ESTUDIO DE HERRAMIENTAS PARA EL ACCESO A ONTOLOGÍAS: CASO APLICADO A UNA ONTOLOGÍA DE PECES DEL CATATUMBO.

Recepción:
10 septiembre de 2010

Aprobación:
Noviembre 2 de 2010

[1]Torcoroma Velásquez Pérez, [2]Sindy Lorena Castro Angarita, [3]Andrés Mauricio Puentes Velásquez

1. Magister en Ciencias Computacionales. Decana Facultad de Ingenierías, Universidad Francisco de Paula Santander Ocaña
tvelasquezp@ufpso.edu.co

2. Ingeniera de Sistemas. Universidad Francisco de Paula Santander Ocaña
lorenacastro_01@hotmail.com

3. Ingeniero de Sistemas. Docente Facultad de Ingenierías, Universidad Francisco de Paula Santander Ocaña
ampuentesv@ufpso.edu.co

Resumen:

Este trabajo presenta un análisis de algunas técnicas y herramientas que pueden utilizarse para el tratamiento y gestión de ontologías: Prolog Server Pages, Prosper y RAP. Esta última, integra numerosas características, entre ellas se distinguen, un motor de consulta en SPARQL, un servidor en RDF, y métodos específicos de vocabulario que permiten la manipulación de modelos RDF; proporcionando una interfaz que permite la consulta de ontologías. Lo anterior indica, que además de proporcionar una relación con el entorno web, ofrece técnicas simples para el manejo de modelos ontológicos. Las técnicas mencionadas se aplicaron a una ontología específica de Peces del Catatumbo.

Palabras claves:

Ontologías, RDF, Web Semántica

Abstract:

This work present an analysis some tools that can be used for the treatment and management of ontologies, which are: Prolog Server Pages, Prosper and RAP. The latter includes many features, among them are distinguished, a SPARQL query engine, an RDF server, and specific methods of vocabulary that allows the manipulation of RDF models, develop an interface that allows the query ontology. This indicates, that besides providing a link to a web environment, offers simple techniques for managing ontological models. Techniques mentioned above were applied to a specific ontology of Catatumbo Fish.

Keywords:

Ontologies, RDF, Semantic Web

Introducción

En la actualidad nuestra experiencia como usuarios permanentes de los contenidos de la web puede resultar a veces complicada, debido en gran parte, a la innumerable cantidad de recursos y documentos publicados y a la dificultad para encontrar de manera rápida y efectiva la información que requerimos. Esta dificultad se origina por la ausencia de estructuras como las ontologías que alberguen el sentido semántico de los contenidos de la web; y por la falta de aplicaciones y buscadores que permitan acceder a las mismas de forma eficiente. Estas dificultades se ven reflejadas en la dificultad de las personas para explorar la web de manera sencilla y en el grado de insatisfacción presente a la hora de hallar los resultados esperados de una consulta.

Muchos de los sistemas que encontramos a diario, están orientados a proporcionar una interfaz agradable al usuario, sin embargo, algunos de ellos, no llenan las expectativas del cliente y no proporcionan las utilidades necesarias para que el mismo optimice su tiempo de trabajo y se sienta satisfecho con los resultados obtenidos; por tal motivo, se ha venido trabajando hace algunos años en el diseño e implementación de metodologías y herramientas que permitan crear y acceder al contenido de ontologías en el marco de la Web Semántica [1]; ésta web no es una web aparte de la existente, surge como una extensión de la web actual, donde se proporciona significado a los recursos disponibles en línea a través de unas especificaciones formales y compartidas [2].

Dentro de las herramientas para el manejo y la manipulación de ontologías [3] se presentaron Prolog Server Pages y Prosper, incluyendo los requisitos para la instalación y configuración de las mismas. De igual forma, se evalúo e implementó el paquete RAP, que integra numerosas características, entre ellas se distinguen, un motor de consulta en SPARQL [4], un servidor en RDF, y métodos específicos de vocabulario que permite la manipulación de modelos RDF.

Finalmente, se desarrolló un prototipo que permite la consulta de ontologías, probándose en la ontología creada para los peces del Catatumbo [5] , trabajada dentro del grupo de investigación de Teleinformática y Desarrollo de Software (GITYD); esta interfaz, además de proporcionar una relación con el entorno web, ofrece técnicas como las tripletas RDF, para el tratamiento de modelos ontológicos.

1. DESARROLLO

1.1 METODOLOGÍA

Estudio de las herramientas básicas para el manejo de ontologías con el fin de determinar cual se adaptaba más a los objetivos planteados.

Inicialmente, se recolectó la información necesaria para establecer los lenguajes más apropiados en cuanto a la manipulación de modelos ontológicos. Lo que se buscaba era que mediante una herramienta, se pudiera inicializar, leer y consultar una ontología en particular.

Por tal motivo, y luego de una minuciosa investigación, se encontraron algunas herramientas que se ajustaban a lo que se deseaba y que a continuación se especifican:

1.1.1 Prolog Server Pages (PSP). El primer sistema que se implementó fue Prolog Server Pages (PSP), un lenguaje de script basado en prolog que se incrusta en los documentos HTML, el cual se puede manejar en diferentes plataformas. Se optó por utilizar esta herramienta, pues es considerada una intérprete que actúa como interfaz entre un servidor web y un navegador. De igual forma, PSP es compatible con peticiones GET y HTTP POST. También proporciona métodos para trabajar con las cookies de http [6]. Además, PSP está compuesto por un script en Prolog, embebido en los documentos HTML.

Los hechos mencionados anteriormente, describen la funcionalidad de PSP por adaptarse al entorno web. Sin embargo, lo que se quería, era

que esta herramienta, permitiera tratar (es decir, inicializar, leer y consultar) una ontología, y fue la más acertada para esta labor debido a su estructura basada en Prolog (Lenguaje de programación simbólico o de Inteligencia Artificial, basado en cálculo de predicados) [7], que utiliza predicados y reglas, la programación estaría más orientada a la lógica, algo muy importante en la consulta de la ontología.

La Lógica simbólica apareció con G.W. Leibniz [8] pero fue olvidada con su muerte, la redescubrió George Boole [9] y se conoció como Lógica Booleana la cual trata de la abstracción de conceptos en símbolos y la interconexión de estos símbolos mediante operadores. La lógica simbólica consta de dos ramas, la lógica proposicional que trata de la verdad o falsedad de una proposición y el cálculo de predicados que incluye relaciones entre objetos y clases de objetos. [10]

1.1.1.1 Requisitos previos a la instalación. Se utiliza el sistema operativo Windows XP ya que soporta PSP, un editor de texto para realizar la modificación de algunos archivos y Swi-Prolog [11] para implementar el código en Prolog.

1.1.1.2 Configuración. Se procede a la configuración e instalación de PSP. Este intérprete, puede ser instalado en plataforma Linux y Windows, para este caso en particular se utilizó el Sistema Operativo Windows XP y el conjunto de servicios denominado Internet Information Server (IIS), versión 6 [12], esta característica solo se puede encontrar en algunos sistemas operativos de Microsoft.

Se instaló SWI-Prolog (versión 5.4.5), inicialmente se tuvo que guardar el archivo webserver.pl que viene por defecto en la carpeta bin de PSP, quedando una ruta como C:\PSP\webserver.pl. A continuación, se le configuraron los permisos correspondientes a SWI-Prolog y a webserver.pl, editando este último archivo e incluyéndole en la línea 40 psp_session_ database ('C: \PSP\session\'), por tal motivo se creó un subdirectorio en la carpeta sesión, que se añadió a la ruta C:\PSP\.

En el Panel de Control se seleccionó Herramientas Administrativas, seguido de ello, Internet Information Services (IIS6) Manager, que es la interfaz correspondiente a la plataforma de administración IIS6, eligiendo la carpeta Web Site, y luego Sitio Web Predeterminado, clic en propiedades, después Home Directory y por último configuration, se complementa incluyendo en las casillas:

Executable: C:\progra~1\pl\bin\plcon.exe -q -s C:\ psp\webserver.pl -g run_page -t halt
Extension: .prolog
Verbs: All Verbs
Script engine: habilitar
Check that file exists: habilitar

1.1.2 Prosper [13]. La principal característica por la cual fue escogida esta aplicación, es su adaptación a la web, ya que cuenta con un módulo FastCgi [14], el cual trabaja como un protocolo para interconectar programas interactivos con un servidor web. De igual forma, este sistema trae incorporado un intérprete PSP, que procesa las peticiones en Prolog, esta última razón, es muy importante para el tratamiento de ontologías.

1.1.2.1 Requisitos previos a la instalación. Para tener una configuración adecuada de Prosper se utilizaron herramientas y lenguajes específicos, entre ellos el sistema operativo (para el caso de estudio Windows XP [15]), el servidor Apache [16] versión 2.0.49. Así mismo se descargó el módulo de FastCGI llamado mod_fastcgi-2.4.2.AP20.dll.

1.1.2.2 Configuración. Luego de haber cumplido con los requisitos previos, se pasó a la configuración de Prosper; para ello se procedió a instalar el servidor de Apache, en este caso, la versión anteriormente especificada, que se aproximaba al requisito de Prosper; lo más recomendable fue ejecutar Prosper, en la ruta C:/Archivos de programa/Prosper; se ingresó a la ruta C:/Archivos de programa/Prosper/ implementation, y se copió la carpeta Prosper a la unidad C: de modo que quedara C:/Prosper/ Implementation/Examples/Modules. De igual forma, se instaló el módulo fastCGI en la ruta C:/

Archivos de programa/Apache Group/Apache 2/ modules.

En la ruta C:/Archivos de programa/Apache Group/ Apache 2/conf, se buscó el archivo httpd.conf y se agregaron las líneas:

```
# Load the FastCGI and the URL rewriting modules
LoadModule fastcgi_module modules/mod_fastcgi.so
LoadModule rewrite_module modules/mod_rewrite.so

# Turn on URL rewriting
RewriteEngine on

# Declare the rewriting rule for files with extension .psp
# The virtual directory is relative to the server root.
RewriteRule ^(.*)\.psp$ /ProsperPages/$1\.xhtml [last]

# Configure FastCGI for communication over TCP/IP
# The directory should be the same as specified in the above RewriteRule but
# should be given as an absolute filesystem path
FastCgiExternalServer "C:/htdocs/ProsperPages" -host localhost:1160

# Configure FastCGI for communication over named pipes on Windows
# FastCgiIpcDir "\\\\.\\pipe\\Prosper"
# FastCgiExternalServer "C: /htdocs/ProsperPages" -socket "FastCGI"
```

Para probar la instalación, fue necesario copiar la carpeta Example que se encontraba en "C:/ Archivos de programa/Prosper/Implementation" y guardarla en C:/Archivos de programa/Apache Group/Apache 2/htdocs.

1.1.3 RAP [17]. Finalmente, se evaluó e implementó RAP, el cual es un paquete de software, que posee una gran cantidad de características orientadas a la web y está centrado en los procedimientos ontológicos para la manipulación de un modelo RDF [18], a través de métodos específicos de vocabulario. RAP, integra lenguajes de consulta, análisis y tratamiento de los modelos RDF (Ontologías).

1.1.3.1 Requisitos previos a la instalación. Para lograr una instalación exitosa, fue necesario descargar el paquete de software RAP-RDF API para PHP versión 0.9.6, Appserv versión 2.5.9, Protégé [19] versión 3.4.4, que es considerada una herramienta para la edición y creación de ontologías y por último un documento con extensión owl [20].

1.1.3.2 Configuración. Para la adecuada instalación e implementación, y luego de cumplir con los requisitos previos, fue preciso realizar la configuración, teniendo en cuenta los siguientes pasos:

- En Protege, se abrió el documento con extensión .owl que es el formato que contiene la ontología y se cambió el formato a RDF para permitir la carga de las tripletas RDF.

- Para realizar la conversión, se seleccionó la pestaña File, a continuación la opción Convert Project to Format, de la cual se eligió RDF Files y posteriormente OK (ver Fig. 1)

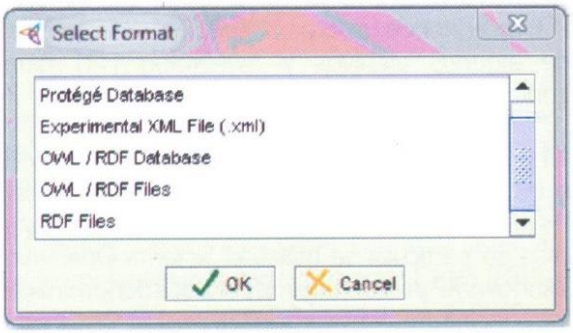

Fig. 1. Vista de selección del formato a convertir

- En la opción Classes file name, se colocó la ubicación y el nombre con la extensión .rdf, de igual forma en Instances file name se realizó el paso anterior pero con la extensión .rdfs y luego OK. El editor comenzó realizando la conversión de OWL a RDF, de la conversión resultó un archivo rdf que se llamó pecescatatumbo.rdf

- Se instaló Appserv en la ruta por defecto C: / Appserv.

• Se creó una carpeta llamada sistemaOntologico, que se guardó en la ruta C:/Appserv/www/
• En sistemaOntológico, se guardó la carpeta que contenía el paquete rdf api-php, anteriormente descrita.

• En el navegador se ingresó http://localhost/phpmyadmin y se creó una base de datos llamada sistemaOntologico, en donde se cargaron los modelos.

1.1.3.3 Codificación. Luego de haber desarrollado paso a paso la configuración, se procedió a modificar el archivo del código fuente de la ontología cargada; para tal efecto fue necesario, tener el archivo rdf resultante de la conversión.

Lo primero que se hizo fue editar la codificación que permite la inicialización y carga del modelo ontológico, este archivo se denominó init.php, el cual es mostrado a continuación:

```
<?php
/*incluir rap es decir el nombre de la libreria que se descargo*/
//se define una carpeta donde se encuentra el API de rdf
define("RDFAPI_INCLUDE_DIR","./rdfapi-php/api/");
//se incluye el api rdf para PHP
Incluye(RDFAPI_INCLUDE_DIR . "RdfAPI.php");
//se crea un modelo DBStore que permite conectarse a la base de datos para guardar y obtener
//rdf_database = ModelFactory::('tipo_BD','server','base_datos','usuario','password');
$rdf_database = ModelFactory::getDbStore('MySQL','localhost','sistemaOntologico','root','root');
//crear tablas para mysql
$rdf_database->createTables('MySQL');
//nombre y direccion del archivo que contiene rdf/OWL
$base = "pecescatatumbo.rdf";
//se crea un modelo en memoria para cargar el archivo RDF/OWL
$memModel = ModelFactory::getDefaultModel();
//se carga y se parsea el documento
$memModel->load($base);
//colocar el modelo en la base, el URI será generado automáticamente
$rdf_database->putModel($memModel);
$modelURI = "pecescatatumbo.rdf";
if ($rdf_database->modelExists($modelURI))
  echo "Modelo con la siguiente URI : '$modelURI' ya existe";
else
  $rdf_database->putModel($memModel , $modelURI);

echo"Mostrando los modelos en la base de datos <br>";
$list= $rdf_database->listModels();
//mostrar el contenido de la base de datos
Foreach($list as $model)
{ echo "modelURI: " .$model['modelURI']."<BR>";
echo "baseURI: " .$model['baseURI']."<BR><BR>";}
?>
```

Luego de haberse inicializado y cargado la tripleta, se elaboró el código referente a la lectura de la ontología, que se llamó read.php y que a continuación se visualiza:

```
<?php
/*incluir rap*/
//se define una carpeta donde se encuentra el API de rdf
define("RDFAPI_INCLUDE_DIR","./rdfapi-php/api/");
//se incluye el api rdf para PHP
Incluye(RDFAPI_INCLUDE_DIR . "RdfAPI.php");
//se crea un modelo DBStore que permite conectarse a la base de datos para guardar y obtener
//rdf_database = ModelFactory::('tipo_BD','server','base_datos','usuario','password');
$rdf_database = ModelFactory::getDbStore('MySQL','localhost','ontoturismo','root','root');
$dbModel = $rdf_database->getModel("pecescatatumbo.rdf");
$dbModel->writeAsHtmlTable();
$rdf_database->close();
?>
```

Por último se codificó el query quien es el que permitiría realizar la respectiva consulta. El código se llamó query.php, se muestra a continuación:

```
<html>
<head>
<title>query</title>
</head>
<body>
<?php
if(!$_POST)
{ }
Else  {
/*incluir rap*/
//se define una carpeta donde se encuentra el API de rdf
define("RDFAPI_INCLUDE_DIR","./rdfapi-php/api/");
//se incluye el api rdf para PHP
Incluye(RDFAPI_INCLUDE_DIR . "RdfAPI.php");
//se crea un modelo DBStore que permite conectarse a la base de datos para guardary obtener
//rdf_database = ModelFactory::('tipo_BD','server','base_datos','usuario','password');
$rdf_database = ModelFactory::getDbStore('MYSQL','localhost','sistemaOntologico','root','root');
$dbModel = $rdf_database-> getModel("pecescatatumbo.rdf");
$nTerm = $_POST['nTerm'];
$split=explode(' ',$nTerm);
// Quero string
$rdql_query = '
SELECT ?NOMBRE ?GENERO ?ESPECIE ?AUTOR ?LOCALIDAD
WHERE (?x, <kb:nombre> , ?NOMBRE)
      (?x, <rdf:type>, ?GENERO)
      (?x, <rdfs:label>, ?ESPECIE)
      (?x, <kb:Descubridor> , ?AUTOR)
      (?x, <kb:Localidad> , ?LOCALIDAD)
AND ?NOMBRE =~ "/.*$split[0].*/"
USING kb FOR <http://protege.stanford.edu/kb#>
   rdf FOR <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
   Rdfs FOR <http://www.w3.org/2000/01/rdf-schema#>';
//modelo del query
$res= $dbModel->rdqlQuery($rdql_query);
//mostrar resultados
RdqlEngine::writeQueryResultASHtmlTable($res);
```

1.2 RESULTADOS

Luego de haber analizado y estudiado cada una de las herramientas anteriormente descritas, se logró concluir, que RAP posee prestaciones y características especiales que permiten el acceso y manipulación de modelos ontológicos previamente implementados.

Por tal motivo, se realizaron las pruebas pertinentes con el fin de determinar y de comprobar la hipótesis planteada inicialmente, que consistía en que la implementación de RAP permitirá demostrar la posibilidad de acceder y manipular contenidos de una ontología implementada en un lenguaje formal, En la Figura 2 se muestra la lectura de la ontología.

URI base: peces.estatumbo.rdf#		Tamaño: 748
Prefijo:	Espacio de nombres:	
kb	http://protege.stanford.edu/kb#	
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	
rdfs	http://www.w3.org/2000/01/rdf-schema#	
N°	Asunto	Predicado	Objeto
1.	Recurso:kb:AnequidensPulcher	RDF Nodes:rdf:type	Recurso:kb:Anequidens
2.	Recurso:kb:AnequidensPulcher	Recurso:kb:Descubridor	Literal:Gill 1858
3.	Recurso:kb:AnequidensPulcher	Recurso:kb:Localidad	Literal:Maravilla de La Quebrada
4.	Recurso:kb:AnequidensPulcher	Recurso:kb:Nombre	Literal:Azul Mejorra
5.	Recurso:kb:AnequidensPulcher	RDF Nodes:rdfs:label	Literal:AnequidensPulcher
6.	Recurso:kb:AnequidensPulcher	Recurso:kb:TieneAletaDorsal	Recurso:kb:AlargadeyAfilada
7.	Recurso:kb:AnequidensPulcher	Recurso:kb:TieneAletaAdiposa	Recurso:kb:Ausente

Fig. 2. Lectura del modelo ontológico

A continuación, se desarrolló una plantilla query.php (Fig. 3), mediante el cual se visualizó la información de la ontología de peces del Catatumbo, producto generado como parte del macroproyecto de Diseño Ontológico de flora y fauna colombiana del grupo de investigación de Teleinformática y Desarrollo de Software (GITYD)

Fig. 3. Interfaz para la prueba de la consulta

No.	?NOMBRE	?GENERO	?ESPECIE	?AUTOR	?LOCALIDAD
1.	Literal: Bocachico	Resource: http://protege.stanford.edu/kb#Prochilodus	Literal: ProchilodusReticulatus	Literal: Valenciennes, 1850	Literal: Rio Antes del asentio el Divizo, cuenca media del rio Catatumbo

Fig 4. Resultado de la consulta "Bocachico"

En la interfaz se debía digitar el nombre de un pez dentro de las opciones que se encontraban en la pantalla principal, en la Figura 4 se muestran los resultados generados del caso "Bocachico", como se representa en la ontología diseñada.

Del mismo modo que el ejemplo planteado, se realizaron distintas pruebas donde se pudo comprobar, que mediante la utilización del prototipo, se puede manipular correctamente la ontología, permitiendo, cargar, leer y consultar el modelo de manera apropiada.

Conclusiones

La implementación de las herramientas descritas anteriormente, proporcionó un enfoque para determinar cuál de ellas se adaptaba mas a los objetivos propuestos. Después de analizar detalladamente los sistemas y las herramientas propuestas, se llegó a la conclusión, de que a pesar de una configuración e instalación simple, requerían de un avanzado conocimiento a la hora de aplicarlos a un dominio de conocimiento específico, resultando a veces insuficiente o incomprensible de la documentación proporcionada por los creadores de estas herramientas.

Luego del estudio realizado se logró establecer que RAP, era la herramienta que más se adecuaba a las características esperadas en este caso particular, así mismo, se logró acceder y manipular una ontología, relacionando la información de acuerdo a un término especifico. Sin embargo, ésta, poseía limitaciones a la hora de realizar una consulta.

La Web Semántica, es abordada dentro de las técnicas y métodos de la Inteligencia Artificial, como una rama de vital importancia, que contribuye al desarrollo y evolución de la forma como accedemos a la información contenida en la innumerable cantidad de documentos disponibles en la Web; esto se logra gracias a la implementación de estructuras donde se representa el conocimiento de un dominio específico y su semántica asociada, además, de la aplicación de técnicas y herramientas para acceder y manipular dichas estructuras, mostrando de manera óptima los resultados de una consulta. Dentro de la Universidad Francisco de Paula Santander de Ocaña, específicamente en la facultad de ingenierías, en el programa de Ingeniería de Sistemas se le está dando gran prioridad a esta área, con el fin de capacitar y apoyar a los estudiantes, brindándoles la oportunidad de desarrollar investigaciones en un campo de la computación que ha recibido gran interés por parte de la comunidad internacional y que requiere el esfuerzo y la dedicación de personas que realicen trabajos en esta área poco explorada.

Este artículo, pretende informar un caso de estudio acerca del manejo de diferentes técnicas y herramientas en el manejo y tratamiento de una ontología, de manera que esto no implique un esfuerzo sustancial por parte del usuario final de una aplicación que implemente características de la Web Semántica. El grupo de Investigación en Teleinformática y Desarrollo de Software (GITYD) a través de la Línea de Investigación de Sistemas Inteligentes, propende por la divulgación a la comunidad de su experiencia en este tipo de trabajos y trabaja continuamente en el desarrollo de bases sólidas que permitan la implementación de proyectos futuros para el beneficio de la Institución y de toda la comunidad interesada.

Bibliografía

[1] CORCHO, Oscar. FERNÁNDEZ LÓPEZ, Mariano. GOMEZ PEREZ, Asunción. LÓPEZ CIMA, Ángel. Construccion de Ontologias juridicas con Methontology y Web Ode, 2005.

[2] PUENTES, Andrés M. "Ontology for Semantic Search of Orchid Genera from the Colombian Florae". Ponencia presentada en el evento: 'VII Pan-American Workshop in Applied and Computational Mathematics' Venezuela, Junio de 2010.

[3] [Citado junio 2, 2010]. Disponible en: www.infor.uva.es/~sblanco/Tesis/Ontologias.pdf

[4] PRUD'HOMMEAUX, E. SEABORNE, A. Lenguaje de consulta SPARQL para RDF, W3C Working Draft. Abril 2005.

[5] Galvis, G. Peces del Catatumbo. Santafé de Bogotá, D C: Editorial Ltda. Mayo, 1997.

[6] SUCIU, Alin. Redes y Arquitectura de Internet. Comell University, marzo 2006.

[7] KING, David. Sistemas Expertos. 1 ed. Madrid: Díaz de Santos, S.A, 1988.

[8] LEIBNIZ, Gottfried. Biografia. [Citado julio 5, 2010]. Disponible en: http://www.dma.eui.upm.es/historia_informatica/Doc/Personajes/GottfriedLeibniz.htm

[9] BOOLE, George. Biografias y vidas. [Citado julio 6, 2010]. Disponible en: http://www.biografiasyvidas.com/biografia/b/boole.htm

[10] RICH, E. & KNIGHT, K.Inteligencia Artificial. Segunda Edición. Mc Graw Hill,1994.

[11] WIELEMAKER, Jan. SWI-Prolog y la Web. University of Amsterdam, 2003.

[12] TULLOCH, Mitch. IIS6 administration. McGraw-Hill: Osborne, 2003.

[13] HUNYADI, Levente. Prosper: Un Marco para la Ampliación de Prolog. Octubre, 2005.