Una Aproximación Dirigida por Modelos para Diseñar y Construir Esquemas XML: Un Caso de Estudio

A Model Driven Approach to Design and Build XML Schemas: A Case Study

Marta Zorrilla, PhD.[1], Belén Vela, PhD.[2], Esperanza Marcos, PhD.[2]
[1]Dpto. Matemáticas, Estadística y Computación. Universidad de Cantabria. Santander, España
[2]Dpto. Lenguajes y Sistemas Informáticos II. Universidad Rey Juan Carlos. Móstoles, Madrid, España
marta.zorrilla@unican.es, {belen.vela, esperanza.marcos}@ujrc.es

Recibido para revisión 26 de Septiembre de 2007, Aceptado 30 de Noviembre de 2007, Versión final 9 de Diciembre de 2007

Resumen—**En este artículo se muestra la utilidad del marco metodológico dirigido por modelos MIDAS para la transformación de esquemas conceptuales en UML (Modelos Independientes de Plataforma) a esquemas XML (Modelos Específicos de Plataforma) mediante su aplicación a un caso de estudio. Además, se analizan las limitaciones que presenta esta propuesta metodológica a la hora de recoger la semántica del esquema conceptual en el esquema XML y, con el fin de solventar estas deficiencias, se definen nuevas reglas de transformación con las que validar y refinar esta propuesta metodológica.**

Palabras Clave—**Esquemas XML, UML, Desarrollo Dirigido por Modelos, MDA.**

Abstract— **In this article we show the usefulness of the model driven methodological framework called MIDAS for the transformation of conceptual schemas in UML (Platform Independent Model) to XML schemes (Platform Specific Model) by means of its application to a case study. In addition, the limitations that this methodological approach presents at the moment of gathering the semantics of the conceptual schema in the schema XML are analyzed and, in order to solve these deficiencies, new transformation rules with which validate and refine this methodological approach are defined..**

Keywords—**XML Schemas, UML, Model Driven Development, MDA.**

I. INTRODUCTION

La globalización en la que estamos inmersos lleva a las organizaciones, y en particular a sus departamentos de informática, a determinar la mejor manera de diseñar y desarrollar sus Sistemas de Información (SI) con objeto de que sean fácilmente integrables y portables. Para conseguir ambos objetivos, se hace necesario trabajar con modelos conceptuales que recojan fielmente la semántica del negocio y que, por medio de herramientas automáticas (o semi-automáticas) de transformación de modelos, obtengan el modelo que se implante físicamente en la plataforma que corresponda. Pero, para realizar con éxito esta tarea, es conveniente trabajar dentro de un marco metodológico que oriente y guíe al informático en el proceso, por lo que se considera que una propuesta dirigida por modelos es la más adecuada, tanto técnica como económicamente, para que las organizaciones puedan fácilmente adaptarse a los cambios tecnológicos que se producen en cada momento.

En la última década, la penetración de las tecnologías de Internet ha introducido en las empresas otros modos de intercambiar y mostrar la información que reside en sus repositorios, lo que las ha obligado a que parte de su información se gestione como documentos XML, encontrándose con el problema de cómo definirlos, almacenarlos y mantenerlos. Algunas de estas cuestiones se han resuelto en los últimos años. Por una parte, el W3C elaboró la especificación de esquemas XML [22] con el fin de poder definir la estructura, el contenido y la semántica de los documentos XML; y, por otra, los vendedores de software han desarrollado sistemas que permiten su almacenamiento y gestión. Así se dispone de gestores de Bases de Datos (BD) XML nativas como Tamino[14], eXcelon XIS [5], eXist [4] o ToX [2]; y de las extensiones para BD XML, que permiten el almacenamiento de documentos XML en sistemas de gestión de BD convencionales, habitualmente relacionales u objeto-relaciones (OR), como Oracle [11], SQL Server [12] o DB2 [6]. Un estudio de las principales soluciones de BD XML se encuentra en [21].

Sin embargo, los diseñadores (*modeladores de datos*) se

encuentran con un nuevo problema, disponer de una metodología que les permita desarrollar esquemas XML con los que puedan validar el contenido de sus documentos, a partir de esquemas conceptuales definidos en un lenguaje de modelado de alto nivel como ER o UML, en los que generalmente tienen recogidos sus modelos de negocio. Esto es, que puedan adaptarse o migrar a una nueva plataforma tecnológica sin tener que redefinir sus modelos conceptuales, ni abandonar sus prácticas habituales, generalmente de transformación UML o ER al Modelo Relacional u OR. Existen algunos trabajos en esta línea, como por ejemplo las propuestas en [3][7][13], que definen reglas para obtener un esquema XML a partir de un diagrama de clases de UML; o la que se utiliza en este trabajo, MIDAS [8], que tiene la ventaja adicional de ofrecer un marco metodológico y una arquitectura dirigida por modelos basada en MDA (*Model Driven Architecture*) [10], donde las reglas de transformación para pasar del Modelo Independiente de Plataforma (PIM - *Platform Independent Model*), nuestro modelo conceptual, al Modelo Específico de Plataforma (PSM - *Platform Specific Model*), que en nuestro caso será el esquema XML, están formalizadas [17][18]. Además en MIDAS se incluye un perfil UML para la representación gráfica de los esquemas XML [19].

Este artículo pretende mostrar la utilidad del marco metodológico MIDAS para la transformación de esquemas conceptuales a esquemas XML mediante su aplicación a un caso de estudio. Además, se analizan las limitaciones que presenta esta propuesta metodológica a la hora de recoger la semántica del esquema conceptual en el esquema XML y se definen nuevas reglas de transformación y se refinan las existentes, solventando así las carencias detectadas. Se ha de mencionar que ninguno de los trabajos anteriormente citados recoge la extensión de las reglas que aquí se proponen.

El resto del artículo se estructura de la siguiente forma: en el apartado 2 se describe brevemente el marco metodológico de MIDAS y se resumen las reglas de transformación que permiten obtener el PSM a partir del PIM. En el apartado 3 se presenta el caso de estudio y se muestra el esquema XML resultante al aplicar las reglas definidas en MIDAS. En el apartado 4 se discuten sus limitaciones y se proponen nuevas reglas de transformación así como un refinamiento de las existentes, principalmente orientadas a reducir la redundancia y los problemas de consistencia. Finalmente, en el apartado 5 se recogen las principales conclusiones obtenidas y se plantean futuros trabajos.

II. Marco de Trabajo: MIDAS

MIDAS [8] es una metodología dirigida por modelos para el desarrollo de Sistemas de Información Web (SIW), que propone el uso de estándares a lo largo de todo el proceso de desarrollo, así como el uso de UML para el modelado del SIW independientemente del nivel de abstracción o del aspecto del sistema a modelar. Dado que UML no permite representar directamente todos los modelos necesarios, MIDAS incorpora algunas extensiones de UML existentes y define, o adapta, otras nuevas, siempre que es necesario [9][19][20].

MIDAS propone una arquitectura dirigida por modelos y considera a la hora de modelar el sistema los aspectos de contenido, hipertexto y comportamiento, como se puede ver en la Figura 1. Todos estos aspectos se contemplan a nivel de Modelos Independientes de Computación (CIM - *Computation Independent Model*), PIM y PSM. La Figura 1 muestra la arquitectura simplificada de MIDAS, donde se proponen los CIM, comunes a todo el sistema, así como los PIMs y PSMs para los aspectos de contenido, hipertexto y comportamiento. También se definen las reglas (o *mappings)* para transformar los modelos en el mismo nivel (PIM-PIM y PSM-PSM) y entre distintos niveles (PIM-PSM).

En MIDAS se contempla además otros aspectos a tener en cuenta en el desarrollo de un SIW y que son ortogonales a los presentados en la Figura 1, como son la arquitectura del sistema o la seguridad, que se describe con detalle en [20].

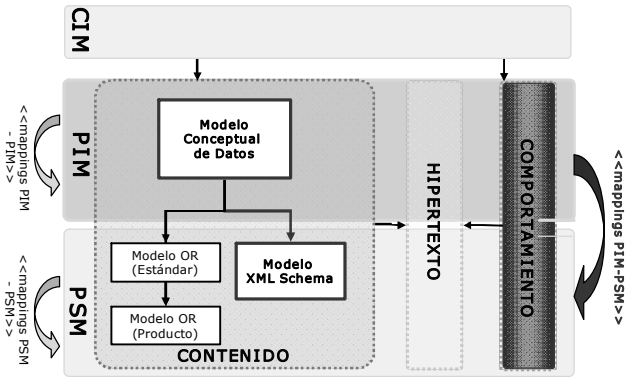

Figura 1. Arquitectura de MIDAS simplificada

Este artículo se centra en el aspecto del ***contenido*** de la metodología, en el que se propone como PIM, el modelo conceptual de datos que se representa mediante un diagrama de clases UML y, como PSM, el modelo OR o el modelo de esquemas XML, dependiendo de la tecnología a utilizar. Para estos dos últimos modelos también se utiliza UML como notación, para lo cual en MIDAS se define un perfil UML para el diseño de BD OR [9] y otro para representar esquemas XML [19]. En este trabajo se aborda la parte relativa al diseño de esquemas XML de MIDAS. Para diseñar esquemas XML (PSM) a partir de un modelo conceptual de datos (PIM), en MIDAS se han definido un conjunto de reglas de transformación (*mappings*). Por una parte, éstas se han descrito en lenguaje natural, y por otra, se han formalizado utilizando gramáticas de grafos [18].

A continuación, se resumen las Reglas de Transformación en lenguaje natural para pasar de PIM a PSM: Modelo Conceptual a Modelo XML Schema.

Regla 0. A la hora de transformar el **PIM completo** se creará en primer lugar el **elemento raíz** del esquema XML de

un tipo complejo que se defina para contener del resto de elementos del esquema (PSM).

Regla 1. Cada **clase** del modelo conceptual corresponderá a un **elemento** en el esquema XML con el mismo nombre de la clase. En el caso de que los elementos XML sean referenciados por otro elemento, por participar en alguna asociación, agregación o composición, estos deben ser definidos como elementos globales. Además, para especificar el tipo del elemento, se incluirá un nuevo *complexType inline*, o si se opta por definirlo con nombre, se utilizará la expresión *nombreclase_type* para nombrarlo

Regla 2. Los **atributos** de una clase se recogerán en el esquema XML como **subelementos** del *complexType* que sirve para definir el tipo del elemento que representa a la clase.

2.1. En el caso de atributos **obligatorios** el valor de *minOccurs* del **subelemento** será **1** (éste es el valor por defecto).

2.2. En el caso de atributos **opcionales** el valor de *minOccurs* del **subelemento** será **0**.

2.3. En el caso de atributos **multivaluados** el valor de la propiedad *maxOccurs* del **subelemento** será *unbounded*.

2.4. En el caso de atributos **compuestos** el **subelemento** será de tipo *complexType* **anónimo**. Dicho *complexType* utilizará el *compositor sequence* para incluir un subelemento por cada componente del atributo compuesto.

2.5. En el caso de atributos **enumerados** el **subelemento** será de un tipo *simpleType* anónimo. Este *simpleType* se relacionará con una clase *enumeration* donde se especificarán los posibles valores del atributo.

Regla 3. Una **asociación** entre dos clases se recogerá incluyendo un **subelemento** en uno de los *complexType* correspondiente a las clases que participan en la asociación, ya que se opta por recogerlas siempre como asociaciones **unidireccionales**. El subelemento recibirá el mismo nombre que la asociación que representa. Dicho subelemento, de tipo **REF**, referenciará al elemento global que corresponde a la otra clase que participa en la asociación. El valor del atributo *minOccurs* del subelemento dependerá de la cardinalidad mínima de la asociación (**0** ó **1**).

3.1. Si la cardinalidad es **1:1**, el subelemento se incluirá en cualquiera de los dos *complexType* que se corresponden a las clases involucradas en la asociación y la decisión la debe tomar el diseñador. El valor del atributo *maxOccurs* será en este caso 1.

3.2. Si la cardinalidad es **1:N**, el subelemento se incluirá forzosamente en el *complexType* que corresponde a la clase de cardinalidad **N**.

3.3. Si la cardinalidad es **N:M,** el valor del atributo *maxOccurs* será *unbounded* y, también en este caso, la decisión de dónde incluir el subelemento la tomará el diseñador.

Regla 4. Las relaciones de **agregación** se recogerán incluyendo un **subelemento** en el *complexType* correspondiente a la clase que actúa como **TODO**. Para nombrarlo se utilizará el nombre de la relación y, en su defecto, la cadena *is_aggregated_of*. El subelemento será de tipo **REF** y referenciará al elemento correspondiente a la clase que actúa como **PARTE**. El valor del atributo *minOccurs* de dicho subelemento dependerá de la cardinalidad mínima de la asociación (**0** ó **1**)..

4.1. Si la **cardinalidad máxima** de la relación es **N**, el valor de la propiedad *maxOccurs* del subelemento será *unbounded*.

Regla 5. Las relaciones de **composición** se recogerán incluyendo un **subelemento** en el *complexType* correspondiente a la clase que actúa como **TODO**. Para nombrarlo se utilizará el nombre de la relación y, en su defecto, la cadena *is_composed_of*. Este subelemento será de tipo *complexType* anónimo y utilizará el *compositor* **all** o **sequence** (este último, en el caso de que la cardinalidad máxima de las partes sea superior a uno) para incluir un conjunto de elementos del tipo del complexType correspondiente a la/s clase/s que actúan como **PARTE** en la **composición**.

Regla 6. En las relaciones de **generalización** el *complexType* utilizado para definir el tipo del elemento que representa a la clase **hija** será una extensión del *complexType* correspondiente a la clase **padre**.

Cabe mencionar que la metodología MIDAS ha sido previamente usada y validada mediante el desarrollo de numerosos casos de laboratorio así como en casos reales. Concretamente, la aproximación metodológica que se usa en este artículo para el diseño de esquemas XML ha sido aplicada en un caso real implantado con éxito en el campo de la gestión y el análisis de imágenes médicas para la investigación en neurociencias [1]. Además se ha usado también para desarrollar el repositorio de la herramienta M2DAT (MIDAS MDA Tool), que integra todas las técnicas propuestas en MIDAS para la generación semiautomática de SIW [16].

III. Caso de Estudio

El caso de estudio, en el que se aplica MIDAS, modela un SI que permite la gestión del glosario de términos y conceptos necesario durante el ciclo de vida de los proyectos de ingeniería software, principalmente en la fase de concepción, con el fin de garantizar la correcta utilización e interpretación del contexto de negocio.

La razón de hacer uso de esquemas XML para su implementación se debe fundamentalmente a la necesidad de

compartir el contenido del diccionario entre las distintas herramientas y aplicaciones software que se utilizan en el desarrollo y explotación de los proyectos informáticos. Como bien es sabido, esta tecnología, incorporada en la mayoría de las herramientas comerciales y de código abierto (*open-source*), es la más adecuada y la que mejor se adapta para realizar intercambios de información entre aplicaciones heterogéneas.

Para mostrar el proceso de desarrollo seguido en este caso de estudio, en primer lugar se presenta el PIM de datos en la Figura 2 y posteriormente, el esquema XML (PSM) que se obtiene tras aplicar las reglas de transformación, resumidas en apartado anterior, en la Figura 3.

A. PIM de Datos

Como se puede observar en la Figura 2, el PIM en el que se apoya, sigue una estructura similar a la utilizada en los tesauros [15] que se definen en el campo de la Bibliotecología. Un tesauro se define como una lista de términos, que pueden estar constituidos por más de una palabra, relacionados entre sí jerárquicamente (términos más genéricos (*broader term*) y términos más específicos (*narrow term*)), y delimitados dentro de un contexto previamente clasificado (*taxonomía*), en el que además pueden existir relaciones asociativas (términos relacionados) y de equivalencia (*sinónimos*) entre términos.

En nuestro caso particular, un glosario (*BusinessGlossary*) se define para un determinado dominio (*BusinessDomain*) de negocio, y está compuesto de un conjunto de términos, cuyo contenido semántico viene delimitado por los conceptos en los que se clasifica, y varias taxonomías (*taxonomy*) en las que se agrupan y relacionan los conceptos con los que se delimita el significado de los términos (reducción de la ambigüedad). Los términos, asimismo, pueden relacionarse entre ellos por sinonimia (*Synonym*), asociación (*RelatedTerm*) y jerarquía (*NarrowTerm, BroaderTerm*).

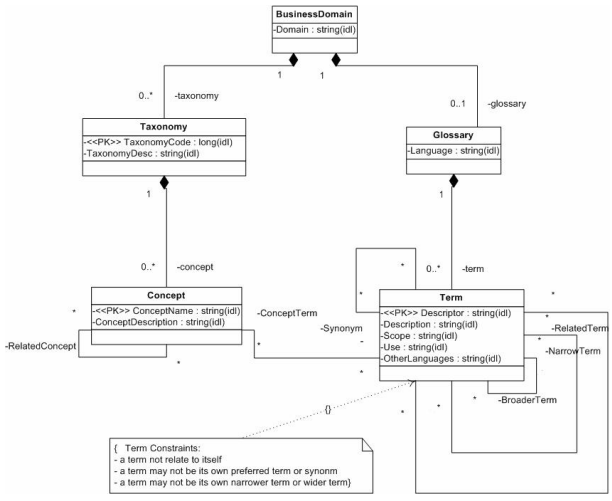

Figura 2. PIM representado mediante diagrama de clases UML

Mediante este modelo, por tanto, se pretende conseguir una unificación en la concepción y definición de los términos y conceptos utilizados en una organización. De modo que se garantice que no existen ambigüedades entre ellos cuando sean utilizados y queden perfectamente especificados, con el objeto de evitar errores de comprensión que conlleven al fracaso de los proyectos informáticos.

B. Descripción del esquema XML obtenido usando MIDAS

Siguiendo las reglas recogidas en la sección II, se transforma el PIM de datos representado en UML en el esquema XML de la Figura 3.

Como todo **documento XML** debe tener un elemento raíz (Regla 0), en nuestro caso este será el elemento denominado *BusinessGlossary* de tipo *BusinessGlossaryType*. Este tipo complejo se define para incluir el elemento *BusinessDomain* y con él, el resto de los componentes del esquema.

Siguiendo la Regla 1, por cada una de las **clases del PIM** se creará un elemento XML. Estos serán elementos globales siempre que necesiten ser referenciados por algún otro elemento del esquema, como es el caso de *Taxonomy, Concept, Glossary* y *Term*.

Como se puede observar en el esquema, las **asociaciones** entre términos (*Term*) y conceptos (*Concepts*) así como las asociaciones reflexivas existentes en términos (*Term*), se transforman utilizando elementos REF, de acuerdo a la Regla 3. Esta transformación puede producir cierta redundancia, además de pérdida de integridad, si se da el caso de que la referencia sea a un término relacionado o a un sinónimo que no existiera como término del diccionario. Además, en el caso de tener múltiples relaciones reflexivas para una misma clase (como es el caso de *Term* que tiene las relaciones reflexivas: *Synonym, RelatedTerm, NarrowTerm* y *BroaderTerm*) se produce una pérdida semántica, ya que, se pierden los nombres de las asociaciones reflexivas, dado que los elementos de tipo REF no pueden tener nombre y, la distinción entre cada una de ellas, ya que en un elemento XML sólo puede tener un único subelemento que se referencie a si mismo.

Una alternativa al uso de elementos de tipo REF para representar asociaciones es la definición de elementos de tipos complejos (*complexTypes*). De esta forma, se evita incluir en el esquema XML elementos globales, y se solucionan los dos problemas planteados. Aunque conviene, con el fin de garantizar la integridad, hacer uso de los elementos *key* y *keyref* del estándar W3C XML Schema [22], con los cuales se establecen las restricciones de unicidad y referenciabilidad con el mismo sentido que en el modelo relacional. El uso de elementos REF, por tanto, se recomienda para transformar asociaciones con multiplicidad **1:1**, es decir, donde una clase A esté asociada exactamente a una instancia de una clase B. Por ejemplo, para el siguiente caso, donde todo profesor es responsable de 0 o varias asignaturas y las asignaturas solo tienen un responsable, con lo que se podría anidar las asignaturas dentro del tipo complejo profesor, sin redundancias.

A la hora de transformar los **atributos** de clases UML se tienen dos posibilidades: en atributos XML o en elementos XML. Una transformación intuitiva y directa podría ser transformar los atributos UML en atributos XML del elemento XML que representa a la clase UML. Sin embargo, sólo se puede hacer de esta forma si los atributos se definen sobre tipos de datos primitivos, es decir, no es válido para atributos multivaluados o compuestos. Por ello, y teniendo en cuenta que los atributos UML son representados como clases en el metamodelo de UML, en MIDAS se propone transformar los atributos de una clase UML en un tipo complejo que tenga como subelementos, los atributos de la clase UML (Regla 2).

Sin embargo, se considera que en algunos casos puede ser interesante que sean atributos XML y, por lo tanto, habría que refinar la regla genérica que MIDAS propone. En [13] se propone utilizar atributos sólo para recoger las claves principales de las relaciones y en [3] ofrece las dos alternativas pero no determina en qué casos son más adecuadas cada una. En [7] se indica que se puede usar atributos o elementos indistintamente, y sólo cuando al atributo sea complejo o multivaluado se debe usar siempre un elemento. En nuestra opinión se deben definir como atributos cuando se trate de información inherente al elemento (meta-información) y como subelementos, cuando su definición puede ser extendida o modificada en un tiempo futuro.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Taxonomy"><!-- Elemento Global-->
  <xs:complexType> <xs:sequence>
    <xs:element name="is_composed_of_2">
     <xs:complexType>
     <xs:sequence>
       <xs:element ref="Concept" minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>  </xs:complexType> </xs:element>
    <xs:element name="TaxonomyCode" type="xs:ID"/>
    <xs:element name="TaxonomyDesc" type="xs:string" minOccurs="0"/>
  </xs:sequence>  </xs:complexType> </xs:element>
<xs:element name="Glossary"><!-- Elemento Global-->
  <xs:complexType> <xs:sequence>
    <xs:element name="is_composed_of_3">
     <xs:complexType>
     <xs:sequence>
       <xs:element ref="Term" minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence> </xs:complexType> </xs:element>
    <xs:element name="Language" type="xs:string"/>
  </xs:sequence> </xs:complexType> </xs:element>
<xs:element name="Concept"> <!-- Elemento Global-->
  <xs:complexType> <xs:sequence>
    <xs:element ref="Concept" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="ConceptName" type="xs:ID"/>
    <xs:element name="ConceptDescription" type="xs:string"/>
  </xs:sequence> </xs:complexType> </xs:element>
<xs:element name="Term"><!-- Elemento Global-->
  <xs:complexType> <xs:sequence>
    <xs:element ref="Term" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="Descriptor" type="xs:ID"/>
    <xs:element name="Description" type="xs:string"/>
    <xs:element name="Scope" type="xs:string" minOccurs="0"/>
    <xs:element name="Use" type="xs:string" minOccurs="0"/>
    <xs:element name="OtherLanguages" type="xs:string" minOccurs="0"/>
    <xs:element ref="Concept" minOccurs="0"/>
  </xs:sequence></xs:complexType> </xs:element>
<xs:element name="BusinessGlossary" type="BusinessGlossaryType"/> <!-- Elemento Raíz-->
        <xs:complexType name="BusinessGlossaryType"> <xs:sequence>
    <xs:element name="BusinessDomain">
     <xs:complexType>
     <xs:sequence>
```

```
      <xs:element name="is_composed_of_1">  <!-- composición -->
      <xs:complexType>
      <xs:sequence>
        <xs:element ref="Taxonomy" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="Glossary" minOccurs="0"/>
      </xs:sequence>  </xs:complexType> </xs:element>
      <xs:element name="Domain" type="xs:string"/>
    </xs:sequence> </xs:complexType> </xs:element>
  </xs:sequence> </xs:complexType>
</xs:schema>
```

Figura 3. PSM: código del esquema XML

IV. NUEVAS REGLAS DE TRANSFORMACIÓN

A continuación, se recogen las nuevas reglas de transformación definidas y las modificadas para pasar del PIM al PSM de datos. Con ellas se pretende que el esquema XML que se obtenga (PSM) recoja la mayor semántica posible, se eviten las redundancias y se conserve la integridad.

Refinamiento de Regla 2.

Los **atributos** de una clase se recogerán en el esquema XML como:

2.a) **atributos XML** cuando se trate de una característica inherente al elemento y que no puede verse modificado por el devenir del modelo. Pero sólo cuando se trate de atributos simples, ya que estos sólo pueden definirse sobre tipos de datos primitivos.

2.b) **subelementos** del ***complexType*** que sirve para definir el tipo del elemento que representa a la clase. De esta forma, un cambio (modificación o ampliación) en el tipo complejo no afectaría al resto. Además será necesario representarlo mediante subelementos cuando se trate de atributos multivaluados o compuestos, ya que estos no se pueden representar por atributos XML.

Extensión de Regla 3.

3.4. Una **asociación** reflexiva se transformará creando un subelemento de tipo complejo con el mismo nombre de la relación reflexiva y añadiendo una restricción de referenciabilidad tal y como se propone en la regla 7, que se define a continuación.

Nueva Regla 7. Transformación de **Restricciones de Referenciabilidad**:

Las restricciones de referenciabilidad definidas a nivel de PIM se ha de especificar en el PSM mediante el lenguaje XPath.

7.a) Cuando la restricción de referenciabilidad se establece entre elementos del mismo tipo (asociación reflexiva), la restricción se puede definir dentro del *complexType* que usa dicho tipo de elemento.

7.b) Sin embargo, cuando la restricción afecta a diferentes tipos complejos, es necesario definir un grupo de elementos (*group*), con el fin de poder establecer la restricción correspondiente.

En la Figura 4 se muestra el esquema XML que se obtiene al aplicar las nuevas reglas de transformación al PIM del caso de estudio.

```
<?xml version="1.0" encoding="UTF-16"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="BusinessGlossary" type="BusinessDomain">
<xs:key name="conceptKey">
      <xs:selector xpath="taxonomy/concept"/>
      <xs:field xpath="ConceptName"/> </xs:key>
   <xs:keyref name="foreignKeyConcept" refer="conceptKey">
      <xs:selector xpath="taxonomy/concept/RelatedConcept"/>
      <xs:field xpath="."/> </xs:keyref>
   <xs:keyref name="foreignKeyTermConcept" refer="conceptKey">
      <xs:selector xpath="glossary/term/Concept"/>
      <xs:field xpath="."/> </xs:keyref>
   </xs:element>
   <xs:complexType name="BusinessDomain">
      <xs:group ref="ConceptTerm"/>
      <xs:attribute name="Domain" type="xs:string" use="required"/>
   </xs:complexType>
<xs:group name="ConceptTerm">
   <xs:sequence>
```

```
   <xs:element name="taxonomy" type="Taxonomy" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="glossary" type="Glossary" minOccurs="0">
   <xs:key name="termKey">
      <xs:selector xpath="term"/>
      <xs:field xpath="Descriptor"/> </xs:key>
   <xs:keyref name="foreignKeyTermSynonym" refer="termKey">
      <xs:selector xpath="term/Synonyms"/>
      <xs:field xpath="."/> </xs:keyref>
   <xs:keyref name="foreignKeyTermNarrow" refer="termKey">
      <xs:selector xpath="term/NarrowTerm"/>
      <xs:field xpath="."/> </xs:keyref>
   <xs:keyref name="foreignKeyTermBroader" refer="termKey">
      <xs:selector xpath="term/BroaderTerm"/>
      <xs:field xpath="."/> </xs:keyref>
   <xs:keyref name="foreignKeyTermRelated" refer="termKey">
      <xs:selector xpath="term/RelatedTerm"/>
      <xs:field xpath="."/> </xs:keyref>
   </xs:element>
   </xs:sequence>
</xs:group>
<xs:complexType name="Taxonomy">
   <xs:sequence>
   <xs:element name="concept" type="Concept" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="TaxonomyCode" type="xs:integer" use="required"/>
   <xs:attribute name="TaxonomyDesc" type="xs:string"/>
</xs:complexType>
<xs:complexType name="Concept">
   <xs:sequence>
            <xs:element name="ConceptName" type="xs:string"/>
            <xs:element name="ConceptDescription" type="xs:string"/>
             <xs:element name="RelatedConcept" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="Glossary">
   <xs:sequence>
      <xs:element name="term" type="Term" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="Language" type="xs:string" use="required"/>
   </xs:complexType>
   <xs:complexType name="Term">
      <xs:sequence>
      <xs:element name="Descriptor" type="xs:string"/>
      <xs:element name="Description" type="xs:string" minOccurs="0"/>
      <xs:element name="ScopeNote" type="xs:string" minOccurs="0"/>
      <xs:element name="Use" type="xs:string" minOccurs="0"/>
      <xs:element name="OtherLanguages" type="xs:string" minOccurs="0"/>
      <xs:element name="Concept" type="xs:string" minOccurs="0"  maxOccurs="unbounded"/>
      <xs:element name="Synonym" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="RelatedTerm" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="NarrowTerm" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="BroaderTerm" type="xs:string" minOccurs="0"  maxOccurs="unbounded"/>
      </xs:sequence> </xs:complexType>
</xs:schema>
```

Figura 4. PSM refinado: código del esquema XML

V. CONCLUSIONES Y TRABAJO FUTURO

Hoy en día se puede afirmar que el intercambio de información en formato XML a través de la red está completamente generalizado. La tecnología para su definición y almacenamiento ha avanzado rápidamente, encontrándose en el mercado soluciones que facilitan su gestión. Pero aún no se tienen métodos y herramientas que faciliten la implementación de un esquema XML (PSM) a partir de un modelo de datos conceptual (PIM). En este sentido, MIDAS, propone una aproximación metodológica con el fin de dar solución a esta problemática y generar de forma automática el esquema XML (PSM) a partir de un PIM de datos descrito en UML.

En este trabajo se presenta un caso de estudio en el que se aplican las reglas de transformación propuestas por MIDAS. En su desarrollo, se han detectado ciertas carencias en las reglas existentes, pues en algunos casos producen redundancias e incluso puede dar lugar a inconsistencias en los datos. Por ello, en este trabajo se han refinado algunas reglas y se han definido otras nuevas. En particular, se añaden dos nuevas reglas de transformación: una, con la que recoger la restricción de referenciabilidad y otra, para realizar la transformación de asociaciones reflexivas. Además, se refinan dos de las reglas existentes: se incluye la posibilidad de definir atributos UML como atributos XML además de como elementos XML, y se matiza la utilización de los elementos de tipo REF.

Actualmente, estamos incorporando estas mejoras en la herramienta MDA que da soporte a la metodología MIDAS, con el fin de automatizar el proceso de generación del SIW. Además, estamos trabajando con otros casos de estudio con los que enriquecer y refinar esta aproximación metodológica.

Paralelamente estamos abordando la especificación de las restricciones recogidas en lenguaje natural que hoy en día, no pueden especificarse directamente en los esquemas XML, si bien, se podrían validar utilizando las extensiones que ofrecen los gestores de bases de datos para la manipulación de documentos XML y su validación por medio de disparadores. Estamos haciendo pruebas para implementar estas validaciones en los gestores Oracle10g y Microsoft SQL Server 2005.

AGRADECIMIENTOS

Esta investigación se ha llevado a cabo en el marco de los siguientes proyectos: GOLD (TIN2005-00010/) financiado por el Ministerio de Educación y Ciencia y FoMDAs (URJC-CM-2006-CET-0387) cofinanciado por la Universidad Rey Juan Carlos y la Comunidad de Madrid.

REFERENCIAS

[1] C. Acuña, E. Marcos, V. de Castro, J. A. Hernández, "A Web Information System for Medical Image Management, Biological and Medical Data Analysis". In Proc. 5th International Symposium, LNCS 3337, Springer-Verlag, 2004, pp. 49-59.

[2] D. Barbosa, A. Barta, A. Mendelzon, G. Mihaila, F. Rizzolo, P. Rodriguez-Gianolli, "ToX - The Toronto XML Engine". In Proc. International Workshop on Information Integration on the Web, Rio de Janeiro, 2001.

[3] D. Carlson. Modeling XML Vocabularies with UML. 2001. Available: www.xml.com.

[4] A.B. Chaudhri, A. Rashid, R. Zicari, (Eds.). XML Data Management. Native XML and XML-Enabled Database Systems. Addison Wesley, 2003.

[5] eXcelon Corporation. Managing DXE. System Documentation Rel. 3.5. eXcelon Corporation. Burlington. Available: www.excelon.corp.com.

[6] IBM DB2 Universal Database - XML Extender Administration and Programming, Product Documentation Version 7. IBM Corporation, 2000.

[7] T. Krumbein, T. Kudrass, "Rule-Based Generation of XML Schemas from UML Class Diagrams". In Berliner XML Tage 2003, Ed. R.Tolksdorf y R. Eckstein, Berlin (Germany), 2003, pp. 213-227.

[8] E. Marcos, B. Vela, P. Cáceres, J.M. Cavero, "MIDAS/DB: a Methodological Framework for Web Database Design". In proc. DASWIS 2001. Yokohama (Japan). Springer-Verlag, LNCS 2465, 2002, pp. 227-238.

[9] E. Marcos, B. Vela, J.M. Cavero, "Methodological Approach for Object-Relational Database Design using UML". Journal on Software and Systems Modeling (SoSyM). Springer-Verlag. Ed.: R. France y B. Rumpe. Vol. SoSyM 2, 2003, pp. 59-72.

[10] OMG. MDA Guide Version 1.0. Document number omg/2003-05-01. Ed.: Miller, J. y Mukerji, J. 2003. Available: www.omg.com/mda.

[11] Oracle Corporation. Oracle XML DB. Technical White Paper. 2003. Available: www.otn.com.

[12] S. Pal, M. Fussell, I. Dolobowsky, XML Support in Microsoft SQL Server 2005. Microsoft Corporation. 2005.

[13] N. Routledge, L. Bird, A. Goodchild. "UML and XML schema". In Proc. ADC '02: Proceedings of the 13th Australasian database conference. 2002.

[14] Software AG. Tamino X-Query. System Documentation V3.1.1. Software AG, Alemania. 2001. Available: www.softwareag.com.

[15] TESE - Thesaurus for Education Systems in Europe - 2006 Edition. 2006. Eurydice.

[16] J.M. Vara, V. de Castro, E. Marcos, "WSDL automatic generation from UML models in a MDA framework". International Journal of Web Services Practices. 2005, Vol. 1 – Issue 1 & 2, pp. 1 – 12.

[17] J.M. Vara, B. Vela, J.M. Cavero, E. Marcos, "Transformaciones de Modelos para el Diseño de BD Objeto-Relacionales". In Proc. XI Jornadas de Ingeniería del Software y Bases de Datos. 2006, pp.225-238.

[18] J.M. Vara, B. Vela, E. Marcos, "Transformación de Modelos para el Desarrollo de Bases de Datos XML". In Proc. III Taller sobre Desarrollo Dirigido por Modelos. MDA y Aplicaciones (DSDM'06). CEUR Worshop Proceedings Eds. A. Vallecillo, V. Pelechado, A. Estevez, 2006, Vol. 227.

[19] B. Vela, C. Acuña, E. Marcos, "A Model Driven Approach for XML Database Development" In Proc. 23rd. International Conference on Conceptual Modelling (ER2004). LNCS 3288, Springer Verlag, 2004, pp. 780-794.

[20] B. Vela, E. Fernandez-Medina, E. Marcos, M. Piattini, "Model Driven Development of Secure XML Databases". Sigmod Record. ACM Press, 2006, Vol. 35, 3, pp. 22-27.

[21] U. Westermann, W. Klas, "An Analysis of XML Database Solutions for the Management of MPEG-7 Media Descriptions". ACM Computing Surveys, 2003, Vol. 35 (4), pp. 331-373.

[22] W3C XML Schema Working Group. XML Schema Parts 0-2:[Primer, Structures, Datatypes]. W3C Recommendation. 2004. Available: http://www.w3.org/XML/Schema.

Marta Zorrilla es Ingeniera de Telecomunicación (1994) y Doctora Ingeniera de Telecomunicación (2001) por la Universidad de Cantabria (España).

Desde marzo de 1995 hasta septiembre de 2006 estuvo contratada como profesora del Área de Ciencias de Computación e Inteligencia Artificial en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad de Cantabria, y en la actualidad ocupa el puesto de Profesor Contratado Doctor en el Área de Lenguajes y Sistemas Informáticos de la misma Universidad. Actualmente ella es la responsable de la docencia de Bases de Datos en la titulación de Ingeniería Informática.

Ha participado y dirigido más de 15 proyectos de investigación, es coautora del libro "Iniciación a las bases de datos con Access 2002" (Díaz de Santos, 2003) y de un número considerable de publicaciones en revistas y congresos nacionales e internacionales. Actualmente, sus líneas principales de investigación se centran en el diseño y desarrollo de sistemas de información y en la aplicación de las tecnologías de inteligencia de negocio (business intelligence) en el ámbito educativo y sanitario.

Belén Vela Sánchez es doctora por la Universidad Rey Juan Carlos e Ingeniera Informática por la Universidad Carlos III de Madrid. Ha trabajado durante 2 años como consultora en la empresa privada (Cronos Ibérica S.A. y PriceWaterhouseCoopers).

Actualmente es profesora contratada doctora en el departamento de Lenguajes y Sistemas Informáticos II de la Escuela Superior de Ingeniería Informática de la Universidad Rey Juan Carlos, donde pertenece al grupo de Bases de Datos e Ingeniería del Software, Kybele, cuya investigación se centra en Ingeniería de Sistemas de Información, abordando temas de desarrollo dirigido por modelos, Sistemas de Información Web, etc.

Ha impartido clases en numerosos cursos y másters de especialización. Actualmente es profesora del "Máster de Sistemas de Información y Comunicación para la Defensa" de la Escuela de Informática para el Ejército.

Es coautora del libro "Tecnología y Diseño de Bases de Datos" (Ra-ma, 2006); es coautora de numerosas publicaciones en revistas y congresos nacionales e internacionales. A su vez, también ha participado en numerosos proyectos de investigación, siendo la investigadora principal en dos de ellos.

Esperanza Marcos Martínez es Ingeniera y Doctora en Informática por la Universidad Politécnica de Madrid y Diplomada en Informática por la Universidad de Valladolid. Durante cinco años ha sido profesora del Departamento de Informática de la Universidad Carlos III de Madrid. Ha sido representante de AENOR en los comités internacionales del ISO/IEC JTC1/SC21 WG3 DBL sobre estandarización del lenguaje SQL:1999.

Actualmente es Profesora Titular de la Escuela Superior de Ingeniería Informática de la Universidad Rey Juan Carlos donde coordina el Grupo de Bases de Datos e Ingeniería del Software, Kybele, cuya investigación se centra en Ingeniería de Sistemas de Información, abordando temas de desarrollo dirigido por modelos, Sistemas de Información Web, etc.

Ha impartido clases y ha dirigido numerosos cursos de especialización relacionados con Bases de Datos e Ingeniería del Software; en la actualidad colabora en el "Máster en Ingeniería del Software" de la Universidad Politécnica de Madrid, en le Master de "Ingeniería de la Decisión" de la Universidad Rey Juan Carlos y es cordinadora académica del Master de "Sistemas de Información y Comunicaciones para la Defensa" impartido conjuntamente entre la Universidad Rey Juan Carlos y la Escuela de Informática del Ejército de Tierra.

Es coautora de diversos libros entre los que destacan: "Diseño de Bases de Datos Relacionales" (Ra-ma, 1999) y "Tecnología y Diseño de Bases de Datos" (Ra-ma, 2006); es coautora de numerosos capítulos de libros y artículos, nacionales e internacionales. Ha participado y dirigido numerosos proyectos de investigación.

Universidad Nacional de Colombia Sede Medellín
Facultad de Minas

120 años TRABAJO Y RECTITUD

Escuela de Ingeniería de Sistemas

Misión

La misión de la Escuela de Ingeniería de Sistemas es fomentar y apoyar la generación o la apropiación de conocimiento, la innovación y el desarrollo tecnológico en el área de ingeniería de sistemas e informática sobre una base científica, tecnológica, ética y humanística.

Visión

La formación integral de profesionales desde el punto de vista científico, tecnológico y social que les permita adoptar, aplicar e innovar conocimiento en el campo de los sistemas e informática en sus diferentes aspectos, aportando con su organización, estructuración, gestión, planeación, modelamiento, desarrollo, procesamiento, validación, transferencia y comunicación; para lograr un desempeño profesional, investigativo y académico que contribuya al desarrollo social, económico, científico y tecnológico del país.

Escuela de Ingeniería de Sistemas
Dirección Postal:
Carrera 80 No. 65 - 223 Bloque M8A
Facultad de Minas. Medellín - Colombia
Tel: (574) 4255350 Fax: (574) 4255365
Email: esistema@unalmed.edu.co
http://pisis.unalmed.edu.co/