Extracción de elementos de una ontología del dominio a partir de documentos tipo esquema

Extracting elements of a domain ontology from scheme documents

Gloria Lucía Giraldo G[1], PhD., Juan Carlos Marín[2], Ing., & German Urrego Giraldo[3], PhD.
1. Profesora Asociada. Grupo de Investigación en Lenguajes Computacionales. Escuela de Sistemas. Facultad de Minas. Universidad Nacional de Colombia, sede Medellín.
2. Ingeniero. Analista de soluciones de TI.
3. Grupo de investigación ITOS-Ingeniería y Tecnología de las Organizaciones y la Sociedad. Departamento de Ingeniería de Sistemas. Universidad de Antioquia.
glgiraldog@unalmed.edu.co; jucamari@bancolombia.com; gaurrego@udea.edu.co

Recibido para revisión 02 de Febrero de 2009, aceptado 25 de Agosto de 2009, versión final 21 de Septiembre de 2009

Resumen— **El presente trabajo, se sitúa en el campo del aprendizaje de ontologías (Ontology Learning), el cual investiga técnicas, automáticas o semiautomáticas, para la construcción de ontologías. Generalmente, la construcción de ellas, presenta dos tipos de problemas, uno de modelado y otro de representación. Este artículo, propone una técnica que aporta en la solución del primer tipo de problema. La técnica propuesta consiste en aplicar ciertas heurísticas, a un conjunto de documentos de tipo esquema XML relativos a un mismo dominio. La idea de la propuesta se funda en dos aspectos. El primero, es el hecho que XML es actualmente, el lenguaje más utilizado para intercambiar información en la Web. El segundo, es que las personas que generan los documentos esquema, utilizan términos del dominio para nombrar las etiquetas de sus documentos. Especialmente, cuando las organizaciones que utilizan dichos documentos, se rigen por un organismo de estandarización o normalización, se espera encontrar en ellos, importante información representativa del dominio.**

Palabras Clave— **Aprendizaje Ontologías, XML, Documentos Esquema.**

Abstract— **This work is relevant to the field of Ontology Learning, which investigates automatic and semiautomatic techniques for the construction of ontologies. Generally, construction of ontologies has two types of problems, a modeling problem and a representation problem. This article proposes a technique that brings in the solution of the first type of problem. The proposed technique applies certain heuristics to a set of XML schema documents related to the same domain. The basic idea for the proposal is based on two aspects. The first is the fact that XML is currently the most widely used language for sharing information on the Web. The second is that people, who generate XML schema documents, use terms of the domain to name tags on their documents. Especially, when the organizations that use those documents, are governed by a body of standardization and normalization, is expected to find in them important information representative of the domain.**

Keywords— **Ontology Learning, XML, Schema Documents.**

I. INTRODUCCIÓN

Actualmente, las aplicaciones de las ontologías son múltiples. Ellas son ampliamente usadas en: gestión de conocimiento, procesamiento de lenguaje natural, comercio electrónico, integración y recuperación inteligente de información, educación, la Web Semántica, entre otros. El concepto de Ontología, proveniente de la filosofía, es tomado por la comunidad de Inteligencia Artificial, a principios de los años noventa, en el marco de la adquisición de conocimiento para los Sistemas a base de conocimiento (SBC). Las ontologías son descripciones formales de los conceptos y las relaciones que intervienen en un dominio o área de interés [11]. Aunque en los SBC las ontologías tienen un objetivo muy importante, el cual es, reducir el esfuerzo durante el proceso de adquisición de conocimiento, adquirir conocimiento para construir una ontología desde cero o para refinar una ontología ya existente, requiere mucho tiempo y recursos.

El aprendizaje de ontologías (*Ontology Learning*), el cual hace parte de las metodologías clásicas para el desarrollo de ontologías, se ocupa de investigar técnicas y métodos para la automatización parcial de la adquisición de conocimiento, en el proceso de construcción de las ontologías.

En [10], los autores clasifican los métodos propuestos por el aprendizaje de ontologías en: aprendizaje desde corpus de texto, aprendizaje desde instancias, aprendizaje desde esquemas y aprendizaje de *mappings* semánticos. El método más conocido, es el aprendizaje desde textos. Estos últimos siendo, normalmente, representativos de un dominio, preparados para ser procesados por un computador y aceptados por un conjunto de expertos del dominio [5]. Las técnicas para el aprendizaje desde corpus de textos, son complejas que implican procesamiento de lenguaje natural.

Trabajos como el realizado por Giraldo en [8], utilizan reglas heurísticas para obtener los componentes de una ontología a partir de un conjunto de documentos DTD (*Description Type Document*). Dado que los DTDs presentan limitaciones en cuanto a definición de tipos y de restricciones, en [8] se reporta una limitada extracción de relaciones, particularmente de relaciones de generalización / especialización. En el 2001, el consorcio W3C crea los esquemas XML con el fin de superar las limitaciones propias de los DTDs. Actualmente, ellos son los más utilizados para definir la estructura de los documentos XML en la Web.

En la línea de trabajos que explota esquemas XML se encuentra el presente artículo, el cual propone una técnica, basada en heurísticas, para automatizar la identificación de los elementos componentes de una ontología del dominio, a partir de un conjunto de documentos esquema asociados a documentos XML.

La propuesta que se presenta en este artículo constituye la base para un trabajo en curso, el cual aplica las heurísticas propuestas, al dominio del turismo, utilizando los esquemas XML definidos por la Open Travel Alliance (OTA) [12], consorcio que agrupa más de 150 organizaciones, entre las cuales se encuentran aerolíneas, empresas dedicadas al alquiler de vehículos, hoteles, proveedores de servicios, agencias de viajes y operadores de tours, entre otras. Este consorcio se ocupa principalmente de definir una estructura normalizada de mensajes, los cuales son, de hecho, un estándar que facilita la interoperabilidad entre los diferentes agentes de este dominio.

Este artículo tiene la siguiente estructura. En la sección 2, se presentan algunos conceptos importantes para la comprensión del resto del artículo. La sección 3 presenta los trabajos relacionados con el nuestro. En la sección 4, se explica la solución propuesta y por último, en la sección 5, se muestran las conclusiones y el trabajo futuro.

II. MARCO TEORICO

A. Ontología

Ontología, en filosofía, es una rama de la metafísica, conocida como la teoría del Ser. En este contexto, ella se ocupa de la definición del Ser y de establecer las categorías fundamentales de ser de las cosas, a partir del estudio de sus propiedades. En el medio informático, el término ontología, es prestado de la filosofía y redefinido en 1991 por Neches et al. [14], de la siguiente manera: "An ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary".

En 1998, Studer et al. [17] define el término ontología, a partir de la definición hecha por Gruber [11] y Borst [1], quien establece que: "una ontología es una especificación explícita y formal de una conceptualización compartida". Ésta, es una de las definiciones de ontología más referenciadas en la literatura.

Según Uschold y Jasper, las ontologías necesariamente incluyen un vocabulario de términos y una especificación de su significado (definiciones e interrelaciones entre conceptos), que impone estructura al dominio y restringe las posibles interpretaciones [18].

Las ontologías proporcionan modelos conceptuales para describir un dominio de interés y para ello utilizan un vocabulario común. Incorporar ontologías a las tecnologías actuales de búsqueda sobre la Web, se traduce en la posibilidad de interrogar las fuentes de información, de una manera más apropiada, que la simple búsqueda por palabras claves. Las búsquedas actuales, consideran poca, o ninguna, información semántica y a pesar de que la mayoría de las veces retornan la informacion deseada, ésta se encuentra inmersa en una gran cantidad de información inútil. Una solución posible a este problema consiste en definir ontologías para cada dominio y describir en términos de esas ontologías, el contenido de las páginas Web. Así, un usuario (o agente de software) podrá navegar en las ontologías para establecer el sentido del término que se usa en la búsqueda.

Lo anterior, evidencia la necesidad de la existencia de métodos de construcción de ontologías, rápidos y fáciles, para evitar que la adquisición de conocimiento se vuelva un cuello de botella. Los métodos manuales, demandan mucho tiempo y resultan costosos [15]. Como una respuesta a esta necesidad aparece la Ingeniería Ontológica y su rama, Aprendizaje Ontológico (Ontology Learning), la cual investiga el desarrollo de métodos (automáticos y semi automáticos) para la adquisición del conocimiento tendiente a la creación de ontologías. El Ontology Learning hace énfasis en la generación de herramientas que permiten importar, extraer, integrar, refinar y evaluar las ontologías bajo la supervisión de un experto humano, el "ingeniero ontológico"[13].

El punto de partida en el proceso de construcción de una ontología es muy importante. La reutilización de ontologías existentes, es un camino posible, que a simple vista parece fácil, pero algunas experiencias muestran que no lo es [2]. Sin embargo, construir una ontología desde cero, es un trabajo largo y tedioso, es por esto que los métodos de extracción y construcción de una ontología deben incorporar procesos de

aprendizaje y adaptación automático, que permitan la renovación de las ontologías frente al cambio constante de la información. Algunos trabajos de investigación proponen métodos de extracción de los términos de una ontología a partir de textos, de documentos tipo esquema, como DTDs (Data Type Definition) [8], y de páginas Web, entre otros.

B. DTD y XSD

Los DTDs (Data Type Definition) al igual que los XSD (XML Schema Definition) reciben el nombre de documentos esquema, ellos contienen meta datos de los documentos XML y son aceptados como un estándar por la W3C. Su propósito es definir la estructura y las restricciones de los documentos XML, tales como: los elementos y atributos que pueden usarse en el documento, el orden y la jerarquía en que deben aparecer, los tipos de datos y valores por defecto que se les puede asignar, etc. Inicialmente aparecieron los DTD como documentos esquema, pero con el tiempo, estos comenzaron a presentar limitaciones y fueron reemplazados por los XSD. Los XSD presentan algunas ventajas sobre los DTDs como son:

o Incorporan un gran número de tipos. Mientras que los DTDs solo permiten definir el tipo texto (#PCDATA), los XSD permiten definir tipos como fechas, string, enteros, decimales, boolean, entre otros. Además, le permite al usuario, definir tipos propios a partir de otros tipos.

o Utilizan, opcionalmente, un mecanismo de "herencia" en la definición de tipos complejos que permite reutilizar y extender elementos definidos con anterioridad.

o Soportan los namespace, los cuales son apuntadores (URI) que permiten diferenciar entre elementos y atributos duplicados. Por ejemplo, dos esquemas pueden definir elementos con igual nombre pero con significado diferente, entonces para evitar ambigüedades y colisiones entre nombres, se debe prefijar cada elemento con su namespace.

o Utilizan la sintaxis propia de los documentos XML, facilitando su procesamiento y permitiendo verificar su validez.

En este artículo se propone un método para extraer los componentes de una ontología a partir de documentos esquema de XML (XSD). En los siguientes subnumerales se explican los principales componentes del lenguaje de definición de esquemas de XML.

C. Elementos del Lenguaje Esquema XML

Como se mencionó anteriormente los XSD definen la estructura de la información que puede contener un documento XML, haciendo uso de los componentes del lenguaje esquema XML (XML Schema).

1) Elementos simples

En XML, un elemento simple es aquel que no posee otros elementos o atributos, es decir que es definido de tipo atómico. El lenguaje esquema XML provee 44 tipos de datos, también llamados tipos atómicos, debido a que no se pueden descomponer en otros componentes. Algunos ejemplos son: string, decimal, integer, boolean, date, etc.

2) Atributos

Los atributos de un elemento se definen con la etiqueta <attribute> y siempre son de tipo atómico.

3) Tipos anónimos y restricciones

Si los elementos y atributos no tienen definido en sus atributos de componente un tipo, es posible definir su tipo mediante tipos anónimos, los cuales permiten restringir (<restriction>) o extender (<extension>) un tipo simple o un tipo complejo. Como ejemplo se tiene que usando la etiqueta <simpleType> y <restricción>, es posible definir un rango de valores permitidos.

Ejemplo:

```
<element name="edad">
 <simpleType>
  <restriction base="integer">
   <minInclusive value="0"/>
   <maxInclusive value="100"/>
  </restriction>
 </simpleType>
</element>
```

Esto significa que el elemento simple "edad" es un número entero entre 0 y 100.

Ejemplo:

```
<element name="automovil" type="TipoAuto"/>
 <simpleType name="TipoAuto">
  <restriction base="string">
   <enumeration value="Renault"/>
   <enumeration value="Chevrolet"/>
   <enumeration value="Audi"/>
  </restriction>
</simpleType>
```

En este caso, “automóvil” es un elemento simple, de tipo string que solo acepta uno de tres valores: Renault, Chevrolet o Audi.

Existen muchos tipos de restricciones que no se explican en este artículo.

4) Elementos complejos.

En XML, los elementos complejos, son aquellos que contienen otros elementos y/o atributos. Ellos son de 4 tipos: elementos vacíos, elementos que contienen solamente otros elementos, elementos que contienen solamente texto, elementos que contienen tanto texto como otros elementos.

Todos ellos pueden poseer además atributos.
Un elemento complejo se puede definir de dos formas.
Ejemplo:
a)

```
<element name="estudiante">
```

```
 <complexType>
   <sequence>
    <element name="nombre" type="string"/>
<element name="apellido" type="string"/>
   </sequence>
 </complexType>
</element>
b)
<element name="estudiante" type="infoPersona"/>
 <complexType name="infoPersona">
  <sequence>
   <element name="nombre" type="string"/>
   <element name="apellido" type="string"/>
  </sequence>
</complexType>
```

Cuando un elemento se define como se definió "estudiante" en a) se dice que el elemento es anónimo, por no poseer el tipo definido con el atributo type, como si se hace en b).

La manera b) de definir el elemento "estudiante", tiene una ventaja sobre la manera a), es que permite que el tipo infoPersona pueda ser referenciado por otros elementos.

Se observa en el ejemplo anterior la utilización de la etiqueta <sequence>, la cual significa que los elementos (hijos) nombre y apellido, es decir los que están contenidos en el elemento (padre) "estudiante", deben aparecer en el documento XML, en el orden definido por dicha etiqueta.

Un elemento complejo sin contenido (vacío), es aquel que no contiene otros elementos, solamente puede contener atributos. Existen varias maneras de definirlo, la siguiente es una de ellas:
Ejemplo:

```
<element name=”carpeta” type=”carpetaTipo”/>
<complexType name=”carpetaTipo”>
<attribute name=”idCarpeta” type=”positiveInteger”/>
</complexType>
```

Existen también los elementos complejos que solo pueden contener otros elementos. Los elementos complejos mixtos son aquellos que pueden contener texto, atributos y otros elementos. Y se definen colocando el atributo *mixed* a la etiqueta *complexType* con valor igual a “*true*”.
Ejemplo:

```
<element name=”letter”>
 <complexType mixed=”true”>
  <sequence>
   <element name=”name” />
     <complexType>
       <sequence>
       <element name=”firstName” type=”xs:string”/>
       <element name=”lastName” type=”xs:string”/>
       </sequence>
     </complexType>
   </element>
   <element name=”orderid” type=”xs:positiveInteger”/>
   <element name=”shipdate” type=”xs:date”/>
  </sequence>
 </complexType>
</element>
```

5) Indicadores

Los indicadores en XML son siete y se clasifican de la siguiente manera:

Indicadores de orden:

All: indica que los elementos hijos pueden aparecer en cualquier orden y que cada elemento hijo puede aparecer solo una vez.

Choice: exige que, uno y solo uno, de los elementos hijos pueda estar presente en el documento XML.

Indicadores de grupo: son utilizados para definir conjuntos de componentes (elementos o atributos) que se relacionan. Los grupos de elemento son definidos, con la etiqueta Group. Los grupos de atributos se definen con la etiqueta attributeGroup.

Indicadores de ocurrencia: son utilizados para indicar cuántas veces pueden aparecer los elementos. Se definen con los atributos maxOccurs y minOccurs, para indicar el número máximo y mínimo de ocurrencias, respectivamente, de un elemento.

Tanto en los indicadores de orden, como en los de grupo, los indicadores de ocurrencia tienen un valor por defecto igual a 1. Cuando se desea que un elemento aparezca un número ilimitado de veces, se coloca como valor del atributo maxOccurs el valor "unbounded".

III. ANTECEDENTES

En esta sección se presentan algunos de los trabajos más representativos enfocados a la generación automática de ontologías. Algunos de ellos extraen los elementos componentes de las ontologías a partir de descripción de recursos RDF o de DTDs. La presente propuesta extrae las clases, las propiedades y las relaciones de la ontología a partir de esquemas XML aplicando un conjunto de heurísticas.

- Delteil et al. [3], presentan una propuesta para el aprendizaje de ontologías a partir de anotaciones de recursos Web en RDF. Inicialmente, extraen de manera parcial las descripciones de los recursos de todo el grafo RDF reuniendo todas las anotaciones y luego aplican un método incremental, que va enriqueciendo la ontología gradualmente.

- El enfoque presentado por Doan et al., en [4], tiene por objeto el aprendizaje semi-automático de mapeos entre los esquemas fuente y los esquemas mediadores, a través del aprendizaje de máquina. Este enfoque puede ser aplicado para obtener conocimientos de fuentes semi-estructuradas. Un sistema de Aprendizaje de Descripciones Fuente (Learning Source Descriptions - LSD) soporta el proceso en general. La

idea subyacente es que después de que una serie de fuentes de datos haya sido mapeada manualmente a un esquema de mediación, el sistema debería ser capaz de extraer información importante de estos mapeos con el fin de proponer con éxito posteriores mapeos de fuentes de datos.

• El método descrito en [16] tiene por objeto desarrollar taxonomías desde repositorios de dominio escritos en XML o RDF, utilizando un enfoque de minería de datos llamado "grupo de minería". Este enfoque, en primer lugar intenta agrupar un conjunto de archivos-metadatos similares para después extraer un vocabulario controlado que se utiliza para la construcción de la taxonomía.

• Volz et al., en [19] describen un prototipo, llamado OntoLiFT, el cual trata de capturar la semántica de esquemas de Bases de Datos relacionales, esquemas XML y especificaciones UML. En lo relacionado a los esquemas XML, ellos traducen símbolos terminales y no-terminales en conceptos y roles de la ontología. El proceso de traducción se realiza por medio de la aplicación secuencial de un conjunto de normas o reglas de traducción, las cuales utilizan una gramática de árbol regular. Esta última ayuda para que el proceso de mapeo sea independiente de la lengua utilizada, para codificar el esquema original y además permite eliminar toda la información innecesaria del esquema. Este enfoque no considera la integridad de las limitaciones del esquema XML cuando se traduce en árboles gramáticos regulares.

• En [8], Giraldo propone un método para la construcción semiautomática de ontologías del dominio. La primera fase del método presenta una técnica basada en heurísticas, para identificar los elementos componentes de una ontología, a partir de un conjunto de DTDs relativos a un dominio particular [7]. En el marco del proyecto PICSEL2, desarrollado en la Universidad Paris XI [8], la ontología construida se incorporó al sistema mediador PICSEL (Producción de Interfaces de Conocimiento para Servicios En Línea), para interrogar un conjunto de fuentes heterogéneas. La ontología se representó en el formalismo propio de PICSEL [9], llamado CARIN, el cual combina una lógica de descripción con un lenguaje basado en reglas de tipo Datalog. El método propuesto fue validado sobre un conjunto de DTDs producidos por la OTA, organismo de normalización de mensajes electrónicos del dominio del turismo. Estos mensajes reflejan el consenso entre expertos de ese dominio. En este trabajo se constató que los DTDs, por su propia esencia, presentan limitaciones para deducir algunos elementos de la ontología, particularmente las relaciones de generalización/especialización. Las heurísticas definidas en el presente artículo superan las limitaciones presentadas en [8] ya que explotan esquemas XML, en lugar de DTDs.

Cabe anotar que un modelo del dominio no es necesariamente una ontología por el hecho de estar representado en un lenguaje ontológico, de la misma manera que un programa no es un sistema a base de conocimiento por estar escrito en Prolog [10]. Por ello no se exploraron propuestas como las de Ferdinand et al., [6], quienes proponen mapear esquemas XML al lenguaje OWL.

La propuesta que se presenta en este artículo busca identificar y extraer los términos del dominio y encontrar las posibles relaciones entre ellos a partir de la aplicación de un conjunto de heurísticas sobre un conjunto de esquemas XML. La aplicación de las heurísticas sobre un solo esquema no sería concluyente. Así, esta propuesta no consiste en una traducción directa de un esquema XML en una ontología, ni se trata de una fusión de esquemas XML, ya que el propósito con el cual los esquemas XML son construidos no es la descripción de un dominio. Sin embargo, si se espera que ellos incorporen los términos pertinentes de un dominio dado, principalmente si los autores de los esquemas son expertos de ese dominio y tienen como propósito crear especificaciones estándares.

IV. SOLUCIÓN PROPUESTA

Desde la perspectiva de la lógica de primer orden, Gruber [11] identificó cinco tipos de componentes de una ontología: clases, relaciones, funciones, axiomas formales e instancias.

Desde la lógica de descripción, se dice que una ontología posee conceptos (clases), roles (relaciones) e individuos (instancias).

En la solución propuesta, la ontología que se desea construir tiene los componentes que se muestran en la Figura 1:

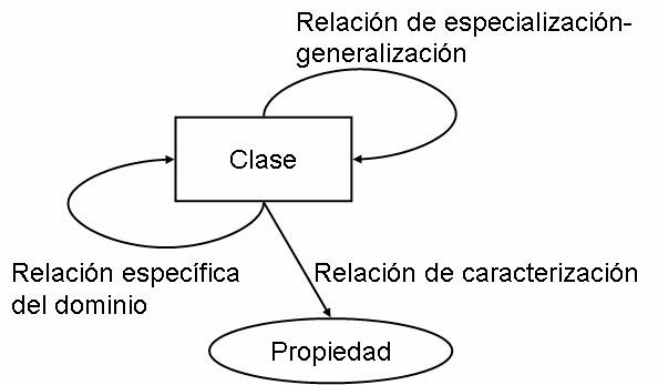

Figura 1. Modelo de la ontología propuesta

Allí se distinguen cinco componentes:

Clases: una clase es una abstracción de un conjunto de objetos con características similares.

Propiedades: una propiedad es un término que caracteriza los objetos de una clase.

Relaciones de especialización-generalización: es una relación entre clases. La clase C2 está ligada a la clase C1 por una relación de especialización, si la clase C2 es más específica que la clase C1. En ese caso, se dice que la clase C2 "es un(a)" C1, en inglés

C2 "is-a" C1. La relación inversa de una relación de especialización es la relación de generalización. Es decir, que el nombre de esta relación depende del sentido de la lectura, si es de C1 a C2 o de C2 a C1.

Relaciones de caracterización: es una relación entre una clase y una propiedad: una clase C está ligada a una propiedad P por una relación de caracterización, si P es una propiedad de C.

Relaciones específicas del dominio: es una relación entre clases, que no corresponde ni a una relación de especialización-generalización ni a una relación de caracterización. Los nombres de estas relaciones son en función del dominio. Las relaciones de composición (parte-de) hacen parte de este tipo de relaciones.

Así, la ontología propuesta comprende un conjunto de jerarquías de clases que describen la categorización de las clases de objetos de un dominio de aplicación, cada clase siendo caracterizada por unas propiedades y ligada a otras clases por relaciones especificas del dominio (Figura 2). Para una clase dada, el modelo precisa, la clase que la generaliza (clase madre en la jerarquía) y eventualmente las propiedades que hacen que un objeto pertenezca a esa clase.

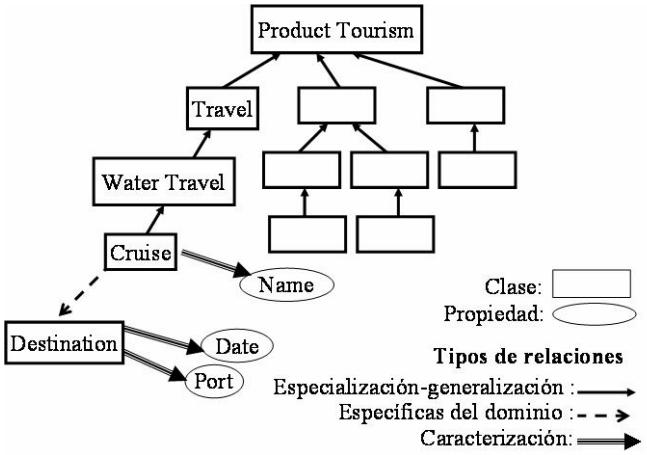

Figura 2. Extracto de una jerarquía de clases

Los elementos extraídos son:

• un conjunto de clases C={Cl1, Cl2, …Cln} donde cada clase Cli corresponde a una clase del dominio.

• un conjunto de propiedades P = {p1, p2,.., pk} donde cada propiedad pi es un término que caracteriza una clase.

• un conjunto de relaciones, las cuales son de dos tipos:

a) Relaciones entre clases: las relaciones de especialización (RS) y las relaciones especificas del dominio (RSD).

Las relaciones de especialización de la clase Cli son representadas por el conjunto de clases Clj que especializan Cli.

RS (Cli) = { Clj j= 0,n / Clj " is-a " Cli }

- Las relaciones específicas del dominio de la clase Cli son representadas por un conjunto de parejas (Clj, ind) donde Clj es una clase ligada a Cli por una relación especifica del dominio e ind corresponde al indicador de cardinalidad asociado a esta relación.

RSD (Cli) = { (Clj, ind) / j= 0,n }

Donde ind {?, +, *," "} (la cardinalidad de 1 es representada por " ").

b) Relaciones entre clases y propiedades: llamadas relaciones de caracterización (RC).

Las relaciones de caracterización de la clase Cli son representadas por el conjunto de parejas (pj, ind) donde pj es una propiedad que caracteriza la clase Cli y donde ind corresponde al indicador de cardinalidad asociado a esta relación.

RC (Cli) = { (pj, ind) / j= 0,n }

Los conjuntos RS (Cli), RSD (Cli) y RC (Cli) pueden estar eventualmente vacíos. En efecto, una clase no necesariamente tiene que estar ligada a otra clase por una relación de especialización o por una relación especifica del dominio o tener propiedades.

Se observa que la cardinalidad no está presente en las relaciones de especialización RS. Esto es debido a que una relación RS siempre tiene una cardinalidad de 1.

Se representa la clase Cli por la 4-tupla (Cli, RS, RSD, RC) donde Cli es el nombre de la clase y RS, RSD y RC representan respectivamente el conjunto de relaciones de especialización, las relaciones específicas del dominio y las relaciones de caracterización.

El método propuesto para la construcción semiautomática de ontologías, utiliza reglas heurísticas, que aplicadas a un conjunto de documentos esquemas (XSD) relativos a un dominio de aplicación, identifican los componentes de la ontología.

Los componentes de la ontología se extraen a través del análisis sintáctico de un conjunto de documentos XSD.

El principio de base para determinar las clases es el siguiente: una clase es considerada como una representación abstracta de un conjunto de objetos complejos, detectados en los documentos esquema, por el hecho de que se trata de elementos que se descomponen en uno o varios elementos hijos. Dado que el enfoque propuesto descansa sobre la explotación de uno o varios documentos XSD representativos del dominio, sus componentes deben aparecer en al menos uno de los documentos de la muestra provista como entrada al sistema. De la misma manera, se dice que, los términos asociados a los elementos que no se descomponen en ninguno de los documentos XSD extraídos, corresponden a propiedades de la ontología. El principio utilizado encuentra su justificación, en el

hecho de que, los documentos que se van a explotar se suponen representativos del dominio que se va a cubrir. La aplicación de este principio se efectúa mediante la aplicación de heurísticas.

Los siguientes son los componentes de los documentos esquema considerados por las heurísticas propuestas:

• **<element>:** los elementos podrían ser clases o propiedades de la ontología dependiendo del tipo de dato que referencien. Es decir que si en su atributo type, tiene como valor un tipo simple, el elemento será una propiedad, de lo contrario será una clase.

• **<complexType>:** los tipos complejos corresponden a clases de la ontología, debido a que en su contenido se encontrarán estructuras compuestas por otros elementos (<element>).

• **<simpleType>:** los tipos simples son propiedades de la ontología debido a que siempre referencian tipos atómicos. En este caso, también se explotan las restricciones (<restriction>) asociadas a los tipos simples.

• **<choice>:** el componente choice define relaciones de generalización y especialización, entre el componente padre y el (los) componente(s) hijo(s).

• **<sequence> y <all>:** el componente sequence identifica relaciones específicas del dominio, entre su componente padre y su(s) componente(s) hijo(s).

La tabla 1 muestra las heurísticas que se proponen para identificar los elementos componentes de la ontología. Estas heurísticas se explican en la siguiente sección.

Tabla 1. Resumen de heurísticas propuestas para la identificación de elementos de la ontología

HEURISTICAS PARA EXTRAER CLASES
H1C: una clase de la ontología corresponde a un componente *element* definido de tipo anónimo, usando el componente *complexType*. Los tipos anónimos se distinguen por la ausencia del atributo "*type*" en la definición del elemento y del atributo "*name*" en la definición del "*complexType*".
H2C: una clase de la ontología corresponde a los tipos complejos (<*complexType*>) no anónimos, es decir aquellos que poseen el atributo "*name*".
H3C: una clase de la ontología corresponde a los elementos que tengan en su atributo "*type*" un tipo complejo.
HEURISTICAS PARA EXTRAER PROPIEDADES
H1P: una propiedad de una clase corresponde a elementos (<*element*>) definidos de tipo atómico.
H2P: una propiedad de una clase corresponde a los atributos definidos con la componente <*attribute*>, los cuales siempre serán de tipo atómico.
H3P: una propiedad de una clase corresponde a los componentes <*element*> que sean anónimos y que tengan como componente hijo un <*simpleType*>
HEURÍSTICA PARA DETERMINAR LAS RELACIONES DE CARACTERIZACIÓN
HRc: cuando al aplicar las heurísticas H1P, H2P y H3P se determine que un componente es una propiedad de una clase, entonces se extrae una relación de caracterización entre esa clase y sus propiedades.
HEURÍSTICA PARA DETERMINAR LAS RELACIONES ESPECÍFICAS DEL DOMINIO
HRed: las instrucciones que definen un elemento complejo como la conjunción de otros elementos (simples y/o complejos), se traducen en una relación especifica del dominio, entre la clase correspondiente al elemento complejo que se está definiendo (padre) y la(s) clase(s) relativa(s) a los elementos complejos hijos de éste. En los XSD esto se define con las etiquetas <*sequence*> y <*all*>.
HEURÍSTICAS PARA DETERMINAR LAS RELACIONES DE ESPECIALIZACIÓN / GENERALIZACIÓN
H1Reg: las instrucciones que definen un elemento complejo como la disyunción de otros elementos (simples y/o complejos), se traducen en una relación de especialización/generalización, entre la clase correspondiente al elemento complejo que se está definiendo (padre) y la(s) clase(s) relativa(s) a los elementos complejos hijos de éste. En los XSD esto se define con la etiqueta <*choice*>.
H2Reg: Los elementos no anónimos que tienen tipo complejo dan lugar a una relación de especialización / generalización. Dependiendo del sentido de lectura, se determina: a) una relación de especialización entre la clase correspondiente al elemento complejo y la clase correspondiente al tipo complejo. b) una relación de generalización entre la clase correspondiente al tipo complejo y la clase correspondiente al elemento complejo.

A. Heurísticas para determinar las clases

H1C: una clase de la ontología corresponde a un componente *element* definido de tipo anónimo, usando el componente *complexType*. Los tipos anónimos se distinguen por la ausencia del atributo "*type*" en la definición del elemento y del atributo "*name*" en la definición del "*complexType*".
Ejemplo:

```
<element name = 'cliente'>
 <complexType>
...
 </complexType>
</element>
```

En este caso, la clase resultante al aplicar la heurística H1C es "cliente".

Cabe anotar que, los elementos que componen a "cliente", es decir lo que iría entre las etiquetas *complexType*, pueden ser elementos de tipo atómico, como por ejemplo "nombre" y "apellido" o elementos de tipo complejo como "tipoDireccion".

H2C: una clase de la ontología corresponde a los tipos complejos (<complexType>) no anónimos, es decir aquellos que poseen el atributo "name".
Ejemplo:

```
<complexType name='tipoDireccion'>
...
</complexType>
```

Así, el resultado de aplicar la heurística H2C es la clase "tipoDirección".

Tabla 1. Resumen de heurísticas propuestas para identificación de elementos de la ontología

H3C: una clase de la ontología corresponde a los elementos que tengan en su atributo "type" un tipo complejo.
Ejemplo:

```
<element name='direccionCasa' type= 'direccion'/>
<complexType name= 'direccion'>
...
</complexType>
```

En este ejemplo la clase "dirección" se obtiene al aplicar la heurística H2C y aplicando la heurística H3C se obtiene la clase "direccionCasa".

B. Heurísticas para determinar Propiedades

H1P: una propiedad de una clase corresponde a elementos (<element>) definidos de tipo atómico.
Ejemplo:

```
<element name='nombre' type='string'/>
```

Ejemplo:

```
<element name='nombre'  type='tipoNombre'/>
<simpleType name='tipoNombre'>
<restriction base="string"/>
</simpleType>
```

En cualquiera de estos dos últimos ejemplos se obtiene la propiedad nombre, al aplicar la heurística H1P.

H2P: una propiedad de una clase corresponde a los atributos definidos con la componente <attribute>, los cuales siempre serán de tipo atómico.
Ejemplo:

```
<attribute name="orderDate" type="date"/>
```

Según la heurística H2P, "orderDate" corresponde a una propiedad. Cabe anotar que la implementación de las heurísticas, guarda también, el tipo de las propiedades (en este ejemplo "date"), debido a que se espera utilizar la ontología como interfaz de interrogación sobre un conjunto de fuentes de información heterogéneas y los tipos servirán para controlar los tipos de los valores que se digiten. De la misma manera, en el caso del siguiente ejemplo, también se guardan las restricciones.

H3P: una propiedad de una clase corresponde a los componentes <element> que sean anónimos y que tengan como componente hijo un <simpleType>
Ejemplo:

```
<element name="quantity">
 <simpleType>
<restriction base="positiveInteger" <maxExclusive value="100"/>
</restriction>
 </simpleType>
</element>
```

Al aplicar la heurística H3P, se obtiene la propiedad "quantity". Se guardan además, el tipo (positiveInteger) y la restricción (maxExclusive=100).

C. Heurística para determinar las Relaciones de Caracterización

HRc: cuando al aplicar las heurísticas H1P, H2P y H3P se determine que un componente es una propiedad de una clase, entonces se extrae una relación de caracterización entre esa clase y sus propiedades.
Ejemplo:

```
<element name='cliente'>
<complexType>
<element name='nombre' type='string'/>
<element name='apellido' type='string'/>
</complexType>
</element>
```

En este ejemplo, aplicando H1P se extraen "nombre" y "apellido" como propiedades de la clase "cliente" y por tanto se deducen dos relaciones de caracterización: una entre "cliente" y "nombre" y otra entre "cliente" y "apellido".

D. Heurística para determinar las Relaciones Específicas del Dominio

HRed: las instrucciones que definen un elemento complejo como la conjunción de otros elementos (simples y/o complejos), se traducen en una relación especifica del dominio, entre la clase correspondiente al elemento complejo que se está definiendo (padre) y la(s) clase(s) relativa(s) a los elementos complejos hijos de éste. En los XSD esto se define con las

etiquetas <sequence> y <all>.
Ejemplo:

```
<element name='cliente'>
<complexType>
<sequence>
        <element name='nombre'  type='string'/>
        <element name='direccion'>
                <complexType>
<element name='calle' type='string'/>
<element name='carrera' type='string'/>
<element name='barrio' type='string'/>
</complexType>
</element>
...
</sequence>
</complexType>
</element>
```

Al aplicar la heurística HRed, en el ejemplo anterior, se obtiene como resultado una relación específica del dominio entre la clase "cliente" (que corresponde a un elemento complejo-padre) y la clase "dirección" (que corresponde a un elemento complejo-hijo). De la misma manera, si en lugar de la etiqueta <sequence> viene la etiqueta <all> también se extraerá una relación específica del dominio.

E. Heurísticas para determinar las Relaciones de Especialización / Generalización

H1Reg: las instrucciones que definen un elemento complejo como la disyunción de otros elementos (simples y/o complejos), se traducen en una relación de especialización/generalización, entre la clase correspondiente al elemento complejo que se está definiendo (padre) y la(s) clase(s) relativa(s) a los elementos complejos hijos de éste. En los XSD esto se define con la etiqueta <choice>.
Ejemplo:

```
<element name = direccionCliente'>
<complexType>
<choice>
<element name='direccionCasa' type='tipoDireccion'/>
<element name='direccionTrabajo' type='tipoDireccion'/>
</choice>
</complexType>
</element>
```

El resultado de aplicar la heurística H1Reg, corresponde a dos relaciones de especialización/generalización, una entre la clase "direccionCliente" y la clase "direccionCasa" y otra entre la clase "direccionCliente" y la clase "direccionTrabajo". En este caso, la clase más general es la clase "direccionCliente" y las clases más específicas son "direccionCasa" y "direccionTrabajo".

H2Reg: Los elementos no anónimos que tienen tipo complejo dan lugar a una relación de especialización / generalización.

Dependiendo del sentido de lectura, se determina:

o una relación de especialización entre la clase correspondiente al elemento complejo y la clase correspondiente al tipo complejo.

o una relación de generalización entre la clase correspondiente al tipo complejo y la clase correspondiente al elemento complejo.
Ejemplo:

```
<complexType name="Persona">
<sequence>
<element name="nombre" type="string"/>
<element name="cedula" type="string"/>
</sequence>
</complexType>
<element name="Estudiante" type="Persona"/>
<element name="Profesor" type="Persona"/>
```

La idea base para determinar esta heurística es que si dos o más elementos tienen nombres diferentes pero tienen el mismo tipo complejo, cada uno de estos elementos tiene una relación de especialización con el tipo complejo. En el ejemplo, aplicando la heurística H2Reg se extraen dos relaciones de especialización/generalización, una Estudiante-Persona y otra Profesor-Persona, donde las clases Estudiante y Profesor son especializaciones de Persona.

Esta heurística permite mejorar la solución propuesta en [8], donde se aplican una serie de heurísticas para determinar los componentes de una ontología a partir de un conjunto de DTDs. Dado que los esquemas XML permiten definir tipos complejos, lo cual no es posible en los DTDs, el trabajar con los esquemas amplía las posibilidades de obtener más relaciones de especialización/generalización. Los DTDs solo permiten definir el tipo texto (#PCDATA), en tanto que los esquemas permiten definir diversos tipos, inclusive tipos propios a partir de otros tipos.

V. CONCLUSIONES Y TRABAJO FUTURO

Este artículo, propone un método heurístico para identificar los elementos componentes de una ontología del dominio a partir de un conjunto de documentos, tipo esquema XML, representativos de un mismo dominio. El método propuesto es un paso preliminar en el proceso de construcción de ontologías del dominio. Los elementos identificados con las heurísticas propuestas, son posteriormente validados por los expertos del dominio. Los pasos siguientes consisten en construir la jerarquía de clases y luego representar la ontología en un lenguaje apropiado para asegurar su formalización y el aprovechamiento de las estructuras de datos y de las lógicas de descripción incorporadas en los lenguajes.

En [8], Giraldo presenta una ontología en el dominio del turismo, construida a partir de DTDs definidos por expertos en este dominio, reunidos en la Open Travel Alliance (OTA). En ese trabajo se plantea la necesidad de explotar esquemas XML en lugar de DTDs, para tratar de superar ciertas limitaciones inherentes a la naturaleza de estos últimos.

Un trabajo en curso consiste en la implementación y aplicación de estas heurísticas sobre un conjunto significativo de documentos esquema XML relativos al dominio antes mencionado y definidos igualmente por la OTA. En este trabajo se constata la aparición de un mayor número de elementos componentes de la ontología, dado que los esquemas XML poseen más información que los DTD.

REFERENCIAS

[1] Borst W. N., Construction of Engineering Ontologies, PhD Thesis, University of Twente, Enschede, 1997.

[2] Charlet J.; Bachimont B.; Bouaud J. y Zweigenbaum P. Ontologie et réutilisabilité : expérience et discussion, In Actes JAVA'94, Strasbourg, C1-C14, 1994.

[3] Delteil A.; Faron C. y Dieng R. (2001) Learning ontologies from RDF annotations. In Proceedings of the IJCAI Workshop in Ontology Learning, Seattle, 2001.

[4] Doan A.; Domingos P. y Levy A. (2000). Learning Source Descriptions for Data Integration. Proceedings of the Third International Workshop on the Web and Databases (pp. 81-86), 2000. Dallas, TX: ACM SIGMOD

[5] Enery TMC y Wilson. A. (2001). Corpus Linguistics: An Introduction Edinburgh: Edinburgh University Press.

[6] Ferdinand M.; Zirpins Ch. y Trastour D. (2004). Lifting XML Schema to OWL. In Proceedings 4th International Conference, ICWE, Munich, Germany, July 26-30.

[7] Giraldo G. y Reynaud C. Construction semi-automatique d'ontologies à partir de DTDs relatives à un même domaine, 13èmes journées francophones d'Ingénierie des Connaissances, Rouen, 28-30 Mai 2002.

[8] Giraldo G., Construction automatisée de l'ontologie de systèmes médiateurs: application à des systèmes intégrant des services standards accessibles via le Web, Tésis doctoral, Université Paris Sud XI, Orsay, Francia, 2005.

[9] Goasdoue F.; Lattes V.y Rousset M.-C. The use of CARIN language and algorithms for Integration Information: the PICSEL system, International Journal of Cooperative Information Systems, 9(4):383-40.1,2000

[10] Gómez-Perez A.; Fernández M. y Corcho O. 2004. Ontological Engineering. Berlin, London: Springer Verlag.

[11] Gruber T. (1993). A translation approach to portable ontologies. Knowledge Acquisition. Volume 5, Issue 2. London:Academic Press Ltd.

[12] http://www.opentravel.org/

[13] Maedche A.; y Staab S. (2002). Ontology Learning for the Semantic Web. IEEE Intelligent Systems 16(2), 72-79.

[14] Neches R.; Fikes R.; Finin T.; Gruber T.; Senator T. y Swartout W. (1991). Enabling technology for knowledge sharing. AI Magazine, 12(3):36-56.

[15] Novacek V. (2005). Ontology Learning. Thesis, Facultas Artis Informatice, Universitas Masarykiana

[16] Papatheodorou C.; Vassiliou A. y Simon B. 2002: Discovery of Ontologies for Learning Resources using Word-based Clustering. ED-MEDIA 2002. Copyright by AACE. Reprinted from the ED-MEDIA 2002 Proceedings, August 2002 with permission of AACE, Denver, USA, August.

[17] Studer R.; Benjamins R.y Fensel D. Knowledge Engineering : Principles and Methods, IEEE Trans. Pm Data and Knowledge Eng., vol 25, nos. 1-2, 1998, pp. 161-197.

[18] Uschold M. y Jasper R. 1999. A Framework for Understanding and Classifying Ontology Applications. Proceedings of the IJCAI99 Workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm.

[19] Volz R.; Oberle D.; Staab S. y Studer, R. (2003) OntoLiFT Prototype. IST Project 2001-33052 WonderWeb Deliverable 11.