PROTOTIPO DE BUSCADOR SEMÁNTICO APLICADO A LA BÚSQUEDA DE LIBROS DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN EN LA BIBLIOTECA JORGE ROA MARTÍNEZ DE LA UNIVERSIDAD TECNOLÓGICA DE PEREIRA

CARLOS ARTURO MORENO AGUDELO
YAKELINE SÁNCHEZ REYES

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS: ELÉCTRICA, ELECTRÓNICA, FÍSICA Y CIENCIAS DE LA COMPUTACIÓN
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
PEREIRA
2012

PROTOTIPO DE BUSCADOR SEMÁNTICO APLICADO A LA BÚSQUEDA DE LIBROS DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN EN LA BIBLIOTECA JORGE ROA MARTÍNEZ DE LA UNIVERSIDAD TECNOLÓGICA DE PEREIRA

CARLOS ARTURO MORENO AGUDELO
YAKELINE SÁNCHEZ REYES

PROYECTO DE GRADO

Director

INGENIERO

JULIO CESAR CHAVARRO PORRAS

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS: ELÉCTRICA, ELECTRÓNICA, FÍSICA Y CIENCIAS DE LA COMPUTACIÓN
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
PEREIRA

Nota Aceptación:

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

Pereira, marzo de 2012

DEDICATORIA

Durante todo el proceso de mi pregrado tengo gratos recuerdos de grandes personas que han enriquecido mi experiencia, profesores, compañeros, tantos y tantas que tendría una lista inmensa. Lo cierto es que en la cúspide de este proceso entiendo cada frase donde me auguraban una satisfacción, un regocijo por ver la culminación de esto, que realmente no es más que el inicio de mi proyecto de vida. A mi madre Patricia Agudelo Rivera excelencia entre las mujeres.

Carlos Arturo Moreno Agudelo

A mis padres, Luzmila y Eliseo, los seres más maravillosos que Dios pudo darme. Siempre los llevo en mi corazón.

Yakeline Sánchez Reyes

AGRADECIMIENTOS

Nuestros más sinceros agradecimientos al Ingeniero Julio César Chavarro Porras, quien nos brindó su apoyo en todo momento durante este proceso, proporcionándonos todo su conocimiento y material necesario para desarrollar este proyecto.

A Dios, nuestros padres, amigos y a todo aquel que directa o indirectamente participó en este proceso.

CONTENIDO

TABLA DE FIGURAS

INTRODUCCIÓN

A través de los últimos años se ha evidenciado la necesidad de dar mayor sentido a los resultados obtenidos en las búsquedas en la web, debido al crecimiento vertiginoso de la información disponible en internet y que a la hora de realizar una búsqueda, la cantidad de resultados obtenidos se empieza a medir en millones. Muchos de los buscadores actuales realizan una búsqueda basada en estadísticas, ilustrando una lista de páginas web que pueden contener la respuesta, dejando el trabajo al usuario de filtrar todos estos resultados de tal forma que pueda encontrar lo que busca, lo cual se configura en una tarea bastante compleja dado la cantidad de resultados obtenidos.

El gigante de los buscadores, Google, cambia ahora su política de privacidad, de donde se destaca el hecho de que con esta nueva política los diferentes servicios de Google como Youtube, Gmail, Picasa, etc, compartirán la información de los usuarios, es decir, como ya es tradicional en estos portales, almacenan la información de lo que se busca para poder enfocar las sugerencias que se muestra, por ejemplo si se busca un video de perros en Youtube las sugerencias que se muestre serán referente a mascotas, pero con la nueva política al buscar el video, posteriormente en los demás servicios como Google+ aparecerán sugerencias basadas en los videos que se buscaron. Por lo que se puede ver que este sistema de búsqueda es de igual forma basado en estadísticas[1].

Los buscadores semánticos en cambio, encuentran resultados en función del contexto, información más exacta acerca de lo que se busca, ofreciendo una cantidad de resultados mas sesgada, facilitando la labor de filtrar los resultados por parte del usuario.

Desde el punto de vista investigativo, la semántica aplicada a sistemas de búsqueda de información es un tema que a nivel mundial ha madurado, debido a la cantidad de beneficios que promete y a su creciente acogida en las funcionalidades de la web, considerándose pues el estudio de esta temática de gran interés y utilidad por parte de las nuevas generaciones de ingenieros que en gran medida se postulan como los desarrolladores de las tecnologías venideras.

En este proyecto se desarrollará un prototipo de buscador semántico, partiendo de los fundamentos teóricos existentes y, del análisis que se llevará a cabo acerca de las tecnologías que se involucran, como son los agentes inteligentes de software, las ontologías implementadas en lenguajes como RDF y XML, y demás herramientas de desarrollo.

[1] Políticas y principios Google: http://www.google.com/policies/

CAPITULO 1. ESPECIFICACIÓN DEL PROBLEMA

1.1 DESCRIPCION DEL PROBLEMA

Las búsquedas de información tradicionales se realizan de una manera sintáctica mediante el uso de palabras claves con las que la información es indexada; esta forma de búsqueda genera resultados con información dispersa y variada que puede no ser precisa, haciendo más tediosa y lenta la obtención de los resultados esperados.

A pesar de la potencia que demuestran los buscadores sintácticos, aún quedan lejos de poder proporcionar al usuario los resultados adecuados a las consultas realizadas, ya que la cantidad de resultados pueden ser demasiados y por ende bastante tedioso encontrar el resultado deseado, o bien no obtener ningún resultado.

Esto debido principalmente a la alta sensibilidad del vocabulario empleado en la búsqueda, es decir, un documento que se busque debe estar descrito con las mismas palabras que se introduzcan al buscar.

Por lo general en los sistemas de información masivos, como son los repositorios de bibliotecas, la información no cuenta con una estructura que facilite su búsqueda, además los buscadores sintácticos en ningún momento generan dicha estructura, sumándose al problema el continuo crecimiento de la información disponible, por ende las búsquedas sintácticas paulatinamente arrojarían muchos mas resultados no deseados o bien serian menos precisos.

1.2 OBJETIVOS

1.2.1 OBJETIVO GENERAL

Implementar un prototipo de buscador semántico del material bibliográfico disponible para el programa de Ingeniería de Sistemas y Computación que se encuentra en la biblioteca Jorge Roa Martínez de la Universidad Tecnológica de Pereira.

1.2.2 OBJETIVOS ESPECÍFICOS

- Realizar un análisis crítico de las tecnologías involucradas en el desarrollo de un buscador semántico.
- Probar el conjunto de tecnologías seleccionadas, que soporten la construcción de un buscador semántico.

- Implementar un prototipo de buscador semántico.

1.3 JUSTIFICACIÓN DEL PROBLEMA

Al mejorar el sistema de búsqueda de libros de la sección de Ingeniería de Sistemas y Computación en la biblioteca Jorge Roa Martínez de la Universidad Tecnológica de Pereira, se le presta a la comunidad estudiantil un mejor servicio, más preciso y coherente.

Al utilizar una búsqueda semántica la información encontrada como resultado de las consultas será información más exacta y completa, haciendo más útil el sistema de búsqueda, y sesgando de manera eficiente los contenidos mostrados como resultados.

Adicionalmente, el uso de las tecnologías semánticas en este proyecto, sirve de base para realizar otros proyectos utilizando las mismas tecnologías en diferentes dominios de interés.

El tema principal de la monografía, es la web semántica, un tema de actualidad en el cual la Universidad Tecnológica de Pereira no ha realizado muchos estudios al respecto, por lo que se abre un nuevo panorama para las generaciones venideras, del cual se pueden seguir generando nuevos proyectos de grado.

1.4 MARCO TEÓRICO

1.4.1 TEORÍA DE AGENTES[2]

Un agente es un proceso computacional que implementa la autonomía y funcionalidad de comunicación sobre una aplicación. Las teorías de agentes son especificaciones para conceptualizar los agentes. Debido a que la definición de agente ha resultado ser tan controvertida como la definición de inteligencia artificial, se ha optado por la definición de un conjunto de propiedades que caracterizan a los agentes, aunque un agente no tiene que poseer todas estas propiedades:

- *Autonomía:* Los agentes pueden operar sin la intervención de humanos o de otros agentes.
- *Sociabilidad:* Los agentes son capaces de interaccionar con otros agentes (humanos o no) a través de un lenguaje de comunicación entre agentes.
- *Reactividad:* Los agentes son capaces de percibir estímulos de su entorno y reaccionar a dichos estímulos.

[2] FONER LN - ¿Qué es un Agente de todos modos? Un caso de estudio sociológico - Memo Agente 93-01, agentes del Grupo, del MIT Media Lab (1993)

- *Proactividad, iniciativa:* Los agentes no son solo entidades que reaccionan ante un estímulo, sino que tienen un carácter emprendedor, y pueden actuar guiados por sus objetivos.
- *Movilidad:* Capacidad de un agente de trasladarse a través de una red telemática.
- *Veracidad:* Asunción de que un agente no comunica información falsa a propósito.
- *Benevolencia:* Asunción de que un agente está dispuesto a ayudar a otros agentes si esto no entra en conflicto con sus propios objetivos.
- *Racionalidad:* Asunción de que un agente actúa de forma racional, intentando cumplir sus objetivos si son viables.

La cuestión de qué es un agente, esta aun siendo debatida corriendo el riesgo de que cualquier programa sea denominado agente. Se pueden distinguir dos nociones extremas de agentes:

- Una *noción débil* de agente consiste en definir un agente como a una entidad que es capaz de intercambiar mensajes utilizando un lenguaje de comunicación de agentes. Esta definición es la más utilizada dentro de la ingeniería software basada en agentes, cuyo fin es conseguir la interoperabilidad entre aplicaciones a nivel semántico utilizando la emergente tecnología de agentes.
- Una *noción mas fuerte* o restrictiva de agentes es la enunciada por Shoham en su propuesta de programación orientada a agentes (AOP), donde un agente se define como una entidad cuyo estado es visto como un conjunto de componentes mentales, tales como creencias, capacidades, elecciones y acuerdos.

Los agentes suelen ser considerados como sistemas intencionales, esto es, sistemas cuya conducta puede ser predicha atribuyendo creencias, deseos y una conducta racional. Para representar estas intenciones, se han empleado diversos formalismos lógicos, de entre los que cabe destacar la teoría de la intención de Cohen y Levesque, la lógica multi-modal BDI (Creencia, Deseo e Intención; *Belief, Desire, Intention*) y la familia de lógicas para especificar sistemas multiagente propuestas por Wooldrige.

Lenguajes de agentes

Los lenguajes de agentes se definen como lenguajes que permiten programar agentes con los términos desarrollados por los teóricos de agentes. Podemos distinguir dos tipos principales de lenguajes de programación:

- **Lenguajes de agentes de propósito general**: lenguajes destinados a programar agentes genéricos utilizables en cualquier aplicación.

- **Lenguajes de agentes específicos**: lenguajes para un tipo de agentes específico, por ejemplo los lenguajes para agentes móviles *Telescrip* o *Agent-Tcl*.

Se pueden distinguir los siguientes niveles en la programación de agentes:

- **Lenguajes de programación de la estructura del agente**: permiten programar las funcionalidades básicas para definir un agente: funciones de creación de procesos (creación del proceso agente y de los procesos recurrentes con él) y funciones de comunicación entre agentes (nivel de transporte).

- **Lenguajes de comunicación de agentes**: definición del formato de los mensajes intercambiados, de las primitivas de comunicación y de los protocolos disponibles.

- **Lenguajes de programación del comportamiento del agente**: permiten definir el conocimiento de agente: conocimiento inicial (modelo de entorno, creencias, deseos, objetivos), funciones de mantenimiento de dicho conocimiento (reglas, planes, etc.), funciones para alcanzar sus objetivos (planes, reglas, etc.) y funciones para desarrollar habilidades (programación de servicios).

Para crear los agentes existen varios lenguajes de programación. Uno de ellos es Java, el cual usa un Middleware llamado Jade, que es muy usado y es la recomendación de muchas comunidades de investigación, entre ellas la W3C (World Wide Web Consortium)

JSP, Java Server Pages

JSP es un acrónimo de Java Server Pages, que en castellano vendría a decir algo como Páginas de Servidor Java. Es, pues, una tecnología orientada a crear páginas web con programación en Java.

Con JSP podemos crear aplicaciones web que se ejecuten en variados servidores web, de múltiples plataformas, ya que Java es en esencia un lenguaje multiplataforma. Las páginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales para programar scripts de servidor en sintaxis Java. Por tanto, las JSP podremos escribirlas con nuestro editor HTML/XML habitual.

Motor JSP

El motor de las páginas JSP está basado en los servlets de Java -programas en Java destinados a ejecutarse en el servidor-, aunque el número de desarrolladores que pueden afrontar la programación de JSP es mucho mayor, dado que resulta mucho más sencillo aprender que los servlets.

En JSP se crean páginas de manera parecida a como se crean en ASP o PHP -otras dos tecnologías de servidor-. Se generan archivos con extensión .jsp que incluyen, dentro de la estructura de etiquetas HTML, las sentencias Java a ejecutar en el servidor. Antes de que sean funcionales los archivos, el motor JSP lleva a cabo una fase de traducción de esa página en un servlet, implementado en un archivo class (Byte codes de Java). Esta fase de traducción se lleva a cabo habitualmente cuando se recibe la primera solicitud de la página

.jsp, aunque existe la opción de precompilar en código para evitar ese tiempo de espera la primera vez que un cliente solicita la página[3].

Servlets Vs JSPs

- La principal diferencia es el enfoque de programación: Un servlet es un programa que luego de compilado devuelve una página web completa; Una JSP es una página HTML con trozos de código que se convierten en servlets y devuelven una parte de la página.
- Ambos se ejecutan en una máquina virtual Java, lo cual permite que, en principio, se puedan usar en cualquier tipo de ordenador, siempre que exista una máquina virtual Java para él.
- Ambos se ejecutan en su propia hebra, es decir, en su propio contexto; pero no se comienza a ejecutar cada vez que recibe una petición, sino que persiste de una petición a la siguiente, de forma que no se pierde tiempo en invocarlo. Su persistencia le permite también hacer una serie de cosas de forma más eficiente: conexión a bases de datos y manejo de sesiones, por ejemplo.
- Ambos necesitan un programa que los contenga, y sea el que envíe efectivamente páginas web al servidor, reciba las peticiones, las distribuya entre los servlets, y lleve a cabo todas las tareas de gestión propias de un servidor web.

Ventajas de las JSPs

- vs. Active Server Pages (ASP). ASP es una tecnología similar de Microsoft. Las ventajas de JSP son dobles. Primero, la parte dinámica se escribe en Java, no Visual Basic u otro lenguaje especifico de Microsoft, así que es de mayor alcance y más fácil de utilizar. En segundo lugar, es portable a otros sistemas operativos y servidores Web que no sean de Microsoft.

- vs. Servlets Puro. JSP no le da ninguna cosa que usted no podría en principio hacer con un servlet pero es más conveniente para escribir y para modificar. Además, separando la apariencia del contenido usted puede poner a diversas personas en diversas tareas: sus expertos en diseño pueden construir el HTML dejando lugar para que sus programadores de servlets inserten el contenido dinámico.

- vs. Server-Side Includes (SSI). SSI es una tecnología ampliamente soportada para incluir pedazos definidos externamente en una página Web estática. JSP es mejor porque le deja utilizar servlets en vez de un programa separado para generar esa parte dinámica. Además, SSI está pensado solamente para hacer inclusiones simples, no para programas "verdaderos" que utilizan formularios, hacen conexiones a las bases de datos, y demás.

[3] ALVAREZ Miguel, Qué es JSP: http://www.desarrolloweb.com/articulos/831.php

1.4.2 JADE [4]

JADE (Java Agent DEvelopment) es un Middleware creado por TILAB para el desarrollo de aplicaciones multi-agente basados en la arquitectura de comunicación par a par (P2P: Peer to Peer).

Todo la plataforma de agentes, incluyendo la inteligencia, la iniciativa, la información, los recursos y el control pueden ser distribuidos a través de diferentes máquinas (sin necesidad de tener el mismo sistema operativo) y la configuración puede ser controlada por un entorno gráfico remoto.

La comunicación entre pares es completamente simétrica sin importar si están conectados de manera cableada o inalámbrica para que ambos estén en la capacidad de ejecutar el papel de iniciador o respondedor.

Características Principales

- Interoperabilidad - JADE obedece a las especificaciones de FIPA (Foundations of Intelligent Physical Agents), como consecuencia los agentes JADE pueden interactuar con otros agentes que también cumplan con el mismo estándar.
- Uniformidad y Portabilidad – JADE proporciona un conjunto homogéneo de APIs que son independientes de la red y la versión de Java, ya que proporciona los mismos APIs para ambientes J2EE, J2SE y J2ME. Esta característica permite a los desarrolladores de aplicaciones re-usar el mismo código para implementarlo ya sea en un PC, un PDA o un teléfono con tecnología Java.
- Fácil de Usar y Movilidad en Aplicaciones – La complejidad de los Middleware queda oculta detrás de un simple e intuitivo conjunto de APIs fáciles de usar y aprender. JADE ha sido diseñado para simplificar el manejo de la comunicación y el transporte de mensajes, haciendo transparente para el desarrollador la administración de las diferentes capas de comunicación usadas para enviar un mensaje de un agente a otro, permitiendo así que los desarrolladores se concentren en la lógica de la aplicación.
- Facilidades de Negociación y Coordinación – JADE facilita el desarrollo de aplicaciones que requieren negociación y coordinación entre agentes, donde los recursos y el control lógico son distribuidos con el ambiente; de hecho JADE provee librerías de fácil uso para implementar comunicación par-a-par y protocolos de interacción (Patrones de interacción entre entidades autónomas).
- Pro-Actividad – Los agentes JADE controlan su propio hilo de ejecución, por lo tanto pueden ser fácilmente programados para iniciar la ejecución de acciones sin la intervención humana sólo con base en metas o cambios de estado. Esta característica hace que JADE sea un ambiente conveniente para la realización de aplicaciones m2m (máquina-a-máquina), como la automatización de una planta industrial, control de tráfico o administración de redes.

[4] IDSAI, Javier Santos Ferreras: http://www.iit.upcomillas.es/pfc/resumenes/46e713b93f323.pdf

- Aplicaciones Multi-Partido – Las arquitecturas par-a-par son más eficientes que las arquitecturas cliente/servidor para el desarrollo de aplicaciones multi-partido, ya que el servidor puede convertirse en el cuello de botella y punto de falla del sistema entero. Como los agentes JADE pueden tanto proveer como consumir servicios esto elimina la necesidad de distinguir entre clientes y servidores ya que los agentes JADE pueden comunicarse cada uno con los demás sin la intervención de un servidor central.

1.4.3 ONTOLOGÍAS[5]

En la filosofía, una ontología es una teoría acerca de la naturaleza de la existencia de las cosas; la ontología como disciplina estudia dichas teorías.
Toda ontología representa cierta visión del mundo con respecto a un dominio. Por ejemplo, una ontología que defina "ser humano" como "espécimen vivo o muerto correspondiente a la especie *Homo sapiens*; primate bípedo que pertenece a la familia de los homínidos, como los chimpancés, gorilas y orangutanes" expresa una visión del mundo totalmente distinta a la de una ontología que lo defina como "sujeto consciente y libre, centro y vértice de todo lo que existe; todos tienen la misma dignidad, pues han sido creados a imagen y semejanza de Dios".
La inteligencia artificial y los investigadores de la Web han reutilizado el término para su propia jerga, y para ellos una ontología es un documento o archivo que formalmente define las relaciones entre términos. La clase más común de ontología para la Web es la taxonomía y un conjunto de reglas de inferencia.
Una ontología es una especificación explícita de una conceptualización compartida (Studer et al 98), esto es, un marco común o una estructura conceptual sistematizada y de consenso no sólo para almacenar la información, sino también para poder buscarla y recuperarla. Define los términos y las relaciones básicas para la compresión de un área del conocimiento, así como las reglas para poder combinar los términos para definir las extensiones de este tipo de vocabulario controlado. Se trata de convertir la información en conocimiento mediante unas estructuras de conocimiento formalizadas (las ontologías) que referencien los datos, por medio de metadatos, quienes no sólo especificarán el esquema de datos que debe aparecer en cada instancia, sino que también podrán contener información adicional de cómo hacer deducciones sobre ellos, es decir, cómo establecer axiomas que podrán, a su vez, aplicarse en los diferentes dominios que trate el conocimiento almacenado.

Cuando se tratan términos poco comunes o cuando se quiere que estos términos sean procesados por máquinas, se precisa explicitar las ontologías; esto es, desarrollarlas en un documento o darles una forma que sea inteligible para las máquinas.

Las ontologías se usan para solucionar problemas semánticos: de dominio y de nombre. Los conflictos de dominio aparecen cuando conceptos similares en cuanto a significado,

[5] María Jesús Lamarca Lapuente, Hipertexto: http://www.hipertexto.info/documentos/ontologias.htm

pero no idénticos, se representan en distintos dominios. Los conflictos de nombre son de dos tipos: sinónimos y homónimos. Los sinónimos ocurren cuando los sistemas usan distintos nombres para referirse al mismo concepto. Por ejemplo, trabajador y empleado. Los homónimos surgen cuando los sistemas usan el mismo nombre para representar cosas distintas. Por ejemplo, asiento (contable) ó asiento (mueble).

Las ontologías se componen de:

Conceptos: son las ideas básicas que se intentan formalizar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc.

Relaciones: representan la interacción y enlace entre los conceptos de un dominio. Suelen formar la taxonomía del dominio. Por ejemplo: subclase-de, parte-de, parte-exhaustiva-de, conectado-a, etc.

Funciones: son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Por ejemplo, pueden aparecer funciones como: asignar-fecha, categorizar-clase, etc.

Instancias: se utilizan para representar objetos determinados de un concepto.

Reglas de restricción o axiomas: son teoremas que se declaran sobre las relaciones que deben cumplir los elementos de la ontología. Por ejemplo: "Si A y B son de la clase C, entonces A no es subclase de B", "Para todo A que cumpla la condición B1, A es C", etc. Los axiomas, junto con la herencia de conceptos, permiten inferir conocimiento que no esté indicado explícitamente en la taxonomía de conceptos.

1.4.4 SEMÁNTICA

Al hablar de la semántica y de las otras ciencias del epígrafe relacionadas con ella, nos estamos refiriendo al significado de las palabras, no al significado conjunto de un enunciado o de un texto en el que intervienen muchos factores, a veces difíciles de delimitar. Dentro de la semántica se encuentra la lexicología, que estudia las palabras pertenecientes al lenguaje general usado por todos, y en paralelismo, hallamos la terminología, que estudia las palabras del lenguaje especializado, es decir, los términos usados para transmitir el conocimiento de un campo del saber[6].

Cabe hablar ahora del termino Semiótica cuya definición, a través de los años ha sido bastante compleja de lograr, por autores como Saussure, en la corriente europea, y Peirce, en la corriente estadounidense, pero se encuentran algunas coincidencias a las cuales haremos mención, entonces se conoce semiótica como una ciencia que estudia las diferentes clases de signos, así como las reglas necesarias para su generación, transmisión e

[6] Las industrias de la lengua: panorámica para los gestores de la información, Carmen Díez Carrera, primera edición, Fesabid, ISBN 84-88699-09-3

intercambio, recepción e interpretación. Así pues la semiótica está vinculada a la comunicación y a la significación y, en última instancia, de forma que las incluye a ambas, a la acción humana[7].

Ahora bien la semántica es pues una parte de la semiótica, ya que es la encargada de estudiar las relaciones entre los significantes y significados.

El significado o imagen mental está compuesto por una serie de rasgos conceptuales que todos los hablantes de una lengua asocian de una manera general a un significante. No obstante lo dicho, hay que tener en cuenta que este significado tiene dos componentes:
Denotación. Es el significado que presenta una palabra fuera de cualquier contexto. Constituyen el núcleo semántico fundamental. Son comunes a todos los hablantes. Es el significado que encontraremos en el diccionario
Connotación. Son las significaciones que lleva añadidas una palabra. Estas significaciones tienen un carácter marcadamente subjetivo. Dependiendo de los hablantes, una misma palabra puede tener connotaciones distintas.
La semántica estudia las diferentes relaciones que contrae un signo con todos los demás, pues todo el léxico constituye un sistema, cuya estructuración facilita a los hablantes la adquisición de ese léxico[8].

Para que las maquinas comprendan el significado de lo que escribimos o decimos, es necesario elaborar una teoría explicativa del conocimiento que un hablante competente tiene del significado de los enunciados, el cual no corresponde a la simple suma de los significados de cada una de las palabras que lo constituyen, sino que depende también de otros elementos, como el hablante, el oyente, el momento de la locución y el contexto. Se trata en parte de las reglas que regulan la comunicación, que hace referencia a las inferencias del mundo. Estos aspectos no son fáciles de programar, de ahí que la inteligencia artificial no los haya resuelto, si bien ha sido posible formular reglas que determinan actos lingüísticos en contextos limitados[8].

[7] Sebastia Serrano. La semiótica: una introducción a la teoría de los signos. Editorial Montesinos 1998. ISBN:84-85859-32-4
[8] www.profesorenlinea.cl: Semántica, http://www.profesorenlinea.cl/castellano/Semantica1.htm
[8] www.profesorenlinea.cl: Semántica, http://www.profesorenlinea.cl/castellano/Semantica1.htm

1.4.5 BUSCADORES

Un buscador es un tipo de software que crea índices de bases de datos o de sitios web en función de los títulos de los ficheros, de palabras clave, o del texto completo de dichos ficheros. El usuario conecta con un buscador y especifica la palabra o las palabras clave del tema que desea buscar. El buscador devuelve una lista de resultados presentados en hipertexto, es decir que se pueden pulsar y acceder directamente al fichero correspondiente[9].

A continuación se listan algunos tipos de buscadores, tal vez los enfoques más utilizados actualmente a la hora de realizar un software que realice consultas sobre índices de bases de datos o en la web y que nos excluyentes entre si: [10]

- Buscadores por palabras clave: son los más comunes, puedes introducir una palabra clave y el buscador examina su base de datos para mostrar los resultados que coincidan.
- Buscadores por categorías: estos están organizados por temas, de forma que al elegir un tema este nos muestra otra pantalla con más temas sobre aquel que hemos elegido, así sucesivamente hasta llegar a los contenidos que existen de un tema específico.
- Metabuscadores: también llamados buscadores múltiples, ya que realizan varias búsquedas simultáneas en otros buscadores y muestran los resultados obtenidos de acuerdo al buscador que los arrojó.
- Buscadores específicos: son buscadores que solo contienen información sobre un tema concreto y se utilizan a nivel local únicamente.
- Buscadores semánticos: son buscadores que a través de ontologías hacen inferencias para realizar las búsquedas por medio de agentes inteligentes de software, por ejemplo swoogle.

[9] Pergamino virtual, Definiciones, http://www.pergaminovirtual.com.ar/definicion/Buscador.html
[10] Informática Computer Science, volumen I, Patricia Ibáñez, Gerardo García, segunda edición, Cengage Learning, ISBN 10: 607-481-091-5

1.5 ESTADO DEL ARTE

Aplicaciones de buscadores semánticos

A través de los últimos años se ha evidenciado la necesidad de dar mayor sentido a los resultados obtenidos en las búsquedas en la web. Muchos de los buscadores actuales realizan una búsqueda basada en estadísticas, ilustrando una lista de páginas web que pueden contener la respuesta.

Los buscadores semánticos en cambio encuentran resultados en función del contexto, información más exacta acerca de lo que se busca.

Aunque en la actualidad no se ha masificado el uso de buscadores semánticos ya se tienen algunos funcionando y sirven de referencia para el futuro de la búsqueda de información.

Entre algunos de los buscadores semánticos que se tienen en la actualidad están:

Wolfram Alpha

El Sitio oficial del buscador es http://www.wolframalpha.com/.

Es un buscador de respuestas desarrollado por la compañía Wolfram Research. Es un servicio en línea que responde a las preguntas directamente, mediante el procesamiento de la respuesta extraída de una base de datos estructurados, se trata de un motor de búsqueda de conocimiento computacional capaz de responder directamente a las preguntas que hace el usuario en lugar de proporcionar una lista de los documentos o páginas web que podrían contener la respuesta, tal y como lo hace Google.

Una vez formulada la pregunta, la herramienta calcula diferentes respuestas eligiendo de forma selectiva la información de la Red para acabar dando una respuesta precisa. La gran innovación de este programa, según Wolfram, es la capacidad de resolver preguntas concretas de inmediato.

Fue anunciado en marzo de 2009 por el físico británico Stephen Wolfram, y esta en marcha desde el 15 de mayo de 2009.

Algunos expertos consideran que su impacto podría llegar a ser similar al que tuvo Google en su momento[11].

Naturalfinder

Es el complemento esencial de cualquier buscador para Internet e intranets: le permite realizar consultas en lenguaje natural sin necesidad de usar operadores sólo aptos para usuarios avanzados. Gracias a la tecnología lingüística, el usuario se concentra en redactar

[11] Wolfram blog, wólfram: http://blog.wolfram.com/?s=wolfram+alpha&submit.x=0&submit.y=0

la consulta en su propio idioma como si preguntara a otra persona. *NaturalFinder* devolverá todos los documentos que sean relevantes para la consulta y devolverá más documentos que los buscadores tradicionales basados en palabras claves[12].

NaturalFinder se integra con cualquier buscador y lo convierte en un buscador semántico. Algunos ejemplos de buscadores con los que ha trabajado son:

- dtSearch Text Retrieval Engine
- Google Search Appliance
- Autonomy
- OmniFind
- Lucene
- Memex, Oracle SES, SharePoint, etc.

Swootti

Swotti, un interesante servicio creado por la española BuzzTrend, nace con la voluntad de ser un buscador que pese a rastrear toda la red intenta tomar en consideración sólo informaciones que recojan la opinión de los usuarios. Un buscador de opiniones.

Swotti trabaja con principios de la muy comentada pero poco habitual web semántica; es decir, con una tecnología capaz de identificar los adjetivos y verbos que definen aquello que estamos buscando, y que por tanto permiten deducir si el comentario es positivo o negativo. Cuando hacemos una búsqueda en Swotti obtenemos no sólo resultados, sino sobre todo una valoración cualitativa[13].

Swotti está desarrollando sus algoritmos semánticos por familias temáticas, y de momento ha empezado a trabajar en familias como la electrónica de consumo, los videojuegos, la informática, el cine, la literatura, la música, el turismo, la automoción y cosas similares.

Los padres del invento son la gente de BuzzTrend, una empresa española creada en julio de 2007 y que, bajo la dirección de David Castro, quiere especializarse en buscadores focalizados en los contenidos creados por la gente en esta cosa que hemos decidido denominar web 2.0. Por lo visto llevan varios años invirtiendo en esta tecnología, desarrollada en Granada.

Hakia

Su sitio oficial es http://www.hakia.com/

[12] Bitext.com, NaturalFinder: el complemento ideal para su buscador: http://www.bitext.com/ES/sol_naturalfinder.html

[13] Infonomia, Swotti, un buscador de opiniones: http://www.infonomia.com/if/articulo.php?id=408&if=64

Hakia es un buscador de respuestas en idioma inglés que ofrece información en los campos de la medicina, las leyes, las finanzas, las ciencias y la literatura.

Hakia es un buscador que trabaja con un software de una tecnología más inteligente: "meaning-based", es decir, basada en el significado. Su principal novedad consiste en que este buscador "entiende" los contenidos de las páginas Web que procesa y no sólo busca palabras en ellas. Actualmente se encuentra en etapa de prueba (Beta). Desde hace varios años, grupos multidisciplinarios trabajan en nuevas formas de indización sobre las bases de la lingüística, las morfologías, el dominio para la aplicación de los analizadores sintácticos, los correctores ortográficos, los algoritmos semánticos, etcétera.

Hakia, por tanto, no es un buscador convencional; se trata de un buscador de recursos de información que trabaja sobre la base del lenguaje natural en que las búsquedas no son relaciones de palabras o frases, sino interrogantes concretas[14].

GRUPOS DE INVESTIGACIÓN[15]

AKT (Advanced Knowledge Technologies): el consorcio AKT agrupa 5 universidades del Reino Unido y fue fundado por el Engineering and Physical Sciences Research Council (EPSRC). Su fin es ayudar a desarrollar la próxima generación de tecnologías del conocimiento para dar soporte a la gestión del conocimiento de las organizaciones. AKT pretende desarrollar y extender métodos integrados y servicios para capturar, modelar, publicar, reutilizar, compartir y gestionar el conocimiento. Para ello se tienen en cuenta los recientes desarrollos en inteligencia artificial, psicología, lingüística, multimedia y tecnologías de Internet. http://www.aktors.org/akt/

ASG (Adaptive Services Grid): es un proyecto integrado dentro del 6º Programa Marco de la Comisión Europea. El proyecto que comenzó en septiembre de 2004 y dura 2 años, agrupa 22 participantes de 7 países. El objetivo es desarrollar un prototipo de plataforma abierta para la innovación, creación, composición y lanzamiento de servicios. ASG cuenta con las principales organizaciones de producción científica y tecnológica que hacen uso del conocimiento y con las instituciones europeas líderes en la investigación y desarrollo del software, las telecomunicaciones y la industria telemática. http://asg-platform.org/cgi-bin/twiki/view/Public/WebHome.

DIP (Data, Information, and Process Integration with Semantic Web Services): el objetivo de DIP es desarrollar y extender la Web Semántica y las tecnologías de los Servicios Web para producir una nueva infraestructura tecnológica para los Servicios de la Web Semántica. http://dip.semanticweb.org/

[14] Hakia, descripción: http://company.hakia.com/

[15] María Jesús Lamarca Lapuente, Hacia la Web Semántica: http://www.hipertexto.info/documentos/web_semantica.htm

ELeGI (The European Learning Grid Infraestructure): Una red semántica para el aprendizaje humano para la puesta en marcha de escenarios futuros de aprendizaje basado en la ubicuidad y la colaboración, y centrados en la experiencia y el aprendizaje contextualizado a través del diseño, implementación y validación del aprendizaje en red. http://www.elegi.org/

IMS Global: se trata de un consorcio en el que participan más de 50 organizaciones y empresas, que tiene como objetivo el aprendizaje global a través de la Web. En este marco, se trabaja con esquemas XML y documentación estructurada en donde RDF juega un papel fundamental, por ejemplo en la definición de vocabularios y taxonomías. http://www.imsglobal.org

KW (Knowledge Web): es una Red de Excelencia FP6 que ayuda a dar soporte de transición a las tecnologías de ontologías desde el sector académico a la industria. El consorcio actual está integrado por 18 participantes que incluyen líderes en Web Semántica, multimedia, tecnologías del lenguaje humano, agentes, etc. http://knowledgeweb.semanticweb.org/

CAPITULO 2. ONTOLOGÍA CCC (Conceptos de las Ciencias de la Computación)

2.1 ¿QUÉ SON ONTOLOGÍAS?

El termino ontología como tal fue introducido en el siglo XVIII en la filosofía, aunque ha sido una disciplina practicada desde hace cientos de años, que intenta explicar la existencia del ser. En las ciencias de la computación, este término fue redefinido en el área de la Inteligencia Artificial, alejándose de sus inicios filosóficos, para adoptar otras connotaciones como son la representación del conocimiento en diferentes dominios[16].

Una ontología es una especificación formal y define el vocabulario de un área mediante un conjunto de términos básicos y de relaciones entre dichos términos, así como a través de las reglas que combinan términos y relaciones, que amplían las definiciones dadas en el vocabulario (Neches, 1991)[17].

Siendo las ontologías entonces una vía para representar el conocimiento a través de conceptos, que son abstracciones del mundo que queremos representar. Ese mundo se delimita con un dominio, un área específica que posee un conocimiento, el cual se quiere representar de una manera más concreta y aterrizada.

2.2 ELEMENTOS QUE COMPONEN UNA ONTOLOGÍA

En términos generales las ontologías se componen de:

- Clases, subclases o conceptos: Los conceptos son la idea básica que se intenta formalizar.
- Slots o propiedades (también llamados roles): Delimitan las propiedades y características de cada concepto describiendo sus rasgos y atributos.
- Restricciones: Describen valores que una propiedad puede tomar.
- Instancias: Representan objetos determinados de un concepto.
- Axiomas: Son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología[18].

[16]Javier Echegoyen Olleta. Origen de la filosofía-Presocráticos-Sofistas y Sócrates: http://www.e-torredebabel.com/Historia-de-la-filosofia/Filosofiagriega/Presocraticos/Ontologia.htm sitio web visitado el 11 de febrero de 2012

[17]Jorge del Río San José. Introducción al tratamiento de datos espaciales en hidrología. 2010. Editorial Bubok. ISBN: 978-84-9981-141-3

[18] Universidad de Valladolid, Departamento de informática: http://www.infor.uva.es/~sblanco/Tesis/Ontolog%C3%ADas.pdf

La comunidad ontológica distingue dos formas de ontologías, que son[19]:

Ontologías livianas: Que se abstienen de usar reglas o axiomas que restrinjan los conceptos y relaciones entre ellos. Incluye conceptos, taxonomía de conceptos, relaciones entre conceptos y propiedades que describen conceptos.

Ontologías pesadas: Hacen uso de reglas o axiomas que condicionan todo lo anterior, los conceptos, la taxonomía de conceptos, relaciones entre conceptos y propiedades que describen conceptos.

En este proyecto se abstendrá de usar axiomas, debido a que la ontología que se va a realizar será una ontología liviana, es decir, carece de reglas o axiomas que rijan las relaciones entre los conceptos.

2.3 TIPOS DE ONTOLOGÍAS

Steve et al. (1998a: 1) distingue tres tipos de ontologías[20]:

- Ontologías de un dominio: Representan el conocimiento especializado de un área en particular.
- Ontologías genéricas: Representan conceptos generales del conocimiento.
- Ontologías representacionales: Especifican conceptualizaciones que subyacen a los formalismos de representación del conocimiento.

Fensel clasifica las ontologías como[21]:

- Ontologías de dominio: Las cuales capturan el conocimiento válido para un dominio específico.
- Ontologías de metadatos: Las cuales proveen el vocabulario para la descripción de fuentes de información online.
- Ontologías genéricas o de sentido común: Éstas capturas el conocimiento sobre concetos básicos del mundo como tiempo, espacio, estado, evento, etc.
- Ontologías de representación: Las cuales definen los conceptos básicos para la representación del conocimiento.
- Ontologías para tareas particulares: Las cuales especifican los términos para una tarea específica, mostrando el punto de vista tenido en cuenta para analizar un dominio.

Según Van Heijst, Schereiber y Wieringa, las ontologías se clasifican según su uso en:

[19] Uso de ontologías en tareas de recupero de información, Tallarico Marcelo, 2008 http://es.scribd.com/doc/55621955/4/Definiciones-de-ontologia
[20] Pérez Chantal, Estudios de Lingüística del Español, http://elies.rediris.es/elies18/index.html
[21] Cavero Barca José María, Aspectos filosóficos, psicológicos y metodológicos de la informática, Editorial Dykinson S.L, ISBN:84-9772-749-5

- Ontologías terminológicas: Donde se especifican cuales son los términos usados para representar el conocimiento.
- Ontologías de información: Que especifican los mecanismos de almacenamiento de información.
- Ontologías de modelado de conocimiento: Las cuales especifican la conceptualización del conocimiento.

Guarino las clasifica según el nivel de generalidad en:

- Ontologías de alto nivel: Donde se encuentran las definiciones de conceptos independientes del dominio.
- Ontologías de dominio: Las cuales describen los conceptos de un dominio o tarea genérica.
- Ontologías de aplicación: Las cuales describen los conceptos para un caso o aplicación específica dentro de un dominio.

2.4 DISEÑO DE ONTOLOGÍAS[22]

Los criterios de diseño que se presentan a continuación fueron propuestos por Thomas R. Gruber, quien ha desarrollado trabajos sobre definición de ontologías en el área de la Inteligencia Artificial.

Criterios de Diseño

1. Claridad: Las definiciones deben ser objetivas e independientes del contexto social.
2. Coherencia: Los axiomas que define deben ser lógicamente consistentes.
3. Extensibilidad: La ontología debe prestarse para definir nuevos términos basada en el vocabulario existente.
4. Sesgo de codificación mínima: Se produce cuando una serie de opciones de representación se realizó únicamente a conveniencia de la notación o ejecución.
5. Compromiso ontológico mínimo: La ontología debe hacer la menor cantidad posible de afirmaciones sobre el mundo que está modelando.

2.4.1 ¿CÓMO SE CONSTRUYE UNA ONTOLOGÍA?

El proceso de desarrollo de una ontología es un proceso de cooperación entre partes que poseen conocimiento sobre el área de interés. Es un proceso de colaboración que depende del contexto en el cual se desarrolla la ontología[19].

[22] Gruber Thomas, Toward Principles for the Design of Ontologies Used for Knowledge Sharing, 1993
[19] Uso de ontologías en tareas de recupero de información, Tallarico Marcelo, 2008 http://es.scribd.com/doc/55621955/4/Definiciones-de-ontologia

Dicho proceso de desarrollo no es único, existentes diferentes formas que dependen de las necesidades y finalidades que las personas involucradas tengan en mente[23].

En términos generales el desarrollo de una ontología incluye:

- Definir clases en la ontología
- Organizar las clases y subclases
- Definir slots y describir valores permitidos para estos
- Llenar los valores de los slots para las instancias

Ahora se presentan los pasos propuestos por Natalya F. Noy y Deborah McGuinness en "Desarrollo de Ontologías-101: Guía para crear tu primera ontología" para crear una ontología.

1. Determine el dominio y alcance de la ontología

Es muy importante el sector específico para el cual se va a desarrollar la ontología, ya que ese será el dominio. El alcance de la misma, depende de qué es lo que se quiere resolver con ella, es decir, qué información contendrá la ontología para responder a las preguntas de quienes la usen.

2. Considere la reutilización de ontologías existentes

Revise en el medio si ya existen ontologías referentes al dominio que usted desea trabajar, esto le puede ayudar a enriquecer su ontología u ofrecer una base para iniciar la construcción de la suya.

3. Enumere términos importantes para la ontología

Liste los conceptos que considere pertinentes para formar su ontología, por ahora no se preocupe por las relaciones entre conceptos, recuerde que la ontología se va podando a través del tiempo.

4. Definir las clases y la jerarquía de clases

Se seleccionan términos que describan de forma general los conceptos enumerados en el paso 3, éstos serán los nombres de las clases.

[23] Natalya F. Noy. Deborah L. McGuinness. Desarrollo de Ontologías-101: Guía Para Crear Tu Primera Ontología. 2005

5. Definir las propiedades de las clases: slots

Los slots son propiedades que poseen los conceptos, atributos que se les puede asociar.

6. Definir las facetas de los slots

Los slots tienen diferentes facetas que describen los tipos de valores que puede admitir el slot. Por ejemplo tipo de valor, valores admitidos, número de valores, entre otros.

7. Crear instancias

Utilizados para representar valores específicos de objetos de una clase.

2.5 APLICACIONES DE LAS ONTOLOGÍAS[24]

Según el Grupo de Trabajo en Ontologías del consorcio W3C, las ontologías tienen las siguientes aplicaciones:

- Portales web: Están basados en sistemas de administración de contenidos, el cual permite publicar información en la web de manera fácil y rápida.
- Colecciones multimedia: Son conjuntos de imágenes, sonidos, videos y películas.
- Administración de sitios web corporativos: Estos sitios ayudan a resolver problemáticas de su organización en internet, las ontologías facilitan la administración de los mismos.
- Agentes inteligentes: Un agente inteligentes es un conjunto de hardware y software capaz de realizar varias actividades cognitivas.
- Servicios web y computación ubicua: Es la integración de diferentes tecnologías para trabajar conjuntamente en tareas variadas.
- Comercio electrónico: Las ontologías se utilizan en este campo para realizar la descripción del dominio y de los usuarios, las tendencias y relaciones entre ellas.

[24]Úcar Ventura, María Pilar. Romana García, María Luisa. Traducción e interpretación: estudios, perspectivas y enseñanzas. Editorial UNE. 2011. ISBN:978-84-8468-373-5

- Búsqueda de información en internet: Las ontologías guían la búsqueda en un dominio concreto, agregando así mayor flexibilidad a la búsqueda.

2.6 HERRAMIENTAS PARA LA CONSTRUCCIÓN SEMIAUTOMÁTICA DE ONTOLOGÍAS

Text-to-Onto[25]

Soporta la creación semiautomática de diccionarios semánticos usando algoritmos de minería de texto. Extrae las estructuras conceptuales usando la frecuencia de términos del texto, la jerarquía de conceptos es extraída usando unos algoritmos de agrupación jerárquica y las relaciones no taxonómicas son extraídas usando algoritmos de minería de asociación de reglas. Este sistema requiere la verificación del usuario en cada etapa del proceso de extracción de ontologías.

Ontolearn[25]

Es un método de extensión de ontologías basado en minería de textos y técnicas de aprendizaje automático. Ontolearn empieza con una ontología genérica existente como WordNet u otras, y un conjunto de documentos de un dominio dado.

Terminae[26]

Terminae es a la vez un método y una herramienta. El objetivo de Terminae es facilitar la modelización de un dominio a partir de textos. El usuario modeliza el dominio circunscrito por el vocabulario del corpus con el fin de construir una representación conceptual. Esta representación está constituida por un conjunto de conceptos estructurado jerárquicamente. Los conceptos son definidos por sus relaciones entre ellos y llevan un nombre. Algunos de estos conceptos tienen como nombre un término del corpus, y han sido elaborados a partir del estudio del sentido de este término en el corpus. Estos últimos son llamados " conceptos terminológicos".

La herramienta Terminae constituye una plataforma de ayuda a la construcción de terminologías y de ontologías a partir del estudio de textos. Integra herramientas lingüísticas.

[25]Lavín Villa, Moisés. Construcción automática de diccionarios semánticos usando la similitud distribucional. Trabajo de grado Maestro en Ciencias de la Computación. México, D.F. Instituto Politécnico Nacional, 2010. 35, 37 p.

[26] http://www-lipn.univ-paris13.fr/~szulman/TERMINAE.html

2.6.1 HERRAMIENTAS PARA LA EDICIÓN ONTOLÓGICA

Kaon[27]

La ontología es una infraestructura de código abierto de gestión orientada a aplicaciones de negocio. Incluye un conjunto de herramientas completo que permite la creación fácil de la ontología y la gestión y proporciona un marco para la construcción de aplicaciones basadas en ontologías. Un objetivo importante de KAON es un razonamiento escalable y eficiente con las ontologías.

Hozo[28]

Es un editor integrado para la construcción de ontologías. La ontología y el modelo resultante están disponibles en diferentes formatos (lisp, texto, XML/DTD, DAML + OIL) que lo hacen portátil y reutilizable. Una de las características más notables de Hozo es que puede tratar el concepto de rol.

Hozo proporciona a los usuarios una interfaz gráfica a través de la cual pueden acceder y modificar la ontología. Los usuarios no tienen que preocuparse por la llamada de codificación para desarrollar una ontología. La representación interna del editor es XML y general el código DAML OIL para exportar la ontología.

WebODE[29]

WebODE fue construido como una plataforma escalable, extensible e integrada que cubre y da soporte a la mayoría de las actividades involucradas en el proceso de desarrollo de ontologías relaciones a las ontologías que permite interoperar con otros sistemas de información. Entre estos servicios, la plataforma integra servicios para la importación y exportación de lenguajes de ontologías para generar documentación, para evaluar, para controlar la evolución, para extraer ontologías, para juntarlas y un motor de inferencia.

El editor de ontologías WebODE permite editar y navegar por las ontologías, y esta basada en formularios HTML y applets de Java.

[27] Kaon Tool Suite sitio oficial. [web en línea] http://kaon.semanticweb.org/
[28] Riichiro Mizoguchi , Tutorial on ontological engineering. The Institute of Scientific and Industrial Research, Osaka University
[29] Ontology Engineer ingGroup. [web en línea]. http://mayor2.dia.fi.upm.es/oeg-upm/index.php/es/downloads/60-webode

OntoConcept[30]

Este framework es una herramienta para la edición ontológica, fue desarrollado en el año 2010, en la tesis doctoral *Marco de referencia para la gestión del cambio en ontologías basados en modelos conceptuales* por el Ingeniero Julio César Chavarro Porras docente de la Universidad Tecnológica de Pereira.

Este marco de referencia (framework) permite probar el funcionamiento de los operadores de cambio ontológico en el nivel conceptual y, posteriormente, trasladar este modelo a un lenguaje de implementación de ontologías. Una de sus características es la capacidad para controlar el ciclo de vida de una ontología desde su creación. Esta característica, facilita la reconstrucción de la ontología a partir del log de cambios.

Tecnológicamente, el marco de referencia es un servicio Web, extensible, portable, con especificación modular basada en WSDL 2.0, y guiado por los principios del software libre.

A continuación se presenta la arquitectura del framework:

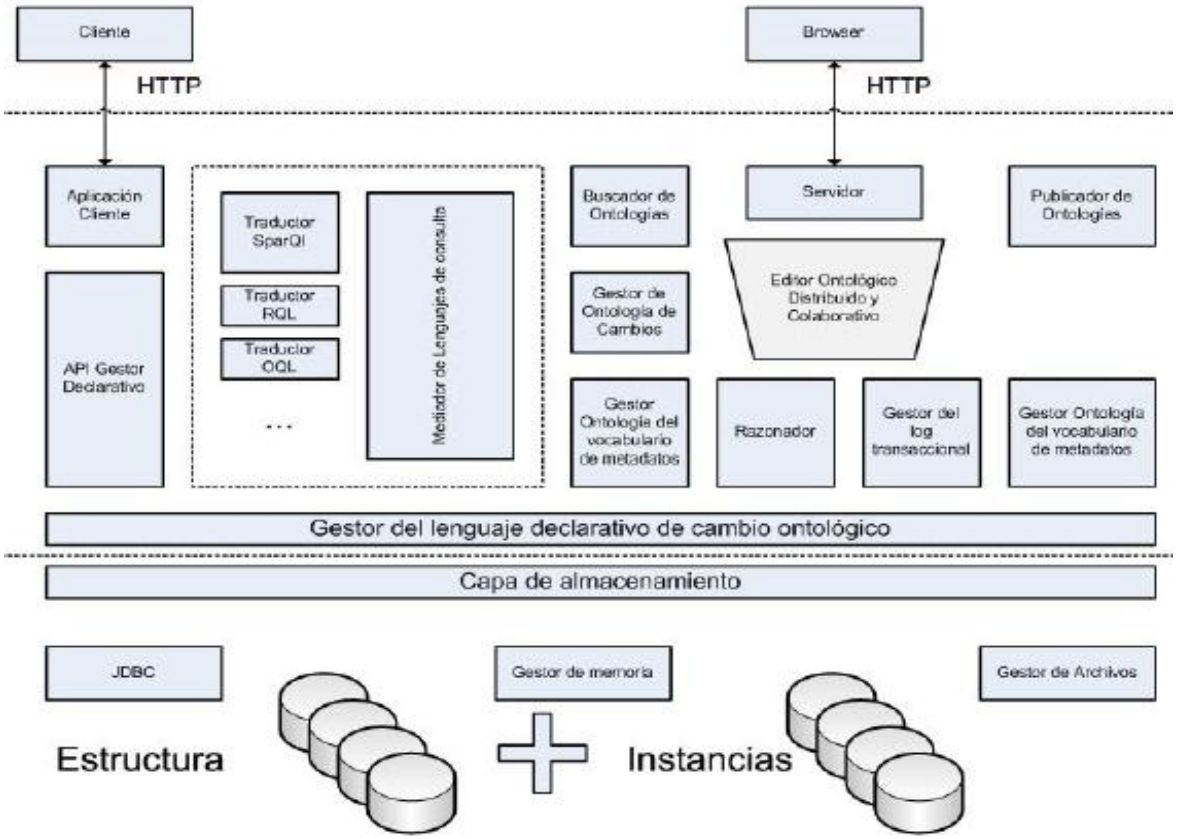

Figura 1 Arquitectura de la herramienta OntoConcept

[30] CHAVARRO Porras, Julio César. Marco de referencia para la gestión del cambio en ontologías basados en modelos conceptuales. 2010.

En el modelo arquitectural propuesto, se presentan las siguientes capas:

En la capa de almacenamiento, es donde se propone gestionar la persistencia ofreciendo las opciones de conectividad a una base de datos, o gestión autónoma de archivos. Además, esta capa se encarga de administrar los bloques de memoria y las relaciones entre la memoria y los bloques de control de persistencia.

En la capa de aplicación, se distribuyen los diferentes módulos, y se ofrecen dos posibilidades de conexión: modo cliente, a través de una aplicación cliente, la cual está conectada al API del gestor declarativo, o en modo servidor Web, donde la conexión se realiza desde un navegador en la estación cliente.

Los principales módulos de la arquitectura propuesta son:

1. Gestor de memoria.
2. Gestor del lenguaje declarativo.
3. Cargador / Descargador de ontologías.
4. Gestor del log transaccional.
5. Razonador
6. Gestor de Ontología de vocabulario de metadatos
7. Gestor de Ontología de cambio genérico
8. Editor Ontológico
9. Buscador de ontologías
10. Publicador de ontologías
11. Mediador de lenguajes de consulta
12. Traductores para lenguajes de consulta de ontologías: Sparql, OQL, RQL
13. API del gestor declarativo
14. Aplicación Cliente

Protégé

Protégé es una plataforma de código abierto que provee a una creciente comunidad de usuarios con un conjunto de herramientas para la construcción de modelos de dominio y las aplicaciones basadas en el conocimiento con ontologías.

Protégé implementa un amplio conjunto de estructuras de modelado del conocimiento y las acciones que apoyan la creación, visualización y manipulación de ontologías en diversos formatos de representación.

La plataforma de Protégé soporta dos formas de modelado de conocimiento, Marcos (frames) y Lógica de descripción (DL) [31].

- El editor Protégé-Frames permite a los usuarios construir y poblar las ontologías

[31] Protégé sitio official [web en línea]. http://protege.stanford.edu/overview/index.html

que están basada en marcos, de acuerdo con la Open Knowledge Base protocolo de conectividad (OKBC). En este modelo, una ontología consta de un conjunto de clases organizadas en una jerarquía de subsunción para representar los conceptos más destacados de un dominio, un conjunto de espacios asociados a las clases para describir sus propiedades y relaciones, y un conjunto de instancias de dichas clases - ejemplares individuales de los conceptos que contienen valores específicos por sus propiedades.

- El editor Protégé-OWL permite a los usuarios construir ontologías para la Web Semántica, en particular en el lenguaje de la Web del W3C (OWL). "Una ontología OWL puede incluir descripciones de las clases, las propiedades y sus instancias. Teniendo en cuenta como una ontología, la semántica formal de OWL especifica cómo derivar sus consecuencias lógicas, hechos, es decir, literalmente, no presentes en la ontología, sino que entraña la semántica. Estas vinculaciones pueden basarse en un solo documento o varios documentos distribuidos que se han combinado con mecanismos definidos búho".

Sus principales características incluyen[32]:

1. Modelo de conocimiento extensible para permitir a los usuarios redefinir las primitivas de representación.
2. Una salida de formato de archivo personalizable para adaptarlo a cualquier lenguaje formal.
3. Una interfaz de usuario personalizable.
4. Un plug-in con arquitectura potente para permitir la interacción con otras aplicaciones.

Siendo la ontología de este proyecto, un modelo consistente con frames, se utilizará Protégé en la versión 3.4 como el editor de ontologías para este proyecto. En él se hará la edición de la ontología, creada para realizar el prototipo de buscador semántico. Su sitio oficial es http://protege.stanford.edu/, de allí se pueden descargar diferentes versiones del software.

[32] STAAB Steffen. Handbook on Ontologies. Editorial Springer. Segunda edición. ISBN: 978-3-540-70999-2

Arquitectura de Protégé

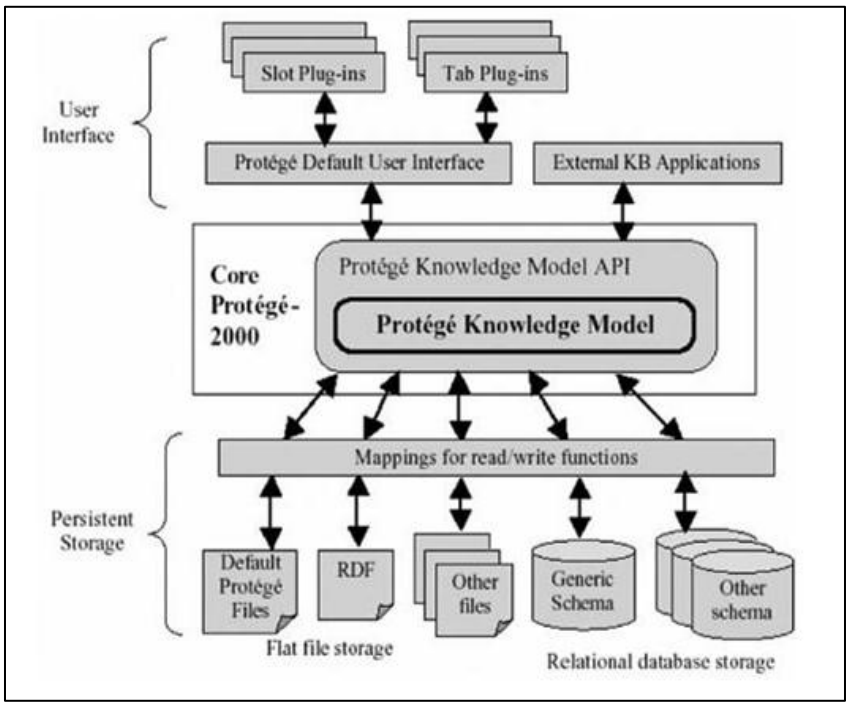

Figura 2 Arquitectura del framework Protégé

2.6.2 LENGUAJES DE PROGRAMACION PARA ONTOLOGIAS

OWL[33]

Este lenguaje de definición de diccionarios semánticos fue creado por el World Wide Web Consortium (W3C), OWL (Lenguaje de ontologías Web) está diseñado para ser usado en aplicaciones que necesitan procesar el contenido de la información en lugar de únicamente representar información para los humanos. OWL facilita un mejor mecanismo de interpretabilidad de contenido Web que los mecanismos admitidos por XML, RDF, y esquema RDF (RDF-S) proporcionando vocabulario adicional junto con una semántica formal.

OWL se puede formular en RDF, incluye toda la capacidad expresiva de RDF(S) y la extiende con la posibilidad de utilizar expresiones lógicas.

OWL permite atribuir ciertas propiedades a las relaciones, como cardinalidad, simetría, transitividad o relaciones inversas.

OWL es un lenguaje para representar el conocimiento de manera descriptiva, basado en lógica. OWL nos permite definir un diccionario semántico expresándose en lenguaje XML.

[33]W3C, Lenguaje de Ontologías Web (OWL), 2004, http://www.w3.org/2007/09/OWL-Overview-es.html#s1.3

Un diccionario semántico OWL puede incluir clases, propiedades y sus instancias. La semántica formal de OWL especifica como derivar sus consecuencias lógicas.

El lenguaje OWL tiene 3 sub-lenguajes que incrementan su expresión: OWL Lite, OWL DL, y OWL Full.

OWL Lite está diseñado para aquellos usuarios que necesitan principalmente una clasificación jerárquica y restricciones simples, proporciona una ruta rápida de migración para tesauros y otras taxonomías.

OWL DL está diseñado para aquellos usuarios que quieren la máxima expresividad conservando completitud computacional y computabilidad. OWL DL incluye todas las construcciones del lenguaje de OWL, pero sólo pueden ser usados bajo ciertas restricciones. OWL DL es denominado de esta forma debido a su correspondencia con la lógica de descripción.

OWL Full está dirigido a usuarios que quieren máxima expresividad y libertad sintáctica de RDF sin garantías computacionales. OWL Full permite una ontología para aumentar el significado del vocabulario prestablecido. Es poco probable que cualquier software de razonamiento sea capaz de obtener un razonamiento completo para cada característica de OWL Full.

RDF (Resource Description Framework)[34]

Es un lenguaje para representar información sobre los recursos de la World Wide Web y las relaciones entre ellos. Está particularmente diseñado para representar metadatos, que son datos que permiten describir otros datos. También puede usarse para representar información sobre cosas que son identificadas en la Web.

El lenguaje RDF o Infraestructura para la Descripción de Recursos es muy útil en situaciones en las que la información necesita ser procesada por aplicaciones que intercambian información legible por máquina, más que por humanos. RDF provee un marco común de trabajo para expresar esta información y para intercambiarla entre aplicaciones distintas mediante una serie de "parsers" o analizadores RDF y otras herramientas de procesamiento automatizado. RDF puede utilizarse en diferentes áreas como en la recuperación de recursos para los buscadores, robots y agentes inteligentes, catalogación para describir el contenido y las relaciones de contenido disponibles en un sitio web o en una colección de documentos, para describir los derechos de propiedad intelectual o las políticas de privacidad de un sitio web, etc.

RDF está basado en la idea de identificar los recursos en la Web usando los Uniform Resource Identifiers o URIs, y describiendo los recursos en términos de propiedades simples y valores. Una descripción RDF es un conjunto de proposiciones simples (también

[34] Lamarca Lapuente María Jesús. RDF. http://www.hipertexto.info/documentos/rdf.htm

llamadas sentencias o declaraciones) y una proposición se conoce también como una tripleta, porque está compuesta de 3 cosas: un sujeto, un predicado y un objeto. Estas sentencias se pueden representar formalmente usando la tripleta (sujeto, predicado, objeto), pero existe otra forma de notación que es mostrar una sentencia mediante grafos dirigidos. Así, en RDF es posible representar declaraciones simples sobre los recursos como un grafo (graph) de nodos y arcos que representan los recursos, y sus propiedades y valores. Los sujetos y objetos son nodos, mientras que los predicados son arcos.

Así pues, una tripleta se representa mediante nodos conectados por líneas con etiquetas. Los nodos representan recursos y las líneas con etiquetas las propiedades de esos recursos. Los 3 elementos de una tripleta se representan mediante URIs.

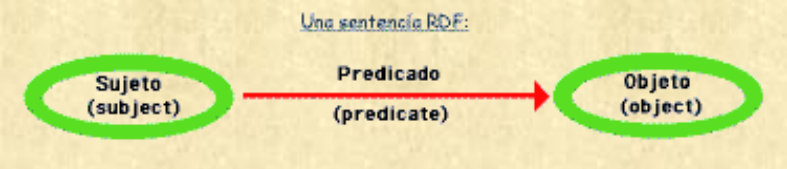

Figura 3 Ilustración de una tripleta RDF

RDF Schema[35]

En RDF es fundamental utilizar palabras que transmitan un significado inequívoco con el fin de que las aplicaciones entiendan el enunciado para un procesamiento correcto. En RDF, este significado se expresa a través de un esquema. Podemos pensar en un esquema como una especie de diccionario que define los términos que se utilizarán en una declaración o sentencia RDF para otorgarle significados específicos. Con RDF se pueden utilizar una gran variedad de formas de esquema, incluyendo la definida en RDFSchema que posee unas características especiales para automatizar tareas utilizando RDF, pero también otras muchas formas.

RDFS permite definir los términos que se usarán en las declaraciones RDF y les otorgará significados específicos. Para evitar definiciones conflictivas del mismo término, RDF utiliza los namespaces de XML. RDFS permite modelar metadatos con una representación explícita de su semántica y permite especificar restricciones de tipos de datos para los sujetos y objetos de las *tripletas* de RDF, introduciendo unas primitivas de modelado orientado a objetos: **rdfs:Class**, **rdfs:Property**, **rdfs:subClassOf**

RDF *Schemas* ofrece un entramado en el cual las comunidades independientes pueden desarrollar vocabularios que se adapten a sus necesidades específicas. Para compartir vocabularios, el significado de los términos debe describirse con detalle. A las descripciones de estos conjuntos de vocabularios se les llaman RDF *Schemas*. Un *schema*

[35] Lamarca Lapuente, María Jesús. Esquema RDF. http://www.hipertexto.info/documentos/esquemas_rdf.htm

define el significado, características y relaciones de un conjunto de propiedades. El lenguaje RDF permite que cada documento que contiene metadatos, sea clarificado con el vocabulario empleado asignando a cada vocabulario una dirección *web*.

El esquema RDF es, básicamente, un conjunto de declaraciones que definen clases y propiedades. Se puede pensar en un RDF *Schema* como en metadatos para una declaración.

XML (eXtensible Markup Language)[36]

XML son las siglas del Lenguaje de Etiquetado Extensible. Con la palabra "Extensible" se alude a la no limitación en el número de etiquetas, ya que permite crear aquellas que sean necesarias.

XML surgió como un lenguaje de marcado para sustituir a HTML. Ambos lenguajes son herederos de SGML, el lenguaje de marcas estándar para la descripción formal y de contenido de los documentos, no solamente para la presentación de dichos documentos.

La primera definición de XML fue la de "Sistema para definir, validar y compartir formatos de documentos en la Web". Para crear XML se tomaron las mejores partes tanto del lenguaje SGML como del HTML. La diferencia fundamental entre HTML y XML es que el primero estaba orientado a la presentación de datos, mientras que XML está orientado a los datos en sí mismos, por lo que cualquier software informático trabajará mejor con XML. Sin duda, esta diferencia es fundamental para los nuevos desarrollos de la Web donde se da suma importancia al contenido de los datos y su tratamiento, y no sólo a su presentación.

XML no es un lenguaje, sino un metalenguaje o lenguaje para definir otros lenguajes.

XML no sustituye a HTML puesto que sirven para cosas distintas: una cosa es presentar la información y otra bien distinta es representar e intercambiar los datos de forma independiente a su presentación.

XML es un lenguaje que permite jerarquizar y estructurar la información y describir los contenidos dentro del propio documento, así como la reutilización de partes del mismo.

XML consiste de una serie de reglas, pautas o convenciones para planificar formatos de texto para tales datos, de manera que produzcan archivos que sean fácilmente generados y leídos por un ordenador, que sean inequívocos y que eviten los problemas más comunes como la falta de extensibilidad, la falta de interoperabilidad entre plataformas o la falta de soporte para universalizar su tratamiento. Los archivos XML son archivos de texto, pero más difíciles de leer por las personas que los archivos HTML. Se puede usar un editor de texto para programar XML, pero cualquier error u olvido de una etiqueta dejará inservible dicho archivo. El lenguaje XML es más estricto que el HTML.

[36] Lamarca Lapuente, María Jesús. XML. http://www.hipertexto.info/documentos/xml.htm

XML no especifica ninguna semántica o conjunto de etiquetas. De hecho, XML es realmente un metalenguaje para describir lenguajes de marcas. En otras palabras, XML facilita definir etiquetas y relaciones estructurales entre ellas. Desde que un conjunto de etiquetas no se predefine, no puede haber una semántica preconcebida. De todas las semánticas de un documento XML, ésta será definida por las aplicaciones del proceso o por las hojas de estilo.

XML se ha creado para enriquecer la estructura de los documentos que pueden ser usados en la Web, puesto que las otras alternativas viables, HTML y SGML, no eran demasiado prácticas para este propósito.

2.6.3 LENGUAJES DE CONSULTA DE ONTOLOGÍAS

RQL (Lenguaje de Consulta de RDF)[37]

RQL es un lenguaje de consulta para RDF y RDF Schema, basado en la sintaxis de OQL.

Una característica de RQL es que aborda la semántica RDF Schema en el propio idioma. Instanciación de clases, la subsunción de clase, propiedad, dominio, rango y demás están todos tratados e inferidos por lenguajes específicos de construcción.

RQL es un lenguaje muy potente y versátil, pero que también toma tiempo dominar.

Una consulta RQL normalmente se construye a partir de tres cláusulas, las cuales se pueden reconocer desde SQL: select, from y where. Sin embargo su uso es ligeramente diferente de SQL.

Considere la siguiente consulta:

```
select X, @P
   from  {X} @P {Y}
   where  Y like "Pablo"
```

El *select* permite especificar las variables que se mostrarán en el resultado y en qué orden. En la consulta anterior, se está interesado en las variables X y @ P, pero no en la variable Y. También es posible especificar que desea todas las variables que están incluidas en el resultado de la consulta, mediante el asterisco (*). Sin embargo, cuando se utiliza el asterisco, este debe ser el único argumento en la cláusula *select*, y el orden, en que las variables son devueltas, no se puede especificar.

El *from* es donde las cosas buenas suceden. Aquí se unen las variables a las localizaciones específicas en el modelo gráfico RDF por expresiones de ruta especificada. En este

[37] Jeen Broekstra. Sesame RQL: a Tutorial. 2004. http://www.openrdf.org/doc/rql-tutorial.html

ejemplo, X e Y están obligados a nodos en el gráfico, mientras que @ P se une a un borde de conexión (la @ es un prefijo variable que representa la variable sólo se une a las propiedades). Por lo tanto, esta estructura corresponde a una declaración donde X es el sujeto, @P el predicado, y por ultimo Y es el objeto.

El *where* es opcional y se puede utilizar para restringir los valores de las variables con destino en la cláusula *from*. En el ejemplo, sólo se quiere los valores de regreso, donde el valor de Y es igual a la cadena "Pablo". Esto corresponde a la selección de todas las declaraciones donde "Pablo" es el objeto.

Namespace

En RDF, los nodos y los bordes se identifican por medio de su identificador universal de recursos, o URI. Estos identificadores pueden ser bastante largos, por lo que las consultas son difíciles de leer. Es por eso que RQL tiene un mecanismo de abreviatura de namespaces (bastante similar al mecanismo utilizado en XML).
Se especifican las abreviaturas de namespaces a través de una cláusula adicional en la parte final de la consulta: *using namespace*. En esta cláusula se especifica un prefijo y el URI a la que corresponde, por ejemplo:

cult = http://www.icom.com/schema.rdf#

Ahora, cada vez que se usa una propiedad o un recurso de este *namespace*, por ejemplo, la propiedad *paint*, puede simplemente escribirse como *cult: paints* en lugar de la URI completa.

Operadores

RQL conoce dos tipos básicos de operadores: los operadores de comparación y operadores lógicos.

Los operadores de comparación

Los operadores de comparación son operadores binarios con los que se comparan los valores de sus operandos y devuelven verdadero o falso según el resultado de la comparación.

Los operadores lógicos

Los operadores lógicos son operadores que hacen una combinación lógica de los valores de verdad de sus dos operandos. Se pueden utilizar en donde se combinan varios operadores de comparación. Los operadores lógicos disponibles son AND, OR y NOT.

Al hacer combinaciones con estos operadores, se puede expresar restricciones muy fuertes en una consulta.

Las funciones estándar

RQL ofrece varias funciones estándar para la recuperación de las relaciones RDFS estándar. Estas funciones se pueden utilizar en una consulta *select-from-where*, pero también pueden ser utilizados como consultas independientes en su propio derecho.

La función *class* recupera todas las clases conocidas. Como una consulta independiente que puede ser utilizada sin ninguna asignación de variables:

class

La función *subClassOf ()* puede usarse para consultar la jerarquía de clases. Como una consulta independiente, se puede utilizar para recuperar todas las subclases de una clase en particular:

subClassOf(http://www.icom.com/schema.rdf#Artist)

La función *typeof ()* permite recuperar las clases a las que pertenece un recurso en particular:

typeOf(http://www.european-history.com/picasso.html)

Características principales de RQL http://139.91.183.30:9090/RDF/RQL/

Compatible con:

- XML esquema tipos de datos (tipos de datos para filtrar valores literales)
- primitivas de grupo (por la construcción de valores anidados)
- operaciones aritméticas (para la conversión de valores literales)
- agregar funciones (para la extracción de estadísticas)
- espacio de nombres, namesspaces (instalaciones para el manejo de distintos esquemas)
- metaschemas de consulta (para los esquemas de navegación)
- duplicado de la eliminación (select distinct)
- cuantificación de los iteradores (existe, FORALL)
- recorrido recursivo de las jerarquías de clase y propiedad (por avanzada de coincidencia de patrones)

Presiona tanto como sea posible la evaluación de la consulta del DBMS subyacente

- Emplea sistemas de etiquetado o de las taxonomías de esquema, para optimizar las consultas de cierre transitivos
- Selecciones forzadas, combinaciones y la creación de resultado (siempre que sea

posible) a la DBMS subyacente
- Beneficios de motores robustos SQL3 de consulta
- Uso extensivo de los índices de DB

Proporciona resultados de la forma genérica en RDF / XML

- XSL de procesamiento para la representación personalizada

El RQL está disponible bajo la licencia RDFSuite. Tiene tanto un tiempo de ejecución y una distribución de código fuente. Para instalar y ejecutar RQL se requiere, C + + (gcc-2.95.1 o superior) y PostgreSQL (v7.3 o superior) DBMS son obligatorios. RQL v2.1 también incluye un script *configure* para facilitar la instalación y personalización en diferentes plataformas. El autor del software es Greg Karvounarakis (CS Departamento - Universidad de Creta y CSI-CUARTO - Grecia).

OQL (Lenguaje de Consulta de Objetos)

Es un lenguaje de consultas similar a SQL para consultar la pila de Java. OQL permite filtrar o seleccionar información deseada de la pila de Java. Mientras consultas predefinidas, tales como "mostrar todas las instancias de la clase X" ya son soportados por Red Hat, OQL añade una mayor flexibilidad. OQL se basa en el lenguaje de expresión JavaScript.

La consulta OQL es de la forma:

select <JavaScript expression to select>
[from [instanceof] <class name> <identifier>
[where <JavaScript boolean expression to filter>]]

Donde el nombre de la clase (*<class name>*) está plenamente cualificado dentro del nombre de una clase de Java (ejemplo: java.net.URL) o como un arreglo, ejemplo: [C es el nombre del arreglo de caracteres, [java.io.File; el nombre de la clase java.io.File; [] y así sucesivamente. Tenga en cuenta que el nombre completo de la clase no siempre se identifica de forma única a una clase Java en tiempo de ejecución. Puede haber más de una clase Java con el mismo nombre pero cargado por diferentes cargadores. Por lo tanto, el nombre de la clase se le permite ser una cadena de caracteres *id* de la clase del objeto.

Si la palabra clave *instanceof* se usa, los objetos subtipo se seleccionan. Si esta palabra clave no se especifica, sólo las instancias de la clase exacta especificada se seleccionan. Tanto desde y donde las cláusulas son opcionales.

La pila Java de objetos se envuelve como objetos de secuencias de comandos convenientes

para que los campos se puedan acceder en la sintaxis natural. Por ejemplo, los campos de Java se pueden acceder con la sintaxis *obj.field_name* y los elementos del arreglo se pueden acceder con *array [indice]*. Cada objeto Java seleccionado se enlaza a una variable de JavaScript del nombre del identificador especificado en la cláusula *FROM*.

Ejemplos de OQL

Selecciona todas las cadenas de caracteres de longitud 100 o más:

select s from java.lang.String s where s.count >= 100

Selecciona todos arreglos de enteros de longitud 256 o mayor:

select a from [I a where a.length >= 256

Muestra el contenido de un string que coincide con una expresión regular:

select s.value.toString() from java.lang.String s
where /java/(s.value.toString())

Muestra el valor de la ruta a todos los archivos de objetos (File objects):

select file.path.value.toString() from java.io.File file

Muestra el nombre de los cargadores de clases:

select classof(cl).name
from instanceof java.lang.ClassLoader cl

Muestra la instancia de una clase identificada por un id dado:

select o from instanceof 0xd404b198 o

Note that 0xd404b198 is id of a Class (in a session).
This is found by looking at the id shown in that class's page.[38]

[38]Herong Yang. Object Query Language (OQL). 2008 http://www.herongyang.com/Java-Tools/jstack-jhat-Object-Query-Language-OQL.html

OQL incorporación de objetos (built-in), funciones[39]

Pila de objetos

La pila built-in de objetos soporta los siguientes métodos:

Heap.forEachClass: Llama a una función de devolución de llamada para cada clase de Java

heap.forEachClass(callback);

heap.forEachObject: Llama a una función de devolución de llamada para cada objeto de Java

heap.forEachObject(callback, clazz, includeSubtypes);

clazz es la clase cuyas instancias son seleccionadas. Si no se especifica, por defecto es *java.lang.Object. includeSubtypes* es un indicador booleano que especifica si se incluyen los casos del subtipo o no. El valor por defecto de este indicador es true.

heap.findClass: Encuentra el nombre de la clase Java.

heap.findClass(className);

donde *className* es el nombre de la clase a encontrar. El Objeto de la clase resultante tiene las siguientes propiedades:

name: nombre de la clase.

superclass: clase de objeto para la super clase (o nulo si *java.lang.Object*).

statics: nombra, pares de valores de los campos estáticos de la clase.

fields: arreglo de objetos de campo. Los objetos de campo tienen nombre, propiedades.

loader: objeto cargador de clases que ha cargado esta clase.

Los objetos de la clase tienen los siguientes métodos:

isSubclassOf: prueba si determinada clase es subclase directa o indirecta de esta clase o no.

[39] VisualVM. Analyzing a Heap Dump Using Object Query Language (OQL). http://visualvm.java.net/oqlhelp.html

isSuperclassOf: verifica si una clase dada es directa o indirectamente una superclase.

subclases: retorna un arreglo de las subclases directas e indirectas.

superclases: retorna un arreglo de las superclases directas o indirectas.

SESAME[40]

Sesame es un framework de código libre de Java para guardar, buscar y razonar con RDF y RDF Schema. Puede ser usado como una base de datos para RDF y RDF Schema, o como una librería Java para aplicaciones que necesiten trabajar con RDF internamente. Por ejemplo, suponga que se necesita leer un documento RDF grande, encontrar la información relevante para la aplicación, y usar esa información. Sesame provee herramientas para parsear, interpretar, buscar y guardar toda esa información, embebidas en la aplicación si así se prefiere, o bien, en una base de datos separada o incluso en un servidor remoto.

Arquitectura de Sesame

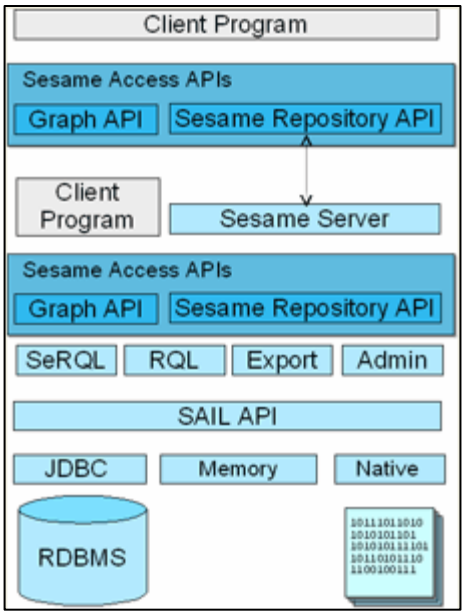

Figura 4 Arquitectura de Sesame

[40] Práctica de recuperación y organización de la información. Universidad Carlos III de Madrid. Susana Corbera Delgado. Ingeniería Superior Informática.

Capa SAIL API (Storage And Interference Layer): Es un API interna de Sesame que se abstrae del formato de almacenamiento usado. Esta capa también proporciona soporte de razonamiento.

Modulos funcionales de Sesame (SeRQL, RQL, Export, Admin): A éstos modulos de funcionalidad se puede acceder a través de la API de acceso Sesame que consiste en dos partes separadas: Repository API y Graph API.

APIs de acceso: Proporcionan acceso directo a los modulos de funcionalidad de Sesame, tanto a un cliente como a un servidor.

Repositorios e Inferencia

Un concepto central en el framework Sesame es el *repositorio*. Un repositorio es un contenedor de RDF. Esto simplemente significa un objeto Java (o un conjunto de objetos Java) en memoria, o en una base de datos relacional, de cualquier manera que se almacene. Es importante saber que casi cualquier operación en Sesame sucede con respecto a un repositorio: cuando se adicionan datos RDF, se adicionan a un repositorio. Cuando se hace una búsqueda, se hace en un repositorio particular.

Sesame, como se mencionó, soporta inferencia de RDF Schema. Esto significa que dado un conjunto de RDF y/o RDF Schema, Sesame puede encontrar la información implícita en los datos. Sesame soporta esto, adicionando toda la información implícita al repositorio cuando los datos estén siendo adicionados.

Es importante saber que la inferencia en Sesame está asociada con el tipo de repositorio que se use. Sesame soporta muchos tipos de repositorios. Algunos de estos soportan inferencia, otros no. El realizar inferencia en Sesame depende mucho de la aplicación que se esté realizando.

2.7 DESARROLLO DE ONTOLOGÍA PARA EL PROTOTIPO DE BUSCADOR SEMÁNTICO

El desarrollo de ontologías ha sido utilizado para representar los elementos de la vida real de un dominio específico, para facilitar la comprensión de la información a las máquinas. Inicialmente fue una temática exclusiva de los laboratorios de inteligencia artificial, pero se ha ido extendiendo hacia todas las áreas del conocimiento a través de expertos, debido a la necesidad de reutilizar la información, de tal forma que se pueda extender una ontología previamente desarrollada.

El desarrollo de la ontología para el prototipo de este proyecto, se basó en la taxonomía *Computing Classification System* (clasificación de los sistemas computacionales). Desarrollada por ACM, Association for Computing Machinery (Asociación para la

Maquinaria Computacional), es la mas grande sociedad informática, educativa y científica del mundo, que proporciona los recursos para desarrollar la computación como una ciencia y profesión. Es la librería digital más grande en el campo computacional.

Por ende el dominio de la ontología desarrollada es las ciencias de la computación. La taxonomía en cuestión se ilustra en la figura 5.

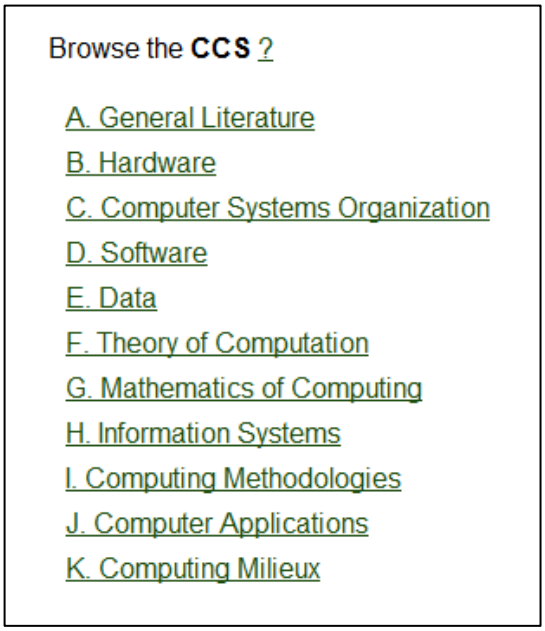

Figura 5 Taxonomía de la clasificación de los sistemas computacionales

La edición de la ontología se desarrolló en Protégé versión 3.4, donde los diferentes conceptos son extraídos de la taxonomía de ACM.

Basados en la guía *Desarrollo de Ontologías-101: Guía Para Crear Tu Primera Ontología* elaborada por Natalya F. Noy y Deborah L. McGuinness, se llevaron a cabo los siguientes pasos para editar la ontología, para el prototipo de buscador semántico.

Paso1

Se selecciona el lenguaje sobre el que se va a soportar la ontología, para el caso en cuestión se selecciona *OWL/RDF Files*. Se deja el URI por defecto ya que la ontología no es necesario que tenga un alcance web. Por la naturaleza de la ontología se selecciona *RDF schema and OWL* como perfil de lenguaje. Finalmente se selecciona *Logical View.*

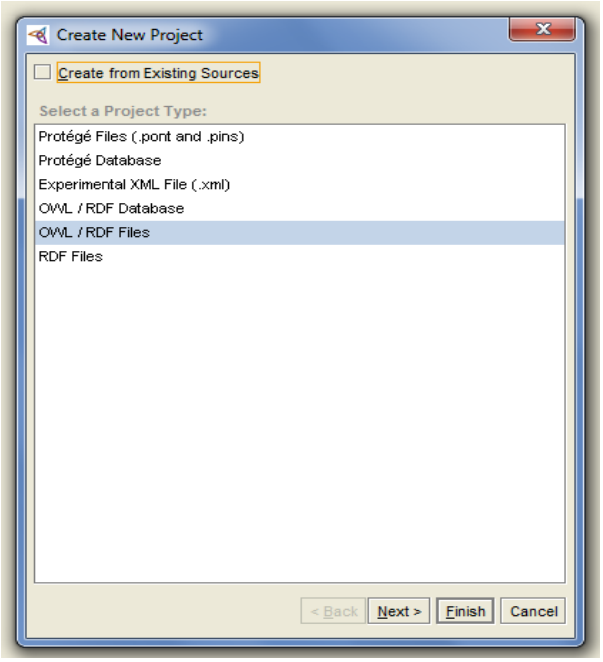

Figura 6 Menú de opciones para crear un nuevo proyecto en protégé 3.4

Paso2

Para construir los conceptos o clases, de la ontología, se ingresa a la pestaña *OWLClasses*, donde se empiezan a crear clases y subclases de acuerdo al dominio, para este proyecto se utilizó una metodología Top-Down, donde se parte de los conceptos más generales hasta llegar a los particulares.

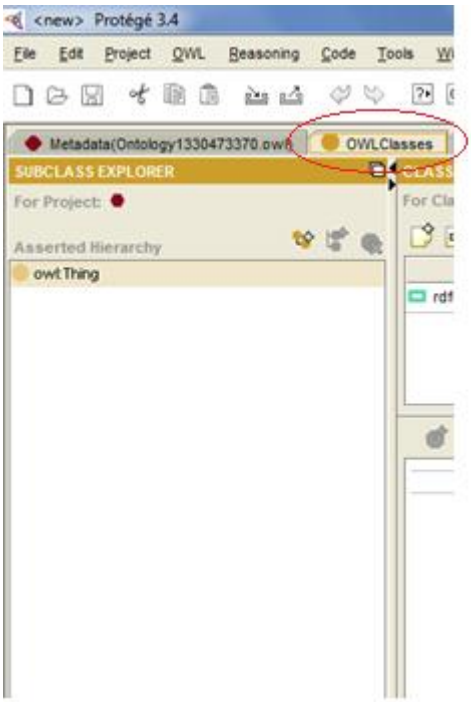

Figura 7 Framework Protégé para la creación de la ontología.

Finalmente en la figura 7 se ilustra el resultado de la ontología desarrollada, llamada CCC (Conceptos de las Ciencias de la Computación).

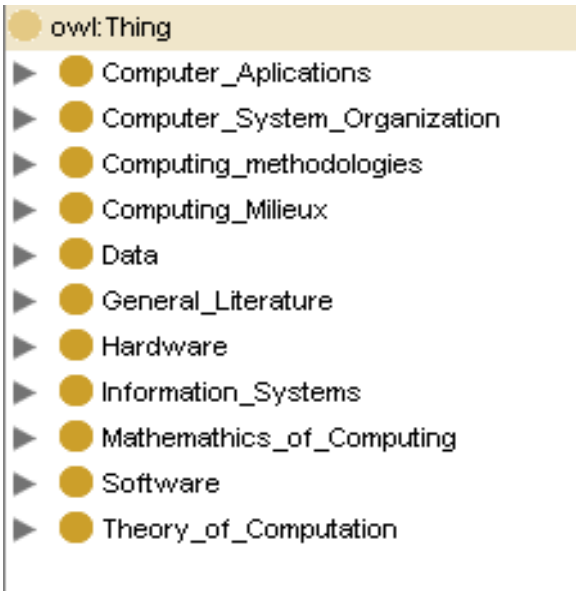

Figura 8 Ontología editada en Protégé

Una vez editada y guardada la ontología, se procede a exportarla al formato OWL, seleccionando *file- export to format- OWL.* El archivo generado puede ser visualizado con un editor de texto cualquiera, y allí se aprecia el código de la ontología.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns="http://www.owl-ontologies.com/Ontology1329146842.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.owl-ontologies.com/Ontology1329146842.owl">
  <owl:Ontology rdf:about=""/>
  <rdfs:Class rdf:ID="Health">
    <rdfs:subClassOf>
      <rdfs:Class rdf:ID="LIFE_AND_MEDICAL_SCIENCES"/>
    </rdfs:subClassOf>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Ordinary_Differential_Equations">
    <rdfs:subClassOf>
      <rdfs:Class rdf:ID="NUMERICAL_ANALYSIS"/>
    </rdfs:subClassOf>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Reference">
    <rdfs:subClassOf>
      <rdfs:Class rdf:ID="General_Literature"/>
    </rdfs:subClassOf>
  </rdfs:Class>
  <rdfs:Class rdf:ID="SPECIAL-PURPOSE_AND_APPLICATION-BASED_SYSTEMS">
    <rdfs:subClassOf>
```

Figura 9 Fragmento del código generado al exportar la ontología CCC al lenguaje OWL

CAPITULO 3.

3.1 HERRAMIENTAS PARA EL PROCESAMIENTO DE LA ONTOLOGIA

3.1.1 JENA[41]

Como se vio en el capítulo anterior la creación de ontologías es el primer paso para hacer que una computadora pueda darle *significado* a la información que procesa, que no se limite a guardar, mostrar, transmitir o comprimir datos sino que pueda hacer inferencias sobre estos datos de tal forma que cuando se realice una consulta, la maquina pueda mostrar los resultados más acertados.

Una parte esencial para poder lograr esto es construir una buena representación del conocimiento, es decir, definir un modelo de información que pueda ser usado por la máquina. Este modelo claramente puede ser una ontología bien definida, que para ser procesada por una maquina debe encontrarse en un lenguaje computacional. Para esto una opción es utilizar RDF, que en resumidas cuentas genera un grafo dirigido donde, los nodos representan los conceptos (cualquier cosa que pueda ser representada por un identificador uniforme de recurso o URI por sus siglas en ingles), los arcos representan las propiedades entre los conceptos, que también pueden ser vistas como las relaciones entre los conceptos. Véase figura 10.

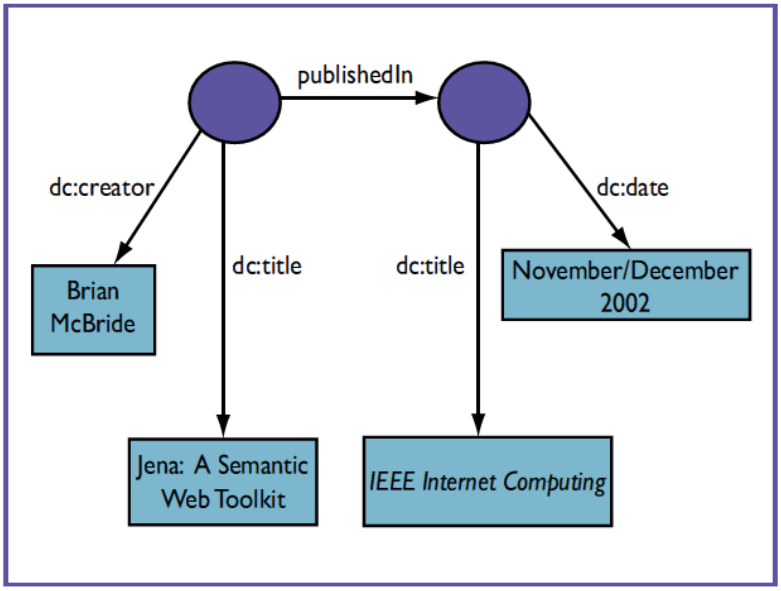

Figura 10 Grafo RDF

[41] Brian McBride, Jena: A Semantic Web Toolkit, Hewlett-Packard Laboratories, Bristol, UK. 2002

Para un humano la comprensión de un grafo RDF resulta una tarea relativamente sencilla, siempre y cuando conozca el contexto o dominio del mismo, ya que solo tendría que echarle un vistazo, mientras que una maquina a la que se le entrega el grafo de la figura 10, lo interpretaría nada mas como una imagen y no haría ningún tipo de procesamiento sobre esta mas que el tradicional de mostrarla o almacenarla. Es en este punto donde se hace importante el uso de herramientas como JENA, que es una herramienta desarrollada por los laboratorios Hewlett Packard, diseñada para facilitar la construcción de aplicaciones que utilicen modelos de información semántica.

El corazón de esta herramienta es el API RDF, que soporta la creación, manipulación y consulta sobre grafos RDF. Su interfaz permite adaptar el grafo sin importar el lenguaje que se haya usado para desarrollarlo.

API RDF

Desde el punto de vista del API hay dos formas de ver un grafo RDF. La primera es como una *vista de declaración centrada*, donde el grafo RDF se compone de una tripleta, un nodo al inicio de un arco, el arco mismo y el nodo al final del arco. Este modelo de vista es importante cuando es necesario ver el grafo como un todo, para leerlo, escribir sobre él o mezclar diferentes grafos.

La segunda forma es como una *vista de un instante centrado,* en esta se ve al grafo como una colección de conceptos, cada uno con sus relaciones o propiedades. Esta forma de vista tiene cierta similitud al paradigma de programación orientado a objetos, donde los objetos serian los conceptos y las relaciones los métodos para interactuar con los demás objetos. Es más conveniente para la navegación del grafo y la manipulación individual de los conceptos. El API de Jena integra ambas formas de vista en una sola.

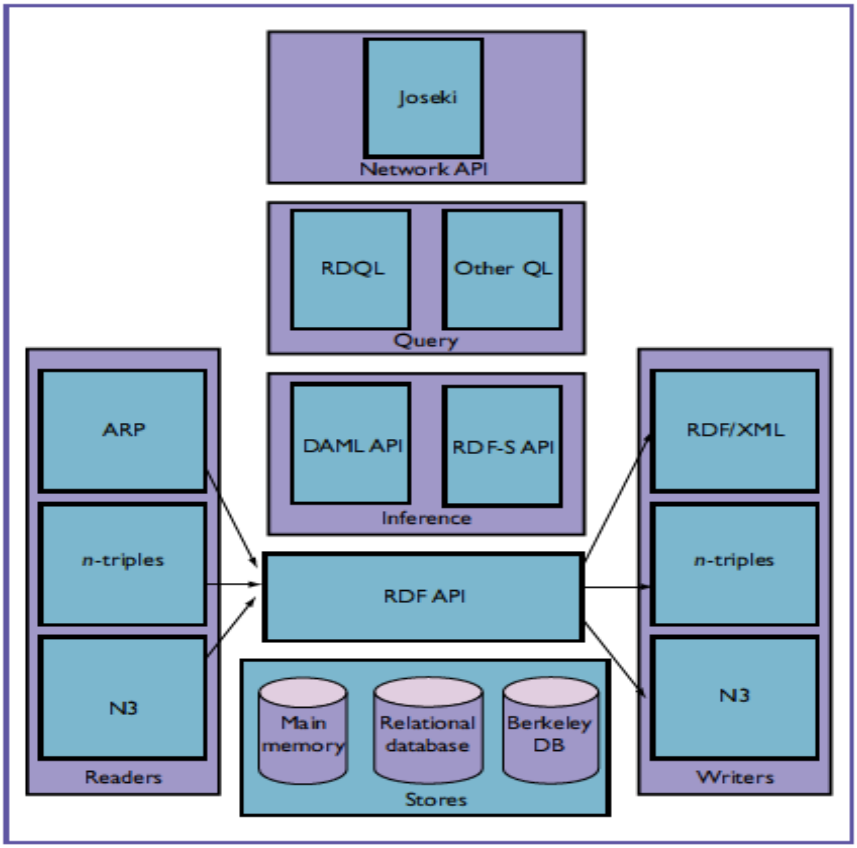

Figura 11 Arquitectura JENA

Jena provee una primitiva de consulta, la cual es un método para la extracción de un subconjunto de tripletas de un grafo específico, llevado a cabo por un objeto selector. Este objeto puede ser tan simple como encontrar las tripletas que coincidan con un parámetro dado o bien puede llegar a ser más complejo de acuerdo al manejo que se tenga de la herramienta.

Determinar si un grafo es similar a otro puede ser sencillo siempre y cuando se tenga un grafo con todos sus arcos etiquetados, sería suficiente con comparar los arcos de cada grafo. El problema aparece cuando se tienen grafos con arcos sin etiqueta, para esto Jena implementa un algoritmo capas de determinar la igualdad entre dos grafos a partir de detectar un isomorfismo entre ambos. Además sobre los grafos se pueden llevar a cabo operaciones de unión, intersección, diferencia y se pueden representar en diferentes lenguajes para su interpretación, ya sea para lectura o para escritura.

Para lectura se utiliza ARP, que es un analizador RDF/XML, el lenguaje de los grafos RDF. Este actúa como un compilador que utiliza el estándar de XML como preprocesador para generar una lectura sobre la gramática de RDF/XML.

Para la escritura es bastante flexible en cuanto a la sintaxis de RDF/XML, permite escribir un grafo de muchas formas diferentes, ya sea un grafo donde no se preocupe por su tamaño

y estilo o bien por el contrario se busque un grafo compacto, con las limitaciones de escalabilidad adheridas a este.

Almacenamiento

El almacenamiento se maneja de tres formas básicamente, una es en memoria principal, otra en bases de datos relacionales y por ultimo en un tipo de bases de datos de software libre llamadas Berkeley. Para las bases de datos relacionales se puede usar cualquier tipo de bases de datos que soporte conectividad con una base de datos Java, para este tipo de almacenamiento se tiene todas las funcionales de personalización normales como son la especificación de las tablas y su estructura. La base de datos Berkeley viene embebido y se comporta como una base de datos relacional, a excepción del manejo de transacciones.

Inferencia

Para almacenar un grafo RDF puede declararse explícitamente cada tripleta o bien se puede construir grafos parciales que luego con el uso de reglas de conclusión se completen. Las reglas de conclusión pueden generar tripletas basadas en algunos conceptos y relaciones generadas previamente, por ejemplo si se tiene que la clase A es subclase de la clase B, y que el recurso R es de tipo A, entonces una regla de conclusión puede añadir al grafo una declaración donde R es del tipo B. Estas operaciones de inferencia pueden llegar a ser un poco más complejas si se desea construir grafos mucho más eficientes en cuanto a su representación, utilizando reglas de transitividad entre las relaciones.

API DAML

En este API se definen las primitivas básicas para el manejo de ontologías. Provee abstracciones para la creación y manipulación de ontologías DAML, con las cuales soporta operaciones algo limitadas de inferencia, conclusiones de subclase y propiedades, transitividad y propiedades inversas.

Consulta

El lenguaje que utiliza Jena para realizar consultas sobre los grafos RDF es RDQL, el cual tiene gran similitud con SQL el lenguaje de consulta sobre bases de datos.

Jena es pues una herramienta básicamente para consultar sobre una base de datos dada, de donde, a partir de una estructura dada (ontología), puede generar un grafo RDF para llevar a cabo su procesamiento.

3.1.2 PLATAFORMA JADE: CREACIÓN DE AGENTES INTELIGENTES[42]

Java Agent Development Framework (JADE) es una plataforma de desarrollo de agentes inteligentes que permite implementar la comunicación entre éstos, su comportamiento y distribución en diferentes equipos. También permite el manejo de información usando ontologías. La comunicación entre agentes determina su comportamiento social y además permite el intercambio de datos y conocimiento.

FIPA (Foundation for Intelligence Physical Agents) es una organización de estándares de la Sociedad de Computación IEEE que promueve la tecnología basada en agentes y la interoperabilidad de sus estándares con otras tecnologías. Es un organismo de estandarización de sistemas de agentes inteligentes.

FIPA propuso un lenguaje para la comunicación entre agentes inteligentes llamado FIPA-ACL (Agents Communication Language). FIPA-ACL se ocupa de la estructura de los mensajes, la cual esta constituida por un conjunto de parámetros, cuya cantidad puede variar de acuerdo a la situación en que se busca establecer una comunicación efectiva entre agentes; el único parámetro que es obligatorio en todo mensaje ACL es la *performative*, que hace referencia al tipo de acto comunicativo, sin embargo se espera que en todo mensaje ACL se defina el *sender* (Emisor), el *reciver* (Receptor) y el *content* que es el contenido del mensaje. El mensaje también puede contener recursivamente otros mensajes.

En esta comunicación se transmiten los mensajes con información requerida entre agentes, el contenido del mensaje está expresado en un lenguaje de contenido como es FIPA-SL o XML, puede hacer referencia a una ontología; el lenguaje de contenido permite representar proposiciones, acciones y términos. El mensaje con su contenido van encapsulados en la carga útil del mensaje de transporte que es el encargado de llevar el mensaje.

El lenguaje de contenidos en este caso es FIPA-SL (Semantic Language). Es un lenguaje definido por la FIPA para facilitar la comunicación entre agentes. SL es basado en una lógica multimodal con operadores modales. El contenido de cada mensaje en FIPA-ACL se define como un conjunto de formulas SL que describen las condiciones necesarias para el emisor y lo que un agente espera que ocurra como resultado de una acción.

[42] Ernesto Jiménez Ruiz, Desarrollo de sistemas multi-agente con jade: aplicación de las ontologías, Universidad Jaume I de Castellón

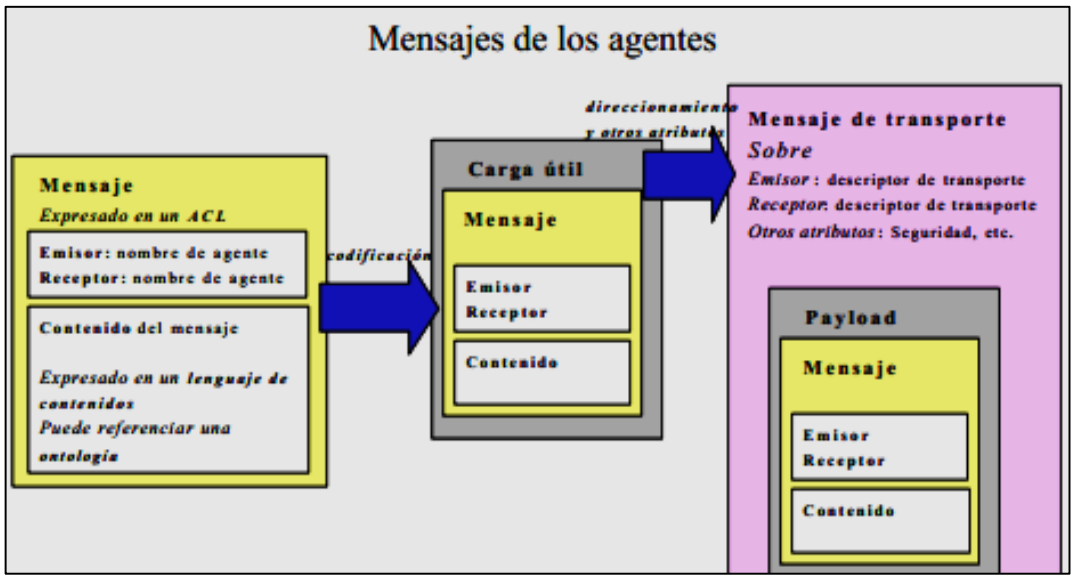

Figura 12 Estructura del mensaje de transporte

El mensaje de transporte consta de un campo de carga útil (payload) y un sobre. Cuando el mensaje es enviado es transformado en un playload e incluido en un mensaje de transporte. El sobre incluye las descripciones de transporte del emisor y del receptor e información adicional que puede ser requerida.

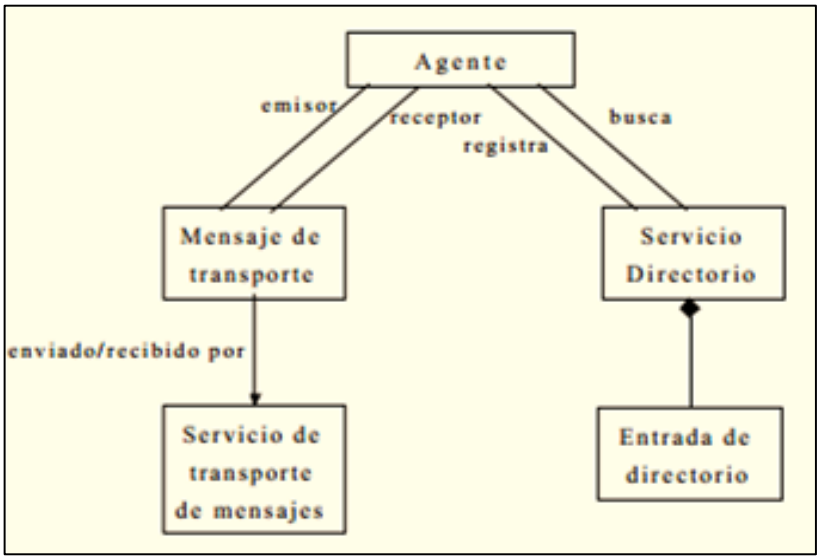

Figura 13 Relaciones básicas del agente

Cada agente tiene una cola de mensajes donde almacena los mensajes que recibe y los lee como le resulte más conveniente, es decir, puede leer el primer mensaje de la cola o leer

uno que satisfaga un requisito determinado.

Para enviar un mensaje se siguen los siguientes pasos:

- Se crea un objeto ACLMessage
- Se usa el método de ACLMessage para rellenar los campos necesarios
- Se llama el método send() de la clase Agent

El método send() recibe como parámetro un ACLMessage, añade el valor al campo sender (emisor) y envía el mensaje a los destinatarios.

Para recibir un mensaje se siguen los siguientes pasos:

- Se usa el método receive() de la clase Agent

El método receive() obtiene el primer mensaje de la cola de mensajes y lo devuelve. En caso de que no existan mensajes en la cola, devuelve null (nulo). Cada vez que un mensaje es recibido el receptor es notificado mediante un evento.

Los mensajes son implementados como un objeto de la clase jade.lang.acl.ACLMessage, que contiene métodos *set()* y *get()* para manejar los campos del mensaje.

Para llevar a cabo las conversaciones entre los agentes es necesario contar con un protocolo de comunicación. En JADE se distinguen dos roles: *initiator* que es el agente que inicia a comunicación y *responder* que es el agente destino de la comunicación. Todos los elementos necesarios para dicha comunicación se encuentran en el paquete *jade.proto*.

CREACIÓN DE AGENTES EN JADE

La creación de un agente en JADE es similar a la definición de una clase Java que extiende la clase *Agent (jade.core.Agent)* e implementa el método *setup()*.

Al crearse el agente se llama al constructor, se crea un identificador para el agente (AID), se registra el agente en el AMS (Agent Management System) y se ejecuta el método *setup()*. El método *setup()* contiene las tareas de inicialización del agente. Los identificadores de los agentes se instancian en la clase *jade.core.AID*.

Comportamiento de los agentes

Las tareas de un agente son objetos de una clase que extiende a la clase *jade.core.behaviours.Behaviour*. Los comportamientos de un agente se pueden agregar (addBehaviour()) o eliminar (removeBehaviour()) en cualquier momento. La clase Behaviour es una clase de los aspectos básicos de los comportamientos de los agentes.

La programación de un agente consiste en definir el comportamiento que el agente debe tener, en otras palabras consiste en implementar la clase Behaviour, donde se especifica lo que el agente debe realizar de acuerdo a un mensaje que reciba.

Las clases que extiendan un comportamiento, o bien hagan uso de la clase Behaviour, deben implementar los métodos *action()* y *done()*.

Método *action ()*

Define lo que se debe hacer según el comportamiento del agente, cada acción es respuesta a un comportamiento determinado.

Método *done ()*

Indica cuándo se ha llevado a cabo un comportamiento, para así eliminarlo de la lista de comportamientos que debe presentar el agente.

En el entorno de los agentes hay un agente específico encargado de prestar el servicio de páginas amarillas, que es donde los agentes publican los servicios que ofrecen para ser utilizados por los agentes que los requieran. Al utilizar este servicio el agente debe proporcionar su AID (identificación del agente), la lista de lenguajes, ontologías necesarias y de servicios proporcionados. También se proporciona el servicio de páginas blancas que está a cargo de un agente el cual mantiene el registro de las direcciones de los agentes de la plataforma.

3.2 MODELO ARQUITECTURAL DEL PROTOTIPO DE BUSCADOR SEMÁNTICO

3.2.1 Cliente

En la arquitectura propuesta para el prototipo del buscador semántico se tiene en la parte superior el entorno del cliente, donde él tendrá acceso a una página donde podrá introducir las consultas que desea, las cuales serán tomadas para procesarlas y generar un código XML, el cual será llevado hasta la plataforma de agentes. De igual forma en la parte cliente se cuenta con el proceso encargado de cargar la página del buscador cuando el cliente acceda a la misma; cabe anotar que en la misma página donde realiza la consulta se mostrará posteriormente los resultados obtenidos. Véase figura 14.

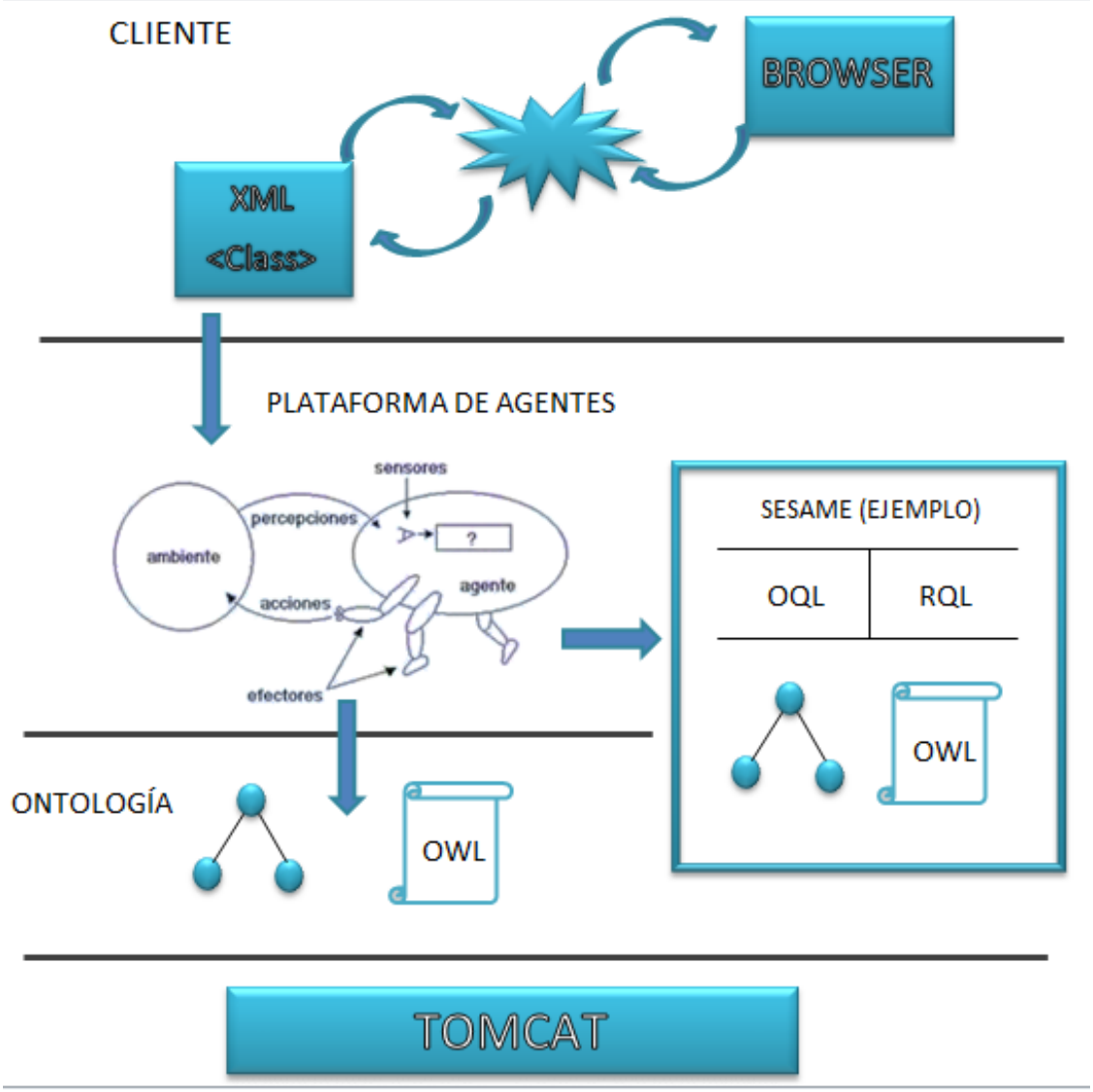

Figura 14 Arquitectura del prototipo buscador semántico

3.2.2 Plataforma de agentes

Para la arquitectura propuesta se cuenta como plataforma para los agentes la plataforma JADE, donde se instanciaran unos agentes inteligentes de software que estarán a la espera de recibir cualquier consulta. Los agentes reciben las consultas de la sección Cliente, las cuales vendrán en código XML.

El agente se encargará de extraer la consulta y convertirla teniendo en cuenta dos opciones:

- El agente se encargará de convertir esta consulta en un código con el que podrá acceder a la ontología directamente y extraer todos los términos relacionados. El código de dicha consulta puede ir en lenguajes como RQL ó OQL.

- La segunda opción es pasarle la consulta a un framework, como Sesame por ejemplo, el cual es un repositorio de ontologías, que al recibir la consulta se encargará de llevar a cabo un proceso de *parser* para convertir esta consulta en un lenguaje como OQL, para luego determinar la ontología, dentro del repositorio, a la que se le extraerán todos los términos relacionados y devolvérselos al agente.

Una vez se han extraído los términos de la consulta el agente se encarga de generar un código XML con dichos resultados, para que estos sean procesados en la parte cliente y sean mostrados en el Browser al usuario.

3.2.3 Repositorio ontologías

Para esta parte de la arquitectura como se mostraba en la sección anterior se pueden tener dos opciones, tener las ontologías en unos archivos a los cuales acceden directamente los agentes o bien utilizar un framework en el que se almacenan las ontologías y a través de éste acceder a las mismas.

3.2.4 Servidor Tomcat[43]

Toda la aplicación funcionaria sobre un servidor Apache Tomcat, el cual es un servidor web de código abierto desarrollado en el proyecto Jakarta de la Apache Software Fundation, basado en las tecnologías Java Servlet y Java Server Pages. Funciona como un contenedor de servlets dando soporte sobre estos mismos y sobre JSP. Un contenedor servlets es un marco de ejecución que invoca y maneja Servlets por cuenta del usuario.

Este servidor contiene un compilador denominado Jasper el cual se encarga de compilar JSPs convirtiéndolos en Servlets para mostrarlos al usuario final. Gestiona solicitudes y respuestas Http utilizando las opciones del servidor Apache.

Para descargas e instrucciones de instalación se puede acceder al sitio oficial a través de la dirección: http://tomcat.apache.org/

[43] JORQUERA Marcos, Administración de servicios de Internet, publicaciones Universidad de Alicante, 2008

CAPITULO 4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- Los sistemas de búsqueda actuales arrojan una cantidad de resultados basados en algunos criterios propios de la naturaleza del buscador, pero siempre dejan la tarea al usuario de tomar esos resultados y filtrarlos para poder hallar en concreto lo que se busca. Siendo en ocasiones una tarea bastante compleja cuando la cantidad de resultados encontrados son millones, como el buscador Google.

- La aplicación de semántica a los buscadores existentes permite que los resultados que se generen por una consulta sean mucho más sesgados y precisos, simplificándole al usuario la tarea de buscar entre los resultados que el buscador arroja, pasándole esta tarea a la máquina aprovechando la capacidad de cálculo de éstas.

- La cantidad de información disponible en Internet aumenta cada día de manera exponencial ofreciendo panoramas cada vez más amplios, siendo este el escenario propicio para la construcción de la web semántica, donde se ha encontrado un sinfín de aplicaciones que mejoran la experiencia del usuario a la hora de navegar en la web en búsqueda de información.

- Las ontologías se muestran como una forma coherente y completa para representar el conocimiento en un dominio especifico. El fuerte de las ontologías radica en que tienen abundante poder expresivo con el rigor del formalismo computacional de la lógica de descripción, pueden tener la extensión que sea suficiente para representar el conocimiento que se desee. El fin de las ontologías es pues llevar esta representación del conocimiento a una forma donde una maquina lo pueda procesar.

- Un modelo de dominio y su representación ontológica no es única, ya que al ser una representación del conocimiento subjetivo, una representación de un dominio específico puede ser representada de distintas maneras todas igualmente válidas. Dependerá entonces del grupo de expertos en el conocimiento específico, determinar los conceptos relevantes con los que se construya la ontología

4.2 RECOMENDACIONES Y TRABAJO FUTURO

- Extender el prototipo desarrollado a un sistema completo con todas las funcionalidades consideradas como posibles de llevar a cabo, de tal forma que permita la entrega de resultados precisos a las consultas realizadas al material bibliográfico del programa Ingeniería de Sistemas y Computación.

- Ampliar el dominio específico de la ontología CCC desarrollada en este proyecto, de tal forma que pueda abarcar la gran mayoría de temas del material bibliográfico disponible en la biblioteca Jorge Roa Martínez, extendiendo así el buscador a la mayoría de los programas existentes en la Universidad Tecnológica de Pereira.

BIBLIOGRAFÍA

[1] CAVERO BARCA, José María; VELA S., Belén y MARCOS, Esperanza. Aspectos filosóficos, psicológicos y metodológicos de la informática. Madrid, Editorial Dykinson S.L, 2005. ISBN:84-9772-749-5.

[2] CHAVARRO PORRAS, Julio César. Marco de referencia para la gestión del cambio en ontologías basados en modelos conceptuales. Doctorado en Ingeniería Énfasis en Ciencias de la Computación. Cali: Universidad del Valle. Escuela de Ingeniería de Sistemas y Computación. 2010.

[3] CORBERA DELGADO, Susana. Estudio sobre Sistemas de Gestión de Conocimiento para la Web Semántica. Ingeniera Superior Informática. Madrid: Universidad Carlos III de Madrid. Ingeniería Superior Informática. 2007.

[4] DÍEZ CARRERA, Carmen. Las industrias de la lengua: panorámica para los gestores de la información. Fesabid, 1994. ISBN 84-88699-09-3.

[5] FONER, L.N. ¿Qué es un Agente de todos modos? Un caso de estudio sociológico - Memo Agente 93-01, agentes del Grupo, del MIT Media Lab (1993).

[6] GARCÍA, Gerardo y IBÁÑEZ, Patricia. Informática Computer Scienc. Segunda edición. México D.F.: Cengage Learning Editores S.A., 2009. ISBN 10: 607-481-091-5. 1 v.

[7] GRUBER, Thomas. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. En: International Workshop on Formal Ontology, 23 de agosto de 1993.

[8] JIMÉNEZ RUIZ, Ernesto. Desarrollo de sistemas multi-agente con jade: aplicación de las ontologías. Castellón: Universidad Jaume I de Castellón. Curso de Doctorado: Robótica Avanzada: Percepción y Manipulación.

[9] JORQUERA, Marcos. Administración de servicios de Internet, publicaciones Universidad de Alicante. Alicante, Publicaciones Universidad de Alicante, 2008. ISBN: 978-84-7908-989-4.

[10] LAVÍN VILLA, Moisés. Construcción automática de diccionarios semánticos usando la similitud distribucional. Trabajo de grado Maestro en Ciencias de la Computación. México, D.F. Instituto Politécnico Nacional, 2010. 35, 37 p.

[11] MCBRIDE, Brian. Jena: A Semantic Web Toolkit. Hewlett-Packard Laboratories, Bristol, UK. 2002.

[12] MCGUINNESS, Deborah L. y NOY, Natalya F.. Desarrollo de Ontologías-101: Guía Para Crear Tu Primera Ontología. Traducido por: Erick Antezana. Stanford University, 2005.

[13] MIZOGUCHI, Riichiro. Tutorial on ontological engineering. The Institute of Scientific and Industrial Research, Osaka University.

[14] RÍO SAN JOSÉ, Jorge del. Introducción al tratamiento de datos espaciales en hidrología. España: Editorial Bubok, 2010. ISBN: 978-84-9981-141-3.

[15] ROMANA GARCÍA, María Luisa; SÁEZ, Juan Manuel y ÚCAR VENTURA, María Pilar. Traducción e interpretación: estudios, perspectivas y enseñanzas. Madrid: Editorial UNE, 2011. ISBN: 978-84-8468-373-5.

[16] SERRANO, Sebastia. La semiótica: una introducción a la teoría de los signos. España. Editorial Montesinos, 1998. ISBN:84-85859-32-4.

[17] STAAB, Steffen y STUDER, Rudi. Handbook on Ontologies. Berlín: Editorial Springer. Segunda edición, 2009. ISBN: 978-3-540-70999-2.

Sitios web

Bitext.com, NaturalFinder: el complemento ideal para su buscador:
http://www.bitext.com/ES/sol_naturalfinder.html

Hakia, descripción:
http://company.hakia.com/

Herong Yang. Object Query Language (OQL). 2008:
http://www.herongyang.com/Java-Tools/jstack-jhat-Object-Query-Language-OQL.html

IDSAI, Javier Santos Ferreras:
http://www.iit.upcomillas.es/pfc/resumenes/46e713b93f323.pdf

Infonomia, Swotti, un buscador de opiniones:
http://www.infonomia.com/if/articulo.php?id=408&if=64

Javier Echegoyen Olleta. Origen de la filosofía-Presocráticos-Sofistas y Sócrates:
http://www.e-torredebabel.com/Historia-de-la-filosofia/Filosofiagriega/Presocraticos/Ontologia.htm

Jeen Broekstra. Sesame RQL: a Tutorial. 2004:
http://www.openrdf.org/doc/rql-tutorial.html

Kaon Tool Suite sitio oficial. [Web en línea]:
http://kaon.semanticweb.org/

Lamarca Lapuente María Jesús. RDF:
http://www.hipertexto.info/documentos/rdf.htm

Lamarca Lapuente, María Jesús. Esquema RDF:
http://www.hipertexto.info/documentos/esquemas_rdf.htm

Lamarca Lapuente, María Jesús. XML:
http://www.hipertexto.info/documentos/xml.htm

María Jesús Lamarca Lapuente, Hipertexto:
http://www.hipertexto.info/documentos/ontologias.htm

María Jesús Lamarca Lapuente, Hacia la Web Semántica:
http://www.hipertexto.info/documentos/web_semantica.htm

Miguel Angel Alvarez, Qué es JSP:
http://www.desarrolloweb.com/articulos/831.php

Ontology Engineer ingGroup. [Web en línea]:
http://mayor2.dia.fi.upm.es/oeg-upm/index.php/es/downloads/60-webode

Pérez Chantal, Estudios de Lingüística del Español:
http://elies.rediris.es/elies18/index.html

Pergamino virtual, Definiciones:
http://www.pergaminovirtual.com.ar/definicion/Buscador.html

Políticas y principios Google:
http://www.google.com/policies/

Protégé sitio oficial [web en línea]:
http://protege.stanford.edu/overview/index.html

Semántica:
http://www.profesorenlinea.cl/castellano/Semantica1.htm

Universidad de Valladolid, Departamento de informática:
http://www.infor.uva.es/~sblanco/Tesis/Ontolog%C3%ADas.pdf

Uso de ontologías en tareas de recupero de información, Tallarico Marcelo, 2008:
http://es.scribd.com/doc/55621955/4/Definiciones-de-ontologia

Uso de ontologías en tareas de recupero de información, Tallarico Marcelo, 2008:
http://es.scribd.com/doc/55621955/4/Definiciones-de-ontologia
http://www-lipn.univ-paris13.fr/~szulman/TERMINAE.html

VisualVM. Analyzing a Heap Dump Using Object Query Language (OQL):
http://visualvm.java.net/oqlhelp.html

W3C, Lenguaje de Ontologías Web (OWL), 2004:
http://www.w3.org/2007/09/OWL-Overview-es.html#s1.3

Wolfram blog, wólfram:
http://blog.wolfram.com/?s=wolfram+alpha&submit.x=0&submit.y=0

ANEXOS

Anexo A. Archivo CCC. owl generado en Protégé en código OWL.

Anexo B. Archivo CCC. repository generado en Protégé.