SISTEMA MULTIAGENTE PARA LA OPTIMIZACION DE BUSQUEDAS EN EL DOMINIO MAMPOSTERIA ESTRUCTURAL BASADO EN TECNICAS DE WEB SEMANTICA

OLGA LUCIA RODRIGUEZ MENESES

CAROL ROSANNA LOAIZA SALAZAR

Anexos

Director: Ing. Juan Carlos Vidal Rojas

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERIA ELECTRONICA Y TELECOMUNICACIONES, DEPARTAMENTO DE SISTEMAS

POPAYAN, ABRIL DE 2005

CONTENIDO

1. METODOLOGIAS PARA DESARROLLO Y CONSTRUCCION DE ONTOLOGIAS 4

1.1 TOVE (Toronto Virtual Enterprise) 4

1.2 METODOLOGIA ENTERPRISE MODEL 5

1.3 METHONTOOGY 5

1.4 KBSI IDEF5 6

2 HERRAMIENTAS DE DESARROLLO DE ONTOLOGIAS 7

2.1 APOLLO 7

2.2 LINKFACTORY 8

2.3 OILED 8

2.4 ONTOEDIT 9

2.5 ONTOLINGUA SERVER 9

2.6 PROTÉGÉ 10

2.7 WEBODE 11

2.8 WEBONTO 11

3 LENGUAJES PARA LA DEFINICION DE ONTOLOGIAS 12

3.1 RDF y RDF Schema 12

3.2 OIL (ONTOLOGY INFERENCE LAYER) 12

3.3 DAML (DARPA AGENT MARKUP LANGUAGE) 13

3.4 DAML+OIL 13

3.5 OWL (WEB ONTOLOGY LANGUAGE) 14

4. DOCUMENTACIÓN DE LA ONTOLOGIA DEL DOMINIO: 14

4.1 DEFINICIÓN DE CLASES: 15

4.2 DEFINICIÓN DE PROPIEDADES: 29

5. DOCUMENTACIÓN DE LA ONTOLOGIA DE CONSULTAS: 30

5.1 DEFINICIÓN DE CLASES: 30

5.2 DEFINICIÓN DE PROPIEDADES: 30

6 ANALISIS DE METODOLOGIAS PARA EL MODELADO DE SISTEMAS MULTIAGENTE. 31

6.1 METODO DE BURMEISTER. 31

6.2 MAS-CommonKADS 31

6.3 MaSE (Multiagent System Engineering) 31

6.4 GAIA 32

6.5 AUML 32

7 SISTEMAS DE BUSQUEDA EN INTERNET 32

7.1 EVOLUCION DE LOS BUSCADORES EN LA WEB. 32

7.1.1 Tipos de Buscadores 33

7.1.1.1 Índices Temáticos o Directorios 33

7.1.1.2 Motores de búsqueda 35

7.1.1.3 Metabuscadores 36

7.1.1.4 Técnicas de filtrado de Información 38

8. REFERENCIAS 39

1. METODOLOGIAS PARA DESARROLLO Y CONSTRUCCION DE ONTOLOGIAS

Al finalizar el estudio de varias metodologías, el grupo de trabajo observó dos importantes corrientes en los procesos de desarrollo de las ontologías. La primera de ellas presenta modelos de desarrollo por etapas y la segunda a través de modelos de desarrollo de prototipos. Ambas corrientes presentan ventajas y desventajas y su utilización dependerá de la claridad con que se hayan podido definir desde el inicio los requerimientos para la construcción de la misma.
Cuando el propósito y los requerimientos están claramente definidos desde el inicio del desarrollo, será más apropiado utilizar modelos de desarrollo por etapas, de lo contrario será más adecuado utilizar un modelo de prototipos (caso de interés).

No existe una metodología de facto para el desarrollo y mantenimiento de ontologías [NOY, 2000]. Sin embargo, existe un conjunto de sugerencias a considerar, surgidas de la experiencia de investigadores en el desarrollo de proyectos que las utilizaron. A continuación se presentan algunas de ellas:

1.1 TOVE (Toronto Virtual Enterprise)

Basado en la experiencia en el desarrollo de TOVE, se desarrolló la siguiente aproximación metodológica:
Definición de escenarios de motivación: El punto inicial es un conjunto de problemas encontrados en un desarrollo particular, el cual se presenta normalmente en forma de narración de problemas o mediante ejemplos.
Preguntas informales de competencia: Describen los requerimientos de la ontología, basados en el escenario de motivación y presentados en forma de preguntas que la ontología debe ser capaz de responder. Esta fase actúa como una evaluación de los propósitos realizados en fases anteriores.
Especificación de la terminología: Los objetos, atributos y relaciones de la ontología son formalmente especificados, normalmente utilizando lógica de primer orden.
Preguntas formales de competencia: Los requerimientos de la ontología son formalizados en términos de la terminología formalmente definida.
Especificación de axiomas: los axiomas que especifican la definición de términos y restricciones sobre sus interpretaciones son dados en lógica de primer orden.

Teoremas de Amplitud: Un estado de evaluación que asegura la competencia de la ontología mediante la definición de las condiciones bajo las cuales la solución de las preguntas de competencia está completa.

1.2 METODOLOGIA ENTERPRISE MODEL

Se trata de una metodología basada en la experiencia obtenida durante el desarrollo del proyecto Enterprise [USC, 1996], la cual se ha refinado hasta definir los siguientes pasos:

Identificación del propósito: determinar el nivel de formalidad al que la ontología debe ser descrita.

Identificación de Alcance: se genera una "Especificación" que defina completamente el rango de información que la ontología debe caracterizar. Esto puede hacerse usando escenarios de motivación y preguntas informales de competencia, como en TOVE o mediante "tormentas de ideas y ajuste", para producir una lista de conceptos potencialmente relevantes y eliminar conceptos irrelevantes y sinónimos.

Formalización: Crear el "código", las definiciones formales y axiomas de términos en la especificación

Evaluación Formal: El criterio usado puede ser general, o específico a una ontología particular, tal como un chequeo entre el propósito y las preguntas de competencia. Esta evaluación causa una revisión de los resultados obtenidos en las etapas 2 y 3.

Al igual que la metodología TOVE, Enterprise Model distingue entre las fases formales e informales del desarrollo de ontologías. La fase informal involucra la identificación de conceptos claves y la generación de definiciones textuales para los conceptos y sus relaciones, mediante la utilización de técnicas de adquisición del conocimiento existentes.

1.3 METHONTOOGY

Methontology inicia con la identificación de las siguientes actividades involucradas en el desarrollo de una ontología.

Especificación: Identifica el propósito de la ontología, incluyendo los posibles usuarios, escenarios de uso, grado de formalidad requerida, etc. Y el alcance de la ontología incluyendo el conjunto de términos que serán representados, sus características la granularidad requerida. De esta fase se obtiene un documento de especificación de la ontología, escrito en lenguaje natural.

Adquisición del conocimiento: Esta fase se realiza en paralelo con la anterior, no es restrictiva en cuanto a la fuente de conocimiento y método de respuesta utilizados, sin embargo, se utilizan específicamente entrevistas con el experto del dominio y el análisis de textos.

Conceptualización: Se identifican los términos del dominio como conceptos, instancias, verbos, relaciones o propiedades y cada una de ellos son representados utilizando una representación informal aplicable.
Integración: Se incorporan estándares para obtener alguna uniformidad entre ontologías y definiciones con otras ontologías.
Implementación: Se representa formalmente la ontología en un lenguaje determinado, por ejemplo Ontolingua.
Evaluación: Methontology hace mucho énfasis en esta fase. Se utilizan técnicas basadas en la validación y verificación para buscar inconsistencias, redundancia e incompletéz de la ontología.
Documentación: Colección de los documentos resultantes de otras actividades.

En methontology el ciclo de vida de una ontología, se basa en el refinamiento de un prototipo, el cual pasa por los siguientes estados: especificación, conceptualización, formalización, integración e implementación. Finalmente la ontología entra en una fase de mantenimiento. La evaluación y documentación se realizan durante el todo el ciclo de vida [COR, 2004].

1.4 KBSI IDEF5

La metodología IDEF5 fue diseñada para asistir en la creación, modificación y mantenimiento de ontologías. Dado que el desarrollo de ontologías es necesariamente un proceso abierto, se sugiere que no es prudente adoptar una metodología rígida para llevarlo a cabo, por lo tanto, la metodología IDF5 es un procedimiento general y un conjunto de guías para su desarrollo:
Organización y alcance: Se establece el alcance, punto de vista y contexto para el desarrollo del proyecto. El propósito de esta fase es proveer un conjunto de “criterios de completitud” para la ontología, incluyendo objetivos y requerimientos. El alcance define los límites de la ontología y especifica las partes del sistema que deberían incluirse y excluirse.
Recolección de datos: Los datos e información necesarios para el desarrollo de la ontología se adquieren utilizando técnicas típicas de Adquisición de Conocimiento (KA), tales como análisis de protocolos y entrevistas con el experto del dominio.
Análisis de los datos: La ontología se extrae a partir de los resultados de la recolección de datos. Se hace una lista de los objetos de interés en el dominio, seguida de la identificación de los objetos y sistemas internos dentro de los límites de la ontología.
Desarrollo inicial de la ontología: Se desarrolla una ontología preliminar que contiene Proto-conceptos como descripciones iniciales de clases, propiedades y relaciones.

Refinamiento y validación de la ontología: Los Proto-conceptos son refinados y validados en forma iterativa. Este es esencialmente un proceso de validación deductiva mediante el cual los diferentes componentes de la ontología son "instanciados" con datos reales y el resultado de dicha instanciación es comparada con la estructura de la ontología.

La metodología IDEF5 puede verse como un modelo de prototipos basado en el refinamiento de las salidas producidas. Este refinamiento gradual se facilita por medio del uso de dos lenguajes de representación: Lenguaje de esquematización y Lenguaje de elaboración.

La ontología inicial es definida en un lenguaje de esquematización, una notación gráfica usada para expresar las formas más comunes de información ontológica durante el proyecto. El lenguaje esquemático es usado principalmente para permitir una adecuada comunicación entre el experto del dominio y el desarrollador de la ontología.

Esta representación inicial se analiza y remodela dentro de un lenguaje de elaboración más estructurado basado en KIF (Knowledge Interchange Format).

2 HERRAMIENTAS DE DESARROLLO DE ONTOLOGIAS

En esta sección, se describen las principales características y funcionalidades de algunas de las herramientas y entornos disponibles para la construcción de ontologías, apropiadas ya sea para generar una ontología totalmente nueva o para reutilizar ontologías ya existentes.

2.1 APOLLO

Apollo es una herramienta de desarrollo de ontologías diseñada buscando para el usuario, la mayor facilidad para el entendimiento de la sintaxis y del entorno.

Soporta las primitivas básicas de modelado del conocimiento: ontologías, clases, instancias, funciones y relaciones; además, hace chequeo de consistencia sobre la ontología, mientras se está editando, detectando por ejemplo, el uso de clases que no están definidas.

Apollo está implementado en Java y tiene su propio lenguaje interno para almacenar las ontologías, pero permite exportarlas hacia diferentes lenguajes de representación según lo requiera el usuario.

La versión 1.1 posee un plugin para exportar la ontología a lenguaje OWL, utilizando sintaxis RDF o XML [KOS, 2003].

2.2 LINKFACTORY

LinkFactory es un sistema de gestión de ontologías formales, desarrollado por Lenguaje & Computing nv, diseñado para construir y gestionar ontologías muy grandes y complejas independientes del lenguaje.

El sistema LinkFactory® consiste de dos componentes principales, desarrollados en Java: LinkFactory® Server, LinkFactory® Workbench.

A nivel del servidor, LinkFactory® almacena los datos en una base de datos relacional, que se accede a través de un conjunto de funciones "naturales". Estas funciones se accedan por los clientes software a través de una API estandarizada que permite construir aplicaciones sobre bases de datos semánticas sin requerir un conocimiento extenso de la estructura interna de la base de datos. Este componente permite manejar la concurrencia de múltiples usuarios, independiente de la plataforma.

El LinkFactory® Workbench permite a los usuarios modelar y navegar varias ontologías, enlazándolas en forma dinámica. Posee una serie de mecanismos para manejar el modelado de las ontologías: versionamiento, seguimiento a los usuarios, jerarquias, detección de sibling, jerarquias de tipos de enlaces, etc.

El principal inconveniente es que LinkFactory® es una herramienta comercial [LAG, 2003].

2.3 OILED

OILEd es un editor gráfico de ontologías, desarrollado en la Universidad de Manchester, que permite a los usuarios construir ontologías usando DAML+OIL.

El modelo de conocimiento de OILEd, se basa en el de DAML+OIL, sin embargo, este es extendido mediante el uso de una representación tipo frames para su modelado. De esta manera, OILEd ofrece una forma familiar de trabajo, al tiempo que ofrece soporte para la rica expresividad de DAML+OIL.

Las clases de definen en términos de sus superclases, de las restricciones sobre sus propiedades y utilizando axiomas adicionales permite capturar relaciones adicionales entre las clases. Este modelo de conocimiento expresivo, permite el uso de descripciones compuestas complejas.

La principal tarea de IOLE es la edición de ontologías o esquemas, en oposición a la adquisición del conocimiento o a la construcción de grandes bases de instancias.

Un aspecto clave del comportamiento de OILEd, es el uso del razonador FaCT, para clasificar ontologías y chequear la consistencia a través de una traducción de DAML+OIL a la lógica descriptiva SHIQ. Esto permite al usuario describir sus clases de

la ontología y permitir que el razonador determine la ubicación apropiada en la jerarquia, para su definición.
OILEd está implementado en Java, y está disponible para ser descargado desde el sitio Web de OILEd [BEC, 2002].

2.4 ONTOEDIT

OntoEdit es un entorno para construcción de ontologías, que soporta el desarrollo y mantenimiento de ontologías utilizando un entorno gráfico. OntoEdit está construido sobre un poderoso modelo de ontologías interno. Este paradigma soporta el modelado mediante un lenguaje neutral de representación, tanto como sea posible para conceptos, relaciones y axiomas. Algunas vistas gráficas dentro de las estructuras contenidas en la ontología, soportan el modelado de las diferentes fases del ciclo de ingeniería de las ontologías.
La herramienta permite a los usuarios editar una jerarquía de conceptos o clases. Estos conceptos puedes ser abstractos o concretos, lo cual indica si es posible o no generar instancias de ellos. Un concepto puede tener varios nombres, lo cual esencialmente es una forma de definir sinónimos para el concepto. La herramienta permite reorganizar los conceptos en la jerarquía de manera similar a la funcionalidad “copy-and-paste”.
La herramienta consiste de un framework flexible basado en plugins, lo cual permite extender fácilmente su funcionalidad en forma modularizada y adaptarla a diferentes escenarios de uso.
Hasta el año 2002 OntoEdit ofreció a los usuarios una versión libre y una versión profesional, después de este año, OntoEdit esta disponible solo dentro del paquete comercial OntoEstudio [ONT, 2002].

2.5 ONTOLINGUA SERVER

El servidor Ontolingua, es un conjunto de herramientas y servicios que dan soporte a la construcción de ontologías compartidas entre grupos distribuidos, desarrollada por el Laboratorio de Sistemas del Conocimiento (KSL) en la Universidad de Stanford. La arquitectura del servidor Ontolingua, provee acceso a una librería de ontologías, traductores a diferentes lenguajes de implementación, y un editor para crear y navegar ontologías. Los editores remotos pueden navegar y editar las ontologías, y aplicaciones locales o remotas pueden acceder cualquiera de las ontologías en la librería utilizando el protocolo OKBC (Open Knowledge Based Connectivity) [RIS, 2002].

2.6 PROTÉGÉ

Protégé provee un entorno gráfico e interactivo para el desarrollo de ontologías y bases de conocimiento. Desarrollado en la Universidad de Stanford.

Los desarrolladores de ontologías pueden acceder información relevante rápidamente cada vez que lo necesiten, manipular directamente la ontología y navegar a través de la jerarquía de clases utilizando árboles de control.

El modelo de conocimiento de Protégé es compatible con OKBC. Incluye soporte para clases y jerarquía de clases con herencia múltiple, plantillas y slots propios, especificación de restricciones predefinidas y arbitrarias las cuales incluyen valores permitidos, cardinalidad, valores por defecto, slots inversos, metaclases y jerarquía de metaclases.

Dos características importantes que distinguen Protégé de la mayoría de entornos de edición de ontologías son su escalabilidad y extensibilidad.

Una de las principales ventajas de la arquitectura de Protégé es que el sistema está construido de forma modular. Esta arquitectura basada en componentes, permite a los desarrolladores, adicionar nueva funcionalidad mediante la creación de plugins apropiados. La librería de plugins de Protégé contiene contribuciones de desarrolladores alrededor del mundo y está clasificada en tres categorías:

- Backends: Permiten a los usuarios almacenar e importar bases de conocimiento en varios formatos
- Slot widgets: Son usados para desplegar y editar valores de slots o combinaciones de ellos, en un dominio específico o para tareas específicas.
- Tab plugins: Son aplicaciones basadas en conocimiento, usualmente enlazadas con las bases de conocimiento de Protégé.

Los plugins estándar actuales, incluyen soporte para el almacenamiento e importación de ontologías en RDF Schema, XML y XML Schema, y OWL.

Los widgets disponibles, incluyen componentes de interfaz de usuario para desplegar imágenes, audio y video.

El tipo más popular de plugins son los tab plugins disponibles, que proveen capacidades para visualización avanzada, mezcla de ontologías, gestión de versiones e inferencia entre otros.

El plugin OWL, es una extensión de Protégé que soporta el Lenguaje de implementación de Ontologías para la Web (OWL) [VEN, 2004].

Este plugin permite:

- Cargar y grabar ontologías OWL y RDF
- Editar y visualizar clases, propiedades y reglas SWRL

- Definir características de clases lógicas como expresiones OWL
- Editar instancias OWL para el marcado Web Semántico.

2.7 WEBODE

WebODE es un entorno de trabajo para el desarrollo de ontologías, que provee una variedad de servicios relacionados y da soporte a la mayoría de actividades envueltos en el proceso de desarrollo de ontologías.

Está construido sobre un servidor de aplicaciones base, que provee alta extensibilidad y usabilidad para permitir una fácil adición de nuevos servicios y el uso de servicios existentes.

Las ontologías WebODE son representadas utilizando un modelo de conocimiento muy amplio, basado en el conjunto de referencias de representaciones inmediatas de la metodología METHONTOLOGY, el cual incluye componentes de la ontología tales como conceptos, instancias y atributos de clases, particiones, relaciones binarias, relaciones predefinidas (taxonomias y parte-de), axiomas, reglas, restricciones y referencias bibliográficas. Además permite la importación de términos desde otras ontologías.

Las ontologías en WebODE se almacenan en una base de datos relacional, sin embargo, WebODE provee una API orientada a servicios que permite acceder la ontología y facilita la integración con otros sistemas, su traducción a una variedad de lenguajes de implementación (RDF(S), OIL, DAML+OIL, CARIN y FLogic) y su exportación del sistema hacia lenguajes y sistemas como Java y Jess.

WebODE soporta la edición colaborativa de ontologías mediante la noción de grupos de usuarios, mecanismos de acceso a las ontologías desarrolladas y mecanismos de sincronización para permitir que varios usuarios puedan editar adecuadamente la misma ontología.

La herramienta provee capacidades de chequeo de restricciones de tipo, de valores numéricos, cardinalidad y verificación de consistencia de la taxonomía de clases [GOM, 2003].

2.8 WEBONTO

WebOnto es una herramienta desarrollada en el the Knowledge Media Institute (KMI) de la Open University (England).

Soporta la creación, edición y navegación colaborativa de ontologías, representadas en el lenguaje de modelado del conocimiento OCML [DOM, 1999].

Sus principales características son:

- Gestión de ontologías usando una interfaz gráfica

- Generación automática de instancias
- Utilización de formularios de edición para la definición de clases
- Manejo de Herencia
- Chequeo de consistencia
- Soporte para el trabajo colaborativo, mediante mecanismos broadcast/recepción de anotaciones
- Librería con más de 100 ontologías, accesible a través de WebOnto.

3 LENGUAJES PARA LA DEFINICION DE ONTOLOGIAS

En esta sección, se describen las principales características de algunos lenguajes disponibles y conocidos para la definición de ontologías.

3.1 RDF y RDF Schema

RDF es un lenguaje de etiquetado, creado mediante sintaxis XML, que define un modelo de datos para describir recursos en la web, mediante enunciados o asertos en forma de tripletas sujeto-predicado-objeto.

Por otro lado, RDF Schema es un vocabulario RDF que nos permite describir recursos mediante una orientación a objetos similar a la de muchos lenguajes de programación como Java. Para ello, proporciona un mecanismo para definir clases, objetos y propiedades; relaciones entre clases y propiedades; y restricciones de dominio y rango sobre las propiedades.

Mientras que la relación entre XML y XML Schema es una relación de control sintáctico, la relación entre RDF y RDFS es de control semántico.

Con RDFS podemos describir jerarquías de clases, tales como ontologías simples, sobre las que poder realizar consultas y razonamiento automático. Aún así, RDFS no es lo suficientemente expresivo para representar ontologías de la complejidad que necesita la Web Semántica, ya que un agente inteligente sólo podría realizar inferencias sobre la herencia de propiedades, pues RDFS no permite declarar axiomas [ALC, 2004].

3.2 OIL (Ontology Inference Layer)

OIL fue desarrollado en el marco del proyecto On-To-Knowledge [DEC, 2002] y es el primer lenguaje de representación de ontologías basado en estándares W3C: tiene

sintaxis XML y está definido como una extensión de RDFS. El modelo utilizado por OIL para la representación del conocimiento lo ha heredado, por una parte de la Lógica Descriptiva (declaración de axiomas o reglas) y, por otra, de los sistemas basados en frames (taxonomía de clases y atributos). OIL se encuentra estructurado en varias capas de sublenguajes. La capa base o núcleo de OIL coincide plenamente con RDFS, a excepción de la reificación, y cada una de las capas superiores añade funcionalidad y complejidad a su capa subyacente. Entre las limitaciones de OIL podíamos destacar las siguientes: no ofrece la posibilidad de sobrescritura de valores heredados de una superclase, presenta falta de expresividad en la declaración de axiomas (reglas) y no soporta dominios concretos (ej.: números enteros, cadenas de caracteres, etc.) [ALC, 2004]

3.3 DAML (DARPA Agent Markup Language)

El DAML (Darpa Agent Markup Language) es una extensión del XML y el RDF. El DAML provee un rico juego de construcciones con el cual se pueden crear ontologías y aumentar la información para que sea legible y comprensible por las máquinas. En diciembre de 2000 el DAML pasó a denominarse DAML+OIL, debido a la revisión de las especificaciones del lenguaje [HEN, 2001]

3.4 DAML+OIL

DAML+OIL es uno de los lenguajes de representación de ontologías más reciente, fruto de la cooperación entre los grupos de trabajo de OIL y DARPA (US Defense Advanced Research Projects Agency), quienes anteriormente habían desarrollado el lenguaje DAML con la finalidad de extender el nivel de expresividad de RDFS. Por tanto DAML+OIL es un lenguaje que unifica estos dos lenguajes, aunando esfuerzos en el camino de la construcción de un lenguaje estándar para la definición de ontologías.

Aunque DAML+OIL hereda muchas de las características de OIL, difiere en algunas. De hecho, se aleja del modelo basado en frames de OIL. A nivel práctico, DAML+OIL demuestra ser más útil como soporte para ontologías que, por ejemplo, RDF Schema, sin embargo aún tiene ciertas carencias como formato de intercambio y modelado de ontologías. Si bien la complejidad técnica en DAML+OIL no es un problema, lo que se puede demostrar por el gran número de herramientas y aplicaciones desarrolladas con soporte para este lenguaje en sus escasos años de vida, su complejidad conceptual (dificultad de uso y aprendizaje por parte del autor de ontologías) sí lo es [HEN, 2001]

3.5 OWL (Web Ontology Language)

El 10 de febrero de 2004 el Consorcio World Wide Web publica las recomendaciones de dos tecnologías clave para la Web Semántica, la Infraestructura para la descripción de Recursos (RDF) revisada y el Lenguaje de Ontologías Web (OWL) [W3C, 2004].
OWL: Lenguaje de Ontologías Web proporciona un lenguaje para la definición de Ontologías estructuradas, basadas en la Web, que ofrece una integración e interoperabilidad de datos más rica entre comunidades descriptivas. Los lenguajes anteriores se utilizaron para desarrollar herramientas y ontologías para comunidades de usuarios específicas (particularmente en las ciencias y en aplicaciones de comercio electrónico de compañías específicas), lo que constituye un problema de compatibilidad con la arquitectura actual de la World Wide Web en general, y en particular de la Web Semántica.
OWL utiliza URIs para fijar nombres y la infraestructura para descripciones en la Web proporcionada por RDF para agregar las siguientes capacidades a las ontologías:

- Habilidad de ser distribuida por muchos sistemas
- Escalabilidad a las necesidades de la Web
- Compatibilidad con estándares Web para la accesibilidad y la internacionalización
- Apertura y extensibilidad

OWL se construye sobre RDF y RDF Esquema y añade más vocabulario para la descripción de clases y propiedades: entre otras, relaciones entre clases, cardinalidad, igualdad, mayor riqueza de tipos en las propiedades, características de propiedades y clases enumeradas.

OWL incluye también OWL Lite, versión reducida de OWL, cuyo objetivo es ser adoptada rápida y ampliamente por los desarrolladores, para lo que concentra las características más comúnmente usadas de OWL y DAML+OIL. En este sentido, OWL se estructura en capas que, al igual que en OIL, difieren en complejidad, para adaptarse a las necesidades de nivel de expresividad requerido por las diferentes comunidades de usuarios (autores de ontologías, programadores, etc.) y los diferentes tipos de aplicaciones (agentes, motores de búsqueda, etc.) [ALC, 2004].

4. DOCUMENTACIÓN DE LA ONTOLOGIA DEL DOMINIO:

Se presenta a continuación la documentación correspondiente a la ontología del dominio proceso constructivo Mampostería Estructural:

4.1 DEFINICIÓN DE CLASES:

Clase:	Construcción_Mampostería
Extiende de:	Thing
Subclases directas:	Elementos componentes, Materiales, Tipos Mampostería

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
Definición		String		0:*	
Sinónimo		String		0:*	

Clase:	Elementos_Componentes
Extiende de:	Construcción_Mampostería
Subclases directas:	Acabados, Mortero, Muro, Refuerzo, Unidad_Mamposteríag

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
Definición		String		0:*	
Sinónimo		String		0:*	

Clase:	Acabados
Extiende de:	Elementos_Componentes
Subclases directas:	Acabados Exteriores, Acabados Interiores.

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
Definición		String		0:*	
Sinónimo		String		0:*	

Clase:	Acabados Exteriores
Extiende de:	Acabados
Subclases directas:	Enchape no Cerámico, Pintura, Recubrimiento Texturizado, Revoque Plástico, Revoque tradicional.

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
Definición		String		0:*	
Sinónimo		String		0:*	

Clase:	Enchape no Cerámico
Extiende de:	Acabados Exteriores
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
Definición		String		0:*	
Sinónimo		String		0:*	

Clase:	Pintura.
Extiende de:	Acabados Exteriores, Acabados Interiores.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
Definición		String		0:*	
Sinónimo		String		0:*	

Clase:	Recubrimiento Texturizado.
Extiende de:	Acabados Exteriores.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
Definición		String		0:*	
Sinónimo		String		0:*	

Clase:	Revoque Plástico.
Extiende de:	Acabados Exteriores, Acabados Interiores.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
Definición		String		0:*	
Sinónimo		String		0:*	

Clase:	Revoque Tradicional.
Extiende de:	Acabados Exteriores, Acabados Interiores.
Subclases directas:	

Propiedades

Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
Definición		String		0:*	
Sinónimo		String		0:*	

Clase:	Acabados Interiores.
Extiende de:	Acabados.
Subclases directas:	Enchape Cerámico, Estuco Plástico, Estuco Tradicional, Pintura, Revoque Plástico, Revoque Tradicional.

Propiedades

Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
Definición		String		0:*	
Sinónimo		String		0:*	

Clase:	Enchape Cerámico.
Extiende de:	Acabados Interiores.
Subclases directas:	

Propiedades

Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
Definición		String		0:*	
Sinónimo		String		0:*	

Clase:	Estuco Plástico.
Extiende de:	Acabados Interiores.
Subclases directas:	

Propiedades

Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
Definición		String		0:*	
Sinónimo		String		0:*	

Clase:	Estuco Tradicional.
Extiende de:	Acabados Interiores.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
Definición		String		0:*	
Sinónimo		String		0:*	

Clase:	Mortero
Extiende de:	Elementos Componentes.
Subclases directas:	Mortero Inyección, Mortero Pega.

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
Definición		String		0:*	
Sinónimo		String		0:*	

Clase:	Mortero Inyección
Extiende de:	Mortero.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	
control_calidad		String		0:*	

Clase:	Muro.
Extiende de:	Elementos Componentes.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	
tipos		String		0:*	
aparejo		String		0:*	

Clase:	Refuerzo.
Extiende de:	Elementos Componentes.
Subclases directas:	Conectores, Refuerzo Horizontal, Refuerzo Vertical

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	
características		String		0:*	
tipos		String		0:*	

Clase:	Refuerzo Horizontal.
Extiende de:	Refuerzo.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	
características		String		0:*	
tipos		String		0:*	

Clase:	Refuerzo Vertical.
Extiende de:	Refuerzo.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	
características		String		0:*	
tipos		String		0:*	

Clase:	Unidad Mampostería.
Extiende de:	Elementos Componentes.
Subclases directas:	Bloque Arcilla, Bloque Concreto

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	
características		String		0:*	
tipos		String		0:*	
control_calidad		String		0:*	

Clase:	Bloque Arcilla.
Extiende de:	Unidad Mampostería.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	
características		String		0:*	
tipos		String		0:*	
control_calidad		String		0:*	

Clase:	Bloque Concreto.
Extiende de:	Unidad Mampostería.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	
características		String		0:*	
tipos		String		0:*	
control_calidad		String		0:*	

Clase:	Materiales.
Extiende de:	Construcción Mampostería.
Subclases directas:	Acero refuerzo, Aditivos, Agregados, Cemento.

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Acero Refuerzo.
Extiende de:	Materiales.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	
tipos		String		0:*	
control_calidad		String		0:*	

Clase:	Aditivos.
Extiende de:	Materiales.
Subclases directas:	Acelerante, Impermeabilizante, Plastificante, Retardador Fraguado.

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Acelerante.
Extiende de:	Aditivos.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Impermeabilizante.
Extiende de:	Aditivos.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Plastificante.
Extiende de:	Aditivos.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Retardador Fraguado.
Extiende de:	Aditivos.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Agregado.
Extiende de:	Materiales.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	
control_calidad		String		0:*	

Clase:	Cemento.
Extiende de:	Materiales.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	
control_calidad		String		0:*	

Clase:	Tipos Mampostería
Extiende de:	Construcción Mampostería
Subclases directas:	Mampostería Arquitectónica, Mampostería Estructural, Mampostería no Estructural, Mampostería.

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Mampostería.
Extiende de:	Tipos Mampostería.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	
características		String		0:*	
tipos		String		0:*	
consideraciones_constructivas		String		0:*	

Clase:	Mampostería Arquitectónica.
Extiende de:	Tipos Mampostería
Subclases directas:	Mampostería Cubierta, Mampostería Expuesta, Mampostería Pegada, Mampostería Tope.

Propiedades

Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Mampostería Cubierta.
Extiende de:	Mampostería Arquitectónica.
Subclases directas:	

Propiedades

Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Mampostería Expuesta.
Extiende de:	Mampostería Arquitectónica.
Subclases directas:	

Propiedades

Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Mampostería Pegada.
Extiende de:	Mampostería Arquitectónica.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Mampostería Tope.
Extiende de:	Mampostería Arquitectónica.
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Mampostería Estructural
Extiende de:	Tipos Mampostería
Subclases directas:	Mampostería Cavidad Reforzada, Mampostería Muros Confinados, Mampostería Muros Diafragma, Mampostería Reforzada

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Mampostería Cavidad Reforzada
Extiende de:	Mampostería Estructural
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Mampostería Muros Confinados
Extiende de:	Mampostería Estructural
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Mampostería Muros Diafragma
Extiende de:	Mampostería Estructural
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Mampostería Reforzada.
Extiende de:	Mampostería Estructural
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Mampostería no Estructural
Extiende de:	Tipos Mampostería
Subclases directas:	Mampostería Parcialmente Reforzada, Mampostería Simple.

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Mampostería Parcialmente Reforzada
Extiende de:	Mampostería no Estructural
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

Clase:	Mampostería Simple
Extiende de:	Mampostería no Estructural
Subclases directas:	

Propiedades					
Nombre	***Documentación***	***Tipo***	***Valores Permitidos***	***Cardinalidad***	***Valores por defecto***
definición		String		0:*	
sinónimo		String		0:*	

4.2 DEFINICIÓN DE PROPIEDADES:

A continuación se presenta la definición de propiedades (atributos) sobre las clases del modelo del dominio:

Propiedades		
Nombre	***Documentación***	***Clases Relacionadas***
Aparejo	Se refiere a la manera como se disponen o traban las unidades de mamposteria en un muro lo cual se manifiesta por el patrón que siguen sus caras	Muro
Características	Se refiere a las propiedades especiales que poseen algunas de las clases modeladas y que las caracterizan	Refuerzo Unidad_mamposteria Cemento Mamposteria
Componentes	Se refiere a la materia prima necesaria para la elaboración del cemento, consistente en una mezcla de materiales calcáreos (piedra caliza) y arcillosos.	Cemento

Consideraciones constructivas	Se refiere a las controles y procesos constructivos que deben seguirse en la elaboración de estructuras de mampostería, especialmente en lo relacionado al levantamiento de muros	Mampostería Muro
Control_calidad	Se refiere a los controles que deben efectuarse sobre los diferentes materiales y elementos componentes de la mamposteria con el fin de asegurar una adecuada calidad de los mismos	Mortero_inyeccion Mortero_pega Unidad_mamposteria Acero_refuerzo Agregado Cemento
Definicion	Corresponde a la definición de cada uno de los conceptos modelados en el dominio	Construccion_ mamposteria
Preparacion_ mortero	Se refiere a los diferentes tipos de preparación del mortero de pega para mamposteria	
Sinonimo	Corresponde a los diferentes términos con que se conocen algunos de los conceptos modelados, de acuerdo a la region, pais, etc	Construccion_ mamposteria
Tipos	Se refiere a los diferentes tipos que conforman algunos de los conceptos del dominio, como por ejemplo: tipos de refuerzo, tipos de muro, etc	Refuerzo Muro Cemento Acero_refuerzo Unidad_mamposteria Mampostería Agregado

5. DOCUMENTACIÓN DE LA ONTOLOGIA DE CONSULTAS:

Se presenta a continuación la documentación correspondiente a la ontología de consultas, desarrollada con el fin de representar las consultas que serán lanzadas al motor de búsqueda utilizado, para recuperar los documentos que serán analizados por el sistema multiagente.

5.1 DEFINICIÓN DE CLASES:

Teniendo en cuenta que la ontología de consultas es una extensión de la ontología del dominio y por lo tanto parten del mismo modelo del dominio, la definición de clases corresponde a la definición de clases de la ontología del dominio.

5.2 DEFINICIÓN DE PROPIEDADES:

Para la implementación de la ontología de consultas se reemplazó la definición de propiedades por una única propiedad denominada consulta, la cual permite almacenar una serie de consultas previamente definidas para cada clase del dominio.

Propiedades		
Nombre	***Documentación***	***Clases Relacionadas***
Consulta	Corresponde a consultas predefinidas que serán utilizadas por el motor de búsqueda para recuperar los documentos que serán analizados por el sistema	

6 ANALISIS DE METODOLOGIAS PARA EL MODELADO DE SISTEMAS MULTIAGENTE.

6.1 METODO DE BURMEISTER.

En este trabajo se especifican tres modelos para el análisis orientado a agentes de un sistema y se dan ciertas guías para la construcción del sistema a partir de aquí. El método está basado en técnicas orientadas a objetos. Esta aproximación, introduce elementos clave a detectar y modelar en el proceso de desarrollo de un sistema multiagente: organización, agente e interacciones, pero esta metodología esta poco detallada sobre todo en su fase de diseño y la falta de ejemplos complica el entendimiento del proceso de modelado [BUR, 1996], [VIC, 2002].

6.2 MAS-CommonKADS

La metodología MASCommonKADS [Iglesias 1998] está basada en CommonKADS [SCH, 2000], aportando una serie de modelos para desarrollar las fases de análisis y de diseño de sistemas multiagente. La principal característica es la incorporación de técnicas orientadas a objetos a CommonKADS, la cual es tomada como eje fundamental a lo largo de todo el proceso.

Siendo esta metodología muy completa, posee características que permiten el modelado de sistemas complejos, por lo tanto, al intentar modelar un sistema sencillo como el planteado en este proyecto, era posible aplicar solo una parte de los conceptos que esta metodología tan completa maneja, dejando el modelado a un nivel demasiado alto y abierto, además, hecho que el grupo de trabajo no había tenido la oportunidad de estudiarla y aplicarla en proyectos anteriores, contrario a lo que sucedía con la metodología GAIA, la cual se había usado y probado en proyectos anteriores del grupo de trabajo [VIC, 2002].

6.3 MaSE (Multiagent System Engineering)

MaSE [WOO, 2000] es una metodología desarrollada en el Air Force Institute of Technology. Dicha metodología trata de cubrir todas las etapas en el proceso de construcción de un sistema multiagente, partiendo de la especificación del mismo

hasta su implementación. Dispone de un lenguaje de especificación basado en UML+OCL [ROB, 2000] y una herramienta de desarrollo denominada AgentTool que trata de cubrir la totalidad de fases de la metodología e intenta facilitar el proceso de desarrollo, pero por el momento se queda en sólo una parte de este, puesto que la herramienta no incorpora la totalidad de las etapas.

6.4 GAIA

La metodología GAIA [WOOL, 1998] [WOOL, 2000] se centra en la idea de que la construcción de sistemas basados en agente es un proceso de diseño organizacional. Cabe resaltar el hecho de que esta metodología que proponen es muy genérica, produce especificaciones a un alto nivel de abstracción, lo cual la hace muy flexible y con ello se consigue desacoplarse de las distintas soluciones de implementación de agentes [VIC, 2002]. Por otra parte, esta metodología fue aplicada con éxito por el grupo de trabajo en proyectos académicos anteriores.

6.5 AUML

AUML [ODE, 2000a] [ODE, 2000b] no es en sí una metodología o un método sino que se centra más en intentar adaptar herramientas de desarrollo ya existentes y que están teniendo éxito para aplicaciones industriales reales, como es el caso de UML, tratando de orientarlas hacia el campo de los agentes. Este método es usado en el desarrollo de este proyecto para integrarse al proceso de modelado planteado en GAIA, lo cual permitirá obtener un modelo más cercano a la implantación.

7 SISTEMAS DE BUSQUEDA EN INTERNET

7.1 EVOLUCION DE LOS BUSCADORES EN LA WEB.

El crecimiento explosivo de la World Wide Web, la heterogeneidad de la información que contiene y la anarquía de su organización, dificultan enormemente el hallazgo de información útil por parte de los usuarios. Para solventar este problema han sido desarrolladas diferentes herramientas de búsqueda que evitan al usuario vagar sin rumbo por la red para localizar la información que necesita [BLA, 2005].

A inicios de la década de los 90's, se crearon los servicios Archie, Verónica y WAIS que son los antepasados de los actuales buscadores Web.

En 1993 se desarrolló World Wide Web Wanderer, que fue el primer robot de la red y tenía como objetivo contabilizar el número de usuarios de la red para medir el crecimiento de ésta.

En 1994 en la Universidad de Standford se creó Yahoo! y en la Universidad de Washington Webcrawler que fue la primera máquina que buscaba a texto completo en Internet.

En 1995 aparece Altavista que incluye nuevos aspectos en las máquinas de búsqueda, como por ejemplo la velocidad o el uso de operadores booleanos.
Por estas fechas se advierte que cuando se realiza una búsqueda en diferentes buscadores los resultados no son los mismos, es por ello que aparecen los metabuscadores. El primero en aparecer sería Metacrawler, que fue desarrollado por estudiantes de la universidad de Washington en 1995.

A finales de 1997 aparece Google, como un proyecto de investigación de la Universidad de Standford. Su novedad principal es que basa sus búsquedas en un ranking de páginas ordenadas por su relevancia [BAE, 2002]. El criterio que sigue para ordenar las páginas es el número de enlaces que otras páginas tienen a la página en concreto.
Algunas de las últimas incorporaciones han sido WiseNut, Teoma o Gigablast en el campo de los motores de búsqueda y Vivísimo en el de los metabuscadores.

7.1.1 Tipos de Buscadores

Actualmente, dentro de los sistemas de Recuperación de Información en la Web basados en Recuperación de Información por medio de palabras clave, se pueden identificar cuatro tipos: motores de búsqueda, directorios, metabuscadores y técnicas de filtrado de información [CHA, 2001]. El uso de motores y directorios son formas bien conocidas y frecuentemente usadas de buscar información en la Web, los motores indexan una porción de los documentos residentes en la globalidad de la Web y permiten localizar información a través de la formulación de una pregunta. Los directorios clasifican documentos Web seleccionados por materia y permiten al usuario navegar por sus secciones, buscar en sus índices o bien acceder a estos recursos a través de una búsqueda por palabras clave.
Los Metabuscadores son sistemas desarrollados para mitigar el problema de tener que acceder a varios motores de búsqueda con el fin de recuperar información mas completa sobre un tema, siendo estos mismos sistemas los que se encargan de efectuarlos por el usuario.

7.1.1.1 Índices Temáticos o Directorios

Los directorios son aplicaciones controladas por humanos que manejan grandes bases de datos con direcciones de páginas, títulos, descripciones, etc. Estas bases de datos son alimentadas cuando sus administradores revisan las direcciones que les son enviadas para luego ir clasificándolas en subdirectorios de forma temática. Los

directorios más amplios cuentan con cientos de trabajadores y colaboradores revisando nuevas páginas para ir ingresándolas en sus bases de datos. Los directorios están organizados en categorías temáticas, que se organizan jerárquicamente en un árbol de materias de información que permite ojear los recursos descendiendo desde los temas más generales a los más específicos. Las categorías presentan un listado de enlaces a las páginas referenciadas en el buscador. Cada enlace contiene una breve descripción de su contenido [POM, 2004] [AGU, 2002].

La mayoría de los índices permiten el acceso a los recursos a través de dos sistemas: navegación, a través de la estructura de las categorías y búsqueda por palabras claves sobre el conjunto de referencias contenidas en el índice. El directorio más grande y famoso es Yahoo!, aunque existen otros bastante conocidos como Dmoz el cual es un directorio alimentado por miles de colaboradores. Looksmart, Infospace e Hispavista. Con mucha diferencia, el más utilizado es Yahoo!. Los directorios son más usados que los motores, especialmente cuando no se conoce exactamente el objetivo de la búsqueda [MAN, 2002] [POM, 2004], ya que resulta difícil acertar con los términos adecuados de búsqueda.

Es oportuno puntualizar que el razonamiento que lleva a considerar que un directorio es mas preciso que un motor, se basa, sin duda alguna, en la fiabilidad de la descripción del registro, realizada manualmente de forma detallada y ajustada [MAR, 2002]. Evidentemente, este nivel de ajuste varía sustancialmente cuando la descripción se ha realizado a través de un proceso automático, como suele ser el caso de los motores de búsqueda.
La tabla 1 resume las ventajas e inconvenientes de usar un directorio cuando se busca información en la Web [BLA, 2005]

Ventajas	Inconvenientes
Son considerados más fáciles de usar que los motores de búsqueda, especialmente para usuarios no experimentados.	El usuario puede distraerse antes de localizar lo que se había propuesto encontrar.
Permiten al usuario a tener una visión general del volumen de información disponible. Algunos de ellos indican en cada uno de los nodos cuántas referencias contienen cada una de las subcategorías posibles desde el nodo actual.	Cubren tan solo una pequeña porción de los recursos existentes en la Web, por lo que si se quiere información sobre algo muy concreto quizás no se encuentre con un directorio y se deba recurrir a un buscador.
Los recursos disponibles han pasado por un proceso de selección de calidad, generalmente efectuado por documentalistas.	Suelen estar desactualizados, debido a la intervención humana en este proceso
Los términos hallados están dentro del contexto de la categoría en la que efectuemos la búsqueda, lo cual disminuye considerablemente el ruido.	La clasificación por parte de un equipo humano se dificulta por la carencia de criterios homogéneos para la selección y clasificación de los recursos, así como para su descripción.
Las descripciones que muestran acerca de los recursos Web suelen estar elaboradas intelectualmente, por lo que son realmente descriptivas de su contenido	Muchos recursos dejan de ser útiles si no se utilizan mecanismos automáticos para seguir los cambios en sus contenidos, direcciones, aparición o desaparición.

Tabla 1 Ventajas e Inconvenientes de usar un directorio cuando se busca información en la Web

7.1.1.2 Motores de búsqueda

Los motores de búsqueda son aplicaciones que manejan también grandes bases de datos de referencias a páginas Web recopiladas automáticamente, sin intervención humana. Uno o varios agentes de búsqueda recorren la Web, a partir de una lista inicial de direcciones y recopilan nuevas direcciones, generando una serie de etiquetas que permiten su indización y almacenamiento en la base de datos. Un motor no cuenta con subcategorías como los directorios, sino con avanzados algoritmos de búsqueda que analizan las páginas y proporcionan el resultado mas adecuado a una búsqueda. También almacenan direcciones que les son remitidas por los usuarios.

Entre los motores mas populares se destacan Altavista, Lycos, Alltheweb, hotbot, Overture, Askjeeves, Direct Hit, Google, Microsoft Network, Terra, WISEnut, entre otros. La tabla 2 resume las ventajas y desventajas de usar un motor cuando se busca información en la Web [BLA, 2005]

Ventajas	Inconvenientes
Las bases de datos de los motores son más amplias que las de los índices. Dado que el proceso de recolección e indexación de recursos es automático.	Su utilización es más compleja que la de los directorios, requiere un mayor esfuerzo por parte del usuario.
Se utilizan también mecanismos automáticos para seguir los cambios en sus contenidos, direcciones, aparición o desaparición. Algunos buscadores incluso guardan una copia en caché de los documentos tal como estaban en el momento en que fueron explorados.	Para obtener resultados precisos se requiere formular la consulta cuidadosamente, eligiendo adecuadamente los términos y los operadores, y delimitando adecuadamente la búsqueda.
Se pueden utilizar sinónimos y traducciones de los términos significativos para tener una mayor amplitud del campo de búsqueda	Cada buscador tiene su propia sintaxis que es preciso conocer y diferenciar para expresar la consulta
	Los recursos indexados por los robots, generalmente no han pasado por ningún proceso de selección de calidad por lo cual entre los resultados puede haber mucha “basura”.

Tabla 2 Ventajas e inconvenientes de usar un motor cuando se busca información en la Web.

La tabla 3 Muestra las características básicas de dos de los métodos de recuperación de información en la Web: motores e índices temáticos [DEL, 1998]

	Descubrimiento de recursos	Representación del contenido	Representación de la consulta	Presentación de resultados
Directorios	Realizado por personas.	Clasificación manual.	Implícita (Navegación por categorías)	Paginas creadas antes de la consulta. Poco exhaustivas. Muy precisas.
Motores de Búsqueda	Principalmente de forma automática por medio de robots	Indización automática.	Explícita (palabras clave, operadores, etc.)	Paginas creadas dinámicamente en cada consulta. Muy exhaustivos, poco precisos.

Tabla 3 Mecanismos de recuperación de información en la WWW. Resumen.

7.1.1.3 Metabuscadores

Un metabuscador colecciona las respuestas recibidas y las unifica, la principal ventaja de los metabuscadores es su capacidad de combinar los resultados de muchas fuentes y el hecho de que el usuario pueda acceder a varias fuentes de forma simultánea a través de una simple interfaz de usuario [BAE, 1999]. Estos sistemas no

almacenan descripciones de páginas en su base de datos. En lugar de eso contienen registros de motores de búsqueda e información sobre ellos. Envían la petición del usuario a todos los motores de búsqueda (basados en directorios y crawlers[1]) que tienen registrados y obtienen los resultados que les devuelven. Algunos mas sofisticados detectan las URL duplicadas provenientes de varios motores de búsqueda y eliminan la redundancia [AGU, 2002], es decir solo presentan una al usuario.

Por muy grande y exhaustiva que pudiera llegar a ser la base de datos de un motor de búsqueda o de un directorio, nunca va a cubrir un porcentaje muy elevado del total de la Web [BRA, 2003].

Estos sistemas se pueden clasificar en dos tipos dependiendo de la forma en que presentan los resultados: multi buscadores y meta buscadores. Los multi buscadores ejecutan la consulta contra varios motores de forma simultánea y presentan los resultados sin mas organización que la derivada de la velocidad respuesta de cada motor como por ejemplo All4One que busca en una gran cantidad de motor de búsqueda y directorios; los meta buscadores funcionan de manera similar a los multi buscadores pero, a diferencia de éstos, eliminan las referencias duplicadas, agrupan los resultados y generan nuevos valores de pertinencia para ordenarlos. Algunos ejemplos son MetaCrawler, Cyber411 y Digisearch [AGU, 2002]. Algunos de esto sistemas presentan los resultados en diferentes ventanas, correspondiendo cada una de ellas a una fuente distinta, como lo hacen Oneseek o Proteus por ejemplo. La tabla 3 resume las ventajas e inconvenientes de usar un metabuscador cuando se busca información en la Web [BLA, 2005].

[1] "Este término se refiere al robot que recopila páginas Web para el índice de los motores de búsqueda."

Ventajas	Inconvenientes
El usuario sólo necesita acceder a una única página Web para ejecutar la búsqueda y no tiene que recordar los nombres ni direcciones de los buscadores	Al formular una única consulta para todos los buscadores puede que la sintaxis utilizada no sea la más adecuada para cada uno de ellos
El usuario ha de aprender a utilizar una única interfaz para realizar sus búsquedas	Son más lentos debido a que el proceso que siguen es más complejo y elaborado.
Permiten ejecutar una búsqueda más extensiva a través de un amplio número de herramientas de búsqueda.	Por lo general, ofrecen las mismas posibilidades de interrogación que los buscadores independientes.
Sólo hay que teclear la consulta una vez y el metabuscador la remitirá a cada una de las herramientas de búsqueda	
Se puede obtener una lista integrada de resultados de manera que se eliminen los duplicados.	

Tabla 4 Ventajas e inconvenientes de usar un metabuscador cuando se busca información en la Web [BLA, 2005].

7.1.1.4 Técnicas de filtrado de Información

El concepto de filtrado tiene que ver con la decisión de considerar a priori si un documento es relevante o no, eliminándolo del índice en caso contrario.

Estas técnicas se basan en una combinación de sistemas de autoaprendizaje y sistemas de Recuperación de Información y han sido empleadas en la construcción de motores de búsqueda especializados [CHA, 2001].

El filtrado de términos mejora la calidad del índice del motor, rechazando términos de escaso o nulo valor de discriminación y contribuyendo a acrecentar la velocidad de la Recuperación de Información, al aligerar las dimensiones del fichero índice. Los agentes encargados de recopilar la información por la Web, someten las páginas que recuperan al sistema de filtrado, el cual si los acepta, los almacenara en el índice del motor, mientras que las rechazadas son descartadas. Entre los motores de búsqueda más conocidos que emplean estas técnicas se destaca Northem Light. [CHA, 2001].

El hecho es que respuestas con un número tan alto de páginas encontradas producen en el usuario el fenómeno bien conocido de la sobrecarga de información (desbordamiento cognitivo). [BER, 2001].

8. REFERENCIAS

[ALC, 2004] Alcaide, R., García, A., Alcañiz, M. and Marcos, A. Sistemas de Representación y Procesamiento Automático del Conocimiento: Ontologias en la web semántica [En línea]. Valencia: Universidad Politécnica, Facultad de Informática. 2004 <http://personales.upv.es/ccarrasc/doc/2003-2004/Onto_WS/OntologiasenlaWebSemantica.htm/>

[AGU, 2002] Aguilar, R. Monografía: Motores de búsqueda [En línea]. <http://www.geocities.com/motoresdebusqueda/inicio.html>

[BAE, 1999] Baeza-Yates, R. and Ribeiro-Neto, B.:Modern Information Retrieval. New York: ACM Press; Harlow [etc.]: Adisson-Wesley, 1999 XX, 513 p. ISBN 0-201-39829-X.

[BAE, 2002] Baeza-Yates, R. and Davis, E.: Ranking Global de Paginas Web Basado en Atributos de los Enlaces. Chile: Universidad, Centro de investigación de la Web. Dpto. de ciencias de la computación. 2002.

[BEC, 2002] Bechhofer, S. and Ng, G.: OilEd. Manchester: Universidad. 2002.. http://oiled.man.ac.uk

[BER, 2001] Berrocal, J., Figuerola, C., Zazo, A. and Rodríguez, E. Agentes Inteligentes: Recuperación Autónoma de Información en el Web [En línea]. Salamanca: Universidad, Departamento de Informática y Automática, Facultad de Documentación. 2001. <http://reina.usal.es>

[BLA, 2005] Blanco, M. Estudio de Buscadores. Vigo: Universidad. Área de lenguajes y sistemas de información. Escuela Superior de Ingeniería Informática. 2005.

[BRA, 2003] Bradley, P. Multi-search Engines - a comparison [En línea]. London. 2003. <http://www.philb.com/msengine.htm>

[BUR, 1996] Burmeister, B. Models and methodologies for agent-oriented analysis and design. Technical Report D-96-06, DFKI.

[CHA, 2001] Chang, G. Mining the World Wide Web: an information search approach. Kluwer Academic Publishers. Massachssetts. 2001.

[COR, 2004] Corcho, O., Fernandez-Lopez, M., Gomez-Perez, A. and Lopez-Cima, A. Building legal anthologies with METHONTOLOGY and WebODE. Madrid: Universidad Politécnica. Facultad de Informática. 2004.

[DEC, 2002] Decker, S., Fensel, D., Harmelen, F., Horrocks, I., McGuinness, D., Patel-Schneider, P., Staab, S. and Stein, L. On-To-Knowledge-Project [En Línea]. Stanford; Universidad, Manchester: Universidad. Department of Computer Science of, Bell Labs Research. Massachusetts: Institute of Technology. <http://www.ontoknowledge.org/oil/index.shtml >

[DEL, 1998] Delgado, A. Mecanismos de recuperación de información en la WWW [En línea]. Palma de Mallorca: Universidad de los Illes Balears. 1998. <http://www.dmi.uib.es/people/adelaida/tice/modul6/memfin.pdf>

[GOM, 2003] Gómez-Pérez, A., Arpírez, J., Fernández-López, M. and Corcho, O. WebODE [En línea]. Madrid: Universidad Técnica. Ontological Engineering Group of Artificial Intelligence. Department of Computer Science. 2003. <http://webode.dia.fi.upm.es/>

[DOM, 1999] Domingue, j. WebOnto [En lína]. London: The Open University Milton Keynes. Knowledge Media Institute. <http://kmi.open.ac.uk/projects/webonto>

[HEN, 2001] Hendler, j. and Harmelen, F. DAML+OIL [En línea]. World Wide Web Consortium (W3C). 2001. <http://www.daml.org/2000/12/daml+oil-index>

[Iglesias 1998] Iglesias, C. Definición de una metodología para el desarrollo de sistemas Multiagente. Madrid: Universidad Politécnica. PhD thesis. Departamento de Ingeniería de Sistemas Telemáticos. 1998.

[KOS, 2003] Koss, M., Mulholland, P. and Zdrahal, Z. Apollo [En línea]. <http://apollo.open.ac.uk>

[LAG, 2003] Laga, M. and Creech, W. LinkFactory®. Languaje & Computing [En línea]. <http://www.landcglobal.com/pages/linkfactory.php>

[MAN, 2002] Manchón, E. Navegación jerárquica o categorial frente al uso de buscador [En línea]. España. 2002. <http://www.ainda.info/navegacion_vs_buscador.html>

[MAR, 2002] Martínez, F. Propuesta y desarrollo de un modelo para la evaluación de la Recuperación de Información en Internet. Murcia: Universidad, Facultad de Ciencias de la Documentación, Tesis de doctoral. 2002.

[NOY, 2000] Noy, N., McGuinness D.: A Guide to Creating Your First Ontology Stanford University.

[ODE, 2000a] Odell, J., Parunak, H., and Bauer, B.: Extending UML for agents. In Proceedings of the Agent-Oriented Information Systems Workshop, pages 3-17.

[ODE, 2000b] Odell, J., Parunak, H., and Bauer, B.: Representing agent interaction protocols in UML. In Proceedings of the AGENTS'2000, España.

[ONT, 2002] Ontoprise. OntoEdit [En línea]. http://www.ontoprise.de/content/e3/e43/index_eng.html

[POM, 2004] Pombert, A. Herramientas para realizar búsquedas en Internet. Informática profesional publicación mensual del Consejo Profesional en Ciencias Informáticas [En línea]. Argentina. <http://www.cpci.org.ar/newsletters/95/busqueda.htm>

[RIC, 2002] Rice, J. Ontolingua [En línea]. Stanford: University Knowledge Systems Laboratory. 2002. <http://www-ksl-svc.stanford.edu:5915/FRAME-EDITOR/&sid=ANONYMOUS&user-id=ALIEN>

[USC, 1996] Uschold, M. "Converting an Informal Ontology into Ontolingua: Some Experiences". Budapest.. En Jones, Dean. Bench-Capon. Trevor and Visser, Pepjin. "Methodologies For Ontology Development".

[VEN, 2004] Vendetti, J., Crubézy, M., Dameron, O., Fergerson, R., Knublauch, H., Musen, M., Noy, N., Rubin, D. and Tu, S. Protégé ontology editor and knowledge-base framework [En línea]. Stanford: University, School of Medicine, Stanford Medical Informatics. 2004. <http://protege.stanford.edu/index.html>

[VIC, 2002] Vicente, J. and Vicente, B. Estudio de métodos de desarrollo de sistemas multiagente. Valencia: Universidad Politécnica. Departamento de Sistemas Informáticos y Computación. 2002.

[WOO, 2000] Wood, M. Multiagent Systems Engineering: A Methodology for Analysis and Design of Multiagent Systems. Air Force: Institute of Technology. MS Thesis. 2000.

[WOOL, 1998] Wooldridge, M., Jennings, N. R. and Kinny, D.: A Methodology for Agent-Oriented Analysis and Design. Seattle. O. Etzioni, J. P. Muller, and J. Bradshaw. 1998.

[ROB, 2000] Robinson, D. A Component Based Approach to Agent Specification. School of Engineering. Air Force Institute of Technology. MS Thesis. 2000.

[WOOL, 2000] Wooldridge, M., Jennings, N. and Kinny, D. The GAIA methodology for agent-oriented analysis and design. Australian: Journal of Autonomous Agents and Multi-Agent Technical Report 59, Artificial Systems. 2000.