados y mantenidos desde empresas.

Las funcionalidades concretas ofrecidas por cada herramienta evolucionan en cada nueva versión. Más que ofrecer una descripción exhaustiva de estas características en las versiones en curso de estas herramientas, a continuación repasamos cuáles son los criterios que hay que tener en cuenta a la hora de elegir una herramienta para razonar con ontologías:

Listado de razonadores

Podéis encontrar un listado actualizado de razonadores para lógica descriptiva y ontologías en la URL siguiente:
"Description logic reasoners"

a) Licencia: Hay diversidad de licencias para conseguir y utilizar estas herramientas, sin coste o de pago.

- **Gratuita**: Los razonadores que tienen un origen académico suelen publicarse utilizando una licencia gratuita de software abierto, como puede ser GPL o LGPL.

- **Usos científicos**: En otros casos, no se ofrece una licencia de software abierto, pero sí que se permite su uso gratuito para aplicaciones académicas o de investigación.

- **Comercial**: Por último, los razonadores desarrollados por empresas se ofrecen con licencias de pago que permiten utilizar el razonador para aplicaciones con afán de lucro. Generalmente, en el precio de la licencia está incluido un servicio de mantenimiento. En algunas herramientas, hay un periodo de prueba gratuito de duración y/o funcionalidades limitadas.

b) Expresividad: Cada razonador admite un lenguaje de lógica descriptiva diferente, con un conjunto de operadores diferentes. Usualmente, estos lenguajes no coinciden perfectamente con los lenguajes de OWL: hay partes del estándar OWL que no se admiten y se añaden extensiones propias. Por este motivo, es muy importante revisar la descripción del razonador y su manual de uso para entender bien qué tipo de propiedades se admiten.

c) Eficiencia: Hay muchas estrategias y optimizaciones para acelerar el proceso de inferencia y cada herramienta opta por implementar algunas. Esto es lo que condiciona la expresividad del razonador y hace que haya herramientas más apropiadas para determinados tipos de consultas.

d) Interfaces de acceso: Los razonadores pueden ofrecer varios canales para acceder a sus funcionalidades:

- **Línea de comandos**: Hay un conjunto de comandos sencillos para indicar al razonador los ficheros donde está almacenada la ontología y el tipo de inferencia que se quiere realizar, generando una salida en formato textual o bien en un fichero.

- **Interfaz gráfica** (**GUI**): El razonador dispone de una interfaz gráfica propia que permite visualizar la ontología y ejecutar razonamientos.

- **API**: El razonador ofrece una interfaz de funciones, clases y/o métodos que pueden ser invocadas desde un programa para cargar una ontología, razonar y recabar el resultado. Generalmente, cada herramienta ofrece la API para algún lenguaje concreto, que acostumbra a ser el mismo que se ha utilizado para implementar la herramienta (por ejemplo, Java, C++ o Lisp). Además de la interfaz propia que pueda proporcionar cada razonador, la mayoría también implementa interfaces genéricas, como por ejemplo OWLAPI[42], DIG[43] o las interfaces de algún entorno de trabajo. El objetivo en este caso es mejorar la modularidad de las aplicaciones de web semántica y facilitar la sustitución de un razonador por otro sin tener que hacer modificaciones en el código.

- **Integración en un editor de ontologías**: Las funcionalidades del razonador se pueden invocar desde la interfaz gráfica de un editor de ontologías, es decir, el razonador se puede añadir como ampliación (*plug-in*) del editor.

(42)OWLAPI es una API Java para crear, manipular y serializar ontologías OWL. Para saber más: The OWL API.

(43)DIG (*description logic implementation group interface*) es una interfaz web (vía peticiones HTTP) para acceder a razonadores de lógica descriptiva:
"The new DIG interface standard (DIG 2.0)"

e) Madurez y mantenimiento: Hay una gran diversidad de madurez entre los razonadores existentes. Algunos razonadores académicos son simplemente prototipos que sirven como prueba de concepto y, por lo tanto, no están pensados para ser utilizados en proyectos reales. Esto significa que es importante comprobar si hay una comunidad de usuarios detrás de la herramienta y si se hace un mantenimiento de la herramienta, ya sea añadiendo nuevas funcionalidades o publicando nuevas versiones que corrigen errores.

Como ejemplo, ilustramos a continuación algunos de los razonadores sobre ontologías más conocidos y evaluamos algunas de las dimensiones que acabamos de explicar. Dentro de esta evaluación, destacamos la falta de soporte en el caso de OWL Full entre los diferentes razonadores, como ya habíamos avanzado al describir los diferentes sublenguajes de OWL.

Razonador	Licencia	Escrito en	Expresividad	Interfaces	Última versión
HermiT OWL Reasoner	Código abierto (LGPL)	Java	**OWL2**: admitido	Línea de comandos OWLAPI (Java) *Plug-in* Protégé	1.3.7 (Marzo 2013)
Pellet	Código abierto (AGPL) Comercial	Java	**OWL Lite**: admitido **OWL DL**: admitido **OWL Full**: sólo propiedades "inverso" **OWL 2**: admitido	Línea de comandos OWLAPI (Java) DIG (HTTP) Jena (Java) *Plug-in* Protégé	2.3 (Agosto 2011)
RacerPro	Usos científicos Comercial Prueba gratuita	Lisp	**OWL Lite**: admitido **OWL DL**: admitido sin nominales ni tipos de datos definidos por el usuario **OWL 2**: sólo restricciones numéricas calificadas	Línea de comandos GUI (RacerPorter) OWLAPI (Java) DIG (HTTP) Java API (JRacer) Lisp API (LRacer) *Plug-in* Protégé	2.0 (Nov. 2012)
FaCT++	Código abierto (LGPL)	C++ Java (JFact)	**OWL 2**: admitido sin claves (*keys*), solo se admiten algunos tipos de datos	OWLAPI DIG (HTTP) *Plug-in* Protégé	1.6.2 (Feb. 2013)

Resumen

Este módulo presenta la web semántica, una evolución de la web donde el conocimiento puede ser utilizado automáticamente por programas inteligentes gracias a anotaciones semánticas que describen su interpretación. En esta visión, la representación del conocimiento y las ontologías en particular juegan un papel muy relevante en tanto que mecanismos para describir la semántica.

Se ha analizado la arquitectura de la web semántica y todos los estándares que la sostienen, destacando entre ellos RDF, RDF Schema y OWL. También se han descrito las aplicaciones de la web semántica que se han desarrollado hasta el momento, subrayando, por ejemplo, iniciativas como *linked data* u *open data*.

Desde una vertiente metodológica, se ha estudiado en detalle el proceso de construcción de una ontología, analizando diferentes alternativas. Este proceso es muy relevante, puesto que es uno de los pasos indispensables en cualquier proyecto relacionado con la web semántica. También se ha presentado la lógica descriptiva, un lenguaje formal clave para entender los procesos de razonamiento en la web semántica.

Finalmente, se han analizado algunos recursos y herramientas disponibles para la web semántica, destacando que algunos son de tipo gratuito o académico, pero que también los hay de carácter comercial. Precisamente, el hecho de que algunos aspectos de la web semántica ya no formen parte del ámbito de la investigación sino del ámbito de los negocios es un buen indicador del grado de progreso de la web semántica.

Ejercicios de autoevaluación

1. Pensad en un proyecto o poned un ejemplo para publicar datos en abierto.

2. La "Real Casa de la Moneda" requiere de una pequeña ontología para publicar su catálogo. Utilizando los pasos descritos, indicad en cada etapa qué se haría, como por ejemplo la reutilización de una ontología ya existente. Cread también algunas tripletas de ejemplo.

3. Modificad el ejemplo 39 "Ampliación del vocabulario de acontecimientos deportivos" con información acerca de la ubicación física y temporal de las competiciones. Haced uso de vocabularios externos.

Solucionario

1. Para ejemplificar el uso de datos en abierto pondremos el caso real de la iniciativa Open Government Data.

Open Government Data es una iniciativa reciente para la publicación en abierto de los datos públicos de la Administración. Esta iniciativa consiste en poner la información de que dispone el sector público (y que no está sujeto a ninguna restricción legal de privacidad o confidencialidad) al alcance de todo el mundo por medio de formatos digitales estandarizados. No es un simple proceso de publicación de datos, sino que va más allá y pretende fomentar el regreso a la sociedad de su información y fomentar así que las utilicen para todo aquello que se desee.

a) ¿Cuáles son los objetivos?

- Construir una comunidad en torno a los datos, y demostrar que el mundo es mucho mejor con datos abiertos que con datos cerrados.
- Crear un espacio común para hacer coincidir políticos y gente interesada en esta iniciativa.
- Construir algo para que la gente entienda el porqué de la apertura de los datos.
- Permitir utilizar datos que tenemos al alcance para poder encontrar soluciones factibles a los problemas con los que los ciudadanos se encuentren.

b) ¿Qué se quiere conseguir?

- Transformar la manera como el ciudadano se relaciona con la Administración.
- Mejorar la democracia, ofrecer un nuevo canal para participar en la sociedad.
- Posicionar al Gobierno como un actor más en la sociedad.
- Construir un modelo generalizado de cómo funciona la sociedad y cómo se organiza.

c) ¿Qué posibilita?

- Dar un valor económico. Permite a empresas, individuos y organizaciones sin ánimo de lucro construir aplicaciones y servicios útiles, interesantes y valiosos para la sociedad. En momentos de crisis, esto fomenta nuevas iniciativas.
- Ser democrático, posibilitar la transparencia y la participación en el Gobierno.
- Detectar posibles focos de corrupción con prácticas injustas e ilegales; muestra cómo se gasta y se invierte el dinero público.
- Capacitar a los ciudadanos para tomar mejores decisiones sobre su vida.
- Hacer más eficiente y efectiva la gestión pública.
- Hacer evidentes las lagunas de información.
- Fomentar debates sobre cómo se utiliza el dinero del contribuyente.

A continuación mostramos tres ejemplos en torno a la iniciativa:

- Como ejemplo de Administración pública que ofrece datos en abierto tenemos el Ayuntamiento de Barcelona (http://opendata.bcn.cat/opendata/es/?cl=1) y el de Nueva York (https://nycopendata.socrata.com/). Cada uno de ellos publica conjuntos de datos de información económica, de territorio, de entorno urbano, de población, de cultura, etc. Se puede consultar qué otras administraciones públicas del mundo también ofrecen datos en el portal web Data.gov.
- Otras iniciativas, como la "¿Dónde va a parar mi dinero?" (Reino Unido: http://wheredoesmymoneygo.org/, España: http://www.dondevanmisimpuestos.es/), o Farmsubsidy.org (http://farmsubsidy.org), organizan datos públicos para mostrar a la sociedad información de quien recibe qué y por qué.
- Por último, hay otras posibilidades que van más allá de suministrar conjuntos de datos o mostrar simplemente información organizada. Por ejemplo, hay proyectos para informar del estado del tráfico o de la contaminación atmosférica que proporcionan servicio mediante el uso de estos datos.

2. Se quiere desarrollar una ontología en el campo de la acuñación de moneda para la Casa Real de la Moneda.

Paso 1 – Determinar el dominio y el alcance de la ontología

Se planifica utilizarla en aplicaciones destinadas a informar acerca de todo lo relacionado con la presentación y venta de monedas. Así, contendrá conceptos que describan la información sobre emisiones de monedas, tipos de material y valor; colecciones conmemorativas y sistemas de almacenamiento, como los estuches. Principalmente, se prevé que sea utilizada por tiendas de venta al por menor.

Preguntas de verificación:

- ¿Cuántas monedas de plata hay en circulación?
- ¿Qué colecciones hay a partir del 2012?
- ¿Hay alguna colección sobre pintores?
- ¿En qué año se pone en circulación la primera moneda de euro?
- ¿Cuál es el máximo valor de una moneda?

Analizando estas preguntas, podemos comprobar que la ontología tendrá información sobre las colecciones de monedas, organizándolas por años, valores, aleación, precios, etc.

Paso 2 – Considerar la reutilización de ontologías existentes

A la hora de definir ciertos componentes de nuestra ontología, será más sencillo aprovechar ontologías ya creadas. En concreto para especificar los precios, tamaños, unidades y códigos de monedas.

Por eso utilizaremos las siguientes ontologías públicas:

- Good Relations
- Measurement Units

Paso 3 – Enumerar los términos relevantes de la ontología

Términos importantes y relativos a la moneda pueden ser: precio, aleación, peso, tirada máxima, estuche, colección, oro, plata, bronce, año de puesta en circulación, temática, etc.

Paso 4 – Definir las clases y la jerarquía

Utilizando la estrategia *top-down*, antes que nada definiremos las clases generales, y a continuación las especializaremos.

Las clases generales serían: colección, moneda, expositor y aleación.

Clase	Especialización (subclases)
colección	colección europea colección española colección monedas del mundo
moneda	moneda de curso legal moneda historicoconmemorativa
expositor	estuche de monedas estuche de monedas y sellos
aleación	aleación de bronce aleación de plata aleación de oro

Paso 5 – Definición de las propiedades de las clases

Cada clase tiene las siguientes propiedades:

Clase	Propiedades
colección	temática, año de publicación, conjunto de monedas, opcionalmente puede traer un expositor
moneda	aleación, peso, diámetro, tirada máxima, año de publicación, unidad monetaria, precio de venta e ISO 4217
moneda de curso legal	año de puesta en circulación
expositor	tirada máxima, precio de venta, número de monedas
estuche de monedas y sellos	número de monedas, número de sellos

Clase	Propiedades
aleación	porcentaje de pureza

Paso 6 – Definir las restricciones de las propiedades

Definimos las propiedades:

Propiedad	Dominio	Rango
hasYear	colección, moneda	fecha
hasAlloy	moneda	aleación
hasWeight	moneda	gr:QuantitativeValue*
hasDiameter	moneda	muo:UnitOfMeasurement**
maxCirculation	moneda, expositor	entero
hasPublishedYear	moneda	fecha
hasUnitPrice	moneda	gr:UnitPriceSpecification*
hasPrice	moneda, expositor	gr:UnitPriceSpecification*
hasCirculationYear	moneda	fecha
hasCoins	colección	moneda
hasDisplayCase	colección	expositor
hasPurity	aleación	muo:UnitOfMeasurement**
hasQuantityCoins	expositor	entero
hasQuantityStamps	estuche de monedas y sellos	entero

* Utiliza el espacio de nombres gr.
** Utiliza el espacio de nombres muo y también habrá que utilizar ucum.

Paso 7 – Crear instancias

Colección de monedas “Pintores españoles V: Joan Miró” (referencia 8833382):

- Moneda “Casa de la palmera” (referencia 92827025): aleación de plata (925), peso 168,75 gramos, 73 mm de diámetro, 5.000 unidades en circulación, unidad de 50 euros, y con precio de 314,60 euros.
- Moneda “Personajes y pájaros con un perro” (referencia 92827022): aleación de plata (925), peso 27 gramos, 40 mm de diámetro, 10.000 unidades en circulación, unidad de 10 euros, y con precio de 60,50 euros.
- Moneda “Retrato de una niña” (referencia 92827026): aleación de oro (999), peso 27 gramos, 38 mm de diámetro, 3.000 unidades en circulación, unidad de 400 euros, y con precio de 1.520,20 euros.

Las tripletas RDF quedarían de la siguiente manera (utilizaremos la URIRef abreviada *crm*, para representar el espacio de nombres inventado http://www.casarealmoneda.es/ont/crm).

Antes de nada declaramos las clases y subclases:

```
crm:Collection rdf:type rdfs:Class
crm:EuropeanCollection rdf:type rdfs:Class
crm:SpanishCollection rdf:type rdfs:Class
crm:WorldCoinsCollection rdf:type rdfs:Class
```

```
crm:EuropeanCollection rdf:subClassOf rdfs:Collection
crm:SpanishCollection rdf:subClassOf rdfs:Collection
crm:orldCoinsCollection rdf:subClassOf rdfs:Collection
crm:Coin rdf:type rdfs:Class
crm:TenderCoin rdf:type rdfs:Class
crm:HistoricallyCommemorativeCoin rdf:type rdfs:Class
crm:TenderCoin rdf:subClassOf rdfs:Coin
crm:HistoricallyCommemorativeCoin rdf:subClassOf rdfs:Coin
crm:DisplayCase rdf:type rdfs:Class
crm:CoinsCase rdf:type rdfs:Class
crm:CoinsAndStampsCase rdf:type rdfs:Class
crm:CoinsCase rdf:subClassOf rdfs:DisplayCase
crm:CoinsAndStampsCase rdf:subClassOf rdfs:DisplayCase
crm:Alloy rdf:type rdfs:Class
crm:BronzeAlloy rdf:type rdfs:Class
crm:SilverAlloy rdf:type rdfs:Class
crm:GoldAlloy rdf:type rdfs:Class
crm:BronzeAlloy rdf:subClassOf rdfs:Alloy
crm:SilverAlloy rdf:subClassOf rdfs:Alloy
crm:GoldAlloy rdf:subClassOf rdfs:Alloy
```

Seguidamente las propiedades:

```
crm:hasYear rdf:type rdfs:Property
crm:hasYear rdfs:domain crm:Collection
crm:hasYear rdfs:domain crm:Coin
xsd:date rdf:type rdfs:Datatype
crm:hasYear rdfs:range xsd:date
crm:hasAlloy rdf:type rdfs:Property
crm:hasAlloy rdfs:domain crm:Coin
crm:hasAlloy rdfs:range crm:Alloy
crm:hasWeight rdf:type rdfs:Property
crm:hasWeight rdfs:domain crm:Coin
crm:hasWeight rdfs:range gr:QuantitativeValue
crm:hasDiameter rdf:type rdfs:Property
crm:hasDiameter rdfs:domain crm:Coin
crm:hasDiameter rdfs:range muo:UnitOfMeasurement
crm:maxCirculation rdf:type rdfs:Property
crm:maxCirculation rdfs:domain crm:Collection
crm:maxCirculation rdfs:domain crm:Coin
xsd:integer rdf:type rdfs:Datatype
crm:maxCirculation rdfs:range xsd:integer
crm:hasPublishedYear rdf:type rdfs:Property
crm:hasPublishedYear rdfs:domain crm:Coin
crm:hasPublishedYear rdfs:range xsd:date
crm:hasUnitPrice rdf:type rdfs:Property
crm:hasUnitPrice rdfs:domain crm:Coin
crm:hasUnitPrice rdfs:range gr:UnitPriceSpecification
crm:hasPrice rdf:type rdfs:Property
crm:hasPrice rdfs:domain crm:Collection
crm:hasPrice rdfs:domain crm:Coin
crm:hasPrice rdfs:range gr:UnitPriceSpecification
crm:hasCirculationYear rdf:type rdfs:Property
crm:hasCirculationYear rdfs:domain crm:Con
crm:hasCirculationYear rdfs:range xsd:date
crm:hasCoins rdf:type rdfs:Property
crm:hasCoins rdfs:domain crm:Collection
crm:hasCoins rdfs:range crm:Coin
crm:hasDisplayCase rdf:type rdfs:Property
crm:hasDisplayCase rdfs:domain crm:Collection
crm:hasDisplayCase rdfs:range crm:DisplayCase
crm:hasPurity rdf:type rdfs:Property
crm:hasPurity rdfs:domain crm:Alloy
crm:hasPurity rdfs:range muo:UnitOfMeasurement
crm:hasQuantityCoins rdf:type rdfs:Property
crm:hasQuantityCoins rdfs:domain crm:DisplayCase
crm:hasQuantityCoins rdfs:range xsd:integer
crm:hasQuantityStamps rdf:type rdfs:Property
crm:hasQuantityStamps rdfs:domain crm:CoinsAndStampsCase
crm:hasQuantityStamps rdfs:range xsd:integer
```

Y finalmente algunas instancias:

```
crm:8833382 rdf:type crm:Collection
crm:8833382 rdf:label "Pintores españoles V: Joan Miró"
crm:8833382 crm:hasCoin crm:92827025
crm:8833382 crm:hasCoin crm:92827022
crm:8833382 crm:hasCoin crm:92827026

crm:92827025 rdf:type crm:Coin
crm:92827025 rdf:label "Casa de la palmera"
crm:92827025 crm:hasAlloy crm:SilverAlloy925
crm:SilverAlloy925 rdf:type crm:SilverAlloy
crm:SilverAlloy925 crm:hasPurity crm:9250per
crm:9250per rdf:type ucum:gram-percent
crm:9250per muo:numericalValue 92,50
crm:92827025 crm:hasYear "2012-01-01"
crm:92827025 crm:hasWeight crm:168_75gr
crm:168_75gr rdf:type gr:QuantitativeValueFloat
crm:168_75gr gr:hasValueFloat 168,75
crm:168_75gr gr:hasUnitOfMesurement GM
crm:92827025 crm:hasDiameter crm:73mm
crm:73mm rdf:type ucum:milli
crm:73mm muo:numericalValue 73
crm:92827025 crm:maxCirculation 5.000
crm:92827025 crm:hasUnitPrice crm:50euros
crm:50euros rdf:type gr:UnitPriceSpecification
crm:50euros gr:hasCurrencyValue 50
crm:50euros gr:hasCurrency EUR
crm:92827025 crm:hasPrice crm:314_60euros
crm:314_60euros df:type gr:UnitPriceSpecification
crm:314_60euros gr:hasCurrencyValue 314,60
crm:314_60euros gr:hasCurrency EUR

crm:92827022 rdf:type crm:Coin
crm:92827022 rdf:label "Personajes y Pájaros con un perro"
crm:92827022 crm:hasAlloy crm:SilverAlloy925
crm:92827022 crm:hasYear "2012-01-02"
crm:92827022 crm:hasWeight crm:27gr
crm:27gr rdf:type gr:QuantitativeValueFloat
crm:27gr gr:hasValueFloat 27
crm:27gr gr:hasUnitOfMesurement GM
crm:92827022 crm:hasDiameter crm:40mm
crm:40mm rdf:type ucum:milli
crm:40mm muo:numericalValue 40
crm:92827022 crm:maxCirculation 10.000
crm:92827022 crm:hasUnitPrice crm:10euros
crm:10euros rdf:type gr:UnitPriceSpecification
crm:10euros gr:hasCurrencyValue 10
crm:10euros gr:hasCurrency EUR
crm:92827022 crm:hasPrice crm:65_50euros
crm:65_50euros df:type gr:UnitPriceSpecification
crm:65_50euros gr:hasCurrencyValue 65,60
crm:65_60euros gr:hasCurrency EUR

crm:92827026 rdf:type crm:Coin
crm:92827026 rdf:label "Retrato de una Niña"
crm:92827026 crm:hasAlloy crm:GoldAlloy999
crm:GoldAlloy999 rdf:type crm:GoldAlloy
crm:GoldAlloy999 crm:hasPurity crm:9990per
crm:9990per rdf:type ucum:gram-percent
crm:9990per muo:numericalValue 99,90
crm:92827026 crm:hasYear "2012-01-01"
crm:92827026 crm:hasWeight crm:27gr
crm:27gr rdf:type gr:QuantitativeValueFloat
crm:27gr gr:hasValueFloat 27
crm:27gr gr:hasUnitOfMesurement GM
crm:92827026 crm:hasDiameter crm:38mm
crm:38mm rdf:type ucum:milli
crm:38mm muo:numericalValue 38
crm:92827026 crm:maxCirculation 3.000
crm:92827026 crm:hasUnitPrice crm:400euros
crm:400euros rdf:type gr:UnitPriceSpecification
```

```
crm:400euros gr:hasCurrencyValue 400
crm:400euros gr:hasCurrency EUR
crm:92827026 crm:hasPrice crm:1520_20euros
crm:1520_20euros df:type gr:UnitPriceSpecification
crm:1520_20euros gr:hasCurrencyValue 1.520,20
crm:1520_20euros gr:hasCurrency EUR
```

3. Se han encontrado en Internet los siguientes recursos externos que nos servirán:

Uso	URI del vocabulario	Espacio de nombres
Localización	http://www.geonames.org/ontology#	gn
Geolocalización	http://www.w3.org/2003/01/geo/wgs84_pos#	wgs84_pos
Ubicación temporal	http://www.w3.org/2002/12/cal#	cal

La primera modificación será en la única propiedad –la fecha de inicio– que identifica la temporalidad del acontecimiento. La podemos ampliar con información –por ejemplo– de la fecha de fin (*cal:dtend*) y la duración (*cal:duration*), y también aprovechando para cambiar el rango de la fecha de inicio (*cal:dtstart*):

```
sport:StartDate rdf:type rdfs:Property
sport:StartDate rdfs:domain sport:Competition
sport:StartDate rdfs:range cal:dtstart

sport:EndDate rdf:type rdfs:Property
sport:EndDate rdfs:domain sport:Competition
sport:EndDate rdfs:range cal:dtend

sport:Duration rdf:type rdfs:Property
sport:Duration rdfs:domain sport:Competition
sport:Duration rdfs:range cal:duration
```

Pero si lo que queremos es hacer más flexible la definición de la temporalidad del acontecimiento, habría que tener un elemento más complejo para identificar frecuencias, intervalos, etc. El recurso que permite esta definición es *cal:Vevent*. Por lo tanto, sustituimos todas las tripletas anteriores por el siguiente conjunto:

```
sport:TimeEvent rdf:type rdfs:Property
sport:TimeEvent rdfs:domain sport:Competition
sport:TimeEvent rdfs:range cal:Vevent
```

La otra propiedad para mejorar es "tener una ubicación", que en el ejemplo únicamente es un literal de texto. Ahora la modificaremos para que exprese más información detallada: el país (*gn:countryCode*), el nombre del lugar (*gn:name*), una latitud (*wgs84_pos:lat*) y una longitud (*wgs84_pos:long*):

```
sport:hasLocation rdfs:domain sport:Competition
sport:hasLocation rdf:type rdfs:Property
sport:hasLocation rdfs:range sport:Location

sport:hasCountry rdf:type rdfs:Property
sport:hasCountry rdfs:domain sport:Location
sport:hasCountry rdfs:range gn:countryCode

sport:hasPlace rdf:type rdfs:Property
sport:hasPlace rdfs:domain sport:Location
sport:hasPlace rdfs:range gn:name

sport:hasLatitude rdf:type rdfs:Property
sport:hasLatitude rdfs:domain sport:Location
sport:hasLatitude rdfs:range wgs84_pos:lat

sport:hasLongitude rdf:type rdfs:Property
sport:hasLongitude rdfs:domain sport:Location
sport:hasLongitude rdfs:range wgs84_pos:long
```

Glosario

ABox *Veáse* descripción del universo.

agente *m* Software inteligente y autónomo que puede percibir su entorno y actuar sobre él de forma racional para alcanzar unos objetivos.

axioma *m* En la lógica descriptiva, cada uno de los elementos de la terminología, que puede ser una definición de concepto o rol o bien establecer una relación de inclusión entre conceptos o roles.

base de conocimiento *f* En la lógica descriptiva, el conjunto formado por una terminología y una descripción del universo.
Véase terminología, descripción del universo.

buscador semántico *m* Motor de búsqueda que explota la información semántica contenida en las páginas web para conseguir mejorar las respuestas a las consultas de los usuarios.

clase *f* *Véase* concepto.

concepto *m* En la lógica descriptiva, elemento que denota un conjunto de instancias y que se puede definir sobre la base de otros conceptos o bien indicar si está incluido en otros conceptos. Es un término análogo a las clases de los documentos RDF.

DAML Sigla de *DARPA Agent Mark-up Language*, un lenguaje para la descripción de ontologías precursor de OWL.

DAML-OIL Fusión de los lenguajes DAML y OIL que inspiró el lenguaje OWL.

descripción del universo *f* En la lógica descriptiva, el conjunto de instancias de los conceptos y roles definidos en la terminología.
Véase terminología, rol.

description logic *Véase* lógica descriptiva.

DL *Véase* lógica descriptiva.

entorno de trabajo para la web semántica *m* Plataforma que proporciona servicios para la web semántica a los que se puede acceder desde un lenguaje de programación a través de llamadas a una API.
en *framework*

enunciado *Véase* tripleta.

folcsonomía *f* Mecanismo de clasificación colaborativa de un conjunto de recursos en el cual cada usuario propone términos para describir un recurso, proceso del que resulta un conjunto de etiquetas simples en un espacio de nombres plano.

HTML Sigla de *Hyper Text Mark-up Language*, el lenguaje de marcas estándar utilizado para describir las páginas web.

linked data Movimiento para favorecer la publicación de datos en la web siguiendo unos criterios que permitan interpretar el contenido y relacionarlos (enlazarlos) con otros conjuntos de datos existentes.

lógica descriptiva *f* Familia de lenguajes formales que permiten expresar propiedades relevantes por la representación del conocimiento y razonar sobre ellas.

N3 *Véase* Notation3.

Notation3 Notación textual estándar para serializar documentos RDF en una sintaxis compacta que facilita la lectura humana.

OIL Sigla de *Ontology Interchange Language*, un lenguaje para la descripción de ontologías inspirado en la lógica descriptiva y precursor de OWL.

ontology engineering Denominación del proceso de construcción de una ontología.

open data Movimiento partidario de la publicación de datos con licencias que permitan el libre acceso, uso, modificación y redistribución.

OWL Sigla de *Ontology Web Language*, un lenguaje estándar para la descripción de ontologías inspirado en la lógica descriptiva y lenguajes previos como DAML-OIL.

predicado *m* Elemento que relaciona un sujeto y un objeto dentro de una tripleta RDF. Este término es equivalente al del *rol* dentro de la lógica descriptiva.

pregunta de verificación / relevancia / competencia *f* Consulta sobre un dominio utilizada en el proceso de construcción de ontologías con doble finalidad: para describir los problemas tipo que la ontología tendría que ser capaz de responder y para validar si la ontología es completa.

propiedad *Véase* predicado.

RDF Sigla de *Resource Description Framework*, un estándar del W3C que describe un modelo de datos para la web semántica basado en enunciados de la forma Sujeto-Predicado-Objeto.

RDF/XML Notación textual estándar que permite serializar documentos RDF siguiendo la sintaxis XML.

RDF* Término genérico que engloba las tecnologías RDF, RDFa y RDF Schema.

RDFS *Véase* RDF Schema.

RDF Schema Estándar que extiende RDF para permitir la definición de vocabularios y conjuntos de propiedades semánticas, definiendo un conjunto de clases y propiedades y la jerarquía de relaciones entre ellos.

RDFa Sigla de *Resource Description Framework in Attributes*, un mecanismo para marcar documentos XML con datos RDF.

rol *m* En la lógica descriptiva, relación binaria que se establece entre dos individuos. Es equivalente a los términos *predicado* y *propiedad* de los documentos RDF.

serialización *f* Proceso de codificación de datos en un formato textual o binario para almacenarlos, transportarlos o intercambiarlos.

servicio web *m* Sistema de software capaz de permitir la interoperabilidad máquina-máquina sobre una red, a través de una interfaz descrita en un formato procesable por máquinas.

SOAP Sigla de *Simple Object Access Protocol*, un protocolo de comunicación basado en XML y diseñado para el intercambio de información estructurada para acceder a servicios web de Internet.
Véase servicio web.

SPARQL Sigla de *Simple Protocol and RDF Query Language*, un lenguaje estándar para la consulta de conocimiento en formato RDF.

TBox *Véase* terminología.

terminología *f* En la lógica descriptiva, conjunto de definiciones de conceptos y roles y sus relaciones jerárquicas de inclusión. En términos de documentos RDF, esta idea se suele denominar vocabulario.

triplestore *f* Herramienta destinada al almacenamiento y consulta de grandes volúmenes de datos RDF. Se puede ver como una base de datos de información semántica.

tripleta *f* Enunciado de un documento RDF que describe una relación Sujeto-Predicado-Objeto.

UML Sigla de *Unified Modeling Language*; una notación estándar del *Object Management Group* para la modelización en el ámbito de la ingeniería del software. Esta notación incorpora diagramas, como por ejemplo el diagrama de clases, que se puede usar para describir una jerarquía de clases.

Unicode *m* Conjunto de caracteres estándar que permite expresar texto en cualquier alfabeto, incluidos alfabetos no latinos, como por ejemplo el árabe, el cirílico o el chino.

URI Sigla de *Uniform Resource Identifier*, un formato estándar para identificar recursos de forma única.

URL Sigla de *Uniform Resource Locator*, es un tipo concreto de URI que permite identificar y localizar un recurso dentro de la web.

vocabulario *m* *Véase* terminología.

W3C Sigla de *World Wide Web Consortium*, una comunidad internacional dedicada al desarrollo de estándares para la web.

WSDL Sigla de *Web Service Description Language*, un formato estándar XML utilizado para describir la funcionalidad pública ofrecida por un servicio web.
Véase servicio web.

XML Sigla de *eXtensible Mark-up Language*, un lenguaje de marcas para intercambiar información que permite, entre otras muchas aplicaciones, la serialización de datos anotados semánticamente.

Bibliografía

Gómez-Pérez, A.; Fernandez-López, M.; Chorcho, O. (2004). *Ontological engineering* (2.ª ed.). Springer-Verlag. (ISBN: 1-85233-551-3)

Miller, P. (enero, 2010). "Linked Data Horizon Scan". *Joint information systems committee*.

Noy, N. F.; McGuinness, D. L. (2001). "Ontology development 101: A guide to creating your first ontology". *Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report* (Informe SMI-2001-0880, marzo).

Uschold, M.; Gruninger, M. (1996). "Ontology: Principles, methods and applications". *Knowledge engineering review* (vol. 2, núm. 11).

Anexo

Serialización del vocabulario de acontecimientos deportivos

A continuación mostramos la sintaxis para describir las clases del ejemplo de "Vocabulario de acontecimientos deportivos".

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xml:base="http://www.schema.org/sport">

   <rdf:Description rdf:ID="SportDiscipline">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
   </rdf:Description>

   <rdf:Description rdf:ID="Athletics">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
      <rdfs:subClassOf rdf:resource="#SportDiscipline" />
   </rdf:Description>

   <rdf:Description rdf:ID="Cycling">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
      <rdfs:subClassOf rdf:resource="#SportDiscipline" />
   </rdf:Description>

   <rdf:Description rdf:ID="Football">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
      <rdfs:subClassOf rdf:resource="#SportDiscipline" />
   </rdf:Description>

   <rdf:Description rdf:ID="SportingEvent">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
   </rdf:Description>

   <rdf:Description rdf:ID="Competition">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
      <rdfs:subClassOf rdf:resource="#SportingEvent" />
   </rdf:Description>

   <rdf:Description rdf:ID="Tournament">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
      <rdfs:subClassOf rdf:resource="#SportingEvent" />
   </rdf:Description>
```

```
</rdf:RDF>
```

Serialización abreviada del vocabulario de acontecimientos deportivos

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xml:base="http://www.schema.org/sport">

   <rdfs:Class rdf:ID="SportDiscipline" />

   <rdfs:Class rdf:ID="Athletics">
      <rdfs:subClassOf rdf:resource="#SportDiscipline" />
   </rdf rdfs:Class>

   <rdf:Class rdf:ID="Cycling">
      <rdfs:subClassOf rdf:resource="#SportDiscipline" />
   </rdf rdfs:Class>

   <rdf:Class rdf:ID="Football">
      <rdfs:subClassOf rdf:resource="#SportDiscipline" />
   </rdf rdfs:Class>

   <rdf:Class rdf:ID="SportingEvent" />

   <rdf:Class rdf:ID="Competition">
      <rdfs:subClassOf rdf:resource="#SportingEvent" />
   </rdf rdfs:Class>

   <rdf:Class rdf:ID="Tournament">
      <rdfs:subClassOf rdf:resource="#SportingEvent" />
   </rdf rdfs:Class>
</rdf:RDF>
```

Serialización de las propiedades del vocabulario de acontecimientos deportivos

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:foaf="xmlns.com/foaf/0.1/"
   xml:base="http://www.schema.org/sport">

   ...

   <rdf:Description rdf:ID="CompetitionType">
```

```
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#Competition"/>
        <rdfs:range rdf:resource="#SportDiscipline"/>
    </rdf:Description>

    <rdf:Description
        rdf:about="http://www.w3.org/2001/XMLSchema#date">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Datatype"/>
    </rdf:Description>

    <rdf:Description rdf:ID="StartDate">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#Competition"/>
        <rdfs:rangerdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
    </rdf:Description>

    <rdf:Description
        rdf:about="http://www.w3.org/2001/XMLSchema#string">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Datatype"/>
    </rdf:Description>

    <rdf:Description rdf:ID="Location">
        <rdf:type rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </rdf:Description>

    <rdf:Description rdf:ID="hasLocation">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#Competition"/>
        <rdfs:range rdf:resource="#Location"/>
    </rdf:Description>

    <rdf:Description rdf:ID="competesIn">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
        <rdfs:range rdf:resource="foaf:Agent"/>
    </rdf:Description>

    <rdf:Description rdf:ID="Team">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="foaf:Agent"/>
    </rdf:Description>

    <rdf:Description rdf:ID="Player">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="foaf:Agent"/>
    </rdf:Description>
</rdf:RDF>
```

Serialización abreviada de las propiedades del vocabulario de acontecimientos deportivos

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:foaf="xmlns.com/foaf/0.1/"
   xml:base="http://www.schema.org/sport">

   ...

   <rdf:Property rdf:ID="CompetitionType">
      <rdfs:domain rdf:resource="#Competition"/>
      <rdfs:range rdf:resource="#SportDiscipline"/>
   </rdf:Property>

   <rdfs:Datatype rdf:about="#date"/>

   <rdf:Property rdf:ID="StartDate">
      <rdfs:domain rdf:resource="#Competition"/>
      <rdfs:range rdf:resource="#date"/>
   </rdf:Property>

   <rdfs:Datatype rdf:about="#string"/>

   <rdf:Description rdf:ID="Location">
      <rdf:type rdf:resource="#string"/>
   </rdf:Description>

   <rdf:Property rdf:ID="hasLocation">
      <rdfs:domain rdf:resource="#Competition"/>
      <rdfs:range rdf:resource="#Location"/>
   </rdf:Property>

   <rdf:Property rdf:ID="competesIn">
      <rdfs:range rdf:resource="foaf:Agent"/>
   </rdf:Property>

   <rdf:Class rdf:ID="Team">
      <rdfs:subClassOf rdf:resource="foaf:Agent"/>
   </rdf:Class>

   <rdf:Class rdf:ID="Player">
      <rdfs:subClassOf rdf:resource="foaf:Agent"/>
   </rdf:Class>
</rdf:RDF>
```