Revista Digital Universitaria
10 de febrero 2006 • Volumen 7 Número 2 • ISSN: 1067-6079

ARTICULO

EXPRESIVIDAD LIMITADA EN EL USO DE EDITORES DE ONTOLOGÍAS

Arturo Cervera García
Investigador Instituto de Robótica, Universidad de Valencia, EspañaArturo.Cervera@robotica.uv.es

José Javier Samper Zapater
Doctor en Ingeniería Informática por la Universidad de Valencia, España.
Jose.j.samper@uv.es

Eduardo Carrillo Zambrano
Profesor Titular de la Universidad Autónoma de Bucaramanga en la Facultad de Ingeniería de Sistemas, Escuela de Ciencias Naturales e Ingeniería.
eduleidy@eudoramail.com

http://www.revista.unam.mx/vol.7/num2/art07/int07.htm

EXPRESIVIDAD LIMITADA EN EL USO DE EDITORES DE ONTOLOGÍAS

Resumen en Español

En su mayoría las herramientas de edición de ontologías no son totalmente satisfactorias en el sentido de que no son capaces de explotar y manejar todas las características de los lenguajes, por ejemplo, el uso de tipos definidos por el usuario. Gracias a la alta expresividad de los lenguajes semánticos, se pueden establecer vías alternativas a ciertas limitaciones impuestas por las herramientas de uso, permitiendo establecer guías de desarrollo para este tipo de expresividad semántica.

Palabras clave: Ontología, Semántica, OWL, tipos, XSD.

LIMITED EXPRESSIVENESS USING ONTOLOGY´S EDITORS

Abstract

The tools of edition of ontologies are not totally satisfactory to the effect that they are not capable of exploiting and handling all the characteristics of the languages, for example, the use of types defined by the user. Thanks to the high expressiveness of the semantic languages, alternative routes can be established to certain limitations imposed by the tools of use, allowing to establish guides of development for this type of semantic expressiveness.

Keywords: Ontology, Semantic, OWL, Types, XSD.

10 de febrero 2006 • Volumen 7 Número 2 • ISSN: 1067-6079

INTRODUCCIÓN

Varios aspectos describen la problemática actual relativos al acceso y distribución de la información de tráfico rodado:

·Un gran volumen de información de tráfico se distribuye entre varios sitios web. El principal problema para un usuario que necesita información de este tipo es encontrar estos sitios web y además tratar con los diferentes accesos a ésta, así como sus diferentes formas de presentación.

·Por otra parte, un usuario puede necesitar información de diferente naturaleza o tipo, y por lo tanto el almacenaje de toda esta información en un solo sitio web no es viable a nivel de costes de almacenamiento ni incluso a nivel de mantenimiento.

·El tratamiento de la información no permite realizar inferencias sobre ella de manera que pueda ser obtenido como resultado, información que a priori no estuviera explícitamente detallada.

Tras el descubrimiento de la problemática existente y la identificación de elementos que pudieran solventar esta situación, el trabajo expuesto en este artículo se centra en el estudio de los sistemas ITS (Sistemas Inteligentes de Tráfico) orientados a facilitar información de tráfico al usuario, y que a su vez permitan la gestión de la información, su tratamiento e intercambio, de manera eficaz entre los diferentes elementos que componen la arquitectura de servicios de información de tráfico, como los usuarios, aplicaciones, y proveedores de la información.

Para la consecución de los objetivos marcados han sido desarrollados diversos elementos como parte integrante de una arquitectura, que supusieron determinados aportes y/o resultados. En este artículo abordaremos la construcción de una infraestructura ontológica cuyo dominio queda enmarcado en la información sobre tráfico vial o rodado. Las ontologías construidas forman el núcleo principal del trabajo, ya que por una parte, constituyen la ontología de dominio utilizada como soporte en el emparejamiento de servicios (búsqueda del servicio que pueda aportar la información más idónea a los requerimientos de un usuario), y a su vez, conforman un mecanismo de descripción de los diferentes servicios de información de tráfico, los cuales son especificados mediante las referencias a conceptos descritos en estas ontologías. La infraestructura ontológica creada puede considerarse una base sólida en el desarrollo de un futuro y completo vocabulario semántico a utilizar por las Administraciones de Tráfico.

En la fase previa a la construcción de las ontologías, fue determinante el proceso de elección de las herramientas a utilizar para la edición de ontologías, para lo cual se procedió a una revisión bibliográfica que mostrara las diferentes comparativas entre éstas, como las aparecidas en [Den02], [EON02], [Ont02] y [Gom04]. Se tuvo en cuenta como punto de partida, la necesidad de hacer uso de los lenguajes DAML+OIL (DARPA Agent Markup Language + Ontology Inference Layer) o la evolución de éste: OWL (Web Ontology Language) [OWL04] [OWL04b]. La elección de este tipo de lenguajes fue debida a su característica de alta expresividad [DAM02], [Cor00].

Tras estas revisiones y diferentes pruebas realizadas, se pudo apreciar que en su mayoría las herramientas de edición no son totalmente satisfactorias en el sentido de que no son capaces de explotar y manejar todas las características de los lenguajes, por ejemplo, el uso de tipos definidos por el usuario. Muchas veces, surge la necesidad de hacer uso de estos tipos, para especificar ciertos conceptos. Por ejemplo, pensemos en la especificación de acotaciones a un tipo entero, decimal etc. Esto en principio no supone ningún problema para ontologías expresadas mediante los lenguajes de especificación de ontologías comentados, tal y como queda reflejado en [Dwa01], aunque por otra parte, los editores de ontologías, en general, no permiten este tipo de definiciones, por lo que la especificación de éstas, es en principio realmente difícil mediante el uso de estas herramientas.

Desarrollo

Las restricciones OWL se optimizan para expresiones de cardinalidad, es decir se puede decir fácilmente que una característica debe tener por lo menos **1** y como mucho 4 valores. Sin embargo, no es actualmente posible especificar rangos numéricos tales como "ruedas con un diámetro sobre 10". Un grupo de trabajo de OWL está colaborando actualmente con el grupo de XML Schema (XSD) [XSD01] para tener en cuenta la definición de los datatypes (tipos de datos) definidos por el usuario, por ejemplo (número entero mayor de 10) para que estos sean utilizados en restricciones del allValuesFrom en OWL. Tan pronto como se estandarice esta característica, herramientas como Protégé [PRO04] podrán hacer uso de ella. OWL provee de mecanismos para la definición 'cualitativa' lógica de clases, pero definitivamente no para la definición ' cuantitativa ', como "ruedas con el diámetro sobre 10". Esto no es un *bug*, sino una característica. La cardinalidad es una característica lógica, pero el valor numérico de un DatatypeProperty no lo es.

Podemos definir propiedades "minDiameter" y "maxDiameter" y definir subclases del concepto "rueda" con restricciones del hasValue sobre ellas. Pero esto será solamente declarativo - los razonadores lógicos no detectarán ninguna inconsistencia si por ejemplo, establecemos "minDiameter" a 10 para la clase de BigWheel, y una instancia de BigWheel se ha declarado con un "diameterValue" de 9. Debido a que OWL internamente no hace ninguna disposición de comprobar su consistencia, sin embargo es posible hacer uso de tal expresividad para transmitirla como caja negra a aplicaciones externas capaces de realizar un tratamiento sobre cantidades. [Vat04]

Dejando al margen el aspecto de la consistencia, trataremos de abordar la especificación de esta semántica mediante un caso particular:

Tomemos por ejemplo la especificación de un *Ciclomotor*: "Vehículo provisto de un motor de cilindrada no superior a 50 centímetros cúbicos, si es de combustión interna, y cuya velocidad máxima por construcción no excede de 45 km/h".

¿Cómo especificar la restricción de velocidad o la de cilindrada?

class-def defined MotorCycleA
subclass-of MotorCycle
slot-constraint Cylinder_capacity value-type integer
has-value (max 50) ---aquí se restringe valores menor o igual a 50
slot-constraint speed value-type integer
has-value (max 45) ---aquí se restringe valores menor o igual a 45

Si hacemos uso de los operadores de cardinalidad no conseguiremos expresar lo que realmente queremos, el siguiente ejemplo clarificará este aspecto:

TYPE	PROPERTY	FILLER
< max 45	speed	decimal

Las restricciones de cardinalidad ponen una restricción en el número de elementos (fillers) que puede tener una propiedad particular. Lo que realmente estamos diciendo con la expresión anterior es que hay menos de 45 valores de la propiedad *speed* que sean decimales.

10 de febrero 2006 • Volumen 7 Número 2 • ISSN: 1067-6079

Lo ideal como ya hemos comentado, sería el uso de tipos definidos por el usuario (traffic_types.xsd). De esta forma la especificación correspondiente a la velocidad sería la siguiente:

```
<daml:Class rdf:about="file:/G:/tmp/typeconstraint.daml#CycleMotorA">
    <rdfs:subClassOf>
      <daml:Class rdf:about="file:/G:/tmp/typeconstraint.daml#CycleMotor"/>
      <daml:Restriction>
        <daml:onProperty rdf:resource="file:/G:/tmp/typeconstraint.daml#speed"/>
        <daml:toClass rdf:resource="http://robotica.uv.es/traffic/traffic_types#under45"/>
      </daml:Restriction>
          ...
    </rdfs:subClassOf>
</daml:Class>
```

Restricción que hace uso de un tipo definido por el usuario.

```
<!-- Definicion de tipos para limitaciones de vehículos (traffic_types.xsd)-->
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#">
                    ...
<xsd:simpleType name="under45">
<!-- under45 es un datatype basado en el tipo definido positiveInteger -->
<!-- con la restriccion añadida de que el valor debe de ser <= 45 -->
<!-- Se utiliza para delimitar la velocidad de determinados vehículos -->
<!-- (no superior a 45 Km/H) -->
<xsd:restriction base="xsd:positiveInteger">
<xsd:maxInclusive value="44"/>
</xsd:simpleType>
                    ...
```

El objetivo marcado es poder introducir clases especializadas con restricciones en sus propiedades, o bien para establecer diferentes tipos de un determinado concepto (por ejemplo ciclomotores de tipo A de los de tipo B), en los que simplemente la diferencia radica en el uso de la restricción, o bien simplemente para definir un determinado concepto.

Afortunadamente DAML + OIL (OWL) es lo suficientemente flexible para permitir la definición de clases con este tipo de restricciones. En otras palabras, definir una clase con restricciones numéricas mediante editores, (por ejemplo OilEd [OEd04], [Rob02]) puede resultar tedioso pero no imposible. Lo que en realidad hace falta es una interfaz que produzca la larga cadena de restricciones de manera más o menos automática.

Tomando como base de estudio el artículo de Gruber y Olson sobre cantidades físicas [Gru94] a continuación serán expuestos los pasos para la consecución de nuestro propósito.

Solución al problema

Tomemos como ejemplo la especificación de un concepto denominado MotorCycleA cuyas restricción sea "Para todo elemento perteneciente a la clase de MotorCycle, que posea velocidad, se deberá cumplir que la magnitud de ésta sea menor que 45 y su unidad de medida deberá ser km/h".

En la restricción anterior se pueden determinar dos cosas, la cantidad física (45) y por otra parte la unidad de medida (km/h). Para entender mejor las claves de la conceptualización de nuestras ontologías se hace necesario revisar algunas definiciones importantes:

·Cantidad Física (Quantity): Es una medida de algún aspecto cuantificable del mundo modelado. Hacemos uso de ella para generalizar la noción de medida cuantitativa. Aunque las cantidades físicas pueden ser de varios tipos, simplemente hacemos uso del tipo escalar. En nuestra conceptualización haremos uso de la generalización para representar cantidades de cualquier cosa cuantificable así como los tipos de ésta, por ejemplo "Cantidades de velocidad" (SpeedQuantity) que permiten especificar el tipo de cantidad cuyos elementos son constantes escalares de la dimensión velocidad.

·Dimensión Física (Physical Dimensions): La principal diferencia entre una cantidad física y una entidad numérica pura es la caracterización de la cantidad por una dimensión física. La dimensión física de una cantidad, la hace diferente de otros tipos de cantidades. (por ejemplo, la dimensión física de la posición de un cuerpo es la longitud).

·Las cantidades escalares (scalar quantities), son cantidades constantes con valores de magnitudes.

·Unidades de Medida (Units of Measure): Aunque la identidad de las cantidades no depende del proceso o la naturaleza de la medida, o unidad de medida, sin embargo ésta es relevante a la hora de especificar constantes en ecuaciones. Una unidad de medida es una cantidad absoluta de alguna cosa que puede ser usada como una cantidad de referencia estándar.(por ejemplo, la unidad de medida de la masa de un cuerpo es el kilogramo).

·Magnitud (Magnitude): La magnitud de una cantidad física no es una propiedad de la cantidad sino que es dada por una función binaria que mapea una cantidad y una unidad de medida a un valor numérico (cantidad sin dimensión). Por ejemplo, la magnitud de 50kg es 50.

Las anteriores definiciones nos permiten especificar tanto los diferentes conceptos como las relaciones (properties) necesarias para el desarrollo de nuestra ontología ejemplo. El primer paso será pues la construcción de la taxonomía de conceptos y su orden jerárquico, tal y como podemos apreciar en la figura 1.

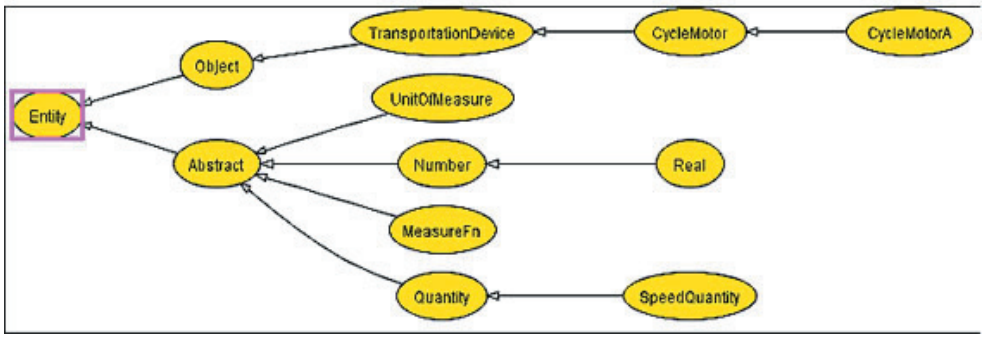

Figura 1: Taxonomía de conceptos

10 de febrero 2006 • Volumen 7 Número 2 • ISSN: 1067-6079

Como se puede observar en la figura anterior existen dos ramas principales en el grafo claramente diferenciadas, que establecen los dominios de *"Object"*, elementos físicos que caracterizan nuestra pequeña ontología, y el dominio *"Abstract"* utilizado para determinar las medidas y cantidades que determinaran una clase en concreto mediante el uso de restricciones.

Hay que hacer constar que aunque el uso de DataTypeProperties y de XML Schema podría simplificar de manera notable el desarrollo de la Ontología se ha preferido definir clases para los conceptos de Number y Real para una mayor claridad en nuestra explicación.

Algunos conceptos como **UnitOfMeasure, Quantity, o SpeedQuantity** se corresponden con las definiciones dadas anteriormente y otros conceptos son usados para aportar una mayor expresividad como:

·**MeasureFn**: Representa las diferentes medidas que se pueden obtener. Es utilizado para restringir el dominio o rango de ciertas propiedades que a continuación se exponen.

Algunas de las relaciones necesarias también tienen su correspondencia con las definiciones aportadas (magnitude y unitOfMeasure) y otras son utilizadas para aportar el grado de expresividad requerido:

·**magnitude** (dominio:MeasureFn, rango: Number)1
·**unitOfMeasure** (dominio:MeasureFn, rango: UnitOfMeasure)
·**speed** (dominio: Object, rango: SpeedQuantity): Esta propiedad es usada para poder establecer distintos valores de velocidad mediante el uso de individuales.
·**hasMeasureFn** (dominio: Quantity, rango: MeasureFn): Propiedad usada para establecer correspondencias entre "cantidades físicas" y "medidas".
·**lessThan** (dominio: Number, rango: Number): Se corresponde con el operador relacional "<" o "menor que".

Llegados a este punto y mediante el uso de propiedades y de individuales, podremos especificar la clase MotorCycleA, objeto de nuestro estudio mediante el uso de la siguiente restricción de su velocidad:

```
(and (to-class speed (hasMeasureFn MeasureFn such that the magnitude of
MeasureFn is lessThan 45)) (to-class speed (hasMeasureFn MeasureFn such
that the unitOfMeasure of MeasureFn is kperHour)).
```

y por tanto la definición completa sería:

```
class MotorCycleA
type: primitive
superclasses: MotorCycle
constraints:
        restriction speed to-class (restriction hasMeasureFn to-class (MeasureFn and
(restriction magnitude to-class (restriction lessThan to-class one-of (45 ))))

        restriction speed to-class (restriction hasMeasureFn to-class (MeasureFn and
(restriction magnitude to-class (restriction unitOfMeasure to-class one-of (kperHour)))))

used in individuals: InstanceOfMotorCycleA
```

[1] magnitude (dominio: MeasureFn, rango: decimal) y relaciones en MeasureFn1: unitOfMeasure mpersec y magnitude decimal con valor 40 si se hubieran utilizado los tipos definidos en XML Schema.

Tomando la primera de las restricciones, la siguiente figura muestra cómo es posible la especificación de este tipo de restricciones mediante el uso de editores de expresiones como se aprecia en la figura 2.

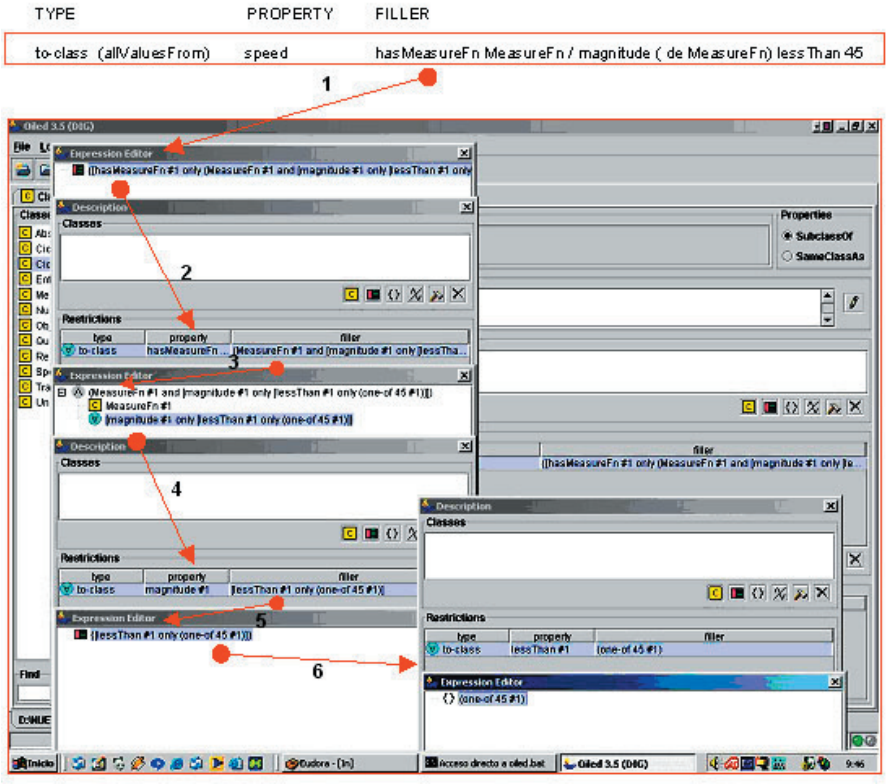

Figura 2: Especificación de restricciones mediante editores.

Creación de instancias

Llegados a este punto, ¿cómo podemos especificar una instancia de ciclomotor de tipo A mediante este conocimiento?

En primer lugar, podemos determinar una instancia *Speed1* mediante relaciones con el individual *MeasureFn1*, haciendo uso de la propiedad *"hasMeasureFn"*, sin especificar directamente ni la magnitud ni la unidad de medida, lo que nos permite tener diferentes medidas y unidades para la velocidad.

1. **Speed1** (instancia de SpeedQuantity)

   Relations: hasMeasureFn **MeasureFn1**

Para poder establecer una unidad de medida concreta (*kperHour*) y una magnitud concreta (40) haremos uso del individual *MeasureFn1*, el cual no ayudará a especificar *Speed1*, aportándonos un alto grado de expresividad y de libertad para la creación de las diferentes unidades de medidas y magnitudes correspondientes:

2. **MeasureFn1** (instancia de MeasureFn)

   Relations:
   magnitude 40 (Instancia de Real)
   unitOfMeasure kperHour (instancia de UnitofMeasure)

10 de febrero 2006 • Volumen 7 Número 2 • ISSN: 1067-6079

De esta forma podremos tener instancias de ciclomotores de tipo A estableciendo la correspondencia entre *Speed1* y la propiedad *speed*, de la siguiente forma:

3. **InstanteOfMotorCycleA** (instancia de MotorCycleA)

   Relations: **speed Speed1**

Es decir, una instancia de MotorCycleA será aquella que tenga una velocidad de tipo *Speed1* y que por tanto tenga una unidad de medida **(kperHour)** y una magnitud **(40)** que lógicamente es consistente con la definición dada para un ciclomotor de este tipo.

Conclusiones

Hemos demostrado que la alta expresividad de los lenguajes semánticos permite establecer vías alternativas a ciertas limitaciones impuestas por las herramientas de uso. A su vez nos ha permitido establecer una guía de desarrollo para este tipo de expresividad semántica, de tal forma que desde una Ontología simple en sus comienzos nos permitiese, tomándola como base, su ampliación para poder incorporar nuevos elementos en la conceptualización, como puedan ser nuevos predicados para el resto de operadores relaciones, nuevas unidades de medida, cantidades físicas etc. El resultado de este experimento queda reflejado en el desarrollo de una completa conceptualización de tipos de vehículos, basada en las definiciones del Reglamento General de Vehículos, fruto de nuestra investigación en el campo de tráfico rodado.

Bibliografía

[Cor00] Corcho, O. and Gómez-Pérez, A. "A Roadmap to Ontology Specification Languages", Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management, October, 2000 at http://delicias.dia.fi.upm.es/articulos/ocorcho/ekaw2000-corcho.pdf. EKAW 2000.

[DAM02] Language feature comparison. Disponible en: http://www.daml.org/language/features.html.
[Den02] Denny, Michael "Ontology Building: A Survey of Editing Tools" Disponible ne: http://www.xml.com/pub/a/2002/11/06/ontologies.html .November 06, 2002.

[Dwa01] *Frank van Harmelen, Peter F. Patel-Schneider and Ian Horrocks, editors.* An annotated version of the example ontology. http://www.daml.org/2001/03/daml+oil-walkthru.

[EON02] Conference on ontology tools: Evaluation of Ontology-based Tools Workshop (EON2002) at the 13th International Conference on Knowledge Engineering and Knowledge Management, at (http://km.aifb.uni-karlsruhe.de/eon2002/). September, 2002.

[Gom04] Gómez-Pérez, Asunción; Fernández-López; Mariano and Corcho, Oscar "Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web", ISBN: 1-85233-551-3 Springer.

[Gru94] Thomas R. Gruber and Gregory R. Olsen "An Ontology for Engineering Mathematics" *Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Gustav Stresemann Institut, Bonn, Germany, Morgan Kaufmann, 1994.

[OEd04] OilEd Ontology Editor web page. En: http://oiled.man.ac.uk/) Última visita Diciembre 2004.

[Ont02] "A survey on ontology tools: OntoWeb report on a comparative study of 11 ontology editors plus several other ontology tools", May, 2002 IST-2000-29243 at http://ontoweb.aifb.uni-karlsruhe.de/About/Deliverables/D13_v1-0.zip.

[OWL04] "OWL Web Ontology Language Reference", W3C Recommendation 10 February 2004. Disponible en: http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

[OWL04b] "OWL Web Ontology Language Semantics and Abstract Syntax". W3C Recommendation 10 February 2004 Disponible en: http://www.w3c.org/TR/owl-semantics.

[PRO04] The Protégé Ontology Editor and Knowledge Acquisition System Disponible en: http://protege.stanford.edu/ Última visita julio 2004.

[Rob02] Roberts Angus. "An Introduction to OilEd". Geodise DAML + OIL Workshop 19-20 February 2002 , Disponible en Internet en http://oiled.man.ac.uk/tutorial/.

[Vat04] Bernard Vatant http://lists.w3.org/Archives/Public/public-swbp-wg/2004Apr/0061.html
[XSD01] XML Schema Part 2: Datatypes W3C Recommendation 02 May 2001 Disponible en: http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.