Infraestructura para la Incorporación de Metadatos en Sitios Web de Departamentos Universitarios Chilenos

Ernesto Krsulovic-Morales, Claudio Gutiérrez
{ekrsulov,cgutierr}@dcc.uchile.cl
Departamento de Ciencias de la Computación, Universidad de Chile
Blanco Encalada 2120, Santiago, Chile.
Fax: +56 2 6895531

Resumen

Este artículo presenta un catálogo Web local y distribuido de los departamentos de universidades chilenas, su arquitectura e implementación. Está compuesto de una herramienta de marcado en-línea y una interfaz de búsqueda parametrizada por una ontología. Comparado con catálogos Web tradicionales, las ventajas de este enfoque son: 1. Extensibilidad; 2. Datos y ontología públicos para reuso en otras aplicaciones; 3. Acceso distribuido y actualización de datos con una granularidad fina (personas, páginas); 4. Simplicidad para uso por no-expertos. Aparte de estas ventajas, estos catálogos pueden contribuir a poblar la Web con metadatos a través de la reestructuración e integración de información de organizaciones de pequeña escala. Esto es un complemento natural a proyectos de gran escala para construir la infraestructura de la Web Semántica.

Keywords: Web Semántica, Herramienta de Marcado, Anuario, RDF.

1. Introducción

Metadatos. Los metadatos se han utilizado desde hace mucho tiempo en el área de ciencias de la información donde se necesita mantener índices para la catalogación de libros. Entre los esquemas clásicos se encuentra el uso de fichas manuales, que indican título, autor, país, ubicación física, o taxonomías como el código Dewey. Actualmente estas fichas son tratadas utilizando índices electrónicos como MARC. Una definición breve de metadatos es ”datos sobre datos”, es decir, descripciones de características y propiedades de los datos. Por ejemplo, metadatos simples son fecha, autor, lenguaje, tipo de documento, formato, etc. Para áreas específicas, sin embargo, se necesitan lenguajes de metadatos mucho más expresivos. Los requerimientos para el manejo de metadatos son diferentes al información semi-estructurada.

En la Web los datos son los documentos, objetos multimediales y enlaces que la componen. Estos datos se denominan recursos. Los metadatos son afirmaciones sobre los recursos. En índices como YAHOO! se aplican esquemas de catalogación para mantener metadatos de recursos. El problema con estos índices es que para poder abarcar todos los sitios Web utilizan descripciones minimalistas y sólo a nivel de documentos (título, URL, descripción). Además almacenan los metadatos de forma centralizada y agentes automáticos no pueden añadir nuevos recursos al índice sino que estos deben ser añadidos por personas que catalogan los documentos. Los índices como DMOZ han aliviado la labor de catalogación, utilizando editores voluntarios por categorías, pero persiste el problema de ser un índice centralizado con descripciones generales a nivel de documento. Proveer la infraestructura necesaria para que cada sitio Web contenga sus propios metadatos permite mantener la información de catalogación y descripción de manera descentralizada. La solución recomendada por el W3C es utilizar RDF (*Resource Description Framework*)[1].

Modelo de metadatos RDF. El lenguaje RDF fue diseñado para mantener metadatos de forma distribuida e interoperables en la Web. Brinda un mejor modelo para el manejo de datos distribuidos que el de XML por diversas razones. XML aporta semántica a los documentos, al incorporar etiquetas que tratan sobre el significado de la información. Esto es un avance sobre HTML, donde las etiquetas expresan sólo la estructura del documento. Para un dominio particular, XML es una buena alternativa como lenguaje de marcado semántico, utilizando esquemas XML para definir

vocabularios [2], o bien una combinación de XML y RDF [3]. Pero para permitir interoperabilidad entre aplicaciones es necesario soportar una diversidad de dominios. Esto significa que hay que soportar distintos vocabularios y las relaciones a nivel lógico que existen entre ellos. Aquí XML no es suficiente, pues se necesitan vocabularios generales. Un ejemplo de tal vocabulario es Dublin Core, un vocabulario minimal para describir documentos que puede usarse como base para el marcado en la Web. Sin embargo, para aplicaciones en dominios específicos esto no es suficiente, y se hace necesario crear nuevos vocabularios. Estos vocabularios se denominan *ontologías*.

¿Como describir los nuevos vocabularios? Los esquemas RDF (RDFS) establecen un sistema de tipos simples para definir vocabularios RDF - que son, un conjunto de recursos, propiedades y clases. RDFS define una notación de clases y permite la definición y descripción de nuevas clases. Clases y propiedades pueden tener subclases y subpropiedades, y esto permite la definición de jerarquías. Las propiedades pueden ser restringidas a una clase tanto en el dominio como en el rango. RDFS en un vocabulario RDF. Clases y propiedades son iguales que otros recursos, así que pueden ser descritos usando RDF.

Web Semántica. La Web Semántica es la visión que propuso Tim Berners-Lee[1] para la próxima generación de la Web, que presenta y requiere múltiples enfoques [4, 5] en las áreas de base de datos, redes, administración de conocimiento, ciencias de la información y agentes inteligentes [6]. Es el enfoque que se le da en el área de agentes inteligentes el que presenta un mayor atractivo [7], pero donde los resultados que hablan de "transformar a la Web en una gran base de conocimientos [8], utilizada por agentes inteligentes [9, 10, 11]" no se espera obtenerlos en el corto plazo. Sin embargo, muchas de las tecnologías desarrolladas en este camino pueden ser aplicadas hoy, con un alcance menos ambicioso, en el área de integración de información.

El problema de reunir sistemas de información heterogéneos y distribuidos se conoce como el problema de integración. La Web es una red altamente distribuida, donde cada vez más se encuentran, por una parte, sistemas de información heterogéneos y por otra, usuarios que demandan un completo acceso a la información disponible [12].

Contribuciones. Este trabajo trata directamente el nivel de integración semántica. Implementando las ideas directrices de la Web Semántica, en otro lugar [13] se propuso una arquitectura para catálogos descentralizados usando RDF. Este trabajo presenta los resultados y el desarrollo de una implementación de esas ideas, un "servicio de anuario". Puede consultarse en la siguiente dirección `http://purl.org/net/depmark`.

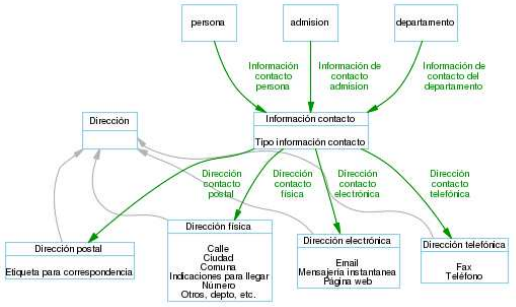
La alternativa clásica para el desarrollo de sistemas de información, es utilizar bases de datos relacionales y mantener un esquema centralizado. En contraposición, el modelo propuesto presenta ventajas al fomentar la descentralización tanto de las fuentes de datos como de los esquemas, y además permite la reutilización de los datos para otras aplicaciones. Además constituye una de las primeras experiencia en el ámbito de aplicaciones de la Web semántica a nivel mundial y la primera en Chile. Junto con obtener un anuario semántico de los departamentos universitarios se adquiere experiencia para desarrolladores de aplicaciones que utilizan el modelo de metadatos en la Web y se obtiene un *testbed* para nuevos proyectos.

Trabajos relacionados. Ejemplos de interoperabilidad, utilizando la tecnología propuesta para la Web semántica, son los que se desarrolla en el área de ciencias de la información, como el proyecto de NDLTD[2], un consorcio de universidades, bibliotecas y otras instituciones mantienen un sistema de tesis electrónicas. Otros son: MusicBrainz [14] centrada en la descripción de música, RSS 1.0 para describir canales de noticias, FOAF para describir personas, Dublin Core para describir documentos, ChefMoz para restaurantes. En el ámbito de departamentos universitarios también existen iniciativas internacionales, como la ontología europea del proyecto AKT [15] o las estadounidenses del proyecto Mangrove [16] y SHOE [17].

Estructura del artículo. En la Sección 2 presentamos algunos conceptos preliminares usados a lo largo del artículo. La Sección 3 describe el desarrollo del proyecto con todo detalle. La Sección 4 trata temas transversales que surgen del proyecto. Finalmente, las conclusiones y proyecciones del trabajo están en la Sección 5.

[1] El creador de la Web y Director del Consorcio de la Web

[2] Incluida la Universidad de Chile, con http://www.cybertesis.cl/



a) Grafo del esquema.

```
<rdfs:Class rdf:about="&depmark;direccion"
  rdfs:label="Direccción">
  <rdfs:subClassOf rdf:resource="&rdfs:Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="&depmark;direccion_postal">
  rdfs:label="Dirección postal">
  <rdfs:subClassOf rdf:resource="&depmark;direccion"/>
</rdfs:Class>
<rdf:Property rdf:about="&depmark;etiqueta_para_correspondencia"
  rdf:label="Etiqueta para correspondencia">
  <rdfs:domain rdf:resource="&depmark;direccion_postal"/>
  <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
```

b) Ejemplo de esquema en RDF/XML.

Figura 1: Sección del esquema RDF para información de contacto.

2. Preliminares

Para aquellos lectores no familiarizados con estos conceptos, describimos aquí brevemente el lenguaje de metadatos RDF y RDFS utilizado, así como una presentación del concepto de ontología.

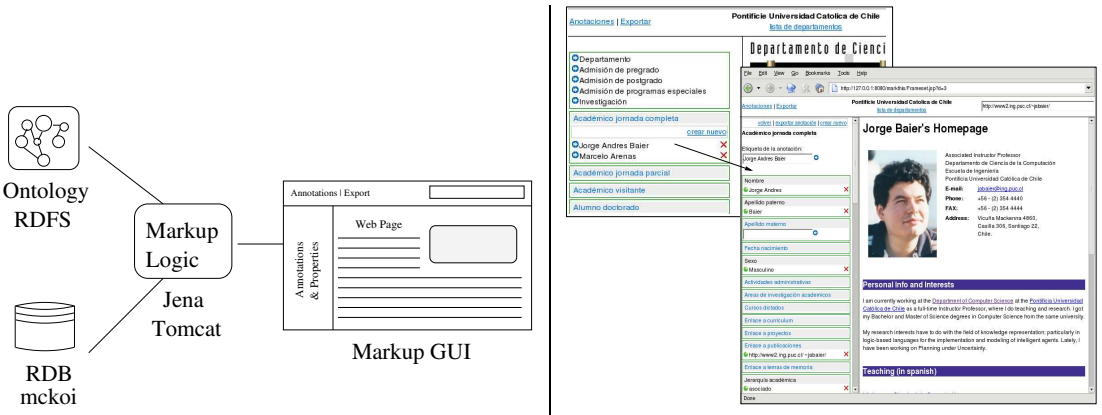
RDF. La sintaxis de RDF es sencilla: una afirmación se compone por triples (una secuencia de tres elementos), que se inspira en los predicados binarios [18], utilizados en la lógica de primer orden: $predicado(sujeto, objeto)$. Cada uno de los elementos de una afirmación son identificados mediante un URI [19], un esquema de identificación universal de recursos. El objeto puede no ser un URI, permitiendo tipos de datos para representar documentos, valores, etc. (denominados literales). Una de las representaciones clásicas del modelo RDF es el de un grafo dirigido y etiquetado. Quizás esta es la notación más pedagógica para describir el modelo. Pero existen también otras representaciones, denominadas serializaciones, como **RDF/XML** donde XML es utilizado como medio de transporte o sintaxis; **N-TRIPLES** [20] que es sólo un listado de los triples; **Notation 3** [21], una notación que alivia el problema de lo difícil que resulta para los desarrolladores la serialización en XML.

RDF Schema El vocabulario RDFS indica cómo describir el uso de los términos definiendo propiedades y clases. Por ejemplo, un vocabulario RDF puede describir limitaciones en el tipo de valores que son apropiados para algunas propiedades, o las clases para las cuales tiene sentido atribuir tales propiedades.

Una característica de RDFS es que *no predetermina cómo* una aplicación debe utilizar la información que RDFS describe. Por ejemplo, mientras un esquema RDF puede afirmar que cierta propiedad del autor está utilizada para indicar los recursos que son instancias de clase persona, no dice cómo una aplicación debe procesar esa información para el rango. Distintas aplicaciones utilizarán esta información de maneras distintas. Por ejemplo, una herramienta de verificación de datos; puede usar esto para ayudar a encontrar errores en algún conjunto de datos, un editor interactivo puede sugerir los valores apropiados; y una aplicación de razonamiento puede usar esto para inferir información adicional desde instancias de datos.

Ejemplo En la figura 1.a se muestra un esquema RDF para describir información de contacto usado por una persona, un departamento y áreas de admisión; en la figura 1.b se muestra una sección de la serialización de ese esquema en RDF/XML abreviado, donde se definen las clases *direccion* y *direccion_postal* (subclase de *direccion*) además de la propiedad *etiqueta_para_correspondencia* (con dominio *direccion_postal* y rango un literal).

Ontologías Ontología, en el área de la filosofía, es una teoría sobre la naturaleza de la existencia. Los investigadores de Inteligencia Artificial (especialmente del área de representación y adquisición del conocimiento) reencarnaron este término en su propia jerga para expresar "un entendimiento compartido y común sobre un dominio que puede ser comunicado entre personas y aplicaciones". Una ontología típica tiene una taxonomía que define clases y sus relaciones, y un conjunto de reglas de inferencia. Visto simplemente, cualquier conjunto organizado de objetos puede ser considerado una ontología de acuerdo a la definición de ontología anterior. Por ejemplo: catálogos; índices de la comunidad



a) Arquitectura de la herramienta b) Secuencia de marcado

Figura 2: Herramienta de marcado

de IR; modelos de entidad relación de la comunidad de bases de datos; diccionarios, tesauros desde la comunidad de ciencias de la información; y definiciones de clases en orientación a objetos desde la comunidad de ingeniería de software.

3. Desarrollo

El proyecto realizado, cuyo nombre es DepMark, es un sistema de información desarrollado sobre un modelo de metadatos distribuidos. El resultado del proyecto son dos componentes independientes: una herramienta de marcado y un sistema de despliegue de información Web basado en metadatos.

La solución propuesta al problema de recolectar la información es publicar en la Web metadatos de cada componente perteneciente a la organización, para que puedan ser extraídos y procesados desde el servicio de anuario. La arquitectura es discutida en [13].

3.1. Herramienta de Marcado

Permite la captura manual de información desde los sitios Web de los departamentos. No se trata de una herramienta típica de anotaciones de sitios Web, las cuales se entienden como sistemas colaborativos para agregar comentarios o notas a las páginas, típicamente implementadas como extensión de un Web *browser* o como *proxies* de marcado. Sin embargo, se optó por utilizar el concepto de anotaciones como metáfora del sistema, ya que puede ayudar tanto al desarrollo [22] como a los usuarios que no son expertos en representación del conocimiento ni menos familiarizados con RDF cuya jerga incluye conceptos como clases, instancias y propiedades.

El tema de marcado basado en esquemas RDF es tratado en [23], donde se describen las características que presenta RDF para proporcionar opciones de marcado de acuerdo al esquema. Entre las características que se deben considerar están: el rango y dominio de las propiedades; con ello es posible diagramar diferentes *widgets* para añadir los valores de estas propiedades. Se optó por implementar un sistema Web, que permita estructurar la información existente en las páginas Web de los distintos departamentos, de manera manual.

En una primera instancia se intento utilizar el concepto de *proxy de marcado,* el que permite añadir anotaciones modificando la página Web, así como lo hace crit.org y coment.it pero fue desechada, por la dificultad que presenta para sitios Web que contienen elementos que no son puramente HTML (javascript y flash en particular). Por esto, la característica de presentar el sitio Web de cada departamento, es provista directamente por el browser, utilizando un *frame*. Como el sistema está implementado sobre frames HTML, este falla cuando hay páginas Web que tienen enlaces con *target* "_top" (e.g. `<a href=''pagina.html'' target=''_top''>link</a>`), lo que tienen como efecto que la nueva página, al hacer *click* en el enlace, utilice todo el área del *browser.* Esto es una limitación del sistema.

El requerimiento de agregar tanto anotaciones como valores de propiedades implica también, el poder modificar

o eliminar. Se decidió, por simplicidad, que el único campo modificable es la etiqueta de una anotación. Si se quiere modificar el valor de una propiedad se deberá eliminar y luego ingresar el valor actualizado.

El vocabulario de RDFS no está orientado a restringir las instancias de los esquemas, a diferencia del objetivo de un esquema en una base de datos relacional, un DTD o un esquema XML. En particular no presenta funcionalidades para restringir la cardinalidad ni establecer el orden de las propiedades de una clase, en la versión utilizada no se tiene incorporado aún tipos de datos (la versión actual utiliza tipos de datos de XML Schema). Dadas estas restricciones se optó por crear un pequeño esquema, que permitiera anotar las propiedades de la ontología con mayores restricciones y así facilitar la creación de *widgets* ad-hoc a cada propiedad. Además se reutilizaron anotaciones provistas por el esquema *protégé*.

La incorporación de metadatos en las páginas Web no se realizará de manera automática por la herramienta, característica deseable por los usuarios. Sin embargo, los metadatos pueden ser exportados para luego ser añadidos manualmente a las páginas Web, siguiendo una de las siguientes alternativas:
1. Referenciar un archivo con los metadatos desde páginas HTML con una etiqueta link
`<link rel="meta" type="application/rdf+xml" href="metadata.rdf" />`.
2. Agregar directamente los metadatos en páginas XHTML.

Resultado La herramienta de marcado tiene la arquitectura presentada en la figura 2.a, donde se destaca los componentes principales que son: la ontología en RDFS, el motor de bases de datos, la lógica de marcado (compuesto de una API basada en *Jena* [24] y páginas *jsp*) y la GUI de marcado.

El sistema desarrollado presenta las opciones de selección de departamento, para luego desplegar la GUI de marcado con opciones de crear nuevas anotaciones y agregar o eliminar valores de propiedades de las anotaciones. Un ejemplo de la secuencia de marcado es presentada en la figura 2.b, donde se selecciona una instancia para agregar valores a sus propiedades.

Anotaciones extras a RDFS utilizadas La interfaz es construida dinámicamente basada en el esquema con anotaciones provistas por *Protégé* y otras diseñadas particularmente para este propósito. El siguiente listado muestra las propiedades del esquema de *Protégé* que fueron usadas

protege:role Indica si el rol de una clase es abstracto, para las cuales no se pueden crear instancias.

protege:range Indica si el rango de una propiedad es *class* o *symbol*. Para propiedades que tienen rango de tipo class, se permiten solamente valores de una clase o subclase. Para las propiedades con rango de tipo *symbol*, se permiten solamente literales con valores definidos por *protege:allowedValues.*

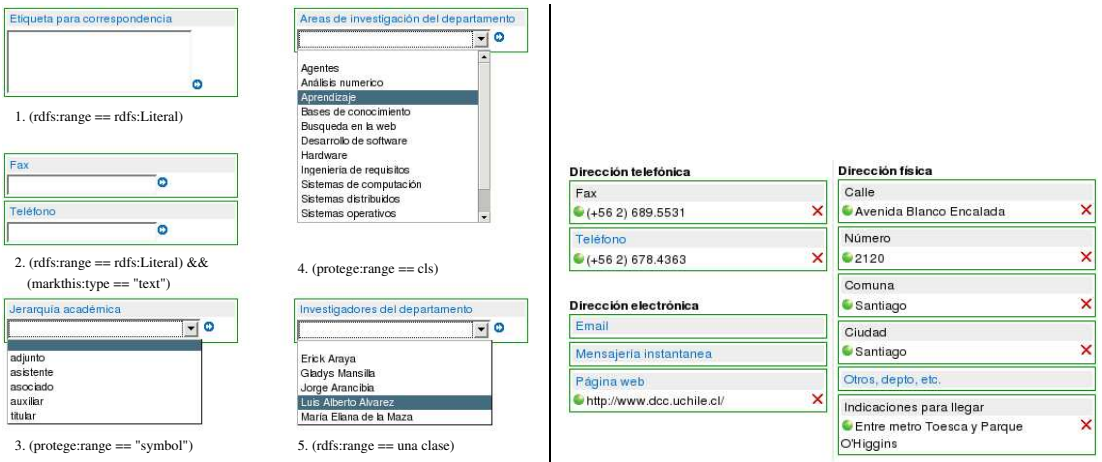
protege:allowedValues Indica los valores permitidos para propiedades con *protege*:*range* de tipo *symbol*.

El esquema creado para facilitar la construcción de la interfaz presenta las siguientes propiedades:

markthis:cardinality Indica la cardinalidad de una clase. En la ontología, esto indica cuales clases que pueden tener sólo una instancia para un departamento (por ejemplo, investigación o admisión). Si una clase tienen cardinalidad igual a uno, la interfaz permite sólo una anotación de este tipo.

markthis:weight Indica el peso que tiene una propiedad para su diagramación en el formulario de marcado. Esto permite ordenar los campos del formulario según el siguiente criterio: se crea una lista con todos las propiedades de una clase, y se le asigna como nombre el valor de "markthis:weight" (un *string* de tamaño cero si no lo tiene) concatenado con "rdfs:label", la lista de propiedades es ordenada por el nombre asignado. Si un esquema no tiene la propiedad "markthis:weight" se ordena alfabéticamente por la etiqueta de la propiedad.

markthis:type Indica el tipo de widget a diagramar para una propiedad. Si es de tipo *text*, se presenta un widget con un campo de texto de una línea.



a) Tipos de *widgets* para valores de propiedades.

b) Ejemplo de clases incrustadas en una instancia.

Figura 3: Widget y Clases incrustadas

Cardinalidad de las clases y clases abstractas Para la creación de anotaciones se toma en consideración la cardinalidad que pueden tener las clases para un departamento. Para el caso de las clases con cardinalidad unitaria se accede directamente a la pantalla de selección de propiedades de la instancia. En cambio, para las clases con cardinalidad n-aria, se presenta la opción para crear una nueva opción de ese tipo y un listado para seleccionar la anotación a editar. Para las clases abstractas como son persona o académico no se permite crear anotaciones de ese tipo, sino que de los subtipos que no son clases abstractas.

Widget para valores de propiedades Al crear una nueva anotación o al seleccionarla para su edición (ya sea de una clase n-aria o de una clase 1-aria) se presenta una pantalla (producida por la página Instance) que permite modificar el nombre de la instancia (rdfs:label) y despliega las propiedades que posee. Estas propiedades son las que tienen como dominio (rdfs:domain) a la clase y a las clases de las cual hereda. Dependiendo del rango de las propiedades se presenta un widget diferente, como se muestra en la figura 3.a que se explica a continuación:

1. Propiedades con rango literal (rdfs:range == rdfs:Literal): para estas propiedades se presenta un cuadro de texto para el ingreso de su valor.

2. Propiedad con rango literal (rdfs:range == rdfs:Literal) y widget de texto simple (markthis:type == text): para estas propiedades se presenta un cuadro de texto de una sola línea.

3. Propiedades con rango symbol (protege:range == symbol): para estas propiedades se presenta una lista de valores permitidos que se han definido en el esquema (con protege:allowedValues)

4. Propiedades con rango una clase (protege:range == cls): para estas propiedades se presenta un cuadro de selección, donde se listan las clases que pueden ser asociadas con esta propiedad.

5. Propiedades con rango instancia de una clase (rdfs:range == un recurso): para estas propiedades se presenta un cuadro de selección con las instancias creadas de la clases (el valor dado en rdfs:range, cuando es distinto a rdfs:Literal y no tiene un protege:range definido) que establece el rango de la propiedad.

Para las clases con cardinalidad n-aria, que son requeridas por otras instancias, se incrustaron las propiedades de las clases en las de la pantalla de propiedades de las instancias. Un ejemplo es el caso de la clase información de contacto, la que a su vez requiere las clases de tipo dirección. Para las clases incrustadas se presenta directamente las propiedades en la instancia. Esto ocurre para las siguientes clases y sus subclases: departamento, admisión, persona; donde se incrustaron las propiedades de información de contacto: direcciones y tipo de información. En la figura 3.b se presenta un ejemplo de clases incrustadas para la instancia de una clase.

RDFSTree La versión de la API *Jena* utilizada no soporta inferencia (una versión en desarrollo, sí proveerá esta funcionalidad), por lo que se desarrolló un módulo denominado RFSTree que mantiene en memoria la representación de un árbol para la jerarquía de clases de un esquema RDFS y presenta funcionalidades para consultar las subclases y superclases.

Optimizaciones de metadatos al exportar Para facilitar el marcado, se realizan optimizaciones al exportar los datos: primero se eliminan clases sin valores que fueron creadas para las clases incrustadas (información de contacto); luego se crean automáticamente las propiedades para las propiedades de la clase docencia (carreras del departamento, cursos del departamento y profesores del departamento) y una propiedad de la clase departamento (personas del departamento).

Prueba de la herramienta de marcado con otra ontología Como una prueba de la herramienta de marcado, se utilizó la ontología FOAF, que es equivalente (en complejidad) a la ontología propuesta para los departamentos universitarios, pero con un alcance más general en la descripción de personas. A partir del esquema FOAF la herramienta permite crear instancias de la clase foaf:person, pero no presenta todas las propiedades esperables para la clase ya que el esquema, aparentemente[3], no está completo (faltan clases y en muchos casos no se señala el dominio ni el rango de la propiedades). Al completar el esquema y agregar los atributos de *Protégé* y Markthis utilizados, la herramienta permite crear instancias para todas las clases del esquema.

La intervención del esquema al definir dominios es válido de acuerdo a la especificación de RDFS, la cual señala que la interpretación de esas propiedades está abierta para cada aplicación, pero en especificaciones ligadas a OWL [25] puede provocar problemas. Señalar un dominio de propiedades con múltiples clases puede conllevar a inconsistencias para un razonador basado en las reglas de clausura de RDF. Un ejemplo es la propiedad foaf:homepage, que en el esquema original no tiene domino, a la cual se agregaron las clases foaf:Project, foaf:Person y foaf:Organization como dominio. Si se aplica sobre el esquema modificado la regla de inferencia rdfs2, de la especificación de semántica para RDF [26], se puede concluir información errónea como: "todos los proyectos son también personas y organizaciones". La regla rdfs2 señala que sí una propiedad p tiene como dominio una clase c, entonces cualquier recurso que tenga la propiedad p es de tipo c, esta regla puede ser escrita ($?p$ rdfs:domain $?c$) $\rightarrow$ [($?x$ rdf:type $?c$) $\leftarrow$ ($?x$ $?p$ $?y$)].

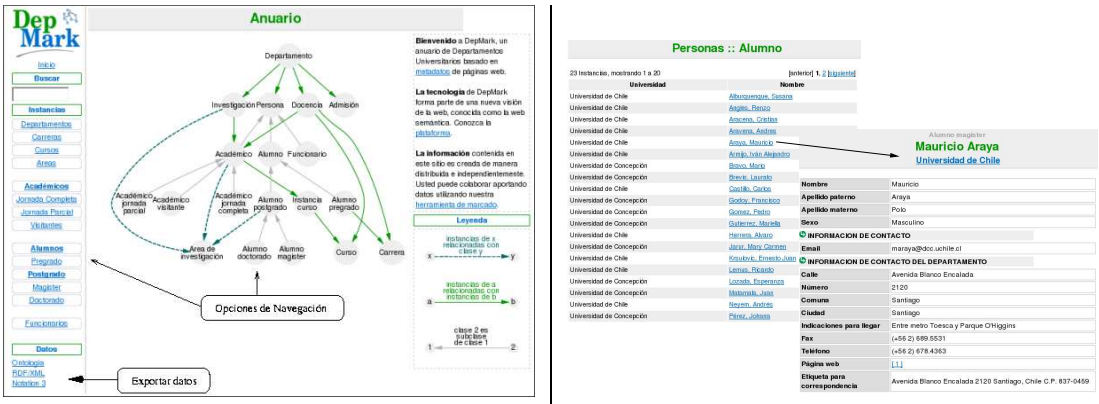
3.2. Anuario

El principal requerimiento para el componente de anuario es presentar la información recopilada en los contenedores RDF (idealmente desde los sitios Web, pero experimentalmente, desde los archivos producidos por la herramienta de marcado), de manera agregada en un sitio Web.

La visión del módulo de anuario es distinta a la requerida para la herramienta de marcado, ya que es un sistema de despliegue y no hay requerimientos para agregar o eliminar triples RDF en un contenedor. Es así como se optó por desarrollar una plataforma de publicación basada en metadatos. La alternativa clásica para la publicación en el caso de sitios guiados por documentos XML es la utilización de XSL, pero esta alternativa fue desechada por la diferencia existente entre el modelo RDF y XML. Además, tomando en cuenta que ya se contaba con experiencia y desarrollo realizado para el módulo de marcado, y la familiarización con la API *Jena*, se optó por seguir en la línea de aplicaciones Web basada en *jsp*. Se decidió crear un "juego" de *jsp custom tag libs* [27] para facilitar la creación de páginas Web basadas en un repositorio de metadatos RDF. El *crawler* de metadatos desarrollado es una versión ideal, ya que los metadatos no están publicados en los sitios de los departamentos, sólo se extraen desde la herramienta de marcado.

Resultado El resultado del desarrollo es un sitio Web que presenta opciones para navegar sobre la información "recopilada" desde las páginas marcadas. La página inicial presenta las opciones de navegación mediante un mapa sensible HTML en el cliente (CMAP), el cual es una versión simplificada de la ontología y fue generado desde el esquema mediante XSL y Graphviz, como se muestra en la figura 4.a. Además se presentan opciones de navegación a base de links textuales (para soportar browsers sin funcionalidades de CMAP). También se dan funcionalidades para exportar en formato RDF y Notation3. El sitio Web sigue un esquema de listados con paginación y fichas de detalle, los cuales están disponibles para los departamentos, carreras, personas, áreas y cursos. En la figura 4.b se muestra el resultado de un ejemplo de listado para alumnos y la ficha de un alumno.

[3] Como se señala en 2, la interpretación de los esquemas son abiertos y dependen del uso que den las aplicaciones.



a)Pagina principal del anuario b) Listado de alumnos y ejemplo de ficha

Figura 4: Anuario

Crawler de metadatos El *crawler* de metadatos fue implementado a base de una lista de direcciones URL cuyo contenido es descargado a un archivo diferente. Antes de cada ejecución del *crawler* los metadatos existente son transferidos a un directorio distinto para mantener un registro histórico.

RDF Custom Tag Lib Como plataforma para la presentación de contenidos desde una fuente de datos RDF se crearon los siguientes *jsp custom tag lib*, una serie de componentes reutilizables para agregar a páginas *jsp*. Todos los *tags* utilizan un mismo contenedor RDF (una instancia de la clase Model en la API de *Jena*). La configuración del contenedor se realiza a nivel de la aplicación utilizando el contexto (esto para permitir su uso en otras aplicaciones).

Buscador El buscador fue implementado como un índice sobre las páginas estáticas. Se utilizó la API *lucene* [28] con el lematizador en español de *Snowball* [29]. Para simplicidad, este módulo puede ser reemplazado con opciones de búsqueda provistas por Google o Todo.cl.

4. Discusión de temas transversales

Instancias de múltiples tipos Uno de los problemas típicos, visto en los cursos de modelamiento de datos, es el de una persona que trabaja en una universidad, quien puede cumplir múltiples roles: ser alumno, funcionario y profesor, todo al mismo tiempo. La herramienta de marcado permite crear instancias con múltiples tipos, gracias a que la URI es dependiente de la etiqueta (*label*).

Por ejemplo, para crear la instancia para un alumno de doctorado, que además es académico jornada parcial, se debe crear primero una anotación para uno de los roles. En tal caso, se puede ingresar la etiqueta y los valores para las propiedades de la anotación como alumno de doctorado; y posteriormente crear una nueva instancia como académico jornada parcial con los valores para las propiedades, pero utilizando la misma etiqueta ingresada para el rol de alumno.

En el anuario, las instancias de múltiples tipos se transforman en una sola ficha con las valores de propiedades de cada uno de los tipos de la instancias. Además la instancia es presentada en todos los listados de los tipos a los que pertenece.

Nombre de las URI En la herramienta de marcado, las URI son impuestas por el sistema y son configurables por departamento a partir de una URI base. Por ejemplo, para el departamento de la Universidad de Chile la base es http://www.dcc.uchile.cl/depmark#; para cada instancia la URI asignada es la versión *UpperCamelCase*, sin acentos ni eñes, de la etiqueta que se ha ingresado para dicha instancia. Por ejemplo, para el Alumno de Pregrado José Muñoz del departamento de la Universidad de Chile la URI asignada es http://www.dcc.uchile.cl/depmark#JoseMunoz. Un caso particular son las clases anidadas utilizadas para la información de contacto. Para ellas se generan URIs del tipo http://udec.cl/depmark#JoseZae_DireccionElectronica. Esto puede ser omitido utilizando nodos blancos. En la versión

dinámica del anuario, la URI es entregada, como un parámetro codificado, a la página que despliega la información de la instancias (una ficha); pero en la versión estática los nombres son transformados para ser URL válidas y compatibles con los servidores Web.

Privacidad de la información Todo el contenido del sitio de anuario desarrollado se ha extraído desde las páginas Web, las cuales ya están publicadas. En particular, se publica información que puede dar pie a discriminación, como la fecha de nacimiento y el sexo, debido a que sólo se publica a partir de los metadatos, la fecha de nacimiento solamente se publicará si la información es pública en al menos una página Web. Al momento de realizar el marcado, el sexo es extrapolado a partir del nombre, sólo cuando es completamente evidente.

Reutilización de ontologías Una de las recomendaciones más frecuentes en el ámbito de aplicaciones para la Web semántica, es la reutilización de ontologías. En nuestro caso, tanto para la ontología como para las herramientas desarrolladas, no se siguió esta recomendación por simplicidad, debido a las particularidades de la realidad nacional y teniendo en cuenta las facilidades para realizar equivalencias a nivel ontológico, que es provisto por OWL. Sin embargo, para impulsar una mayor adopción, tanto de la herramienta de marcado como de la infraestructura de publicación, se hace necesario experimentar en la mezcla de ontologías. Por ejemplo, en una versión internacional de la ontología se podría tomar como base para la sección de personas el esquema FOAF, para la sección de información de contacto *Contact Schema* del W3C [30] y para departamentos universitarios la ontología europea del proyecto AKT [15] o las estadounidenses del proyecto Mangrove [16] y/o SHOE [17].

5. Conclusiones

Se logró desarrollar un esquema conceptual para la organización de Departamentos Universitarios y las herramientas necesarias para la implementación de un anuario distribuido basado en metadatos RDF. Este enfoque tiene ventajas sobre el enfoque clásico, especialmente al fomentar la descentralización y permitir el acceso a las fuentes de datos.

El tener herramientas prácticas y de fácil uso, que son dirigidas por una ontología, permiten gatillar el *feedback* directo por parte de la comunidad sobre la que trata la ontología.

Con las herramientas existentes para el desarrollo de sistemas basados en RDF, es posible crear aplicaciones como las producidas durante el trabajo descrito. Desgraciadamente la amplia movilidad en las especificaciones de las tecnologías utilizadas, produce que las herramientas (APIs y lenguajes) no están implementadas con un nivel de optimización adecuadas para llevarlas a producción, sino que más bien, se encuentran resolviendo los problemas que se derivan de ceñirse a las especificaciones que las guían. En particular se hace necesario tener una interfaz de consulta amigable para RDQL y una herramienta que permita hacer *diff* de contenedores RDF. Esto es útil tanto para mantener las distintas versiones de los contenedores para el anuario, como también para implementar el marcador con control de versiones y así actuar como un *wiki*.

Proyecciones y trabajo futuro Se presenta como un gran desafío el transformar la ontología desarrollada en una versión oficial por parte de la Sociedad Chilena de Ciencias de la Computación, que es la responsable natural para la comunidad tratada.

El trabajo futuro pasa por adaptar la herramienta de marcado a ámbitos de aplicación diferentes y usar lenguajes de esquemas diferentes (como son DAML y OWL), para así obtener la experiencia necesaria en el desarrollo de la herramienta como un “Framework” de marcado.

El hecho que una gran cantidad de las propiedades de la ontología sean del tipo “enlace a”, no solamente permite la extensión del sistema para mantener marcado distribuido, sino que también puede ser utilizado para ayudar al marcado asistido. Si tomamos como ejemplo la propiedad “enlace a becas” que tiene la clase admisión de todos los departamentos, tenemos aquí una clasificación de todas las páginas que tratan sobre becas y se puede construir una herramienta semiautomática de marcado. Por ejemplo, utilizando un clasificador *bayesiano* y realizando un proceso de aprendizaje de las páginas ya clasificadas para sugerir nuevas clasificaciones.

El sistema desarrollado asume una condición bastante ideal: el que todos los sitios Web de los departamentos tendrán en una URL específica todos los metadatos de su sitio Web. Esto es claramente impracticable (debido a la dificultad para mantener un archivo donde se centralizan todos los metadatos), pero por el momento es una solución

parcial para enfrentar la realidad que son muy pocos los que tienen su página con metadatos. Una perspectiva más realista, si la conducta de poner metadatos a las páginas Web se transforma en una tarea cotidiana para los desarrolladores de contenidos, será accesar a las recolecciones con un crawler como Todo.cl, el cual recolectará los metadatos RDF que tienen relación con el sistema de anuario.

Agradecimientos. Financiado parcialmente por Núcleo Científico Milenio, Centro de Investigación de la Web, Grant P01-029-F, Mideplan, Chile.

Referencias

[1] Ora Lassila and Ralph Swick. Resource Description Framework (RDF) Model and Syntax Specification. http://www.w3.org/ TR/ 1999/ REC-rdf-syntax-19990222/, Feb 1999.

[2] Varun Ratnakar and Yolanda Gil. A Comparison of (Semantic) Markup Languages. *Proceedings of the 15th International FLAIRS Conference, Special Track on Semantic Web*, May 2002.

[3] Peter Patel-Schneider and Jérôme Siméon. The Yin/ Yang Web: XML Syntax and RDF Semantics. In *The Eleventh International World Wide Web Conference, WWW2002*, May 2002.

[4] Vladimir Geroimenko. Bringing sense to the web. Aplication Development Advisor, Mar 2002.

[5] Michael R. Genesereth and Steven P. Ketchpel. Software Agents. *Communications of the ACM*, 37(7), 1997.

[6] Michael Wooldridge. Intelligent Agents. In Gerhard Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 27–78. The MIT Press, Cambridge, MA, USA, 1999.

[7] Otis Port. The Next Web. BusinessWeek, March 2002.

[8] Dieter Fensel and Mark A. Musen. The Semantic Web: A Brain for Humankind. *IEEE Intelligent Systems Journal*, 16(2):24–25, Mar 2001.

[9] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. Scientific American, May 2001.

[10] Poul Ford. August 2009: How Google beat Amazon and Ebay to the Semantic Web. http://www.ftrain.com/ google_takes_all.html, Jul 2002.

[11] James Hendler. Agents and the Semantic Web. *IEEE Intelligent Systems Journal*, 16(2):24–25, Mar 2001.

[12] H. Wache, V. Ogele, T. Visser, U. Stuckenschmidt, H. Schuster, G.Ñeumann, and H. Ubner. Ontology-Based Integration of Information A Survey of Existing Approaches. In H. Stuckenschmidt, editor, *IJCAI-01 Workshop: Ontologies and Information Sharing*, pages 108–117, Seattle, USA, Aug 2001.

[13] Ernesto Krsulovic and Claudio Gutiérrez. Building Yearbooks with RDF. In A. Abraham et al., editor, *Soft Computing Systems: Design, Management and Applications*, pages 593–601. IOS Press, Dec 2002.

[14] Aaron Swartz. MusicBrainz: A Semantic Web Service. *IEEE Intelligent Systems*, Jan 2002.

[15] AKT. Hyphen.info, an information source for UK Researchers. http://www.hyphen.info/ .

[16] Mangrove: An Evolutionary Approach to the Semantic Web. http://www.cs.washington.edu/ research/ semweb/ .

[17] Parallel Understanding Systems Group, Department of Computer Science, University of Maryland at College Park. SHOE and DAML. http://www.cs.umd.edu/ projects/ plus/ DAML/ .

[18] John F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, CA, 2000.

[19] Tim Berners-Lee, R. Fielding, U.C. Irvine, and L. Masinter. URI Generic Syntax. Network Working Group, RFC 2396, Aug 1998.

[20] Dave Beckett and Art Barstow. N-Triples. http://www.w3.org/ 2001/ sw/ RDFCore/ ntriples/, May 2001.

[21] Tim Berners-Lee. Notation 3, Ideas about Web Architecture - yet another notation. http://www.w3.org/ DesignIssues/ Notation3.html, October 2000.

[22] System Metaphor. http://c2.com/ cgi/ wiki?SystemMetaphor.

[23] Tobias Kunze, Jan Brase, and Wolfgang Nejdl. Editing Learning Object Metadata: Schema Driven Input of RDF Metadata with the OLR3-Editor. In *Proceedings of Semantic Authoring, Annotation and Knowledge Markup Workshop, 15th European Conference on Artificial Intelligence*, Lyon, France, July 2002.

[24] Brian McBride. Jena: A Semantic Web Toolkit. DS Online ISSN: 1541-4922 http://dsonline.computer.org/ 0211/ f/ wp6jena.htm, Nov 2002.

[25] Mike Dean, Dan Connolly, Frank van Harmelen, James Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language 1.0 Reference. http://www.w3.org/ TR/ owl-ref/, July 2002.

[26] Patrick Hayes. RDF Semantics. http://www.w3.org/ TR/ 2003/ WD-rdf-mt-20030123/, Jan 2003.

[27] Marty Hall. *Core Servlets and JavaServer Pages (JSP)*, chapter 14: Creating Custom JSP Tag Libraries, pages 308–350. Prentice Hall PTR, first edition, May 2000.

[28] Jakarta, lucene. http://jakarta.apache.org/ lucene/ docs/ index.html.

[29] Snowball: A language for stemming algorithms. http://snowball.tartarus.org/ .

[30] W3C. Contact schema. http://www.w3.org/ 2000/ 10/ swap/ pim/ contact, Oct 2000.