



# **EL FUTURO DE LA WEB**

## **XML, RDF/RDFS, ontologías y la Web semántica**

**Miguel Ángel Abián**



# EL FUTURO DE LA WEB

## XML, RDF/RDFS, ontologías y la Web semántica

**Resumen:** El propósito de este trabajo es explicar desde un punto de vista práctico las limitaciones de la Web actual (por ejemplo, la baja precisión de los motores de búsqueda basados en palabras clave, como Google y Yahoo) y por qué necesitamos la Web semántica (una Web “inteligente”). El término *Web semántica* corresponde a una visión en que tanto los ordenadores –software– como las personas podrán encontrar, leer, comprender y usar los datos de la *World Wide Web*. La Web semántica, basada en crear datos procesables por las máquinas, permitirá el razonamiento automático, la gestión del conocimiento, la mejora del comercio electrónico y la búsqueda de información de manera eficaz y precisa. Estas características se explicarán en el tutorial, especialmente aplicadas al comercio electrónico.

En este trabajo se describen las principales tecnologías de la Web semántica: XML, XML Schema, RDF, RDF Schema (RDFS) y ontologías. XML proporciona una sintaxis superficial para documentos estructurados, pero no impone restricciones semánticas sobre el significado de los documentos. XML Schema es un lenguaje para restringir la estructura de los documentos XML. RDF es un modelo de datos para objetos (“recursos”) y las relaciones entre ellos; proporciona una semántica (“significado”) sencilla para este modelo de datos, y estos modelos de datos se pueden representar en una sintaxis de XML. RDFS es un lenguaje universal que deja a los usuarios describir los recursos con sus propios vocabularios; RDFS es necesario porque RDF no define la semántica de ningún dominio particular. Las ontologías son especificaciones formales de cómo representar los objetos, conceptos y otras entidades que se asume que existen en un área de interés, así como las relaciones que mantienen entre sí.

**Abstract:** The goal of this work is to explain the limitations of the current Web (e.g., the low precision of keyword-based search engines, such as Google and Yahoo) from a practical approach, and why we need the Semantic Web (an “intelligent” Web). The term *Semantic Web* stands for a vision in which computers –software– as well as people can find, read, comprehend, and make use of data over the World Wide Web. The Semantic Web, based on creating machine-processable data, will allow automatic reasoning, knowledge management, e-commerce improvement, and information search in an accurate, efficient and less time-consuming way. These issues will be explained in this tutorial, specially applied to e-commerce.

In this work the key Semantic Web technologies are described: XML, XML Schema, RDF, RDFS and ontologies. XML provides a surface syntax for structured documents but imposes no semantic constraints on the meaning of these documents. XML Schema is a language for restricting the structure of XML documents. RDF is a data model for objects (“resources”) and relations between them; it provides a simple semantics (“meaning”) for this data model, and these data models can be represented in XML syntax. RDFS is a universal language that lets users describe resources using their own vocabularies; RDFS is necessary because RDF does not define the semantics of any particular domain. An ontology is an explicit formal specification of how to represent the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them.

**Keywords:** e-commerce, e-business, b2c, b2b, Semantic Web, agents, intelligent agents, shopbots, XML, RDF, RDFS, ontologies, metadata, machine-processable data, web services, EDI, conventional EDI approach, VAN, VAN networks, OWL, DAML+OIL, knowledge, knowledge base, knowledge representation, interoperability, syntactic interoperability, semantic interoperability, Java, JENA, Protégé, SMORE, VKB, Enron, 11-S



# ÍNDICE

<b>1. Introducción</b>	<b>Página 4</b>
<b>2. La Web actual. Glosario</b>	<b>Página 7</b>
<b>3. Problemas de la Web actual</b>	<b>Página 13</b>
<b>3.1 Introducción</b>	<b>Página 13</b>
<b>3.2 Dificultad para encontrar información</b>	<b>Página 17</b>
<b>3.3 Dificultad para implantar comercio electrónico B2B</b>	<b>Página 21</b>
<b>4. La Web semántica</b>	<b>Página 30</b>
<b>5. XML: un primer paso hacia la Web semántica</b>	<b>Página 49</b>
<b>6. RDF y RDFS: el pegamento semántico</b>	<b>Página 59</b>
<b>6.1 RDF y RDFS</b>	<b>Página 59</b>
<b>6.2 Aplicaciones de RDF y RDFS</b>	<b>Página 71</b>
<b>6.3 Nadie es perfecto: desventajas de RDFS</b>	<b>Página 82</b>
<b>7. Ontologías</b>	<b>Página 84</b>
<b>7.1 Introducción</b>	<b>Página 84</b>
<b>7.2 Lenguajes de representación de ontologías y herramientas</b>	<b>Página 88</b>
<b>7.3 Aplicaciones de las ontologías</b>	<b>Página 99</b>
<b>8. Algunas reflexiones sobre la Web semántica</b>	<b>Página 102</b>
<b>Nota biográfica del autor</b>	<b>Página 103</b>



# EL FUTURO DE LA WEB

## XML, RDF/RDFS, ontologías y la Web semántica

Fecha de creación: 15.09.2005

Miguel Ángel Abián  
mabian ARROBA aidima PUNTO es

**Copyright (c) 2005, Miguel Ángel Abián.** Este documento puede ser distribuido sólo bajo los términos y condiciones de la licencia de Documentación de javaHispano v1.0 o posterior (la última versión se encuentra en <http://www.javahispano.org/licencias/>).

Me agradaban, aunque no era consciente de ello, las demostraciones típicas de la mentalidad matemática. Cuando fui mayor, encontré a otros que opinaban como yo en este asunto. Mi amigo G. H. Hardy gozaba de este placer con una intensidad muy grande. Una vez me dijo que, si pudiera encontrar una prueba de que yo me iba a morir antes de cinco minutos, lamentaría naturalmente perderme, pero que ese pesar sería completamente sobrepasado por el placer que le produciría la prueba. Estuve enteramente de acuerdo con él y no me ofendí en absoluto.

**Bertrand Russell**, *Portraits from Memory and other Essays* (1956)

Propongo considerar la pregunta: “¿Pueden pensar las máquinas?”. Ésta debería empezar con las definiciones del significado de los términos “maquina” y “pensar”.

**Alan Turing**, *Computing Machinery and Intelligence* (1950)

## 1. Introducción

Este trabajo de divulgación no trata sobre Java, si bien esta palabra se mencionará varias veces e incluso aparecerá algo de código escrito en Java. Trata del presente y, en mayor medida, del futuro de la Web. La manera como se desarrolle la Web influirá en nuestra forma de usarla (consultas, compras...), así que conviene saber hacia adónde podría dirigirse la Web. En el caso de los programadores, aún resulta más acuciante conocer las tecnologías que posiblemente dominarán la Web de aquí a unos años.

Actualmente, se admite que la Web del futuro será “inteligente”. Es decir, que estará llena de información que las máquinas podrán “comprender” y a partir de la cual extraerán conclusiones de interés para los usuarios. La necesidad de una Web “inteligente” aparece en cuanto consideramos que la Web actual está formada por unas 11.500 millones de páginas estáticas (no creadas en respuesta a las acciones de los usuarios). Los buscadores web más usados cubren bastante bien el total de la red: Google cubre 8.800 millones de páginas; Yahoo, 8.000 millones, y MSN, 7.100 millones. Ahora bien, ninguno de ellos ofrece búsquedas “inteligentes”, adaptadas a las intenciones de los usuarios. Los procedimientos que usan para valorar la relevancia de los resultados devueltos obedecen a criterios estadísticos, no a los del usuario. Por ejemplo, un buscador web puede encontrar miles de páginas que contengan las palabras



“García”, “México D. F” y “Venezuela”; pero es incapaz de aceptar una petición del tipo “Busco a una persona apellidada García que viven en la calle Venezuela de la ciudad de México D. F.”.

La inteligencia de la Web actual brilla por su ausencia cuando se intentan búsquedas del estilo “¿Qué generales soviéticos defendieron la ciudad de Stalingrado durante el cerco nazi?”, “¿Cómo se llamaba el sacerdote español que encabezó el levantamiento popular contra las tropas napoleónicas?”, “¿Cómo se llamaba el conquistador asiático de cuyo caballo se decía que no crecía la hierba por donde pisaba?”, “¿Qué vicealmirante tuerto, manco y aquejado de depresión pronunció la frase *Inglaterra espera que cada hombre cumpla con su deber?*” o “¿Qué valores del Ibex-35 reparten dividiendo la próxima semana?”. Como veremos más adelante, la “estupidez” de la Web no sólo exaspera a los usuarios (“Quien espera, desespera”), sino que tiene consecuencias económicas muy relevantes para los usuarios y para las empresas que desean realizar negocios electrónicos.

En este trabajo se explicarán las tecnologías que parecen más prometedoras para lograr una Web mucho más “inteligente” (la Web semántica) que la actual: XML, RDF/RDFS y ontologías. Todas ellas se enfocarán desde un punto de vista práctico y autocontenido. Más que dedicarme a explicarlas exhaustivamente, las abordaré desde un punto de vista divulgativo, más centrado en explicar para qué sirven y por qué son como son que en dar una lista exacta de sus propiedades. En el caso de XML, como es una tecnología muy popular, he dedicado bastante más espacio a explicar sus ventajas y el enorme impacto que ha tenido sobre los sistemas de información y los negocios electrónicos que a explicarlo técnicamente. En concreto, he hecho bastante hincapié sobre las ventajas que XML ofrece sobre el enfoque EDI convencional (que dominó el comercio electrónico en los años 70 y 80).

Una sugerencia: no deje que la abundancia de siglas o que palabras raras como “ontologías” le desconcierten o le tiran para atrás. No se necesita ser un especialista o un fanático de la informática para leer este trabajo, ni siquiera para escribirlo. Si está familiarizado con la Web, con Java, con la orientación a objetos (OO) y ha leído algo sobre XML y sobre comercio electrónico, este tutorial es idóneo para usted. Si desconoce la OO o sus conocimientos de ella son muy superficiales, en los tutoriales

- a) <http://www.javahispano.org/tutorials.item.action?id=25>
- b) <http://www.javahispano.org/tutorials.item.action?id=33>

puede encontrar información sobre ella y sobre los lenguajes orientado a objetos (Java, C++, Smalltalk, C#). Hay, eso sí, otro requisito que facilitará la lectura de este trabajo: curiosidad por el futuro.

Quizás alguien se quede un tanto desconcertado porque vinculo –al final del apartado 4– el futuro uso masivo de las tecnologías de la Web semántica con decisiones estratégico-militares estadounidenses que proceden de los sucesos del 11 de septiembre de 2001. No hay para tanto: muchas tecnologías que usamos hoy día (microondas, radares, láser, Internet) proceden de proyectos militares; hasta el punto de que la historia de las telecomunicaciones es la historia de la Segunda Guerra Mundial y la Guerra Fría. La misma mano que ha arrojado bombas nucleares, *napalm* y bombas recubiertas de uranio empobrecido ha creado y difundido algunas de las tecnologías que usamos hoy. Me costaría bien poco ponerme la toga de la Santa Imparcialidad Científica y contar los avances científicos o técnicos sin referirme a las circunstancias sociales, históricas o políticas que los han hecho posibles o los han acelerado; pero cada vez soy más



consciente de que obrar así en nombre de la objetividad es traicionarla completamente. Imaginemos que un extraterrestre desconocedor de la cultura del *homo sapiens* llegara a la Tierra y que, paseando por alguna calle de un país hispanohablante, se preguntara de dónde habían venido los átomos de una escultura que representa a un escritor manco y vestido a la usanza del siglo XVI. Un astrofísico podría explicar al alienígena que esos átomos se originaron en el centro de una estrella moribunda e incluso podría escribir las ecuaciones matemáticas de las trayectorias que recorrieron en el espacio-tiempo hasta alcanzar su posición actual. Una persona más avispada podría explicarle que los humanos de un país levantan estatuas en homenaje a aquellos compatriotas suyos que han destacado en el arte o en la ciencia, a modo de recordatorio para las futuras generaciones. Ambas explicaciones son ciertas. La primera parece más objetiva que la segunda, pero ¿cuál de las dos sería más útil para el extraterrestre?

En la página 47 incluyo un ejemplo de lo indigesta que resultó la mezcla de las nuevas tecnologías con la rafiña humana. Cabe suponer que situaciones similares sucederán también con la Web semántica. ¿Qué empresa tendrá el dudoso honor de ser la Enron de la nueva Web?

Con *El futuro de la Web* no busco engañar a ningún lector o lectora: nadie puede predecir el curso de los avances científicos o tecnológicos, y es posible –si bien improbable– que la Web que usemos de aquí a diez o quince años se base en tecnologías distintas a las mencionadas antes. Dicho de otro modo, alguna de estas tecnologías podría tener la vida del pollo: veintiocho días y al degollo. De lo que no me cabe ninguna duda es de que la Web del mañana no será como piensan las gentes bienpensantes. En realidad, el futuro jamás ha sido como esperaban los bienpensantes.

Los físicos de finales del siglo XIX pensaban que la Física era una disciplina acabada y completa, como la termodinámica clásica; faltaba solucionar dos o tres detalles, sí, pero nada más. Afortunadamente (al menos para los físicos), el estudio de los espectros atómicos y el experimento de Michelson-Morley acabaron con esa fúnebre actitud y modificaron radicalmente la vida de los habitantes del siglo XX. Nada hacía predecir que ese siglo sería el de las bombas nucleares, el láser, la biología molecular, la bomba de cobalto, las sondas espaciales, los semiconductores, los computadores; pero así fue. Al final, la disciplina prácticamente acabada se reveló incompleta y hasta errónea. Acaso suceda algo similar con la idea de que ya tenemos las tecnologías que permitirán, dentro de poco, dotar de inteligencia a la Web y expresar los datos de una manera que sea comprensible para las máquinas y que permita hacer deducciones automáticas a partir de ellos. Incluso admitiendo esa posibilidad, considero interesante conocer las tecnologías que ahora parecen más prometedoras para que las máquinas nos liberen de muchas tareas tediosas. Si de aquí a diez años las ontologías y RDF son vías muertas, tan muertas como el átomo de Rutherford o la teoría del éter, espero que no me juzgue demasiado severamente: he tratado de mirar lo más lejos que he podido.



## 2. La Web actual. Glosario

La característica más llamativa de la Web es su estructura hipertextual: la *World Wide Web* contiene una gigantesca cantidad de documentos de hipertexto (llamados páginas web). Los documentos de hipertexto contienen enlaces que conectan con otros documentos; un enlace puede ser una palabra, una frase o una imagen. A través de los enlaces de un documento de hipertexto se puede acceder a otras secciones del documento y a otros documentos, imágenes, vídeos, sonidos, etc. En general, se dice que la red permite acceder a **recursos web**. Los recursos web son dispositivos físicos (impresoras, ordenadores, agendas electrónicas, etc.) o estructuras de datos de software accesibles (páginas web, imágenes, vídeos, directorios, etc.). Frente a los documentos convencionales, cuya lectura es lineal, los documentos de hipertexto permiten una lectura no lineal, en la que el lector puede saltar a información relacionada con el documento que lee.

El término “hipertexto” fue introducido por Ted Nelson en 1965; con él se refería a una colección de documentos (“nodos”) con referencias cruzadas o enlaces que, mediante la ayuda de programas navegadores, permiten al lector moverse sencillamente de un documento a otro. Nelson presentó así el nuevo concepto: “Déjeme introducir la palabra hipertexto para designar un cuerpo de material escrito o gráfico interconectado de una manera que podría no ser conveniente presentar o representar en papel”.

La Web, al igual que Internet, basa su funcionamiento en la pila o familia de protocolos TCP/IP (encontrará una rápida descripción de estos protocolos en la primera parte de *Java y las redes*, <http://www.javahispano.org/tutorials.item.action?id=45>). Dentro de esta pila de protocolos, se destacan los protocolos de la capa de aplicación, destinados a permitir la comunicación directa entre aplicaciones. Concretamente, el protocolo de aplicación HTTP (*HyperText Transfer Protocol*, protocolo de transferencia de hipertexto) permite acceder a documentos de hipertexto. Tales documentos se escriben en HTML (*HyperText Markup Language*, lenguaje de etiquetado de hipertexto), un lenguaje que especifica la forma como se muestran y los enlaces entre ellos. Por ello, los términos *página web* y *página HTML* son intercambiables.

Cuando la Web comenzó a popularizarse, las páginas web se escribían a mano y eran estáticas. Ahora hay muchísimas herramientas para la generación automática de páginas web y abundan las páginas web activas (es decir, creadas como respuesta a acciones y peticiones de los usuarios). Por ejemplo, cuando se hace una compra electrónica, la página web que muestra los detalles de la transacción no existía antes de la compra: se genera basándose en la información introducida por el comprador. Lo mismo sucede cuando se consulta una base de datos remota mediante una interfaz web: las páginas web con los resultados no existían antes de la consulta.

El uso actual de la Web se puede resumir en este texto de un grupo de trabajo del W3C (W3C XML ProtocolWorking Group):

Hoy, el principal uso de la Web es para acceso interactivo a documentos y aplicaciones. En casi todos los casos, este acceso es hecho por usuarios humanos, que típicamente trabajan con navegadores web, reproductores de audio u otros sistemas interactivos en el lado del usuario. La Web puede crecer significativamente en poder y alcance si se extiende para permitir comunicaciones entre aplicaciones, de un programa a otro.



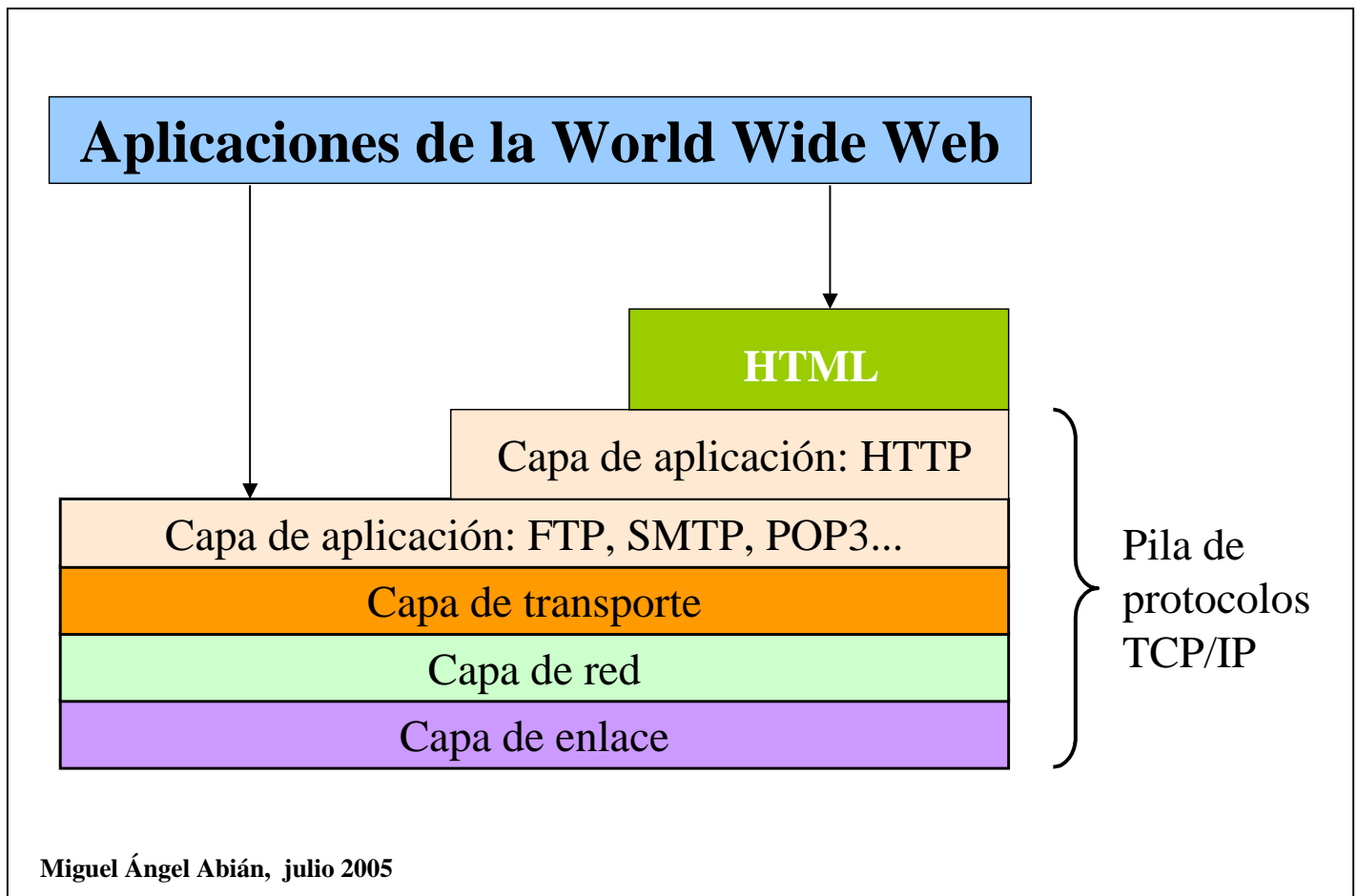


Figura 1. Estructura interna de la Web

Antes de continuar, conviene definir unos cuantos conceptos que aparecerán en este documento:

- **Agente.** Es una aplicación capaz de cumplir sus objetivos mediante acciones autónomas y flexibles. Frente a los objetos o los componentes, los agentes se caracterizan por su autonomía y flexibilidad.
- **Agente inteligente.** Una aplicación que automatiza la búsqueda de información en la Web. Por ejemplo, hay agentes inteligentes que recopilan para el usuario todas las nuevas noticias que sobre un asunto determinado se van publicando en la Web, procedentes de agencias de prensa, de periódicos, de *chats*, etc. Véase también la definición de *softbot*.
- **Analizador sintáctico (*parser*).** Programa que analiza, con respecto a una gramática, la estructura (sintaxis) de un documento (en inglés, el proceso se conoce como *parsing*). Una gramática (o *sintaxis*: son términos equivalentes) es un conjunto de reglas formales que especifican la estructura de los documentos aceptables o correctos. Por ejemplo, una DTD (*Document Type Definition*, definición de tipos de documentos) define una gramática para los documentos XML. Mediante un analizador sintáctico de XML, se puede averiguar si un



cierto documento XML cumple la gramática definida en la DTD; esto es, si es válido para esa gramática. Aunque aquí sólo interesan los analizadores de lenguajes de programación, existen también analizadores de lenguajes naturales, capaces de averiguar si una frase como “El gata pardo bien sentada en el sillón está” es o no correcta.

- **API (*Application Program Interface*, interfaz de programación de aplicaciones).** Conjunto de subrutinas, protocolos y herramientas para construir aplicaciones de software. Las buenas API proporcionan a los programadores todos los bloques que necesitan para construir aplicaciones de cierto tipo.
- **Base de conocimiento (*knowledge base*).** Hechos (“Luis es un empleado”), relaciones (“Los empleados tienen jefes”) y procedimientos que forman el conocimiento sobre un dominio, expresados en una forma que las máquinas puedan procesar. Si bien el término se usará en el sentido anterior, también se emplea *base de conocimiento* para denotar el conjunto de todo el conocimiento explícito de una organización, que puede estar formado por manuales, presentaciones, tutoriales, reglas internas, etc.
- **Cadena de aprovisionamiento (*supply chain*).** Secuencia de actividades que las organizaciones desarrollan para producir y entregar un producto o servicio. En una empresa manufacturera, la cadena de aprovisionamiento comienza con el procesamiento de las materias primas, continúa con la fabricación de piezas y termina con el ensamblado de las piezas y la distribución del producto final.
- **Comercio electrónico.** Intercambio y procesamiento de la información por medios electrónicos, con el fin de permitir transacciones económicas.
- **Comercio electrónico B2B (*business-to-business*).** Comercio electrónico entre empresas.
- **Comercio electrónico B2C (*business-to-customer*).** Comercio electrónico entre una empresa y los consumidores finales.
- **Dominio.** Parte del mundo real que resulta de interés. Es sinónimo de “área de interés”.
- **Integración (de sistemas).** El proceso de juntar sistemas, dispositivos y programas en una misma arquitectura, de modo que puedan compartir e intercambiar datos.
- **Interoperabilidad.** Según el proyecto IDEAS, es la capacidad de un sistema o un producto para trabajar con otros sistemas o productos sin especial esfuerzo por parte del cliente. Según la norma ISO 16100, es la capacidad para compartir e intercambiar información, mediante una sintaxis y una semántica comunes, para conseguir una relación funcional específica mediante el uso de una interfaz común. Según el IEEE (*Institute of Electrical and Electronics Engineers*, Instituto de Ingenieros Eléctricos y Electrónicos) es “la capacidad de dos o más sistemas o componentes para intercambiar información y usar la información que ha sido intercambiada” ([*IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries, 1990*]). Según Lisa L. Brownsword *et al.* ([*Current*



**Perspectives on Interoperability, marzo de 2004]**), es la capacidad de una colección de entidades comunicativas de a) compartir información especificada; y b) de operar sobre esa información de acuerdo con una semántica operacional establecida.

- **Modelo de negocio.** Representación del negocio de una o más empresas, definida según la terminología de los expertos del dominio y sin vinculación a ningún sistema específico ni a ninguna aplicación de software. El principal objetivo de un modelo de negocio es representar formalmente el negocio mediante sus objetos de negocio y las relaciones entre ellos. Por lo general, un modelo de negocio consiste en una lista de objetos y de casos de uso.
- **Modelado empresarial.** El proceso de construir modelos que representen a las empresas (modelos de negocio, de datos, de procesos, de toma de decisiones, etc.).
- **Negocio electrónico.** Compra y venta de bienes y servicios mediante Internet. Suele usarse como sinónimo de “comercio electrónico”, si bien algunas empresas piensan que “comercio electrónico” es un término más amplio que “negocio electrónico”.
- **Objeto de negocio.** Una representación de cualquier elemento activo en el dominio del negocio de una o más empresas. Como mínimo, debe incluir estas características: nombre, definición, atributos, comportamiento, relaciones y restricciones.
- **Portal.** Sitio web que es o intenta ser un punto de partida para la gente que entra en la Web. Generalmente, ofrecen servicios como motores de búsqueda, noticias, charla o plática (*chat*), correo electrónico, acceso a contenidos personalizados, etc. Hay portales “generales”, que ofrecen información para el público en general. Otros son empresariales, y permiten que las empresas, bien de un mismo sector o de varios, se pongan en contacto, accedan a la información que les interesa y que puedan intercambiar información e incluso hacer negocios electrónicos.
- **Representación del conocimiento.** Se refiere al modo en que el conocimiento humano se describe formalmente en una máquina. El propósito de cualquier representación del conocimiento es capturar toda la información relevante de un dominio y codificarla de manera que las máquinas puedan acceder a ella. Para representar el conocimiento se emplean lenguajes formales (PROLOG, LISP, OIL, DAML, RDF/RDFS, etc.) que expresan cosas e ideas mediante sentencias que pueden ser procesadas por máquinas. Estos *lenguajes de representación del conocimiento* tienen una sintaxis y una semántica. La sintaxis especifica qué configuraciones de símbolos son sentencias válidas (en español, por ejemplo, la sentencia *\*La grande sol* no es válida). La semántica determina a qué hechos se refieren las sentencias.
- **Semántica.** En informática, este término se usa para diferenciar entre el significado de una sentencia y su formato. Supongamos que empleamos en Java una sentencia así: *suma = sumando1 – sumando2*. La sintaxis de la expresión es correcta, pero su semántica



no (salvo que se defina la suma como una resta de números). Los constructores en Java y en C++, por ejemplo, tienen semánticas distintas. En C++, los compiladores llaman internamente a los constructores de las clases; en Java, en cambio, las llamadas a los constructores deben ser explícitamente escritas por el programador. Como sabe cualquier desarrollador, un programa puede compilar correctamente (su sintaxis es correcta), pero eso no entraña que funcione bien: lo que hace (semántica) puede diferir de lo deseado.

- **Servicio web.** Aplicación modular que puede ser llamada a través de Internet y que usa HTTP y estándares abiertos de XML (como SOAP [*Simple Object Access Protocol*, protocolo simple de acceso a objetos], WSDL [*Web Services Description Language*, lenguaje de descripción de los servicios web] y UDDI [*Universal Description, Discovery, and Integration*; integración, descubrimiento y descripción universales]). Mediante los servicios web, las aplicaciones escritas en distintos lenguajes de programación y ejecutadas en distintas plataformas pueden intercambiar datos o, por mejor decir, interoperar. Las aplicaciones que acceden a un servicio web (es decir, los *consumidores del servicio*) no necesitan conocer cómo se ha implementado éste. Los servicios web pueden combinarse para ejecutar operaciones complejas.
- **Sistema de información (SI).** Colección organizada de hardware y software que una organización usa para sus tareas diarias (pagos, pedidos, gestión de nóminas, etc.). Los SI almacenan información, la procesan y permiten acceder a ella. Normalmente, los SI empresariales incluyen bases de datos relacionales.
- **Sofbot (*software robot*, *robot de software*).** Agente inteligente que usa herramientas de software y servicios basados en el comportamiento de una persona. Especialmente populares son los *shopbots* (*shopping robots*, robots de compra o agentes autónomos de compra). Unos son pasivos (a partir de la información introducida por el usuario, buscan información sobre productos); otros son activos (tratan de anticiparse a los deseos de los usuarios ofreciendo propuestas de compra). Los *shopbots* más populares (p. ej., *The Bargain Finder* [El buscador de gangas]) permiten comparar los precios que las tiendas virtuales asignan a un producto. Los *shopbots* más avanzados buscan al mejor precio el producto buscado por el usuario y lo compran automáticamente.
- **Software ERP (*Enterprise Resource Planning*, *planificación de los recursos empresariales*).** Software usado por las empresas para planear y gestionar las funciones comerciales de su negocio: planificación de la producción, seguimiento de los pedidos, contabilidad, gestión de inventarios, recursos humanos, atención a los clientes. Las siglas *ERP* suelen usarse como abreviatura de *software ERP* o *sistema ERP*. Las empresas más conocidas que ofrecen sistemas ERP son SAP, PeopleSoft, Oracle y BAAN. Todo ERP es un SI.
- **W3C (*World Wide Web Consortium*).** Consorcio fundado en 1994 para desarrollar estándares comunes para la Web. Inicialmente, el



W3C estuvo vinculado al CERN (*Centre Européen pour la Recherche Nucléaire*, Centro Europeo de Investigación Nuclear), donde se originó la Web, y tuvo el apoyo de DARPA (*Defense Advanced Research Projects Agency*, Agencia de Proyectos de Investigación Avanzada para la Defensa) y de la Comisión Europea.

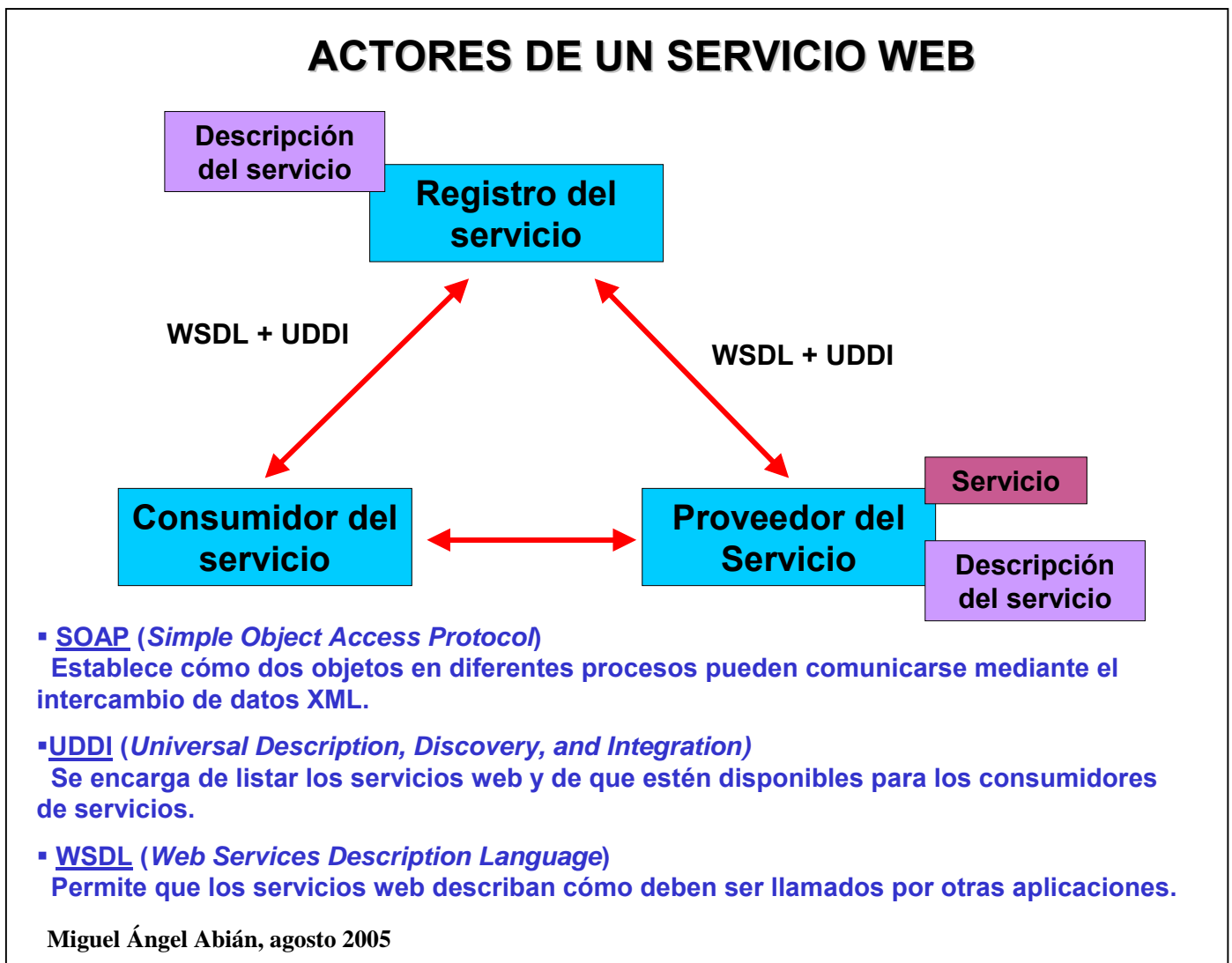


Figura 2. Estructura interna de los servicios web



## 3. Problemas de la Web actual

### 3.1 Introducción

Si bien la Web actual constituye un avance tecnológico impresionante, adolece de dos carencias cruciales. En primer lugar, **no incorpora mecanismos que permitan el procesamiento automático de la información**; esto es, que hagan que la información pueda ser procesada por máquinas. Por ejemplo, un programa buscador de información que acceda a la página web de un profesor universitario desconoce que en el mundo hay personas que trabajan en unas instituciones llamadas universidades y que imparten clases a las que asisten unas personas llamadas alumnos, a quienes el profesor asigna calificaciones según sus conocimientos de la materia impartida en las clases. Ignorando toda esta información, el programa sólo puede realizar un procesamiento muy primitivo de la información presentada en la página del profesor, basado en la búsqueda de palabras clave en el texto de la página. Así, actualmente es impensable hacer consultas del estilo “Obtenga la lista de los antiguos alumnos a los que el catedrático Julio Pellicer impartió la asignatura de Termodinámica en la Universidad de Valencia y que ahora son profesores en universidades españolas”. Ningún programa, por supuestamente inteligente que sea, puede contestar a dicha consulta basándose en las páginas web actuales, pues éstas no almacenan información sobre relaciones del tipo profesor-Universidad, alumno-profesor, alumno-es-ahora-profesor, etc.

En segundo lugar, **la Web actual no incluye mecanismos para la interoperabilidad completa de los sistemas de información (SI) basados en la Web**. Dicho en otras palabras: no facilita la creación de una comprensión común y compartida de un dominio, de forma que ésta pueda ser usada por personas, organizaciones y máquinas. Para que haya una comprensión común de un dominio, los sistemas de información de las partes interesadas en él deben cumplir tres tipos de interoperabilidad:

- **Interoperabilidad técnica.** Se refiere a la capacidad de los SI para intercambiar señales. Esta interoperabilidad exige una conexión física (cable, fibra óptica, ondas electromagnéticas) entre los sistemas y un conjunto de protocolos de comunicaciones (como la pila TCP/IP). La interoperabilidad técnica constituye el primer nivel para lograr la interoperabilidad: si dos ordenadores no pueden intercambiar señales, no puede existir interoperabilidad alguna. Así como al principio los ordenadores de distintos fabricantes no podían comunicarse, actualmente lo raro es lo contrario. La existencia de Internet ha sido posible gracias a la interoperabilidad técnica entre sistemas informáticos muy distintos. Que dos SI interoperen técnicamente **no significa** que puedan intercambiar datos, pues para intercambiar datos se precisa que los SI usen un mismo formato para el intercambio de datos. En lo que sigue no me referiré a la interoperabilidad técnica, pues la Web goza de esta característica. De hecho, es uno de los motivos de su popularidad. **Siempre que me refiera a interoperabilidad de la Web o de las aplicaciones basadas en ella, me estaré refiriendo a la interoperabilidad sintáctica o a la semántica, o a ambas.**
- **Interoperabilidad sintáctica.** Se refiere a la capacidad de los SI para leer datos procedentes de otros SI y obtener una representación que puedan usar (objetos, p. ej.). En definitiva, significa que, en todos los SI involucrados, tanto la codificación de los datos como los protocolos de



acceso son compatibles. Desde luego, esto no implica que los SI procesen los datos de una manera consistente con su significado. La interoperabilidad sintáctica es alta cuando se encuentran disponibles **analizadores sintácticos** (*parsers*) y **APIs** para manipular los datos intercambiados, de manera que las organizaciones pueden usarlos para incorporar esos datos a sus SI. El grado más alto de interoperabilidad sintáctica se alcanza cuando dos SI cualesquiera intercambian datos cualesquiera sin ningún acuerdo previo y de manera completamente automática.

El primer paso para conseguir la interoperabilidad sintáctica consiste en compartir algún tipo de formato de representación de datos (p. ej., almacenar los datos en forma de cadenas de texto dentro de un archivo de texto) para los datos enviados.

El segundo paso pasa por que los datos intercambiados compartan, además de un mismo formato, una misma estructura. Por ejemplo, si el SI de una empresa almacena cada factura en un archivo de texto y reserva una línea del archivo para el campo “Código Postal”, habrá problemas de interoperabilidad sintáctica cuando trabaje con otro cuyas facturas se almacenen en archivos de texto donde no haya una línea para el código postal (porque éste se incluye en el campo “Dirección”, p. ej.). Para que el primer SI pudiese trabajar con facturas del segundo, debería procesar las facturas de éste de modo que a) se extrajera del campo “Dirección” el código postal; b) se almacenara el código postal en el campo “Código Postal”. Desde luego, la solución más sencilla sería usar una misma estructura para las facturas de ambas empresas.

- **Interoperabilidad semántica.** Es la capacidad de los SI para intercambiar información basándose en un común significado de los términos y expresiones que usan. En otras palabras: denota la capacidad de los SI para intercambiar información consistente con el significado que se le supone. Consideremos, por ejemplo, dos SI en que uno trabaja con facturas donde el campo “Importe” significa “Importe de la factura en dólares”; y el otro, con facturas donde ese campo significa “Importe de la factura en euros”. Evidentemente, los dos SI carecen de interoperabilidad semántica.

La interoperabilidad semántica no debe confundirse con la sintáctica: ésta se refiere a la dificultad de procesar automáticamente los datos, no a su “contenido”. Aquella se refiere a la dificultad de relacionar los términos desconocidos que aparecen en los datos con los ya conocidos o definidos. La interoperabilidad semántica no puede existir antes de la interoperabilidad técnica y la sintáctica.

En el comercio electrónico B2B, la falta de interoperabilidad semántica produce grandes problemas: ¿cómo van a entenderse empresas que definen de manera distinta objetos de negocio como “pedido” o “producto”? ¿Cómo van a realizar transacciones electrónicas dos empresas que entienden “precio” de distintas maneras (una, como la cantidad que figura en su catálogo; otra, como esa cantidad más impuestos y más gastos de envío)?

En campos como la medicina, la interoperabilidad semántica resulta crucial: ni las aplicaciones médicas ni los médicos pueden comunicarse – en el sentido estricto del término– si no definen las enfermedades de un



mismo modo. Así, si una aplicación médica tiene como criterio imprescindible para diagnosticar una leucemia que el número de monocitos sea superior al 30%, mientras que otra emplea un 20%, jamás se pondrán de acuerdo en si los pacientes con un 25% de monocitos padecen o no la enfermedad.

La interoperabilidad no es una palabra de moda: las sociedades industriales y postindustriales deben su existencia a la interoperabilidad. El gran logro de Henry Ford no consistió en introducir correas que pasaban los automóviles de trabajador en trabajador, sino en introducir un sistema donde las piezas se podían intercambiar y ensamblar de manera sencilla, es decir, un sistema de componentes interoperables. Antes de Ford, las piezas se construían y se ensamblaban de manera personalizada para cada automóvil.

Igualmente, el éxito de la industria electrónica estriba en la interoperabilidad: pueden crearse circuitos ensamblando componentes de distintos fabricantes; si uno se estropea, puede cambiarse por un componente de otro fabricante. Como saben muchos usuarios informáticos, un ordenador puede montarse ensamblando piezas de distintos fabricantes. Por último, el éxito de la telefonía, ya sea fija o móvil, se debe a la interoperabilidad: redes telefónicas de países distintos, cada una con sus estándares y sus componentes telemáticos, interoperan para que dos personas separadas por miles de kilómetros puedan hablar.

## DEFINICIÓN DE INTEROPERABILIDAD (según IEEE)

Capacidad de dos o más sistemas o componentes para intercambiar información y usar la información que ha sido intercambiada.

1. Interoperabilidad  
técnica

2. Interoperabilidad  
sintáctica

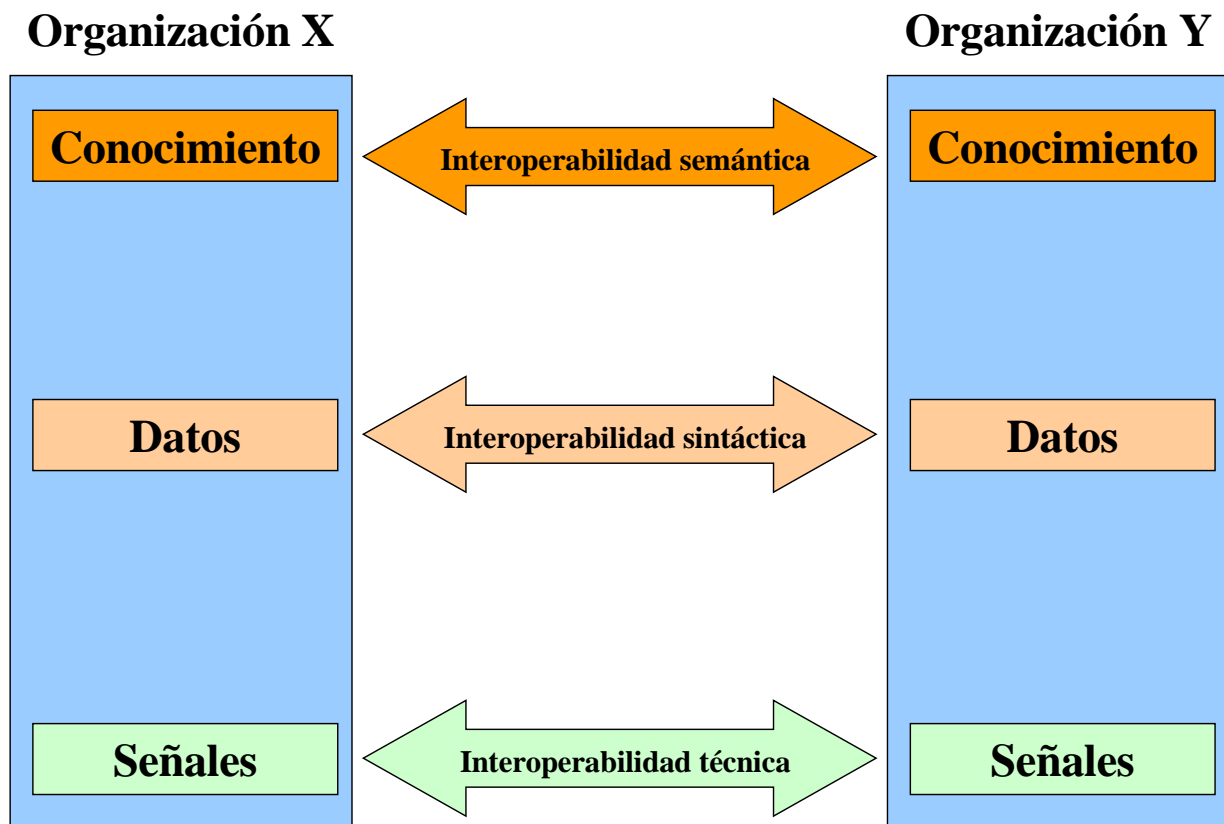
3. Interoperabilidad  
semántica

Miguel Ángel Abián, julio de 2005

Figura 3. Tres tipos de interoperabilidad



## ORDENACIÓN DE LOS NIVELES DE INTEROPERABILIDAD



Miguel Ángel Abián, julio de 2005

**Figura 4. Tres niveles de interoperabilidad. Para conseguir la interoperabilidad en un nivel, debe haberse conseguido en los niveles inferiores**

De la falta de interoperabilidad de la Web se derivan muchos problemas, demasiados para enumerarlos aquí. En lo que sigue consideraré los dos más relevantes desde el punto de vista económico. Esto es, los dos problemas que más caros resultan a las organizaciones y a los usuarios que usan la Web.

Si bien la falta de interoperabilidad no mete directamente la mano en la cartera de nadie, sí tiene dos efectos sobre los consumidores. En primer lugar, causa que los usuarios de la Web no aprovechen todas las ventajas económicas que proporciona Internet. En segundo lugar, hace que tengan que pagar más por los productos ofrecidos en Internet, pues las empresas repercuten en sus productos los costes derivados de la falta de interoperabilidad.



## 3.2 Dificultad para encontrar información

Antes de nada, conviene entender cómo funcionan los buscadores web basados en palabras clave (*Google*, *Yahoo* y *Alta Vista* son algunos de los más populares). En general, constan de tres grandes componentes:

- 1) Un “merodeador de la web” (*webcrawler*), encargado de descargar documentos de la Web.
- 2) Un indexador que extrae los términos clave de los documentos descargados. En *Alta Vista*, estos documentos se representan mediante vectores de términos, en los que aparece la frecuencia de cada término clave en el documento.
- 3) Una interfaz gráfica de búsqueda, mediante la cual hacen sus consultas los usuarios. En *Alta Vista*, las palabras introducidas por el usuario se buscan en una base de datos que almacena los vectores de términos. Luego, se presentan al usuario los documentos que tienen buena correlación con las palabras introducidas.

En el caso de *Google*, la relevancia de los términos no se determina como en *Alta Vista*. *Google* usa un índice de citas o menciones: cuantos más hipervínculos apuntan a un documento, más relevancia se le concede. Los documentos que se muestran antes son los citados más veces en otros documentos.

Vista esa somera explicación de los buscadores web, es momento de considerar el problema de la búsqueda de información. Un ejemplo bastará para entender la magnitud del problema. Supongamos que alguien busca información sobre las antenas de los insectos; para ello, escribe la palabra “antenas” en un buscador web (*Google*, por ejemplo). En la búsqueda aparecerán miles de resultados: muchos se referirán a aparatos de transmisión y recepción de ondas electromagnéticas; otros se referirán a las antenas de los caracoles; algunos otros se referirán a las antenas de insectos que sólo los especialistas saben que existen; unos pocos se referirán a las antenas de los deformes peces que pueblan las fosas oceánicas... El usuario sabe sobre qué tipo de antenas busca información, pero no puede transmitirla directamente al buscador. En otras palabras: no puede dar al navegador órdenes como “Busco información sobre las antenas de los insectos, no sobre los artilugios de ondas electromagnéticas”.

El ejemplo manifiesta que *Google* carece de “inteligencia”: concede la misma importancia a una página web sobre la venta de antenas parabólicas, siempre que esté poco referenciada, que a una sobre la viscosidad de las antenas del *Helix aspersa* (por definición, estas páginas tienen pocas referencias). También muestra que cualquier buscador web se enfrenta con una cruda realidad: en cualquier lenguaje natural abundan las palabras sinónimas o polisémicas (“antena” es un buen ejemplo). El funcionamiento de los buscadores web actuales, explicado brevemente al principio de este subapartado, resulta bien simple: se buscan las palabras de la búsqueda en los documentos de la Web, sin atender a la sinonimia o polisemia de las palabras. Como el lector o la lectora sabrá por propia experiencia, eso provoca que a veces sea necesario consultar docenas o cientos de páginas hasta encontrar la información deseada. Hasta el momento, la intervención humana resulta imprescindible para seleccionar la información que uno desea.

Si las páginas web se escribieran en un lenguaje que permitiera almacenar **semántica**, los buscadores web encontrarían no sólo las páginas donde aparecieran las



palabras de la búsqueda, sino también todas aquellas páginas donde hubiera sinónimos de esas palabras. Por ejemplo, una búsqueda basada en el término “modelado empresarial” encontraría automáticamente páginas donde aparecieran términos como “modelado de la empresa”, “modelado de las compañías”, etc. De la misma manera, una búsqueda basada en el término “termitas” mostraría páginas con los términos *Reticulitermes lucifugus* Rossi, *Cryptotermes brevis* Walker y *Kaloterms flavicollis* Fabr.

No son los buscadores web los responsables últimos de la dificultad para encontrar información en la Web, sino el lenguaje con que se construyen las páginas web: HTML. Este lenguaje de etiquetado sólo permite especificar cómo se va a mostrar una página HTML, pero resulta inútil para dar “contexto” a una página. Por ejemplo, HTML no permite expresar que la palabra “antenas” en un documento está relacionada con un insecto blancuzco y famoso por su voracidad hacia la madera, y no con un emisor o receptor de radiofrecuencias. En pocas palabras: HTML no proporciona semántica a los datos. Una sentencia HTML como

<H1> Soy lector de javaHispano </H1>

sólo indica que el dato “Soy lector de javaHispano” debe mostrarse como un encabezado de tipo 1. Con esta información, una aplicación sabe cómo procesar gráficamente el texto, pero desconoce cómo interpretarlo. En definitiva, con HTML es imposible generar información susceptible de ser “comprendida” por máquinas (en el apartado 4 explicaré qué significa que una máquina comprenda la información), de modo que se evite la intervención humana para seleccionar la información relevante.

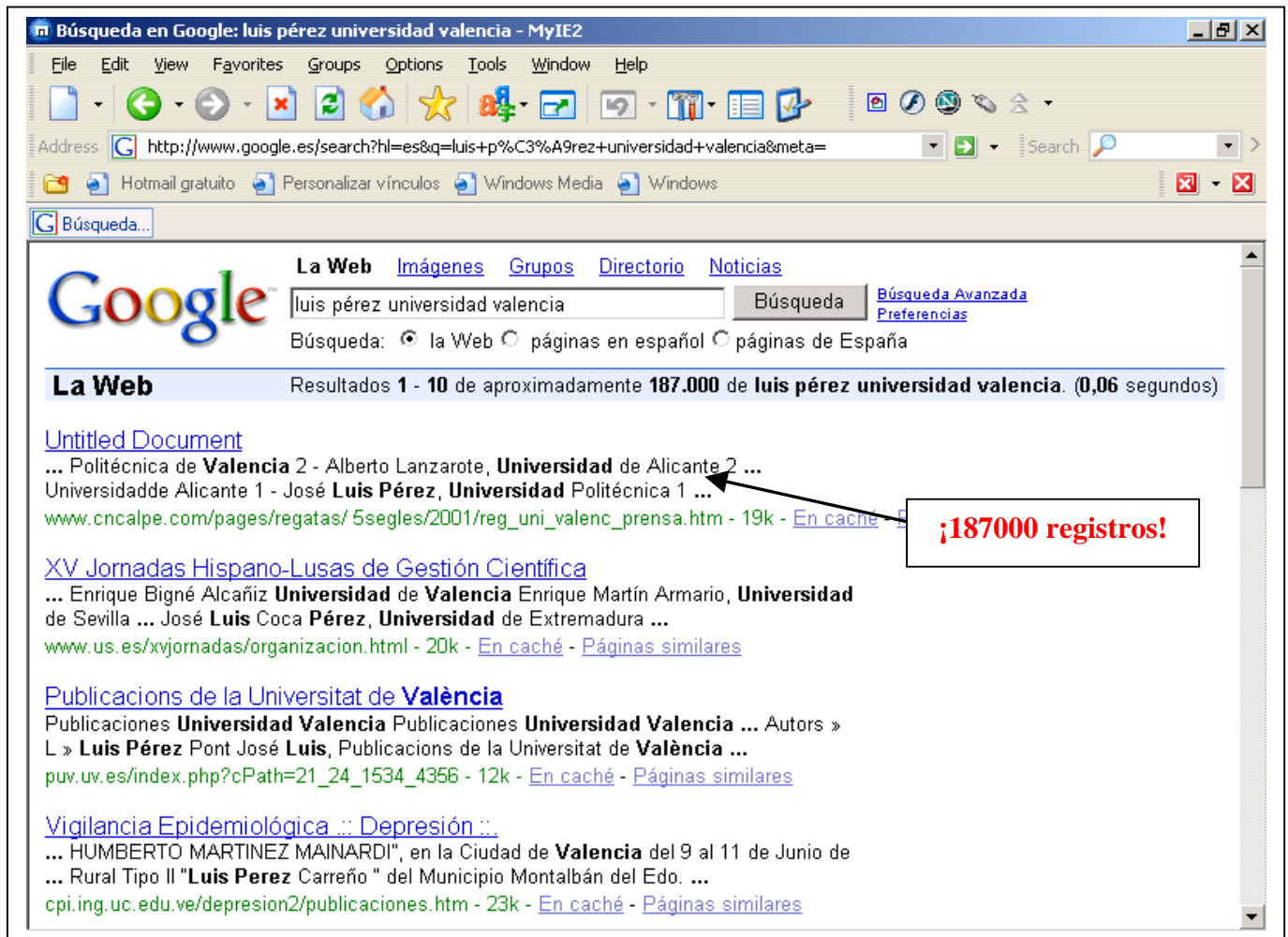
Resumiendo, éstos son los dos principales problemas con que se encuentran los usuarios cuando buscan información en la Web:

- **Escasa precisión en los resultados.** Algunas búsquedas generan decenas de miles de páginas que carecen de interés; otras, por el contrario, no generan ninguna.
- **Alta sensibilidad al vocabulario empleado en la búsqueda.** Si los documentos que verdaderamente interesan no emplean el mismo vocabulario que la búsqueda, jamás aparecerán. Así, si uno busca mediante la palabra clave “espín”, no encontrará ninguno de los documentos en español donde sólo se usa el término inglés “spin”.

Ambos problemas ocasionan que el usuario nunca tenga la garantía de que ha encontrado toda la información relevante para su consulta. En el caso de Google, vaya por caso, se concede más importancia a mostrar los documentos más citados (los que mayor índice de citas tienen) que a mostrar todos los documentos que podrían interesar al usuario.

Por si la falta de inteligencia de los buscadores web no fuera bastante obstáculo para encontrar información, no debemos olvidar que no pueden acceder a la información almacenada en imágenes, vídeos o archivos de sonido. En todo caso, pueden acceder a las descripciones textuales de esos archivos. Por ejemplo, una imagen donde aparezcan las antenas de una abeja reina podrá mostrarse en un buscador si junto a la imagen hay alguna etiqueta HTML que proporcione una descripción de la imagen (“antenas abeja reina”, por ejemplo) o si el nombre de la imagen contiene esa descripción. En caso contrario, el buscador jamás la encontrará.





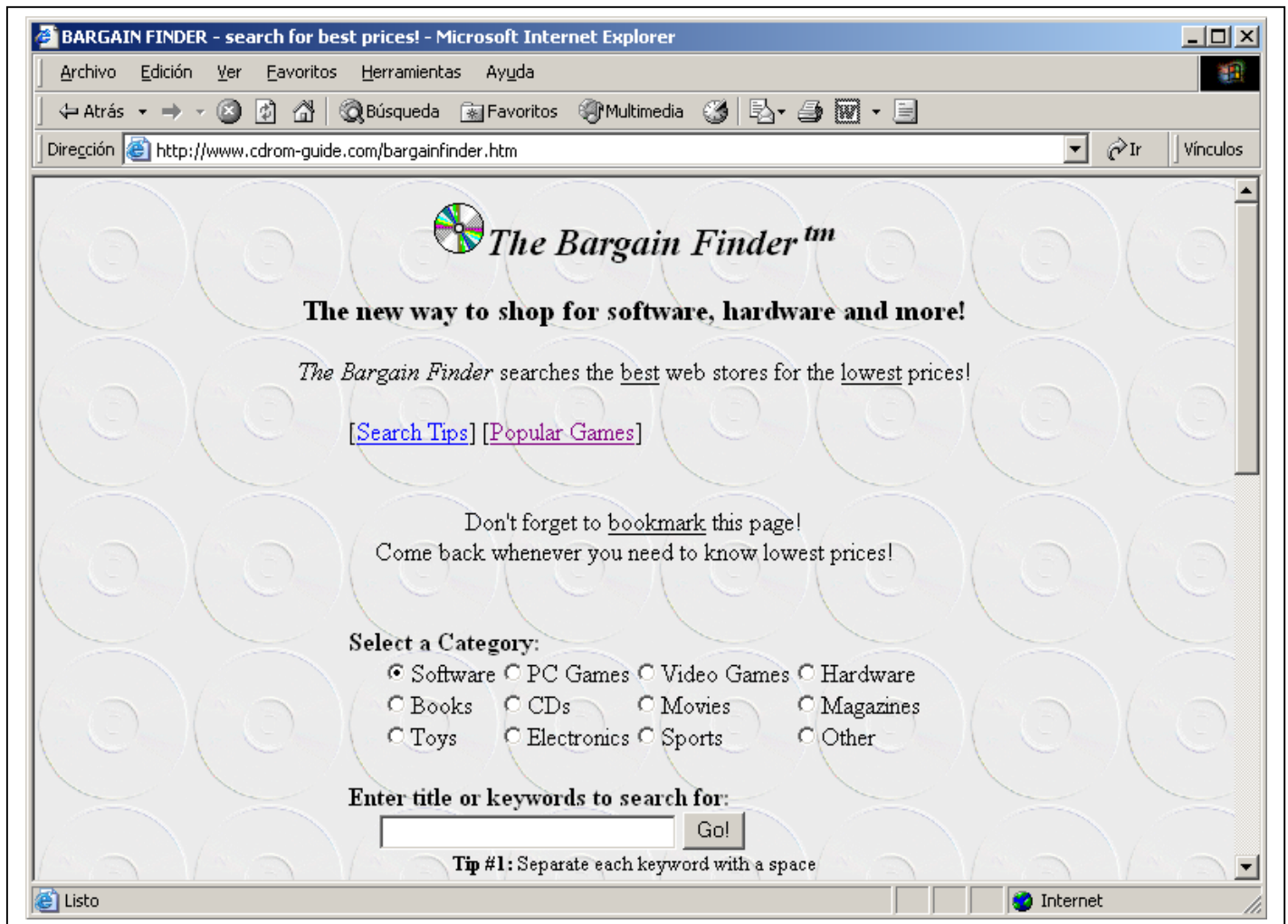
**Figura 5. Ejemplo del problema de buscar información. Si uno pretende encontrar a un compañero de universidad mediante un buscador, se encuentra con miles de las páginas, de las cuales sólo unas pocas (o ninguna) serán útiles**

Las dificultades para encontrar información aparecen también en el comercio electrónico. En el caso del comercio **B2C**, los consumidores encuentran tedioso comparar precios de un mismo producto. Supongamos, vaya por caso, que una persona desea comprar una crema de afeitar muy cara. Si quiere ahorrar dinero, comparará el precio de la crema en distintas tiendas virtuales y la comprará en la más barata. Enseguida llegarán los problemas: en unas tiendas, la crema aparecerá en la categoría de "Higiene masculina"; en otras, en la de "Cosmética"; en algunas otras, en la de "Parafarmacia"... Para encontrar la crema, el aspirante a cliente tendrá que buscar de un modo distinto en cada tienda, y las tiendas abundan en la Web. La búsqueda de precios mediante un buscador web tampoco resulta muy útil: al introducir el nombre de la crema y la palabra "precio" aparecerán cientos o miles de páginas donde "precio" no se refiere a esa crema (por ejemplo, en una página web de opinión, un usuario puede haber mencionado la crema en cuestión y haber escrito el precio de otro producto).

El uso de **agentes inteligentes** tampoco resuelve el problema de encontrar la tienda donde un producto esté más barato. Uno puede pensar que la solución pasa por escribir o comprar un **shopbot** o agente autónomo de compra y configurarlo para que visite todas las tiendas virtuales que ofrecen el producto y obtenga su precio en cada



una; pero no es así: debido a las múltiples categorías en que se clasifica un mismo producto, a las diferentes presentaciones de los catálogos (tablas HTML, archivos Word, Excel, PDF, etc.) y a las variadas estructuras internas de las páginas web de las tiendas y de los proveedores, cualquier agente actual deja fuera de la comparación a muchísimas tiendas. Además, cualquier agente autónomo de compra se siente desconcertado por la gran cantidad de palabras que usan los vendedores para definir sus productos ("producto", "ítem", "artículo", "product"...), y su binario desconcierto se traduce en que no reconoce muchos productos. El sueño de cualquier consumidor –encontrar los productos al precio más reducido posible– sigue requiriendo la intervención humana.



**Figura 6. The Bargain Finder es un *shopbot* (agente autónomo de compra) pasivo. Permite encontrar las tiendas que ofrecen los mejores precios para ciertos artículos**

Las dificultades para encontrar información no atañen sólo a los particulares. El tiempo que las empresas dedican a buscar información entre los millones de páginas de Internet y de las intranets empresariales equivale a billones de euros. Por ejemplo, un informe de una consultora estadounidense, realizado en 2000, analizó el tiempo que cuatrocientas organizaciones europeas y estadounidenses (públicas y privadas) invertían en buscar información en la Web y en sus intranets. El resultado fue que los empleados



empleaban una media de ocho horas semanales en buscar información, lo que equivalía a unos cien billones de dólares.

Las empresas de tamaño pequeño y medio no suelen tener problemas para encontrar información en sus intranets. Por el contrario, las empresas grandes que almacenan en sus intranets miles o millones de páginas deben compensar la falta de inteligencia de sus buscadores mediante la inteligencia y el tiempo de sus empleados.

La solución más inmediata para el problema de buscar información pasa por incluir **metadatos** –esto es, datos que describen a los datos– en las páginas web. Los metadatos son *como* etiquetas que describen qué significa cada dato. Por ejemplo, cualquier sistema de gestión de bases de datos relacionales (SGBDR) almacena metadatos sobre los datos. Imaginemos una base de datos de películas: el SGBDR correspondiente asigna a todos los datos de una misma columna una descripción (*Nombre\_Director*, *Fecha\_Estreno*, *Actor\_Principal*, *Actriz\_Principal*, *Productor*, *Recaudacion*, etc.) y un tipo de datos (*INTEGER*, *STRING*, *DATE*, *CURRENCY*, etc.). Toda estos metadatos se guardan en la propia base de datos de películas. (A veces, los metadatos son realmente etiquetas: por ejemplo, el código de barras de cualquier producto es un metadato.)

En el caso concreto de la crema de afeitar, si cada tienda virtual que la vende incluyera un metadato numérico *Precio* cada vez que apareciera el nombre de la crema (este metadato no tiene por qué ser visible para el usuario), los buscadores web tendrían más fácil su tarea: sólo tendrían que buscar las páginas donde el nombre de la crema aparece vinculado al metadato *Precio*. En general, los metadatos permiten que las consultas sean más precisas y no devuelvan tantas páginas inútiles para el usuario.

Si bien el uso de metadatos facilita la búsqueda automática de información, dista mucho de ser una solución perfecta. Volvamos al ejemplo de la crema: si consideramos tiendas por todo el mundo, resulta impensable que todas usen un metadato *Precio*. Unas usarán *Price*, otras *Prix*, algunas otras *Preis*... ¿Cómo puede saber el buscador web que estos metadatos se refieren al mismo concepto? Mejor aún: ¿en verdad se refieren esos metadatos al mismo concepto? En el Reino Unido, los precios vienen expresados en libras esterlinas; en la Europa Occidental, en euros; en Estados Unidos, en dólares... Para que el buscador pudiera comparar precios en tiendas de todo el mundo, necesitaría una especie de biblioteca de términos equivalentes (*Precio*, *Price*, *Prix*, *Preis*...) y conocer la relación exacta entre ellos, o precisaría que las tiendas usaran unos mismos metadatos a la hora de describir los productos.

### 3.3 Dificultad para implantar comercio electrónico B2B

En el presente, el principal reto al que se enfrentan las empresas es la **integración** con otras (clientes, proveedores, fabricantes y subcontratistas). Para lograr esto, los SI de las empresas deben permitir el intercambio de información interempresarial, ya sea comercial (albaranes, facturas, pedidos, etc.) o de carácter logístico o de planificación. Compartir información no resulta fácil: cada SI usa distintos formatos, definiciones, modelos de negocio...

La mejor manera de integrar los procesos de negocio y los SI de las organizaciones que forman parte de una misma **cadena de aprovisionamiento** se basa en usar EDI (*Electronic Data Interchange*, intercambio electrónico de datos). EDI es el intercambio



automatizado de documentos comerciales electrónicos entre una organización y sus socios comerciales, de modo que apenas se requiere la intervención humana. La DISA (*Data Interchange Standards Association*, Asociación de Estándares de Intercambio de Datos), proporciona la siguiente definición de EDI:

[es] el intercambio, de ordenador a ordenador, de datos de negocio en formatos estándares. En EDI, la información se organiza de acuerdo con un conjunto de formatos especificado por ambas partes, los cuales permiten una transacción informática automática [en el original, “hands off”] que no requiere en cada extremo ninguna intervención humana ni volver a teclear. La información contenida en un conjunto de transacciones EDI es, en su mayor parte, la misma que en los documentos impresos convencionalmente.

EDI se basa en la idea de que los datos deberían introducirse manualmente una vez y, luego, ser pasados electrónicamente a las otras partes. Así se evita introducir los datos una y otra vez en cada organización (lo cual requiere la intervención humana y resulta propenso a errores) y se eliminan los documentos en papel. Mediante el uso de EDI, el procesado y la transferencia de información es mucho más rápida, eficaz y exacta que mediante el intercambio manual de papel o mediante la comunicación por teléfono o fax. Indudablemente, EDI favorece la productividad de las organizaciones (menos errores, menos papeleo, transmisión rápida de las órdenes, entregas más rápidas de los pedidos).

En cualquier empresa típica, un proceso EDI comienza cuando la empresa envía a su proveedor una orden EDI de compra. A continuación, el proveedor devuelve al comprador una confirmación de la orden de compra. Cuando el proveedor tiene ya todos los productos solicitados, envía al comprador una nota EDI de próxima entrega. Cuando se produce la entrega, el proveedor envía una factura EDI; finalmente, el comprador envía una orden de pago al banco con que trabaja, por la cual se transfiere dinero de la cuenta del comprador a la del vendedor.

La meta última de EDI radica en **compartir el conocimiento** de todas las empresas que forman parte de una misma cadena de aprovisionamiento, de manera que cada empresa amplíe sus SI más allá de sus fronteras e integre en sus procesos de negocio los de otras empresas involucradas en su cadena de aprovisionamiento. Este conocimiento compartido alberga un gran valor económico y optima la cadena de aprovisionamiento: permite suprimir intermediarios, gestiona mejor los procesos de suministro, proporciona plazos precisos y actualizados de la entrega de productos, permite hacer menos inventarios y utilizar mejor los almacenes, avisa de posibles retrasos, hace posible que el cliente especifique sin ambigüedades el diseño de los productos, posibilita que los distribuidores aprovechen mejor los transportes y ahorren costos, etc. En resumen, EDI permite integrar los SI de los fabricantes, proveedores y distribuidores para que los productos se produzcan y se distribuyan en las cantidades adecuadas, en el momento correcto y de la manera más económica posible.

Hasta hace pocos años, el comercio electrónico B2B seguía el **enfoque EDI convencional o tradicional**, consistente en que los SI de las empresas intercambian mensajes EDI –es decir, mensajes de contenido, significado y formato normalizados– a través de redes de valor añadido (VAN, *Value Added Networks*). ANSI X12 (Estados Unidos) y UN/EDIFACT (resto del mundo) son los estándares más conocidos para establecer la estructura y el significado de los mensajes entre organizaciones, si bien existen muchos otros formatos (la mayoría, propietarios e incompatibles con los demás). ANSI X12 y UN/EDIFACT definen una serie de documentos comerciales –facturas,



pedidos, notas de entrega, etcétera—, así como documentos más especializados (p. ej., para usos gubernamentales o médicos). Para los negocios entre dos partes, sólo se implementa un subconjunto del estándar elegido (estos subconjuntos se conocen como directrices de implementación).

Las VAN son redes de telecomunicaciones, privadas, a las que las empresas suscriptoras se conectan para intercambiar documentos EDI. Estas redes proporcionan servicios de buzón de correo (los documentos que se envían a una empresa se almacenan en un buzón, de donde ésta puede recogerlos cuando desee) y se encargan de gestionar la red: garantizan la seguridad y confidencialidad de las comunicaciones, certifican ante terceros la entrega de los mensajes... Muchas VAN son redes de punto a punto o X.25 en las que se paga por la cantidad de información transmitida; a diferencia de Internet, los protocolos de red que usan suelen ser propietarios.

Como los SI no usan internamente los contenidos, formatos y significados de los documentos EDI, las organizaciones deben instalar “traductores” EDI que transformen los formatos y significados de los documentos internos a los formatos y significados de los mensajes interempresariales, y viceversa. Si el volumen de datos intercambiados no es muy grande, una o más personas se pueden encargar de convertir los datos de un formato a otro, pero esta solución resulta imposible en cualquier empresa que intercambie diariamente cientos o miles de mensajes. Lógicamente, el intercambio de mensajes normalizados se torna innecesario si los SI de las organizaciones son idénticos o si se han construido según una arquitectura de integración, casos en que pueden intercambiarse directamente los datos de cada SI.

En un sistema EDI convencional, el envío de un mensaje EDI conlleva cinco pasos:

- 1) En el SI de la organización emisora se exportan los datos de interés a un archivo.
- 2) Se convierte el archivo de datos al formato ANSI X12 o UN/EDIFACT
- 3) Se transfiere el archivo de datos al SI de la segunda organización (organización receptora).
- 4) Se convierte el formato del archivo (ANSI X12 o UN/EDIFACT) al formato del SI receptor.
- 5) El SI receptor importa los datos, los verifica, los edita y los distribuye.

Actualmente, los **sistemas ERP** suelen tener módulos EDI que facilitan la traducción de unos formatos a otros. Estas soluciones resultan caras para las pequeñas y medianas empresas y, además, requieren un cierto trabajo de personalización.



**PURCHASE ORDER**  
**UNB+UNOB:1+003897733:01:MFGB-PO+PARTNER ID:ZZ+000101:1050**  
**+000000000000916++ORDERS'**  
**UNH+1+ORDERS:S:93A:UN'**  
**BGM+221+P1M24987E+9'**  
**DTM+4:20000101:102'**  
**FTX+PUR+3++PURCHASE ORDER BEFORE LINE ITEM**  
**INSTRUCTIONS'**  
**RFF+CT:123-456'**  
**NAD+SE+10025392::92++SUPPLIER NAME'**  
**CTA+SR+:STEVE'**  
**NAD+BT+B2::92++COMPAQ COMPUTER CORPORATION+P O BOX**  
**692000+HOUSTON+TX+77692000+US'**  
**NAD+BY+MFUS::92++COMPAQ COMPUTER CORPORATION'**  
**CTA+PD+:CLARETTA STRICKLAND-FULTON'**  
**NAD+ST+CM6::92++COMPAQ COMPUTER CORPORATION+CCM6**  
**RECEIVING DOCK:20555 SH**  
**249+HOUSTON+TX+77070+US'**  
**TAX+9++++++3-00105-5135-3'**  
**CUX+2:USD:9'**  
**PAT+1++1:1:D:45'**  
**PCD+12:2'**  
**TDT+20++++::AIRBORNE'**  
**LOC+16+COMPAQ DOCK'**  
**TOD+2+NS+:::ORIGIN COLLECT'**  
**IA+1+AA:EC+123456:VP'**  
**IMD+F+8+:::PART DESCRIPTION INFORMATION**

**Figura 7. Ejemplo de una orden de compra en formato UN/EDIFACT (*United Nations/Electronic Data Interchange for Administration, Commerce and Transport*). Como puede apreciarse, la legibilidad es sumamente baja**



```
ISA*00*0000000000*01*01*PASSWORDME*01*123456789bbbbbb*9
87654321bbbbbb*890714*2210*U*00204*000000008*0*P*:
GS*IN*012345678*087654321*900509*2210*000001*X*002040
ST*810*0001
BIG*900713* 1001 *950625*P98932
N1*BT*ROYAL DISTRIBUTING COMPANY
N3*P.O. BOX 22327
N4*MYTOWN*NJ*42131
N1*ST*MYCORNER
N3*502 STREET
N4*CROSSROADS*MI*43103
N1*SE*GLOBAL CORPORATION
N3*920 MY STREET
N4*MY CITY*NJ*15445
PER*AD*A.MARTIN*TE*623435812
ITD*01*3*2**IO
ITI**3*CA*12.75**VC*7100
ITI**12*EA*.475**VC*P450
ITI**4*EA*.94**VC*1350Y
ITI**I*DZ*3.4**VC*1507
TDS*5111
CAD*M***TRAVELLER COMPANY
CTT*4*20
SE*21*000001
GE*1*000001
IEA*1*000000008
```

**Figura 8. Ejemplo de una factura en formato ANSI X12 (*American National Standards Institute*). El número que aparece tras la etiqueta (810) identifica el documento como una factura**

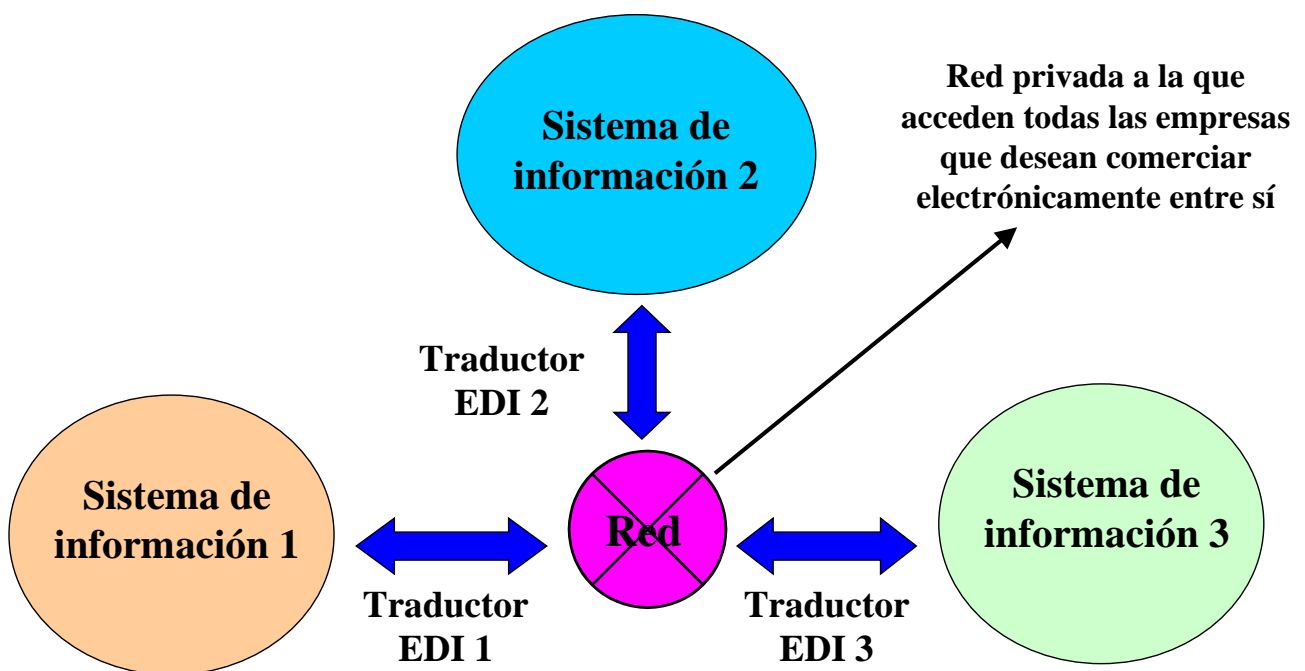
El enfoque EDI convencional tiene dos problemas fundamentales. Por un lado, el elevado coste que supone desarrollar aplicaciones traductoras, completamente hechas a medida (dependen del SI de cada empresa y del formato elegido para intercambiar datos). Consideremos, por ejemplo, una pequeña y mediana empresa (PYME) que fabrica sofás. Si tenemos en cuenta que tiene que trabajar con tiendas y con proveedores de madera, tablero aglomerado, colas y telas, nos daremos cuenta de la dificultad económica que supone desarrollar un traductor para cada par empresa-proveedor o empresa-cliente. Incluso si todas las empresas se pusieran de acuerdo en usar un formato común para el intercambio de datos, cada una debería desarrollar e instalar su propio traductor.

Por otro lado, este enfoque adolece de **inflexibilidad**: una empresa que quiera hacer negocios con un grupo de empresas que ya tienen un sistema de comercio B2B no podrá intercambiar información con las otras hasta que no tenga en marcha su traductor EDI. Si bien el enfoque EDI convencional proporciona interoperabilidad sintáctica, ésta se halla restringida a pequeños entornos cerrados de empresas, casi siempre



pertenecientes a un mismo sector. Imaginemos una empresa que vende un producto sanitario a un precio un 30% más barato que sus competidores. Si la empresa decide integrarse en un grupo de empresas que hacen negocios B2B (verbigracia, para venderles directamente su producto o automatizar su sistema de compra de materias primas), necesitará un tiempo para desarrollar, depurar y configurar el sistema traductor encargado de transformar sus documentos internos al formato normalizado que las otras empresas usan para intercambiar mensajes (y viceversa). Cuando tenga listo el traductor, habrá desaparecido la ventaja competitiva que tenía y su oportunidad de hacer negocio se habrá esfumado.

## INTERCAMBIO DE DOCUMENTOS EN EL ENFOQUE EDI CONVENCIONAL



Miguel Ángel Abián, julio de 2005

Figura 9. Intercambio de documentos en el enfoque EDI convencional

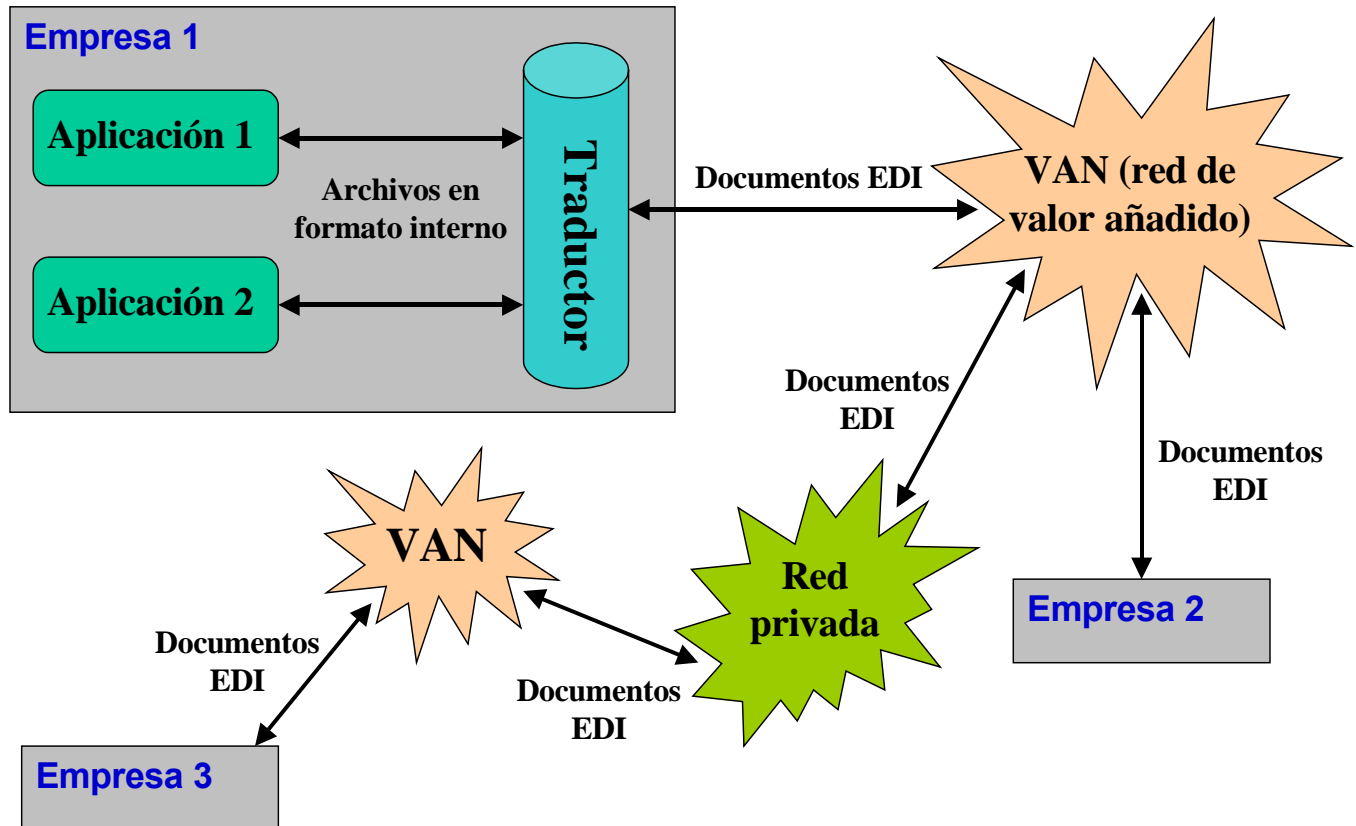
Antes de la llegada de Internet, el comercio B2B era un lujo. Pocas empresas podían permitirse el comercio B2B, pues la transferencia de los mensajes EDI se realizaba mediante redes privadas, como las VAN. Las VAN se encargaban de transmitir físicamente los mensajes EDI, de almacenarlos y también de los aspectos concernientes a la seguridad y a la administración (identificación de los usuarios, certificación ante terceros de la correcta entrega de los mensajes, confidencialidad, etc.). Así pues, la empresa que deseaba hacer **negocios electrónicos** con otras no sólo debía desarrollar sus traductores EDI, sino que también tenía que pagar el coste de la VAN.

(Las redes de valor añadido no se han extinguido completamente. Aún tienen una cierta relevancia en Japón, Europa y Estados Unidos, sobre todo en el sector del automóvil. Si sobreviven es, primero, porque las empresas involucradas quieren



rentabilizar la enorme inversión que estas redes supusieron. Segundo, porque las VAN garantizan un nivel de servicio –ancho de banda, fiabilidad– muy superior a Internet: es natural que se prefiera una VAN para aplicaciones críticas (bancarias, militares, de telemedicina).

## COMERCIO B2B CON EL ENFOQUE EDI CONVENCIONAL



Miguel Ángel Abián, julio de 2005

**Figura 10. El enfoque EDI convencional**

Internet ha facilitado grandemente que cualquier empresa puede desarrollar comercio B2B. Los protocolos de aplicación que se usan en la Web (HTTP, FTP, SMTP, POP) permiten que las empresas puedan intercambiar documentos a través de Internet, sin necesidad de usar redes de valor añadido. Sin embargo, la Web no ha resuelto el problema fundamental del enfoque EDI convencional: la exigencia de usar programas traductores para intercambiar datos entre distintos SI. La Web no proporciona por sí misma ningún protocolo de intercambio de mensajes comerciales: actúa como medio de transporte barato entre los mensajes que se envían las empresas, pero no “traduce” el formato y el significado de éstos.

El lenguaje de la Web, HTML, sirve para decir a los navegadores cómo deben mostrar los documentos, pero no especifica la estructura de los datos ni su semántica. Consideremos, vaya por caso, una empresa que tiene un sitio web donde publica los



pedidos que hace a sus proveedores (obvio cualquier aquí consideración sobre la seguridad y la confidencialidad). El proveedor que acceda al sitio web encontrará un documento HTML semejante a éste:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

```
<HTML><HEAD><TITLE>Pedido 1453</TITLE>
```

```
<META http-equiv=Content-Type content="text/html; charset=windows-1250">
```

```
<BODY text=#ffffff bgColor=#000000>
```

```
<P>Estimado señor García:</P>
```

```
<P>Necesitamos que nos entreguen lo antes posible 500 estanterías metálicas de dimensiones 1.500 x 450 mm, que corresponden al código A341DF de su catálogo. El catálogo que manejamos es del 2004.</P>
```

```
<P>Tenga en cuenta el descuento que nuestra empresa tiene por ser cliente habitual (12%). Necesitamos que nos envíen urgentemente las estanterías. El pago será el habitual (giro a 90 días).</P>
```

```
<P>Atentamente,</P>
```

```
<P>Luis Pérez</P>
```

```
<P>Gerente de PRODUCTOS METÁLICOS S.L.</P>
```

```
</BODY>
```

```
</HTML>
```

Ningún sistema de información puede manejar este tipo de documentos: no saben qué hacer con ellos. Para que el pedido llegue a buen puerto, se necesita a una persona que lea el documento y extraiga los datos relevantes: unidades, 500; producto, estantería; dimensiones: 1.500 x 450 mm; referencia del producto: A341DF; descuento: 12% (revísese si realmente PRODUCTOS METÁLICOS S.L. es cliente habitual, que hay mucho pícaro suelto); urgencia: alta; forma de pago: giro a 90 días. Después, esa persona u otra deberá introducir manualmente esos datos en el sistema de información de la empresa proveedora. Por ahora, ningún SI puede deducir del documento anterior que debe realizar acciones tales como insertar un pedido de 500 estanterías en la base de datos de pedidos o verificar que PRODUCTOS METÁLICOS S.L. es un buen cliente.

Así las cosas, hacer negocios electrónicos con el lenguaje estándar de la Web no difiere mucho de hacerlos mediante teléfono o fax: al final, algunos seres humanos deben traducir la información a términos que los SI puedan procesar e introducirla en ellos.

En resumidas cuentas, la Web proporciona interoperabilidad técnica para el comercio B2B, pero no interoperabilidad sintáctica ni semántica. Estas carencias suponen un gran coste para las empresas, cuantificable en un 50-60% del coste de los proyectos relacionados con el comercio B2B.



**Nota:** En la Web, dos o más empresas siempre pueden hacer negocios B2B de este modo: poniéndose antes de acuerdo sobre el formato y el significado de los mensajes que intercambien (como sucede con el enfoque EDI convencional). Estrictamente hablando, tendrían interoperabilidad; pero de un modo parcial y restringido, pues no es ése el tipo de interoperabilidad que interesa en un entorno web. Ese enfoque del comercio electrónico no considera las propiedades de la Web, donde una empresa tiene a menudo que hacer negocios con empresas con las que jamás había trabajado antes y con las que quizá nunca vuelva a trabajar. En un entorno de negocios con muchas empresas y donde éstas cambian continuamente, no resulta práctico que todas se reúnan y se pongan de acuerdo *a priori* sobre el formato y el significado de los mensajes comerciales, ni que adapten sus SI a estas necesidades.



## 4. La Web semántica

La solución a los problemas anteriores vendrá de lo que se conoce como la **Web semántica**. Tim Berners-Lee, creador de la WWW, es el inventor del concepto. Dejemos que él mismo aclare qué es la Web semántica:

El primer paso es colocar los datos en la Web de un modo en que las máquinas puedan entenderlos naturalmente o convertirlos a esa forma. Esto crea lo que yo llamo una Web semántica: una red de datos que pueden ser procesados directa o indirectamente por máquinas.

**[Weaving the Web, 1999]**

La Web semántica es una extensión de la actual Web en la cual la información se da mediante un significado bien definido, lo que facilita que los ordenadores y la gente trabajen en cooperación.

**[The Semantic Web, Scientific American, mayo de 2001]**

La Web semántica proporcionará estructura al contenido importante de las páginas web, creando un entorno donde los agentes de software que viajan de página en página puedan enseguida llevar a cabo complicadas tareas para los usuarios. Así, un agente que venga de la página web de la clínica no sólo sabrá que la página tiene palabras como “tratamiento, medicina, médico, terapia” (como puede codificarse hoy), sino también que el Dr. Hartman trabaja en la clínica los lunes, miércoles y viernes, y que el *script* [programa no compilado realizado en un lenguaje de programación sencillo] toma un rango de datos en el formato año-mes-día y devuelve fechas y horas de consulta.

**[The Semantic Web, Scientific American, mayo de 2001]**

Otra definición interesante es ésta:

Otro esquema de representación del conocimiento que procede de la Inteligencia Artificial, la red semántica, se basa en redes similares de conceptos interdependientes, pero aquí las dependencias se clasifican en distintos tipos con interpretaciones específicas. Por ejemplo, los diferentes tipos de relaciones podrían especificar que un concepto *a* “causa” otro concepto *b*, que “es una parte de *b*”, o que *a* “es un caso especial de *b*”. La motivación tras las redes semánticas es que los conceptos obtienen su significado mediante las relaciones semánticas que tienen con otros conceptos.

[\[http://framework.v2.nl/archive/archive/node/text/default.xslt/nodenr-156647\]](http://framework.v2.nl/archive/archive/node/text/default.xslt/nodenr-156647)

La definición oficial de la Web semántica dada por el **W3C** se resume aquí:

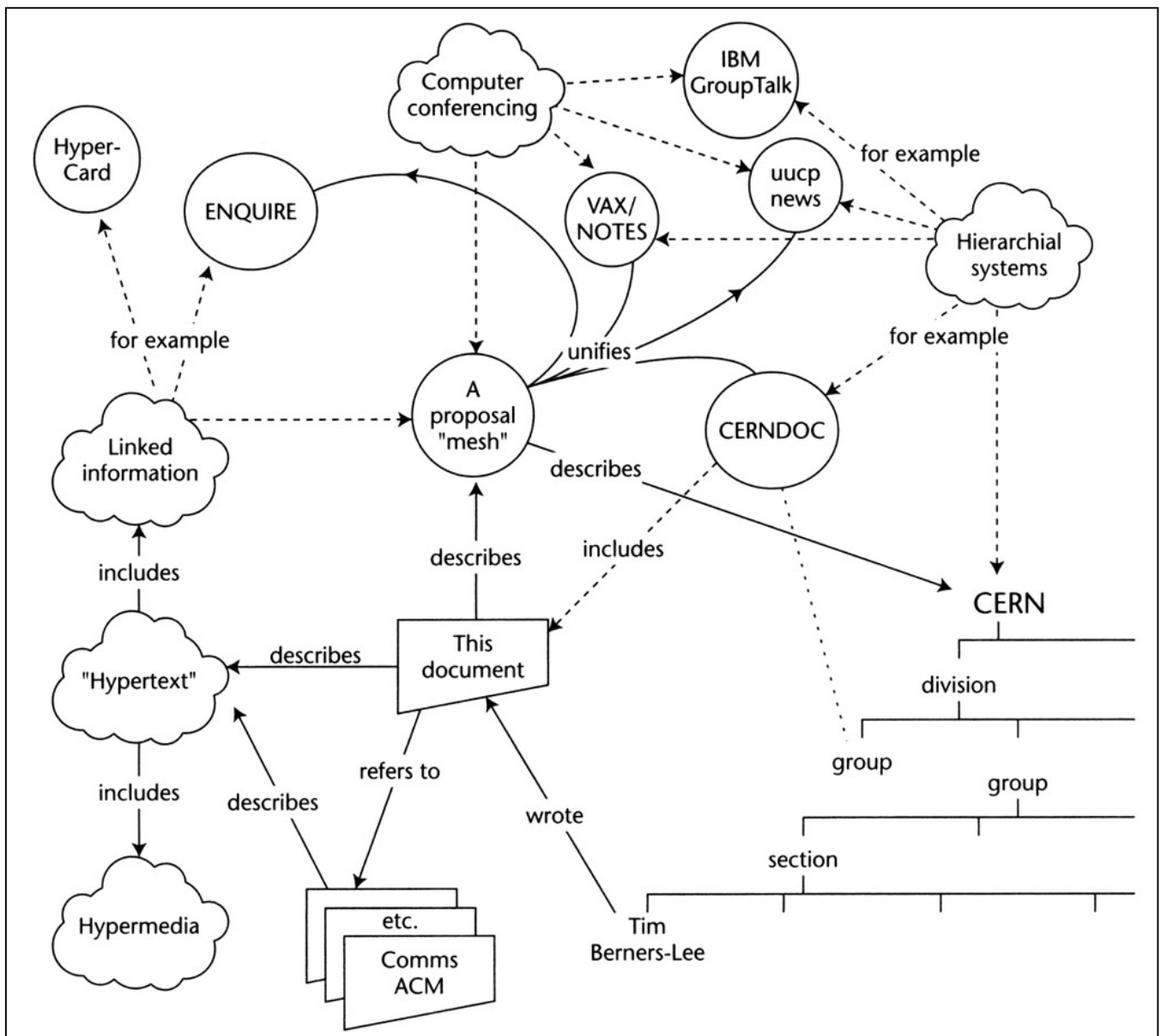
El propósito de la iniciativa de la Web semántica es tan amplio como el de la Web: crear un medio universal para el intercambio de datos. Se considera para interconectar eficazmente la gestión de la información personal, la integración de las aplicaciones empresariales y compartir globalmente datos comerciales, científicos y culturales. Los servicios para poner datos comprensibles por las máquinas se están convirtiendo rápidamente en una prioridad para muchas organizaciones, individuos y comunidades.

La Web sólo puede alcanzar todo su potencial si llega a ser un sitio donde se puedan compartir datos y sean procesados por herramientas automáticas, así como por personas. Para adaptar la Web, los programas del mañana deben ser capaces de compartir y procesar datos incluso cuando estos programas se hayan diseñado de forma completamente independiente. La Web semántica es



una iniciativa del consorcio World Wide Web (W3C), diseñada para desempeñar un papel de liderazgo en la definición de la Web. Éste desarrolla especificaciones abiertas para aquellas tecnologías que están preparadas para distribuciones a gran escala, e identifica –mediante el desarrollo avanzado de código abierto– los componentes de la infraestructura que en el futuro se necesitarán adaptar en la Web.

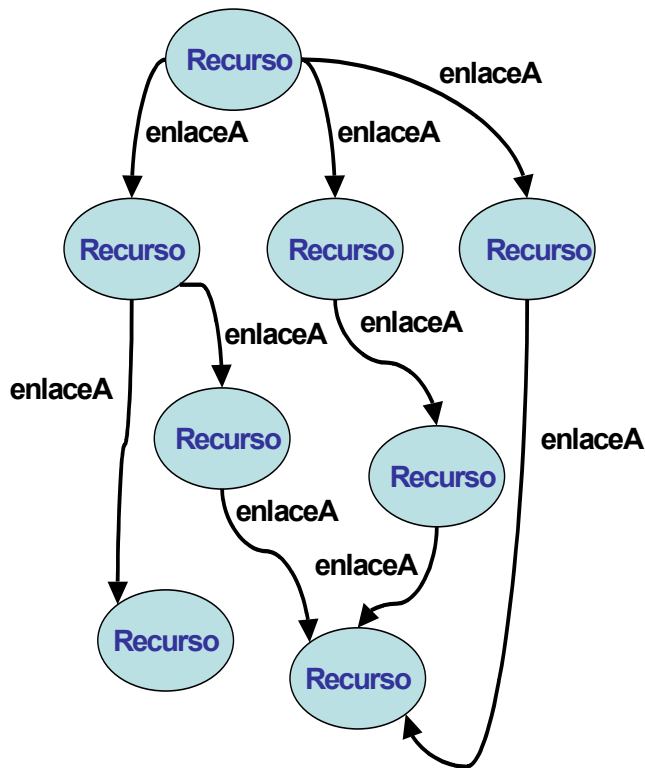
Berners-Lee plasmó su visión inicial de la Web semántica en la siguiente figura.



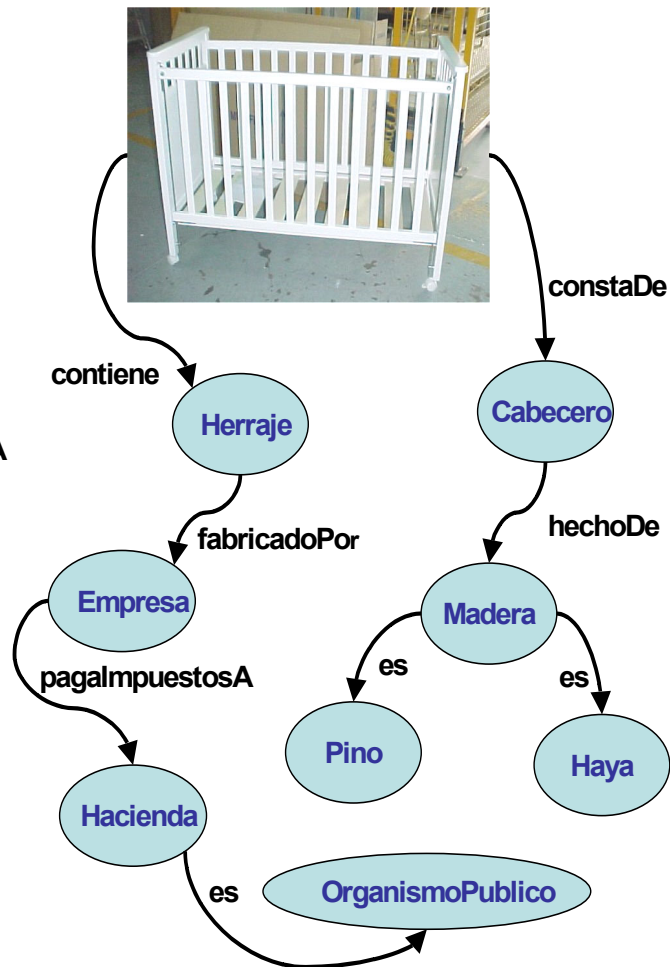
**Figura 11. La Web semántica tal como la concibió Tim Berners-Lee. Figura de Berners-Lee**



## Red de datos



## Red semántica



Miguel Ángel Abián, julio de 2005

Figura 12. Comparación entre una red de datos y una red semántica

La Web semántica se basará en que las máquinas comprendan el significado de la información disponible en ella; de ahí el adjetivo "semántica". Para los humanos, comprender un signo o una palabra no es nada extraordinario: lo hacemos gratuitamente a cada momento. Por sí solas, las palabras y los símbolos no son más que manchas negras sobre el monitor o el papel. Si esas manchas cobran sentido (es decir, tienen semántica) es porque nuestros cerebros se lo dan: al ver un símbolo, una palabra o una imagen, el cerebro lo asocia automáticamente con los conceptos que ha ido almacenando a lo largo de los años. Por decirlo de otra manera: la **interpretación semántica** la proporcionamos nosotros, mejor dicho, nuestras estructuras neuronales. Por ejemplo, la educación ha hecho que  $7 + 3$  tenga para todos la semántica (significado) de "la suma de dos números naturales: el 7 y el 3" o, dicho manera más formal, de "una operación algebraica definida en el cuerpo de los números naturales ( $\mathbb{N}$ ), donde  $7$  y  $4 \in \mathbb{N}$ ". En cierta medida, el cerebro es una máquina traductora que convierte símbolos en conceptos. ¿Por qué obra así? Seguramente, porque pasar del mundo simbólico al conceptual ha sido muy provechoso para la supervivencia de la especie.



Para las máquinas actuales, “comprender” no debe entenderse en el sentido de “comprensión humana”, sino en el de “inferir, deducir”. Que las máquinas comprendan o entiendan los datos significa que, mediante procesos lógico-matemáticos, sean capaces de inferir conclusiones a partir de los datos. Veamos un ejemplo: una máquina a la que se le programe la regla lógica “Toda persona es un ser vivo” puede comprender (deducir) que, si Luis es una persona, es un ser vivo (la información de que Luis es una persona debe ser suministrada por humanos).

Como vemos, hay poco de inteligencia en ese tipo de deducciones: la máquina nunca sabrá qué es una persona ni podrá deducir conclusiones que no se deriven de la aplicación de reglas lógicas (definidas por humanos) a los datos suministrados. Tim Berners-Lee no es muy optimista sobre la inteligencia de las máquinas:

El concepto de documentos que las máquinas puedan entender no implica ninguna inteligencia artificial que permita a la máquina comprender los murmullos humanos. Sólo indica la habilidad de la máquina para resolver un problema bien definido desarrollando operaciones bien definidas sobre datos bien definidos. En lugar de pedir a las máquinas que entiendan el lenguaje humano, implica pedir a la gente que haga un esfuerzo.

**[*What the Semantic Web can represent*,  
<http://www.w3.org/DesignIssues/RDFnot.html>, septiembre de 1998]**

La expresión “pedir a la gente que haga un esfuerzo” significa que los humanos deben representar los datos en algún lenguaje formal (lógico y axiomático), de manera que las máquinas puedan usarlos para extraer inferencias lógicas. Estos lenguajes, vinculados a la **representación del conocimiento**, se conocen como lenguajes de representación del conocimiento.



## EJEMPLO DE BÚSQUEDA DE DOCUMENTOS EN LA WEB SEMÁNTICA (1)

**Regla lógica:** Si la página X se relaciona con la página Y y la página Y se relaciona con la página Z, entonces las páginas X y Z están relacionadas

Se buscan todos los documentos relacionados con el libro “Tristana”, de Pérez Galdós

**Inferencia lógica:** Si la página X aparece relacionada con “Tristana”, la página Y con la página X y la página Y con la Z, entonces Z está relacionada con “Tristana”

Por lo tanto, en la búsqueda aparecerán documentos que en la Web no estaban referenciados directamente con el asunto de la búsqueda.

Miguel Ángel Abián, julio de 2005

**Figura 13. Ejemplo de una búsqueda en la Web semántica. Por un lado, las páginas deberían almacenar información sobre las páginas con las que se relacionan; por otro lado, el programa de búsqueda debería usar la regla lógica para efectuar inferencias lógicas a partir de la información que encuentre en las páginas. Nótese que en el proceso no hay nada que se parezca a la comprensión humana: sólo se aplican literalmente unas reglas**



## EJEMPLO DE BÚSQUEDA DE DOCUMENTOS EN LA WEB SEMÁNTICA (2)

**Regla lógica:** Si una tienda A ha vendido más de 1000 unidades del producto X, entonces pertenece a la lista de “Mejores tiendas del año”

Se buscan todas las tiendas que figuran en la lista de “Mejores tiendas del año”

**Inferencia lógica:** Si en la página web de la tienda A aparecen vendidas más de 1000 unidades del producto X, entonces la tienda A pertenece a la lista de “Mejores Tiendas del año”.

Por lo tanto, en la búsqueda aparecerán documentos en los que no figura explícitamente el término “Mejores Tiendas de año”

Miguel Ángel Abián, julio de 2005

**Figura 14. Ejemplo de una búsqueda en la Web semántica. El buscador inferirá deducciones a partir de los datos expresados en las páginas.**

Salvo que la investigación en inteligencia artificial (IA) experimente un espectacular avance en los últimos años y podamos tener algún HAL 9000 sin tendencias asesinas, la Web semántica sólo se podrá construir incorporando a los documentos información sobre ellos (metadatos), de modo que las máquinas puedan procesarla mediante la aplicación de reglas lógicas.

La historia de la Lógica (disciplina que estudia los principios formales del razonamiento humano) comienza con Aristóteles. La Lógica es relevante para la Web semántica por tres motivos: 1) permite desarrollar lenguajes formales que permiten representar el conocimiento; 2) proporciona semánticas bien definidas: en un sistema lógico, cada símbolo y cada expresión tiene un significado único e inambiguo; 3) proporciona reglas de inferencia, a partir de las cuales se pueden extraer consecuencias del conocimiento (estas reglas permiten validar demostraciones, es decir, comprobar si una consecuencia se deriva de unas premisas). Lo que se conoce como **interpretación semántica automática** de documentos no pasa de ser la aplicación de reglas lógicas a unos datos presentados en algún lenguaje formal (como OWL, DAML+OIL o KIF/CL).



Estos lenguajes se basan en la lógica descriptiva (lógica que proporciona un formalismo para representar y expresar el conocimiento humano, basándose en métodos de razonamiento bien establecidos y procedentes de un subconjunto de la lógica de predicados o de primer orden). Por ejemplo, la frase “El único matemático de la Universidad es profesor del hijo de Ferreira”, se expresa en lógica de primer orden como

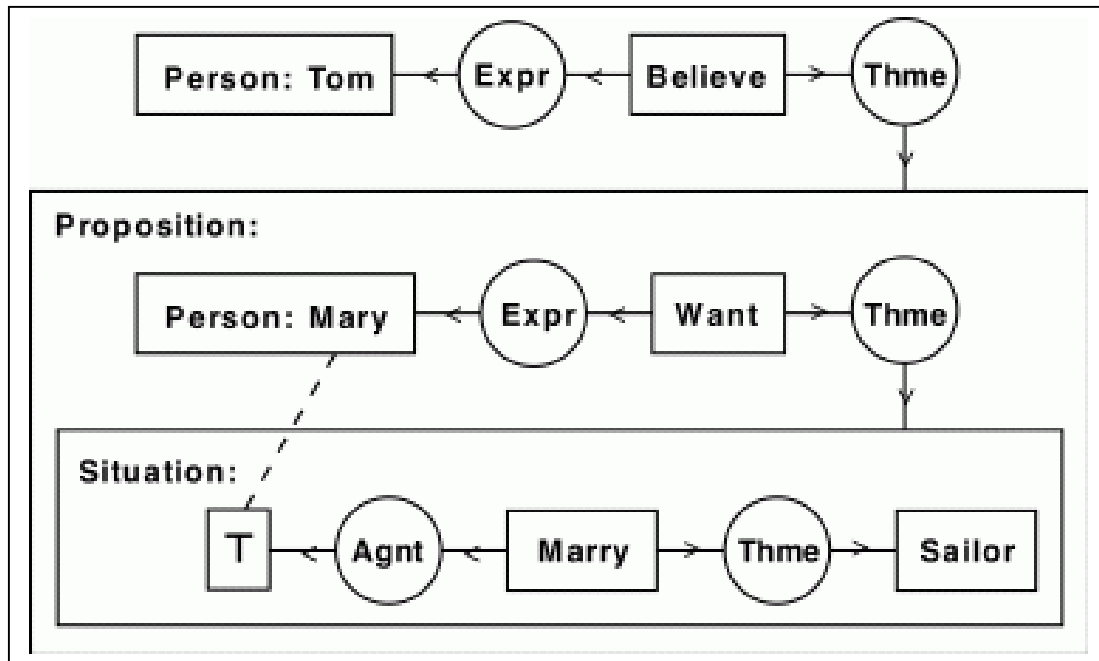
$$\exists x. [M(x) \wedge (\forall y. [M(y) \rightarrow x = y]) \wedge \exists z. [P(x, z) \wedge H(z, Ferreira) \wedge (\exists u. [H(u, Ferreira) \wedge z \neq u]) \wedge \forall t. [P(x, t) \rightarrow t = z]]]$$

Donde  $M(x)$  significa “x es matemático”;  $P(x, y)$  denota “x es profesor de y”;  $H(x, y)$  significa “x es hijo de y”.  $M(x)$ ,  $P(x, y)$  y  $H(x, y)$  son funciones proposicionales.

Espero que no se cumpla aquí la consabida regla editorial de que cada fórmula en un libro de divulgación reduce el número inicial de lectores a la mitad. El anterior ejemplo busca tan sólo resaltar que los lenguajes en que se expresarán los metadatos en la Web semántica se basan en la lógica, de modo que son inambiguos y no admiten contradicciones. Los lenguajes naturales están muy bien para seres que se consideran inteligentes, aunque sus acciones apunten hacia la conclusión contraria, pero no para las máquinas. Frases como “El premio Nóbel Nash habló de su travesía por la locura en el congreso” (¿tan mal organizado estaba el congreso?: tendré que revisar mi agenda), “Luis trabaja en un pueblo abandonado por todos” (¿quién está abandonado: Luis o el pueblo?) o “He visto al hijo de Juan, que ha ganado mil millones en la lotería (¿quién tachará de su vocabulario la palabra “trabajar”: Juan o su hijo?) ejemplifican que los lenguajes naturales no resultan los más apropiados para expresar conceptos inambiguos.

Con la lógica que hay tras los lenguajes de la Web semántica, las máquinas podrán realizar inferencias como hacen los humanos –especialmente, los matemáticos y los filósofos– y justificarlas. Por ejemplo, un agente inteligente que tenga programada la regla “Las compras de más de 300 € tienen un descuento del 10%” podrá justificar lógicamente por qué permite descuentos a unos clientes y a otros no. Del mismo modo, un agente que reclame a una persona 12 € por una entrada de cine, será capaz de justificar formalmente por qué lo hace, basándose en el número de la tarjeta de crédito y en el código de la entrada (es de esperar que los morosos usen métodos menos formales para escapar de sus obligaciones). Los usuarios de la Web semántica no tendrán que preocuparse de conocer la lógica formal de los lenguajes: solamente deberán modelar con ellos las áreas de conocimiento de interés; luego, los lenguajes deducirán de los datos introducidos las inferencias correspondientes.





**Figura 15.** Representación de la frase “Tom believes that Mary wants to marry a sailor” en el lenguaje lógico CG (Conceptual Graphs). Lenguajes formales como éste permiten expresar los datos de manera que las máquinas puedan “interpretarlos”. Figura de Tim Berners-Lee

```
<<daml:Class rdf:ID="Persona">
  <rdfs:subClassOf rdf:resource="#SerVivo"/>
  <rdfs:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#tieneProgenitor"/>
      <daml:toClass rdf:resource="#Persona"/>
    </daml:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction daml:cardinality="1">
      <daml:onProperty rdf:resource="#tieneMadre"/>
    </daml:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction daml:cardinality="1">
      <daml:onProperty rdf:resource="#tienePadre"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

**Figura 16.** Representación en el lenguaje formal DAML de las relaciones entre una persona y sus padres. Una persona es un ser vivo con dos características: “tieneMadre” y “tienePadre”, que son especializaciones (subclases) de la característica “tieneProgenitor”



**Nota para los lectores con inclinaciones lógicas:** La lógica de primer orden (o lógica de predicados) permite establecer formalmente sentencias (predicados) sobre cosas y colecciones de cosas, sus tipos y propiedades, así como clasificarlas y describirlas. En la lógica de primer orden, cada sentencia se divide en un sujeto y un predicado; éste modifica o define las propiedades del sujeto. En esta lógica, los predicados siempre se refieren a individuos u objetos y los cuantificadores (“todo”, “algún”) sólo se permiten para individuos. Las sentencias se expresan en la lógica de primer orden en la forma  $R(x)$ , donde  $R$  es el predicado y  $x$  es el sujeto, formulado como una variable.  $R(x)$  es una función proposicional; estas funciones se convierten en proposiciones de sujeto-predicado cuando se asignan valores a sus variables. Así, la función proposicional  $G(x)$  (“ $x$  es un gato”) origina una proposición de sujeto-predicado cuando se asigna un valor a  $x$ :  $G(\text{Silvestre})$  (“Silvestre es un gato”).

Por ejemplo, “Todos las personas son mortales” (bueno, la he actualizado un poco) se expresa así en la lógica de primer orden:

$$\forall x : P(x) \rightarrow M(x)$$

$P$  es el predicado “es una persona”,  $M$  es el predicado “es mortal”. Por la regla del *modus ponens*, Sócrates, por ser persona, es mortal (vamos, que el filósofo no se escapa de la guadaña):

$$P(\text{Sócrates}) \rightarrow M(\text{Sócrates})$$

La lógica de segundo orden se forma como extensión de la de primer orden y, a diferencia de ésta, permite cuantificar propiedades. En la lógica de primer orden no pueden expresarse sentencias como “existe una propiedad...” o “para toda propiedad...”. Una frase como “Si Juan es una persona y María es una persona, hay una propiedad que Juan y Rosa comparten” no puede expresarse con la LPO; en cambio, en LSO se expresa fácilmente:

$$\text{Persona}(\text{Juan}) \wedge \text{Persona}(\text{Rosa}) \rightarrow \exists P (P(\text{Juan}) \wedge P(\text{Rosa}))$$

La LSO no es tan “perfecta” como la LPO y no admite una teoría formal de demostraciones como la de la LPO, que cuenta con algoritmos de demostración perfectamente definidos. Algunas sentencias de la LSO originan situaciones paradójicas. Por ejemplo, la paradoja de Epiménides el Cretense, quien dijo “Todos los cretenses mienten”, plantea una contradicción (esta sentencia no puede expresarse mediante la lógica de primer orden). Si Epiménides mentía, era mentira que mintiera y, por tanto, decía la verdad; si decía la verdad, mentía, porque precisamente eso era lo que decía que estaba haciendo. La actitud de los pensadores hacia este tipo de paradojas varía mucho: San Pablo pensaba que, más que una paradoja lógica, era una prueba irrefutable de la perversidad de los paganos; en cambio, los filósofos hegelianos no veían nada raro en la paradoja, pues veían contradicciones en todas partes (salvo en el Absoluto), incluso en  $1 + 1 = 2$ . Corolario: los filósofos no hegelianos y las máquinas aman la LPO, pues no da lugar a enunciados contradictorios.

Ciertos subconjuntos de la LPO, especializados en clasificar cosas, se denominan lógicas descriptivas (cada familia de lógicas descriptivas se origina de un determinado subconjunto de la LPO). Si bien las lógicas descriptivas carecen de la potencia expresiva de la LPO –es decir, no pueden representar tantas cosas como ésta–, son matemáticamente trazables y son computables.



La Web semántica permitirá a los usuarios buscar información de una manera imposible hoy. Por ejemplo, no sólo permitirá saber qué páginas web almacenan *Manifeste du Surréalisme*, publicado por André Breton en 1924, sino que también informará al usuario de que Breton escribió también un segundo manifiesto llamado *Second Manifeste du Surréalisme*, en que abordaba cuestiones tratadas en el primero y cuyo tema principal era el papel de los sueños en la vida, y proporcionará –si así se desea– una lista de los textos en los que Breton abordó ese tema, así como de libros y artículos relacionados con Breton, el surrealismo y las vanguardias. Asimismo, el usuario podrá encontrar, mediante una sola consulta, una lista de todos los escritores cuyas obras han sido influidas por el surrealismo (Fernando Arrabal, por ejemplo) y enlaces a estudios críticos sobre esa influencia. Como los críticos literarios se caracterizan por una tendencia gregaria a no ponerse de acuerdo, el usuario del futuro podrá disfrutar de la búsqueda muchísimo más que el usuario actual.

Los buscadores del futuro permitirán consultas como “Busco todos los mecánicos que tengan su taller a menos de 1 km de la calle Manuel Candela y que trabajen para la compañía de seguros Siempre Seguro. Muéstrame primero los más baratos”. Mediante la información codificada en la página web de cada mecánico (mediante metadatos como *Trabaja\_Para\_Compañía* y *Tarifa\_Por\_Hora*), el agente podrá encontrar a esos mecánicos en un santiamén.

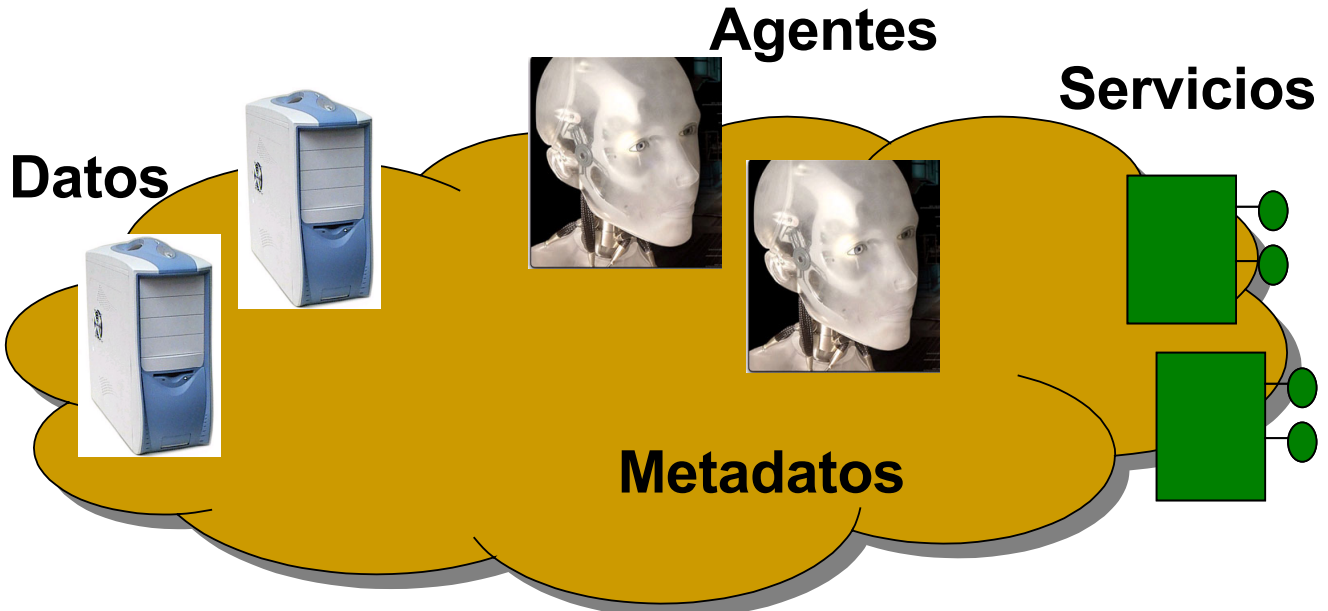
La Web semántica también permitirá consultas basadas en otras consultas. Por ejemplo, imagínese un agente inteligente que busque información sobre pintores. Cuando se encuentre con sitios web que proporcionan interfaces de búsqueda para bases de datos sobre pinturas y pintores (la interfaz típica sería un formulario donde se pueda escribir nombres de pintores, cuadros o movimientos pictóricos), el agente introducirá en esas interfaces los nombres de los pintores sobre los cuales busca información y buscará luego en los resultados devueltos por las consultas. La Web semántica proporcionará maneras de buscar datos que forman parte de lo que hoy se conoce como la “Internet oculta”.

Asimismo, la Web semántica también permitirá el uso de agentes personales encargados de extraer información de múltiples fuentes, cada una con su propia manera de representar los datos, y de informar a los usuarios de los sucesos que son de interés (conciertos, charlas, estrenos de películas, etc.). Por ejemplo, un agente personal al que se le hayan proporcionado las preferencias musicales del usuario Miguel Ángel Abián le avisará cada vez que Lou Reed saque un nuevo disco (sobre todo si contiene material nuevo, que ya va siendo hora) o que vaya de gira por Europa (sobre todo si la gira pasa por España o Francia). Para lograrlo, el agente trabajará con fuentes muy distintas: periódicos virtuales, páginas de agencias de noticias internacionales, páginas de venta de entradas por Internet...

Por último, los buscadores web del mañana no sólo encontrarán las páginas donde aparezcan los términos de la búsqueda, sino también todas aquellas páginas donde haya sinónimos de esas palabras. Así, una búsqueda basada en el término “B2C” encontrará también páginas donde aparezcan términos como “business-to-consumer”, “business-2-consumer” o “B-to-C”.



# LA WEB DEL FUTURO



Miguel Ángel Abián, julio de 2005

**Figura 17. En el futuro, la Web estará repleta de metadatos legibles para las máquinas. Los agentes de la Web semántica usarán esos metadatos para concertar citas, realizar transacciones monetarias y de bienes, comparar precios, obtener información, reservar entradas...**

El lector habrá notado que casi siempre hablo de la Web semántica usando el futuro simple: esta Web no existe aún, pero ya tenemos las tecnologías que nos llevarán hasta ella. En la figura 18 se detallan las más importantes.



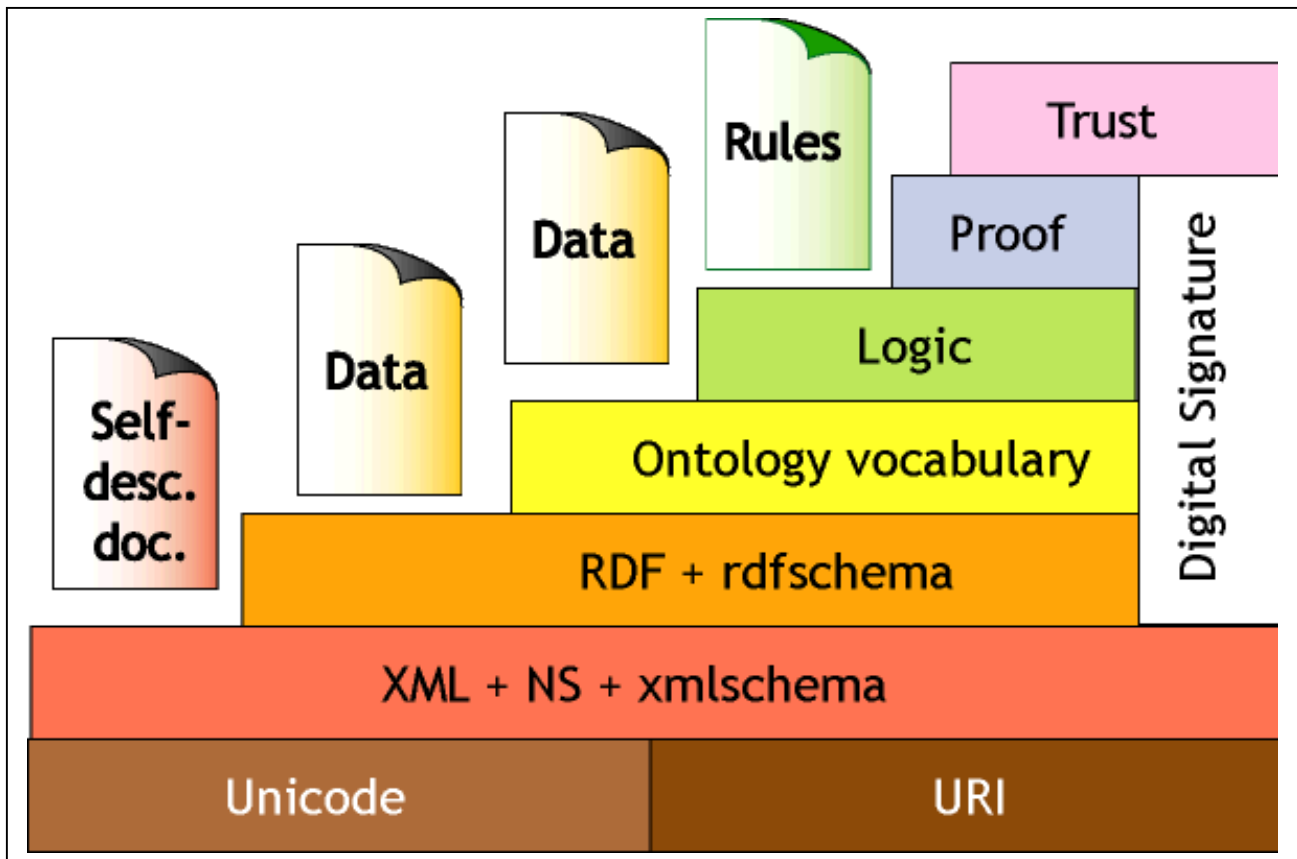


Figura 18. Figura extraída de la documentación del W3C sobre la Web semántica

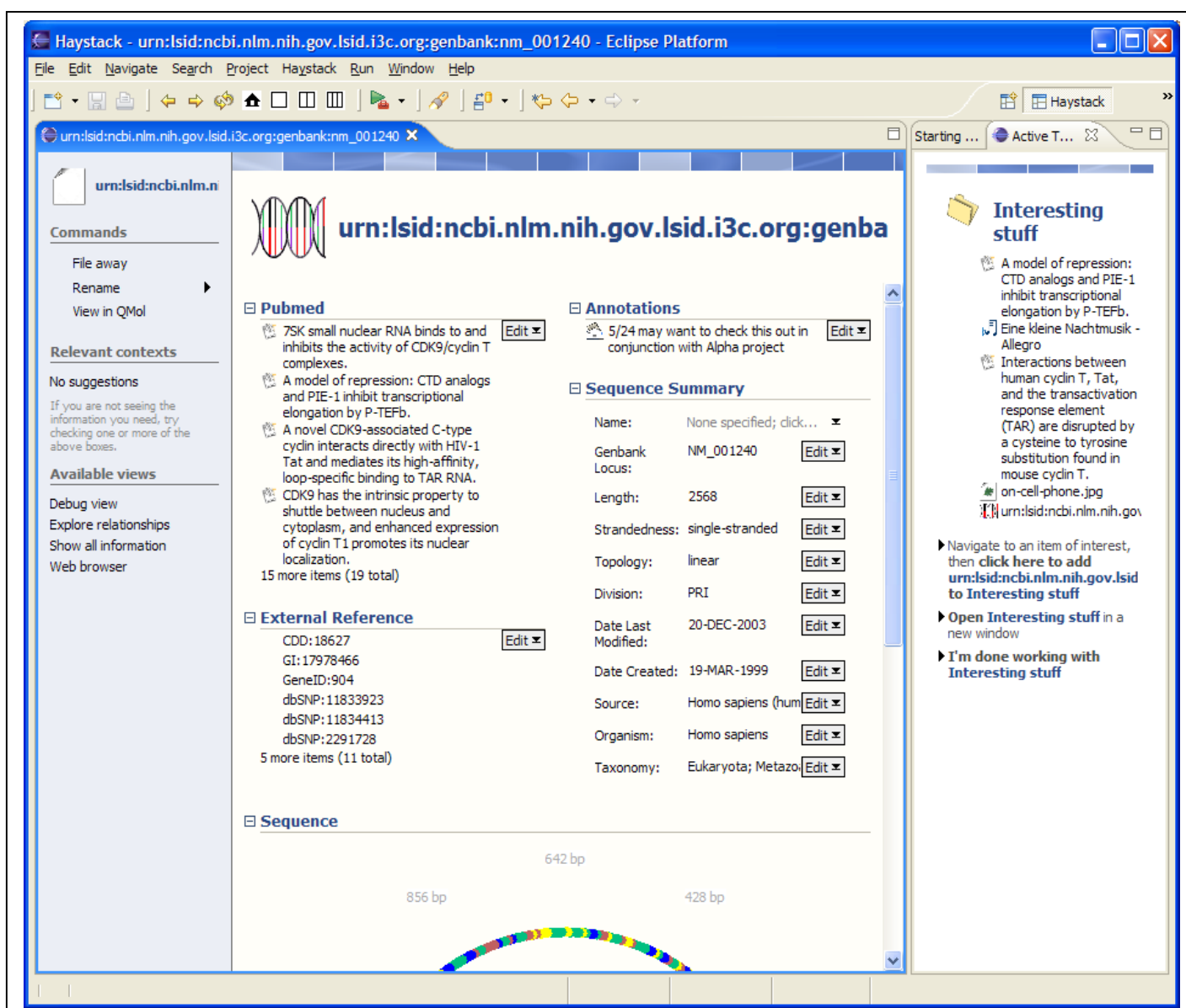
Desglosemos un poco los principales componentes de la figura:

- a) **XML (*Extensible Markup Language*)**. Es el lenguaje que actualmente se usa para intercambiar datos en la red.
- b) **RDF (*Resource Description Framework*)**. Es un lenguaje que describe metadatos y fuentes de información (documentos, imágenes, etc.).
- c) **Ontologías**. Son conceptualizaciones que definen el significado de un conjunto de conceptos para un determinado dominio. Las ontologías se expresan en lenguajes formales como OWL.
- d) **Lógica**. El razonamiento lógico permite determinar si los datos son correctos e inferir conclusiones a partir de ellos.
- e) **Prueba**. Las pruebas explican y verifican los pasos de los razonamientos lógicos.
- f) **Confianza**. Se refiere a técnicas que aseguran la identidad y fiabilidad de los datos y servicios.

En los próximos apartados estudiaremos algunas de estas tecnologías. Con ellas, la Web pasará de ser una colección de documentos a ser una **base de conocimiento**, de la cual se podrán extraer conclusiones a partir de premisas.



Al lector interesado en ver cómo funcionan a pequeña escala las tecnologías de la Web semántica, le recomiendo el **proyecto Haystack** (<http://haystack.lcs.mit.edu/>). Este proyecto desarrolla aplicaciones que usan las tecnologías de la Web semántica para personalizar la navegación, de acuerdo con los gustos del usuario. Especialmente interesantes son el “navegador semántico” y la extensión (*plug-in*) *Piggy-Bank* para el navegador *Firefox*. El primero está basado en **Eclipse** e incluye muchas funciones interesantes: navegar por páginas web ordinarias y por páginas que usan tecnologías de la Web semántica, organizar fotografías digitales y archivos de música, recopilar información en minoportales...



**Figura 19. Captura de pantalla del navegador semántico del proyecto Haystack.**  
El navegador se puede descargar de la dirección  
<http://haystack.lcs.mit.edu/staging/eclipse-download.html>

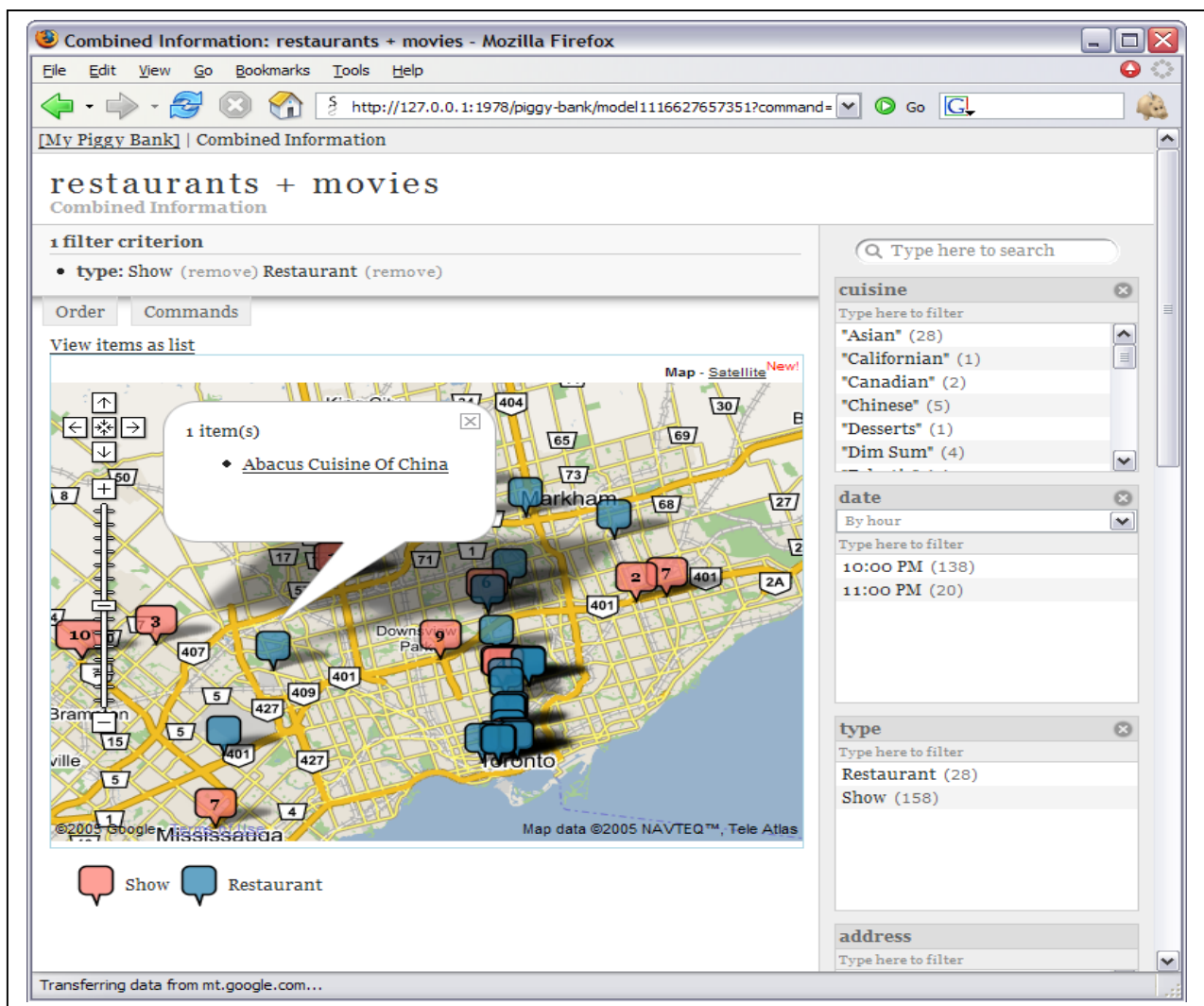


**Nota informativa:** Si no conoce el maravilloso entorno integrado de desarrollo que es Eclipse o si lo conoce y quiere ampliar sus conocimientos, puede consultar los siguientes artículos de javaHispano:

- 1) <http://www.javahispano.org/articles.article.action?id=75>
- 2) <http://www.javahispano.org/articles.article.action?id=81>
- 3) <http://www.javahispano.org/articles.article.action?id=97>

Para conocer la impactante interfaz gráfica que Eclipse ofrece gratuitamente a los programadores, le recomiendo **JLibrary** (<http://jlibrary.javahispano.net/es/index.html>), obra de Martín Pérez, miembro de javaHispano y autor de recomendables artículos y tutoriales sobre patrones e interfaces gráficas.

La extensión *Piggy-Bank* (<http://simile.mit.edu/piggy-bank/>) para el navegador web Firefox permite recoger metadatos semánticos de las páginas web e incluso crearlos a partir del código HTML. Con esta información, los usuarios pueden combinar informaciones de múltiples fuentes, almacenar la información de interés en “bancos semánticos” y navegar de forma eficaz por sitios web, según sus preferencias.

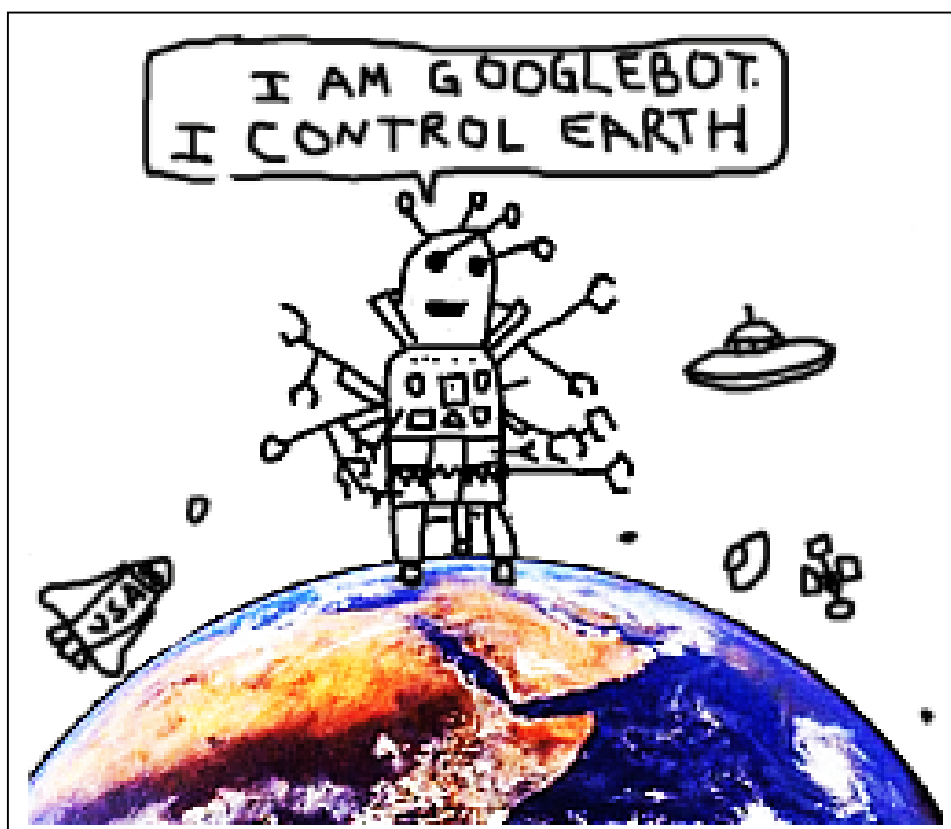


**Figura 20.** Captura de pantalla de la extensión Piggy-Bank para Firefox. Extraída de <http://simile.mit.edu/piggy-bank/>



Tímidamente todavía, algunas empresas están incorporando metadatos a sus productos, con vistas a que no se queden rezagados respecto a la futura Web semántica. Por ejemplo, la compañía Adobe trabaja en su XMP (*eXtensible Metadata Platform*, plataforma extensible de metadatos), una tecnología que permite incluir directamente metadatos en aplicaciones, archivos, bases de datos, incluso en datos binarios. Independientemente del formato de los datos, XMP permite introducir “paquetes XMP” en ellos. Se puede encontrar más información sobre esta tecnología en <http://www.adobe.com/products/xmp/main.html>. La inclusión de metadatos en los archivos permitirá que las aplicaciones de la Web semántica puedan conocer su contenido sin necesidad de abrirlos y leerlos. Por el momento, XMP se encuentra disponible para Photoshop 7.0, Acrobat 5.0, FrameMaker 7.0, GoLive 6.0, InCopy 2.0, InDesign 2.0, Illustrator 10 y LiveMotion 2.0.

En [http://www.ftrain.com/google\\_takes\\_all.html](http://www.ftrain.com/google_takes_all.html) se encuentra el interesante trabajo de ficción *August 2009: How Google beat Amazon and Ebay to the Semantic Web*, escrito por Paul Ford, quien describe lo que podría suceder si Google adoptara técnicas de la Web semántica. El artículo se publicó en 2002. Curiosamente, menciona de pasada que Apple compró España y la renombró como *Different-thinking Capitalist Republic of Information* (¿?). Si es un chiste se me escapa la gracia, la verdad.



**Figura 21. El futuro según Paul Ford. Imagen extraída del trabajo de ficción *August 2009: How Google beat Amazon and Ebay to the Semantic Web* ([http://www.ftrain.com/google\\_takes\\_all.html](http://www.ftrain.com/google_takes_all.html))**



Por el momento, las tecnologías de la Web semántica no se han probado a gran escala, si bien están despertando cierto interés en el ámbito militar y en el de los servicios de inteligencia y espionaje. Baste una sola muestra: como consecuencia de la guerra contra el terrorismo en la que se haya enfrascado el reelecto presidente George Walker Bush, el **programa VKB** (*Virtual Knowledge Base*) del Departamento de Defensa de los Estados Unidos ha recibido un gran impulso. Sus fines son detectar relaciones entre todos los datos que circulan por la Web, mediante **servicios web** (para permitir la interoperabilidad de los datos) y ontologías. A toro pasado, no cabe duda de que habría sido muy útil una Web semántica que relacionara la información que la CIA tenía sobre los terroristas del 11-S y la que tenían los escuelas de pilotos y los organismos oficiales encargados de expedir licencias de vuelo. Según los servicios de inteligencia estadounidenses, toda esa información estaba ahí, pero dispersa y en formatos no interoperables.

(Las lágrimas de cocodrilo que derramaron algunos responsables del FBI y de la CIA –Robert Mueller, por ejemplo– por no haber tenido algo similar a la Web semántica antes del once de septiembre de 2001 se podrían haber evitado si hubieran prestado más atención a la información que tenían. Según el informe oficial sobre el 11-S, redactado por una comisión especial del Senado y de la Cámara de Representantes estadounidenses, la CIA no tuvo en cuenta las advertencias de las otras agencias de información y de seguridad; en particular, el informe prueba que la Administración de Bush recibió, en el verano de 2001, avisos de que los terroristas de Bin Laden preparaban “un ataque espectacular contra Estados Unidos” y que estaban entrenándose para secuestrar aviones de línea. Así las cosas, una web semántica hubiera determinado que ni George Tenet ni Robert Mueller –directores en 2001 de la CIA y del FBI, respectivamente– merecían estar en sus puestos. Curiosamente, nadie habló entonces de dimisiones ni de destituciones. A la vista de la forma como se trató al ex presidente Bill Clinton por mentir sobre su relación con una oronda becaria –estuvo muy cerca de ser destituido–, cualquier observador imparcial concluiría que es muchísimo más peligroso mantener relaciones extramatrimoniales que descuidar gravemente la seguridad nacional. En fin, de muy poco sirve la interoperabilidad de los datos si la gente que toma decisiones mira hacia otro lado.)

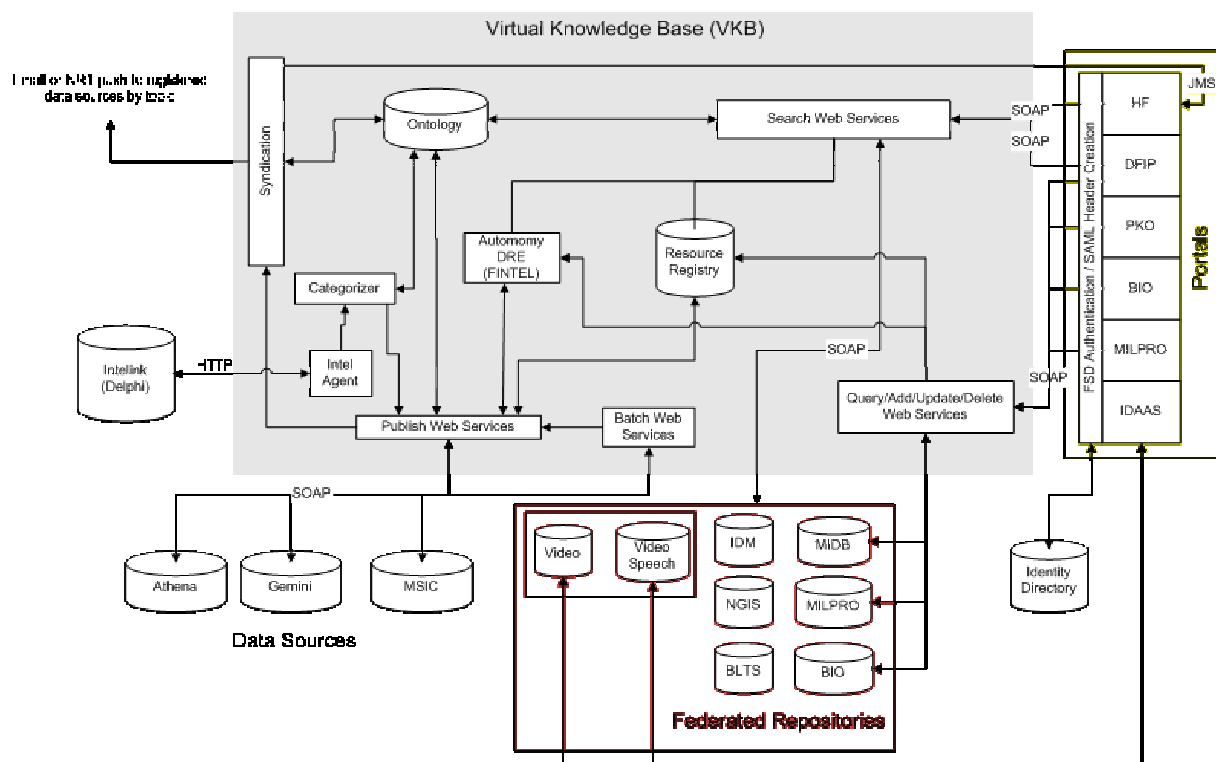
¿Cómo acabará el programa VKB? Tecnológicamente, no lo sé: es muy ambicioso. Económicamente, a la vista de lo que sucedió con muchos productos de alta tecnología del siglo pasado (*chips*, ordenadores, el proyecto de la *guerra de las galaxias* [que nombre tan bien elegido..., tan comercial, ¿verdad?], Internet), supongo que unas pocas empresas privadas sacarán tajada del dinero público y podrán comercializar, de aquí a unos cuantos años, productos basados en todo lo que los contribuyentes estadounidenses han costado. Como es habitual, el Gobierno estadounidense elige a las empresas ganadoras, y ya se sabe que *the winner takes all*.

Según James Woolsey (ex director de la CIA), “esta cuarta guerra mundial [la guerra contra el terrorismo] durará para nosotros considerablemente más que la Primera o la Segunda. Esperemos que no más que las cuatro décadas completas que duró la Guerra Fría” (frase pronunciada en una conferencia en la Universidad de California, el 3 de abril de 2003). Parece, pues, que unas cuantas privilegiadas empresas estadounidenses van a tener fondos de sobra durante unos cuantos años. Asegurada su supremacía tecnológica mediante la inversión pública en la empresa privada, Estados Unidos –mejor dicho, su gobierno– se permitirá sermonear al mundo sobre las ventajas y virtudes de la liberalización de los mercados y de la no intervención del Estado en asuntos empresariales (salvo cuando se trata de rescatar a grandes bancos que han quebrado por inversiones especulativas o mala gestión, claro está, pues *alguien* debe



pagar los platos rotos), e incluso seguirá acusando a la Unión Europea de ser “un club de subvenciones” y continuará alegrándose por el progresivo deterioro de modelos sociales como el sueco. Tal comportamiento resulta *extremadamente razonable*: los titiriteros no quieren compartir los hilos que controlan a los muñecos; sus recetas para lograr el éxito son suyas y de nadie más.

Alguien debería recordar lo que era IBM antes de conseguir un jugoso contrato del Gobierno estadounidense para desarrollar *chips* y ordenadores: una poco prometedora empresa que fabricaba máquinas de escribir y tarjetas perforadas. ¿Alguien la imagina cotizando en la bolsa si no hubiera llegado ese providencial contrato? Sin él, hoy sólo sería recordada como la empresa cuyas tarjetas perforadas del censo fueron utilizadas por los nazis para localizar a los judíos y aniquilarlos de forma eficaz y despiadada, y cuyo fundador –Thomas J. Watson– recibió una medalla de manos de Hitler en 1937. ¿Cuántos pueden recordar a los competidores de IBM (en el mercado de las máquinas de escribir o de las tarjetas perforadas) que no obtuvieron contratos del Gobierno? Seguramente, tantos como los que creen de corazón en el libre mercado y lo practican en ambas direcciones, no sólo en una...



**Figura 22. Arquitectura del proyecto *Virtual Knowledge Base* (VKB) del Departamento de Defensa de los Estados Unidos. Como es de suponer, no hay disponible mucha información sobre él**



**Nota para los lectores con inclinaciones económicas o bursátiles:**

Enron es mi ejemplo favorito de lo que sucede cuando se deja que el libre mercado campe a sus anchas, y éste se encuentra con las nuevas tecnologías (la Web) y los viejos vicios. Esta compañía de distribución de gas y electricidad, que había comenzado como una pequeña empresa de gas en Tejas, basó su éxito –y su pronto y estrepitoso fracaso– en la estrategia de libre mercado aplicada a la distribución de gas natural y electricidad en Estados Unidos. En su momento, Enron se citaba como ejemplo de que, gracias a las bondades del libre mercado –en este caso, la desregulación del mercado estadounidense de gas y electricidad–, las empresas estadounidenses serían más competitivas, productivas y atractivas para el consumidor. En otras palabras: las empresas eran buenas por naturaleza, sólo había que dejarlas trabajar sin ahogarlas con normas medioambientales y laborales, regulaciones y cosas similares... Los hechos demostraron más bien lo contrario. Si bien la cantinela del libre mercado decía lo habitual (eliminar regulaciones y normas conduce a rentabilidades mayores y a la competencia entre empresas, lo cual beneficia a los consumidores), la realidad fue muy distinta: si el precio de la electricidad era de unos 30 dólares por megavatio/hora entre abril de 1998 y junio de 2000, en el primer semestre de 2001 se había multiplicado por 8. Además, había constantes interrupciones del suministro, es decir, apagones; y la compañía eléctrica más importante de California quebró. Visto lo visto, en junio de 2001 volvió la regulación, y la Comisión Federal Reguladora de Energía impuso unos precios máximos por el megavatio/hora (como sucede en Francia y España).

¿Estaría equivocado Ronald Reagan cuando dijo “El gobierno no es la solución a nuestros problemas, es el problema”? ¿Por qué funcionó tan mal el libre mercado? Pues porque los beneficios se antepusieron a los consumidores: Enron se dedicó a especular con contratos a largo plazo (prohibidos antes de la desregulación), de modo que casi todo el suministro a largo plazo de electricidad y gas quedaba comprado o vendido. En consecuencia, el suministro al contado (para *hoy*) era muy escaso, y se pedían por él precios cada vez más altos. Curiosamente, cuanto más escasez había de electricidad, más “averías” ocurrían en las centrales hidroeléctricas, que obtenían grandes beneficios cuando se disparaba el precio al contado del megavatio/hora. Curiosamente también, Enron enviaba electricidad fuera de California para aumentar la escasez de electricidad al contado en ese estado y subir más los precios. Mientras tanto, la cotización bursátil de Enron subía como la espuma. Una empresa que compraba y vendía electricidad y gas por Internet, ¿cómo no iba a subir en plena fiebre de las *puntocom*, cuando subían hasta las sillas de la Bolsa?

¿Cómo acabó este experimento de libre mercado? Enron quebró (los 1000 millones de dólares para las pensiones de los empleados se esfumaron), los consumidores –particulares y empresas– pagaron precios exorbitantes por el gas y la electricidad, y el Gobierno de los Estados Unidos gastó más de 40.000 millones de dólares en restablecer las condiciones de suministro propias de un país desarrollado. Al final, cuando los especuladores ya habían expoliado a los clientes, papá Estado –ese Estado al que tan poco aprecian algunos partidarios del libre mercado, salvo cuando la codicia los pone en apuros– pagó el pato. ¿Final feliz?

(Cuando Enron se derrumbaba, se solicitó al Gobierno que redujera la deuda de la empresa. Curioso: ¿no debía el Estado ocuparse de sus propios asuntos y dejar al mercado en paz? ¿Por qué debía salvar a las empresas que se arruinaban por sus fraudes y su avaricia?)



Las fraudulentas prácticas contables de Enron dan para un manual de contabilidad moderna. Me limitaré a señalar aquí una muy llamativa (la difunta Arthur Andersen no pensaba lo mismo): Enron contabilizaba las ventas de gas y electricidad que se iban a entregar al año siguiente como ingresos actuales, sin contabilizar como gasto el dinero necesario para comprar el gas o la electricidad. Es decir, ¡se obtenían beneficios sin gasto alguno! Enron fue durante un tiempo el equivalente financiero de la máquina del movimiento perpetuo.

Claro está, en la historia de Enron hubo algunos hechos que fueron bastante miserables. Que altos directivos (entre ellos el presidente de la empresa, Ken Lay) vendieran acciones de Enron por valor de cientos de millones de dólares mientras recomendaban a sus empleados que mantuvieran las suyas demostró dos cosas: a) la ética de los dirigentes de la empresa se había desplomado a niveles sorprendentes; b) era el momento de abandonar el barco.

No le quepa ninguna duda de que la Web semántica tendrá su Enron. Vigile su cartera.



## 5. XML: un primer paso hacia la Web semántica

Hoy día se acepta que la Web semántica se construirá basándose en XML (*Extensible Markup Language*, lenguaje extensible de etiquetado) y en las tecnologías asociadas. La ubicación exacta del lugar que ocupará XML se muestra en la figura 18. XML aparece sobre dos rectángulos marcados como Unicode y URI porque se basa en estas dos tecnologías. Unicode es un estándar cuya meta es asignar enteros no negativos a los caracteres escritos de los idiomas más extendidos y proponer distintas maneras de codificar binariamente esos números (las codificaciones más comunes son UTF-8, UTF-16 y UTF-32), amén de definir algoritmos para asegurar la interoperabilidad cuando se procesan textos que usan Unicode. Como XML se basa en Unicode, los documentos XML pueden usar caracteres de casi todas las lenguas que se hablan en la Tierra (la versión 3.1 de Unicode incluía más de 90.000 caracteres). Si desea más información sobre Unicode, la encontrará en la primera parte del tutorial *Java y las redes* (<http://www.javahispano.org/tutorials.item.action?id=45>). XML usa URIs (*Uniform Resource Identifiers*, identificadores uniformes de recursos) para especificar de forma única los recursos de la Web semántica, del mismo que la Web actual usa URLs (*Uniform Resource Locators*, localizadores uniformes de recursos).

Mientras que HTML es un lenguaje de marcado para documentos de hipertexto, XML es un lenguaje de marcado para documentos de todas clases. Se dice que XML es extensible porque permite al programador asociar sus propias etiquetas (metadatos) a los datos. El aspecto de un documento XML es similar al mostrado en la figura 23.

```
<empresa>
  <nombreEmpresa>
    AIDIMA
  </nombreEmpresa>
  <direccion>
    <calle>
      Benjamín Franklin
    </calle>
    <numero>
      13
    </numero>
    <poblacion>
      Paterna
    </poblacion>
    <codigoPostal>
      46017
    </codigoPostal>
    <pais>
      España
    </pais>
  </direccion>
  <telefono>
    +(34) 961265070
  </telefono>
</empresa>
```

Figura 23. Ejemplo de un documento XML



Como vemos, un documento XML consiste en una serie anidada de etiquetas abiertas (<) y cerradas (>), donde cada etiqueta tiene ciertos valores. Es potestad del programador definir los nombres de las etiquetas y sus combinaciones permitidas (por ejemplo, que una etiqueta <cal/e> no puede encontrarse fuera de unas etiquetas <direccion></direccion>). Para ello, puede usar documentos **DTD** (*Document Type Definition*, definición de tipos de documentos) o esquemas XML. Las DTD y los esquemas XML son usadas por los **analizadores sintácticos** (*parsers*) para comprobar si un documento XML es válido; es decir, si cumple la gramática.

```
<?xml version="1.0" encoding="utf-8"?>

<!ELEMENT E-MAIL (DE, A, ASUNTO, CUERPO)>
<!ELEMENT DE (#CDATA)*>
<!ELEMENT A (#CDATA)*>
<!ELEMENT ASUNTO (#CDATA)*>
<!ELEMENT CUERPO (#CDATA)*>
```

Figura 24. Ejemplo de una DTD

```
<E-MAIL>
<DE>Miguel Ángel Abián</DE>
<A>javaHispano</A>
<ASUNTO>Saludos</ASUNTO>
<CUERPO>Saludos a los lectores de javaHispano.</CUERPO>
</E-MAIL>
```

Figura 25. Ejemplo de un documento XML válido según la anterior DTD



```
<?xml version="1.0" encoding="utf-8"?>

<!-- Esquema XML para un pedido -->
<!-- Un pedido se compone de varias líneas de pedido, cada una con el -->
<!-- nombre del producto y el número de unidades -->

<xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'>

    <xsd:element name="pedido">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="lineaPedido"
                    minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="lineaPedido">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="nombreProducto"/>
                <xsd:element ref="cantidad"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="nombreProducto" type="xsd:string"/>
    <xsd:element name="cantidad" type="xsd:string"/>

</xsd:schema>
```

**Figura 26. Ejemplo de un esquema XML. Los esquemas permiten introducir más información (tipos de datos, por ejemplo) que las DTD.**



```
<?xml version="1.0" encoding="utf-8"?>

<pedido
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xsi:noNombrespaceSchemaLocation="lineaPedido.xsd">
  <lineaPedido>
    <nombre>silla de oficina</nombre>
    <cantidad>2</cantidad>
  </lineaPedido >
  <lineaPedido>
    <nombre>mesa de reuniones</nombre>
    <cantidad>3</cantidad>
  </lineaPedido>
  <lineaPedido>
    <nombre>cajonera</nombre>
    <cantidad>6</cantidad>
  </lineaPedido>
</pedido>
```

**Figura 27. Ejemplo de un documento XML. Los esquemas permiten introducir más información (tipos de datos, por ejemplo) que las DTD**

Resulta innegable que XML constituye un gran paso adelante para lograr la interoperabilidad sintáctica, tan necesaria para el comercio electrónico. He aquí algunas ventajas generales de usar XML:

- Cualquier documento y cualquier tipo de dato puede expresarse como un documento XML. Este lenguaje permite definir los datos independientemente de cualquier lenguaje o plataforma y, por ello, constituye un formato universal para el intercambio de datos.
- Por su propia naturaleza, XML es autodescriptivo. Un documento XML no sólo almacena datos, sino también la estructura de éstos. Resulta fácil compartir documentos XML entre empresas y personas, porque una persona que lea un documento XML descubrirá enseguida su estructura.
- Los documentos XML pueden desarrollarse tal y como quiera una persona o una organización. Nada impide a una empresa que desarrolle una serie de esquemas o DTDs para los tipos de documentos con que trabaja (facturas, pedidos, órdenes de producción, albaranes, reclamaciones...) y que exija a sus clientes y proveedores que usen documentos XML conformes a esos esquemas o DTDs.
- La estructura de los documentos XML puede comprobarse automáticamente, de manera que se rechacen aquellos documentos que no están contruidos siguiendo los esquemas o DTDs elegidos. Si dos empresas trabajan con un conjunto común de esquemas o DTDs, pueden analizar cualquier documento que una vaya a enviar a la otra, con lo que se evita el envío de documentos incorrectos. Por ejemplo, si una empresa



va a enviar una factura en la que falta el importe y usa un analizador de XML antes de hacerlo, se ahorrará enviar un documento que el SI de la otra empresa rechazará.

- Como XML se basa en texto ordinario, los documentos XML son idóneos para transportarlos a través de Internet; red donde conviven muchas plataformas y sistemas informáticos distintos, cada uno con sus propios formatos de archivos.
- Como Unicode es el juego de caracteres predefinido para XML, se pueden crear documentos XML para casi todos los lenguajes humanos.

Ha sido y es enorme el impacto que han tenido XML y las tecnologías que lo emplean (XSL, XSLT, UDDI, etc.) en las empresas relacionadas con las tecnologías de la información:

- 1) Sin lugar a dudas, XML se ha convertido en el formato universal para intercambiar datos entre organizaciones, ya sean públicas o privadas. Tanto la Comisión Europea como los gobiernos estadounidense y japonés promueven la adopción de tecnologías basadas en XML. La llegada de XML y de la Web ha hecho caer en desuso las redes privadas VAN, que usaban aplicaciones “traductoras” para pasar los mensajes de un miembro de la red al formato de otro miembro y transmitían los mensajes mediante formatos binarios propios. Las rígidas arquitecturas de las redes VAN, de costoso mantenimiento, a duras penas han podido resistir a XML y a la popularización de la Web.
- 2) El panorama de las transacciones B2B se ha visto radicalmente modificado por la aparición de XML. Hay en el mercado docenas de productos compatibles con las especificaciones para el intercambio comercial de documentos XML desarrolladas por organizaciones como OASIS (*Organization for the Advancement of Structured Information Standards*), UN/CEFACT (*United Nations Center for Trade Facilitation and Electronic Business*), el consorcio ebXML, RosettaNet, etc. Estas organizaciones tienen por fin promover **estándares universales** para el uso de XML, con el propósito de facilitar el intercambio de datos comerciales electrónicos. Por poner un solo ejemplo, la arquitectura ebXML comienza con un modelo de los datos y procesos de la empresa, convierte el modelo en esquemas XML y define los requisitos para las aplicaciones que procesan los documentos y los intercambian con sus socios.
- 3) XML ha sido la pieza clave (el “pegamento”) a la hora de integrar aplicaciones empresariales. Como las empresas odian tirar a la basura el dinero invertido en sistemas de información, XML se ha hecho muy popular para la **integración** de aplicaciones, sistemas y bases de datos ya existentes, con el objetivo de que los sistemas resultantes puedan integrarse con sistemas ERP y comerciar electrónicamente mediante la Web. Indirectamente, XML ha cambiado mucho las intenciones de los vendedores de sistemas ERP (SAP, Oracle, PeopleSoft, IBM, etc.), que ven ahora que utilizar formatos propios y no tender a la integración de sus productos con los de la competencia les dejaría, a medio plazo, fuera del mercado.



- 4) XML ha conseguido introducirse en el corazón de las dos plataformas de software más populares (J2EE y .Net). Por una parte, ambas tienen APIs para trabajar directamente con todas las tecnologías basadas en XML. Por otra parte, tanto J2EE como .Net usan XML para configurar y distribuir las aplicaciones ASP .Net (*Active Server Pages*) y JSP (*Java Server Pages*), respectivamente.
- 5) XML ha liberado a las empresas de desarrollar en una plataforma u en otra, pues los servicios web, basados en XML, permiten integrar aplicaciones de distintas plataformas.
- 6) Casi todos los vendedores de sistemas de gestión de bases de datos relacionales han incorporado XML a sus productos. Algunos incluso ofrecen bases de datos que sólo almacenan XML; la mayoría incluye la conversión automática de las tablas relacionales (modelo Entidad-Relación) a esquemas XML.
- 7) El impacto de XML en los **portales** ha sido grande. Distintos portales pueden tener un mismo “esqueleto” en XML y distintas presentaciones (mediante distintas plantillas XSLT [*Stylesheet Language Transformation*, transformación del lenguaje de hojas de estilo]). Así se evita tener que escribir y mantener el código HTML para cada portal: si se desea cambiar la apariencia gráfica de uno no se necesita modificar todas sus etiquetas HTML; basta usar otra plantilla XSLT. La unión de XML y XSLT ha resultado muy ventajosa para separar contenido y presentación.

En el ámbito empresarial, **el uso de XML y de la Web** para intercambiar mensajes se conoce como el enfoque EDI actual o el *nuevo* enfoque EDI. Tanto el nuevo enfoque como el convencional se basan en el intercambio de mensajes consensuados entre las partes, si bien el convencional no emplea XML ni Internet (y, en consecuencia, tampoco la Web). ¿Qué ventajas proporciona el nuevo enfoque frente al anterior? En primer lugar, XML es un estándar universal y abierto, para el cual hay disponibles muchas herramientas gratuitas o muy baratas. En el caso de los estándares ANSI X12 y UN/EDIFACT, ambos abiertos e incompatibles entre sí, el problema residía en implementarlos (la norma UN/EDIFACT ocupa unas tres mil hojas) y en que las implementaciones eran propiedad de las grandes empresas capaces de tal proeza. Sorprende, vaya por caso, el espacio que la norma ANSI X12 dedica al conjunto de transacciones 864 (*Text Message*). Es más: puede que hielen la sangre a quien tenga que implementarlas. Sin embargo, tantas páginas sólo vienen a definir lo que hoy consideramos un correo electrónico sin documentos adjuntos. En teoría, los estándares anteriores buscaban que el proceso de intercambiar mensajes EDI fuera independiente de cualquier hardware o software; pero la realidad ha sido otra: los sistemas traductores se hallan fuertemente ligados a las plataformas para las cuales se desarrollan.

En segundo lugar, XML aprovecha las ventajas de la Web: los documentos XML son de texto plano y pueden transmitirse perfectamente por HTTP. Los documentos EDI usados en el enfoque EDI convencional se transmitían a través de redes privadas, mediante protocolos de red (como X.25) que no pertenecían a TCP/IP.

En tercer lugar, XML permite crear documentos legibles para los humanos, quienes pueden intuir su contenido sin necesidad de usar ninguna aplicación; XML se basa en Unicode y es lo bastante flexible como para que cualquier empresa cree sus propios documentos. En el enfoque EDI convencional, los mensajes intercambiados solían



codificarse en formato binario, lo que obligaba a usar aplicaciones muy específicas para leerlos y mostrarlos. Frente a la flexibilidad de XML para generarla infinidad de documentos internacionales, los mensajes EDI destacaban por su rigidez y sólo permitían usar el juego de caracteres ASCII, insuficiente para otros idiomas que no sean el inglés.

En cuarto lugar, existen muchos analizadores sintácticos (*parsers*) de XML y APIs que pueden usarse para transformar los documentos XML en representaciones que las aplicaciones pueden procesar (objetos, registros), así como muchos SI que permiten importar y exportar datos en XML (hay bases de datos que manejan XML como formato propio). Incluso si una empresa debe desarrollar algún tipo de traductor formato propio-XML, puede usar muchas herramientas y *parsers* que ya existen (la mayoría, de código abierto o libre), lo cual abarata mucho el desarrollo, implantación y configuración del traductor. En otras palabras: cuando XML no permite una **interoperabilidad sintáctica completa** entre dos SI, facilita enormemente conseguirla. Recurriendo a una metáfora zoológica, podemos decir que los documentos XML son camaleones (pueden ser procesados por muchas herramientas y aplicaciones, y pueden mostrarse de muchas maneras distintas). Huelga decir que el enfoque EDI convencional nunca contó con herramientas y APIs como las descritas, ni con ninguna cualidad camaleónica: cada empresa debía reinventar la rueda. Como los traductores EDI se hacían a medida para el SI de cada empresa, no se podían aprovechar para empresas que usaran otros SI (por lo general, los traductores eran propiedad de las compañías arrendadoras de las VAN).

Finalmente, el acceso a la Web es muy barato y permite hacer negocios con empresas de todo el mundo. Las empresas que en las décadas de 1970, 1980 o 1990 usaban el enfoque EDI convencional se veían obligadas a trabajar con redes privadas, cuyo alquiler resultaba muy caro y a las que estaban conectadas pocas empresas. Incluso hoy, cuesta unos 40.000 €uros unirse a una VAN de tamaño medio. Trabajar con estas redes limita las oportunidades de hacer negocios; pues una empresa sólo puede negociar con otras conectadas a la misma VAN que ella, y las transacciones van de uno a uno (punto a punto): del emisor al buzón del receptor. En cambio, la Web permite hacer negocios con un número de empresas virtualmente infinito y las transacciones son de uno a muchos. Consideremos, por ejemplo, el caso de un hospital que necesita comprar material sanitario en grandes cantidades. Mediante la web, el hospital podrá acceder a un *e-marketplace* y presentar a la vez su pedido a docenas de distribuidores y fabricantes de material sanitario (negocio de uno a muchos). Una vez recibidas las correspondientes ofertas, podrá elegir la más barata.



**Nota:** En el mundo del comercio electrónico no cabe sólo el blanco y el negro: hay muchos grises. Así, hay enfoques combinados que mezclan las características del enfoque EDI convencional y del nuevo (basado en la Web y XML). Unos sustituyen las VAN (redes de valor añadido) por Internet, pero mantienen formatos que no se basan en XML. Otros usan a la vez las VAN, Internet y la Web, pero mantienen sus propios formatos propietarios. Algunos otros usan traductores XML/EDI. Por poner un solo ejemplo de estos enfoques combinados, mencionaré el sistema *TradeWeb* de la empresa General Electric Information Systems (antes propiedad de Global eXchange Services, comprada por General Electric en 2002). TradeWeb se basa en un servidor web que expone los formularios de los documentos comerciales más usuales (facturas, pedidos, etc.). Los usuarios sin sistemas EDI convencionales –PYMES, tiendas– acceden al servidor mediante Internet y un navegador web. Cuando un usuario de este tipo rellena un formulario, TradeWeb lo envía a una VAN donde están conectadas las empresas que pueden atender su petición (grandes empresas con sistemas EDI convencionales). Como los pequeños usuarios carecen de aplicaciones para recibir y mostrar mensajes EDI, las empresas conectadas a la VAN se los envían en formato web (HTML). En Estados Unidos y México, unos 6.000 proveedores de DaimlerChrysler usan TradeWeb.

Pese a los laureles y excelencias de XML, no proporciona una interoperabilidad completa, pues no incorpora ningún mecanismo para garantizar la interoperabilidad semántica. XML sirve para especificar el formato y estructura de cualquier documento; pero no impone ninguna interpretación común de los datos del documento. Desde el punto de vista semántico, XML resulta casi inútil: dado un documento XML, no se pueden reconocer en él los objetos y relaciones del dominio de interés.

El marcado de los documentos XML es una forma de metadatos. Etiquetas como *<precio>* o *<autor>* ayudan a que los humanos intuyamos el significado de lo que vaya entre las etiquetas. Cuando una persona lee una DTD, puede extraer la semántica de sus elementos prestando atención a los nombres de los elementos (*<fechaEntrega>*, p. ej.) o a los comentarios que figuran en ella. En otras palabras: puede interpretar semánticamente la DTD. Por el contrario, para un programa que procese documentos XML, las etiquetas carecen de sentido o significado.

En el caso del comercio B2B, que XML no incluya ningún mecanismo para la interpretación semántica automática constituye un gran problema. Varias empresas pueden intercambiar documentos XML si todas las partes se han puesto de acuerdo sobre las DTD (o los esquemas) que van a usar, pero surgirán problemas cuando aparezcan empresas que usen otras DTD, aunque sean equivalentes. Por ejemplo, un SI que acepte etiquetas *<precio>* no será capaz de procesar etiquetas *<precioUnidad>*, aunque sean semánticamente equivalentes. Por ahora, la única manera de integrar las DTD procedentes de empresas que no han establecido acuerdos previos consiste en que una persona establezca la relación entre las etiquetas de una DTD y las de las otras.

Las DTD apenas admiten semántica: no hay nada en las DTD que corresponde al concepto de herencia, y el único tipo de datos primitivo que admiten es *PCDATA*, es decir, cadenas de texto. Para ser justo, dejo constancia de que los esquemas XML son más ricos en semántica que las DTD, pues permiten incluir algunas expresiones semánticas (*X* contiene a *Y*; *X* sigue a *Y*; si *X*, también *Y*; si *X*, no *Y*). Ahora bien, éstas



no bastan para representar los objetos del complejo mundo real. De todos modos, incluso si un esquema de XML contiene la escasa semántica que permiten las anteriores expresiones, ningún procesador de XML podrá validarla en un documento.

Si retornamos al ejemplo de la crema de afeitar y suponemos que todas las tiendas virtuales han decidido usar documentos XML en los que aparecen las etiquetas `<precio>x</precio>` (donde *x* es un número que corresponde al precio en euros y en que la coma señala los decimales), podremos hablar de interoperabilidad completa (sintáctica y semántica). Desgraciadamente, esta situación resulta quimérica: en el mundo real, las tiendas usan etiquetas del estilo `<precio>34,45</precio>`, `<coste>34,45</coste>`, `<precioOferta>34,45</precioOferta>`, `<price>30.50</price>`... Huelga decir que ni la Web actual ni XML proporcionan mecanismos para que una máquina sepa que esas etiquetas significan lo mismo o algo similar.

En los últimos años se ha hablado mucho de los servicios web, a veces con entusiasmo monomaniático. El principal argumento comercial que se esgrime a favor de los servicios web consiste en que una empresa cualquiera puede encontrar y usar automáticamente los servicios web ofrecidos por otras (compra de productos, realización de operaciones financieras, reserva de billetes o entradas). Actualmente, las tecnologías basadas en XML no permiten este uso automático de los servicios web, pues no hay asociada una interpretación semántica de tipo automático. Por ejemplo, un servicio web que ofrezca un método *CalcularAmortizacion()* carece de interés para las empresas, pues no sabrán si el cálculo se refiere a la amortización lineal, a la basada en coeficientes, etc. (salvo que exista un acuerdo previo entre las empresas consumidoras del servicio o que haya personas encargadas de interpretar semánticamente el método; esto es, de leer la documentación del servicio web).

Quizás algún lector piense que eso de la semántica es algo muy abstracto y que importa poco o nada, o que lo verdaderamente importante para los negocios electrónicos son los protocolos de red, la conexión física y las políticas de seguridad, o que con XML basta para hacer negocios en el mundo real. No es así: las consecuencias económicas que tiene la falta de interoperabilidad semántica son enormes. En Europa, se calcula que el 20-30% de los gastos relacionados con el comercio electrónico deriva de la falta de interoperabilidad semántica. Este gasto se debe a que, aunque los SI empresariales usen las mismas tecnologías (por ejemplo, XML para describir los datos), el significado de los **modelos de negocio** suele diferir. Incluso nociones comunes tan habituales como “pedido” o “cliente” pueden significar cosas muy distintas en cada empresa. Como consecuencia, el intercambio de información entre empresas suele ser, en el mejor de los casos, un proceso semimanual. La principal dificultad a la que se enfrenta el **modelado empresarial** reside en traducir los **objetos de negocio** de una empresa a los de otras, y viceversa. Sin esa traducción, la integración automática de los SI de las empresas que forman parte de una misma cadena de aprovisionamiento resulta imposible: no existe un entendimiento común.

Para mostrar los problemas que derivan de no contar con un entendimiento común (es decir, con una misma semántica), consideraré un caso real: una empresa que comercializa escaleras encargó tres mil escaleras a otra. El mensaje intercambiado era parecido a éste (por motivos de confidencialidad, he cambiado el formato):



```
<pedido>
  <emisor>empresa XXX</emisor>
  <receptor>empresa YYY</receptor>
  <familiaProducto> A1234VU</familiaProducto>
  <producto> A547</producto>
  <peldaños>6</peldaños>
  <descripcion>Escalera doble “PLUS” de seis peldaños (6P) </descripcion>
  ...
</pedido>
```

Al cabo de unas semanas, la primera empresa recibió tres mil escaleras de siete peldaños (seis peldaños y la plataforma superior). ¿Por qué? Pues porque la segunda interpretó que la plataforma no es un peldaño, interpretación opuesta a la de la primera. Finalmente, ambas empresas se repartieron solidariamente los costes derivados de no compartir un vocabulario común.



## 6. RDF y RDFS: el pegamento semántico

### 6.1 RDF y RDFS

RDF (*Resource Description Framework*, marco de descripción de recursos) es un lenguaje para representar información sobre recursos de la Web (metadatos). Según el W3C, el objetivo de RDF radica en “especificar semántica para los datos basados en XML, de una manera interoperable y estandarizada”. Debido a su carácter general, también puede representar datos.

¿Por qué se recurre a RDF para describir recursos, y no a XML? El motivo estriba en que XML no es apropiado para incluir semántica, tal como se vio en el apartado anterior. El modelo de un dominio puede representarse con varias DTD o varios esquemas XML, y una DTD o un esquema pueden corresponder a muchos modelos distintos. Por así decirlo, XML no está orientado a objetos.

### Ejemplo de la ambigüedad semántica de XML



Rubén compra la silla ref. 120 en la compra con código 112

```
<compra codigo="112">
  <usuario>Rubén </usuario>
  <producto>silla ref. 120</producto>
</compra>
```

```
<producto nombre="silla ref. 120">
  <comprador>Rubén
    <compra codigo="112" />
  </comprador>
</producto>
```

```
<usuario nombre="Rubén">
  <compra codigo="112">
    <producto>silla ref. 120</producto>
  </compra>
</usuario>
```

Miguel Ángel Abián, agosto de 2005

**Figura 28.** Una relación tan sencilla como ésta puede codificarse en XML de distintas formas. Partiendo de los DTD o de los esquemas XML correspondientes a los tres ejemplos, resulta imposible determinar cuál es la relación (*compra*) y cuáles son los objetos (*usuario* y *producto*)

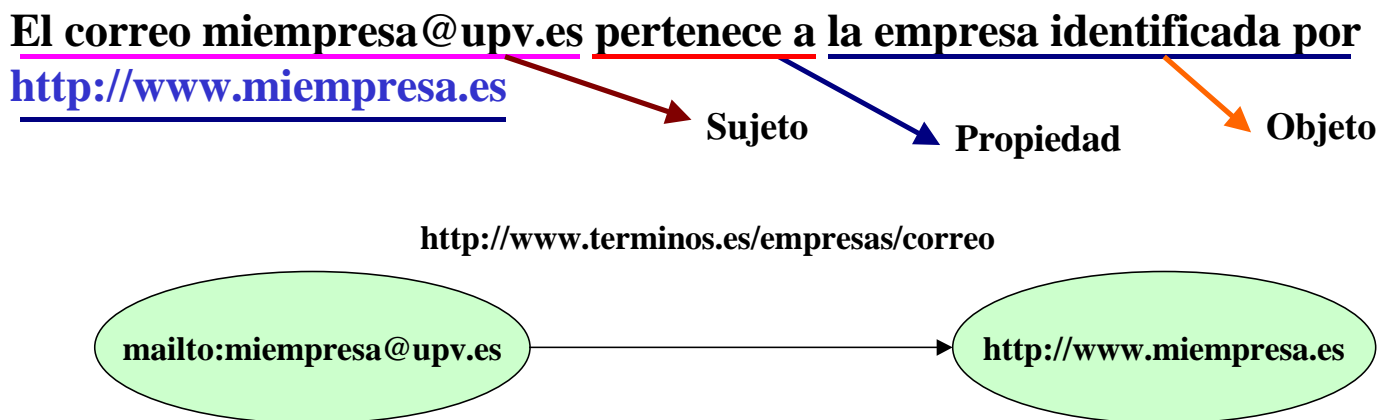


Si bien suele decirse que RDF es un lenguaje (como he escrito al principio de este apartado), lo cierto es que resulta más exacto describirlo como un modelo de datos para las instancias de metadatos o, por abreviar, como un modelo de metadatos. En RDF, la construcción básica es la tripleta (sujeto, propiedad, objeto). Toda tripleta RDF corresponde a una sentencia RDF: la parte que identifica a qué se refiere la sentencia es el **sujeto**; la parte que identifica la característica del sujeto a la que se refiere la sentencia es la **propiedad** o el **predicado**; la parte que identifica el valor de la propiedad es el **objeto**.

En la figura 29 se incluye la representación gráfica de una tripleta de RDF.

## RDF (Resource Description Framework )

- **RDF es una recomendación del W3C para describir recursos**
- **En RDF, el concepto fundamental es la tripleta, que se representa como nodos conectados por líneas con etiquetas (los nodos representan recursos; las líneas con etiquetas, propiedades de estos recursos). Los tres elementos de una tripleta se representan mediante URIs**



Miguel Ángel Abián, julio de 2005

**Figura 29. Fundamentos de RDF**

Los sujetos, las propiedades y los objetos son recursos. Los recursos con que trabaja RDF no son necesariamente recursos presentes en la Web: pueden ser personas, animales, objetos materiales, números de teléfono o de fax, reuniones, ideas o conceptos, etc. En general, un recurso RDF es cualquier cosa con identidad. Para identificarlos se recurre a los URI (*Uniform Resource Identifier*, identificadores uniformes de recursos). Estos localizadores son como unos URL universales (*Uniform Resource Locator*, localizadores uniformes de recursos), que no se limitan a identificar entidades que tengan localizaciones en la Web o que sean accesibles mediante aplicaciones



informáticas. Cualquier organización o persona puede crear URIs y usarlos para trabajar con sus dominios de interés. Frente a los URL, los URI permiten referirse a un mismo recurso ubicado en distintas localizaciones.

Los URI no tienen significado por sí mismos: son identificadores, y la persona o la organización que los crea se responsabiliza de darles significado. Así, para [www.miempresa.com](http://www.miempresa.com), “correo” puede significar “correo general de una empresa”, mientras que para otra empresa ([www.aidima.es](http://www.aidima.es), vaya por caso) puede significar “correo de la persona de contacto de una empresa”. Una aplicación que extraiga datos RDF de [www.miempresa.com](http://www.miempresa.com) y [www.aidima.es](http://www.aidima.es) considerará que las respectivas propiedades <http://www.miempresa.es/empresas#correo> y <http://www.aidima.es/asociados#correo> se refieren a cosas distintas, si bien será incapaz de averiguar en qué difieren. Igualmente, una aplicación que encuentre dos o más recursos vinculados a un mismo URI, sabrá que se refieren a un mismo concepto, aun cuando no podrá saber cuál es. El vocabulario de RDF se define en el recurso <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

Aparte de en forma gráfica o en forma de tripletas (sujeto, propiedad, objeto), las sentencias RDF pueden representarse en XML (RDF/XML). Consideremos, vaya por caso, la sentencia “La página web <http://www.uv.es/documentacion/java> ha sido creada por Luisa Rodenas”. En la forma de (sujeto, propiedad, objeto) quedaría así:

(<http://www.uv.es/documentacion/java>, <http://www.definiciones.org/definiciones/creador>, “Luisa Rodenas”)

En forma de XML quedaría así::

```
<?xml version="1.0", encoding="UTF-8"?>
```

```
<rdf:RDF
```

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

```
  xmlns:definiciones="http://www.definiciones.org/definiciones/">
```

**Encabezado**

```
  <rdf:Description rdf:about="http://www.uv.es/documentacion/java">
```

```
    <definiciones:creador>
```

```
      Luisa Rodenas
```

```
    </definiciones:creador>
```

```
  </rdf:Description>
```

**Cuerpo**

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

**Pie**

Nótese que la propiedad *creador* y la página web se representan como URIs, mientras que el objeto se representa como una cadena de texto. No hay impedimento ninguno a que el objeto (la persona llamada Luisa Rodenas) se represente mediante URIs (por ejemplo, como <http://www.uv.es/alumnos/lrodenas> o <mailto:lrodenas@uv.es>). De hecho, en este caso es mucho más conveniente usar URIs, pues así los programas no confundirán a la Luisa Rodenas que ha creado una página en la web de la Universidad de Valencia (España) con otra Luisa Rodenas que, vaya por caso, estudie en una universidad colombiana. Si usamos una dirección de correo para identificar el objeto, el código XML correspondiente a la sentencia anterior queda así:



```
<?xml version="1.0", encoding="UTF-8"?>
```

```
<rdf:RDF
```

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

```
  <rdf:Description rdf:about="http://www.uv.es/documentacion/java">
```

```
    <definiciones:creador rdf:resource="mailto:lrodenas@uv.es" />
```

```
  </rdf:Description>
```

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

Encabezado

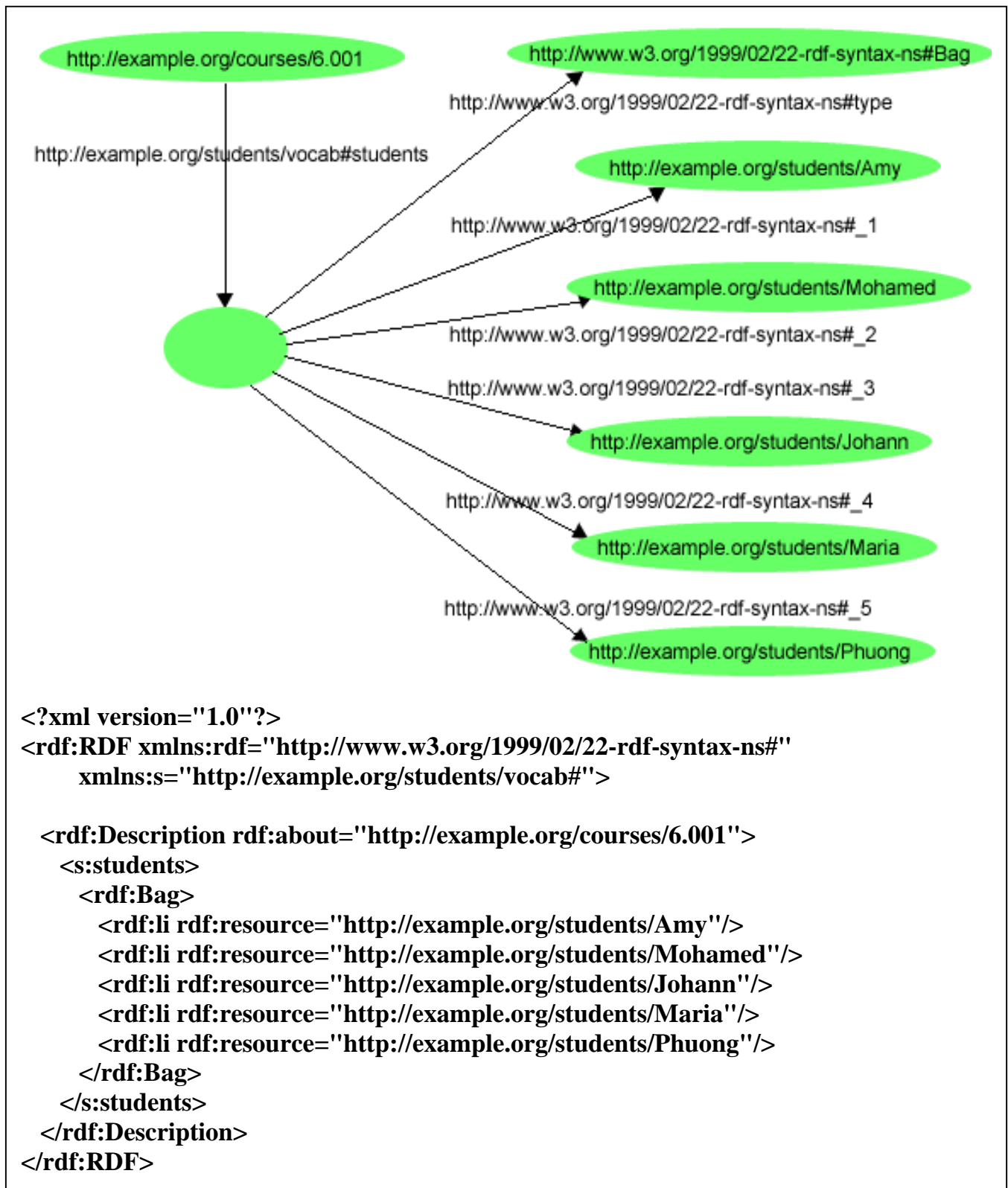
Cuerpo

Pie

La ventaja de usar RDF/XML, esto es, la representación en XML de RDF, estriba en poder reutilizar todas las herramientas existentes para XML. Se pueden aprovechar los analizadores sintácticos, las transformaciones XSLT, la representación en memoria de **los objetos** XML (mediante DOM/SAX), etc. Resulta importante **no identificar RDF con XML**: existe una representación en XML de RDF, pero eso es todo. RDF no está ligado a XML; si mañana se decide usar otra representación para RDF, no habrá que modificar la sintaxis de tripletas ni la semántica de RDF.

En la siguiente página se muestra la representación en RDF de una sentencia más complicada que la anterior.





**Figura 30. Representaciones en RDF de la sentencia “El curso 6.001 tiene como estudiantes a Amy, Mohamed, Johann, Maria y Phuong”. Extraído de la documentación oficial sobre RDF (W3C)**



RDF no se asocia a ningún dominio en particular: se puede emplear en cualquier campo. Cada persona u organización puede definir su propia terminología o vocabulario mediante lo que se conoce como RDFS (*RDF Schema*, esquema RDF), que se define en función de sentencias RDF. Además, RDFS permite especificar las entidades a las que pueden aplicarse los atributos del vocabulario. Por caso, en el dominio de una biblioteca se podría usar RDFS para definir un vocabulario concreto (*autorDe*, *tieneCarnetDeSocio*, *estaSancionado*, etc.) y especificar condiciones tales como que *estaSancionado* sólo puede aplicarse a socios de la biblioteca.

¿Resulta verdaderamente necesario definir un esquema para RDF? Sí: un esquema RDF permite comprobar si un conjunto de tripletas RDF (un conjunto de metadatos, en definitiva) resulta válido o no para ese esquema. Al disponer de un esquema RDF, se puede comprobar si las propiedades aplicadas a los recursos son correctas (un reloj no puede tener una propiedad que sea *NumeroSeguridadSocial*, p. ej.) y si los valores vinculados a las propiedades tienen sentido (p. ej., carece de sentido que una propiedad *vendeAccionDeBolsa* tenga como valor a un simpático marsupial). RDFS permite controlar la validez de los valores y restringir las entidades a las cuales pueden aplicarse ciertas propiedades.

Pese a la similitud entre los términos “esquema RDF” (RDFS) y “esquema XML”, sus significados son muy distintos. Los esquemas XML, al igual que las DTD, especifican el orden y las combinaciones de etiquetas que son aceptables en un documento XML. En cambio, los esquemas RDF especifican qué interpretación hay que dar a las sentencias de un modelo de datos RDF; mas dejan libre la representación sintáctica del modelo.

Así, un esquema XML puede obligar a que las etiquetas *<Producto>* de los documentos XML estén dentro de etiquetas *<Vendedor>*; pero carece de medios para expresar que hay una relación *vende* entre las instancias de *Vendedor* y las de *Producto*, o que un *Vendedor* es una especialización (subclase) de *Persona*. En síntesis, los esquemas XML modelan datos expresados en XML, mientras que los esquemas RDF (RDFS) modelan conocimiento. Dicho de otra manera: XML y los esquemas XML son un lenguaje de modelado de datos, en tanto que RDF y RDFS son un lenguaje de modelado de metadatos.

El modelo de metadatos que RDFS permite expresar coincide con el de los lenguajes de programación orientada a objetos (Smalltalk, Java, C#), pues permite expresar clases e instancias. Si no conoce bien estos conceptos, puede consultar estos tutoriales:

- 1) <http://www.javahispano.org/tutorials.item.action?id=25>
- 2) <http://www.javahispano.org/tutorials.item.action?id=33>

**Advertencia:** Debo señalar que, si bien el modelo de datos de RDFS se basa en el de los modernos lenguajes orientados a objetos (LOO), existe una diferencia bastante llamativa. A saber: las propiedades definidas en RDFS son globales; esto es, no están encapsuladas como atributos en las definiciones de las clases. A diferencia de lo que ocurre en los LOO, RDFS permite asignar a una clase, sin modificarla, nuevas propiedades. Si desea conocer más sobre las peculiaridades de los lenguajes de programación orientados a objetos, le remito al tutorial <http://www.javahispano.org/tutorials.item.action?id=33>).



Las restricciones que RDFS introduce son análogas a las de los lenguajes OO con tipos (C++, Java, C#). Del mismo modo que una propiedad *come* no puede tener como valor una *piedra*, no se puede llamar al método *come()* de un objeto *piedra*.

RDFS incluye tres clases principales:

- 1) **rdfs:Resource**. Todas las entidades descritas por expresiones RDF son recursos y se consideran instancias de la clase *rdfs:Resource*. Páginas atrás se definió el concepto de recurso en RDF.
- 2) **rdfs:Class**. Corresponde al concepto de clase en los lenguajes de programación orientados a objetos. Las clases RDF pueden representar páginas web, organizaciones, personas, búsquedas en la Web, etc. Cada clase es miembro de *rdfs:Class*, clase de todas las clases; cuando se crea una nueva clase mediante un esquema RDF, la propiedad *rdf:type* del recurso representado por la clase adquiere como valor el recurso *rdfs:Class* o alguna subclase de *rdfs:Class*. Cuando un recurso tiene una propiedad *rdf:type* cuyo valor es una determinada clase, se dice que el recurso es una instancia de esa clase. Todas las clases son recursos. Una tripleta RDF/RDFS como (*Persona*, ***rdf:type***, ***rdfs:Class***) indica que *Persona* es una clase.
- 3) **rdf:Property**. Representa el subconjunto de recursos RDF que son propiedades. El dominio de estos recursos se describe mediante la propiedad *rdfs:domain*, y el rango mediante *rdfs:range*; las jerarquías de propiedades se describen mediante *rdfs:subPropertyOf*.  
*rdfs:range* es una instancia de *rdf:Property* que se usa para establecer que los valores de una propiedad son instancias de una o más clases; *rdfs:domain* es una instancia de *rdf:Property* empleada para establecer que un recurso con una cierta propiedad es instancia de una o más clases. Por ejemplo: las tripletas (*BonoTesoro*, ***rdf:type***, ***rdfs:Class***), (*vende*, ***rdf:type***, ***rdf:Property***) y (*vende*, ***rdfs:domain***, *BonoTesoro*) establecen que las sentencias RDF/RDFS con la propiedad *vende* tienen como sujetos instancias de la clase *BonoTesoro*. Otro ejemplo: las tripletas (*Banco*, ***rdf:type***, ***rdfs:Class***), (*vende*, ***rdf:type***, ***rdf:Property***) y (*vende*, ***rdfs:range***, *Banco*) establecen que las sentencias RDF/RDFS con la propiedad *vende* tienen como objetos instancias de la clase *Banco*.

Una propiedad relevante de RDFS es ***rdfs:subClassOf***, que describe una clase como subclase de otra. Solamente las instancias de *rdfs:Class* pueden tener la propiedad *rdfs:subClassOf*.

Para los metadatos, RDFS define varias propiedades:

- **rdfs:comment** proporciona la descripción en lengua natural de un recurso. Por ejemplo, `<rdfs:comment> "Las neveras enfrían lo que se introduce en ellas"</rdfs:comment>`.



- **rdfs:label** proporciona una versión legible para los humanos del nombre del recurso. P. ej., `<rdfs:label>Frigorífico</rdfs:label>`.
- **rdfs:seeAlso** especifica un recurso que proporciona información adicional sobre el recurso principal. P. ej., `<rdfs:seeAlso http://www.vocabulario.es/aparatos</rdfs:seeAlso>`.
- **rdfs:isDefinedBy** indica cuál es el recurso donde se define el recurso principal. Esta propiedad es una subpropiedad (*rdfs:subPropertyOf*) de la propiedad *rdfs:seeAlso*.

## VOCABULARIO DE RDFS (RDF SCHEMA)

Los usuarios de RDF pueden definir sus propias terminologías mediante el lenguaje de esquemas RDFS. Con RDFS se puede definir un vocabulario especializado, especificar las propiedades aplicables a las clases de objetos y describir las relaciones entre clases. El vocabulario de RDFS se define en <http://www.w3.org/2000/01/rdf-schema#>

### Clases de RDFS:

- a) **rdfs:Resource**
- b) **rdfs:Class**
- c) **rdfs:Literal**
- d) **rdfs:Datatype**
- e) **rdfs:Container**
- f) **rdfs:ContainerMembershipProperty**

### Propiedades de RDFS:

- a) **rdfs:domain**
- b) **rdfs:range**
- c) **rdfs:subPropertyOf**
- d) **rdfs:subClassOf**
- e) **rdfs:member**
- f) **rdfs:seeAlso**
- g) **rdfs:isDefinedBy**
- h) **rdfs:comment**
- i) **rdfs:label**

Miguel Ángel Abián, julio de 2005

**Figura 31. Vocabulario de RDFS**

El vocabulario completo de RDFS se define en el URI <http://www.w3.org/2000/01/rdf-schema#>. En resumen, puede aceptarse que RDFS proporciona:

- Clases
- Jerarquías de clases
- Propiedades
- Jerarquías de propiedades
- Restricciones sobre los dominios y los rangos



La semántica de RDF y RDFS, expresada mediante la lógica de predicados (subconjunto de la lógica de primer orden), se explica en <http://www.w3.org/TR/rdf-mt/>. Codificar la semántica en un lenguaje formal resulta muy conveniente, pues la hace inambigua y comprensible para las máquinas. Con un lenguaje formal se proporciona una base firme para razonar automáticamente.

Por eficacia computacional, RDF y RDFS tienen también sus semánticas expresadas mediante tripletas (sujeto, propiedad, objeto). En las semánticas se incluye un sistema de inferencia robusto y completo. Las reglas del sistema de inferencia son de la forma

SI	C contiene determinadas tripletas
ENTONCES	se añaden a C ciertas tripletas

Donde C es un conjunto no vacío de tripletas. Veamos un ejemplo de regla de inferencia:

SI	C contiene la tripleta (?x, <b>rdfs:subclassOf</b> , ?y) y (y?, <b>rdfs:subclassOf</b> , ?z)
ENTONCES	C también contiene la tripleta (?x, <b>rdfs:subClassOf</b> , ?z)

La notación ?x significa “cualquier recurso x”; los nombres de las variables empiezan con ?. La sentencia (?x, **rdfs:subclassOf**, ?y) define el conjunto de recursos x que son subclases del conjunto de recursos y. En román paladino, la anterior regla de inferencia significa “Si el recurso x es subclase del recurso y y y, al mismo tiempo, el recurso y es subclase del recurso z, entonces el recurso x es subclase del recurso z”.

Reglas de inferencia como la anterior pueden ser aprovechadas por las máquinas para realizar deducciones. Así, una aplicación que busque información sobre marsupiales mostrará páginas sobre canguros rojos, aunque en ellas no figure la palabra “marsupial”, pues podrá deducir que *Canguro* es una subclase (**rdfs:subClassOf**) de *Marsupial* y que *rojo* es una propiedad de *Canguro*. Basándose en esas deducciones, el buscador concluirá que las páginas sobre canguros pueden interesar al usuario.

Veamos un ejemplo completo de inferencia con tripletas RDF/RDFS. Consideremos las siguientes tripletas, que actúan como axiomas para la inferencia:

(InspectorSanitario, **rdf:type**, **rdfs:Class**)  
(InspectorSanitario, **rdfs:SubclassOf**, Funcionario)  
(Restaurante, **rdf:type**, **rdfs:Class**)  
(inspecciona, **rdf:type**, **rdfs:property**)  
(inspecciona, **rdf:domain**, InspectorSanitario)  
(inspecciona, **rdf:range**, Restaurante)

Aceptando lo anterior, se sigue que:

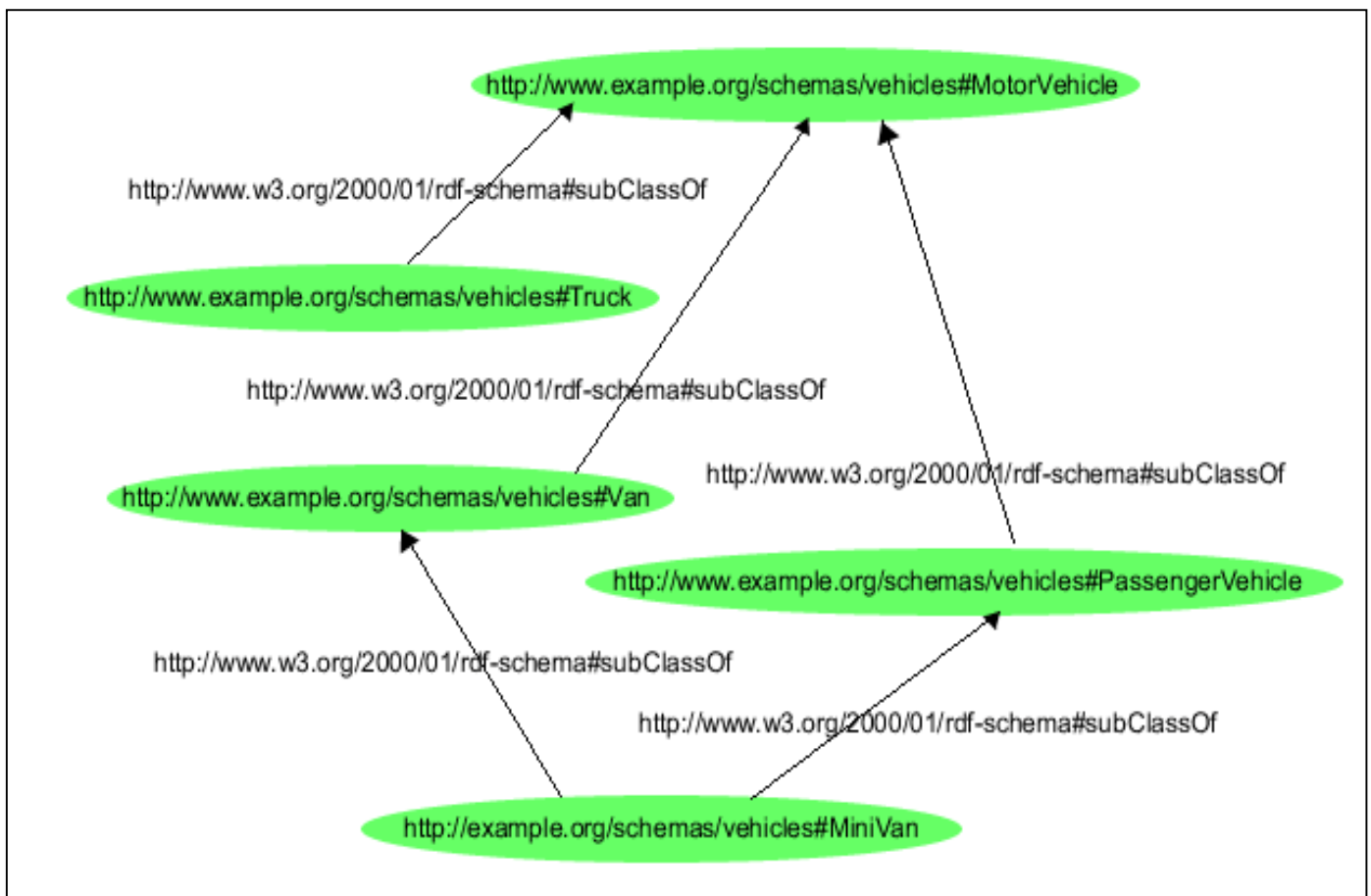
(JuanRodrigo, inspecciona, elBuenTenedor) → (JuanRodrigo, **rdf:type**, Funcionario)

Es decir: si Juan Rodrigo inspecciona el restaurante “El buen tenedor”, entonces Juan Rodrigo es funcionario. Un agente inteligente podría usar esta deducción para buscar los datos sobre Juan Rodrigo en las bases de datos de empleados públicos o para pedir a la Administración información sobre él.



El ejemplo propuesto es muy simple; pero no debemos olvidar que habrá situaciones con cientos o miles de clases, subclases, instancias, propiedades y subpropiedades. En tales situaciones se mostrará la verdadera potencia de la interpretación semántica automática, que permitirá extraer deducciones escondidas entre avalanchas de datos aparentemente inconexos.

La figura 32 incluye un ejemplo de la representación gráfica en RDFS de una jerarquía de clases de vehículos. La clase *Camion* (*Truck*) es subclase directa de la clase *VehiculoMotor* (*MotorVehicle*). La clase *Furgoneta* (*Van*) es subclase directa de la clase *VehiculoMotor* y tiene una subclase *MiniFurgoneta* (*MiniVan*). A su vez, *MiniFurgoneta* es subclase de *VehiculoPasajeros* (*PassengerVehicle*), que a su vez es subclase directa de *VehiculoMotor*.



**Figura 32. Representación en RDF de una jerarquía de clases. Extraída de la documentación oficial sobre RDF (W3C). En el apartado 7 veremos que esta jerarquía es una ontología**

En la figura 33 se muestra en XML el esquema RDF correspondiente a la figura anterior. A diferencia de lo que ocurre con XML, RDFS proporciona una representación única de un modelo de datos.



```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<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://example.org/schemas/vehicles">

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

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

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

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

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

</rdf:RDF>
```

Figura 33. Representación en RDF/XML de la jerarquía de clases representada en la figura anterior. Código extraído de la documentación oficial sobre RDF (W3C). Los atributos ID se utilizan como abreviaturas. Por ejemplo, el ID “MiniVan” representa el URI completo `http://example.org/schemas/vehicles#MiniVan`



En la siguiente figura se muestra la descripción RDF/XML correspondiente a la clase *Persona*.

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

  <rdfs:Class rdf:ID="Person">
    <rdfs:comment>The class of people.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/03/example/classes#Animal"/>
  </rdfs:Class>

  <rdf:Property ID="maritalStatus">
    <rdfs:range rdf:resource="#MaritalStatus"/>
    <rdfs:domain rdf:resource="#Person"/>
  </rdf:Property>

  <rdf:Property ID="ssn">
    <rdfs:comment>Social Security Number</rdfs:comment>
    <rdfs:range rdf:resource="http://www.w3.org/2000/03/example/classes#Integer"/>
    <rdfs:domain rdf:resource="#Person"/>
  </rdf:Property>

  <rdf:Property ID="age">
    <rdfs:range rdf:resource="http://www.w3.org/2000/03/example/classes#Integer"/>
    <rdfs:domain rdf:resource="#Person"/>
  </rdf:Property>

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

  <MaritalStatus rdf:ID="Married"/>
  <MaritalStatus rdf:ID="Divorced"/>
  <MaritalStatus rdf:ID="Single"/>
  <MaritalStatus rdf:ID="Widowed"/>

</rdf:RDF>
```

**Figura 34. Representación en RDF/XML de la clase *Persona*. Código procedente de la documentación oficial sobre RDFS**

Según el código RDF/XML de la figura superior, *Persona* es subclase de la clase *Animal* (es decir, todas las personas son animales). Una persona tiene un atributo de edad (*age*), cuyo valor es un entero. También tiene un número de seguridad social (*ssn*), que es un entero. Por último, tiene una propiedad de estado marital (*MaritalStatus*), que sólo puede tomar uno de estos valores: casado/a (*Married*), divorciado/a (*Divorced*), soltero/a (*Single*), viudo/a (*Widowed*).



## 6.2 Aplicaciones de RDF y RDFS

La aplicación más conocida que usa RDF/RDFS es RSS (al principio RSS significaba *Rich Site Summary*, luego *RDF Site Summary* y, finalmente, *Real Simple Syndication*). RSS es un vocabulario RDF usado para describir información de manera que pueda ser reutilizada por muchas partes; su propósito es distribuir (*syndication*) un conjunto de titulares llamados canales (*feeds* o *channels*). Un canal RSS siempre contiene un título o cabecera, un breve resumen de la información del canal (una noticia, p. ej.) y un enlace a la página web donde está el texto completo. Los enlaces a canales RSS suelen indicarse mediante alguno de estos iconos rectangulares:

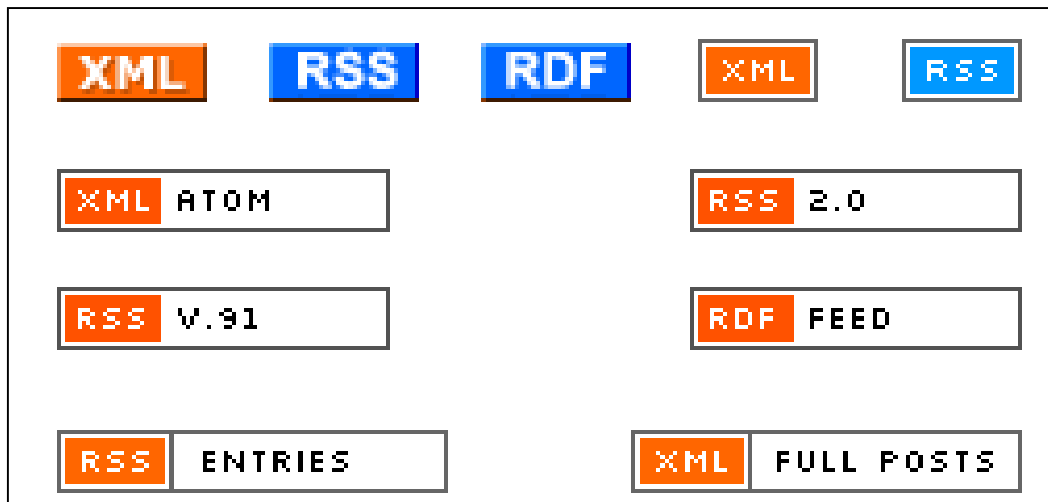


Figura 35. Iconos que indican enlaces a canales RSS

En la Web, RSS suele usarse para distribuir las noticias o los contenidos de una página web. Un uso típico consiste en incluir los titulares de una página web en otra. Incluso hay páginas web formadas sólo por titulares de otras páginas. Existen programas llamados lectores o agregadores de canales (*feed readers* o *feed aggregators*), que procesan las páginas web que usan RSS e informan al usuario de los nuevos artículos y noticias que van publicándose, así como de las modificaciones y actualizaciones que se van produciendo en las noticias y los artículos ya publicados. El usuario que incluya varios canales RSS en un lector de canales (es decir, que se suscriba a ellos) accederá a una versión resumida de las noticias y artículos publicados en distintas páginas web, sin tener que visitarlas de una en una; eso sí, si desea leer un artículo completo, deberá ir a la página donde se almacena.

Debo señalar que no todas las especificaciones de RSS están relacionadas con RDF (las especificaciones 0.91, 0.92, 0.93, 0.94, 2.0 y 2.0.1 no lo están; la 0.9 y la 1.0 sí). En consecuencia, sólo las páginas web que usen la especificación RSS 0.9 o 1.0 (<http://purl.org/rss/1.0/>) podrán participar, al menos directamente, en la Web semántica. Aun cuando RSS es la aplicación RDF/RDFS más popular, no constituye un buen ejemplo de las posibilidades de ambas tecnologías. Por ejemplo, no muestra la estructura (sujeto, propiedad, objeto), un rasgo diferenciador frente a XML.



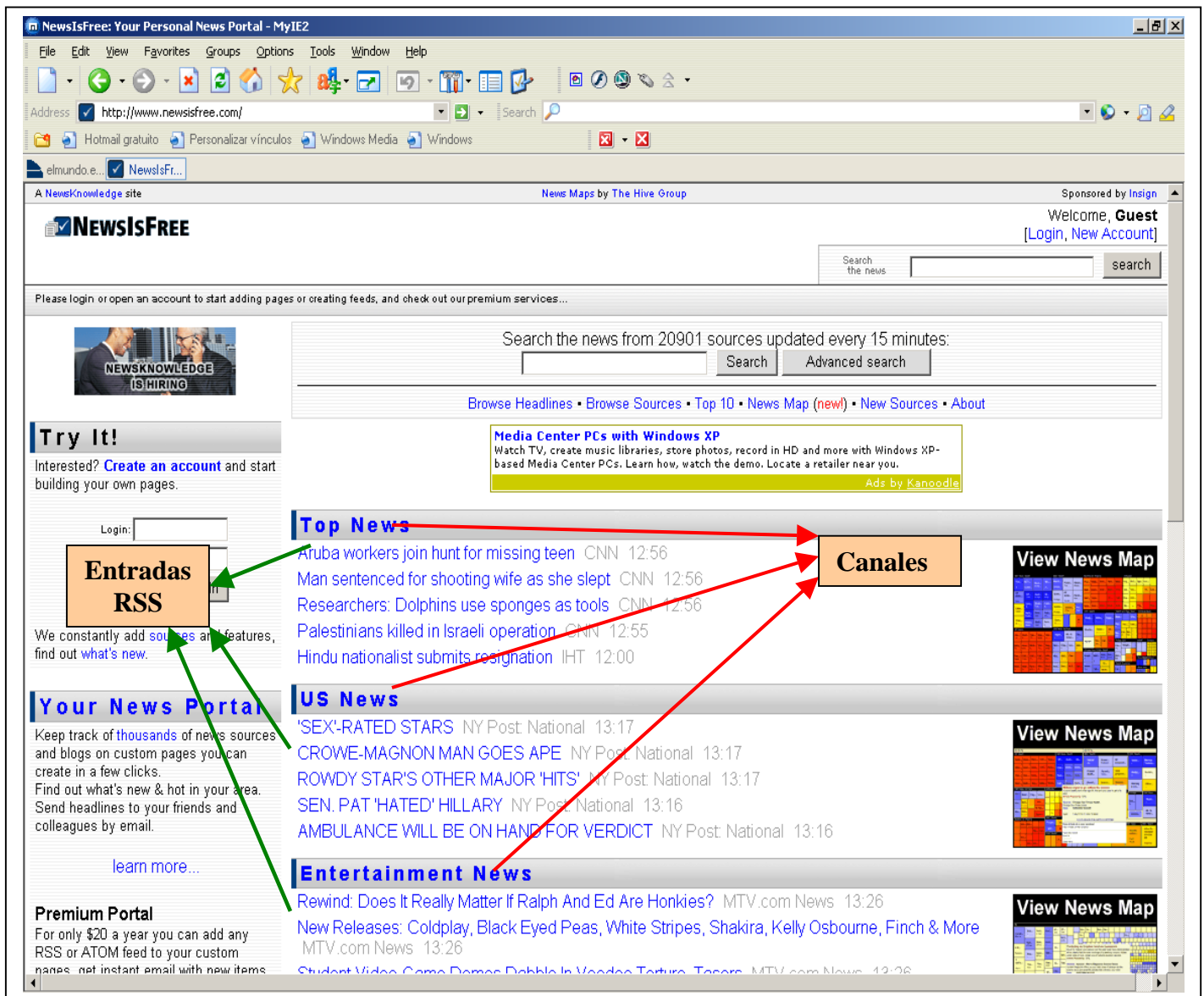


Figura 36. Captura de pantalla de una página web (<http://www.newsisfree.com/>) que es una compilación de titulares

Por lo general, al trabajar con RSS hay que distinguir entre **proveedores de información** y **consumidores de información**. Los primeros publican canales RSS donde cada entrada contiene un resumen de alguna noticia. Los segundos usan lectores de canales RSS para leer los resúmenes de las noticias; si lo desean, acceden a las noticias completas siguiendo los enlaces de las entradas RSS. Los lectores de canales RSS ahorran mucho tiempo a los usuarios habituales de muchos sitios web, pues pueden echar un vistazo rápido a los contenidos de decenas de sitios, sin necesidad de visitarlos de uno en uno.



## VOCABULARIO BÁSICO DE RSS 1.0

**RSS 1.0 ha sido diseñado para ser extensible. Cualquiera puede desarrollar añadir vocabularios adicionales (módulos, en la jerga de RSS) y añadirlos a RSS 1.0. Esta flexibilidad permite incluir y procesar tantos metadatos como uno quiera. El vocabulario que se incluye aquí es el básico.**

- **<channel>**
- **<title>**
- **<link>**
- **<description>**
- **<image>**
- **<items>**
- **<textinput>**
- **<item>**
- **<title>**
- **<link>**
- **<description>**
- **<textinput>**
- **<title>**
- **<description>**
- **<name>**
- **<link>**
- **<image>**
- **<title>**
- **<url>**
- **<link>**

Miguel Ángel Abián, julio de 2005

**Figura 37. Vocabulario básico de RSS 1.0. Todos los metadatos básicos de una noticia o un artículo se pueden capturar con este vocabulario**



```
<?xml version="1.0" encoding="ISO-8859-1"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://my.netscape.com/rdf/simple/0.9/">

<channel>
<title>Barrapunto</title>
<link>http://barrapunto.com/</link>
<description>La informaci&#243;n que te interesa</description>
</channel>

<image>
<title>Barrapunto</title>
<url>http://barrapunto.com/topics/topicbarrapunto.png</url>
<link>http://barrapunto.com/</link>
</image>

<item>
<title>Ondas en el fondo c&#243;smico de neutrinos</title>
<link>http://barrapunto.com/article.pl?sid=05/06/24/129250</link>
</item>

<item>
<title>Visualizaci&#243;n de informaci&#243;n geogr&#225;fica utilizando J2EE</title>
<link>http://barrapunto.com/article.pl?sid=05/06/24/1635215</link>
</item>

<item>
<title>Otro nuevo estado de la materia</title>
<link>http://barrapunto.com/article.pl?sid=05/06/24/123219</link>
</item>

<textinput>
<title>Search Barrapunto</title>
<description>Search Barrapunto stories</description>
<name>query</name>
<link>http://barrapunto.com/search.pl</link>
</textinput>

</rdf:RDF>
```

Figura 38. Archivo RSS extraído de Barrapunto (<http://barrapunto.com/>)



Existe otra aplicación, bastante conocida, que usa RDF/RDFS: **Dublin Core**. Dublin Core (<http://dublincore.org>) es un vocabulario desarrollado por la comunidad de bibliotecología para representar metadatos sobre documentos y recursos. Incluye las etiquetas que se muestran en la figura 39. Su descripción completa está en <http://purl.org/dc/elements/1.1/>).

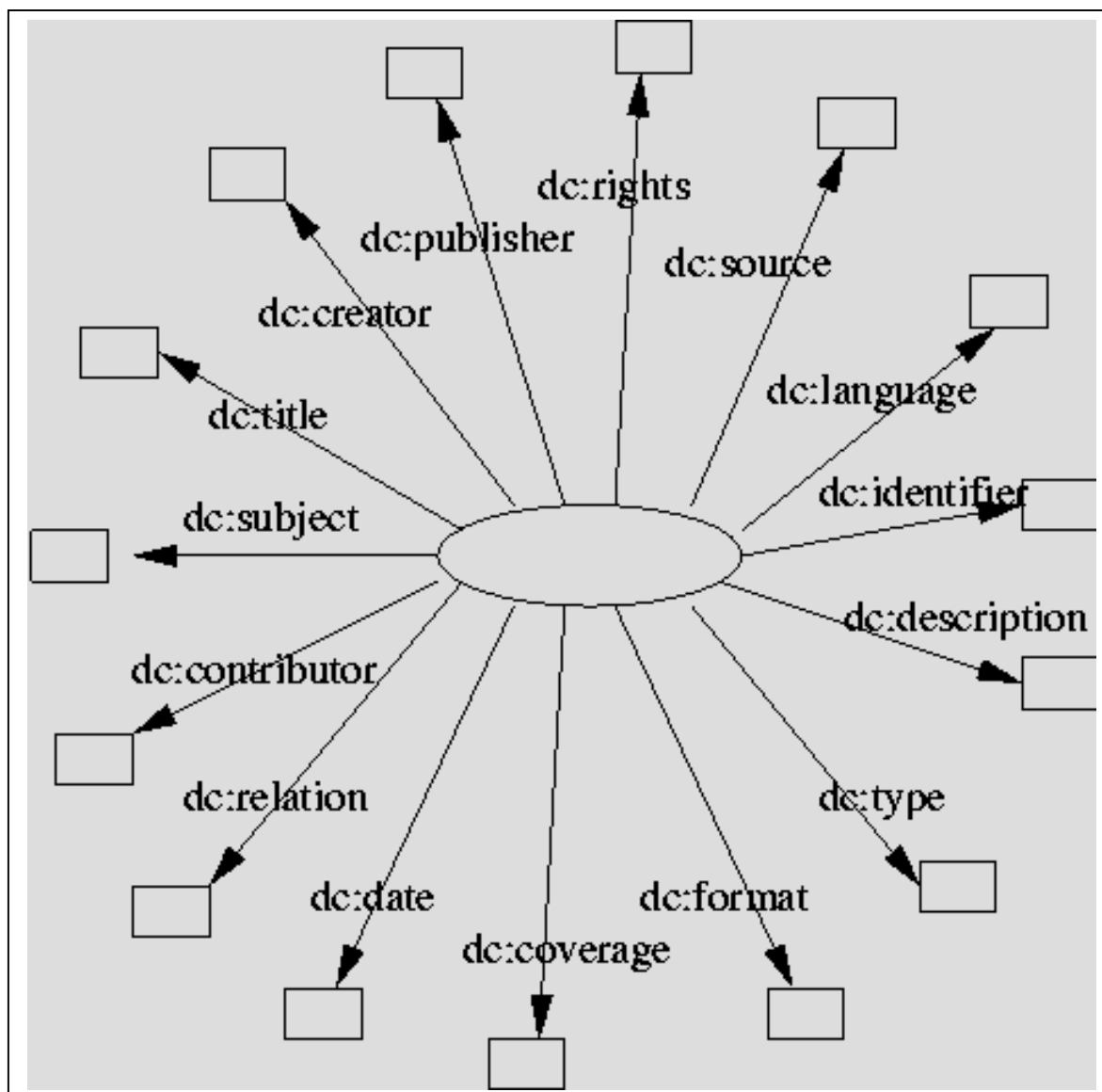


Figura 39. El vocabulario DublinCore. Extraído de <http://dublincore.org/>



```
<?xml version="1.0" encoding="iso-8859-1" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://purl.org/rss/1.0/" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:taxo="http://purl.org/rss/1.0/modules/taxonomy/"
xmlns:syn="http://purl.org/rss/1.0/modules/syndication/">

<channel rdf:about="http://www.elmundo.es/">
<title>BLOGs de elmundo.es</title>
<link>http://www.elmundo.es/</link>
<description>RSS de elmundo.es - Diario El Mundo del siglo XXI</description>
<dc:language>es</dc:language>
<dc:rights>Copyright 2005, Mundinteractivos</dc:rights>
<dc:date>2005-06-06T15:08+02:00</dc:date>
<dc:publisher>Mundinteractivos</dc:publisher>
<dc:creator>Mundinteractivos - El Mundo</dc:creator>
<dc:description>Blog, James Blog</dc:description>
<dc:subject>Noticias, news, última hora, breaking news, españa, spain, europa</dc:subject>
<syn:updatePeriod>hourly</syn:updatePeriod>
<syn:updateFrequency>1</syn:updateFrequency>
<syn:updateBase>2001-01-01T00:00+00:00</syn:updateBase>
<items>
<rdf:Seq>
<rdf:li rdf:resource="http://www.elmundo.es/elmundo/2005/06/03/cineclu/1117794469.html" />
<rdf:li rdf:resource="http://www.elmundo.es/elmundo/2005/06/02/cineclu/1117713388.html" />
</rdf:Seq>
</items>
<image rdf:resource="http://www.elmundo.es/imagen/canalima.gif" />
</channel>
<image rdf:about="http://www.elmundo.es/imagen/canalima.gif">
<title>elmundo.es</title>
<url>http://www.elmundo.es/imagen/canalima.gif</url>
<link>http://www.elmundo.es/</link>
<dc:creator>Mundinteractivos (internet at elmundo.es)</dc:creator>
</image>
<item rdf:about="http://www.elmundo.es/elmundo/2005/06/02/cineclu/1117713388.html">
<title>¡Menos humos! (prohibido fumar en la pantalla)</title>
<link>http://www.elmundo.es/elmundo/2005/06/02/cineclu/1117713388.html</link>
</item>
</rdf:RDF>
```

**Figura 40.** Este documento, extraído de la versión digital del diario español El Mundo, usa el vocabulario RDF conocido como Dublin Core

Por ejemplo, está es la definición en RDFS de la propiedad *rights*:

```
<rdf:Property rdf:about="http://purl.org/dc/elements/1.1/rights">
<rdfs:label xml:lang="en-US">Rights Management</rdfs:label>
<rdfs:comment xml:lang="en-US">Information about rights held in and over the
resource.</rdfs:comment>
```



```
<dc:description xml:lang="en-US">Typically, a Rights element will contain a rights management statement for the resource, or reference a service providing such information. Rights information often encompasses Intellectual Property Rights (IPR), Copyright, and various Property Rights. If the Rights element is absent, no assumptions can be made about the status of these and other rights with respect to the resource.</dc:description>  
<rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />  
<dcterms:issued>1999-07-02</dcterms:issued>  
<dcterms:modified>2002-10-04</dcterms:modified>  
<dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />  
<dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#rights-004" />  
</rdf:Property>
```

En la página siguiente vemos cómo se describiría con el vocabulario DublinCore el vídeo al cual pertenece el siguiente fotograma.



**Figura 41.** Fotograma del vídeo musical de la canción *Atmosphere* del grupo Joy Division, dirigido por Anton Corbijn. Se reproduce sin ánimo alguno de lucro



```
<?xml version="1.0" encoding="iso-8859-1" ?>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">

  <rdf:Description rdf:about="http://videos.ejemplo.com/joydivision/atmosphere.mpg">

    <dc:creator>Anton Corbijn</dc:creator>
    <dc:title>Vídeo musical de Atmosphere</dc:title>
    <dc:description>Vídeo rodado en España en 1988. Se desconoce en qué playa se rodó.
      Duración: 4 min 26 seg. La calidad de la imagen es buena. En el Reino Unido, fue editado
      por Factory Communications Ltd (FACV 213), como apoyo promocional al recopilatorio
      Substance de Joy Division. </dc:description>
    <dc:date>1988-06-20</dc:date>

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

Los metadatos basados en XML o en RDF pueden usarse no sólo para describir recursos audiovisuales, sino también para describir fragmentos de éstos (escenas, secuencias, fotogramas), de modo que puedan encontrarse los fragmentos de interés mediante búsquedas. Por ejemplo, un aficionado a las películas de Stanley Kubrick podría “etiquetar” semánticamente cada fotograma de *2001: A Space Odyssey*, con el fin de encontrarlos rápidamente mediante búsquedas por palabras clave.



## EJEMPLO DE ANOTACIONES SEMÁNTICAS CON MPEG-7

### Descripción en MPEG-7 de una secuencia de una película

```
<Mpeg7>
  <Description xsi:type="SemanticDescriptionType">
    <Semantics>
      <Label>
        <Name> Fotograma de 2001 </Name>
      </Label>
      <Definition>
        <FreeTextAnnotation>
          Momento en que el mono golpea contra un esqueleto el hueso que le ha servido de arma. Este fotograma precede a una de las elipsis cinematográficas más brillantes de la historia del cine, la cual representa un salto de miles de años.
        </FreeTextAnnotation>
      </Definition>
      <MediaOccurrence>
        <MediaLocator>
          <MediaUri> sec9-12.jpg </MediaUri>
        </MediaLocator>
      </MediaOccurrence>
    </Semantics>
  </Description>
</Mpeg7>
```

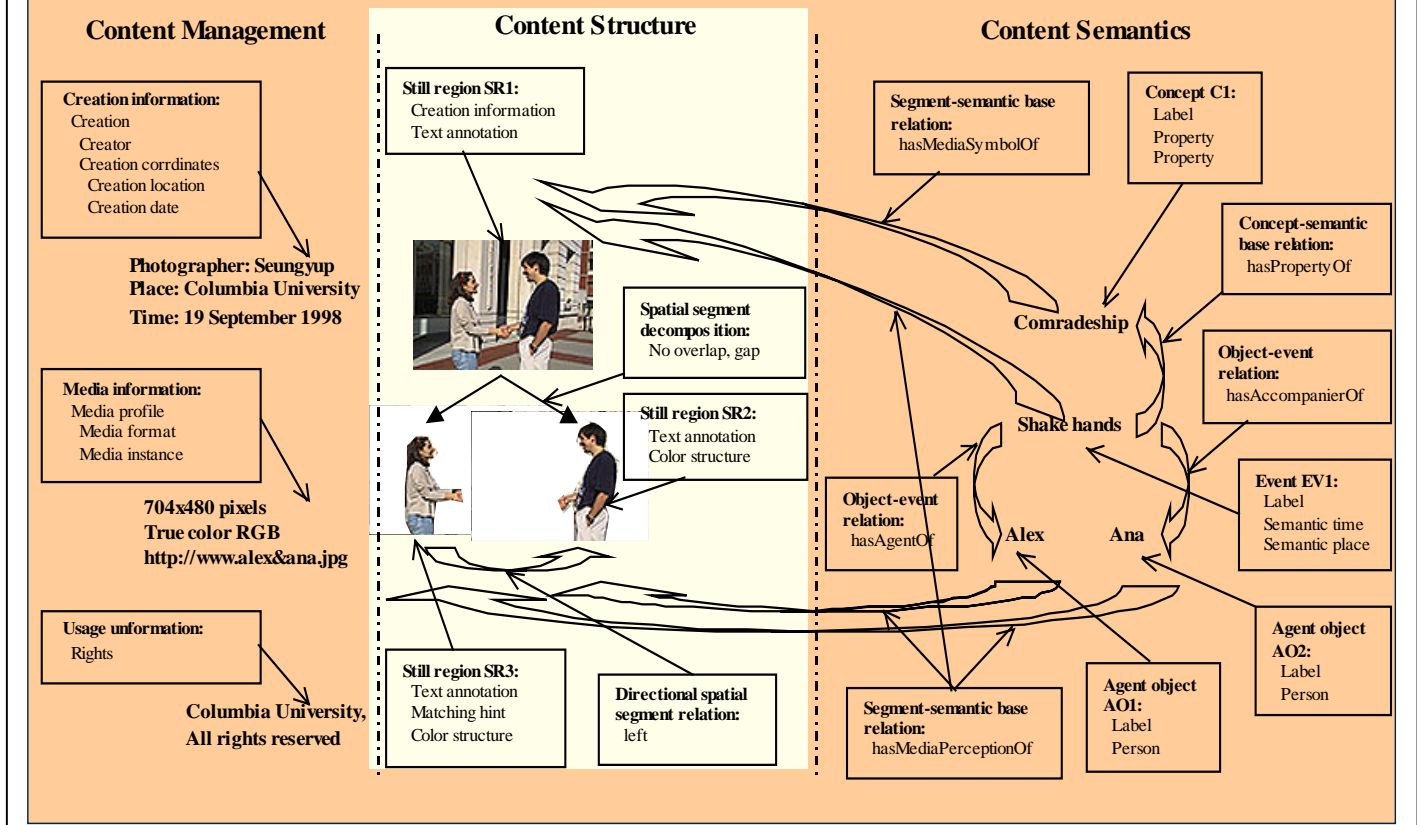


Miguel Ángel Abián, agosto de 2005

**Figura 42. Anotación semántica del fotograma de una película. MPEG-7 es un estándar para describir los datos multimedia (<http://www.cselt.it/mpeg>). Se basa en XML y RDF**



# MPEG-7 Description



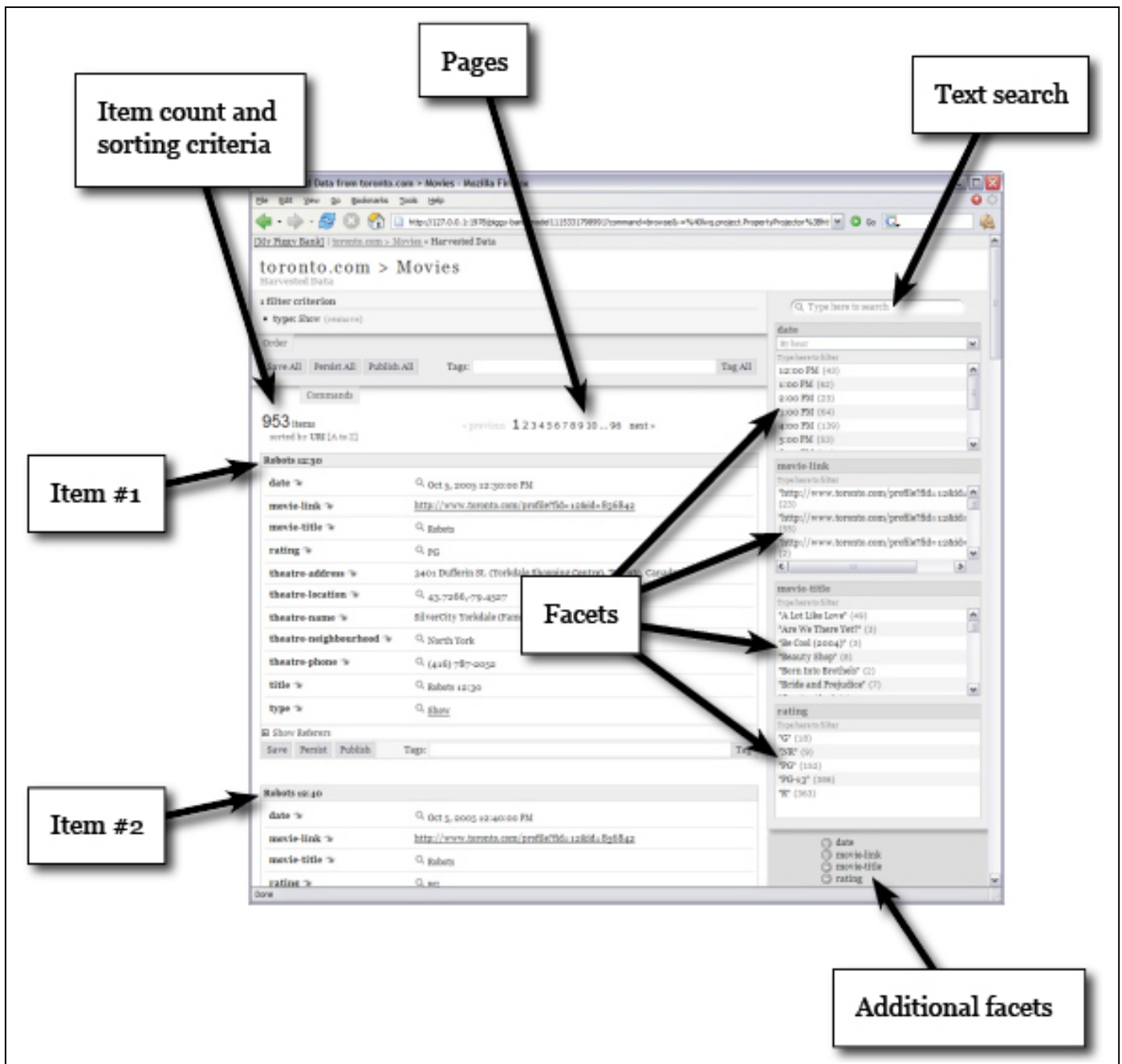
**Figura 43. Ejemplo de las posibilidades que ofrece MPEG-7 para describir recursos audiovisuales. Imagen de Ana B. Benítez**

La extensión *Piggy-Bank* (<http://simile.mit.edu/piggy-bank/>) para el navegador web *Firefox* (<http://www.mozilla.org/products/firefox/>), mencionada ya en el apartado 3, extrae la información RDF asociada a las páginas web visitadas. Si las páginas web no contienen información RDF, permite generarla a partir del código HTML de la página y almacenarla en el ordenador. Es decir, esta extensión separa los metadatos, del código de formato de la página, y permite almacenarlos en RDF. En consecuencia, el usuario puede extraer información RDF procedente de distintas fuentes, guardarla en su ordenador y navegar uniformemente por ella.

La posibilidad de obtener una vista unificada de los datos provenientes de múltiples fuentes resulta muy útil. Imagine el caso de un usuario que acaba de comprar un automóvil y desea comparar las condiciones que le ofrecen las compañías aseguradoras. Con un navegador ordinario, el usuario tendría que mantener abiertas distintas ventanas, cada una con la página web de una aseguradora. Empleando la extensión *Piggy-Bank*, el usuario verá toda la información relevante en una misma ventana, de una forma



homogénea (no con los diferentes formatos de las páginas web de las aseguradoras), y podrá navegar por ella sin tener que saltar de una ventana a otra.



**Figura 44. La extensión Piggy-Bank del navegador Firefox presenta, en una vista unificada, información procedente de múltiples fuentes. Imagen extraída de la documentación oficial sobre Piggy-Bank**

Ahora que hemos visto las características fundamentales de RDF/RDFS y algunas de sus aplicaciones, podemos apreciar mejor sus ventajas semánticas frente a XML. De un lado, las unidades semánticas de un dominio se expresan naturalmente mediante la estructura (sujeto, propiedad, objeto) de RDF. Esto es, RDF tiene un expresividad casi universal (permite representar casi todo lo que uno desee). El modelo de un dominio, con



sus objetos y relaciones, se representa naturalmente en RDF, sin ambigüedades. Por ejemplo, la relación de la figura 28 (usuario compra producto) se representa en RDF de una sola manera; con XML había distintas posibilidades.

De otro lado, mientras que una expresión XML no tiene semántica propia (unas etiquetas anidadas, p. ej., pueden interpretarse como un relación *tipoDe*, *parteDe* o *subclaseDe*), una expresión RDF/RDFS tiene una semántica propia (verbigracia, el significado de la relación *subClassOf* se encuentra perfectamente definido, con independencia de cualquier analizador o procesador de las expresiones RDF/RDFS). Así como la semántica de un documento XML la proporciona la aplicación que lo procesa (porque alguien se la ha programado: p. ej., cuando se recibe una etiqueta *<Producto>* hay que transformar la cadena de texto en un *double* y sumarle el 16% de IVA...) o la persona que lo lee, la semántica de un documento RDF/RDFS ya va insertada en el documento, independientemente de cualquier aplicación que lo procese. Es más: un procesador de RDF/RDFS que no respete la semántica del lenguaje será un procesador mal programado.

### 6.3 Nadie es perfecto: desventajas de RDFS

Habiendo llegado hasta aquí, deberíamos preguntarnos si la Web semántica no sería posible sólo con XML y RDFS (considero que RDF está incluido en RDFS): XML proporcionaría la interoperabilidad sintáctica; RDFS, la interoperabilidad semántica. La respuesta es no. RDFS tiene muchas carencias:

- No se pueden declarar restricciones de rango (*rdfs:range*) que sean válidas sólo para algunas clases. *rdfs:range* define el rango de una propiedad para **todas las clases**. Así, RDFS no permite expresar que los televisores sólo funcionan con corriente alterna, mientras que otros aparatos pueden funcionar con corriente alterna o continua. Veamos un ejemplo más zoológico: en RDFS no se puede especificar que el rango de *tieneCria* es *Liebre* cuando se aplica a liebres, y *Tigre* cuando se aplica a tigres.
- No se pueden representar algunas características de las propiedades. En concreto, no se puede declarar que una propiedad es transitiva (como *menorQue*), simétrica (como *tocaA*), inversa (como *raizCuadradaDe* y *CuadradoDe*) o única (como *AbueloMaternoDe*).
- No se puede reflejar que unas determinadas clases son disjuntas. Por ejemplo, la clase *Persona* tiene asociada dos clases disjuntas (*Hombre* y *Mujer*). En RDFS, puede declararse que *Hombre* y *Mujer* son subclases de *Persona*, pero no que son disjuntas. En otras palabras: resulta imposible especificar que ninguna persona puede ser a la vez hombre y mujer).
- No permite expresar restricciones de cardinalidad. Por lo tanto, una propiedad no está restringida en cuanto al número de valores que puede tomar. Así, no se puede expresar que una cuenta bancaria no puede tener más de seis titulares o que un hijo siempre tiene un padre y una madre.



- Existen algunas expresiones cuya semántica no es estándar (es decir, no pueden expresarse mediante la lógica de primer orden). Esta característica es sumamente indeseable, pues causa que haya sentencias indecibles (sentencias de las que, dado un sistema de axiomas o premisas, no se puede afirmar o negar nada).

En suma, RDFS no es lo bastante completo para describir los recursos de la Web con el detalle que se precisa. Bien: ¿por qué se utiliza? Porque es tan general que puede emplearse en muchos dominios y porque sirve como “puente” entre los vocabularios de cada uno.



## 7. Ontologías

### 7.1 Introducción

Como vimos en el apartado anterior, RDFS no es la mejor solución para describir recursos de una manera compatible con los objetivos de la Web semántica. Para conseguirlos, hay que recurrir a lenguajes de descripción del conocimiento más avanzados. Estos lenguajes son lenguajes formales de ontologías.

La descripción de un dominio de interés (esto es, de sus conceptos y de las relaciones entre ellos) se llama modelo conceptual del dominio. En el campo de la informática, los modelos conceptuales deben transformarse en una forma que pueda almacenarse en la memoria de los ordenadores y que permita aplicar algoritmos.

El propósito de las ontologías, que provienen del campo de la Inteligencia Artificial, estriba en proporcionar modelos conceptuales con las características descritas en el párrafo anterior. La definición que el Diccionario de la Real Academia Española (DRAE) da al término *Ontología* es la convencional: “Parte de la metafísica que trata del ser en general y de sus propiedades trascendentales”. Dicho de otra manera: la Ontología estudia las categorías fundamentales del ser (*ser* debe entenderse en el sentido de *cualidad de la existencia*, no en el de *un ser*). En informática, una *ontología* (nótese la minúscula inicial) designa la descripción de un dominio mediante un vocabulario común de representación. **Alto:** no crea que las ontologías son abstracciones para distraer a los programadores con veleidades filosóficas, inservibles para quienes tienen que llegar a fin de mes. En realidad, todos los seres humanos usamos ontologías:

La Ontología es la primera ciencia. La Ontología implica descubrir categorías e incluir objetos en ellas de maneras que tengan sentido. Cuando Aristóteles miró alrededor del Mundo Antiguo, vio que era valioso empezar a categorizar sus partes. Dividió el mundo según sus elementos y procesos constituyentes, de los primeros intentos de clasificación de animales y plantas a la Física y la Metafísica. Desde Aristóteles, esta tarea se ha dividido entre varias ciencias discretas. La mayoría de los científicos no piensan en lo que hacen como una ontología; pero, en muchas maneras, es exactamente lo que comenzó Aristóteles hace dos mil años. Este trabajo continúa no sólo en las ciencias “duras”, sino también en las ciencias sociales. Es algo que hacemos todos nosotros, cada día.

Cuando hacemos una lista de cosas que hacer, o de los discos y libros que más queremos comprar, o de vídeos que intentamos alquilar, estamos categorizando: estamos dedicándonos a una rudimentaria ontología. Concediendo prioridad a los elementos de una lista, asignamos relaciones entre varias cosas. La ontología puede ser relativamente simple o puede ser bastante compleja.

**[Center for Commercial Ontolog. Prospectus,**  
<http://www.acsu.buffalo.edu/~koepsell/center.htm>]

Toda ontología modela, mediante conceptos y relaciones, lo que conocemos sobre un dominio o un área de conocimiento. A menudo, las ontologías se representan mediante clases, propiedades y atributos de las clases, relaciones entre clases y restricciones sobre los atributos y propiedades de las clases (por ejemplo, una bicicleta tiene dos ruedas, pero no una o tres).



Veamos otra definición de *ontología*:

Especificación de la conceptualización de un dominio del conocimiento [nota mía: una conceptualización es una visión, abstracta y simplificada, del dominio que se quiere representar]. Una ontología es un vocabulario controlado que describe de manera formal objetos y las relaciones entre ellos, y que tiene una gramática para usar los términos del vocabulario con el fin de expresar algo con significado dentro de un dominio de interés específico. El vocabulario se usa para hacer búsquedas y aserciones. Los compromisos ontológicos son acuerdos para usar el vocabulario de una manera consistente para compartir conocimiento. Las ontologías pueden incluir glosarios, taxonomías y diccionarios, pero normalmente tienen más expresividad y reglas más estrictas que estas herramientas. Una ontología formal es un vocabulario controlado que se expresa en un lenguaje de representación de ontologías.

[<http://members.optusnet.com.au/~webindexing/Webbook2Ed/glossary.htm>,  
octubre de 2004]

Aunque la definición anterior es la más común, debo señalar que se refiere a ontologías formales (aquellas que pueden ser usadas por las máquinas). No todas las ontologías deben ser formales: hay ontologías que se expresan con una forma restringida y estructurada del lenguaje natural, e incluso mediante el lenguaje natural (español, francés, inglés...).

En lo que sigue consideraré sólo ontologías formales, que tienen semánticas formales. Es decir, semánticas que describen el significado del conocimiento de una forma precisa. La palabra *precisa* significa aquí que la semántica no se refiere a opiniones ni intuiciones, y que las máquinas y las personas deben interpretarla de una misma forma. En definitiva, una semántica formal usa la lógica de primer orden o un subconjunto de ella (como las lógicas descriptivas). Disponer de una semántica formal resulta indispensable para implementar sistemas de inferencia o de razonamiento automático (esto es, sin intervención humana). Las utilidades del razonamiento automático son varias:

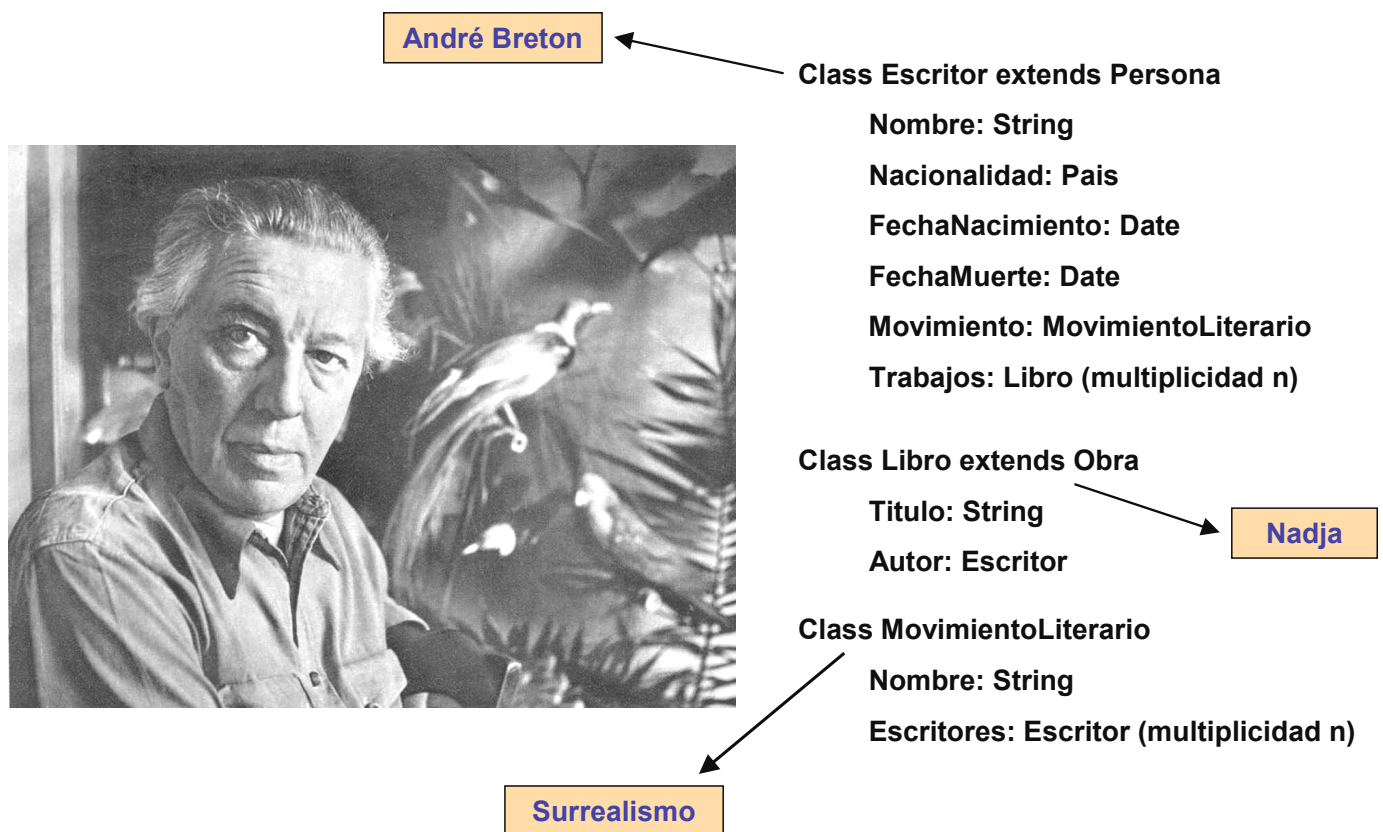
- Se puede comprobar automáticamente si una ontología es consistente con el conocimiento del dominio de interés al cual está asociada.
- Se puede comprobar automáticamente si las relaciones entre las clases corresponden a los propósitos de la ontología y se pueden detectar relaciones espurias.
- Permite clasificar automáticamente las instancias en clases.



Las ontologías son una herramienta para compartir información y conocimiento, es decir, para conseguir la interoperabilidad. Al definir un vocabulario formal de los conceptos del dominio y un conjunto de relaciones entre ellos, permiten que las aplicaciones *comprendan* la información (uso *comprender* en el sentido explicado en el apartado 4). Una DTD se puede considerar como una paupérrima ontología, compuesta sólo por términos y no por relaciones. También un esquema XML se puede considerar como una ontología muy pobre (como ya se explicó en el apartado anterior, la semántica de los esquemas resulta muy rudimentaria para describir el mundo).

¿Qué aspecto tienen las ontologías? Por lo general, toman la forma de una jerarquía de términos que representan los conceptos básicos de un determinado dominio.

## Ejemplo de ontología (1)



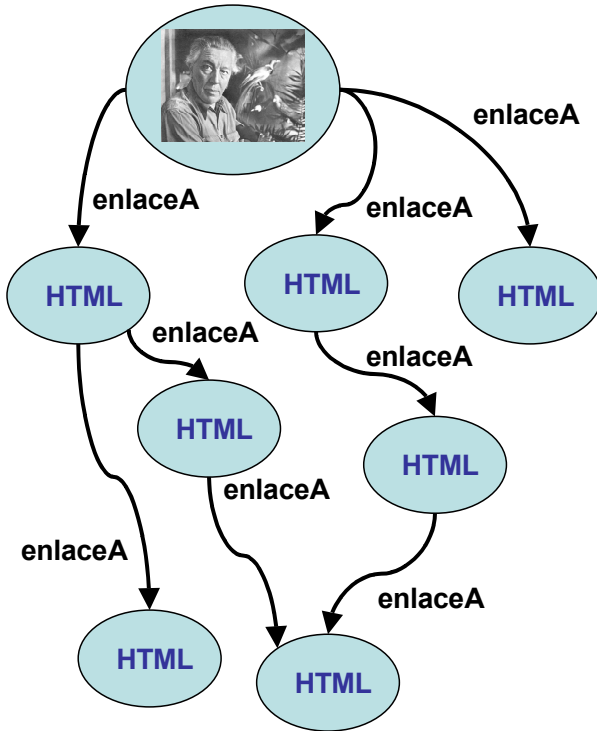
Miguel Ángel Abián, julio de 2005

**Figuro 45. Una ontología de escritores. La ontología mostrada en la figura define las clases Escritor, Libro y MovimientoLiterario. Las dos primeras son subclases de otras dos (Persona y Obra) que, por motivos de espacio, no aparecen aquí. André Breton, Nadja y Surrealismo son instancias de las clases de la ontología**

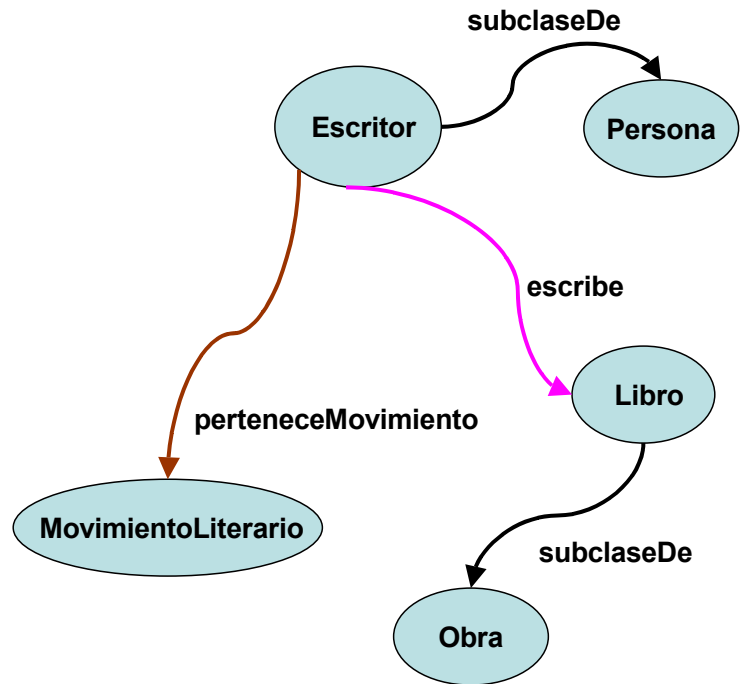


## Ejemplo de ontología (2)

# Red actual



# Red semántica



**Miguel Ángel Abián, julio de 2005**

**Figura 46. Las ontologías sirven para establecer redes semánticas**

En el entorno de la Web, lo usual es que las ontologías se representen en formato XML. Este hecho no debe vincularlas indisolublemente con XML, pues una ontología representa conocimiento; no es un formato de mensajes. A diferencia de las ontologías, XML carece de una semántica que permita el razonamiento automático. Por ejemplo, una descripción XML de André Breton y de los libros que escribió no permitiría inferir conclusiones tan sencillas como que André Breton, por ser *Escritor*, debe ser una *Persona*.



## 7.2 Lenguajes de representación de ontologías y herramientas

Hay muchos lenguajes que se usan para representar las ontologías o, por mejor decir, el conocimiento. En el entorno de la Web, los más conocidos son RDFS, OWL y DAML+OIL. Como DAML+OIL es un lenguaje bastante similar a OWL y está siendo superado por él, no lo consideraré aquí. De todos modos, el lector interesado en DAML+OIL puede consultar <http://www.daml.org/language/features.html>.

**RDFS** se vio en el apartado anterior, donde también se dio fe de sus limitaciones. Si bien RDF es un modelo de representación de metadatos, puede usarse como lenguaje general para la representación del conocimiento. Como ejemplo de ontología representada en RDF/XML (la sintaxis XML para RDF), incluyo una ontología sobre derechos de acceso a los recursos de la Web (está extraída de <http://www.w3.org/2001/Talks/0710-ep-grid/slide21-0.html>).

```
<rdf:RDF
```

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

La ontología se describe mediante sentencias RDFS (toda sentencia RDFS válida es una sentencia RDF).

```
<rdf:Description about="http://www.w3.org/2001/02/acls/ns#">
```

```
<rdfs:comment>A namespace for describing Access Control Lists</rdfs:comment>
```

```
<rdfs:comment>$Revision: 1.3 $ $Date: 2001/07/11 00:11:43 $</rdfs:comment>
```

```
<rdfs:seeAlso resource="http://www.w3.org/2001/02/acls/acls-for-ns"/>
```

```
</rdf:Description>
```

Define una nueva clase. En la ontología, nos referiremos a ella como *ResourceAccessRule*.

```
<rdfs:Class ID="ResourceAccessRule">
```

```
<rdfs:label xml:lang="en">Access Rule</rdfs:label>
```

```
<rdfs:comment>An assertion of access privileges to a resource.</rdfs:comment>
```

```
<rdfs:isDefinedBy resource="http://www.w3.org/2001/02/acls/ns#" />
```

```
</rdfs:Class>
```

```
<rdf:Class ID="Identity">
```

```
<rdfs:label xml:lang="en">Identity</rdfs:label>
```

```
<rdfs:comment>Any entity to which access may be granted to a resource.</rdfs:comment>
```

```
<rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource" />
```

```
</rdf:Description>
```

Subclase de *Resource*.

```
<rdf:Class ID="Principle">
```

```
<rdfs:label xml:lang="en">Principle</rdfs:label>
```

```
<rdfs:comment>An Identity to which credentials or other uniquely distinguishing characteristics may be assigned.</rdfs:comment>
```

```
<rdfs:subClassOf rdf:resource="#Identity" />
```

```
</rdf:Description>
```

Subclase de *Identity*.

```
<rdf:Class ID="Group">
```

```
<rdfs:label xml:lang="en">Group</rdfs:label>
```

```
<rdfs:comment>Collection of Principles.</rdfs:comment>
```

```
<rdfs:subClassOf rdf:resource="#Identity" />
```

Subclase de *Identity*.



```
</rdf:Description>
<rdf:Property ID="accessor">
  <rdfs:label xml:lang="en">accessor</rdfs:label>
  <rdfs:comment>The resource identifying an entity (for instance, a user) to whom access privileges
have been granted.</rdfs:comment>
  <rdfs:range rdf:resource="#Identity"/>
  <rdfs:domain rdf:resource="#ResourceAccessRule"/>
  <rdfs:isDefinedBy resource="http://www.w3.org/2001/02/acls/ns#" />
</rdf:Property>
```

Propiedad con rango *Identity* y dominio *Resource*.

```
<rdf:Property ID="access">
  <rdfs:label xml:lang="en">access</rdfs:label>
  <rdfs:comment>The access privileges extended to an accessor.</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
  <rdfs:domain rdf:resource="#ResourceAccessRule"/>
  <rdfs:isDefinedBy resource="http://www.w3.org/2001/02/acls/ns#" />
</rdf:Property>
```

Propiedad con rango *Literal* y dominio *ResourceAccessRule*.

```
<rdf:Property ID="hasAccessTo">
  <rdfs:label xml:lang="en">has access to</rdfs:label>
  <rdfs:comment>Relates an Access Rule to the resources to which the rule applies. The inverse
relation is 'accessedBy'</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
  <rdfs:domain rdf:resource="#ResourceAccessRule"/>
  <rdfs:isDefinedBy resource="http://www.w3.org/2001/02/acls/ns#" />
</rdf:Property>
```

Propiedad con rango *Resource* y dominio *ResourceAccessRule*.

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

La clase *Identity* se define como una subclase de la clase RDFS *Resource*, y *Principle* se define como subclase de *Identity*. Por lo tanto, la ontología anterior define una jerarquía de clases así:

Resource  
  Identity  
    Principle

**OWL** (*Web Ontology Language* [sic]) es un lenguaje desarrollado por W3C. Es una extensión de RDFS y emplea el modelo de tripletas de RDF. En cierto modo, es un RDFS mejorado, que mantiene una buena relación entre eficacia computacional y poder expresivo. Entre las propiedades que incluye para restringir las instancias de una clase, se destacan *disjointWith* (expresa la exclusividad mutua de las clases), *unionOf* (expresa uniones de clases) y *oneOf* (enumera todas las instancias de una clase). Para restringir los valores de las propiedades, se puede usar *rdfs:domain* y *rdfs:range* (proviene de RDFS), *TransitiveProperty* (indica que una propiedad es transitiva), *inverseOf* (indica que una propiedad es inversa de otra), *minCardinality* (especifica el número mínimo de elementos que participan en una relación), *maxCardinality* (especifica el número máximo de elementos que participan en una relación).



Como puede verse, OWL tiene mucha más capacidad expresiva que RDFS. Además, OWL facilita la importación y exportación de clases: incluye propiedades como *sameAs*, *equivalentClass*, *equivalentProperty*, *differentFrom*, etc. Por ejemplo, *equivalentClass* permite establecer que la clase *onto1:Persona* (de la ontología *onto1*) y la clase *onto2:SerHumano* (de otra ontología, llamada *onto2*) son equivalentes; esto es, que cada instancia de una clase es también instancia de la otra clase, y viceversa. La capacidad de expresar que dos entidades son iguales resulta muy útil cuando se integran o se mezclan ontologías y permite la interoperabilidad.

OWL tiene una sintaxis abstracta (independiente de cualquier representación computacional), una basada en XML y otra basada en RDF. La siguiente ontología la expreso mediante la sintaxis abstracta de OWL, con el fin de no vincular las ontologías con XML. La ontología, con algunos cambios y traducida, se ha extraído de <http://www.cs.man.ac.uk/~horrocks/ISWC2003/Tutorial/>.

Ontology(  
 Class(pp:planta partial)  
 Class(pp:hierba partial pp:planta)  
 Class(pp:arbol partial pp:planta)  
 Class(pp:macho partial)  
 Class(pp:hembra partial)  
 Class(pp:joven partial)  
 Class(pp:adulto partial)  
 Class(pp:anciano partial pp:adulto)  
 Class(pp:mascota complete restriction(pp:es\_mascota\_de someValuesFrom(owl:Thing)))  
 Class(pp:animal partial restriction(pp:come someValuesFrom(owl:Thing)))  
 Class(pp:gato partial pp:animal)  
 Class(pp:perro partial pp:animal)  
 Class(pp:oveja partial pp:animal restriction(pp:come allValuesFrom pp:hierba))  
 Class(pp:perro partial pp:animal)

Definición OWL de la clase *planta*.

Definición OWL de que *hierba* es una subclase de *planta*. OWL permite expresar lo mismo con *SubClassOf*.

Definición OWL de que *anciano* es una subclase de *adulto*. OWL permite expresar lo mismo con *SubClassOf*.

La clase *mascota* sólo puede tomar valores de la clase OWL *Thing* cuando participa en la relación *es\_mascota\_de*.

Expresión OWL de que una *oveja* sólo *come* (relación) *hierba*.



```
Class(pp:persona partial pp:animal)
Class(pp:hombre complete intersectionOf(pp:persona pp:macho pp:adulto))
```

La clase *hombre* es la intersección de las clases *persona*, *macho* y *adulto*.

```
Class(pp:mujer complete intersectionOf(pp:hembra pp:persona pp:adulto))
Class(pp:senyora+anciana complete intersectionOf(pp:anciano pp:hembra pp:persona))
Class(pp:senyora+anciana partial
  intersectionOf(restriction(pp:tiene_mascota allValuesFrom(pp:gato))
    restriction(pp:tiene_mascota someValuesFrom(pp:animal))))
```

La clase *senyoraanciana* es la intersección de las clases *persona*, *hembra* y *anciano*. Además, para la clase *senyoraanciana*, la relación *tiene\_mascota* sólo puede tomar valores de la clase *gato*. En una palabra: una señora anciana sólo puede tener gatos como mascotas.

Un amante de los animales tiene como mínimo tres mascotas.

```
Class(pp:amante+animales complete
  intersectionOf(pp:persona restriction(pp:tiene_mascota minCardinality(3))))
Class(pp:propietario+mascota complete
  intersectionOf(restriction(pp:tiene_mascota someValuesFrom(pp:animal)) pp:persona))
Class(pp:propietario+gato complete
  intersectionOf(pp:persona restriction(pp:tiene_mascota someValuesFrom(pp:gato))))
```

```
DisjointClasses(pp:perro pp:gato)
DisjointClasses(pp:joven pp:adulto)
```

La propiedad *come* es la inversa de *comido\_por*.

Una instancia no puede pertenecer a la clase *gato* y a la clase *perro*.

```
ObjectProperty(pp:comido_por)
ObjectProperty(pp:come inverseOf(pp:comido_por) domain(pp:animal))
ObjectProperty(pp:tiene_mascota domain(pp:persona) range(pp:animal))
ObjectProperty(pp:es_mascota_de inverseOf(pp:tiene_mascota))
```

```
SubPropertyOf(pp:tiene_mascota pp:gusta_a)
```

Las personas tienen animales como mascotas.

```
Individual(pp:Miguel type(owl:persona))
Individual(pp:Dolly type(pp:oveja))
Individual(pp:Fido type(pp:dog) value(pp:es_mascota_de pp:Luis))
```

Sentencias sobre individuos: Miguel es una persona, Dolly es una oveja, Fido es un perro y es mascota de Luis.



```
Individual(pp:Consolacion type(pp:anciano) type(pp:hembra)
value(pp:tiene_mascota pp:Sonrisas))
```

)

La señora Consolación es una anciana que tiene una mascota llamada Sonrisas.

La ontología anterior, pese a su simplicidad, ya permite realizar algunos razonamientos automáticos. De la última sentencia, p. ej., un programa deduciría que Sonrisas es un gato, pues Consolación es una persona (los propietarios de mascotas son personas) y, por tanto, es una señora anciana (pues es persona, mujer y anciana). Como todas las mascotas de las señoras ancianas son gatos, Sonrisas debe ser un minino.

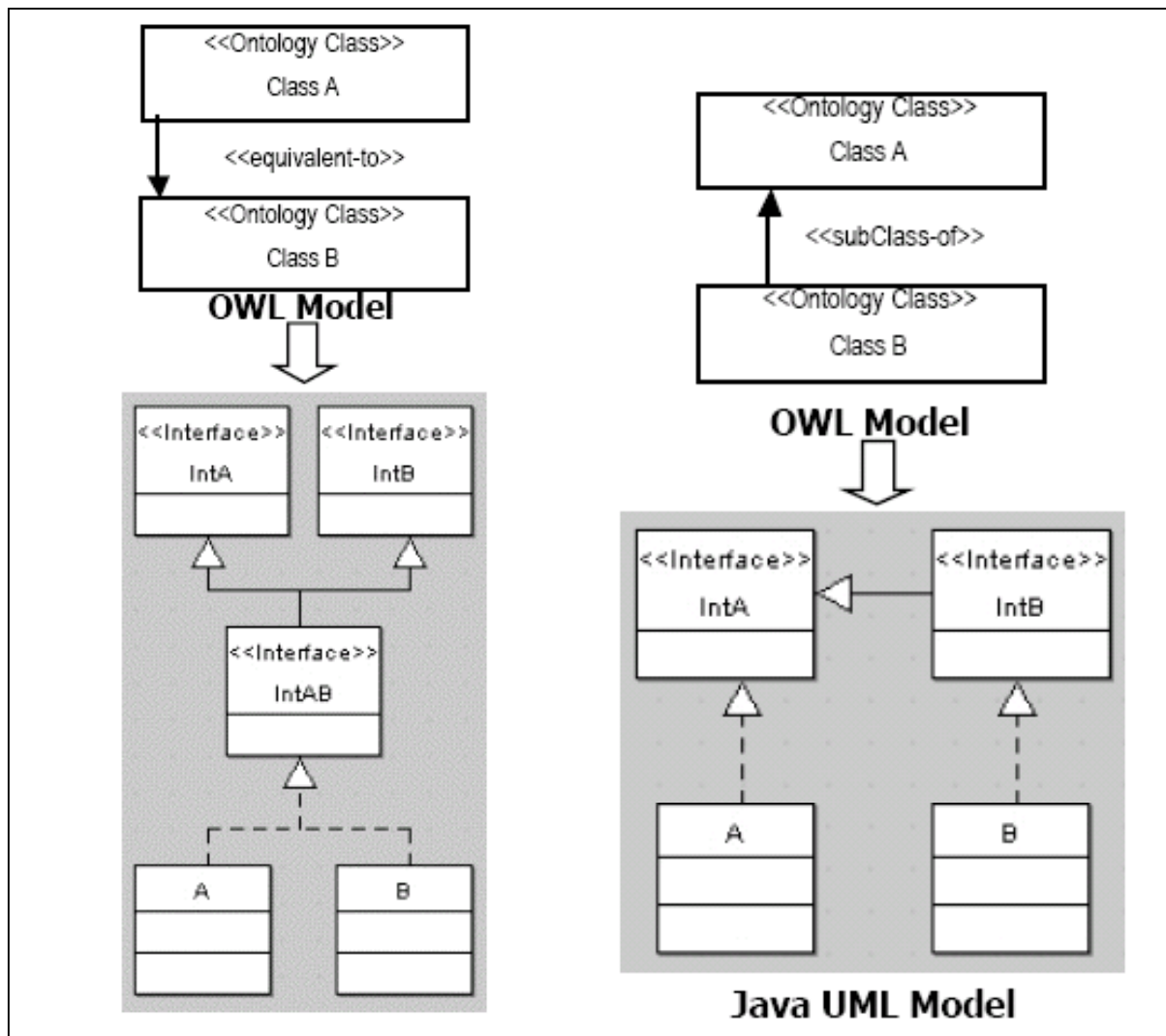
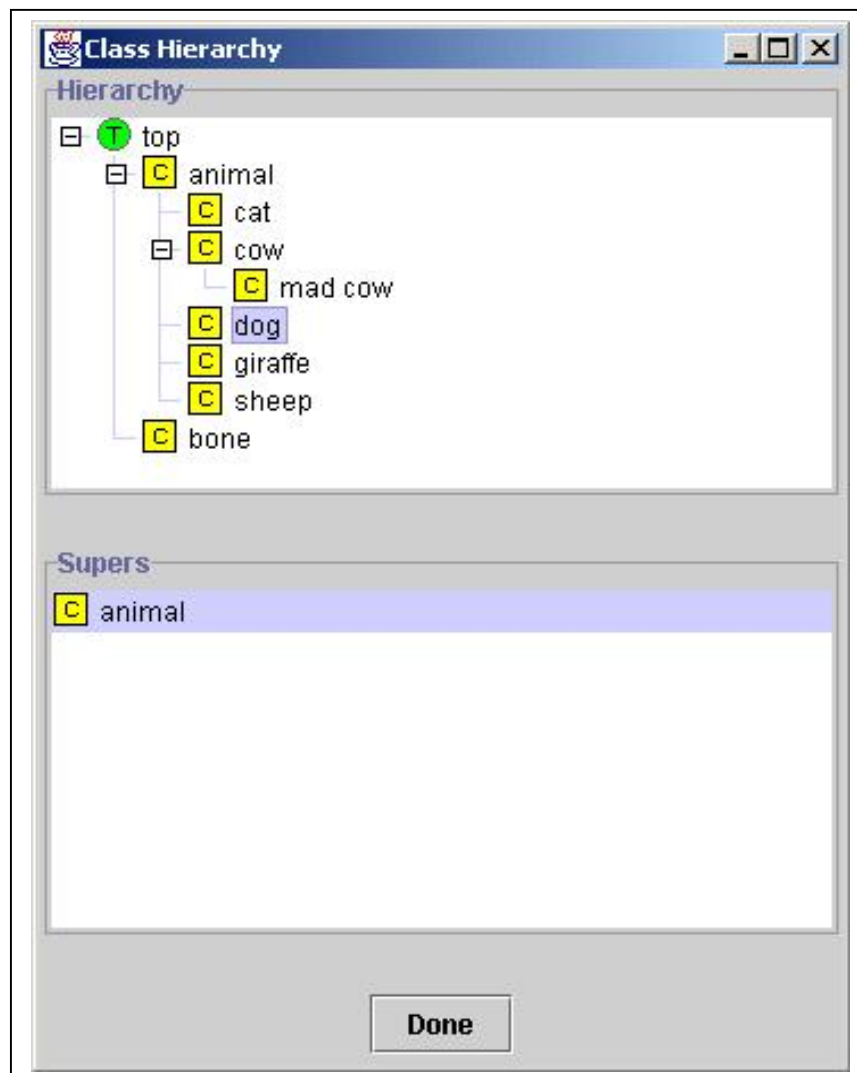


Figura 47. Traducción de las clases de OWL al modelo UML de Java



En <http://onto.stanford.edu:8080/wino/index.jsp> puede hallarse un agente que recomienda vinos a partir de una base de conocimiento, compuesta por una ontología de vinos y comidas y por un sistema de razonamiento automático. Si quiere saber cómo funcionarán los agentes del futuro, le recomiendo que eche una ojeada a este agente de vinos, pues muestra perfectamente las tres partes en que se descompone el funcionamiento del agente: consultar la ontología, realizar búsquedas y dar formato a los resultados.

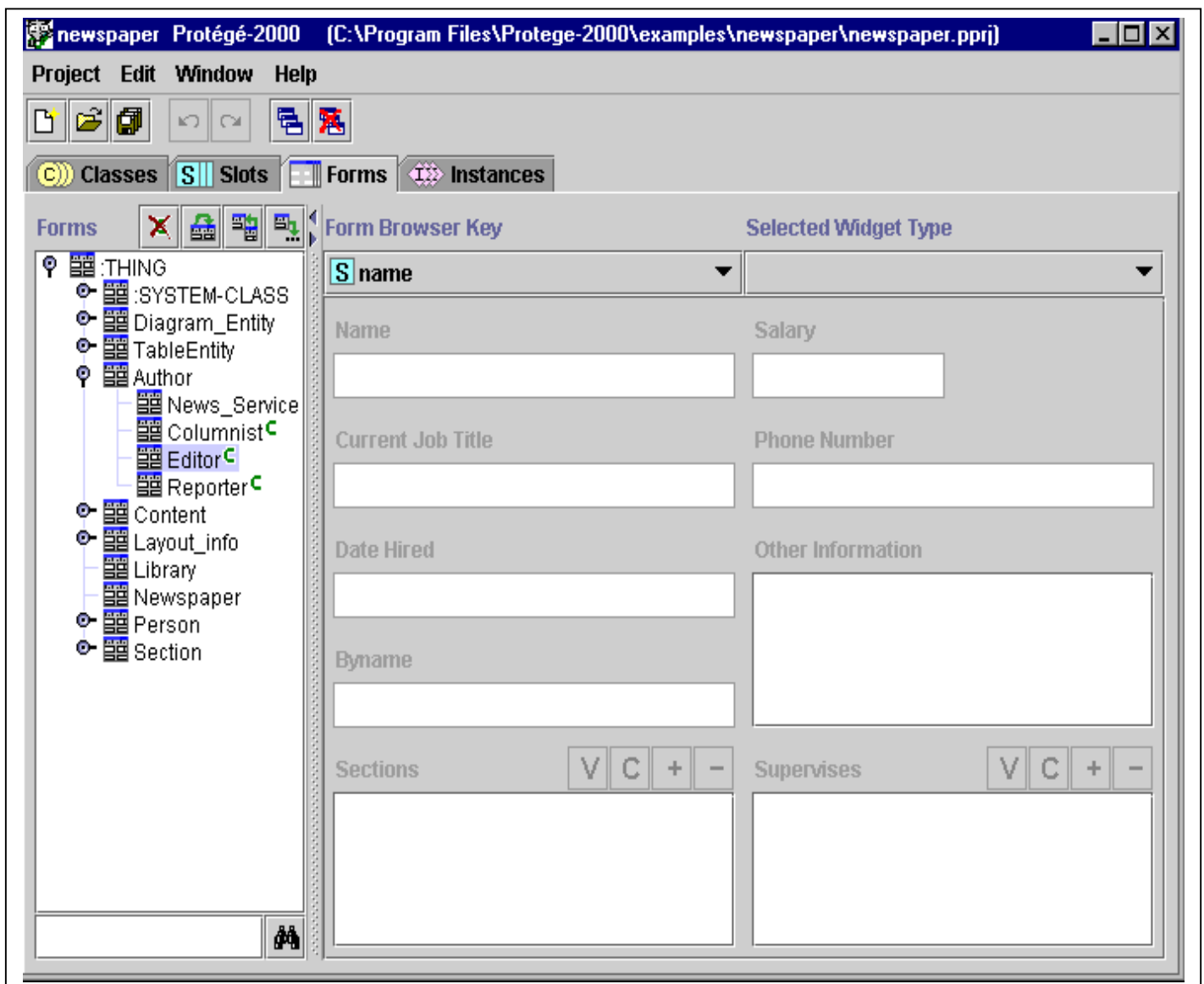
Existen varios editores de ontologías gratuitos. Entre ellos, destacan Protégé (<http://protege.stanford.edu/>), Kaon (<http://kaon.semanticweb.org/>), OILed (<http://oiled.man.ac.uk>) y ORIENT (<http://www.alphaworks.ibm.com/tech/semanticstk>). Exceptuando el penúltimo, todos se basan en Java, lo cual da cuenta del interés que Java sigue despertando entre los desarrolladores de tecnologías avanzadas. ORIENT es una extensión (*plug-in*) de Eclipse.



**Figura 48. Edición de una ontología de animales con Kaon**  
(<http://kaon.semanticweb.org/>)

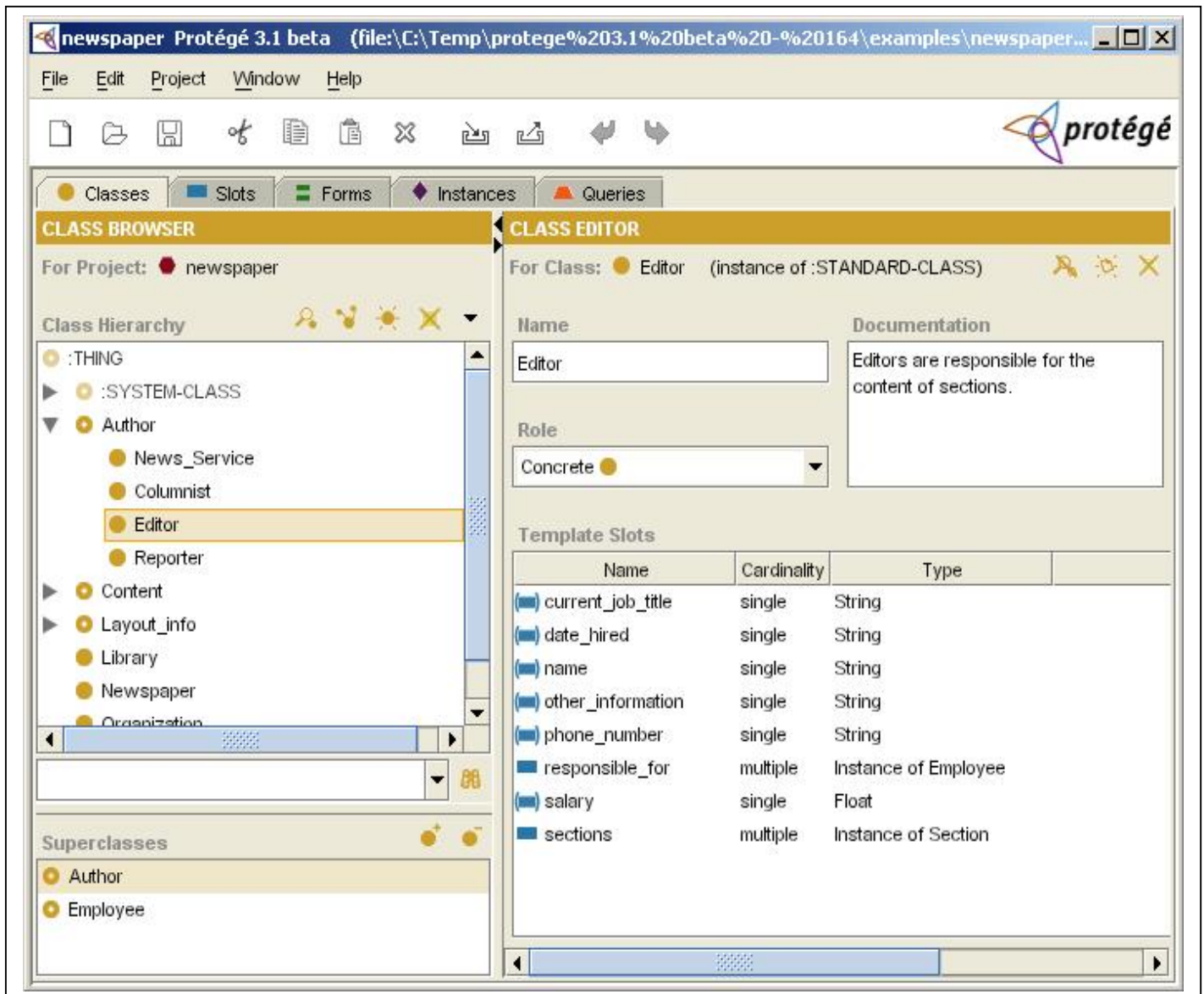


Después de evaluar exhaustivamente las anteriores herramientas de edición de ontologías y algunas más, he llegado a la conclusión de que la más recomendable es **Protégé** (<http://protege.stanford.edu/>). Es un editor de ontologías escrito en Java, gratuito y de código abierto. Tras él hay una gran comunidad de desarrolladores y de usuarios universitarios, empresariales y gubernamentales. Actualmente, Protégé permite trabajar con RDFS (las ontologías se pueden exportar a RDFS) y dispone de una extensión para OWL. Su sencillez y su buena documentación lo hacen ideal para los neófitos en ontologías.



**Figura 49. Protégé en acción. Se está usando una ontología que describe un periódico. En la imagen se muestra el formulario que permite introducir instancias de las clases. Captura de pantalla extraída de la documentación oficial de Protégé**





**Figura 50. Protégé en acción. Se está usando una ontología que describe un periódico. En la imagen se muestra tanto el navegador de clases como el editor de clases. Captura de pantalla extraída de la documentación oficial de Protégé**

JENA (<http://jena.sourceforge.net/>) es un *framework* desarrollado por los laboratorios de HP. Su fin es manipular metadatos desde aplicaciones escritas en Java. En su versión 2, JENA incluye un analizador sintáctico de RDF, una API para RDF, una API de ontologías que permite trabajar con OWL, DAML y RDFS, un subsistema de razonamiento y un sistema de persistencia. Por si fuera poco, permite trabajar con el lenguaje de consultas de RDF (RDQL).

Consideremos el siguiente código en OWL (corresponde a una ontología de cámaras de fotografía y se ha extraído de la documentación oficial de JENA):



```
<owl:Class rdf:ID="SLR">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Camara"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#Visor"/>
      <owl:hasValue rdf:resource="#ATravesDeLaLente"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

SLR (*Single Lens Reflex*) es una cámara cuya lente se usa tanto para enfocar como para tomar la fotografía.

El código Java que generaría JENA a partir del anterior código en OWL sería éste:

```
// Se crea la instancia ATravesDeLaLente
OntClass Window = m.createClass(camNS + "Window");
Individual throughTheLens = m.createIndividual(camNS + "ATravesDeLaLente", Window);

// Se crea la propiedad visor
ObjectProperty visor = m.createObjectProperty(camNS + "visor");

// Se crea la restricción tieneValor
RestriccionTieneValor mirarATravesDeLaLente =
    m.createHasValueRestriction(null, visor, ATravesDeLaLente);

// Se crea la clase Camara
OntClass Camara = m.createClass(camNS + "Camara");

// Se crea la intersección para definir la clase SLR (un tipo especial de cámara)
IntersectionClass SLR = m.createIntersectionClass(camNS + "SLR",
    m.createList(new RDFNode[] { mirarATravesDeLaLente, Camara }));
```

La manera como se carga un modelo RDF desde Java se ilustra con este ejemplo, extraído también de la documentación de JENA:

```
String personURI = "http://enalgunaparte/JuanPerez";
String givenName = "Juan";
String familyName = "Perez";
String fullName = givenName + " " + familyName;

Model model = ModelFactory.createDefaultModel();
Resource juanPerez = model.createResource(personURI);

johnSmith.addProperty(VCARD.FN, fullName);
johnSmith.addProperty(VCARD.N, model.createResource()
    .addProperty(VCARD.Given, givenName)
    .addProperty(VCARD.Family, familyName));
```



Una vez cargado el modelo en la memoria, la información puede recuperarse como se muestra en este ejemplo:

```
// Se recupera el recurso Juan Pérez
String johnSmithURI = "http://enalgunaparte/JuanPerez";
Resource jPerez = model.getResource(juanPerezURI);

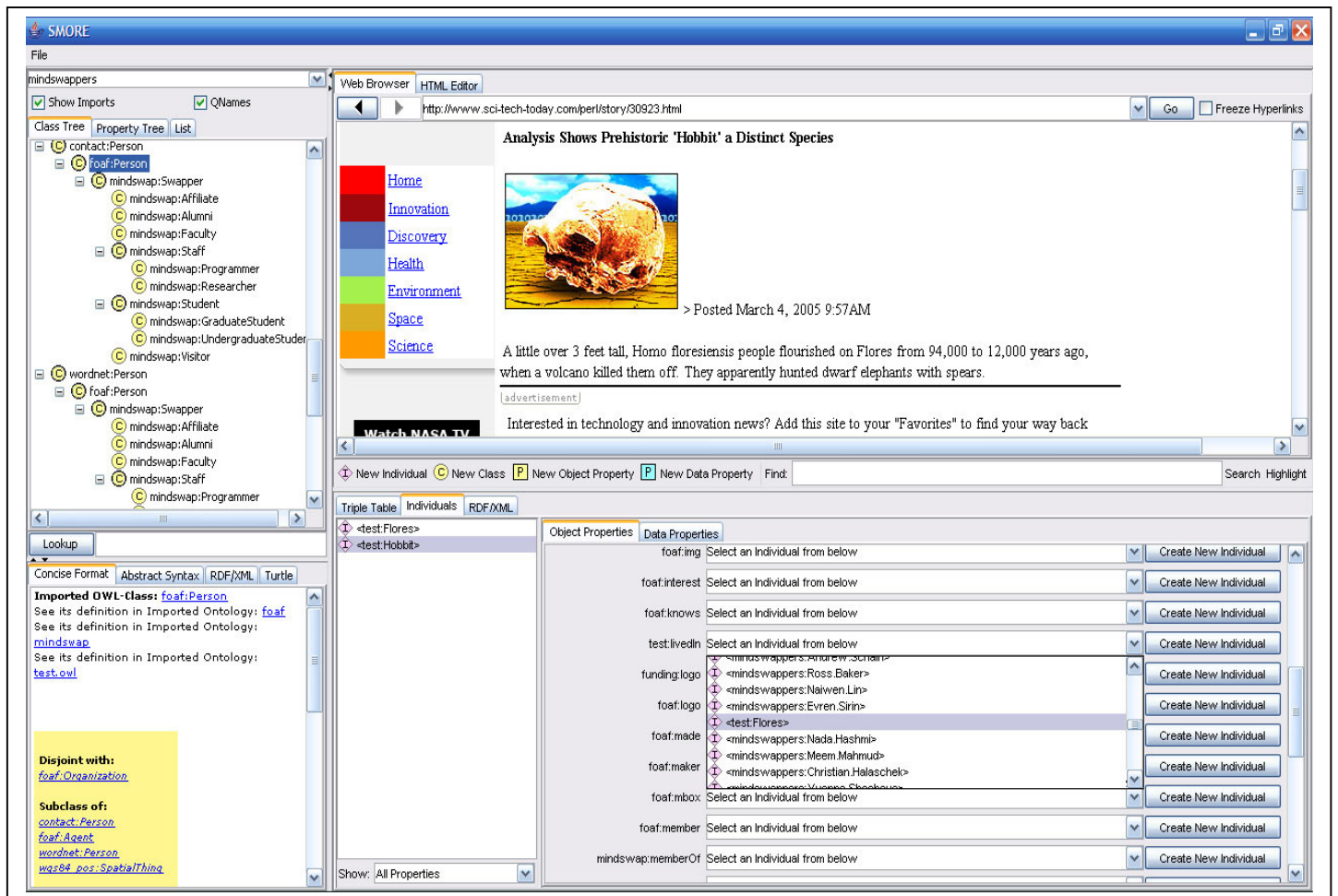
// Se recupera el valor de la propiedad N
Resource name = (Resource) jPerez.getProperty(VCARD.N).getObject();

// Se recupera el valor de la propiedad FN
String fullName = (String) jPerez.getProperty(VCARD.FN).getObject();
```

*VCARD* es una clase que representa una tarjeta para negocios electrónicos. El modelo de datos representa la tarjeta de Juan Pérez. La propiedad *VCARD.N* designa el nombre del usuario de la tarjeta; *VCARD.FN* designa el nombre completo (nombre y apellidos).

Java es también el lenguaje con que se ha escrito SMORE (<http://www.mindswap.org/2005/SMORE/>), una herramienta que posibilita incluir anotaciones OWL en documentos HTML. Sus dos principales objetivos son permitir que el usuario maneje clases, propiedades e instancias de las ontologías que ya existen (también permite crear ontologías desde cero) y permitir que éste haga el marcado semántico de sus páginas web al mismo tiempo que las crea.





**Figura 51. En la imagen pueden apreciarse algunas de las utilidades que SMORE incorpora: editor de HTML, de RDF/XML, navegador de clases, etc. Captura de pantalla extraída de la documentación oficial sobre SMORE**



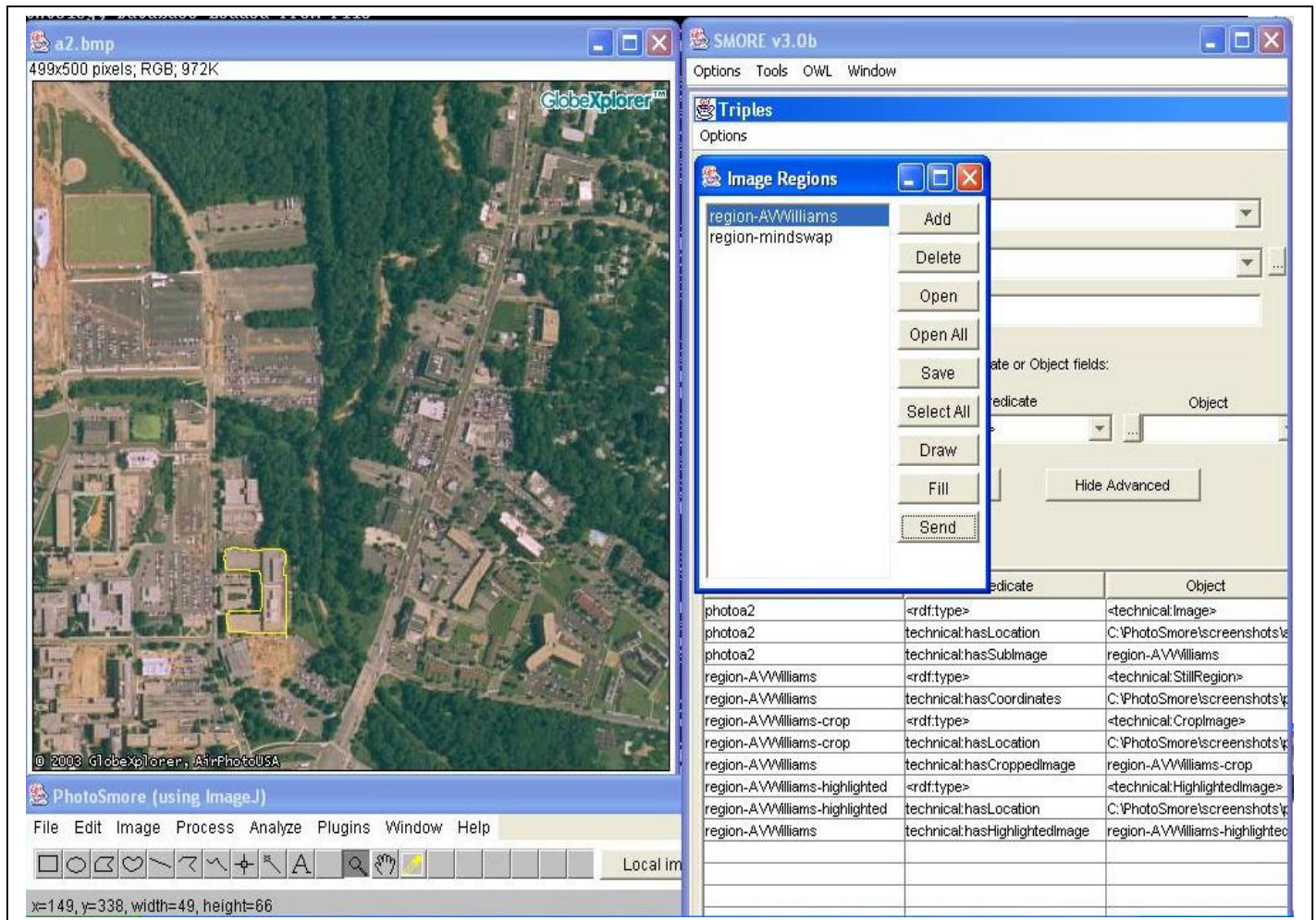


Figura 52. SMORE en acción: haciendo anotaciones semánticas en una página web

## 7.3 Aplicaciones de las ontologías

En primer lugar, las ontologías pueden usarse para mejorar la **búsqueda de información** en la Web y en las intranets de las organizaciones, así como para **navegar** por ellas. Si se definieran una o más ontologías para cada dominio, los contenidos web podrían describirse en función de los términos ontológicos. De este modo, se podrían expandir las búsquedas mediante términos de las categorías más específicas de la ontología. Las ontologías evitarían así las ambigüedades en las búsquedas, pues el usuario podría subir uno o más niveles en la jerarquía de clases para establecer el sentido del término que usa en la búsqueda. Por ejemplo, en el ejemplo de las antenas (apdo. 3.1), el buscador web notaría que la palabra “antena” aparece en varias ontologías y preguntaría al usuario si está interesado en aparatos o en animales. En el segundo caso, el buscador podría mostrar una lista de especies animales con antenas, de suerte que el usuario pudiera concretar más su búsqueda. Asimismo, podrían usarse axiomas que definieran los términos en función de otros términos (por ejemplo, *persona* sería lo mismo que ser *humano*), lo que solucionaría el problema de los sinónimos. La facilidad con que las ontologías se integran en sistemas de razonamiento automático permitirá hacer búsquedas como las descritas en anteriores apartados y evitar las dificultades que se expusieron en el subapartado 3.1.



En segundo lugar, las ontologías favorecen la **interoperabilidad**. Las ontologías que permiten especificar cómo un término se relaciona con otros términos (las escritas en OWL, vg.) facilitan la interoperabilidad semántica. Por ejemplo, una ontología podría especificar que *Funcionario* es una *Persona* cuya propiedad *organizacion* toma el valor “Estado”. Cualquier aplicación que comprenda los términos *Persona* y *organizacion* podrá usar *Funcionario*, aunque no entienda (esto es, aunque en su ontología no exista) el término *Funcionario*.

En tercer lugar, las ontologías pueden usarse para comprobar la **validez de los datos**. Por ejemplo, la anterior ontología de plantas y animales podría usarse para comprobar si ciertas afirmaciones son válidas o no. En ella, afirmaciones como “El perro Fido tiene una mascota llamada Miau” serían falsas, pues sólo las personas tienen mascotas.

Especialmente interesante es el uso de ontologías para la validación de datos procedentes de bases de datos. Por ejemplo, una ontología que establezca que una instancia de la clase *TrabajadorAutonomo* debe estar vinculada a una o más instancias de la clase *ActividadEconomica* podría usarse para comprobar que todos los autónomos registrados en una base de datos tienen al menos una actividad. Del mismo modo, podría usarse para buscar aquellas actividades sin ningún trabajador asociado (incompletitud de los datos).

En cuarto lugar, las ontologías son útiles para organizar las **colecciones de recursos multimedia**. Las ontologías permiten incluir anotaciones semánticas en colecciones de imágenes, audio, vídeos y otros recursos no textuales. Actualmente, esos recursos se indexan mediante metadatos que pueden usarse para buscar mediante palabras clave. El principal problema de esa forma de trabajar estriba en que cada persona u organización escoge sus propias etiquetas, lo cual dificulta encontrar los resultados correctos. Por ejemplo, una búsqueda de imágenes mediante las palabras clave “presidente” y “Adolfo Suárez” no producirá los mismos resultado que una basada en “jefe de Estado” y “Adolfo Suárez”. Con las ontologías, los recursos no textuales se pueden relacionar con información concreta del dominio al que pertenecen, de manera que encontrarlos sea más sencillo y eficaz. Si no he incluido la clasificación de recursos multimedia en el punto anterior es porque la búsqueda de recursos multimedia siempre ha sido mucho más difícil que la de recursos textuales.

En quinto lugar, las ontologías se usarán para programar **agentes inteligentes**, que entenderán e integrarán las informaciones procedentes de distintas fuentes. En el futuro, los servicios web se describirán mediante ontologías. Los agentes las usarán para buscar los servicios web que les interesen y utilizarlos automáticamente, sin intervención humana.

En sexto lugar, las ontologías facilitarán el **comercio electrónico**. En el subapartado 3.1 se mostró que compradores y vendedores sufren un mismo problema: la falta de integración de las heterogéneas descripciones de los productos. A menudo, no existe un consenso sobre cómo describir los productos de un dominio (alimentación, por caso) y cómo catalogarlos; y las diversas descripciones que los vendedores usan para sus productos acrecientan la dificultad de buscar productos y servicios de forma rápida y precisa. Construir catálogos que puedan ser usados por muchos usuarios y organizaciones no es más que construir una ontología para un dominio.



Por si fuera poco, las ontologías dotan a los datos de semántica comprensible para las máquinas y, por tanto, permiten la automatización de muchos procesos. Así, los agentes que usen ontologías podrán buscar productos, negociar compras, localizar servicios de interés para los usuarios. En el caso de que varias empresas usen una misma ontología, sus sistemas de información podrán procesar automáticamente los datos que intercambien, pues compartirán una misma semántica. Incluso en el caso de empresas que usen distintas ontologías, existen herramientas que permiten rápidamente traducir los términos de una ontología a otra o mezclar ontologías, incluso de manera automática o semiautomática. En suma, el uso de ontologías en las empresas permitirá hacer negocios electrónicos con otras empresas, de modo automático y sin que sea necesario ponerse de acuerdo *a priori* sobre la semántica de los datos. Dicho con otras palabras: las ontologías y las tecnologías que las usan permitirán, por un lado, que los compradores localicen productos y servicios con características específicas y, por otro, que los vendedores localicen a posibles compradores que correspondan a ciertos perfiles.

RosettaNet (<http://www.rosettanet.org/>) proporciona un excelente ejemplo de ontología vertical (propia de un dominio, no de varios) para el comercio electrónico. Esta ontología semiformal describe en detalle los productos de las industrias de hardware y software. RosettaNet crea, en primer lugar, definiciones de las propiedades que tienen ciertos productos habituales en el comercio electrónico. Luego distribuye estas propiedades entre las empresas y organizaciones relacionadas con el hardware y el software, con el fin de que enumeren los valores que pueden tener esas propiedades. Finalmente, distribuye las definiciones de las propiedades y sus valores entre las empresas, para que usen esas propiedades y valores como formato de descripción de sus productos. Así, RosettaNet define varios metadatos para la propiedad CPU: *Property Name*, *Synonym*, *Property Definition*, *Dictionary References*, *Where Used*, *Property Type*... Incluso hay metadatos que contienen metadatos. Por caso, *Property Name* consta de los metadatos *Acronym* y *Abbreviation*.

Dos empresas que usan RosettaNet pueden intercambiar automáticamente datos, pues todas las propiedades tienen un significado bien establecido (interoperabilidad semántica) y la organización RosettaNet proporciona una sintaxis XML para los documentos comerciales más habituales (interoperabilidad sintáctica).



## 8. Algunas reflexiones sobre la Web semántica

La Web semántica se acerca ahora más a una realidad que a una visión, pero sería imprudente omitir los problemas que plantea implantarla y las críticas que ha merecido. Para empezar, el paso de la Web actual a la semántica tardará un tiempo. No se pretende que la Web semántica sustituya a la actual, sino que sea una extensión lógica de ésta. La transición de una a otra no será inmediata, pues los millones de páginas de la Web no pueden rellenarse de metadatos de un día para otro. Existen herramientas que permiten, de forma semiautomática, incorporar metadatos a las páginas web, pero no son prácticas cuando se trata con terabytes de información.

En segundo lugar, la creación de ontologías consensuadas dista mucho de ser una tarea rápida. En cualquier área de interés, resulta difícil que los actores se pongan de acuerdo en definir una ontología común. Las ontologías no son verdades inmutables o absolutas: dependen del punto de vista de los humanos que las crean. Por ejemplo, una ontología médica describirá a las personas de una forma completamente distinta a como lo hace una ontología fiscal o filosófica. La frase “Según el cristal con que se mira, la botella está medio llena o medio vacía” es muy apropiada para las ontologías. Cualquier ontología refleja unas concepciones específicas del mundo y del área de interés. No puede haber ontologías universales, pues cada persona o cada comunidad tienen sus intereses. Incluso hay diferencias entre las traducciones de una misma ontología, ya que las palabras pueden alterar su significado al ser traducidas. La palabra “person”, vaya por caso, tiene en inglés un sentido legal del que “persona” carece en español. Una posible solución para permitir la interoperabilidad de las ontologías pasa por definir unas ontologías comunes, útiles para el intercambio de información entre distintas partes, que sean compatibles con las ontologías particulares de cada área de interés.

En tercer lugar, algunos piensan que la Web semántica tardará muchísimo en materializarse; pues la relacionan con la Inteligencia Artificial, campo donde los avances han sido muy lentos. Poco peso tiene esta crítica: ya vimos en el apartado 4 que asociar la Web semántica con la IA resulta incorrecto. El planteamiento de la Web del futuro estriba en que los humanos nos tomemos la molestia de añadir metadatos a los datos, de suerte que las máquinas puedan interpretarlos; no en confiar en que las máquinas puedan asemejarse a las mentes humanas o en que sean capaces de entender los lenguajes naturales.

En cuarto lugar, algunos opinan que la Web semántica requiere un esfuerzo inútil. En otras palabras, creen que no se necesita una nueva Web. Me resulta difícil comprender esta creencia, pues los problemas de la Web actual son bien patentes para cualquiera que busque información. Supongo que, cuando Tim Berners-Lee comenzó a escribir en HTML los nombres y números de teléfono de la gente del CERN, habría mucha gente que le comentaría la inutilidad de su esfuerzo: ¿acaso no se podían buscar los números de teléfono en la guía telefónica o en los listines del CERN? En una de las empresas para las que trabajo, los viejos del lugar recuerdan muy bien su reacción cuando se introdujo el correo electrónico. Exceptuando al informático de la empresa y al gerente, todos pensaban que el correo electrónico era bien inútil: ¿para qué iban a enviar un correo a un compañero, cuando podían dejarle una nota sobre la mesa? (Quizá como venganza tardía, el informático conserva, enmarcada en la pared de su despacho, una nota de las que usaban antes del correo electrónico.) Moraleja: mucha gente no percibe



las ventajas de las nuevas tecnologías hasta que no las han probado, aunque las desventajas de las existentes resulten incuestionables.

Por último, existe la percepción de que la Web semántica resultará demasiado compleja (muchas tecnologías distintas, muchos acrónimos). Algunos críticos incluso la han denominado “Pedantic Web” (el chiste es intraducible al español). No negaré que la proliferación de nuevos conceptos y lenguajes desconcierta al principio, mas creo que la pregunta que debemos hacernos es sencilla: ¿son necesarios? En la Web semántica, serán las máquinas las que razonen a partir de los datos. Como cualquiera puede intuir, una meta tan ambiciosa no puede conseguirse usando HTML. ¿Tendrá el usuario que dominar todas las nuevas tecnologías y cursar algún doctorado en Ciencias de la Computación? No: los usuarios no necesitarán saber RDF ni OWL para navegar por la Web semántica, del mismo modo que el usuario actual no precisa saber HTML ni conocer las especificaciones del protocolo HTTP. Incluso los usuarios avanzados que necesiten escribir sus propias ontologías contarán con editores de ontologías (Protégé, p. ej.) que les evitarán escribir directamente código RDF o OWL.

**Nota biográfica del autor:** Miguel Ángel Abián es licenciado en Ciencias Físicas por la U. de Valencia y obtuvo la suficiencia investigadora dentro del Dpto. Física Aplicada de la U.V. con una tesina acerca de relatividad general y electromagnetismo. Además, ha realizado diversos cursos de postgrado sobre bases de datos, lenguajes de programación Web, sistemas Unix, comercio electrónico, firma electrónica, UML y Java. Ha colaborado en diversos programas de investigación TIC relacionados con el estudio de fibras ópticas y cristales fotónicos, ha obtenido becas de investigación del IMPIVA y de la Universidad Politécnica de Valencia y ha publicado artículos en el *IEEE Transactions on Microwave Theory and Techniques* y en el *European Congress on Computational Methods in Applied Sciences and Engineering*, relacionados con el análisis de guías de onda inhomogéneas y guías de onda elípticas.

En el ámbito laboral ha trabajado como gestor de carteras y asesor fiscal para una agencia de bolsa y actualmente trabaja simultáneamente en los departamentos de **I+D** y de **Sistemas de la Información** de AIDIMA (Instituto Tecnológico del Mueble y Afines), ubicado en Paterna (Valencia), en tareas relacionadas con proyectos nacionales e internacionales de I+D e I+D+i. En dicho instituto se están desarrollando proyectos europeos de comercio electrónico B2B para la industria del mueble basados en Java y XML (más información en [www.aidima.es](http://www.aidima.es)). Ha impartido formación en calidad, normalización y programación para ELKEDE (Grecia), CETEBA (Brasil) y CETIBA (Túnez), entre otros.

Últimamente, aparte de asesorar técnica y financieramente a diversas empresas de la Comunidad Valenciana, es investigador en las Redes de Excelencia **INTEROP** ([www.interop-noe.org](http://www.interop-noe.org)) y **ATHENA** ([www.athena-ip.org](http://www.athena-ip.org)) del Sexto Programa Marco de la Comisión Europea, que pretenden marcar las pautas para tecnologías de la información en la próxima década y asegurar el liderazgo de Europa en las tecnologías de la sociedad del conocimiento. Ambos proyectos tienen como fin la interoperabilidad del software (estudian tecnologías como J2EE, .Net y CORBA, servicios web, tecnologías orientadas a aspectos, ontologías, etc.), y en ellos participan empresas como IBM U.K., COMPUTAS, SIEMENS, FIAT, TXT, GRAISOFT, SAP, EADS, además de numerosas universidades europeas y centros de investigación en ingeniería del software.

Sus intereses actuales son el diseño asistido por ordenador de guías de ondas y cristales fotónicos, la evolución de la programación orientada a objetos, Java, UEMML, el intercambio electrónico de datos, el surrealismo y París, siempre París.